МИР
программирования

Р. ХАГГАРТИ

Дискретная математика для программистов

Издание 2-е, исправленное

Перевод с английского
под редакцией С.А. Кулешова
с дополнениями А.А. Ковалева,
В.А. Головешкина, М.В. Ульянова

Допущено УМО вузов РФ по образованию в области прикладной математики в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки «Прикладная математика»

ТЕХНОСФЕРА
Москва
2012

УДК 519.854
ББК 22.176
Х13
Х13 Хаггарти Р.
Дискретная математика для программистов
Издание 2-е, исправленное

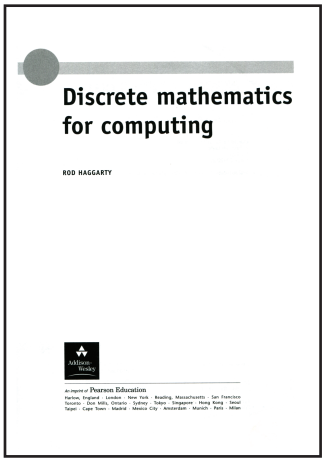
Москва: Техносфера, 2012. – 400 с., ISBN 978-5-94836-303-5

Основополагающее введение в дискретную математику, без знания которой невозможно успешно заниматься информатикой и программированием. Ни одно из многочисленных изданий по этой дисциплине, вышедших на русском языке, не читается с таким удовольствием и пользой. В доступной и весьма увлекательной форме автор рассказывает о фундаментальных понятиях дискретной математики – о логике, множествах, графах, отношениях и булевых функциях. Теория изложена кратко и иллюстрируется многочисленными простыми примерами, что делает ее доступной даже школьнику. После каждой главы (начиная со второй) рассматривается приложение описанных методов к информатике.

Дополнения в издании на русском языке посвящены актуальным задачам теории графов, рекурсивным алгоритмам, общей проблеме перебора и задачам целочисленного программирования.

Книга будет полезна студентам, изучающим курс дискретной математики, а также всем желающим проникнуть в технику написания и проверки корректности алгоритмов, включая программистов-практиков.

УДК 519.854
ББК 22.176



© Pearson Education Limited 2002
This translation of DISCRETE MATHEMATICS FOR COMPUTING, First Edition is published by arrangement with Pearson Education Limited.
© 2012, ЗАО «РИЦ «Техносфера», перевод на русский язык, дополнения,
оригинал-макет, оформление

ISBN 978-5-94836-303-5
ISBN 0-201-73047-2 (англ.)

Содержание

Указатель обозначений

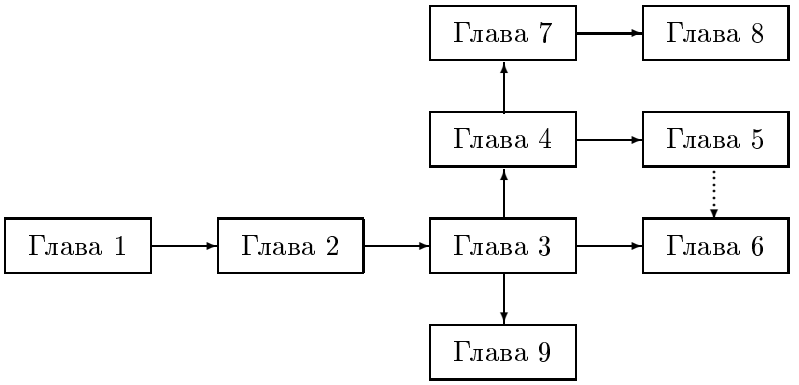
R^*	замыкание отношения R	75
E_x	класс эквивалентности элемента x	78
$x \prec y$	x — непосредственный предшественник y	80
проект	операция «проект»	87
соединение	операция «соединение»	88
выбор	операция «выбор»	89
R^{-1}	обратное отношение	91
$S \circ R$	композиция отношений R и S	92
MN	булево произведение матриц M и N	94
$f(x)$	образ элемента x	97
$f\colon A \longrightarrow B$	функция из A в B	97
$f(A)$	множество значений функции f	97
$f^{-1}\colon\longrightarrow A$	обратная функция	102
$g \circ f$	композиция функций f и g	104
$\lvert x \rvert$	модуль числа x	110
$\lfloor x \rfloor$	целая часть числа x	110
$P(n,\, k)$	число всех $(n,\, k)$-размещений без повторений	121
$C(n,\, k)$	число всех $(n,\, k)$-сочетаний без повторений	123
$O(g(n))$	класс функций, растущих не быстрее, чем $g(n)$	137
$\delta(v)$	степень вершины	143
$G = (V,\, E)$	граф с множеством вершин V и множеством ребер E	143
$c(G)$	число компонент связности	146
K_n	полный граф с n вершинами	148
МОД	минимальное остовное дерево	154
P	граф Петерсена	160
ПЕРТ	система планирования и руководства разработками	171
M^k	булево произведение k экземпляров матрицы M	176
M^*	матрица достижимости	176
$d[v]$	расстояние до вершины v	182
$\bar{p}$	отрицание булевой переменной p	195
$p \vee q$	дизъюнкция переменных p и q	195
$p \wedge q$	конъюнкция переменных p и q	195
НЕ–И	функция **НЕ–И**	200

Предисловие

Основная цель этой книги — рассказать об основной математической технике, необходимой студентам, изучающим информатику. Представленные здесь темы интересны и сами по себе, и в связи с их широкой применимостью как непосредственно в математике, так и в дисциплинах, использующих математический аппарат. В частности, формальные методы, применяемые в информатике, опираются на такие фундаментальные понятия дискретной математики, как логика, множества, отношения и функции.

Теория излагается преднамеренно кратко, а обсуждаемые здесь математические идеи вполне доступны студентам со скромной математической подготовкой. В многочисленных примерах обобщаются и развиваются ключевые идеи курса, а каждая глава, начиная со второй, снабжена приложением теории к практике. Приложения наглядно демонстрируют, как математика, о которой рассказывается в книге, решает задачи информатики. Каждая глава заканчивается набором упражнений, а чтобы поощрить читателя заниматься ими, полное решение приводится только в конце книги.

Основной материал книги появился при подготовке к чтению начального (годового) курса информатики в Оксфорде. Он рассчитан на 20 лекций. Зависимость глав друг от друга представлена на диаграмме, которая показывает, что существует некоторая свобода выбора очередности изучения материала. Это вместе с возможностью опускать отдельные приложения или заменять их альтернативными, делает книгу более гибкой и удобной для изучения.



Есть несколько доступных текстов по дискретной математике, охватывающих схожий материал. Их список для дальнейшего изучения предмета приведен в конце книги. Более продвинутые учебники по дискретной математике требуют большей математической зрелости, и я надеюсь, что читатели, успешно овладевшие содержанием настоящей книги, смогут изучать их более легко и уверенно.

Я хотел бы поблагодарить своих студентов, кто выдержал все трудности этого материала и чей рост собственных математических способностей поощрял меня писать книгу. Моя благодарность адресована также рецензентам предварительного варианта, сделавшим много полезных замечаний, и сотрудникам издательства «Pearson Education» за их усилия, предпринятые при оформлении текста. И наконец, моя признательность — супруге, за ее неизменную заботу и поддержку.

Род Хаггарти
Оксфорд
Март 2001

ГЛАВА 1

ВВЕДЕНИЕ

Дискретная математика и логика лежат в основе любого современного изучения информатики. Слово «дискретный» означает «составленный из отдельных частей», а дискретная математика имеет дело с совокупностями объектов, называемых множествами, и определенными на них структурами. Элементы этих множеств как правило изолированы друг от друга и геометрически не связаны. Действительно, большинство интересующих нас множеств конечны или, по крайней мере, счетны.

Эта область математики привлекается для решения задачи на компьютере в терминах аппаратных средств и программного обеспечения с привлечением организации символов и манипуляции данными. Современный цифровой компьютер — по существу конечная дискретная система. Понимания того, как такая машина работает, можно достигнуть, если представить машину как дискретную математическую систему. Поэтому наша главная цель при изучении дискретной математики — приобрести инструменты и технику, необходимые для понимания и проектирования компьютерных систем. Когда и как использовать эти инструменты и технику — основа раздела математики, известного как математическое моделирование.

В настоящей главе мы бросим взгляд на процесс моделирования и применим стандартный алгоритм к решению практической задачи. Выбранный пример проиллюстрирует не только вид математики, о которой идет речь в этой книге, но и ее использование при решении насущных задач. Затем мы разовьем паскалеподобный[1] псевдокод в качестве средства выражения алгоритмов, вводимых далее, для однозначной трактовки их команд.

1.1. Моделирование

Процесс математического моделирования на диаграмме можно представить так, как показано на рис. 1.1.
В качестве примера моделирования рассмотрим следующую задачу:

> *Расстояние (в милях) между шестью шотландскими городами: Абердин, Эдинбург, Форт Уильям, Глазго, Инвернесс*

[1] Pascal — язык программирования высокого уровня. — *Прим. перев.*

и Перт дано в табл. 1.1. Требуется найти дорожную сеть минимальной длины, связывающую все шесть городов.

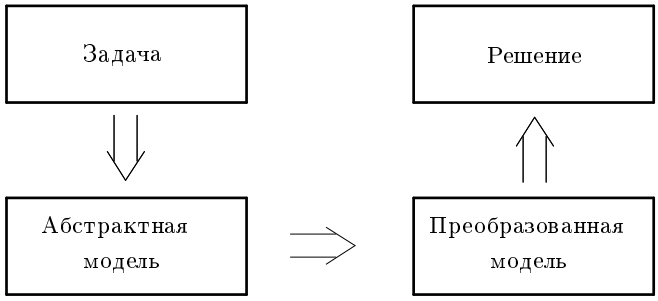


Рисунок 1.1. Схема моделирования

Сама таблица является абстрактной моделью реальной задачи. Однако для нашего решения мы преобразуем ее в геометрическую модель.

Таблица 1.1

	Абердин	Эдинбург	Форт Уильям	Глазго	Инвернесс	Перт
Абердин	—	120	147	142	107	81
Эдинбург	120	—	132	42	157	45
Форт Уильям	147	132	—	108	66	105
Глазго	142	42	108	—	168	61
Инвернесс	107	157	66	168	—	112
Перт	81	45	105	61	112	—

Мы нарисуем *граф*, чьи *вершины* обозначают города, а *ребра* — дороги их связывающие. Более подробно о графах рассказано в главе 7. Каждое ребро нашего графа, изображенного на рис. 1.2, снабжено *весом*, который означает расстояние между соответствующими городами согласно табл. 1.1.

Для решения поставленной задачи с помощью подходящего *алгоритма* (последовательности однозначных инструкций, выполнение которых влечет решение за конечное время), мы построим новый граф, имеющий минимальный общий вес, в котором все шесть городов будут соединены дорогами.

Алгоритм Прима

Шаг 1 Выберите произвольную вершину и ребро, соединяющее ее с ближайшим (по весу) соседом.

Шаг 2 Найдите не присоединенную (еще) вершину, ближе всего лежащую к одной из присоединенных, и соедините с ней.

Шаг 3 Повторяйте шаг 2 до тех пор пока все вершины не будут присоединены.

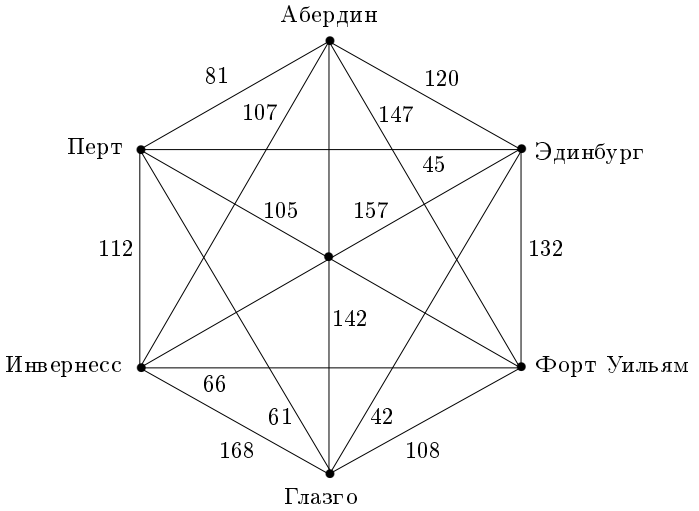


Рисунок 1.2.

На рисунках 1.3, 1.4 и 1.5 изображена последовательность графов, которая получается в результате применения алгоритма Прима, если начинать с вершины Перт. Последний граф (с общим весом 339) представляет собой минимальную сеть дорог, охватывающую все шесть городов.

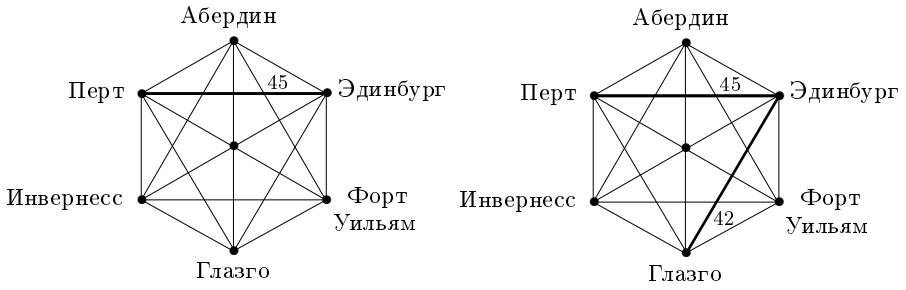


Рисунок 1.3.

Алгоритм, который мы применяли, написан на обычном русском языке. Разговорный язык может оказаться слишком многоречивым, неоднозначным и, в следствие этого, не соответствующим запутанной проблеме. Мы могли бы написать программу для компьютера,

реализующую алгоритм, но какой язык выбрать? Кроме того, язык программирования зачастую скрывает истинный смысл алгоритма от неопытного читателя! Подходящий компромисс в этой ситуации — использовать так называемый *псевдокод*, состоящий из небольшого числа структурных языковых элементов вместе с русскоподобным описанием действий реализуемого алгоритма. О нем идет речь в следующем параграфе.

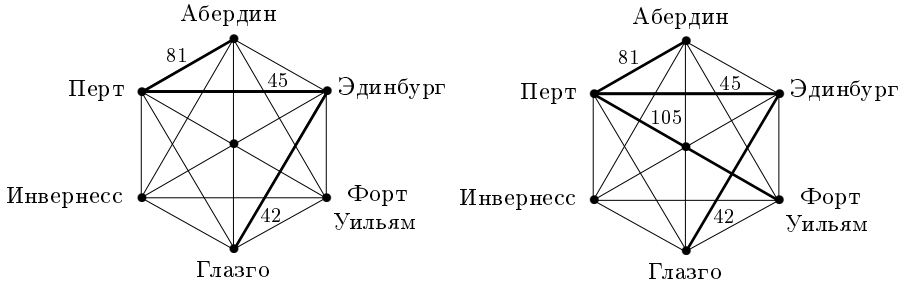


Рисунок 1.4.

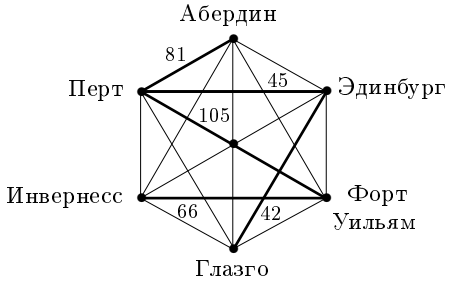


Рисунок 1.5.

1.2. Псевдокод

Мы будем использовать псевдокод, основанный на Паскале. Алгоритм в нем выглядит следующим образом.

begin
операторы исполняемых действий
операторы, управляющие порядком выполнения
end

Строительными блоками алгоритмического языка являются операторы, которые можно разбить на две категории: операторы присваивания и управляющие операторы.

Оператор присваивания приписывает переменным определенные величины и имеют такую общую форму:

имя переменной := *выражение*

Пример 1.2.1. (Алгоритм сложения двух чисел, *First* и *Second*, и присвоение результата переменной *Sum*.)

begin
Input First and Second;
Sum:= *First* + *Second;*
end

Управляющий оператор определяет порядок, в котором должны выполняться шаги алгоритма. Операторы управления бывают трех типов:

- составные операторы;
- условные операторы;
- оператор цикла.

Составные операторы представляют собой список операторов, которые должны выполняться как отдельная команда в том порядке, в котором они записаны. Составные операторы имеют следующий вид:

begin
оператор 1;
оператор 2;
.....
оператор n;
end

Пример 1.2.2. (Алгоритм обмена значений двух переменных: *One* и *Two*.)

begin
Input One and Two;
Temp := *One*;
One := *Two*;
Two := *Temp*;
end

Чтобы проследить за работой алгоритма, предположим, что начальные значения переменных *One* и *Two* равны 5 и 7 соответственно, и обратимся к табл. 1.2.

Таблица 1.2

	Temp	*One*	*Two*
Строка 1	—	5	7
Строка 2	5	5	7
Строка 3	5	7	7
Строка 4	5	7	5

Условные операторы позволяют делать выбор между двумя альтернативными ситуациями. Они записываются в виде **if-then** или **if-then-else**. На псевдокоде условные операторы изображают так:

begin
  if *условие* **then** *оператор*
end

или так:

begin
  if *условие* **then** оператор 1
    else *оператор 2*
end

Пример 1.2.3. (Алгоритм вычисления модуля числа n и присвоение результата переменной *abc*.)

begin
  Input n;
  if $n < 0$ **then** $abc := -n$
    else $abc := n$;
  Output abc;
end

В этом алгоритме оператор, стоящий во второй строке, выполняется при отрицательных значениях переменной n, а в третьей — при положительных (и нулевом). Можно написать и другой алгоритм, решающий ту же задачу, но не использующий **else**:

begin
  Input n;
  if $n < 0$ **then** $n := -n$;
  $abc := n$;
  Output abc;
end

Здесь оператор во второй строчке выполняется только при отрицательных значениях n и игнорируется при любом другом значении.

В последнем случае выполняется оператор, записанный в третьей строке.

Оператор цикла или просто *цикл* может иметь одну из форм записи:

for $X := A$ **to** Z **do** оператор; (1)

while выражение **do** оператор; (2)

repeat
оператор 1;
оператор 2;
.....
оператор n;
until условие. (3)

Здесь X — переменная, а A и Z — ее начальное и конечное значения.

В случае (1) цикл повторяется определенное число раз. Его разновидность выглядит следующим образом:

for *всех элементов множества* **do** *оператор*

В случае (2) цикл выполняется не определенное число раз, а до тех пор, пока выражение, о котором в нем идет речь, остается верным. Как только выражение становится ложным, цикл заканчивается.

И наконец, в последней ситуации (3) цикл выполняется до тех пор, пока конечное условие остается ложным. Единственное различие между (2) и (3) заключается в том, что в последнем цикл выполнится по крайней мере один раз, поскольку истинность условия в нем проверяется *после* каждого прохода цикла.

Пример 1.2.4. (Алгоритм вычисления суммы квадратов первых n натуральных чисел.)

```
begin
    sum := 0;
    for i := 1 to n do
        begin
            j := i * i;
            sum := sum + j;
        end
    Output sum;
end
```

Проследим алгоритм в случае $n = 4$, записав результаты в табл. 1.3

Таблица 1.3

	i	j	Sum
Перед выполнением цикла	—	—	0
Первый проход цикла	1	1	1
Второй проход цикла	2	4	5
Третий проход цикла	3	9	14
Четвертый проход цикла	4	16	30

Выводимый результат: $sum = 30$.

Пример 1.2.5. (Алгоритм выделения графа с минимальным общим весом, связывающего все вершины в данном связном взвешенном графе.)

```
begin
    v := произвольная вершина;
    u := ближайшая соседняя вершина;
    связать v и u;
    while остаются неприсоединенные вершины do
        begin
            u := неприсоединенная вершина, ближайшая
                 к одной из присоединенных вершин;
            соединить u с ближайшей
            из присоединенных вершин;
        end
end
```

Это — написанная на псевдокоде версия алгоритма Прима, с которым мы познакомились ранее.

Замечание. *Связным называется такой граф, в котором существует путь (по ребрам) между любыми двумя вершинами (подробнее об этом см. главу 7, стр. 146).*

Превращение алгоритма в работающую программу — дело программирования или курса структуры данных, поэтому мы не будем обсуждать этот процесс в нашей книге. Однако мы познакомимся со множеством алгоритмов, некоторые из которых представлены в форме псевдокода, а другие оформлены как математические теоремы. Доказательство истинности теорем — необходимая и далеко нетривиальная часть математического процесса. Аналогично необходимо проверять корректность написанного на псевдокоде алгоритма. Например, откуда мы можем знать, что алгоритм из примера 1.2.5 действительно дает минимальную сеть дорог?

В том случае, когда есть несколько различных алгоритмов, решающих одну и ту же задачу, возникает вопрос: какой из них является более эффективным? В упражнении 1.5 приведен еще один алгоритм, суммирующий квадраты натуральных чисел (как и в примере 1.2.4). Оба работают. Но какой это делает быстрее, использует при этом меньше памяти? Короче говоря, какой из этих алгоритмов является наилучшим?

Обе эти проблемы: корректности и эффективности алгоритмов — будут обсуждаться в последующих главах после того, как мы освоим необходимый для этого аппарат дискретной математики.

Набор упражнений 1

1.1. Граф на рисунке рис. 1.6 изображает сеть дорог, связывающих семь деревень. Расстояние между деревнями задано в милях. Используя алгоритм Прима, найдите сеть дорог минимальной общей длины, охватывающую все деревни.

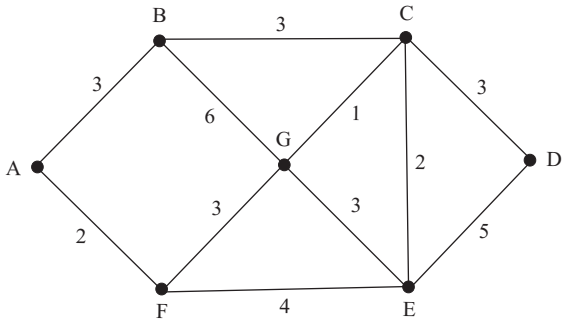


Рисунок 1.6.

1.2. Найдите результаты вычислений следующего алгоритма в случаях

(а) $n = 3$;

(б) $n = 5$.

```
begin
    f := 1;
    Input n;
    for i := 1 to n do
        f := f * i;
    Output f;
end
```

Что получится на выходе алгоритма при произвольном натуральном числе n?

1.3. Проследите за изменением значений переменных i и j в следующем алгоритме при $m = 3$ и $n = 4$:

```
begin
    Input m, n;
    i := 1;
    j := m;
    while i ≠ n do
        begin
            i := i + 1;
            j := j * m;
        end
    Output j;
end
```

Опишите на словах выходные данные этого алгоритма при произвольных целых m и $n > 0$. Что получится при $n = 0$?

1.4. Найдите целые числа, получающиеся в результате работы следующего алгоритма:

```
begin
    first := 1;
    Output first;
    second := 1;
    Output second;
    next := first + second;
    while next < 100 do
        begin
            Output next;
            first := second;
            second := next;
            next := first + second;
        end
end
```

Опишите полученную последовательность чисел в терминах ее членов.

1.5. Проследите эволюцию значений переменных l, sum и k в алгоритме, приведенном на следующей странице при $n = 6$.

```
begin
    Input n;
    k := 1;
    l := 0;
    sum := 0;
    while k < 2n do
        begin
            l := l + k;
            sum := sum + l;
            k := k + 2;
        end
    Output sum;
end
```

Опишите результат работы алгоритма при вводе произвольного натурального значения n.

1.6. Проследите работу алгоритма на примере сети дорог, из упр. 1.1. Какой получился результат?

```
begin
    Упорядочите ребра графа по убыванию веса
    и пронумеруйте их числами: 1, 2, 3, ... и т. д.;
    m := число вершин;
    остаток := число ребер;
    текущее := 1;
    while остаток > m − 1 do
        begin
            if удаление ребра с номером «текущее»
               не нарушает связности графа then
                  begin
                      удалить ребро «текущее»;
                      остаток := остаток − 1;
                  end;
            текущее := текущее + 1;
        end
end
```

Краткое содержание главы

Дискретная математика представляет собой математический аппарат и технику, необходимую для проектирования и понимания вычислительных систем.

Математическое моделирование — это процесс, привлекающий математику для решения реальных практических задач.

Граф (модель) данной сети дорог между городами состоит из набора **вершин**, изображающих города, соединенных друг с другом (взвешенными) **ребрами**, обозначающими дороги.

Алгоритм — это последовательность однозначных команд, выполнение которых влечет решение поставленной задачи за конечное время.

Алгоритм Прима может быть использован для выделения сети ребер минимального общего веса, соединяющей все вершины данного взвешенного графа.

Псевдокодом называется набор структурных элементов языка, подходящий для выражения алгоритма в однозначных терминах.

Оператор присваивания присваивает переменным определенные значения.

Управляющий оператор определяет порядок, в котором должны выполняться шаги алгоритма.

Составной оператор представляет собой список инструкций (операторов), которые должны выполняться как отдельная команда в том порядке, в котором они записаны.

Условный оператор дает возможность сделать выбор между альтернативными возможностями.

Оператор цикла или просто **цикл** позволяет выполнить определенный набор команд подходящее число раз.

ГЛАВА 2

ЛОГИКА И ДОКАЗАТЕЛЬСТВО

Логика необходима в любой формальной дисциплине и состоит из правил получения обоснованного вывода (заключения). Логику можно выделить из контекста тех дисциплин, в которых она используется, и изучать как отдельный раздел науки. Акцент в этой главе будет сделан именно на логике, лежащей в основе неоспоримых рассуждений и доказательств.

Мы познакомимся с логикой высказываний, имеющей дело с истинностью (или ложностью) простых описательных утверждений, что можно рассматривать как короткое введение в логику предикатов. Скажем сразу, что предикатами принято называть утверждения, содержащие переменные величины[1]. Кроме того, в этой главе описаны различные методы доказательств (прямое рассуждение, метод «от противного» и обратное рассуждение), снабженные простыми примерами проверки фактов о четных и нечетных числах, иллюстрирующими методологию рассуждений. Наконец, мы рассмотрим сильный метод доказательства, называемый методом математической индукции.

После упражнений, размещенных в конце главы, мы встретимся с первыми приложениями изучаемых методов к информатике. В них мы увидим, как логические методы доказательств используются при проверке корректности алгоритмов.

2.1. Высказывания и логика

Стандартными блоками формальной логики являются высказывания. *Высказыванием* называется утверждение, которое имеет значение истинности, т. е. может быть **истинным** (обозначается буквой И) или **ложным** (обозначается Л). Например,

- земля плоская;
- Сара — доктор;
- 29 — простое число.

[1]Здесь нужно сказать, что логика предикатов обобщает логику высказываний, и мы ею тоже займемся.

Каждое из высказываний можно обозначить своей буквой. Пусть, например, P обозначает высказывание «земля плоская», Q — «Сара — доктор» и R — «29 — простое число».

Используя такие логические операции, как **не**, **или**, **и**, можно построить новые, так называемые *составные высказывания*, компануя более простые. Например,

- (**не** P) — это высказывание «земля не плоская»;
- (P **или** Q) — «земля плоская или Сара — доктор»;
- (P **и** Q) — «земля плоская и Сара — доктор».

Пример 2.1. Обозначим через P высказывание «логика — забава», а через Q — «сегодня пятница». Требуется выразить каждое из следующих составных высказываний в символьной форме.

(а) Логика — не забава, и сегодня пятница.

(б) Сегодня не пятница, да и логика — не забава.

(в) Либо логика — забава, либо сегодня пятница.

Решение.

(а) (**не** P) **и** Q.

(б) (**не** P) **и** (**не** Q).

(в) P **или** Q.

Чтобы уметь определять значение истинности составных высказываний, нам необходимо разобраться со смыслом логических операций, т. е. какой эффект они оказывают на истинностное значение простых высказываний. Это можно аккуратно сделать с помощью так называемых *таблиц истинности*.

Отрицанием произвольного высказывания P называется высказывание вида (**не** P), чье истинностное значение строго противоположно значению P. Определяющая таблица истинности отрицания высказывания приведена в табл. 2.1.

Таблица 2.1

P	(**не** P)
И	Л
Л	И

Конъюнкцией или *логическим умножением* двух высказываний P и Q называют составное высказывание вида (P **и** Q). Оно принимает истинное значение только в том случае, когда истинны обе его составные части. Такое определение хорошо согласуется с обычным пониманием союза «и» в разговорном языке. Соответствующая таблица истинности — табл. 2.2.

Таблица 2.2

P	Q	(P **и** Q)
И	И	И
И	Л	Л
Л	И	Л
Л	Л	Л

Дизъюнкцией или *логическим сложением* двух высказываний P и Q называется составное высказывание (P **или** Q). Оно истинно, если хотя бы одна из ее составных частей имеет истинное значение, что в некотором смысле также согласуется с обыденным пониманием союза «или». Другими словами, (P **или** Q) означает, что «или P, или Q, или и то, и другое». Таблица истинности дизъюнкции обозначена как табл. 2.3.

Таблица 2.3

P	Q	(P **или** Q)
И	И	И
И	Л	И
Л	И	И
Л	Л	Л

Пример 2.2. Что можно сказать об истинности составного высказывания: «либо луна делается из зеленого сыра и Генрих VIII имел шесть жен, или не верно, что дронт[1] вымер»?

Решение. Обозначим через P высказывание «луна делается из зеленого сыра», через Q — «Генрих VIII имел шесть жен» и через R — «дронт вымер». Символьная запись данного высказывания имеет вид: (P **и** Q) **или** (**не** R). Известно, что высказывание P ложно, а Q и R истинны. Поэтому высказывание (P **и** Q) **или** (**не** R) имеет такое истинностное значение: (Л **и** И) **или** Л, что эквивалентно Л.

[1] Дронт — вымершая птица отряда голубеобразных, обитавшая на островах Индийского океана и истребленная в XVII–XVIII в.в. завезенными туда свиньями. — *Прим. перев.*

Два составных высказывания, построенные из одних и тех же простых утверждений, но разными путями, могут принимать одинаковые значения истинности на любом возможном наборе значений истинности своих составных частей. Такие высказывания называются *логически эквивалентными*.

Пример 2.3. Показать, что высказывание (**не** (P **и** (**не** Q))) логически эквивалентно утверждению ((**не** P) **или** Q).

Решение. Заполним совместную таблицу истинности (табл. 2.4) для составных высказываний:

$$R = (\textbf{не}\ (P\ \textbf{и}\ (\textbf{не}\ Q))) \qquad \text{и} \qquad S = ((\textbf{не}\ P)\ \textbf{или}\ Q).$$

Вспомогательные колонки используются для построения обоих выражений из P и Q.

Таблица 2.4

P	Q	**не** P	**не** Q	P **и** (**не** Q)	R	S
И	И	Л	Л	Л	И	И
И	Л	Л	И	И	Л	Л
Л	И	И	Л	Л	И	И
Л	Л	И	И	Л	И	И

Две последние колонки таблицы идентичны. Это означает, что высказывание R логически эквивалентно высказыванию S.

Важно изучить еще один тип логического оператора, результатом которого является *условное высказывание*. Примером такого высказывания является следующее: «если завтра будет суббота, то сегодня — пятница». При определении истинностного значения условного высказывания, необходимо различать фактическую истину и логическую.

Рассмотрим высказывание «если P, то Q». В том случае, когда предпосылка P истинна, мы не можем получить лигически корректного заключения, если Q ложно. Однако если посылка P ложна, мы имеем логически корректное высказывание и когда Q ложно, и когда оно истинно.

Пример 2.4. Пусть P — (ложное) высказывание $1 = 5$, Q — (тоже ложное) высказывание $3 = 7$ и R — (истинное) утверждение $4 = 4$. Показать, что условные высказывания: «если P, то Q» и «если P, то R», — оба истинны.

Решение. Если $1 = 5$, то, прибавляя 2 к обеим частям равенства, мы получим, что $3 = 7$. Следовательно, высказывание «если P, то Q»

справедливо. Вычтем теперь из обеих частей равенства $1 = 5$ число 3 и придем к $-2 = 2$. Поэтому $(-2)^2 = 2^2$, т. е. $4 = 4$. Таким образом, «если P, то R» тоже верно.

В логике условное высказывание «если P, то Q» принято считать ложным только в том случае, когда *предпосылка* P истинна, а *заключение* Q ложно. В любом другом случае оно считается истинным.

Используя символ импликации «$\Rightarrow$», мы пишем $P \Rightarrow Q$ для обозначения условного высказывания «если P, то Q». Такая запись читается как «из P следует Q» или, «P влечет Q», или «P достаточно для Q», или «Q необходимо для P».

Таблица истинности импликации приведена в табл. 2.5.

Таблица 2.5

P	Q	$(P \Rightarrow Q)$
И	И	И
И	Л	Л
Л	И	И
Л	Л	И

Пример 2.5. Высказывание $((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$ называется *противоположным* или *контрапозитивным* к высказыванию $(P \Rightarrow Q)$. Показать, что $((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$ логически эквивалентно высказыванию $(P \Rightarrow Q)$.

Решение. Рассмотрим совместную таблицу истинности (табл. 2.6).

Таблица 2.6

P	Q	**не** P	**не** Q	$(P \Rightarrow Q)$	$((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$
И	И	Л	Л	И	И
И	Л	Л	И	Л	Л
Л	И	И	Л	И	И
Л	Л	И	И	И	И

Поскольку два последних столбца этой таблицы совпадают, то и высказывания, о которых идет речь, логически эквивалентны.

2.2. Предикаты и кванторы

Логика высказываний применяется к простым декларативным высказываниям, где базисные высказывания — либо истинны, либо ложны. Утверждения, содержащие одну и более переменных, могут

быть верными при некоторых значениях переменных и ложными при других.

Предикатом называется утверждение, содержащее переменные, принимающее значение истины или лжи в зависимости от значений переменных. Например, выражение «x — целое число, удовлетворяющее соотношению $x = x^2$» является предикатом, поскольку оно истинно при $x = 0$ или $x = 1$ и ложно в любом другом случае.

Логические операции можно применять и к предикатам. В общем случае истинность составного предиката в конечном счете зависит от значений входящих в него переменных. Однако существуют некоторые, еще незнакомые Вам логические операторы (называемые *кванторами*), применение которых к предикатам превращает последние в ложные или истинные высказывания.

Пример 2.6. Какие из следующих высказываний истинны, а какие ложны?

(а) Сумма внутренних углов любого треугольника равна 180°.

(б) У всех кошек есть хвост.

(в) Найдется целое число x, удовлетворяющее соотношению $x^2 = 2$.

(г) Существует простое четное число.

Решение.

(а) Истинно.

(б) Ложно. У бесхвостой[1] кошки хвоста нет.

(в) Ложно.

(г) Истинно. Число 2 является и простым, и четным.

В примере 2.6 мы имеем дело с набором объектов и утверждениями о том, что некоторое свойство имеет место *для всех* рассматриваемых объектов, или что *найдется* (*существует*) по крайней мере один объект, обладающий данным свойством.

Выражения «для всех» и «найдется» («существует») называются кванторами и обозначаются, соответственно, $\forall$ и $\exists$. Включая в предикат кванторы, мы превращаем его в высказывание. Поэтому предикат с кванторами может быть истинным или ложным.

[1] Бесхвостая кошка — разновидность домашней кошки. — *Прим. перев.*

Пример 2.7. Обозначим через $P(x)$ предикат «x — целое число и $x^2 = 16$». Выразите словами высказывание: $\exists\, x :\ P(x)$ и определите его истинностное значение.

Решение. Высказывание $\exists\, x :\ P(x)$ означает, что найдется целое число x, удовлетворяющее уравнению $x^2 = 16$. Высказывание, конечно, истинно, поскольку уравнение $x^2 = 16$ превращается в верное тождество при $x = 4$. Кроме того, $x = -4$ — также решение данного уравнения. Однако нам не требуется рассуждать о знаке переменной x, чтобы проверить истинность высказывания $\exists\, x :\ P(x)$.

Пример 2.8. Пусть $P(x)$ — предикат: «x — вещественное число и $x^2+1 = 0$». Выразите словами высказывание: $\exists\, x :\ P(x)$ и определите его истинностное значение.

Решение. Данное высказывание можно прочитать так: существует вещественное число x, удовлетворяющее уравнению $x^2 + 1 = 0$. Поскольку квадрат любого вещественного числа неотрицателен, т. е. $x^2 \geqslant 0$, мы получаем, что $x^2 + 1 \geqslant 1$. Следовательно, утверждение $\exists\, x :\ P(x)$ ложно.

Отрицание высказывания из примера 2.8 записывается в следующем виде: **не** $\exists\, x :\ P(x)$. Это, естественно, истинное высказывание, которое означает, что не существует вещественного числа x, удовлетворяющего условию $x^2 + 1 = 0$. Иными словами, каково бы ни было вещественное x, $x^2 + 1 \neq 0$. В символьной форме это можно записать как $\forall\, x$ **не** $P(x)$.

Для общего предиката $P(x)$ есть следующие логические эквивалентности[2]:

$$\textbf{не}\ \exists\, x :\ P(x) \Leftrightarrow \forall\, x\ \textbf{не}\ P(x);$$

$$\textbf{не}\ \forall\, x\ P(x) \Leftrightarrow \exists\, x :\ P(x).$$

Как показывает следующий пример, некоторые трудности возникают, когда в высказывании участвует более одного квантора.

Пример 2.9. Предположим, что x и y — вещественные числа, а $P(x,\, y)$ обозначает предикат $x + y = 0$. Выразите каждое из высказываний словами и определите их истинность.

(а) $\forall\, x\ \exists\, y :\ P(x,\, y)$;

(б) $\exists\, y :\ \forall\, x\ P(x,\, y)$.

[2] В символьной форме логически эквивалентные высказывания обозначаются значком «$\Leftrightarrow$». — *Прим. перев.*

Решение.

(а) Высказывание $\forall x\, \exists y:\ P(x, y)$ говорит о том, что для любого вещественного числа x найдется такое вещественное число y, что $x+y=0$. Оно, очевидно, верно, поскольку какое бы число x мы ни взяли, число $y = -x$ обращает равенство $x + y = 0$ в верное тождество.

(б) Высказывание $\exists y:\ \forall x\ P(x, y)$ читается следующим образом: существует такое вещественное число y, что для любого вещественного числа x выполнено равенство $x + y = 0$. Это, конечно, не так: не существует вещественного числа y, обладающего указанным свойством. Следовательно, высказывание ложно.

2.3. Методы доказательств

При доказательстве теорем применяется логическая аргументация. Доказательства в информатике — неотъемлемая часть проверки корректности алгоритмов. Необходимость доказательства возникает, когда нам нужно установить истинность высказывания вида $(P \Rightarrow Q)$. Существует несколько стандартных типов доказательств, включающих следующие:

1. Прямое рассуждение. Предполагаем, что высказывание P истинно и показываем справедливость Q. Такой способ доказательства исключает ситуацию, когда P истинно, а Q — ложно, поскольку именно в этом и только в этом случае импликация $(P \Rightarrow Q)$ принимает ложное значение (см. табл. 2.5 на стр. 27).

2. Обратное рассуждение. Предполагаем, что высказывание Q ложно и показываем ошибочность P. То есть, фактически, прямым способом проверяем истинность импликации $((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$, что согласно примеру 2.5, логически эквивалентно истинности исходного утверждения $(P \Rightarrow Q)$.

3. Метод «от противного». Предположив, что высказывание P истинно, а Q ложно, используя аргументированное рассуждение, получаем противоречие. Этот способ опять-таки основан на том, что импликация $(P \Rightarrow Q)$ принимает ложное значение только тогда, когда P истинно, а Q ложно.

Пример 2.10. Покажите прямым способом рассуждений, что произведение xy двух нечетных целых чисел x и y всегда нечетно.

Решение. Прежде всего заметим, что любое нечетное число, и в частности x, можно записать в виде $x = 2m + 1$, где m — целое число. Аналогично, $y = 2n + 1$ с некоторым целым n.

Значит, произведение

$$xy = (2m + 1)(2n + 1) = 4mn + 2m + 2n + 1 = 2(2mn + m + n) + 1$$

тоже является нечетным числом.

Пример 2.11. Пусть n — натуральное число. Покажите, используя обратный способ доказательства, что если n^2 нечетно, то и n нечетно.

Решение. Отрицанием высказывания о нечетности числа n^2 служит утверждение «n^2 четно», а высказывание о четности n является отрицанием утверждения «число n нечетно». Таким образом, нам нужно показать прямым способом рассуждений, что четность числа n влечет четность его квадрата n^2.

Так как n четно, то $n = 2m$ для какого-то целого числа m. Следовательно, $n^2 = 4m^2 = 2(2m^2)$ — четное число.

Пример 2.12. Методом «от противного» покажите, что решение уравнения $x^2 = 2$ является иррациональным числом, т. е. не может быть записано в виде дроби с целыми числителем и знаменателем.

Решение. Здесь нам следует допустить, что решение x уравнения $x^2 = 2$ рационально, т. е. записывается в виде дроби $x = \frac{m}{n}$ с целыми m и n, причем $n \neq 0$. Предположив это, нам необходимо получить противоречие либо с предположением, либо с каким-то ранее доказанным фактом.

Как известно, рациональное число неоднозначно записывается в виде дроби. Например, $x = \frac{m}{n} = \frac{2m}{2n} = \frac{3m}{3n}$ и т. д. Однако можно считать, что m и n не имеют общих делителей. В этом случае неоднозначность записи пропадает.

Итак, предполагаем дополнительно, что дробь $x = \frac{m}{n}$ несократима (m и n не имеют общих делителей). По условию число x удовлетворяет уравнению $x^2 = 2$. Значит, $\left(\frac{m}{n}\right)^2 = 2$, откуда $m^2 = 2n^2$.

Из последнего равенства следует, что число m^2 четно. Следовательно, m тоже четно (см. упр. а(б)) и может быть представлено в виде $m = 2p$ для какого-то целого числа p. Подставив эту информацию в равенство $m^2 = 2n^2$, мы получим, что $4p^2 = 2n^2$, т. е. $n^2 = 2p^2$. Но тогда n тоже является четным числом. Таким образом, мы показали, что как m, так и n — четные числа. Поэтому они обладают

общим делителем 2. Если же теперь вспомнить, что мы предполагали отсутствие общего делителя у числителя и знаменателя дроби $\frac{m}{n}$, то увидим явное противоречие.

Найденное противоречие приводит нас к однозначному выводу: решение уравнения $x^2 = 2$ не может быть рациональным числом, т. е. оно иррационально.

2.4. Математическая индукция

Компьютерную программу в информатике называют *правильной* или *корректной*, если она делает то, что указано в ее спецификации. Несмотря на то, что тестирование программы может давать ожидаемый результат в случае каких-то отдельных начальных данных, необходимо доказать приемами формальной логики, что правильные выходные данные будут получаться при любых вводимых начальных значениях. О доказательствах такого сорта будет рассказано в приложении, размещенном в конце этой главы.

Проверка корректности алгоритма, содержащего циклы, нуждается в довольно мощном методе доказательства, который называется «*математическая индукция*». Продемонстрируем преимущества этого важного метода, доказав корректность следующего рекуррентного алгоритма, определяющего максимальный элемент из набора a_1, a_2, a_3, ..., a_n натуральных чисел.

```
begin
    i := 0;
    M := 0;
        while i < n do
            begin
                j := j + 1;
                M := max(M, a);
            end
end
```

Действие алгоритма на наборе данных: $a_1 = 4$, $a_2 = 7$, $a_3 = 3$ и $a_4 = 8$ прослежено в табл. 2.7.

Таблица 2.7

j	M	$j < 4$?
0	0	Да
1	4	Да
2	7	Да
3	7	Да
4	8	Нет

В качестве выходных данных мы получили $M = 8$, что безусловно правильно. Заметим, что после каждого прохода цикла переменная M равна наибольшему из чисел набора, просмотренных к этому моменту.

Но будет ли алгоритм работать правильно при любом вводимом наборе чисел длины n?

Рассмотрим вводимый набор $a_1, a_2, a_3, \ldots, a_n$ длины n и обозначим через M_k значение переменной M после k-го прохода цикла.

1. Если мы вводим набор a_1 длины 1, то цикл сделает только один проход и M присвоится наибольшее значение из 0 и a_1, которым, очевидно, будет a_1 (натуральные числа больше 0). В этом простом случае вывод будет правильным.

2. Если после k-го прохода цикла M_k — наибольший элемент из набора $a_1, a_2, \ldots, a_k$, то после следующего прохода M_{k+1} будет равно $\max(M_k, a_{k+1})$, т. е. максимальному элементу набора $a_1, a_2, \ldots, a_k, a_{k+1}$.

В п. 1 мы показали, что алгоритм работает правильно при любом вводимом наборе длины 1. Поэтому согласно п. 2, он будет правильно работать и на любом наборе длины 2. Вновь применяя п. 2 рассуждений, мы убеждаемся, что алгоритм работает правильно и на любых наборах длины 3, и т. д. Таким образом, алгоритм правильно работает на любых наборах произвольной длины n, т. е. он корректен.

На формальном языке использованный метод доказательства выглядит следующим образом.

Принцип математической индукции

Пусть $P(n)$ — предикат, определенный для всех натуральных чисел n.
Предположим, что

1. *$P(1)$ истинно и*
2. *$\forall k \geqslant 1$ импликация $(P(k) \Rightarrow P(k+1))$ верна.*

Тогда $P(n)$ истинно при любом натуральном значении n.

Пример 2.13. Докажите по индукции, что равенство

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

выполнено при всех натуральных n.

Решение. Пусть $P(n)$ — предикат $1+2+\cdots+n=\frac{n(n+1)}{2}$.

В случае $n=1$ левая часть равенства — просто 1, а вычисляя правую часть, получаем

$$\frac{1(1+1)}{2}=1.$$

Следовательно, $P(1)$ истинно.

Предположим теперь, что равенство $1+2+\cdots+k=\frac{k(k+1)}{2}$ имеет место для какого-то натурального числа k. Тогда

$$\begin{aligned}1+2+\cdots+k+(k+1) &= (1+2+\cdots+k)+(k+1)= \\ &= \frac{k(k+1)}{2}+(k+1)= \\ &= \frac{1}{2}\big(k(k+1)+2(k+1)\big)= \\ &= \frac{1}{2}\big((k+2)(k+1)\big)= \\ &= \frac{(k+1)(k+2)}{2}.\end{aligned}$$

Таким образом, при любом натуральном k импликация

$$P(k)\Rightarrow P(k+1)$$

справедлива. Значит, по принципу математической индукции, предикат $P(n)$ имеет истинное значение при всех натуральных n.

Пример 2.14. Методом математической индукции докажите, что 7^n-1 делится на 6 при любом натуральном показателе n.

Решение. Прежде всего напомним, что целое число a делится на целое число b тогда и только тогда, когда выполняется равенство $a=mb$ при каком-то целом числе m. Например, 51 делится на 17, поскольку $51=3\cdot 17$. Кроме того, для наших рассуждений потребуется простое свойство делимости чисел, которое утверждает, что сумма делящихся на b чисел делится на b.

Пусть $P(n)$ обозначает предикат «7^n-1 делится на 6».

При $n=1$ имеем

$$7^n-1=7-1=6,$$

т. е. предикат $P(1)$ имеет истинное значение.

Предположим, что 7^k-1 делится на 6 при каком-то натуральном k. Тогда

$$\begin{aligned} 7^{k+1} - 1 &= 7(7^k) - 1 = \\ &= 7(7^k - 1) + 7 - 1 = \\ &= 7(7^k - 1) + 6. \end{aligned}$$

Так как $7^k - 1$ делится на 6, то по упомянутому свойству делимости сумма $7(7^k - 1) + 6$ тоже делится на 6.

Итак, $7^{k+1} - 1$ делится на 6, так что при любом натуральном k импликация $(P(k) \Rightarrow P(k+1))$ истинна.

Индуктивным рассуждением мы доказали истинность предиката $P(n)$ для всех натуральных n.

Пример 2.15. Последовательность целых чисел x_1, x_2, ..., x_n определена рекуррентной формулой:

$$x_1 = 1 \qquad \text{и} \qquad x_{k+1} = x_k + 8k \quad \text{при } k \geqslant 1.$$

Доказать, что имеет место формула: $x_n = (2n-1)^2$ для всех $n \geqslant 1$.

Решение. Предикат $x_n = (2n-1)^2$ обозначим через $P(n)$. Если $n = 1$, то $(2n-1)^2 = (2-1)^2 = 1$, что показывает истинность высказывания $P(1)$.

Допустим теперь, что $x_k = (2k-1)^2$ для некоторого $k \geqslant 1$. Тогда

$$\begin{aligned} x_{k+1} &= x_k + 8k = \\ &= (2k-1)^2 + 8k = \\ &= 4k^2 + 4k + 1 = \\ &= (2k+1)^2. \end{aligned}$$

Мы видим, что $x_{k+1} = \big(2(k+1)-1\big)^2$ и поэтому истинность импликации $(P(k) \Rightarrow P(k+1))$ доказана при всех $k \geqslant 1$. Следовательно, согласно индуктивному принципу, предикат $P(n)$ превращается в истинное высказывание при любом натуральном значении переменной n.

Набор упражнений 2

2.1. Пусть P, Q и R — определенные следующим образом высказывания:

P: Я умираю от жажды.

Q: Мой стакан пуст.
R: Сейчас три часа.

Запишите каждое из следующих высказываний как логическое выражение, включающее P, Q и R.

(а) Я умираю от жажды и мой стакан не пуст.

(б) Сейчас три часа, а я умираю от жажды.

(в) Если сейчас три часа, то я умираю от жажды.

(г) Если я умираю от жажды, то мой стакан пуст.

(д) Если я не умираю от жажды, то мой стакан не пуст.

2.2. Обозначим через P высказывание: «розы красные», а через Q — «фиалки синие». Запишите каждое из следующих высказываний:

(а) если розы не красные, то фиалки не синие;

(б) розы красные или фиалки не синие;

(в) либо розы красные, либо фиалки синие (но не одновременно)

как логическое выражение.

Используя таблицы истинности, докажите логическую эквивалентность высказываний (а) и (б).

2.3. Составные высказывания, принимающие истинные значения при любых истинностных значениях своих компонент, называются *тавтологиями*. С помощью таблиц истинности найдите тавтологии среди следующих высказываний:

(а) **не** (P **и** (**не** P));

(б) $P \Rightarrow$ (**не** P);

(в) (P **и** ($P \Rightarrow Q$)) $\Rightarrow Q$.

2.4. Покажите, что высказывание $(P \Rightarrow Q) \Rightarrow R$ логически эквивалентно высказыванию ((**не** P) $\Rightarrow R$) **и** ($Q \Rightarrow R$).

2.5. Обозначим через x слово «кошка», а через $P(x)$ предикат «у x есть усы». Запишите каждое из высказываний в символьной форме:

(а) усы есть у всех кошек;

(б) найдется кошка без усов;

(в) не бывает кошек с усами.

Запишите отрицание высказывания (б) в символьной форме, а отрицание высказывания (в) запишите как символами, так и словами.

2.6. Пусть $P(x)$ означает «x высокий», а $Q(x)$ — «x толстый», где x — какой-то человек. Прочитайте высказывание:

$$\forall\, x\ (P(x)\ \textbf{и}\ Q(x)).$$

Найдите его отрицание среди следующих утверждений:

(а) найдется некто короткий и толстый;

(б) нет никого высокого и худого;

(в) найдется некто короткий или худой.

2.7. (а) Прямым рассуждением докажите истинность высказывания:

$$n \text{ и } m \text{ — четные числа} \Rightarrow n+m \text{ — число четное.}$$

(б) Дайте обратное доказательство высказывания:

$$n^2 \text{ — четное число} \Rightarrow n \text{ — четное.}$$

(в) Методом «от противного» докажите, что

$$n+m \text{ — нечетное число} \Rightarrow \text{одно из слагаемых является четным, а другое — нечетным.}$$

2.8. Докажите каждое из высказываний методом математической индукции.

(а) $1+5+9+\cdots+(4n-3)=n(2n-1)$ для всех натуральных чисел n.

(б) $1^2+2^2+\cdots+n^2=\frac{1}{6}n(n+1)(2n+1)$ для всех натуральных чисел n.

(в) $\frac{1}{1\cdot3}+\frac{1}{3\cdot5}+\cdots+\frac{1}{(2n-1)\cdot(2n+1)}=\frac{n}{2n+1}$ для всех натуральных чисел n.

(г) Число n^3-n делится на 3 при всех натуральных значениях числа n.

(д) $1\cdot1!+2\cdot2!+\cdots+n\cdot n!=(n+1)!-1$ для всех натуральных чисел n.

(Символ $n!$ читается как «n факториал» и обозначает произведение всех натуральных чисел от 1 до n включительно: $n!=1\cdot2\cdot3\cdots(n-1)\cdot n$.)

2.9. Последовательность натуральных чисел $x_1, x_2, \ldots, x_n$ определяется рекуррентной формулой

$$x_1 = 1 \qquad \text{и} \qquad x_{k+1} = \frac{x_k}{x_k + 2} \quad \text{при } k \geqslant 1.$$

Вычислите x_2, x_3 и x_4. Докажите по индукции, что

$$x_n = \frac{1}{2^n - 1}$$

для всех $n \geqslant 1$.

2.10. Последовательность натуральных чисел $x_1, x_2, \ldots, x_n$ определяется рекуррентной формулой

$$x_1 = 1,\ x_2 = 2 \qquad \text{и} \qquad x_{k+1} = 2x_k - x_{k-1} \quad \text{при } k > 1.$$

Вычислите x_3, x_4 и x_5. Найдите общую формулу для x_n и докажите ее истинность индуктивным методом.

Краткое содержание главы

Логика представляет собой набор правил для получения обоснованных выводов.

Высказыванием называется утверждение, имеющее истинностное значение, т. е. оно может быть *истинным* или *ложным*.

Составное высказывание может быть построено из других с помощью логических операций. Наиболее употребительными операциями являются **и**, **или**, **если ... то** и **не**.

В табл. 2.8 сведены **таблицы истинности** логических операций **и**, **или** и $\Rightarrow$.

Таблица 2.8

P	Q	P **и** Q	P **или** Q	$(P \Rightarrow Q)$
И	И	И	И	И
И	Л	Л	И	Л
Л	И	Л	И	И
Л	Л	Л	Л	И

Два составных высказывания называются **логически эквивалентными**, если они принимают одинаковые значения истинности на любом наборе истинностных значений своих составных частей.

Высказывание о свойствах переменной x называют **предикатом** и обозначают, например, так: $P(x)$.

Для всех ($\forall$) и **существует** ($\exists$) — это **кванторы**.

При доказательстве **прямым рассуждением** утверждения вида $(P \Rightarrow Q)$ из предположения об истинности P выводят истинность Q.

Обратное рассуждение в доказательстве основано на логической эквивалентности высказываний (**не** $Q \Rightarrow$ **не** P) и $(P \Rightarrow Q)$.

Метод доказательства импликации $(P \Rightarrow Q)$, при котором из предположения о ложности Q и истинности P приходят к противоречию, называют методом **«от противного»**.

Математическая индукция полезна при доказательстве высказывания, истинного для всех натуральных чисел.

Принцип математической индукции — это следующая теорема:

> *Пусть $P(n)$ — предикат, определенный для всех натуральных n.*
> *Предположим, что*
>
> 1. $P(1)$ *истинно и*
> 2. $\forall k \geqslant 1$ *импликация* $(P(k) \Rightarrow P(k+1))$ *верна.*
>
> *Тогда $P(n)$ истинно при любом натуральном значении n.*

Приложение. Корректность алгоритмов

Чтобы доказать корректность алгоритма (иными словами, убедиться, что он делает именно то, что и предусмотрено), нам нужно проверить все изменения переменных, в нем используемых *до*, *в течение* и *после* работы алгоритма. Эти изменения и условия можно рассматривать как небольшие утверждения или предикаты.

Пусть P — предикат, истинный для входных данных алгоритма A, и Q — предикат, описывающий условия, которым должны удовлетворять выходные данные. Высказывание $\{P\}\,A\,\{Q\}$ означает, что «если работа алгоритма A начинается с истинного значения предиката P, то она закончится при истинном значении Q». Предикат P называется *входным условием* или *предусловием*, а Q — *выходным*

условием или *постусловием*. Высказывание $\{P\}\,A\,\{Q\}$ само является предикатом. Поэтому доказательство корректности алгоритма A равносильно доказательству истинности $\{P\}\,A\,\{Q\}$. Для простых алгоритмов это делается достаточно прямолинейно.

Задача 1. Докажите корректность алгоритма ***Разность***.

Разность
begin
$z := x - y;$
end

Решение. В данном случае предусловием P являются равенства: $x = x_1$ и $y = y_1$. Постусловие Q — это $z = x_1 - y_1$. Предикат

$$\{P\}\,\textbf{\textit{Разность}}\,\{Q\}$$

читается как «если $x = x_1$ и $y = y_1$, то $z = x_1 - y_1$». Истинность последнего предиката легко проверяется подстановкой $x = x_1$ и $y = y_1$ в тело алгоритма, содержащего переменные z, x и y. С формальной точки зрения соотношения: $z = x - y$, $x = x_1$ и $y = y_1$ влекут тождество $z = x_1 - y_1$.

Когда в алгоритме A происходит много различных действий с переменными, мы разбиваем его на подходящие отрезки $A_1, \ldots, A_n$ и доказываем цепочку утверждений вида

$$\{P\}\,A_1\,\{Q_1\},\ \{Q_1\}\,A_2\,\{Q_2\},\ \ldots,\ \{Q_{n-1}\}\,A_n\,\{Q\},$$

где постусловие любого отрезка служит предусловием следующего.

Задача 2. Докажите правильность алгоритма ***«Квадратный многочлен»***.

Квадратный многочлен
$\{x$ — *вещественное число*$\}$
begin
$y := ax;$
$y := (y + b)x;$
$y := y + c;$
end
$\{y = ax^2 + bx + c\}$

Решение. Разобьем алгоритм на кусочки, зафиксировав при этом обозначения пред- и постусловий.

$P \to$ $\{x = x_1\}$
begin
$y := ax;$
$Q_1 \to$ $\{y = ax_1$ **и** $x = x_1\}$
$y := (y + b)x;$
$Q_2 \to$ $\{y = ax_1^2 + bx_1\}$
$y := y + c;$
end
$Q \to$ $\{y = ax_1^2 + bx_1 + c\}$

Подстановки, сделанные выше, показывают, что все высказывания:

$\{P\}$ $\boldsymbol{y := ax}$ $\{Q_1\}$,
$\{Q_1\}$ $\boldsymbol{y := (y + b)x}$ $\{Q_2\}$,
$\{Q_2\}$ $\boldsymbol{y := y + c}$ $\{Q\}$, —

верны. Следовательно, предикат

$$\{P\}\ \textbf{\textit{Квадратный многочлен}}\ \{Q\}$$

истинен, т. е. алгоритм ***Квадратный многочлен*** корректен.

Алгоритм с условными высказываниями тоже поддается технике доказательства. Когда в алгоритме появляется условный оператор **if ... then**, во входных и выходных условиях должны быть отражены альтернативные пути через весь алгоритм.

Более точно: предположим, что условное составное высказывание

if *условие* **then**
высказывание 1;
else
высказывание 2;

вводит предусловие P, а на выходе дает условие Q. Тогда следует доказать истинность обоих предикатов:

$$\{P\ \textbf{и}\ \text{условие}\}\ \textbf{\textit{высказывание 1}}\ \{Q\}$$

и

$$\{P\ \textbf{и не}\ (\text{условие})\}\ \textbf{\textit{высказывание 2}}\ \{Q\}.$$

Задача 3. Докажите, что алгоритм ***Модуль*** корректен.

Модуль
{x — *вещественное число*}
begin
  if $x \geqslant 0$ **then**
    $abc := x$;
  else
    $abs := -x$;
end
{abs — *модуль числа* x}

Решение. Предусловием P в нашем алгоритме служит $\{x = x_1\}$, а соответствующим постусловием Q является {abs — модуль числа x}.

Предикат $\{P$ **и** $x \geqslant 0\}$ $\boldsymbol{abs := x}$ $\{Q\}$ имеет истинное значение, поскольку модуль неотрицательного числа x_1 совпадает с ним самим.

Предикат $\{P$ **и не** $(x \geqslant 0)\}$ $\boldsymbol{abs := -x}$ $\{Q\}$ тоже истинен, так как модуль отрицательного числа x_1 отличается от него знаком.

Использование пред- и постусловий при проверке алгоритмов, в которых участвуют циклы типа **while ... do**, довольно громоздко. Предпочтительнее доказывать корректность таких алгоритмов методом математической индукции.

Задача 4. Докажите по индукции корректность алгоритма ***Квадрат***.

Квадрат
{n — *натуральное число*}
begin
  $sq := 0$;
  for $i := 1$ **to** n **do**
    $sq := sq + 2i - 1$;
end
$\{sq = n^2\}$

Решение. Пусть $P(n)$ обозначает предикат «$sq = n^2$ после n-го прохода цикла», а sq_k — значение переменной sq после k-го прохода цикла. Покажем, что

(1) $sq_1 = 1^2$;

(2) если $sq_k = k^2$, то $sq_{k+1} = (k+1)^2$.

Очевидно, что после первого прохода цикла $sq_1 = 1$ и пункт (1) выполнен. Предположим, что после k-ой петли цикла $sq_k = k^2$. Тогда после следующего прохода

$$sq_{k+1} = sq_k + 2(k+1) - 1 = k^2 + 2k + 1 = (k+1)^2.$$

Таким образом, пункт (2) тоже имеет место.

Итак, мы установили, что $P(1)$ истинно (п. (1)). Кроме того, по второму пункту импликация $((P(k) \Rightarrow P(k+1))$ справедлива при любом $k \geqslant 1$. Следовательно, согласно принципу математической индукции, $P(n)$ истинно для всех натуральных n.

В задаче 4 цикл **for** ограничен определенным числом итераций (проходов). В том случае, когда число петель цикла заранее не определено, как в цикле **while ... do**, при доказательстве индукцией следует предположить, что число проходов все же ограничено и показать правильность выходных данных. После чего необходимо будет проверить, что число петель такого цикла действительно конечно.

ГЛАВА 3

ТЕОРИЯ МНОЖЕСТВ

Теория множеств — один из краеугольных камней математики, обеспечивающий удобный язык для описания массы концепций как в математике, так и в и информатике.

В этой главе мы введем понятие множества и опишем различные способы комбинирования разных множеств для получения новых. Результат операций объединения, пересечения, дополнения и симметрической разности иллюстрируется на диаграммах Венна. Аналогии, которые мы обнаружим между операциями над множествами и логическими операциями из предыдущей главы, побудят нас сформулировать определенный набор тождеств. Некоторое число этих тождеств, собранных вместе, определяют законы алгебры множеств и, в свою очередь, будут использованы для вывода более сложных соотношений. Здесь мы освоим такие новые понятия, как, например, принцип включения исключения, упорядоченные пары, декартово произведение (последнее используется для имитации операций над множествами манипулированием строками бит). Они нам потребуются для освоения последующего материала.

В приложении к этой главе с помощью предикатов и множеств будет создана простая база знаний для извлечения информации из базы данных о британских королях и королевах, начиная с Георга I.

3.1. Множества и операции над ними

Множество — это совокупность объектов, называемых *элементами* множества. Например,

- {Эссекс, Йоркшир, Девон};
- {2, 3, 5, 7, 11};
- {сыр, яйцо, молоко, сметана}.

В этом примере элементы каждого множества заключены в *фигурные скобки*. Чтобы обеспечить возможность ссылок, мы обычно будем обозначать множества прописными латинскими буквами. На-

пример, $S = \{3,\ 2,\ 11,\ 5,\ 7\}$ — множество, содержащее данные элементы. Заметим, что множество S совпадает с одним из множеств, выписанных выше, поскольку порядок, в котором записываются элементы множества, значения не имеет.

В общем случае запись $a \in S$ означает, что объект a — элемент множества S. Часто говорят, что a принадлежит множеству S. Если объект a не принадлежит S, то пишут: $a \notin S$.

Мы не можем выписать все элементы очень больших, в особенности бесконечных множеств. В этом случае множества определяются с помощью подходящих предикатов. Формально мы пишем

$$S = \{x :\ P(x)\}$$

для описания множества, состоящего из элементов x, для которых предикат $P(x)$ имеет истинное значение. Например, запись

$$S = \{x :\ x \text{ — нечетное натуральное число}\}$$

описывает множество

$$S = \{1,\ 3,\ 5,\ 7,\ \ldots\}.$$

Поскольку любое натуральное нечетное число может быть записано в виде $2n - 1$, где n — любое натуральное число, альтернативное допустимое определение того же множества задается формулой:

$$S = \{2n - 1 :\ n \text{ — натуральное число}\}.$$

Пример 3.1. Найдите более простое описание множеств, перечисляющее их элементы.

(а) $A = \{x :\ x$ — целое и $x^2 + 4x = 12\}$;

(б) $B = \{x :\ x$ — название дня недели, не содержащее буквы «е»$\}$;

(в) $C = \{n^2 :\ n$ — целое$\}$.

Решение.

(а) Если $x^2 + 4x = 12$, то $x(x+4) = 12$. Поскольку x — целое число, делящее 12, то оно может быть равно только ± 1, ± 2, ± 3, ± 4, ± 6 и ± 12. С другой стороны, $x + 4$ тоже делит 12. Поэтому остается только два значения: $x = -6$ или $x = 2$.

Другой способ решения заключается в отыскании корней квадратного уравнения $x^2 + 4x - 12 = 0$. Он приводит к тому же ответу $x = -6$ или $x = 2$.

Следовательно, $A = \{-6,\ 2\}$.

(б) $B = \{\text{вторник, пятница, суббота}\}$.

(в) $C = \{0,\ 1,\ 4,\ 9,\ 16,\ \ldots\}$.

Некоторые множества чисел столь часто используются, что имеют стандартные названия и обозначения.

$\varnothing$ — пустое множество;

$\mathbb{N} = \{1,\ 2,\ 3,\ \ldots\}$ — множество *натуральных чисел*;

$\mathbb{Z} = \{0,\ \pm 1,\ \pm 2,\ \pm 3,\ \ldots\}$ — множество *целых чисел*;

$\mathbb{Q} = \{\frac{p}{q} :\ p,\, q \in \mathbb{Z},\ q \neq 0\}$ — множество *рациональных чисел*;

$\mathbb{R} = \{\text{все десятичные дроби}\}$ — множество *вещественных чисел*.

Читатель должен учитывать, что в некоторых книгах натуральные числа $\mathbb{N}$ включают и 0.

В современных языках программирования требуется, чтобы переменные объявлялись как принадлежащие к определенному *типу данных*. Тип данных представляет собой множество объектов со списком стандартных операций над ними. Определение типа переменных равносильно указанию множества, из которого переменным присваиваются значения.

Существует несколько способов конструирования нового множества из двух данных. Опишем коротко эти *операции* на множествах. Прежде всего отметим, что в вышеприведенных примерах все элементы некоторых множеств принадлежали другим бóльшим множествам. Например, все элементы множества $C = \{0,\ 1,\ 4,\ 9,\ 16,\ \ldots\}$ содержатся в множестве $\mathbb{Z} = \{0,\ \pm 1,\ \pm 2,\ \pm 3,\ \ldots\}$.

Говорят, что множество A является *подмножеством* множества S, если каждый его элемент автоматически является элементом множества S. Довольно часто при этом говорят, что множество A *содержится* в множестве S. Этот факт обозначают так: $A \subset S$. На рис. 3.1 дана иллюстрация этого определения. Такого сорта картинки называются диаграммами Венна.

Два множества считаются *равными*, если каждое из них содержится в другом. Поэтому для доказательства равенства множеств нам нужно показать, что они состоят из одних и тех же элементов. На формальном языке для равенства множеств $A = B$ необходимо проверить истинность двух импликаций:

$$\{x \in A \Rightarrow x \in B\} \qquad \text{и} \qquad \{x \in B \Rightarrow x \in A\}.$$

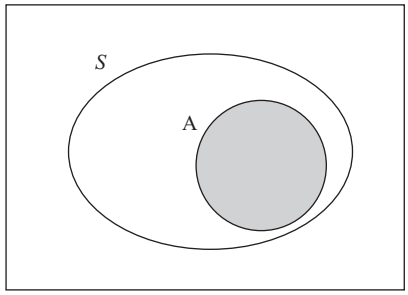


Рисунок 3.1. Диаграмма Венна подмножества $A \subset S$

Пример 3.2. Пусть

$$A = \{n : \ n^2 \text{ — нечетное целое число}\}$$

и

$$B = \{n : \ n \text{ — нечетное целое число}\}.$$

Показать, что $A = B$.

Решение. Если $x \in A$, то x^2 — нечетное целое число. Как мы проверили в примере 2.11, отсюда вытекает, что само число x — целое и нечетное. Следовательно, $x \in B$, т. е. $A \subset B$.

В обратную сторону, пусть $x \in B$. Тогда x — нечетное целое число. Согласно примеру 2.10, в этом случае x^2 тоже будет нечетным целым числом, а значит, $x \in A$. В виду произвольности взятого элемента $x \in B$ мы можем утверждать, что все элементы из B принадлежат A, т. е. $B \subset A$. Итак, $A = B$.

Объединением двух множеств A и B называется множество

$$A \cup B = \{x : \ x \in A \quad \textbf{или} \quad x \in B\}.$$

Оно состоит из тех элементов, которые принадлежат либо множеству A, либо множеству B, а возможно и обоим сразу. Диаграмма Венна объединения показана на рис. 3.2.

Пересечением двух множеств A и B называется множество

$$A \cap B = \{x : \ x \in A \quad \textbf{и} \quad x \in B\}.$$

Оно состоит из элементов, которые принадлежат как множеству A, так и множеству B. Диаграмма Венна пересечения приведена на рис. 3.3.

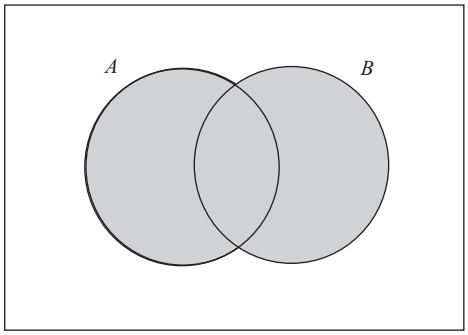


Рисунок 3.2. Диаграмма Венна объединения $A \cup B$

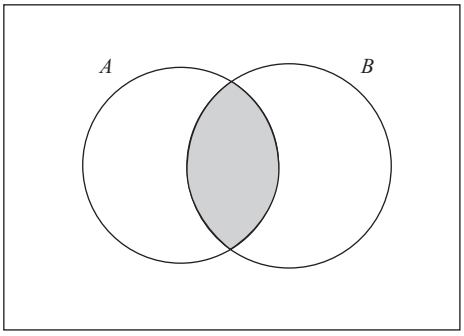


Рисунок 3.3. Диаграмма Венна пересечения $A \cap B$

Дополнением[1] множества B до множества A называется

$$A \setminus B = \{x : \ x \in A \ \textbf{и} \ x \notin B\}.$$

Дополнение $A \setminus B$ состоит из всех элементов множества A, которые не принадлежат B (см. рис. 3.4).

Если мы оперируем подмножествами некоего большого множества U, мы называем U *универсальным множеством* для данной задачи. На наших диаграммах Венна прямоугольник как раз и символизирует это универсальное множество.

Для подмножества A универсального множества U можно рассматривать дополнение A до U, т. е. $U \setminus A$. Поскольку в каждой конкретной задаче универсальное множество фиксировано, множество $U \setminus A$ обычно обозначают $\overline{A}$ и называют просто дополнением

[1] Довольно часто эту же операцию называют *разностью* множеств. — *Прим. перев.*

множества A. Таким образом, понимая, что мы работаем с подмножествами универсального множества U, можно записать

$$\overline{A} = \{x : \ \textbf{не}\ (x \in A)\} \quad \Leftrightarrow \quad \overline{A} = \{x : \ x \notin A\}.$$

Диаграмма Венна дополнения изображена на рис. 3.5.

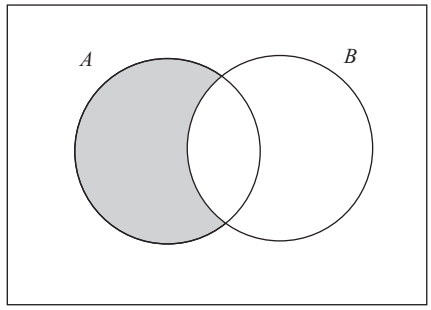


Рисунок 3.4. Диаграмма Венна разности $A \setminus B$

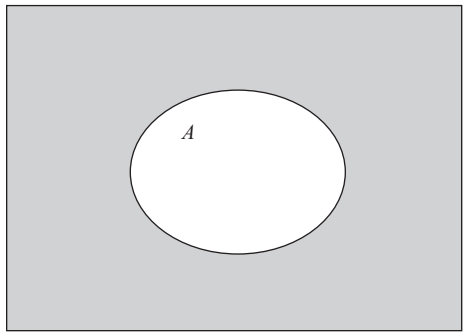


Рисунок 3.5. Диаграмма Венна дополнения $\overline{A}$

Симметрической разностью двух множеств A и B называют множество

$$A \triangle B = \{x : \ (x \in A \ \ \textbf{и}\ x \notin B) \ \ \textbf{или} \ \ (x \in B \ \ \textbf{и} \ \ x \notin A)\}.$$

Оно состоит из всех тех и только тех элементов универсального множества, которые либо принадлежат A и не принадлежат B, либо наоборот, принадлежат B, но не A. Грубо говоря, симметрическая разность состоит из элементов, лежащих либо в A, либо в B, но не одновременно. Диаграмма Венна, иллюстрирующая новое понятие, начерчена на рис. 3.6.

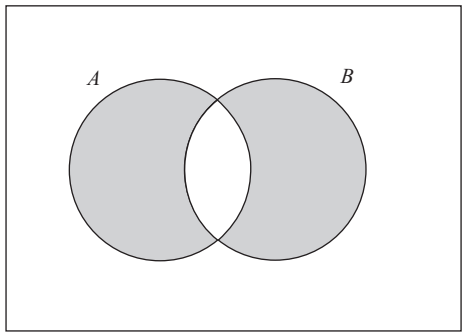


Рисунок 3.6. Диаграмма Венна симметрической разности $A \triangle B$

Пример 3.3. Пусть

$$A = \{1,\ 3,\ 5,\ 7\}; \quad B = \{2,\ 4,\ 6,\ 8\}; \quad C = \{1,\ 2,\ 3,\ 4,\ 5\}.$$

Найдите $A \cup C$, $B \cap C$, $A \setminus C$ и $B \triangle C$.

Решение.

$A \cup C = \{1,\ 3,\ 5,\ 7,\ 2,\ 4\}$;

$B \cap C = \{2,\ 4\}$;

$A \setminus C = \{7\}$;

$B \triangle C = (B \setminus C) \cup (C \setminus B) = \{6,\ 8\} \cup \{1,\ 3,\ 5\} = \{6,\ 8,\ 1,\ 3,\ 5\}$.

Пример 3.4. Пусть

$$A = \{x:\ 1 \leqslant x \leqslant 12 \text{ и } x \text{ четное целое число}\},$$
$$B = \{x:\ 1 \leqslant x \leqslant 12 \text{ и } x \text{ целое число, кратное } 3\}.$$

Убедитесь, что $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$.

Решение. Прежде всего заметим, что универсальным множеством здесь служит

$$U = \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12\}.$$

Кроме того,

$$A = \{2,\ 4,\ 6,\ 8,\ 10,\ 12\} \quad \text{и} \quad B = \{3,\ 6,\ 9,\ 12\}.$$

Поэтому

$$\overline{(A \cap B)} = \overline{\{6,\ 12\}} = \{1,\ 2,\ 3,\ 4,\ 5,\ 7,\ 8,\ 9,\ 10,\ 11\}$$

и

$$\begin{aligned}\overline{A} \cup \overline{B} &= \{1,\ 3,\ 5,\ 7,\ 9,\ 11\} \cup \{1,\ 2,\ 4,\ 5,\ 7,\ 8,\ 10,\ 11\} = \\ &= \{1,\ 2,\ 3,\ 4,\ 5,\ 7,\ 8,\ 9,\ 10,\ 11\}.\end{aligned}$$

Следовательно, $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$.

3.2. Алгебра множеств

Из операций, о которых шла речь в предыдущем параграфе, можно вывести многочисленные свойства множеств. Некоторые из них кажутся очевидными, другие — меньше, но все требуют доказательства. Доказательства будут основываться на соответствии[1] между операциями на множествах и логическими операциями над предикатами, которое отражено в табл. 3.1.

Таблица 3.1

Операции над множествами	Логические операции
—	**не**
$\cup$	**или**
$\cap$	**и**
$\subset$	$\Rightarrow$

Пример 3.5. Докажите, что для любых множеств A и B имеет место соотношение: $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$.

Решение.

$$\begin{aligned}\overline{(A \cap B)} &= \{x:\ x \notin (A \cap B)\} = \\ &= \{x:\ \textbf{не}\ (x \in (A \cap B))\} = \\ &= \{x:\ \textbf{не}\ \big((x \in A)\ \textbf{и}\ (x \in B)\big)\};\end{aligned}$$

$$\begin{aligned}\overline{A} \cup \overline{B} &= \{x:\ (x \notin A)\ \textbf{или}\ (x \notin B)\} = \\ &= \{x:\ (\textbf{не}\ (x \in A))\ \textbf{или}\ (\textbf{не}\ (x \in B))\}.\end{aligned}$$

Сравнивая таблицы истинности, легко установить логическую эквивалентность составных предикатов:

$$\textbf{не}\ (P\ \textbf{и}\ Q) \qquad \text{и} \qquad (\textbf{не}\ P)\ \textbf{или}\ (\textbf{не}\ Q),$$

[1] Строго говоря, его тоже необходимо обосновать. — *Прим. перев.*

где P и Q — простые высказывания. Опираясь теперь на соответствие между логическими операциями и операциями над множествами (табл. 3.1), можно увидеть, что предикат **не** (P **и** Q) соответствует множеству $\overline{(A \cap B)}$, а (**не** P) **или** (**не** Q) — множеству $\overline{A} \cup \overline{B}$. Следовательно, $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$.

Свойство, доказанное в примере 3.5, известно, как один из законов де Моргана. Фундаментальные свойства, аналогичные законам де Моргана, составляют законы *алгебры множеств*. Эти свойства перечислены в таблице 3.2. Каждое из них может быть доказано с помощью логических аргументов, аналогичных тем, что использованы в примере 3.5.

Внимательное изучение свода законов алгебры множеств (табл. 3.2) позволяет заметить, что каждое из тождеств правой колонки может быть получено из соответствующего тождества левой путем замены $\cup$ на $\cap$, $\varnothing$ на U и наоборот. Такое соответствие тождеств называется *законом двойственности*, а соответствующие тождества — *двойственными* друг другу.

Таблица 3.2. Законы алгебры множеств

Законы ассоциативности	
$A \cup (B \cup C) = (A \cup B) \cup C$	$A \cap (B \cap C) = (A \cap B) \cap C$
Законы коммутативности	
$A \cup B = B \cup A$	$A \cap B = B \cap A$
Законы тождества	
$A \cup \varnothing = A$	$A \cap U = A$
$A \cup U = U$	$A \cap \varnothing = \varnothing$
Законы идемпотентности	
$A \cup A = A$	$A \cap A = A$
Законы дистрибутивности	
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
Законы дополнения	
$A \cup \overline{A} = U$	$A \cap \overline{A} = \varnothing$
$\overline{U} = \varnothing$	$\overline{\varnothing} = U$
$\overline{\overline{A}} = A$	$\overline{\overline{A}} = A$
Законы де Моргана	
$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$	$\overline{(A \cap B)} = \overline{A} \cup \overline{B}$

Закон двойственности является довольно сложной теоремой алгебры множеств. Его доказательство выходит за рамки данного

простого учебника. Однако, приняв его на веру, можно упростить себе жизнь, поскольку доказав какое-то тождество множеств и обратив операции, можно обосновать и двойственное тождество.

Пример 3.6. Опираясь на законы алгебры множеств, докажите, что произвольные множества A и B удовлетворяют свойству:

$$A \triangle B = (A \cup B) \cap \overline{(A \cap B)}.$$

Решение. Напомним определение симметрической разности множеств:

$$A \triangle B = (A \cap \overline{B}) \cup (B \cap \overline{A}).$$

Согласно законам алгебры множеств, имеем.

$$\begin{aligned}
&(A \cup B) \cap \overline{(A \cap B)} = && \text{(з. де Моргана)}\\
&= (A \cup B) \cap (\overline{A} \cup \overline{B})) = && \text{(з. дистрибутивн.)}\\
&= ((A \cup B) \cap \overline{A}) \cup ((A \cup B) \cap \overline{B}) = && \text{(з. коммутативн.)}\\
&= (\overline{A} \cap (A \cup B)) \cup (\overline{B} \cap (A \cup B)) = && \text{(з. дистрибутивн.)}\\
&= ((\overline{A} \cap A) \cup (\overline{A} \cap B)) \cup ((\overline{B} \cap A) \cup (\overline{B} \cap B)) = && \text{(з. коммутативн.)}
\end{aligned}$$

$$\begin{aligned}
&= ((A \cap \overline{A}) \cup (B \cap \overline{A})) \cup ((A \cap \overline{B}) \cup (B \cap \overline{B})) = && \text{(з. дополнения)}\\
&= (\varnothing \cup (B \cap \overline{A})) \cup ((A \cap \overline{B}) \cup \varnothing) = && \text{(з. коммутативн.}\\
& && \text{и тождества)}\\
&= (A \cap \overline{B}) \cup (B \cap \overline{A})
\end{aligned}$$

Следовательно,

$$A \triangle B = (A \cup B) \cap \overline{(A \cap B)},$$

как и утверждалось.

3.3. Дальнейшие свойства множеств

В главе 6 мы будем изучать *комбинаторику*, область математики, имеющую дело с подсчетом количества элементов в тех или иных множествах. Вопросы пересчета становятся очень важными, когда у Вас ограничены ресурсы. Например, сколько пользователей может поддерживать данная компьютерная сеть? Или сколько операций будет сделано при работе данного алгоритма?

Мощностью конечного множества S называется число его элементов. Она обозначается символом $|S|$.

Следующая теорема дает простое правило вычисления мощности объединения двух множеств. Используя индукцию, его можно обобщить на произвольное конечное число множеств.

Формула включений и исключений.

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Доказательство. Как показано на рис. 3.7, множество $A \cup B$ состоит из подмножеств: $A \setminus B$, $A \cap B$ и $B \setminus A$, которые не имеют общих элементов. Более того,

$$A = (A \setminus B) \cup (A \cap B) \qquad \text{и} \qquad B = (B \setminus A) \cup (A \cap B).$$

Введем обозначения:

$$|A \setminus B| = m, \qquad |A \cap B| = n, \qquad |B \setminus A| = p.$$

Тогда $|A| = m + n$, $|B| = n + p$ и

$$\begin{aligned} |A \cup B| &= m + n + p = \\ &= (m + n) + (n + p) - n = \\ &= |A| + |B| + |A \cap B|. \end{aligned}$$

■

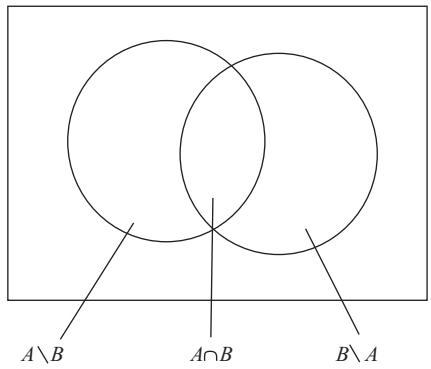


Рисунок 3.7.

Пример 3.7. Каждый из 63 студентов первого курса, изучающих информатику в университете, может посещать и дополнительные лекции. Если 16 из них слушают еще курс бухгалтерии, 37 — курс коммерческой деятельности, и 5 изучают обе эти дисциплины, то сколько студентов вообще не посещают упомянутых дополнительных занятий?

Решение. Введем обозначения.

$A = \{$студенты, слушающие курс бухгалтерии$\}$;

$B = \{$студенты, слушающие курс коммерческой деятельности$\}$.

Тогда

$$|A| = 16, \qquad |B| = 37, \qquad |A \cap B| = 5.$$

Поэтому,

$$|A \cup B| = 16 + 37 - 5 = 48.$$

Следовательно, $63 - 48 = 15$ студентов не посещают дополнительных курсов.

Формула для подсчета мощности объединения двух множеств, как уже говорилось, может быть обобщена на произвольное число множеств. Случай трех множеств приведен в упражнениях к этой главе.

При обсуждении конечных множеств, порядок, в котором перечисляются их элементы, значения не имеет. Однако бывает необходимо работать и с упорядоченными наборами. Чтобы хорошо освоить этот новый тип данных, мы сначала познакомимся с упорядоченной парой.

Упорядоченной парой называется запись вида (a, b), где a — элемент некоторого множества A, а b — элемент множества B. Множество всех таких упорядоченных пар называется *декартовым* или *прямым произведением* множеств A и B и обозначается $A \times B$.

Следовательно, $A \times B = \{(a, b) : \ a \in A$ **и** $b \in B\}$. Операция прямого произведения множеств имеет практическое значение, поскольку вплотную подводит нас к понятиям «*отношение*» и «*функция*», играющим заметную роль в информатике и составляющим предмет изучения следующих глав.

Пример 3.8. Пусть $A = \{x, y\}$ и $B = \{1, 2, 3\}$. Найдите декартовы произведения: $A \times B$, $B \times A$ и $B \times B$.

Решение. Прямым произведением $A \times B$ является множество

$$\{(x, 1), (x, 2), (x, 3), (y, 1), (y, 2), (y, 3)\}.$$

Прямое произведение $B \times A$ — это

$$\{(1, x),\ (2, x),\ (3, x),\ (1, y),\ (2, y),\ (3, y)\}.$$

Заметим, что множества $A \times B$ и $B \times A$ различны! Прямым произведением $B \times B$ служит множество

$$\{(1, 1),\ (1, 2),\ (1, 3),\ (2, 1),\ (2, 2),\ (2, 3),\ (3, 1),\ (3, 2),\ (3, 3)\}.$$

Основываясь на примере 3.8, можно предположить, что мощность прямого произведения конечных множеств A и B равна[1]

$$|A \times B| = mn, \quad \text{если } |A| = m \text{ и } |B| = n.$$

Если же одно из них или сразу оба бесконечны, то и произведение будет иметь бесконечное число упорядоченных пар.

Как и в случае предыдущих операций на множествах мы можем нарисовать диаграмму Венна, иллюстрирующую прямое произведение. Например, диаграмма Венна множества $A \times B$ из примера 3.8 представлена на рис. 3.8

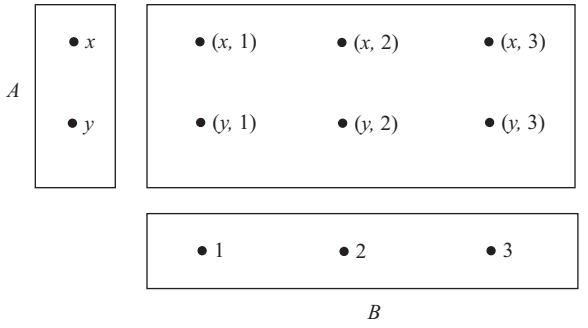


Рисунок 3.8.

В качестве следующего примера рассмотрим прямое произведение множества $\mathbb{R}$ вещественных чисел на само себя. Множество $\mathbb{R}\times\mathbb{R}$ или $\mathbb{R}^2$, как его часто обозначают, состоит из всех упорядоченных пар вещественных чисел (x, y). Их можно представлять себе как координаты точек на плоскости. Множество $\mathbb{R}^2$ называется *декартовой плоскостью*. Она изображена на рис. 3.9

[1] Заметим, что это действительно так, поскольку каждый элемент множества A (а их ровно m штук) участвует ровно в n различных упорядоченных парах. — *Прим. перев.*

Декартовым произведением произвольного числа множеств A_1, A_2, ..., A_n называется множество

$$A_1 \times A_2 \times \cdots A_n = \{(a_1, a_2, \ldots, a_n) : \ a_i \in A_i, \ i = 1, 2, \ldots, n\}.$$

Элементы этого множества — просто *конечные упорядоченные наборы*, объекты, с которыми работают все языки программирования. Подмножества прямых произведений также представляют собой объект обработки в базах данных. Все эти приложения будут рассмотрены в следующих главах.

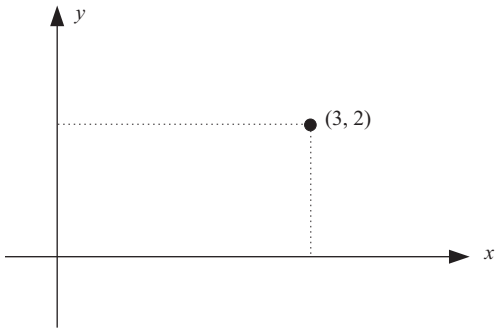


Рисунок 3.9. Декартова плоскость

В том случае, когда каждое из множеств A_1, A_2, ..., A_n совпадает с множеством A, мы пишем A^n для обозначения прямого произведения n экземпляров A.

Пример 3.9. Пусть $B = \{0, 1\}$. Опишите множество B^n.

Решение. Множество B состоит из последовательностей нулей и единиц длины n. Они называются *строкой бит* или *битовой строкой* длины n.

В заключение нашего обсуждения множеств мы покажем, как строка бит применяется для моделирования операций на конечных множествах. Пусть $S = \{s_1, s_2, \ldots, s_n\}$, причем элементы множества мы пометили числовыми индексами исключительно для удобства ссылок. Если $A \subset S$, мы поставим ему в соответствие n-битную строку $(b_1, b_2, \ldots, b_n)$, где $b_i = 1$, если $s_i \in A$ и $b_i = 0$ в противном случае. Такая строка бит называется *характеристическим вектором* подмножества A. Теперь мы можем имитировать операции на множествах логическими операциями, применяемыми к соответствующим характеристическим векторам, условившись считать 1 за И, а 0 за Л.

Пример 3.10. Пусть $S = \{1, 2, 3, 4, 5\}$, $A = \{1, 3, 5\}$ и $B = \{3, 4\}$. Выписать характеристические векторы A и B, а затем определить характеристические векторы множеств $A \cup B$, $A \cap B$ и $\overline{B}$.

Решение. Нетрудно заметить, что характеристическим вектором множества A является $a = (1, 0, 1, 0, 1)$, а характеристический вектор множества B равен $b = (0, 0, 1, 1, 0)$. Значит,

a **или** $b = (1, 0, 1, 0, 1)$ **или** $(0, 0, 1, 1, 0) = (1, 0, 1, 1, 1)$;
a **и** $b = (1, 0, 1, 0, 1)$ **и** $(0, 0, 1, 1, 0) = (0, 0, 1, 0, 0)$;
не $b =$ **не** $(0, 0, 1, 1, 0) = (1, 1, 0, 0, 1)$.

Полученные векторы позволяют нам «без запинки прочитать» элементы требуемых подмножеств: $A \cup B = \{1, 3, 4, 5\}$, $A \cap B = \{3\}$ и $\overline{B} = \{1, 2, 5\}$.

Набор упражнений 3

3.1. (а) Перечислите элементы следующих множеств:

$$A = \{x : \ x \in \mathbb{Z} \text{ и } 10 \leqslant x \leqslant 17\};$$
$$B = \{x : \ x \in \mathbb{Z} \text{ и } x^2 < 24\};$$
$$C = \{x : \ x \in \mathbb{Z} \text{ и } 6x^2 + x - 1 = 0\};$$
$$D = \{x : \ x \in \mathbb{R} \text{ и } 6x^2 + x - 1 = 0\}.$$

Указание: $6x^2 + x - 1 = (3x - 1)(2x + 1)$.

(б) Определите с помощью предикатов следующие множества:

$$S = \{2, 5, 8, 11, \ldots\};$$
$$T = \{1, \tfrac{1}{3}, \tfrac{1}{7}, \tfrac{1}{15}, \ldots\}.$$

3.2. В качестве универсального множества данной задачи зафиксируем $U = \{p, q, r, s, t, u, v, w\}$. Пусть $A = \{p, q, r, s\}$, $B = \{r, t, v\}$ и $C = \{p, s, t, u\}$. Найдите элементы следующих множеств:

(а) $B \cap C$; (б) $A \cup C$; (в) $\overline{C}$;
(г) $A \cap B \cap C$; (е) $\overline{(A \cup B)}$; (з) $B \triangle C$.
(д) $(A \cup B) \cap (A \cap C)$; (ж) $B \setminus C$;

3.3. Рассмотрим подмножества стандартного словаря русского языка.

$A = \{x :\ x$ — слово, стоящее перед «собака»$\}$;

$B = \{x :\ x$ — слово, стоящее после «кошка»$\}$;

$C = \{x :\ x$ — слово, содержащее двойную букву$\}$.

Выясните, какие из следующих высказываний истинны:

(а) $C \subset A \cup B$;

(б) «бассейн» $\in \overline{B} \cap C$;

(в) «стресс» $\in B \triangle C$;

(г) $A \cap B = \varnothing$.

Опишите на словах элементы следующих множеств:

(д) $A \cap B \cap C$;

(е) $(A \cup B) \cap \overline{C}$.

3.4. Рассмотрим подмножества целых чисел:

$A = \{3n :\ n \in \mathbb{Z}$ и $n \geqslant 4\}$;

$B = \{2n :\ n \in \mathbb{Z}\}$;

$C = \{n :\ n \in \mathbb{Z}$ и $n^2 \leqslant 100\}$.

(а) Используя операции на множествах, выразите следующие подмножества через A, B и C:

(i) множество всех нечетных целых чисел;

(ii) $\{-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10\}$;

(iii) $\{6n :\ n \in \mathbb{Z}$ и $n \geqslant 2\}$;

(iv) $\{-9, -7, -5, -3, -1, 1, 3, 5, 7, 9\}$.

(б) Запишите определение множества $A \setminus B$ в предикатах.

3.5. Нарисуйте серию диаграмм Венна, иллюстрирующих закон дистрибутивности:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

Докажите, что закон действительно справедлив для любых множеств A, B и C.

3.6. Нарисуйте серию диаграмм Венна, иллюстрирующих следующее тождество:

$$A \cap (B \triangle C) = (A \cap B) \triangle (A \cap C).$$

Покажите на примере, что множество $A \cup (B \triangle C)$ не обязательно совпадает с множеством $(A \cup B) \triangle (A \cup C)$.

3.7. Докажите с помощью законов алгебры множеств следующие тождества:

(а) $\overline{(A \cap \overline{B})} \cup B = \overline{A} \cup B$;

(б) $\overline{(\overline{A} \cap \overline{(B \cup C)})} = A \cup B \cup C$;

(в) $(A \cup B \cup C) \cap (A \cup \overline{B} \cup C) \cap \overline{(A \cup C)} = \varnothing$;

(г) $(A \setminus B) \setminus C = A \setminus (B \cup C)$;

(д) $A \triangle A \triangle A = A$.

3.8. Определим операцию «$*$» по формуле:

$$A * B = \overline{(A \cap B)}.$$

Изобразите на диаграмме Венна множество $A * B$. С помощью законов алгебры множеств докажите тождества:

(а) $A * A = \overline{A}$;

(б) $(A * A) * (B * B) = A \cup B$;

(в) $(A \cup B \cup C) \cap (A \cup \overline{B} \cup C) \cap \overline{(A \cup C)} = \varnothing$;

(г) $(A * B) * (A * B) = A \cap B$.

3.9. (а) Покажите с помощью диаграмм Венна, что любые множества A, B и C удовлетворяют соотношению:

$$\begin{aligned} &|A \cup B \cup C| = \\ =&|A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|. \end{aligned}$$

(б) Студенты первого курса, изучающие информатику в университете, могут посещать и дополнительные дисциплины. В этом году 25 из них предпочли изучать бухгалтерию, 27 выбрали бизнес, а 12 решили заниматься туризмом. Кроме того, было 20 студентов, слушающих курс бухгалтерии и бизнеса, пятеро изучали бухгалтерию и туризм, а трое — туризм и бизнес. Известно, что никто

из студентов не отважился посещать сразу три дополнительных курса. Сколько студентов посещали по крайней мере один дополнительный курс? Сколько из них были увлечены только туризмом?

3.10. Что можно сказать о непустых множествах A и B, если имеет место равенство $A \times B = B \times A$?
Непустые множества A, B и C удовлетворяют соотношению $A \times B = A \times C$. Следует ли отсюда, что $B = C$? Объясните свой ответ.

3.11. Пусть A, B и C — произвольные множества. Докажите, что

(а) $A \times (B \cap C) = (A \times B) \cap (A \times C)$;

(б) $(A \cup B) \times C = (A \times C) \cup (B \times C)$.

3.12. *Показательным множеством* $\mathcal{P}(A)$ называется множество, элементами которого являются подмножества множества A. Иначе говоря, $\mathcal{P}(A) = \{C : \ C \subset A\}$.

(а) Найдите $\mathcal{P}(A)$, если $A = \{1, 2, 3\}$.

(б) Докажите, что $\mathcal{P}(A) \cap \mathcal{P}(B) = \mathcal{P}(A \cap B)$ для любых множеств A и B.

(в) Покажите на примере, что $\mathcal{P}(A) \cup \mathcal{P}(B)$ не всегда совпадает с $\mathcal{P}(A \cup B)$.

3.13. Пусть $U = \{1, 2, 3, 4, 5, 6\}$ — универсальное множество. Выпишите характеристические векторы подмножеств:

$$A = \{1, 2, 4, 5\} \qquad \text{и} \qquad B = \{3, 5\}.$$

Найдите характеристические векторы подмножеств $A \cup \overline{B}$ и $A \triangle B$, после чего перечислите их элементы.

Краткое содержание главы

Множество — это совокупность объектов, называемых его элементами.

Символом $\varnothing$ обозначается **пустое** множество, а U — универсальное.

$\mathbb{N} = \{1, 2, 3, \ldots\}$ — множество **натуральных** чисел.

$\mathbb{Z} = \{0, \ \pm 1, \ \pm 2, \ \pm 3, \ \ldots\}$ — множество **целых** чисел.

$\mathbb{Q} = \{\frac{p}{q} : \ p, \ q$ целые, $q \neq 0\}$ — множество **рациональных** чисел.

$\mathbb{R} = \{$все десятичные дроби$\}$ — множество **вещественных** чисел.

Подмножеством множества S называется множество A, все элементы которого принадлежат S. Этот факт обозначается так: $A \subset S$.

Два множества называются **равными** тогда и только тогда, когда каждое из них является подмножеством другого.

Объединением множеств A и B называется множество

$$A \cup B = \{x : \ x \in A \textbf{ или } x \in B\}.$$

Пересечением множеств A и B называется множество

$$A \cap B = \{x : \ x \in A \textbf{ и } x \in B\}.$$

Дополнением множества B до A называется множество

$$A \setminus B = \{x : \ x \in A \textbf{ и } x \notin B\}.$$

Дополнением множества A (до универсального множества U) называется множество

$$\overline{A} = \{x : \ x \notin A\}.$$

Симметрической разностью двух множеств A и B называется множество

$$A \triangle B = \{x : \ (x \in A \textbf{ и } x \notin B) \textbf{ или } (x \in B \textbf{ и } x \notin A)\}.$$

Из любого тождества множеств можно получить **двойственное**, если заменить $\cap$ на $\cup$, $\varnothing$ на U и наоборот.

Мощностью конечного множества S называется число его элементов. Оно обозначается через $|S|$.

Формула включений и исключений утверждает, что

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Декартовым произведением множеств A и B является множество

$$A \times B = \{(a, b) : \ a \in A \textbf{ и } b \in B\}.$$

Элементы $A \times B$ называются **упорядоченными парами.**

Множество $\mathbb{R} \times \mathbb{R}$ или $\mathbb{R}^2$ называется **декартовой плоскостью**.

Битовой строкой (длины n) называется элемент множества B^n, где $B = \{0, 1\}$.

Приложение. Система с базой знаний

Экспертная система создается с целью подменить собой специалистов в данной области. Это достигается накоплением *базы знаний* известных фактов вместе с определением набора *правил вывода*. Вследствие чего ответы на запросы системы могут быть выведены логическим путем из базы знаний.

Мы построим простую экспертную систему «КОРОЛЕВСКАЯ ДИНАСТИЯ» для ответа на вопросы об английских королях и королевах и их семьях, начиная с Георга I. Прежде всего мы подготовим список фактов, используя предикаты *родитель* и *жена*.

родитель(Георг I, Георг II)
родитель(Георг III, Георг IV)
родитель(Георг III, Вильгельм IV)
родитель(Георг III, Эдвард)
родитель(Эдвард, Виктория)
родитель(Виктория, Эдвард VII)
родитель(Эдвард VII, Георг V)
родитель(Георг V, Эдвард VIII)
родитель(Георг V, Георг VI)
родитель(Георг VI, Елизавета II)
родитель(Виктория, Элис)
родитель(Элис, Виктория Альберта)
родитель(Виктория Альберта, Элис–Моунтбаттен)
родитель(Элис–Моунтбаттен, Филипп)

жена(София, Георг I)
жена(Вильгельмина, Георг II)
жена(Шарлотта, Георг III)
жена(Каролина, Георг IV)
жена(Аделаида, Вильгельм IV)
жена(Виктория, Альберт)
жена(Александра, Эдвард VII)
жена(Виктория Мари, Георг V)
жена(Елизавета, Георг VI)
жена(Елизавета II, Филлип)

Условимся, что *родитель*(x, y) означает, что x является родителем y, а *жена*(x, y) означает, что x — жена y. Это стандартное чтение предикатов, используемых языками программирования, такими, как, например, PROLOG.

Чтобы извлечь информацию, мы будем ставить вопросы перед базой данных. Например, если мы спрашиваем: «является ли Георг I отцом Георга III?», то ответ будет отрицательным, поскольку предикат *родитель*(Георг I, Георг 3) отсутствует в нашем списке фактов.

Запросы записываются в виде: «? — предикат». Кроме того предполагается, что наличие переменной в предикате равносильно вопросу о существовании. Например, запрос «? — *жена*$(x$, Георг IV) понимается как «была ли жена у Георга IV?». В этом случае ответ положителен, так как, заменяя x на «Каролина», мы получим высказывание, присутствующее в списке фактов.

Задача 1. Найдите ответы на следующие запросы:

(а) ? — *жена*(Елизавета II, Филипп);

(б) ? — *родитель*(София, Георг II);

(в) ? — *женщина*(Каролина);

(г) ? — *жена*(Филипп, Елизавета II).

Решение. Положительный ответ будет выдан только на первый запрос, так как только для него в списке фактов есть соответствующий предикат. Напомним, что отрицательный ответ на запрос дается в том случае, если список фактов не содержит предиката из запроса.

Чтобы система с базой знаний могла решать более сложные задачи, мы введем так называемые правила вывода. Правило вывода определяет новый предикат в терминах тех, которые присутствуют в исходном списке фактов. Ответы на запросы о новых предикатах могут быть логически выведены из списка фактов, генерируя, таким образом, новую информацию.

В системе «КОРОЛЕВСКАЯ ДИНАСТИЯ» кажется очевидным, что переменная x, попавшая в *жена*(x, y), соответствует женщине. В правиле (1) определим новый предикат, который будет означать, что «если x — жена y, то x — женщина».

(1) *женщина*(x) **from** *жена*(x, y).

Аналогично, введем правило (2), определяющее предикат *муж*. Он означает, что «если x — жена y, то y — муж x».

(2) *муж*(y, x) **from** *жена*(x, y).

Задача 2. Как изменятся ответы на запросы из задачи 1? Ответьте на следующие дополнительные запросы:

(д) ? — *женщина*(Элис–Моунтбаттен);

(е) ? — *муж*(Альберт, Виктория);

(ж) ? — *мужчина*(Альберт).

Решение. Теперь на запрос (в) из задачи 1 будет дан положительный ответ, согласно правилу (1), примененному к основному факту: *жена*(Каролина, Георг IV).

На запрос (д) ответ будет отрицателен, так как Элис–Моунтбаттен в основном списке не упомянута в качестве чьей-либо жены.

Ответ в случае (е) — положителен, ввиду наличия в списке предиката *жена*(Виктория, Альберт) и правила (2).

Отрицательный ответ будет дан на запрос (ж), так как предикат *мужчина* пока еще не определен.

Подходящее правило вывода, дающее информацию о принадлежности к мужской половине человечества, аналогично правилу (1):

(3) *мужчина*(y) **from** *жена*(x, y)

Можно сформулировать правило, представляющее информацию о сыновьях:

(3) *сын*(x, y) **from** (*мужчина*(x) **и** *родитель*(y, x))

Задача 3. Ответьте на следующие запросы:

(а) ? — *мужчина*(Вильгельм IV);

(б) ? — *сын*(Вильгельм IV, Георг III);

(в) ? — *сын*(Вильгельм IV, Шарлотта);

(г) ? — *сын*(Эдвард VIII, Георг V).

Решение.

(а) Положительный ответ следует из предиката *жена*(Аделаида, Вильгельм IV) по правилу (3).

(б) Положительный ответ следует из предиката *родитель*(Георг III, Вильгельм IV) ввиду положительного ответа на запрос (а) и правила вывода (4).

Ответы на последние два запроса — отрицательны.

Обратите внимание, что при ответах на (в) и (г) необходимо твердо придерживаться фактов и правил вывода, ввиду ограничений, наложенных на систему. Только монархи считаются родителями, в то время как их супруги появляются только в предикате *жена*. Так, хотя Шарлотта была замужем за Георгом III, и Вильгельм IV — один из их сыновей, база данных считает его родителем только Георга III. Следовательно, правило вывода (4) не может дать положительный ответ на запрос (в). Причина отрицательного ответа в случае (г) заключается в том, что в списке отсутствует жена у Эдварда VIII. Поэтому правило (3) дает ответ «Нет» на запрос «? — *мужчина*(Эдвард VIII)».

Как мы увидели, в случае неполной информации, содержащейся в системе данных, как это часто бывает в реальных экспертных системах, то отрицательный ответ на запрос может означать, что нам просто ничего не известно. Должное внимание к формулировке правил вывода и выбору исходных предикатов базы данных может частично решить эту проблему. К сожалению, при неопределенности отрицательных ответов мы не можем полностью доверять и положительным, если в предикатах участвует операция **не**.

Рассмотрим, например, следующие, разумные на первый взгляд правила вывода (A) и (B):

(A) *мужчина*(x) **from** *жена*(x, y);

(B) *женщина*(x) **from** (**не** *мужчина*(x)).

Попробуем ответить на запрос: «? — *женщина*(Эдвард VIII)», основываясь на исходном списке фактов, но пользуясь только правилами (A) и (B). На запрос «? — *женщина*(Эдвард VIII)» будет получен отрицательный ответ. Поэтому высказывание «**не** *женщина*(Эдвард VIII)» становится выведенным истинным фактом. По правилу (B) на запрос «? — *женщина*(Эдвард VIII)» будет дан положительный ответ! Следовательно, прежде чем разрешать употребление отрицаний в правилах вывода, необходимо убедиться в полноте исходной информации.

Задача 4. Сформулируйте правило вывода для извлечения информации о матерях из экспертной системы «КОРОЛЕВСКАЯ ДИНАСТИЯ». Определите правило *мать*(x) так, чтобы положительный ответ на соответствующий запрос выдавался в том случае, если x — жена чьего-то родителя или x — женщина и чей-то родитель. Примените новое правило совместно с правилом (1) к исходной базе данных для определения максимально возможного числа матерей. Удовлетворительным ли получилось новое правило вывода?

Решение. Требуемое правило вывода может быть определено так:

мать(x) **from** ([*жена*(x, y) **и** *родитель*(z, y)] **или**
или [*женщина*(x) **и** *родитель*(x, y)]).

Часть [*жена*(x, y) **и** *родитель*(z, y)] нашего правила определит как «мать» следующих королев: Александра, Шарлотта, Елизавета, София и Виктория Мари. Вторая часть [*женщина*(x) **и** *родитель*(x, y)] выявит Викторию.

Однако сформулированное правило вывода найдет не всех матерей, поскольку база данных неполна. В ней, например, не записаны дети Едизаветы II.

Первая часть правила будет считать матерями и мачех. Кроме того, проблема будет возникать и тогда, когда у монарха было несколько жен. Это показывает трудности, возникающие при попытке ограничить реальный мир рамками простой математической модели.

ГЛАВА 4

ОТНОШЕНИЯ

Когда говорят о родстве двух человек, Хораса и Анны, то подразумевают, что есть некая семья, к членам которой они относятся. Упорядоченная пара (Хорас, Анна) отличается от других упорядоченных пар людей тем, что между Хорасом и Анной есть некое родство (кузина, отец, и т. д.). В математике среди всех упорядоченных пар прямого произведения $A \times B$ двух множеств A и B тоже выделяются некоторые пары в связи с тем, что между их компонентами есть некоторые «родственные» отношения, которых нет у других.

В качестве примера рассмотрим множество S студентов какого-нибудь института и множество K читаемых там курсов. В прямом произведении $S \times K$ можно выделить большое подмножество упорядоченных пар (s, k), обладающих свойством: студент s слушает курс k. Построенное подмножество отражает отношение «... слушает ...», естественно возникающее между множествами студентов и курсов.

Для строгого математического описания любых связей между элементами двух множеств мы введем понятие бинарного отношения. В этой главе мы расскажем о различных путях определения отношений и обсудим некоторые их свойства. В частности, мы изучим два важных специальных типа отношений: отношение эквивалентности и частичного порядка. Они часто появляются как в математике, так и в информатике. Отношения между элементами нескольких множеств задаются в виде таблиц данных. В приложении к этой главе такие n-арные отношения применяются для описания простой *системы управления базами данных*.

4.1. Бинарные отношения

Бинарным отношением между множествами A и B называется подмножество R прямого произведения $A \times B$. В том случае, когда $A = B$, мы говорим просто об *отношении* R на A.

Пример 4.1. Рассмотрим генеалогическое древо, изображенное на рис. 4.1. Выпишите упорядоченные пары, находящиеся в следующих отношениях на множестве P членов этой семьи:

(а) $R = \{(x, y) : \ x$ — дедушка $y\}$;

(б) $S = \{(x, y) : \ x$ — сестра $y\}$.

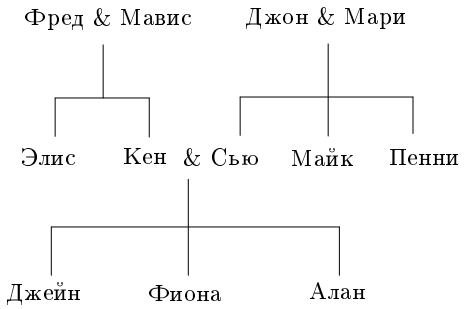


Рисунок 4.1.

Решение.

(а) R содержит упорядоченные пары: (Фред, Джейн), (Фред, Фиона), (Фред, Алан), (Джон, Джейн), (Джон, Фиона) и (Джон, Алан).

(б) S состоит из пар: (Сью, Пенни), (Пенни, Сью), (Джейн, Фиона), (Фиона, Джейн), (Алис, Кен), (Сью, Майк), (Пенни, Майк), (Джейн, Алан) и (Фиона, Алан).

Пример 4.2. Выпишите упорядоченные пары, принадлежащие следующим бинарным отношениям на множествах $A = \{1, 3, 5, 7\}$ и $B = \{2, 4, 6\}$:

(а) $U = \{(x, y) : \ x + y = 9\}$;

(б) $V = \{(x, y) : \ x < y\}$.

Решение.

(а) U состоит из пар: (3, 6), (5, 4) и (7, 2);

(б) $V = \{(1, 2), \ (1, 4), \ (1, 6), \ (3, 4), \ (3, 6), \ (5, 6)\}$.

Пример 4.3. Множество

$$R = \{(x, y) : \ x \text{ — делитель } y\}$$

определяет отношение на множестве $A = \{1, 2, 3, 4, 5, 6\}$. Найдите все упорядоченные пары, ему принадлежащие.

Решение. R состоит из пар: (1, 1), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 2), (2, 4), (2, 6), (3, 3), (3, 6), (4, 4), (5, 5) и (6, 6).

Теперь мы познакомимся с двумя более удобными способами перечисления упорядоченных пар, принадлежащих данному отношению. Первый из них основан на понятии «ориентированный граф», а второй опирается на матрицы.

Пусть A и B — два конечных множества и R — бинарное отношение между ними. Мы изобразим элементы этих множеств точками на плоскости. Для каждой упорядоченной пары отношения R нарисуем стрелку, соединяющую точки, представляющие компоненты пары. Такой объект называется *ориентированным графом* или *орграфом*, точки же, изображающие элементы множеств, принято называть вершинами графа.

В качестве примера рассмотрим отношение V между множествами $A = \{1, 3, 5, 7\}$ и $B = \{2, 4, 6\}$ из примера 4.2 (б). Соответствующий ориентированный граф показан на рис. 4.2.

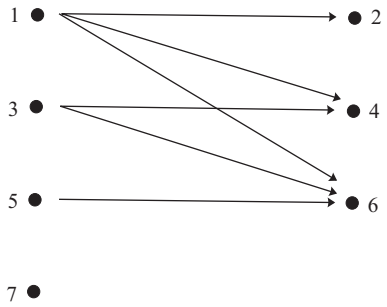


Рисунок 4.2. Отношение V между A и B

Для иллюстрации отношения на отдельном множестве A мы чертим орграф, чьи вершины соответствуют одному лишь множеству A, а стрелки, как обычно, соединяют элементы упорядоченных пар, находящихся в отношении.

Пример 4.4. Изобразите граф, представляющий отношение R из примера 4.3.

Решение. Поскольку R — отношение на множестве

$$A = \{1, 2, 3, 4, 5, 6\},$$

то ориентированный граф будет иметь шесть вершин. Он приведен на рис. 4.3.

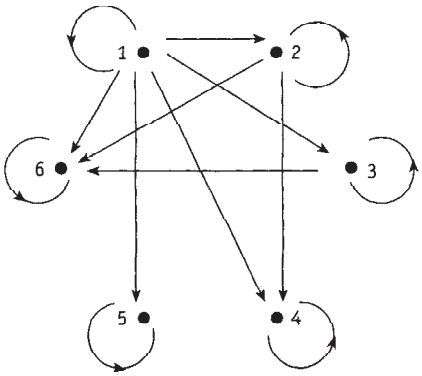


Рисунок 4.3. Отношение R на множестве A

Второй способ задания бинарного отношения на конечных множествах основан на использовании таблиц. Предположим, что мы хотим определить бинарное отношение R между множествами A и B. Необходимо обозначить элементы множеств и выписать их в каком-нибудь порядке. Сделаем это так:

$$A = \{a_1, a_2, \ldots, a_n\}, \qquad B = \{b_1, b_2, \ldots, b_m\}.$$

Для определения отношения R заполним таблицу M с n строками и m столбцами. Строки «перенумеруем» элементами множества A, а столбцы — элементами множества B в соответствии с порядком, в котором мы выписали элементы. Ячейку таблицы, стоящую на пересечении i-той строки и j-того столбца будем обозначать через $M(i, j)$, а заполнять ее будем следующим образом:

$$M(i, j) = \text{И}, \quad \text{если } (a_i, b_j) \in R,$$
$$M(i, j) = \text{Л}, \quad \text{если } (a_i, b_j) \notin R,$$

Такого сорта таблицы называются $n \times m$ *матрицами*.

В этих терминах, отношение U из примера 4.2(а) с помощью матрицы задается следующим образом:

$$\begin{array}{c} \\ 1 \\ 3 \\ 5 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 4 & 6 \end{array} \\ \left[\begin{array}{ccc} Л & Л & Л \\ Л & Л & И \\ Л & И & Л \\ И & Л & Л \end{array}\right] \end{array}.$$

Чтобы лучше понять такой способ задания отношений, мы явно пометили столбцы и строки матрицы. В общем случае это делать не обязательно.

Пример 4.5. Отношение R на множестве $A = \{a, b, c, d\}$ задается матрицей:

$$\begin{bmatrix} Л & И & И & Л \\ Л & Л & И & И \\ Л & И & Л & Л \\ И & И & Л & И \end{bmatrix},$$

порядок строк и столбцов в которой соответствует порядку выписанных элементов множества A. Назовите упорядоченные пары, принадлежащие R.

Решение. Отношение R содержит упорядоченные пары: (a, b), (a, c), (b, c), (b, d), (c, b), (d, a), (d, b) и (d, d).

Пример 4.6. Выпишите матрицу, представляющую отношение R из примера 4.3.

Решение. Матрица отношения R имеет вид:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} И & И & И & И & И & И \\ Л & И & Л & И & Л & И \\ Л & Л & И & Л & Л & И \\ Л & Л & Л & И & Л & Л \\ Л & Л & Л & Л & И & Л \\ Л & Л & Л & Л & Л & И \end{bmatrix} \end{array}.$$

Если R — бинарное отношение, то вместо записи $(x, y) \in R$ можно употреблять обозначение $x\,R\,y$. Например, предикат «x — сестра y» определяет отношение на множестве всех людей. В примере 4.3 предикат «x — делитель y» дает ясное словесное описание еще одного отношения.

Подводя итог вводной части теории отношений, полезно напомнить, что бинарное отношение между конечными множествами может быть задано одним из следующих способов:

- словами (с помощью подходящих предикатов);
- как множество упорядоченных пар;
- как орграф;
- как матрица.

Пример 4.7. Отношение R на множестве $A = \{1, 2, 3, 4\}$ представлено графом на рис. 4.7.
Перечислите упорядоченные пары, принадлежащие R, выпишите соответствующую матрицу и определите это отношение с помощью предикатов.

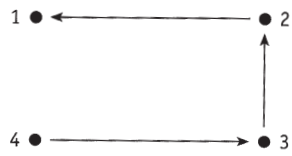


Рисунок 4.3.

Решение. В терминах упорядоченных пар $R = \{(2, 1), (3, 2), (4, 3)\}$.

Матрица (относительно данного в условии порядка элементов множества) имеет вид:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{cccc} Л & Л & Л & Л \\ И & Л & Л & Л \\ Л & И & Л & Л \\ Л & Л & И & Л \end{array}\right] \end{array}.$$

С помощью предикатов данное отношение может быть описано как

$$x - y = 1.$$

4.2. Свойства отношений

Ограничимся рассмотрением бинарных отношений, заданных на одном множестве и введем некоторый набор их свойств.

Говорят, что отношение R на множестве A

рефлексивно, если для всех $x \in A \quad x\,R\,x$;

симметрично, если $x\,R\,y \Rightarrow y\,R\,x$ для каждой пары x и y из A;

кососимметрично, если ($x\,R\,y$ и $y\,R\,x \Rightarrow x = y$) для всех x и y из A;

транзитивно, если ($x\,R\,y$ и $y\,R\,z \Rightarrow x\,R\,z$) для любой тройки элементов $x, y, z \in A$.

В терминах упорядоченных пар эти свойства определяются следующим образом. Данное отношение R рефлексивно, если $(x, x) \in R$ для любого возможного значения переменной x; симметрично, если из включения $(x, y) \in R$ следует, что $(y, x) \in R$; кососимметрично, если из предположений: $(x, y) \in R$ и $x \neq y$ вытекает, что $(y, x) \notin R$; транзитивно, если включения $(x, y) \in R$ и $(y, z) \in R$ влекут $(x, z) \in R$.

У ориентированного графа, изображающего рефлексивное отношение, каждая вершина снабжена петлей, т. е. стрелкой, начинающейся и заканчивающейся в одной и той же вершине. Орграф симметричного отношения вместе с каждой стрелкой из вершины x в вершину y имеет стрелку, направленную в обратную сторону: из y в x. Если отношение кососимметрично, то при наличии стрелки из вершины x в несовпадающую с ней вершину y, стрелка из y в x будет обязательно отсутствовать. И, наконец, орграф транзитивного отношения устроен так, что вместе со стрелками из вершины x в y и из y в z у него будет стрелка и из x в z.

Перечислим свойства матриц, задающих отношения. Прежде всего заметим, что матрица отношения на отдельном множестве A будет квадратной, т. е. количество ее строк будет равно количеству столбцов. Так вот, матрица M, задающая рефлексивное отношение, отличается от других тем, что каждый ее элемент, стоящий на главной диагонали ($M(i,\, i)$), равен И; матрица M симметричного отношения будет симметричной, т. е. $M(i,\, j) = M(j,\, i)$; в матрице кососимметричного отношения выполнено условие:

$$\big(M(i,\, j) = \text{И}\ \textbf{и}\ i \neq j\big) \Rightarrow M(j,\, i) = \text{Л}.$$

К сожалению, отличительное свойство матрицы транзитивного отношения довольно трудно сформулировать четко и наглядно.

Пример 4.8. Что можно сказать о свойствах (рефлексивности, симметричности, кососимметричности и транзитивности) следующих отношений:

(а) «x делит y» на множестве натуральных чисел;

(б) «$x \neq y$» на множестве целых чисел;

(в) «количество лет x совпадает с возрастом y» на множестве всех людей.

Решение.

(а) Поскольку x всегда делит сам себя, то это отношение рефлексивно. Оно, конечно, не симметрично, поскольку, например, 2 является делителем 6, но не наоборот: 6 не делит 2. Проверим, что отношение делимости транзитивно. Предположим, что x делит y, а y в свою очередь делит z. Тогда из первого предположения вытекает, что $y = mx$ для некоторого натурального

числа m, а из второго — $z = ny$, где n — натуральное число. Следовательно, $z = ny = (nm)x$, т. е. x делит z. Значит, данное отношение транзитивно. Наконец, наше отношение кососимметрично, поскольку из предположений: x делит y и y делит x немедленно вытекает, что $y = x$.

(б) Так как высказывание «$x \neq x$» ложно, то это отношение не рефлексивно. Оно симметрично, поскольку $x \neq y$ тогда и только тогда, когда $y \neq x$. Наше отношение не обладает свойством транзитивности, так как, например, $2 \neq 3$ и $3 \neq 2$, но, тем не менее, $2 = 2$. Наше отношение не кососимметрично, поскольку из условий $x \neq y$ и $y \neq x$ нельзя заключить, что $x = y$.

(в) Отношение этого пункта рефлексивно, так как возраст любого человека x совпадает с количеством прожитых им лет. Оно симметрично, поскольку высказывание «количество лет x совпадает с возрастом y» равносильно высказыванию «количество лет y совпадает с возрастом x». Отношение и транзитивно, так как, если найдутся такие три человека x, y и z, что «количество лет x совпадает с возрастом y», а «количество лет y совпадает с возрастом z», то все трое будут одинакового возраста. Так как мы можем найти много ровесников, то данное отношение не кососимметрично.

Если отношение R на множестве A не обладает тем или иным свойством, то его стоит попытаться продолжить до отношения R^*, которое будет иметь нужное свойство. Под «продолжением» мы понимаем присоединение некоторых упорядоченных пар к подмножеству $R \subset A \times A$ так, что новое полученное множество R^* уже будет обладать требуемым свойством. Ясно, что исходное множество R будет подмножеством в R^*. В том случае, если вновь построенное множество R^* будет минимальным среди всех расширений R с выделенным свойством, то говорят, что R^* является замыканием R относительно данного свойства.

Более строго, R^* называется *замыканием отношения* R относительно свойства P, если

1. R^* обладает свойством P;
2. $R \subset R^*$;
3. R^* является подмножеством любого другого отношения, содержащего R и обладающего свойством P.

Пример 4.9. Пусть $A = \{1, 2, 3\}$, а отношение R на A задано упорядоченными парами:

$$R = \{(1, 1),\ (1, 2),\ (1, 3),\ (3, 1),\ (2, 3)\}.$$

Оно не рефлексивно, не симметрично и не транзитивно. Найдите соответствующие замыкания.

Решение. Замыкание относительно рефлексивности должно содержать все пары вида (x, x). Поэтому, искомое замыкание имеет вид:

$$R^* = \{(1, 1),\ (1, 2),\ (1, 3),\ (3, 1),\ (2, 3);\ \ (2, 2),\ (3, 3)\},$$

где добавленные пары отделены от исходных точкой с запятой.

Замыкание относительно симметричности должно содержать все пары, симметричные исходным. Значит,

$$R^* = \{(1, 1),\ (1, 2),\ (1, 3),\ (3, 1),\ (2, 3);\ \ (2, 1),\ (3, 2)\}.$$

Чтобы найти замыкание относительно транзитивности, необходимо выполнить несколько шагов. Так как R содержит пары (3, 1) и (1, 2), замыкание обязано включать в себя и пару (3, 2). Аналогично, пары (2, 3) и (3, 1) добавляют пару (2, 1), а пары (3, 1) и (1, 3) — пару (3, 3). Добавим сначала эти пары:

$$R^* \supset \{(1, 1),\ (1, 2),\ (1, 3),\ (3, 1),\ (2, 3);\ \ (3, 2),\ (2, 1),\ (3, 3)\}.$$

Теперь у нас возникло сочетание (2, 1) и (1, 2). Стало быть, замыкание R^* должно содержать пару (2, 2). Теперь можно увидеть, что все необходимые пары мы добавили (хотя бы потому, что перебрали все пары из A^2). Следовательно,

$$R^* = \{(1, 1),\ (1, 2),\ (1, 3),\ (3, 1),\ (2, 3);\ \ (3, 2),\ (2, 1),\ (3, 3),\ (2, 2)\}.$$

Метод, которым мы нашли замыкание по транзитивности в примере 4.9, довольно специфичен. В главе 8 мы обсудим более систематический подход, использующий алгоритм, который по матрице отношения вычисляет матрицу замыкания относительно транзитивности.

Замыкание по транзитивности имеет массу приложений. Допустим, нам дан ориентированный граф, отражающий коммуникационную сеть. В этом случае матрица замыкания по транзитивности позволит нам определить, существует ли возможность переправить сообщение из одного места в другое.

4.3. Отношения эквивалентности и частичного порядка

В этом параграфе мы сосредоточимся на двух важных специальных типах бинарных отношений.

Рефлексивное, симметричное и транзитивное бинарное отношение на множестве A называется *отношением эквивалентности*. Отношение эквивалентности в некотором смысле обобщает понятие равенства. *Эквивалентные элементы* (т. е. находящиеся в отношении эквивалентности), как правило, обладают какими-то общими признаками.

Приведем примеры отношения эквивалентности.

- Отношение «... имеет те же углы, что и ...» на множестве всех треугольников. Очевидно, треугольники эквивалентны относительно этого отношения тогда и только тогда, когда они подобны.

- Отношение R, заданное условием: $x\,R\,y$, если и только если $xy > 0$ на множестве ненулевых целых чисел является отношением эквивалентности. При этом эквивалентные числа имеют одинаковый знак.

- Отношение «... имеет тот же возраст, что и ...» на множестве всех людей. «Эквивалентные» люди принадлежат к одной и той же возрастной группе.

Примеры наводят на мысль, что если на множестве задано отношение эквивалентности, то все его элементы можно естественным способом разбить на непересекающиеся подмножества. Все элементы в любом из таких подмножеств эквивалентны друг другу в самом прямом смысле. Наличие такого разбиения — движущая сила любой классификационной системы.

Разбиением множества A называется совокупность непустых подмножеств A_1, A_2, $\ldots A_n$ множества A, удовлетворяющих следующим требованиям:

1) $A = A_1 \cup A_2 \cup \cdots \cup A_n$;

2) $A_i \cap A_j = \varnothing$ при $i \neq j$.

Подмножества A_i называются *блоками* разбиения.

Диаграмма Венна разбиения множества A на пять блоков показана на рис. 4.4. Заметим, что блоки изображены как лоскуты, не

заходящие один на другой. Это связано с тем, что блоки разбиения не могут иметь общих элементов.

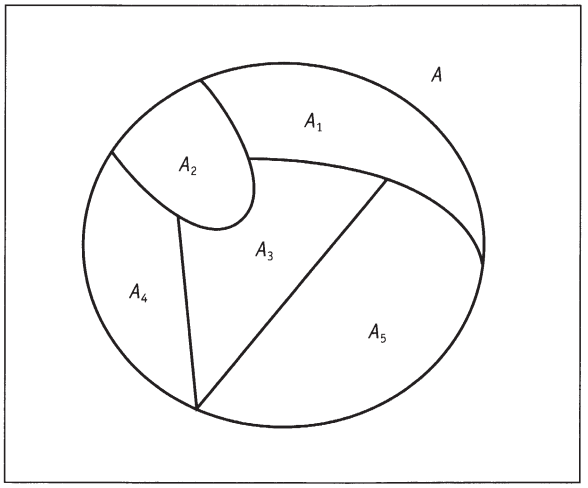


Рисунок 4.4.

Как мы уже говорили, отношение эквивалентности R на множестве A задает на нем разбиение. Блоки разбиения при этом состоят из эквивалентных друг другу элементов. Мы сейчас докажем это утверждение. Но прежде определим *класс эквивалентности* E_x произвольного элемента $x \in A$ как подмножество $E_x = \{z \in A : \ z\,R\,x\}$. Докажем теорему.

Теорема. Пусть R — отношение эквивалентности на непустом множестве A. Тогда различные классы эквивалентности определяют разбиение A.

Доказательство. Доказательство состоит из четырех частей.

Сначала покажем, что классы эквивалентности являются непустыми подмножествами в A. По определению, E_x — подмножество в A. Кроме того, R — рефлексивное отношение, т. е. $x\,R\,x$. Следовательно, $x \in E_x$ и E_x не пусто.

Далее проверим, что из $x\,R\,y$ вытекает равенство $E_x = E_y$. Предположим, что $x\,R\,y$ и возьмем произвольный $z \in E_x$. Тогда $z\,R\,x$ и $x\,R\,y$. Поскольку R — транзитивное отношение, мы получаем, что $z\,R\,y$. Иными словами, $z \in E_y$. Следовательно, $E_x \subset E_y$. Аналогично можно показать, что $E_y \subset E_x$, откуда $E_x = E_y$, что и требовалось.

Теперь мы покажем, что классы эквивалентности удовлетворяют первому свойству разбиения, а именно, что A является объеди-

нением всех классов эквивалентности. Как уже отмечалось в первой части нашего доказательства, E_x — подмножество в A и поэтому объединение всех классов эквивалентности тоже будет подмножеством в A. В обратную сторону, если $x \in A$, то $x \in E_x$. В частности, x принадлежит объединению классов эквивалентности. Значит, и A является подмножеством нашего объединения. Следовательно, A совпадает с объединением классов эквивалентности.

И, наконец, в последней части мы покажем, что два разных класса эквивалентности не пересекаются. Этим мы проверим, что классы удовлетворяют второму свойству разбиения. Воспользуемся методом «от противного». Допустим, что $E_x \cap E_y \neq \varnothing$. Тогда найдется элемент z в A, принадлежащий пересечению $E_x \cap E_y$. Следовательно, $z\,R\,x$ и $z\,R\,y$. Так как R — симметричное отношение, можно утверждать, что $x\,R\,z$ и $z\,R\,y$. Ввиду транзитивности R, это влечет $x\,R\,y$. Значит, по второй части доказательства, $E_x = E_y$. Итак, мы предположили, что *разные* классы эквивалентности E_x и E_y пересекаются и доказали, что они на самом деле совпадают. Полученное противоречие доказывает последнюю часть наших рассуждений. Теорема доказана. ■

Заметим, чтобы показать, что классы эквивалентности служат блоками разбиения множества A, мы использовали *все* определяющие свойства отношения эквивалентности: рефлексивность, симметричность и транзитивность.

Пример 4.10. Отношение R на вещественной прямой $\mathbb{R}$ задано условием: $x\,R\,y$, если и только если $x - y$ — целое число. Докажите, что R — отношение эквивалентности и опишите классы эквивалентности, содержащие 0, $\frac{1}{2}$ и $\sqrt{2}$.

Решение. Так как $x - x = 0 \in \mathbb{Z}$ для любого вещественного числа x, отношение R рефлексивно. Если $x - y$ число целое, то и противоположное к нему $y - x = -(x - y)$ является целым. Следовательно, R — симметричное отношение. Пусть $x - y$ и $y - z$ — целые числа. Тогда $x - z = (x - y) + (y - z)$ — сумма целых чисел, т. е. целое число. Это означает, что R транзитивно.

Итак, мы показали, что R рефлексивно, симметрично и транзитивно. Следовательно, R — отношение эквивалентности.

Класс эквивалентности E_x произвольного вещественного числа x определяется по формуле:

$$E_x = \{z \in \mathbb{R} :\ z - x \text{ — целое число}\}.$$

Поэтому,

$$E_0 = \mathbb{Z};$$

$$E_{\frac{1}{2}} = \{z \in \mathbb{R} : \ z - \frac{1}{2} \text{ — целое число}\} =$$
$$= \{\ldots, -1\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, 1\frac{1}{2}, 2\frac{1}{2}, \ldots\};$$

$$E_{\sqrt{2}} = \{z \in \mathbb{R} : \ z - \sqrt{2} \text{ — целое число}\} =$$
$$= \{\ldots, -1 + \sqrt{2}, \sqrt{2}, 1 + \sqrt{2}, 2 + \sqrt{2}, \ldots\}.$$

Рефлексивное, транзитивное, но кососимметричное отношение R на множестве A называется *частичным порядком*. Частичный порядок важен в тех ситуациях, когда мы хотим как-то охарактеризовать старшинство. Иными словами, решить при каких условиях считать, что один элемент множества превосходит другой.

Примеры частичных порядков.

- «$\leqslant$» на множестве вещественных чисел;
- «$\subset$» на подмножествах универсального множества;
- «... делит ...» на множестве натуральных чисел.

Множества с частичным порядком принято называть *частично упорядоченными множествами*.

Если R — отношение частичного порядка на множестве A, то при $x \neq y$ и $x\,R\,y$ мы называем x *предшествующим элементом* или *предшественником*, а y — *последующим*. У произвольно взятого элемента y может быть много предшествующих элементов. Однако если x предшествует y, и не существует таких элементов z, для которых $x\,R\,z$ и $z\,R\,y$, мы называем x *непосредственным предшественником*[1] y и пишем $x \prec y$.

Непосредственных предшественников можно условно изобразить с помощью графа, известного как *диаграмма Хассе*. Вершины графа изображают элементы частично упорядоченного множества A, и если $x \prec y$, то вершина x помещается ниже вершины y и соединяется с ней ребром.

Диаграмма Хассе выдаст полную информацию об исходном частичном порядке, если мы не поленимся подняться по всем цепочкам ребер.

[1] Иногда также говорят, что y покрывает x. — *Прим. перев.*

Пример 4.11. Дано, что отношение «... делитель ...» определяет частичный порядок на множестве $A = \{1, 2, 3, 6, 12, 18\}$. Составьте таблицу предшественников и непосредственных предшественников, после чего постройте соответствующую диаграмму Хассе.

Решение. Таблица и диаграмма приведены ниже.

Таблица 4.1

элемент	предшественник	непосредственный предшественник
1	нет	нет
2	1	1
3	1	1
6	1, 2, 3	2, 3
12	1, 2, 3, 6	6
18	1, 2, 3, 6	6

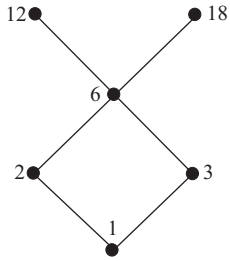


Рисунок 4.5. Диаграмма Хассе

Линейным порядком на множестве A называется отношение частичного порядка, при котором из любой пары элементов можно выделить предшествующий и последующий.

Примеры линейного порядка.

- «$\leqslant$» на множестве вещественных чисел;
- лексикографическое упорядочение слов в словаре.

Различные сортирующие процедуры в информатике требуют, чтобы элементы сортируемых множеств были линейно упорядочены. В этом случае они могут выдавать упорядоченный список. Другие приложения используют частичный порядок, предполагая, что в любом частично упорядоченном множестве найдется[2] *минималь-*

[2]Заметим, что в случае бесконечных множеств это не так. Например, в множестве $\mathbb{Z}$ относительно порядка «$\leqslant$» нет не минимального, ни максимального элемента. — *Прим. перев.*

ный элемент (не имеющий предшественников) и *максимальный* (не имеющий последующих элементов).

Частично упорядоченное множество из примера 4.11 обладает одним минимальным элементом, а именно, числом 1. С другой стороны, в нем есть два максимальных: 12 и 18. В этом множестве содержится несколько линейно упорядоченных подмножеств. Каждое из них соответствует цепочке ребер на диаграмме Хассе. Например, множество $\{1, 2, 6, 18\}$ линейно упорядочено относительно отношения «... делитель ...».

Набор упражнений 4

4.1. Выпишите множество упорядоченных пар и начертите ориентированный граф отношения, заданного матрицей:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \left[\begin{array}{cccc} И & Л & И & Л \\ И & Л & И & Л \\ Л & И & И & Л \end{array}\right] \end{array}.$$

4.2. Для каждого из следующих отношений на множестве натуральных чисел $\mathbb{N}$ опишите упорядоченные пары, принадлежащие отношениям:

$$R = \{(x, y) : \ 2x + y = 9\};$$
$$S = \{(x, y) : \ x + y < 7\};$$
$$T = \{(x, y) : \ y = x^2\}.$$

4.3. Пусть R — отношение на множестве $\{1, 2, 3, 4\}$, определяемое условием: $u\,R\,v$ тогда и только тогда, когда $u + 2v$ — нечетное число. Представьте R каждым из способов:

(а) как множество упорядоченных пар;

(б) в графической форме;

(в) в виде матрицы.

4.4. Определите, какие из следующих отношений на множестве людей рефлексивны, симметричны или транзитивны:

(а) «... имеет тех же родителей, что и ...»;

(б) «... является братом ...»;

(в) «... старше или младше, чем ...»;

(г) «... не выше, чем ...».

4.5. Определите, какие из приведенных ниже отношений на $\mathbb{Z}$ являются рефлексивными, симметричными, а какие транзитивными?

(а) «$x + y$ — нечетное число»;

(б) «$x + y$ — четное число»;

(в) «xy — нечетное число»;

(г) «$x + xy$ — четное число».

4.6. Перечислите упорядоченные пары, принадлежащие отношениям, заданным на множестве $\{x : \ x \in \mathbb{Z} \text{ и } 1 \leqslant x \leqslant 12\}$.

(а) $R = \{(x, y) : \ xy = 9\}$;

(б) $S = \{(x, y) : \ 2x = 3y\}$;

(в) замыкание R по транзитивности;

(г) замыкание S по транзитивности.

4.7. Ниже определены отношения на множествах. Опишите на словах замыкание по транзитивности в каждом случае.

(а) «x на один год старше, чем y» на множестве людей;

(б) $x = 2y$ на множестве $\mathbb{N}$ натуральных чисел;

(в) $x < y$ на множестве $\mathbb{R}$ вещественных чисел;

(г) «x является дочерью y» на множестве женщин.

4.8. Найдите замыкания по рефлексивности, по симметричности и по транзитивности отношения

$$\big\{(a, a),\ (b, b),\ (c, c),\ (a, c),\ (a, d),\ (b, d),\ (c, a),\ (d, a)\big\},$$

заданного на множестве $\{a, b, c, d\}$. Имеет ли смысл строить замыкание по антисимметричности?

4.9. Для каждого из следующих отношений эквивалентности на данном множестве A опишите блоки, на которые разбивается множество A:

(а) A — множество книг в библиотеке, а R определяется условием: $x\,R\,y$, если и только если цвет переплета x совпадает с цветом переплета y;

(б) $A = \mathbb{Z}$, R задается условием: $x\,R\,y$ тогда и только тогда, когда $x - y$ — четное число;

(в) A — множество людей, и $x\,R\,y$, если x имеет тот же пол, что и y;

(г) $A = \mathbb{R}^2$, R задается по правилу: $(a, b)\, R\, (c, d)$ в том случае, когда $a^2 + b^2 = c^2 + d^2$.

4.10. Отношение R на множестве $\mathbb{Z}$ определяется так: $x\, R\, y$ в том и только том случае, когда $x^2 - y^2$ делится на 3. Покажите, что R является отношением эквивалентности и опишите классы эквивалентности.

4.11. Нарисуйте диаграмму Хассе для каждого из следующих частично упорядоченных множеств:

(а) множество $\{1, 2, 3, 5, 6, 10, 15, 30\}$ с отношением «x делит y»;

(б) множество всех подмножеств в $\{1, 2, 3\}$ с отношением «X — подмножество Y».

4.12. Диаграмма Хассе частичного порядка R на множестве $A = \{a, b, c, d, e, f, g, h\}$ показана на рис. 4.6. Перечислите элементы R и найдите минимальный и максимальный элементы частично упорядоченного множества A.

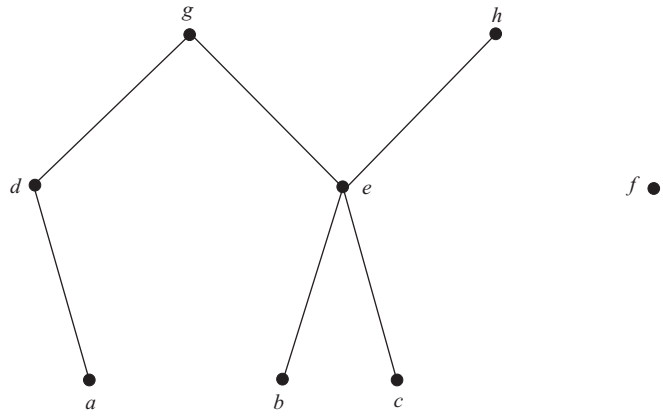


Рисунок 4.6.

4.13. Лексикографический (алфавитный) порядок работает следующим образом: у данных слов X и Y сравниваем букву за буквой, оставляя без внимания одинаковые, пока не найдем пару разных. Если в этой паре буква слова X стоит раньше (по алфавиту), нежели соответствующая буква слова Y, то X предшествует Y; если все буквы слова X совпадают с соответствующими буквами Y, но оно короче, то X предшествует Y, в противном случае, Y предшествует X.

Упорядочите следующие слова лексикографически: *бутылка, банан, бисквит, бивень* и *банджо*. Объясните, почему Вы выбрали именно такой порядок.

Краткое содержание главы

Бинарным отношением между множествами A и B называется подмножество R в $A \times B$. Если $A = B$, то говорят, что R — **отношение на** A.

Бинарное отношение между конечными множествами может быть описано на словах (при помощи подходящих предикатов), как множество упорядоченных пар, как орграф и с помощью матрицы.

Отношение R на множестве A называется

рефлексивным, если $x\,R\,x$ для всех $x \in A$;
симметричным, если $x\,R\,y \Rightarrow y\,R\,x$ для всех x, $y \in A$;
кососимметричным, если ($x\,R\,y$ и $y\,R\,x \;\Rightarrow\; x = y$) для всех x, $y \in A$;
транзитивным, если ($x\,R\,y$ и $y\,R\,z$) $\Rightarrow x\,R\,z$ для всех x, $y\,z \in A$.

Отношение R^* называют **замыканием отношения** R относительно свойства P, если

1) R^* обладает свойством P;
2) $R \subset R^*$;
3) R^* — подмножество любого другого отношения, содержащего R и обладающего свойством P.

Рефлексивное, симметричное и транзитивное отношение R на множестве A называется **отношением эквивалентности**. **Классом эквивалентности** элемента $x \in A$ является подмножество

$$E_x = \{z \in A :\ z\,R\,x\}.$$

Разбиение множества A представляет собой совокупность подмножеств A_1, A_2, ..., A_n в A, удовлетворяющих требованиям:

$$A = A_1 \cup A_2 \cup \cdots \cup A_n \text{ и } A_i \cap A_j = \varnothing \text{ при } i \neq j.$$

Подмножества A_i из предыдущего определения называются **блоками** разбиения. Если R — отношение эквивалентности на A, то различные классы эквивалентности образуют **разбиение** A.

Рефлексивное, кососимметричное и транзитивное отношение R на множестве A называется **частичным порядком**. Множества, на которых определено такое отношение, в свою очередь, называются **частично упорядоченными множествами**.

Линейный порядок на множестве — это такой частичный порядок, при котором можно сравнить любую пару элементов.

Если R — отношение частичного порядка на множестве A и $x\,R\,y$, $x \neq y$, то x называется **предшественником** y. В том случае, когда x предшествует y и нет такого элемента z, для которого $x\,R\,z$ и $z\,R\,y$, то говорят, что x — **непосредственный предшественник** y. Последний факт обозначают так: $x \prec y$.

Диаграмма Хассе представляет собой граф, чьи вершины изображают элементы частично упорядоченного множества. В том случае, когда $x \prec y$, вершина x располагается непосредственно под вершиной y и соединяется с ней ребром.

Приложение. Системы управления базами данных

Данные, хранящиеся в компьютере, называются *базой данных*. Программы, с помощью которых пользователь извлекает информацию из базы данных или вносит в нее изменения, называются *системами управления базами данных* (СУБД).

Таблица 4.2. T1 = Персональные данные

Личный номер	Фамилия	Пол	Дата рожд.	Семейное положение	Адрес
4000123	Джонс	Ж	1.2.83	не замужем	2 Мотт, Ньютон
5001476	Сингх	М	4.5.84	женат	4А Ньюраод, Сифорт
5112391	Смит	Ж	21.3.84	не замужем	17 Креснт, Сифорт
5072411	Смит	М	12.12.84	холост	21 Паддинг Лэйн, Витэм
5532289	Чинг	М	15.8.83	холост	4А Ньюраод, Сифорт
5083001	Грант	М	9.7.83	женат	18 Иффлейроад, Сифорт
5196236	Маккай	Ж	21.3.84	не замужем	133 Уффроад, Реадинг
4936201	Френк	Ж	7.10.77	замужем	11 Финнроад, Ньютон

Данные в компьютере, как правило, организованы в виде таблиц. Например, табл. 4.2 содержит информацию о группе студентов: личный номер студента, фамилию, пол, дату рождения, семейное положение и адрес. В табл. 4.3 занесена информация об успеваемости некоторых студентов по отдельным курсам. Эти таблицы составят

основу для наших обсуждений, хотя и не представляют практического интереса. Например, проблемы при работе с табл. 4.2 могут возникнуть при попытке извлечь информацию о двух различных Смитах, а в табл. 4.2 отсутствует детальная информация о некоторых из студентов, появляющихся в табл. 4.3.

Таблица 4.3. Т2 = Успеваемость

Фамилия	Основы матем.	Прогр.	Дискр. матем.	Вычисл. системы
Каммингс	отл	хор	удовл	отл
Джонс	хор	удовл	хор	неуд
Грант	удовл	хор	отл	удовл
Сингх	удовл	хор	отл	неуд
Френк	неуд	неуд	удовл	удовл
Маккай	отл	отл	хор	отл
Куксон	удовл	отл	отл	хор

Строки таблицы с n колонками, помеченными множествами A_1, A_2, ..., A_n можно представить как подмножество в прямом произведении $A_1 \times A_2 \times \cdots \times A_n$. Строки образуют список из n элементов, по одному из каждого A_i, а вся таблица представляет собой **n-арное отношение**.

Например, табл. 4.3 можно рассматривать как подмножество $T2$ в $A_1 \times A_2 \times A_3 \times A_4 \times A_5$, где A_1 — множество фамилий студентов, а $A_2 = A_3 = A_4 = A_5 = \{$отл, хор, удовл, неуд$\}$. Один из элементов этого пятинарного отношения — строка (Джонс, хор, удовл, хор, неуд), в которой записаны оценки Джонса, полученные им за четыре предмета.

Для извлечения информации и изменения содержания таблиц, соответствующих набору отношений, мы определим несколько основных операций над ними, а именно: **проект**, **соединение** и **выбор**. Это только три из многочисленных операций, созданных для манипулирования базами данных, теория которых опирается на язык множеств, отношений и функций.

Операция **проект** формирует новую таблицу из определенных столбцов старой. Например, **проект**(Т1, {**Фамилия, Адрес**}) создает табл. 4.4.

Задача 1. Найти
проект(Т2, {**Фамилия, Основы матем., Дискр. матем.**}).

Решение. Смотри табл. 4.5

Таблица 4.4. Т3 = проект(Т1, {Фамилия, Адрес})

Фамилия	Адрес
Джонс	2 Мотт, Ньютон
Сингх	4А Ньюраод, Сифорт
Смит	17 Креснт, Сифорт
Смит	21 Паддинг Лэйн, Витэм
Чинг	4А Ньюраод, Сифорт
Грант	18 Иффлейроад, Сифорт
Маккай	133 Уффроад, Реадинг
Френк	11 Финнроад, Ньютон

Таблица 4.5

Фамилия	Основы матем.	Дискр. матем.
Каммингс	отл	удовл
Джонс	хор	хор
Грант	удовл	отл
Сингх	удовл	отл
Френк	неуд	удовл
Маккай	отл	хор
Куксон	удовл	отл

Операция **соединение** объединяет две таблицы в бóльшую, выписывая в одну строку информацию, соответствующую общему атрибуту. Предположим, что R и S — отношения, представленные двумя таблицами, причем R — подмножество в прямом произведении $A_1 \times \cdots \times A_m \times B_1 \times \cdots \times B_n$, а S — в прямом произведении $A_1 \times \cdots \times A_m \times C_1 \times \cdots \times C_p$. В этом случае общие атрибуты представлены множествами $A_1, A_2, \ldots, A_m$. Соединение R и S — это подмножество в $A_1 \times \cdots \times A_m \times B_1 \times \cdots \times B_n \times C_1 \times \cdots \times C_p$, состоящее из элементов вида $(a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m, c_1, c_2, \ldots, c_p)$, где $(a_1, \ldots, a_m, b_1, \ldots, b_m)$ лежит в R, а $(a_1, \ldots, a_m, c_1, \ldots, c_p)$ — в подмножестве S.

Например, **соединение**(Т3, Т2) дает табл. 4.6.

Таблица 4.6

Фамилия	Адрес	Основы матем.	Прогр.	Дискр. матем.	Вычисл. системы
Джонс	2 Мотт, Ньютон	хор	удовл	хор	неуд
Грант	18 Иффлейроад, Сифорт	удовл	хор	отл	удовл
Сингх	4А Ньюраод, Сифорт	удовл	хор	отл	неуд
Френк	11 Финнроад, Ньютон	неуд	неуд	удовл	удовл
Маккай	133 Уффроад, Реадинг	отл	отл	хор	отл

Операция **выбор** отбирает строки таблицы, удовлетворяющие подходящему критерию. Например, **выбор**(T1, **Пол** = M и **Семейное положение** = Женат) верстает табл. 4.7.

Таблица 4.7

Личный номер	Фамилия	Пол	Дата рожд.	Семейное положение	Адрес
5001476	Сингх	M	4.5.84	женат	4A Ньюраод, Сифорт
5083001	Грант	M	9.7.83	женат	18 Иффлейроад, Сифорт

Задача 2. Найдите **выбор**(T2, **Дискр. матем.** = отл).

Решение. В новую таблицу (табл. 4.8) войдут только те строки таблицы T2, у которых в столбце, помеченном **Дискр. математика** будет стоять «отл».

Таблица 4.8

Фамилия	Основы матем.	Прогр.	Дискр. матем.	Вычисл. системы
Грант	удовл	хор	отл	удовл
Сингх	удовл	хор	отл	неуд
Куксон	удовл	отл	отл	хор

Как иллюстрируют следующие задачи, комбинация всех трех операций позволит нам извлекать различную информацию из баз данных.

Задача 3. Найдите таблицу, которая получится в результате операций:
R1 = **проект**(T2, {**Фамилия, Прогр., Вычисл. системы**});
R2 = **выбор**($R1$, **Вычисл. системы** = отл или **Прогр.** = отл);

Решение. Во-первых, все столбцы таблицы T2, отличные от **Фамилия**, **Прогр.** и **Вычисл. системы**, удаляются. В результате получится таблица R1. Затем, в новой таблице нужно оставить только те строки, в которых есть хотя бы одна оценка «отл», а остальные отбросить. Это даст нам требуемую таблицу R2 (табл. 4.9).

Таблица 4.9

Фамилия	Прогр.	Вычисл. системы
Каммингс	хор	отл
Маккай	отл	отл
Куксон	отл	хор

Задача 4. Найдите результат действий следующих операций:

R1 = **выбор**(T1, **пол** = Ж);
R2 = **проект**(T2,{**Фамилия, Дискр. матем.**});
R3 = **соединение**(R1, R2).

Решение. Прежде всего выберем из таблицы T1 строки, соответствующие студенткам, и составим из них таблицу R1. Затем удалим из T2 все столбцы, кроме двух выбранных, и получим таблицу R2. Общим атрибутом таблиц R1 и R2 является **Фамилия**. Соединив R1 и R2, получим искомую таблицу (табл. 4.10).

Таблица 4.10

Личный номер	Фамилия	Пол	Дата рожд.	Семейное положение	Адрес	Дискр. матем.
4000123	Джонс	Ж	1.2.83	не замужем	2 Мотт, Ньютон	хор
5196236	Маккай	Ж	21.3.84	не замужем	133 Уффроад, Реадинг	хор
4936201	Френк	Ж	7.10.77	замужем	11 Финнроад, Ньютон	удовл

Задача 5. Выпишите последовательность операций (**выбор**, **проект** и **соединение**) для определения имен и адресов всех тех студенток, которые получили оценку не ниже «хор» по обоим предметам: основы математики и дискретная математика.

Решение. Одна из последовательностей операций выглядит следующим образом.

R1 = **выбор**(T1, **пол** = Ж);
R2 = **выбор**(T2, **Дискр. матем.** = «отл» или **Дискр. матем.** = «хор»);
R3 = **выбор**(R2, **Основы матем.** = «отл» или **Основы матем.** = «хор»);
R4 = **соединение**(R1, R3);
R = **проект**(R4,{**Фамилия, Адрес**}).

ГЛАВА 5

ФУНКЦИИ

Функции играют центральную роль в математике, где они используются для описания любых процессов, при которых элементы одного множества каким-то образом переходят в элементы другого. Такие преобразования элементов — фундаментальная идея, имеющая первостепенное значение для всех вычислительных процессов.

Как мы увидим, функции представляют из себя специальный тип бинарных отношений. Однако перед тем, как мы определим функции, в этой главе мы продолжим работу над бинарными отношениями, начатую в главе 4. Здесь мы изучим два важнейших способа построения новых бинарных отношений из уже имеющихся. Эти способы основаны на вычислении обратного отношения и определении композиции отношений. Упомянутые операции особенно важны, когда мы от отношений переходим к функциям. После определения мы обсудим некоторые свойства функций, а закончим это обсуждение законом, известным как *принцип Дирихле*. Этот, на первый взгляд очень простой, факт даст нам возможность решать несвязанные друг с другом явным образом вычислительные задачи.

Как набор упражнений, так и приложение к этой главе посвящены применению функций в информатике. Мы познакомимся с *языком функционального программирования*, хотя и сильно ограниченным (из методологических соображений). Покажем, как с помощью композиции из элементарных функций строятся довольно сложные.

5.1. Обратные отношения и композиция отношений

Пусть R — бинарное отношение между множествами A и B. Определим *обратное отношение* R^{-1} между B и A по формуле:

$$R^{-1} = \{(b,\, a) :\ (a,\, b) \in R\}.$$

Например, обратным к отношению «... родитель ...» на множестве всех людей будет отношение «... ребенок ...».

На графическом языке обратное отношение получается обращением всех стрелок в орграфе, изображающем исходное отношение.

Вторая конструкция более сложна для понимания. Пусть R — бинарное отношение между множествами A и B, а S — бинарное отношение между B и третьим множеством C. *Композицией* R и S называется бинарное отношение между A и C, которое обозначается $S \circ R$ и определяется формулой:

$$S \circ R = \{(a,\, c) :\ a \in A,\ c \in C \text{ и } a\, R\, b,\ b\, S\, c \text{ для некоторого } b \in B\}.$$

Новое отношение устанавливает связь между элементами множеств A и C, используя элементы из B в качестве посредников.

Пример 5.1. Пусть R — отношение «a — сестра b», а S обозначает отношение «b — мать c» на множестве всех людей. Опишите на словах композиции: $S \circ R$ и $S \circ S$.

Решение. Если a — сестра b, а b — мать c, то a, очевидно, будет сестрой матери c, т. е. a приходится тетей c. Стало быть, отношение $S \circ R$ есть ни что иное, как «a — тетя c».

Аналогичными рассуждениями легко установить, что $S \circ S$ — это «a — бабушка c».

Пример 5.2. Предположим, что отношения R и S заданы орграфами, представленными на рис. 5.1. Найдите орграф, соответствующий композиции $S \circ R$.

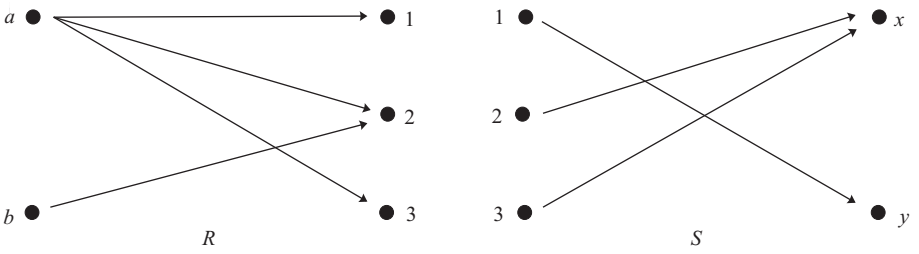


Рисунок 5.1. Орграфы отношений R и S

Решение. Используя орграфы, выпишем упорядоченные пары, принадлежащие отношениям.

$$R = \{(a,\, 1),\ (a,\, 2),\ (a,\, 3),\ (b,\, 2)\} \qquad \text{и} \qquad S = \{(1,\, y),\ (2,\, x),\ (3,\, x)\}.$$

Применим определение композиции отношений.

$$a\, R\, 1 \text{ и } 1\, S\, y \Rightarrow (a,\, y) \in S \circ R;$$
$$a\, R\, 2 \text{ и } 2\, S\, x \Rightarrow (a,\, x) \in S \circ R;$$

$$a\,R\,3 \text{ и } 3\,S\,x \Rightarrow (a,\,x) \in S \circ R;$$
$$b\,R\,2 \text{ и } 2\,S\,x \Rightarrow (b,\,x) \in S \circ R.$$

Теперь на рис. 5.2 изобразим орграф композиции.

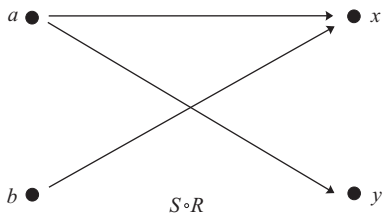


Рисунок 5.2. Орграф отношения $S \circ R$

Композицию бинарных отношений можно вычислить и с помощью матриц, их определяющих. Имея матрицы двух отношений, мы построим матрицу их композиции. Она называется *логическим* или *бу́левым произведением* матриц.

Рассмотрим три множества:

$$A = \{a_1,\,a_2,\,\ldots,\,a_n\},\ B = \{b_1,\,b_2,\,\ldots,\,b_m\} \text{ и } C = \{c_1,\,c_2,\,\ldots,\,c_p\}.$$

Предположим, что R — отношение между A и B, а S — отношение между B и C. Напомним, что матрица M отношения R определяется условием:

$$M(i,\,j) = \text{И если } (a_i,\,b_j) \in R,$$
$$M(i,\,j) = \text{Л если } (a_i,\,b_j) \notin R.$$

Аналогично, матрица N отношения S заполняется по правилу:

$$N(i,\,j) = \text{И если } (b_i,\,c_j) \in S,$$
$$N(i,\,j) = \text{Л если } (b_i,\,c_j) \notin S.$$

Если найдется такой элемент $b_k \in B$, что $a_i\,R\,b_k$ и $b_k\,S\,c_j$, то в i-ой строке матрицы M на k-ом месте стоит И. Кроме того, в j-ом столбце матрицы N на k-ом месте тоже будет стоять значение И. С другой стороны, поскольку по определению композиции отношений $a_i\,(S \circ R)\,c_j$, то значение $P(i,\,j)$ логической матрицы P отношения $S \circ R$ тоже равно И. Если же в i-ой строке матрицы M нет значений И, соответствующих такому же значению в j-ом столбце матрицы N, то $P(i,\,j) = \text{Л}$.

Таким образом, логическая матрица P композиции $S \circ R$ заполняется по следующему правилу:

$$\begin{aligned} P(i,\, j) = [M(i,\, 1)\ \textbf{и}\ N(1,\, j)]\ \textbf{или}\ & \\ \textbf{или}\ \ [M(i,\, 2)\ \textbf{и}\ N(2,\, j)]\ & \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ & \\ \textbf{или}\ \ [M(i,\, n)\ \textbf{и}\ N(n,\, j)]. & \end{aligned}$$

Будем писать $P = MN$ для обозначения булева произведения матриц.

Пример 5.3. Пусть R и S — отношения из примера 5.2. Вычислите логическую матрицу отношения $S \circ R$ с помощью булева произведения логических матриц отношений R и S.

Решение. Отношение R между $A = \{a,\, b\}$ и $B = \{1,\, 2,\, 3\}$ задается матрицей

$$M = \begin{bmatrix} \text{И} & \text{И} & \text{И} \\ \text{Л} & \text{И} & \text{Л} \end{bmatrix},$$

строки и столбцы которой помечены элементами множеств A и B в том порядке, в котором они выписаны.

Аналогично, S — отношение между $B = \{1,\, 2,\, 3\}$ и $C = \{x,\, y\}$, заданное матрицей

$$N = \begin{bmatrix} \text{Л} & \text{И} \\ \text{И} & \text{Л} \\ \text{И} & \text{Л} \end{bmatrix}.$$

Значит, логическая матрица P отношения $S \circ R$ равна булеву произведению

$$P = \begin{bmatrix} \text{И} & \text{И} & \text{И} \\ \text{Л} & \text{И} & \text{Л} \end{bmatrix} \begin{bmatrix} \text{Л} & \text{И} \\ \text{И} & \text{Л} \\ \text{И} & \text{Л} \end{bmatrix}.$$

В матрице M есть две строки, а в матрице N — два столбца. Поэтому матрица P состоит из двух строк и двух столбцов.

Ячейка $P(1,\, 1)$ заполняется по первой строке матрицы M и первому столбцу матрицы N. Более точно,

$$\begin{aligned} P(1,\, 1) = \begin{bmatrix} \text{И} & \text{И} & \text{И} \end{bmatrix} \begin{bmatrix} \text{Л} \\ \text{И} \\ \text{И} \end{bmatrix} &= (\text{И}\ \textbf{и}\ \text{Л})\ \textbf{или}\ (\text{И}\ \textbf{и}\ \text{И})\ \textbf{или}\ (\text{И}\ \textbf{и}\ \text{И}) \\ &= \text{Л}\ \textbf{или}\ \text{И}\ \textbf{или}\ \text{И} = \text{И}. \end{aligned}$$

Заметим, что в первой строке матрицы M на втором и третьем местах стоит И, так же как и в первом столбце матрицы N. Этого достаточно для обоснованного заключения: $P(1,\ 1) = \text{И}$.

Сравнивая первую строку матрицы M со вторым столбцом матрицы N, мы видим, что в обоих случаях на первом месте стоит значение И. Следовательно, $P(1,\ 2) = \text{И}$.

Таким же способом, смотря на вторую строку матрицы M и первый столбец матрицы N, определяем $P(2,\ 1) = \text{И}$.

Наконец, $P(2,\ 2) = \text{Л}$, так как вторая строка матрицы M и второй столбец матрицы N не имеют значения И на одинаковых местах.

Итак,

$$N = \begin{bmatrix} \text{И} & \text{И} \\ \text{И} & \text{Л} \end{bmatrix}.$$

Пример 5.4. Отношение R на множестве $A = \{1,\ 2,\ 3,\ 4,\ 5\}$ задается матрицей

$$\begin{bmatrix} \text{Л} & \text{Л} & \text{И} & \text{И} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix}.$$

Вычислить матрицу композиции $R \circ R$ и объяснить, почему отношение R не обладает свойством транзитивности.

Решение. Матрица композиции $R \circ R$ равна

$$\begin{bmatrix} \text{Л} & \text{Л} & \text{И} & \text{И} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix} \begin{bmatrix} \text{Л} & \text{Л} & \text{И} & \text{И} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix} = \begin{bmatrix} \text{Л} & \text{Л} & \text{И} & \text{И} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \end{bmatrix}.$$

Элементы композиции $R \circ R$ имеют вид $(x,\ z)$, где $x\,R\,y$ и $y\,R\,z$ для какого-нибудь $y \in A$. Поэтому в случае транзитивности R композиция $R \circ R$ должна быть подмножеством R. Однако из расположения значения И в матрицах, выписанных выше, видно, что $R \circ R$ содержит пары, которые не лежат в R. Именно поэтому отношение R не транзитивно.

5.2. Функции

Отношения эффективно применяются для описания связей между парами элементов, выбранных из двух множеств A и B. Функции — это частный случай бинарных отношений, на которые наложены дополнительные ограничения.

Функцией из множества A в множество B называется бинарное отношение, при котором *каждый* элемент множества A связан с *единственным* элементом множества B. Другими словами, для каждого $a \in A$ существует ровно одна пара из отношения вида (a, b).

В графических терминах функция описывается таким графом, у которого из каждой вершины, изображающей элементы множества A, выходит *ровно одна* стрелка.

Например, на рис. 5.3 изображен граф, представляющий функцию из множества $\{a, b, c\}$ в $\{1, 2\}$, состоящую из пар $(a, 1)$, $(b, 1)$ и $(c, 2)$.

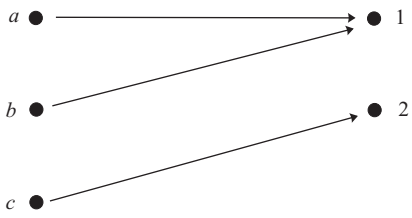


Рисунок 5.3.

Пример 5.5. Определите, какие из следующих отношений между множествами $A = \{a, b, c\}$ и $B = \{1, 2, 3\}$ являются функциями из множества A в B.

(а) $f = \{(a, 1),\ (a, 2),\ (b, 3),\ (c, 2)\}$;

(б) $g = \{(a, 1),\ (b, 2),\ (c, 1)\}$;

(в) $h = \{(a, 1),\ (c, 2)\}$.

Решение.

(а) Отношение f — не функция, поскольку элементу a соответствуют два разных элемента множества B: 1 и 2.

(б) Отношение g является функцией.

(в) Последнее отношение функцией не является, поскольку элементу b не соответствует ни одного элемента.

Пример 5.6. Какие из отношений:

(а) «x — брат или сестра y» на множестве всех людей;

(б) отношение на множестве $\mathbb{Z}$, заданное парами: $\{(x,\ x^2) :\ x \in \mathbb{Z}\}$;

(в) отношение на множестве $\mathbb{R}$, заданное парами: $\{(x,\ y) :\ x = y^2\}$

являются функциями?

Решение.

(а) Это не функция, поскольку есть люди с несколькими братьями и сестрами, а также бывают семьи с единственным ребенком.

(б) А вот это отношение как раз функция, поскольку по каждому целому числу x его квадрат x^2 определяется однозначно.

(в) Последнее отношение — не функция, так как, например, обе упорядоченные пары: $(2,\ \sqrt{2})$ и $(2,\ -\sqrt{2})$ — ему принадлежат. Кроме того, в нем отсутствуют пары $(x,\ y)$ с отрицательными x.

Пусть f — функция из множества A в множество B. Поскольку для каждого $x \in A$ существует единственным образом определенный $y \in B$, такой, что $(x,\ y) \in f$, мы будем писать: $y = f(x)$, и говорить, что функция f *отображает* множество A в множество B, а $f(x)$ называть *образом* x при отображении f или *значением* f, соответствующим аргументу x.

Кроме того, можно написать $f : A \longrightarrow B$, чтобы подчеркнуть, что функция f переводит элементы из A в элементы из B. Множество A принято называть *областью определения*, а B — *областью значений* функции[1] f.

Для уточнения подмножества элементов, в которые переводятся элементы из A функцией f, вводят понятие «образ» или «множество значений функции». А именно, *множеством значений* функции f называется подмножество в B, состоящее из образов всех элементов $x \in A$. Оно обозначается символом $f(A)$ и формально определяется так:

$$f(A) = \{f(x) :\ x \in A\}.$$

Диаграмма Венна на рис. 5.4 служит удобной иллюстрацией функции, определенной на множестве A со значениями в множестве B.

[1] Довольно часто говорят, что функция f *определена* или *задана* на множестве A со значениями в множестве B. — *Прим. перев.*

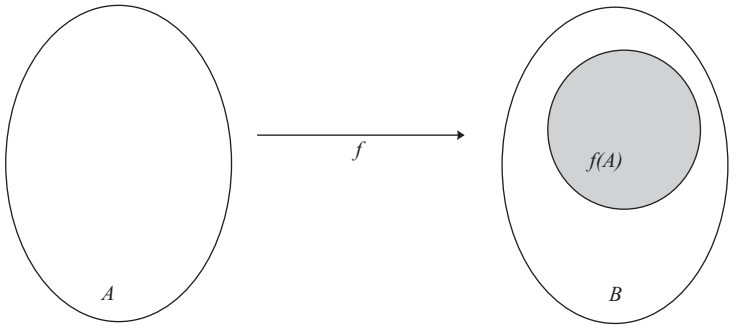


Рисунок 5.4. Диаграмма Венна функции $f : A \longrightarrow B$

Когда мы работаем с функцией $f : A \longrightarrow B$, где A и B — бесконечные множества, мы не можем нарисовать граф этого отношения. Следует обратиться к традиционной математической идее графического представления функции, а именно, ее *графику*.

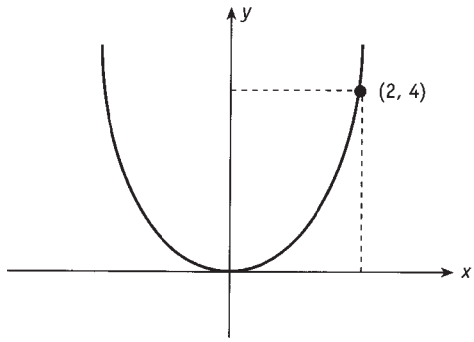


Рисунок 5.5. График функции $y = f(x)$

Например, график функции $f : \mathbb{R} \longrightarrow \mathbb{R}$, заданной формулой $f(x) = x^2$, изображен на рис. 5.5. Горизонтальная ось помечается x и обозначает множество определения $\mathbb{R}$. Вертикальна ось помечается y и заменяет собой область значений функции (в нашем случае тоже $\mathbb{R}$). Кривая на рисунке, т. е. график функции, состоит из точек (x, y) прямого произведения $\mathbb{R} \times \mathbb{R}$, для которых $y = f(x)$. Так как, например, $f(2) = 4$, точка $(2, 4)$ лежит на этой кривой, что видно на рисунке.

Пример 5.7. Сделайте набросок графика функции $g : \mathbb{R} \longrightarrow \mathbb{R}$, заданной формулой $g(x) = 2 - x$. Найдите ее значения при $x = 2$ и $x = 3$. Отметьте соответствующие точки на графике.

Решение. Чертеж графика приведен на рис. 5.6. Отмеченные точки соответствуют парам: $g(2) = 0$ и $g(3) = -1$.

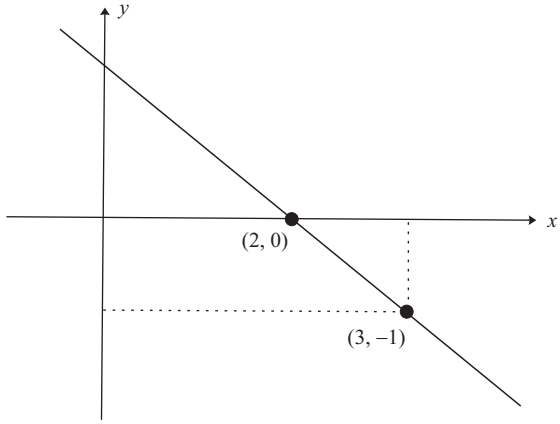


Рисунок 5.6. График функции $g(x) = 2 - x$

Теперь займемся некоторыми важными свойствами функций.

Пусть $f : A \longrightarrow B$ — функция. Мы будем называть ее *инъективной* или *инъекцией*, или *взаимно однозначной*, если

$$f(a_1) = f(a_2) \Rightarrow a_1 = a_2$$

для всех $a_1, a_2 \in A$.

Это определение логически эквивалентно тому, что

$$a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2),$$

т. е. у инъективной функции нет повторяющихся значений. Иными словами, разные входные данные дают различные выходные данные.

Будем называть функцию f *сюръективной* или *сюръекцией*, или *функцией «на»*, если множество ее значений совпадает с областью значений. Это означает, что для каждого $b \in B$ найдется такой $a \in A$, что $b = f(a)$. Таким образом, каждый элемент области значений является образом какого-то элемента из области определения f.

Мы называем f *биективной* функцией или просто *биекцией*, если она как инъективна, так и сюръективна.

Пример 5.8. Определите, какие из функций, изображенных на рис. 5.7, инъективны, а какие сюръективны. Перечислите все биекции.

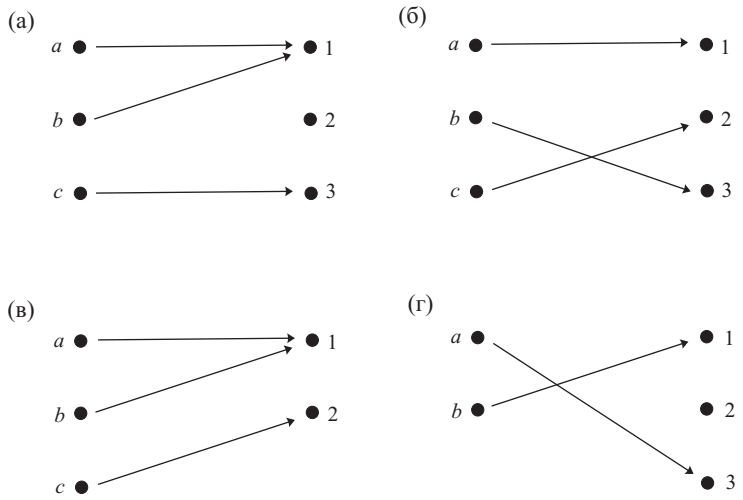


Рисунок 5.7.

Решение.

(а) Данная функция не инъективна, поскольку значение 1 соответствует как a, так и b. Она не является и сюръекцией, ввиду того, что в элемент 2 ничего не переходит.

(б) Данная функция инъективна, так как не имеет повторяющихся значений. Она же и сюръективна, поскольку множество ее значений совпадает со всей областью значений.

(в) Значение 1 эта функция принимает как на a, так и на b. Следовательно, она не инъективна. Однако данная функция сюръективна, поскольку в ее множество значений входят все элементы области значений.

(г) Последняя функция инъективна, но сюръективна.

Только в случае (б) мы имеем биекцию.

Пример 5.9. Покажите, что функция $h : \mathbb{Z} \longrightarrow \mathbb{Z}$, заданная формулой $h(x) = x^2$ и не инъективна, и не сюръективна.

Решение. Прежде всего заметим, что данная функция не совпадает с функцией $f : \mathbb{R} \longrightarrow \mathbb{R}$, заданной той же формулой $f(x) = x^2$, чей график был приведен выше. Несмотря на одинаковые формулы, области определения и значений функции h ограничены только

целыми числами. Фактически график h, изображенный на рис. 5.8, состоит из серии изолированных точек.

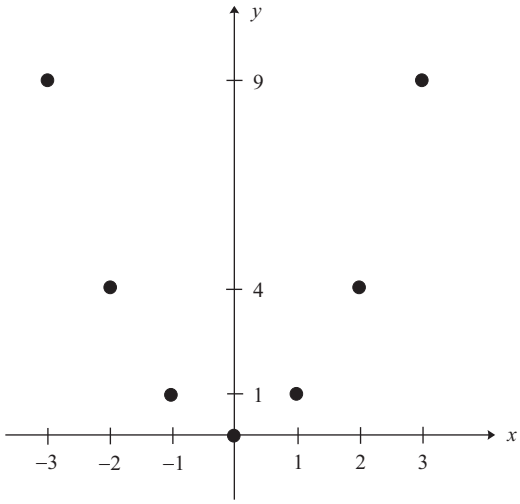


Рисунок 5.8. График функции $h(x) = x^2$, определенной на множестве $\mathbb{Z}$.

Чтобы показать, что h не инъективна, достаточно найти такие разные целые числа $a_1 \neq a_2$, для которых $h(a_1) = h(a_2)$. На графике видно много таких пар целых чисел. Например, $a_1 = 2$ и $a_2 = -2$.

Что касается противоречия с сюръективностью, то можно найти такое число, которое содержится в области значений, но не является значением функции h. Опять-таки наш график помогает при решении поставленной задачи. Любое отрицательное число, в частности -1, годится в качестве примера.

Пример 5.10. Покажите, что функция $k : \mathbb{R} \longrightarrow \mathbb{R}$, заданная формулой $k(x) = 4x + 3$, является биекцией.

Решение. Предположим, что $k(a_1) = k(a_2)$, т. е.

$$4a_1 + 3 = 4a_2 + 3.$$

Из равенства следует, что $4a_1 = 4a_2$, откуда $a_1 = a_2$. Значит, k — инъекция.

Пусть $b \in \mathbb{R}$. Покажем, что найдется такое вещественное число $a \in \mathbb{R}$, что $h(a) = b$. Ясно, что в качестве a можно взять $a = \frac{1}{4}(b-3)$. Итак, k — сюръективная функция.

Поскольку k является одновременно и сюръекцией и инъекцией, то она — биективная функция.

5.3. Обратные функции и композиция функций

Напомним, что любая функция $f : A \longrightarrow B$ — бинарное отношение. Поэтому мы можем построить обратное отношение f^{-1}. Если при этом мы снова получим функцию, то исходную функцию будем называть *обратимой* и писать $f^{-1} : B \longrightarrow A$ для обозначения *обратной функции*.

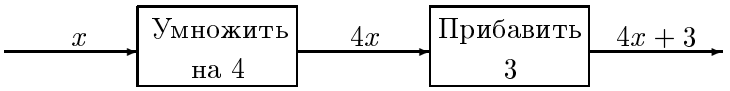
Функция f состоит из пар вида $(a,\ b)$, где $b = f(a)$. Когда f обратима, обратная функция f^{-1} состоит из пар $(b,\ a)$, где $a = f^{-1}(b)$. Значит, обратимая функция должна удовлетворять условию: если $f(a) = b$, то $f^{-1}(b) = a$. Другими словами, обратная функция переворачивает действие исходной.

Пример 5.11. Какие из функций примера 5.8 обратимы?

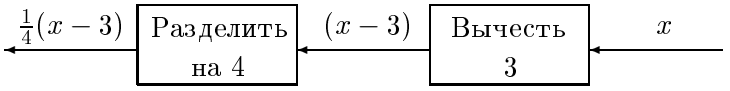
Решение. Обратное отношение получается простым обращением стрелок в орграфе, его представляющем. Очевидно, только в случае (б) мы имеем обратимую функцию.

Аккуратная доводка идеи переориентации стрелок может быть использована для обращения не очень сложных функций.

Рассмотрим функцию $k : \mathbb{R} \longrightarrow \mathbb{R}$, $k = 4x + 3$ (см. пример 5.10). Действие k можно разбить на несколько шагов:



Две элементарных команды: «умножить на 4» и «прибавить 3» — легко обращаются: «разделить на 4» и «вычесть 3», соответственно. Таким образом, обращение стрелок дает обратную функцию:



Итак, $k^{-1} : \mathbb{R} \longrightarrow \mathbb{R}$ задается формулой $k^{-1} = \frac{1}{4}(x - 3)$.

Необходимо учитывать, что тот же самый ответ может быть получен и алгебраическим путем. Пусть $y = k(x)$, так что $x = k^{-1}(y)$. Нам известно, что $y = 4x + 3$. Выразив из этого равенства y, получим, что $x = \frac{1}{4}(y-3)$. Следовательно, $k^{-1}(y) = \frac{1}{4}(y-3)$ или, поскольку мы обычно используем x в качестве аргумента, $k^{-1}(x) = \frac{1}{4}(x-3)$, как нам уже известно.

Как мы видели в примерах, обе функции, обладающие свойством обратимости, были биективными. Это не случайное совпадение: обратимы только биекции. Сейчас мы докажем критерий обратимости для общей функции $f : A \longrightarrow B$.

Теорема. Функция f обратима тогда и только тогда, когда она биективна.

Доказательство. Доказательство состоит из двух частей.

Сначала мы докажем, что биективная функция обратима. Пусть $f : A \longrightarrow B$ — биекция. Как отношение, f можно определить с помощью предикатов:

$$f = \big\{(a,\, b) :\ a \in A \textbf{ и } f(a) = b\big\}.$$

По определению обратного отношения имеем:

$$f^{-1} = \big\{(b,\, a) :\ a \in A \textbf{ и } f(a) = b\big\}.$$

Поскольку f сюръективна, для любого элемента $b \in B$ найдется такой $a \in A$, что $f(a) = b$. Кроме того, ввиду инъективности функции f такой элемент a определяется по b единственным образом. Следовательно, все пары отношения f^{-1} обладают тем свойством, что каждый элемент множества B соответствует единственному элементу множества A. А это, по определению, и означает, что f^{-1} является функцией, как и утверждалось.

Теперь покажем, что обратимая функция обязана быть биективной. Предположим, что обратное отношение f^{-1} — функция. Тогда для любого $b \in B$ существует единственный элемент $a \in A$, для которого $(b,\, a) \in f^{-1}$. Следовательно, $(a,\, b) \in f$, т. е. $b = f(a)$. Этим доказана сюръективность f.

Для проверки инъективности функции f поступим следующим образом. Предположим, что $f(a_1) = f(a_2)$. Тогда обе пары: $(f(a_1),\, a_1)$ и $(f(a_2),\, a_2)$ — лежат в f^{-1}. Так как f^{-1} является функцией, имеет место равенство: $a_1 = a_2$, так что f инъективна.

Таким образом, f является биекцией, как и утверждалось. Доказательство проведено полностью: обратимая функция биективна и, в обратную сторону, биективная функция обратима. ■

Пример 5.12. Пусть

$$A = \{x :\ x \in \mathbb{R} \text{ и } x \neq 1\}$$

и $f : A \longrightarrow A$ задается формулой:

$$f(x) = \frac{x}{x-1}.$$

Показать, что f биективна и найти обратную ей функцию.

Решение. Предположим, что $f(a_1) = f(a_2)$. Тогда

$$\frac{a_1}{a_1 - 1} = \frac{a_2}{a_2 - 1}.$$

Значит,

$$a_1 a_2 - a_1 = a_1 a_2 - a_2,$$

откуда $a_1 = a_2$. Следовательно, f инъективна.

Пусть $b \in A$ — элемент области значений f. Найдем элемент a из множества A, удовлетворяющий условию: $f(a) = b$, т. е.

$$\frac{a}{a - 1} = b.$$

Разрешая полученное уравнение относительно a, найдем

$$a = \frac{b}{b - 1}.$$

Нам удалось найти элемент $a = \frac{b}{b-1} \in A$, для которого $f(a) = b$. Это свидетельствует о сюръективности f.

Итак, мы показали, что функция f как сюръективна, так и инъективна. Значит, она является биекцией.

Обратная функция определяется условием: $f^{-1}(b) = a$ всегда, когда $f(a) = b$. Но, как мы уже выяснили при доказательстве сюръективности f,

$$a = \frac{b}{b - 1}.$$

Таким образом, $f^{-1} : A \longrightarrow A$,

$$f^{-1}(x) = \frac{x}{x - 1},$$

т. е. функция f обратна сама себе.

Мы закончим этот параграф изучением композиции функций. Этим понятием более легко овладеть, нежели композицией общих отношений, с которой мы познакомились в начале главы.

Если $f : A \longrightarrow B$ и $g : B \longrightarrow C$ — функции, то композиция отношений $g \circ f$ между A и C состоит из пар вида (a, c), где для некоторого $b \in B$ $(a, b) \in f$ и $(b, c) \in g$. Однако элемент $b = f(a)$ однозначно определяется по a, поскольку f — функция. Более того, элемент $c = g(b)$ также однозначно определяется по b (g тоже

функция). Следовательно, элемент $c = g(f(a))$ единственным образом определяется элементом a и, стало быть, композиция функций f и g — снова функция.

Итак, композиция $g \circ f : A \longrightarrow C$ является функцией, действующей по правилу $(g \circ f)(x) = g(f(x))$.

Пример 5.13. Рассмотрим две функции: $f : \mathbb{R} \longrightarrow \mathbb{R}$, $f(x) = x^2$ и $g : \mathbb{R} \longrightarrow \mathbb{R}$, $g(x) = 4x + 3$. Вычислить $g \circ f$, $f \circ g$, $f \circ f$ и $g \circ g$.

Решение. Все четыре новых функции определены на $\mathbb{R}$ со значениями в $\mathbb{R}$.

$$(g \circ f)(x) = g(f(x)) = g(x^2) = 4x^2 + 3;$$

$$(f \circ g)(x) = f(g(x)) = f(4x + 3) = (4x + 3)^2 = 16x^2 + 24x + 9;$$

$$(f \circ f)(x) = f(f(x)) = f(x^2) = x^4;$$

$$(g \circ g)(x) = g(g(x)) = g(4x + 3) = 4(4x + 3) + 3 = 16x + 15.$$

В современных языках программирования функции используются очень широко. Они дают нам возможность выделить отдельные вычисления в *подпрограммы*. В большинстве языков есть специальные библиотеки с наиболее часто применяющимися функциями, такими как $\sin x$, $\log x$, $|x|$ и т. д. Кроме того, в них легко создавать собственные функции.

В некоторых особенно мощных языках, известных как языки функционального программирования, основные операторы определены в терминах функций. Главная особенность таких языков — возможность построения новых, более сложных, операторов из основных. Чтобы уметь это делать, нам необходимо в совершенстве овладеть композицией функций. В приложении к этой главе как раз и будет проиллюстрирована такая необходимость.

5.4. Принцип Дирихле

Пусть $f : A \longrightarrow B$ — функция, причем как A, так и B — конечные множества. Предположим, что A состоит из n элементов: $a_1, a_2, \ldots, a_n$. Принцип Дирихле гласит, что если $|A| > |B|$, то по крайней мере одно значение f встретится более одного раза[1]. Проще говоря, найдется пара элементов $a_i \neq a_j$, для которых $f(a_i) = f(a_j)$.

[1] Допуская некоторую вольность, принцип Дирихле можно переформулировать в легко запоминающейся форме: нельзя рассадить 10 зайцев в 9 клеток так, чтобы в каждой клетке сидел один заяц. — *Прим. перев.*

Чтобы убедиться в истинности принципа, предположим, что для любой пары разных индексов $i \neq j$ мы имеем: $f(a_i) \neq f(a_j)$. Тогда множество B содержит по крайней мере n различных элементов: $f(a_1)$, $f(a_2)$, ..., $f(a_n)$. И уж во всяком случае, $|B| \geqslant n$, что противоречит предположению: $n = |A| > |B|$. Следовательно, есть хотя бы два разных элемента $a_i, a_j \in A$, для которых $f(a_i) = f(a_j)$.

Пример 5.14. В автобусе едет 15 людей. Покажите, что по крайней мере у двоих из них день рождения в одном и том же месяце.

Решение. Множество людей в автобусе обозначим буквой A, а множество всех 12 месяцев обозначим через B. Рассмотрим функцию $f : A \longrightarrow B$, сопоставляющую каждому человеку из автобуса месяц его рождения. Так как $|A| = 15$, а $|B| = 12$, то $|A| > |B|$. По принципу Дирихле функция f должна иметь повторяющиеся значения, т. е. найдутся два человека с одним и тем же месяцем рождения.

Задачу из примера 5.14 легко решить и менее формальным рассуждением. Дано 15 человек и 12 месяцев. Поэтому совершенно очевидно, что хотя бы двое из них родились в один и тот же месяц. При этом трудно понять, зачем нам применять формальное рассуждение, как это было в предыдущем примере. Однако, как мы увидим дальше, более сложные задачи могут быть решены только с помощью принципа Дирихле, если, конечно, нам удастся обнаружить подходящую функцию. Поиск нужной функции — всегда самая трудная часть решения. Ключевая идея, на которой основан принцип Дирихле, состоит в том, что функция f размещает некоторое количество объектов (элементов множества A) в меньшее число клеток (элементы множества B). Поэтому по крайней мере два объекта попадут в одну.

Пример 5.15. Какое наименьшее число фамилий должно быть записано в телефонном справочнике, чтобы с гарантией можно было утверждать, что хотя бы две фамилии начинаются с одной и той же буквы и заканчиваются одинаковыми буквами?

Решение. Пусть A — множество фамилий в справочнике, а B — множество пар букв, выписанных из стандартного алфавита русского языка, насчитывающего 33 буквы. Обозначим через $f : A \longrightarrow B$ функцию, которая каждой фамилии справочника ставит в соответствие пару букв: первую и последнюю буквы фамилии. Например, $f(Кузнецов) = (к, в)$. Множество B содержит $33 \cdot 33 = 1\,089$ пар букв. Принцип Дирихле гарантирует нам, что если $|A| > |B| = 1\,089$,

то найдется по крайней мере две фамилии, начинающиеся и оканчивающиеся на одинаковые буквы. Поэтому телефонный справочник должен содержать не менее 1 090 фамилий[2].

Пример 5.16. Покажите, что какие бы пять цифр из 1, 2, 3, 4, 5, 6, 7 и 8 мы ни выбрали, найдутся хотя бы две из них, сумма которых равна 9.

Решение. Перечислим пары цифр, дающих в сумме 9.

$$\{1, 8\},\ \{2, 7\},\ \{3, 6\},\ \{4, 5\}.$$

Обозначим через A множество выбранных пяти цифр (не важно каких конкретно), а через B следующее множество пар:

$$B = \big\{\{1, 8\},\ \{2, 7\},\ \{3, 6\},\ \{4, 5\}\big\}.$$

Рассмотрим функцию $f : A \longrightarrow B$, сопоставляющую каждой цифре из пятерки пару из множества B, которая в ней содержится. Например, $f(3) = \{3, 6\}$. По принципу Дирихле хотя бы две цифры из множества A попадут в одну и ту же пару. Короче говоря, две из пяти цифр дадут в сумме 9.

Принцип можно обобщить следующим образом. Рассмотрим функцию $f\colon A \longrightarrow B$, где A и B — конечные множества. Если $|A| > k|B|$ для некоторого натурального числа k, то найдется такое значение функции f, которое она будет принимать по крайней мере $k+1$ раз. Это утверждение верно потому, что если каждое значение функция f принимает не более чем k раз, то все множество A состоит не более чем из $k|B|$ элементов.

Пример 5.17. Какое наименьшее число фамилий должно быть записано в телефонном справочнике, чтобы с гарантией можно было утверждать, что хотя бы пять фамилий начинаются с одной и той же буквы алфавита и заканчиваются одинаковыми буквами?

Решение. Пусть $f : A \longrightarrow B$ — функция из примера 5.15. Как мы уже подсчитали, B состоит из 1 089 элементов. Чтобы по крайней мере пять фамилий начинались и оканчивались одинаковыми буквами, нам нужно, чтобы $|A| > 4|B| = 4\,356$. Таким образом, телефонный справочник должен содержать не менее чем 4 357 абонентов.

[2]Если учесть, что фамилии не могут начинаться с букв «Ь» и «Ъ», то требуемый объем справочника окажется меньше. Попытайтесь его найти. — *Прим. перев.*

Пример 5.18. Покажите, что в любой группе из шести человек найдутся трое, знакомые друг с другом, или наоборот, совершенно не знающие друг друга.

Решение. Пусть x — один из шести людей, A — множество оставшихся пяти людей в группе и $B = \{0, 1\}$. Определим функцию $f : A \longrightarrow B$ по правилу:

$$f(a) = \begin{cases} 0, & \text{если } a \text{ не знаком с } x, \\ 1, & \text{если } a \text{ знаком с } x. \end{cases}$$

Поскольку $5 = |A| > 2|B|$, то найдется три человека, которые либо все знакомы с x, либо все трое его не знают.

Предположим теперь, что три человека a, b и c знакомы с x. Если все три друг с другом не знакомы, то мы получаем решение задачи. В противном случае, какая-то пара, скажем a и b знает друг друга. Но они же знакомы и с x. Стало быть, трое людей: a, b и x — хорошие знакомые. Аналогично разбирается случай, когда нашлась тройка людей, которые с x не знакомы.

Предыдущие примеры наглядно свидетельствуют, что применение принципа Дирихле требует аккуратной постановки задачи и, довольно часто, тонких логических рассуждений. Удачно, что в наших примерах было сравнительно несложно подсчитать количество элементов в множествах A и B. К сожалению, так бывает далеко не всегда, и в следующей главе мы разовьем разнообразные методы пересчета, которые предоставят нам возможность определять мощность конечных множеств, чьи элементы выбираются определенными способами.

Набор упражнений 5

5.1. Пусть R — отношение между множествами

$$\{1, 2, 3\} \qquad \text{и} \qquad \{1, 2, 3, 4\},$$

заданное перечислением пар:

$$R = \big\{(1, 1),\ (2, 3),\ (2, 4),\ (3, 1),\ (3, 4)\big\}.$$

Кроме того, S — отношение между множествами

$$\{1, 2, 3, 4\} \qquad \text{и} \qquad \{1, 2\},$$

состоящее из пар:

$$S = \{(1,\ 1),\ (1,\ 2),\ (2,\ 1),\ (3,\ 1),\ (4,\ 2)\}.$$

Вычислите R^{-1}, S^{-1} и $S \circ R$. Проверьте, что

$$(S \circ R)^{-1} = R^{-1} \circ S^{-1}.$$

5.2. Пусть R — отношение «... родитель ...», а S — отношение «... брат ...» на множестве всех людей. Дайте краткое словесное описание отношениям: R^{-1}, S^{-1}, $R \circ S$, $S^{-1} \circ R^{-1}$ и $R \circ R$.

5.3. Покажите, что если R — отношение частичного порядка на множестве A, то обратное к нему отношение R^{-1} тоже устанавливает частичный порядок на множестве A. Какова связь между максимальным и минимальным элементами относительно R и R^{-1}?

5.4. Отношения R и S заданы матрицами M и N соответственно, где

$$M = \begin{bmatrix} \text{И} & \text{Л} & \text{Л} \\ \text{И} & \text{Л} & \text{И} \end{bmatrix} \qquad \text{и} \qquad N = \begin{bmatrix} \text{И} & \text{Л} & \text{И} & \text{И} \\ \text{И} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{И} & \text{И} \end{bmatrix}.$$

Вычислите булево произведение MN. Какое отношение задается этим произведением?

5.5. Пусть $A = \{0,\ 2,\ 4,\ 6\}$ и $B = \{1,\ 3,\ 5,\ 7\}$. Какие из нижеперечисленных отношений между множествами A и B являются функциями, определенными на A со значениями в B?

(а) $\{(6,\ 3),\ (2,\ 1),\ (0,\ 3),\ (4,\ 5)\}$;

(б) $\{(2,\ 3),\ (4,\ 7),\ (0,\ 1),\ (6,\ 5)\}$;

(в) $\{(2,\ 1),\ (4,\ 5),\ (6,\ 3)\}$;

(г) $\{(6,\ 1),\ (0,\ 3),\ (4,\ 1),\ (0,\ 7),\ (2,\ 5)\}$.

Какие из найденных функций инъективны, а какие сюръективны?

5.6. Про каждую из следующих функций, чьи области определения и значений совпадают с $\mathbb{Z}$, скажите, являются ли они инъекциями, сюръекциями или биекциями.

(а) $f(n) = 2n + 1$;

(б) $g(n) = \begin{cases} \frac{n}{2}, & \text{если } n \text{ четно,} \\ 2n, & \text{если } n \text{ нечетно;} \end{cases}$

(в) $h(n) = \begin{cases} n+1, & \text{если } n \text{ четно,} \\ n-1, & \text{если } n \text{ нечетно.} \end{cases}$

5.7. Изобразите графики функций:

(а) $f : \mathbb{Z} \longrightarrow \mathbb{Z}$, $f(x) = x^2 + 1$;

(б) $g : \mathbb{N} \longrightarrow \mathbb{N}$, $g(x) = 2^x$;

(в) $h : \mathbb{R} \longrightarrow \mathbb{R}$, $h(x) = 5x - 1$;

(г) $j : \mathbb{R} \longrightarrow \mathbb{R}$, $j(x) = \begin{cases} 2x - 3 & \text{если } x \geqslant 1, \\ x + 1 & \text{если } x < 1; \end{cases}$

(д) $k : \mathbb{R} \longrightarrow \mathbb{R}$, $k(x) = x + |x|$;

(е) $l : \mathbb{R} \longrightarrow \mathbb{R}$, $l(x) = 2x - |x|$.

Назовите их множество значений и скажите, какие из них инъективны, а какие сюръективны ($|x|$ здесь обозначает модуль числа x, совпадающий с x при $x \geqslant 0$ и равный $-x$ при $x < 0$).

5.8. Функция, называемая *целой частью числа*, сопоставляет вещественному числу x наибольшее целое число, не превосходящее x, и обозначается так: $\lfloor x \rfloor$.

(а) Пусть $A = \{-1, 0, 1, 2\}$ и функция $f : A \longrightarrow \mathbb{Z}$ определяется условием: $f(x) = \left\lfloor \frac{x^2+1}{3} \right\rfloor$. Найдите множество значений f.

(б) Определите, является ли функция $g : \mathbb{Z} \longrightarrow \mathbb{Z}$, заданная формулой

$$g(n) = \left\lfloor \frac{n}{2} \right\rfloor$$

инъективной, сюръективной или биективной.

5.9. Функция $f : A \longrightarrow B$ задана формулой: $f(x) = 1 + \frac{2}{x}$, где A обозначает множество вещественных чисел, отличных от 0, а B — множество вещественных чисел без 1. Покажите, что f биективна и найдите обратную к ней функцию.

5.10. Функции $f : \mathbb{R} \longrightarrow \mathbb{R}$ и $g : \mathbb{R} \longrightarrow \mathbb{R}$ заданы условием:

$$f(x) = x^2 \qquad \text{и} \qquad g(x) = \begin{cases} 2x + 1, & \text{если } x \geqslant 0, \\ -x, & \text{если } x < 0. \end{cases}$$

Выразите формулами композиции: $f \circ g$, $g \circ f$ и $g \circ g$.

5.11. Пусть $f : A \longrightarrow B$ и $g : B \longrightarrow C$ — функции. Докажите, что

(а) если f и g инъективны, то $g \circ f$ тоже инъективна;

(б) если f и g сюръективны, то $g \circ f$ тоже сюръективна;

(в) если f и g обратимые функции, то $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$.

5.12. (а) Сколько раз нужно бросить игральную кость, чтобы какое-то число на ней выпало по крайней мере дважды?

(б) Сколько раз нужно бросить две игральные кости, чтобы с гарантией можно было утверждать: сумма выпавших очков появится по крайней мере дважды?

(в) Сколько карт необходимо вытащить из стандартной колоды в 52 карты, чтобы обязательно попались хотя бы две одной масти?

(г) Сколько карт необходимо вытащить из стандартной колоды в 52 карты, чтобы обязательно попались хотя бы четыре одной масти?

5.13. Известно, что в одном селе проживает 79 семей, в каждой из которых по 2 ребенка.

(а) Покажите, что найдется по крайней мере две семьи, в которых совпадают месяцы рождения обоих детей, т. е., если в первой семье дети родились в январе и мае, то и во второй — в январе и мае.

(б) Докажите, что по крайней мере у шестерых детей имена начинаются с одной и той же буквы.

5.14. Пусть $S = \{1, 2, \ldots, 20\}$.

(а) Какое наименьшее количество четных чисел необходимо взять из множества S, чтобы по крайней мере два из них в сумме давали 22?

(б) Покажите, что если взять 11 элементов из множества S, то по крайней мере одно из выбранных чисел будет делить какое-то из оставшихся в выборке.
(Указание: используйте функцию f, которая сопоставляет каждому целому числу его наибольший нечетный делитель. Например, $f(12) = 3$.)

Краткое содержание главы

Обратное отношение к отношению R между множествами A и B обозначается как R^{-1}; оно является отношением между множествами B и A и состоит из пар: $R^{-1} = \{(b, a) : \ (a, b) \in R\}$.

Пусть R — отношение между множествами A и B, и S — отношение между множеством B и третьим множеством C. **Композицией** отношений R и S называется отношение между A и C, которое определяется условием:

$$S \circ R = \{(a, c) : \ a \in A, \ c \in C \text{ и } a\,R\,b, \ b\,S\,c \text{ для некоторого } b \in B\}.$$

Пусть M и N — логические матрицы отношений R и S соответственно. **Логическим** или **булевым произведением матриц** MN называется логическая матрица композиции $S \circ R$.

Функцией, определенной на множестве A со значениями в B, называется отношение f между A и B, при котором каждому элементу множества A ставится в соответствие единственный элемент из B.

Запись $f : A \longrightarrow B$ обозначает функцию из множества A в множество B. Множество A при этом называют областью определения f, а B — областью значений функции f. Мы пишем $y = f(x)$, чтобы подчеркнуть, что $y \in B$ — **значение функции** f, принимаемое на аргументе x. Тот же y еще называют **образом** x при отображении f.

Множеством значений функции f называют подмножество в B: $f(A) = \{f(x) : \ x \in A\}$ (не путайте с *областью* значений).

Функция $f : A \longrightarrow B$ называется **инъективной** (или **взаимно однозначной**), если $f(a_1) = f(a_2) \Rightarrow a_1 = a_2$ для всех $a_1, a_2 \in A$.

Функция $f : A \longrightarrow B$ называется **сюръективной**, если ее множество значений совпадает с областью значений. Иначе говоря, если для каждого $b \in B$ найдется такой $a \in A$, что $f(a) = b$.

Функцию, которая как инъективна, так и сюръективна, называют **биекцией** или **биективной**.

Если обратное отношение к функции f снова функция, то мы называем f **обратимой**. Функция $f : A \longrightarrow B$ обратима тогда и только

тогда, когда она биективна. Обратную функцию к f мы обозначаем символом $f^{-1} : B \longrightarrow A$. Если $f(a) = b$, то $f^{-1}(b) = a$.

Принцип Дирихле утверждает, что если $f : A \longrightarrow B$ — функция, отображающая конечное множество A в конечное множество B, причем $|A| > |B|$, то по крайней мере одно из значений f встретится более одного раза. Если $|A| > k|B|$ для некоторого натурального k, то одно из значений функции f повторится по крайней мере $k + 1$ раз.

Приложение. Языки функционального программирования

Язык функционального программирования оперирует символами, используя основные примитивные функции. Такие языки успешно применяются для создания экспертных систем, моделирования общесмысловых рассуждений, построения естественных языковых интерфейсов и поддерживают исследования в области компьютерной речи и изображений.

Сила этих языков заключается в их способности строить сложные процедуры из простых, комбинируя основные примитивные функции. В результате, созданные алгоритмы производят сложные вычисления, используя доступные и весьма примитивные основные функции.

Здесь мы опишем некоторый функциональный алгоритм, который оперирует с текстом так, как это может происходить в простом текстовом редакторе.

Пусть $\mathbf{C} = \{$«а», «б», «в», ..., «я»$\}$ — множество **литер** нижнего регистра клавиатуры компьютера с кириллицей, а $\mathbf{P}$ обозначает множество $\{0, 1, 2, \ldots\}$. Обозначим через $\mathbf{S}$ множество **строк** (последовательностей) этих литер. Например, «мышь» — элемент множества $\mathbf{S}$, как и пустая строка « ».

Допустим, что мы можем использовать следующие основные примитивные функции:

CHAR : $\mathbf{S} \longrightarrow \mathbf{C}$, где CHAR($s$) — первая буква непустой строки s.

REST : $\mathbf{S} \longrightarrow \mathbf{S}$, где REST($s$) — строка, полученная из непустой строки s удалением ее первой литеры.

ADDCHAR : $\mathbf{C} \times \mathbf{S} \longrightarrow \mathbf{S}$, где ADDCHAR($c$, s) — строка, полученная из s добавлением к ее началу литеры c.

LEN : $\mathbf{S} \longrightarrow \mathbf{P}$, где LEN($s$) — число литер в строке s.

Поскольку это наши специфические базисные функции, нас не должно волновать, как они устроены на более низком уровне. Можно считать, что доступ к ним осуществляется практически так же, как и к обыкновенным функциям на калькуляторе — простым нажатием кнопки.

Задача 1. Вычислите

$\textsc{char}(s)$,

$\textsc{len}(\textsc{rest}(s))$ и

$\textsc{addchar}\Big(\textsc{char}(s),\ \textsc{addchar}\big(\text{«л»},\ \textsc{rest}(s)\big)\Big)$,

если $s = \text{«сон»}$.

Решение.

$$\textsc{char}(s) = \textsc{char}(\text{«сон»}) = \text{«с»};$$

$$\textsc{len}(\textsc{rest}(s)) = \textsc{len}(\textsc{rest}(\text{«сон»})) = \textsc{len}(\text{«он»}) = 2;$$

$$\begin{aligned}\textsc{addchar}\Big(\textsc{char}(s),\ \textsc{addchar}\big(\text{«л»},\ \textsc{rest}(s)\big)\Big) &= \\ = \textsc{addchar}\Big(\text{«с»},\ \textsc{addchar}\big(\text{«л»},\ \textsc{rest}(\text{«сон»})\big)\Big) &= \\ = \textsc{addchar}\big(\text{«с»},\ \textsc{addchar}\big(\text{«л»},\ \text{«он»}\big)\big) &= \\ = \textsc{addchar}(\text{«с»},\ \text{«лон»}) = \text{«слон»}.&\end{aligned}$$

Задача 2. Опишите на словах действие функции

$$\textsc{addchar}(\textsc{char}(s),\ \textsc{addchar}(c, \textsc{rest}(s))),$$

где s — произвольная непустая строка, а c — любая литера.

Решение. Как мы видели при решении предыдущей задачи, эта функция вставляет новую литеру «c» непосредственно после первой литеры строки s.

Задача 3. Функция $\textsc{third} : \mathbf{S} \longrightarrow \mathbf{C}$ нужна для определения третьей по счету литеры в строке из трех и более литер. Выразите функцию third через char и rest.

Решение. Третья литера произвольной строки s необходимой длины совпадает с первым символом строки, полученной из s удалением первых двух. Поэтому

$$\textsc{third}(s) = \textsc{char}(\textsc{rest}(\textsc{rest}(s))).$$

Задача 4. Опираясь на базисные примитивные функции, найдите функцию REVERSE2 : **S** $\longrightarrow$ **S**, которая переставляет первые две литеры в строке длины 2 и более.

Решение. Пусть s — вводимая строка. Первая литера выводимой строки равна CHAR(REST(s)), а вторая — CHAR(s). Остальные литеры остаются неизменными и совпадают с REST(REST(s)). Следовательно, значение функции REVERSE2(s) выражается следующим образом:

ADDCHAR(CHAR(REST(s)), ADDCHAR(CHAR(s), REST(REST(s)))).

Задача 5. Проследите следующий алгоритм, взяв в качестве вводной строки s = «клоп».

Input s
begin
  $u :=$ « »;
  $t := s$;
  $i := 0$;
  while $i <$ LEN(s) **do**
    $c :=$ CHAR(t);
    $t :=$ REST(t);
    $u :=$ ADDCHAR(c, u);
    $i := i + 1$;
end
Output u

Что делает этот алгоритм?

Решение. Проследим за изменением значений переменных c, t, u и i в течение работы цикла **while** и сведем полученную информацию в табл. 5.1

Таблица 5.1

Проход цикла	c	t	u	i	$i < 4$?
0	—	«клоп»	« »	0	Да
1	«к»	«лоп»	«к»	1	Да
2	«л»	«оп»	«лк»	2	Да
3	«о»	«п»	«олк»	3	Да
4	«п»	« »	«полк»	3	Нет

Алгоритм переставляет литеры строки в обратном порядке.

Как Вы могли заметить, некоторые из наших функций определены не для всех вводимых строк. Например, REST не определена на строке $s = \text{« »}$, а REVERSE2 не определена на строках длины меньше 2. Причина заключается в том, что желательно ограничиться малым числом стандартных множества, из которых берутся входные данные. Поэтому используются так называемые *частично вычислимые функции*. У них стандартные входные и выходные данные, но ограничены области определения и значений.

Например, частично вычислимая функция REST по существу определяется так:

$\text{REST} : \mathbf{S} \longrightarrow \mathbf{S}$,

область определения: $s \in \mathbf{S}$ и $s \neq \text{« »}$,

где $\text{REST}(s)$ — строка, полученная из s, удалением ее первой литеры.

Работая с композицией частично вычислимых функций, необходимо быть очень внимательным. Поскольку, например, функция $\text{REST}(\text{REST}(s))$ не определена на строках длины 1 или меньше, то область определения композиции $\text{REST} \circ \text{REST}$ — $s \in \mathbf{S}$ и $\text{LEN}(s) > 1$.

ГЛАВА 6

КОМБИНАТОРИКА

Комбинаторика представляет собой область математики, занимающейся подсчётом элементов конечных множеств. На простейший, казалось бы, вопрос о мощности множества часто очень трудно дать ответ. Мы уже решали такого сорта задачи, используя формулу включений и исключений (глава 3) и принцип Дирихле (глава 5). В этой же главе мы обратимся к другим задачам пересчета, чьи решения получаются с помощью двух новых принципов: правил суммы и произведения.

Общие задачи пересчета связаны с выборкой некоторого числа элементов из заданного базисного множества. Такие задачи полезно делить на типы в зависимости от того, как выбираются элементы: с повторением или без повторений, с учетом порядка выбора или без оного. Мы выведем формулы для каждого из перечисленных типов задач. В последнем параграфе этой главы мы познакомимся с биномом Ньютона и установим связь между его коэффициентами и одной из формул подсчета, полученных ранее. Эта связь может быть обобщена на коэффициенты, получающиеся при раскрытии скобок в выражении:

$$(x_1 + x_2 + \cdots + x_k)^n.$$

Они совпадают с числом выборок элементов из множества, некоторые из которых могут повторяться.

После упражнений и краткого содержания главы, мы займемся *эффективностью алгоритмов*. Это еще одно приложение функций к проблемам информатики, в котором используются также и некоторые формулы комбинаторики.

6.1. Правила суммы и произведения

Начнем этот параграф с формулировки ряда простых задач.

Задача 1. В небольшой кондитерской к концу рабочего дня осталось несколько пирожных: четыре ванильных, два шоколадных и три фруктовых. Один покупатель собирается купить пирожные перед закрытием кондитерской. Сколько пирожных может выбрать покупатель?

Задача 2. Необходимо выбрать смешанную команду, которая будет представлять местный теннисный клуб на соревнованиях. В спортивном клубе состоят 6 женщин и 9 мужчин. Сколько различных пар можно выбрать для участия в соревнованиях?

Задача 3. Сколько трехзначных чисел начинаются с 3 или 4?

Первая задача решается простым подсчетом. Поскольку все пирожные различны, мы просто можем сложить их количества. Это дает $4+2+3=9$ пирожных, из которых покупатель может сделать выбор.

Во второй задаче у нас есть 6 женщин, из которых мы можем выбрать представительницу клуба, и для каждой из них мы можем подобрать партнера среди девяти мужчин. Таким образом, общее число различных пар, которые мы можем составить, равно $6 \cdot 9 = 54$.

Эти задачи иллюстрируют два фундаментальных правила пересчета.

Правило суммы гласит, что если A и B — несвязанные события, и существует n_1 возможных исходов события A, и n_2 возможных исходов события B, то возможное число исходов события «A или B» равно сумме $n_1 + n_2$.

Правило произведения утверждает, что если дана последовательность k событий с n_1 возможными исходами первого, n_2 — второго, и т. д., вплоть до n_k возможных исходов последнего, то общее число исходов последовательности k событий равно произведению $n_1 \cdot n_2 \cdots n_k$.

Правило суммы, по существу, — частный случай формулы включений и исключений. Действительно, если рассматривать A и B как множества исходов, то $|A| = n_1$, $|B| = n_2$; а поскольку события A и B не связаны друг с другом, то можно считать, что соответствующие множества не пересекаются. Тогда, по формуле включений и исключений, $|A \cup B| = |A| + |B|$, т. е. множество $A \cup B$ содержит $n_1 + n_2$ элементов. Это означает, что существует $n_1 + n_2$ возможных исхода события «A или B».

Правило произведения тоже можно сформулировать на языке теории множеств. Пусть A_1 обозначает множество n_1 исходов первого события, A_2 — множество n_2 исходов второго, и т. д. Тогда любую последовательность k событий можно рассматривать как элемент декартова произведения $A_1 \times A_2 \times \cdots \times A_k$, чья мощность равна $|A_1| \cdot |A_2| \cdots |A_k|$.

Теперь мы готовы решить третью из сформулированных задач. При этом мы будем использовать как правило суммы, так и произведения. Трехзначные числа, о которых идет речь в задаче, есте-

ственным образом разбиваются на два непересекающихся класса. К одному из них относятся числа, начинающиеся с 3, а ко второму — с 4. Для подсчета чисел в первом классе заметим, что существует один возможный исход для первой цифры (она должна быть равна 3), 10 исходов для второй и 10 исходов для последней цифры. По правилу произведения получаем, что всего чисел в первом классе насчитывается ровно $1 \cdot 10 \cdot 10 = 100$. Аналогично можно подсчитать количество чисел во втором классе. Оно тоже равно 100. Наконец, по правилу суммы получаем, что существует $100 + 100 = 200$ трехзначных чисел, начинающихся с 3 или 4.

Пример 6.1. Я хочу взять с собой для ланча два фрукта. У меня есть три банана, четыре яблока и две груши. Сколькими способами я могу выбрать два фрукта разного вида из имеющихся у меня?

Решение. Если я собираюсь взять один из трех бананов и одно из четырех яблок, то сделать я это могу $3 \cdot 4 = 12$ различными способами[1]. Банан и грушу я могу взять $3 \cdot 2 = 6$ возможными способами. Наконец, грушу и яблоко можно выбрать $4 \cdot 2 = 8$ различными способами. Поскольку все три множества возможностей различны, то всего количество способов, которыми можно выбрать два фрукта, равно $12 + 6 + 8 = 26$.

Пример 6.2. Государственный регистрационный знак легкового автомобиля состоит из трех цифр и трех букв русского алфавита (не считая кода города). Будем считать, что в номере можно задействовать любую последовательность букв и цифр. Сколько различный автомобильных номеров может выдать ГИБДД?

Решение. Каждую из трех букв номера можно выбрать из 33 букв алфавита. По правилу произведения число различных последовательностей из трех букв равно $33 \cdot 33 \cdot 33 = 35\,937$. Аналогично, число последовательностей трех цифр равно $10 \cdot 10 \cdot 10 = 1\,000$. Наконец, поскольку каждый из автомобильных номеров состоит из трех букв и трех цифр, правило произведения дает искомый ответ: 35 937 000 различных автомобильных номеров может выдать ГИБДД.

[1] В условии задачи опущена существенная деталь: два фрукта одного наименования считаются непохожими один на другой. Если бы мы не могли отличить один банан от другого и одну грушу от другой, ответ был бы иным. — *Прим. перев.*

6.2. Комбинаторные формулы

Допустим, что ребенку предложили мешок с конфетами трех наименований: «Мишка на севере» (A), «А ну-ка отними» (B) и «Золотой петушок» (C). Сколькими способами ребенок может выбрать две конфеты из мешка?

На этот вопрос можно дать несколько ответов в зависимости от уточнения его формулировки. Ставя задачу, мы не уточнили, можно ли брать конфеты одного наименования или нет. Например, можно ли взять две конфеты «Мишка на севере», т. е. AA? Кроме того, имеет ли значение порядок выбора? Иными словами, отличается ли выбор AB от BA или нет?

Таким образом мы имеем четыре разных уточнения формулировки.

1. Повторения разрешены и порядок выбора существенен. В этом случае у нас есть 9 возможностей: AA, AB, AC, BA, BB, BC, CA, CB и CC.

2. Запрещено брать конфеты одного наименования, но порядок существенен. В этой ситуации — 6 случаев: AB, AC, BA, BC, CA и CB.

3. Повторения разрешены, но порядок выбора не имеет значения. Тогда ответ — тоже 6 возможностей: AA, AB, AC, BB, BC и CC.

4. И, наконец, если нельзя брать одинаковые конфеты, а порядок не имеет значения, то у ребенка есть только три варианта выбора: AB, AC и BC.

При решении конкретных задач на подсчет количества способов необходимо четко понимать, о каком типе уточнения формулировки идет речь. Чтобы различать на терминологическом уровне тип конкретной задачи, введем несколько определений.

Начнем с вспомогательных терминов. Предположим, что мы берем элементы x_1, x_2, …, x_k из множества X мощности k. Каждый такой набор принято называть *выборкой* объема k из n элементов или, иначе, (n, k)-выборкой. Выборка называется *упорядоченной*, если порядок следования элементов в ней задан. При этом две упорядоченные выборки, различающиеся лишь порядком следования элементов, считаются разными. Если же порядок следования элементов в выборке не имеет значения, то выборка называется *неупорядоченной*. Теперь введем основные термины в соответствии с типом уточнений, приведенных выше.

- (n, k)-*размещением с повторениями* называется упорядоченная (n, k)-выборка, элементы в которой могут повторяться;
- (n, k)-*размещением без повторений* называется упорядоченная (n, k)-выборка, элементам в которой повторяться запрещено;
- неупорядоченная (n, k)-выборка с повторяющимися элементами называется (n, k)-*сочетанием с повторениями*;
- неупорядоченная (n, k)-выборка без повторяющихся элементов называется (n, k)-*сочетанием без повторений*.

Попробуем подсчитать количество всех различных (n, k)-размещений с повторениями. На первое место выборки мы можем поставить любой из n элементов множества. Поскольку повторения разрешены, то на второе место мы опять можем поставить любой элемент из этого же множества, и т. д. Поскольку у нас k мест в выборке, то опираясь на правило произведения, получаем, что число всех (n, k)-размещений с повторениями равно n^k.

Пример 6.3. Целые числа в компьютере представляются строчкой из N двоичных знаков. Первый из них отведен на знак ($+$ или $-$), а остальные $N - 1$ отвечают за модуль целого числа. Сколько различных целых чисел может использовать компьютер?

Решение. Двоичная цифра — это 0 или 1. Для записи числа используется N таких цифр. Заметим, что двоичные строки, представляющие числа, могут иметь повторяющиеся цифры, и порядок их следования, естественно, существенен для данной задачи. Поэтому мы имеем дело с $(2, N)$-размещениями с повторениями. По выведенной формуле получаем, что общее количество таких строк равно 2^N. Практически всегда различные размещения изображают различные числа, за исключением двух строк:

$$-000000\cdots00 \qquad \text{и} \qquad +000000\cdots00,$$

которые изображают 0. Стало быть, компьютер может оперировать $(2^N - 1)$ целыми числами.

Для числа всех (n, k)-размещений без повторений зафиксировано специальное обозначение[1]: $P(n, k)$. Подсчитаем это число. На первое место выборки мы можем поставить любой из n элементов. Поскольку здесь нам не разрешены повторения, то для второго места

[1] По старой русской традиции это число обозначается символом A_n^k. — *Прим. перев.*

мы можем выбрать любой из $(n-1)$ оставшихся элементов. На третье — из $(n-2)$ и так далее, вплоть до k-го места, куда можно написать любой из $(n-k+1)$ элементов. Теперь для окончательного ответа нам нужно применить правило произведения. Имеем

$$P(n,\,k) = n(n-1)(n-2)\cdots(n-k+1).$$

Для сокращения записи напомним, что произведение всех натуральных чисел от 1 до n включительно называется n факториал и обозначается символом $n!$. Попробуем с помощью этого символа выразить $P(n,\,k)$, для чего проделаем легкие, хоть и не очевидные преобразования.

$$\begin{aligned} P(n,\,k) &= n(n-1)(n-2)\cdots(n-k+1) = \\ &= n(n-1)(n-2)\cdots(n-k+1)\frac{(n-k)(n-k-1)\cdots 2\cdot 1}{(n-k)(n-k-1)\cdots 2\cdot 1} = \\ &= \frac{n(n-1)(n-2)\cdots(n-k+1)(n-k)(n-k-1)\cdots 2\cdot 1}{(n-k)(n-k-1)\cdots 2\cdot 1} = \\ &= \frac{n!}{(n-k)!}. \end{aligned}$$

Итак, число различных $(n,\,k)$-размещений без повторений равно

$$P(n,\,k) = \frac{n!}{(n-k)!}.$$

Пример 6.4. Сколько различных четырехбуквенных «слов» можно написать, используя буквы: *a, c, n, o* и *e*, если под «словом» мы будем понимать любую последовательность неповторяющихся букв, даже если эта последовательность не несет в себе никакого смысла.

Решение. Как договорились, под «словом» мы понимаем любую последовательность четырех разных букв, которые можно выбрать из шести данных. Значит мы имеем дело с подсчетом числа размещений без повторений $P(6,\,4)$. Следовательно,

$$P(6,\,4) = \frac{6!}{(6-4)!} = \frac{6!}{2!} = \frac{6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1}{2\cdot 1}.$$

После сокращения получаем окончательный ответ:

$$P(6,\,4) = \frac{6\cdot 5\cdot 4\cdot 3\cdot \cancel{2}\cdot \cancel{1}}{\cancel{2}\cdot \cancel{1}} = 6\cdot 5\cdot 4\cdot 3 = 360.$$

Теперь займемся сочетаниями без повторений, т. е. выборками, в которых порядок не существенен и повторы запрещены. Число всех (n, k)-сочетаний без повторений обозначается символом[2] $C(n, k)$. Найдем его.

Мы воспользуемся уже известным нам фактом: число всех (n, k)-размещений без повторений равно $P(n, k) = \frac{n!}{(n-k)!}$. Поскольку размещение без повторений отличается от сочетания без повторений наличием порядка, то число $P(n, k)$, естественно больше, чем $C(n, k)$. Если мы поймем соотношение между ними, то получим ответ.

Проведем эксперимент. Пусть $n = 4$, а $k = 3$. Зафиксируем множество $A = \{1, 2, 3, 4\}$, откуда мы будем брать элементы. Каждое $(4, 3)$-сочетание без повторений — это выбор последовательности трех разных цифр из четырех данных, причем порядок, в котором будут идти выбранные цифры, значения не имеет. Например, подмножество $\{1, 2, 3\}$ является $(4, 3)$-сочетанием без повторений. Перемешав цифры в выбранном подмножестве $\{2, 1, 3\}$, мы получим то же самое сочетание (порядок не важен), но совершенно другое размещение (порядок существенен). Так сколько же различных размещений можно получить из одного сочетания? Вот основной вопрос, ответ на который приводит к победе. В данном конкретном случае ($n = 4$, $k = 3$) ответ легко получить, перечислив вручную все варианты. Нам же надо разобраться с общим случаем. Сформулируем его более четко.

Дано (n, k)-сочетание без повторений, т. е. выбрано подмножество $B \subset A$, где $|B| = k$ и $|A| = n$. Сколько из него можно получить разных (n, k)-размещений без повторений?

Фактически, нам нужно подсчитать количество (k, k)-размещений без повторений! (Подумайте, почему.) Но это число мы знаем[3]:

$$P(k, k) = \frac{k!}{(k-k)!} = k!.$$

Таким образом, на каждое (n, k)-сочетание без повторений приходится $k!$ различных (n, k)-размещений без повторений. Стало быть,

$$C(n, k) = \frac{P(n, k)}{k!} = \frac{n!}{(n-k)!\,k!}.$$

[2]Раньше в России это число было принято обозначать C_n^k, а теперь у нас, как и практически всюду в мире, его обозначают так: $\binom{n}{k}$. — *Прим. перев.*

[3]Заметим, что $0! = 1$. — *Прим. перев.*

Пример 6.5. Меню в китайском ресторане дает Вам возможность выбрать ровно три из семи главных блюд. Сколькими способами Вы можете сделать заказ?

Решение. Здесь мы имеем дело с (7, 3)-сочетаниями без повторений. Поэтому ответ получить легко:

$$C(7,\,3) = \frac{7!}{(7-3)!\,3!} = \frac{7!}{4!\,3!} = \frac{7\cdot 6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1}{(4\cdot 3\cdot 2\cdot 1)(3\cdot 2\cdot 1)} =$$
$$= \frac{7\cdot 6\cdot 5\cdot \cancel{4}\cdot \cancel{3}\cdot \cancel{2}\cdot \cancel{1}}{(\cancel{4}\cdot \cancel{3}\cdot \cancel{2}\cdot \cancel{1})(3\cdot 2\cdot 1)} = \frac{7\cdot \cancel{6}\cdot 5}{\cancel{3}\cdot \cancel{2}\cdot \cancel{1}} = 35.$$

Итак, у Вас есть 35 возможностей для различных заказов.

Последнее, что мы исследуем, это сочетания с повторениями. Напомним, что это выборки, в которых порядок не важен, а вот повторы элементов допускаются. Поскольку порядок в наших выборках значения не имеет, а повторы разрешены, мы можем сгруппировать вместе одинаковые элементы, разделив группы какими-нибудь метками.

Предположим, например, что мы сделали выборку, состоящую из пяти букв, каждая из которых может быть одной из *а*, *б* и *в*. Выборку, состоящую из двух «а», одной «б» и двух «в», можно записать как *аа*|*б*|*вв*, а выборка из одной буквы «а» и четырех букв «в» будет выглядеть так: *а*||*вввв*. Договоримся, что слева от первой метки либо стоят буквы «а», либо ничего, справа от второй метки — либо «в», либо ничего, а буквы «б», если они присутствуют в выборке, стоят между метками. Таким образом, можно считать, что мы всегда смотрим на *семь* ячеек (пять букв и две метки), причем различные выборки будут отличаться ячейками, в которых стоят метки. Значит, число всех таких сочетаний с повторениями совпадает с количеством способов, которыми мы можем поместить две метки в семь ячеек. Осталось понять, что это количество есть не что иное, как число всех (7, 2)-сочетаний без повторений, т. е. равно $C(7,\,2)$. Действительно, первую метку можно поставить в любую из семи ячеек, а вторую — в любую из шести, поскольку одна ячейка уже занята. Это дает нам $7\cdot 6$ возможностей. Заметим теперь, что поменяв расставленные метки местами, мы получим то же самое заполнение ячеек. Стало быть, $7\cdot 6$ нужно разделить на 2. Итак,

количество способов равно

$$\frac{7\cdot 6}{2} = \frac{7\cdot 6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1}{(2\cdot 1)(5\cdot 4\cdot 3\cdot 2\cdot 1)} = \\ = \frac{7!}{5!\cdot 2!} = \frac{7!}{(7-2)!\cdot 2!} = C(7,\,2).$$

Возвращаясь к общему случаю $(n,\,k)$-сочетаний без повторений (k объектов из n данных), заметим, что нам потребуется $n-1$ метка и k объектов. Таким образом, у нас будет $(n-1)+k$ ячеек для заполнения. Значит, число $(n,\,k)$-сочетаний без повторений совпадает с количеством способов размещения $(n-1)$ метки в $(n+k-1)$ ячейку. Итак, общее число $(n,\,k)$-сочетаний без повторений равно

$$C(n+k-1,\,n-1) = \frac{(n+k-1)!}{(n+k-1-(n-1))!\,(n-1)!} = \frac{(n+k-1)!}{k!\,(n-1)!}.$$

Пример 6.6. Сколько различных вариантов можно получить, бросая пять игральных костей?

Решение. На каждой из костей может выпасть от одного до шести очков, т. е. каждая кость дает шесть вариантов. Если бросили пять костей, то каждый вариант можно рассматривать как неупорядоченный набор пяти объектов (для каждого из которых есть 6 возможностей) с повторениями, т. е. (6, 5)-сочетание с повторениями. Согласно общей формуле, общее число исходов равно

$$C(6+5-1,\,6-1) = C(10,\,5) = \frac{10!}{5!\,5!} = 252.$$

В табл. 6.1 собраны вместе все формулы для подсчета количества выборок k элементов из n-элементного множества, которые мы вывели в этом параграфе.

Таблица 6.1

	Порядок существенен		Порядок не существенен	
Элементы повторяются	размещения с повторениями	n^k	сочетания с повторениями	$\frac{(n+k-1)!}{k!\,(n-1)!}$
Элементы не повторяются	размещения без повторений	$\frac{n!}{(n-k)!}$	сочетания без повторений	$\frac{n!}{(n-k)!\,k!}$

Разберем еще несколько примеров, которые наглядно показывают, что нужно быть очень внимательным при выборе комбинаторной формулы для решения конкретной задачи.

В Национальной Английской Лотерее[4] дважды в неделю случайным образом выбирается шесть разных номеров из первых 49 натуральных чисел. Любой, кто угадает все шесть выпавших номеров, выиграет главный приз, который может достигать миллиона фунтов стерлингов.

Подсчитаем шансы выигрыша главного приза. Шестерка выигрышных номеров — это неупорядоченная выборка шести чисел из 49 возможных. Поскольку общее количество (49, 6)-сочетаний без повторений равно

$$\frac{49!}{(49-6)!\,6!} = \frac{49!}{43!\,6!} = 13\,983\,816,$$

то шансов выиграть практически нет: 1 из 13 983 816. Гораздо больше шансов за то, что в Вас попадет молния, что тоже маловероятно.

Тем, кто угадал пять, четыре или три номера, тоже присуждается приз, хотя и не такой впечатляющий. Премия также достается и тем участникам, кто угадал пять номеров, а шестой, названный ими, совпадает со специальным, случайно названным «призовым» номером.

Величина денежного приза, за одним исключением, зависит от числа проданных лотерейных билетов. Каждый билет, в котором можно выписать шесть номеров, стоит £1 (один фунт), и любой, кто угадает ровно три (и не больше) выигрышных номера, автоматически выигрывает £10. Подсчитаем же вероятность выигрыша £10.

Итак, Вы можете рассматривать свои шесть номеров как объединение двух несвязанных событий: *выборка трех правильных номеров* и *выборка трех неверных номеров*. Существует $C(6,\,3)$ возможностей назвать три из шести номеров, которые объявляются в среду или субботу членами жюри, и $C(43,\,3)$ возможности неудачного выбора. Тогда общее число комбинаций, выигрывающих £10, равно

$$C(6,\,3)\cdot C(43,\,3) = \frac{6!}{3!\,3!}\cdot\frac{43!}{40!\,3!} = 246\,820.$$

Вероятность выигрыша — это доля удачно заполненных карточек ко всем возможным заполнениям, т. е.

$$\frac{246\,820}{13\,983\,816} \approx \frac{1}{57} \approx 0,018.$$

[4] Она похожа на наше Спортлото. — *Прим. перев.*

Пример 6.7. Двенадцать человек, включая Мари и Петера, являются кандидатами в комитет пяти. Сколько разных комитетов можно набрать из 12 кандидатов? Сколько из них

(а) включают как Мари, так и Петера;

(б) не включают ни Мари, ни Петера;

(в) включают либо Мари, либо Петера, но не обоих?

Решение. Существует

$$C(12,\,5) = \frac{12!}{7!\,5!} = 792$$

возможных комитета.

(а) Если Мари и Петер уже выбраны в комитет, нам остается отобрать в него только трех членов из оставшихся десяти кандидатов. Это можно сделать

$$C(10,\,3) = \frac{10!}{7!\,3!} = 120$$

способами. Значит, Мари и Петер могут быть членами 120 разных комитетов.

(б) Если Мари и Петер не участвуют в комитете, то мы выбираем всех его членов из десяти кандидатов. Поэтому у нас есть

$$C(10,\,5) = \frac{10!}{5!\,5!} = 252$$

возможности для разных комитетов, не включающих ни Мари, ни Петера.

(в) Один из способов дать ответ на этот вопрос заключается в подсчете комитетов, включающих Мари, но без Петера. Их ровно $C(10,\,4)$. То же число комитетов включают Петера, но без Мари. Значит, $2\cdot C(10,\,4)$ комитета имеют в качестве члена либо Мари, либо Петера, но не обоих сразу.

Альтернативный подход к решению основан на том, что каждый из 792 возможных составов комитета можно отнести в точности к одной из категорий: (а), (б) или (в). Значит, число комитетов, относящихся к последней, равно

$$792 - 120 - 252 = 420.$$

6.3. Бином Ньютона

Числа $C(n, k)$ возникают как коэффициенты при раскрытии скобок в биноме $(a+b)^n$. Например,

$$\begin{aligned}(a+b)^3 &= (a+b)(a+b)(a+b) = \\ &= aaa + aab + aba + abb + baa + bab + bba + bbb = \\ &= a^3 + 3a^2b + 3ab^2 + b^3.\end{aligned}$$

Каждое из восьми слагаемых, стоящих во второй строке наших преобразований, получается при умножении трех переменных, выбираемых по одной из каждой скобки. Мы видим, в частности, что ровно три слагаемых содержат одну переменную a и две b. Это происходит потому, что у нас есть $C(3, 2) = 3$ способа выбора двух скобок из трех, откуда мы возьмем переменную b (а из оставшейся берем a).

Аналогично получаются и остальные коэффициенты этого выражения: $C(3, 0) = 1$, $C(3, 1) = 3$, $C(3, 2) = 3$ и $C(3, 3) = 1$. Чтобы согласовать полученные числа с формулой для $C(n, k)$, выведенной в предыдущем параграфе, мы должны предполагать, что $0! = 1$. Иначе говоря, существует единственная возможность не сделать никакого выбора из конечного множества объектов.

В общем случае, раскрывая скобки в биноме $(a+b)^n$, мы будем получать члены вида $a^{n-k}b^k$ (где k принимает каждое из значений от 0 до n) при перемножении символов b, взятых из k скобок, и a, взятых из оставшихся $(n-k)$ скобок. Так как есть ровно $C(n, k)$ способов выбора k скобок из n, то у нас будет в точности $C(n, k)$ членов вида $a^{n-k}b^k$ при $k = 0, 1, \ldots, n$. Следовательно,

$$(a+b)^n = C(n, 0)a^n + C(n, 1)a^{n-1}b + C(n, 2)a^{n-2}b^2 + \cdots + C(n, n)b^n.$$

Эта формула называется *биномом Ньютона*. Ровно поэтому коэффициенты $C(n, k)$ часто называют *биномиальными коэффициентами*.

Биномиальные коэффициенты полезно выстроить в так называемый *треугольник Паскаля* (см. рис. 6.1).

$$\begin{array}{ccccccccccc}
 & & & & & C(0,0) & & & & & \\
 & & & & C(1,0) & & C(1,1) & & & & \\
 & & & C(2,0) & & C(2,1) & & C(2,2) & & & \\
 & & C(3,0) & & C(3,1) & & C(3,2) & & C(3,3) & & \\
 & C(4,0) & & C(4,1) & & C(4,2) & & C(4,3) & & C(4,4) & \\
C(5,0) & & C(5,1) & & C(5,2) & & C(5,3) & & C(5,4) & & C(5,5) \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
C(n,0) & C(n,1) & \ldots & \ldots & \ldots & \ldots & \ldots & C(n,n-1) & & C(n,n) &
\end{array}$$

Рисунок 6.1. Треугольник Паскаля

Каждая $(n+1)$-ая строка этого треугольника состоит из биномиальных коэффициентов, получающихся при раскрытии скобок в выражении $(a+b)^n$.

Вычислив несколько первых коэффициентов треугольника Паскаля, мы получим

					1					
				1		1				
			1		2		1			
		1		3		3		1		
	1		4		6		4		1	
1		5		10		10		5		1
...		...		...		...		...		...

Так как $C(n,\,0) = C(n,\,n) = 1$, на внешних сторонах треугольника Паскаля всегда стоят единицы. Симметрия относительно вертикальной высоты треугольника следует из тождества:

$$C(n,\,k) = C(n,\,n-k),$$

которое легко доказывается с помощью формулы для $C(n,\,k)$.

Есть и другие закономерности, которые бросаются в глаза при взгляде на треугольник Паскаля. Например, сложив два последовательных числа, стоящих в строке треугольника, мы получим число из следующей строки, которое стоит между двумя сложенными. Это свойство известно как *формула Паскаля*:

$$C(n-1,\,k-1) + C(n-1,\,k) = C(n,\,k),$$

справедливая при $0 < k < n$.

Доказательство формулы состоит в последовательности преобразований:

$$\begin{aligned} C(n-1,\,k-1) + C(n-1,\,k) &= \frac{(n-1)!}{(n-k)!\,(k-1)!} + \frac{(n-1)!}{(n-k-1)!\,k!} = \\ &= \frac{(n-1)!}{(n-k-1)!\,(k-1)!}\left(\frac{1}{n-k} + \frac{1}{k}\right) = \\ &= \frac{(n-1)!}{(n-k-1)!\,(k-1)!}\left(\frac{n}{(n-k)k}\right) = \\ &= \frac{n!}{(n-k)!\,k!} = C(n,\,k). \end{aligned}$$

Наше знакомство с комбинаторикой не будет полным, если мы не рассмотрим задачу о количестве перестановок букв в слове «КОЛОБОК». Оно состоит из семи букв, которые можно переставить

7! способами. Однако в нем есть три буквы «О» и две буквы «К». Поэтому меняя местами буквы «О» или переставляя буквы «К», мы не получим новых «слов». Так как количество перестановок трех элементов равно 3!, а двух — 2!, то мы можем получить всего

$$\frac{7!}{3!\,2!} = 420$$

разных «слов» из слова «КОЛОБОК».

В общей ситуации справедлива следующая теорема о перестановках.

Теорема. Существует

$$\frac{n!}{n_1!\,n_2!\cdots n_r!}$$

различных перестановок n объектов, n_1 из которых относятся к типу 1, n_2 — к типу 2, и т. д. вплоть до n_r объектов типа r.

Пример 6.8. Сколькими способами можно распределить 15 студентов по трем учебным группам по пять студентов в каждой?

Решение. У нас есть 15 объектов, которые нужно организовать в три группы по пять. Это можно сделать

$$\frac{15!}{5!\,5!\,5!} = 68\,796$$

различными способами.

Коэффициенты $\frac{n!}{n_1!\,n_2!\cdots n_r!}$ носят название *мультиномиальных*. Они стоят при произведениях $x_1^{n_1}x_2^{n_2}\cdots x_r^{n_r}$ в разложении степени $(x_1 + x_2 + \cdots + x_r)^n$.

В этом легко убедиться, поскольку член вида $x_1^{n_1}x_2^{n_2}\cdots x_r^{n_r}$ получается, когда мы перемножаем переменные x_1, выбранные из n_1 скобок, x_2, выбранные из n_2 скобок, и т. д. Таким образом, коэффициент при $x_1^{n_1}x_2^{n_2}\cdots x_r^{n_r}$ в точности равен числу перестановок n объектов, n_1 из которых относятся к первому типу, n_2 — ко второму и т. д.

Пример 6.9. Найдите

(а) коэффициент при $x^3y^2z^4$ из разложения степени $(x + y + z)^9$;

(б) коэффициент при x^3y^2 из разложения степени $(x + y + 3)^7$.

Решение.

(а) Коэффициент при $x^3y^2z^4$ из разложения степени $(x+y+z)^9$ равен

$$\frac{9!}{3!\,2!\,4!} = 1\,260.$$

(б) Коэффициент при $x^3y^2z^2$ из разложения степени $(x+y+z)^7$ равен

$$\frac{7!}{3!\,2!\,2!} = 210.$$

Поэтому разложение степени $(x+y+z)^7$ содержит член $210x^3y^2z^2$. Положив $z=3$, мы увидим, что в разложении степени $(x+y+3)^7$ присутствует член $1\,890x^3y^2$. Итак, коэффициент при x^3y^2 из разложения степени $(x+y+3)^7$ равен 1 890.

Набор упражнений 6

6.1. (а) У человека есть пять пиджаков, восемь рубашек и семь галстуков. Сколько различных костюмов можно составить из этих предметов?

(б) У женщины в шкафу висит шесть платьев, пять юбок и три блузки. Сколько разных нарядов она может составить из своей одежды?

(в) В холодильнике стоит мороженое шести разных наименований. На десерт можно взять одну, две или даже три порции мороженого сразу. Сколько возможностей есть у Вас для различных десертов?

6.2. (а) Перевертыш — это многозначное число, которое не поменяет своего значения, если все его цифры записать в обратном порядке. Сколько существует шестизначных перевертышей? А сколько семизначных?

(б) Сколько четырехзначных чисел, не превосходящих 6 000, можно составить, используя только нечетные цифры?

(в) Пароль, открывающий доступ к компьютеру, состоит из шести символов. Первые два из них — строчные буквы латинского алфавита (всего 26 букв), а оставшиеся четыре могут быть как цифрами, так и строчными буквами. Сколько можно придумать различных паролей?

6.3. Пусть S — множество четырехзначных чисел, в чьей десятичной записи участвуют цифры: 0, 1, 2, 3, и 6, причем 0 на первом месте, естественно, стоять не может.

(а) Какова мощность множества S?

(б) Сколько чисел из S в своей десятичной записи не имеют повторяющихся цифр?

(в) Как много четных среди чисел пункта (б)?

(г) Сколько чисел из пункта (б) окажутся больше, чем 4 000?

6.4. Вычислите следующие величины:

(а) $P(7, 2)$, $P(8, 5)$, $P(6, 4)$ и $P(n, n-1)$;

(б) $C(10, 7)$, $C(9, 2)$, $C(8, 6)$ и $C(n, n-1)$.

Убедитесь, что $C(n, k) = C(n, n-k)$.

6.5. (а) Сколько существует возможностей для присуждения первого, второго и третьего мест семнадцати участницам соревнований по икебане?

(б) Комитет из 20 членов избирает председателя и секретаря. Сколькими способами это можно сделать?

(в) Пароль, открывающий доступ к компьютеру, составляется по правилам задачи 6.2 (в). Сколько разных паролей можно написать из неповторяющихся символов?

6.6. (а) Хоккейная команда насчитывает 18 игроков. Одиннадцать из них входят в основной состав. Подсчитайте количество возможных основных составов.

(б) Жюри из 5 женщин и 7 мужчин должно быть выбрано из списка в 8 женщин и 11 мужчин. Сколько можно выбрать различных жюри?

(в) Предстоит выбрать команду четырех игроков в гольф из пяти профессиональных игроков и пяти любителей. Сколько разных команд может состоять из трех профессионалов и одного любителя? Сколько команд состоит только из профессионалов или только из любителей?

6.7. В один из комитетов парламента нужно отобрать трех членов, причем выбирать надо из пяти консерваторов, трех лейбористов и четырех либерал-демократов.

(а) Сколько разных комитетов можно составить?

(б) Сколько разных комитетов можно составить, если в него должен входить по крайней мере один либерал-демократ?

(в) Сколько разных комитетов можно составить, если лейбористы и консерваторы не могут быть его членами одновременно?

(г) Сколько разных комитетов можно составить, если в него должен войти по крайней мере один консерватор и хотя бы один лейборист?

6.8. В небольшой фирме восемь человек работают на производстве, пятеро — в отделе сбыта, и трое — в бухгалтерии. Для обсуждения новой продукции было решено пригласить на совещание шестерых работающих. Сколькими способами это можно сделать, если

(а) необходимо пригласить по два представителя от каждого отдела;

(б) необходимо пригласить по крайней мере двоих представителей производства;

(в) необходимы представители каждого из трех отделов?

6.9. (а) Ресторан в своем меню предлагает пять различных главных блюд. Каждый из компании в шесть человек заказывает свое главное блюдо. Сколько разных заказов может получить официант?

(б) Цветочница продает розы четырех разных сортов. Сколько разных букетов можно составить из дюжины роз?

6.10. Вы покупаете пять рождественских открыток в магазине, который может предложить четыре разных типа приглянувшихся Вам открыток.

(а) Как много наборов из пяти открыток Вы можете купить?

(б) Сколько наборов можно составить, если ограничиться только двумя типами открыток из четырех, но купить все равно пять открыток?

6.11. Вот восьмая строка треугольника Паскаля:

$$1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1.$$

(а) Найдите девятую и десятую его строки.

(б) Проверьте, что если a, b и c — три последовательных числа в восьмой строке треугольника Паскаля, то одно из чисел десятой строки можно получить как сумму: $a + 2b + c$.

(в) Воспользуйтесь формулой Паскаля для доказательства равенства:

$$C(n,\, k) + 2C(n,\, k+1) + C(n,\, k+2) = C(n+2,\, k+2)$$

при $0 \leqslant k \leqslant n-2$.
(Эта формула обобщает факт (б) на весь треугольник Паскаля.)

6.12. (а) Положив в биноме Ньютона $a = b = 1$, покажите, что для любого $n = 0, 1, 2, \ldots$ справедлива формула:

$$C(n,\, 0) + C(n,\, 1) + \cdots + C(n,\, n) = 2^n.$$

Выведите отсюда, что в множестве S из n элементов содержится ровно 2^n различных подмножеств.
(Указание: определите сначала, сколько подмножеств мощности k содержится в S.)

(б) Покажите, что

$$C(n,\, 0) - C(n,\, 1) + C(n,\, 2) - \cdots + (-1)^n C(n,\, n) = 0.$$

6.13. (а) Сколько разных «слов» можно получить из слова

«АБРАКАДАБРА»?

(б) Сколько из них начинаются с буквы «К»?

(в) В скольких из них обе буквы «Б» стоят рядом?

6.14. (а) Найдите коэффициент при a^3b^5 после раскрытия скобок в выражении $(a+b)^8$.

(б) Найдите коэффициент при xy^3z^4 после раскрытия скобок в выражении $(x+y+z)^8$.

(в) Найдите коэффициент при xy^2z после раскрытия скобок в выражении $(x+2y+z-1)^5$.

Краткое содержание главы

Правило суммы гласит, что если A и B — несвязанные события, причем существует n_1 возможных исходов события A и n_2 возможных исхода события B, то возможное число исходов события «A или B» равно сумме $n_1 + n_2$.

Правило произведения утверждает, что если дана последовательность k событий с n_1 возможными исходами первого, n_2 — второго, и т. д., вплоть до n_k возможных исходов последнего, то общее число исходов последовательности k событий равно произведению $n_1 \cdot n_2 \cdots n_k$.

Мы выбираем k элементов из множества S мощности n. Если при этом порядок последовательности имеет значение, то мы получаем (n, k)-**размещение**, а в противном случае — (n, k)-**сочетание**. **Размещение с повторениями** получается в том случае, если в последовательности выбираемых элементов мы разрешаем появляться одинаковым, иначе мы имеем дело с **размещением без повторений**. Аналогично определяются **сочетания с повторениями** и **без повторений**.

Бином Ньютона — это формула:

$$(a+b)^n = C(n,\,0)a^n + C(n,\,1)a^{n-1}b + C(n,\,2)a^{n-2}b^2 + \cdots + C(n,\,n)b^n,$$

где

$$C(n,\,k) = \frac{n!}{(n-k)!\,k!}.$$

Общие количества всех (n, k)-размещений и (n, k)-сочетаний, как с повторениями, так и без оных даны в табл. 6.1.

Таблица 6.1

	Порядок существенен		Порядок не существенен	
Элементы повторяются	размещения с повторениями	n^k	сочетания с повторениями	$\dfrac{(n+k-1)!}{k!\,(n-1)!}$
Элементы не повторяются	размещения без повторений	$\dfrac{n!}{(n-k)!}$	сочетания без повторений	$\dfrac{n!}{(n-k)!\,k!}$

Теорема о перестановках утверждает, что существует

$$\frac{n!}{n_1!\,n_2!\,\cdots\,n_r!}$$

различных перестановок n объектов, n_1 из которых относятся к типу 1, n_2 — к типу 2, и т. д. вплоть до n_r объектов типа r.

Приложение. Эффективность алгоритмов

Одна из центральных задач информатики — создание и анализ «эффективности» компьютерных алгоритмов. Для успешного выполнения такого анализа нам необходимо уметь измерять затраты алгоритма в терминах времени и компьютерной памяти. Для этого, в частности, мы оцениваем время, необходимое для вычисления значения числовой функции. Один из способов оценки заключается в подсчете элементарных операций, которые производятся при вычислениях.

Например, чтобы установить, есть ли данное слово X в словаре, содержащем n слов, мы могли бы применить *последовательный поиск*. Названный алгоритм сравнивает слово X с первым словом в словаре, затем со вторым, и т. д. , пока слово X не будет найдено в словаре или, в наихудшем случае, не будет найдено. Очевидно, в наихудшем случае будет произведено n сравнений. С другой стороны, при *двоичном (дихотомическом) способе* слово X сравнивается со «средним» из словаря, а потом, учитывая лексикографическое упорядочение слов, принимается решение о том, в какой части словаря (до «среднего» слова или после него) продолжать поиск. Этот процесс повторяется в выбранной половине словаря и т. д. В наихудшем случае (слово отсутствует в словаре) двоичный поиск сделает $1 + \log_2 n$ сравнений. Как видно из табл. 6.2, двоичный поиск куда более эффективен, чем последовательный.

Таблица 6.2

n	$1 + \log_2 n$
8	4
64	7
$2^{18} = 250\,000$	19

Напомним, что $\log_2 2^k = k$.

Задача 1. На выполнения алгоритмов A, B, C, D и E требуется n, $3n^2$, $2n^2 + 4n$, n^3 и 2^n элементарных операций соответственно. Подсчитайте время, необходимое на работу алгоритмов при $n = 1$, $n = 10$, $n = 100$ и $n = 1000$, если одна элементарная операция совершается за 1 миллисекунду.

Решение. Ответы сведены в табл. 6.3.

Как видно из таблицы, существует огромная качественная разница между формулами, включающими в себя степени n (*полиномиальные функции*), и теми, в которых n выступает в качестве пока-

зателя (*экспоненциальные функции*). Полиномиальные функции отличаются друг от друга величиной старшей степени переменной n. Если функции имеют одну и ту же старшую степень n, то рабочие периоды соответствующих алгоритмов близки друг к другу (см. алгоритмы B и C).

Таблица 6.3

	A	B	C	D	E
	n	$3n^2$	$2n^2+4n$	n^3	2^n
1	1 мс	3 мс	6 мс	1 мс	2 мс
10	10 мс	300 мс	240 мс	1 с	1,024 с
100	100 мс	30 с	20,4 с	0,28 ч	$4 \cdot 10^{17}$ веков
1000	1000 мс	0,83 ч	0,56 ч	11,6 дней	10^{176} веков

Предположим, что функции $f(n)$ и $g(n)$ измеряют эффективность двух алгоритмов, их обычно называют *функциями временно́й сложности*. Будем говорить, что *порядок роста* функции $f(x)$ не больше, чем у $g(x)$, если найдется такая положительная константа C, что $|f(n)| \leqslant C|g(n)|$ для всех достаточно больших значений n. Этот факт обозначают как[1] $f(x) = O(g(n))$.

Задача 2. Покажите, что $2n^2 + 4n = O(n^2)$.

Решение. Так как $n \leqslant n^2$ при $n \geqslant 1$, мы получаем, что

$$2n^2 + 4n \leqslant 2n^2 + 4n^2 = 6n^2$$

для всех $n \in \mathbb{N}$. Следовательно, положив $C = 6$ в определении порядка роста, мы можем заключить, что $2n^2 + 4n = O(n^2)$.

Кроме того, легко заметить, что $n^2 \leqslant 2n^2 + 4n$ при $n \geqslant 1$. Другими словами, $n^2 = O(2n^2 + 4n)$. Две функции, такие как n^2 и $2n^2 + 4n$, каждая из которых имеет порядок роста другой, называются *функциями одного порядка роста*. Функции, ассоциированные с алгоритмами B и C в задаче 1 имеют один и тот же порядок роста и, как следствие, соответствующие длительности работы алгоритмов близки.

Мы можем определить некоторую иерархическую структуру на множестве функций, каждая из которых имеет бóльший порядок ро-

[1] Заметим, что $O(g(n))$ обозначает не одну, а целый класс функций, растущих не быстрее $g(n)$. Поэтому знак равенства здесь надо понимать условно, а именно, как знак «$\in$». — *Прим. перев.*

ста, чем предыдущая. Один из примеров такой иерархии имеет вид:

$$\underset{\text{константа}}{1} \qquad \underset{\text{логарифм}}{\log n} \qquad \underbrace{\underset{\text{полином}}{n\ n^2\ n^3\ \ldots\ n^k\ \ldots}} \qquad \underset{\text{экспонента}}{2^n}$$

Впоследствии мы могли бы детализировать эту иерархию, вставив $(n \log n)$ между n и n^2, $(n^2 \log n)$ между n^2 и n^3 и т. д.

В этой иерархии важно то, что двигаясь от ее левого края к правому, мы встречаем функции все большего порядка роста. Следовательно, чем правее в этом ряду стоит функция, тем быстрее растут ее значения по сравнению с ростом аргумента n. Сравнительные графики некоторых из перечисленных функций приведены на рис. 6.2.

Теперь любой функции временно́й сложности $f(n)$ мы можем сопоставить некоторую функцию $g(n)$ из описанной иерархии таким образом, что $f(n)$ будет иметь порядок роста не более чем $g(n)$, но больше, чем любая из функций иерархии, стоящая левее $g(n)$. Чтобы сделать это, мы с помощью нашей иерархии выделяем в данной функции наиболее быстро растущий член (его еще называют старшим членом) и сопоставляем нашей функции временно́й сложности соответствующую функцию в иерархии.

Рассмотрим, например, функцию $f(n) = 9n + 3n^6 + 7 \log n$. Ясно, что мультипликативные константы (числовые коэффициенты, на которые умножается то или иное выражение) не влияют на порядок роста функций. Поэтому $9n = O(n)$, $3n^6 = O(n^6)$ и $7 \log n = O(\log n)$. Поскольку n и $\log n$ появляются в иерархии раньше, чем n^6, мы получаем, что как $9n$, так и $7 \log n$ принадлежат классу $O(n^6)$. Следовательно, $f(n) = O(n^6)$, и наиболее быстро растущим членом у $f(n)$ является член $3n^6$. Таким образом, значения функции $f(n)$ растут не быстрее, чем возрастают значения n^6. Фактически, в этом примере функция $f(n)$ имеет тот же порядок роста, что и n^6.

Задача 3. Используя иерархию функций, о которой шла речь выше, определите порядок роста у следующих функций временно́й сложности:

(а) $n^4 + 2n^3 + 3$;

(б) $6n^5 + 2^n$;

(в) $5n + n^2 \log n$.

Решение.

(а) Данная функция принадлежит классу $O(n^4)$, поскольку n^4 — старший ее член.

(б) Эта функция принадлежит классу $O(2^n)$, поскольку 2^n — ее старший член.

(в) Старший член последней функции — это $n^2 \log n$. Такой функции нет в иерархии, а если бы она была, то стояла бы между n^2 и n^3. Следовательно, можно утверждать, что данная функция растет не быстрее, чем функции, лежащие в классе $O(n^3)$.

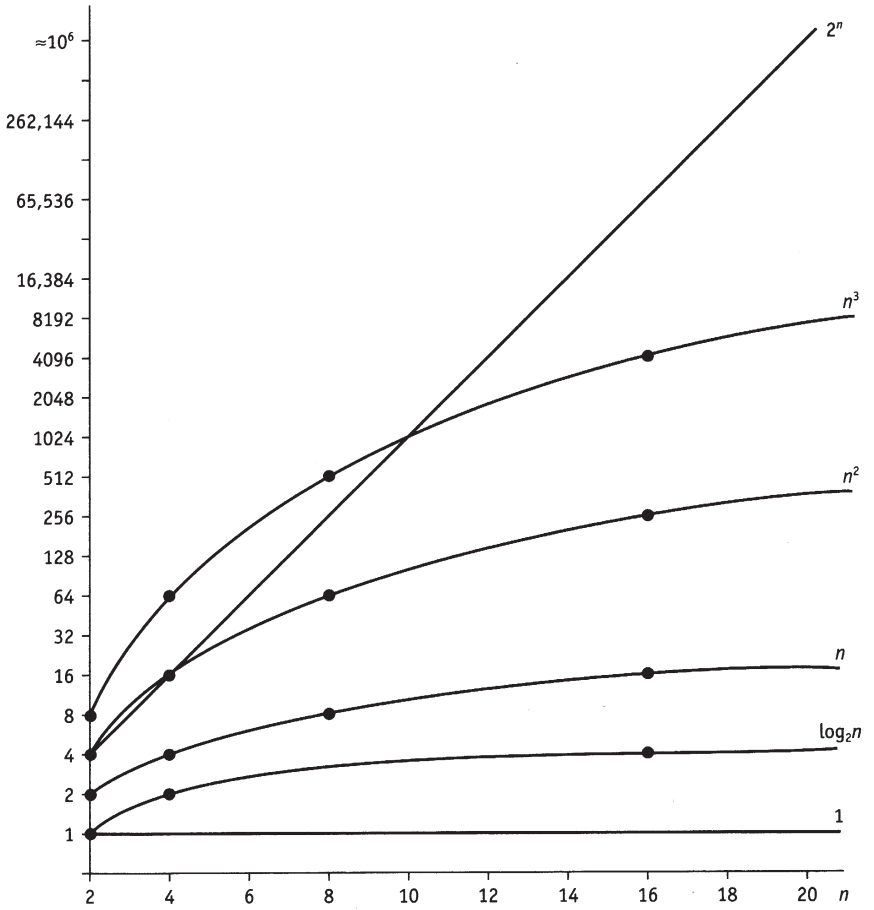


Рисунок 6.2. Относительный порядок роста функций

При вычислении функции временно́й сложности любого алгоритма, необходимо решить, что в данной задаче следует взять в качестве параметра n и какие элементарные операции стоит учитывать при расчетах.

Задача 4. Найдите функцию временно́й сложности следующего фрагмента алгоритма, написанного на псевдокоде, подсчитав количество операторов присваивания $x := x + 1$, которые в нем выполняются.

```
begin
    for i := 1 to 2n do
        for j := 1 to n do
            for k := 1 to j do
                x := x + 1;
end
```

Решение. Внешний цикл (параметризованный переменной i) повторяется $2n$ раз. Цикл, помеченный переменной j, повторяется n раз. При каждом значении j операция $x := x + 1$ выполняется j раз. Следовательно, при каждом значении параметра внешнего цикла i операция $x := x+1$ выполняется $1+2+\cdots+n$ раз, что равно $\frac{1}{2}n(n+1)$. Значит функция временно́й сложности $T(n)$ определяется формулой:

$$T(n) = 2n \cdot \frac{1}{2}n(n+1) = n^2(n+1).$$

Итак, $T(n) = O(n^3)$.

ГЛАВА 7

ГРАФЫ

Графы возникли в восемнадцатом столетии, когда известный математик, Леонард Эйлер, пытался решить теперь уже классическую задачу о Кенигсбергских мостах. В то время в городе Кенигсберге было два острова, соединенных семью мостами с берегами реки Преголь и друг с другом так, как показано на рис. 7.1. Задача состоит в следующем: осуществить прогулку по городу таким образом, чтобы, пройдя ровно по одному разу по каждому мосту, вернуться в то же место, откуда начиналась прогулка. Решая эту задачу, Эйлер изобразил Кенигсберг в виде графа, отождествив его вершины с частями города, а ребра — с мостами, которыми связаны эти части. Как мы покажем в § 7.1, Эйлеру удалось доказать, что искомого маршрута обхода города не существует.

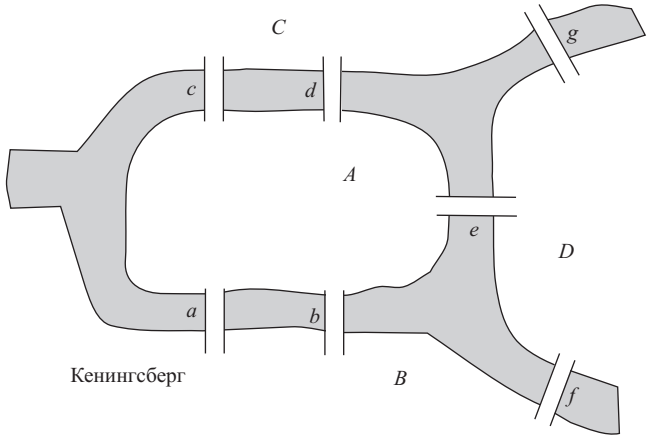


Рисунок 7.1. Схема старого Кенигсберга

В этой главе мы вводим стандартную терминологию, используемую в теории графов, и разбираем несколько конкретных задач, решаемых с помощью графов. В частности, мы познакомимся с классом графов, называемым деревьями. Деревья — естественная модель, представляющая данные, организованные в иерархичную систему. Поиск по дереву для выделения отдельных предметов и сортировка данных в дереве представляют собой важные точки

приложения усилий в информатике. В приложении к этой главе мы и займемся *сортировкой и поиском* данных, организованных в деревья.

7.1. Графы и терминология

На рис. 7.1 изображены семь мостов Кенигсберга так, как они были расположены в восемнадцатом столетии. В задаче, к которой обратился Эйлер, спрашивается: можно ли найти маршрут прогулки, который проходит ровно один раз по каждому из мостов и начинается и заканчивается в одном и том же месте города?

Модель задачи — это *граф*, состоящий из множества *вершин* и множества *ребер*, соединяющих вершины. Вершины A, B, C и D символизируют берега реки и острова, а ребра a, b, c, d, e, f и g обозначают семь мостов (см. рис. 7.2). Искомый маршрут (если он существует) соответствует обходу ребер графа таким образом, что каждое из них проходится только один раз. Проход ребра, очевидно, соответствует пересечению реки по мосту.

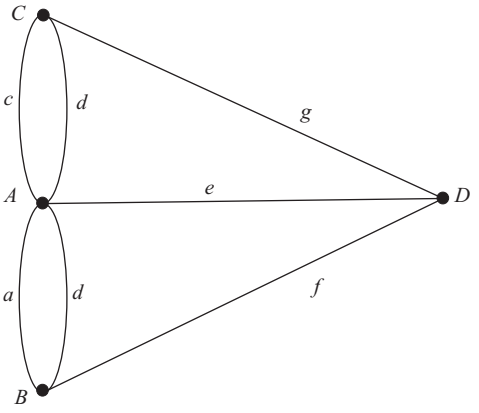


Рисунок 7.2. Модель задачи о мостах Кенигсберга

Граф, в котором найдется маршрут, начинающийся и заканчивающийся в одной вершине, и проходящий по всем ребрам графа ровно один раз, называется *эйлеровым графом*. Последовательность вершин (может быть и с повторениями), через которые проходит искомый маршрут, как и сам маршрут, называется *эйлеровым циклом*. Эйлер заметил, что если в графе есть эйлеров цикл, то для каждого ребра, ведущего в какую-то вершину, должно найтись другое ребро, выходящее из этой вершины[1], и получил из этого простого наблю-

[1]Зайдя в вершину, мы не можем выйти по тому же ребру, соблюдая условия задачи. — *Прим. перев.*

дения такой вывод: если в данном графе существует эйлеров цикл, то к каждой вершине должно подходить четное число ребер.

Кроме того, Эйлеру удалось доказать и противоположное утверждение, так что граф, в котором любая пара вершин связана некоторой последовательностью ребер, является Эйлеровым тогда и только тогда, когда все его вершины имеют четную степень. *Степенью* вершины v называется число $\delta(v)$ ребер, ей *инцидентных*[2].

Теперь совершенно очевидно, что в графе, моделирующем задачу о мостах Кенигсберга, эйлерова цикла найти нельзя. Действительно, степени всех его вершин нечетны: $\delta(B) = \delta(C) = \delta(D) = 3$ и $\delta(A) = 5$. С легкой руки Эйлера графы, подобные тому, который мы исследовали при решении задачи о мостах, стали использоваться при решении многих практических задач, а их изучение выросло в значительную область математики.

Простой граф определяется как пара $G = (V, E)$, где V — конечное множество вершин, а E — конечное множество ребер, причем G не может содержать *петель* (ребер, начинающихся и заканчивающийся в одной вершине) и *кратных ребер* (кратными называются несколько ребер, соединяющих одну и ту же пару вершин). Граф, изображенный на рис. 7.2, не является простым, поскольку, например, вершины A и B соединяются двумя ребрами (как раз эти ребра и называются кратными).

Две вершины u и v в простом графе называются *смежными*, если они соединяются каким-то ребром e, про которое говорят, что оно *инцидентно* вершине u (и v). Таким образом, мы можем представлять себе множество E ребер как множество пар смежных вершин, определяя тем самым нерефлексивное, симметричное отношение на множестве V. Отсутствие рефлексивности связано с тем, что в простом графе нет петель, т. е. ребер, оба конца которых находятся в одной вершине. Симметричность же отношения вытекает из того факта, что ребро e, соединяющее вершину u с v, соединяет и v с u (иначе говоря, ребра не ориентированы, т. е. не имеют направления). Единственное ребро простого графа, соединяющее пару вершин u и v, мы будем обозначать как uv (или vu).

Логическая матрица отношения на множестве вершин графа, которое задается его ребрами, называется *матрицей смежности*. Симметричность отношения в терминах матрицы смежности M означает, что M симметрична относительно главной диагонали. А из-за нерефлексивности этого отношения на главной диагонали матрицы M стоит символ «Л».

[2]Заметим, что если вершина v является концом ребра x, то говорят, что v и x инцидентны. — *Прим. перев.*

Пример 7.1. Нарисуйте граф $G(V, E)$ с множеством вершин $V = \{a, b, c, d, e\}$ и множеством ребер $E = \{ab, ae, bc, bd, ce, de\}$. Выпишите его матрицу смежности.

Решение. Граф G показан на рис. 7.3.

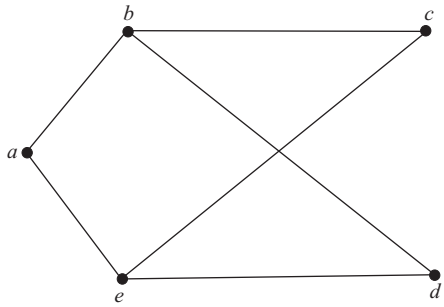


Рисунок 7.3.

Его матрица смежности имеет вид:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \left[\begin{array}{ccccc} \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{И} & \text{И} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{И} & \text{И} & \text{Л} \end{array}\right] \end{array}.$$

Для восстановления графа нам достаточно только тех элементов матрицы смежности, которые стоят над главной диагональю.

Подграфом графа $G = (V, E)$ называется граф $G' = (V', E')$, в котором $E' \subset E$ и $V' \subset V$.

Пример 7.2. Найдите среди графов H, K и L, изображенных на рис. 7.4, подграфы графа G.

Решение. Обозначим вершины графов G, H и K как показано на рис. 7.5. Графы H и K — подграфы в G, как видно из наших обозначений. Граф L не является подграфом в G, поскольку у него есть вершина индекса 4, а у графа G такой нет.

Маршрутом длины k в графе G называется такая последовательность вершин $v_0, v_1, \ldots, v_k$, что для каждого $i = 1, \ldots, k$ пара $v_{i-1}v_i$ образует ребро графа. Мы будем обозначать такой маршрут через $v_0\ v_1\ \ldots\ v_k$. Например, 1 4 3 2 5 — это маршрут длины 4 в графе G из примера 7.2.

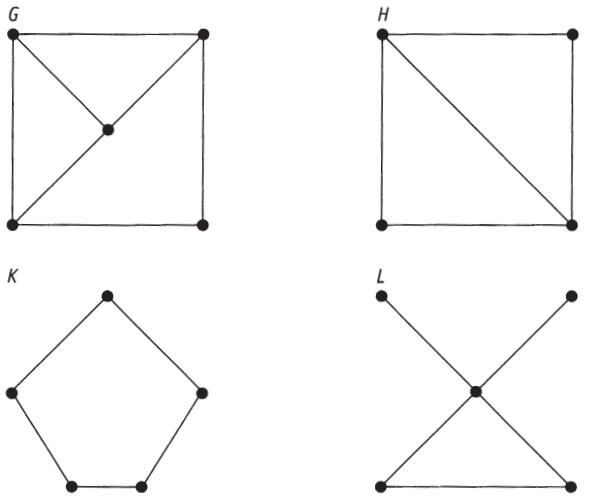


Рисунок 7.4.

Циклом в графе принято называть последовательность вершин $v_0, v_1, \ldots, v_k$, каждая пара которых является концами одного ребра, причем $v_0 = v_1$, а остальные вершины (и ребра) не повторяются. Иными словами, цикл — это замкнутый маршрут, проходящий через каждую свою вершину и ребро только один раз.

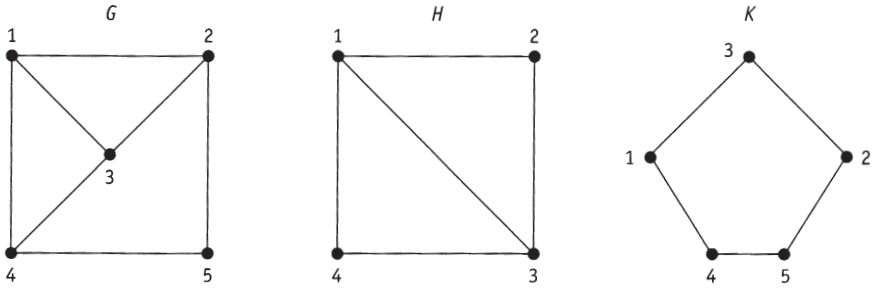


Рисунок 7.5.

Пример 7.3. Найдите циклы в графе G из примера 7.2.

Решение. В этом графе есть два разных цикла длины 5:

$$1\,3\,2\,5\,4\,1 \qquad \text{и} \qquad 1\,2\,5\,4\,3\,1.$$

Мы можем пройти эти циклы как в одном направлении, так и в дру-

гом, начиная с произвольной вершины цикла. Кроме того, в графе есть три разных цикла длины 4:

$$1\,2\,5\,4\,1, \qquad 1\,2\,3\,4\,1 \qquad \text{и} \qquad 2\,5\,4\,3\,2,$$

и два цикла длины 3:

$$1\,2\,3\,1 \qquad \text{и} \qquad 1\,3\,4\,1.$$

Граф, в котором нет циклов, называется *ациклическим*. Структуры деревьев, которые возникают в вычислениях, представляют собой частный случай ациклических графов. Позже в этой главе мы ими займемся.

Граф называют *связным*, если любую пару его вершин соединяет какой-нибудь маршрут. Любой общий граф можно разбить на подграфы, каждый из которых окажется связным. Минимальное число таких связных компонент называется *числом связности* графа и обозначается через $c(G)$. Вопросы связности имеют важное значение в приложениях теории графов к компьютерным сетям. Следующий алгоритм применяется для определения числа связности графа.

Алгоритм связности.

Пусть $G = (V, E)$ — граф. Алгоритм предназначен для вычисления значения $c = c(G)$, т. е. числа компонент связности данного графа G.

```
begin
    V' := V;
    c := 0;
    while V' ≠ ∅ do
        begin
            Выбрать y ∈ V';
            Найти все вершины, соединенные маршрутом с y;
            Удалить вершину y из V' и
            соответствующие ребра из E;
            c := c + 1;
        end
end
```

Пример 7.4. Проследите за работой алгоритма связности на графе, изображенном на рис. 7.6.

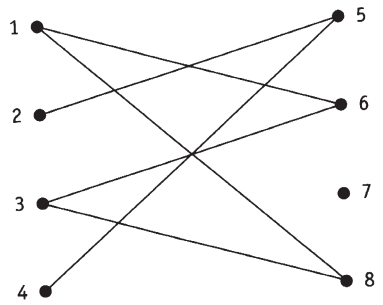


Рисунок 7.6.

Решение. Смотри табл. 7.1.

Таблица 7.1

	V'	c
Исходные значения	{1, 2, 3, 4, 5, 6, 7, 8}	0
Выбор $y = 1$	{2, 4, 5, 7}	1
Выбор $y = 2$	{7}	2
Выбор $y = 7$	$\varnothing$	3

Итак, $c(G) = 3$. Соответствующие компоненты связности приведены на рис. 7.7.

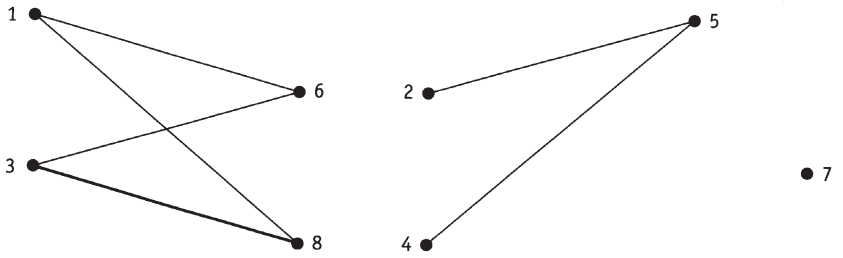


Рисунок 7.7.

7.2. Гамильтоновы графы

Мы начали эту главу с изучения эйлеровых графов, обладающих замкнутым маршрутом, который проходит по всем ребрам графа ровно один раз. Похожая задача состоит в поиске цикла, проходящего

через каждую вершину графа в точности один раз. Такой цикл, если он существует, называется *гамильтоновым*, а соответствующий граф — *гамильтоновым графом*.

Гамильтоновы графы служат моделью при составлении расписания движения поездов, для телекоммуникационных сетей, и т. д. В отличие от задачи Эйлера, простого критерия гамильтоновости графа пока не известно. Поиск хорошего критерия остается одной из главных нерешенных задач теории графов.

Тем не менее, многие графы являются гамильтоновыми. Предположим, что в каком-то графе любая пара вершин соединена ребром. Такой граф называется *полным* и обозначается через K_n, где n — число его вершин. Очевидно, в любом полном графе можно отыскать гамильтонов цикл.

Полный граф K_5 изображен на рис. 7.8. Его цикл $a\ b\ c\ d\ e\ a$, очевидно, является гамильтоновым. В нем есть и другие гамильтоновы циклы. Поскольку каждая вершина смежна с остальными, то начиная с вершины a, в качестве второй вершины цикла мы можем выбрать любую из четырех оставшихся. Далее у нас будет три варианта для выбора третей вершины и два для четвертой, после чего мы вернемся в вершину a. Таким образом, у нас есть $4 \cdot 3 \cdot 2 = 24$ цикла. Поскольку каждый цикл можно проходить как в одном направлении, так и в другом, то реально в графе K_5 есть только 12 разных гамильтоновых циклов[1].

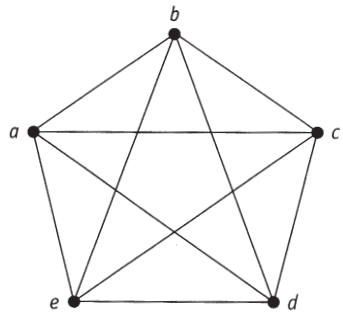


Рисунок 7.8. Полный граф K_5

Поиск гамильтонова цикла (если он существует) в произвольном (связном) графе — задача далеко не всегда простая. Ответ на вопрос о гамильтоновости графа может оказаться довольно трудоемким.

[1] Две разные последовательности вершин: $a\ b\ c\ d\ e\ a$ и $a\ e\ d\ c\ b\ a$ задают, очевидно, один и тот же цикл. — *Прим. перев.*

Пример 7.5. Покажите, что граф, изображенный на рис. 7.9, не является гамильтоновым.

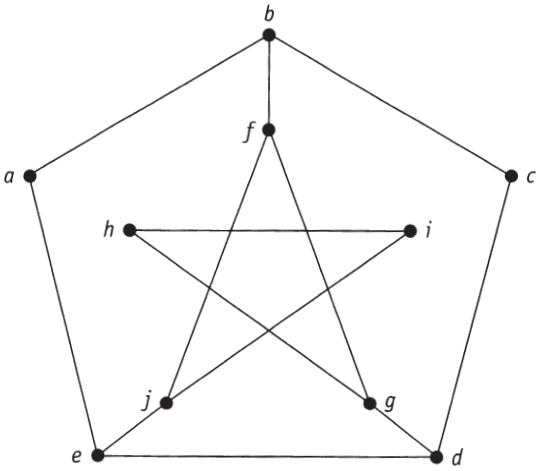


Рисунок 7.9. Пример не гамильтонова графа

Решение. Предположим, что в связном графе найдется гамильтонов цикл. Каждая вершина v включается в гамильтонов цикл C выбором двух инцидентных с ней ребер, а значит, степень каждой вершины в гамильтоновом цикле (после удаления лишних ребер) равна 2. Степени вершин данного графа — 2 или 3. Вершины степени 2 входят в цикл вместе с обоими инцидентными с ними ребрами. Следовательно, ребра ab, ae, cd, cb, hi, hg и ij в том или ином порядке входят в гамильтонов цикл C (см. рис. 7.10).

Ребро bf не может быть частью цикла C, поскольку каждая вершина такого цикла должна иметь степень 2. Значит, ребра fj и fg обязаны входить в цикл C, чтобы включить в него вершину f. Но тогда ребра je и gd никак не могут принадлежать циклу C, поскольку в противном случае в нем появятся вершины степени три. Это вынуждает нас включить в цикл ребро ed, что приводит нас к противоречию: ребра, которые мы были вынуждены выбрать, образуют два несвязных цикла, а не один, существование которого мы предполагали. Вывод: граф, изображенный на рис. 7.10, не является гамильтоновым.

Гамильтоновы графы применяются для моделирования многих практических задач. Основой всех таких задач служит классическая *задача коммивояжера*, формулировка которой приведена на следующей странице.

Коммивояжер должен совершить поездку по городам и вернуться обратно, побывав в каждом городе ровно один раз, сведя при этом затраты на передвижения к минимуму.

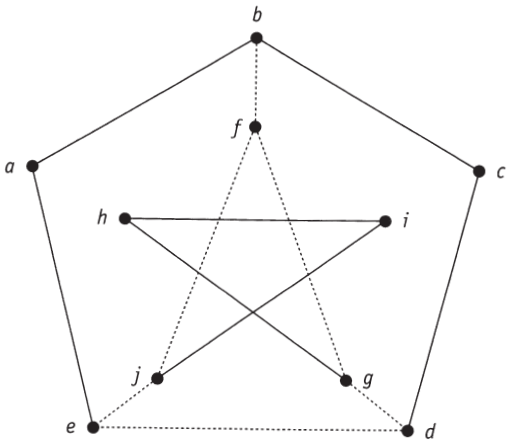


Рисунок 7.10. Ребра, входящие в гамильтонов цикл C

Графическая модель задачи коммивояжера состоит из гамильтонова графа, вершины которого изображают города, а ребра — связывающие их дороги. Кроме того, каждое ребро оснащено *весом*, обозначающим транспортные затраты, необходимые для путешествия по соответствующей дороге, такие, как, например, расстояние между городами или время движения по дороге[2]. Для решения задачи нам необходимо найти гамильтонов цикл минимального общего веса.

К сожалению, эффективный алгоритм решения данной задачи пока не известен. Для сложных сетей число гамильтоновых циклов, которые необходимо просмотреть для выделения минимального, непомерно огромно. Однако существуют алгоритмы поиска *субоптимального решения*. Субоптимальное решение необязательно даст цикл минимального общего веса, но найденный цикл будет, как правило, значительно меньшего веса, чем большинство произвольных гамильтоновых циклов! Один из таких алгоритмов мы сейчас и изучим.

Алгоритм ближайшего соседа.

Этот алгоритм выдает субоптимальное решение задачи коммивояжера, генерируя гамильтоновы циклы в нагруженном графе с

[2]Граф, каждое ребро которого оснащено весом, называется *нагруженным* — *Прим. перев.*

множеством вершин V. Цикл, полученный в результате работы алгоритма, будет совпадать с конечным значением переменной *маршрут*, а его общая длина — конечное значение переменной w.

begin
  Выбрать $v \in V$;
  маршрут $:= v$;
  $w := 0$;
  $v' := v$;
  Отметить v';
  while *остаются неотмеченные вершины* **do**
    begin
      Выбрать неотмеченную вершину u,
      ближайшую к v';
      маршрут $:=$ *маршрут* u;
      $w := w +$ вес ребра $v'u$;
      $v' := u$;
      Отметить v';
    end
  маршрут $:=$ *маршрут* v;
  $w := w +$ вес ребра $v'v$;
end

Пример 7.6. Примените алгоритм ближайшего соседа к графу, изображенному на рис. 7.11. За исходную вершину возьмите вершину D.

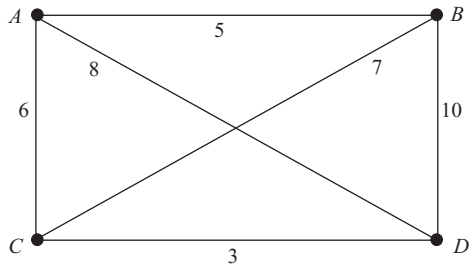


Рисунок 7.11.

Решение. Смотри табл. 7.2.

Таблица 7.2

	u	*маршрут*	w	v'
Исходные значения	—	D	0	D
	C	DC	3	C
	A	DCA	9	A
	B	$DCAB$	14	B
Последний проход	B	$DCABD$	24	B

В результате работы алгоритма был найден гамильтонов цикл $DCABD$ общего веса 24. Делая полный перебор всех циклов в этом маленьком графе, можно обнаружить еще два других гамильтоновых цикла: $ABCDA$ общего веса 23 и $ACBDA$ общего веса 31. В полном графе с двадцатью вершинами существует приблизительно $6,1 \cdot 10^{16}$ гамильтоновых циклов, перечисление которых требует чрезвычайно много машинной памяти и времени.

7.3. Деревья

Как упоминалось ранее в этой главе, есть класс графов, называемых деревьями, которые особенно интенсивно используются в вычислительных приложениях. Граф $G = (V, E)$ называется *деревом*, если он связен и ацикличен (т. е. не содержит циклов).

Пусть $G = (V, E)$ — граф с n вершинами и m ребрами. Можно сформулировать несколько необходимых и достаточных условий, при которых G является деревом:

- Любая пара вершин в G соединена единственным путем.
- G связен и $m = n - 1$.
- G связен, а удаление хотя бы одного его ребра нарушает связность графа.
- G ацикличен, но если добавить хотя бы одно ребро, то в G появится цикл.

Эквивалентность большинства из этих условий устанавливается без особого труда. Наиболее сложно разобраться со вторым из них. В следующем примере мы докажем, что дерево с n вершинами имеет ровно $n - 1$ ребро.

Пример 7.7. Докажите с помощью индукции по числу вершин, что для дерева T с n вершинами и m ребрами выполнено соотношение: $m = n - 1$.

Решение. Поскольку дерево с единственной вершиной вообще не содержит ребер, то доказываемое утверждение справедливо при $n = 1$.

Рассмотрим дерево T с n вершинами (и m ребрами), где $n > 1$ и предположим, что любое дерево с $k < n$ вершинами имеет ровно $k - 1$ ребро.

Удалим ребро из T. По третьему свойству дерево T после этой процедуры превратится в несвязный граф. Получится ровно две компоненты связности, ни одна из которых не имеет циклов (в противном случае исходный граф T тоже содержал бы циклы и не мог

бы быть деревом). Таким образом, полученные компоненты связности — тоже деревья.

Обозначим новые деревья через T_1 и T_2. Пусть n_1 — количество вершин у дерева T_1, а n_2 — у T_2. Поскольку $n_1 + n_2 = n$, то $n_1 < n$ и $n_2 < n$.

По предположению индукции дерево T_1 имеет $n_1 - 1$ ребро, а T_2 — $n_2 - 1$. Следовательно, исходное дерево T насчитывало (с учетом одного удаленного) $(n_1 - 1) + (n_2 - 1) + 1 = n - 1$ ребро, что и требовалось доказать.

Несложно доказать, что в любом связном графе найдется подграф, являющийся деревом. Подграф в G, являющийся деревом и включающий в себя все вершины G, называется *остовным деревом*. Остовное дерево в графе G строится просто: выбираем произвольное его ребро и последовательно добавляем другие ребра, не создавая при этом циклов, до тех пор, пока нельзя будет добавить никакого ребра, не получив при этом цикла. Благодаря примеру 7.7, мы знаем, что для построения остовного дерева в графе из n вершин необходимо выбрать ровно $n - 1$ ребро.

Пример 7.8. Найдите два разных остовных дерева в графе, изображенном на рис. 7.12.

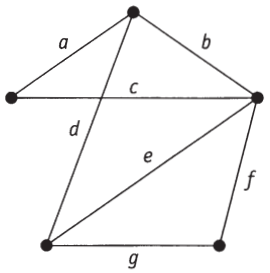


Рисунок 7.12. Связный граф G

Решение. В этом графе существует несколько остовных деревьев. Одно из них получается последовательным выбором ребер: a, b, d и f. Другое — b, c, e и g. Названные деревья показаны на рис. 7.13.

Процесс, описанный в примере 7.8, можно приспособить для решения *задачи поиска кратчайшего соединения*:

> *Нужно построить железнодорожную сеть, связывающую некоторое число городов. Известна стоимость строительства отрезка путей между любой парой городов. Требуется найти сеть минимальной стоимости.*

На языке теории графов нам нужно в нагруженном графе найти остовное дерево наименьшего общего веса. Такое дерево принято называть *минимальным остовным деревом* или, сокращенно, *МОД*. В отличие от задачи коммивояжера, здесь есть эффективный алгоритм, находящий действительно минимальное остовное дерево. Он похож на алгоритм Прима, с которым мы познакомились в главе 1 при решении задачи поиска кратчайшего соединения для набора из шести шотландских городов.

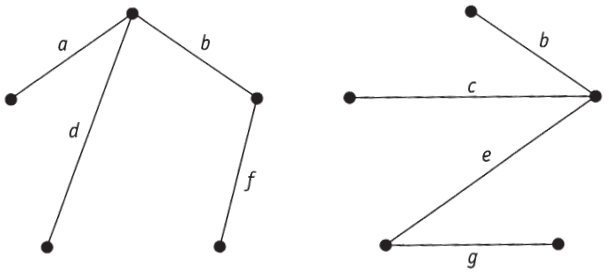


Рисунок 7.13. Остовные деревья графа G

Алгоритм поиска минимального остовного дерева. Пусть $G = (V, E)$ — связный взвешенный граф. Алгоритм строит МОД в графе G, последовательно выбирая ребра наименьшего возможного веса до образования остовного дерева. МОД в памяти компьютера хранится в виде множества T ребер.

```
begin
    e := ребро графа G с наименьшим весом;
    T := {e};
    E' := E \ {e}
    while E' ≠ ∅
        begin
            e' := ребро из E' наименьшего веса;
            T := T ∪ {e'};
            E' := множество ребер из E' \ {T},
                  чье добавление к T не ведет
                  к образованию циклов;
        end
end
```

Пример 7.9. В табл. 7.3 дано расстояние (в милях) между пятью деревнями A, B, C, D и E. Найдите минимальное остовное дерево.

Таблица 7.3

	A	B	C	D	E
A	—	13	3	9	9
B	13	—	11	11	13
C	3	11	—	9	7
D	9	11	9	—	2
E	9	13	7	2	—

Решение. Ребра выбираются следующим образом: первое — ребро DE веса 2; второе — AC веса 3; третье — CE веса 7. На этой стадии строящееся дерево выглядит так, как на рис. 7.14

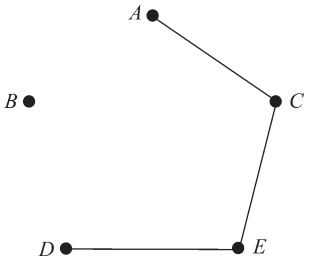


Рисунок 7.14. Вид дерева после трех шагов

Следующие по весу ребра — AD, AE и CD, каждое из которых имеет вес 9. Однако какое бы из них мы ни добавили, получится цикл. Поэтому перечисленные ребра следует исключить из числа доступных для строительства. Далее идут ребра BC и BD веса 11. Можно присоединить любое из них, получив при этом два разных МОД: $\{AC, BC, CE, DE\}$ или $\{AC, BD, CE, DE\}$ веса 23 каждое.

Зачастую нам хотелось бы иметь деревья, представляющие информацию с учетом естественной иерархической структуры, такие, как, например, генеалогическое древо (рис. 7.15). На нем показаны некоторые члены семьи Бернулли, каждый из которых был известным швейцарским математиком.

Гинеалогическое древо можно изобразить и более сжато. Схема, приведенная на рис. 7.16, представляет собой пример так называемого дерева с корнем. *Деревом с корнем* называется дерево с одной выделенной вершиной. Именно эта выделенная вершина и является *корнем* дерева. Корень, в некотором смысле, можно назвать «величайшей» вершиной (например, родоначальник математиков Бернулли). Вершины дерева, лежащие непосредственно под данной, называются *сыновьями*. С другой стороны, вершина, стоящая непосредственно перед сыном, называется ее *отцом*.

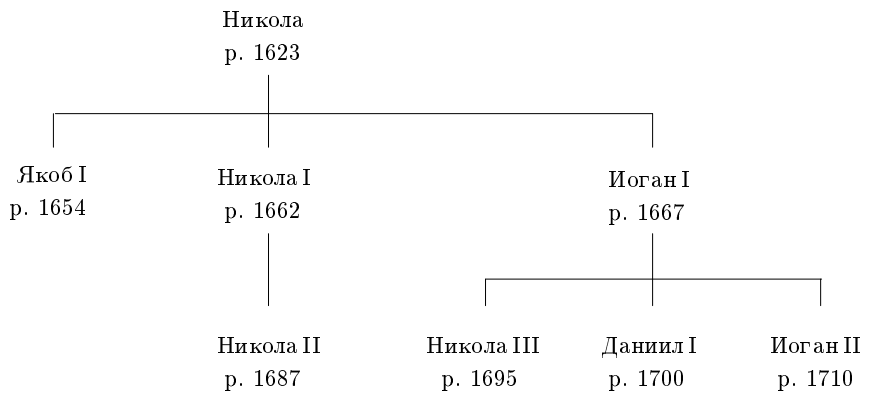


Рисунок 7.15. Династия Бернулли

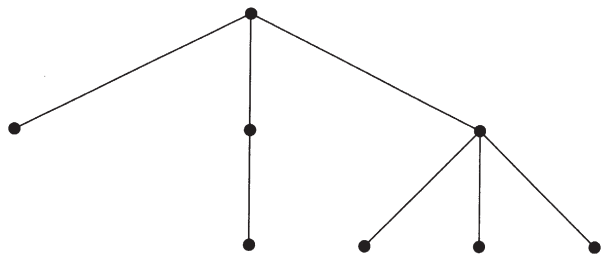


Рисунок 7.16. Схема генеалогического древа Бернулли

Дерево с корнем можно определить рекуррентным образом. Отдельная вершина является деревом с корнем (она же служит и корнем такого дерева). Если $T_1, T_2, \ldots, T_k$ — несвязанные друг с другом деревья с корнями v_1, v_2, $\ldots$, v_k, то граф, получающийся присоединением новой вершины v к каждой из вершин v_1, v_2, $\ldots$, v_k отдельным ребром, является деревом T с корнем v. Вершины v_1, $\ldots$, v_k графа T — это сыновья корня v. Мы изображаем такое дерево с корнем, расположенным наверху, и сыновьями, стоящими ниже, непосредственно под корнем (см. рис. 7.17). Каждую вершину дерева с корнем можно рассматривать как корень другого дерева, которое «растет» из него. Мы будем называть его *поддеревом* дерева T.

Как мы уже говорили, вершина на самом верху дерева — его корень, а вот те, которые находятся в самом низу дерева (и не имеют сыновей) принято называть *листьями*. Остальные вершины, отличные от корня и листьев, называют *внутренними*.

Область применения деревьев с корнем обширна. Это, например, и информатика, и биология, и менеджмент. Для приложения к

информатике наиболее важны так называемые *двоичные* или *бинарные* деревья с корнем. Двоичное дерево отличает от остальных то, что каждая его вершина имеет не более двух сыновей. В двоичном дереве с корнем вниз от каждой вершины идет не более двух ребер.

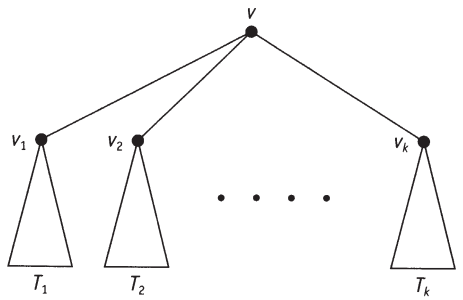


Рисунок 7.17.

Таким образом, можно сказать, что каждой вершине двоичного дерева с корнем соответствует не более, чем два поддерева, которые принято называть *левым* и *правым* поддеревьями этой вершины. Если оказалось, что у какой-то вершины дерева отсутствует потомок слева, то ее левое поддерево называют *нулевым деревом* (т. е. нулевое дерево — это дерево без единой вершины). Аналогично, если у вершины отсутствует правый потомок, то ее правое поддерево будет нулевым.

Пример 7.10. Пусть T — двоичное дерево с корнем, изображенное на рис. 7.18.

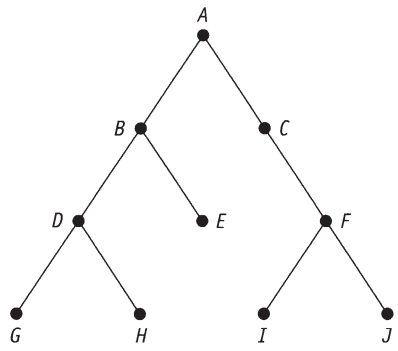


Рисунок 7.18. Двоичное дерево с корнем T

Определите

(а) корень T;
(б) корень левого поддерева вершины B;
(в) листья T;
(г) сыновей вершины C.

Нарисуйте двоичное дерево с корнем T', полученное из T перестановкой левых и правых поддеревьев у каждой вершины.

Решение. (а) A; (б) D; (в) G, H, E, I и J; (г) F.

Двоичное дерево с корнем T' начерчено на рис. 7.19.

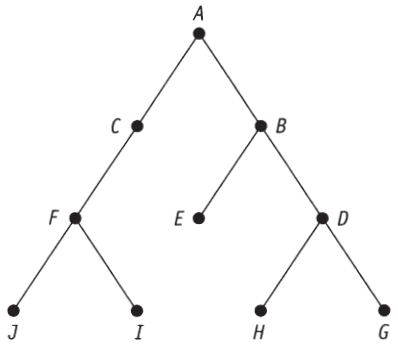


Рисунок 7.19. Двоичное дерево с корнем T'

Набор упражнений 7

7.1. Объясните, почему сумма степеней всех вершин простого графа G совпадает с удвоенным числом его ребер. Этот факт называют леммой об эстафете.

Используя эту лемму, покажите, что в любом полном графе K_n с n вершинами есть ровно $\frac{n(n-1)}{2}$ ребер.

Для каких значений n граф K_n будет эйлеровым?

7.2. Опираясь на принцип Дирихле, докажите, что если простой граф G имеет более одной вершины, то у него найдутся по крайней мере две вершины одинаковой степени.

7.3. Нарисуйте граф, чья матрица смежности имеет вид:

$$\begin{bmatrix} Л & И & Л & И & Л & И \\ И & Л & И & Л & И & Л \\ Л & И & Л & И & Л & И \\ И & Л & И & Л & И & Л \\ Л & И & Л & И & Л & Л \\ И & Л & И & Л & Л & Л \end{bmatrix}.$$

Опишите матрицу смежности полного графа K_n.

7.4. Введя подходящие обозначения вершин, для каждого из графов на рис. 7.20 подберите соответствующую матрицу смежности из перечисленных ниже.

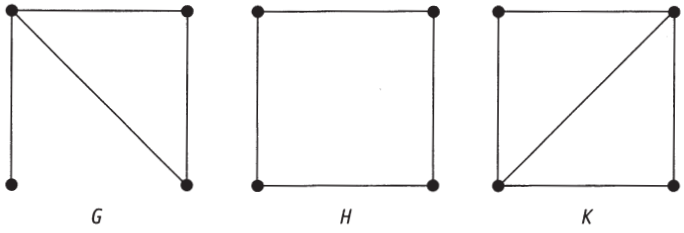


Рисунок 7.20.

$$\begin{bmatrix} Л & И & И & Л \\ И & Л & Л & И \\ И & Л & Л & И \\ Л & И & И & Л \end{bmatrix} \quad \begin{bmatrix} Л & И & И & Л \\ И & Л & И & И \\ И & И & Л & И \\ Л & И & И & Л \end{bmatrix} \quad \begin{bmatrix} Л & И & Л & Л \\ И & Л & И & И \\ Л & И & Л & И \\ Л & И & И & Л \end{bmatrix}$$

(а) (б) (с)

7.5. Какие из графов на рис. 7.21 могут являться подграфами графа из упражнения 7.3?

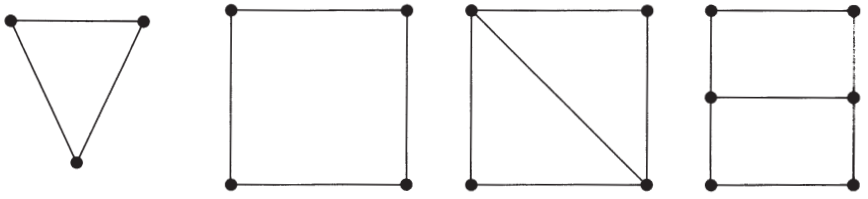


Рисунок 7.21. Кандидаты в подграфы

7.6. Найдите гамильтоновы циклы в графе на рис. 7.22.

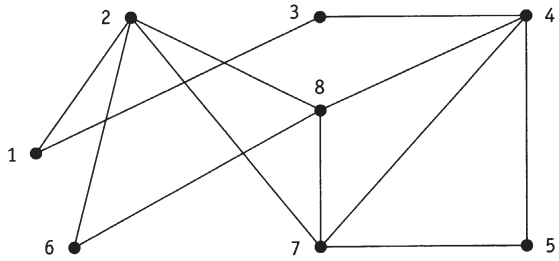


Рисунок 7.22.

Найдите в нем циклы длины 3, 4, 5, 6 и 7.

7.7. На рисунке рис. 7.23 изображен граф Петерсена P.

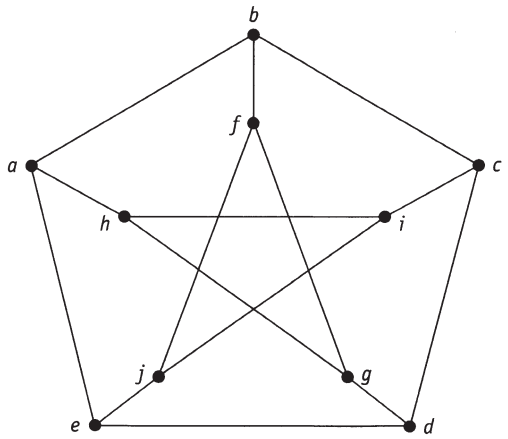


Рисунок 7.23. Граф Петерсена

Найдите в нем цикл длины 9. Покажите, что P не является гамильтоновым.

7.8. Используйте алгоритм ближайшего соседа для поиска гамильтонова цикла в нагруженном графе (рис. 7.24), взяв за исходную

(а) вершину A;

(б) вершину D.

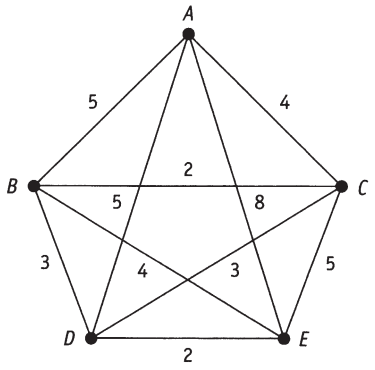


Рисунок 7.24. Нагруженный граф

7.9. Выясните, являются ли графы, задаваемые следующими матрицами смежности, деревьями:

(а)
$$\begin{bmatrix} Л & Л & Л & Л & Л & И \\ Л & Л & Л & И & Л & И \\ Л & Л & Л & Л & И & И \\ Л & И & Л & Л & И & Л \\ Л & Л & И & И & Л & Л \\ И & И & И & Л & Л & Л \end{bmatrix};$$

(б)
$$\begin{bmatrix} Л & Л & Л & Л & Л & И \\ Л & Л & И & Л & Л & Л \\ Л & И & Л & И & Л & И \\ Л & Л & И & Л & И & Л \\ Л & Л & Л & И & Л & Л \\ И & Л & И & Л & Л & Л \end{bmatrix}.$$

7.10. Известно, что дерево T имеет три вершины степени 3 и четыре вершины степени 2. Остальные вершины дерева имеют степень 1. Сколько вершин степени 1 есть у дерева T?
(Указание: обозначьте число вершин дерева T через n и примените лемму об эстафете.)

7.11. *Лесом* называют граф (не обязательно связный), каждая компонента связности которого — дерево. Пусть G — лес с n вершинами и k компонентами связности.

(а) Докажите, что G имеет $n - k$ ребер.

(б) Покажите, что если в каждой компоненте связности леса G есть более одной вершины, то G содержит по крайней мере $2k$ вершин степени 1.

(в) Нарисуйте лес с девятью вершинами и шестью ребрами, в котором не больше пяти вершин степени 1.

7.12. Найдите минимальное остовное дерево графа, изображенного на рис. 7.25.

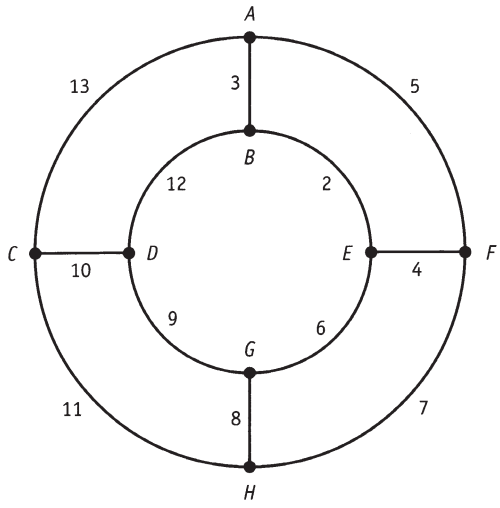


Рисунок 7.25.

7.13. В табл. 7.4 приведены расстояния (в милях) между шестью городами Ирландии.

Таблица 7.4

	Атлон	Дублин	Голуэй	Лимерик	Слайго	Уэксфорд
Атлон	—	78	56	73	71	114
Дублин	78	—	132	121	135	96
Голуэй	56	132	—	64	85	154
Лимерик	73	121	64	—	144	116
Слайго	71	135	85	144	—	185
Уэксфорд	114	96	154	116	185	—

Используя алгоритм поиска минимального остовного дерева, найдите сеть дорог минимальной общей длины, связывающую все шесть городов.

7.14. *Глубина вершины* v дерева с корнем T определяется как длина единственного пути от нее к корню дерева. *Глубина графа* T — это максимальная глубина его вершин.

(а) Начертите следующие деревья:
(i) дерево с корнем глубины 1 с шестью вершинами;
(ii) полное двоичное дерево с корнем глубины 2;
(iii) дерево с корнем глубины 3, каждая вершина глубины i ($i \geqslant 0$) которого имеет $(i+1)$ сына.

(б) Покажите индукцией по n, что полное двоичное дерево с корнем глубины n имеет 2^n листьев.

(*Полным* называется двоичное дерево с корнем, у которого все вершины (за исключением листьев) имеют по два сына.)

Краткое содержание главы

Граф $G = (V, E)$ состоит из множества V, чьи элементы называют **вершинами** графа, и множества E его ребер, соединяющих некоторые пары вершин.

Вершины u и v графа называют **смежными**, если они соединены каким-то ребром e, про которое говорят, что оно инцидентно вершинам u и v.

Степенью вершины v считают число $\delta(v)$ ребер графа, инцидентных v.

Граф, в котором существует маршрут (называемый **эйлеровым**), начинающийся и заканчивающийся в одной и той же вершине и проходящий по каждому ребру графа ровно один раз, называется **Эйлеровым** графом. Связный граф с двумя или более вершинами является эйлеровым тогда и только тогда, когда каждая его вершина имеет четную степень.

Лемма об эстафете утверждает, что сумма степеней вершин произвольного графа $G = (V, E)$ равна удвоенному числу его ребер.

Простым принято называть граф $G = (V, E)$ с конечным множеством вершин V и конечным множеством ребер E, в котором нет петель и кратных ребер.

Логическая матрица отношения на множестве вершин простого графа G, которое задается его ребрами, называется **матрицей смежности**.

Подграфом графа $G = (V, E)$ называют граф $G' = (V', E')$, в котором $E' \subset E$ и $V' \subset V$.

Маршрутом длины k в графе называют такую последовательность различных вершин $v_0, v_1, \ldots, v_k$, в которой каждая пара соседних вершин $v_{i-1}v_i$ соединена ребром.

Циклом в графе является замкнутый маршрут $v_0, v_1, \ldots, v_0$, у которого все вершины, кроме первой и последней, различны.

Граф, не содержащий циклов, называют **ацикличным**.

Связным является тот граф, в котором каждая пара вершин соединена маршрутом.

Количество компонент связности графа можно подсчитать с помощью **алгоритма связности**.

Гамильтоновым называют такой цикл в графе, который проходит через каждую вершину графа, причем только один раз. Граф, в котором существует гамильтонов цикл, называют **гамильтоновым**.

Задача коммивояжера состоит в поиске гамильтонова цикла минимального общего веса в нагруженном графе. **Алгоритм ближайшего соседа** позволяет найти субоптимальное решение задачи коммивояжера.

Связный ацикличный граф $G = (V, E)$ является **деревом**. Следующие утверждения о связном графе $G = (V, E)$ с n вершинами и m ребрами эквивалентны:

(а) G — дерево;

(б) любую пару вершин G связывает единственный путь;

(в) G связен и $m = n - 1$;

(г) G связен, а удаление любого его ребра нарушает это свойство;

(д) G ацикличен, но соединяя любую пару вершин новым ребром, мы получаем цикл.

Остовным деревом графа G называют такой его подграф, который является деревом и содержит все вершины графа G. **Алгоритм поиска минимального остовного дерева** позволяет найти остовное дерево минимального общего веса в нагруженном графе и может быть использован для решения **задачи поиска кратчайшего соединения**.

Дерево с одной выделенной вершиной называют **деревом с корнем**, а выделенную вершину — его **корнем**. Вершины, стоящие непосредственно под вершиной v (и соединенные с ней ребрами), называются **сыновьями** вершины v. Вершины, расположенные в самом низу дерева (они не имеют сыновей), называются **листьями**. Вершины, отличные от корня и листьев, называют **внутренними** вершинами графа. **Нулевое дерево** — это дерево, не имеющее ни одной вершины.

Каждая вершина дерева с корнем T является корнем какого-то другого дерева, называемого **поддеревом** T. В **двоичном дереве с корнем** каждая вершина имеет не более двух сыновей, а два поддерева вершины v называют **левым** и **правым** поддеревьями, ассоциированными с v. Двоичное дерево с корнем называют **полным**, если каждая его вершина, за исключением листьев, имеет ровно по два сына.

Глубиной вершины v дерева с корнем T принято считать длину единственного маршрута, соединяющего ее с корнем. **Глубиной графа** T называют максимальную глубину его вершин.

Приложение. Сортировка и поиск

Двоичные деревья с корнем очень полезны при решении задач о выборе, в частности, о выборе такого сорта, при котором нужно классифицировать упорядоченные данные или вести в них поиск.

Упорядоченные данные, такие как множество чисел, упорядоченных по величине или множество строк литер, упорядоченных лексикографически (в алфавитном порядке), можно организовать в виде вершин двоичного дерева с корнем в соответствии с их порядком. При этом мы стремимся к тому, чтобы данные, стоящие в левом поддереве данной вершины v были бы меньше данных, соответствующих этой вершине, а данные, расположенные в правом ее поддереве — больше. Дерево данных, удовлетворяющее указанному условию, называют *двоичным деревом поиска*.

Например, в дереве двоичного поиска, приведенном на рис. 7.26, слова фразы «У МОЕГО КОМПЬЮТЕРА ЕСТЬ ЧИП НА МАТЕРИНСКОЙ ПЛАТЕ» организованы именно таким образом. Заметим, что каждое слово в левом поддереве любой вершины предшествует (относительно алфавитного порядка) слову, стоящему в этой вершине, а каждое слово ее правого поддерева следует за словом выбранной вершины.

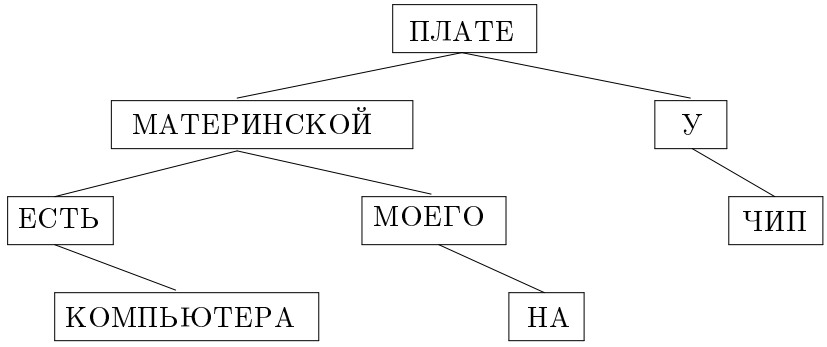


Рисунок 7.26. Дерево двоичного поиска

Преимущество организации упорядоченных данных в виде двоичного дерева поиска заключается в возможности создания эффективного алгоритма поиска каких-то конкретных данных, включения новых данных в дерево и печати всей информации, содержащейся в дереве в виде упорядоченного списка.

Предположим, что в университете хранится список студентов (упорядоченный в алфавитном порядке), в котором кроме фамилий и имен имеются дополнительные важные сведения о студентах. Допустим также, что возникла необходимость найти какую-то информацию в списке или добавить новые записи к списку. Мы сейчас познакомимся с алгоритмами, которые осуществляют поиск конкретной информации, добавляют новых студентов к списку и выводят на печать все записи в алфавитном порядке.

Записи о студентах организованы в двоичное дерево поиска (каждая запись соответствует одной вершине), и наши алгоритмы будут исследовать вершины этого дерева. Поскольку каждая вершина является также и корнем некоторого двоичного дерева поиска, алгоритмы будут последовательно проверять левые и правые поддеревья вершин. Чтобы это осуществить, необходимо приписать каждой вершине некоторый *ключ* для идентификации и ссылок на ее левое и правое поддеревья (в структурах данных для этих целей используются так называемые дважды связанные списки). Из всех ключей

организуется линейно упорядоченное множество (в нашей ситуации оно упорядочено лексикографически).

Алгоритм *поиска* определяет, является ли данная запись (*ключ поиска*) вершиной в двоичном дереве поиска, сравнивая ключ поиска с ключом корня дерева, и, при необходимости, осуществляет аналогичные сравнения в левом или правом поддеревьях.

```
Поиск(дерево)
    begin
        if дерево нулевое then
            поиск := ложь;
        else
            if ключ поиска = ключ корня then
                поиск := истина;
            else
                if ключ поиска < ключ корня then
                    поиск := поиск(левое поддерево);
                else
                    поиск := поиск(правое поддерево);
    end
```

Задача 1. Проследите за работой алгоритма над двоичным деревом поиска, изображенном на рис. 7.27. Известно, что ключ поиска — буква R, а ключи вершин упорядочены лексикографически.

Решение. Поскольку $R > K$, то поиск продолжается в правом поддереве вершины K. Так как $R < T$, процесс поиска переключается на левое поддерево вершины T. Наконец, ввиду неравенства $R \neq M$ и отсутствия поддеревьев у вершины M, алгоритм заканчивается и сообщает, что искомая вершина не была найдена.

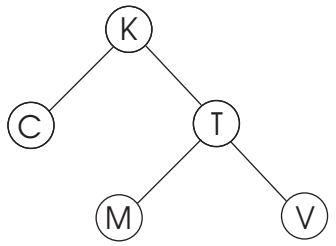


Рисунок 7.27.

Алгоритм вставки вставляет новые вершины (*ключи вставок*) в двоичное дерево поиска, создавая при этом новую вершину слева или справа от уже существующей. Это делается таким образом,

чтобы все ключи вершин в получившемся дереве подчинялись установившемуся порядку.

```
Вставка(запись, дерево)
    begin
        if дерево нулевое then
            добавить новую вершину;
        else
            if ключ вставки = ключ корня then
                вывести на печать:
                «запись содержится в дереве»;
            else
                if ключ вставки < ключ корня then
            вставка:=вставка(запись, левое поддерево);
                else
            вставка:=вставка(запись, правое поддерево);
    end
```

Задача 2. Проследите за работой алгоритма вставки на примере вершин R, A и L в дерево из задачи 1.

Решение. Поскольку $R > K$, мы применяем алгоритм вставки к правому поддереву вершины K. Далее мы видим, что $R < T$. Значит, алгоритм вставки переключается на левое поддерево вершины T. Так как $R > M$ и правое поддерево вершины M нулевое, то мы ставим вершину R справа от M и получаем дерево, изображенное на рис. 7.28. Теперь вставим A и L, построив дерево, показанное на рис. 7.29.

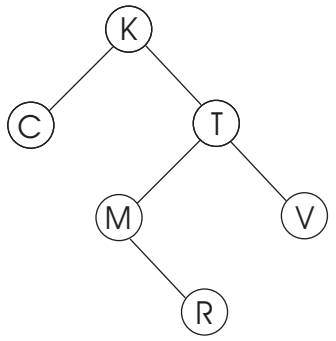


Рисунок 7.28.

Алгоритм вставки можно использовать для создания двоичного дерева поиска, начиная с нулевого дерева и последовательно добавляя новые данные в удобном для нас порядке. Например, двоичное

дерево поиска на рис. 7.26 является результатом применения алгоритма вставки к нулевому дереву в процессе добавления слов фразы «У МОЕГО КОМПЬЮТЕРА ЕСТЬ ЧИП НА МАТЕРИНСКОЙ ПЛАТЕ» в том порядке, в котором они в ней записаны.

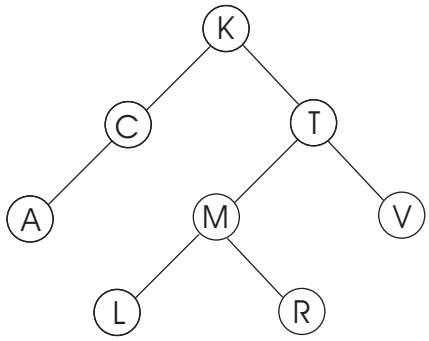


Рисунок 7.29.

Алгоритм правильного обхода выводит на печать всю информацию, содержащуюся в двоичном дереве поиска, в надлежащем порядке. При этом все вершины дерева осматриваются в определенном порядке. Алгоритм работает следующим образом. Для каждой вершины, начиная с корня, печатается вся информация, содержащаяся в вершинах левого поддерева. Затем выводится информация, хранящаяся в этой вершине, и наконец, информация, соответствующая вершинам правого поддерева.

```
Правильный обход(дерево)
    begin
        if дерево нулевое then
            ничего не делать;
        else
            begin
                правильный обход(левое поддерево);
                напечатать корневой ключ;
                правильный обход(правое поддерево);
            end
    end
```

Задача 3. Примените алгоритм правильного обхода к дереву, полученному в задаче 2 после вставки R, A и L.

Решение. После работы алгоритма над указанным деревом получается список:

$$A,\ C,\ K,\ L,\ M,\ R,\ T,\ V.$$

Он соответствует обходу дерева против часовой стрелки (рис. 7.30) и печати информации, содержащейся в вершинах, как только Вы прошли *под* вершиной.

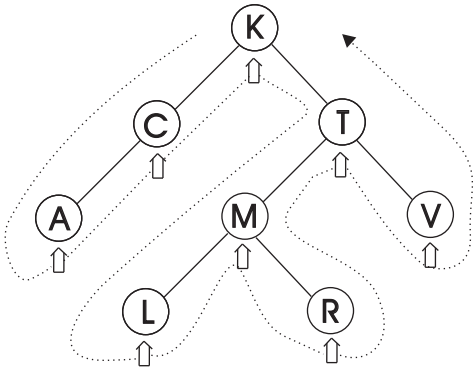


Рисунок 7.30.

ГЛАВА 8

ОРИЕНТИРОВАННЫЕ ГРАФЫ

Мы впервые столкнулись с ориентированными графами в главе 4 при описании двойных отношений. Ориентированные графы или, для краткости, орграфы используются для моделирования ситуаций, в которых есть отношение частичного порядка между объектами. Возникающие при этом схемы служат для изображения схем информационных потоков, сетевого планирования и планирования заданий. В этой главе мы введем стандартную терминологию из теории орграфов и обсудим систему ПЕРТ. ПЕРТ — это система планирования и руководства разработками. На английском языке ее называют PERT — сокращение от «Program Evaluation and Review Technique». ПЕРТ была разработана для помощи в конструировании подводной лодки военно-морского флота США.

Основная часть главы посвящена проблеме поиска путей в сетях. Существование путей мы будем устанавливать с помощью матриц достижимости (это матрицы замыкания (относительно транзитивности) отношения, задаваемого ребрами сети). В § 8.2 описан эффективный алгоритм вычисления матрицы достижимости, известный как алгоритм Уоршелла. Этим мы, наконец, завершим работу, начатую в § 4.2. Далее мы обсудим алгоритм Дейкстры, предназначенный для поиска кратчайшего пути в сетях.

Приложение к этой главе доказывает, что именно алгоритм Дейкстры играет центральную роль при создании динамических таблиц маршрутов в коммуникационных сетях.

8.1. Ориентированные графы

Ориентированный граф или *орграф* представляет собой пару $G = (V, E)$, где V — конечное множество *вершин*, а E — отношение на V. Графическое изображение графа состоит из множества помеченных вершин с ориентированными ребрами (называемых *дугами*), соединяющими пары вершин. Совокупность всех дуг образует множество E.

Дугу, соединяющую пару (u, v) вершин u и v орграфа G, будем обозначать через uv. В простом орграфе отсутствуют петли и

кратные дуги. Следовательно, для любой пары вершин u и v в орграфе найдется не более одной дуги uv из вершины u в v, и не более одной дуги vu из v в u. Если uv — дуга орграфа, то u называют *антецедентом* v.

На рис. 8.1 приведен пример простого орграфа с множеством вершин $V = \{a, b, c, d\}$ и множеством дуг $E = \{ab, bd, cb, db, dc\}$.

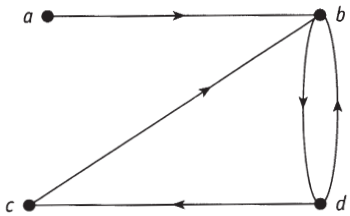


Рисунок 8.1. Пример орграфа

Матрицей смежности данного графа служит (несимметричная) матрица

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \left[\begin{array}{cccc} \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{И} & \text{Л} \end{array}\right] \end{array}$$

(вершины a, c и d здесь — антецеденты b).

Путем длины k в орграфе называют последовательность различных вершин v_0, v_1, ..., v_k, каждая пара $v_{i-1}v_i$ которой образует дугу $(i = 1, \ldots, k)$.

Контуром в орграфе G принято называть последовательность вершин v_0, v_1, ..., v_k, образующую путь, в которой первая вершина v_0 совпадает с последней v_k , а других повторяющихся вершин в ней нет. Орграф G называют *бесконтурным*, если в нем нет контуров.

Бесконтурные орграфы полезны в качестве моделей ситуаций, задачи в которых должны выполняться в определенном порядке (контур в такой интерпретации означает, что та или иная задача выполняется с некоторой периодичностью и предшествует сама себе). В задаче о планировании заданий соответствующий бесконтурный орграф имеет кодовое название «система ПЕРТ».

Пример 8.1. Для получения степени магистра биологии студенту университета, в частности, необходимо прослушать восемь курсов, которые некоторым образом зависят друг от друга. Эта зависимость представлена в табл. 8.1. Изобразите систему ПЕРТ, иллюстрирующую приоритетную структуру курсов.

Таблица 8.1

		Предварительные курсы
(**A**)	Биотехнология	**B**
(**B**)	Начальный курс биотехнологии	**C**
(**C**)	Цитология	**H**
(**D**)	Структура ДНК	**C**
(**E**)	Энзимология	**D, G**
(**F**)	Диетология	**E**
(**G**)	Генная инженерия	**C**
(**H**)	Биология человека	Никаких требований

Решение. Система ПЕРТ (см. рис. 8.2) — это просто орграф, представляющий данную приоритетную структуру. Вершины орграфа в данном случае — восемь курсов. Для краткости ссылок мы обозначим курсы буквами латинского алфавита от **A** до **H**. Дуги орграфа отражают представленные в таблице требования, необходимые для усвоения данного курса.

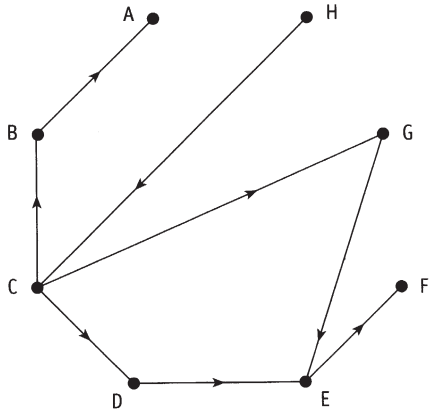


Рисунок 8.2. Система ПЕРТ: приорететная структура курсов

Предположим, что студент из примера 8.1 намерен определить порядок, в котором ему следует изучать предметы, учитывая их

зависимость друг от друга. Он может сделать это с помощью *алгоритма топологической сортировки*. Алгоритм создает *последовательность согласованных меток* для вершин бесконтурного орграфа таким образом, что если 1, 2, 3, ..., n — метки вершин и uv — дуга орграфа, идущая от вершины u с меткой i к вершине v с меткой j, то $i < j$.

Алгоритм топологической сортировки. Алгоритм генерирует последовательность согласованных меток для вершин бесконтурного орграфа $G = (V, E)$. В самом начале работы алгоритма антецеденты каждой вершины v записываются в множество $A(v)$.

```
begin
    for v ∈ V do
      вычислить A(v);
      label := 0;
      whileостаются неотмеченные вершины, для которых
            A(v) = ∅ do
          begin
              label := 1;
              u := вершина с A(u) = ∅;
              присвоить метку вершине u;
              for каждой неотмеченной вершины v ∈ V do;
              A(v) := A(v) \ {u};
          end
end
```

Алгоритм успешно присваивает метки вершинам. Каждая вершина получает очередную метку в том случае, если у нее нет неотмеченных антецедентов.

Пример 8.2. Найдите последовательность меток для орграфа, изображенного на рис. 8.2.

Решение.

Шаг 0 Множество антецедентов выглядит следующим образом: $A(\mathbf{A}) = \{\mathbf{B}\}$, $A(\mathbf{B}) = \{\mathbf{C}\}$, $A(\mathbf{C}) = \{\mathbf{H}\}$, $A(\mathbf{D}) = \{\mathbf{C}\}$, $A(\mathbf{E}) = \{\mathbf{D}, \mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$, $A(\mathbf{G}) = \{\mathbf{C}\}$ и $A(\mathbf{H}) = \varnothing$.

Шаг 1 Первый проход цикла **while**. Назначить метку 1 вершине **H** и удалить вершину **H** из всех оставшихся множеств $A(v)$. $A(\mathbf{A}) = \{\mathbf{B}\}$, $A(\mathbf{B}) = \{\mathbf{C}\}$, $A(\mathbf{C}) = \varnothing$, $A(\mathbf{D}) = \{\mathbf{C}\}$, $A(\mathbf{E}) = \{\mathbf{D}, \mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$ и $A(\mathbf{G}) = \{\mathbf{C}\}$.

Шаг 2 Второй проход цикла **while**. Назначить метку 2 вершине **C** и удалить вершину **C** из всех оставшихся множеств $A(v)$.

$A(\mathbf{A}) = \{\mathbf{B}\}$, $A(\mathbf{B}) = \varnothing$, $A(\mathbf{D}) = \varnothing$, $A(\mathbf{E}) = \{\mathbf{D}, \mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$ и $A(\mathbf{G}) = \varnothing$.

Шаг 3 Третий проход цикла **while**. Теперь у нас появился выбор: какой вершине присвоить очередную метку? В зависимости от нашего выбора, получатся разные последовательности меток. Присвоим, например, метку 3 вершине **B**, и удалим **B** из множеств $A(v)$. $A(\mathbf{A}) = \varnothing$, $A(\mathbf{D}) = \varnothing$, $A(\mathbf{E}) = \{\mathbf{D}, \mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$ и $A(\mathbf{G}) = \varnothing$.

Шаг 4 Четвертый проход цикла **while**. Мы снова стоим перед выбором. Назначим метку 4 вершине **A** и удалим вершину **A** из $A(v)$. $A(\mathbf{D}) = \varnothing$, $A(\mathbf{E}) = \{\mathbf{D}, \mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$ и $A(\mathbf{G}) = \varnothing$.

Шаг 5 Пятый проход цикла **while**. Назначим метку 5 вершине **D** и удалим вершину **D** из $A(v)$. $A(\mathbf{E}) = \{\mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$ и $A(\mathbf{G}) = \varnothing$.

Шаг 6 Шестой проход цикла **while**. Назначим метку 6 вершине **G** и удалим вершину **G** из $A(v)$. $A(\mathbf{E}) = \varnothing$, и $A(\mathbf{F}) = \varnothing$.

Шаг 7 Седьмой проход цикла **while**. Назначаем метку 7 вершине **E** и удаляем **E** из списка $A(v)$. Останется только $A(\mathbf{F}) = \varnothing$.

Шаг 8 Последний проход цикла **while**. Назначаем метку 8 вершине **F**.

Итак, один из возможных приоритетных списков: **H, C, B, A, D, G, E, F**. Он дает нам порядок, в котором можно изучать курсы, соблюдая должную последовательность.

8.2. Пути в орграфах

Ориентированные графы успешно применяются для схематичного изображения аэролиний, соединяющих города всего мира, или коммуникационных сетей между компьютерами. В таких сетях важно знать последовательность выключений любого соединения (дуги или вершины) по всей сети.

Например, если самолет не может приземлиться для дозаправки в некотором городе вследствие неблагоприятных погодных условий, то ошибка в его переадресации грозит катастрофой: ему может не хватить горючего для достижения неправильно назначенного аэропорта. Аналогично, если одна или несколько цепей в компьютерной сети не работают, то для некоторых пользователей отдельные серверы могут оказаться вообще недоступными.

Таким образом, мы приходим к задаче о поиске путей между произвольной парой вершин в ориентированном графе.

Пусть $G = (V, E)$ — орграф с n вершинами, а M — его матрица смежности. Напомним, что буквой И на пересечении i-той строки и j-го столбца мы обозначаем наличие дуги от вершины с номером i к вершине с номером j. Дуга, по определению, является путем длины 1. Булево произведение матрицы M с самой собой обозначается через M^2. В этой матрице буква И символизирует наличие пути длины 2. По матрице $M^3 = M \cdot M \cdot M$ можно определить все пути длины 3, и вообще, матрица M^k хранит сведения о путях длины k. Наконец, в *матрице достижимости*

$$M^* = M \textbf{ или } M^2 \textbf{ или } \ldots \textbf{ или } M^n$$

записаны пути любой длины между вершинами.

Если у нас есть две логические матрицы одного размера, то в результате логической операции **или** получится матрица, чьи элементы — результат применения этой операции к соответствующим элементам двух данных матриц. Более точно,

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2(n-1)} & a_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{m(n-1)} & a_{mn} \end{bmatrix} \textbf{ или } \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1(n-1)} & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2(n-1)} & b_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b_{m1} & b_{m2} & \ldots & b_{m(n-1)} & b_{mn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11} \textbf{ или } b_{11} & a_{12} \textbf{ или } b_{12} & \ldots & a_{1n} \textbf{ или } b_{1n} \\ a_{21} \textbf{ или } b_{21} & a_{22} \textbf{ или } b_{22} & \ldots & a_{2n} \textbf{ или } b_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} \textbf{ или } b_{m1} & a_{m2} \textbf{ или } b_{m2} & \ldots & a_{mn} \textbf{ или } b_{mn} \end{bmatrix}.$$

Матрица достижимости орграфа $G = (V, E)$ фактически является матрицей замыкания по транзитивности E^* отношения E на вершинах орграфа G.

Пример 8.3. Вычислите матрицу достижимости орграфа, представленного на рис. 8.3.

Решение. Прежде всего напишем матрицу смежности орграфа.

$$M = \begin{bmatrix} \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \end{bmatrix}.$$

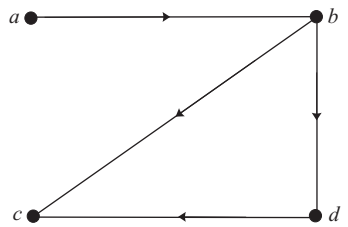


Рисунок 8.3.

Квадрат матрицы M равен

$$M^2 = \begin{bmatrix} Л & И & Л & Л \\ Л & Л & И & И \\ Л & Л & Л & Л \\ Л & Л & И & Л \end{bmatrix} \cdot \begin{bmatrix} Л & И & Л & Л \\ Л & Л & И & И \\ Л & Л & Л & Л \\ Л & Л & И & Л \end{bmatrix} = \begin{bmatrix} Л & Л & И & И \\ Л & Л & И & Л \\ Л & Л & Л & Л \\ Л & Л & Л & Л \end{bmatrix}.$$

Заметим, что буква И в матрице M^2 соответствует путям длины 2 в орграфе G, а именно: $a\,b\,c$, $a\,b\,d$ и $b\,d\,c$.

Дальнейшие вычисления приводят к третьей и четвертой степеням матрицы M.

$$M^3 = \begin{bmatrix} Л & Л & И & Л \\ Л & Л & Л & Л \\ Л & Л & Л & Л \\ Л & Л & Л & Л \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} Л & Л & Л & Л \\ Л & Л & Л & Л \\ Л & Л & Л & Л \\ Л & Л & Л & Л \end{bmatrix}.$$

Следовательно,

$$M^* = \begin{bmatrix} Л & И & И & И \\ Л & Л & И & И \\ Л & Л & Л & Л \\ Л & Л & И & Л \end{bmatrix}.$$

Отметим, например, что буква И в верхнем правом углу матрицы M^* появляется из матрицы M^2 и соответствует пути $a\,b\,d$.

Для больших орграфов вычисление матрицы M^* с помощью возведения M все в бóльшую степень утомительно и неэффективно. Более удобный путь определения M^* дает так называемый алгоритм Уоршелла.

Пусть $G = (V, E)$ — орграф с вершинами $v_1, v_2, \ldots, v_n$. Алгоритм Уоршелла генерирует последовательность матриц $W_0 = M$, $W_1, W_2, \ldots, W_n$, причем элемент матрицы W_k $(k \geqslant 1)$, стоящий на пересечении i-ой строки и j-го столбца $W_k(i, j)$, равен И в том и только том случае, когда существует путь (произвольной длины) из

вершины v_i в вершину v_j с внутренними вершинами из множества $\{v_1, v_2, \ldots, v_k\}$.

Матрица W_0 совпадает с матрицей смежности M орграфа, а W_n — искомая матрица достижимости M^*. Удачное использование цикла **for** придает алгоритму особенное изящество. Последовательные проходы этого цикла (пронумерованные индексом k) вычисляют матрицы $W_1, W_2, \ldots, W_n$.

Алгоритм Уоршелла.

Этот алгоритм вычисляет матрицу достижимости $W = M^*$ ориентированного графа $G = (V, E)$ с матрицей смежности M.

```
begin
    W := M;
    for k = 1 to n do
        for i = 1 to n do
            for j = 1 to n do
                W(i, j) = W(i, j) или (W(i, k) и W(k, j));
end
```

Для лучшего понимания работы алгоритма Уоршелла, необходимо с его помощью вручную вычислить матрицу достижимости какого-нибудь орграфа. С этой целью полезно более подробно рассказать о шагах алгоритма.

За каждый проход цикла (пронумерованный индексом k) алгоритм Уоршелла генерирует матрицу W_k, используя элементы предыдущей матрицы W_{k-1}.

Чтобы найти i-ую строку матрицы W_k, нам следует вычислить выражения:

$$W_{k-1}(i, j) \text{ **или** } \big(W_{k-1}(i, k) \text{ **и** } W_{k-1}(k, j)\big) \qquad (1)$$

при разных значениях j.

Если $W_{k-1}(i, k) = \text{Л}$, то $\big(W_{k-1}(i, k)$ **и** $W_{k-1}(k, j)\big) = \text{Л}$, и значение выражения (1) совпадает со значением $W_{k-1}(i, j)$. Иначе говоря, *i-ая строка матрицы остается неизменной*. В том же случае, когда $W_{k-1}(i, k) = \text{И}$, вычисление выражения (1) сводится к вычислению $\big(W_{k-1}(i, k)$ **и** $W_{k-1}(k, j)\big)$. При этом *i-ая строка получается с помощью логической операции* **или** *из текущей строки i и текущей строки k*. Говоря более аккуратно, при вычислении W_k поступают следующим образом.

1. Берем k-ый столбец матрицы W_{k-1}.

2. Строку с номером i ($i = 1, \ldots, n$), у которой на k-ом месте стоит Л, переписываем в i-ую строку матрицы W_k.

3. Строку с номером i ($i = 1, \ldots, n$), у которой на k-ом месте стоит И, спариваем с помощью операции **или** с k-ой строкой, а результат записываем в i-ую строку матрицы W_k.

Пример 8.4. С помощью алгоритма Уоршелла вычислите матрицу достижимости орграфа, изображенного на рис. 8.4.

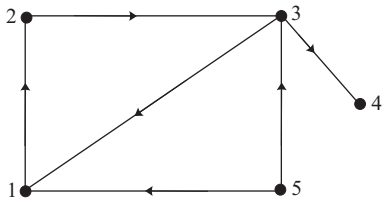


Рисунок 8.4.

Решение. Матрица W_0 совпадает с матрицей смежности данного орграфа.

$$W_0 = \begin{bmatrix} Л & И & Л & Л & Л \\ Л & Л & И & Л & Л \\ И & Л & Л & И & Л \\ Л & Л & Л & Л & Л \\ И & Л & И & Л & Л \end{bmatrix}.$$

Теперь вычисляем W_1. Учитывая первый шаг, мы рассматриваем 1-ый столбец матрицы W_0. Следуя указаниям шага 2, скопируем строки матрицы W_0 с номерами 1, 2 и 4 в матрицу W_1 на те же места. Таким образом,

$$W_1 = \begin{bmatrix} Л & И & Л & Л & Л \\ Л & Л & И & Л & Л \\ ? & ? & ? & ? & ? \\ Л & Л & Л & Л & Л \\ ? & ? & ? & ? & ? \end{bmatrix}.$$

Далее, согласно шагу 3, строка с номером 3 матрицы W_1 получается с помощью логической операции **или** из 1-ой и 3-ей строк матрицы W_0. Поэтому

$$W_1 = \begin{bmatrix} Л & И & Л & Л & Л \\ Л & Л & И & Л & Л \\ И & И & Л & И & Л \\ Л & Л & Л & Л & Л \\ ? & ? & ? & ? & ? \end{bmatrix}.$$

Опять применяем шаг 3 алгоритма для вычисления 5-ой строки матрицы W_1 с помощью операции **или**, примененной к 5-ой и 1-ой строкам матрицы W_0. Получаем

$$W_1 = \begin{bmatrix} Л & И & Л & Л & Л \\ Л & Л & И & Л & Л \\ И & И & Л & И & Л \\ Л & Л & Л & Л & Л \\ И & И & И & Л & Л \end{bmatrix}.$$

Матрица W_1 вычислена. Теперь строим матрицу W_2, по матрице W_1. Взгляд на 2-ой столбец матрицы W_1 показывает, что строки с номерами 2 и 4 копируются в W_2. Первая строка матрицы W_2 — результат применения операции **или** к 1-ой и 2-ой строкам из W_1. Третья строка в W_2 получается спариванием 3-ей и 2-ой строк матрицы W_1. И, наконец, пятая строка W_2 — результат логической операции **или**, примененной к 5-ой и 2-ой строкам W_1. Значит,

$$W_2 = \begin{bmatrix} Л & И & И & Л & Л \\ Л & Л & И & Л & Л \\ И & И & И & И & Л \\ Л & Л & Л & Л & Л \\ И & И & И & Л & Л \end{bmatrix}.$$

Отметим, в частности, что на пересечении 3-ей строки и 3-го столбца матрицы W_2 стоит буква И. Это означает, что существует контур, начинающийся и заканчивающийся в вершине 3, проходящий через одну или обе вершины с номерами 1 и 2. Посмотрев на изображение орграфа (рис. 8.4), убеждаемся, что действительно существует контур длины 3: 3 1 2 3.

Аналогичным образом вычисляется матрица W_3.

$$W_3 = \begin{bmatrix} И & И & И & И & Л \\ И & И & И & И & Л \\ И & И & И & И & Л \\ Л & Л & Л & Л & Л \\ И & И & И & И & Л \end{bmatrix}.$$

Поскольку из вершины 4 не выходит ни одной дуги, то мы не сможем построить ни одного пути, проходящего через вершину 4. Следовательно, матрица W_4 совпадает с W_3. Кроме того, в орграфе отсутствуют дуги, ведущие в вершину 5. Значит нет и путей, через нее проходящих, т. е. $W_5 = W_4$. Наконец, $W_5 = M^*$, поскольку граф состоит только из пяти вершин.

8.3. Кратчайший путь

Чтобы подготовить технику для решения задач из приложения к этой главе, мы рассмотрим задачу поиска кратчайшего пути, связывающего пару данных вершин в нагруженном орграфе. Слово «кратчайший» здесь вполне уместно, поскольку довольно часто веса в орграфе — это расстояния между пунктами.

Типичная ситуация, которая моделируется нагруженным орграфом, это транспортная сеть (по которой товары доставляются из города в город) и коммуникационная сеть (по которой переправляется информация).

Рассмотрим *нагруженный граф* на рис. 8.5. Он может представлять, например, длины дорог в милях между шестью деревнями. Поскольку количество вершин в этом графе невелико, то перебрать все возможные пути между любой парой заданных вершин нам вполне по силам. При этом, естественно, мы найдем наиболее короткий путь, соединяющий соответствующие деревни. В реальной задаче, возникающей в профессиональной деятельности, число вершин, как правило, настолько велико, что такой упрощенный подход к поиску кратчайшего пути слишком неэффективен.

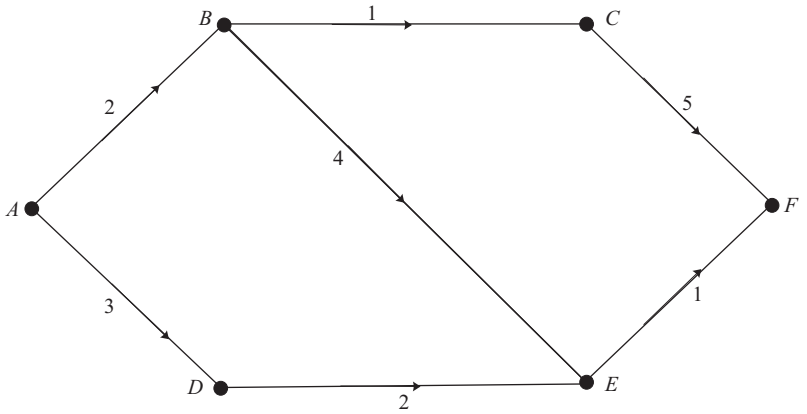


Рисунок 8.5. Нагруженный граф

Существует множество алгоритмов поиска кратчайшего пути, но мы познакомимся только с одним, *алгоритмом Дейкстры*. Перед формальным изложением алгоритма мы опишем его действия и проиллюстрируем работу алгоритма на нашем примере. Допустим, что нужно найти кратчайший путь от вершины A к любой другой вершине орграфа (см. рис. 8.5). *Кратчайший путь* — это путь минимального общего веса, соединяющий выбранные вершины. Об-

щий вес, по определению, равен сумме весов всех дуг, составляющих путь. Общий вес кратчайшего пути, ведущего из вершины u в вершину v, называют *расстоянием* от u до v.

Определим *весовую матрицу* w, чьи элементы $w(u, v)$ задаются формулой

$$w(u, v) = \begin{cases} 0, & \text{если } u = v, \\ \infty, & \text{если } u \text{ и } v \text{ не соединены дугой}, \\ d, & \text{если } uv \text{ — дуга веса } d. \end{cases}$$

Для нашего графа весовая матрица выглядит следующим образом:

$$w = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \left[\begin{array}{cccccc} 0 & 2 & \infty & 3 & \infty & \infty \\ \infty & 0 & 1 & \infty & 4 & \infty \\ \infty & \infty & 0 & \infty & \infty & 5 \\ \infty & \infty & \infty & 0 & 2 & \infty \\ \infty & \infty & \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & \infty & \infty & 0 \end{array}\right] \end{array}.$$

В течение работы алгоритма каждой вершине v орграфа присваивается число $d[v]$, равное расстоянию от вершины A до v. Перед началом работы $d[v]$ совпадает с весом дуги (A, v), если такая существует, или равно ∞ в противном случае. Мы будем проходить вершины орграфа и уточнять значения $d[v]$.

На каждом шагу алгоритма отмечается одна вершина u, до которой уже найден кратчайший путь от A и расстояние $d[u]$ до нее. Далее полученное значение $d[u]$ отмеченной вершины не меняется. Для оставшихся, неотмеченных вершин v, число $d[v]$ будет меняется с учетом того, что искомый кратчайший путь до них от A будет проходить через последнюю отмеченную вершину u. Алгоритм завершится в тот момент, когда все возможные вершины будут отмечены и получат свои окончательные значения $d[v]$.

Для каждого шага алгоритма, описанного ниже, в соответствующую строку табл. 8.2 заносится отмеченная вершина, текущие значения $d[v]$ и оставшиеся неотмеченные вершины. При этом полужирным шрифтом выделяется наименьшее из значений $d[v]$ среди неотмеченных вершин. Соответствующую вершину следует отметить. Кроме того, в таблице все значения $d[v]$ для уже отмеченных вершин отделены от остальных ломаной линией.

Таблица 8.2

Шаг	Отмеченные вершины	Расстояние до вершины						Неотмеченные вершины
		A	B	C	D	E	F	
0	A	0	**2**	∞	3	∞	∞	B, C, D, E, F
1	B	0	2	3	**3**	6	∞	C, D, E, F
2	D	0	2	**3**	3	5	∞	C, E, F
3	C	0	2	3	3	**5**	8	E, F
4	E	0	2	3	3	5	**6**	F
5	F	0	2	3	3	5	6	

Шаг 0 Поскольку мы интересуемся кратчайшими путями от вершины A, мы ее отмечаем и используем первую строку весовой матрицы w для определения начальных значений $d[v]$. Таким образом получается первая строка таблицы. Наименьшее число из всех $d[v]$ для неотмеченных вершин — это $d[B] = 2$.

Шаг 1 Отмечаем вершину B, так как она является ближайшей к A. Вычисляем длины путей, ведущих от A к неотмеченным вершинам через вершину B. Если новые значения $d[v]$ оказываются меньше старых, то меняем последние на новые. Итак, при этом проходе цикла путь $A\,\boldsymbol{B}\,C$ имеет вес 3, а путь $A\,\boldsymbol{B}\,E$ — 6, в то время как старые расстояния до этих вершин от A были ∞. Следовательно, заполняя вторую строку таблицы, мы заменим $d[C]$ на 3 и $d[E]$ на 6.

Шаг 2 Из оставшихся неотмеченными, вершины C и D находятся ближе всех к A. Отметить можно любую из них. Возьмем вершину D. Так как длина пути $A\,\boldsymbol{D}\,E$ равна 5, текущее значение $d[E]$ следует уменьшить до 5. Теперь можно заполнить третью строчку таблицы. Наименьшее значение $d[v]$ среди неотмеченных к этому моменту вершин оказывается у вершины C.

Шаг 3 Отмечаем вершину C и подправляем значения $d[v]$. Теперь можно дойти и до вершины F, следуя путем $A\,B\,\boldsymbol{C}F$. Его длина, а стало быть и значение $d[F]$, равны 8. К этому моменту остались неотмеченными две вершины: E и F.

Шаг 4 Мы отмечаем вершину E, что позволяет нам уменьшить величину $d[F]$ с 8 до 6.

Шаг 5 Отмечаем вершину F.

Алгоритм Дейкстры.

Пусть (V, E) — нагруженный граф, и A — его вершина. Алгоритм выбирает кратчайший путь от вершины A до любой вершины v и присваивает его длину переменной $d[v]$. Для вершин u и v через $w(u, v)$ мы обозначаем вес дуги uv, а в списке PATHTO(v) перечисляются вершины кратчайшего пути от A до v.

```
begin
    for каждой v ∈ V do
        begin
            d[v] := w(A, v);
            PATHTO(v) := A;
        end
    Отметить вершину A;
    while остаются неотмеченные вершины do
        begin
            u := неотмеченную вершину с минимальным
                 расстоянием от A;
            Отметить вершину u;
            for каждой неотмеченной вершины v
                с условием uv ∈ E do
                 begin
                 d' := d[u] + w(u, v)
                      if d' < d[v] then
                           begin
                           d[v] := d';
                           PATHTO(v) := PATHTO(u), v;
                           end
                 end
        end
end
```

Набор упражнений 8

8.1. Изобразите орграф с вершинами $\{1, 2, 3, 4, 5, 6\}$ и матрицей смежности

$$\begin{bmatrix} И & Л & Л & И & Л & Л \\ И & Л & И & Л & Л & И \\ Л & И & И & Л & И & Л \\ И & Л & Л & И & И & И \\ Л & Л & Л & И & Л & И \\ Л & И & И & И & И & Л \end{bmatrix}.$$

Предположите, что вес каждой дуги равен 1 и найдите (если он существует)

(а) кратчайший путь (пути) от вершины 1 до вершины 2;

(б) кратчайший путь (пути) от вершины 3 до вершины 6;

(в) контур длины 5.

8.2. *Полустепенью исхода* вершины v орграфа G называется число дуг $\delta^+(v)$ орграфа, исходящих из v, а *полустепенью захода* этой вершины называют число дуг $\delta^-(v)$, заходящих в нее.

Объясните, почему обе суммы — полустепеней исхода и полустепеней захода всех вершин орграфа — совпадают с числом его дуг.

Что можно сказать о числе букв И в любой строке матрицы смежности орграфа? А как проинтерпретировать их число в любом столбце?

8.3. *Связным* называется такой орграф, из которого получается связный граф, если забыть про ориентацию дуг. С другой стороны, если для любой упорядоченной пары его вершин существует путь, ведущий из первой во вторую, то такой орграф называют *сильно связным*.

(а) Определите, какие из связных орграфов, представленных на рис. 8.6, являются сильно связными.

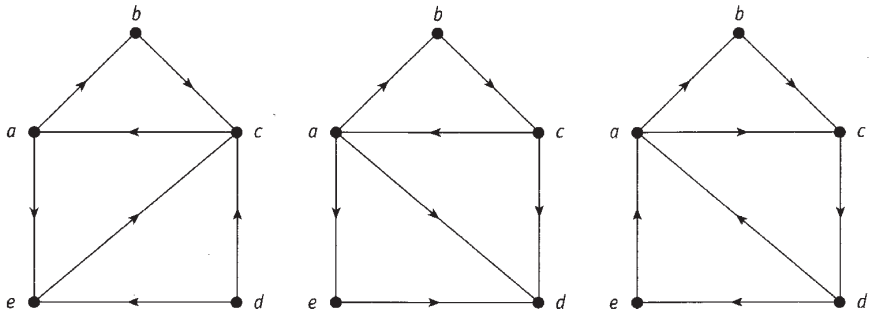


Рисунок 8.6. Связные орграфы

(б) Объясните, как нужно ориентировать ребра гамильтонова графа (т. е. нарисовать на каждом ребре стрелку, превратив ее в дугу), чтобы из него получился сильно связный орграф.

(в) Объясните важность требования: орграф, представляющий систему односторонних дорог в городе, должен быть сильно связным.

8.4. Примените алгоритм топологической сортировки к (ациклличному) орграфу со следующей матрицей смежности:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \left[\begin{array}{cccccc} \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{Л} \end{array}\right] \end{array}.$$

Напишите новую матрицу смежности, строки и столбцы которой упорядочены в соответствии с новыми обозначениями вершин. Что можно сказать о новой матрице?

Что можно ожидать от алгоритма топологической сортировки в случае орграфа из упр. 8.1?

8.5. В табл. 8.3 приведен список действий по приготовлению цыпленка с расставленными приоритетами. Упорядочите список согласно приоритетам.

Таблица 8.3

Задания		Предварительные действия
А	Добавить лук к цыпленку	И
Б	Вымыть салат-латук	Л
В	Приготовить салатную заправку	Л
Г	Перемешать жаркое	К
Д	Перемешать салат	Б, В
Е	Разрезать цыпленка	никаких
Ж	Растереть имбирь	И
З	Подать готовое блюдо	И
И	Замариновать цыпленка	Е
К	Поставить казанок на огонь	А, Ж, З, Л
Л	Приготовить рис	никаких

8.6. Пусть $M = M(i, j)$ — матрица смежности орграфа G с множеством вершин $V = \{1, 2, 3, \ldots, n\}$. Напишите на псевдокоде алгоритм, вычисляющий множество антецедентов $A(v)$ вершины $v \in V$.

8.7. Матрица смежности орграфа G имеет вид

$$M = \begin{bmatrix} \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix}.$$

Вычислите M^2, M^3 и M^4. Найдите матрицу достижимости M^*.

8.8. С помощью алгоритма Уоршелла вычислите W_1, W_2, W_3 и W_4 для орграфа G из предыдущего упражнения, после чего найдите матрицу достижимости M^*.

8.9. Проследите за работой алгоритма Дейкстры на примере орграфа, изображенного на рис. 8.7, и найдите кратчайшие пути до каждой вершины

(а) от вершины A;

(б) от вершины C.

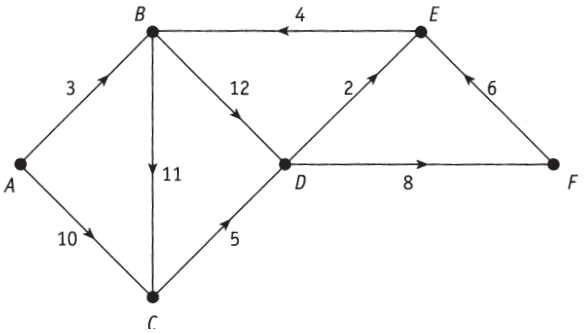


Рисунок 8.7.

8.10. С помощью алгоритма Дейкстры найдите кратчайший путь от вершины S до всех остальных вершин в нагруженном графе из рис. 8.8. Найдите два кратчайших пути от S до T.

Краткое содержание главы

Ориентированным графом или **орграфом** называют пару $G = (V, E)$, где V — конечное множество **вершин**, а E — отношение на V. Элементы множества E называют **дугами** орграфа.

Если uv — дуга орграфа, то вершину u называют **антецедентом** v.

Путем длины k в орграфе называют такую последовательность различных вершин v_0, v_1, ..., v_k, в которой каждая пара $v_{i-1}v_i$ образует дугу $(i = 1, \ldots, k)$.

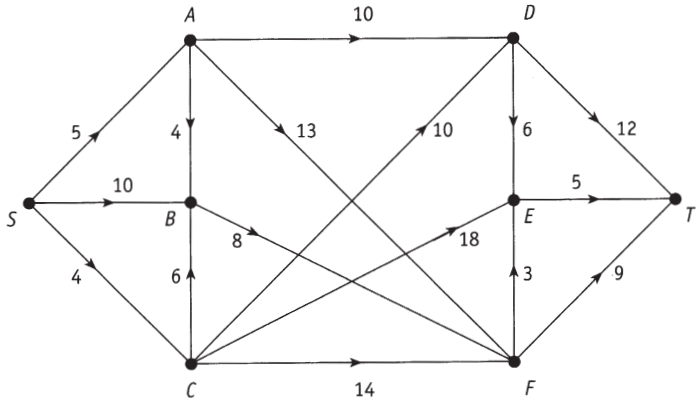


Рисунок 8.8.

Контуром в орграфе G принято называть последовательность вершин v_0, v_1, ..., v_k, образующую путь, в которой первая v_0 совпадает с последней, а других повторяющихся вершин в ней нет.

Орграф называют **бесконтурным**, если в нем нет контуров. В задаче о планировании заданий соответствующий бесконтурный орграф называют системой **ПЕРТ**.

Последовательность согласованных меток бесконтурного орграфа $G = (V, E)$ — это метки: 1, 2, 3, ..., n вершин, причем если uv — дуга орграфа, соединяющая вершину u с меткой i и вершину v с меткой j, то $i < j$.

Для орграфа можно выписать последовательность согласованных меток тогда и только тогда, когда он не имеет контуров. **Алгоритм топологической сортировки** генерирует последовательность согласованных меток бесконтурного орграфа.

Пусть $G = (V, E)$ — орграф с n вершинами и матрицей смежности M. Логическая степень M^k матрицы смежности хранит информацию о существовании путей длины k между произвольной парой вершин орграфа G. Матрица

$$M^* = M \textbf{ или } M^2 \textbf{ или } \ldots \textbf{ или } M^n$$

называется **матрицей достижимости** орграфа. В ней записана информация о существовании путей между вершинами.

Алгоритм Уоршелла используется для вычисления матрицы достижимости орграфа. Алгоритм генерирует последовательность матриц W_0, W_1, W_2, ..., W_n, где $W_0 = M$, $W_n = M^*$ а для любого $k \geqslant 1$ матрица W_k строится по W_{k-1} следующим образом:

1. Берем k-ый столбец матрицы W_{k-1}.
2. Каждую строчку, у которой на k-ом месте стоит Л, переписываем в соответствующую строку матрицы W_k.
3. Каждую строчку, у которой на k-ом месте стоит И, спариваем с помощью операции **или** с k-ой строкой, а результат записываем в соответствующую строку матрицы W_k.

Кратчайшим путем между произвольной парой вершин в нагруженном орграфе называется путь наименьшего общего веса. Общий вес кратчайшего пути, ведущего от вершины u к v, называется **расстоянием** от u до v. Если пути от u до v не существует, то расстояние между ними принято считать бесконечным и обозначать символом ∞.

Алгоритм Дейкстры определяет кратчайшие пути в нагруженном графе от данной вершины до любой другой.

Приложение. Коммуникационные сети

Удачной моделью компьютерной сети может служить ориентированный граф, чьи вершины (или *узлы*) представляют компьютерные компоненты, а дуги — коммуникационные линии связи. Каждая дуга такого графа снабжена весом, обозначающим пропускную способность соответствующей линии.

Рисунок 8.9 показывает, например, простую сеть из семи узлов.

Сразу после ввода в действие сети возникает вопрос о том, как передавать сообщения между несмежными узлами. Процедура *статической маршрутизации* учитывает информацию о пропускной способности линий для определения фиксированного пути передач между узлами. В целях оптимизации таких путей в сети применяют алгоритм, подобный алгоритму Дейкстры. Однако при этом подходе могут возникать сбои в линиях или узлах сети. Задержки передач могут происходить в тех случаях, когда превышается пропускная способность линии.

Процедура *Динамической маршрутизации* постоянно корректирует пропускную способность линий с учетом потребности. Чтобы дать возможность индивидуальным узлам решать, когда и куда передавать новую информацию, разработан *протокол* или множество правил. Каждый узел поддерживает свою таблицу путей, так что задача оптимизации передачи сообщений *рассредоточена* по всей сети.

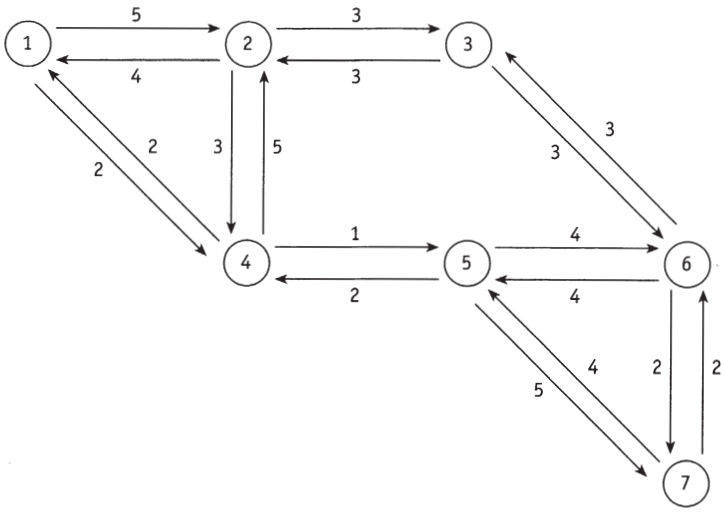


Рисунок 8.9. Семиузловая сеть

Каждый узел сети, изображенной на рис. 8.9, прогоняет алгоритм Дейкстры для определения наилучших путей к другим узлам и распространяет эту информацию по дереву, чей корень соответствует «домашнему узлу». Например, для узла 1 соответствующее дерево показано на рис. 8.10.

Реально, для передачи сообщений любому узлу требуется таблица, в которой указаны ближайшие соседи для передачи сообщения тому или иному адресату[1]. Такая таблица, относящаяся к узлу 1, приведена ниже (табл. 8.4).

Таблица 8.4

Адресат	2	3	4	5	6	7
Следующий узел	2	2	4	4	4	4

[1]Таблицу ближайших соседей в этом контексте называют таблицей маршрутов. — *Прим. перев.*

Задача 1. Используя алгоритм Дейкстры, найдите кратчайшие пути от узла 2 к любому другому по сети, о которой шла речь выше. После этого изобразите дерево кратчайших путей и заполните таблицу маршрутов для узла 2.

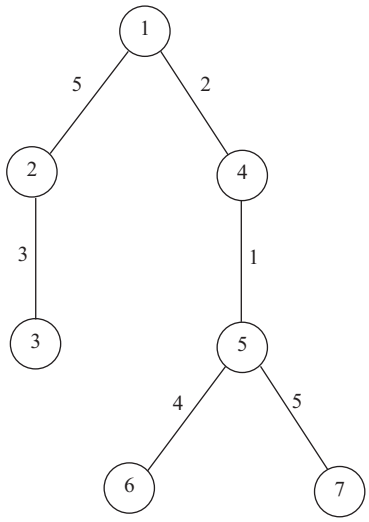


Рисунок 8.10. Дерево передачи информации узла 1

Решение. Смотри табл. 8.5.

Таблица 8.5

Отмеченные вершины	Расстояние до вершины							Неотмеченные вершины
	1	2	3	4	5	6	7	
2	4	0	**3**	3	∞	∞	∞	1, 3, 4, 5, 6, 7
3	4	0	3	**3**	∞	6	∞	1, 4, 5, 6, 7
4	4	0	3	3	**4**	6	∞	1, 5, 6, 7
5	**4**	0	3	3	4	6	9	1, 6, 7
1	4	0	3	3	4	**6**	9	6, 7
6	4	0	3	3	4	6	**8**	7
7	4	0	3	3	4	6	8	

Кроме того, алгоритм Дейкстры определяет кратчайшие пути от вершины 2 к любой другой. Например, РАТНТО(6) = 2, 3, 6.

Дерево кратчайших путей и таблица маршрутов показаны на рис. 8.11 и табл. 8.6 соответственно.

Таблица 8.6

Адресат	1	3	4	5	6	7
Следующий узел	1	3	4	4	3	3

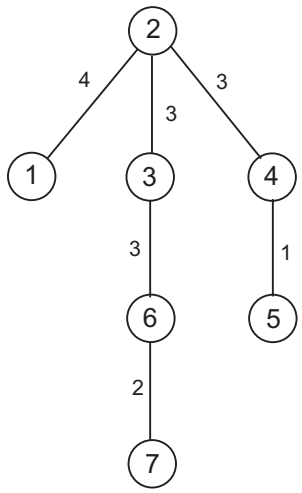


Рисунок 8.11.

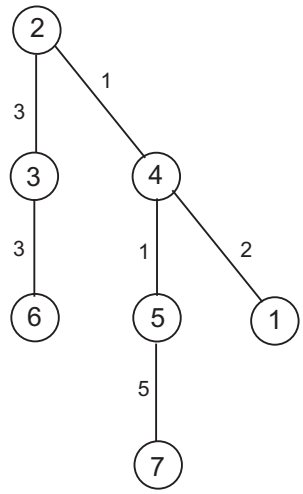


Рисунок 8.12.

Задача 2. Предположим, что временна́я задержка передачи от узла 2 к узлу 4 уменьшилась с 3 до 1. Как изменятся при этом дерево кратчайших путей и таблица маршрутов для узла 2?

Решение. Перезапустим алгоритм Дейкстры, установив временну́ю задержку на линии 2 4, равную 1. Это изменение дает нам несколько новых деревьев кратчайших путей. Одно из них приведено на рис. 8.12.

Соответствующая таблица маршрутов — табл. 8.7.

Таблица 8.7

Адресат	1	3	4	5	6	7
Следующий узел	4	3	4	4	3	4

Задача 3. Какими будут дерево кратчайших путей и таблица маршрутов для узлов 1 и 2, если удалить линии между узлами 5 и 6?

Решение. Поскольку линия 5 6 не задействована при передаче информации от узла 2, то его дерево кратчайших путей и таблица маршрутов, найденные в задаче 1, останутся без изменений.

Что касается узла 1, то мы можем ограничиться поиском кратчайшего пути от узла 1 к узлу 6. Алгоритм Дейкстры находит такой путь без особых затруднений: PATHTO(6) = 1, 4, 5, 7, 6.

Новое дерево кратчайших путей начерчено на рис. 8.13.

Таблица 8.8 — соответствующая таблица маршрутов.

Таблица 8.8

Адресат	2	3	4	5	6	7
Следующий узел	2	2	4	4	4	4

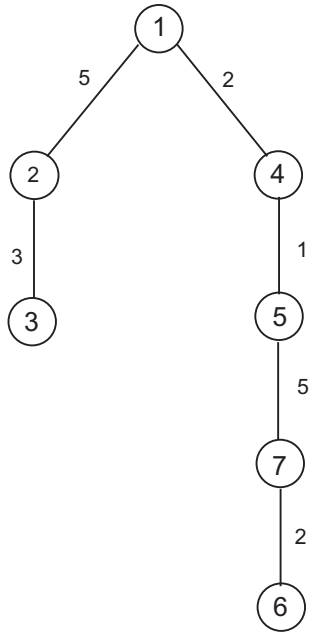


Рисунок 8.13.

ГЛАВА 9

БУЛЕВА АЛГЕБРА

Булева алгебра — это название области математики, занимающейся логическим анализом. Операции и законы булевой алгебры применяются к логическим символам так же, как обычная алгебра оперирует символами, представляющими численные величины.

В настоящей главе мы изучаем булеву алгебру, а именно множество $\{0, 1\}$ с определенными на нем операциями дизъюнкции, конъюнкции и отрицания. Здесь будут сформулированы законы булевой алгебры и проведена параллель между булевой алгеброй, логикой высказываний (см. главу 2) с одной стороны, и с алгеброй множеств (глава 3) с другой. Мы покажем, как булевы выражения могут быть записаны в стандартной форме, носящей название «дизъюнктивная нормальная форма». После этого здесь будет описан способ, называемый «карта Карно», который применяется для упрощения булевых выражений.

Последний параграф главы демонстрирует, как вся развитая теория применяется к конструированию и упрощению логических схем. Такие схемы встречаются в электронных устройствах, используемых в компьютерах, калькуляторах, телефонных системах и ряде других устройств. Указанную тему продолжает приложение к главе. В нем, опираясь на булеву алгебру, мы будем моделировать 2-битный сумматор.

9.1. Булева алгебра

Простейшая булева алгебра состоит из множества $B = \{0, 1\}$ вместе с определенными на нем операциями *дизъюнкции* ($\vee$), *конъюнкции* ($\wedge$) и *отрицания* ($\bar{}$). Действие операций ($\vee$) и ($\wedge$) на символах 0 и 1 показаны на рис. 9.1.

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

Рисунок 9.1.

Действие отрицания на 0 и 1 определяется следующим образом: $\bar{0} = 1$ и $\bar{1} = 0$.

Для *булевых переменных* p и q (т. е. переменных, принимающих значения 0 и 1) можно построить таблицы, определяющие действия операций $\bar{p}$, $p \vee q$ и $p \wedge q$ (см. табл. 9.1 и табл. 9.2).

Таблица 9.1

p	$\bar{p}$
0	1
1	0

Таблица 9.2

p	q	$p \vee q$	$p \wedge q$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Эти таблицы напоминают таблицы истинности логических операций **не**, **или** и **и**, с которыми мы встретились в главе 2. Действительно, мы можем легко трансформировать табл. 9.1 и табл. 9.2 в таблицы истинности, возникающие в логике высказываний, заменив переменные p и q на высказывания P и Q, и используя истинностные значения Л и И вместо 0 и 1 соответственно. Таким образом, $\bar{p}$ заменится на (**не** P), $p \vee q$ — на (P **или** Q), а $p \wedge q$ — на (P **и** Q). Поэтому и в контексте булевой алгебры мы будем называть такого сорта таблицы таблицами истинности.

Мы можем комбинировать булевы переменные с помощью операций $(\vee)$, $(\wedge)$ и $(\bar{\ })$, получая *булевы выражения*, так же, как мы строили составные высказывания из более простых, компануя их с помощью логических операций.

Таблица 9.3. Законы булевой алгебры

Законы коммутативности:	$p \wedge q = q \wedge p$,
	$p \vee q = q \vee p$;
Законы ассоциативности:	$p \wedge (q \wedge r) = (p \wedge q) \wedge r$,
	$p \vee (q \vee r) = (p \vee q) \vee r$;
Законы дистрибутивности:	$p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$,
	$p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$;
Законы идемпотентности:	$p \wedge p = p$,
	$p \vee p = p$;
Законы поглощения:	$p \wedge (p \vee q) = p$,
	$p \vee (p \wedge q) = p$;
Законы де Моргана:	$\overline{(p \wedge q)} = \bar{p} \vee \bar{q}$,
	$\overline{(p \vee q)} = \bar{p} \wedge \bar{q}$.

Два булевых выражения называются *эквивалентными*, если они имеют одинаковые таблицы истинности. Вычисляя таблицы истинности, легко установить справедливость некоторых эквивалентностей, которые принято называть законами булевой алгебры (см. таблицу на предыдущей стр.).

Пример 9.1. Докажите закон дистрибутивности:

$$p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r).$$

Решение. Требуемые таблицы истинности приведены в табл. 9.4. Поскольку два последних столбца таблицы полностью совпадают, булевы выражения $p \wedge (q \vee r)$ и $(p \wedge q) \vee (p \wedge r)$ эквивалентны.

Таблица 9.4

p	q	r	$p \wedge q$	$p \wedge r$	$q \vee r$	$p \wedge (q \vee r)$	$(p \wedge q) \vee (p \wedge r)$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

Схожесть названий и форм законов булевой алгебры и соответствующих законов алгебры множеств, изученных в главе 3, далеко не случайна. В таблице, приведенной ниже, устанавливается соответствие между булевыми операциями, логическими операторами логики высказываний и операциями над множествами.

Таблица 9.5

Логические операторы	Операции над множествами	Булевы операции
не	$-$	$-$
или	$\cup$	$\vee$
и	$\cap$	$\wedge$

Стоит раз и навсегда убедиться в справедливости законов булевой алгебры[1], чтобы в дальнейшем доказывать эквивалентность булевых выражений с их помощью, не обращаясь к таблицам истинности. Это можно делать так же, как мы проверяли равенства множеств, опираясь на законы алгебры множеств.

[1] Попытайтесь сделать это самостоятельно. — *Прим. перев.*

Пример 9.2. Покажите, что булево выражение $\overline{(\bar{p} \wedge q)} \wedge (p \vee q)$ эквивалентно p.

Решение. Сделаем несложные преобразования.

$$\begin{aligned}
\overline{(\bar{p} \wedge q)} \wedge (p \vee q) &= \left(\overline{(\bar{p})} \vee \bar{q}\right) \wedge (p \vee q) = && \text{(закон де Моргана)} \\
&= (p \vee \bar{q}) \wedge (p \vee q) = && \text{(так как } \overline{(\bar{p})} = p\text{)} \\
&= p \vee (\bar{q} \wedge q) = && \text{(закон дистрибутивности)} \\
&= p \vee 0 = && \text{(так как } \bar{q} \wedge q = 0\text{)} \\
&= p && \text{(по определению } \vee\text{).}
\end{aligned}$$

Булевой функцией от n булевых переменных $p_1, p_2, \ldots, p_n$ называется такая функция $f : B^n \longrightarrow B$, что $f(p_1, p_2, \ldots, p_n)$ — булево выражение.

Наша ближайшая задача — показать, как произвольную булеву функцию можно записать в стандартном виде (он называется *дизъюнктивной нормальной формой*). Начнем с небольшого примера. Рассмотрим булеву функцию $m(p, q, r)$ от булевых переменных p, q и r, чья таблица истинности дана ниже (табл. 9.6).

Таблица 9.6

p	q	r	m
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Функция m — пример *минтерма*, т. е. булевой функции, которая принимает значение 1 только на одном наборе значений аргументов.

Так как $m(p, q, r) = 1$ только если $p = 0$, $q = 1$ и $r = 1$, то[2]

$$m(p, q, r) = \bar{p} \wedge q \wedge r.$$

Выражение $\bar{p} \wedge q \wedge r$ называют *элементарной конъюнкцией*.

[2] Действительно, по определению конъюнкции функция $p \wedge q \wedge r$ принимает значение 1 только тогда, когда $p = q = r = 1$. Поэтому

$$\bar{p} \wedge q \wedge r = 1 \Leftrightarrow p = 0,\ q = r = 1.$$

Пример 9.3. Объясните, каким образом любой минтерм можно записать в виде элементарной конъюнкции, т. е. как конъюнкцию переменных p_i или их отрицаний. $m(p_1, p_2, \ldots, p_r)$

Решение. Пусть $m(p_1, p_2, \ldots, p_r)$ — минтерм. Тогда в последнем столбце таблицы истинности функции m будет стоять только одна единица. Возьмем строку таблицы истинности, последний символ в которой — 1. Если в этой строке переменная $p_i = 1$, то в элементарной конъюнкции, представляющей функцию m, участвует p_i; а если $p_i = 0$, то участвует $\bar{p}_i$.

Например, для минтерма из табл. 9.6 такой строкой является последовательность 0, 1, 1, 1. Это означает, что $m(0, 1, 1) = 1$. Поэтому $m(p, q, r) = \bar{p} \wedge q \wedge r$.

Теперь, используя элементарные конъюнкции, мы запишем произвольную булеву функцию как дизъюнкцию минтермов. Более того, можно доказать, что такая запись (она называется *дизъюнктивной нормальной формой*) для каждой функции определена единственным образом с точностью до перестановки элементарных конъюнкций.

Рассмотрим булеву функцию трех переменных $f(p, q, r)$, чья таблица истинности — табл. 9.7. Единицы последнего столбца в этой таблице соответствуют трем минтермам:

$$\bar{p} \wedge \bar{q} \wedge r, \qquad \bar{p} \wedge q \wedge r, \qquad p \wedge \bar{q} \wedge \bar{r}.$$

Таблица истинности функции f может быть получена наложением таблиц истинности выписанных минтермов.

Таблица 9.7

p	q	r	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Поскольку дизъюнкция с 1 «поглощает» все нули (иными словами, $f_1 \vee f_2 \vee \cdots \vee f_s$ равно 1 тогда и только тогда, когда среди

значений f_i найдется хотя бы одна 1), то наша функция f равна дизъюнкции трех минтермов:

$$f(p,\, q,\, r) = (\bar{p} \wedge \bar{q} \wedge r) \vee (\bar{p} \wedge q \wedge r) \vee (p \wedge \bar{q} \wedge \bar{r}).$$

Это и есть нормальная дизъюнктивная форма функции f. Очевидно, в той же форме можно записать произвольную булеву функцию с любым числом переменных.

Пример 9.4. Найдите дизъюнктивную нормальную форму булевой функции $f = (p \wedge q) \vee (\bar{q} \wedge r)$.

Решение. В табл. 9.8 запишем таблицу истинности функции f.

Таблица 9.8

p	q	r	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Выпишем минтермы:

$$\bar{p} \wedge \bar{q} \wedge r, \qquad p \wedge \bar{q} \wedge r, \qquad p \wedge q \wedge \bar{r}, \qquad p \wedge q \wedge r.$$

Следовательно, искомая дизъюнктивная нормальная форма:

$$f(p,\, q,\, r) = (\bar{p} \wedge \bar{q} \wedge r) \vee (p \wedge \bar{q} \wedge r) \vee (p \wedge q \wedge \bar{r}) \vee (p \wedge q \wedge r).$$

Мы убедились на опыте, что любую булеву функцию можно единственным образом представить в виде дизъюнкции минтермов. Значит, каждая булева функция может быть выражена через две функции от двух аргументов: $p \vee q$, $p \wedge q$ и одной функции одной переменной $\bar{p}$. Множество функций, через которые можно выразить любую булеву функцию, называется *полной системой функций*. Итак, $\{p \vee q,\ p \wedge q,\ \bar{p}\}$ — полная система функций. Однако можно ограничиться и меньшим количеством функций. Например, по закону де Моргана $\overline{(p \vee q)} = \bar{p} \wedge \bar{q}$. Следовательно,

$$p \vee q = \overline{(\bar{p} \wedge \bar{q})}.$$

Значит, любую булеву функцию можно записать только с помощью двух операций: $\wedge$ и $\bar{}$, т. е. $\{p \wedge q,\ \bar{p}\}$ — тоже полная система функций. Расплатой за малое количество операций, посредством которых записывается функция, становится громоздкость формул.

Пример 9.5. Функция **НЕ–И** определяется формулой:

$$p \ \textbf{НЕ–И}\ q = \overline{(p \wedge q)}.$$

Покажите, что $\{\ \textbf{НЕ–И}\ \}$ — полная система функций.

Решение. Для решения задачи достаточно показать, что каждая из функций $\bar{p}$, $p \vee q$ и $p \wedge q$ может быть выражена через **НЕ–И**.

Ввиду закона идемпотентности

$$\bar{p} = \overline{(p \wedge p)} = p \ \textbf{НЕ–И}\ p.$$

По закону де Моргана

$$\begin{aligned} p \vee q = \overline{(\bar{p} \wedge \bar{q})} &= \overline{\big((p \ \textbf{НЕ–И}\ p) \wedge (q \ \textbf{НЕ–И}\ q)\big)} = \\ &= (p \ \textbf{НЕ–И}\ p)\ \textbf{НЕ–И}\ (q \ \textbf{НЕ–И}\ q). \end{aligned}$$

Наконец,

$$\begin{aligned} p \wedge q = \overline{\big(\overline{(p \wedge q)}\big)} &= \overline{(p \ \textbf{НЕ–И}\ q)} = \\ &= (p \ \textbf{НЕ–И}\ q)\ \textbf{НЕ–И}\ (p \ \textbf{НЕ–И}\ q). \end{aligned}$$

Итак, $\{$ **НЕ–И** $\}$ — действительно полная система операций.

9.2. Карта Карно

Теперь мы попытаемся решить задачу об «упрощении» выражения для булевой функции. Под «упрощением» мы подразумеваем эквивалентное выражение, использующее меньше символов, чем исходное. Метод состоит в упрощении дизъюнктивной нормальной формы булевой функции, несмотря на то, что эта форма может оказаться более громоздкой, чем сама функция. Мы обсудим только булевы функции не более, чем от трех переменных. При желании разработанную технику можно будет обобщить на функции с любым числом аргументов.

Прежде всего, записывая функцию мы будем опускать символ $\wedge$ аналогично тому, как в обычной алгебре опускают символ умножения. Например, мы будем писать

$$\bar{p}\bar{q}r \vee \bar{p}qr \vee p\bar{q}\bar{r} \quad \text{вместо} \quad (\bar{p} \wedge \bar{q} \wedge r) \vee (\bar{p} \wedge q \wedge r) \vee (p \wedge \bar{q} \wedge \bar{r}).$$

Это выражение — дизъюнктивная нормальная форма. Его можно упростить следующим образом:

$$\begin{aligned}
\bar{p}\bar{q}r \vee \bar{p}qr \vee p\bar{q}\bar{r} &= (\bar{p}r\bar{q} \vee \bar{p}rq) \vee p\bar{q}\bar{r} = && \text{по законам коммутативности} \\
&&& \text{и ассоциативности} \\
&= \bar{p}r(\bar{q} \vee q) \vee p\bar{q}\bar{r} = && \text{по закону дистрибутивности} \\
&= \bar{p}r \vee p\bar{q}\bar{r} && \text{так как } \bar{q} \vee q = 1.
\end{aligned}$$

Мы получили выражение, существенно более простое, чем исходная функция. Оказывается, что последовательность шагов, которые мы совершили, упрощая функцию, можно делать почти автоматически. Заметим, что на первом шаге мы сделали перегруппировку: переставили и взяли в скобки два минтерма, отличающиеся только в одном символе. Закон дистрибутивности позволил нам на втором шаге вынести один минтерм за скобки, исключив из него булеву переменную q.

Упрощение функций можно делать с помощью *карты Карно*, метода, изобретенного в 1950-ых годах для разработки логических схем. Образно говоря, Карта Карно — это наглядная схема, предназначенная для обнаружения пар минтермов, которые можно сгруппировать и преобразовать в одно простое выражение.

В случае булевых функций трех переменных p, q и r карта Карно представляет собой таблицу с двумя строками и четырьмя столбцами (рис. 9.2). Столбцы обозначены дизъюнкциями, которые можно получить из двух переменных p и q и их отрицаний, а строки — переменной r и ее отрицанием $\bar{r}$.

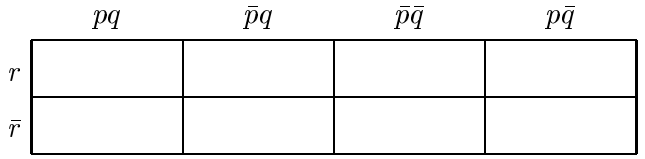


Рисунок 9.2. Карта Карно для функций трех переменных

Метки расставлены таким образом, что от столбца к столбцу в них происходит изменение ровно в одном символе. Ячейки карты Карно соответствуют восьми минтермам, которые можно построить из трех булевых переменных. Если нам дано булево выражение в дизъюнктивной нормальной форме, то в ячейки, соответствующие минтермам, участвующим в ней, мы записываем цифру 1. Например, карта Карно булева выражения $\bar{p}\bar{q}r \vee \bar{p}qr \vee p\bar{q}\bar{r}$ изображена на рис. 9.3.

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r		1	1	
$\bar{r}$				1

Рисунок 9.3. Карта Карно выражения $\bar{p}\bar{q}r \vee \bar{p}qr \vee p\bar{q}\bar{r}$

Затем предлагается «группировать» пары «соседних» единиц в Карте Карно (похожих на ту, которая выделена на рис. 9.3). Такая пара в нашем примере только одна. Она соответствует именно тем минтермам, которые мы объединили в сделанных ранее алгебраических преобразованиях.

Вообще говоря, при какой-то разметке карты Карно (отвечающей требованию, при котором метки соседних столбцов отличились только одним символом) может оказаться, что возможность группировки минтермов будет скрыта.

Например, на карте Карно (рис. 9.4) выражения $pqr \vee \bar{p}q\bar{r} \vee p\bar{q}r$ не видно, что члены pqr и $p\bar{q}r$ можно сгруппировать.

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r	1			1
$\bar{r}$		1		

Рисунок 9.4. Неудачное обозначение столбцов карты Карно выражения $pqr \vee \bar{p}q\bar{r} \vee p\bar{q}r$

Переобозначая столбцы с соблюдением основного требования, мы получим альтернативную карту Карно (рис. 9.5). На ней уже члены для группировки стоят рядом.

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r			1	1
$\bar{r}$	1			

Рисунок 9.5. Альтернативная карта Карно выражения $pqr \vee \bar{p}q\bar{r} \vee p\bar{q}r$

Следовательно,

$$pqr \vee \bar{p}q\bar{r} \vee p\bar{q}r = \bar{p}q\bar{r} \vee (p\bar{q}r \vee pqr) =$$
$$= \bar{p}q\bar{r} \vee pr(\bar{q} \vee q) = \bar{p}q\bar{r} \vee pr.$$

Пример 9.6. Упростите булево выражение

$$pqr \vee \bar{p}\bar{q}\bar{r} \vee \bar{p}qr \vee pq\bar{r} \vee \bar{p}q\bar{r}.$$

Решение. Обратим внимание на рис. 9.6, где изображена соответствующая карта Карно.

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r	1	1		
$\bar{r}$	1	1	1	

Рисунок 9.6. Карта Карно выражения $pqr \vee \bar{p}\bar{q}\bar{r} \vee \bar{p}qr \vee pq\bar{r} \vee \bar{p}q\bar{r}$

Из нее следует, что в данном выражении есть группа из четырех минтермов:

$$pqr \vee \bar{p}qr \vee pq\bar{r} \vee \bar{p}q\bar{r},$$

которую мы обозначим через (А), и группа из двух минтермов:

$$\bar{p}q\bar{r} \vee \bar{p}\bar{q}\bar{r}.$$

Ее мы обозначим через (Б).

Сначала поработаем над группой (А).

$$\begin{aligned} pqr \vee \bar{p}qr \vee pq\bar{r} \vee \bar{p}q\bar{r} &= (p \vee \bar{p})qr \vee (p \vee \bar{p})q\bar{r} = \\ &= qr \vee q\bar{r} = q(r \vee \bar{r}) = q. \end{aligned}$$

Теперь займемся группой (Б).

$$\bar{p}q\bar{r} \vee \bar{p}\bar{q}\bar{r} = \bar{p}\bar{r}(q \vee \bar{q}) = \bar{p}\bar{r}.$$

Таким образом, исходное выражение упрощается до[1] $q \vee \bar{p}\bar{r}$.

[1]Внимательно следя за преобразованиями, можно заметить, что минтерм $\bar{p}q\bar{r}$ здесь используется дважды, хотя в исходном выражении он задействован только один раз. Тем не менее, ошибки никакой нет. Это связано с законом идемпотентности: $f = f \vee f$, где f — произвольная булева функция. Аналогичный прием используется автором и далее. — *Прим. перев.*

Пример 9.7. Упростите булеву функцию

$$f(p,\, q,\, r) = \Big(\overline{(p \vee q)} \wedge r\Big) \vee \overline{(q \vee r)}.$$

Решение. Заполним таблицу истинности функции f (табл. 9.9).

Таблица 9.9

p	q	r	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

По таблице строим дизъюнктивную нормальную форму функции f:

$$\bar{p}\bar{q}\bar{r} \vee \bar{p}\bar{q}r \vee p\bar{q}\bar{r}.$$

Ее карта Карно показана на рис. 9.7.

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r			1	
$\bar{r}$			1	1

Рисунок 9.7. Карта Карно выражения $\bar{p}\bar{q}r \vee \bar{p}\bar{q}\bar{r} \vee p\bar{q}\bar{r}$

Из карты Карно видно, что в нашем выражении присутствуют две пары минтермов для группировки: $\bar{p}\bar{q}\bar{r} \vee \bar{p}\bar{q}r$ и $\bar{p}\bar{q}\bar{r} \vee p\bar{q}\bar{r}$. После их упрощения получаются функции: $\bar{p}\bar{q}$ и $\bar{q}\bar{r}$. Следовательно, исходная функция сводится к выражению $\bar{p}\bar{q} \vee \bar{q}\bar{r}$.

Наглядный способ поиска минтермов для группировки, который предоставляет карта Карно, можно обобщить на булевы функции четырех, пяти и даже шести переменных. Однако при этом возникают трехмерные диаграммы и дополнительные осложнения делают метод малопроизводительным. Есть и другие способы упрощения булевых выражений. Особенно привлекателен метод Куина–Мак-Класки. Он довольно систематичный (см., например, [**??**]) и применим к функциям любого количества булевых переменных.

9.3. Функциональные схемы

Одно из основных приложений булевых функций лежит в области создания схем функциональных элементов или *функциональных схем*, которые можно реализовать в виде электронных устройств с конечным числом входов и выходов, причем на каждом входе и выходе может появляться только два значения. Такие устройства собраны из *функциональных элементов*, генерирующих основные булевы операции. Стандартные обозначения основных функциональных элементов показаны на рис. 9.8.

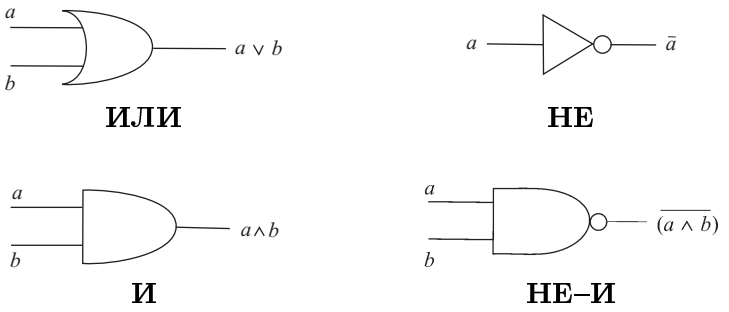


Рисунок 9.8. Стандартные обозначения функциональных элементов

Соединяя функциональные элементы вместе, мы получаем *функциональную схему*. С ее помощью можно реализовать любую булеву функцию.

Пример 9.8. Что получится на выходе функциональной схемы, представленной на рис. 9.9?

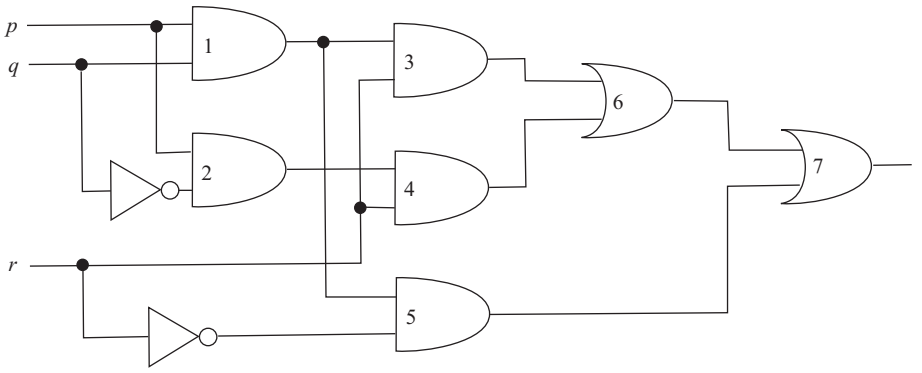


Рисунок 9.9. Функциональная схема

Решение. В табл. 9.10 перечислены входы и соответствующие выходы для каждого функционального элемента в соответствии с нумерацией из рис. 9.9.

Таблица 9.10

Вентиль	Вход	Выход
1	$p,\ q$	pq
2	$p,\ \bar{q}$	$p\bar{q}$
3	$pq,\ r$	pqr
4	$p\bar{q}, r$	$p\bar{q}r$
5	$pq,\ \bar{r}$	$pq\bar{r}$
6	$pqr,\ p\bar{q}r$	$pqr \vee p\bar{q}r$
7	$pqr \vee p\bar{q}r,\ pq\bar{r}$	$pqr \vee p\bar{q}r \vee pq\bar{r}$

Таким образом, на выходе схемы получится функция $pqr \vee p\bar{q}r \vee pq\bar{r}$.

Диаграммы функциональных схем можно упростить, если разрешить функциональным элементам **И** и **ИЛИ** иметь не по два входа, а больше. Но более впечатляющего упрощения можно добиться, если привлечь карту Карно для преобразования функции, полученной на выходе сложной схемы.

Пример 9.9. Упростите функцию, генерируемую схемой из примера 9.8 и найдите более простую функциональную схему, ее реализующую.

Решение. Карта Карно требуемого выражения представлена на рис. 9.10. Она имеет две пары минтермов для группировки (одна из них не видна при данном обозначении столбцов).

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r	1			1
$\bar{r}$	1			

Рисунок 9.10. Карта Карно выражения $pqr \vee p\bar{q}r \vee pq\bar{r}$

Итак,

$$pqr \vee pq\bar{r} = pq(r \vee \bar{r}) = pq$$

и

$$pqr \vee p\bar{q}r = (q \vee \bar{q})pr = pr.$$

Это сводит функцию к выражению $pq \vee pr$, которое, ввиду дистрибутивности, редуцируется к функции $p(q \vee r)$.

Более простая схема, реализующая функцию из примера 9.8, показана на рис. 9.11.

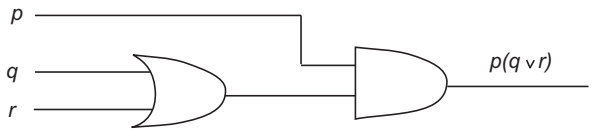


Рисунок 9.11.

При вычерчивании функциональных схем нет необходимости использовать все типы функциональных элементов. Как мы уже видели, множество $\{\vee, \bar{}\}$ является полной системой функций. Поэтому мы можем построить любую схему, ограничившись функциональными элементами **И** и **НЕ**.

Более того, если по той или иной причине нам неудобно использовать большое число компонент, мы могли бы использовать только функциональный элемент **НЕ–И** .

Пример 9.10. Начертите функциональную схему, реализующую булеву функцию $p(q \vee r)$, используя только **НЕ–И** .

Решение. Во-первых, заметим, что

$$p(q \vee r) = \big(p \textbf{ НЕ–И } (q \vee r)\big) \textbf{ НЕ–И } \big(p \textbf{ НЕ–И } (q \vee r)\big).$$

А во-вторых,

$$q \vee r = (q \textbf{ НЕ–И } q) \textbf{ НЕ–И } (r \textbf{ НЕ–И } r).$$

Искомая схема показана на рис. 9.12.

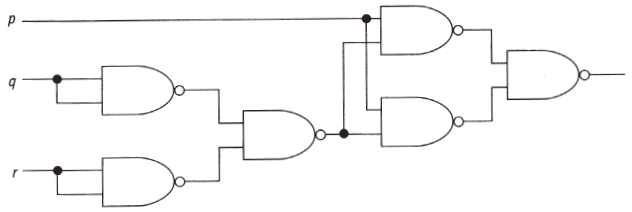


Рисунок 9.12. Функциональная схема функции $p\,(q \vee r)$

Набор упражнений 9

9.1. Заполняя подходящие таблицы истинности, докажите законы де Моргана.

9.2. Опираясь на законы булевой алгебры, проверьте соотношения:

(а) $\overline{(p \wedge \bar{q})} \vee \bar{r} = \bar{p} \vee q \vee \bar{r}$;

(б) $\overline{((p \wedge \bar{q}) \wedge (r \vee (p \wedge \bar{q})))} = \bar{p} \vee q$.

9.3. Найдите дизъюнктивную нормальную форму булевой функции $g(p, q, r, s)$, чья таблица истинности — (табл. 9.11).

Таблица 9.11

p	q	r	s	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

9.4. Заполните таблицу истинности булева выражения

$$\big(p \wedge (\bar{q} \vee r)\big) \vee \big(\bar{p} \wedge (q \vee \bar{r})\big)$$

и найдите его дизъюнктивную нормальную форму.

9.5. Запишите выражение $(p \wedge \bar{q}) \wedge r$, используя только:

(а) операции $\vee$ и $^{-}$;

(б) функцию **НЕ–И**.

9.6. Булева функция **НЕ–ИЛИ** определяется по формуле

$$p \textbf{ НЕ–ИЛИ } q = \overline{(p \vee q)}.$$

Покажите, что {**НЕ–ИЛИ**} является полной системой функций.

9.7. Изобразите карту Карно булева выражения с дизъюнктивной нормальной формой

$$\bar{p}\bar{q}r \vee \bar{p}qr \vee pq\bar{r} \vee pqr$$

и найдите его упрощенную версию.

9.8. Найдите дизъюнктивную нормальную форму булевой функции $f(p, q, r)$ с таблицей истинности — табл. 9.12.

Таблица 9.12

p	q	r	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Изобразите ее карту Карно и упростите функцию f.

9.9. Вычислите булеву функцию, генерируемую функциональной схемой, показанной на рис. 9.13.

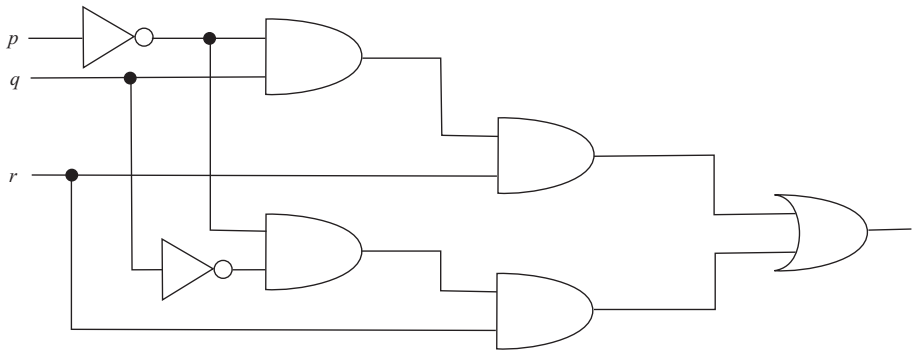


Рисунок 9.13.

Используя карту Карно, найдите эквивалентную схему, состоящую из двух функциональных элементов: **И** и **НЕ**.

9.10. С помощью законов булевой алгебры проверьте, что выражение

$$\bar{p}\ \textbf{НЕ–И}\ (\bar{q}\ \textbf{НЕ–И}\ r)$$

эквивалентно выражению

$$p \vee (\bar{q} \wedge r).$$

Затем замените функциональную схему, представленную на рис. 9.14, на эквивалентную ей, но состоящую из двух функциональных элементов: **И** и **ИЛИ** и одного инвертора[1].

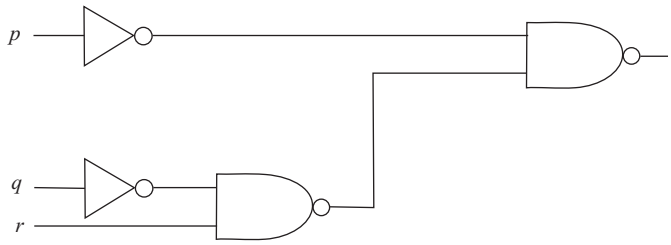


Рисунок 9.14.

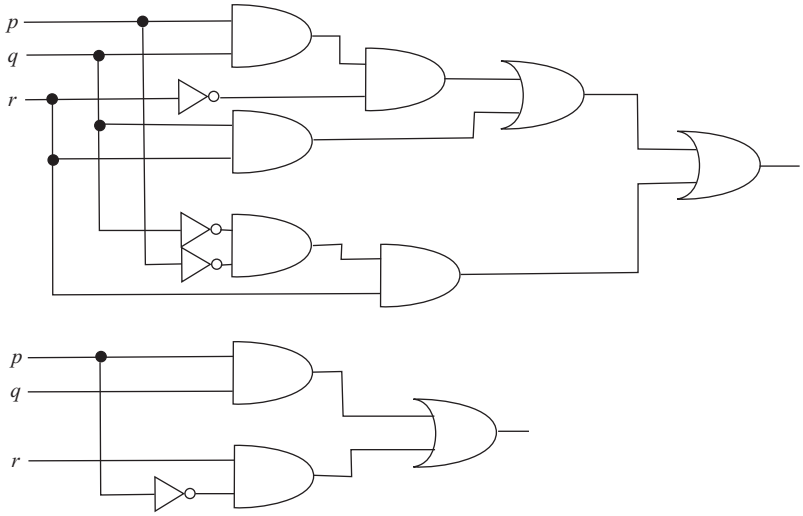


Рисунок 9.15.

[1] Инвертором называют функциональный элемент **НЕ**. — *Прим. перев.*

9.11. Докажите эквивалентность функциональных схем, изображенных на рис. 9.15.

9.12. Начертите функциональную схему выражения p **НЕ–ИЛИ** q, используя только функциональный элемент **НЕ–И** . (Указание: начните с проверки соотношения q **НЕ–ИЛИ** $q = \overline{(\bar{p}\ \textbf{НЕ–И}\ \bar{q})}$, а затем убедитесь, что любая булева переменная r удовлетворяет тождеству $\bar{r} = r$ **НЕ–И** r.)

Краткое содержание главы

Булева переменная принимает только два значения: 0 и 1.

Булевы переменные можно комбинировать, используя операции $\vee$, $\wedge$ и $^-$ для создания **булевых выражений**.

Булевой функцией от n булевых переменных $p_1, p_2, \ldots, p_n$ называется такая функция $f : B^n \longrightarrow B$, что $f(p_1, p_2, \ldots, p_n)$ — булево выражение.

Булева функция называется **минтермом**, если столбец таблицы истинности, в котором записаны ее значения, содержит только одну единицу.

Пусть p и q — булевы переменные. Тогда имеют место следующие **законы булевой алгебры**:

Таблица 9.13. Законы булевой алгебры

Законы коммутативности:	$p \wedge q = q \wedge p$,
	$p \vee q = q \vee p$;
Законы ассоциативности:	$p \wedge (q \wedge r) = (p \wedge q) \wedge r$,
	$p \vee (q \vee r) = (p \vee q) \vee r$;
Законы дистрибутивности:	$p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$,
	$p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$;
Законы идемпотентности:	$p \wedge p = p$,
	$p \vee p = p$;
Законы поглощения:	$p \wedge (p \vee q) = p$,
	$p \vee (p \wedge q) = p$;
Законы де Моргана:	$\overline{(p \wedge q)} = \bar{p} \vee \bar{q}$,
	$\overline{(p \vee q)} = \bar{p} \wedge \bar{q}$.

Любая булева функция может быть единственным образом записана как дизъюнкция минтермов. Такое представление функции называется **дизъюнктивной нормальной формой**.

Множество функций, через которые можно выразить любую булеву функцию, называется **полной системой функций**.

Булево выражение можно упростить, используя **карту Карно**, прямоугольную таблицу, чьи строки и столбцы обозначены конъюнкциями булевых переменных и их отрицаний. В клетках этой таблицы, соответствующих минтермам данной дизъюнктивной нормальной формы, помещаются единицы.

Двоичное устройство — это устройство, как правило электронное, снабженное конечным числом двоичных входов и выходов.

Функциональная схема строится из **функциональных элементов**, которые реализуют основные булевы операции (рис. 9.16).

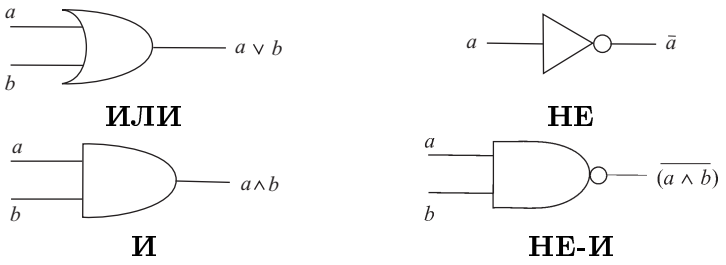


Рисунок 9.16. Стандартные обозначения функциональных элементов

Функциональную схему можно упростить, применяя **карту Карно** для уменьшения сложности булева выражения, генерируемого схемой.

Приложение. Проектирование 2-битного сумматора

2-битный сумматор — это устройство, которое вычисляет сумму двузначных двоичных[2] чисел, выдавая в качестве ответа трехзначное двоичное число. Например, $10 +_2 11 = 101$. Для создания функциональной схемы 2-битного сумматора мы сначала построим *полубитный сумматор* предназначенный для сложения двух двоичных цифр. Ответ при этом представляется двузначным двоичным числом. Например, $1 +_2 1 = 10$.

[2]Записанных в двоичной системе счисления. — *Прим. перев.*

Полубитный сумматор.

Пусть x и y обозначают двоичные цифры, которые предстоит сложить, а u и v — двоичные цифры суммы, получающейся на выходе сумматора, как показано на рис. 9.17.

Рисунок 9.17.

Таблица истинности (табл. 9.14) проясняет связь между вводимыми и выводимыми цифрами. Следовательно, $u = xy$ (*разряд переноса*) и $v = \bar{x}y \vee x\bar{y}$ (*сложение по модулю 2*).

Таблица 9.14

x	y	u	v
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Задача 1. Проверьте, что функциональная схема, изображенная на рис. 9.18, реализует полубитный сумматор.

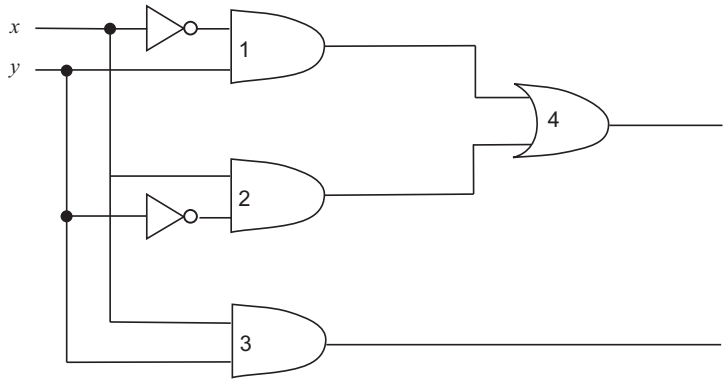


Рисунок 9.18. Схема полубитного сумматора

Решение. Входными данными элементов 3 и 4 являются разряд переноса и сумма по модулю 2 соответственно (смотри табл. 9.15).

Таблица 9.15

Логический элемент	Ввод	Вывод
1	$\bar{x},\ y$	$\bar{x}y$
2	$x,\ \bar{y}$	$x\bar{y}$
3	$x,\ y$	xy
4	$\bar{x}y,\ x\bar{y}$	$\bar{x}y \vee x\bar{y}$

2-битный сумматор.

На входе 2-битный сумматор получает два двузначных двоичных числа, а на выходе у него оказывается трехзначное число, равное сумме вводимых чисел. Иными словами, 2-битный сумматор складывает числа в двоичной системе счисления, например: $11 +_2 10 = 101$.

Обозначим через a и b цифры первого вводимого в сумматор числа, а через c и d — цифры второго (рис. 9.19). Пусть e, f и g — цифры вычисляемой суммы.

Рисунок 9.19.

Далее мы могли бы, как и в случае с полубитным сумматором, заполнить таблицы истинности цифр e, f и g, считая их булевыми функциями от вводимых цифр, упростить полученные выражения с помощью карты Карно и начертить функциональную схему. Однако мы поступим иначе: используем полубитный сумматор в качестве блока функциональной схемы. Схема, представленная на рис. 9.20, использует два полубитных сумматора для вычисления сумм: $a +_2 c$ и $b +_2 d$.

Сумма по модулю 2 (переменная v_2) дает нам цифру g. Складывая разряд переноса u_2 с v_1 с помощью третьего полубитного сумматора, мы получаем двузначное число с цифрами u_3 и f. Наконец, последняя цифра суммы, e, может быть получена из u_1 и u_3 с помощью функционального элемента **ИЛИ**.

Задача 2. Проверьте описанные действия 2-битного сумматора на примере суммы $11 +_2 10 = 101$.

Решение. Нам дано:

$$a = 1, \quad b = 1, \qquad c = 1, \quad d = 0.$$

Так как $a +_2 c = 1 +_2 1 = 10$, то $u_1 = 1$ и $v_1 = 0$. Ввиду того, что $b +_2 d = 1 +_2 0 = 01$, мы получаем: $u_2 = 0$ и $v_2 = 1$, откуда $g = 1$.

Далее, $u_2 +_2 v_1 = 0 +_2 0 = 00$, т. е. $u_3 = 0$. Значит $f = 0$.

Наконец, $u_1 \vee u_3 = 1 \vee 0 = 1$, что дает равенство $e = 1$.

Итак, $11 +_2 10 = 101$, как и ожидалось.

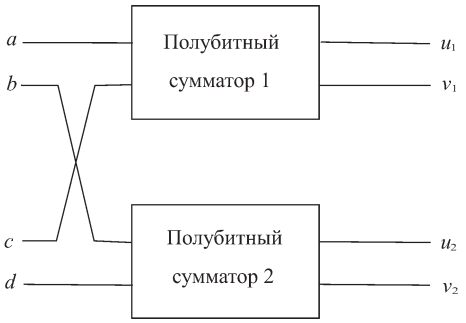


Рисунок 9.20.

Разработанная функциональная схема 2-битного сумматора представлена на рис. 9.21.

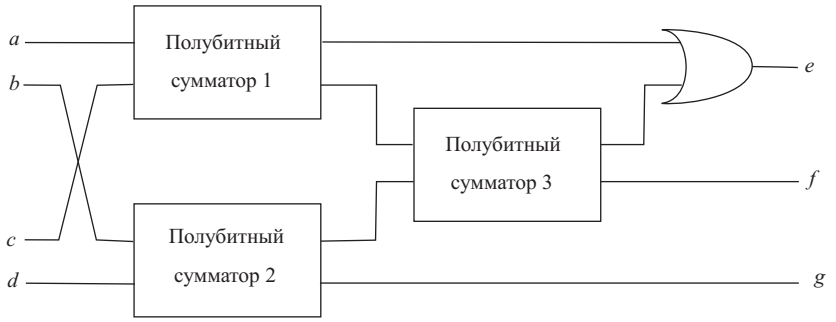


Рисунок 9.21. Функциональная схема 2-битного сумматора

Задача 3. Проследите процесс сложения двоичных чисел 11 и 01 функциональной схемой из рис. 9.21.

Решение. Обозначим через u_i и v_i цифры, генерируемые полубитным сумматором i при $i = 1$, 2 и 3. Так как $a = 1$, $b = 1$, $c = 0$ и $d = 1$, то

$$a +_2 c = 1 +_2 0 = 01 \;\Rightarrow\; u_1 = 0 \text{ и } v_1 = 1;$$

$$b +_2 d = 1 +_2 1 = 10 \;\Rightarrow\; u_2 = 1 \text{ и } v_2 = 0 \quad (\text{откуда } g = 0).$$

Теперь в качестве входных данных третьего полубитного сумматора мы имеем $v_1 = 1$ и $u_2 = 1$. Значит

$$v_1 +_2 u_2 = 1 +_2 1 = 10 \;\Rightarrow\; u_3 = 1 \text{ и } v_3 = 0 \quad (\text{откуда } f = 0).$$

Наконец,

$$u_1 \vee u_3 = 0 \vee 1 = 1, \qquad (\text{откуда } e = 1).$$

Итак, окончательная сумма, генерируемая сумматором, равна $100 = 11 +_2 01$, что соответствует истине.

На рис. 9.22 изображена функциональная схема 3-битного сумматора, складывающего два трехзначных двоичных числа с цифрами a, b, c и d, e, f соответственно. В качестве суммы получается четырехзначное число с цифрами g, h, i и j.

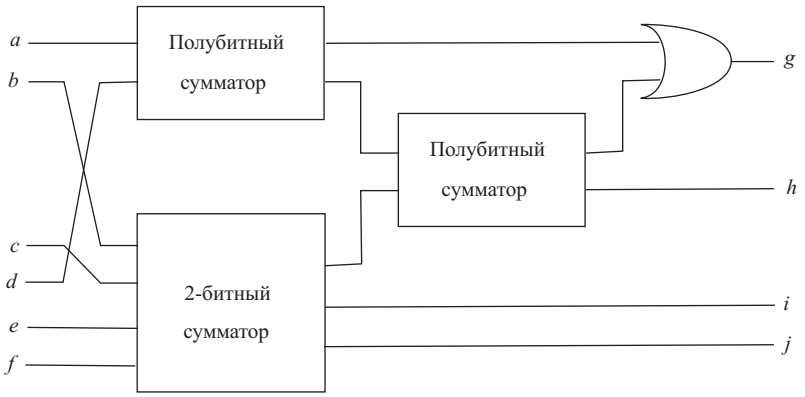


Рисунок 9.22. Функциональная схема 3-битного сумматора

Задача 4. Проверьте, что схема из рис. 9.22 правильно вычисляет следующие суммы:

(а) $110 +_2 011$;

(б) $101 +_2 111$.

Решение. Вам следует получить 1001 в случае (а) и 1100 в случае (б).

Решения упражнений

Набор упражнений 1

1.1. Одна из возможных сетей дорог наименьшей общей длины с началом в вершине B показана на рис. P1.1.

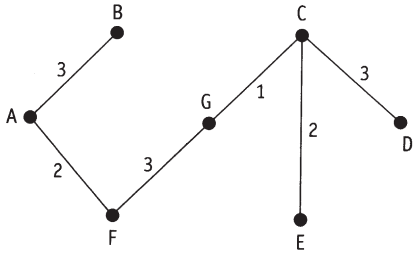


Рисунок P1.1.

Общая длина любой минимальной сети равна 14.

1.2. (а) $f = 6$,

(б) $f = 120$.

В случае произвольного натурального n мы получим $f = n!$.

1.3. Смотри табл. P1.1

Таблица P1.1

i	j	$i \neq 4$?
1	3	Да
2	9	Да
3	27	Да
4	81	Нет

При $n > 0$ алгоритм вычисляет m^n.

В случае $n = 0$ алгоритм будет работать некорректно. Более того, проходы цикла будут повторяться бесконечно, поскольку условие останется истинным для всех значений параметра цикла[1] i.

[1] Для устранения такой неприятности достаточно заменить условие $i \neq n$ на $i < n$. — *Прим. перев.*

1.4. Выходные данные алгоритма: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89. Алгоритм вычисляет члены последовательности Фибоначчи, каждый из которых, начиная с третьего, равен сумме двух предшествующих.

1.5. Смотри табл. Р1.2

Таблица Р1.2

l	sum	k	$k < 12$?
0	0	1	Да
1	1	3	Да
4	5	5	Да
9	14	7	Да
16	30	9	Да
25	55	11	Да
36	91	13	Нет

В результате работы алгоритма получается сумма квадратов первых n натуральных чисел.

1.6. Смотри табл. Р1.3

Таблица Р1.3

текущее	ребро	вес
1	BG	6
2	DE	5
3	EF	4
4	AB	3
5	BC	3
6	CD	3
7	EG	3
8	FG	3
9	AF	2
10	CE	2
11	CG	1

У нас есть некоторая свобода выбора при упорядочении ребер одинакового веса. Изначально *остаток* $= 11$ и $m = 7$. Цикл выполняется пять раз. При этом удаляется пять ребер: BG, DE, EF, AB и EG. Остается минимальная сеть, изображенная на рис. Р1.2.

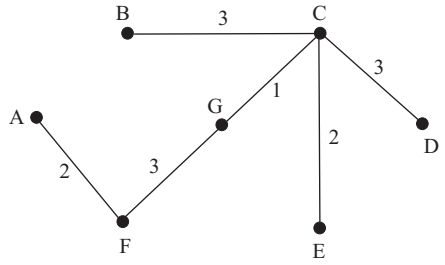


Рисунок Р1.2. Минимальная сеть

Набор упражнений 2

2.1. (а) P **и** (**не** Q),
(б) R **и** P,
(в) $R \Rightarrow P$,
(г) $P \Rightarrow Q$,
(д) (**не** P) $\Rightarrow$ (**не** Q).

2.2. (а) (**не** P) $\Rightarrow$ (**не** Q).
(б) P **или** (**не** Q).
(в) (P **или** Q) **и** (**не** (P **и** Q)).

Таблица истинности приведена в табл. Р2.1.

Таблица Р2.1

P	Q	**не** P	**не** Q	P **или** (**не** Q)	((**не** P) $\Rightarrow$ (**не** Q))
И	И	Л	Л	И	И
И	Л	Л	И	И	И
Л	И	И	Л	Л	Л
Л	Л	И	И	И	И

Поскольку два последних столбца таблицы идентичны, высказывания (а) и (б) логически эквивалентны.

2.3. Таблицы истинности высказываний приведены в табл. Р2.2 и табл. Р2.3. Из них следует, что высказывания (а) и (в) — тавтологии.

Таблица Р2.2

P	**не** P	P **и** (**не** P)	**не** (P **и** (**не** P))	$P \Rightarrow$ **не** P
И	Л	Л	И	Л
Л	И	Л	И	И

Таблица Р2.3

P	Q	$P \Rightarrow Q$	P **и** $(P \Rightarrow Q)$	$(P$ **и** $(P \Rightarrow Q)) \Rightarrow Q$
И	И	И	И	И
И	Л	Л	Л	И
Л	И	И	Л	И
Л	Л	И	Л	И

2.4. Смотри табл. Р2.4.

Таблица Р2.4

P	Q	R	$P \Rightarrow Q$	(**не** P) $\Rightarrow R$	$Q \Rightarrow R$	$(P \Rightarrow Q) \Rightarrow R$	((**не** P) $\Rightarrow R$) **и** $(Q \Rightarrow R)$
И	И	И	И	И	И	И	И
И	И	Л	И	И	Л	Л	Л
И	Л	И	Л	И	И	И	И
И	Л	Л	Л	И	И	И	И
Л	И	И	И	И	И	И	И
Л	И	Л	И	Л	Л	Л	Л
Л	Л	И	И	И	И	И	И
Л	Л	Л	И	Л	И	Л	Л

Мы видим, что последние два столбца таблицы идентичны. Это означает, что высказывания

$$\Big((P \Rightarrow Q) \Rightarrow R\Big) \quad \text{и} \quad \Big(\big((\textbf{не } P) \Rightarrow R\big) \textbf{ и } \big(Q \Rightarrow R\big)\Big)$$

логически эквивалентны.

2.5. (а) $\forall\, x\ P(x)$;

(б) $\exists\, x$: **не** $P(x)$;

(в) $\forall\, x$ **не** $P(x)$.

Отрицание высказывания (б) в символьной форме выглядит так: $\forall\, x\ P(x)$ (т. е. высказывание (а) является отрицанием высказывания (б)).

Отрицание высказывания (в) имеет вид: $\exists\, x\colon\ P(x)$. Его следует читать следующим образом: найдется кошка с усами.

2.6. Требуемое высказывание читается так: любой человек — высокий и толстый. Его отрицание сформулировано в п. (в).

2.7. (а) Четные числа n и m мы можем представить в виде $n = 2a$ и $m = 2b$, где как a, так и b — целые числа. Следовательно,

$$n + m = 2a + 2b = 2(a + b),$$

откуда вытекает четность суммы $n + m$.

(б) Если n — нечетное число, то $n = 2a + 1$ для какого-то целого числа a. Возведя n во вторую степень, мы получим

$$n^2 = (2a + 1)^2 = 4a^2 + 4a + 1 = 2(2a^2 + 2a) + 1.$$

Значит, n^2 — нечетное число. Итак, если n^2 — число четное, то и n — четное.

(в) Допустим, что сумма целых чисел $n + m$ — нечетное число, а заключение утверждения задачи: одно из слагаемых n и m является четным, а другое нечетным, — ложно. Отрицание заключения означает, что оба слагаемых либо четны, либо нечетны. Рассмотрим эти возможности отдельно.

Если как m, так и n — четные числа, то $m = 2a$ и $n = 2b$ для каких-то целых чисел a и b. Стало быть, их сумма

$$n + m = 2a + 2b = 2(a + b)$$

оказывается четной, что противоречит предположению.

Пусть теперь m и n — нечетны. Тогда их можно записать в виде $m = 2a + 1$, $n = 2b + 1$ с целыми числами a и b. Сложив их, получим

$$n + m = (2a + 1) + (2b + 1) = 2(a + b) + 2 = 2(a + b + 1),$$

четное число. Опять пришли к противоречию с предположением.

Итак, отрицание заключения утверждения в обоих возможных случаях ведет к противоречию. Следовательно, утверждение истинно.

2.8. (а) Обозначим предикат $1 + 5 + 9 + \cdots + (4n - 3) = n(2n - 1)$ через $P(n)$. При $n = 1$ левая часть равенства состоит только из 1. Правая часть после подстановки $n = 1$ тоже окажется равной 1:

$$n(2n - 1) = 1 \cdot (2 \cdot 1 - 1) = 1.$$

Поэтому высказывание $P(1)$ истинно.

Предположим, что $P(k)$ истинно при некотором $k \geqslant 1$. Нам следует показать, что такое предположение влечет истинность $P(k + 1)$. Сделаем это.

$$\begin{aligned}1+\cdots+\big(4(k+1)-3\big) &= 1+\cdots+(4k-3)+(4k+1) = \\ &= k(2k-1)+(4k+1) = \\ &= 2k^2+3k+1 = \\ &= (k+1)(2k+1) = \\ &= (k+1)\big(2(k+1)-1\big).\end{aligned}$$

Таким образом, в силу принципа математической индукции $P(n)$ истинно при любом $n \geqslant 1$.

(б) Здесь $P(n)$ будет обозначать предикат:

$$1^2+2^2+\cdots+n^2 = \frac{1}{6}n(n+1)(2n+1).$$

Так как $1^2 = 1$ и $\frac{1}{6}n(n+1)(2n+1) = \frac{1}{6}\cdot 1\cdot 2\cdot 3 = 1$ (при $n=1$), то высказывание $P(1)$ верно.

Допустим, что $P(k)$ истинно при некотором целом $k \geqslant 1$ и покажем, что отсюда вытекает истинность $P(k+1)$.

$$\begin{aligned}1^2+\cdots+(k+1)^2 &= 1^2+\cdots+k^2+(k+1)^2 = \\ &= \frac{1}{6}k(k+1)(2k+1)+(k+1)^2 = \\ &= \frac{1}{6}(k+1)\big(k(2k+1)+6(k+1)\big) = \\ &= \frac{1}{6}(k+1)(2k^2+7k+6) = \\ &= \frac{1}{6}(k+1)\big((2k+3)(k+2)\big) = \\ &= \frac{1}{6}(k+1)\big((k+1)+1\big)\big(2(k+1)+1\big).\end{aligned}$$

Итак, по индукции заключаем, что $P(n)$ верно для всех натуральных чисел $n \geqslant 1$.

(в) Пусть теперь $P(n)$ обозначает предикат

$$\frac{1}{1\cdot 3}+\frac{1}{3\cdot 5}+\cdots+\frac{1}{(2n-1)\cdot(2n+1)} = \frac{n}{2n+1}.$$

Если $n=1$, то левая часть равенства будет равна $\frac{1}{3}$, а правая —

$$\frac{n}{2n+1} = \frac{1}{3}.$$

Следовательно, $P(1)$ — верное высказывание.

Пусть высказывание $P(k)$ истинно для какого-то натурального k. Тогда

$$\frac{1}{1\cdot 3}+\frac{1}{3\cdot 5}+\cdots+\frac{1}{\big(2(k+1)-1\big)\big(2(k+1)+1\big)}=$$
$$=\frac{1}{1\cdot 3}+\cdots+\frac{1}{(2k-1)(2k+1)}+\frac{1}{(2k+1)(2k+3)}=$$
$$=\frac{k}{2k+1}+\frac{1}{(2k+1)(2k+3)}=\frac{k(2k+3)+1}{(2k+1)(2k+3)}=$$
$$=\frac{2k^2+3k+1}{(2k+1)(2k+3)}=\frac{(2k+1)(k+1)}{(2k+1)(2k+3)}=\frac{k+1}{2k+3}.$$

Этими вычислениями мы вывели высказывание $P(k+1)$ из предположения об истинности $P(k)$. Значит, $P(n)$ имеет место для всех натуральных чисел n.

(г) Обозначим предикат: «n^3-n делится на 3» через $P(n)$. Так как $1^3-1=0$ делится на 3, то высказывание $P(1)$ истинно.

Предположим, что $P(k)$ верно для некоторого целого $k\geqslant 1$. Тогда

$$(k+1)^3-(k+1)=k^3+3k^2+2k=$$
$$=(k^3-k)+3k^2+3k.$$

Число k^3-k делится на 3 по предположению индукции, сумма $3k^2+3k=3(k^2+k)$ тоже делится на 3. Значит, и $(k^3-k)+3k^2+3k$ должно делиться на 3. Тем самым мы показали истинность импликации: $P(k)\Rightarrow P(k+1)$, и ввиду принципа математической индукции $P(n)$ верно для всех натуральных чисел n.

(д) Обозначим через $P(n)$ предикат:

$$1\cdot 1!+2\cdot 2!+\cdots+n\cdot n!=(n+1)!-1.$$

Подставив в это равенство $n=1$, получим легко проверяемое тождество:

$$1\cdot 1!=(1+1)!-1.$$

Значит, $P(1)$ верно.

Предположение об истинности $P(k)$ при каком-то натуральном k влечет цепочку равенств:

$$\begin{aligned}&1\cdot 1!+2\cdot 2!+\cdots+(k+1)\cdot(k+1)!=\\&=1\cdot 1!+2\cdot 2!+\cdots+k\cdot k!+(k+1)\cdot(k+1)!=\\&=(k+1)!-1+(k+1)!(k+1)=\\&=(k+1)!(k+2)-1=(k+2)!-1,\end{aligned}$$

в конце которой стоит высказывание $P(k+1)$, что и требовалось. Значит, $P(n)$ верно при любом натуральном значении n.

2.9. $x_2=\frac{1}{3}$, $x_3=\frac{1}{7}$ и $x_4=\frac{1}{15}$.

Пусть $P(n)$ — предикат $x_n=\frac{1}{2^n-1}$. Подстановка $n=1$ в равенство доказывает истинность $P(1)$. Предположим, что $P(k)$ верно при каком-то $k\geqslant 1$. Тогда

$$x_{k+1}=\frac{x_k}{x_k+2}=\frac{\frac{1}{2^k-1}}{\frac{1}{2^k-1}+2}=\frac{1}{1+2(2^k-1)}=\frac{1}{2^{k+1}-1}.$$

Следовательно, истинность $P(k)$ влечет истинность $P(k+1)$. Значит, $P(n)$ — верное высказывание при любом натуральном n.

2.10. Вычислим первые члены последовательности:

$$\begin{aligned}x_3&=2x_2-x_1=2\cdot 2-1=3;\\x_4&=2x_3-x_2=2\cdot 3-2=4;\\x_5&=2x_4-x_3=2\cdot 4-3=5.\end{aligned}$$

Наш опыт наводит на мысль, что общий член последовательности равен своему номеру. Попытаемся это доказать. Обозначим через $P(n)$ предикат $x_n=n$. Истинность $P(1)$ и $P(2)$ следует из условия задачи. Более того, вычисления, которые мы провели, доказывают истинность $P(3)$, $P(4)$ и $P(5)$.

Предположим, что для некоторого $k>1$ истинны высказывания $P(k-1)$ и $P(k)$. Тогда

$$x_{k+1}=2x_k-x_{k-1}=2k-(k-1)=k+1.$$

Итак, мы получили истинность высказывания $P(k+1)$ из предположения об истинности $P(k-1)$ и $P(k)$. Тем самым,

помня о математической индукции, можно утверждать, что предикат $P(n)$ имеет истинное значение для любого натурального n.

Набор упражнений 3

3.1. (а) $A = \{10, 11, 12, 13, 14, 15, 16, 17\}$;
$B = \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$;
$C = \varnothing$;
$D = \left\{-\frac{1}{2}, \frac{1}{3}\right\}$;

(б) $S = \{3n - 1 : \ n \in \mathbb{N}\}$;
$T = \{1/(2^n - 1) : \ n \in \mathbb{N}\}$.

3.2. (а) $\{t\}$; (г) $\varnothing$; (ж) $\{r, v\}$;
(б) $\{p, q, r, s, t, u\}$; (д) $\{p, s\}$; (з) $\{p, r, s, u, v\}$;
(в) $\{q, r, v, w\}$; (е) $\{u, w\}$;

3.3. (а) Истинно, поскольку $A \cup B$ состоит из всех слов словаря.

(б) Истинно, так как слово «бассейн» стоит перед словом «кошка» и тем самым не содержится в B; кроме того, данное слово имеет двойную букву «с».

(в) Ложно, так как симметрическую разность $B \triangle C$ можно записать как $(B \cup C) \setminus (B \cap C)$, а слово «стресс» принадлежит как множеству B, так и C, т. е. оно лежит в их пересечении.

(г) Ложно, ибо в словаре русского языка между словами «кошка» и «собака» содержится немало других слов, например, слово «мышь».

(д) Слова словаря, находящиеся между словами «кошка» и «собака» и имеющие двойную букву.

(е) Все слова словаря, несодержащие двойную букву.

3.4. (а) (i) $\overline{B}$; (iii) $A \cap B$;
(ii) $B \cap C$; (iv) $C \setminus B$.

(б) $A \setminus B = \{3n : \ n \in \mathbb{Z}$ и $n \geqslant 4$, и n— нечетно$\}$.
Заметьте, если представить n как $2k + 1$, то ответ будет выглядеть так: $A \setminus B = \{6k + 3 : \ k \in \mathbb{Z}$ и $k \geqslant 2\}$.

3.5. Соответствующие диаграммы Венна приведены на рис. Р3.1.

Если $x \in A \cap (B \cup C)$, то $x \in A$ и $\big((x \in B)$ **или** $(x \in C)\big)$. Следовательно,

$$\big((x \in A) \textbf{ и } (x \in B)\big) \textbf{ или } \big((x \in A) \textbf{ и } (x \in C)\big).$$

Иными словами, $x \in (A \cap B) \cup (A \cap C)$. Отсюда вытекает истинность включения: $A \cap (B \cup C) \subset (A \cap B) \cup (A \cap C)$. Обратное включение проверяется аналогично.

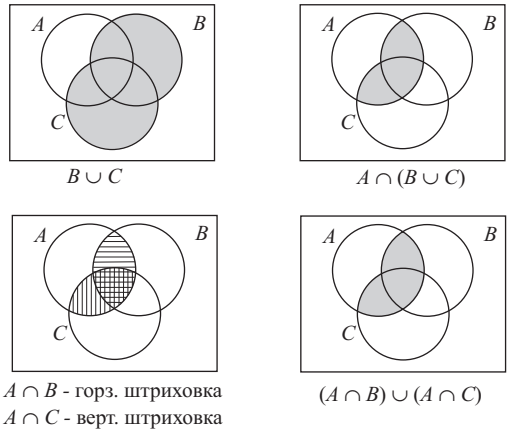


Рисунок Р3.1.

3.6. Соответствующие диаграммы Венна приведены на рис. Р3.2.

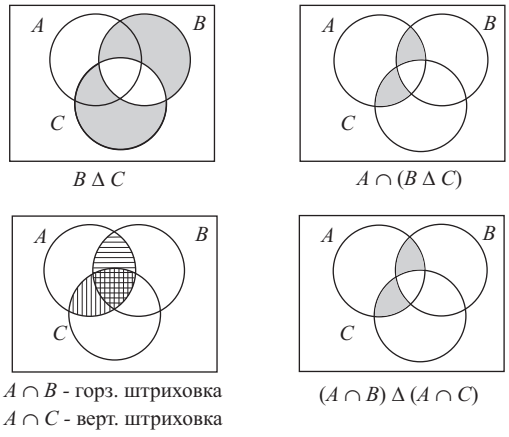


Рисунок Р3.2.

Любой набор множеств A, B и C, в котором множество A не содержит элементов не из B, не из C, противоречит равен-

ству $A \cup (B \triangle C) = (A \cup B) \triangle (A \cup C)$. В частности, пусть $A = \{1, 2\}$, $B = \{3, 4\}$ и $C = \{4, 5\}$. Тогда

$$B \triangle C = \{3, 5\} \qquad \text{и} \qquad A \cup (B \triangle C) = \{1, 2, 3, 5\}.$$

С другой стороны,

$$A \cup B = \{1, 2, 3, 4\}, \quad A \cup C = \{1, 2, 4, 5\}$$

и

$$(A \cup B) \triangle (A \cup C) = \{3, 5\}.$$

3.7. (а) Выполним преобразования:

$$\begin{aligned} \overline{(A \cap \overline{B})} \cup B &= (\overline{A} \cup B) \cup B = && \text{(з. де Моргана и дополн.)} \\ &= \overline{A} \cup (B \cup B) = && \text{(з. ассоциативности)} \\ &= \overline{A} \cup B && \text{(з. идемпотентности)} \end{aligned}$$

(б) Воспользуемся законами алгебры множеств:

$$\begin{aligned} \overline{(\overline{A} \cap \overline{(B \cup C)})} &= A \cup (B \cup C) = && \text{(з. де Моргана и дополн.)} \\ &= A \cup B \cup C && \text{(следствие з. ассоциат.)} \end{aligned}$$

(в) Опираясь на законы коммутативности и ассоциативности, имеем

$$\begin{aligned} &(A \cup B \cup C) \cap (A \cup \overline{B} \cup C) \cap \overline{(A \cup C)} = \\ &= \big((B \cup (A \cup C)\big) \cap \big(\overline{B} \cup (A \cup C)\big) \cap \overline{(A \cup C)}. \end{aligned}$$

Учитывая дистрибутивность, получаем

$$\begin{aligned} &\big((B \cup (A \cup C)\big) \cap \big(\overline{B} \cup (A \cup C)\big) \cap \overline{(A \cup C)} = \\ &= \big((B \cap \overline{B}) \cup (A \cup C)\big) \cap \overline{(A \cup C)}. \end{aligned}$$

Теперь применяем закон дополнения.

$$\big((B \cap \overline{B}) \cup (A \cup C)\big) \cap \overline{(A \cup C)} = \big(\varnothing \cup (A \cup C)\big) \cap \overline{(A \cup C)}.$$

По законам тождества и дополнения приходим к окончательному ответу:

$$\big(\varnothing \cup (A \cup C)\big) \cap \overline{(A \cup C)} = (A \cup C) \cap \overline{(A \cup C)} = \varnothing.$$

(г) Сделаем преобразования:

$$\begin{aligned}(A \setminus B) \setminus C = (A \cap \overline{B}) \setminus C = &\quad \text{(по определению «\»)}\\ = (A \cap \overline{B}) \cap \overline{C} = &\quad \text{(по определению «\»)}\\ = A \cap (\overline{B} \cap \overline{C}) = &\quad \text{(ввиду ассоциативности)}\\ = A \cap \overline{(B \cup C)} = &\quad \text{(по закону де Моргана)}\\ = A \setminus (B \cup C) &\quad \text{(по определению «\»)}\end{aligned}$$

(д) Учитывая определение симметрической разности и законы алгебры множеств, получаем:

$$\begin{aligned}A \triangle A = (A \setminus A) \cup (A \setminus A) = &\quad \text{(по определению «△»)}\\ = (A \setminus A) = &\quad \text{(по з. идемпотентности)}\\ = A \cap \overline{A} = &\quad \text{(по определению «\»)}\\ = \varnothing &\quad \text{(по з. дополнения)}\end{aligned}$$

Следовательно,

$$\begin{aligned}A \triangle A \triangle A = A \triangle \varnothing = &\\ = (A \setminus \varnothing) \cup (\varnothing \setminus A) = &\quad \text{(по определению «△»)}\\ = (A \cap \overline{\varnothing}) \cup (\varnothing \cap \overline{A}) = &\quad \text{(по определению «\»)}\\ = A \cup \varnothing = &\quad \text{(по з. дополнения,}\\ = A &\quad \text{коммутат. и тожд.)}\end{aligned}$$

3.8. Диаграмма Венна изображена на рис. Р3.3.

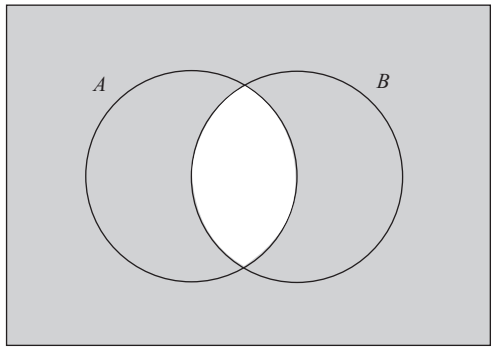


Рисунок Р3.3.

(а) По определению операции «∗» и закону идемпотентности получаем:

$$A * A = \overline{(A \cap A)} = \overline{A}.$$

(б) Учитывая предыдущий пункт задачи, воспользуемся определением операция «$*$», а затем применим законы де Моргана и дополнения.

$$(A * A) * (B * B) = \overline{A} * \overline{B} = \overline{(\overline{A} \cap \overline{B})} = A \cup B.$$

(в) Аналогичные соображения помогают решить и последний пункт задачи.

$$\begin{aligned}(A * B) * (A * B) &= \overline{(A * B)} = \\ &= \overline{(\overline{(A \cap B)})} = \\ &= A \cap B.\end{aligned}$$

3.9. (а) Диаграмма Венна приведена на рис. Р3.4.

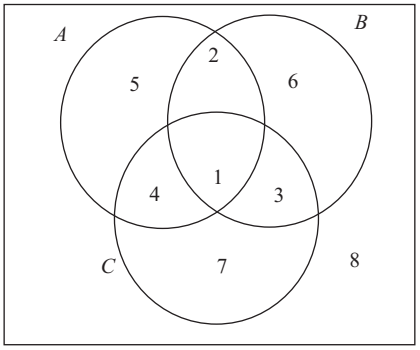


Рисунок Р3.4.

Обозначим различные области метками **1, 2, ... , 8**, как показано на рисунке, и предположим, что область **i** содержит n_i элементов. Тогда

$$\begin{aligned}&|A| + |B| + |C| = \\ &= (n_1 + n_2 + n_4 + n_5) + (n_1 + n_2 + n_3 + n_6) + \\ &\quad + (n_1 + n_3 + n_4 + n_7) = \\ &= 3n_1 + 2n_2 + 2n_3 + 2n_4 + n_5 + n_6 + n_7.\end{aligned}$$

С другой стороны,

$$\begin{aligned}&|A \cap B| + |B \cap C| + |A \cap C| = \\ &= (n_1 + n_2) + (n_1 + n_3) + (n_1 + n_4) = \\ &= 3n_1 + n_2 + n_3 + n_4.\end{aligned}$$

Кроме того,

$$|A \cap B \cap C| = n_1.$$

Следовательно,

$$\begin{aligned}
&|A| + |B| + |C| - \\
&\quad - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C| = \\
&= 3n_1 + 2n_2 + 2n_3 + 2n_4 + n_5 + n_6 + n_7 - \\
&\quad - 3n_1 - n_2 - n_3 - n_4 + n_1 = \\
&= n_1 + n_2 + n_3 + n_4 + n_5 + n_6 + n_7 = \\
&= |A \cup B \cup C|,
\end{aligned}$$

что и утверждалось[1].

(б) Обозначим через A множество студентов, изучающих бухгалтерию, через B — множество студентов, слушающих курс бизнеса, а через C — множество студентов, занимающихся туризмом. Из условия задачи и предыдущего пункта следует, что

$$|A \cup B \cup C| = 25 + 27 + 12 - 20 - 5 - 3 = 36.$$

Итак, 36 студентов посещают хотя бы один дополнительный курс. Изобразим схематически множества студентов (см. рис. 3.5).

Из нее видно, что четверо из студентов посещали исключительно лекции по туризму.

Можно предложить более аккуратное решение данной задачи[2]. Следуя введенным обозначениям, нам достаточно подсчитать мощность множества $C \setminus (A \cup B)$, поскольку в него входят студенты, изучающие туризм, но не посещающие ни бухгалтерию, ни бизнес. Для начала заметим, что множество C можно представить в виде объединения непересекающихся множеств:

$$C = (C \setminus A) \cup (C \cap A).$$

Поэтому

$$|C| = |C \setminus A| + |C \cap A|.$$

[1] Используя этот пример как подсказку, попытайтесь доказать формулу для вычисления мощности объединения трех множеств самостоятельно. — *Прим. перев.*

[2] Оно дано переводчиком. — *Прим. перев.*

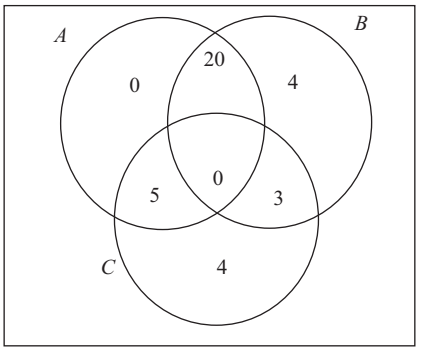


Рисунок Р3.5.

Из условия задачи нам известно, что $|C| = 12$ и $|C \cap A| = 5$. Значит, $|C \setminus A| = 12 - 5 = 7$. Теперь нужно обратить внимание на то, что к множеству $C \setminus A$ относятся студенты, изучающие туризм, но не посещающие бухгалтерию. Однако в множестве $C \setminus A$ могут остаться студенты, которые кроме туризма изучают еще и бизнес. Выбросив их, мы получим искомое множество: $(C \setminus A) \setminus B$. Для подсчета его мощности применим тот же прием.

$$\big|(C \setminus A) \setminus B\big| = |C \setminus A| - \big|(C \setminus A) \cap B\big|.$$

Самое трудное здесь вычислить мощность множества $(C \setminus A) \cap B$. К нему относятся те элементы множества C, которые принадлежат B, но не принадлежат A. Но по условию задачи любой элемент из C, принадлежащий B, не может принадлежать A, так как все три множества общих элементов не имеют. Значит,

$$(C \setminus A) \cap B = C \cap B \Rightarrow \big|(C \setminus A) \cap B\big| = \big|C \cap B\big| = 3.$$

Окончательный ответ:

$$12 - 5 - 3 = 4.$$

3.10. Элементы прямого произведения $A \times B$ имеют вид (a, b), где $a \in A$ и $b \in B$. Если $A \times B = B \times A$, то пара (a, b) из множества $A \times B$ должна принадлежать и $B \times A$, т. е. $a \in B$ и $b \in A$. Поскольку это верно для любого $a \in A$ и любого $b \in B$, мы имеем равенство множеств $A = B$.

Множество $A \times C$ состоит из упорядоченных пар (a, c), в которых $a \in A$ и $c \in C$. Если $A \times B = A \times C$, то $(a, c) \in A \times B$, откуда $c \in B$. Это нам дает включение: $C \subset B$. Меняя в нашем рассуждении множества B и C местами, можно увидеть, что $B \subset C$. Следовательно, $B = C$.

3.11. (а) Пусть $x \in A \times (B \cap C)$. Тогда $x = (a, t)$, где $a \in A$ и $t \in (B \cap C)$. Следовательно, $t \in B$, т. е. $(a, t) \in A \times B$ и, одновременно, $t \in C \Rightarrow (a, t) \in A \times C$. Значит, $(a, t) \in (A \times B) \cap (A \times C)$. Иными словами,

$$A \times (B \cap C) \subset (A \times B) \cap (A \times C).$$

И наоборот. Если $x \in (A \times B) \cap (A \times C)$, то $x \in A \times B$ и $x \in A \times C$. Следовательно, $x = (a, t)$, причем $a \in A$, а t принадлежит как B, так и C, т. е. $t \in B \cap C$. Таким образом, $x \in A \times (B \cap C)$. Этим рассуждением мы доказали обратное включение: $(A \times B) \cap (A \times C) \subset A \times (B \cap C)$.

Из доказанного вытекает требуемое равенство множеств.

(б) Пусть $x \in (A \cup B) \times C$. Тогда $x = (s, c)$, где $s \in (A \cup B)$ и $c \in C$. Поскольку s — элемент объединения, то либо $s \in A$, и тогда $x \in A \times C$, либо $s \in B$, т. е. $x \in B \times C$. В любом случае x принадлежит либо $A \times C$, либо $B \times C$. Значит, $x \in (A \times C) \cup (B \times C)$.

И наоборот. Если $x \in (A \times C) \cup (B \times C)$, то он представляется в виде $x = (s, c)$ где $c \in C$, а s лежит либо в множестве A, либо в множестве B. Иными словами, $s \in (A \cup B)$. Таким образом, $x = (s, c) \in (A \cup B) \times C$. Ввиду произвольности x можно заключить, что

$$(A \times C) \cup (B \times C) \subset (A \cup B) \times C.$$

Из двух включений следует нужное равенство множеств.

3.12. (а) $\mathcal{P}(A) = \big\{\varnothing, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\big\}$.

(б) Пусть $C \in \mathcal{P}(A) \cap \mathcal{P}(B)$. Тогда $C \subset A$ и $C \subset B$, поэтому $C \subset (A \cap B)$. Следовательно, $\mathcal{P}(A) \cap \mathcal{P}(B) \subset \mathcal{P}(A \cap B)$. Теперь возьмем $C \in \mathcal{P}(A \cap B)$. Тогда $C \subset (A \cap B)$, т. е. $C \subset A$ и $C \subset B$. Значит, $\mathcal{P}(A \cap B) \subset \mathcal{P}(A) \cap \mathcal{P}(B)$. Отсюда вытекает равенство $\mathcal{P}(A) \cap \mathcal{P}(B) = \mathcal{P}(A \cap B)$.

(в) Элементами объединения $\mathcal{P}(A) \cup \mathcal{P}(B)$ являются подмножества, лежащие в A, и подмножества, лежащие в B. Следовательно, $\mathcal{P}(A) \cup \mathcal{P}(B) \subset \mathcal{P}(A \cup B)$. Обратного же включения нет, поскольку подмножество объединения $A \cup B$ не обязательно целиком содержится либо в A, либо в B. Пусть, например, $A = \{1, 2, 3\}$, $B = \{4, 5\}$, а $C = \{1, 2, 5\}$. Тогда, конечно, $C \in \mathcal{P}(A \cup B)$, но, очевидно, $C \notin \mathcal{P}(A) \cup \mathcal{P}(B)$.

3.13. Характеристическим вектором множества A является вектор $a = (1, 1, 0, 1, 1, 0)$. Характеристический вектор множества B равен $b = (0, 0, 1, 0, 1, 0)$.

Вычислим характеристический вектор множества $A \cup \overline{B}$. Он равен

$$a \textbf{ или } (\textbf{не } b) = (1, 1, 0, 1, 1, 0) \textbf{ или } (1, 1, 0, 1, 0, 1) = \\ = (1, 1, 0, 1, 1, 1).$$

Следовательно, $A \cup \overline{B} = \{1, 2, 4, 5, 6\}$.

Характеристический вектор множества $A \triangle B$ равен

$$\begin{aligned} &(a \textbf{ и } (\textbf{не } b)) \textbf{ или } (b \textbf{ и } (\textbf{не } a)) = \\ &= \big((1, 1, 0, 1, 1, 0) \textbf{ и } (1, 1, 0, 1, 0, 1)\big) \textbf{ или} \\ &\quad \textbf{или } \big((0, 0, 1, 0, 1, 0) \textbf{ и } (0, 0, 1, 0, 0, 1)\big) = \\ &= (1, 1, 0, 1, 0, 0) \textbf{ или } (0, 0, 1, 0, 0, 0) = \\ &= (1, 1, 1, 1, 0, 0). \end{aligned}$$

Таким образом, $A \triangle B = \{1, 2, 3, 4\}$.

Набор упражнений 4

4.1. $\big\{(1, a), (1, c), (2, a), (2, c), (3, b), (3, c)\big\}$. Графическая форма отношения показана на рис. Р4.1.

4.2. $R = \big\{(1, 7), (2, 5), (3, 3), (4, 1)\big\}$.

$S = \big\{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 1), (2, 2), (2, 3), (2, 4), (3, 1), (3, 2), (3, 3), (4, 1), (4, 2), (5, 1)\big\}$.

$T = \{(n, n^2) : \ n \in \mathbb{N}\}$.

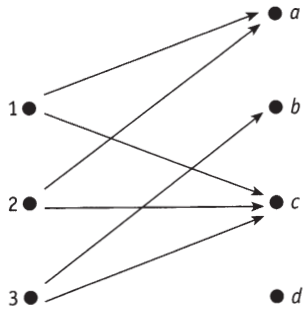


Рисунок Р4.1.

4.3. (а) Множество упорядоченных пар: $R = \{(1, 1), (1, 2), (1, 3), (1, 4), (3, 1), (3, 2), (3, 3), (3, 4)\}$.

(б) Графическая форма отношения представлена на рис. Р4.2.

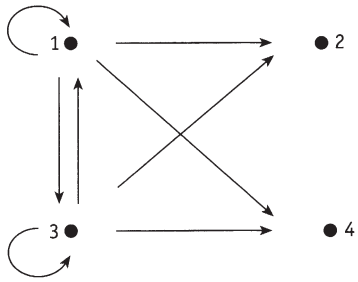


Рисунок Р4.2. Графическая форма отношения R

(в) Матрица отношения R имеет вид:

$$\begin{bmatrix} \text{И} & \text{И} & \text{И} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} \\ \text{И} & \text{И} & \text{И} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix}.$$

4.4. (а) Рефлексивно, симметрично и транзитивно.

(б) Транзитивно, но не рефлексивно и не симметрично.

(в) Симметрично, но не рефлексивно и не транзитивно.

(г) Рефлексивно и транзитивно, но не симметрично.

4.5. (а) Отношение симметрично, поскольку сумма $x + y$ совпадает с $y + x$.

Оно не рефлексивно, так как число $x + x$ всегда четно.

Отношение не транзитивно, ибо, например, 1+4 и 4+3 — нечетные числа, а $1 + 3$ — четное.

(б) Отношение рефлексивно ввиду того, что при любом x число $x + x$ — четно.

Ввиду равенства $x + y = y + x$ оно симметрично.

Транзитивность данного отношения следует из того факта, что суммы $x + y$ и $y + z$ будут четными, только если все слагаемые имеют одинаковую четность (все четные

или все нечетные). В любом из этих случаев сумма $x+z$ окажется четной.

(в) Это отношение симметрично ввиду коммутативности произведения: $xy = yx$.

Оно транзитивно, потому что из нечетности произведений xy и yz следует нечетность всех сомножителей: x, y и z. В частности, произведение xz тоже нечетно.

Но оно не рефлексивно, так как, например, $2 \cdot 2 = 4$ — число четное.

(г) Последнее отношение рефлексивно, поскольку $x + x^2 = x(x+1)$ — произведение двух последовательных натуральных чисел, одно из которых обязательно четно. Поэтому и их произведение всегда будет четным числом.

Оно транзитивно. Действительно, если $x+xy$ и $y+yz$ — четные числа, то либо x — четно (тогда и $x + xz$ — четно), либо x — нечетно (тогда все переменные: x, y и z — нечетны и, снова, $x + xz$ — четное число). Итак, в любом случае $x + xz$ — число четное, что доказывает транзитивность отношения.

Благодаря примеру: $2 + 2 \cdot 3 = 8$ — четное число, но $3 + 3 \cdot 2 = 9$ — нечетное число, можно утверждать, что наше отношение не симметрично.

4.6. (а) $R = \{(1, 9), (3, 3), (9, 1)\}$.

(б) $S = \{(3, 2), (6, 4), (9, 6), (12, 8)\}$.

(в) Транзитивным замыканием R является отношение

$$R \cup \{(1, 1), (9, 9)\}.$$

(г) Транзитивным замыканием S служит отношение

$$S \cup \{(9, 4)\}.$$

4.7. (а) «x на несколько лет старше, чем y».

(б) $x = 2^n y$ для некоторого натурального показателя n.

(в) $x < y$ (поскольку данное отношение транзитивно).

(г) «x является потомком y женского пола».

4.8. Замыканием по рефлексивности служит отношение, заданное упорядоченными парами:

$$\{(a, a),\ (b, b),\ (c, c),\ (d, d),\ (a, c),\ (a, d),\ (b, d),\ (c, a),\ (d, a)\}.$$

Замыкание по симметричности:

$$\{(a, a),\ (b, b),\ (c, c),\ (a, c),\ (a, d),\ (b, d),\ (c, a),\ (d, a),\ (d, b)\}.$$

Чтобы построить замыкание по транзитивности, добавим пару (b, a) (учитывая наличие в отношении пар (b, d) и (d, a)), (c, d) (из-за (c, a) и (a, d)) и (d, d) (так как в отношении уже есть пары (d, a) и (a, d)).

Теперь добавляем пару (b, c) (учитывая новую пару (b, a) и старую (a, c)).

Итак, мы получили замыкание по транзитивности:

$$\{(a, a),\ (b, b),\ (c, c),\ (a, c),\ (a, d),\ (b, d),\ (c, a),\ (d, a),\ (b, a)\ (c, d),\ (d, c),\ (d, d),\ (b, c)\}.$$

Если отношение не является антисимметричным, оно содержит пары (x, y) и (y, x), где $x \neq y$. Какие бы пары мы ни добавили к отношению, мы не сможем преодолеть эту неприятность. Поэтому строить замыкание по антисимметричности бессмысленно.

4.9. (а) Каждый класс эквивалентности состоит из всех книг, переплет которых имеет один и тот же цвет.

(б) Здесь есть два класса эквивалентности: множество всех четных целых чисел и множество всех нечетных целых чисел.

(в) Здесь тоже два класса эквивалентности. К одному относятся все женщины, а к другому — все мужчины.

(г) Каждый класс эквивалентности представляет собой окружность с центром в точке с координатами $(0, 0)$.

4.10. Так как $x^2 - x^2 = 0$ делится на 3, то отношение R рефлексивно.

Если $x^2 - y^2$ делится на 3, то $y^2 - x^2 = -(x^2 - y^2)$ тоже делится на 3. Значит отношение R симметрично.

Предположим, что разности $x^2 - y^2$ и $y^2 - z^2$ делятся на 3. Тогда $x^2 - z^2 = (x^2 - y^2) + (y^2 - z^2)$ — сумма чисел, делящихся на 3. Поэтому она тоже делится на 3, откуда следует транзитивность отношения R.

Класс эквивалентности, содержащий число 0, определяется по правилу: $E_0 = \{n : \ n^2$ делится на $3\}$. Так как n^2 кратно трем тогда и только тогда, когда само число n делится на 3, то $E_0 = \{0, \pm 3, \pm 6, \ldots\}$.

Класс эквивалентности, включающий 1, имеет вид:

$$E_1 = \{n : \ n^2 - 1 \text{ делится на } 3\}.$$

Поскольку $n^2 - 1 = (n-1)(n+1)$, то одно из чисел: $(n-1)$ или $(n+1)$ должно делиться на 3. Следовательно, $n = 3m \pm 1$, где m — целое число и, окончательно, $E_1 = \{\ldots -1, \ 1, \ 2, \ 4, \ 5, \ \ldots\}$.

Итак, ввиду равенства $E_0 \cup E_1 = \mathbb{Z}$, у нас есть только два класса эквивалентности: E_0 и E_1.

4.11. (а) Диаграмма Хассе изображена на рис. Р4.3.

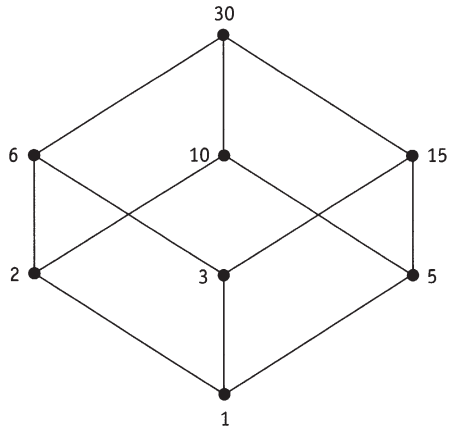


Рисунок Р4.3. Диаграмма Хассе

(б) Диаграмма Хассе частично упорядоченного множества данной подзадачи идентична предыдущей. Поэтому мы приведем только соответствие между вершинами диаграмм.

$$1 \leftrightarrow \varnothing, \quad 2 \leftrightarrow \{1\}, \quad 3 \leftrightarrow \{2\}, \quad 5 \leftrightarrow \{3\},$$

$$6 \leftrightarrow \{1, 2\}, \quad 10 \leftrightarrow \{1, 3\}, \quad 15 \leftrightarrow \{2, 3\}, \quad 30 \leftrightarrow \{1, 2, 3\}.$$

4.12. $R = \{(a, d),\ (a, g),\ (b, e),\ (b, g),\ (b, h),\ (c, e),\ (c, g),\ (c, h),\ (d, g),\ (e, g),\ (e, h)\}$.

Минимальные элементы: a, b, c и f.

Максимальные элементы: f, g и h.

4.13. Упорядоченный список слов выглядит следующим образом: *банан, банджо, бивень, бисквит, бутылка.*

Набор упражнений 5

5.1. $R^{-1} = \{(1, 1),\ (1, 3),\ (3, 2),\ (4, 2),\ (4, 3)\}$.

$S^{-1} = \{(1, 1),\ (1, 2),\ (1, 3),\ (2, 1),\ (2, 4)\}$.

$S \circ R = \{(1, 1),\ (1, 2),\ (2, 1),\ (2, 2),\ (3, 1),\ (3, 2)\}$.

$(S \circ R)^{-1} = \{(1, 1),\ (1, 2),\ (1, 3),\ (2, 1),\ (2, 2),\ (2, 3)\}$.

$R^{-1} \circ S^{-1} = \{(1, 1),\ (1, 2),\ (1, 3),\ (2, 1),\ (2, 2),\ (2, 3)\} = (S \circ R)^{-1}$.

5.2. R^{-1} — это отношение «... ребенок ...».

S^{-1} — это отношение «... брат или сестра ...».

$R \circ S$ — это отношение «... дядя ...».

$S^{-1} \circ R^{-1}$ — это отношение «... племянник или племянница ...».

$R \circ R$ — это отношение «... бабушка или дедушка ...».

5.3. Так как R — отношение частичного порядка, то оно рефлексивно, антисимметрично и транзитивно. Ввиду рефлексивности R, оно содержит все возможные пары вида (x, x). Значит и R^{-1} будет содержать все такие пары, т. е. R^{-1} — рефлексивно. Предположим, что $x\,R^{-1}\,y$ и $y\,R^{-1}\,x$. Тогда, по определению обратного отношения $y\,R\,x$ и $x\,R\,y$. Благодаря свойству антисимметричности, получаем, что $x = y$, откуда отношение R^{-1} тоже антисимметрично.

Предположим, что $x\,R^{-1}\,y$ и $y\,R^{-1}\,z$. Тогда $y\,R\,x$ и $z\,R\,y$. Поскольку R — транзитивное отношение, получаем, что $z\,R\,x$. Следовательно, $x\,R^{-1}\,z$, т. е. R^{-1} транзитивно.

Подводя итог нашим рассуждениям, можно сказать, что R^{-1} действительно является отношением частичного порядка. Очевидно, максимальный элемент относительно частичного порядка R^{-1} совпадает с минимальным элементом относитель-

но порядка R и наоборот, минимальный элемент относительно R^{-1} — это максимальный элемент относительно R.

5.4.

$$MN = \begin{bmatrix} \text{И} & \text{Л} & \text{Л} \\ \text{И} & \text{Л} & \text{И} \end{bmatrix} \begin{bmatrix} \text{И} & \text{Л} & \text{И} & \text{И} \\ \text{И} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{И} & \text{И} \end{bmatrix} = \begin{bmatrix} \text{И} & \text{Л} & \text{И} & \text{И} \\ \text{И} & \text{И} & \text{И} & \text{И} \end{bmatrix}.$$

Матрица MN представляет композицию отношений $S \circ R$.

5.5. Отношения пунктов (а) и (б) являются функциями.

Отношение пункта (в) функцией не является, поскольку элементу $0 \in A$ не поставлено в соответствие никакого элемента из множества B.

Отношения пункта (г) тоже не функция, ибо элементу $0 \in A$ в нем соответствуют два элемента множества B: 3 и 7.

Функция пункта (а) не инъективна, так как она переводит два разных элемента множества A (6 и 0) в один и тот же элемент $3 \in B$. Эта функция не является и сюръективной, ввиду того, что элемент $7 \in B$ не входит в ее множество значений.

Функция пункта (б) и инъективна, и сюръективна, т. е. она является биекцией.

5.6. (а) Если $f(a_1) = f(a_2)$, то $2a_1 + 1 = 2a_2 + 1$, откуда $a_1 = a_2$. Следовательно, f — инъективная функция. С другой стороны, функция f принимает только нечетные значения, значит, f — не сюръективна.

(б) Функция g не инъективна, поскольку, например, $g(4) = g(1) = 2$. Далее, так как $g(2m) = m$, где m — произвольное целое число, то множество значений этой функции совпадает со всем множеством $\mathbb{Z}$. Это означает, что g — сюръективная функция.

(в) Прежде всего заметим, что $h(n)$ — нечетное число, если n — четно, и четное, если n — нечетно. Следовательно, если $h(a_1) = h(a_2)$, то либо $a_1 + 1 = a_2 + 1$, либо $a_1 - 1 = a_2 - 1$. В обоих случаях получаем равенство: $a_1 = a_2$. Поэтому h — инъективная функция.

Если m — нечетное число, число $m - 1$ является четным и, по определению функции h,

$$h(m-1) = (m-1) + 1 = m.$$

Если же m — четное число, то число $m+1$ является нечетным и

$$h(m+1) = (m+1) - 1 = m.$$

Этим рассуждением мы показали, что каково бы ни было целое число m, найдется такое число n (равное $(m-1)$ или $(m+1)$), для которого $h(n) = m$. Значит, h — сюръективная функция.

5.7. Графики функций изображены на рис. Р5.1.

(а) Множество значений функции f совпадает с множеством

$$\{n^2 + 1 : \ n \in \mathbb{Z}\} = \{1, 2, 5, 10, \ldots\}.$$

Так как $f(-1) = f(1) = 2$, она не инъективна. Более того, уравнение $f(x) = 3$ не имеет целочисленного решения. Поэтому f не сюръективна.

(б) Множество значений функции g равно

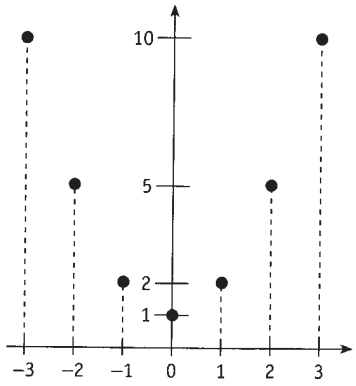
$$\{2, 4, 8, 16, \ldots\}.$$

Предположение $g(a) = g(b)$ равносильно равенству $2^a = 2^b$, которое возможно только при равных a и b. Таким образом, g — инъективная функция. Однако она не сюръективна, потому что уравнение $g(x) = 3$ не имеет целочисленного решения.

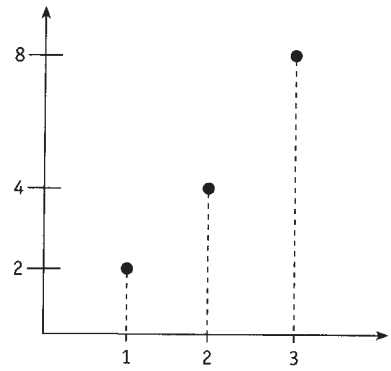
(в) Множество значений функции h совпадает с множеством $\mathbb{R}$. Если $h(a) = h(b)$, то $5a - 1 = 5b - 1$, т. е. $a = b$. Следовательно, h — инъективная функция. А так как h принимает любое вещественное значение, то она является и сюръективной функцией.

(г) Множество значений функции j совпадает со всем множеством $\mathbb{R}$. Так как $j(-1) = j(\frac{3}{2}) = 0$, j — не инъективна. Ее множество значений совпадает со всей областью значений, откуда следует ее сюръективность.

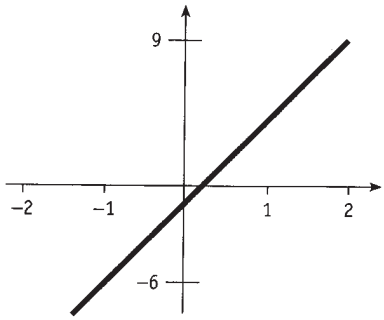
(д) К множеству значений функции k относится любое неотрицательное вещественное число, а отрицательных значений она не имеет. Тем самым мы показали, что k не сюръективна. Более того, она не является и инъективной функцией, так как для любых $x_1 < 0$ и $x_2 < 0$ имеет место равенство: $k(x_1) = k(x_2) = 0$.



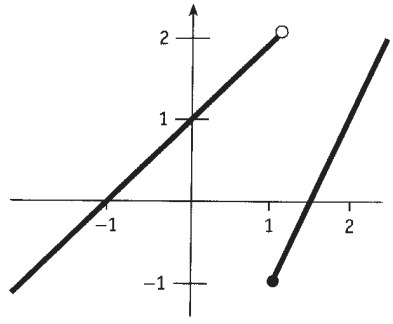
(а) $f : \mathbf{Z} \to \mathbf{N}$
$f(x) = x^2 + 1$



(б) $g : \mathbf{N} \to \mathbf{N}$
$g(x) = 2^x$

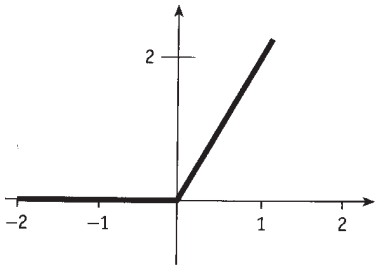


(в) $h : \mathbf{R} \to \mathbf{R}$
$h(x) = 5x - 1$

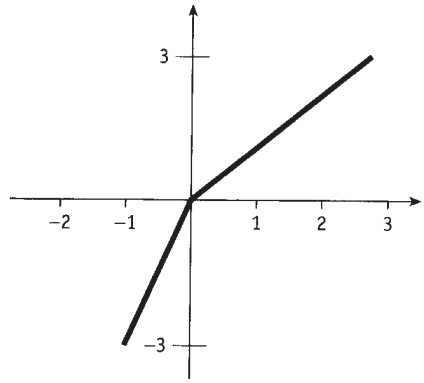


(г) $j : \mathbf{R} \to \mathbf{R}$

$$j(x) = \begin{cases} 2x - 3, & \text{если } x > 1 \\ x + 1, & \text{если } x < 1 \end{cases}$$



(д) $k : \mathbf{R} \to \mathbf{R}$
$k(x) = x + |x|$



(е) $l : \mathbf{R} \to \mathbf{R}$
$l(x) = 2x - |x|$

Рисунок Р5.1.

(е) Множество значений функции l — все вещественные числа, т. е. множество $\mathbb{R}$. В частности, она сюръективна.

Если $x \geqslant 0$, то $l(x) = x \geqslant 0$, а при $x < 0$ $l(x) = 3x < 0$. Значит, в случае совпадения значений $l(a) = l(b)$ аргументы a и b будут иметь один и тот же знак.

Если аргументы неотрицательны, то равенство значений функции l можно переписать в виде: $a = b$. Если же $a < 0$ и $b < 0$, то $l(a) = l(b)$ тогда и только тогда, когда $3a = 3b$, откуда $a = b$. Итак, в любом случае предположение $l(a) = l(b)$ влечет равенство $a = b$, т. е. l — инъективная функция.

5.8. (а) $f(-1) = \left\lfloor \frac{(-1)^2+1}{3} \right\rfloor = \left\lfloor \frac{2}{3} \right\rfloor = 0$. Аналогично, $f(0) = 0$, $f(1) = 0$ и $f(2) = 1$. Таким образом, множество значений функции f — это $\{0, 1\}$.

(б) Функция g не инъективна, так как, например, $g(0) = g(1) = 0$. Однако она сюръективна, поскольку

$$g(2n) = \left\lfloor \frac{2n}{2} \right\rfloor = n,$$

где n — произвольное целое число.

5.9. Если $f(a) = f(b)$, то $1+\frac{2}{a} = 1+\frac{2}{b}$, что влечет равенство $a = b$. Это означает инъективность функции g. Уравнение $f(x) = y$ переписывается в виде $1 + \frac{2}{x} = y$, откуда $x = \frac{2}{y-1}$. Последнее выражение не определено только для значения $y = 1$, которое не входит в область значений функции. По любому же другому элементу из множества B можно построить x, который переводится в этот элемент функцией f. Значит, f — сюръективна.

Мы показали, что f как инъективна, так и сюръективна. Таким образом, она является биекцией. Обратная к ней функция $f^{-1} : B \longrightarrow A$ определяется формулой $f^{-1}(x) = \frac{2}{x-1}$.

5.10. $(f \circ g)(x) = f(g(x)) = \begin{cases} (2x+1)^2, & \text{если } x \geqslant 0, \\ x^2, & \text{если } x < 0; \end{cases}$

$(g \circ f)(x) = g(f(x)) = 2x^2 + 1;$

$(g \circ g)(x) = g(g(x)) = \begin{cases} g(2x+1), & \text{если } x \geqslant 0, \\ g(-x), & \text{если } x < 0. \end{cases}$

Если $x \geqslant 0$, то $2x + 1 \geqslant 1$ и поэтому

$$g(2x + 1) = 2(2x + 1) + 1 = 4x + 3.$$

Если $x < 0$, то $-x > 0$. Значит, $g(-x) = 2(-x) + 1 = 1 - 2x$. Следовательно,

$$(g \circ g)(x) = \begin{cases} 4x + 3, & \text{если } x \geqslant 0, \\ 1 - 2x, & \text{если } x < 0. \end{cases}$$

5.11. (а) Равенство $(g \circ f)(a_1) = (g \circ f)(a_2)$ можно переписать в более удобном виде: $g(f(a_1)) = g(f(a_2))$. Ввиду инъективности функции g из него следует, что $f(a_1) = f(a_2)$. Кроме того, f — тоже инъективная функция. Значит, $a_1 = a_2$. Другими словами, $g \circ f$ — инъективная функция.

(б) Пусть $c \in C$. Так как g — сюръективная функция, найдется элемент $b \in B$, для которого $g(b) = c$. Для этого b, как следует из сюръективности функции f, отыщется элемент $a \in A$, при котором $f(a) = b$. Пропуская промежуточные рассуждения, получаем вывод: для любого $c \in C$ найдется такой элемент $a \in A$, что $g(f(a) = c$, что доказывает сюръективность композиции $g \circ f$.

(в) Предположим, что $f(a) = b$ и $g(b) = c$. Тогда, по определению композиции функций $(g \circ f)(a) = c$. Следовательно, $(g \circ f)^{-1}(c) = a$. Но $f^{-1}(b) = a$ и $g^{-1}(c) = b$. Поэтому

$$(f^{-1} \circ g^{-1})(c) = f^{-1}(g^{-1}(c)) = f^{-1}(b) = a.$$

Мы показали, что для любого элемента $c \in C$ имеет место равенство

$$(g \circ f)^{-1}(c) = (f^{-1} \circ g^{-1})(c),$$

что и требовалось.

5.12. (а) На игральной кости может выпасть одно из шести чисел. Иными словами, бросая кость, мы можем получить шесть различных исходов. Поэтому, бросая кость семь раз, мы увидим какое-то число по крайней мере два раза.

(б) Обозначим через A множество, состоящее из пар очков, которые выпадают на двух игральных костях. Количество же этих пар в множестве A будет равно числу бросаний костей. Пусть множество $B = \{2, 3, 4, \ldots, 12\}$ состоит из всех возможных сумм очков на паре костей.

Рассмотрим функцию $f : A \longrightarrow B$, сопоставляющую паре очков их сумму. Благодаря принципу Дирихле, мы можем утверждать, что если $|A| > |B|$, то какая-то из сумм выпадет дважды. Следовательно, в результате 12 бросаний костей одна из сумм очков встретится дважды.

(в) В игральных картах существует четыре масти: трефы, пики, бубны и червы. Следовательно, вытащив пять карт из колоды, мы обнаружим, что по крайней мере две из них одной масти.

(г) Обозначим через A множество вытащенных карт, а через B — множество четырех карточных мастей. Пусть функция $f : A \longrightarrow B$ сопоставляет каждой вытащенной карте ее масть. Из обобщенного принципа Дирихле следует, что при $|A| > 3|B|$ можно утверждать, что не менее четырех карт окажутся одной масти. Получаем ответ: достаточно вытащить 13 карт.

5.13. (а) Пусть A — множество семей, проживающих в селе, а B — множество различных пар месяцев. Рассмотрим функцию $f : A \longrightarrow B$, которая сопоставляет каждой семье пару месяцев, в которые родились их дети.

В году всего 12 месяцев, поэтому число *упорядоченных* пар месяцев равно $12 \cdot 12 = 144$. В двенадцать из них входит название одного и того же месяца, а в 132 остальных парах — месяцы разные. Поэтому, переставив их в паре, мы формально получим другую упорядоченную пару месяцев, но для нашей задачи порядок месяцев в паре значения не имеет. Поэтому число различных *неупорядоченных* пар месяцев, т. е. мощность множества B, равна

$$|B| = 12 + \frac{132}{2} = 78.$$

Так как $|A| = 79$, а $|B| = 78$, то, согласно принципу Дирихле, найдется две семьи, дети в которых родились в одни и те же месяцы.

(б) Обозначим через A множество детей, а через B — множество букв в алфавите и рассмотрим функцию $f\colon A \to B$, сопоставляющую ребенку первую букву его имени. По условию задачи $|A| = 2 \cdot 79 = 158$. С другой стороны, количество букв в алфавите, с которых теоретически может начинаться имя ребенка, — 31 (мы выбросили

знаки: «Ъ» и «Ь», с которых не может начинаться ни одно слово). Значит, $|B| = 31$. Таким образом, $|A| > 5|B|$, что, ввиду обобщенного принципа Дирихле, гарантирует нам, что найдется по крайней мере 6 детей в селе, имена которых начинаются с одной буквы.

5.14. (а) Выпишем пары четных чисел из множества S, дающих в сумме число 22: {2, 20}, {4, 18}, {6, 16}, {8, 14}, {10, 12}. Обозначим через A множество четных чисел, выбранных из множества S, а через B — множество выписанных выше пар. Пусть функция $f : A \longrightarrow B$ ставит в соответствие выбранному числу ту пару, в которой оно присутствует. Например, $f(6) = \{6, 16\}$. Опираясь на принцип Дирихле, мы можем утверждать, что если $|A| > |B|$, то найдется два числа из множества A, которые отображаются функцией f в одну и ту же пару. А это и означает, что среди выбранных найдутся два числа с суммой, равной 22. Осталось заметить, что $|B| = 6$. Следовательно, из множества S достаточно взять 6 чисел.

(б) Следуя указанию, рассмотрим функцию f, определенную на множестве S, и определим ее множество значений.

В множестве S находится 10 нечетных чисел, максимальное из которых равно 19. Наибольший нечетный делитель четных чисел из S — это один из 1, 3, 5, 7 или 9. Таким образом, множество значений B функции f состоит из всех нечетных чисел множества S.

Пусть A — подмножество в S, элементы которого — произвольные 11 чисел. Рассмотрим функцию $f : A \to B$, сопоставляющую числу $n \in A$ его наибольший нечетный делитель. Так как $|B| = 10$, то по принципу Дирихле найдутся два числа, у которых будет один и тот же наибольший нечетный делитель. Обозначим их через n и m. Тогда $n = 2^i b$, где b — нечетно[1], а $i = 0, 1, 2, 3, 4$. Аналогично, $m = 2^j b$, где $j = 0, 1, 2, 3, 4$. Если $i > j$, то

[1] Действительно, пусть наибольший нечетный делитель числа n равен b. Тогда число $n' = n/b$ — целое. Если n' делится не какое-то нечетное число, отличное от 1, скажем, на k, то и исходное число n будет на него делиться. Более того, число n должно делиться на произведение kb, которое является нечетным. Это противоречит максимальности b среди нечетных делителей числа n. Значит, $n' = n/b$ не может иметь ни одного нечетного делителя, отличного от 1. Поэтому $n' = 2^i$ для некоторого натурального показателя i. — *Прим. перев.*

m делит n. В противном случае число n делит m. Таким образом, мы доказали, что при выборе 11 различных чисел из множества S обязательно попадется пара чисел, одно из которых делит другое.

Набор упражнений 6

6.1. (а) По правилу произведения можно составить $5 \cdot 8 \cdot 7 = 280$ разных костюмов.

(б) Женщина может составить $5 \cdot 3 = 15$ разных нарядов из пяти юбок и трех блузок. В качестве альтернативного решения она может сменить блузку и юбку на платье. Следовательно, по правилу суммы у нее есть $15 + 6 = 21$ возможностей для смены нарядов.

(в) У Вас есть шесть различных десертов из одной порции мороженого, $6 \cdot 6 = 36$ десертов из двух порций и $6^3 = 216$ из трех. Это дает всего $6 + 36 + 216 = 258$ возможностей для выбора десерта.

6.2. (а) Как известно, существует 10 цифр: $0, 1, 2, 3, 4, 5, 6, 7, 8$ и 9. Однако в старшем разряде любого числа (в частности, шестизначного перевертыша) не может стоять цифра 0. Следовательно, у нас есть $9 \cdot 10^2 = 900$ возможностей для выбора первых трех цифр шестизначного перевертыша. Оставшиеся его цифры однозначно определяются по первым трем. Следовательно, всего существует 900 шестизначных перевертышей.

Аналогичное рассуждение можно применить и для подсчета семизначных перевертышей. Но проще всего заметить, что добавив в середину шестизначного перевертыша произвольную цифру, мы получим семизначный перевертыш. Таким образом, ответ в этом случае: $900 \cdot 10 = 9000$.

(б) На первом месте справа у такого числа может стоять 1, 3 или 5. Оставшиеся разряды можно заполнять любыми нечетными цифрами (их всего 5). Поэтому существует $3 \cdot 5^3 = 375$ четырехзначных чисел, не превосходящих 6000, в чьей записи участвуют только нечетные цифры.

(в) У нас есть $26^2 = 676$ возможностей для выбора первых двух символов пароля. Каждый из оставшихся четырех

может быть выбран из 36 символов (26 букв и 10 цифр). Это можно сделать $36^4 = 1\,679\,616$ способами. Следовательно, у нас есть $676 \cdot 1\,679\,616 = 1\,135\,420\,416$ различных паролей.

6.3. (а) На первом месте такого числа может стоять любая из четырех цифр (1, 2, 3, и 6), а на оставшихся — любая из пяти. Поэтому к множеству S относятся $4 \cdot 5^3 = 500$ чисел.

(б) Первый знак числа может быть одним из четырех. Оставшиеся три знака можно рассматривать как упорядоченную выборку без повторений трех объектов из четырех возможных (одного 0 и трех не задействованных еще цифр). Всего таких выборок — $4 \cdot 3 \cdot 2 = 24$. Значит, $4 \cdot 24 = 96$ чисел из S не имеют в своей записи повторяющихся цифр.

(в) Четное число из множества S может оканчиваться цифрами: 0, 2 или 6. Если последняя цифра равна 0, то первые три мы можем выбрать $4 \cdot 3 \cdot 2 = 24$ способами. Если последняя цифра — 2 или 6, то (так как число не может начинаться с 0) первые три можно выбрать $3 \cdot 3 \cdot 2 = 18$ способами. По правилу суммы среди чисел пункта (б) ровно $24 + 18 + 18 = 60$ четных.

(г) Чтобы число из пункта (б) было больше, чем 4 000, оно должно начинаться с 4, 5 или 6. Однако мы можем составлять числа из цифр 0, 1, 2, 3, и 6. Значит, нужные нам числа начинаются с 6. Остальные 3 цифры мы можем написать $4 \cdot 3 \cdot 2 = 48$ способами. Значит, 48 чисел, удовлетворяющих условиям п. (б), превосходят 4 000.

6.4. (а) $P(7,\, 2) = 42$, $P(8,\, 5) = 6\,720$, $P(6,\, 4) = 360$, $P(n,\, 1) = n$ и $P(n,\, n-1) = n!$.

(б) $C(10,\, 7) = 120$, $C(9,\, 2) = 36$, $C(8,\, 6) = 28$, $C(n,\, 1) = C(n,\, n-1) = n$.

Число способов, которыми мы можем выбрать k различных элементов из n, совпадает с числом способов выбора $n - k$ элементов из n. Действительно, отбирая элементы, мы можем либо взять нужное количество, либо выбросить лишнее. Поэтому $C(n,\, k) = C(n,\, n-k)$.

Альтернативное доказательство этого равенства получается из преобразований:

$$C(n,\, n-k) = \frac{n!}{(n-(n-k))!(n-k)!} = \frac{n!}{k!(n-k)!} = C(n,\, k).$$

6.5. (а) При выборе призеров порядок существенен, а повторения запрещены. Поэтому существует $P(17,\, 3) = 4080$ возможностей распределения призовых мест.

(б) Здесь, как и в предыдущей задаче, порядок существенен и повторения запрещены. Следовательно, у нас есть $P(20,\, 2) = 380$ способов выбора председателя и секретаря комитета.

(в) Придумывая пароль, на первые два места мы можем поставить любые две строчные буквы. Это можно сделать $P(26,\, 2) = 650$ способами. После того как мы использовали две буквы для начала пароля, у нас осталось 10 цифр и 24 буквы для продолжения пароля, поскольку символы в нем не могут повторяться. Значит, для второй части пароля у нас есть $P(34,\, 4) = 1\,113\,024$ возможности. Следовательно, по правилу произведения, мы можем написать $650 \cdot 1\,113\,024 = 723\,465\,600$ различных паролей.

6.6. (а) Порядок отбора игроков в основной состав значения не имеет и повторы не разрешены. Стало быть, у нас есть $C(18,\, 11) = 31\,824$ возможностей для выбора основного состава команды.

(б) Женскую часть жюри мы можем выбрать $C(8,\, 5) = 56$ способами, а мужскую — $C(11,\, 7) = 330$ способами. Таким образом, существует $56 \cdot 330 = 18\,480$ различных составов жюри.

(в) Существует $C(5,\, 3) = 10$ способов отобрать трех профессиональных игроков и $C(5,\, 1) = 5$ возможностей для выбора любителя. Это дает $10 \cdot 5 = 50$ возможных команд, в которые входит только один любитель. Число возможных команд профессионалов равно $C(5,\, 4) = 5$. Аналогично мы можем набрать 5 любительских команд. Следовательно, у нас есть $5 + 5 = 10$ команд, состоящих только из профессионалов или только из любителей.

6.7. (а) Можно составить $C(12,\, 3) = 220$ разных комитетов.

(б) Существует $C(8,\,3) = 56$ комитетов, не включающих в себя представителей Либерал-Демократической партии. Так как можно составить 220 комитетов без каких-либо ограничений на его членов, то количество комитетов, в которые входит хотя бы один либерал-демократ, равно $220 - 56 = 164$.

(в) Количество разных комитетов, в которые не входит представитель партии лейбористов, равно $C(9,\,3) = 84$. Если в комитет не входят консерваторы, то у нас есть $C(7,\,3) = 35$ возможностей для их составления. Таким образом, у нас есть $84 + 35 = 119$ различных комитетов. Однако при таком подсчете комитеты, состоящие только из либерал-демократов, мы учитывали дважды. Их количество равно $C(4,\,3) = 4$. Окончательный ответ: существует 115 разных комитетов, удовлетворяющих условию задачи.

(г) Можно составить $C(5,\,2)\cdot C(3,\,1) = 10\cdot 3 = 30$ комитетов, состоящих из двух консерваторов и одного лейбориста, $C(5,\,1)\cdot C(3,\,2) = 15$ комитетов, состоящих из одного консерватора и двух лейбористов. Кроме того, у нас есть $C(5,\,1)\cdot C(3,\,1)\cdot C(4,\,1) = 60$ комитетов, в которые входят представители всех партий. Это дает нам $30 + 15 + 60 = 105$ возможностей.

6.8. (а) Существует $C(8,\,2)\cdot C(5,\,2)\cdot C(3,\,2) = 840$ разных составов совещания, на которое приглашены по два представителя от каждого отдела.

(б) Если в совещании не участвуют представители производства, то существует $C(8,\,6) = 28$ возможностей созыва совещания. Если в совещании участвует один представитель производства, то существует

$$C(8,\,1)\cdot C(8,\,5) = 448$$

возможностей. Таким образом, всего $28 + 448 = 476$ разных составов совещаний включают в себя меньше, чем двоих представителей производства. Так как количество составов совещаний, на которые не наложены ограничения, равно $C(16,\,6) = 8\,008$, то существует 7532 требуемых составов.

(в) Обозначим через U множество всех составов совещаний. Как следует из предыдущего п. задачи, его мощность

равна $|U| = 8\,008$. Пусть X — множество составов совещаний, на которых присутствуют представители каждого из отделов. В этом п. задачи нам необходимо определить мощность множества X. Однако, после зрелого размышления, можно понять, что проще вычислить мощность его дополнения $\overline{x} = U \setminus X$, откуда легко будет получить ответ. Действительно, множество $\overline{X}$ включает в себя составы тех совещаний, на которых отсутствовали представители хотя бы одного из отделов.

Пусть A — множество составов совещаний, на которые не пригласили производственников, B — множество составов совещаний без представителей отдела сбыта и, наконец, C — множество составов совещаний, на которых не было бухгалтеров. Очевидно,

$$\overline{X} = A \cup B \cup C.$$

Мы уже умеем вычислять мощность объединения трех конечных множеств (см. стр. 60). Применим наши знания к этой конкретной ситуации.

$$\begin{aligned} |\overline{X}| = |A \cup B \cup C| = \\ = |A| + |B| + |C| - \\ - |A \cap B| - |A \cup C| - |B \cap C| + |A \cap B \cap C|. \end{aligned}$$

Вычислим мощность каждого из множеств, участвующих в формуле, отдельно:

$$\begin{aligned} |A| &= C(8,\, 6) = 28, \\ |B| &= C(11,\, 6) = 462, \\ |C| &= C(13,\, 6) = 1\,716. \end{aligned}$$

Множество $A \cap B$ включает в себя составы совещаний, на которые пригласили только бухгалтеров. Но в совещании должны принимать участие 6 человек, а бухгалтеров — всего трое. Следовательно, $A \cap B = \varnothing$ и $|A \cap B| = 0$. Аналогично, $A \cap C$ состоит из тех составов совещаний, на которых присутствовали только представители отдела сбыта. Ввиду малочисленности таких представителей, имеем: $|A \cap C| = 0$.

Непустым оказывается только пересечение $B \cap C$, состоящее из составов совещаний, на которых были только

производственники. По стандартной формуле комбинаторики имеем:

$$|B \cap C| = C(8,\, 6) = 28.$$

Осталось заметить, что $A \cap B \cap C = \varnothing$, поскольку, например, $A \cap B = \varnothing$. Итак,

$$\begin{aligned} |\overline{X}| &= |A \cup B \cup C| = \\ &= 28 + 462 + 1\,716 - 28 = 2\,178. \end{aligned}$$

Следовательно,

$$|X| = |U| - |\overline{X}| = 5\,830.$$

6.9. (а) Каждый заказ можно рассматривать как неупорядоченную выборку с повторениями шести объектов пяти возможных типов. Их число равно $C(n+k-1,\, n-1)$, где $n = 5$ и $k = 6$. Значит, официант может получить $C(10,\, 4) = 210$ различных заказов.

(б) Каждый букет — это неупорядоченная выборка 12 роз с повторениями четырех возможных типов. Следовательно, у Вас есть $C(4 + 12 - 1, 4 - 1) = C(15,\, 3) = 455$ возможностей для составления букета.

6.10. (а) Набор открыток — это неупорядоченная выборка с повторениями пяти объектов четырех возможных типов. Поэтому Вы можете составить

$$C(4 + 5 - 1,\, 4 - 1) = C(8,\, 3) = 56$$

наборов открыток.

(б) Выбрать два типа открыток из четырех Вы можете шестью способами, поскольку $C(4,\, 2) = 6$. При каждом выборе Вы можете сделать неупорядоченную выборку с повторениями пяти объектов из двух выбранных типов. Это можно осуществить

$$C(2 + 5 - 1,\, 2 - 1) = C(6,\, 1) = 6$$

способами.

Хотелось бы теперь перемножить шесть на шесть и получить ответ. Однако при таком поспешном решении легко допустить ошибку. Действительно, предположим, мы

ограничились типами A и B рождественских открыток. Тогда среди всевозможных наборов пяти открыток этих типов окажутся два набора, целиком состоящие из открыток одного типа: $5A$ и $5B$. Если же мы ограничимся типами A и C, то среди наборов открыток таких типов будут наборы $5A$ и $5C$. Значит, воспользовавшись правилом произведения, мы несколько раз учтем однотипные наборы открыток.

Заметим теперь, что число выборок пяти открыток двух типов с условием, что в них присутствуют открытки обоих типов, равно $6-2=4$. Следовательно, количество способов, которыми мы можем отобрать 5 не однотипных открыток двух видов, равно $4\cdot 6 = 24$. Осталось учесть 4 однотипных набора. Окончательный ответ: 28.

6.11. (а) Восьмая, девятая и десятая строки треугольника Паскаля:

$$\begin{array}{ccccccccccc} & & & 1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \\ & & 1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1 \\ 1 & 9 & 36 & 84 & 126 & 126 & 84 & 36 & 9 & 1. \end{array}$$

(б) Взяв подходящие числа из восьмой строки треугольника Паскаля, получим

$$\begin{aligned} 1+2\cdot 7+21 &= 36,\\ 7+2\cdot 21+35 &= 84,\\ 21+2\cdot 35+35 &= 126, \end{aligned}$$

и т. д. Таким образом мы можем получить все числа десятой строки.

(в) По формуле Паскаля

$$C(n,\,k)+C(n,\,k+1)=C(n+1,\,k+1)$$

и

$$C(n,\,k+1)+C(n,\,k+2)=C(n+1,\,k+2).$$

Сложив эти тождества, придем к равенству

$$\begin{aligned} C(n,\,k)+2C(n,\,k+1)+C(n,\,k+2) &=\\ =C(n+1,\,k+1)+C(n+1,\,k+2). & \end{aligned}$$

Вновь воспользовавшись формулой Паскаля, получим требуемое:

$$C(n,\,k)+2C(n,\,k+1)+C(n,\,k+2)=C(n+2,\,k+2).$$

6.12. (а) Положив $a = b = 1$ в биноме Ньютона, мы получим

$$C(n,\,0) + C(n,\,1) + \cdots + C)n,\,n) = (1+1)^n = 2^n.$$

Число k-элементных подмножество множества S мощности n совпадает с числом способов выбора k элементов из n возможных без учета порядка и повторений. Как мы уже знаем, оно равно в точности $C(n,\,k)$. Подмножества в S могут состоять из 0 или 1, или 2, или ... или n элементов. Значит, в n-элементном подмножестве содержится

$$C(n,\,0) + C(n,\,1) + \cdots + C)n,\,n) = 2^n$$

различных подмножеств.

(б) Подстановка $a = 1$, $b = -1$ в бином Ньютона дает

$$C(n,\,0) - C(n,\,1) + C(n,\,2) - \cdots + (-1)^n C(n,\,n) = \\ = (1-1)^n = 0.$$

6.13. (а) Существует 11! перестановок букв А, Б, Р, А, К, А, Д, А, Б, Р и А. Так как пять букв А, две буквы Б и две буквы Р невозможно отличить друг от друга, то существует

$$\frac{11!}{5!2!2!} = 83\,160$$

различных «слов».

(б) Переставляя буквы А, Б, Р, А, А, Д, А, Б, Р и А, мы можем составить

$$\frac{10!}{5!2!2!} = 7560$$

различных «слов». Ровно столько «слов», начинающихся с буквы К, можно получить, переставляя буквы в слове «АБРАКАДАБРА».

(в) Эта задача легко решается, если сочетание «ББ» считать одной буквой. Итак, у нас есть

$$\frac{10!}{5!2!} = 15\,120$$

разных слов, обладающих указанным в условии задачи свойством.

6.14. (а) Искомый коэффициент равен $C(8,\,5) = 56$.

(б) Коэффициент при xy^3z^4, получающийся при раскрытии скобок в выражении $(x+y+z)^8$, равен

$$\frac{8!}{1!3!4!} = 280.$$

(в) Коэффициент при ab^2cd, получающийся при раскрытии скобок в выражении $(a+b+c+d)^5$, равен

$$\frac{5!}{1!2!1!1!} = 60.$$

Другими словами, пятая степень суммы $a+b+c+d$ имеет член $60ab^2cd$. Пусть $a = x$, $b = 2y$, $c = z$ и $d = -1$. Тогда, раскрывая скобки в $(x+2y+z-1)^5$, мы получим член $60x(2y)^2z(-1) = -240xy^2z$. Следовательно, коэффициент при xy^2z в выражении $(x+2y+z-1)^5$ равен -240.

Набор упражнений 7

7.1. Каждое ребро графа G соединяет две вершины. Стало быть, оно вносит вклад в степень каждой из них, равный 1. Таким образом, суммируя степени всех вершин, мы можем подсчитать и количество ребер графа. Единственное, о чем нужно помнить, это то, что каждое ребро uv мы подсчитаем дважды, учитывая степени вершины u и v. Следовательно, сумма степеней всех вершин равна удвоенному числу ребер в простом графе.

Каждая вершина полного графа K_n соединена с любой другой его вершиной. Значит, степень произвольной вершины K_n равна $(n-1)$. При этом сумма степеней всех его вершин, будет равна $n(n-1)$. По предыдущей лемме получаем, что число ребер полного графа равно $\frac{n(n-1)}{2}$.

Как мы уже поняли, степень любой вершины полного графа K_n равна $(n-1)$. С другой стороны, нужно вспомнить, что граф G является эйлеровым тогда и только тогда, когда любая его вершина имеет четную степень. Следовательно K_n — эйлеров граф, если и только если n — нечетное число, большее 1.

7.2. Пусть $G = (V,\,E)$ — простой граф с $n \geqslant 2$ вершинами. Обозначим через S множество степеней всех его вершин. Ввиду

простоты графа G, максимальная степень его вершины может быть равна $n-1$. Следовательно,

$$S \subset \{0,\ 1,\ 2,\ \ldots,\ n-1\}.$$

Однако S не может содержать одновременно и 0, и $n-1$, в противном случае, вершина степени $n-1$ соединена со всеми остальными вершинами графа. В частности и с той, которая имеет степень 0, что невозможно. Итак, $|S| \leqslant n-1$. Рассмотрим функцию $f : V \longrightarrow S$, сопоставляющую каждой вершине графа G ее степень. Поскольку $|V| = n$, то имеет место неравенство: $|V| > |S|$. Так что, по принципу Дирихле, найдутся две вершины, на которых функция f принимает одно и то же значение, т. е. две вершины одинаковых степеней.

7.3. Пронумеруем строки и столбцы матрицы смежности числами от 1 до 6, соответственно номерам вершин графа. Изобразим граф на рис. Р7.1.

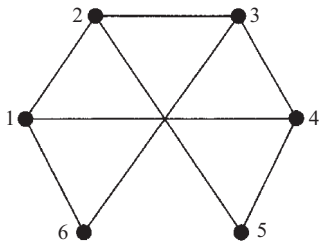


Рисунок Р7.1.

Матрица смежности полного графа K_n имеет n строк и n столбцов. На главной ее диагонали (пересечение строк со столбцами того же номера) стоит Л, а вне ее — И.

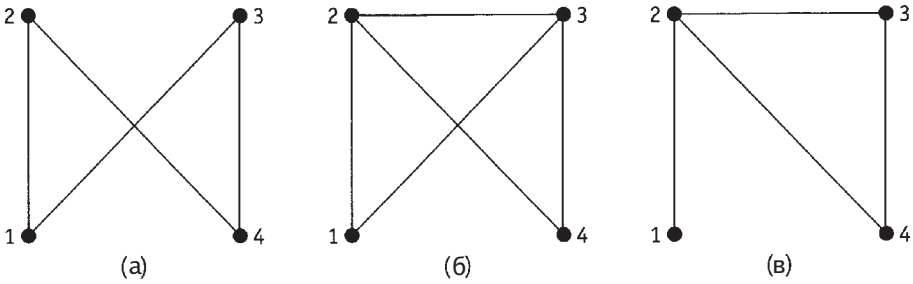


Рисунок Р7.2.

7.4. Пронумеровав строки и столбцы матриц числами от 1 до 4, изобразим графы, соответствующие этим матрицам (рис. Р7.2).

Обозначив подходящим образом вершины графов G, H и K, можно убедиться, что на рис. Р7.2 (а) изображен граф H, на рис. Р7.2 (б) — граф K, а на рис. Р7.2 (в) — G.

7.5. На рис. Р7.3 приведены те графы из условия задачи, которые являются подграфами графа из упражнения 7.3.

7.6. Единственным гамильтоновым циклом является цикл

$$1\,2\,6\,8\,7\,5\,4\,3\,1.$$

Существует несколько циклов длин 3, 4, 5, 6 и 7.

Циклы длины 3: $2\,8\,6\,2$, $2\,8\,7\,2$, $4\,7\,8\,4$ и $4\,5\,7\,4$.

Циклы длины 4: $2\,7\,8\,6\,2$, $2\,8\,4\,7\,2$ и $4\,5\,7\,8\,4$.

Циклы длины 5: $1\,2\,7\,4\,3\,1$, $1\,2\,8\,4\,3\,1$, $2\,6\,8\,4\,7\,2$ и $2\,7\,5\,4\,8\,2$.

Циклы длины 6: $1\,2\,7\,5\,4\,3\,1$, $1\,2\,8\,7\,4\,3\,1$, $1\,2\,7\,8\,4\,3\,1$ и $1\,2\,6\,8\,4\,3\,1$.

Циклы длины 7: $1\,2\,6\,8\,7\,4\,3\,1$ и $1\,3\,4\,5\,7\,8\,2\,1$.

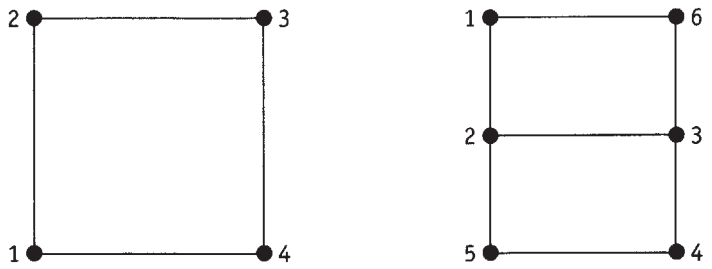


Рисунок Р7.3.

7.7. Цикл длины 9: $a\,b\,f\,j\,i\,c\,d\,g\,h\,a$.

Обозначим через C гипотетический гамильтонов цикл графа P. Поскольку гамильтонов цикл должен обойти как вершины, расположенные в вершинах внешнего пятиугольника, так и помещающиеся на лучах звезды, он должен содержать одно из ребер: ah, bf, ci, dg или ej. Без ограничения общности можно предположить, что цикл C включает в себя ребро ah. Теперь, чтобы включить в цикл C вершину a, нужно использовать ребро ab или ae. Ввиду симметричности графа, не нарушая общности рассуждений, будем считать, что в цикл C включено ребро ae. Напомним, что каждая вершина

в гамильтоновом цикле должна иметь индекс 2 (одно ребро подходит к ней, а другое выходит из нее). Поэтому цикл C не может содержать ребра ab. С другой стороны, мы должны пройти через вершину b (войти в нее и выйти). Поэтому цикл C содержит оба ребра: bf и bc. Вершина e должна входить в гамильтонов цикл со степенью 2. Поэтому цикл C должен содержать еще одно из ребер: ej или de. Разберем эти случаи отдельно.

Предположим, что в цикл C входит ребро ej. Тогда ребро ed нам придется исключить из рассмотрения. Изобразим граф Петерсена на рис. Р7.4, выделив на нем ребра, которые мы включили в цикл C, и удалив те, которые в C уже содержаться не могут.

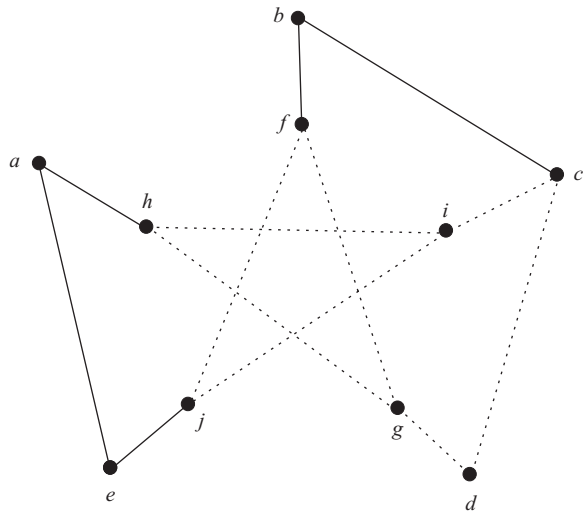


Рисунок Р7.4. Отдельные ребра гипотетического цикла C

Из рисунка видно, что через вершину d можно пройти только по ребрам dg и dc. Значит, их необходимо включить в цикл C. После этого ребро ci оказывается лишним и его приходится удалить. Теперь в нашем графе осталось два ребра, инцидентных вершине i. Включая вершину i в цикл C, мы с необходимостью должны добавить к нему и ребра hj и ji, удалив при этом hg и jf. Подводя итог этому этапу рассуждений, изобразим наши результаты на рис. Р7.5.

Теперь мы вынуждены включить в цикл C ребро fg, дабы иметь возможность пройти через вершины f и g. В результа-

те C будет состоять из двух несвязных частей, т. е. вообще не будет циклом, не говоря уже о его гамильтоновости. Итак, выбор ребра ej ведет к неудаче в построении гамильтонова цикла.

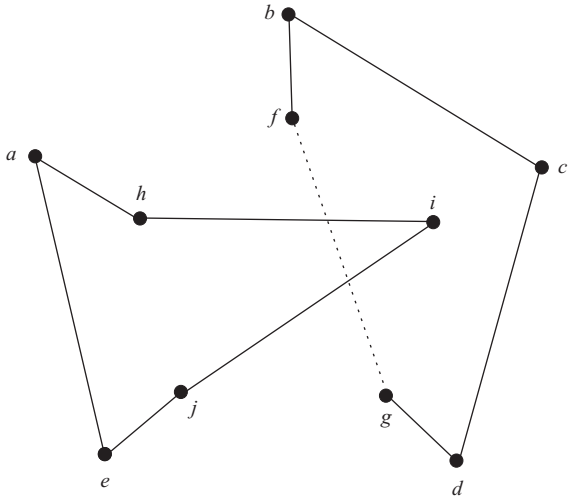


Рисунок Р7.5.

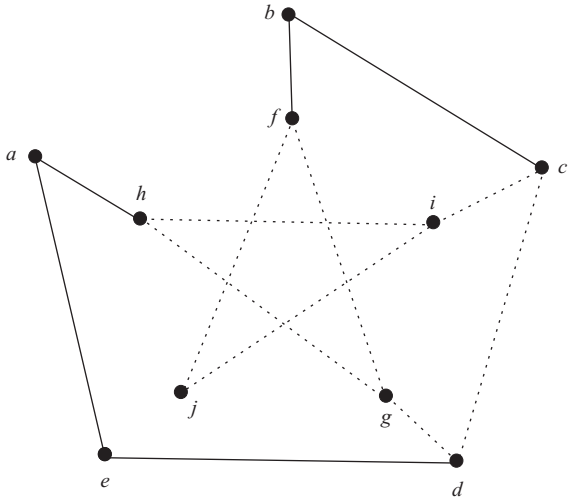


Рисунок Р7.6.

Допустим теперь, что мы включили в цикл C ребро ed, а ребро ej, наоборот, выбросили. В этом случае наши построения будут выглядеть так, как показано на рис. Р7.6.

На рисунке рис. P7.6 отчетливо видно, что для присоединения вершины j к циклу C нам необходимо добавить к нему ребра fj и ji. Но тогда ребро fg оказывается лишним, поскольку два остальных ребра, подходящие к вершине f, уже вошли в цикл C. Выбросим ребро fg.

Теперь у вершины g осталось только два ребра: gd и gh. Поэтому нам нужно присоединить их к циклу, а ребра hi и cd отбросить как ненужные. В результате получим рис. P7.7, одного взгляда на который достаточно, чтобы понять: мы должны присоединить к циклу ребро ic, что опять приводит нас к двум несвязным компонентам.

Полученное противоречие доказывает, что в графе P нет гамильтоновых циклов. Иными словами, P — не гамильтонов граф.

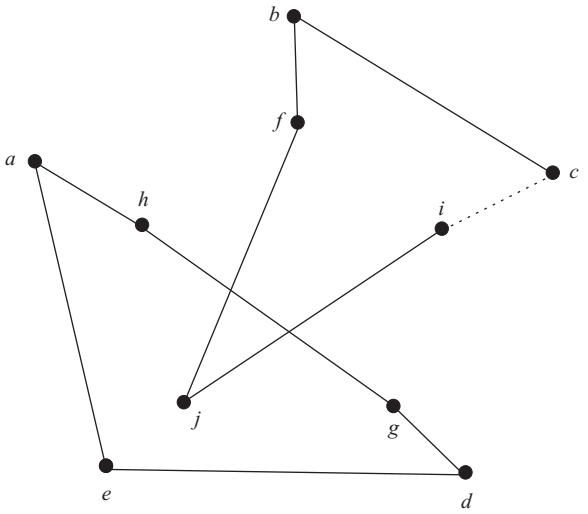


Рисунок P7.7.

7.8. (а) Алгоритм генерирует маршрут $A\,C\,B\,D\,E\,A$ общей длины $4+2+3+2+8=19$.

(б) Алгоритм генерирует маршрут $D\,E\,B\,C\,A\,D$ общей длины $2+4+2+4+5=17$.

7.9. Пронумеровав строки и столбцы матриц смежности числами от 1 до 6, построим соответствующие графы на рис. P7.8.

(а) Граф, изображенный на рис. P7.8 (а) не является деревом, поскольку содержит цикл 2 4 5 3 6 2.

(б) Граф, изображенный на рис. Р7.8 (б) — дерево, поскольку он связен, имеет шесть вершин и пять ребер.

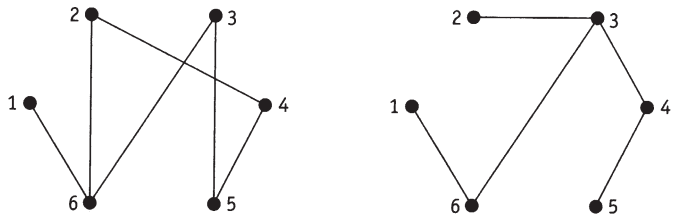


Рисунок Р7.8.

7.10. Пусть n — общее число вершин нашего дерева. Тогда сумма их степеней равна

$$3+3+3+2+2+2+2+(n-7)=n+10.$$

С другой стороны, число ребер дерева T должно быть ровно на единицу меньше числа вершин, т. е. число ребер равно $n-1$. Опираясь на лемму об эстафете, получаем соотношение:

$$n+10=2(n-1).$$

Следовательно, $n=12$, откуда вытекает, что число вершин дерева T со степенью 1 равно пяти.

7.11. (а) Обозначим компоненты леса G через T_1, T_2, ..., T_k. Будем считать, что дерево T_i имеет n_i вершин $(i=1,\ldots,k)$. У каждого дерева T_i есть (n_i-1) ребер. Поэтому общее количество ребер леса G равно

$$(n_1-1)+(n_2-1)+\cdots+(n_k-1)=n_1+n_2+\cdots+n_k-k=n-k.$$

(б) Достаточно показать, что любое дерево с двумя и более вершинами обладает по крайней мере двумя вершинами степени 1. Пусть T — дерево с $r\geqslant 2$ вершинами. Допустим, что у него не более одной вершины степени 1. Тогда по крайней мере $r-1$ его вершина будет иметь степень 2 и более. Следовательно, сумма степеней его вершин будет не меньше, чем $1+2(r-1)=2r-1$. С другой стороны, у дерева T должно быть $r-1$ ребро и, по лемме об эстафете, удвоенное количество ребер $2(r-1)$

должно совпадать с суммой степеней вершин, которая, как мы выяснили, не меньше, чем $2r - 1$. Полученное противоречие доказывает требуемый факт.

(в) Опираясь на решенный п. (а) задачи, можно сказать, что искомый лес G имеет $9-6 = 3$ компоненты связности. По п. (б) по крайней мере одна из этих компонент должна состоять из единственной вершины (без ребер). Оставшиеся две компоненты должны иметь восемь вершин и шесть ребер. Можно нарисовать несколько графов, удовлетворяющих последнему условию. Один из них изображен на рис. Р7.9. Он имеет пять вершин степени 1.

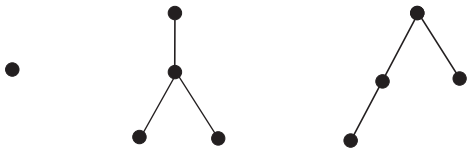


Рисунок Р7.9.

7.12. Последовательно выбираем ребра наименьшего веса до тех пор, пока не присоединим все вершины графа, следя при этом, чтобы не появлялись циклы. Одна из последовательностей выбора ребер следующая: BE, AB, EF, EG, FH, DG и CD. МОТ, полученное в результате, представлено на рис. Р7.10.

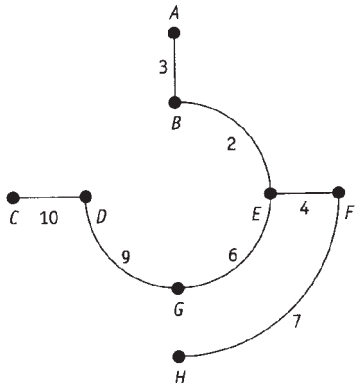


Рисунок Р7.10.

7.13. Обозначим через A город Атлон, через D — Дублин, через G — Голуэй, через L — Лимерик, через S — Слайго и через

W — Уэксфорд. Алгоритм выбирает следующие ребра графа: AG (веса 56), GL (веса 64) и AS (веса 71). Он забракует ребро AL (веса 73), поскольку добавив его, получит цикл. Далее алгоритм отберет ребро AD (веса 78), а ребро GS (веса 85) забракует (иначе образуется цикл). Последним алгоритм присоединит ребро DW (весом 96). В результате получится MOT (общего веса 365), изображенное на рис. P7.11.

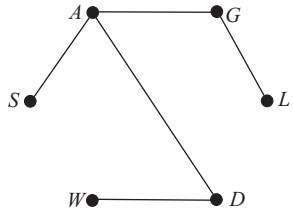


Рисунок P7.11. MOT городов Ирландии

7.14. (а) Смотри рис. P7.12.

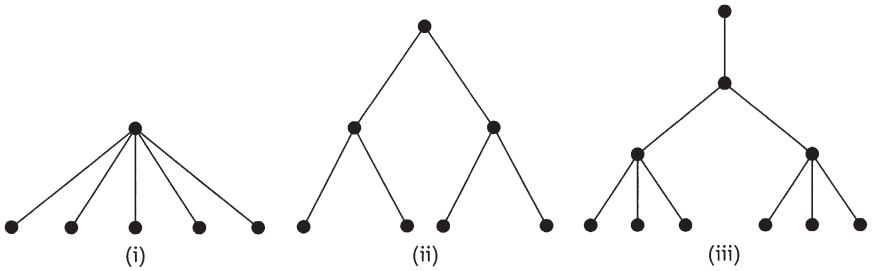


Рисунок P7.12.

(б) Полное двоичное дерево с корнем глубины 1 имеет 2 листа. Кроме того, $2^1 = 2$. Значит, утверждение верно при $n = 1$.

Предположим, что полное двоичное дерево с корнем глубины k имеет 2^k листа для некоторого $k \geqslant 1$. Значит, в полном двоичном дереве с корнем глубины $k+1$ будет 2^k внутренних вершин глубины k. Так как каждая из этих вершин имеет двух сыновей, в таком графе получится $2 \cdot 2^k = 2^{k+1}$ вершин глубины $k+1$. Осталось заметить, что любая вершина такой глубины в дереве глубины $k+1$ является листом. Используя принцип математической индукции, можно утверждать, что любое

полное двоичное дерево с корнем глубины n имеет 2^n листьев.

Набор упражнений 8

8.1. Смотри рис. Р8.1.

(а) Кратчайшим путем от вершины 1 до вершины 2 является путь 1 4 6 2.

(б) Есть два кратчайших пути от вершины 3 до вершины 6: 3 5 6 и 3 2 6.

(в) Один из контуров длины 5 — это контур 1 4 6 3 2 1.

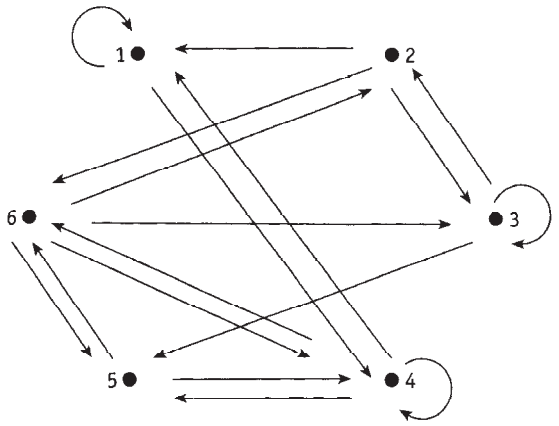


Рисунок Р8.1.

8.2. Каждая дуга uv вносит вклад, равный 1, в полустепень исхода вершины u и такой же вклад в полустепень захода вершины v. Следовательно, сумма полустепеней исхода всех вершин совпадает с числом дуг графа. Аналогичное утверждение можно сделать и относительно суммы полустепеней захода.

Рассмотрим строку матрицы смежности, соответствующую вершине v орграфа. Буква «И» в этой строке появляется в столбце, соответствующем вершине u в том и только том случае, когда в графе есть дуга, ведущая от вершины v к вершине u. Таким образом, число букв «И» в строке матрицы, соответствующей вершине v, совпадает с полустепенью исхода $\delta^+(v)$. Аналогично, число букв «И», стоящих в столбце

матрицы смежности, равно полустепени захода соответствующей вершины $\delta^-(u)$.

8.3. (а) Орграф, изображенный на рисунке в условии задачи слева, не является сильно связным, поскольку полустепень захода вершины d равна 0 (в вершину d не ведет никакая дуга). Значит, мы не сможем найти пути из любой наперед выбранной вершины этого орграфа в вершину d.

Орграф, расположенный в середине рисунка, тоже не является сильно связным, только по другой причине: из вершины d не выходят дуги. Иными словами, ее полустепень исхода равна 0. Как следствие, мы не сможем найти пути, ведущего из вершины d куда бы то ни было.

Последний орграф — сильно связный. В нем присутствует контур $a\,b\,c\,d\,e\,a$, проходящий через все вершины. Поэтому для любой пары вершин орграфа найдется путь, ведущий от первой ко второй. Например $b\,c\,d\,e\,a$ — путь из вершины b к a.

(б) Пусть C — гамильтонов цикл в гамильтоновом графе. Он содержит все вершины графа. Обозначим вершины графа таким образом, чтобы цикл C был равен

$$v_0\,v_1\,v_2\,\ldots\,v_k\,v_0.$$

После этого ориентируем ребра графа так, чтобы ребро $v_{i-1}v_i$ превратилось в дугу с началом v_{i-1} и концом v_i. Кроме того, потребуем, чтобы вершина v_k была началом дуги v_kv_0. Оставшиеся ребра графа можно ориентировать произвольным образом. Получившийся в результате орграф будет сильно связным, поскольку мы сможем найти путь от произвольной вершины к любой другой, воспользовавшись образовавшимся замкнутым путем $v_0\,v_1\,v_2\,\ldots\,v_k\,v_0$, проходящим через все вершины.

(в) Если бы соответствующий орграф не был сильно связным, то могло оказаться, что из одного района города нельзя было бы попасть в другой.

8.4. Выпишем исходные множества антецедентов:

$$A(a)=\{b\},\ A(b)=\{c\},\ A(c)=\varnothing,\ A(d)=\{a,\,e\},$$
$$A(e)=\{c\},\ A(f)=\{b,\,d,\,e\}.$$

Присвоив номер 1 вершине c, удалим ее вместе с инцидентными ей дугами. Множества антецедентов изменятся следу-

ющим образом:

$$A(a) = \{b\},\ A(b) = \varnothing,\ A(d) = \{a,\ e\},$$
$$A(e) = \varnothing,\ A(f) = \{b,\ d,\ e\}.$$

Теперь у нас возникло две вершины, которые можно отбросить. Присвоим номер 2 вершине b и удалим ее вместе с соответствующими дугами. Получим новые множества антецедентов:

$$A(a) = \varnothing,\ A(d) = \{a,\ e\},\ A(e) = \varnothing,\ A(f) = \{d,\ e\}.$$

Опять есть свобода в выборе вершины для удаления. Присвоим номер 3 вершине a и удалим ее вместе с соответствующими дугами. Подправим множества антецедентов:

$$A(d) = \{e\},\ A(e) = \varnothing,\ A(f) = \{d,\ e\}.$$

Удалим вершину e вместе с инцидентными ей дугами, присвоив ей номер 4. Тогда

$$A\{d\} = \varnothing,\ A(f) = \{d\}.$$

Присвоим номер 5 вершине d и удалим ее, выбросив и подходящие к ней дуги. Теперь осталась одна вершина с пустым множеством антецедентов:

$$A(f) = \varnothing.$$

Присвоим вершине f номер 6.

В результате мы получим следующую последовательность согласованных меток: $c(1)$, $b(2)$, $a(3)$, $e(4)$, $d(5)$ и $f(6)$. Новая матрица смежности выглядит следующим образом:

$$\begin{array}{c} \\ c \\ b \\ a \\ e \\ d \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} c & b & a & e & d & f \end{array} \\ \left[\begin{array}{cccccc} \text{Л} & \text{И} & \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{И} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{Л} & \text{Л} \end{array}\right] \end{array}.$$

Поскольку столбцы и строки новой матрицы смежности записаны в порядке согласованной последовательности меток,

то ее верхнетреугольная часть дает полную информацию об орграфе.

Все множества антецедентов орграфа из упр. 8.1 не пусты. Следовательно, в этом орграфе присутствуют контуры и последовательность согласованных меток создать невозможно. Алгоритм топологической сортировки не сможет присвоить первую метку ни одной из вершин.

8.5. Соответствующая система ПЕРТ приведена на рис. Р8.2.

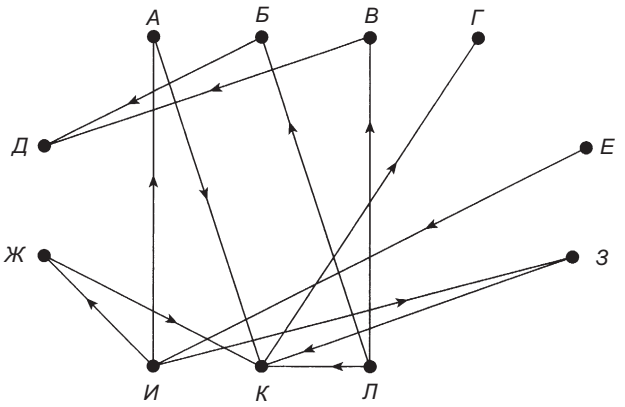


Рисунок Р8.2. Система ПЕРТ для приготовления цыпленка

Исходные множества антецедентов:

$$A(\text{А}) = \{\text{И}\},\ A(\text{Б}) = \{\text{Л}\},\ A(\text{В}) = \{\text{Л}\},\ A(\text{Г}) = \{\text{К}\},$$
$$A(\text{Д}) = \{\text{Б, В}\},\ A(\text{Е}) = \varnothing,\ A(\text{Ж}) = \{\text{И}\},\ A(\text{З}) = \{\text{И}\},$$
$$A(\text{И}) = \{\text{Е}\},\ A(\text{К}) = \{\text{А, Ж, З, Л}\},\ A(\text{Л}) = \varnothing.$$

Итак, присвоив метки 1 и 2 вершинам Е и Л соответственно, удалим их из орграфа вместе с инцидентными им дугами. Тогда

$$A(\text{А}) = \{\text{И}\},\ A(\text{Б}) = \varnothing,\ A(\text{В}) = \varnothing,\ A(\text{Г}) = \{\text{К}\},$$
$$A(\text{Д}) = \{\text{Б, В}\}, A(\text{Ж}) = \{\text{И}\},\ A(\text{З}) = \{\text{И}\},$$
$$A(\text{И}) = \varnothing,\ A(\text{К}) = \{\text{А, Ж, З}\}.$$

Обозначив вершину Б меткой 3, В — меткой 4 и И — меткой 5, удалим Б, В и И вместе с соответствующими дугами.

Теперь

$$A(\text{А}) = \varnothing,\ A(\text{Г}) = \{\text{К}\},\ A(\text{Д}) = \varnothing,\ A(\text{Ж}) = \varnothing,$$
$$A(\text{З}) = \varnothing,\ A(\text{К}) = \{\text{А, Ж, З}\}.$$

Обозначим А как 6, Д как 7, Ж как 8 и З как 9 и удалим эти вершины вместе с соответствующими дугами. После этого

$$A(\text{Г}) = \{\text{К}\},\ A(\text{К}) = \varnothing.$$

Наконец, удалим вершину К, присвоив ей метку 10, а затем присвоим метку 11 вершине Г. Это дает нам последовательность согласованных меток Е Л Б В И А Д Ж З К Г.

8.6.

```
begin
    for j := 1 to n do
        A(j) := ∅;
        k := 1;
        begin
            while k ⩽ n do
                if M(k, j) = И then
                A(j) := A(j) ∪ {k};
                k := k + 1;
        end
end
```

8.7.

$$M^2 = \begin{bmatrix} \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix} \begin{bmatrix} \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix} = \begin{bmatrix} \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} \end{bmatrix};$$

$$M^3 = \begin{bmatrix} \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix} \begin{bmatrix} \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} \end{bmatrix} = \begin{bmatrix} \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \end{bmatrix};$$

$$M^4 = \begin{bmatrix} \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \end{bmatrix} \begin{bmatrix} \text{Л} & \text{Л} & \text{Л} & \text{И} \\ \text{И} & \text{Л} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \end{bmatrix} = \begin{bmatrix} \text{И} & \text{Л} & \text{Л} & \text{Л} \\ \text{Л} & \text{И} & \text{Л} & \text{Л} \\ \text{Л} & \text{Л} & \text{И} & \text{Л} \\ \text{Л} & \text{Л} & \text{Л} & \text{И} \end{bmatrix}.$$

$$M^* = M \textbf{ или } M^2 \textbf{ или } M^3 \textbf{ или } M^4 = \begin{bmatrix} И & И & И & И \\ И & И & И & И \\ И & И & И & И \\ И & И & И & И \end{bmatrix}.$$

8.8.

$$M = W_0 = \begin{bmatrix} Л & И & Л & Л \\ Л & Л & И & Л \\ Л & Л & Л & И \\ И & Л & Л & Л \end{bmatrix}; \qquad W_1 = \begin{bmatrix} Л & И & Л & Л \\ Л & Л & И & Л \\ Л & Л & Л & И \\ И & И & Л & Л \end{bmatrix};$$

$$W_2 = \begin{bmatrix} Л & И & И & Л \\ Л & Л & И & Л \\ Л & Л & Л & И \\ И & И & И & Л \end{bmatrix}; \qquad W_3 = \begin{bmatrix} Л & И & И & И \\ Л & Л & И & И \\ Л & Л & Л & И \\ И & И & И & И \end{bmatrix}.$$

Наконец,

$$M^* = W_4 = \begin{bmatrix} И & И & И & И \\ И & И & И & И \\ И & И & И & И \\ И & И & И & И \end{bmatrix}.$$

8.9. (а) Смотри табл. Р8.1.

Таблица Р8.1

Шаг	Отмеченные вершины	Расстояние до вершины						Неотмеченные вершины
		A	B	C	D	E	F	
0	A	0	**3**	10	∞	∞	∞	B, C, D, E, F
1	B	0	3	**10**	15	∞	∞	C, D, E, F
2	C	0	3	10	**15**	∞	∞	D, E, F
3	D	0	3	10	15	**17**	23	E, F
4	E	0	3	10	15	17	**23**	F
5	F	0	3	10	15	17	23	

Кратчайшие пути от вершины A:

$A\,B$ длины 3,
$A\,C$ длины 10,
$A\,C\,D$ длины 15.
$A\,C\,D\,E$ длины 17,
$A\,C\,D\,F$ длины 23.

(б) Смотри табл. Р8.2.

Таблица Р8.2

Шаг	Отмеченные вершины	Расстояние до вершины						Неотмеченные вершины
		A	B	C	D	E	F	
0	C	∞	∞	0	**5**	∞	∞	A, B, D, E, F
1	D	∞	∞	0	5	**7**	13	A, B, E, F
2	E	∞	**11**	0	5	7	13	A, B, F
3	B	∞	11	0	5	7	**13**	A, F
4	F	∞	11	0	5	7	13	A

Кратчайшие пути от вершины C:

$C\,D$ длины 5,
$C\,D\,E$ длины 7,
$C\,D\,E\,B$ длины 11,
$C\,D\,F$ длины 13.

8.10. Смотри табл. Р8.3.

Таблица Р8.3

Шаг	Отмеченные вершины	Расстояние до вершины								Неотмеченные вершины
		S	A	B	C	D	E	F	T	
0	S	0	5	10	**4**	∞	∞	∞	∞	A, B, C, D, E, F, T
1	C	0	**5**	10	4	14	22	18	∞	A, B, D, E, F, T
2	A	0	5	**9**	4	14	22	18	∞	B, D, E, F, T
3	B	0	5	9	4	**14**	22	17	∞	D, E, F, T
4	D	0	5	9	4	14	20	**17**	26	E, F, T
5	F	0	5	9	4	14	**20**	17	26	E, T
6	E	0	5	9	4	14	20	17	**25**	T
7	T	0	5	9	4	14	20	17	25	

Кратчайшие пути от вершины S:

$S\,C$ длины 4,
$S\,A$ длины 5,
$S\,A$, B длины 9,
$S\,C\,D$ длины 14,
$S\,A\,B\,F$ длины 17,
$S\,A\,B\,F\,E$ длины 20,
$S\,A\,B\,F\,E\,T$ длины 25,
$S\,C\,D\,E\,T$ длины 25.

Набор упражнений 9

9.1. Таб. Р9.1 и табл. Р9.2 — это таблицы истинности булевых выражений, участвующих в законах де Моргана. Последние два столбца в каждой из таблиц одинаковы. Значит, законы де Моргана справедливы.

Таблица Р9.1

p	q	$\bar{p}$	$\bar{q}$	$p \vee q$	$\bar{p} \wedge \bar{q}$	$\overline{(p \vee q)}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Таблица Р9.2

p	q	$\bar{p}$	$\bar{q}$	$p \wedge q$	$\bar{p} \vee \bar{q}$	$\overline{(p \wedge q)}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

9.2. (а) Опираясь на законы де Моргана, законы ассоциативности и тот факт, что $\overline{(\bar{q})} = q$, получаем

$$\overline{(p \wedge \bar{q})} \vee \bar{r} = (\bar{p} \vee q) \vee \bar{r} = \bar{p} \vee q \vee \bar{r}.$$

(б) Используя законы де Моргана, дистрибутивности, идемпотентности и поглощения, имеем

$$\begin{aligned} &\overline{((p \wedge \bar{q}) \wedge (r \vee (p \wedge \bar{q})))} = \\ &= \overline{(p \wedge \bar{q})} \vee \overline{(r \vee (p \wedge \bar{q}))} = \\ &= (\bar{p} \vee q) \vee (\bar{r} \wedge \overline{(p \wedge \bar{q})}) = \\ &= (\bar{p} \vee q) \vee (\bar{r} \wedge (\bar{p} \vee q)) = \\ &= ((\bar{p} \vee q) \vee \bar{r}) \wedge (\bar{p} \vee q) = \\ &= \bar{p} \vee q. \end{aligned}$$

9.3. $\bar{p}\bar{q}\bar{r}s \vee \bar{p}\bar{q}r\bar{s} \vee p\bar{q}rs \vee pq\bar{r}s$.

9.4. Пусть

$$f = \big(p \wedge (\bar{q} \vee r)\big) \vee \big(\bar{p} \wedge (q \vee \bar{r})\big).$$

Таблица истинности функции f представлена в табл. Р9.3.

По таблице можно написать дизъюнктивную нормальную форму:

$$f = \bar{p}\bar{q}\bar{r} \vee \bar{p}q\bar{r} \vee \bar{p}qr \vee p\bar{q}\bar{r} \vee p\bar{q}r \vee pqr.$$

9.5. (а) $(p \wedge \bar{q}) \wedge r = \overline{\Big(\overline{((p \wedge \bar{q}) \wedge r)}\Big)} = \overline{\Big(\overline{(p \wedge \bar{q})} \vee \bar{r}\Big)} =$
$= \overline{((\bar{p} \vee q) \vee \bar{r})} = \overline{(\bar{p} \vee q \vee \bar{r})}.$

(б) $(p \wedge \bar{q}) \wedge r =$

$$
\begin{aligned}
&= \big(p \wedge (q \textbf{ НЕ–И } q)\big) \wedge r = \\
&= \Big(\big(p \wedge (q \textbf{ НЕ–И } q)\big) \textbf{ НЕ–И } r\Big) \textbf{ НЕ–И} \\
&\quad \textbf{НЕ–И } \Big(\big(p \wedge (q \textbf{ НЕ–И } q)\big) \textbf{ НЕ–И } r\Big) = \\
&= \Big[\Big(\big(p \textbf{ НЕ–И } (q \textbf{ НЕ–И } q)\big) \textbf{ НЕ–И} \\
&\quad \textbf{НЕ–И } \big(p \textbf{ НЕ–И } (q \textbf{ НЕ–И } q)\big)\Big) \textbf{ НЕ–И } r\Big] \textbf{ НЕ–И} \\
&\quad \textbf{НЕ–И } \Big[\Big(\big(p \textbf{ НЕ–И } (q \textbf{ НЕ–И } q)\big) \textbf{ НЕ–И} \\
&\quad \textbf{НЕ–И } \big(p \textbf{ НЕ–И } (q \textbf{ НЕ–И } q)\big)\Big) \textbf{ НЕ–И } r\Big].
\end{aligned}
$$

Таблица Р9.3

p	q	r	$\bar{q} \vee r$	$q \vee \bar{r}$	$p \wedge (\bar{q} \vee r)$	$\bar{p} \wedge (q \vee \bar{r})$	f
0	0	0	1	1	0	1	1
0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	1
0	1	1	1	1	0	1	1
1	0	0	1	1	1	0	1
1	0	1	1	0	1	0	1
1	1	0	0	1	0	0	0
1	1	1	1	1	1	0	1

9.6. Для решения задачи нам достаточно выразить функции $\bar{p}$, $p \vee q$ и $p \wedge q$ через p **НЕ–ИЛИ** q. Сделаем это.

$$
\begin{aligned}
\bar{p} &= \overline{(p \vee p)} = p \textbf{ НЕ–ИЛИ } p; \\
p \vee q &= \overline{\big(\overline{(p \vee q)}\big)} = \overline{(p \textbf{ НЕ–ИЛИ } q)} = \\
&= (p \textbf{ НЕ–ИЛИ } q) \textbf{ НЕ–ИЛИ } (p \textbf{ НЕ–ИЛИ } q); \\
p \wedge q &= \overline{(\bar{p} \vee \bar{q})} = \bar{p} \textbf{ НЕ–ИЛИ } \bar{q} = \\
&= (p \textbf{ НЕ–ИЛИ } p) \textbf{ НЕ–ИЛИ } (q \textbf{ НЕ–ИЛИ } q).
\end{aligned}
$$

Полнота системы функций { **НЕ–ИЛИ** } доказана.

В качестве альтернативного доказательства можно заметить, что полнота системы функций { **НЕ–И** } была нами доказана ранее. Поэтому для решения задачи достаточно выразить функцию p **НЕ–И** q через p **НЕ–ИЛИ** q.

Заметим, что

$$\begin{aligned} p \textbf{ НЕ–И } q &= \overline{(p \wedge q)} = \\ &= \overline{\Big(\overline{(\bar{p} \vee \bar{q})}\Big)} = \\ &= \overline{(\bar{p} \textbf{ НЕ–ИЛИ } \bar{q})} = \\ &= \overline{\big((p \textbf{ НЕ–ИЛИ } p) \textbf{ НЕ–ИЛИ } (q \textbf{ НЕ–ИЛИ } q)\big)}. \end{aligned}$$

Следовательно,

$$\begin{aligned} p \textbf{ НЕ–И } q =& \big((p \textbf{ НЕ–ИЛИ } p) \textbf{ НЕ–ИЛИ } (q \textbf{ НЕ–ИЛИ } q)\big) \\ & \textbf{НЕ–ИЛИ } \big((p \textbf{ НЕ–ИЛИ } p) \textbf{ НЕ–ИЛИ } (q \textbf{ НЕ–ИЛИ } q)\big). \end{aligned}$$

9.7. Карта Карно показана на рис. Р9.1.

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r	1	1	1	
$\bar{r}$	1			

Рисунок Р9.1. Карта Карно выражения $\bar{p}\bar{q}r \vee \bar{p}qr \vee pq\bar{r} \vee pqr$

Она имеет две пары соседних единиц. Поэтому

$$\begin{aligned} \bar{p}\bar{q}r \vee \bar{p}qr \vee pq\bar{r} \vee pqr =\ & (\bar{p}\bar{q}r \vee \bar{p}qr) \vee (pq\bar{r} \vee pqr) = \\ & = \bar{p}r(\bar{q} \vee q) \vee pq(\bar{r} \vee r) = \\ & = \bar{p}r \vee pq. \end{aligned}$$

9.8. $f = \bar{p}\bar{q}\bar{r} \vee \bar{p}q\bar{r} \vee p\bar{q}r \vee pqr$. Карта Карно функции f представлена на рис. Р9.2.

	pq	$\bar{p}q$	$\bar{p}\bar{q}$	$p\bar{q}$
r	1			1
$\bar{r}$		1	1	

Рисунок Р9.2. Карта Карно функции f

На рисунке легко заметить только одну пару соседних единиц. Однако их две (одна не видна при таком обозначении столбцов). Упрощение соответствующих пар минтермов дает:

$$\bar{p}\bar{q}\bar{r} \vee \bar{p}q\bar{r} = \bar{p}\bar{r} \qquad \text{и} \qquad p\bar{q}r \vee pqr = pr.$$

Окончательно имеем: $f = \bar{p}\bar{r} \vee pr$.

9.9. Функциональная схема генерирует функцию $\bar{p}qr \vee \bar{p}\bar{q}r$. Ее карта Карно изображена на рис. Р9.3.

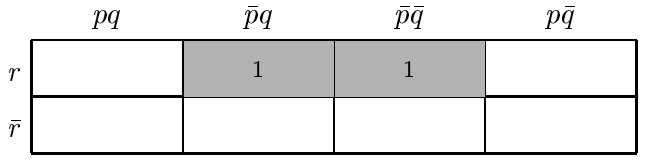


Рисунок Р9.3. Карта Карно функции $\bar{p}qr \vee \bar{p}\bar{q}r$

Упрощая это выражение, получаем $\bar{p}r$. Новая функциональная схема представлена на рис. Р9.4.

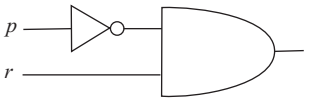


Рисунок Р9.4. Функциональная схема функции $\bar{p}r$

9.10. Последовательность преобразований доказывает требуемую эквивалентность:

$$\begin{aligned}
\bar{p}\ \textbf{НЕ–И}\ (\bar{q}\ \textbf{НЕ–И}\ r) &= \bar{p}\ \textbf{НЕ–И}\ \overline{(\bar{q} \wedge r)} = \\
&= \bar{p}\ \textbf{НЕ–И}\ (q \vee \bar{r}) = \\
&= \overline{\big(\bar{p} \wedge (q \vee \bar{r})\big)} = \\
&= p \vee \overline{(q \vee \bar{r})} = \\
&= p \vee (\bar{q} \wedge r).
\end{aligned}$$

Искомая функциональная схема изображена на рис. Р9.5.

9.11. Первая из функциональных схем генерирует функцию $\bar{p}\bar{q}r \vee pq\bar{r} \vee qr$, которая равна функции $\bar{p}\bar{q}r \vee pq\bar{r} \vee pqr \vee \bar{p}qr$, поскольку $qr = (p \vee \bar{p})qr$.

Вторая схема генерирует функцию $\bar{p}r \vee pq$. Из решения задачи 9.7, в частности, следует, что

$$\bar{p}r \vee pq = \bar{p}\bar{q}r \vee pq\bar{r} \vee pqr \vee \bar{p}qr.$$

Таким образом, эти схемы генерируют равные функции, т. е. они эквивалентны.

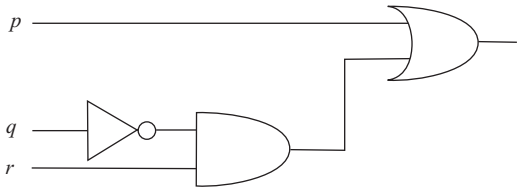


Рисунок Р9.5. Функциональная схема функции $p \vee (\bar{q} \wedge r)$

9.12. Следуя указанию к задаче, получаем

$$p \textbf{ НЕ–ИЛИ } q = \overline{(p \vee q)} = \bar{p} \wedge \bar{q} = \overline{\overline{(\bar{p} \wedge \bar{q})}} = \overline{(\bar{p} \textbf{ НЕ–И } \bar{q})} =$$
$$= \big((p \textbf{ НЕ–И } p) \textbf{ НЕ–И } (q \textbf{ НЕ–И } q)\big) \textbf{ НЕ–И}$$
$$\textbf{НЕ–И } \big((p \textbf{ НЕ–И } p) \textbf{ НЕ–И } (q \textbf{ НЕ–И } q)\big).$$

Требуемая схема показана на рис. Р9.6.

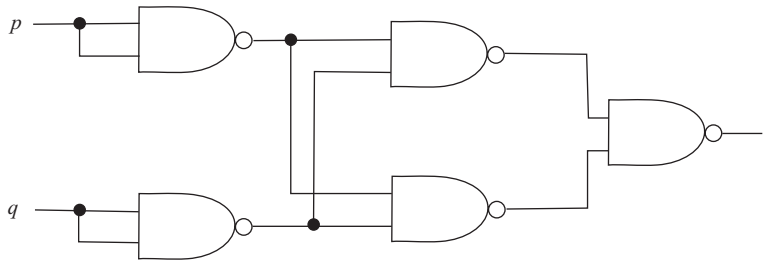


Рисунок Р9.6. Функциональная схема функции p **НЕ–ИЛИ** q

Дополнение к первому изданию

А. А. Ковалев

Д.1. Генератор случайных графов

Математике всегда было присуще стремление разрабатывать эффективные методы решения как можно более широких классов задач. Многолетний опыт развития теории дискретных и комбинаторных проблем и практика их решения показали, что эти две стороны — общность метода и его эффективность — находятся в известном антагонизме. Вместе с тем, очень важно знать, и в особенности это касается задач дискретной математики, можно ли в принципе надеяться на создание достаточно общих и эффективных методов или надо сознательно идти по пути разбиения задач на все более узкие классы и, пользуясь их спецификой, разрабатывать для них эффективные алгоритмы.

Большинство дискретных и комбинаторных проблем, вообще говоря, допускает решение с помощью некоторого процесса перебора. Однако число шагов переборного метода растет экспоненциально в зависимости от размерности задачи. Для некоторых проблем этого типа удается построить эффективные (существенно менее трудоемкие, чем полный перебор вариантов) методы решения. К сожалению, число таких задач невелико. Для отличия «удобных» и «неудобных» задач на терминологическом уровне вводятся специальные понятия. Так задачу, для которой существует алгоритм решения существенно более экономичный, чем перебор экспоненциального числа вариантов, называют *алгоритмически разрешимой*. Если же такого алгоритма найти невозможно, то задачу называют *алгоритмически неразрешимой*. Стало общепринятым считать переборную задачу решаемой эффективно, если имеется алгоритм, решающий ее за время, ограниченное полиномом от размерности задачи.

Основные подходы, применяемые для решения алгоритмически неразрешимых задач, можно разбить на две категории. К первой категории относятся подходы, в которых делается попытка максимального сокращения объема перебора, хотя при этом и признается неизбежность экспоненциального времени работы. Для сокращения перебора наиболее широко используются приемы, основанные на методе «ветвей и границ». Они заключаются в построении «частичных решений», представленных в виде дерева поиска, и применении мощных методов построения оценок, позволяющих распознать бесперспективные частичные решения, в результате чего от дерева поиска на одном шаге отсекается целая ветвь.

Подходы, относящиеся ко второй категории, применимы исключительно к оптимизационным задачам. Они основаны на приеме, который можно назвать «снижение требований». Метод заключается в отказе от поиска оптимального решения и в нахождении вместо этого «хорошего» решения за приемлемое время. Алгоритмы, основанные на этом приеме, обычно называются «эвристически-

ми», поскольку они используют различные разумные соображения без строгих обоснований.

Для алгоритмов первой и второй категорий важным является исследование их поведения или эффективности «на практике» или «в среднем». Часто изучение поведения алгоритмов «в среднем» сводится к формированию множества предположительно типичных задач, прогонке соперничающих алгоритмов на этом множестве и сравнению полученных результатов.

Если можно считать, что задачи, предположительно встречающиеся на практике, подчиняются некоторому конкретному распределению вероятностей, то используют метод построения «случайных» задач, подчиняющихся заданному распределению вероятностей, и на построенной таким образом выборке исследуют поведение алгоритмов.

Для задач, формулируемых в терминах графов, такой подход приводит к необходимости создания генераторов случайных графов с теми или иными свойствами. При этом на начальном этапе, при общей формулировке переборных задач и разработке универсальных методов их решения, оказывается достаточным формирование выборки случайных графов с теми или иными свойствами, подчиняющейся равномерному закону распределения вероятностей.

Метод генерирования случайного графа состоит в построении его матрицы смежности, в ячейках которой истинностные значения *И* размещены случайным образом. Для построения такой матрицы можно использовать датчик случайных чисел, равномерно распределенных на отрезке $[0, 1]$. Для читателей, знакомых с теорией вероятности только понаслышке, можно сказать, что главное отличие равномерно распределенной случайной величины от других состоит в том, что все случайные значения, которые они принимают — равновероятны. Алгоритмы построения последовательностей случайных чисел на отрезке $[0, 1]$ и их таблицу можно найти в книге [21]. Кроме того, любой язык программирования высокого уровня содержит в своих библиотеках датчики случайных чисел.

Пусть генерируемый граф имеет n вершин и m ребер или дуг. Тогда его матрица смежности будет иметь размер $n \times n$, а в ее ячейках должно быть размещено $2m$ букв «*И*» для неориентированного графа (так как каждое ребро неориентированного графа порождает две буквы «*И*», расположенные симметрично относительно главной диагонали матрицы) и m букв «*И*» — для ориентированного.

Наша задача — расположить требуемое количество букв «*И*» в ячейках матрицы смежности случайным образом, следя за тем, чтобы распределение вероятностей было равномерным. Обсудим, как это можно сделать.

Напомним, что мы можем пользоваться датчиком случайных чисел R, равномерно распределенных на отрезке $[0, 1]$. Нам же нужно выбрать ячейку матрицы, т. е. как столбец, так и строку, в которых она расположена. Прежде всего необходимо масштабировать случайную величину R, а именно, умножить ее на подходящий коэффициент k. Так, например, случайная величина $12R$ будет равномерно распределена на отрезке $[0, 12]$, а nR — на отрезке $[0, n]$. Но число nR как правило не будет целым. Поэтому необходимо перейти к целой части $\lfloor nR \rfloor$ (напомним, что это наименьшее целое число, не превосходящее nR). Од-

нако такое преобразование нас тоже не может устроить. Дело в том, что если случайная величина R окажется меньше n, то $\lfloor nR \rfloor$ будет равна 0, а нумерация строк и столбцов нашей матрицы начинается с 1. В итоге мы приходим к новой случайной величине $\lfloor nR \rfloor + 1$, значения которой — натуральные числа от 1 до[1] n.

Итак, мы имеем возможность случайным образом выбирать как номер столбца, так и номер строки матрицы, используя два случайных значения величины $\lfloor nR \rfloor + 1$. К сожалению, при таком выборе мы не получим равномерного распределения вероятностей, поскольку будут задействованы не одна, а две случайных величины. Из создавшейся ситуации есть красивый выход. Расположим все ячейки матрицы в одну строку: сначала будут идти ячейки первой строки матрицы, за ними — ячейки второй строки и т. д. В результате получим строчку из n^2 ячеек, причем ячейка матрицы, стоящая в i-ой строке и j-ом столбце будет иметь номер $N = (n-1)i + j$. Теперь величина $N = \lfloor n^2 R \rfloor + 1$ будет давать нам ячейки матрицы случайным образом, а закон распределения окажется требуемым, т. е. равномерным. Осталось восстановить строку i и столбец j матрицы, содержащих найденную ячейку. Это делается с помощью следующих, довольно понятных, формул:

$$i = \left\lfloor \frac{N}{n} \right\rfloor + 1, \quad j = N - (i-1)n.$$

Можно предложить, в качестве примера, несколько алгоритмов построения выборок случайных графов с заданными свойствами.

Д.1.1. Алгоритм построения случайного неориентированного графа

Матрица смежности M неориентированного графа, состоящего из n вершин и m ребер, будет иметь размер $n \times n$ и содержать $2m$ истинностных значений *И*, расположенных симметрично относительно главной диагонали. Как обычно в этой книге считаем, что граф не содержит петель, т. е. для любого $i = 1, 2, \ldots, n$, $M(i, i) = Л$. На каждом k-ом шаге алгоритма ($k = 1, 2, \ldots, m$) будем получать случайное число с помощью датчика случайных чисел, вычислять адрес ячейки (i, j) матрицы и записывать в нее букву «*И*». Если ячейка с вычисленным адресом (i, j) уже содержит такую букву или $i = j$, будем считать данный шаг алгоритма неудачным и увеличим m на единицу.

```
Input
    n — количество вершин графа;
    m — количество ребер графа;
    M — матрица размера n × n,
          содержащая во всех ячейках Л;
      begin
          for k = 1 to m do
              begin
                  получить с помощью датчика
```

[1] Может так случиться, что величина R примет значение 1. Тогда, очевидно, $\lfloor nR \rfloor + 1 = n + 1$. Однако вероятность этого события настолько мала, что им можно пренебречь.

```
                случайное число R;
                N := ⌊n²R⌋ + 1;
                i := ⌊N : n⌋ + 1;
                j := N − (i − 1) · n;
                if M(i, j) ≠ И and i ≠ j then
                    begin
                     M(i, j) := И;
                     M(j, i) := И;
                    end
                else
                     m := m + 1
            end
        end
```

Output M — *матрица смежности неориентированного графа.*

Д.1.2. Алгоритм построения случайного ориентированного графа

Матрица смежности M орграфа, состоящего из n вершин и m дуг, будет иметь размер $n \times n$ и содержать m истинностных значений *И*. Считаем, что орграф не содержит петель. На каждом k-ом шаге алгоритма ($k = 1, 2, \ldots, m$) будем получать случайное число с помощью датчика случайных чисел, вычислять адрес ячейки (i, j) матрицы и записывать в нее букву «*И*». Если ячейка с вычисленным адресом (i, j) уже содержит *И* или $i = j$, будем считать данный шаг алгоритма неудачным, и увеличим m на единицу.

```
Input
    n — количество вершин орграфа;
    m — количество дуг орграфа;
    M — матрица размера n × n,
          содержащая во всех ячейках Л;
    begin
        for k = 1 to m do
            begin
                получить с помощью датчика
                случайное число R;
                N := ⌊n²R⌋ + 1;
                i := ⌊N : n⌋ + 1;
                j := N − (i − 1) · n;
                if M(i, j) ≠ И and i ≠ j then
                    begin
                     M(i, j) := И;
                    end
                else
```

$m := m + 1$

end

end

Output M — *матрица смежности орграфа.*

В целом ряде случаев необходимо, чтобы сгенерированный орграф обладал какими-то дополнительными свойствами. Одно из таких свойств — отсутствие контуров. Приведем алгоритм генерирования бесконтурного орграфа.

Д.1.3. Алгоритм построения случайного ориентированного бесконтурного графа

Матрица смежности орграфа, состоящего из n вершин и m дуг, имеет размер $n \times n$ и содержит m букв «*И*». Будем размещать истинностные значения *И* выше главной диагонали матрицы смежности, задавая дуги орграфа, идущие от вершины с меньшим номером к вершине с большим номером. В этом случае, очевидно, построенный орграф не сможет содержать контуров. Как и раньше, на k-ом шаге алгоритма ($k = 1, 2, \ldots, m$) получаем случайное число с помощью датчика случайных чисел, вычисляем адрес ячейки (i, j) матрицы и в случае $i < j$, в ячейку (i, j), а в случае $i > j$, в ячейку (j, i) записываем букву «*И*». Если ячейка с вычисленным адресом (i, j) (в случае $i < j$) или (j, i) (в случае $i > j$) уже содержит «*И*» или $i = j$, будем считать данный шаг алгоритма неудачным и увеличим m на единицу.

Input

n — *количество вершин орграфа;*
m — *количество дуг орграфа;*
M — *матрица размера* $n \times n$,
содержащая во всех ячейках *Л*;

begin

for $k = 1$ **to** m **do**

begin

получить с помощью датчика
случайное число R;
$N := \lfloor n^2 R \rfloor + 1$;
$i := \lfloor N : n \rfloor + 1$;
$j := N - (i - 1) \cdot n$;
if $i \neq j$ **and** $i < j$ **and** $M(i, j) \neq И$ **then**
$M(i, j) := И$;
if $i \neq j$ **and** $i > j$ **and** $M(i, j) \neq И$ **then**
$M(j, i) := И$;
else
$m := m + 1$

end

end

Output M — *матрица смежности бесконтурного орграфа.*

Принцип работы представленных алгоритмов одинаков, поэтому рассмотрим на примере работу только одного из них, а именно, построим ориентированный

граф. Воспользуемся таблицей случайных чисел из книги Дж. Макконнелл, уже упоминавшейся здесь.

Пример Д.1.1. Сгенерировать орграф, содержащий 5 вершин и 8 дуг, и изобразить его графически.

Решение. В наших обозначениях $n = 5$ и $m = 8$. Процесс работы алгоритма представлен в следующей таблице:

Таблица Д.1.1

k	R	N	i	j	m	M
1	0,21132	6	2	1	8	Л Л Л Л Л И Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л
2	0,26215	7	2	2	9	Л Л Л Л Л И Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л Л
3	0,79253	20	4	5	9	Л Л Л Л Л И Л Л Л Л Л Л Л Л Л Л Л Л Л И Л Л Л Л Л
4	0,28052	8	2	3	9	Л Л Л Л Л И Л И Л Л Л Л Л Л Л Л Л Л Л И Л Л Л Л Л
5	0,93648	24	5	4	10	Л Л Л Л Л И Л И Л Л Л Л Л Л Л Л Л Л Л И Л Л Л И Л
7	0,35606	9	2	4	10	Л Л Л Л Л И Л И И Л Л Л Л Л Л Л Л Л Л И Л Л Л И Л

Продолжение табл. Д.1.1

8	0,16043	5	1	5	10	$\begin{bmatrix} Л & Л & Л & Л & И \\ И & Л & И & И & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ Л & Л & Л & И & Л \end{bmatrix}$
9	0,40480	11	3	1	10	$\begin{bmatrix} Л & Л & Л & Л & И \\ И & Л & И & И & Л \\ И & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ Л & Л & Л & И & Л \end{bmatrix}$
10	0,74225	19	4	4	11	$\begin{bmatrix} Л & Л & Л & Л & И \\ И & Л & И & И & Л \\ И & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ Л & Л & Л & И & Л \end{bmatrix}$
11	0,70183	18	4	3	11	$\begin{bmatrix} Л & Л & Л & Л & И \\ И & Л & И & И & Л \\ И & Л & Л & Л & Л \\ Л & Л & И & Л & И \\ Л & Л & Л & И & Л \end{bmatrix}$

Полученный орграф представлен на рис. Д.1.

Д.2. Связность в графах

При решении практических задач, формулируемых в терминах графов, бывает важно не только определить число компонент связности графа, но и выяснить структуру самих компонент, т. е. изобразить отдельно каждую компоненту или найти их матрицы смежности.

Например, при решении задачи планирования дорожного строительства в каком-либо регионе страны, существующую сеть дорог между населенными пунктами удобно представить в виде графа. В этом случае компонента связности графа — сеть дорог между группой населенных пунктов, обеспечивающая доступ из каждого населенного пункта группы в каждый. Если таких групп несколько, то можно планировать строительство дорог (то есть добавление ребер в граф)

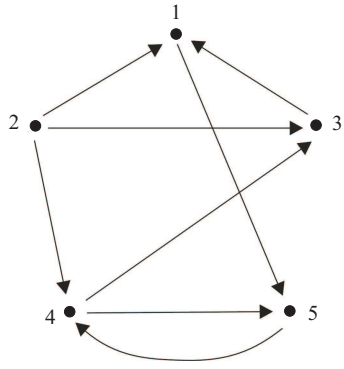


Рисунок Д.1.

между населенными пунктами различных групп с
целью расширения возможности доступа.

Алгоритмы определения самих компонент связности графа основаны на использовании матриц связности графов. В этой книге уже говорилось о матрице достижимости W орграфа (см. стр. 176), в которой хранится информация о всех путях между его вершинами. Нам потребуется аналогичное понятие для неориентированного графа. В этом случае такого сорта матрицу называют *матрицей связности*. Матрицей связности графа G с n вершинами называют квадратную матрицу S размера $n \times n$, у которой на пересечении i-ой строки и j-ого столбца стоит истинностное значение $S(i,j) = И$, если $i = j$ или существует маршрут, соединяющий i-ую вершину с j-ой. Поскольку ребра графа неориентированы, то $S(i,j) = S(j,i)$, т. е. матрица связности симметрична.

Кроме того, для ориентированного графа введем *матрицу сильной связности* (понятие сильной связности обсуждалось на стр. 185). *Матрицей сильной связности* орграфа называют квадратную матрицу C размера $n \times n$, у которой $C(i,j) = И$, если $i = j$ или вершина с номером i достижима из вершины с номером j и, одновременно, j-ая вершина достижима из i-ой. Если же это условие не выполнено, то $S(i,j) = Л$ (т. е. $S(i,j) = И$ тогда и только тогда, когда вершины с номерами i и j принадлежат одной компоненте сильной связности орграфа).

Матрица C сильной связности орграфа отличается от матрицы W достижимости только тем, что $C(i,j) = И$ тогда и только тогда, когда существуют пути как из вершины i в j, так и из j в i. Кроме того, $C(i,i) = И$ для любой вершины i. Согласно определению матрицы достижимости, при наличии пути из i-ой вершины в j-ую $W(i,j) = И$, а $W(j,i) = И$ только тогда, когда есть путь из j-ой вершины в i-ую. Таким образом, ячейки матрицы сильной связности орграфа можно заполнять по правилу:

$$C(i,j) = \begin{cases} И, & \text{если } i = j, \\ W(i,j) \text{ } \textbf{и} \text{ } W(j,i), & \text{если } i \neq j. \end{cases}$$

Существует достаточно большое количество методов вычисления матриц связности и достижимости. В этой книге уже был описан алгоритм Уоршелла, предназначенный для вычисления матрицы достижимости ориентированного графа (см. стр. 177). Оказывается, с его помощью можно найти и матрицу связности неориентированного графа. Известная уже схема переносится на этот случай почти без изменений. Только в самое ее начало добавляется один шаг, обеспечивающий значения «И» на главной диагонали матрицы связности S. Опишем алгоритм и разберем его действие на конкретном примере.

Д.2.1. Алгоритм Уоршелла, вычисляющий матрицу связности

Пусть дан граф $G(V,E)$ с n вершинами и матрицей смежности M. Алгоритм строит последовательность булевых матриц $W_0, W_1, \ldots, W_n$, первая из которых равна

$$W_0 = M \text{ или } E,$$

где E — единичная матрица вида

$$\begin{bmatrix} И & Л & ... & Л \\ Л & И & ... & Л \\ ... & ... & ... & ... \\ Л & Л & ... & И \end{bmatrix},$$

последняя W_n — искомая матрица связности S, а ячейки промежуточных матриц определяются по формуле:

$$W_k(i,j) = W_{k-1}(i,j) \textbf{ или } \Big(W_{k-1}(i,k) \textbf{ и } W_{k-1}(k,j)\Big),$$

$$k = 1, ..., n; \quad i = 1, ..., n; \quad j = 1, ..., n.$$

Подробное объяснение этой формулы можно найти на стр. 178.

Пример Д.2.1. Используя алгоритм Уоршелла, вычислить матрицу достижимости орграфа $G = (V, E)$, изображенного на рисунке рис. Д.2, и на ее основе получить матрицу сильной связности.

Решение. Матрица смежности данного орграфа имеет вид

$$M = \begin{bmatrix} Л & Л & И & Л & Л \\ Л & Л & Л & И & Л \\ Л & И & Л & И & И \\ Л & И & Л & Л & Л \\ И & Л & Л & И & Л \end{bmatrix}.$$

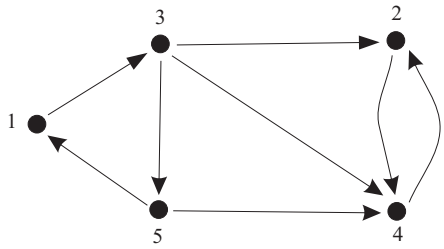


Рисунок Д.2.

Согласно алгоритму, первая матрица W_0 отличается от матрицы смежности тем, что у нее на главной диагонали стоят буквы «*И*». Таким образом,

$$W_0 = M \textbf{ или } E = \begin{bmatrix} И & Л & И & Л & Л \\ Л & И & Л & И & Л \\ Л & И & И & И & И \\ Л & И & Л & И & Л \\ И & Л & Л & И & И \end{bmatrix}.$$

Для определения матрицы W_1 рассмотрим первый столбец матрицы W_0 и скопируем все строки, у которых на первом месте стоит буква «*Л*», в матрицу W_1. Получим

$$W_1 = \begin{bmatrix} ? & ? & ? & ? & ? \\ Л & И & Л & И & Л \\ Л & И & И & И & И \\ Л & И & Л & И & Л \\ ? & ? & ? & ? & ? \end{bmatrix}.$$

Чтобы определить первую строку матрицы W_1, действуем, следуя указаниям на стр. 178. Первую строку матрицы W_0 спариваем с первой же строкой матрицы W_0 с помощью операции **или**, а результат записываем в первую строку матрицы W_1. Иными словами, первую строку матрицы W_0 переписываем без изменения в первую строку матрицы W_1.

$$W_1 = \begin{bmatrix} И & Л & И & Л & Л \\ Л & И & Л & И & Л \\ Л & И & И & И & И \\ Л & И & Л & И & Л \\ ? & ? & ? & ? & ? \end{bmatrix}.$$

Осталось определить только пятую строку матрицы W_1. Спариваем первую строку матрицы W_0 с пятой ее строкой с помощью операции **или** и записываем результат в пятую строку матрицы W_1. Итак,

$$W_1 = \begin{bmatrix} И & Л & И & Л & Л \\ Л & И & Л & И & Л \\ Л & И & И & И & И \\ Л & И & Л & И & Л \\ И & Л & И & И & И \end{bmatrix}.$$

Строим матрицу W_2. Поскольку в первой и пятой строках матрицы W_1 на втором месте стоит «*Л*», то копируем эти строки в матрицу W_2. Вторую строку матрицы W_1 последовательно спариваем операцией **или** со второй, третьей и четвертой строками матрицы W_1, записывая результат в соответствующие строки матрицы W_2. Получаем

$$W_2 = \begin{bmatrix} И & Л & И & Л & Л \\ Л & И & Л & И & Л \\ Л & И & И & И & И \\ Л & И & Л & И & Л \\ И & Л & И & И & И \end{bmatrix}.$$

При формировании матрицы W_3 важную роль играет третий столбец матрицы W_2. Поскольку в этом столбце и строках с номерами 2 и 4 стоит буква «*Л*», переписываем указанные строки в матрицу W_3 без изменения. Затем первую, третью и пятую строки из W_2 мы должны спарить с третьей и записать результат в соответствующие строки матрицы W_3. Получаем

$$W_3 = \begin{bmatrix} И & И & И & И & И \\ Л & И & Л & И & Л \\ Л & И & И & И & И \\ Л & И & Л & И & Л \\ И & И & И & И & И \end{bmatrix}.$$

Строим матрицу W_4. Для этого рассмотрим четвертый столбец матрицы W_3. Все строки этого столбца содержат буквы «*И*», следовательно нам необходимо спарить операцией **или** четвертую строку с остальными строками матрицы W_3

и результаты записать в соответствующие строки матрицы W_4. Но все строки W_3 содержат буквы «*И*» в тех же столбцах, что и четвертая строка (столбцы 2 и 4), следовательно они останутся без изменений. То есть матрицы W_3 и W_4 совпадают.

$$W_4 = \begin{bmatrix} И & И & И & И & И \\ Л & И & Л & И & Л \\ Л & И & И & И & И \\ Л & И & Л & И & Л \\ И & И & И & И & И \end{bmatrix}.$$

Найдем последнюю матрицу W_5. Пятый столбец матрицы W_4 содержит буквы «*Л*» во второй и четвертой строках, поэтому эти строки перейдут в матрицу W_5 без изменений. Остальные строки матрицы W_4 спариваем операцией **или** с пятой строкой и переписываем в матрицу W_5. Получаем

$$W_5 = \begin{bmatrix} И & И & И & И & И \\ Л & И & Л & И & Л \\ И & И & И & И & И \\ Л & И & Л & И & Л \\ И & И & И & И & И \end{bmatrix}.$$

Таким образом, матрица достижимости данного орграфа равна

$$W = \begin{bmatrix} И & И & И & И & И \\ Л & И & Л & И & Л \\ И & И & И & И & И \\ Л & И & Л & И & Л \\ И & И & И & И & И \end{bmatrix}.$$

Используя формулы для вычисления матрицы сильной связности (стр. 282), получим

$$C = \begin{bmatrix} И & Л & И & Л & И \\ Л & И & Л & И & Л \\ И & Л & И & Л & И \\ Л & И & Л & И & Л \\ И & Л & И & Л & И \end{bmatrix}.$$

Если граф имеет небольшое количество вершин и ребер, то глядя на его матрицу связности или графическое изображения графа, можно легко выделить компоненты связности. Так орграф из примера Д.2.1 имеет две компоненты связности (см. рис. Д.3).

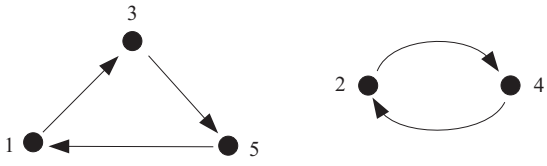


Рисунок Д.3.

Однако в задачах, возникающих на практике, число вершин графа очень велико и ручной метод выделения компонент нерентабелен. Поэтому необходим алгоритм выделения компонент связности, реализуемый на компьютере.

Д.2.2. Выделение компонент связности

Опишем алгоритм нахождения числа компонент сильной связности ориентированного графа, а также выделения самих компонент. Аналогичным образом решается задача нахождения числа компонент связности и их выделения в случае неориентированного графа.

Идея алгоритма основана на том, что элементы «*И*» любой i-ой строки матрицы связности графа или матрицы сильной связности орграфа соответствуют всем вершинам, содержащимся в одной компоненте связности с вершиной i. Удаление из матриц смежности и связности строк и столбцов, соответствующих этим вершинам, соответствует удалению из исходного графа одной компоненты связности. С полученным в результате новым графом можно повторить описанные действия, выделив следующую компоненту связности, и так далее, до тех пор, пока не получится матрица, не имеющая ни столбцов, ни строк. Такую матрицу принято называть пустой и обозначать символом $\varnothing$.

Input
V — множество вершин орграфа G;
M — матрица смежности орграфа G;
C — матрица сильной связности орграфа G;
begin
$p := 0$;
$B := C$;
while $B \neq \varnothing$ **do**
begin
$p := p + 1$;
включить в множество V_p вершины из V,
соответствующие элементам И первой строки
матрицы B;
в качестве матрицы M_p взять подматрицу
матрицы M, находящуюся на пересечении
строк и столбцов, соответствующих
вершинам из V_p;
вычеркнуть из B строки и столбцы,
соответствующие вершинам из V_p;
удалить из V вершины V_p;
end
end
Output
p — число компонент сильной связности орграфа G;
V_i, $i = 1, 2, \ldots p$, — множество вершин i-той компоненты
связности G_i орграфа G;
M_i, $i = 1, 2, \ldots p$, — матрица смежности i-той
компоненты связности G_i орграфа G.

Пример Д.2.2. Определить компоненты сильной связности орграфа G, полученного с помощью генератора случайных орграфов в примере Д.1.1, и изобразить их отдельно.

Решение. Матрица смежности орграфа равна

$$M = \begin{bmatrix} Л & Л & Л & Л & И \\ И & Л & И & И & Л \\ И & Л & Л & Л & Л \\ Л & Л & И & Л & И \\ Л & Л & Л & И & Л \end{bmatrix}.$$

Применив метод Уоршелла, получим матрицу достижимости:

$$W = \begin{bmatrix} И & Л & И & И & И \\ И & И & И & И & И \\ И & Л & И & И & И \\ И & Л & И & И & И \\ И & Л & И & И & И \end{bmatrix}.$$

Затем, используя формулы для вычисления матрицы сильной связности (стр. 282), находим

$$C = \begin{bmatrix} И & Л & И & И & И \\ Л & И & Л & Л & Л \\ И & Л & И & И & И \\ И & Л & И & И & И \\ И & Л & И & И & И \end{bmatrix}.$$

В соответствии с алгоритмом полагаем $p{:=}0$, $B = C$. Матрица $B \neq \varnothing$, потому выполняем тело цикла.

- полагаем $p := p + 1 = 0 + 1 = 1$;
- $V_1 := \{1, 3, 4, 5\}$;
- формируем M_1, взяв из M элементы, стоящие на пересечении строк и столбцов, соответствующих элементам множества V_1:

$$M_1 := \begin{bmatrix} Л & Л & Л & И \\ И & Л & Л & Л \\ Л & И & Л & И \\ Л & Л & И & Л \end{bmatrix}.$$

На этом этапе мы сформировали множество вершин V_1 и матрицу смежности M_1 первой компоненты сильной связности орграфа G.

- вычеркиваем из B строки и столбцы, соответствующие вершинам из V_1:

$$B := [И];$$

- вычеркиваем из V вершины, соответствующие V_1, $V := \{2\}$.

Так как $B \neq \varnothing$, повторяем тело цикла.

- полагаем $p := p + 1 = 1 + 1 = 2$;
- формируем $V_2 := \{2\}$;

– формируем M_2, взяв из M элементы, стоящие на пересечении строки и столбца, соответствующих вершине 2 множества V_2:

$$M_2 := [Л].$$

Таким образом, сформировано множество вершин V_2 и матрица смежности M_2 второй компоненты сильной связности орграфа G.

– вычеркиваем из B строки и столбцы, соответствующие вершинам из V_2:

$$B := \varnothing.$$

Так как $B = \varnothing$, то завершаем цикл и, соответственно, весь алгоритм. В результате получены множества вершин и матрицы смежности двух компонент сильной связности орграфа G. Компоненты связности изображены на рис. Д.4.

Д.3. Эйлеровы циклы

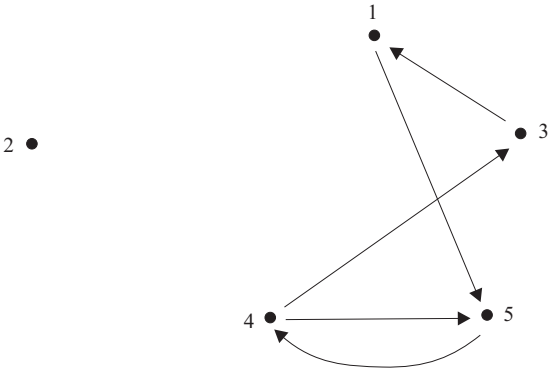


Рисунок Д.4.

В начале седьмой главы основной части книги Вы познакомились с классической задачей о кенигсбергских мостах, при решении которой вводилось понятие эйлерова цикла и эйлерова графа. Там же было доказано, что степень каждой вершины эйлерова графа должна быть четной, и сообщался такой факт: условие четности степеней всех вершин является достаточным для существования эйлерова цикла в графе. Задача о поиске эйлерова цикла в данном графе имеет практическое значение. Не углубляясь в высокую науку, можно привести задачу уборки улиц города, состоящую в следующем. Необходимо составить маршрут движения уборочной машины по улицам заданного района города таким образом, чтобы он начинался и заканчивался в одном и том же месте и проходил по каждой улице не более одного раза, если это возможно. Схему улиц района можно представить в виде графа, ребрам которого соответствуют улицы, а вершинам — перекрестки. В этом случае сформулированная задача сводится к построению эйлерова цикла в графе.

Задачу построения эйлерова цикла (если он существует) можно решить, например, с помощью алгоритма, основанного на следующем правиле. Начнем маршрут P в произвольной вершине a графа G и будем продолжать его, насколько возможно, все время через новые ребра. Так как к каждой вершине подходит четное число ребер, этот процесс может закончиться только в исходной вершине a. При этом получится замкнутый маршрут, проходящий через каждое свое

ребро по одному разу, т. е. цикл $P = a\,a_1\,a_2\,\ldots\,a_k\,a$. Если P содержит не все ребра G, удалим из G ребра, входящие в P.

Вершины графа G и цикла P имеют четные степени. То же можно сказать и про остающийся подграф G'. Так как граф G связен, то в P должна найтись вершина a_s, инцидентная какому-то ребру из G'. Из a_s можно построить новый маршрут P', содержащий ребра только из подграфа G'. Такой маршрут может закончиться только при возвращении в вершину a_s, т. е. $P' = a_s\,b_1\,\ldots\,b_m\,a_s$. Но тогда из P и P' можно составить новый цикл

$$P_1 = a\,a_1\ \ldots\,a_s\,b_1\,\ldots\,b_m\,a_s\,a_{s+1}\,\ldots\,a_m\,a,$$

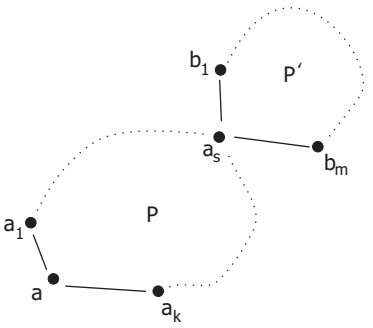


Рисунок Д.5.

который возвращается в a и содержит больше ребер, чем P (см. рис. Д.5). Если P_1 не является эйлеровым циклом, то это построение повторяется. Когда процесс закончится, эйлеров цикл будет построен.

Д.3.1. Алгоритм построения эйлерова цикла в графе

Input
$G(V, E)$ — конечный связный граф с четными степенями всех вершин;
begin
выбрать любую вершину $d \in V$;
$P := \varnothing$;
$a := d$;
while $E \neq \varnothing$ **do**
begin
выбрать вершину $d \in P$, для которой существует ребро $(d, x) \in E$;
$c := d$;
$P' := \varnothing$;
repeat
begin
выбрать ребро $(d, y) \in E$ с условием:
$(d, y) \notin P'$;
$P' := P' \cup \{(d, y)\}$;
удалить ребро (d, y) из E;
$d := y$;
end;
until $y \neq c$
if $P \neq \varnothing$ **then**
$P := P(a, c) \cup P' \cup P(c, a)$;

else
$P := P'$;
end
end
Output P — *эйлеров цикл графа* G.

Пример Д.3.1. Построить эйлеров цикл в графе $G(V, E)$, если

$V = \{1, 2, 3, 4, 5, 6\}$ и
$E = \{(1,2), (1,4), (2,3), (2,4), (2,5), (3,4), (4,6), (5,6)\}$.

Решение. Условимся для определенности: просматривая элементы слева направо в соответствующих множествах, выбирать первый подходящий элемент. На начальном этапе алгоритма выбираем вершину $d \in V$.

- Положим $d := 1$;
- $P := \varnothing$;
- $a := d = 1$.

Выполняем операторы внешнего цикла **while** $E \neq \varnothing$ **do ... end**. Так как $P = \varnothing$, новую вершину выбрать невозможно и d сохраняет значение, равное 1.

- $c := d = 1$;
- $P' := \varnothing$.

Выполняем операторы внутреннего цикла **repeat ... until** $y \neq c$.

- Выбираем ребро $(d, y) \in E$ с условием: $(d, y) \notin P'$, где $d = 1$ (это ребро $(1, 2)$);
- включаем его в маршрут: $P' := 1\,2$;
- удаляем ребро $(1, 2)$ из E ($E := \{(1,4), (2,3), (2,4), (2,5), (3,4), (4,6), (5,6)\}$);
- $d := y = 2$.

- Выбираем ребро $(d, y) \in E$ с условием: $(d, y) \notin P'$, где $d = 2$ (это ребро $(2, 3)$);
- включаем его в маршрут: $P' := 1\,2\,3$;
- удаляем ребро $(2, 3)$ из E ($E := \{(1,4), (2,4), (2,5), (3,4), (4,6), (5,6)\}$);
- $d := y = 3$.

- Выбираем такое ребро $(d, y) \in E$, что $(d, y) \notin P'$, где $d = 3$ (это ребро $(3, 4)$);
- включаем его в маршрут: $P' := 1\,2\,3\,4$;

– удаляем ребро $(3,4)$ из E ($E := \{(1,4), (2,4), (2,5), (4,6), (5,6)\}$);
– $d := y = 4$.

– Выбираем такое ребро $(d,y) \in E$, что $(d,y) \notin P'$, где $d = 4$ (это ребро $(4,1)$);
– включаем его в маршрут: $P' := 1\,2\,3\,4\,1$;
– удаляем ребро $(1,4)$ из E ($E := \{(2,4), (2,5), (4,6), (5,6)\}$);
– $d := y = 1$.

Выполнилось условие окончания внутреннего цикла, поскольку y стал равен c. Так как $P = \varnothing$, полагаем $P := P' = 1\,2\,3\,4\,1$. Поскольку $E \neq \varnothing$, повторяем внешний цикл.

– Выбираем вершину $d \in P$, для которой существует ребро (d,x), принадлежащее E (это вершина $d = 2$);
– $c := d = 2$;
– $P := \varnothing$.

Снова переходим к выполнению операторов внутреннего цикла **repeat ... until** $y \neq c$.

– Выбираем такое ребро $(d,y) \in E$, что $(d,y) \notin P'$, где $d = 2$ (это ребро $(2,4)$);
– включаем его в маршрут: $P' := 2\,4$;
– удаляем ребро $(2,4)$ из E ($E := \{(2,5), (4,6), (5,6)\}$);
– $d := y = 4$.

– Выбираем такое ребро $(d,y) \in E$, что $(d,y) \notin P'$, где $d = 4$ (это ребро $(4,6)$);
– включаем его в маршрут: $P' := 2\,4\,6$;
– удаляем ребро $(4,6)$ из E ($E := \{(2,5), (5,6)\}$);
– $d := y = 6$.

– Выбираем такое ребро $(d,y) \in E$, что $(d,y) \notin P'$, где $d = 6$ (это ребро $(6,5)$);
– включаем его в маршрут: $P' := 2\,4\,6\,5$;
– удаляем ребро $(6,5)$ из E ($E := \{(2,5)\}$);
– $d := y = 5$.

- Выбираем такое ребро $(d, y) \in E$, что $(d, y) \notin P'$, где $d = 5$ (это ребро $(5, 2)$);
- включаем его в маршрут: $P' := 2\,4\,6\,5\,2$;
- удаляем ребро $(5, 2)$ из E $(E := \varnothing)$;
- $d := y = 2$.

Поскольку $y = c = 2$, внутренний цикл закончен. Так как $P \neq \varnothing$, полагаем

$$P := P(a, c) \cup P' \cup P(c, a) = P(1, 2) \cup P' \cup P(2, 1) =$$
$$= 1\,2\,4\,6\,5\,2\,3\,4\,1.$$

Кроме того, к настоящему моменту $E = \varnothing$. Следовательно, закончен внешний цикл и весь алгоритм. В результате построен эйлеров цикл: $P = 2\,4\,6\,5\,2\,3\,4\,1$.

Д.3.2. Алгоритм Терри

Если G не является эйлеровым графом, а что-то вроде эйлерова цикла построить необходимо, то следует рассмотреть задачу построения нескольких маршрутов, покрывающих все его ребра. Решить эту задачу можно следующим образом. Обозначим через k число вершин графа G нечетной степени (известно, что k — четно). Очевидно, каждая вершина нечетной степени должна быть концом хотя бы одного из покрывающих G маршрутов P_i. Следовательно, количество таких маршрутов равно $k/2$. С другой стороны, таким количеством маршрутов граф G покрыть можно. Чтобы убедиться в этом, расширим G до нового графа G', добавив $k/2$ ребер E', соединяющих различные пары вершин нечетной степени. Тогда G' оказывается эйлеровым графом и имеет эйлеров цикл P'. После удаления из P' ребер E' граф разложится на $k/2$ участков, покрывающих G.

В качестве графа с эйлеровым циклом можно рассматривать, например, разумную схему обхода выставки, по различным коридорам которой посетители должны, согласно указателям, пройти так, чтобы увидеть каждый экспонат по одному разу. Если экспонаты расположены по обе стороны коридоров, то можно направлять посетителей через них дважды, чтобы каждая сторона была осмотрена отдельно.

Это не накладывает ограничений на основной план расположения выставки, так как в конечном связном графе всегда можно построить ориентированный цикл, проходящий через каждое ребро по одному разу в каждом из двух направлений. Для этого достаточно удвоить ребра в данном графе G, расщепляя каждое на пару дуг противоположной ориентации. Такая процедура называется «удвоением» графа. Получившееся удвоение G_d графа G, очевидно, является ориентированным графом, удовлетворяющим условиям существования эйлерова цикла.

Имеет смысл рассматривать и задачу составления экскурсионного маршрута для осмотра достопримечательностей города, то есть составления маршрута

движения автобуса по заданной схеме городских улиц. Ясно, что маршрут желательно составить так, чтобы он не проходил более одного раза по каждой из улиц.

Для нахождения цикла, проходящего через каждое ребро графа по одному разу в каждом из двух направлений, можно использовать правило, предложенное Тэрри.

Начиная с произвольной вершины a_0, необходимо идти по какому-нибудь маршруту P, отмечая на каждом ребре направление, в котором оно было пройдено. Если мы приходим в некоторую вершину g в первый раз, то особо отмечается первое входящее ребро. Из вершины g всегда следуем по ребру (g, r), которое или вообще еще не было пройдено, или же было пройдено в противоположном направлении. При этом по первому входящему в g ребру можно идти только тогда, когда других возможностей не остается.

Очевидно, при каждом проходе через вершину g в цепи P будет одно входящее ребро и одно выходящее ребро; следовательно, P может закончиться только в a_0. Цепь P должна покрывать все ребра G по одному разу в каждом направлении. Сначала проверим это для всех ребер с концом в a_0. Так как P в вершине a_0 кончается, все ребра с концом в a_0 должны быть уже покрыты в направлении от a_0; так как в P число входящих и выходящих ребер одно и то же для каждой вершины, каждое ребро в a_0 должно оказаться покрытым в обоих направлениях.

Требуемое утверждение получается индукцией по остальным вершинам из P. Пройдем по P от a_0 до некоторой вершины a_n и предположим, что в предыдущих вершинах a_i все ребра покрыты в обоих направлениях. Входящее в a_n ребро имеет вид (a_i, a_n) и, по предположению, покрыто в обоих направлениях. Но первое входящее в a_n ребро может быть использовано в P только при отсутствии других свободных выходов; следовательно, все ребра в a_n также должны оказаться покрытыми в обоих направлениях. Приведем алгоритм Терри.

Input
n — число вершин графа;
V — множество вершин графа;
m — число ребер графа;
E — множество ребер графа;
M — матрица смежности графа;
begin
выбрать произвольную вершину $a \in V$;
$P := \varnothing$;
$k := 2m$;
$V' := V$;
while $k \neq 0$ **do**

begin
выбрать ребро (a,b), *для которого* $M(a,b) = И$, причем в последнюю очередь выбирать ребро (a,b), для которого $M(b,a) = Л$;
if $b \in V'$ **then**
begin
удалить b *из* V';
положить $M(a,b) := Л$;
end
else
положить $M(a,b) := Л$;
$P := P \cup \{(a,b)\}$;
$k := k - 1$;
$a := b$;
end
end

Output P — *ориентированный цикл, проходящий через каждое ребро графа по одному разу в каждом из двух направлений.*

Пример Д.3.2. С помощью алгоритма Терри постройте ориентированный цикл, проходящий через каждое ребро изображенного на рис. Д.6 графа по одному разу в каждом из двух направлений.

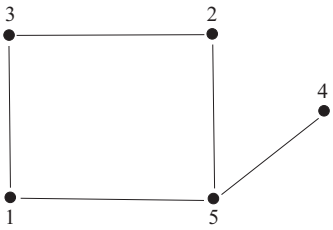


Рисунок Д.6.

Решение. Для определенности, при выборе ребер графа для добавления в цикл, будем выбирать первое подходящее ребро. В качестве начальной возьмем вершину с номером 1. Процесс работы алгоритма представлен в табл. Д.4.2. Первая строка таблицы соответствует начальному состоянию массивов и переменных.

Таким образом, в результате работы алгоритма построен следующий цикл:

$$P = \big\{(1,2),\ (2,3),\ (3,1),\ (1,5),\ (5,2),\ (2,1),(1,3),\ (3,2),\ (2,5),\ (5,4),\ (4,5),\ (5,1)\big\}.$$

Д.4. Операции над множествами

Наиболее продуктивный подход к разработке эффективного алгоритма для решения любой задачи — изучить ее сущность. Довольно часто задачу можно сформулировать на языке теории множеств, относящейся к фундаментальным

разделам математики. В этом случае алгоритм ее решения можно изложить в терминах основных операций над множествами. К таким задачам относятся и задачи информационного поиска, в которых решаются проблемы, связанные с линейно упорядоченными множествами (определение линейного порядка приведено на стр. 81).

Таблица Д.4.2

k	a	(a,b)	V'	P	M
12	1		$\{1,2,3,4,5\}$	$\varnothing$	$\begin{bmatrix} Л & И & И & Л & И \\ И & Л & И & Л & И \\ И & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & И & Л & И & Л \end{bmatrix}$
11	2	$(1,2)$	$\{1,3,4,5\}$	$\{(1,2)\}$	$\begin{bmatrix} Л & Л & И & Л & И \\ И & Л & И & Л & И \\ И & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & И & Л & И & Л \end{bmatrix}$
10	3	$(2,3)$	$\{1,4,5\}$	$\{(1,2),\ (2,3)\}$	$\begin{bmatrix} Л & Л & И & Л & И \\ И & Л & Л & Л & И \\ И & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & И & Л & И & Л \end{bmatrix}$
9	1	$(3,1)$	$\{4,5\}$	$\{(1,2),\ (2,3),\ (3,1)\}$	$\begin{bmatrix} Л & Л & И & Л & И \\ И & Л & Л & Л & И \\ Л & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & И & Л & И & Л \end{bmatrix}$
8	5	$(1,5)$	$\{4\}$	$\{(1,2),\ (2,3),\ (3,1),\ (1,5)\}$	$\begin{bmatrix} Л & Л & И & Л & Л \\ И & Л & Л & Л & И \\ Л & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & И & Л & И & Л \end{bmatrix}$
7	2	$(5,2)$	$\{4\}$	$\{(1,2),\ (2,3),$ $(3,1),\ (1,5),\ (5,2)\}$	$\begin{bmatrix} Л & Л & И & Л & Л \\ И & Л & Л & Л & И \\ Л & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & Л & Л & И & Л \end{bmatrix}$
6	1	$(2,1)$	$\{4\}$	$\{(1,2),\ (2,3),\ (3,1),$ $(1,5),\ (5,2),\ (2,1)\}$	$\begin{bmatrix} Л & Л & И & Л & Л \\ Л & Л & Л & Л & И \\ Л & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & Л & Л & И & Л \end{bmatrix}$

Продолжение табл. Д.4.2

5	3	$(1,3)$	$\{4\}$	$\{(1,2),\ (2,3),\ (3,1),\ (1,5),$ $(5,2)\,(2,1),\ (1,3)\}$	$\begin{bmatrix} Л & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ Л & И & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & Л & Л & И & Л \end{bmatrix}$
4	2	$(3,2)$	$\{4\}$	$\{(1,2),\ (2,3),\ (3,1),\ (1,5),$ $(5,2)\,(2,1),\ (1,3),\ (3,2)\}$	$\begin{bmatrix} Л & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & Л & Л & И & Л \end{bmatrix}$
3	5	$(2,5)$	$\{4\}$	$\{(1,2),\ (2,3),\ (3,1),\ (1,5),\ (5,2),$ $(2,1),\ (1,3),\ (3,2),\ (2,5)\}$	$\begin{bmatrix} Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & Л & Л & И & Л \end{bmatrix}$
2	4	$(5,4)$	$\varnothing$	$\{(1,2),\ (2,3),\ (3,1),\ (1,5),\ (5,2),$ $(2,1),\ (1,3),\ (3,2),\ (2,5),\ (5,4)\}$	$\begin{bmatrix} Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & И \\ И & Л & Л & Л & Л \end{bmatrix}$
1	5	$(4,5)$	$\varnothing$	$\{(1,2),\ (2,3),\ (3,1),\ (1,5),\ (5,2)\,(2,1),$ $(1,3),\ (3,2),\ (2,5),\ (5,4),\ (4,5)\}$	$\begin{bmatrix} Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ И & Л & Л & Л & Л \end{bmatrix}$
0	4	$(5,1)$	$\varnothing$	$\{(1,2),\ (2,3),\ (3,1),\ (1,5),\ (5,2)\,(2,1),$ $(1,3),\ (3,2),\ (2,5),\ (5,4),\ (4,5),\ (5,1)\}$	$\begin{bmatrix} Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \\ Л & Л & Л & Л & Л \end{bmatrix}$

Многие задачи, встречающиеся на практике, сводятся к одной или нескольким подзадачам, которые можно абстрактно сформулировать как последовательность основных команд, выполняемых на некоторой базе данных (универсальном множестве элементов). Например, выполнение последовательности команд, состоящих из операций **поиск**, **вставка**, **удаление** и **минимум**, составляет существенную часть задач поиска. Об этих и других подобных командах и пойдет речь ниже.

Рассмотрим несколько основных операций над множествами, типичных для задач информационного поиска.

- **Поиск**(a, S) определяет, принадлежит ли элемент a множеству S; если $a \in S$, результат операции — ответ «да»; в противном случае — ответ «нет».

- **Вставка**(a, S) добавляет элемент a в множество S, то есть заменяет S на $S \cup \{a\}$.
- **Алгоритм правильного обхода**(S) печатает элементы множества S с соблюдением порядка.
- **Удаление**(a, S) удаляет элемент a из множества S, то есть заменяет S на $S \setminus \{a\}$.
- **Объединение**(S_1, S_2, S_3) объединяет множества S_1 и S_2, т. е. строит множество $S_3 = S_1 \cup S_2$.
- **Минимум**(S) выдает наименьший элемент множества S.

С первыми тремя операциями Вы уже знакомы. Алгоритмы их решения приведены в приложении к главе 7 на стр. 167 и 169 . Следующая операция, которой необходимо владеть, — это операция **минимум**(S). Для нахождения наименьшего элемента в двоичном дереве поиска T строится путь v_0, v_1, ..., v_p, где v_0 — корень дерева T, v_i — левый сын вершины v_{i-1} $(i = 1, 2, \ldots, p)$, а у вершины v_p левый сын отсутствует. Ключ вершины v_p и является наименьшим элементом в дереве T. В некоторых задачах полезно иметь указатель на вершину v_p, чтобы обеспечить доступ к наименьшему элементу за постоянное время.

Алгоритм выполнения операции **минимум**(S) использует рекурсивную процедуру **левый сын**(v), определяющую левого сына вершины v. Алгоритм и процедура выглядят следующим образом.

Input
двоичное дерево поиска T для множества S
begin
if $T = \varnothing$ **then output** *«дерево пусто»;*
else
вызвать процедуру ***левый сын***(r)
(здесь r — корень дерева T);
минимум $:= l(v)$, где v — возврат
процедуры **левый сын**;
end
Output *ответ «дерево пусто», если T не содержит вершин;*
минимум *— наименьший элемент дерева T*
в противном случае.

Procedure левый сын(v).

begin
if v имеет левого сына w **then return левый сын**(w)
else return v
end

Пример Д.4.1. Проследите работу алгоритма **минимум** на дереве поиска, изображенном на рис. Д.7

Решение. Дерево T не пусто, следовательно вызывается процедура **левый сын**(1).

Вершина 1 имеет левого сына — вершину 2, значит вызывается процедура **левый сын**(2).

Вершина 2 имеет левого сына — вершину 4, значит вызывается процедура **левый сын**(4).

Вершина 4 не имеет левого сына, поэтому вершина 4 и будет возвратом процедуры **левый сын**.

Ключом вершины 4 является слово **begin**, следовательно наименьшим элементом дерева T является значение **минимум**= **begin**.

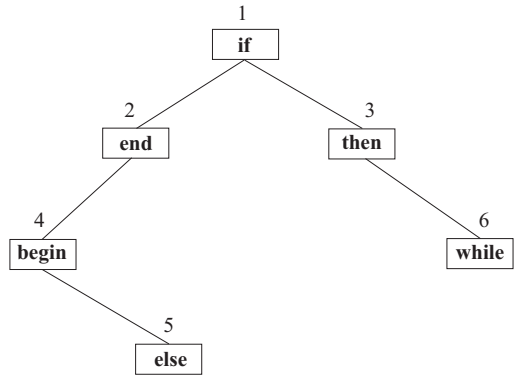


Рисунок Д.7.

Реализовать операцию **удаление**(a, S) на основе бинарных поисковых деревьев тоже просто. Допустим, что элемент a, подлежащий удалению, расположен в вершине v. Возможны три случая:

- вершина v является листом; в этом случае v просто удаляется из дерева;
- у вершины v в точности один сын; в этом случае объявляем отца вершины v отцом сына вершины v, удаляя тем самым v из дерева (если v — корень, то его сына делаем новым корнем);
- у вершины v два сына; в этом случае находим в левом поддереве вершины v наибольший элемент b, рекурсивно удаляем из этого поддерева вершину, содержащую b, и присваиваем вершине v ключ b. (Заметим, что среди элементов, меньших a, элемент b будет наибольшим элементом дерева. Кроме того, вершина, содержащая b, может быть или листом, являющимся чьим-то правым сыном, или иметь только левое поддерево.)

Ясно, что до выполнения операции **удаление**(a, S) необходимо проверить, не является ли дерево пустым и содержится ли элемент a в множестве S. Для этого придется выполнить операцию **поиск**(a, S), причем в случае положительного ответа необходимо выдать кроме ответа «да» и номер вершины v, ключ которой совпадает с a (далее ключ вершины v будем обозначать через $l(v)$). Кроме этого, для реализации операции **удаление**(a, S) требуется знать и номер вершины w, являющейся отцом вершины v. Саму же рекурсивную процедуру **удаление**(a, S) можно реализовать так, как показано на следующей странице.

```
Procedure удаление(a, S)
    begin
(1)     if v — лист then удалить v из T
        else
(2)         if v имеет только левого или только
```

```
                правого сына u then
(3)             if v имеет отца w then
                    назначить u сыном w
                else
(4)                 сделать u корнем дерева,
                    присвоив ему номер v
            else
(5)             найти в левом поддереве v наибольший
                элемент b, содержащийся в вершине y;
(6)             return удаление(b, S);
(7)             l(v) := b;
    end
```

Пример Д.4.2. Проследите за работой алгоритма **удаление**(a, S) для двоичного дерева поиска S, изображенного на рис. Д.7, если a — это слово «**if**».

Решение. Слово «**if**» расположено в корне с номером 1, у которого два сына (вершины 2 и 3), поэтому выполняем строку (5) алгоритма. Наибольшее слово, меньшее «**if**» (лексикографически), расположенное в левом поддереве корня и находящееся в вершине 2, — это **end**. Вызываем процедуру **удаление**(end, S).

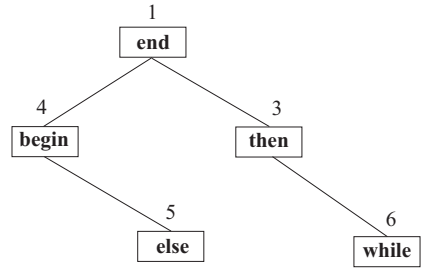


Рисунок Д.8.

Условие строки (2) алгоритма выполняется, так как вершина 2 с ключом **end** имеет только левого сына. Условие строки (3) не выполняется, так как удаляемая вершина является корнем. Поэтому переходим к строке (4): делаем вершину 2 корнем, сыновьями которого становятся вершины 4 (левым) и 3 (правым). Работа процедуры **удаление**(end, S) закончена.

Продолжаем выполнение алгоритма (выполняем строку (7)): полагаем ключ вершины 1 равным «**end**» ($l(1) :=$ **end**).

Полученное в результате дерево изображено на рис. Д.8.

Заметим, что при работе рассмотренных алгоритмов необходимо находить сыновей, отца и ключ заданной вершины двоичного дерева поиска. Это можно сделать довольно просто, если дерево в памяти компьютера хранится в виде трех массивов: LEFTSON, RIGHTSON и KEY. Эти массивы устроены следующим образом:

$$\text{LEFTSON}(i) = \begin{cases} v, \text{ если } v \text{ — левый сын вершины } i, \\ \star, \text{ если у вершины } i \text{ левого сына нет;} \end{cases}$$

$$\text{RIGHTSON}(i) = \begin{cases} v, \text{ если } v \text{ — правый сын вершины } i, \\ \star, \text{ если у вершины } i \text{ правого сына нет;} \end{cases}$$

$$\text{KEY}(i) = l(i) \text{ — ключ вершины } i.$$

Например, бинарное поисковое дерево, изображенное на рис. Д.7, с помощью этих массивов представляется следующим образом.

i	LEFTSON	RIGHTSON	KEY
1	2	3	**if**
2	4	$\star$	**end**
3	$\star$	6	**then**
4	$\star$	5	**begin**
5	$\star$	$\star$	**else**
6	$\star$	$\star$	**while**

Правила поиска сыновей и ключа заданной вершины следуют из определения массивов. Определение отца заданной вершины состоит в нахождении строки массивов LEFTSON или RIGHTSON, в которой находится номер заданной вершины. Например, отцом вершины 4 является вершина 2, так как число 4 находится во второй строке массива LEFTSON.

Д.4.1. Объединение множеств

Обратимся теперь к объединению множеств, то есть к операции **объединение**(S_1, S_2, S_3).

Если множества S_1 и S_2 линейно упорядочены, то естественно требовать такого порядка и от их объединения. Один из возможных способов объединения состоит в последовательном выполнении описанной выше операции **вставка** для добавления каждого элемента одного множества в другое. При этом, очевидно, операцию **вставка** предпочтительнее выполнять над элементами меньшего множества с целью сокращения времени на объединение. Отметим, что такой способ работы подходит как для непересекающихся, так и для пересекающихся множеств.

Во многих задачах объединяются непересекающиеся множества. Это упрощает процесс, так как исчезает необходимость каждый раз проверять, содержится ли добавляемый элемент в множестве, либо удалять повторяющиеся экземпляры одного и того же элемента из S_3.

Процедура объединения непересекающихся множеств применяется, например, при построении минимального остовного дерева в данном нагруженном графе.

Рассмотрим алгоритм объединения непересекающихся множеств на основе списков.

Будем считать, что элементы объединяемых множеств пронумерованы натуральными числами $1, 2, \ldots, n$. Кроме того, предположим, что имена множеств — также натуральные числа. Это не ограничивает общности, поскольку имена множеств можно просто пронумеровать натуральными числами.

Представим множества, рассматриваемые в рамках решения какой-то задачи, в виде совокупности линейных связанных списков. Такую структуру данных можно организовать следующим образом.

Сформируем два массива R и NEXT размерности n, в которых $\mathrm{R}(i)$ — имя множества, содержащего элемент i, а $\mathrm{NEXT}(i)$ указывает номер следующего элемента массива R, принадлежащий тому же множеству, что и элемент i. Если i — последний элемент какого-то множества, то положим $\mathrm{NEXT}(i) = 0$. Для указателей на первые элементы каждого множества будем использовать массив LIST, число элементов которого равно числу рассматриваемых в задаче множеств, и элемент которого $\mathrm{LIST}(j)$ содержит номер первого элемента множества с именем j в массиве R.

Кроме того, сформируем массив SIZE, каждый элемент которого $\mathrm{SIZE}(j)$ содержит количество элементов множества с именем j.

Будем различать внутренние имена множеств, содержащиеся в массиве R, и внешние имена, получаемые множествами в результате выполнения операции **объединение**. Соответствие между внутренними и внешними именами множеств можно установить с помощью массивов EXT-NAME и INT-NAME. Элемент массива $\mathrm{EXT\text{-}NAME}(j)$ содержит внешнее имя множества, внутреннее имя которого есть j, а $\mathrm{INT\text{-}NAME}(k)$ — внутреннее имя множества, внешнее имя которого есть k.

Пример Д.4.3. Используя только что определенные массивы, опишите множества $\{1, 3, 5, 7\}, \{2, 4, 8\}, \{6\}$ с внешними именами 1, 2, 3 и внутренними именами 2, 3, 1 соответственно.

Решение. Прежде всего отметим, что общее количество элементов всех множеств равно восьми. Поэтому размерность массивов R и NEXT будет $n = 8$. Кроме того, напомним, что номера элементов массивов LIST, SIZE, EXT-NAME и значения элементов массива R — это внутренние имена множеств, а массива INT-NAME — внешние. Структуру данных для представления указанных множеств занесем в таблицы.

Таблица Д.4.3

	R	NEXT
1	2	3
2	3	4
3	2	5
4	3	8
5	2	7
6	1	0
7	2	0
8	3	0

Таблица Д.4.4

	LIST	SIZE	EXT-NAME
1	6	1	3
2	1	4	1
3	2	3	2

Таблица Д.4.5

	INT-NAME
1	2
2	3
3	1

Алгоритм выполнения операции **объединение**(S_1, S_2, S_3) состоит в добавлении к списку элементов большего из множеств S_1 и S_2 элементов меньшего множества и присвоение полученному множеству внешнего имени S_3. При этом вычисляется также размер построенного множества S_3.

объединение(i, j, k)
Input
i, j — внешние имена объединяемых множеств
(пусть размер i меньше размера j);
массивы R, NEXT, LIST, SIZE, EXT-NAME, INT-NAME;

```
        begin
(1)         A := INT-NAME(i);
(2)         B := INT-NAME(j);
(3)         element := LIST(A);
(4)             while element ≠ 0 do
                begin
(5)                 R(element) := B;
(6)                 last := element;
(7)                 element := NEXT(element)
                end
(8)         NEXT(last) := LIST(B);
(9)         LIST(B) := LIST(A);
(10)        SIZE(B) := SIZE(B) + SIZE(A);
(11)        INT-NAME(K) := B;
(12)        EXT-NAME(B) := k
        end
```

Output *Объединение множеств i, j с внешним именем k.*

В процессе работы алгоритм совершает следующие действия:

1) определяет внутренние имена множеств i и j (строки (1) и (2));

2) определяет начало списка элементов меньшего множества (строка (3));

3) просматривает список элементов меньшего множества и изменяет соответствующие элементы массива R на внутреннее имя большего множества (строки (4)–(7));

4) объединяет множества путем подстановки меньшего множества перед бóльшим (строки (8)–(10));

5) присваивает новому множеству внешнее имя k.

Заметим, что время выполнения операции объединения двух множеств с помощью рассмотренного алгоритма пропорционально мощности меньшего множества.

Пример Д.4.4. Примените алгоритм **объединение**$(1, 2, 4)$ для объединения множеств из примера Д.4.3.

Решение. Структура данных, получившаяся в результате работы алгоритма, представлена в таблицах Д.4.6–Д.4.8.

Рассмотрим еще один важный способ объединения непересекающихся множеств. Допустим, что каждое множество представлено неупорядоченным деревом, вершинам которого поставлены в соответствие элементы множества, а корню дерева — имя самого множества. Операцию **объединение**(S_1, S_2, S_3) в этом случае можно выполнить, преобразуя корень дерева, соответствующего множеству S_1 в сына корня дерева, соответствующего S_2, и заменяя имя дерева, соответствующего S_2 на S_3 (при этом будем присоединять меньшее дерево к бóльшему). В этом случае время выполнения операции объединения двух множеств постоянно и не зависит от мощности множеств.

Таблица Д.4.6

	R	NEXT
1	2	3
2	2	4
3	2	5
4	2	8
5	2	7
6	1	0
7	2	0
8	2	1

Таблица Д.4.7

	LIST	SIZE	EXT-NAME
1	6	1	3
2	2	7	4
3	–	–	–
4	–	–	–

Таблица Д.4.8

	INT-NAME
1	–
2	–
3	1
4	2

Структуру данных для такого способа объединения множеств можно организовать с помощью массивов FATHER, ROOT, NAME, SIZE, элементы которых определяются следующим образом.

Массив FATHER содержит элементы всех рассматриваемых множеств, представленных в виде деревьев: FATHER(i) равен номеру отца вершины i. Если i — корень, то FATHER$(i) = 0$. Массив ROOT содержит номера корней деревьев для соответствующих множеств, то есть ROOT(i) равен номеру корня дерева, представляющего множество i. SIZE(i) равен числу элементов множества i. NAME(i) содержит имя множества, представленного деревом с корнем i. Если вершина i не является корнем, то NAME$(i) = 0$.

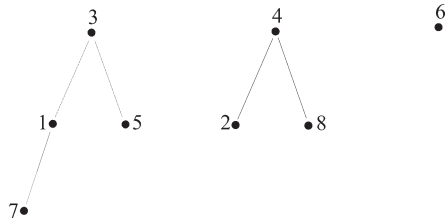


Рисунок Д.9.

Пример Д.4.5. Используя массивы FATHER, ROOT, NAME и SIZE, выпишите структуру данных для множеств из примера Д.4.3, исходя из предположения, что эти множества

$$1 = \{1, 3, 5, 7\}, \qquad 2 = \{2, 4, 8\}, \qquad 3 = \{6\}$$

представлены деревьями, изображенными на рис. Д.9.

Решение. Искомая структура данных сведена в таблицы Д.4.9 и Д.4.10.

Приведем алгоритм операции **объединение**(i, j, k).

Input

i, j — внешние имена объединяемых множеств
(пусть размер i больше размера j);
массивы FATHER , NAME , ROOT , SIZE ;

begin

$large := \text{ROOT}\,(i)$;
$small := \text{ROOT}\,(j)$;
$\text{FATHER}\,(small) := large$;
$\text{SIZE}\,(k) := \text{SIZE}\,(large) + \text{SIZE}\,(small)$;
$\text{NAME}\,(large) := k$;
$\text{NAME}\,(small) := 0$;
$\text{ROOT}\,(k) := large$;
$\text{ROOT}\,(small) := 0$;
$\text{ROOT}\,(large) := 0$;

end

Output *объединение множеств i, j с именем k.*

Таблица Д.4.9

	FATHER	NAME
1	3	0
2	4	0
3	0	1
4	0	2
5	3	0
6	0	3
7	1	0
8	4	0

Таблица Д.4.10

	ROOT	SIZE
1	3	4
2	4	3
3	6	1

Таблица Д.4.11

	FATHER	NAME
1	3	0
2	4	0
3	0	4
4	3	0
5	3	0
6	0	3
7	1	0
8	4	0

Пример Д.4.6. Примените алгоритм **объединение**(i, j, k) к множествам из примера Д.4.3. Результат работы алгоритма изобразите в виде дерева.

Решение. В результате работы операции **объединение**$(1, 2, 4)$ над данными множествами структура данных отражена в таблицах Д.4.11 и Д.4.12. Полученное в результате объединения дерево представлено на рис. Д.10.

Таблица Д.4.12

	ROOT	SIZE
1	0	0
2	0	0
3	6	1
4	3	7

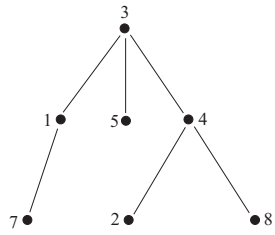


Рисунок Д.10.

Дополнение ко второму изданию

В. А. Головешкин, М. В. Ульянов

Предисловие

В 2004 году издательство «Техносфера» опубликовало в серии «Мир программирования» книгу Р. Хаггарти «Дискретная математика для программистов». Книга является хорошо зарекомендовавшим себя учебным пособием для студентов, преподавателей и программистов, и методически грамотно излагает базовые разделы дискретной математики, связанные с программированием. По своему содержанию книга Р. Хаггарти — развернутое введение в дискретную математику с большим количеством тщательно продуманных примеров и задач.

Но теория и практика программирования диктуют необходимость более широкого знакомства с дискретной математикой, в связи с чем представляется целесообразным расширить данное издание за счет включения дополнительных глав дискретной математики, посвященных вопросам оценки и исчисления конечных сумм, конечным разностям, рекурсии и методам дискретной математики, применяемым при анализе рекурсивных алгоритмов. Представляется также важным познакомить читателя с проблемой перебора и методами решения задач целочисленного программирования, играющими важную роль в практике программирования.

Материал дополнения опирается на лекционные курсы «Дискретная математика» и «Разработка эффективных алгоритмов», читаемые авторами в Московской государственной академии приборостроения и информатики.

Д.5. Дополнительные главы дискретной математики

Введение

Исчисление конечных сумм, исследование конечных разностей функций целочисленного аргумента, методы решения рекуррентных соотношений уже давно являются классическими разделами дискретной математики. Использование этих разделов в практике программирования связано, прежде всего, с теоретическим анализом алгоритмов. Очевидно, что человеку, владеющему навыками программирования, достаточно просто определить, например, количество проходов цикла путем размещения в его теле соответствующего счетчика. Однако когда речь идет о получении теоретических оценок трудоемкости алгоритма, что важно для определения области его рационального использования и обоснования правильности выбора, то использование указанных разделов дискретной математики является необходимым для получения явных функциональных зависимостей трудоемкости от длины входа.

Проиллюстрируем сказанное на небольшом фрагменте программы:

```
for i = 1 to n
  for j = i to n − i + 1
    <тело цикла>
  end for j
end for i
```

Сколько раз будет выполнено тело цикла, если $n = 10$? Ответ можно получить вычислительным экспериментом. Но ответ на вопрос, каково количество выполнений цикла как функция от n, требует применения аппарата исчисления конечных сумм. Математически задача сводится к получению функции $S(n)$ для суммы вида

$$S(n) = n + (n-2) + (n-4) + \ldots$$

Теоретический анализ рекурсивных алгоритмов неизбежно приводит к необходимости решения рекуррентных соотношений для оценки их трудоемкости. Определение количества возможных вариантов решений в задачах перебора, которым посвящен второй раздел дополнения, требует в целом ряде случаев специальных комбинаторных подсчетов, для которых одним из мощных методов получения результатов в функциональном виде является метод производящих функций.

Изложению основ соответствующих разделов дискретной математики, важных и необходимых для программистов в целях анализа и рационального выбора алгоритмов посвящен данный раздел дополнения.

Д.5.1. Исчисление и оценка конечных сумм

Понятия и обозначения. Понятие суммы часто встречается в математике, практике программирования и анализе алгоритмов. Мы опишем кратко некоторые основные методы, которые используются при их исследовании и оценке, но в начале напомним некоторые основные понятия и обозначения. Речь будем вести о конечных суммах общего вида:

$$a_1 + a_2 + \cdots + a_n,$$

где a_k — слагаемое суммы с номером k — есть некоторым образом определенное число. Каждый элемент суммы a_k называется ее членом. При этом предполагается, что он определяется по некоторому достаточно очевидному правилу, например, вычисляется, как некоторая заданная функция аргумента k, т. е. $a_k = f(k)$. В качестве примера приведем следующие суммы:

$$1 + 4 + \cdots + n^2, \tag{Д.1}$$

$$1 + \frac{1}{4} + \cdots + \frac{1}{n^2}, \tag{Д.2}$$

для суммы (Д.1) — $f(k) = k^2$, а для суммы (Д.2) — $f(k) = \frac{1}{k^2}$.

Запись суммы с использованием многоточия распространена в литературе, она наглядна, поскольку позволяет при наличии минимального воображения мы-

сленно дополнить недостающие слагаемые, однако обладает некоторыми недостатками. Во-первых — эта запись громоздка, а во-вторых — не всегда допускает однозначное толкование. Поэтому при более строгом подходе используются другие обозначения, общепринятыми являются следующие:

$$\sum_{k=1}^{n} a_k, \quad \text{или} \quad \sum_{1 \leqslant k \leqslant n} a_k. \tag{Д.3}$$

Данное обозначение суммы введено в математику Жозефом Фурье в 1820 году и называется сигма-обозначением. Отметим, что когда мы пишем сумму вида (Д.3), то предполагаются два момента. Первый — существует сравнительно простое правило вычисления членов суммы a_k по известному номеру k. Второй — нас интересует достаточно большое множество возможных значений n. Например, если нам требуется только найти сумму $1+2+3+4+5$, то нет смысла писать $1+2+\cdots+5$ или тем более $\sum\limits_{k=1}^{5} k$, а необходимо сразу написать $1+2+3+4+5=15$, и забыть про подобные «глупости».

Прежде чем перейти к дальнейшему материалу, приведем две «классические» суммы — суммы арифметической и геометрической прогрессий. Сумма арифметической прогрессии — это сумма вида

$$\sum_{k=1}^{n} (a + (k-1) \cdot d) = na + \frac{(n-1)n}{2} d, \tag{Д.4}$$

а сумма геометрической прогрессии — это сумма вида

$$\sum_{k=1}^{n} aq^{k-1} = a\frac{q^n - 1}{q - 1}. \tag{Д.5}$$

Формулы (Д.4) и (Д.5) мы получим ниже, используя различные методы исчисления конечных сумм.

Сформулируем некоторые цели, которые мы ставим перед собой, исследуя суммы. Конечно, идеальной целью является нахождение сравнительно простого вида функции $S(n)$, такой что

$$S(n) = \sum_{k=1}^{n} a_k,$$

т. е. получение явной функциональной зависимости суммы от верхнего предела суммирования. Получение $S(n)$ и является основной целью исчисления конечных сумм. Однако найти такую функцию удается не всегда, хотя ниже мы приведем некоторые методы для нахождения $S(n)$. Чаще удается решить более скромную задачу — задачу нахождения оценок суммы.

Методы нахождения оценок сумм. В задаче нахождения оценок сумм нас интересуют, как правило, оценки при достаточно больших значениях n. Прежде чем перейти к постановке задач оценки сумм, напомним некоторые основные обозначения для асимптотических оценок функций. Вводя эти обозначения, мы

заранее предполагаем, что речь идет о неотрицательных функциях, стремящихся, как правило, к бесконечности при стремлении к бесконечности аргумента n.

Обозначения асимптотических оценок функций

1. *Обозначение о-малое.* Если $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=0$, то пишем $f(n)=o(g(n))$ (читается «эф от эн есть о малое от же от эн»).

2. *Обозначение O-большое.* Если существует число c, такое что $\frac{f(n)}{g(n)}<c$ для всех значений n, то пишем $f(n)=O(g(n))$ (читается «эф от эн есть O большое от же от эн»). Заметим, что всякое o-малое является одновременно и O-большим, но не наоборот.

3. *Обозначение* Θ. Если найдутся две неотрицательные константы $c_1>0$ и $c_2>0$, а так же число n_0, такое что при всех $n>n_0$ выполнено $c_1g(n)\leqslant f(n)\leqslant c_2g(n)$, то пишем $f(n)=\Theta(g(n))$. Если $f(n)=\Theta(g(n))$ то говорят, что $g(n)$ является асимптотически точной оценкой для $f(n)$.

4. *Эквивалентность.* Если $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=1$, то пишем $f(n)\sim g(n)$ (читается «эф от эн эквивалентно же от эн»). Заметим, что если справедливо: $f(n)\sim g(n)$, то $f(n)=g(n)+o(g(n))$.

Получение подобных оценок и является основной задачей исследования при оценке сумм. Если не удается найти точное выражение для суммы

$$S(n)=\sum_{k=1}^{n}a_k,$$

то лучшим результатом является такая оценка суммы $S(n)$ — функция $g(n)$, что $S(n)\sim g(n)$. Но такую оценку в общем случае удается найти достаточно редко. Неплохой оценкой считается оценка $S(n)=\Theta(g(n))$, особенно если известны значения c_1 и c_2, а не просто доказано, что они существуют. Оценка $S(n)=O(g(n))$ также позволяет делать некоторые выводы о характере поведения суммы.

Пример Д.5.1. Получить оценку O-большое для суммы $S(n)=\sum\limits_{k=1}^{n}k^3$.

Решение. Очевидно, что для всех $1\leqslant k\leqslant n$ справедливо неравенство $k^3\leqslant n^3$, тогда получаем

$$S(n)=\sum_{k=1}^{n}k^3\leqslant\sum_{k=1}^{n}n^3=n^4.$$

Этот результат позволяет сделать вывод, что

$$S(n)=\sum_{k=1}^{n}k^3=O(n^4).$$

Метод математической индукции. Суть метода состоит в том, что мы «угадываем» решение, проверяем его правильность при $n=1$, и, считая, что результат верен при $n=n$, доказываем его для $n+1$.

Используя этот метод, попытаемся доказать, что функция $g(n) = n^4$ является асимптотически точной оценкой для следующей суммы

$$S(n) = \sum_{k=1}^{n} k^3, \quad \text{т. е.} \quad S(n) = \Theta(n^4).$$

Для этого в соответствии с определением оценки Θ требуется найти такое значение константы c_1 и такое значение n_0, что при всех $n > n_0$ выполнено

$$S(n) = \sum_{k=1}^{n} k^3 \geqslant c_1 n^4. \tag{Д.6}$$

Не будем конкретизировать значение c_1. Сам метод позволит дать оценку этому числу. Пусть для некоторого значения n выполнено условие (Д.6). Требуется так подобрать c_1, что бы для всех $n > n_0$ было выполнено неравенство

$$S(n+1) = \sum_{k=1}^{n+1} k^3 \geqslant c_1 (n+1)^4. \tag{Д.7}$$

Проведем следующую цепочку рассуждений. Имеем

$$S(n+1) = \sum_{k=1}^{n+1} k^3 = \sum_{k=1}^{n} k^3 + (n+1)^3,$$

тогда

$$S(n+1) = \sum_{k=1}^{n} k^3 + (n+1)^3 = S(n) + (n+1)^3 \geqslant c_1 n^4 + (n+1)^3.$$

Если нам удастся подобрать такое значение c_1, что

$$c_1 n^4 + (n+1)^3 \geqslant c_1 (n+1)^4 \tag{Д.8}$$

при всех значениях n, то заведомо будет выполнено неравенство (Д.7), которое нам и требуется доказать. Тогда для определения c_1 имеем неравенство (Д.8), преобразуя которое, получаем

$$(n+1)^3 \geqslant c_1 \cdot ((n+1)^4 - n^4), \quad c_1 \leqslant \frac{(n+1)^3}{4n^3 + 6n^2 + 4n + 1}.$$

Заметим, что для всех $n \geqslant 1$ выполнено

$$\frac{(n+1)^3}{4n^3 + 6n^2 + 4n + 1} > \frac{n^3}{4n^3 + 6n^3 + 4n^3 + n^3} = \frac{1}{15},$$

тогда выберем $c_1 = \frac{1}{15}$, очевидно, что $S(1) = 1^3 \geqslant \frac{1}{15} 1^4$, следовательно, мы можем утверждать, что при всех $n \geqslant 1$ выполнено

$$S(n) = \sum_{k=1}^{n} k^3 \geqslant \frac{1}{15} n^4.$$

Ранее мы показали, что при всех $n \geqslant 1$ выполнено $S(n) \leqslant n^4$, следовательно, при значениях $c_1 = \frac{1}{15}$, $c_2 = 1$, $S(n) = \Theta(n^4)$, и функция $g(n) = n^4$ является асимптотически точной оценкой для функции $S(n)$.

Метод почленных сравнений. Прежде чем излагать данный метод, напомним некоторые основные свойства конечных сумм. Первое из них — конечные суммы допускают любую перестановку слагаемых. Второе — это свойство линейности конечной суммы, которое математическим языком записывается следующим образом

$$\sum_{k=1}^{n}(ca_k + db_k) = c\sum_{k=1}^{n}a_k + d\sum_{k=1}^{n}b_k.$$

Отметим также, что при построении асимптотических оценок мы можем пренебрегать любым конечным числом слагаемых.

Метод почленных сравнений основан на следующем факте. Пусть имеются две конечные суммы

$$U(n) = \sum_{k=1}^{n}a_k,\ V(n) = \sum_{k=1}^{n}b_k,$$

при этом для всех значений k выполнено условие $a_k \leqslant b_k$. Тогда можно утверждать, что $U(n) \leqslant V(n)$. Этот факт удобен для построения асимптотических оценок. При этом для сравнения с заданной суммой выбирается такая сумма, значение которой или ее асимптотическая оценка известны.

Пример Д.5.2. Получить асимптотическую оценку для суммы

$$U(n) = \sum_{k=1}^{n}\frac{k}{k+1}2^{k-1}.$$

Решение. Для сравнения выберем геометрическую прогрессию — сумму

$$V(n) = \sum_{k=1}^{n}2^{k-1},$$

сумма которой известна — формула (Д.5) и равна $V(n) = \sum\limits_{k=1}^{n}2^{k-1} = 2^n - 1$.

Поскольку $\frac{k}{k+1} < 1$, то $U(n) \leqslant V(n) = 2^n - 1$, и можно утверждать, что

$$U(n) = O(2^n).$$

Метод сравнения с определенным интегралом. Еще один полезный прием — это сравнение с определенным интегралом. Этот прием удобно применять для исследования сумм, слагаемые которых имеют вид $a_k = f(k)$, и при этом функция действительной переменной $y = f(x)$ неотрицательна и монотонна для положительных значений переменной x.

Если функция $y = f(x)$ не убывает, то нетрудно видеть, что

$$a_k \leqslant \int\limits_{k}^{k+1} f(x)\,dx \quad \text{и} \quad a_k \geqslant \int\limits_{k-1}^{k} f(x)\,dx,$$

тогда для значения суммы — функции $S(n) = \sum\limits_{k=1}^{n} a_k$ имеем следующие оценки

$$S(n) \leqslant \int\limits_{1}^{n+1} f(x)\,dx \quad \text{и} \quad S(n) \geqslant a_1 + \int\limits_{1}^{n} f(x)\,dx.$$

Если функция $y = f(x)$ не возрастает, то нетрудно видеть, что

$$a_k \geqslant \int\limits_{k}^{k+1} f(x)\,dx \quad \text{и} \quad a_k \leqslant \int\limits_{k-1}^{k} f(x)\,dx,$$

и тогда для функции $S(n)$ имеем

$$S(n) \geqslant \int\limits_{1}^{n+1} f(x)\,dx \quad \text{и} \quad S(n) \leqslant a_1 + \int\limits_{1}^{n} f(x)\,dx.$$

Пример Д.5.3. Исследовать сумму гармонического ряда $S(n) = \sum\limits_{k=1}^{n} \frac{1}{k}$ с целью получения асимптотической оценки.

Решение. Для сравнения рассмотрим функцию $f(x) = \frac{1}{x}$, которая монотонно убывает при положительных значениях аргумента. Имеем

$$\int\limits_{1}^{n+1} \frac{1}{x}\,dx \leqslant S(n) \leqslant 1 + \int\limits_{1}^{n} \frac{1}{x}\,dx,$$

следовательно, $\ln(n+1) \leqslant S(n) \leqslant 1 + \ln(n)$. Из данного неравенства легко показать, что $S(n) \sim \ln(n)$. Отметим, что можно доказать и более сильное утверждение, а именно, что

$$S(n) = \ln(n) + O(1).$$

Пример Д.5.4. Применить метод сравнения с определенным интегралом для исследования суммы $S(n) = \sum\limits_{k=1}^{n} \ln k$.

Решение. Поскольку функция $y = \ln x$ монотонно возрастает при положительных значениях аргумента, то имеют место следующие оценки для данной суммы

$$\int\limits_{1}^{n} \ln x dx < \sum_{1}^{n} \ln k < \int\limits_{1}^{n+1} \ln x dx.$$

Поскольку $\int \ln x dx = x \ln x - x$, то получаем следующую оценку суммы

$$n \ln n - n < \sum_{k=1}^{n} \ln k < (n+1)\ln(n+1) - (n+1).$$

Полученная оценка позволяет сделать вывод, что

$$\sum_{k=1}^{n} \ln k \sim n \ln n.$$

В качестве приложения к полученному соотношению мы получим оценку значения $n!$. Напомним, что $n! = 1 \times 2 \times 3 \times \cdots \times n$, а поскольку

$$n! = e^{\ln(n!)} = e^{\sum\limits_{k=1}^{n} \ln k}, \quad \text{то} \quad \frac{n^n}{e^n} < n! < \frac{(n+1)^{n+1}}{e^{n+1}}.$$

Отметим, что в литературе существуют и более точные оценки значения $n!$, в частности формула Стирлинга (см., например [16]).

Пример Д.5.5. Доказать, что $\sum\limits_{k=1}^{n} \frac{1}{k^2} = O(1)$.

Решение. Для этого требуется доказать, что найдется значение c, такое что

$$\sum_{k=1}^{n} \frac{1}{k^2} \leqslant c$$

для всех значений n. Рассмотрим функцию $f(x) = \frac{1}{x^2}$, которая монотонно убывает при положительных значениях аргумента. Тогда

$$\sum_{k=1}^{n} \frac{1}{k^2} \leqslant 1 + \int_{1}^{n} \frac{1}{x^2}\, dx = 1 - \frac{1}{n} + 1 = 2 - \frac{1}{n} \leqslant 2,$$

что и требовалось доказать. На всякий случай напомним, что $\int \frac{dx}{x^2} = -\frac{1}{x}$.

Пример Д.5.6. Используя метод сравнения с определенным интегралом, найти эквивалентную оценку для суммы $S(n) = \sum\limits_{k=1}^{n} k^3$.

Решение. Для получения эквивалентной оценки воспользуемся следующими соображениями. Заметим, что при $k \geqslant 1$ выполнены очевидные неравенства

$$\int_{k-1}^{k} x^3\, dx < k^3 < \int_{k}^{k+1} x^3\, dx,$$

откуда

$$S(n) = \sum_{k=1}^{n} k^3 > \int_{0}^{n} x^3\, dx = \frac{1}{4} n^4,$$

но другой стороны

$$S(n) = \sum_{k=1}^{n} k^3 < \int_{1}^{n+1} x^3\, dx = \frac{1}{4}((n+1)^4 - 1).$$

Покажем, что функция $g(n) = \frac{1}{4}n^4$ эквивалентна $S(n)$ для больших n. Имеем

$$1 \leqslant \frac{S(n)}{(1/4)n^4} \leqslant \frac{(1/4)((n+1)^4-1)}{(1/4)n^4},$$
$$\lim_{n\to\infty} \frac{(1/4)((n+1)^4-1)}{(1/4)n^4} = \lim_{n\to\infty} \frac{(1+1/n)^4 - 1/n^4}{1} = 1.$$

Из полученного результата следует, что

$$\lim_{n\to\infty} \left(\frac{S(n)}{(1/4)\,n^4}\right) = 1,$$

и, следовательно

$$S(n) = \sum_{k=1}^{n} k^3 \sim \frac{1}{4}n^4.$$

Методы точного исчисления конечных сумм. Для получения функции $S(n)$ в явном виде в дискретной математике используется целый ряд методов, из которых мы изложим наиболее простые и употребительные.

Метод математической индукции. Очевидно, что этот достаточно мощный и широко используемый метод можно применить и для исчисления конечных сумм. Проиллюстрируем его применение для доказательства формулы суммы арифметической прогрессии

$$\sum_{k=1}^{n} (a + (k-1)\cdot d) = na + \frac{(n-1)n}{2}d.$$

Обозначим через

$$S(n) = \sum_{k=1}^{n} (a + (k-1)\cdot d).$$

При $n = 1$ справедливость данной формулы очевидна. Предположим, что она верна для некоторого значения n, тогда для значения $(n+1)$ имеем

$$S(n+1) = \sum_{k=1}^{n+1} (a + (k-1)\cdot d) = \sum_{k=1}^{n} (a + (k-1)\cdot d) + a + nd = S(n) + a + nd,$$

поскольку $S(n) = na + \dfrac{(n-1)n}{2}d$, то

$$S(n+1) = na + \frac{(n-1)n}{2}d + a + nd = (n+1)a + \frac{n(n+1)}{2}d,$$

что и требовалось доказать.

Метод неопределенных коэффициентов. Метод основан на том, что вид функции $S(n)$ известен, например, на основе полученных асимптотических оценок, а конкретные значения коэффициентов могут быть получены с использованием математической индукции.

Проиллюстрируем этот метод на уже знакомом примере — найдем точное значение суммы $S(n) = \sum\limits_{k=1}^{n} k^3$. В силу полученных выше оценок попытаемся искать значение суммы в виде многочлена четвертой степени от n, т. е. в виде

$$S(n) = An^4 + Bn^3 + Cn^2 + Dn + E. \qquad (Д.9)$$

Поскольку $S(1) = 1$ — базис индукции, то имеем

$$A + B + C + D + E = 1.$$

Пусть $S(n)$ имеет вид (Д.9) — предположение индукции, тогда

$$S(n+1) = A(n+1)^4 + B(n+1)^3 + C(n+1)^2 + D(n+1) + E,$$

с другой стороны

$$S(n+1) = S(n) + (n+1)^3. \qquad (Д.10)$$

Тогда, подставляя в (Д.10) значение $S(n)$ из формулы (Д.9), имеем равенство

$$\begin{aligned} &An^4 + Bn^3 + Cn^2 + Dn + E + (n+1)^3 = \\ &= A(n+1)^4 + B(n+1)^3 + C(n+1)^2 + D(n+1) + E, \end{aligned}$$

которое должно выполняться для всех значений n. Приравнивая коэффициенты при одинаковых степенях n, получаем систему уравнений для определения коэффициентов A, B, C, D.

$$\begin{cases} 4A = 1; \\ 6A + 3B = 3; \\ 4A + 3B + 2C = 3; \\ A + B + C + D = 1. \end{cases}$$

Решая эту систему, получаем $A = \frac{1}{4}$, $B = \frac{1}{2}$, $C = \frac{1}{2}$, $D = 0$. Зная коэффициенты A, B, C, D из уравнения

$$A + B + C + D + E = 1,$$

находим, что $E = 0$, и, подставляя значения коэффициентов в (Д.9) имеем

$$S(n) = \sum_{k=1}^{n} \frac{1}{k^3} = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2 = \frac{n^2(n+1)^2}{4}.$$

Заметим, что в качестве «бесплатного приложения» мы получили любопытное равенство. Поскольку

$$(1 + 2 + \cdots + n) = \frac{n(n+1)}{2}, \quad \text{то} \quad (1^3 + 2^3 + \cdots + n^3) = (1 + 2 + \cdots + n)^2.$$

Метод прибавления нуля. Идея метода состоит в использовании специального приема доказательств, а именно прибавление нуля, записанного своеобразным образом. Наряду с этим используется также тождественное переписывание выражений другим образом и умножение на единицу, которая также записывается своеобразным образом.

Пример Д.5.7. Найти функциональную зависимость $S(n)$ для суммы

$$S(n) = \sum_{k=1}^{n} \frac{1}{k(k+2)}.$$

Решение. Нетрудно вычислить первые два значения $S(n)$

$$S(1) = \frac{1}{3};\ S(2) = \frac{1}{3} + \frac{1}{8} = \frac{11}{24}.$$

Перепишем выражение для этой суммы несколько иначе, а именно

$$S(n) = \sum_{k=1}^{n} \left(\frac{1}{2k} - \frac{1}{2(k+2)} \right),$$

и воспользуемся свойством линейности, тогда получаем

$$S(n) = \sum_{k=1}^{n} \left(\frac{1}{2k} - \frac{1}{2(k+2)} \right) = \frac{1}{2} \sum_{k=1}^{n} \frac{1}{k} - \frac{1}{2} \sum_{k=1}^{n} \frac{1}{k+2}.$$

Введем обозначения для сумм в правой части

$$U(n) = \sum_{k=1}^{n} \frac{1}{k}, \quad V(n) = \sum_{k=1}^{n} \frac{1}{k+2},$$

и исследуем вторую сумму — $V(n)$. Для ее исследования удобно прибавить ноль, записанный в следующем виде — $0 = \frac{1}{1} + \frac{1}{2} - \frac{1}{1} - \frac{1}{2}$, тогда имеем

$$\begin{aligned} V(n) &= \sum_{k=1}^{n} \frac{1}{k+2} = \frac{1}{1} + \frac{1}{2} + \sum_{k=1}^{n} \frac{1}{k+2} - \frac{1}{1} - \frac{1}{2} = \sum_{k=1}^{n+2} \frac{1}{k} - \frac{1}{1} - \frac{1}{2} = \\ &= \sum_{k=1}^{n} \frac{1}{k} + \frac{1}{n+1} + \frac{1}{n+2} - \frac{1}{1} - \frac{1}{2}, \end{aligned}$$

следовательно,

$$V(n) = U(n) + \frac{1}{n+1} + \frac{1}{n+2} - \frac{1}{1} - \frac{1}{2},$$

откуда

$$S(n) = \frac{1}{2}(U(n) - V(n)) = \frac{1}{2} \left(U(n) - U(n) - \frac{1}{n+1} - \frac{1}{n+2} + \frac{1}{1} + \frac{1}{2} \right).$$

Получаем, что при $n \geqslant 3$ сумма $S(n) = \dfrac{3}{4} - \dfrac{1}{2(n+1)} - \dfrac{1}{2(n+2)}$.

Пример Д.5.8. В качестве еще одного примера вычисления суммы методом прибавления нуля, вычислим сумму геометрической прогрессии и докажем формулу (Д.5).

Решение. Пусть имеется сумма $S(n) = \sum\limits_{k=1}^{n} aq^{k-1}$. Преобразуем ее к виду

$$S(n) = \sum_{k=1}^{n} aq^{k-1} = a + \sum_{k=2}^{n} aq^{k-1} = a + q\sum_{k=2}^{n} aq^{k-2} = a + (q\sum_{k=2}^{n} aq^{k-2} + aq^n) - aq^n,$$

мы снова прибавили ноль, записанный в виде $0 = (aq^n - aq^n)$, далее получаем

$$S(n) = a + (q\sum_{k=2}^{n} aq^{k-2} + aq^n) - aq^n = a + q(\sum_{k=2}^{n} aq^{k-2} + aq^{n-1}) - aq^n =$$
$$= a + q(\sum_{k=2}^{n+1} aq^{k-2}) - aq^n = a(1-q^n) + q(\sum_{k=1}^{n} aq^{k-1}).$$

Так как

$$\sum_{k=1}^{n} aq^{k-1} = S(n),$$

то для определения суммы $S(n)$ мы получили уравнение

$$S(n) = a(1-q^n) + qS(n),$$

решая которое, получаем искомый результат

$$S(n) = \sum_{k=1}^{n} aq^{k-1} = a\frac{q^n - 1}{q-1}.$$

Метод дифференцирования непрерывных функций. Использование операции дифференцирования непрерывных функций иногда также оказывается полезным для вычисления сумм. Покажем это на следующем примере. Рассмотрим сумму вида

$$S(n) = \sum_{k=1}^{n} k3^k. \qquad (Д.11)$$

Вначале «усложним» задачу. Вместо данной суммы рассмотрим функцию

$$f(x) = \sum_{k=1}^{n} kx^k,$$

тогда интересующая нас сумма есть просто значение указанной выше функции в точке $x = 3$, т. е.

$$\sum_{k=1}^{n} k3^k = f(3).$$

Проведем некоторые преобразования, которые в конечном итоге позволят найти данную сумму.

$$f(x) = \sum_{k=1}^{n} kx^k = x\sum_{k=1}^{n} kx^{k-1} = x\frac{d}{dx}(\sum_{k=1}^{n} x^k).$$

Сумма $\sum\limits_{k=1}^{n} x^k$ является суммой геометрической прогрессии и может быть легко вычислена

$$\sum_{k=1}^{n} x^k = \sum_{k=1}^{n} x \cdot x^{k-1} = x\frac{x^n - 1}{x - 1} = \frac{x^{n+1} - x}{x - 1}.$$

Вычисляя производную, имеем

$$\frac{d}{dx}(\sum_{k=1}^{n} x^k) = \frac{nx^{n+1} - (n+1)x^n + 1}{(x-1)^2}.$$

Таким образом,

$$f(x) = \sum_{k=1}^{n} kx^k = x\frac{nx^{n+1} - (n+1)x^n + 1}{(x-1)^2}, \qquad (Д.12)$$

и, следовательно, подставляя значение $x = 3$, окончательно получаем

$$\sum_{k=1}^{n} k3^k = f(3) = \frac{3}{4}(n3^{n+1} - (n+1)3^n + 1).$$

Отметим, что полученный результат позволяет найти точные значения сумм вида (Д.11) с разными основаниями степени. Так, например, сумма

$$S(n) = \sum_{k=1}^{n} k2^k = f(2),$$

которая встречается при анализе трудоемкости некоторых алгоритмов, может быть без труда получена на основе формулы (Д.12).

Мы рассмотрели некоторые методы оценки и исчисления конечных сумм. Отметим, что дальнейшим обобщением исчисления конечных сумм является рассмотрение бесконечных сумм вида $\sum\limits_{k=1}^{\infty} a_k$, которые называются рядами, и исследование которых является предметом изучения специальных разделов классического курса математического анализа.

Набор упражнений Д.5.1

Д.5.1.1. Найдите сумму $\sum\limits_{k=1}^{n} (2n - 2k)$.

Д.5.1.2. Найдите сумму $\sum\limits_{k=1}^{n} \frac{1}{k(k+1)}$.

Д.5.1.3. Найдите точную асимптотическую оценку для суммы $\sum\limits_{k=1}^{n} \sqrt{k}$.

Д.5.1.4. Найдите точную асимптотическую оценку для суммы $\sum\limits_{k=1}^{n} \frac{1}{\sqrt{k}}$.

Д.5.1.5. Покажите, что $\sum\limits_{k=1}^{n} \frac{1}{\sqrt{k^3}} = O(1)$.

Д.5.2. Элементы теории рекурсии

Рекурсивно заданные последовательности. Прежде чем перейти к изложению общей теории рекурсии, рассмотрим две задачи. Первая задача, которая может показаться читателю немного странной, состоит в том, что, предположим, нам требуется решить на компьютере следующее уравнение $0,7x = 2$, но наш компьютер таков, что он умеет складывать, вычитать, умножать и сравнивать, но не умеет делить. Предложим алгоритм, который позволит этому странному компьютеру решить данное уравнение (говоря более высоким стилем, мы покажем, что данная задача разрешима в предложенной модели вычислений).

Перепишем уравнение в виде $x = 0,3x + 2$, зададим значение $x_1 = 1$ и рассмотрим последовательность значений x_n, определенных соотношением

$$\begin{cases} x_1 = 1; \\ x_{n+1} = 0,3x_n + 2. \end{cases} \tag{Д.13}$$

Мы получили рекурсивно заданную последовательность, в которой следующее значение определяется на основе предыдущего. Изучим свойства этой последовательности. Обозначим через $r_n = |x_{n+1} - x_n|$. Поскольку по формуле (Д.13) $x_{n+1} = 0,3x_n + 2$, и $x_{n+2} = 0,3x_{n+1} + 2$, то имеем

$$x_{n+2} - x_{n+1} = 0,3 \cdot (x_{n+1} - x_n).$$

Следовательно, $r_{n+1} = 0,3 \cdot r_n$. Покажем, что последовательность x_n ограничена для всех значений n. Так как $x_1 = 1$, то $x_2 = 2,3$, тогда при $n > 2$ имеем

$$|x_n - x_1| = |(x_n - x_{n-1}) + (x_{n-1} - x_{n-2}) + \cdots + (x_3 - x_2) + (x_2 - x_1)|.$$

Воспользовавшись свойством модуля, мы можем записать неравенство

$$|x_n - x_1| \leqslant |x_n - x_{n-1}| + |x_{n-1} - x_{n-2}| + \cdots + |x_3 - x_2| + |x_2 - x_1|,$$

следовательно,

$$|x_n - x_1| \leqslant r_{n-1} + r_{n-2} + \cdots + r_2 + r_1,$$

а поскольку $r_{n+1} = 0,3 \cdot r_n$, то

$$|x_n - x_1| \leqslant 0,3^{n-2} r_1 + 0,3^{n-3} r_1 + \cdots + 0,3 r_1 + r_1.$$

Используя формулу суммы геометрической прогрессии, получаем

$$|x_n - x_1| \leqslant r_1 \frac{1 - 0,3^{n-1}}{1 - 0,3},$$

таким образом доказана ограниченность последовательности x_n.

Далее докажем, что данная последовательность имеет предел. Для этого обозначим через $d_{mn} = |x_n - x_m|$, и для определенности будем полагать, что $n > m$. Аналогично только что проделанным рассуждениям получаем следующее неравенство

$$d_{mn} \leqslant r_{n-1} + r_{n-2} + \cdots + r_{m+1} + r_m,$$

следовательно,

$$d_{mn} \leqslant 0,3^{n-2}r_1 + 0,3^{n-3}r_1 + \cdots + 0,3^m r_1 + 0,3^{m-1}r_1 = 0,3^{m-1}r_1\frac{1-0,3^{n-m}}{1-0,3}.$$

Тогда для любого $\varepsilon > 0$, найдется число $n_0 = n_0(\varepsilon)$, такое, что при любых $n > n_0$, $m > n_0$ будет выполнено $d_{mn} < \varepsilon$. Из свойства полноты множества действительных чисел следует, что последовательность x_n имеет предел при n стремящемся к бесконечности.

Обозначим $\lim\limits_{n\to\infty} x_n = z$, и перейдем к пределу в соотношении $x_{n+1} = 0,3x_n + 2$. Имеем $\lim\limits_{n\to\infty} x_{n+1} = \lim\limits_{n\to\infty}(0,3x_n + 2)$, тогда получаем $z = 0,3z + 2$, следовательно, значение предела и является корнем уравнения. Таким образом, предложенный алгоритм позволяет решать уравнение с любой заданной точностью, не используя операцию деления. Отметим, что в соотношении (Д.13) мы использовали определение следующего члена последовательности через его предыдущее значение.

Вторая задача. Необходимо найти $\sqrt{a}$ $(a > 0)$, если разрешается использовать только четыре арифметические операции — сложение, вычитание, умножение, деление и операцию сравнения. Значение $\sqrt{a}$ является положительным корнем квадратного уравнения $x^2 = a$. Преобразуем это уравнение в уравнение $2x^2 = x^2 + a$, которое представим в виде

$$x = \frac{1}{2}\left(x + \frac{a}{x}\right).$$

Рассмотрим последовательность x_n, определяемую соотношениями:

$$\begin{cases} x_1 = 1; \\ x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n}). \end{cases} \qquad (Д.14)$$

Очевидно, что данная последовательность принимает только положительные значения. Для того чтобы исследовать свойства данной последовательности, нам понадобится некоторая информация о свойствах функции

$$y(x) = \frac{1}{2}\left(x + \frac{a}{x}\right),$$

более точно, нас интересует ее производная

$$\frac{dy}{dx} = \frac{1}{2}\left(1 - \frac{a}{x^2}\right).$$

При положительных значениях аргумента производная равна нулю при $x = \sqrt{a}$. Функция $y(x)$ убывает на интервале $(0; \sqrt{a})$ и возрастает на интервале $(\sqrt{a}; \infty)$. Тогда для положительных значений аргумента точка $x = \sqrt{a}$ является точкой минимума, и, следовательно, при всех неотрицательных его значениях имеем

$$y \geqslant \frac{1}{2}\left(\sqrt{a} + \frac{a}{\sqrt{a}}\right) = \sqrt{a}.$$

Значит, все члены последовательности, кроме может быть первого, больше либо равны $\sqrt{a}$: $x_n \geqslant \sqrt{a}$ при $n > 1$. Далее покажем, что все члены последовательности, начиная со второго, не возрастают, т. е. $x_{n+1} \leqslant x_n$ при $n \geqslant 2$. Нетрудно заметить, что при $x \geqslant \sqrt{a}$ выполнено

$$\frac{1}{2}\left(x + \frac{a}{x}\right) \leqslant \frac{1}{2}\left(x + x\right) = x.$$

Следовательно, члены последовательности не возрастают, начиная со второго, но из курса математического анализа известно, что невозрастающая ограниченная последовательность имеет предел. Обозначим $\lim\limits_{n\to\infty} x_n = z$ и, переходя к пределу в соотношении

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right),$$

получаем, что значение предела является положительным корнем уравнения

$$z = \frac{1}{2}\left(z + \frac{a}{z}\right),$$

который равен $z = \sqrt{a}$. Записанная последовательность, в которой опять следующее значение определено по некоторой формуле через предыдущее, может быть легко преобразована в алгоритм, который позволяет вычислять квадратные корни с нужной степенью точности.

Отметим, что предложенные методы основаны на более широком методе, применяемом в математике, и известном под названием «метод сжатых отображений». Сам математический метод опирается на идею построения рекуррентного соотношения. Далее мы приведем некоторые определения, поясняющие смысл понятия рекурсии.

Рекурсивно заданные функции. Непосредственное задание функции в виде формулы, в большинстве случаев, является предпочтительным. Однако это удается не всегда, довольно часто функцию приходится задавать при помощи «рекурсивного метода», суть которого состоит в следующем.

Предположим, что требуется определить значения некоторой функции $T(n)$, где аргумент n является целым числом. (Обычно предполагается, что $n \geqslant 0$ или $n \geqslant 1$).

Рекурсивное задание функции включает два этапа.

Первый этап. Функция $T(n)$ задается непосредственно в виде числовых значений для некоторого множества начальных значений n: $1 \leqslant n \leqslant m$.

Второй этап. Задается метод или формула, которые позволяют, зная все значения функции $T(n)$ при $n \leqslant k$ $(k \geqslant m)$, вычислить ее значение при $n = k + 1$, т. е. найти $T(k + 1)$ — мы получаем рекуррентные соотношения, описывающие рекурсивно заданную функцию.

Полученная запись этих двух этапов, обычно объединенная фигурной скобкой носит название *рекуррентного соотношения.*

Отметим, что сумма $S(n) = \sum\limits_{k=1}^{n} a_k$ также может быть задана как рекурсивная функция следующими рекуррентными соотношениями

$$\begin{cases} S(0) = 0; \\ S(k) = S(k-1) + a_k, \quad k \geqslant 1. \end{cases} \tag{Д.15}$$

Приведем еще один пример. Рассмотрим целочисленную функцию $f(n) = n!$, называемую факториалом. Непосредственно она определяется в виде

$$n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n,$$

но современные компьютеры пока многоточие не воспринимают. Поэтому в целях программирования приходится задавать эту функцию несколько иначе. Прибегнем к рекурсивному заданию функции

$$\begin{cases} f(0) = 1; \\ f(k) = k \cdot f(k-1), \quad k \geqslant 1. \end{cases} \tag{Д.16}$$

очевидно, что вычисление по предложенным рекуррентным соотношениям может быть легко реализовано на компьютере.

Таким образом, с помощью *рекуррентных соотношений* можно задавать как числовые последовательности (см. формулы (Д.13), (Д.14)), так и функции (см. формулы (Д.15), (Д.16)). Последовательности или функции, определенные с помощью рекуррентных соотношений, называются *рекурсивно заданные* или *рекурсивные*. Отметим, что рекурсивно заданная последовательность может быть представлена в виде функции целочисленного аргумента $f(k) = a_k$.

Классификация рекурсивно заданных последовательностей. При классификации рекурсивно заданных последовательностей используется следующая терминология. Последовательность $x_1, x_2, ..., x_n$ называется последовательностью с *частичной предысторией*, если следующий член последовательности зависит от некоторого фиксированного числа предыдущих членов. Она может быть выражена соотношением

$$x_n = f(x_{n-1}, x_{n-2}, ..., x_{n-i}), \quad \text{где} \quad n > i.$$

Если x_n зависит от всех предшествующих членов, то говорят, что рекурсивно заданная последовательность имеет *полную предысторию*.

Цели, которые ставятся при исследовании рекурсивно заданных последовательностей или функций, совпадают с теми, которые возникали при исследовании конечных сумм. В идеале, конечно, желательно найти явное выражение для рекурсивно заданной функции или общего члена рекурсивной последовательности. Однако наше идеальное желание далеко не всегда, а точнее, крайне редко, совпадает с нашими возможностями. Поэтому чаще приходится ограничиваться изучением асимптотики поведения рекурсивной последовательности или функции при больших значениях аргумента. Тем не менее, существует важный класс рекуррентных соотношений, для которых удается найти явное выражение для соответствующей рекурсивно заданной функции или члена последовательности. Задача определения явного выражения для рекурсивно заданной функции или нахождения общего члена рекурсивной последовательности часто называется *задачей решения рекуррентного соотношения*.

Линейные однородные рекуррентные соотношения с постоянными коэффициентами с частичной предысторией. Линейным однородным рекуррентным соотношением с постоянными коэффициентами порядка p называется соот-

ношение вида

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_p a_{n-p}, \quad \text{где} \quad c_p \neq 0.$$

Решения данного рекуррентного соотношения обладают свойством линейности, которое означает следующее. Пусть последовательности $b_1, b_2, ..., b_n$ и $c_1, c_2, ..., c_n$ являются решениями данного рекуррентного соотношения. Для краткости обозначим их через $B = \{b_1, b_2, ..., b_n\}$, и $C = \{c_1, c_2, ..., c_n\}$, тогда последовательность $D = b \cdot B + c \cdot C$, (где b и c произвольные постоянные величины), являющаяся линейной комбинацией последовательностей B и C, также будет решением данного рекуррентного соотношения.

Решение исследуемых рекуррентных соотношений во многом аналогично решению линейных однородных дифференциальных уравнений с постоянными коэффициентами порядка p. Чтобы дать качественную картину, рассмотрим случай, когда $p = 2$. Представим эту аналогию. Однородное линейное дифференциальное уравнение второго порядка имеет вид

$$\frac{d^2y}{dx^2} + d_1 \frac{dy}{dx} + d_2 y = 0. \qquad (Д.17)$$

Линейное однородное рекуррентное соотношение с постоянными коэффициентами порядка 2 записывается в виде

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} \quad \Rightarrow \quad a_n - c_1 a_{n-1} - c_2 a_{n-2} = 0. \qquad (Д.18)$$

Решение линейного однородного дифференциального уравнения ищем в виде $y = e^{\lambda x}$. Подставляя в уравнение (Д.17), получаем

$$\lambda^2 e^{\lambda x} + d_1 \lambda e^{\lambda x} + d_2 e^{\lambda x} = 0; \quad \lambda^2 + d_1 \lambda + d_2 = 0.$$

Для определения значения λ получено уравнение $\lambda^2 + d_1 \lambda + d_2 = 0$, которое называется характеристическим уравнением.

Решение линейного однородного рекуррентного соотношения с постоянными коэффициентами порядка 2 ищем в виде $a_n = r^n$. Подставляя в уравнение (Д.18), получаем

$$r^n = c_1 r^{n-1} + c_2 r^{n-2}; \quad r^2 - c_1 r - c_2 = 0.$$

Для определения значения r получено уравнение $r^2 - c_1 r - c_2 = 0$, которое также называется характеристическим уравнением.

Возможны три случая, при рассмотрении которых мы через A и B будем обозначать некоторые произвольные постоянные.

Случай I. Если характеристическое уравнение $\lambda^2 + d_1 \lambda + d_2 = 0$ имеет два различных действительных корня $\lambda = k_1$, $\lambda = k_2$, то общее решение линейного однородного дифференциального уравнения представляется в виде $y = Ae^{k_1 x} + Be^{k_2 x}$. Если характеристическое уравнение $r^2 - c_1 r - c_2 = 0$ имеет два различных действительных корня $r = r_1$, $r = r_2$, то общее решение линейного однородного рекуррентного соотношения представляется в виде $a_n = Ar_1^n + Br_2^n$.

Случай II. Если характеристическое уравнение $\lambda^2 + d_1 \lambda + d_2 = 0$ имеет два равных действительных корня $\lambda_1 = \lambda_2 = k$, то общее решение линейного

однородного дифференциального уравнения представляется в виде $y = Ae^{kx} + Bxe^{kx}$. Если характеристическое уравнение $r^2 - c_1 r - c_2 = 0$ имеет два равных действительных корня $r_1 = r_2 = a$, то общее решение линейного однородного рекуррентного соотношения представляется в виде $a_n = Aa^n + Bna^n$.

Поясним этот факт. Очевидно, что последовательность $a_n = a^n$ является решением рекуррентного соотношения, поскольку число $r = a$ является корнем характеристического уравнения. Рассмотрим последовательность $a_n = na^n$. Подставим эту последовательность в рекуррентное соотношение (Д.18)

$$na^n = c_1(n-1)a^{n-1} + c_2(n-2)a^{n-2},$$

и преобразуем данное соотношение к виду

$$na^{n-2}(a^2 - c_1 a - c_2) + a^{n-1}(c_1 a + 2c_2) = 0.$$

Поскольку a — корень характеристического уравнения, то $(a^2 - c_1 a - c_2) = 0$, а т. к. a является корнем кратности 2, то $(c_1 a + 2c_2) = 0$. Таким образом, последовательность $a_n = na^n$ действительно является решением рекуррентного соотношения.

Случай III. Если характеристическое уравнение $\lambda^2 + d_1\lambda + d_2 = 0$ имеет два комплексно сопряженных корня $\lambda = \alpha \pm \beta i$, то общее решение линейного однородного дифференциального уравнения представляется в виде $y = Ae^{\alpha x}\cos\beta x + Be^{\alpha x}\sin\beta x$.

Если характеристическое уравнение $r^2 - c_1 r - c_2 = 0$ имеет два комплексно сопряженных корня $r = \alpha \pm \beta i$, то общее решение линейного однородного рекуррентного соотношения представляется в виде $a_n = A \cdot (\alpha + \beta i)^n + B \cdot (\alpha - \beta i)^n$.

Воспользуемся тем, что мы рассматриваем уравнение с действительными коэффициентами и перепишем полученную формулу в более удобном виде. Очевидно, что если некоторая комплексная последовательность является решением линейного однородного рекуррентного соотношения с действительными коэффициентами, то решениями такого рекуррентного соотношения будут являться действительная и мнимая части последовательности. Комплексное число $\alpha + \beta i$ может быть записано в тригонометрической форме

$$\alpha + \beta\, i = \rho(\cos\varphi + i\sin\varphi),$$

где модуль комплексного числа ρ равен $\rho = \sqrt{\alpha^2 + \beta^2}$, а аргумент φ определяется из условия

$$\cos\varphi = \frac{\alpha}{\rho}, \qquad \sin\varphi = \frac{\beta}{\rho} \qquad (0 \leqslant \varphi < 2\pi).$$

Согласно формуле Муавра $(\rho(\cos\varphi + i\sin\varphi))^n = \rho^n(\cos n\varphi + i\sin n\varphi)$. Поскольку и действительная и мнимая части последовательности являются решениями рекуррентного соотношения, то последовательности $b_n = \rho^n\cos n\varphi$ и $c_n = \rho^n\sin n\varphi$ будут решениями рекуррентного соотношения. Тогда общее решение может быть записано в виде $a_n = A\rho^n\cos n\varphi + B\rho^n\sin n\varphi$.

Примеры решения рекуррентных соотношений.

Пример Д.5.9. Найти решение для последовательности (ряда) Фибоначчи $fb(n)$, заданной в виде рекурсивной функции соотношениями

$$\begin{cases} fb(1) = 1,\ fb(2) = 1; \\ fb(n) = fb(n-1) + fb(n-2)\,, \quad n \geqslant 3. \end{cases} \tag{Д.19}$$

Решение. Заметим, что мы определили рекурсивно заданную функцию для числовой последовательности. Характеристическое уравнение имеет вид $r^2 = r + 1$ или $r^2 - r - 1 = 0$, решая которое, получаем

$$r_{1,2} = \frac{1 \pm \sqrt{5}}{2},$$

тогда

$$fb(n) = c \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n + d \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Определим значения постоянных c и d. Поскольку $fb(1) = 1$, то

$$c \cdot \left(\frac{1+\sqrt{5}}{2}\right) + d \cdot \left(\frac{1-\sqrt{5}}{2}\right) = 1,$$

а поскольку в силу рекуррентного соотношения (Д.19) $fb(2) = 1$, то

$$c \cdot \left(\frac{1+\sqrt{5}}{2}\right)^2 + d \cdot \left(\frac{1-\sqrt{5}}{2}\right)^2 = 1.$$

Мы получили систему двух линейных уравнений с двумя неизвестными для определения c и d. Решая эту систему, получаем $c = \frac{1}{\sqrt{5}}$, $d = -\frac{1}{\sqrt{5}}$, таким образом

$$fb(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Пример Д.5.10. Найти функциональную зависимость для общего члена рекурсивно заданной последовательности (решить рекуррентное соотношение)

$$\begin{cases} a_1 = 7,\ a_2 = 19; \\ a_n = 4a_{n-1} - 3a_{n-2},\ n \geqslant 2\,. \end{cases}$$

Решение. Характеристическое уравнение имеет вид $r^2 = 4r - 3$ или $r^2 - 4r + 3 = 0$, решая его получаем $r_1 = 1$, $r_2 = 3$, тогда общее решение имеет вид

$$a_n = c + d \cdot 3^n.$$

Поскольку $a_1 = 7$, то $c + 3d = 7$, а поскольку $a_2 = 19$, то $c + 9d = 19$. Решая полученную систему уравнений, получаем $c = 1$, $d = 2$, что приводит к

$$a_n = 1 + 2 \cdot 3^n.$$

Пример Д.5.11. Решить рекуррентное соотношение

$$\begin{cases} a_1 = 4,\ a_2 = 12; \\ a_n = 4a_{n-1} - 4a_{n-2},\ n \geqslant 2. \end{cases}$$

Решение. Характеристическое уравнение имеет вид $r^2 = 4r - 4$ или $r^2 - 4r + 4 = 0$, решая уравнение, получаем $r_1 = r_2 = 2$. Поскольку характеристическое уравнение имеет два равных действительных корня, то общее решение имеет вид

$$a_n = c \cdot 2^n + d \cdot n \cdot 2^n.$$

Поскольку $a_1 = 4$, то $2c + 2d = 4$, т. к. в силу рекуррентного соотношения $a_2 = 12$, то $4c + 8d = 12$. Решая полученную систему уравнений, получаем $c = 1$, $d = 1$, тогда

$$a_n = 2^n + n \cdot 2^n.$$

Пример Д.5.12. Решить рекуррентное соотношение

$$\begin{cases} a_1 = 1,\ a_2 = -2; \\ a_n = 2a_{n-1} - 4a_{n-2},\ n \geqslant 2. \end{cases}$$

Решение. Характеристическое уравнение имеет вид $r^2 = 2r - 4$ или $r^2 - 2r + 4 = 0$. Решая уравнение, получаем $r_{1,2} = 1 \pm \sqrt{3}i$. Поскольку характеристическое уравнение имеет два комплексно сопряженных корня, то представим их в тригонометрической форме. Модуль комплексного числа $\rho = \sqrt{1+3} = 2$. Для определения аргумента имеем

$$\cos\varphi = \frac{1}{2}, \quad \sin\varphi = \frac{\sqrt{3}}{2}; \quad \varphi = \frac{\pi}{3},$$

тогда

$$a_n = c \cdot 2^n \cos\left(\frac{n\pi}{3}\right) + d \cdot 2^n \sin\left(\frac{n\pi}{3}\right).$$

Используя явно заданные значения первых двух членов последовательности, получаем систему уравнений

$$\begin{cases} c + \sqrt{3}d = 1; \\ -2c + 2\sqrt{3}d = -2, \end{cases}$$

решая которую, получаем $c = 1$, $d = 0$, тогда

$$a_n = 2^n \cos\left(\frac{n\pi}{3}\right).$$

Обобщение на случай линейного однородного рекуррентного соотношения с постоянными коэффициентами порядка *p*. Для рекуррентного соотношения

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots c_p a_{n-p}, \text{ где} \quad c_p \neq 0,$$

характеристическое уравнение имеет вид

$$r^p = c_1 r^{p-1} + c_2 r^{p-2} + \cdots + c_p,$$

или

$$r^p - c_1 r^{p-1} - c_2 r^{p-2} - \cdots - c_p = 0.$$

Если действительное число $r = b$ является корнем характеристического уравнения кратности m, то оно порождает m решений вида

$$a_n = b^n; \quad a_n = nb^n; a_n = n^2 b^n; \quad \ldots; \quad a_n = n^{m-1} b^n.$$

Если каждое число из пары комплексно сопряженных чисел $r = \rho(\cos\varphi \pm \pm i \sin\varphi)$ является корнем характеристического уравнения кратности m, то они порождают $2m$ решений вида:

$$\begin{cases} a_n = \rho^n \cos n\varphi\,; & a_n = \rho^n \sin sn\varphi, \\ a_n = n\rho^n \cos n\varphi\,; & a_n = n\rho^n \sin n\varphi, \\ a_n = n^{m-1}\rho^n \cos n\varphi\,; & a_n = n^{m-1}\rho^n \sin n\varphi, \end{cases}$$

и в этом случае общее решение может быть представлено в виде суммы указанных решений, умноженных на произвольные постоянные.

Пример Д.5.13. Решить рекуррентное соотношение

$$\begin{cases} a_1 = 3;\ a_2 = 10;\ a_1 = 19; \\ a_n = 4a_{n-1} - 5a_{n-2} + 2a_{n-2},\ n \geqslant 3. \end{cases}$$

Решение. Характеристическое уравнение имеет вид $r^3 = 4r^2 - 5r + 2$ или $r^3 - 4r^2 + 5r - 2 = 0$. Данное характеристическое уравнение имеет следующие корни $r_1 = r_2 = 1$, $r_3 = 2$. Поскольку $r = 1$ является корнем кратности 2, то этот корень порождает два решения: $a_n = 1$; $a_n = n$, а корень $r = 2$ порождает решение $a_n = 2^n$. Тогда общее решение рекуррентного соотношения имеет вид $a_n = b + cn + d2^n$. Для определения значений b, c, d имеем систему уравнений

$$\begin{cases} b + c + 2d = 3, \\ b + 2c + 4d = 10, \\ b + 3c + 8d = 19, \end{cases}$$

решая которую, получаем $b = 0,\ c = 1,\ d = 1$, тогда

$$a_n = n + 2^n.$$

Линейные неоднородные рекуррентные соотношения с постоянными коэффициентами порядка p. Линейным неоднородным рекуррентным соотношением с постоянными коэффициентами порядка p называется соотношение вида

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_p a_{n-p} + g(n), \qquad (Д.20)$$

где $c_p \neq 0$, а $g(n)$ — известная функция.

Метод решения таких соотношений состоит в следующем. На первом этапе удаляется $g(n)$, (то есть полагается равной нулю) и находится общее решение соответствующего однородного соотношения. Далее ищется какое-либо частное решение неоднородного уравнения. Тогда общее решение неоднородного соотношения представляется как сумма частного решения и общего решения соответствующего однородного соотношения. Поскольку существует универсальный

метод решения однородных соотношений, то проблема решения неоднородного состоит в том, чтобы «угадать» любое частное решение неоднородного соотношения.

Для некоторых частных видов функции $g(n)$ существует универсальный прием отыскания частных решений. Об этом приеме мы скажем ниже, а пока отметим один полезный факт. Если функция $g(n)$ представима в виде суммы двух слагаемых $g(n) = g_1(n) + g_2(n)$ и каким-то образом удалось найти частные решения соотношений

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_p a_{n-p} + g_1(n),$$
$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_p a_{n-p} + g_2(n),$$

то сумма таких частных решений будет также частным решением соотношения (Д.20). Касаясь вопроса отыскания частных решений, отметим следующее. Если функция $g(n)$ имеет вид $g(n) = P_m(n) \cdot b^n$, где $P_m(n)$ многочлен степени m, то частное решение может быть найдено в виде $a_n = n^s \cdot Q_m(n) \cdot b^n$, где $Q_m(n)$ — многочлен той же степени что и $P_m(n)$, s — кратность корня $r = b$ в характеристическом уравнении. Если $r = b$ не является корнем характеристического уравнения, то $s = 0$. Отметим, что многочлен $P_m(n)$ представим в виде $P_m(n) = P_m(n) \cdot 1^n$.

Пример Д.5.14. Найти общее решение неоднородного рекуррентного соотношения

$$a_n = 4a_{n-1} - 3a_{n-2} + 5 \cdot 2^n + (5n + 4).$$

Решение. Соответствующее однородное соотношение исследовалось ранее, и были найдены корни характеристического уравнения $r_1 = 1$, $r_2 = 3$. Общее решение однородного соотношения имеет вид $a_n = c + d \cdot 3^n$, где c и d произвольные постоянные. В данном примере функция $g(n) = 5 \cdot 2^n + (5n + 4)$. Она состоит из двух слагаемых $g_1(n) = 5 \cdot 2^n$ и $g_2(n) = (5n + 4)$.

Вначале найдем частное решение соотношения $a_n = 4a_{n-1} - 3a_{n-2} + 5 \cdot 2^n$. Имеем функцию $g_1(n)$ вида $g_1(n) = P_m(n) \cdot b^n$, где $P_m(n) = 5$ — многочлен нулевой степени и значит $m = 0$; $b = 2$ и поскольку число $b = 2$ не является корнем характеристического уравнения, то можно положить его кратность $s = 0$. Тогда частное решение соотношения ищем в виде $a_n = n^0 \cdot Q_0(n) \cdot 2^n = A \cdot 2^n$, где A — неизвестная величина, которую мы определим из рекуррентного соотношения. Подставляя данное выражение в рекуррентное соотношение, получаем

$$A \cdot 2^n = 4A \cdot 2^{n-1} - 3A \cdot 2^{n-2} + 5 \cdot 2^n \quad \Rightarrow \quad -\frac{1}{4}A = 5; \quad A = -20,$$

следовательно, частное решение имеет вид $a_n = -20 \cdot 2^n$.

Далее найдем частное решение соотношения $a_n = 4a_{n-1} - 3a_{n-2} + (5n + 4)$. Имеем функцию $g_2(n)$ вида $g_2(n) = P_m(n) \cdot b^n$, где $P_m(n) = 5n + 4$ — многочлен первой степени и значит $m = 1$; $b = 1$, и поскольку число $b = 1$ является корнем характеристического уравнения кратности 1, то можно положить $s = 1$. Тогда частное решение соотношения ищем в виде $a_n = n^1 \cdot Q_1(n) \cdot 1^n = n(An + B)$, где

A и B — неизвестные величины, которые мы определим из рекуррентного соотношения. Подставляя данное выражение в рекуррентное соотношение, получаем

$$n(An+B) = 4(n-1)(A(n-1)+B) - 3(n-2)(A(n-2)+B) + (5n+4).$$

Приравнивая коэффициенты при одинаковых степенях n, получаем систему уравнений для определения A и B.

$$\begin{cases} 4A + 5 = 0; \\ -8A + 2B + 4 = 0, \end{cases}$$

решая эту систему, получаем $A = -\frac{5}{4}$, $B = -7$, тогда частное решение рекуррентного соотношения имеет вид

$$a_n = -20 \cdot 2^n + n(-\frac{5}{4}n - 7).$$

Общее решение соответственно равно

$$a_n = -20 \cdot 2^n + n(-\frac{5}{4}n - 7) + c + d \cdot 3^n,$$

где c и d — произвольные постоянные.

Рекуррентные соотношения первого порядка с переменными коэффициентами. Рекуррентным соотношением первого порядка с переменными коэффициентами называется соотношение вида

$$a_n = b(n) \cdot a_{n-1} + c(n),\ n \geqslant 1,$$

где значение a_0 предполагается заранее известной величиной.

Идея решения данного соотношения состоит во введении суммирующего множителя $F(n)$, равного

$$F(n) = \frac{1}{\prod\limits_{k=1}^{n} b(k)}.$$

Умножим обе части соотношения на суммирующий множитель. Получим

$$\frac{a_n}{\prod\limits_{k=1}^{n} b(k)} = \frac{a_{n-1}b(n)}{\prod\limits_{k=1}^{n} b(k)} + \frac{c(n)}{\prod\limits_{k=1}^{n} b(k)}.$$

Обозначим $y_n = b(n+1) \cdot F(n+1) \cdot a_n$, тогда полученное соотношение может быть переписано в виде $y_n = y_{n-1} + F(n) \cdot c(n)$. Это соотношение решается простым суммированием

$$y_n = y_0 + \sum_{k=1}^{n} F(k) \cdot c(k), \quad \text{тогда} \quad a_n = \frac{a_0 + \sum\limits_{k=1}^{n} F(k) \cdot c(k)}{b(n+1) \cdot F(n+1)}.$$

Поскольку проблема вычисления произведений следующим образом сводится к проблеме исчисления сумм

$$\prod_{k=1}^{n} b(k) = e^{\ln \prod\limits_{k=1}^{n} b(k)} = e^{\sum\limits_{k=1}^{n} \ln b(k)},$$

то и исследование таких рекуррентных соотношений сводится к вопросу исчисления сумм.

Пример Д.5.15. Найти решение рекуррентного соотношения

$$\begin{cases} a_0 = 0; \\ a_n = \frac{n+1}{n} a_{n-1} + 2, \ n \geqslant 1. \end{cases}$$

Решение. Суммирующий множитель имеет вид

$$F(n) = \frac{1}{\prod\limits_{k=1}^{n} \frac{k+1}{k}} = \frac{1}{n+1}.$$

Обозначим

$$y_n = b(n+1) \cdot F(n+1) \cdot a_n = \frac{n+1}{n+2}(n+2) \cdot a_n = (n+1) \cdot a_n.$$

Для определения y_n получаем соотношение

$$\begin{cases} y_0 = 0; \\ y_n = y_{n-1} + \frac{2}{n+1}, \end{cases}$$

откуда

$$y_n = \sum_{k=1}^{n} \frac{2}{k+1}, \quad a_n = 2(n+1) \sum_{k=1}^{n} \frac{1}{k+1}.$$

Методы асимптотической оценки функций, заданных рекуррентными соотношениями. В дальнейшем будем рассматривать рекурсивно заданную функцию целого положительного аргумента, обозначая ее через $T(n)$.

Один из приемов, который используется для построения подобных оценок — это построение такой функции $f(n)$, которая мажорировала бы $T(n)$ для всех значений n, т. е. при всех $n \geqslant 1$ должно выполняться $T(n) \leqslant f(n)$. Иногда для построения таких функций используется следующий прием. Задается вид функции $f(n)$, в предположении, что она зависит от некоторых параметров, а значения параметров уточняются в процессе доказательства.

Проиллюстрируем этот прием на примере.

Пример Д.5.16. Оценить функцию, заданную рекуррентным соотношением

$$\begin{cases} T(1) = c_1; \\ T(n) = 2T([\frac{n}{2}]) + c_2 n, \ n > 1, \end{cases} \tag{Д.21}$$

напомним, что через $[x]$ обозначается целая часть аргумента.

Решение. Будем искать $f(n)$ в виде $f(n) = a \cdot n \cdot \log_2 n + b$, где a и b пока неизвестные параметры. Воспользуемся методом математической индукции. При $n = 1$ оценка работает, если положить $b \geqslant c_1$. В соответствии с методом математической индукции полагаем, что для всех $k < n$ выполняется неравенство

$$T(k) \leqslant a \cdot k \cdot \log_2 k + b.$$

Полагая $k = [\frac{n}{2}]$, и подставляя в (Д.21) получаем

$$T(n) = 2T\left(\left[\frac{n}{2}\right]\right) + c_2 n \leqslant 2\left(a \cdot k \cdot \log_2 k + b\right) + c_2 n \leqslant 2\left(a\frac{n}{2}\log_2 \frac{n}{2} + b\right) + c_2 n =$$
$$= a \cdot n \cdot \log_2 n - a \cdot n + c_2 n + 2b \leqslant a \cdot n \cdot \log_2 n + b.$$

Последнее неравенство получено в предположении, что $a \geqslant c_2 + b$. Таким образом, приведенная оценка будет справедлива при условиях, что $b \geqslant c_1$ и $a \geqslant c_2 + b$. Можно положить $b = c_1$, $a = c_2 + c_1$. Тогда можно сделать вывод, что при всех $n \geqslant 1$ имеет место оценка

$$T(n) \leqslant (c_1 + c_2) \cdot n \cdot \log_2 n + c_1.$$

Это означает, что $T(n) = O(n \cdot \log_2 n)$. Отметим, что при приведенном доказательстве, мы *не можем утверждать*, что полученная оценка является точной асимптотической оценкой. Мы лишь доказали тот факт, что функция $T(n)$ растет не быстрее чем $f(n) = n \cdot \log_2 n$. И уж если быть совсем строгим, то полученная оценка должна быть записана в виде

$$T(n) \leqslant (c_1 + c_2) \cdot n^* \log_2 n^* + c_1,$$

где $n^* = n$, если n четное число, и $n^* = n + 1$ в противном случае. Используя операцию получения остатка от деления, n^* можно записать в виде

$$n^* = n + n \mod 2.$$

Метод подстановки. Другой прием — это оценка рекуррентного соотношения методом подстановки, который мы продемонстрируем на следующем примере.

Пример Д.5.17. Оценить рекурсивно заданную функцию $T(n) = 3T([\frac{n}{4}]) + n$.

Решение. Выберем целое число m, такое что $4^{m-1} < n \leqslant 4^m$, и подставляя в правую часть соотношения получаем

$$T(n) \leqslant 3T(4^{m-1}) + 4^m,$$

но в свою очередь $T(4^{m-1}) = 3T(4^{m-2}) + 4^{m-1}$, и подставляя полученный результат в первое неравенство, получаем

$$T(n) \leqslant 3\left(3T(4^{m-2}) + 4^{m-1}\right) + 4^m = 3^2 T\left(4^{m-2}\right) + 4^m\left(1 + \frac{3}{4}\right).$$

Поскольку

$$T(4^{m-2}) = 3T(4^{m-3}) + 4^{m-2},$$

то повторим подстановку

$$T(n) \leqslant 3^3 T(4^{m-3}) + 4^m \left(1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2\right),$$

и продолжая этот процесс, мы получим следующее неравенство

$$T(n) \leqslant 3^m T(1) + 4^m \left(1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2 + \cdots + \left(\frac{3}{4}\right)^{m-1}\right).$$

Последняя сумма в круглых скобках является суммой геометрической прогрессии со знаменателем $q = 3/4$, и ее значение не превышает суммы аналогичной бесконечной геометрической прогрессии, которая равна 4, тогда

$$T(n) \leqslant 3^m T(1) + 4^{m+1}.$$

На основании выбора числа m имеем $m \leqslant \log_4 n + 1$, и

$$T(n) \leqslant 3^{\log_4 n + 1} \cdot T(1) + 16n,$$

но так как $3^{\log_4 n} = n^{\log_4 3} = o(n)$, то окончательно $T(n) = O(n)$.

Основная теорема об асимптотических оценках рекурсивно заданных функций. Следующая теорема, доказанная в 1980 г. Дж. Бентли, Д. Хакен и Дж. Саксом [20] является достаточно мощным средством асимптотической оценки функциональных рекуррентных соотношений определенного вида. Такого рода рекурсивно заданные функции получаются при оценке вычислительной сложности рекурсивных алгоритмов, основанных на методе декомпозиции задачи. В частности, рекуррентные соотношения из примера Д.5.16 соответствуют рекурсивному алгоритму сортировки, разработанному Дж. Фон Нейманом — алгоритму сортировки слиянием.

Теорема. (J. L. Bentley, Dorothea Haken, J. B. Saxe, 1980) Пусть $a \geqslant 1$ и $b > 1$ — константы, $f(n)$ — известная функция, $T(n)$ определено при неотрицательных значениях n формулой

$$T(n) = aT\left(\left[\frac{n}{b}\right]\right) + f(n),$$

тогда:
1) если $f(n) = O(n^{\log_b a - \varepsilon})$ для некоторого $\varepsilon > 0$, то

$$T(n) = \Theta(n^{\log_b a});$$

2) если $f(n) = \Theta(n^{\log_b a})$, то

$$T(n) = \Theta(n^{\log_b a} \log_b n);$$

3) если найдется $c > 0$ и $\varepsilon > 0$, что при достаточно больших n выполнено $f(n) > cn^{\log_b a + \varepsilon}$ и найдется положительная константа $d < 1$, такая что, при достаточно больших n выполнено условие

$$a \cdot f(\frac{n}{b}) \leqslant d \cdot f(n), \quad \text{то} \quad T(n) = \Theta(f(n)).$$

Рассмотрим примеры оценки рекурсивно заданных функций с применением данной теоремы.

Пример Д.5.18. Получить асимптотическую оценку функции $T(n) = 8T([\frac{n}{4}])+n$.

Решение. В этом случае $a = 8$, $b = 4$, $f(n) = n$, а

$$n^{\log_b a} = n^{\log_4 8} = n^{\frac{3}{2}} = n\sqrt{n}.$$

Поскольку $f(n) = O(n\sqrt{n})$ для $\varepsilon = 1/2$, то, применяя первое утверждение теоремы, делаем вывод, что $T(n) = \Theta(n\sqrt{n})$.

Пример Д.5.19. Получить асимптотическую оценку функции $T(n) = T([\frac{3n}{4}])+5$.

Решение. Здесь $a = 1$, $b = 4/3$, $f(n) = 5$, а

$$n^{\log_b a} = n^{\log_{4/3} 1} = n^0 = \Theta(1),$$

поэтому воспользуемся вторым случаем теоремы. Поскольку

$$f(n) = 5 = \Theta(n^{\log_b a}) = \Theta(1),$$

то получаем оценку $T(n) = \Theta(\log_{4/3} n)$.

Пример Д.5.20. Получить асимптотическую оценку функции

$$T(n) = 2T\left(\left[\frac{n}{4}\right]\right) + n\log_4 n.$$

Решение. Мы имеем $a = 2$, $b = 4$, $f(n) = n\log_4 n$, а $n^{\log_b a} = n^{\log_4 2} = n^{1/2} = \sqrt{n}$, функция $f(n) = n\log_4 n \geqslant n^{0,5+\varepsilon}$, при больших n и $\varepsilon = 0,5$. Остается проверить последнее условие для третьего случая теоремы. Для достаточно больших значений n имеем

$$2f\left(\frac{n}{4}\right) = 2\frac{n}{4}\log_4\frac{n}{4} \leqslant \frac{1}{2}n\log_4 n = d\cdot f(n), \quad \text{где} \quad d = \frac{1}{2} < 1,$$

условие выполнено, и тогда $T(n) = \Theta(n\log_4 n)$.

Набор упражнений Д.5.2

Д.5.2.1. Записать рекурсивную последовательность для вычисления квадратного корня в виде рекурсивно заданной функции.

Д.5.2.2. Показать, что рекурсивная последовательность

$$\begin{cases} a_1 = 1; \\ a_n = \frac{1}{3}\left(2a_{n-1} + \frac{8}{a_{n-1}^2}\right), \ n > 1, \end{cases}$$

имеет предел при $n \to \infty$ и найти значение этого предела.

Д.5.2.3. Найти общее решение рекуррентного соотношения

$$a_n = \frac{1}{3}a_{n-1} + \frac{2}{3}n^2 + \frac{2}{3}n - \frac{1}{3}.$$

Д.5.2.4. Найти общее решение рекуррентного соотношения

$$a_n = 2a_{n-1} - n + 2.$$

Д.5.2.5. Найти общее решение рекуррентного соотношения

$$a_n = -5a_{n-1} - 6a_{n-2}.$$

Д.5.2.6. Найти общее решение рекуррентного соотношения

$$a_n = -6a_{n-1} - 9a_{n-2}$$

Д.5.2.7. Найти общее решение рекуррентного соотношения

$$a_n = -2a_{n-1} - 2a_{n-2}.$$

Д.5.2.8. Если Вы считаете, что на основании рекуррентного соотношения очень просто вычислить следующее значение рекурсивно заданной функции, то попробуйте вычислить значение дважды рекурсивной функции Аккермана — $A(3,5)$, которая задается следующими рекуррентными соотношениями

$$A(m,n) = \begin{cases} n+1, & m = 0; \\ A(m-1,n), & n = 0; \\ A(m-1,A(m,n-1)). \end{cases}$$

Д.5.3. Конечные разности. Разностный и суммирующий операторы

Конечные разности как дискретный аналог дифференцирования. Исчисление конечных разностей разработано по аналогии с традиционным исчислением бесконечно малых. В основе исчисления бесконечно малых лежит дифференциальный оператор D, определяемый соотношением

$$Df(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

Напомним, что оператором называется некоторое отображение из множества функций во множество функций.

Исчисление конечных разностей основано на свойствах разностного оператора Δ, определяемого формулой

$$\Delta f(x) = f(x+1) - f(x).$$

Оператор Δ является конечно-разностным аналогом производной, в котором ограничились целочисленными значениями аргумента и, вместо $h \to 0$, взято ближайшее к нулю целое число — $h = 1$.

Далее следовало бы изучить свойства введенного оператора, но мы это сделаем позднее. Сейчас мы уйдем немного вперед, чтобы объяснить смысл того, ради чего нам понадобился такой оператор.

Суммирующий оператор и неопределенный интеграл. Оператор D в дифференциальном исчислении имеет обратный оператор — интегральный, обозначаемый символом $\int$. Связь дифференциального и интегрального операторов определяется следующим образом $g(x) = Df(x)$, тогда и только тогда, когда $\int g(x)dx = f(x) + C$. То есть $\int g(x)dx$ — *неопределенный интеграл* от функции $g(x)$ является классом функций, производная которых равна $g(x)$.

Точно так же для конечно-разностного оператора Δ введем обратный ему суммирующий оператор $\sum$. Связь конечно-разностного и суммирующего операторов определяется следующим образом.

$$g(x) = \Delta f(x),$$

тогда и только тогда, когда

$$\sum g(x)\delta x = f(x) + C.$$

Здесь обозначение $\sum g(x)\delta x$ есть *неопределенная сумма* функции $g(x)$ — класс функций, конечная разность которых есть $g(x)$. Буква «C» в случае неопределенного интеграла означает произвольную постоянную, в случае неопределенной суммы «C» означает любую функцию $p(x)$, для которой $p(x+1) = p(x)$.

Пока, собственно, не совсем понятно, для каких целей мы ввели суммирующий оператор, и все это пока напоминает некоторые математические развлечения — придется еще совсем немного подождать.

Определенные суммы. Классический курс интегрального и дифференциального исчисления содержит также понятие определенного интеграла. Мы надеемся, что его смысл читатель немного помнит.

В основе вычисления определенного интеграла лежит известная формула Ньютона–Лейбница:

$$g(x) = Df(x) \quad \Rightarrow \quad \int_a^b g(x)dx = f(x)\Big|_a^b = f(b) - f(a).$$

Аналогично введем понятие определенной суммы для конечных разностей

$$g(x) = \Delta f(x) \quad \Rightarrow \quad \sum_a^b g(x)\delta x = f(x)\Big|_a^b = f(b) - f(a).$$

Определенная сумма $\sum\limits_a^b g(x)\delta x$ введена просто по аналогии, но что она представляет на самом деле, предстоит еще разобраться. Пока это просто некоторый набор символов. Для того чтобы понять смысл введенной суммы вернемся немного назад. Пусть имеется обычная сумма

$$\sum_{k=a}^{k=b} g(k), \quad b \geqslant a. \tag{Д.22}$$

Предположим, что каким-то образом нам удалось отыскать *неопределенную сумму* функции $g(x)$, и она равна $f(x)$, т. е. $g(x) = f(x+1) - f(x)$. Тогда для

любого k имеем $g(k) = f(k+1) - f(k)$. Сумму (Д.22) перепишем с помощью многоточия

$$\sum_{k=a}^{k=b} g(k) = g(a) + g(a+1) + g(a+2) + \cdots + g(b-1) + g(b).$$

Далее воспользуемся связью между функциями $g(x)$ и $f(x)$, тогда выражение для суммы (Д.22) приобретает более интересный вид

$$\sum_{k=a}^{k=b} g(k) = (f(a+1) - f(a)) + (f(a+2) - f(a+1)) +$$
$$+ (f(a+3) - f(a+2)) + \cdots + (f(b) - f(b-1)) + (f(b+1) - f(b)) .$$

Интересный вид полученной суммы заключается в том, что *одинаковые* слагаемые в соседних скобках имеют разные знаки и взаимно уничтожают друг друга. После этого значение суммы приобретает «вполне приличный» вид

$$\sum_{k=a}^{k=b} g(k) = f(b+1) - f(a),$$

и полученный результат проясняет смысл определенной суммы, т. е.

$$\sum_{a}^{b} g(x)\delta x = \sum_{k=a}^{k=b-1} g(k).$$

По аналогии со свойствами определенного интеграла

$$\int_a^a g(x)dx = 0, \quad \int_a^b g(x)dx = -\int_b^a g(x)dx,$$

по определению для определенных сумм положим:

1) $\sum\limits_{a}^{a} g(x)\delta x = 0$;

2) при $b < a$ имеем $\sum\limits_{a}^{b} g(x)\delta x = -\sum\limits_{b}^{a} g(x)\delta x$.

Для определенного интеграла имеет место формула

$$\int_a^b g(x)dx + \int_b^c g(x)dx = \int_a^c g(x)dx.$$

Читатель сам без особого труда может доказать аналогичную формулу для определенных сумм

$$\sum_{a}^{b} g(x)\delta x + \sum_{b}^{c} g(x)\delta x = \sum_{a}^{c} g(x)\delta x.$$

Вспомним знаменитую формулу о производной определенного интеграла по переменному верхнему пределу

$$D\int_a^x g(t)dt = g(x).$$

Аналогом этой формулы является соотношение, доказательство которого мы приводим ниже

$$\sum_a^{b+1} g(x)\delta x - \sum_a^b g(x)\delta x = (f(b+1)-f(a)) - (f(b)-f(a)) = f(b+1)-f(b) = g(b).$$

Подведем итог, ради которого мы занимались определенными суммами. Пусть нам требуется вычислить сумму вида $\sum\limits_{k=a}^{k=b} g(k)$, и каким-то образом нам удалось найти неопределенную сумму функции $g(x)$

$$\sum g(x)\delta x = f(x),$$

т. е. найти такую функцию $f(x)$, для которой выполнено

$$f(x+1) - f(x) = g(x).$$

Тогда вычисление суммы уже не представляет принципиальной трудности (конечно, если сам вид функции $f(x)$ достаточно простой), а именно значение суммы выражается следующей формулой

$$\sum_{k=a}^{k=b} g(k) = \sum_a^{b+1} g(x)\delta x = f(b+1) - f(a).$$

Следовательно, проблема нахождения сумм сводится к проблеме отыскания неопределенных сумм соответствующих функций. Для того, что бы придать этому поиску некоторое разумное направление, нам понадобится изучить некоторые свойства разностного оператора Δ.

Конечные разности высших порядков. *Первая разность* или *разность первого порядка* функции f, обозначаемая Δf, определена выше формулой

$$\Delta f(x) = f(x+1) - f(x).$$

Вторая разность или *разность второго порядка* функции f, обозначаемая $\Delta^2 f$, определяется как

$$\Delta^2 f(x) = \Delta(\Delta f(x)).$$

Следовательно,

$$\begin{aligned}\Delta^2 f(x) &= \Delta(\Delta f(x)) = \Delta(f(x+1) - f(x)) = \\ &= (f(x+2) - f(x+1)) - (f(x+1) - f(x)) = f(x+2) - 2f(x+1) + f(x)\,.\end{aligned}$$

В общем случае разность порядка n определяется рекуррентным соотношением $\Delta^n f(x) = \Delta(\Delta^{n-1} f(x))$. Рассмотрим пока примеры вычисления конечных разностей, ниже мы получим общую формулу для $\Delta^n f(x)$.

Пример Д.5.21. Найти конечные разности от Δ до Δ^4 для функции $f(x) = x^3$.

Решение.

$$\Delta(x^3) = (x+1)^3 - x^3 = 3x^2 + 3x + 1.$$
$$\Delta^2(x^3) = \Delta(\Delta(x^3)) = \Delta(3x^2 + 3x + 1) =$$
$$= (3(x+1)^2 + 3(x+1) + 1) - (3x^2 + 3x + 1) = 6x + 6.$$
$$\Delta^3(x^3) = \Delta(\Delta^2(x^3)) = \Delta(6x+6) = (6(x+1)+6) - (6x+6) = 6.$$
$$\Delta^4(x^3) = \Delta(\Delta^3(x^3)) = \Delta(6) = 0.$$

Пример Д.5.22. Найти конечную разность для функции $f(x) = a^x$.

Решение.

$$\Delta(a^x) = a^{x+1} - a^x = (a-1)a^x.$$

Легко получить формулу $\Delta^n(a^x) = (a-1)^n a^x$, в частности $\Delta(2^x) = 2^x$. Из курса математического анализа читателю известно, что $D(e^x) = e^x$, таким образом в рамках аппарата конечных разностей функция $f(x) = 2^x$ является аналогом экспоненты в исчислении бесконечно малых.

Пример Д.5.23. С использованием выражения для $\Delta(a^x)$ доказать формулу для суммы геометрической прогрессии.

Решение. Поскольку геометрическая прогрессия задается суммой $\sum\limits_{k=1}^{k=n} aq^{k-1}$, то рассмотрим функцию $g(x) = q^{x-1}$. Неопределенной суммой для этой функции является функция

$$f(x) = \frac{q^{x-1}}{q-1}, \qquad \text{т. е.} \quad \sum q^{x-1}\delta x = \frac{q^{x-1}}{q-1},$$

тогда

$$\sum_{k=1}^{k=n} aq^{k-1} = \sum_{1}^{n+1} aq^{x-1}\delta x = a\frac{q^{x-1}}{q-1}\Bigg|_1^{n+1} = a\frac{q^n - 1}{q-1}.$$

Для дальнейшего исследования введем оператор E, определяемый соотношением $E(f(x)) = f(x+1)$, а через I обозначим тождественный оператор

$$I(f(x)) = f(x),$$

тогда оператор Δ может быть представлен в виде

$$\Delta(f(x)) = f(x+1) - f(x) = E(f(x)) - I(f(x)) = (E - I)(f(x)).$$

Положим $E^0 = I$, тогда

$$\begin{aligned}\Delta^n(f(x)) = (E-I)^n(f(x)) = E^n(f(x)) - n\cdot E^{n-1}(f(x)) + \cdots\\ \cdots + (-1)^k\cdot C_n^k\cdot E^{n-k}(f(x)) + \cdots + (-1)^n E^0(f(x)) =\\ = f(x+n) - nf(x+n-1) + \cdots\\ \cdots + (-1)^k\cdot C_n^k\cdot f(x+n-k) + \cdots + (-1)^n\cdot f(x),\end{aligned}$$

где $C_n^k = \frac{n!}{k!(n-k)!}$ — обозначение биномиальных коэффициентов. Отметим, что в отличие от принятого в основном тексте книги Р. Хаггарти обозначения $C(n,k)$, авторы используют обозначение, общепринятое в отечественной литературе — C_n^k.

В качестве применения полученной формулы вычислим $\Delta^4(f(x))$, имеем

$$\Delta^4(f(x)) = f(x+4) - 4f(x+3) + 6f(x+2) - 4f(x+1) + f(x).$$

Перечислим некоторые свойства операторов E и Δ, которые читатель без особого труда может проверить самостоятельно.

Свойства операторов E и Δ.

1. Линейность. Для любых функций f и g, и любых действительных чисел a и b выполнено $E(af+bg) = aE(f) + bE(g)$; $\Delta(af+bg) = a\Delta(f) + b\Delta(g)$. Кроме того, имеем: $E\Delta = \Delta E$; $E(a) = a$; $\Delta a = 0$.

2. Конечная разность произведения. Ниже мы получим аналоги формул производных произведения и частного. В курсе математического анализа известна формула

$$D(fg) = fD(g) + gD(f).$$

Для конечной разности имеем

$$\begin{aligned}&\Delta(f(x)g(x)) = f(x+1)g(x+1) - f(x)g(x) =\\ =&f(x+1)g(x+1) - f(x)g(x+1) + f(x)g(x+1) - f(x)g(x) =\\ =&g(x+1)(f(x+1) - f(x)) + f(x)(g(x+1) - g(x)) =\\ =&\Delta(f(x))E(g(x)) + f(x)\Delta(g(x)) = (f\cdot\Delta g + Eg\cdot\Delta f)(x),\end{aligned}$$

таким образом

$$\Delta(fg) = f\cdot\Delta(g) + E(g)\cdot\Delta(f). \qquad (Д.23)$$

3. Конечная разность частного. Для производной частного используется известная формула

$$D\left(\frac{f}{g}\right) = \frac{gD(f) - fD(g)}{g^2}.$$

В случае конечной разности имеем

$$\begin{aligned}&\Delta\left(\frac{f(x)}{g(x)}\right) = \frac{f(x+1)}{g(x+1)} - \frac{f(x)}{g(x)} = \frac{f(x+1)g(x) - f(x)g(x+1)}{g(x)g(x+1)} =\\ =&\frac{f(x+1)g(x) - f(x)g(x) + f(x)g(x) - f(x)g(x+1)}{g(x)g(x+1)} =\\ =&\frac{g(x)(f(x+1) - f(x)) - f(x)(g(x+1) - g(x))}{g(x)g(x+1)} = \frac{g(x)\Delta(f(x)) - f(x)\Delta(g(x))}{g(x)g(x+1)},\end{aligned}$$

таким образом

$$\Delta\left(\frac{f}{g}\right) = \frac{g\cdot\Delta(f) - f\cdot\Delta(g)}{g\cdot E(g)}. \qquad (Д.24)$$

Рассмотрим несколько примеров, решение которых основано на формулах (Д.23) и (Д.24).

Пример Д.5.24.

Найти следующие конечные разности

а) $\Delta(2x^3+3x^2+4x+5)$; б) $\Delta((x^2+4x+3)(x+5))$; в) $\Delta(\frac{x^2+4x+3}{x+5})$.

Решение. В начале напомним, что $\Delta(x) = (x+1)-x = 1$, $\Delta(x^2) = (x+1)^2 - x^2 = 2x+1$, и ранее было показано, что $\Delta(x^3) = 3x^2+3x+1$.

а) Найти $\Delta(2x^3+3x^2+4x+5)$.

$$\Delta(2x^3+3x^2+4x+5) = 2\Delta(x^3)+3\Delta(x^2)+4\Delta(x)+5\Delta(1) =$$
$$= 2(3x^2+3x+1)+3(2x+1)+4(1)+0 = 6x^2+12x+9.$$

б) Найти $\Delta((x^2+4x+3)(x+5))$. Используем формулу (Д.23), и получаем

$$\Delta((x^2+4x+3)(x+5)) = (x^2+4x+3)\cdot\Delta(x+5) + E(x+5)\Delta(x^2+4x+3) =$$
$$= (x^2+4x+3)\cdot 1 + ((x+1)+5)(2x+1+4)) =$$
$$= (x^2+4x+3) + (x+6)(2x+5).$$

в) Найти $\Delta(\frac{x^2+4x+3}{x+5})$. На основании формулы (Д.24) имеем

$$\Delta\left(\frac{x^2+4x+3}{x+5}\right) = \frac{(x+5)\cdot\Delta(x^2+4x+3) - (x^2+4x+3)\cdot\Delta(x+5)}{(x+5)E(x+5)} =$$
$$= \frac{(x+5)\cdot(2x+1+4) - (x^2+4x+3)\cdot 1}{(x+5)((x+1)+5)} =$$
$$= \frac{(x+5)(2x+5) - (x^2+4x+3)}{(x+5)(x+6)}.$$

Факториальные многочлены. В целях дальнейшего исследования найдем $\Delta(x^n)$ — используя бином Ньютона получаем

$$\Delta(x^n) = (x+1)^n - x^n = nx^{n-1} + \frac{n(n-1)}{2}x^{n-2} + \cdots + C_n^k\cdot x^{n-k} + \cdots + 1,$$

но обычная формула классического анализа дает следующий результат

$$D(x^n) = nx^{n-1}.$$

Мы видим, что конечная разность для функции $f(x) = x^n$ имеет довольно сложное выражение и это, конечно, не самый приятный факт, поскольку с многочленами приходится часто иметь дело при различных вычислениях. Попытаемся выбраться из этой ситуации, при минимальных потерях.

Введем в рассмотрение функцию $f(x) = x^{(n)}$, которую определим следующим образом. Пусть при $x \geqslant n \geqslant 0$

$$\begin{cases} x^{(n)} = 1, & n = 0; \\ x^{(n)} = x \cdot (x-1) \cdot (x-2) \cdots (x-n+1), & n > 0. \end{cases}$$

Фактически функция $f(x) = x^{(n)}$ представляет собой многочлен степени n, правда не самого приятного вида, если раскрыть скобки. Но сейчас мы обнаружим интересный факт, вычислив его конечную разность.

$$\begin{aligned} \Delta(x^{(n)}) &= (x+1) \cdot (x) \cdot (x-1) \cdots (x-n+2) - x \cdot (x-1) \cdot (x-2) \cdots (x-n+1) = \\ &= (x \cdot (x-1) \cdot (x-2) \cdots (x-n+2)) \cdot ((x+1) - (x-n+1)) \\ &= nx \cdot (x-1) \cdot (x-2) \cdots (x-n+2) = nx^{(n-1)}. \end{aligned}$$

Таким образом, с точки зрения аппарата конечных разностей функция $f(x) = x^{(n)}$ является, в некотором смысле, аналогом функции $f(x) = x^n$.

Факториальным многочленом степени n назовем функцию вида

$$f(x) = a_n x^{(n)} + a_{n-1} x^{(n-1)} + \cdots + a_2 x^{(2)} + a_1 x^{(1)} + a_0.$$

Вычисление конечных разностей для факториальных многочленов не представляет серьезных проблем.

Пример Д.5.25. Вычислить конечную разность для функции

$$f(x) = 4x^{(3)} - 5x^{(2)} + 7x^{(1)} + 2.$$

Решение.

$$\begin{aligned} \Delta(f(x)) &= \Delta(4x^{(3)} - 5x^{(2)} + 7x^{(1)} + 2) = 4\Delta(x^{(3)}) - 5\Delta(x^{(2)}) + 7\Delta(x^{(1)}) + \Delta(2) = \\ &= 4 \cdot 3x^{(2)} - 5 \cdot 2x^{(1)} + 7 \cdot 1 + 0 = 12x^{(2)} - 10x^{(1)} + 7. \end{aligned}$$

Представление обычных многочленов факториальными многочленами. Остается нерешенным вопрос о том, каким образом обычный многочлен можно представить в виде факториального многочлена. Прежде чем перейти к этому вопросу, напомним читателям одну теорему, известную как теорема Безу.

Теорема. Остаток, полученный при делении многочлена

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

(степени $n \geqslant 1$) на многочлен первой степени $(x-a)$, равен значению многочлена в точке $x = a$, т. е. $f(a)$.

Схему представления обычного многочлена в виде факториального поясним на примере — пусть $f(x) = x^4$. Ясно, что факториальный многочлен должен иметь ту же степень, что и исходный, значит имеем

$$x^4 = a_4 x^{(4)} + a_3 x^{(3)} + a_2 x^{(2)} + a_1 x^{(1)} + a_0.$$

Распишем это выражение более подробно

$$x^4 = a_4 x(x-1)(x-2)(x-3) + a_3 x(x-1)(x-2) + a_2 x(x-1) + a_1 x + a_0.$$

Теперь левую и правую часть этого выражения поделим с остатком на многочлен первой степени x. Остатки, полученные при делении, соответственно равны: в левой части — нулю, а в правой — a_0, тогда $a_0 = 0$. Частное в левой части равно x^3, в правой получаем

$$a_4(x-1)(x-2)(x-3) + a_3(x-1)(x-2) + a_2(x-1) + a_1,$$

тогда имеем

$$x^3 = a_4(x-1)(x-2)(x-3) + a_3(x-1)(x-2) + a_2(x-1) + a_1.$$

Теперь поделим обе части на многочлен первой степени — $(x-1)$. Остатки: в левой части остаток равен 1, в правой — a_1, тогда $a_1 = 1$. Частное в левой части — $x^2 + x + 1$, в правой — $a_4(x-2)(x-3) + a_3(x-2) + a_2$, тогда имеем

$$x^2 + x + 1 = a_4(x-2)(x-3) + a_3(x-2) + a_2.$$

Поделим обе части на $(x-2)$. Остатки: в левой части остаток равен 7, в правой — a_2, тогда $a_2 = 7$. Частные: в левой части — $x+3$, в правой — $a_4(x-3) + a_3$, тогда $x + 3 = a_4(x-3) + a_3$.

И последнее деление — поделим обе части на $(x-3)$. Остатки: в левой части остаток равен 6, в правой — a_3, тогда $a_3 = 6$. Частные: в левой части — 1, в правой — a_4, значит $a_4 = 1$. В результате окончательно получаем

$$x^4 = x^{(4)} + 6x^{(3)} + 7x^{(2)} + x^{(1)}.$$

Отметим, что приведенная схема применима для представления любого многочлена как факториального. Отметим так же, что коэффициенты при старших степенях и свободные члены не меняются при переходе от обычного многочлена к факториальному.

Поскольку $\Delta\left(x^{(n)}\right) = nx^{(n-1)}$, и, следовательно, $\Delta\left(\frac{x^{(n+1)}}{n+1}\right) = x^{(n)}$, то мы можем найти неопределенную сумму

$$\sum x^{(n)}\delta x = \frac{x^{(n+1)}}{n+1},$$

мы опускаем при этом «C» в выражении для неопределенной суммы. Воспользуемся полученным представлением x^4 для вычисления суммы $\sum\limits_{k=0}^{n} k^4$.

Функция $f(x) = x^4$ представима как факториальный многочлен в виде

$$x^4 = x^{(4)} + 6x^{(3)} + 7x^{(2)} + x^{(1)},$$

тогда

$$\sum_{k=0}^{n} k^4 = \sum_{x=0}^{n+1} x^4\delta x = \sum_{x=0}^{n+1} (x^{(4)} + 6x^{(3)} + 7x^{(2)} + x^{(1)})\delta x =$$

$$= \left(\frac{x^{(5)}}{5} + 6\frac{x^{(4)}}{4} + 7\frac{x^{(3)}}{3} + \frac{x^{(2)}}{2}\right)\Bigg|_0^{n+1} = \left(\frac{x(x-1)(2x-1)(3x^2-3x-1}{30}\right)\Bigg|_0^{n+1} =$$

$$= \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}.$$

Факториальные многочлены отрицательных степеней. До сих пор запись $x^{(n)}$ предполагала, что $n \geqslant 0$. Сейчас мы попытаемся ввести аналог для отрицательных значений степени. Для неотрицательных степеней нетрудно установить связь

$$x^{(n-1)} = \frac{x^{(n)}}{x-n+1}. \qquad (Д.25)$$

С помощью этой формулы попытаемся определить $x^{(n)}$ для отрицательных значений степени. Полагая $n = 1$, получаем

$$x^{(0)} = \frac{x^{(1)}}{x-1+1} = 1,$$

а полагая $n = 0$, имеем

$$x^{(-1)} = \frac{x^{(0)}}{x-0+1} = \frac{1}{x+1}.$$

Используя метод индукции и соотношение (Д.25) покажем, что

$$x^{(-m)} = \frac{1}{(m+x)^{(m)}},$$

для всех $m \geqslant 1$, при которых знаменатель не равен нулю. При $m = 1$ получаем

$$x^{(-1)} = \frac{1}{(1+x)} = \frac{1}{(1+x)^{(1)}}.$$

Предположим, что $x^{(-k)} = \frac{1}{(k+x)^{(k)}}$, тогда

$$\begin{aligned} x^{(-k-1)} &= \frac{x^{(-k)}}{x+k+1} = \frac{1}{(x+k)^{(k)}} \cdot \frac{1}{x+k+1} = \\ &= \frac{1}{(x+k)(x+k-1)\cdots(x+1)} \cdot \frac{1}{x+k+1} = \\ &= \frac{1}{(x+k+1)(x+k)(x+k-1)\cdots(x+1)} = \frac{1}{(x+k+1)^{(m)}}. \end{aligned}$$

Используя полученный результат, вычислим $\Delta(x^{(-n)})$, получаем

$$\begin{aligned} \Delta(x^{(-n)}) &= \Delta\left(\frac{1}{(x+n+1)^{(n)}} - \frac{1}{(x+n)^{(n)}}\right) = \\ &= \left(\frac{1}{(x+n+1)\cdot(x+n)\cdots(x+2)} - \frac{1}{(x+n)\cdot(x+n-1)\cdots(x+1)}\right) = \\ &= \frac{1}{(x+n)\cdot(x+n-1)\cdots(x+2)}\left(\frac{1}{(x+n+1)} - \frac{1}{(x+1)}\right) = \\ &= \frac{-n}{(x+n+1)\cdot(x+n)\cdots(x+1)} = \frac{-n}{(x+n+1)^{(n+1)}} = -nx^{(-n-1)}. \end{aligned}$$

Сравните полученную конечную разность с действием дифференциального оператора на функцию $f(x) = x^{-n}$, результат известен: $D(x^{-n}) = -nx^{-n-1}$.

Следовательно, неопределенная сумма от функции $x^{(-n)}$ равна

$$\sum x^{(-n)}\delta x = \frac{x^{(-n+1)}}{(-n+1)}.$$

В классическом курсе математического анализа хорошо известна формула интегрирования по частям. Сейчас мы получим аналогичную формулу для суммирующего оператора — формулу суммирования по частям.

Формула суммирования по частям и ее применение. Для произведения функций f и g ранее была получена формула

$$\Delta(fg) = f\cdot\Delta(g) + E(g)\cdot\Delta(f),$$

поэтому

$$\sum \Delta(fg)\delta x = \sum f\cdot\Delta(g)\delta x + \sum E(g)\cdot\Delta(f)\delta x,$$

следовательно, получаем формулу

$$\sum f\cdot\Delta(g)\delta x = (fg) - \sum E(g)\cdot\Delta(f)\delta x,$$

которая и называется формулой суммирования по частям.

Пример Д.5.26. Найти сумму $S(n) = \sum\limits_{k=0}^{n} k2^k$.

Решение.

$$\sum_{k=0}^{k=n} k2^k = \sum_{0}^{n+1} x2^x\delta x,$$

но так как $\sum 2^x\delta x = 2^x$, $\quad x^{(1)} = x$, то полученную сумму перепишем в виде

$$\sum_{0}^{n+1} x2^x\delta x = \sum_{0}^{n+1} x^{(1)}\Delta(2^x)\delta x,$$

и используя формулу суммирования по частям, получаем

$$\begin{aligned}\sum_{0}^{n+1} x2^x\delta x &= \sum_{0}^{n+1} x^{(1)}\Delta(2^x)\delta x = (x^{(1)}(2^x))\Big|_0^{n+1} - \sum_{0}^{n+1} E(2^x)\Delta(x^{(1)})\delta x = \\ &= (x2^x)\Big|_0^{n+1} - \sum_{0}^{n+1}(2^{x+1})\delta x = (x2^x)\Big|_0^{n+1} - (2^{x+1})\Big|_0^{n+1} = \\ &= (n+1)\cdot 2^{n+1} - 2^{n+2} + 2,\end{aligned}$$

таким образом

$$S(n) = \sum_{k=0}^{n} k2^k = (n+1)\cdot 2^{n+1} - 2^{n+2} + 2.$$

Пример Д.5.27. Найти сумму $S(n)=\sum\limits_{k=0}^{n}k^2 3^k$.

Решение. Сделаем некоторые предварительные замечания.

1) $\sum 3^x\delta x=\frac{3^x}{2}$.

2) $x^2=x(x-1)+x=x^{(2)}+x^{(1)}$.

3) $E(3^x)=3^{x+1}=3\cdot 3^x$.

С учетом этих замечаний, используя формулу суммирования по частям, имеем

$$\begin{aligned}
\sum_{k=0}^{n}k^2 3^k &= \sum_{0}^{n+1}x^2 3^x\delta x=\sum_{0}^{n+1}\left(x^{(2)}+x^{(1)}\right)\frac{1}{2}\Delta(3^x)\delta x=\\
&=\left(\left(x^{(2)}+x^{(1)}\right)\frac{1}{2}(3^x)\right)\Big|_0^{n+1}-\frac{1}{2}\sum_{0}^{n+1}E(3^x)\Delta\left(x^{(2)}+x^{(1)}\right)\delta x=\\
&=\frac{1}{2}\left(x^2 3^x\right)\Big|_0^{n+1}-\frac{3}{2}\sum_{0}^{n+1}\left(2x^{(1)}+1\right)3^x\delta x=\\
&=\frac{1}{2}\left(x^2 3^x\right)\Big|_0^{n+1}-\frac{3}{2}\sum_{0}^{n+1}\left(2x^{(1)}+1\right)\Delta\left(\frac{3^x}{2}\right)\delta x=\\
&=\frac{1}{2}\left(x^2 3^x\right)\Big|_0^{n+1}-\frac{3}{4}\left(3^x(2x^{(1)}+1)\right)\Big|_0^{n+1}+\frac{3}{4}\sum_{0}^{n+1}E(3^x)\Delta\left(2x^{(1)}+1\right)\delta x=\\
&=\frac{1}{2}\left(x^2 3^x\right)\Big|_0^{n+1}-\frac{3}{4}\left(3^x(2x^{(1)}+1)\right)\Big|_0^{n+1}+\frac{9}{2}\sum_{0}^{n+1}3^x\delta x=\\
&=\frac{1}{2}\left(x^2 3^x\right)\Big|_0^{n+1}-\frac{3}{4}\left(3^x(2x^{(1)}+1)\right)\Big|_0^{n+1}+\frac{9}{2}\cdot\frac{1}{2}\left(3^x\right)\Big|_0^{n+1}=\\
&=\frac{1}{2}(n+1)^2\cdot 3^{n+1}-\frac{3}{4}(2n+3)\cdot 3^{n+1}+\frac{3}{4}+\frac{9}{4}\cdot 3^{n+1}-\frac{9}{4}=\\
&=\frac{3^{n+1}}{2}(n^2-n+1)-\frac{3}{2}.
\end{aligned}$$

Таким образом

$$S(n)=\sum_{k=0}^{n}k^2 3^k=\frac{3^{n+1}}{2}(n^2-n+1)-\frac{3}{2}.$$

Набор упражнений Д.5.3

Д.5.3.1. Найти конечную разность $\Delta\left(x^2\cdot 5^x\right)$.

Д.5.3.2. Найти конечную разность $\Delta\left(\dfrac{3^x}{2x+1}\right)$.

Д.5.3.3. Найти конечную разность $\Delta\left(x^3+7^{3x}\right)$.

Д.5.3.4. Представить в виде факториального многочлена функцию $f(x) = x^3$.

Д.5.3.5. Представить в виде факториального многочлена функцию

$$f(x) = 2x^4 - 3x^2 + 5x - 1.$$

Д.5.3.6. Найти сумму $\sum\limits_{k=0}^{n} k^2$.

Д.5.3.7. Найти сумму $\sum\limits_{k=0}^{n} k^3 2^k$.

Д.5.4. Производящие функции и комбинаторные подсчеты

Довольно часто объектом исследования в дискретной математике является некоторая последовательность чисел

$$(a_0, a_1, a_2, \ldots, a_n, \cdots).$$

Мы будем говорить далее о бесконечных последовательностях чисел. При этом мы не ограничиваем общности, поскольку любую конечную последовательность можно рассматривать как бесконечную, считая все ее члены, начиная с некоторого номера, равными нулю. Использование производящих функций ставит своей целью рассмотрение последовательностей не как набора чисел, а как единого целого. Формальный подход аппарата производящих функций начинается с того, что указанной выше последовательности ставится в соответствие степенной ряд

$$\sum_{k=0}^{\infty} a_k z^k, \tag{Д.26}$$

где z рассматривается как комплексное число.

Сумма этого ряда (мы ее будем обозначать через $G(z)$) и называется производящей функцией для данной последовательности. Таким образом, производящая функция $G(z)$ для последовательности $(a_0, a_1, a_2, \ldots, a_n, \cdots)$ равна

$$G(z) = \sum_{k=0}^{\infty} a_k z^k.$$

Одно преимущество производящей функции уже видно. Вместо бесконечного набора чисел мы будем иметь дело с одной функцией. Конечно, это достоинство будет действительно таковым, если сама функция будет иметь не очень сложный вид. Но возникает резонный вопрос — каким образом, зная функцию $G(z)$, восстановить последовательность $(a_0, a_1, a_2, \ldots, a_n, \cdots)$, — без ответа на него все дальнейшие рассуждения теряют всякий смысл.

Мы не будем вдаваться в тонкости теории функции комплексного переменного. При не слишком больших ограничениях степенной ряд (Д.26) абсолютно сходится в некотором круге на комплексной плоскости с центром в точке $z = 0$, а функция $G(z)$ является аналитической в этом круге. Тогда члены последовательности $(a_0, a_1, a_2, \ldots, a_n, \cdots)$ являются коэффициентами ряда Тейлора для

этой функции в точке $z = 0$. Для вычисления коэффициентов ряда Тейлора известна формула

$$a_n = \frac{1}{n!} D^n(G(z))\Big|_{z=0}, \qquad (Д.27)$$

где через $D^n(G(z))\Big|_{z=0}$ обозначена производная порядка n функции $G(z)$ в точке $z = 0$.

Сейчас мы установим связь между некоторыми конкретными последовательностями и соответствующими производящими функциями.

Пример Д.5.28. Определить последовательность, задаваемую производящей функцией $G(z) = (az + b)^n$.

Решение. Мы надеемся, что читатель помнит простейшие правила вычисления производных, используя которые мы получаем:

$$D((az+b)^n) = n(az+b)^{n-1}a;$$
$$D^2((az+b)^n) = n(n-1)(az+b)^{n-2}a^2;$$
$$D^k((az+b)^n) = n(n-1)\cdots(n-k+1)(az+b)^{n-k}a^k, \text{ при } k \leqslant n;$$
$$D^k((az+b)^n) = 0, \text{ при } k > n.$$

Так как для членов соответствующей последовательности имеет место формула (Д.27), то имеем

$$\begin{cases} a_0 = b^n; \quad a_1 = nab^{n-1}; \quad a_2 = \dfrac{1}{2!}n(n-1)a^2b^{n-2}; \\ a_k = \dfrac{1}{k!}n(n-1)\cdots(n-k+1)a^kb^{n-k}, \quad k \leqslant n; \\ a_k = 0, \quad k > n. \end{cases}$$

Заметим, что

$$\frac{1}{k!}n(n-1)\cdots(n-k+1) = \frac{1}{k!}\cdot\frac{n(n-1)\cdots2\cdot1}{(n-k)(n-k-1)\cdots2\cdot1} = \frac{n!}{k!(n-k)!} = C_n^k,$$

где C_n^k — биномиальные коэффициенты или, что тоже самое, число сочетаний без повторений из n по k. Таким образом, данной производящей функции соответствует последовательность $(a_0, a_1, a_2, \ldots, a_n, \cdots)$, где

$$\begin{cases} a_k = C_n^k a^k b^{n-k}, \quad k \leqslant n; \\ a_k = 0, \quad k > n. \end{cases} \qquad (Д.28)$$

Рассмотрим частные случаи этой формулы. Пусть $G(z) = (z - b)^n$, тогда

$$\begin{cases} a_k = C_n^k(-1)^{n-k}b^{n-k}, \quad k \leqslant n; \\ a_k = 0, \quad k > n. \end{cases}$$

И еще очень важный частный случай, когда $G(z) = (1 + z)^n$, по (Д.28) имеем

$$\begin{cases} a_k = C_n^k, \quad k \leqslant n; \\ a_k = 0, \quad k > n, \end{cases}$$

т. е. функция $G(z) = (1+z)^n$ является производящей функцией для числа сочетаний без повторений из n по k.

Теперь рассмотрим функцию

$$G(z) = \frac{1}{(az+b)^{m+1}}, \quad m \geqslant 0,$$

$$D^k(g(z)) = D^k\left(\frac{1}{(az+b)^{m+1}}\right) = (-1)^k \frac{(m+1)(m+2)\cdots(m+k)}{(az+b)^{m+1+k}} a^k,$$

тогда

$$a_k = \frac{1}{k!}(-1)^k \frac{(m+1)(m+2)\cdots(m+k)}{b^{m+1+k}} a^k = (-1)^k \frac{a^k}{b^{m+1+k}} \cdot \frac{1}{k!} \cdot \frac{1\cdot 2\cdots(m+k)}{1\cdot 2\cdots m},$$

или в окончательном виде

$$a_k = (-1)^k \frac{a^k}{b^{m+1+k}} \cdot C^m_{m+k}. \tag{Д.29}$$

Рассмотрим частные случаи этой формулы, пусть

$$G(z) = \frac{1}{(z-b)^{m+1}},$$

тогда

$$a_k = (-1)^{m+1} \frac{1}{b^{m+1+k}} \cdot C^m_{m+k}. \tag{Д.30}$$

В случае, если

$$G(z) = \frac{1}{(1+z)^{m+1}} \quad \Rightarrow \quad a_k = (-1)^k \cdot C^m_{m+k},$$

а, если

$$G(z) = \frac{1}{(1-z)^m} \quad \Rightarrow \quad a_k = C^m_{m+k}.$$

Отдельно рассмотрим случай, когда $m = 0$ — в этом случае

$$G(z) = \frac{1}{(1+z)},$$

тогда $a_k = (-1)^k$, т. е. последовательность состоит из единиц с чередующимися знаками, и, наконец, для функции

$$G(z) = \frac{1}{(1-z)}$$

имеем $a_k = 1$, т. е. все члены данной последовательности равны единице.

Найдем соответствующую последовательность для производящей функции

$$G(z) = \frac{1-z^{n+1}}{(1-z)}.$$

Мы надеемся, что читатель к этому моменту еще не забыл формулу суммы геометрической прогрессии, и он вспомнит, что данная функция есть не что иное, как сумма

$$G(z) = \frac{1 - z^{n+1}}{(1 - z)} = 1 + z + z^2 + \cdots + z^n.$$

Поэтому последовательность, соответствующая данной производящей функции, имеет вид: $a_k = 1$ при $k \leqslant n$; $a_k = 0$ при $k > n$. Этот вывод можно было получить другим способом, если обратить внимание на одно очень важное свойство производящих функций.

Пусть производящей функции $G(z)$ соответствует последовательность $(a_0, a_1, a_2, \ldots, a_n, \cdots)$. Тогда функции $G_1(z) = z^n \cdot G(z)$, $n > 0$ соответствует последовательность $(b_0, b_1, b_2, \ldots, b_n, \cdots)$ и между этими последовательностями существует следующая связь: $b_k = 0$ при $k < n$; $b_k = a_{k-n}$ при $k \geqslant n$. То есть при умножении производящей функции на z^n соответствующая последовательность сдвигается на n шагов вправо. С учетом указанного замечания полученный результат для функции

$$G(z) = \frac{1 - z^{n+1}}{(1 - z)}$$

становится очевидным.

Отметим также тот факт, что производящие функции обладают свойством линейности по отношению к порождающим их последовательностям. Линейность понимается в следующем аспекте. Пусть имеются две последовательности $(a_0, a_1, a_2, \ldots, a_n, \cdots)$ и $(b_0, b_1, b_2, \ldots, b_n, \cdots)$, производящие функции которых равны, соответственно $G_1(z)$ и $G_2(z)$, тогда для последовательности

$$(c \cdot a_0 + d \cdot b_0, c \cdot a_1 + d \cdot b_1, \ldots, c \cdot a_n + d \cdot b_n, \ldots)$$

производящая функция $G(z)$ равна

$$G(z) = c \cdot G_1(z) + d \cdot G_2(z).$$

Многочлены и рациональные дроби. В дальнейшем мы рассмотрим методы восстановления последовательности, когда производящая функция является рациональной дробью, но в начале напомним некоторые факты, касающиеся многочленов в комплексной области. Многочленом степени n (его будем обозначать через $P_n(z)$) называется выражение вида

$$P_n(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_2 z^2 + a_1 z + a_0,$$

где $a_0, a_1, a_2, \ldots, a_n$ в общем случае являются комплексными числами, $a_n \neq 0$. Корнем многочлена называется число c, такое что $P_n(c) = 0$. Если число c является корнем многочлена, то многочлен может быть представлен в виде $P_n(z) = (z - c)P_{n-1}(z)$, где $P_{n-1}(z)$ — многочлен соответствующей степени. Число c является корнем многочлена кратности r, если многочлен может быть представлен в виде $P_n(z) = (z - c)^r P_{n-r}(z)$, где $P_{n-r}(c) \neq 0$. Многочлен степени n имеет

ровно n корней с учетом их кратности. Последний факт означает, что если многочлен

$$P_n(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_2 z^2 + a_1 z + a_0$$

имеет k различных корней $c_1, c_2, ..., c_k$ кратности $r_1, r_2, ..., r_k$ соответственно, т. е. $(r_1 + r_2 + \cdots + r_k = n)$, то его можно представить в виде

$$P_n(z) = a_n (z - c_1)^{r_1} (z - c_2)^{r_2} \cdots (z - c_k)^{r_k}.$$

Рациональной дробью $R(z)$ называется выражение вида $R(z) = \frac{P_n(z)}{P_m(z)}$, где $P_n(z)$ и $P_m(z)$ — многочлены соответствующих степеней. Рациональная дробь называется правильной дробью, если степень многочлена в знаменателе строго больше степени многочлена в числителе. В противном случае дробь называется неправильной. Если рациональная дробь $R(z) = \frac{P_n(z)}{P_m(z)}$ является неправильной дробью, то она может быть представлена в виде

$$R(z) = P_{n-m}(z) + R_1(z),$$

где $P_{n-m}(z)$ многочлен степени $n - m$, $R_1(z)$ — правильная рациональная дробь. Элементарной (или простейшей) рациональной дробью называется выражение вида

$$R(z) = \frac{A}{(z - a)^m}.$$

Любая правильная рациональная дробь может быть разложена в сумму элементарных дробей. Этот факт позволяет представить схему нахождения соответствующих последовательностей в случае, когда производящая функция является рациональной дробью. На первом этапе для неправильной дроби ее представляют в виде суммы многочлена и правильной дроби. Проблема нахождения соответствующей последовательности для многочлена решается просто. Правильная дробь раскладывается в сумму элементарных дробей. Способ определения соответствующей последовательности для элементарной дроби мы изложили ранее.

Разложение на элементарные дроби. Схема представления правильной дроби в виде суммы элементарных дробей выглядит следующим образом. Пусть имеется правильная дробь вида

$$R(z) = \frac{P_n(z)}{P_m(z)}, \quad m > n,$$

при этом многочлен в знаменателе $P_m(z) = b_m z^m + b_{m-1} z^{m-1} + \cdots + b_2 z^2 + b_1 z + b_0$ имеет k различных корней $c_1, c_2, ..., c_k$ кратности $r_1, r_2, ..., r_k$ соответственно и, значит, его можно представить в виде

$$P_m(z) = b_m (z - c_1)^{r_1} (z - c_2)^{r_2} \cdots (z - c_k)^{r_k},$$

тогда рациональная дробь $R(z)$ может быть представлена в следующем виде

$$R(z) = \frac{P_n(z)}{P_m(z)} = \frac{P_n(z)}{b_m(z-c_1)^{r_1}(z-c_2)^{r_2}\cdots(z-c_k)^{r_k}} =$$
$$= \frac{A_{1,r_1}}{(z-c_1)^{r_1}} + \frac{A_{1,r_1-1}}{(z-c_1)^{r_1-1}} + \cdots + \frac{A_{1,1}}{(z-c_1)} +$$
$$+ \frac{A_{2,r_2}}{(z-c_2)^{r_2}} + \frac{A_{2,r_2-1}}{(z-c_2)^{r_2-1}} + \cdots + \frac{A_{2,1}}{(z-c_2)} + \cdots$$
$$\cdots + \frac{A_{k,r_k}}{(z-c_k)^{r_k}} + \frac{A_{k,r_k-1}}{(z-c_k)^{r_k-1}} + \cdots + \frac{A_{k,1}}{(z-c_k)}.$$

Способ определения коэффициентов мы покажем на примерах.

Пример Д.5.29. Найти последовательность $(a_0, a_1, a_2, \ldots, a_n)$, для которой производящая функция равна

$$G(z) = \frac{4z^2 - z - 9}{(z+2)(z+1)(z-1)}.$$

Решение. Выражение для производящей функции является правильной рациональной дробью. Представление этой дроби в виде суммы простейших дробей ищем в виде

$$\frac{4z^2 - z - 9}{(z+2)(z+1)(z-1)} = \frac{A}{(z+2)} + \frac{B}{(z+1)} + \frac{C}{(z-1)}.$$

Осталось определить неизвестные пока коэффициенты A, B, C. Приведем правую часть к общему знаменателю, который равен $(z+2)(z+1)(z-1)$, т. е. к знаменателю в левой части выражения, и приравнивая числители в правой и левой частях, получаем

$$4z^2 - z - 9 = A(z+1)(z-1) + B(z+2)(z-1) + C(z+2)(z+1). \qquad (Д.31)$$

В соотношении (Д.31) приравниваем коэффициенты при одинаковых степенях z и получаем систему уравнений для определения A, B, C.

$$\begin{cases} A + B + C = 4, \\ B + 3C = -1, \\ -A - 2B + 2C = -9. \end{cases}$$

Решая эту систему уравнений, находим коэффициенты. Отметим, что в данном случае нет необходимости решать систему уравнений, поскольку неизвестные коэффициенты можно определить более простым путем. Равенство (Д.31) рассматриваем как равенство многочленов, верное для любых значений z. Подставим в левую и правую части этого равенства $z = -2$, получим $9 = 3A$, и $A = 3$, подставляя $z = -1$, имеем $-4 = -2B$, и, следовательно $B = 2$. Подстановка $z = 1$ дает $-6 = 6C$, откуда $C = -1$. Таким образом,

$$\frac{4z^2 - z - 9}{(z+2)(z+1)(z-1)} = \frac{3}{(z+2)} + \frac{2}{(z+1)} - \frac{1}{(z-1)}.$$

Для элементарной дроби $\frac{1}{z+2}$, согласно (Д.29) соответствующая последовательность $(b_0, b_1, b_2, \ldots, b_n, \cdots)$ имеет вид $b_n = \frac{(-1)^n}{2^{n+1}}$, аналогично для элементарной дроби $\frac{1}{z+1}$ получаем, что $b_n = (-1)^n$, для элементарной дроби $\frac{1}{z-1}$ согласно (Д.29) соответствующая последовательность имеет вид $b_n = (-1)$. Надеемся, что читатель без особых усилий доведет решение задачи до конца.

Пример Д.5.30. Найти последовательность, для которой производящая функция равна

$$G(z) = \frac{4z-8}{(z-3)(z-1)^2}.$$

Решение. Поскольку дробь правильная, то представим ее в виде суммы элементарных дробей вида

$$\frac{4z-8}{(z-3)(z-1)^2} = \frac{A}{(z-3)} + \frac{B}{(z-1)^2} + \frac{C}{(z-1)}.$$

Приводя к общему знаменателю и приравнивая числители в левой и правой частях, получаем

$$4z-8 = A(z-1)^2 + B(z-3) + C(z-3)(z-1). \qquad (Д.32)$$

Приравнивая коэффициенты при одинаковых степенях z, получаем систему уравнений для определения A, B, C.

$$\begin{cases} A + C = 0, \\ -2A + B - 4C = 4, \\ A - 3B + 3C = -8. \end{cases}$$

Решая эту систему уравнений, находим коэффициенты. Поступая аналогично примеру 1.29, рассматриваем равенство (Д.32) как равенство многочленов, верное для любых значений z. Подставляя в левую и правую части этого равенства $z = 3$, получим $4 = 4A$, следовательно $A = 1$, подстановка в (Д.32) $z = 1$ дает $-4 = -2B$, и $B = 2$. Из первого уравнения системы находим, что $C = -1$, тогда

$$\frac{4z-8}{(z-3)(z-1)^2} = \frac{1}{(z-3)} + \frac{2}{(z-1)^2} - \frac{1}{(z-1)}.$$

Для элементарной дроби $\frac{1}{z-3}$ согласно (Д.29) соответствующая последовательность имеет вид $b_n = -\frac{1}{3^{n+1}}$, а для элементарной дроби $\frac{1}{(z-1)^2}$ последовательность $(b_0, b_1, b_2, \ldots, b_n, \cdots)$ имеет вид $b_n = C^1_{1+n} = (n+1)$. Для элементарной дроби $\frac{1}{z-1}$ соответствующая последовательность найдена ранее. Определение окончательного ответа предоставим читателю.

Пример Д.5.31. Найти последовательность $(a_0, a_1, a_2, \ldots, a_n, \cdots)$, для которой производящая функция равна $G(z) = \frac{4}{z^2-2z+2}$.

Решение. Представим $G(z)$ в виде

$$G(z) = \frac{4}{(z-(1+i))(z-(1-i))}.$$

Разложение на элементарные дроби ищем в виде

$$\frac{4}{(z-(1+i))(z-(1-i))} = \frac{A}{(z-(1+i))} + \frac{B}{(z-(1-i))}.$$

Аналогичным образом, определяя неизвестные коэффициенты, мы получаем $A=-2i$, $B=2i$, тогда

$$\frac{4}{(z-(1+i))(z-(1-i))} = \frac{-2i}{(z-(1+i))} + \frac{2i}{(z-(1-i))}.$$

Для элементарной дроби $\frac{1}{(z-(1+i))}$, согласно (Д.29), соответствующая последовательность имеет вид $b_n = -\frac{1}{(1+i)^{n+1}}$, а для дроби $\frac{1}{(z-(1-i))}$, соответствующая последовательность $(c_0, c_1, c_2, \ldots, c_n, \cdots)$ выражается соотношением

$$c_n = -\frac{1}{(1-i)^{n+1}}.$$

Тогда

$$a_n = -2ib_n + 2ic_n = 2i\left(\frac{1}{(1+i)^{n+1}} - \frac{1}{(1-i)^{n+1}}\right) = \frac{i}{2^n}((1-i)^{n+1} - (1+i)^{n+1}).$$

Заметим, что

$$1+i = \sqrt{2}(\cos\frac{\pi}{4} + i\sin\frac{\pi}{4}), \qquad 1-i = \sqrt{2}\left(\cos\left(-\frac{\pi}{4}\right) + i\sin\left(-\frac{\pi}{4}\right)\right),$$

тогда

$$(1+i)^{n+1} = 2^{\frac{n+1}{2}}\left(\cos(n+1)\frac{\pi}{4} + i\sin(n+1)\frac{\pi}{4}\right),$$
$$(1-i)^{n+1} = 2^{\frac{n+1}{2}}\left(\cos(n+1)\left(-\frac{\pi}{4}\right) + i\sin(n+1)\left(-\frac{\pi}{4}\right)\right).$$

Следовательно,

$$a_n = \frac{i}{2^n}2^{\frac{n+1}{2}}\left(-2i\sin(n+1)\left(\frac{\pi}{4}\right)\right) = 2^{\frac{3-n}{2}}\sin(n+1)\left(\frac{\pi}{4}\right).$$

Производящие функции и рекуррентные соотношения. До этого момента мы не использовали аппарат производящих функций для решения других задач. Покажем на примерах, как можно использовать этот аппарат для решения рекуррентных соотношений.

Пример Д.5.32. Найти a_n для рекуррентного соотношения

$$\begin{cases} a_0 = 0; \\ a_n = \frac{1}{2}a_{n-1} + \frac{1}{2^n}, \quad n \geqslant 1. \end{cases}$$

Решение. Введем производящую функцию $G(z) = \sum\limits_{n=0}^{\infty} a_n z^n$. Обе части рекуррентного соотношения

$$a_n = \frac{1}{2}a_{n-1} + \frac{1}{2^n}$$

умножим на z^n и просуммируем по n от 1 до ∞, в результате получаем

$$\sum_{n=1}^{\infty} a_n z^n = \frac{1}{2}\sum_{n=1}^{\infty} a_{n-1} z^n + \sum_{n=1}^{\infty} \frac{1}{2^n} z^n,$$

поскольку $a_0 = 0$, то

$$\sum_{n=1}^{\infty} a_n z^n = \sum_{n=0}^{\infty} a_n z^n = G(z).$$

Преобразуем суммы в правой части, имеем

$$\sum_{n=1}^{\infty} a_{n-1} z^n = \sum_{n=0}^{\infty} a_n z^{n+1} = z\sum_{n=0}^{\infty} a_n z^n = zG(z),$$

$$\sum_{n=1}^{\infty} \frac{1}{2^n} z^n = \sum_{n=1}^{\infty} \left(\frac{z}{2}\right)^n = \frac{z}{2}\cdot\frac{1}{1-\frac{z}{2}} = -\frac{z}{z-2},$$

как сумма геометрической прогрессии. Тогда мы можем определить $G(z)$

$$G(z) = \frac{1}{2} zG(z) - \frac{z}{z-2},$$

следовательно,

$$G(z) = \frac{2z}{(z-2)^2}.$$

Для элементарной дроби $\frac{1}{(z-2)^2}$, согласно (Д.30), соответствующая последовательность имеет вид

$$b_n = \frac{1}{2^{n+2}} \cdot C_{n+1}^1 = \frac{n+1}{2^{n+2}}.$$

Поскольку умножение на z сдвигает последовательность на единицу вправо, то мы получаем при $n \geqslant 1$

$$a_n = 2b_{n-1} = \frac{n}{2^n}.$$

Отметим, что поскольку $a_0 = 0$, то эта формула верна и при $n = 0$.

Пример Д.5.33. Решить рекуррентное соотношение

$$\begin{cases} a_0 = 0; \\ a_1 = 1; \\ a_n = \frac{3}{4} a_{n-1} - \frac{1}{8} a_{n-2} + \frac{9}{32} n + \frac{3}{8}, \quad n \geqslant 2. \end{cases}$$

Решение. Введем производящую функцию $G(z) = \sum\limits_{n=0}^{\infty} a_n z^n$. Обе части рекуррентного отношения умножим на z^n и просуммируем по n в пределах $2 \leqslant n < \infty$.

$$\sum_{n=2}^{\infty} a_n z^n = \frac{3}{4}\sum_{n=2}^{\infty} a_{n-1} z^n - \frac{1}{8}\sum_{n=2}^{\infty} a_{n-2} z^n + \frac{9}{32}\sum_{n=2}^{\infty} n z^n + \frac{3}{8}\sum_{n=2}^{\infty} z^n.$$

Заметим, что

$$\sum_{n=2}^{\infty} a_n z^n = \sum_{n=0}^{\infty} a_n z^n - a_0 - a_1 z = G(z) - z;$$

$$\sum_{n=2}^{\infty} a_{n-1} z^n = \sum_{n=1}^{\infty} a_n z^{n+1} = \sum_{n=0}^{\infty} a_n z^{n+1} - a_0 z = zG(z);$$

$$\sum_{n=2}^{\infty} a_{n-2} z^n = \sum_{n=0}^{\infty} a_n z^{n+2} = z^2 G(z);$$

$$\sum_{n=2}^{\infty} n z^n = z \sum_{n=2}^{\infty} n z^{n-1} = z \frac{d}{dz} \left(\sum_{n=2}^{\infty} z^n \right) = z \frac{d}{dz} \left(\frac{z^2}{1-z} \right) = \frac{-z^3 + 2z^2}{(1-z)^2};$$

$$\sum_{n=2}^{\infty} z^n = \frac{z^2}{1-z}.$$

Тогда для определения функции $G(z)$ имеем уравнение

$$G(z) - z = \frac{3}{4} z G(z) - \frac{1}{8} z^2 G(z) - \frac{9(z^3 - 2z^2)}{32(1-z)^2} + \frac{3z^2}{8(1-z)},$$

тогда

$$G(z) = \frac{11z^3 - 34z^2 + 32z}{4(z-1)^2(z^2 - 6z + 8)} = \frac{11z^3 - 34z^2 + 32z}{4(z-1)^2(z-2)(z-4)}.$$

Разложим это выражение на элементарные дроби

$$\frac{11z^3 - 34z^2 + 32z}{(z-1)^2(z-2)(z-4)} = \frac{A}{(z-1)^2} + \frac{B}{(z-1)} + \frac{C}{z-2} + \frac{D}{z-4}.$$

Определяем коэффициенты

$$11z^3 - 34z^2 + 32z = A(z-2)(z-4) + B(z-1)(z-2)(z-4) + \\ + C(z-1)^2(z-4) + D(z-1)^2(z-2).$$

Приравнивая коэффициенты при одинаковых степенях z, получаем

$$\begin{cases} B + C + D = 11; \\ A - 7B - 6C - 4D = -34; \\ -6A + 15B + 9C + 5D = 32; \\ 8A - 12B - 4C - 2D = 0. \end{cases}$$

Часть коэффициентов определим независимым путем. Полагая последовательно $z = 1$, $z = 2$ и $z = 4$ получаем $3A = 9$, $A = 3$; $-2C = 16$, $C = -8$; $18D = 288$, $D = 16$. Тогда $B = 3$. Следовательно,

$$G(z) = \frac{3}{4(z-1)^2} + \frac{3}{4(z-1)} - \frac{2}{z-2} + \frac{4}{z-4},$$

используя (Д.30), получаем

$$a_n = \frac{3}{4}(n+1) - \frac{3}{4} + \left(\frac{1}{2}\right)^n - \left(\frac{1}{4}\right)^n = \frac{3}{4}n + \frac{1}{2^n} - \frac{1}{4^n}.$$

Пример Д.5.34. Используя производящие функции получить общий член последовательности для чисел Фибоначчи. Числа Фибоначчи могут быть заданы как рекуррентная последовательность вида

$$\begin{cases} a_0 = 0; \\ a_1 = 1; \\ a_n = a_{n-1} + a_{n-2}, \quad n \geqslant 2. \end{cases}$$

Решение. Введем производящую функцию $G(z) = \sum\limits_{n=0}^{\infty} a_n z^n$, умножим обе части рекуррентного отношения на z^n и просуммируем по n в пределах $2 \leqslant n < \infty$, в результате получим

$$\sum_{n=2}^{\infty} a_n z^n = \sum_{n=2}^{\infty} a_{n-1} z^n + \sum_{n=2}^{\infty} a_{n-2} z^n.$$

Рассматривая полученные суммы в отдельности, имеем:

$$\sum_{n=2}^{\infty} a_n z^n = \sum_{n=0}^{\infty} a_n z^n - a_0 - a_1 z = G(z) - z,$$

$$\sum_{n=2}^{\infty} a_{n-1} z^n = \sum_{n=1}^{\infty} a_n z^{n+1} = \sum_{n=0}^{\infty} a_n z^{n+1} - a_0 z = z \sum_{n=0}^{\infty} a_n z^n = zG(z),$$

$$\sum_{n=2}^{\infty} a_{n-2} z^n = \sum_{n=0}^{\infty} a_n z^{n+2} = z^2 G(z).$$

На основании этого получаем уравнение для производящей функции

$$G(z) - z = zG(z) + z^2 G(z),$$

$$G(z) = \frac{-z}{z^2 + z - 1} = \frac{-z}{\left(z - \frac{-1+\sqrt{5}}{2}\right)\left(z - \frac{-1-\sqrt{5}}{2}\right)} =$$

$$= \frac{1}{\sqrt{5}} \cdot \left(- \frac{\frac{-1+\sqrt{5}}{2}}{z - \frac{-1+\sqrt{5}}{2}} + \frac{\frac{-1-\sqrt{5}}{2}}{z - \frac{-1-\sqrt{5}}{2}} \right),$$

тогда

$$a_n = \frac{1}{\sqrt{5}} \left(\frac{2^n}{\left(-1+\sqrt{5}\right)^n} - \frac{2^n}{\left(-1-\sqrt{5}\right)^n} \right) = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n.$$

Свертки последовательностей. Производящие функции обладают еще одним интересным свойством. Пусть даны две последовательности $A = (a_0, a_1, a_2, \ldots, a_n, \cdots)$ и $B = (b_0, b_1, b_2, \ldots, b_n, \cdots)$, производящие функции которых равны $G_1(z)$ и $G_2(z)$ соответственно. Тогда сверткой $C = A * B$ данных последовательностей называется последовательность $C = (c_0, c_1, c_2, \ldots, c_n, \cdots)$, такая что

$$c_n = \sum_{k=0}^{n} a_k b_{n-k}.$$

Производящая функция $G(z)$ для последовательности C равна

$$G(z) = G_1(z) \cdot G_2(z).$$

Отметим интересный факт. Если последовательность $B = (b_0, b_1, b_2, \ldots, b_n, \cdots)$ такова, что $b_n = 1$ для всех значений n, то свертка $C = A * B$ представляет последовательность частичных сумм последовательности $A = (a_0, a_1, a_2, \ldots, a_n, \cdots)$, т. е.

$$c_n = \sum_{k=0}^{n} a_k.$$

Поскольку для последовательности B, состоящей из единиц, производящая функция $G_2(z)$ равна $G_2(z) = \frac{1}{1-z}$, то, следовательно, производящая функция $G(z)$, для последовательности C — частичных сумм последовательности A, равна

$$G(z) = \frac{1}{1-z} \cdot G_1(z).$$

Используем этот факт для вычисления суммы квадратов натуральных чисел. Рассмотрим последовательность A: $(a_n = n^2)$. Наша цель — найти производящую функцию $G_1(z)$ для последовательности A, такую, что

$$G_1(z) = \sum_{n=0}^{\infty} n^2 z^n.$$

Имеем

$$\begin{aligned} G_1(z) &= \sum_{n=0}^{\infty} n^2 z^n = z \sum_{n=1}^{\infty} n^2 z^{n-1} = z \frac{d}{dz}\left(\sum_{n=1}^{\infty} n z^n\right) = z \frac{d}{dz}\left(z \sum_{n=1}^{\infty} n z^{n-1}\right) = \\ &= z \frac{d}{dz}\left(z \frac{d}{dz}\left(\sum_{n=1}^{\infty} z^n\right)\right) = z \frac{d}{dz}\left(z \frac{d}{dz}\left(\frac{z}{1-z}\right)\right) = z \frac{d}{dz}\left(\frac{z}{(1-z)^2}\right) = \\ &= \frac{z+z^2}{(1-z)^3}. \end{aligned}$$

Тогда для последовательности $C = A * B$, $B = (1, 1, \ldots, 1)$ производящая функция имеет вид

$$G(z) = \frac{1}{1-z} \cdot G_1(z) = \frac{z+z^2}{(1-z)^4}.$$

Разложим это выражение на элементарные дроби

$$G(z) = \frac{z+z^2}{(z-1)^4} = \frac{A}{(z-1)^4} + \frac{B}{(z-1)^3} + \frac{C}{(z-1)^2} + \frac{D}{(z-1)},$$

и после известных читателю преобразований получаем

$$z + z^2 = A + B(z-1) + C(z-1)^2 + D(z-1)^3,$$

тогда система уравнений для определения коэффициентов имеет вид

$$\begin{cases} D = 0; \\ C - 3D = 1; \\ B - 2C + 3D = 1; \\ A - B + C - D = 0. \end{cases}$$

Решая эту систему получаем: $D = 0$, $C = 1$, $B = 3$, $A = 2$, таким образом

$$G(z) = \frac{2}{(z-1)^4} + \frac{3}{(z-1)^3} + \frac{1}{(z-1)^2}.$$

Используя соотношение (Д.29), получаем

$$s_n = \sum_{k=0}^{n} k^2 = 2 \cdot C_{n+3}^3 - 3 \cdot C_{n+2}^2 + C_{n+1}^1 = \frac{n(n+1)(2n+1)}{6}.$$

Комбинаторные подсчеты и производящие функции. В заключение приведем примеры использования аппарата производящих функций для доказательства некоторых свойств биномиальных коэффициентов и решения некоторых задач комбинаторики. В начале уточним наше понимание биномиальных коэффициентов C_n^k — числа сочетаний из n по k. Будем считать, что $n \geqslant 0$, тогда

$$\begin{cases} C_n^k = 0, & k < 0,\ k > n; \\ C_n^k = \dfrac{n!}{k!(n-k)!}, & 0 \leqslant k \leqslant n. \end{cases}$$

Ранее мы показали, что производящая функция $(1+z)^n$ соответствует последовательности $(a_0, a_1, a_2, \ldots, a_n, \ldots)$, где $a_k = C_n^k$. Отметим, что при $k > n$ все члены последовательности равны нулю. Рассмотрим два положительных целых числа m и r, таких что $m + r = n$.

Производящей функции $G_1(z) = (1+z)^m$ соответствует последовательность $(b_0, b_1, b_2, \ldots, b_k, \ldots)$, где $b_k = C_m^k$, а производящей функции $G_2(z) = (1+z)^r$ соответствует последовательность $(c_0, c_1, c_2, \ldots, c_k, \ldots)$, где $c_k = C_r^k$.

Произведению этих функций соответствует функция

$$G(z) = G_1(z) \cdot G_2(z) = (1+z)^m (1+z)^r = (1+z)^{m+r} = (1+z)^n.$$

Этой производящей функции соответствует последовательность, в которой $a_k = C_n^k$. С другой стороны последовательность $(a_0, a_1, a_2, \ldots, a_k, \ldots)$ является сверткой последовательностей $(b_0, b_1, b_2, \ldots, b_k, \ldots)$ и $(c_0, c_1, c_2, \ldots, c_k, \ldots)$, следовательно, для любого k имеем $a_k = \sum\limits_{l=0}^{k} b_l c_{k-l}$. Таким образом, имеем равенство

$$C_n^k = \sum_{l=0}^{k} C_m^l \cdot C_r^{k-l},$$

так как $n = r + m$, то это равенство можно записать в виде

$$C_{r+m}^k = \sum_{l=0}^{k} C_m^l \cdot C_r^{k-l},$$

это соотношение называется правилом свертки Вандермонда.

Рассмотрим еще один пример. Производящей функции $G_1(z) = (1+z)^n$ соответствует последовательность $(b_0, b_1, b_2, \ldots, b_n, \ldots)$, где $b_k = C_n^k$. Производящей функции $G_2(z) = (1-z)^n$ соответствует последовательность $(c_0, c_1, c_2, \ldots, c_k, \ldots)$, где $c_k = (-1)^k C_n^k$.

Произведению этих функций соответствует функция

$$G(z) = G_1(z)G_2(z) = (1+z)^n(1-z)^n = (1-z^2)^n.$$

Этой производящей функции соответствует последовательность $(a_0, a_1, a_2, \ldots, a_k, \ldots)$, где

$$a_{2m} = (-1)^m C_n^m, \quad a_{2m+1} = 0.$$

Поскольку

$$a_k = \sum_{l=0}^{k} b_l c_{k-l},$$

то получаем следующее свойство для a_{2m}

$$a_{2m} = \sum_{l=0}^{2m} b_l \cdot c_{2m-l} = \sum_{l=0}^{2m} C_n^l \cdot (-1)^{2m-l} \cdot C_n^{2m-l},$$

и окончательно получаем следующее соотношение для биномиальных коэффициентов

$$(-1)^m \cdot C_n^m = \sum_{l=0}^{2m} (-1)^l \cdot C_n^l \cdot C_n^{2m-l}.$$

Теперь рассмотрим примеры применения производящих функций для решения комбинаторных задач.

Пример Д.5.35. Сколько существует способов выдать сумму в 12 копеек, используя монеты достоинством только в 1 и 5 копеек.

Решение. Число способов выдать сумму в k копеек однокопеечными монетами образует последовательность $(b_0, b_1, b_2, \ldots, b_k, \ldots)$, где $b_k = 1$. Производящая функция этой последовательности $G_1(z)$ имеет вид

$$G_1(z) = 1 + z + z^2 + \cdots + z^k + \cdots = \frac{1}{1-z}.$$

Для монет достоинством в 5 копеек соответствует последовательность $(c_0, c_1, c_2, \ldots, c_k, \ldots)$, где $c_k = 1$, если k кратно 5 и $c_k = 0$ в противном случае. Производящая функция этой последовательности $G_2(z)$ имеет вид

$$G_2(z) = 1 + z^5 + z^{10} + \cdots + z^{5k} + \cdots = \frac{1}{1-z^5}.$$

Тогда последовательность числа способов для сумм, которые можно выдать однокопеечными и пятикопеечными монетами является сверткой данных последовательностей и, значит, ее производящая функция $G(z)$ равна

$$G(z) = G_1(z) \cdot G_2(z) = \frac{1}{1-z} \cdot \frac{1}{1-z^5}.$$

Тогда число способов, которыми можно выдать сумму в n копеек равно коэффициенту при z^n. Найдем этот коэффициент для $n = 12$. Для нахождения этого

коэффициента воспользуемся представлением функций в виде ряда. Нетрудно заметить, что z^{12} можно представить тремя способами:

$$z^{12} = z^{12} \cdot 1, z^{12} = z^7 \cdot z^5, \quad z^{12} = z^2 \cdot z^{10},$$

следовательно, всего существует три способа.

Набор упражнений Д.5.4

Д.5.4.1. Найти последовательности, которые соответствуют следующим производящим функциям $G(z)$:

(а) $G(z) = \dfrac{1}{4 - z^2}$; (в) $G(z) = \dfrac{z}{z^2 - 6z + 8}$;

(б) $G(z) = \dfrac{1}{9 + z^2}$; (г) $G(z) = \dfrac{1}{(z-1)^2(2z+1)}$.

Д.5.4.2. С помощью производящих функций решить следующие рекуррентные отношения:

(а) $\begin{cases} a_0 = 2; \\ a_n = 3a_{n-1} - 2n + 3, \quad n > 0. \end{cases}$

(б) $\begin{cases} a_0 = 2; \\ a_1 = 6; \\ a_n = 6a_{n-1} - 8a_{n-2} + 3n^2 - 23n + 36, \quad n > 1. \end{cases}$

Д.5.4.3. Имеются числа 1, 2, 3, 4, 5. Сколькими способами, суммируя эти числа, можно получить 6, если каждое слагаемое можно использовать не более одного раза. Порядок суммирования не играет роли.

Д.6. Общая проблема перебора и некоторые точные методы решения задач целочисленного программирования

Введение

Предположим, что у Вас есть монеты номиналом 2, 3, 5, 6 и 7 флоринов, по одной каждого номинала, и Вы хотите без сдачи оплатить покупку стоимостью 21 флорин. Можно ли набрать из данных монет сумму равную 21? А если понравившийся Вам товар стоит 22 флорина? Мы сталкиваемся с задачей, в которой для получения ответа необходимо перебрать различные варианты — в случае, если Вы утверждаете, что задача не имеет решения, то для доказательства этого может быть необходимо перебрать вообще все возможные варианты. Более детальное рассмотрение этой задачи приводит к целому ряду вопросов: сколько вообще существует вариантов, какой компьютер необходим нам для решения этой задачи, является ли полный перебор единственно возможным алгоритмом

решения или мы можем предложить идею существенного сокращения перебора? Это простой пример очень «не простой» задачи перебора, которая известна как задача о сумме — можно ли получить из известного набора чисел путем суммирования некоторых из них определенное, заранее заданное число?

Дадим более строгую формулировку задачи о сумме. Каким образом мы можем описать решение, если оно есть? Классический подход таков: зафиксируем порядок следования исходных чисел, присвоив им номера — $A = (a_1, \ldots, a_n)$ и введем в рассмотрение упорядоченное множество $\mathbf{x} = (x_1, \ldots x_n)$, $x_i \in \{0, 1\}$. Значение $x_i = 1$ говорит о том, что число a_i с номером i включено в сумму, например, $\mathbf{x} = (0, 1, 1, 1, 1)$ есть описание решения для суммы в 21 флорин. Мы можем рассматривать элементы множества $\mathbf{x}$ как значения координат некоторой точки в целочисленном n- мерном пространстве, или как координаты вектора в этом пространстве. Таким образом, задача сводится к поиску такой точки (вектора) $\mathbf{x}$ с целочисленными $(0, 1)$ координатами в n- мерном пространстве, в которой целевая функция $f(\mathbf{x})$ удовлетворяет определенным условиям.

При таком подходе задача о сумме формулируется теперь следующим образом. Дано упорядоченное множество $A = (a_1, \ldots, a_n)$ и число V. Пусть точка $\mathbf{x} = (x_1, \ldots, x_n)$, множество $X = \{\mathbf{x} |\ x_i \in \{0, 1\}\}$, и целевая функция имеет вид

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot a_i,$$

необходимо найти одну или все точки $\mathbf{x} \in X$ такие, что $f(\mathbf{x}) = V$ или доказать, что решения нет.

Заметим также, что теперь мы можем ответить и на вопрос о том, сколько существует различных вариантов суммирования. Если интерпретировать компоненты вектора $\mathbf{x}$ как значения битов двоичного числа, то становится ясно, что полный перебор требует исследования 2^n вариантов, именно столько чисел мы можем записать, используя n битов, или более корректно — $|X| = 2^n$. В таком объеме полного перебора задача не решается за реальное время даже при небольших значениях n. Приведем приближенный расчет. Пусть на операцию сложения компьютер затрачивает 10 тактов (это достаточно реальное значение), и мы решаем задачу о сумме из 40 чисел. Для сложения 40 чисел и проверки будет затрачено не менее 400 тактов, тогда процессор с частотой 4 Ггц позволяет перебрать 10 млн. вариантов в секунду. Число $2^{40} \approx 10^{12}$, таким образом, мы получаем ожидаемое время порядка 100 000 сек., что составляет около 28 часов, т. е. более суток. Заметьте, что для задачи о сумме из 41 числа время будет увеличено вдвое.

Такого рода задачи называются «задачами перебора» и их исследование привело в 1960-х годах к важным результатам в области теории алгоритмов и разработке специального раздела современной дискретной математики, изучающего задачи целочисленного программирования, методы и алгоритмы их решения. Введению в данную проблематику и посвящен данный раздел дополнения.

Д.6.1. Понятие m-мерного евклидова целочисленного пространства

При изложении задач целочисленного программирования удобно, как это мы уже видели, использовать геометрическую терминологию, обобщающую наше представление о трехмерном пространстве. Поскольку в рассматриваемых задачах результатом является набор целых, как правило, неотрицательных чисел, то нас будет интересовать многомерное целочисленное пространство. Вначале опишем классическое m-мерное евклидово пространство [18].

Будем назвать *m-мерным координатным пространством* множество упорядоченных совокупностей $(x_1, x_2, \ldots, x_m)$, состоящих из m вещественных чисел $x_1, x_2, \ldots, x_m$, и обозначать это пространство символом A^m. Каждую такую упорядоченную совокупность будем называть *точкой* m-мерного координатного пространства, обозначая ее через $\mathbf{x}$, а числа $x_1, x_2, \ldots, x_m$ будем называть *координатами точки* $\mathbf{x}$.

Если рассматривать m-мерное координатное пространство как множество *векторов* $\mathbf{x}$, ввести понятие суммы векторов как вектор с покоординатной суммой и определить произведение вектора на вещественное число как покоординатное произведение, то мы получим m-мерное линейное пространство.

Координатное пространство A^m называется *m-мерным евклидовым пространством*, если между любыми двумя точками $\mathbf{x}$, $\mathbf{y}$ m-мерного пространства A^m определено расстояние, обозначаемое символом $\rho(\mathbf{x}, \mathbf{y})$ и выражающееся соотношением

$$\rho(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}. \qquad (Д.33)$$

Общеупотребительным для m-мерного евклидова пространства является обозначение E^m. Пространство, в котором указано правило, ставящее в соответствие любым двум элементам $\mathbf{x}, \mathbf{y}$ вещественное число, называемое расстоянием между этими элементами — $\rho(\mathbf{x}, \mathbf{y})$, которое удовлетворяет следующим трем аксиомам:

1) $\rho(\mathbf{x}, \mathbf{y}) = \rho(\mathbf{y}, \mathbf{x})$;
2) $\rho(\mathbf{x}, \mathbf{y}) \leqslant \rho(\mathbf{x}, \mathbf{z}) + \rho(\mathbf{z}, \mathbf{y})$;
3) $\rho(\mathbf{x}, \mathbf{y}) \geqslant 0$, $\rho(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$,

называется *метрическим пространством*. Легко проверить, что расстояние, введенное формулой (Д.33) удовлетворяет этим аксиомам, и, следовательно, пространство E^m является метрическим пространством.

Подробные сведения по общей топологии и аксиоматике линейных, нормированных и метрических пространств содержатся, например, в [18] и [27].

Но пространство E^m содержит точки, координаты которых являются вещественными числами. В целях формализации задач целочисленного программирования нам необходимо подмножество пространства E^m, содержащее только те точки из E^m, которые имеют целочисленные координаты. Будем называть это подмножество *m-мерным евклидовым целочисленным пространством*, и обозначать его через E_z^m:

$$E_z^m = \{\mathbf{x} |\ \mathbf{x} \in E^m,\ \mathbf{x} = (x_1, x_2, \ldots, x_m), \quad x_i \in \mathbb{Z}\ \forall i = \overline{1, m}\},$$

где $\mathbb{Z}$ — множество целых чисел.

Ограничения, возникающие в задачах целочисленного программирования, приводят к тому, что мы рассматриваем не всё множество E_z^m, а некоторую ограниченную совокупность его точек — в общем случае — некоторый многогранник с вершинами, имеющими целочисленные координаты. В связи с этим введем понятие m-мерного координатного куба. Множество Y всех точек $\mathbf{y}$ из E_z^m, координаты $(y_1, y_2, \ldots, y_m)$ которых удовлетворяют неравенствам

$$y_1 - x_1 \leqslant k,\ y_2 - x_2 \leqslant k,\ \ldots,\ y_m - x_m \leqslant k, \quad y_i \geqslant x_i\ \forall i = \overline{1, m}, \qquad (\text{Д.34})$$

будем называть *m-мерным координатным кубом* с ребром k с началом в точке $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, $\mathbf{x} \in E_z^m$, и обозначать его через $Cub_z^m(\mathbf{x}, k)$.

Поскольку многие задачи целочисленного программирования приводят к перебору точек m-мерного координатного куба с началом в точке $\mathbf{0} = (0, \ldots, 0)$, т. е. куба $Cub_z^m(\mathbf{0}, k)$, то представляется целесообразным ввести специальное обозначение: $k_z^m = Cub_z^m(\mathbf{0}, k)$. Таким образом, например, 1_z^m — это m-мерный координатный куб с началом в нуле, все точки которого имеют целочисленные координаты — 0 или 1. Мы уже знакомы с этим множеством точек — в рамках перебора точек из 1_z^n формулируется известная читателю задача о сумме n чисел.

Аналогично, множество всех точек Y из E_z^m, координаты $(y_1, y_2, \ldots, y_m)$ которых удовлетворяют равенству $\rho(\mathbf{x}, \mathbf{y}) = r$, будем называть *$m$-мерной сферой* радиуса r с центром в точке $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, $\mathbf{x} \in E_z^m$, и обозначать ее через $S_z^m(\mathbf{x}, r)$.

Сколько целочисленных точек содержит m-мерный координатный куб? Ответ на этот вопрос является важным, так как позволяет указать верхнюю оценку границы размерности задачи целочисленного программирования, если мы пытаемся решить ее путем полного перебора точек. Поскольку в силу условия (Д.34) каждая координата точки куба может принимать не более чем $k+1$ целочисленных значений, и мы имеем m независимых координат для точки в пространстве E_z^m, то приходим к выводу, что $|Cub_z^m(\mathbf{x}, k)| = (k+1)^m$ вне зависимости от начальной точки куба. В частности, даже куб 1_z^m содержит 2^m точек. Этот неутешительный результат и заставляет искать эффективные алгоритмы решения задач целочисленного программирования, существенно сокращающие перебор.

Д.6.2. Общая постановка, типизация и примеры задач целочисленного программирования

Прежде чем перейти к описанию общей постановки задачи целочисленного программирования, авторы дополнения хотели бы сделать замечание о том, что они принимают расширенное толкование термина «задача целочисленного программирования». Под этим термином авторы понимают задачу нахождения экстремума любой вычислимой вещественнозначной функции, определенной в некоторой области m-мерного евклидова целочисленного пространства при ограничениях (условиях) общего вида, задающих такое сужение области определения функции, которое заключено в некоторый конечный m-мерный целочисленный координатный куб.

Общая постановка задачи. Пусть задано подмножество G в целочисленном пространстве E_z^m. На точках $\mathbf{x}$ подмножества G определена вещественнозначная функция $f(\mathbf{x}) : G \to R$, где R — множество действительных чисел. Эту функцию мы будем в дальнейшем, следуя [27], называть *целевым функционалом задачи*. В литературе встречаются также равно положенные термины — показатель качества решения задачи, функция потерь, критерий эффективности и т. д.

Пусть заданы также ограничения (неравенства) на значения вещественнозначных функций $g_i(\mathbf{x}) : G \to R$, $i = \overline{1,k}$ в виде

$$g_i(\mathbf{x}) \leqslant a_i, \quad a_i \in R, \quad i = \overline{1,k},$$

тогда общая постановка задачи целочисленного программирования имеет вид

$$f(\mathbf{x}) \to \min, \quad \text{при} \quad g_i(\mathbf{x}) \leqslant a_i, \quad a_i \in R, \quad i = \overline{1,k}, \quad \mathbf{x} \in G. \tag{Д.35}$$

В задаче требуется найти такой вектор $\mathbf{x}$, удовлетворяющий ограничениям — неравенствам, и принадлежащий подмножеству G, который доставляет минимум целевому функционалу $f(\mathbf{x})$. Решить задачу (Д.35) точно — значит найти какой-нибудь или все векторы $\mathbf{x}$, удовлетворяющие (Д.35), или доказать, что задача несовместна.

Типизация задач целочисленного программирования. Алгоритмы и методы решения задач вида (Д.35) существенно зависят от того, что представляет собой область G, какой вид имеет целевой функционал и какова структура функций ограничений. По структуре целевого функционала, области определения и функциям ограничения имеет место следующая типизация задач целочисленного программирования:

- задачи *линейного* целочисленного программирования, в которых $f(\mathbf{x})$ и $g_i(\mathbf{x})$ являются линейными функционалами;
- задачи *нелинейного* целочисленного программирования, в которых хотя бы один из функционалов $f(\mathbf{x})$ или $g_i(\mathbf{x})$ является нелинейным, достаточно часто возникающие в различных прикладных областях.

Из нелинейных задач наиболее полно разработан класс задач выпуклого программирования, в которых множество G является выпуклой областью, а целевой функционал и функции ограничений — выпуклые функции. В этом классе выделяются также задачи квадратичного выпуклого целочисленного программирования.

Напомним, что множество точек в пространстве E_m называется *выпуклым*, если вместе с любой парой своих точек, оно содержит и весь соединяющий их отрезок. Непрерывная функция $f(x)$, определенная на выпуклом множестве G называется выпуклой вниз, если для всех $x, y \in G$ справедливо неравенство

$$\forall \alpha, \quad 0 \leqslant \alpha \leqslant 1, \quad f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y).$$

Приведем некоторые примеры задач целочисленного программирования.

Задача оптимального резервирования. Пусть некоторая система состоит из n основных элементов, относящихся к n разным типам. Каждый элемент может быть резервирован в целях повышения надежности функционирования всей системы в целом. Пусть значение x_i есть число элементов i- ого типа, включенных в систему, $x_i \geqslant 1$, $i = \overline{1,n}$. Один из этих элементов находится в штатном режиме функционирования, остальные находятся в резерве. Мы предполагаем, что заданы функции $f_i(x_i)$, дающие значение показателя надежности системы по элементам типа i, в зависимости от их количества. Если известны вероятности отказа элемента типа i — p_i, то функция $f_i(x_i)$ может иметь, например, следующий вид

$$f_i(x_i) = 1 - p_i^{x_i}.$$

Мы хотим максимизировать целевой функционал, отражающий показатель надежности системы в целом, при условии, что существует ограничение на суммарные затраты. В предположении, что показатель надежности системы есть произведение показателей надежности по типам элементов, а значение c — есть заданное ограничение на затраты, задача оптимального резервирования есть задача определения числа резервных элементов каждого типа, максимизирующих нелинейный целевой функционал при заданных ограничениях

$$\prod_{i=1}^{n} f_i(x_i) \to \max, \quad \sum_{i=1}^{n} a_i \cdot x_i \leqslant c, \quad \mathbf{x} = (x_1, \ldots, x_n) \in E_z^n, \quad x_i \geqslant 1.$$

Отметим, что особенностью данной задачи является сепарабельность целевого функционала и ограничений — изменение показателя надежности и показателя затрат при изменении значений по одному типу элементов не зависит от значений остальных переменных. Иными словами, изменение значения x_i не влияет на другие значения x_j, $j \neq i$. Задачи с сепарабельными функционалами и ограничениями могут быть решены методом динамического программирования, который мы рассмотрим позднее.

Задача оптимального снабжения. Нам необходимо организовать рациональное снабжение предприятия, занимающегося сборкой, комплектующими деталями в контейнерах, которые изготавливаются и доставляются из других городов. Мы предполагаем, что в снабжении предприятия участвуют n городов. Пусть x_i — есть количество контейнеров, поставляемых из i-ого города, а c_i — суммарная стоимость производства и доставки одного контейнера деталей из города i, $i = \overline{1,n}$. Тогда стоимость всех привезенных контейнеров задается линейной функцией

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i \cdot x_i.$$

Потребность предприятия в комплектующих деталях определяется величиной b (контейнеров), что приводит к первому ограничению

$$\sum_{i=1}^{n} x_i = b.$$

Производство деталей в городе i ограничено величиной b_i, а обратные перевозки исключены, что приводит к очевидным ограничениям. В этих предположениях постановка задачи имеет вид

$$f(\mathbf{x}) \to \min, \quad \sum_{i=1}^{n} x_i = b, \quad 0 \leqslant x_i \leqslant b_i, \quad i = \overline{1,n}, \quad \mathbf{x} \in E_z^n.$$

В классической непрерывной постановке ($\mathbf{x} \in E^n$) — это просто решаемая задача линейного программирования, мы обращаем ее в задачу целочисленного линейного программирования путем помещения товаров в контейнеры, которые являются минимальными единицами перевозки. Заметим, что область перебора — это n-мерный координатный параллелепипед с началом в нуле, и ребрами длины b_i.

Оптимизация многопроцессорного расписания. Приведем еще одну постановку задачи, более близкую и понятную программистам. Суть задачи сводится к составлению оптимального расписания работы многопроцессорной системы.

Пусть система состоит из m неоднородных процессоров, по которым необходимо распределить n заданий, причем $n > m$. Если такое распределение есть, то каждый процессор последовательно выполняет указанные ему задания. Мы заранее знаем время выполнения каждого задания на каждом процессоре, или имеем его оценку. Пусть t_{ij} — есть время выполнения задания i на процессоре j. Наша цель — минимизировать суммарное время решения всего набора заданий.

Рассмотрим вектор $\mathbf{x}$ в пространстве E_z^n, и будем интерпретировать значение компоненты x_i этого вектора как номер процессора, на котором должно выполняться задание с номером i. Этот вектор и будет являться «расписанием» работы процессоров. Поскольку имеется m процессоров, и каждое задание должно быть назначено на какой-то процессор, то возникает очевидная система ограничений

$$1 \leqslant x_i \leqslant m, \quad i = \overline{1,n},$$

таким образом, областью G для этой задачи является координатный куб с началом в точке $\mathbf{1} = (1, 1, \ldots, 1)$ в n-мерном целочисленном пространстве с ребром $(m-1)$ — $Cub_z^n(\mathbf{1}, m-1)$.

Поскольку весь набор заданий считается выполненным, когда последний (по времени) процессор завершит выполнение последнего задания, то целевой функционал может быть записан в виде

$$f(\mathbf{x}) = \max_{j=1,m} \{ T_j \} \to \min,$$

где T_j — время работы процессора j, определяемое как

$$T_j = \sum_{i=1}^{n} [x_i = j] \cdot t_{ij},$$

где мы использовали нотацию Айверсона — $[k = l] = 1$, если $k = l$, иначе — 0.

Ниже мы рассмотрим и другие постановки, как иллюстрации к точным методам решения задач целочисленного программирования.

Д.6.3. NP-полные задачи и проблема перебора

Из общей постановки задачи целочисленного программирования следует, что ее решение, полным перебором точек в некотором подмножестве куба многомерного целочисленного пространства, требует временных затрат, экспоненциально зависящих от размерности пространства. Это, однако, не означает, что такова теоретическая оценка трудоемкости этих задач. Мы можем, например, сформулировать задачу сортировки чисел a_i, $i = \overline{1,n}$, как задачу целочисленного программирования, в которой каждая точка пространства E_z^n описывает перестановку этих чисел, а функционал имеет вид

$$f(\mathbf{x}) = \sum_{i=1}^{n} i \cdot v(x_i) \to \max, \quad 1 \leqslant x_i \leqslant n \quad \forall i = \overline{1,n}, \quad v(x_i) = a_{x_i}.$$

Но мы знаем, что задача сортировки решается с оценкой $\Theta(n \ln n)$. Можем ли мы все задачи целочисленного программирования решить так быстро? Получение теоретических оценок трудоемкости этих задач приводит к необходимости их рассмотрения с точки зрения теории сложности вычислений.

Исследование сложностных классов задач связано с введением в середине 1960-х годов класса P как класса задач, для которых известны полиномиальные алгоритмы решения. Более точно: задача относится к классу P, если существует алгоритм, правильно решающий эту задачу за время $O(n^k)$, где n — длина входа алгоритма, а k есть константа, не зависящая от n [20]. Интуитивно класс P — это класс задач, которые можно быстро решить, или класс реально решаемых задач. Принадлежность задачи к классу P позволяет положительно ответить на вопрос о границах практической применимости данного алгоритма решения задачи в смысле ограничений на ее размерность — мы можем получить решение задачи из класса P за приемлемое время даже для больших размерностей.

Представим себе, что некоторый алгоритм получает решение некоторой задачи. Мы вправе задать вопрос — соответствует ли полученный ответ поставленной задаче, и насколько быстро мы можем проверить его правильность? В рамках исследования этого вопроса в теории сложности рассматривается класс NP — класс быстро проверяемых задач. Задача относится к классу NP, если существует алгоритм, проверяющий полученное решение за время $O(n^k)$.

Покажем, что задача целочисленного программирования (ЦП) в общей постановке принадлежит этому классу. Следуя подходу теории сложности, будем рассматривать не оптимизационные постановки задачи ЦП, а постановки в виде задач разрешения

$$\begin{aligned} &\exists \mathbf{x}: \ f(\mathbf{x}) \leqslant C, \quad \mathbf{x} \in E_z^n, \text{ при условиях:} \\ &g_i(\mathbf{x}) \leqslant a_i, \quad a_i \in R, \quad i = \overline{1,k} \text{ и } \mathbf{x} \in G. \end{aligned} \tag{Д.36}$$

Таким образом, алгоритм решения задачи ЦП в такой постановке должен либо найти точку $\mathbf{x}$, удовлетворяющую (Д.36), либо определить, что такой точки нет. Если нам предъявлен ответ, и мы предполагаем, что вычисление значений $f(\mathbf{x})$ и $g_i(\mathbf{x})$ имеет сложность, полиномиально зависящую от n, то поскольку количество ограничений не зависит от размерности задачи, то сложность проверки

решения задачи (Д.36) полиномиальна, и, следовательно, задача ЦП принадлежит классу NP.

Каково соотношение классов P и NP? На сегодня отсутствуют теоретические доказательства, как совпадения этих классов (P = NP), так и их несовпадения. Предположение состоит в том, что класс P является собственным подмножеством класса NP. Это предположение опирается на существование еще одного класса, а именно класса NP-полных задач. В 1971 году Кук ввел понятие NP-полных задач как подкласса в NP — это задачи, к которым за время, полиномиальное относительно длины входа, могут быть сведены любые другие задачи из класса NP.

Определение класса NPC (*NP-complete*) требует выполнения следующих двух условий: во-первых, задача должна принадлежать классу NP, и, во-вторых, к ней полиномиально должны сводиться все задачи из класса NP. Исторически первое доказательство NP-полноты было предложено Куком для задачи о выполнимости схемы. Метод сведения за полиномиальное время был предложен Р. Карпом и использован им для доказательства NP-полноты целого ряда задач. Описание ряда NP-полных задач читатель может найти в [20], более подробное введение в теорию сложности вычислений содержится, например, в [22].

В настоящее время для NP-полных задач не найдены полиномиальные алгоритмы решения, и большинство специалистов полагают, что их вообще нельзя решить за полиномиальное время, но теоретически проблема P = NP остается открытой. NP-полные задачи, формулируемые как задачи оптимизации, перестают принадлежать к классу NP, т. к. оптимальность полученного решения нельзя проверить за полиномиальное время. Такие задачи называют NP-трудными задачами [27].

В теории сложности вычислений доказано, что в оптимизационной постановке целый ряд задач целочисленного программирования относится к NP-трудным [20], в частности, это задача о составлении расписаний и задача оптимальной упаковки, а задача о сумме — NP-полной. Поскольку на сегодня отсутствие полиномиальных алгоритмов решения NP-полных задач приводит к отрицательному ответу на проблему P = NP [20], то, скорее всего, нет и алгоритмов решения задач ЦП в оптимизационной постановке с полиномиальной сложностью относительно размерности вектора решения (длины входа алгоритма). Сегодня в научных публикациях продолжают предлагаться и исследоваться как эвристические, так и ε-полиномиальные алгоритмы [20] для решения разнообразных практических постановок задач целочисленного программирования [14], [9]. Другой интересный аспект изучения NP-полных задач состоит в рассмотрении их как задач в многомерном целочисленном пространстве, с последующей оценкой числа вершин полиэдрального графа задачи. Более подробную информацию читатель может получить, например, в [13] и [12].

В литературе часто встречается термин «задачи перебора», как аналог обозначения NP-полных задач. Возникновение этого термина в 1960-е годы можно видимо объяснить тем, что единственным очевидным точным методом решения таких задач является прямой перебор вариантов с экспоненциальной сложностью по размерности искомого вектора решения. Рассматриваемые ниже точные алгоритмы решения задач ЦП, принадлежащих классу NPC, в силу выше сказанного

также не являются полиномиальными. Эти методы позволяют в целом ряде практически важных постановок задач целочисленного программирования получить точное решение за приемлемое время путем существенного сокращения объема перебора.

Д.6.4. Обзор точных методов решения задач целочисленного программирования

Первые работы по целочисленному программированию были опубликованы в конце 1950-х — начале 1960-х годов. За прошедшее время было разработано достаточно много различных алгоритмов и методов, обеспечивающих как точное, так и приближенное решение целочисленных задач.

Мы уже знаем, что точное решение не может быть получено за полиномиальное время для тех задач целочисленного программирования, которые относятся к классу NPC. Но, оказывается, что построение приближенных методов (с гарантированной погрешностью) для многих классов дискретных задач является столь же трудоемкой проблемой, что и поиск точного решения [27]. Тем не менее, для ряда задач такие эффективные приближенные алгоритмы существуют. В частности, к таким относится уже известные читателю задачи о составлении расписания многопроцессорной системы, и об оптимальном резервировании элементов.

Отметим, что в последнее десятилетие широко разрабатываются новые интересные алгоритмические методы, основанные, в частности, на биологических аналогиях — мы имеем в виду генетические и муравьиные алгоритмы, успешно применяемые для решения задач целочисленного программирования. Небольшое введение в эти методы содержится в [22].

При достаточном разнообразии, точные алгоритмы решения могут быть описаны несколькими общими методами, универсального и ограниченного применения. Большинство универсальных алгоритмов основано на идеях метода отсечения и метода ветвей и границ. Алгоритмы ограниченного применения являются, в основном, модификациями метода динамического программирования. Ниже мы излагаем основные принципы этих методов, которые далее проиллюстрируем на конкретных задачах.

Метод полного перебора. Поскольку число возможных решений задачи целочисленного программирования ограничено сверху — мы можем гарантировать, что существует координатный куб, в который вписан многогранник, формируемый ограничениями задачи, то очевидным методом является полный перебор всех возможных вариантов — точек целочисленного пространства. В англоязычной литературе этот метод называется British museum technique — техника британского музея. Мы пытаемся решить задачи перебора методом перебора, но нас останавливает проблема размерности — как только размерность целочисленного пространства становится большой — перебор требует, хотя и конечного, но астрономического времени. Попробуйте, например, оценить, при каких значениях n ваш компьютер решит задачу перебора в кубе 1_z^n не более чем за один час.

Метод отсечений. Идея метода принадлежит Г. Данцигу и Р. Гомори [27]. Изначально метод был предложен для задач линейного целочисленного программирования. Суть идеи заключается в том, чтобы вначале пренебречь требованием целочисленности и попытаться решить соответствующую непрерывную задачу линейного программирования. Если оптимальное решение задачи непрерывного линейного программирования удовлетворяет требованию целочисленности, то тогда оно является одновременно и решением целочисленной задачи.

Если же на первом шаге получено нецелочисленное решение, то к исходным ограничениям добавляется новое линейное неравенство, которое формируется таким образом, что бы полученное нецелочисленное решение не удовлетворяло новому неравенству, а любое целочисленное решение заведомо удовлетворяло ему. Описанная процедура повторяется с новой задачей линейного программирования, система ограничений которой содержит на одно неравенство больше. Полученное решение вновь проверяется на целочисленность, и, при необходимости дополняется новым неравенством. Через некоторое число итераций, будет либо найдено целочисленное решение, либо очередная система неравенств окажется несовместной. Очевидно, что число итераций существенно зависит от способа формирования дополнительных ограничений. В общем виде метод может быть представлен следующим образом

Procedure *Метод отсечений*
Begin
1. *Решить непрерывную задачу линейного программирования с исходной системой ограничений*
While *(пока не найдено целочисленное решение или система ограничений несовместна)*
begin
2.1. *Сформировать новое ограничение, отсекающее полученное нецелочисленное решение.*
2.2. *Решить непрерывную задачу линейного программирования с новой системой ограничений.*
end *(while)*
End.

В терминах геометрии многомерного пространства добавление каждого нового линейного неравенства означает построение гиперплоскости, отсекающей от многогранного множества, сформированного исходными ограничениями задачи линейного программирования оптимальную, но нецелочисленную, вершину и сохраняющей все целочисленные точки области определения задачи. Эта аналогия и дала название методам, основанных на идеях Г. Данцига — «методы отсечений».

Метод ветвей и границ. Основная идея метода была предложена в работах А. Лэнда и А. Дойга по линейному целочисленному программированию в начале 1960-х годов. Метод стал широко известен благодаря детально разработанному и эффективному алгоритму Дж. Литла, К. Мерти, Д. Суини и К. Кэрол [7] для точного решения задачи коммивояжера, которая является NP-полной (класс NPC).

Основная идея метода состоит в попытке существенного сокращения перебора, за счет анализа подмножеств общего множества точек многогранника, подлежащих перебору. Наиболее важным этапом метода является выявление подмножеств точек, гарантированно не содержащих оптимального решения.

Метод основан на последовательном разбиении исходного множества точек на подмножества — этот процесс называется ветвлением, и вычислении оценок целевого функционала задачи (нижних при решении задачи на минимум) на выделенных подмножествах — этот процесс называется вычислением границ. Таким образом, строится поисковое дерево решений, и основная идея метода состоит в отсечении тех ветвей этого дерева, которые заведомо не содержат оптимального решения.

Проблема применения метода к конкретной задаче состоит в выборе стратегии ветвления и принципов построения оценок. В качестве простейшего способа вычисления оценок предлагается следующий подход — решить непрерывную задачу при ограничениях, соответствующих рассматриваемому подмножеству точек. Обычно стратегии ветвления и оценивания существенно опираются на специфику задачи. При удачно выбранных стратегиях сокращение перебора оказывается настолько существенно, что позволяет решить задачи практически значимых размерностей за приемлемое время, именно такими свойствами обладает указанный выше алгоритм решения задачи коммивояжера.

Метод динамического программирования. Метод динамического программирования был предложен и обоснован Р. Беллманом в начале 1960-х годов [11]. Условия применения метода требуют, чтобы целевой функционал представлял собой аддитивную функцию, т. е.

$$f(\mathbf{x}) = \sum_{i=1}^{n} g_i(x_i),$$

заметим, что требование аддитивности может быть ослаблено до требования сепарабельности [27]. Приведем описание идеи этого метода, опираясь на оригинальное изложение и терминологию его автора [11]. Экономическая интерпретация постановки задачи приводит к тому, что ограничение вида

$$\sum_{i=1}^{n} x_i = C,$$

рассматривается как ограничение на общий ресурс, который должен быть распределен по n процессам, приносящим доход, задаваемый функциями $g_i(x_i)$, $i = \overline{1,n}$ в условиях целочисленности и неотрицательности значений x_i. Для решения задачи максимизации целевого функционала $f(\mathbf{x}) \to \max$, вместо рассмотрения одной задачи с данным количеством ресурсов и фиксированным числом процессов, рассматривается целое семейство задач, в которых число n может принимать любые целые значения. Если исходная задача представляет собой статический процесс распределения, то подход Р. Беллмана переводит ее в динамический процесс, требуя распределения ресурсов последовательно по каждому

процессу, что собственно и отражено в названии метода — динамическое программирование [11].

Максимум целевого функционала $f(\mathbf{x})$ в указанной области зависит от количества процессов n, и от общего ограничения ресурса C. Эта зависимость в подходе динамического программирования записывается явно путем задания последовательности функций $\{f_m(c)\}$, $m = \overline{1,n}$, $0 \leqslant c \leqslant C$, следующим образом

$$f_m(c) = \max_{x_m} f(\mathbf{x}), \quad x_m \geqslant 0, \quad \sum_{k=1}^{m} x_k = c.$$

Функция $f_m(c)$ выражает оптимальный доход, получаемый от распределения ресурса c по m процессам. В двух частных случаях значения этой функции вычисляются элементарно. В разумном предположении, что доход от нулевого ресурса равен нулю — $g_m(0) = 0$, $\forall m = \overline{1,n}$, очевидно, что $f_m(0) = 0$, $\forall m = \overline{1,n}$. Также очевидно, что при значениях $c \geqslant 0$ функция $f_1(c) = g_1(c)$.

Для совместного распределения ресурса между несколькими процессами достаточно просто находятся рекуррентные соотношения, связывающие $f_m(c)$ и $f_{m-1}(c)$ для произвольных значений m и c. Если x_m — количество ресурса, назначенное для процесса с номером m, то остающееся количество — $(c - x_m)$ должно быть оптимально распределено для получения максимального дохода от остающихся $m-1$ процессов. Таким образом, при некотором значении x_m совокупный доход от распределения по m процессам составит

$$g_m(x_m) + f_{m-1}(c - x_m).$$

Очевидно, что оптимальным будет такой выбор значения x_m, который максимизирует эту функцию, что приводит к основному функциональному уравнению динамического программирования [11].

$$\begin{cases} f_1(c) = g_1(c); \\ f_m(c) = \max\limits_{0 \leqslant x_m \leqslant c} [g_m(x_m) + f_{m-1}(c - x_m)], \ \forall m = \overline{2,\ n}, \ c \geqslant 0. \end{cases}$$

Таким образом, задача целочисленной оптимизации в многомерном пространстве сводится к последовательности задач одномерной целочисленной оптимизации, что существенно сокращает трудоемкость получения решения. Но условия применимости метода накладывают ограничения аддитивности (сепарабельности) на целевой функционал. Кроме того, система ограничений не должна быть слишком сложной, и допускать построение рекуррентно связанных между собой функций Беллмана через возможно меньшее число аргументов, которые определяются «существенными» ограничениями задачи.

Дальнейшее развитие идеи выделения «существенных» ограничений привело к разработке новых методов решения задач целочисленного программирования, в частности — метода ядер (подробнее см., например [27], [26]).

Д.6.5. Точное решение задачи одномерной упаковки методом динамического программирования

Начиная с конца 50-х годов XX века, когда Р. Беллман предложил и обосновал основные идеи метода динамического программирования, задача оптимальной по стоимости одномерной упаковки привлекает внимание программистов и ученых, занимающихся разработкой и анализом алгоритмов. Повышенный интерес к этой задаче, равно как и к ее многочисленным модификациям, связан с разнообразными практическими областями применения. Заметим в этой связи, что метод динамического программирования хотя и позволил существенно сократить полный перебор вариантов, но реализующие этот метод алгоритмы не являются полиномиальными относительно длины входа.

В настоящее время рост производительности и, прежде всего, доступной оперативной памяти современных компьютеров обеспечивает приемлемое время решения для ряда практических задач, сводящихся к одномерной оптимальной по стоимости упаковке, с применением точных классических алгоритмов, реализующих метод динамического программирования.

Изложение применения метода динамического программирования для решения задачи одномерной оптимальной по стоимости упаковки основано на классической книге Р. Беллмана и С. Дрейфуса [11] и статье [25], посвященной анализу трудоемкости классических беллмановских алгоритмов. Мы хотим не только проиллюстрировать метод динамического программирования, но, и используя методы дискретной математики, получить оценки вычислительной сложности реализующих этот метод алгоритмов.

Содержательная постановка задачи упаковки. Представим себе, что у нас есть рюкзак с прямоугольным дном и нерастяжимыми стенками определенной высоты. У нас есть также несколько групп коробок, с таким же дном, как у рюкзака. В любой группе очень много коробок, и каждая коробка в группе одинакова по высоте и имеет определенную стоимость. Наша задача так упаковать рюкзак, чтобы он закрывался, и сумма стоимостей упакованных коробок была бы наибольшей. Хотя мы имеем дело с объемом, но в реальности мы рассматриваем только высоту рюкзака и высоты коробок — в этом смысле задача является одномерной. Поскольку у нас нет ограничений на состав коробок в рюкзаке, то интуитивное решение состоит в том, чтобы выбрать коробки из группы, обладающей максимальной удельной (на единицу высоты) стоимостью. Но, к сожалению, мы не можем разрезать коробки по высоте — задача является целочисленной, и интуитивное решение не всегда оптимально. Например, из двух групп коробок с высотами 5 и 7 и стоимостями 10 и 18, в рюкзак высотой 10 лучше положить две коробки из первой группы, чем одну из второй. Другая идея состоит в том, что можно рассмотреть все возможные варианты упаковки рюкзака и выбрать наилучший, но если рюкзак очень высокий, и у нас много разных групп коробок, то такой полный перебор может потребовать месяцев счета, даже на очень мощном компьютере.

Эта задача, известная, как задача об упаковке рюкзака, более корректно — задача оптимальной по стоимости одномерной упаковки, имеет разнообразные

практические применения, для которых сегодня актуальным является получение именно точных решений. К такой постановке сводится задача одномерного раскроя материала, формирования оптимального пакета акций на фиксированную сумму. На основе полученных оптимальных решений при значительном количестве групп коробок можно даже построить достаточно надежную криптосистему.

Математическая постановка задачи. Рассмотрим общую постановку задачи одномерной оптимальной по стоимости упаковки:

Пусть задано множество типов грузов

$$Y = \{y_i\}, \quad y_i = \{v_i, c_i\}, \quad i = \overline{1,n},$$

где каждый элемент y_i обладает целочисленным линейным размером v_i, или «объемом» в общепринятых терминах задач упаковки, и ценовой характеристикой c_i, которая содержательно отражает практически значимые предпочтения для загрузки объектов данного типа.

Так же целочисленным значением задан основной объем упаковки V. В классической постановке элементы y_i называются типами грузов. Для описания количества загружаемых в объем V элементов y_i введем в рассмотрение следующий характеристический вектор:

$$\mathbf{x} = \{x_i\}, \quad x = \overline{1,n},$$

где x_i — неотрицательное целое, т. е. $\mathbf{x} \in E_z^n$, $x_i \geqslant 0$. Значение компонента вектора $x_i = k$ соответствует загрузке k элементов типа y_i в объем V.

Среди всех возможных упаковок объема V грузами из Y должна существовать, по крайней мере, одна упаковка, максимизирующая суммарную стоимость, что приводит к следующей постановке задачи линейного целочисленного программирования.

Максимизировать линейный функционал:

$$P_n(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot c_i \to \max,$$

при выполнении следующего условия

$$\sum_{i=1}^{n} x_i \cdot v_i \leqslant V.$$

Содержательно условие (ограничение) означает, что суммарный объем, занимаемый грузами всех типов в количествах, указанных характеристическим вектором $\mathbf{x}$, не должен превышать общего объема упаковки.

Параметризация задачи. Реализация метода динамического программирования, обеспечивающего точное решение задачи одномерной упаковки, может быть осуществлена следующими двумя принципиально разными алгоритмами:

– табличным алгоритмом, опирающимся на вычисление и хранение векторов оптимальной упаковки для всех последовательных целочисленных значений объема;

– рекурсивным алгоритмом, непосредственно реализующим основное функциональное уравнение Беллмана для задачи одномерной оптимальной упаковки.

Оба алгоритма упаковки: — и табличный, и рекурсивный, являются количественно параметрическими. Это означает, что количество элементарных операций, задаваемых алгоритмом, зависит не только от количества данных на входе, но и от их значений. Оценка вычислительной сложности этих алгоритмов будет зависеть от значений $n, V, v_1, \ldots, v_n$, что существенно затрудняет анализ. Мы вводим параметр $k = \dfrac{V}{\bar{v}}$, характеризующий, сколько грузов среднего объема

$$\bar{v} = \frac{1}{n} \cdot \sum_{i=1}^{n} v_i$$

размещается в объеме упаковки V. Очевидно, что в реальности количество любых грузов, размещенных в объеме V является целым числом, но для оценки вычислительной сложности в среднем, необходимо учитывать, что параметр k является действительным (вещественным) числом.

Функциональное уравнение Беллмана для задачи упаковки. Опираясь на терминологию метода динамического программирования (см. Д.6.4), будем считать, что распределяемым ресурсом является объем упаковки V, а функции дохода — $g_i(x_i) = c_i \cdot x_i$. Наша задача — максимизировать доход (стоимость упаковки), заданный линейным функционалом $P_n(\mathbf{x})$, путем распределения ограниченного ресурса объема упаковки между грузами указанных типов. В этих условиях мы можем непосредственно использовать основное функциональное уравнение Беллмана, которое с учетом обозначений математической постановки задачи упаковки имеет вид [11].

$$\begin{cases} f_0(v) = 0; \\ f_m(v) = \max\limits_{x_m} \{x_m \cdot c_m + f_{m-1}(v - x_m \cdot v_m)\}, \\ m = \overline{1,n},\ v = \overline{0,V},\ x_m = 0, 1, \ldots, \left[\dfrac{v}{v_k}\right]. \end{cases} \qquad (Д.37)$$

Таким образом, метод предполагает последовательное решение одномерных задач целочисленной оптимизации с использованием информации об оптимальной упаковке любого дискрета объема предыдущими типами грузов.

Табличный алгоритм решения задачи упаковки. Один из вариантов точного решения задачи методом динамического программирования — это табличный алгоритм, позволяющий получить набор оптимальных решений не только для заданного объема V, но и для всех промежуточных дискретных значений этого объема.

Обозначим вектор оптимальной упаковки для объема v элементами y_i, $i = \overline{1,m}$, $m \leqslant n$, через $\mathbf{x}_v^m$, а через $f_m(v)$ стоимость оптимальной упаковки в этом объеме. В силу принципа оптимальности [11] $f_m(v) = \max\limits_{x_m} P_m(\mathbf{x})$, а порядок предъявления элементов y_i не является существенным.

Решение табличным алгоритмом осуществляется на основе функционального уравнения метода динамического программирования (Д.37).

Результатом решения задачи является набор оптимальных значений целевой функции $f_n(v)$ и соответствующих векторов оптимальной упаковки $\mathbf{x}_v^n$ для всех объемов v от 0 до V исходными элементами (грузами различных типов) из множества Y. Отметим, что поскольку табличный способ позволяет получить оптимальные решения для всех промежуточных объемов упаковки, то эту информацию можно использовать, например, для исследования чувствительности целевой функции $f_n(v)$ по изменению объема упаковки.

Рассмотрим пример решения задачи одномерной упаковки с использованием табличного алгоритма. Мы решаем задачу с тремя типами грузов, при этом общий объем упаковки $V = 10$. Информация об объемах и стоимостях типов грузов приведена в табл. Д.6.1.

Таблица Д.6.1.
Таблица исходных данных

i	v_i	c_i
1	2	3
2	3	5
3	4	7

Табличная реализация основного функционального уравнения Беллмана приводит к последовательному рассмотрению типов грузов, для каждого из которых мы получаем оптимальную упаковку для всех дискретов объема от 1 до 10, приведенную в табл. Д.6.2. Заметим, что рекурсивный вызов в функциональном уравнении (Д.37) в табличном алгоритме есть не что иное, как обращение к предыдущей таблице оптимальной упаковки для определенного значения объема.

Таблица Д.6.2. Таблица оптимального дохода

v	x_1	$f_1(v)$
1	0	0
2	1	3
3	1	3
4	2	6
5	2	6
6	3	9
7	3	9
8	4	12
9	4	12
10	5	15

$\Rightarrow$

v	x_1	x_2	$f_2(v)$
1	0	0	0
2	1	0	3
3	0	1	5
4	2	0	6
5	1	1	8
6	0	2	10
7	2	1	11
8	1	2	13
9	0	3	15
10	2	2	16

$\Rightarrow$

v	x_1	x_2	x_3	$f_3(v)$
1	0	0	0	0
2	1	0	0	3
3	0	1	0	5
4	0	0	1	7
5	1	1	0	8
6	0	2	0	10
7	0	1	1	12
8	0	0	2	14
9	0	3	0	15
10	?	?	?	?

Обратите внимание на то, как изменяется состав грузов, оптимально заполняющих объем, при изменении объема упаковки на единицу — такая чувстви-

тельность оптимального решения по ограничениям характерна для задач целочисленного программирования.

Табличный алгоритм решения задачи упаковки.

```
Procedure Табличный алгоритм для задачи упаковки
  Инициализация
    (boxV[1..n], boxC[1..n] — массивы исходных данных)
    (x[1..n, 0..V], x1[1..n, 0..V] — массивы векторов оптимальной
    упаковки для текущего и предыдущего типов грузов)
    (f[0..V] — функция Беллмана — оптимальная стоимость
    упаковки в объеме v)
Input
  n — количество типов грузов
  V — объем упаковки
  boxV[1..n] — объемы типов грузов
  boxC[1..n] — стоимости типов грузов
Begin
  Формирование таблицы для грузов типа 1
  for i = 0 to V (по всем дискретам объема)
    begin
      x[1, i] = i div boxV[1] (число грузов типа 1 в объеме i)
      x1[1, i] = x[1, i]
      f[i] = x[1, i] * boxC[1]
    end
Основная часть алгоритма
for i = 2 to N (цикл по типам грузов)
  begin
    Vi = boxV[i]
    Ci = boxC[i]
    for v = 0 to V (цикл по дискретам объема упаковки)
      begin
        maxKol = 0;
        maxC = f[v]
        if Vi ⩽ v then (груз типа i размещается в объеме v)
          begin
            z = v div Vi (z — максимальное количество груза
                                типа i в объеме v)
            for k = 0 toz (цикл нахождения максимума f)
              begin
                if (maxC < (Ci * k + f[v − k * Vi])) then
                  begin
                    maxC = Ci * k + f[v − k * Vi]
                    maxKol = k
                  end
              end (конец цикла по k)
          end (end if Vi ⩽ v)
```

```
      (сохранение оптимального решения —
      формирование оптимального вектора упаковки)
        for l = 1 to i − 1
          x[l, v] = x1[l, v − maxKol ∗ Vi]
        end (for l)
        x[i, v] = maxKol
    end (конец цикла по дискретам объема)

    Переход к новому типу груза
    (копирование массива векторов оптимальной упаковки)
    объявить массив x массивом x1
    (формирование массива оптимальных стоимостей
    упаковок всех дискретов объема грузами i типов)

    for l = 1 to V
      sum = 0
        for j = 1 to i
          sum = sum + x[j, l] ∗ boxC[j]
        end (end for j)
        f[l] = sum
    end (end for l)
  end (конец цикла по типам грузов)
```

Output

$f[V]$ — *стоимость оптимальной упаковки в объеме* V
$x[1..n, V]$ — *вектор оптимальной упаковки*

End

Отметим, что если мы создадим трехмерный массив векторов оптимальной упаковки, то в результате табличный алгоритм получит все оптимальные решения для всех дискретов объема и всей последовательности предъявления типов грузов.

Оценка вычислительной сложности табличного алгоритма. Оценим вычислительную сложность в среднем для основной части алгоритма в условиях принятой параметризации. Мы считаем, что все грузы имеют одинаковый объем, равный среднему значению, и, следовательно, не более k грузов каждого типа может быть размещено в объеме V.

Заметим, что количество операций в цикле нахождения максимума целевого функционала имеет порядок $\Theta(1)$, равно как и количество операций, управляемых собственно циклом по дискрету объема. Цикл по нахождению максимума целевого функционала зависит от цикла по дискретам объема, и их совместное рассмотрение приводит к следующим результатам, которые сведены в табл. Д.6.3.

Таким образом, совокупное количество проходов внешнего (по v) и внутреннего (по z) циклов равно

$$V \cdot \Theta(1) + (1 \cdot \overline{v} + 2 \cdot \overline{v} + \cdots + (k-1) \cdot \overline{v} + k) \cdot \Theta(1) = \\ = \Theta(1) \cdot (V + k + \overline{v} \cdot \sum_{i=1}^{k-1} i) = \Theta(1) \cdot \left(V + k + \overline{v} \cdot \frac{k \cdot (k-1)}{2} \right), \qquad (Д.38)$$

мы воспользовались формулой для суммы арифметической прогрессии. Поскольку в силу введенной параметризации $k = V/\overline{v}$, то подстановка в (Д.38) дает следующую оценку для двух внутренних циклов

$$\left(\frac{V^2}{2\overline{v}} + \frac{V}{2} + \frac{V}{\overline{v}}\right) \cdot \Theta(1),$$

и учитывая, что цикл по типам грузов выполняется n раз, окончательно получаем вычислительную сложность основной части табличного алгоритма в виде

$$\Theta\left(\frac{n \cdot V^2}{2 \cdot \overline{v}}\right).$$

Таблица Д.6.3. Анализ трудоемкости циклов

Значения счетчика цикла по дискретам объема	Количество проходов цикла вычисления максимума
от 0 до $\overline{v} - 1$	0
от $\overline{v}$ до $2 \cdot \overline{v} - 1$	1
...	...
от $(k-1) \cdot \overline{v}$ до $k \cdot \overline{v} - 1$	$k - 1$
при $k \cdot \overline{v} = V$	k

Рекурсивный алгоритм решения задачи упаковки. Другим вариантом точного решения задачи динамического программирования является рекурсивный алгоритм, напрямую реализующий функциональное уравнение Беллмана — рекуррентные соотношения (Д.37). Рекурсия выполняется для вычисления оптимальной стоимости упаковки грузами предыдущих типов в меньшем объеме. Схематично выполнение этого алгоритма может быть представлено в виде дерева рекурсий. Такое дерево для нахождения оптимальной упаковки объема 10 типами грузов, характеристики которых приведены в таблице табл. Д.6.1 показано на рис Д.11.

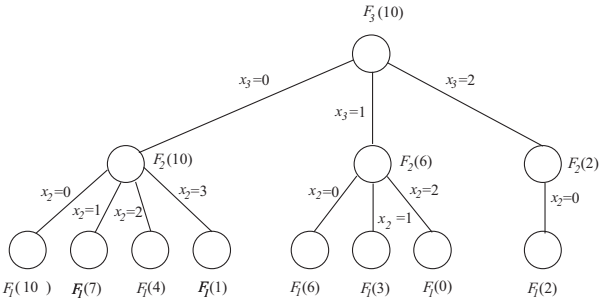


Рисунок Д.11. Дерево рекурсий, порождаемое алгоритмом упаковки

Рекурсивная функция для решения задачи упаковки.

```
Function F(V, x) (описание рекурсивной функции)
                 (V — текущий объем упаковки)
                 (x — количество рассматриваемых типов грузов)
                 (boxV[1..n], boxC[1..n] — массивы исходных данных)
                 (F — функция Беллмана — оптимальная стоимость упаковки
                 в объеме V грузами x типов)
Begin
  if (x = 0 или V = 0)
    then (Останов рекурсии — прямое вычисление при x = 0
          или V = 0)
      F = 0 (функция возвращает оптимальную ст-ть)
    else
      begin
        if x = 1
          then (Останов рекурсии — прямое вычисление при x = 1)
            begin
              k = V/boxV[1]
              F = k * boxC[1] (функция возвращает оптим. ст-ть)
              (формирование вектора оптимальной упаковки)
            end
          else (Рекурсивные вызовы для определения max F)
            begin
              Max = 0;
              k = V div boxV[x]
              if k = 0
                then
                  F = F(V, x − 1)
                else
                begin
                  for i = 0 to k
                    Cost = i * boxC[x] + F((V − boxV[x] * i), x − 1)
                      if Cost > Max
                        then
                          Max = Cost
                          Optx = i
                  end (for i)
                  F = Max(функция возвращает оптимальную
                          стоимость)
                  Формирование вектора оптимальной упаковки
                  Оптимальное количество грузов типа x
                  в объеме V равно Optx
                end (if k = 0 else)
            end (if x = 1 else)
      end (if x = 0 или V = 0 else)
```

(функция возвращает оптимальную стоимость упаковки в объеме V грузами x типов)

Return

End

Пример вызова из основной программы — F(1000, 15).

Оценка вычислительной сложности рекурсивного алгоритма. Трудоемкость данного алгоритма прямо пропорциональна количеству вершин в дереве рекурсии. Мы считаем, что подготовка и обработка результатов рекурсивного вызова включена в этот вызов. Таким образом, трудоемкость цикла нахождения максимума разнесена по рекурсивным вызовам, и, следовательно, рекурсивная функция выполняет порядка $\Theta(1)$ операций на один вызов. Следовательно, наша задача — попытаться получить явную формулу для количества вершин дерева рекурсий в условиях введенной параметризации. Обозначим через $R(n,k)$ функцию, задающую количество вершин, в зависимости от значений n и k. Рассмотрим структуру дерева рекурсии при некоторых значениях n и k. Фрагменты дерева для значений $k=1$ и $k=2$ показаны на рис. Д.12 и Д.13.

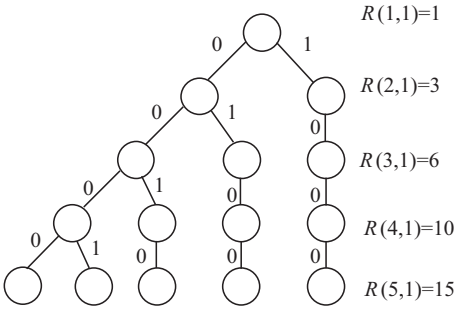


Рисунок Д.12. Фрагмент дерева рекурсий для значения $k=1$

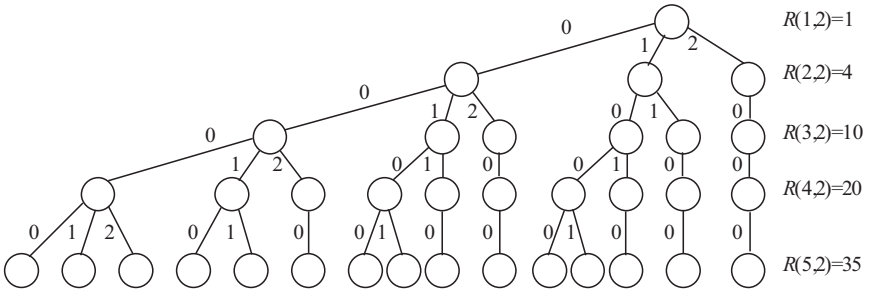


Рисунок Д.13. Фрагмент дерева рекурсий для значения $k=2$.

Для того чтобы найти закономерность для значений $R(n,k)$ построим треугольник Паскаля, первые семь строк которого показаны на рис. Д.13.

Теперь видно, что значения $R(n,k)$ это элементы треугольника Паскаля — биномиальные коэффициенты, и устанавливая соответствие между номерами строк треугольника и значениями n и k — $m=(n+k)$, окончательно получаем

$$R(n,k)=C_{n+k}^{n-1}, \qquad (Д.39)$$

и, следовательно, вычислительная сложность рекурсивного алгоритма решения задачи оптимальной упаковки имеет вид

$$\Theta(1) \cdot R(n,k) = \Theta(C_{n+k}^{n-1}),$$

и при фиксированном значении n принимает максимальное значение при $k = n-2$, в силу свойств биномиальных коэффициентов.

$m=1$						1						
$m=2$					**1**		2		1			
$m=3$				**1**		**3**		3		1		
$m=4$			1		**4**		**6**		4		1	
$m=5$		1		5		**10**		**10**		5		1
$m=6$	1		6		15		**20**		**15**		6	1
$m=7$		1		7		21		35		**35**		**21**		7		1

Рисунок Д.14. Треугольник Паскаля — биномиальные коэффициенты.

Д.6.6. Метод ветвей и границ и задача коммивояжера

Мы излагаем классический алгоритм Дж. Литла, К. Мерти, Д. Суини и К. Кэрол для точного решения задачи коммивояжера методом ветвей и границ опираясь на оригинальную статью авторов [7], и ее изложение в [17]. Используемый нами иллюстрирующий пример для задачи коммивояжера содержится в [17]. Мы хотим показать, насколько не просто метод адаптируется к конкретной задаче, и как красиво все получается, когда все трудности преодолены.

Содержательная постановка задачи. Коммивояжер — это сотрудник торговой организации, задача которого состоит в том, чтобы посетить ряд городов с целью рекламы и продажи товаров. Мы предполагаем, что коммивояжер живет в стране с развитой транспортной сетью, и между каждой парой городов существует собственное транспортное сообщение. Будем называть туром порядок посещения всех городов, и потребуем, чтобы каждый город был посещен только один раз — это так называемая задача коммивояжера без возвратов. Стоимости проезда между городами известны, причем, в общем случае, стоимость проезда туда и обратно различны — это несимметричная задача коммивояжера.

Собственно сама задача состоит в том, чтобы найти такой тур коммивояжера, который имел бы минимальную стоимость. Очевидно, что число туров — конечно, и мы можем решить задачу прямым перебором туров. Оценим объем такого перебора в задаче с n городами. Поскольку тур — это замкнутый цикл, то в качестве начального пункта можно выбрать любой город. Отправляясь из него, у нас есть $n-1$ возможных вариантов, и мы выбираем один их них. В следующем городе таких вариантов будет $n-2$, т. к. возврат обратно запрещен. Поскольку выбор является независимым, то полное множество будет содержать $(n-1)!$ туров, и переборный алгоритм решения становится для реальных размерностей задачи совершенно неприемлемым.

Постановка в терминах теории графов. Ассоциируя города с вершинами графа, а пути сообщения и стоимости проезда с нагруженными ребрами — мы получаем полный ориентированный асимметричный граф без собственных петель на n вершинах. Граф может быть описан матрицей стоимости C, значение каждого элемента которой — c_{ij} равно стоимости (измеряемой реально в единицах времени, денег или расстояния) прямого проезда из города i в город j. Задача коммивояжера называется симметричной, если $c_{ij} = c_{ji}, \forall i \neq j,\ i, j = \overline{1,n}$, т. е. если стоимость проезда между каждыми двумя городами не зависит от направления, и несимметричной, если это не так. Отсутствие собственных петель может быть обозначено как $c_{ii} = \infty,\ \forall i = \overline{1,n}$ (подумайте о том, как Вы будете реализовывать это в реальной программе?).

Задача в терминах теории графов формулируется как задача нахождения покрывающего (полного) цикла наименьшей стоимости, который мы называем туром, на полном ориентированном графе, заданном несимметричной (в общем случае) матрицей стоимостей C.

Постановка в терминах целочисленного программирования. Задача коммивояжера, как задача целочисленного программирования допускает несколько разных формулировок, авторы предлагают два следующих подхода.

1. Постановка в пространстве $E_z^{n^2-n}$. Для формулировки задачи коммивояжера, как целочисленной задачи, определим вначале размерность целочисленного пространства. Поскольку наша цель — выбор некоторых ребер полного графа, составляющих тур, из всего множества ребер, то предлагается рассматривать компоненты вектора $\mathbf{x}$, как указатели на выбираемые ребра. Полный ориентированный граф на n вершинах содержит $n \cdot (n-1) = n^2 - n$ ребер — это и есть размерность нашего целочисленного пространства. Компоненты вектора $\mathbf{x}$ принимают значения 0 или 1, указывая на выбираемые ребра и, следовательно, $\mathbf{x} \in 1_z^{n^2-n}$. Поскольку тур содержит ровно n ребер, то вектор $\mathbf{x}$ содержит ровно n компонент, равных единице. Таким образом, множество точек перебора — это некоторое подмножество точек положительной полусферы ($x_i \geqslant 0$) с радиусом равным $\sqrt{n}$, и с центром в нуле. В принятых обозначениях (см. Д.6.1) это — $S_z^{n^2-n}(\mathbf{0}, \sqrt{n})$. Заметим, что не все точки этой сферы с положительными целыми координатами являются возможными решениями задачи коммивояжера, поскольку ребра, выбранные по единичным компонентам вектора $\mathbf{x}$, должны составлять тур.

Для построения целевого функционала и ограничений нам необходимо ввести два отображения. Обозначим далее через N — конечное множество целых чисел

$$N = \{i|\ i = \overline{1,n}\},$$

через M — множество

$$M = \{i|\ i = \overline{1,n^2-n}\},$$

и через $\mathbb{R}^+$ — множество положительных вещественных чисел. Определим взаимно однозначное отображение

$$M \xrightarrow{v} N \times N,\ v(i) = (k,l),$$

которое каждому номеру ребра i ориентированного графа ставит в соответствие номера пар инцидентных ребру вершин — (k,l), и отображение

$$M \xrightarrow{c} \mathbb{R}^+,\ c(i) = c_{v(i)} = c_{kl},$$

которое задает веса (стоимости) для последовательно пронумерованных ребер. Поскольку задача состоит в минимизации общей стоимости объезда городов, то постановка задачи имеет вид

$$f(\mathbf{x}) = \sum_{i=1}^{n^2-n} c(i)\cdot x_i,\ f(\mathbf{x}) \to \min,\ \text{при}$$

$$\begin{cases} x_i \in \{\,0,1\,\},\ \sum\limits_{i=1}^{n^2-n} x_i = n,\ \mathbf{x} \in S_z^{n^2-n}(\mathbf{0},\sqrt{n})\ ; \\ n \text{ ребер, соответствующих вектору } \mathbf{x},\ \text{образуют тур.} \end{cases}$$

Математическую формулировку последнего условия мы оставляем читателю в качестве упражнения.

2. Постановка в пространстве E_z^{n-1}. Предыдущая постановка, несмотря на кажущуюся простоту структуры вектора $\mathbf{x}$, имеет серьёзный недостаток — среди всех рассматриваемых векторов только немногие образуют тур (см. упражнения). Мы хотим, чтобы все точки некоторого множества в пространстве E_z^{n-1} гарантированно представляли тур. Если это так, то опадает необходимость проверки условия тура для точек этого множества, кроме того мы хотим уменьшить размерность пространства.

Новая постановка задачи опирается на понятие перестановки на множестве целых чисел. Будем далее обозначать через $\pi(k,l)$, $l \geqslant k$ множество всех перестановок целых чисел $k, k+1, \ldots, l$, очевидно, что таких перестановок $(l-k+1)!$. Рассмотрим множество перестановок $\pi(2,n)$ — оно содержит $|\pi(2,n)| = (n-2+1)! = (n-1)!$ различных перестановок — ровно столько имеется различных туров коммивояжера в задаче с n городами. Поскольку тур — это полный цикл по всем вершинам, то начальная вершина тура может быть выбрана произвольно. Мы фиксируем вершину с номером один, и считаем, что некоторая перестановка из множества $\pi(2,n)$ задает порядок обхода вершин, начиная с первой, и после последней вершины, заданной этой перестановкой, мы снова возвращаемся в начало тура.

Таким образом, компоненты нашего вектора $\mathbf{x}$ в пространстве E_z^{n-1} — это числа от двух до n, а сам вектор ассоциирован с некоторой перестановкой из $\pi(2,n)$, и мы можем записать

$$x_i \in \{\,2,n\,\},\ i = \overline{1,n-1},\ x_i \neq x_j,\ \text{при } i \neq j, \quad \mathbf{x} \in \pi(2,n). \tag{Д.40}$$

Множество векторов $\mathbf{x}$ координаты которых совпадают с перестановками из $\pi(2,n)$ обозначим через $\pi_z^{n-1}(2,n)$. Где расположены точки различных векторов $\mathbf{x}$ в пространстве E_z^{n-1}, удовлетворяющие (Д.40)? Заметим, что поскольку операция сложения ассоциативна, то сумма квадратов компонент различных векторов $\mathbf{x}$ одинакова — это разные перестановки чисел от двух до n. Таким образом, точки различных векторов $\mathbf{x}$ являются точками положительной полусферы в E_z^{n-1}

с центром в нуле и радиусом

$$r = \sqrt{\sum_{i=2}^{n} i^2} = \sqrt{\frac{n \cdot (n+1) \cdot (2n+1)}{6} - 1}, \quad \mathbf{x} \in S_z^{n-1}(\mathbf{0}, r),$$

отметим, что мы воспользовались формулой для суммы квадратов, полученной в разделе Д.5.

Нам осталось сформулировать, как задаются стоимости ребер — воспользуемся уже введенными в первой постановке задачи множествами, и определим отображение

$$N \times N \underset{c}{\to} R^+, \; c(i,j) = c_{ij}, \; c(i,i) = \infty,$$

которое каждой упорядоченной паре вершин графа ставит в соответствие стоимость, инцидентного этой паре, ребра.

В построенном формализме задача коммивояжера в пространстве E_z^{n-1} имеет следующую постановку

$$f(\mathbf{x}) = c(1, x_1) + c(x_{n-1}, 1) + \sum_{i=1}^{n-2} c(x_i, x_{i+1}) \; , \quad f(\mathbf{x}) \to \min, \text{ при}$$
$$\mathbf{x} = (x_1, \ldots, x_n) \in \pi_z^{n-1}(2, n).$$

Рассматриваемый далее метод ветвей и границ работает, хотя и не явно, именно с этой постановкой задачи коммивояжера.

Реализация идей метода ветвей и границ для задачи коммивояжера. В соответствии с общей идеей метода ветвей и границ все множество допустимых решений должно последовательно разделяться на подмножества с целью сокращения перебора — это процедура ветвления. С каждым таким подмножеством должна быть связана оценка (нижняя граница при поиске минимума), обеспечивающая отсечение тех подмножеств, которые заведомо не содержат оптимального решения — это процедура построения границ. Таким образом, мы приходим к исследованию древовидной модели пространства решений.

В рассматриваемой задаче таким исходным множеством является множество всех туров коммивояжера, и мы минимизируем целевой функционал, определяющий стоимость тура. Излагаемые ниже идеи авторов алгоритма — это своего рода классика метода ветвей и границ. Для построения алгоритма необходимо описать две основные процедуры — ветвление и построение границ. Начнем рассмотрение с процедуры ветвления.

Мы начинаем построение поискового дерева решений с корня, который будет соответствовать множеству «всех возможных туров», т. е. эта вершина дерева представляет множество R всех $(n-1)!$ возможных туров в задаче с n городами. Ветви, выходящие из корня, определяются выбором одного ребра, скажем ребра (k,l). Идея авторов алгоритма состоит в том, чтобы разделить текущее множество туров на два множества: одно, которое, весьма вероятно, содержит оптимальный тур, и другое, которое, вероятно, этого тура не содержит. Для этого выбираем ребро (k,l) — оно, как мы надеемся, входит в оптимальный тур,

и разделяем R на два множества $\{k,l\}$ и $\{\overline{k,l}\}$. Во множество $\{k,l\}$ входят все туры из R, содержащие ребро (k,l), т. е. проходящие через него, а во множество $\{\overline{k,l}\}$ — туры не содержащие это ребро.

Заметим, что идея авторов алгоритма очень перспективна — если мы так организуем процесс ветвления, что на каждом шаге будем выбирать «правильное» ребро, то весь процесс будет завершен за n шагов.

Проиллюстрируем сказанное на конкретном примере. В табл. Д.6.4 приведена матрица стоимостей для несимметричной задачи коммивояжера с пятью городами.

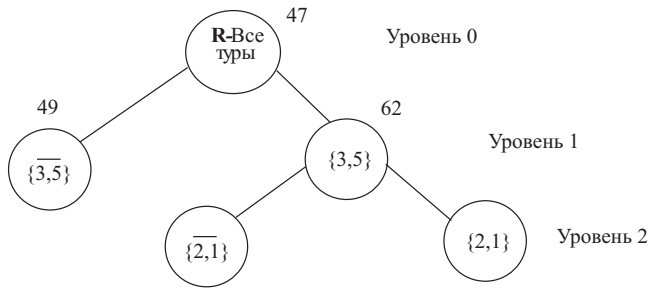
В качестве претендентов на ребро ветвления можно рассмотреть ребра, имеющие небольшие стоимости. Предположим, что производится ветвление на ребре $(3,5)$, имеющем наименьшую стоимость во всей матрице. Ниже мы рассмотрим другой алгоритм выбора ребра ветвления. Корень и первый уровень поискового дерева решений будут тогда такими, как показано на рис. Д.15. Заметим, что каждый тур содержится только в одном множестве уровня 1. Если бы мы как-то могли сделать вывод, что множество $\{\overline{3,5}\}$ не содержит оптимального тура, то нам нужно было бы исследовать только множество $\{3,5\}$ — в этой идее, собственно говоря, и заключается суть метода ветвей и границ.

Таблица Д.6.4. Матрица стоимостей

	1	2	3	4	5
1	∞	25	40	31	27
2	2	∞	17	30	25
3	19	15	∞	6	1
4	9	50	24	∞	6
5	22	8	7	10	∞

Продолжая процедуру ветвления, мы разделяем множество $\{3,5\}$ также, как и множество R. Следующее по дешевизне ребро в матрице — это ребро $(2,1)$ со стоимостью 5. Поэтому можно разделить множество $\{3,5\}$ на подмножество туров, включающих ребро $(2,1)$, и подмножество туров, не включающих это ребро. Полученное поисковое дерево решений показано на рис. Д.15.

Рисунок Д.15. Фрагмент поискового дерева решений



Путь от корня к любой вершине дерева выделяет определенные ребра, которые должны, или не должны быть включены во множество, представленное

данной вершиной. Например, левая вершина уровня 2 на рис. Д.15 представляет множество всех туров, проходящих через ребро $(3,5)$ и не проходящих через ребро $(2,1)$. Таким образом, процедура ветвления состоит в разделении текущего подмножества туров на два подмножества путем выбора некоторого ребра.

Теперь расскажем об идее авторов алгоритма, связанной с процедурой вычисления границ. С каждой вершиной дерева мы связываем нижнюю границу стоимости любого тура из множества, представленного данной вершиной. Вычисление нижних границ — основной фактор, дающий возможность сокращения перебора в любом алгоритме, реализующем метод ветвей и границ. Очевидно, что мы хотим получить как можно более точные нижние границы. Причина этого следующая. Предположим, что мы построили конкретный полный тур со стоимостью Ct. Если нижняя граница, связанная с множеством туров, представленных некоторой вершиной поискового дерева, больше, чем Ct, то до конца процесса поиска не нужно рассматривать эту и все следующие за ней вершины, т. к. если мы гарантируем нижнюю границу, то все содержащиеся в этой вершине туры имеют стоимость большую, чем уже найденный тур.

Основной шаг при вычислении нижних границ известен как процедура *приведения матрицы стоимостей*. Эта процедура основана на следующих двух соображениях:

1. В терминах матрицы стоимостей каждый полный тур содержит только один элемент (ребро и соответствующую стоимость) из каждого столбца и каждой строки матрицы. Заметим, что обратное утверждение не всегда верно — множество, содержащее один и только один элемент из каждой строки и из каждого столбца, не обязательно представляет тур.

2. Если вычесть константу h из каждого элемента какой-то строки или столбца матрицы стоимостей, то стоимость любого тура при новой матрице будет ровно на h меньше. Поскольку любой тур должен содержать ребро из данной строки или данного столбца, стоимость всех туров уменьшается на h. Это вычитание называется приведением строки (или столбца) матрицы стоимостей. Пусть t — оптимальный тур при матрице стоимостей C. Стоимость тура t может быть определена как $z(t) = \sum\limits_{(i,j)\in t} c_{ij}$.

Если матрица C' получается из матрицы C приведением строки (или столбца), то тур t должен оставаться оптимальным туром и в матрице C', при этом стоимости туров связаны соотношением

$$z(t) = h + z'(t),$$

где $z'(t)$ — стоимость тура t в матрице C'.

Под процедурой приведения всей матрицы стоимостей понимается следующее: мы последовательно проходим строки и вычитаем значение наименьшего элемента h_i каждой строки из каждого элемента этой строки. Затем аналогичные действия выполняются для каждого столбца. Если для некоторого столбца или строки значение $h_i = 0$, то рассматриваемый столбец или строка уже приве-

дены, и мы переходим к следующему столбцу или строке матрицы стоимостей:

$$h = \sum_{\substack{\text{все строки} \\ \text{и столбцы}}} h_i.$$

Полученную в результате матрицу стоимостей назовем *приведенной из* C. В табл. Д.6.5 показано приведение исходной матрицы стоимостей. Значения h_i даны в конце каждой строки и столбца (строки и столбцы последовательно пронумерованы).

Общее приведение составляет $h = 25 + 5 + 1 + 6 + 7 + 3 = 47$, следовательно, нижняя граница стоимости любого тура из множества R равна 47, т. е.

$$z(t) = h + z'(t) \geqslant h = 47,$$

так как $z'(t) \geqslant 0$ для любого тура t при приведенной матрице C'. Эта граница и указана около корня дерева на рис. Д.15. Заметим, что если ребра, имеющие нулевую стоимость в приведенной матрице, составляют тур, то мы сразу получили оптимальное решение (к сожалению, это почти что всегда не так).

Следующий вопрос — это вопрос о вычислении границ для некорневых вершин поискового дерева решений. Авторы алгоритма строят эти оценки следующим образом.

Таблица Д.6.5. Приведенная матрица стоимостей

	1	2	3	4	5	
1	∞	25	40	31	27	$h_1 = 25$
2	2	∞	17	30	25	$h_2 = 5$
3	19	15	∞	6	1	$h_3 = 1$
4	9	50	24	∞	6	$h_4 = 6$
5	22	8	7	10	∞	$h_5 = 7$
	$h_6 = 0$	$h_7 = 0$	$h_8 = 0$	$h_9 = 3$	$h_{10} = 0$	

По определению ребро $(3, 5)$ содержится в каждом туре множества $\{3, 5\}$. Этот факт препятствует выбору ребра $(5, 3)$, так как ребра $(3, 5)$ и $(5, 3)$ образуют цикл, а это не допускается ни для какого тура. Поэтому ребро $(5, 3)$ исключаем из рассмотрения, положив значение $c_{53} = \infty$. Строку 3 и столбец 5 также можно исключить из дальнейшего рассмотрения по отношению к множеству туров $\{3, 5\}$, потому что уже есть ребро из вершины 3 в вершину 5. Часть приведенной матрицы стоимостей, из табл. Д.6.5, которая будет необходима для дальнейшего поиска на множестве туров $\{3, 5\}$, показана в табл. Д.6.6. Она может быть приведена к матрице стоимостей, показанной в табл. Д.6.7 со значением $h = 15$. Теперь нижняя граница для любого тура из множества $\{3, 5\}$ равна 47+15=62, что указано около вершины $\{3, 5\}$ на рис. Д.15.

Таблица Д.6.6.
Матрица стоимостей
для множества $\{3,5\}$

	1	2	3	4
1	∞	0	15	3
2	0	∞	12	22
4	3	44	18	∞
5	5	1	∞	0

Таблица Д.6.7.
Приведенная матрица стоимостей
для множества $\{3,5\}$

	1	2	3	4	
1	∞	0	3	3	0
2	0	∞	0	22	0
4	0	41	3	∞	$h_3 = 3$
5	15	1	∞	0	0
	0	0	$h = 12$	0	

Нижняя граница для множества $\{\overline{3,5}\}$ получается несколько иным способом. Ребро $(3,5)$ не может находиться в этом множестве, поэтому полагаем $c_{35} = \infty$ в матрице, приведенной в табл. Д.6.5. В любой тур из множества $\{\overline{3,5}\}$ будет входить какое-то ребро из вершины 3 и какое-то ребро к вершине 5. Самое дешевое ребро из вершины 3, исключая старое значение $(3,5)$, имеет стоимость 2, а самое дешевое ребро к вершине 5 имеет стоимость 0. Следовательно, нижняя граница любого тура во множестве $\{\overline{3,5}\}$ равна $47 + 2 + 0 = 49$ (рис. Д.15).

При приведении матрицы стоимости для множества $\{3,5\}$ нам удалось сократить ее размер. Очевидно, что это сокращение будет происходить, каждый раз для множества $\{k,l\}$, так что мы значительно сокращаем при этом вычислительные затраты. Еще один вывод состоит в том, что если мы сможем найти тур из множества $\{\overline{3,5}\}$ со стоимостью, меньшей или равной 62, то тогда мы отбрасываем вершину поискового дерева решений $\{3,5\}$, и в этом случае будем говорить, что вершина $\{3,5\}$ в дереве отработана. Тогда следующей целью может быть ветвление из вершины $\{\overline{3,5}\}$ в надежде найти тур со стоимостью в пределах $49 \leqslant Ct \leqslant 62$.

Алгоритм метода ветвей и границ для задачи коммивояжера. Теперь, когда ясно, как в общих чертах, выглядят процедуры ветвления и построения границ, можно предложить следующую схему алгоритма метода ветвей и границ (МВГ) для задачи коммивояжера, в которой приняты следующие обозначения. Пусть X — текущая вершина поискового дерева, а (k,l) — ребро по которому происходит ветвление, обозначим вершины, непосредственно следующие за X, через Y и $\overline{Y}$. Множество Y есть подмножество туров из X, проходящих через ребро (k,l), а множество $\overline{Y}$ — подмножество туров из X, не проходящих через ребро (k,l). Вычисленные нижние границы для множеств Y и $\overline{Y}$ обозначим через $w(Y)$ и $w(\overline{Y})$ соответственно. Самый дешевый тур, известный алгоритму в данный момент обозначим через z_0, причем в момент инициализации $z_0 = \infty$.

Procedure *Алгоритм МВГ для задачи коммивояжера*
Input
Ввод размерности и матрицы стоимостей
Begin
1. *Инициализация*
2. *Приведение матрицы стоимостей C*
3. *Установка корня поискового дерева решений* $X = R$
While $(w(X) < z0)$

begin
4. *Выбор ребра ветвления* (k, l)
5. *Процесс ветвления. Создание вершины* $\overline{Y}$
и вычисление $w(\overline{Y})$.
6. *Процесс ветвления. Создание вершины* Y
и вычисление $w(Y)$.
if *(размер матрицы стоимостей в вершине* $Y = 2$*)*
then
begin
7. *Проведение исчерпывающей оценки для вершины* Y
if $(w(Y) < z0)$
then
begin
$z0 = w(Y)$ *(запоминаем тур)*
end $(if(w(Y) < z0)$
end *(if)*
8. *Выбор следующей вершины поискового дерева*
решений, и установка X
9. *Вычисление или чтение фрагмента матрицы* C,
соответствующей выбранной вершине X
end *(while* $w(X) < z0$*)*
Оптимальное решение со стоимостью $z0$ *найдено*
End

Особенности реализации этапов алгоритма. В приведенной укрупненной схеме данного алгоритма остались нераскрытыми некоторые важные детали, которые необходимо обсудить.

Этап 1. Установление начальных значений переменных, или инициализация, не представляет труда для программиста. Единственный вопрос — это выбор структуры данных для хранения поискового дерева решений — имейте в виду, что количество вершин может быть значительно. Это замечание касается способа выделения памяти под структуру дерева.

Этап 2. Приведение исходной матрицы стоимостей — это непосредственная реализация описанной ранее процедуры; детали мы оставляем в качестве упражнения.

Этап 3. Инициализация корня поискового дерева. Основной вопрос в том, будет ли вместе с вершиной дерева храниться и матрица стоимостей. Альтернативой является перевычисление матрицы стоимостей для текущей вершины на основе исходной. Это классический выбор между производительностью и требуемым объемом памяти.

Этап 4. Как мы уже обсуждали, выбор следующего ребра ветвления (k, l) определяет множества Y и $\overline{Y}$, непосредственно следующие за текущим множеством решений X. Ребро (k, l) для обеспечения сокращения перебора, нужно выбирать так, чтобы попытаться получить большую по величине нижнюю границу на множестве $\overline{Y}$, что облегчит проведение оценки для множества Y. Таким

образом, мы пытаемся заставить алгоритм выбирать на каждом шаге множество Y, пытаясь решить задачу всего за n шагов. Как применить эти идеи к выбору конкретного ребра ветвления (k,l)? В приведенной матрице стоимостей C', связанной с вершиной X, каждая строка и столбец имеют хотя бы по одному нулевому элементу (если это не так, то матрица C' не полностью приведена). Можно предположить, что ребра, соответствующие этим нулевым стоимостям, будут с большей вероятностью входить в оптимальный тур, чем ребра с большими стоимостями. Поэтому в качестве ребра ветвления мы выберем одно из них, при этом мы хотим, чтобы у множества $\overline{Y} = \{\overline{k,l}\}$ была как можно большая нижняя граница. Пусть ребро (k,l) имеет $c_{kl} = 0$, и обращаясь к процедуре вычисления нижней границы, мы видим, что для множества $\overline{Y}$ эта граница задается в виде

$$w(\overline{Y}) = w(X) + \min_{i,\, i \neq k} c_{il} + \min_{j,\, j \neq l} c_{kj}.$$

Следовательно, из всех ребер (k,l), у которых $c_{kl} = 0$ в текущей матрице C' мы выбираем то, которое дает наибольшее значение для нижней границы — $w(\overline{Y})$. Более подробный алгоритм выбора ребра ветвления на этапе 4 имеет вид

Procedure *Выбор ребра ветвления*
Begin
(*Пусть* S *— множество ребер* (i,j), *таких,*
что $c[i,j] = 0$ *в текущей матрице стоимостей* C.
Положим $D_{ij} =$ min (*стоимости в строке* i, *исключая* $c[i,j]$)
+
min (*стоимость в столбце* j, *исключая* $c[i,j]$)
)
1. *Вычисляем* D_{ij} *для всех* $(i,j) \in S$
2. *Выбираем следующее ребро ветвления* (k,l) *из условия*
$D_{kl} = \max(D_{ij})$ *для всех* $(i,j) \in S$
End

Этап 5. На этом этапе мы реализуем ветвление, и добавляем в поисковое дерево вершину $\overline{Y}$, следующую за X. Вычисляем нижнюю границу для множества $\overline{Y}$ следующим образом

$$w(\overline{Y}) = w(X) + D_{kl},$$

где D_{kl} уже вычислено на этапе 4.

Этап 6. Вершине Y, следующей за X, соответствует подмножество туров из множества X, содержащих то ребро (k,l), которое выбрано на этапе 4. При формировании матрицы стоимости, соответствующей вершине Y необходимо учитывать, что наша задача — нахождение тура, и все ребра, которые могут привести к более коротким циклам, должны быть запрещены. Детальное рассмотрение приводит к следующему алгоритму.

Procedure *Формирование матрицы для вершины* Y
begin
1. *Из матрицы* C *для вершины* X *исключаем*
строку k *и столбец* l.

if *(ребро (k,l) не изолировано от других ребер, включенных в тур по пути от корня дерева до вершины X)*
then
begin
2. *Находим начальный — p и конечный — q город пути*
3. $C[p,q]=\infty$
end
4. *Выполняем процедуру приведения полученной матрицы стоимостей. h = сумма констант приведения.*
5. *Вычисляем* $w(Y)=w(X)+h$
end

Этап 7. В конце концов, процесс ветвления приводит нас к множествам, содержащим так мало туров, что можно рассмотреть каждый из них и провести оценку для этой вершины без дальнейшего ветвления. Каждое ребро, про которое мы знаем, что оно содержится во всех турах из множества Y, сокращает размер матрицы стоимостей на одну строку и один столбец. Если в исходной задаче было n городов, а текущая матрица имеет размер 2×2, то уже $n-2$ ребер содержатся в туре из Y, поэтому множество Y содержит самое большее два тура, и мы можем провести исчерпывающую оценку, и получить конкретный тур коммивояжера. Вопрос о том, как их распознать, мы оставим в качестве упражнения.

Этап 8. Теперь нужно выбрать следующую вершину X, от которой необходимо проводить ветвление. Этот выбор довольно очевиден. Мы выбираем ту вершину, которая имеет в данный момент наименьшую нижнюю границу, и из которой в данный момент не выходят ветви, т. е.— это лист поискового дерева решений с минимальной нижней границей.

Этап 9. Поскольку мы выбрали новую вершину поискового дерева X в качестве текущей, то наша задача на этом этапе — получить соответствующую вершине X матрицу стоимостей. Если мы храним матрицы стоимостей вместе с вершинами поискового дерева, то матрица уже есть. В противном случае нам необходимо найти путь от корня до этой вершины и последовательно корректировать исходную матрицу стоимостей.

В заключение обсуждения этапов приведем (рис. Д.16) дерево решений для рассматриваемого примера, построенное на основе приведенных алгоритмов.

Обсуждение и дополнительные улучшения. Каких результатов в смысле вычислительной сложности мы можем ожидать? Обо всех более или менее эффективных алгоритмах, реализующих метод ветвей и границ для задачи коммивояжера, известно или имеются предположения, что их трудоемкость в худшем случае является экспоненциальной. В лучшем случае — если мы все время сокращаем размерность матрицы стоимостей, то оценка вычислительной сложности является, полиномиальной. Это очевидно, т. к. оценка трудоемкости каждого из этапов алгоритма — полиномиальна по линейному размеру (n) матрицы стоимостей, и в лучшем случае основной цикл выполняется не более чем n раз — мы на каждом шаге выбираем одно ребро в тур, который состоит из n ребер. Та-

ким образом, мы имеем очень большой разброс ожидаемого времени выполнения при фиксированной размерности матрицы стоимости n, зависящий от численных значений ее элементов. Мы имеем количественно-параметрический алгоритм с сильной параметрической зависимостью. Теоретический анализ ожидаемой трудоемкости очень сложен и часто выходит за рамки наших аналитических возможностей.

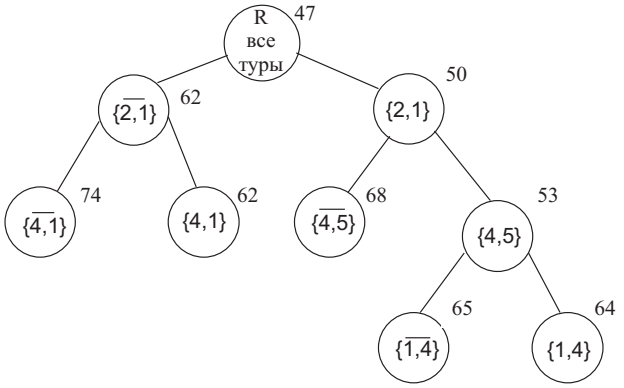


Рисунок Д.16. Дерево решений для задачи коммивояжера, полученное алгоритмом метода ветвей и границ для матрицы стоимостей из табл. Д.6.4

Несмотря на определенный пессимизм в теории, экспериментальные данные показывают, что тщательно реализованный алгоритм справляется, в большинстве случаев, с задачами размерности порядка 50 за реальное время. Особо подчеркнем, что мы получаем гарантированно точное решение задачи целочисленного программирования, принадлежащей классу NPC.

Можно рассматривать ряд дополнительных улучшений этого алгоритма, касающихся способов выбора ребра ветвления и использования простых (полиномиальных) алгоритмов для получения начального тура с целью возможного отбрасывания некоторых вершин поискового дерева решений уже на ранних шагах ветвления.

Набор упражнений Д.6

К введению.

Д.6.1. Попробуйте найти решение для задачи о сумме в 22 флорина, напомним, что у Вас есть монеты номиналом 2, 3, 5, 6 и 7 флоринов. Какой у Вас получился ответ, и сколько вариантов суммы Вы перебрали?

К решению задачи упаковки методом динамического программирования

Д.6.2. Из постановки задачи одномерной упаковки известны значение объема упаковки — V и объемы типов грузов — v_i. Попытайтесь, как

можно точнее, ограничить множество точек перебора координатным прямоугольным параллелепипедом, и найти формулу для определения количества точек, которое он содержит.

Д.6.3. Если Вы обратили внимание, то последняя строка табл. Д.6.2, иллюстрирующая решение задачи упаковки, заполнена знаками вопроса — на основании имеющейся информации об упаковки всех дискретов объема грузами первого, и второго типа найдите, используя основное функциональное уравнения Беллмана, решение для объема 10 и трех типов грузов.

Д.6.4. Как будет выглядеть фрагмент дерева рекурсий, если в среднем груз каждого типа размещается в объеме упаковки не более трех раз — $k = 3$. Нарисуйте несколько уровней этого дерева аналогично рис. Д.13.

Д.6.5. Используя рекурсивный алгоритм решения задачи упаковки, вычислите значение $f_3(14)$ и постройте дерево решений, аналогичное тому, что приведено на рис. Д.11. Объемы и стоимости типов грузов приведены в таблице.

i	v_i	c_i
1	3	5
2	4	6
3	5	7

К решению задачи коммивояжера методом ветвей и границ

Д.6.6. Сформулируйте математически условие для тура в постановке задачи коммивояжера, как задачи целочисленного программирования в пространстве $E_z^{n^2-n}$. Ребра, указанные компонентами вектора $x_i = 1$, должны составлять тур. Воспользуйтесь, например, математическим описанием перестановок n чисел, и отображениями, введенными в постановке задачи.

Д.6.7. В постановке задачи коммивояжера в пространстве $E_z^{n^2-n}$ достаточно просто определить M — количество точек этого пространства, у которых только n координат равны единице, а остальные нулевые — именно они и составляют полное множество перебора в этой постановке

$$M = C^n_{n^2-n},$$

это есть количество способов распределить n единиц по $n^2 - n$ позициям.

Попытайтесь оценить отношение $(n-1)!/M$ — именно такова доля «настоящих» туров в общем объеме перебора. Воспользуйтесь формулой для оценки факториала из раздела 1, и свойством логарифма: $\ln(1-x) \approx -x$, $0 \leqslant x \ll 1$.

Д.6.8. Оцените, сколько туров входит в подмножество $\{k, l\}$ на первом шаге ветвления при решении задачи коммивояжера с n городами?

Д.6.9. Предложите другие идеи по выбору ребра ветвления в алгоритме метода ветвей и границ для задачи коммивояжера. Как вы думаете, может ли усложнение этого этапа, например, за счет рассмотрения большего количества ребер, привести к сокращению общего времени решения? На наш взгляд наилучшим ответом будет проведение вычислительных экспериментов.

Литература

[1] Garnier, R. and Taylor, J. (1992) *Discrete Mathematics for New Technology*, Bristol: Institute of Physics Publishing.

[2] Gersting, J. L. (1999) *Mathematical Structures for Computer Science*, 4th edn, San Francisco: W. H. Freeman.

[3] Johnsonbaugh, R. (2001) *Discrete Mathematics*, 5th edn, New Jersey: Prentice Hall.

[4] Rosen, K. H. (1998) *Discrete Mathematics and Its Applications*, New York: McGraw-Hill.

[5] Ross, K. A. and Wright, C. R. B. (1999) *Discrete Mathematics*, 4th edn, New Jersey: Prentice Hall.

[6] Truss, J. K. (1999) *Discrete Mathematics for Computer Scientists*, 2nd edn, Harlow: Addison-Wesley.

[7] Little R. D. C., Murty K. G., Sweeney D. W., Karel C. *An Algorithm for the Traveling Salesman Problem*. Oper. Res., 11: 979–989 (1963) (IM).

[8] Андерсон Дж. *Дискретная математика и комбинаторика*: Пер. с англ. М.: Издателький дом «Вильямс». 2003.

[9] Андреева К.Г., Никитов С.А. *Оптимизация использования сырья в задаче линейного раскроя*// Информационные технологии. 2003. № 11. с. 24 – 29.

[10] Ахо А., Хопкрофт Дж., Ульман Дж. *Структуры данных и алгоритмы*: Пер. с англ.: — М.: Издательский дом «Вильямс», 2001 г.

[11] Беллман Р., Дрейфус Р. *Прикладные задачи динамического программирования*: Пер. с англ. М.: Наука, Главная редакция физико-математической литературы, 1965.

[12] Бондаренко В. А. *О сложности дискретных задач* (http://edu.yar.ru/russian/pedbank/sor-pro/bondarenko).

[13] Бондаренко В. А. *Полиэдральные графы и сложность в комбинаторной оптимизации*. Ярославль: Ярославский госуниверситет, 1995.

[14] Валеева А.Ф., Гареев И.Ф., Мухачева Э.А. *Задача одномерной упаковки: рандомизированный метод динамического перебора и метод перебора с усечением*// Информационные технологии. Приложение. 2003. № 2. 24 с.

[15] Грин Д., Кнут Д. *Математические методы анализа алгоритмов*. М.: Мир, 1987. — 120 с.

[16] Грэхем Р., Кнут Д., Паташник О. *Конкретная математика. Основание информатики*: Пер. с англ. М.: Мир, 1998.

[17] Гудман С., Хидетниеми С. *Ведение в разработку и анализ алгоритмов*. М.: Мир, 1981.

[18] Ильин В. А., Садовничий В. А., Сендов Бл. Х. *Математический анализ.* М.: Наука. Главная редакция физико-математической литературы, 1979.

[19] Кнут Д. *Искусство программирования.* Тома 1, 2, 3. 3-е изд. Пер. с англ.: Уч. пос. М.: Изд. дом «Вильямс», 2001 г.

[20] Кормен Т., Лейзерсон Ч., Ривест Р. *Алгоритмы: построение и анализ.* М.: МЦНМО, 2001 г.

[21] Макконелл Дж. *Анализ алгоритмов. Вводный курс.* М.: РИЦ Техносфера, 2002.

[22] Макконелл Дж. *Основы современных алгоритмов. 2-е дополненное издание.* М.: Техносфера, 2004.

[23] Нефедов В. Н., Осипова В. А. *Курс дискретной математики.* М.: МАИ, 1992.

[24] Новиков Ф. А. *Дискретная математика для программистов.* СПб.: Питер, 2001.

[25] Ульянов М. В., Гурин Ф. Е., Исаков А. С., Бударагин В. Е. *Сравнительный анализ табличного и рекурсивного алгоритмов точного решения задачи одномерной упаковки*// Exponenta Pro Математика в приложениях. 2004. № 2(6). С. 64–70.

[26] Шевченко В. Н. *Качественные вопросы целочисленного программирования*, М.: Физматлит, 1995.

[27] Юдин Д.Б., Горяшко А.П., Немировский А.С. *Математические методы оптимизации устройств и алгоритмов АСУ.* М.: Радио и связь, 1982.

[28] Яблонский С. В. *Введение в дискретную математику.* М.: Наука, 1986.

Предметный указатель

Производство книг на заказ
Издательство «Техносфера»
тел.: (495) 234-01-10
e-mail: knigi@technosphera.ru

Реклама в книгах:
• модульная
• статьи

Подробная информация о книгах на сайте
http://www.technosphera.ru

Хаггарти Род

Дискретная математика для программистов
Издание 2-е, исправленное

Компьютерная верстка – С.А. Кулешов
Дизайн – М.В. Лисусина
Выпускающий редактор – О.Н. Кулешова
Ответственный за выпуск – С.А. Орлов

Формат 70 x 100/16. Печать офсетная.
Гарнитура Computer modern LaTeX
Печ.л. 25. Тираж 1000 экз., Зак. №
Бумага офсет №1, плотность 65 г/м^2

Издательство «Техносфера»
Москва, ул. Краснопролетарская, д.16, стр.2

Отпечатано в типографии ОАО ПИК «Идел-Пресс»
420066 г. Казань, ул. Декабристов, 2.