ПЕДАГОГИЧЕСКОЕ ОБРАЗОВАНИЕ

С. М. Окулов

ДИСКРЕТНАЯ МАТЕМАТИКА
ТЕОРИЯ И ПРАКТИКА РЕШЕНИЯ ЗАДАЧ ПО ИНФОРМАТИКЕ

БИНОМ

ПЕДАГОГИЧЕСКОЕ ОБРАЗОВАНИЕ

С. М. Окулов

ДИСКРЕТНАЯ МАТЕМАТИКА
ТЕОРИЯ И ПРАКТИКА РЕШЕНИЯ ЗАДАЧ ПО ИНФОРМАТИКЕ

Учебное пособие

Москва
БИНОМ. Лаборатория знаний
2 0 0 8

УДК 519.85(075)
ББК 22.174я7
О-52

Рецензенты:
академик РАО, доктор педагогических наук, профессор *А. А. Кузнецов*
доктор технических наук, профессор *В. Н. Комаров*

Окулов С. М.

О-52 Дискретная математика. Теория и практика решения задач по информатике : учебное пособие / С. М. Окулов. — М. : БИНОМ. Лаборатория знаний, 2008. — 422 с. : ил. — (Педагогическое образование).

ISBN 978-5-94774-498-9

В учебном пособии даны ключевые разделы дискретной математики с практической реализацией алгоритмических решений. Книга написана на основе лекционного курса и практических занятий для студентов факультета информатики Вятского государственного гуманитарного университета, а также спецкурса, читаемого автором для школьников, занимающихся информатикой по углубленной программе.

Для студентов высших учебных заведений, а также старшеклассников, углубленно изучающих информатику.

УДК 519.85(075)
ББК 22.174я7

По вопросам приобретения обращаться:
«БИНОМ. Лаборатория знаний»
Телефон: (499) 157-5272
e-mail: Lbz@aha.ru, http://www.Lbz.ru

ISBN 978-5-94774-498-9

© Окулов С. М., 2008
© БИНОМ. Лаборатория знаний, 2008

ОГЛАВЛЕНИЕ

Предисловие 7

Глава 1. Основные методы дискретной математики (счет и перебор) 10

1.1. Счет и перебор 10
1.2. Асимптотические обозначения и основная теорема . . 17
1.3. Эффект «комбинаторного взрыва» 20
Упражнения и задачи 22
Комментарии 24

Глава 2. Основные комбинаторные принципы и понятия в примерах 25

2.1. Принципы сложения и умножения 25
2.2. Подмножества 25
2.3. Принцип включения и исключения 26
2.4. Выборки 28
2.5. Размещения с повторениями 28
2.6. Размещения без повторений 29
2.7. Сочетания без повторений 30
2.8. Бином Ньютона и полиномиальная формула (комбинаторный смысл) 32
2.9. Сочетания с повторениями 33
2.10. Перестановки без повторений 33
2.11. Перестановки с повторениями 38
2.12. Задача о размещениях 39
2.13. Разбиения 42
2.14. Разбиения на циклы 43
2.15. Разбиение числа на слагаемые 45
Упражнения и задачи 46
Комментарии 51

Глава 3. Перечисление комбинаторных объектов 52
3.1. Общая схема генерации комбинаторных объектов . . 52
3.2. Генерация перестановок без повторений 53
3.3. Генерация сочетаний без повторений 54
3.4. Генерация размещений без повторений 55
3.5. Генерация перестановок с повторениями 57
3.6. Генерация сочетаний с повторениями 57
3.7. Генерация размещений с повторениями 57
3.8. Генерация подмножеств 58
3.9. Генерация разбиений 60
3.10. Генерация разбиений на циклы 66
3.11. Генерация разбиений числа на слагаемые 73
Упражнения и задачи 74
Комментарии 75

Глава 4. Рекуррентные и нерекуррентные формулы 76
4.1. Простые примеры 76
4.2. Числа Фибоначчи 77
4.3. Числа Каталана 82
4.4. Схема нахождения общего решения линейных рекуррентных уравнений 86
4.5. Производящие функции 90
4.6. Ладейные полиномы 97
4.7. Аддитивность задач, или динамическое программирование 101
Упражнения и задачи 106
Комментарии 110

Глава 5. Понятие графа, основные методы просмотра вершин графа 111
5.1. Терминология 111
5.2. Способы представления графа 112
5.3. Поиск в глубину 114
5.4. Поиск в ширину 116
5.5. Основные понятия 117
Упражнения и задачи 124
Комментарии 129

Глава 6. Деревья 130
6.1. Определение дерева 130
6.2. Перечисление остовных деревьев связного помеченного графа 131
6.3. Матричная формула Кирхгофа 134

6.4. Алгоритм представления дерева в виде последовательности чисел . . . 135
6.5. Остовные деревья минимального веса . . . 137
6.6. Задача Штейнера . . . 141
Упражнения и задачи . . . 143
Комментарии . . . 144

Глава 7. Связность . . . 145

7.1. Вершинная и реберная связность . . . 145
7.2. Метод нахождения блоков графа . . . 147
7.3. Теорема Менгера . . . 149
7.4. Связность в орграфе . . . 151
Упражнения и задачи . . . 154
Комментарии . . . 155

Глава 8. Циклы . . . 156

8.1. Эйлеровы графы . . . 156
8.2. Гамильтоновы графы . . . 158
8.3. Фундаментальное множество циклов . . . 161
8.4. Матроиды . . . 166
Упражнения и задачи . . . 172
Комментарии . . . 173

Глава 9. Покрытия и независимость . . . 174

9.1. Основные понятия . . . 174
9.2. Метод генерации всех максимальных независимых множеств вершин графа . . . 175
9.3. Клики . . . 179
9.4. Доминирующие множества . . . 180
9.5. Паросочетания . . . 185
9.6. Матроиды трансверсалей . . . 196
9.7. Диаграмма взаимосвязей между задачами . . . 198
Упражнения и задачи . . . 201
Комментарии . . . 203

Глава 10. Планарные графы . . . 204

10.1. Основные понятия . . . 204
10.2. Формула Эйлера . . . 204
10.3. Алгоритм укладки графа на плоскости . . . 206
Упражнения и задачи . . . 214
Комментарии . . . 215

Глава 11. Раскраска вершин графа 216
11.1. Хроматическое число 216
11.2. Метод правильной раскраски 217
11.3. Методы поиска минимальной раскраски 219
Упражнения и задачи 222
Комментарии 223

Глава 12. Кратчайшие пути в графе 224
12.1. Постановка задачи. Вывод пути 224
12.2. Алгоритмы поиска кратчайших путей 226
Упражнения и задачи 234
Комментарии 235

Глава 13. Потоки в сетях 236
13.1. Основные понятия и постановка задачи 236
13.2. Алгоритм К. Эдмондса—Р. Карпа 237
13.3. Введение в метод блокирующих потоков или алгоритм Е. А. Диница 244
13.4. Модификация алгоритма Е. А. Диница 252
Упражнения и задачи 260
Комментарии 262

Ответы и решения 263
Задачи для самостоятельного решения 353

Приложение 1. Математические факты и доказательства отдельных теорем 375
Приложение 2. Описание основных элементов языков программирования Паскаль, визуального Бейсика и C++ 396

Литература 414
Предметный указатель 416

Надежде Григорьевне Окуловой
посвящаю

ПРЕДИСЛОВИЕ

С момента появления компьютера и начала использования его для решения самых разнообразных задач получила развитие системная область знаний математики, связанная с программированием. Компьютер, при всех его огромных возможностях, может работать только с конечным множеством объектов (задача проецируется на конечномерный случай), причем на этом множестве объектов должно быть определено отношение порядка [21]. Только в этом случае проблема (задача) подвластна компьютеру при условии, что найден эффективный способ ее решения. Эффект «комбинаторного взрыва» показывает, что компьютер бывает бессилен при решении даже очень простых задач. Обобщая, можно сказать, что практически компьютер решает только две задачи: подсчет количества объектов определенной природы (счет) или поиск объекта (из известного множества объектов), удовлетворяющего определенным условиям (перебор).

Потребности практики, а именно программирования, определяют развитие «стыка» информатики и математики. Эта область знаний не исчерпывается одним предметом — «дискретной математикой», но он является характерным, основополагающим.

Структура книги

Главы 1, 2, 3, 4 — это материал по комбинаторике. Первой вводной темой являются проблемы счета и перебора (или выбора). К этим двум проблемам сводится большинство задач, решаемых на компьютере. Раскрывается суть «комбинаторного взрыва» — показывается ограниченность возможностей компьютера. И весь дальнейший курс строится под лозунгом преодоления этой огра-

ниченности. Тема прекрасно «ложится» на компьютерный вариант практики и служит пропедевтикой всех тем, изучаемых в дальнейшем. При изучении комбинаторных объектов (комбинаторики) следует, на наш взгляд, не ограничиваться задачами подсчета, рассматривая и вопросы перечисления комбинаторных объектов, а также задачи вычисления номера комбинаторного объекта в соответствии с установленным отношением порядка так, как это сделано, например, в работе [20]. Рассмотрение комбинаторных чисел «плавно подводит» к рекуррентным соотношениям и методам исчисления конечных сумм.

Завершается курс темой «Алгоритмы на графах» (главы с 5 по 13). Эта тема тесно связана с рассмотренными комбинаторными проблемами. Отметим возможность различного по сложности уровня обобщения теоретического материала и построения компьютерно-ориентированных практических занятий.

Доказательства теорем приведены в приложении 1. Они даются автором, как правило, в курсе дискретной математики по специальности «прикладная математика и информатика».

Особенности книги

Книга является учебным пособием по дискретной математике и написана по результатам педагогической деятельности автора, осуществляемой в рамках:

— спецкурса для школьников старших классов физико-математического лицея;

— курса дискретной математики для студентов педагогической специальности «учитель информатики»;

— курса дискретной математики для студентов, специализирующихся в прикладной математике и информатике.

Текст книги — это синтез лекционных и практических занятий, проводимых по всем трем направлениям деятельности. Автор преследовал основную цель — написать просто даже о сложном так, чтобы пособие было полезно и школьнику, и учителю информатики, и студенту, и преподавателю вуза.

Используемые обозначения

Для записи алгоритмов используется минимальное подмножество языка программирования Паскаль, синтаксис которого счита-

ется очевидным. Перевод в другие системы программирования не вызывает трудностей. Соответствие между используемой нотацией Паскаля и языками C++ и визуальным Бейсиком приведено в приложении 2. Не используются различного рода псевдокоды, которые есть как в отечественной, так и в переводной литературе, по той простой причине, чтобы читатель, набрав текст и уточнив логику, мог получить действующую программу. Заглавные буквы, используемые для выделения конструкций языка, кое-кого приводят в недоумение. Автор осознанно делает это, ибо фрагмент текста при этом становится не монотонным, а образным и легче воспринимаемым. Сочетание зрительного образа, вербального разъяснения и действий с объектом, в данном случае с программой, по мнению психологов, является наиболее действенным в системе восприятия и запоминания человека.

В фигурных скобках {} даются комментарии. В угловых скобках <> — фрагменты логики, описанные словами. Если в формулировке задачи не оговариваются параметры входных данных, например значение *n*, то право выбора предоставлено читателю.

Благодарности

Автор выражает благодарность Тамаре Николаевне Котельниковой, которая много лет приводит его рукописи в приличный с точки зрения русского языка вид, а также многочисленным школьникам и студентам, творчески изучавшим курс. Роман Александрович Веснин, Максим Анатольевич Корчемкин, Артем Николаевич Алалыкин оказали помощь автору в разработке материалов для отдельных параграфов, за что им огромная признательность. Ульяна Александровна Токмакова помогла автору в оформлении рукописи, профессионально переоформив все рисунки. Особо хотелось бы поблагодарить Ирину Анатольевну Маховую, главного редактора издательства «БИНОМ. Лаборатория знаний» — ее профессионализм, приложенный к книгам автора, внес в них то, чего им так не хватало много лет.

ГЛАВА 1

ОСНОВНЫЕ МЕТОДЫ ДИСКРЕТНОЙ МАТЕМАТИКИ (СЧЕТ И ПЕРЕБОР)

1.1. Счет и перебор

Приведем несколько примеров, в которых требуется подсчитать количество объектов определенной природы.

Пример 1.1. Подсчитать количество последовательностей из натуральных чисел от 1 до n, в которые каждое из этих чисел входит по одному разу.

На первое место в последовательности можно записать любое из n чисел; на второе любое из оставшихся $n-1$ чисел и т. д. Общее количество последовательностей равно произведению $1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$. Это произведение обозначают $n!$ (читается как n факториал).

При $n = 7$ значение $n! = 5040$, а при $n = 8$ — уже 40 320. Для вычисления и хранения чисел такого порядка в компьютере использовать величину типа *Integer* нельзя. Аналогично и с величинами типа *LongInt*, так как последнее значение n, для которого можно сохранить значение факториала, равно 12 ($13! = 6\,227\,020\,800$). Вычисление для больших значений n рассмотрено в книге [20]. ◀

Пример 1.2. Подсчитать количество единиц в двоичном представлении целого положительного числа.

Рассмотрим величины типа *Word*. Длина двоичного представления чисел равна 16 битам ($n = 16$). Например, в двоичной последовательности 0101101001001001 содержится 7 единиц.

Первый вариант решения

```
Const n=16;
Var a, cnt, i:Word;
Begin
  ReadLn(a);
  cnt:=0;
```

```
  For i:=1 To n Do
    If a And (1 ShL (i-1))=1 Then cnt:=cnt+1;
    {a ShL b – сдвиг величины a влево на b разря-
    дов}
  WriteLn(cnt);
End.
```

За 16 операций сравнения вычисляется результат.

Примечание. Для вывода двоичного представления числа можно использовать следующую простую процедуру.

```
Procedure Rec(x:Word);
  Begin
    If x>1 Then Rec(x Div 2);
    Write(x Mod 2);
  End;
```

Второй вариант решения

```
...
While a<>0 Do Begin
  cnt:=cnt+1;
  a:=a And (a-1);
End;
...
```

В этом случае количество операций пропорционально количеству единиц в двоичном представлении числа. Используется тот факт, что операция a *And* $(a-1)$ «убирает» одну единицу из двоичного представления числа.

Например:

a	0101101001000000
$a-1$	0101101000111111
a *And* $(a-1)$	0101101000000000

Третий вариант решения. Вычисление за 4 операции независимо от количества единиц в представлении числа.

Рассмотрим идею решения на примере. Дано 0101101001001001. Пронумеруем биты справа налево. Выделим нечетные биты, а на место четных битов запишем нулевые значения. Затем выделим четные биты, на место нечетных битов запишем нулевые значения, сдвинем получившуюся величину на 1 разряд вправо. Выполним обычное сложение двух найденных величин. Результаты этих «манипуляций» приведены в табл. 1.1.

Таблица 1.1

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	0	1	0	1	1	0	1	0	0	1	0	0	1	0	0	1
Нечетные биты *a*	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1
Четные биты *a*, сдвинутые на один разряд вправо	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0
Новое значение *a*	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1

С новым значением *a* выполним аналогичную операцию, но теперь будем работать с группами из двух соседних битов. Действие представлено в табл. 1.2.

Таблица 1.2

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1
Биты 1, 2, 5, 6, 9, 10, 13, 14 из *a*	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
Биты 3, 4, 7, 8, 11, 12, 15, 16 из *a*, сдвинутые на 2 разряда вправо	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
Новое значение *a*	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0

Продолжим вычисления. На этот раз группировать будем по 4 бита. Результаты преобразований см. в табл. 1.3.

Таблица 1.3

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0
Биты 1, 2, 3, 4, 9, 10, 11, 12 из *a*	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
Биты 5, 6, 7, 8, 13, 14, 15, 16 из *a*, сдвинутые на 4 разряда вправо	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
Новое значение *a*	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1

И наконец, последний шаг представлен в табл. 1.4.

Таблица 1.4

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1
Биты 1, 2, 3, 4, 5, 6, 7, 8 из *a*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Биты 9, 10, 11, 12, 13, 14, 15, 16 из *a*, сдвинутые на 8 разрядов вправо	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Новое значение *a*	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Значение *a* равно 7, это и есть ответ. Количество единиц в двоичном представлении числа (или сумма значений разрядов слова)

вычислено за четыре описанных действия. При n, равном 32, 64, 128, потребуется соответственно 5, 6 и 7 действий.

```
Const L=4;
      C=Array[1..2*L] Of Word=($5555, $AAAA,
        $3333, $CCCC, $0F0F, $F0F0, $00FF, $FF00);
Var a,s,i:Word;
Begin
  ReadLn(a);
  s:=1;
  For i:=1 To L Do Begin
    a:=(a And C[2*i-1])+((a And C[2*i]) ShR s);
    s:=s ShL 1;
  End;
  WriteLn(a);
End.
```

Разъяснение записи констант приведено в табл. 1.5.

Таблица 1.5

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
$5555	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$AAAA	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
$3333	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$CCCC	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
$0F0F	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$F0F0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
$00FF	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$FF00	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Четвертый вариант решения (гипотетический). В таблице заранее зафиксирован ответ для каждого целого положительного числа заданного диапазона. Размер таблицы 2^n (при $n = 16$ требуется хранить таблицу из 64 Кбайт при условии, что адресация осуществляется на уровне байтов). Для больших значений n объем требуемой памяти весьма значителен. Пусть $n = 3$. Данные приведены в табл. 1.6.

Таблица 1.6

Двоичное представление числа (является адресом ячейки памяти)	Количество единиц (хранится в памяти)
000	0
001	1
010	1
011	2
100	1
101	2
110	2
111	3

За одно обращение к памяти получается результат. Значение величины является адресом ячейки памяти. Этот вариант решения иллюстрирует один из фундаментальных принципов информатики —

«принцип косвенной адресации». Это первый аспект. Второй аспект заключается в том, что, «ускоряясь» в одном, мы проигрываем в другом. Выигрывая в быстродействии, проигрываем в памяти. А если смысл в том, чтобы искать более эффективные (по какому-то из критериев!) алгоритмы? Компьютер обладает огромным быстродействием, а мы говорим, что 16 операций — это плохо, а 4 — хорошо, одна еще лучше, но слишком много требуется памяти. Образно выражаясь, вся история информатики связана с борьбой за эффективность алгоритмов работы как самого компьютера, так и программ, и увеличение производительности компьютера не уменьшает накала этой борьбы, ибо, как будет показано, этот рост не решает многих проблем. ◀

Общая постановка проблемы перебора. Пространство решения задачи описывается семейством n упорядоченных множеств $A_1, A_2, \ldots, A_n$ (A_i могут и совпадать). Требуется ответить на вопрос, существует ли вектор $(a_1, a_2, \ldots, a_n)$, где $a_1 \in A_1$, $a_2 \in A_2$, $\ldots$, $a_n \in A_n$, удовлетворяющий определенным условиям, или перечислить все возможные векторы, удовлетворяющие этим условиям. На стадии предварительной обработки (до фактического перебора вариантов) обычно исследуется структура множеств A_i с целью уменьшения размерности или установления определенных зависимостей (задача о ферзях, см. пример 1.3), позволяющих сократить перебор.

Пример 1.3. На шахматной доске $n \times n$ требуется расставить n ферзей, не атакующих друг друга. Пусть $n = 8$, и наша задача заключается в поиске всех возможных расстановок ферзей.

Первый вариант решения. Из 64 клеток доски требуется выбрать 8 и проверить условие: будут ли ферзи атаковать друг друга, если их поставить на эти клетки. Все A_i совпадают, и каждое множество состоит из всех 64 клеток доски ($i = 1, 2, \ldots, 8$). Сокращение перебора определяется только тем фактом, что на одну клетку поля нельзя ставить двух и более ферзей. Число способов расстановки определяется числом способов выбора 8 клеток из 64 (число сочетаний — C_{64}^8), а это порядка $4{,}4 \cdot 10^9$ вариантов. Если пренебречь временем проверки условия «атакуют — не атакуют», то и время решения задачи пропорционально этому значению.

Второй вариант решения. Каждый столбец поля может содержать только одного ферзя. В каждом A_i уже не 64 элемента, а восемь, что существенно сокращает перебор. Не задумываясь, ставим ферзя

[. . .]

Рассмотрим схему применения «правила 72» к оценке времени работы программы на примере. Предположим, что есть алгоритм типа AL_3 с экспоненциальным временем работы, и задача с размерностью $n = 40$ решается за 10 секунд. При увеличении n на единицу время возрастет на 12%. Это следует из анализа экспоненциальной зависимости. Примем это как факт. «Правило 72» гласит, что при увеличении n на 6 единиц время увеличится в 2 раза, а при увеличении n на 60 время увеличится в 1000 раз и составит 10^4 секунд. Таким образом, при $n = 100$ алгоритм работает 10^4 секунд, а при $n = 160$ — в течение 10^7 секунд или примерно 8 часов.

Упражнения и задачи

1.1. Написать программу поиска всех способов расстановки n ферзей на шахматной доске $n \times n$ клеток так, чтобы они не били друг друга.

Примечание. В формулировках следующих задач слово «шахматная» не используется.

1.2. Путем поворота доски и зеркальных отображений часть расстановок ферзей совпадает. Назовем их симметричными. Изменить решение предыдущей задачи так, чтобы осуществлялся поиск только несимметричных расстановок.

1.3. Изменить решение задачи 1.1 так, чтобы находилась только одна расстановка ферзей.

1.4. Написать программу поиска обхода конем доски $n \times m$. Конь должен побывать на каждой клетке доски только один раз. Начальная позиция коня произвольна.

1.5. Решить задачу 1.4, используя следующее правило: конь очередным ходом ставится на непройденную клетку не по правилу обхода возможных продолжений по «часовой стрелке», а в очередности возрастания количества возможных ходов, которые можно сделать из клетки-кандидата на очередной ход. Если таких клеток несколько, то выбирается любая.

1.6. Магараджа — это фигура, которая объединяет в себе ходы коня и ферзя. Для доски 10×10 написать программу поиска расстановки 10 мирных (не бьющих друг друга) магараджей.

1.7. Написать программу заполнения таблицы размером 5×5 числами от 1 до 25 по следующему принципу. Если в клетку с координатами (x, y) записано число i $(1 \leqslant i \leqslant 25)$, то число $i + 1$ записывается в клетку с координатами (z, w), вычисляемыми по одному

[. . .]

1.16. *Задача о рюкзаке.* Даны предметы n различных типов. Количество предметов каждого типа не ограничено. Каждый предмет типа i имеет вес w_i и стоимость v_i, $i = 1, 2, \ldots, n$. Написать программу для определения максимальной стоимости груза, вес которого равен W.

1.17. *Задача о коммивояжере.* Классическая формулировка задачи известна уже более 200 лет: имеются n городов, расстояния между которыми заданы; коммивояжеру необходимо выйти из какого-то города, посетить остальные $n-1$ городов точно по одному разу и вернуться в исходный город. При этом маршрут коммивояжера должен быть минимальной длины (стоимости). Написать программу решения задачи.

1.18. Написать программу расстановки на доске $n \times n$ $(n = 5)$ n ферзей так, чтобы наибольшее число полей оказалось вне боя ферзей.

1.19. Написать программу определения наименьшего числа ферзей, которых можно расставить на доске $n \times n$ так, чтобы они держали под боем все ее свободные поля.

1.20. Написать программу расстановки на доске как можно большего количества ферзей так, чтобы при снятии любого из них появлялось ровно одно неатакованное поле.

1.21. Найти решение рекуррентных соотношений [14]:

а) $T(n) = 2T(\sqrt{n}) + \log n$;
б) $T(n) = 3T(n/4) + n$;
в) $T(n) = T(2n/3) + 1$;
г) $T(n) = 3T(n/4) + n \log n$;
д) $T(n) = 2T(n/2) + n \log n$.

Комментарии

Факты, изложенные в данной главе, так или иначе приводятся в любом учебнике по дискретной математике. Пример 1.2 рассматривается в [22]. Задача о расстановке ферзей общеизвестна, главное — в особенностях ее изложения. Данные и действия над данными интегрируются в изложении в единое целое, и в результате получается органическое целое под названием программа. Наиболее подробное изложение основной теоремы приведено в [14]. На качественном уровне в сжатом виде она приводится в [2] и сверхдетально изложена в [29]. Эффект «комбинаторного взрыва» — компьютерный фольклор, и его использование — это скорее дань уважения и привязанности старым и добрым книгам (хоть и переизданным — *Бентли Дж.* Жемчужины программирования. — СПб. : Питер, 2002), входящим в «золотой фонд» классики по информатике.

ОКУЛОВ Станислав Михайлович
Декан факультета информатики Вятского государственного гуманитарного университета, кандидат технических наук, доктор педагогических наук, профессор. Автор 9 изобретений и автор (соавтор) 12 книг по информатике для школьников и студентов.

Область интересов: развитие интеллектуальных способностей школьника при активном изучении информатики, исследование ассоциативных систем обработки информации.

С 1993 по 2003 год деятельность в вузе совмещал с работой учителя информатики. За это время его ученики отмечены 33 дипломами (1-й и 2-й степени) на Российских олимпиадах школьников по информатике; трое из них представляли Россию на международных олимпиадах.