

РАЗБОР ЗАДАЧ РАЙОННОЙ ОЛИМПИАДЫ

Аннотация

Ниже предлагается разбор большинства задач муниципального этапа Всероссийской олимпиады школьников по информатике 2011–2012 учебного года. Не рассмотрены только задачи «Числа» для 7–8 и 9 классов, но они аналогичны одноимённой задаче для 10–11 классов. Возможно, составители заданий олимпиады имели в виду какие-то иные способы решения задач. Но предложенные ниже решения проходят все тесты составителей и работают весьма быстро.

1 Задача «Фишка»

Эта задача предлагалась в 7–8 классах.

Задача. Фишка может двигаться только вперед по полю длины N . Длина хода фишки не более K . Найти число различных вариантов ходов, при которых фишка может пройти поле от начала до конца. Например, при $N = 3$, $K = 2$ — возможные пути: $(1,1,1)$, $(1,2)$, $(2,1)$, т.е. возможны 3 варианта.

Формат входных данных

Файл содержит два числа N и K , разделенных пробелами. $1 \leq N \leq 15$, $1 \leq K \leq 15$, $K \leq N$.

Формат выходных данных

Выходной файл содержит число вариантов.

Пример входных и выходных файлов

Task2.in	Task2.out
3 2	3

Решение

Идея

Для примера рассмотрим случай $N = 5$, $K = 3$ (рис. 1.1).

Определение. Назовём размерностью задачи длину поля, ещё не пройденного фишкой.

После каждого хода фишки размерность задачи уменьшается на величину сделанного хода. Фишка успешно прошла всё поле, если после очередного хода размерность задачи стала равна нулю. Чтобы подсчитать количество вариантов прохода фишки через всё поле длины 5, надо сложить решения той же задачи для поля длиной 4, поля длиной 3 и поля длиной 2. Происходит уменьшение размерности задачи. Соответствующее дерево игры представлено на рис. 1.1. Вес дуг равен длинам сделанных ходов. Метка вершины — размерность задачи после очередного хода. Нам надо подсчитать количество путей, ведущих из корня в листья дерева, или, что то же, количество листьев этого дерева. (Все листья — это задачи размерности ноль.)

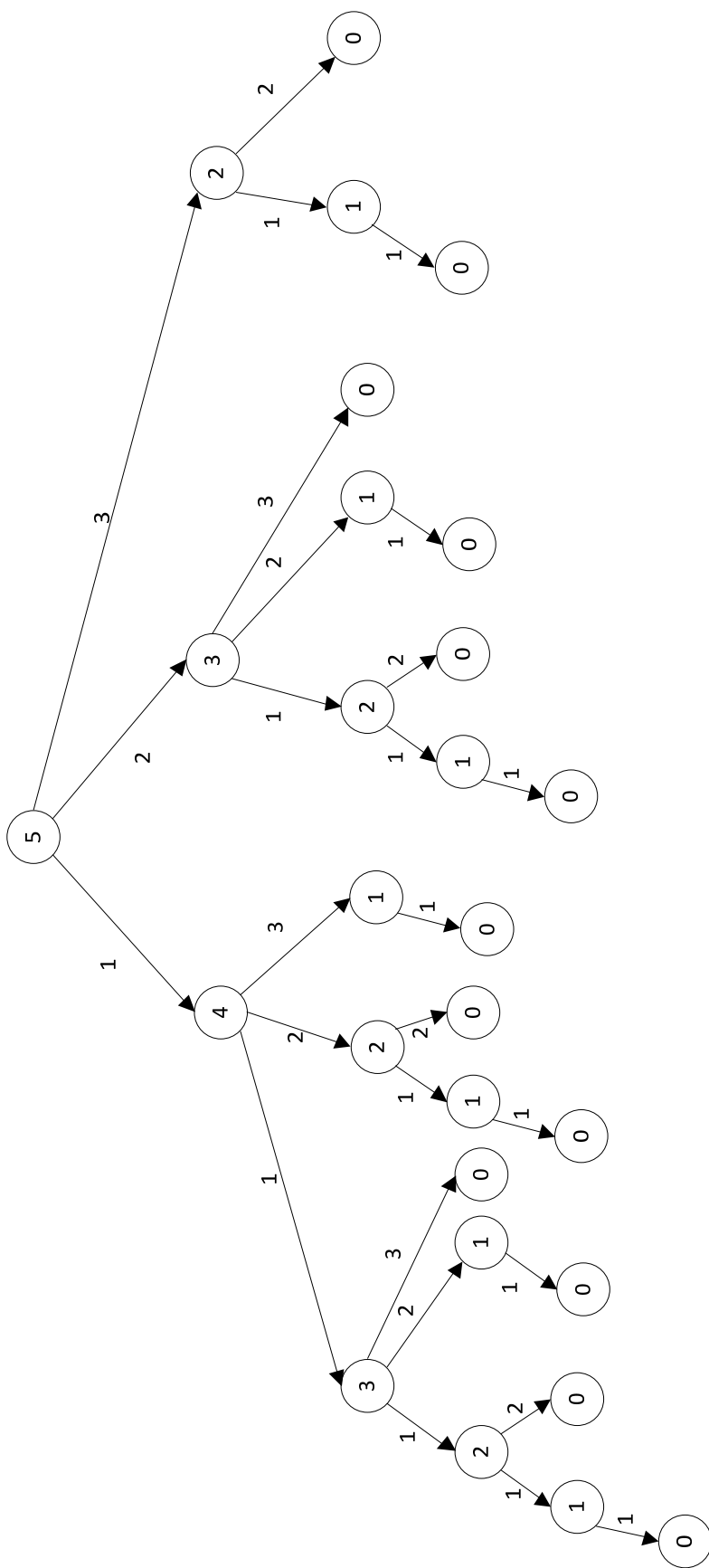


Рис. 1.1: Задача «Фишка». Случай $N = 5$, $K = 3$

До тех пор, пока метка вершины не меньше K , из каждой вершины исходят ровно K дуг, затем количество исходящих дуг уменьшается. Это значит, что решение задачи некоторой размерности komponуется из решения не более K задач меньшей размерности. Причём эти задачи не являются независимыми. На рис. 1.1 видно, что одна и та же задача повторяется несколько раз. Например, задачу размерности 3 надо решить и для решения задачи размерности 5 непосредственно, и для решения задачи размерности 4, нужной для решения задачи размерности 5. Ещё большее количество раз нужно решить задачу размерности 2. Понятно, что эффективный алгоритм не должен каждый раз решать одну и ту же задачу заново. Вместо этого надо запоминать ответы уже решённых задач и при необходимости их использовать¹.

Расчётные формулы

Обозначим через c_p решение задачи размерности p .

Заметим, что

$$c_0 = 1 \quad (1.1)$$

. Это значит, что существует единственный путь из листа в него же самого.

При $p \geq K$ получим²

$$c_p = \sum_{i=1}^K c_{p-i} \quad (1.2)$$

При $p < K$ имеет место формула

$$c_p = \sum_{i=1}^p c_{p-i} \quad (1.3)$$

Объединяя соотношения (1.2) и (1.3), получим

$$c_p = \sum_{i=1}^{\min(p,K)} c_{p-i} \quad (1.4)$$

Через $\min(p, K)$ обозначено меньшее из чисел p и K .

Пути реализации

Одним из возможных путей реализации описанных выше идей является написание рекурсивной подпрограммы. Однако, эта подпрограмма должна использовать массив для хранения уже найденных решений. При каждом вызове подпрограммы надо будет сначала проверить, нет ли уже готового решения, и только в случае его отсутствия совершать рекурсивный вызов и тут же запоминать найденное решение. Для выхода из рекурсии можно использовать соотношение (1.1)

Другой вариант решения итерационный. Используя начальное условие (1.1), в цикле по формуле (1.4) вычисляем все c_p , наращивая размерность задачи p от 1 до N с шагом 1. Ответ содержится в c_N .

2 Задача «Шеренга»

В 10–11 классах была предложена следующая задача.

Задача. Сколько вариантов имеется у учеников для того, чтобы стать в шеренгу из N человек?

¹ Таким образом, эта задача решается методом динамического программирования.

² Знак $\sum_{i=1}^m x_i$ обозначает сумму $x_1 + x_2 + \dots + x_m$

Формат входных данных

Строка входного файла содержит число N — количество учеников. ($1 \leq N \leq 150$).

Формат выходных данных

Выходной файл содержит количество вариантов расстановки.

Пример входных и выходных файлов

Task2.in	Task2.out
10	3628800

Решение

Хорошо известно, что решением этой задачи является факториал числа N , который для $N \geq 1$ определяется так:

$$N! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot N$$

Однако факториал очень быстро растёт, и нет такого стандартного целочисленного типа данных ни в Паскале, ни в C/C++, в котором можно было бы хранить $150!$ Использование вещественных типов данных для хранения результата грозит потерей значащих цифр, так как числа с плавающей точкой хранятся в памяти приближённо, с ограниченным числом разрядов в мантиссе. Поэтому придётся программировать умножение большого числа на обычное самим. Под обычным числом здесь понимается число N , уместяющееся в стандартные целочисленные типы данных языков программирования.

Большое число будем хранить в одномерном массиве. Пусть каждый элемент массива хранит одну цифру десятичной записи числа, причём младшие разряды числа находятся в конце массива.

Любое неотрицательное целое число x можно представить в виде

$$x = a \cdot 10 + b \quad (2.1)$$

где a и b — неотрицательные целые числа.

Если наложить ограничение $b < 10$, то числа a и b будут определены однозначно, причём b будет равно последней цифре в записи числа x в десятичной системе счисления, а a получается «отрыванием» от числа x последней цифры.

Рассмотрим произведение числа x на целое неотрицательное число k . Используя соотношение 2.1, получим

$$x \cdot k = (a \cdot 10 + b) \cdot k = a \cdot k \cdot 10 + b \cdot k \quad (2.2)$$

Здесь $b \cdot k$ уже не будет, вообще говоря, последней цифрой в записи произведения, т.к. может оказаться больше девяти. Снова применим 2.1.

$$b \cdot k = p \cdot 10 + b' \quad (2.3)$$

Потребуем, выполнения неравенства $b' < 10$. Это значит, что b' — последняя цифра в записи произведения $b \cdot k$. Подставим 2.3 в 2.2. Получим

$$x \cdot k = (a \cdot k + p) \cdot 10 + b' \quad (2.4)$$

Из равенства 2.4 следует, что b' — последняя цифра и в десятичной записи произведения $x \cdot k$, а не только $b \cdot k$, а p — перенос в старшие разряды. Для получения остальных цифр произведения $x \cdot k$ можно вместо $x \cdot k$ рассмотреть $a \cdot k + p$. Чтобы не выделять младший разряд числа среди

остальных только потому, что перенос в него невозможен, будем считать, что перенос в него тоже есть, но он равен нулю.

По сути мы сейчас обосновали алгоритм умножения в столбик, но при этом нигде не предполагалось, что число k — однозначное. Следовательно, оно может быть многозначным.

Итак, получаем следующий алгоритм.

Алгоритм 1. *Mult(x,k) — умножение большого целого неотрицательного числа, заданного массивом x , на обычное целое неотрицательное k*

1. $p := 0$
2. Для каждого разряда числа x , начиная с младшего,
 - (a) $r := x[i] * k + p$
 - (b) $p := r \text{ div } 10$
 - (c) $x[i] := r \text{ mod } 10$
3. Результат — в массиве x

Дальнейшие рассуждения не должны вызвать затруднений.

Алгоритм 2. *Вычисление $N!$*

1. Обнулить все разряды числа (массива) x , кроме младшего
2. Младший разряд числа x установить в 1
3. Для каждого целого i от 2 до N включительно
 - (a) Mult(x, i)

Оценим размер массива для хранения большого числа x .

$$150! < 150^{150} = (1,5 \cdot 10^2)^{150} < 10^{150} \cdot 10^{300} = 10^{450}$$

Это довольно грубая оценка. Но из неё следует, что массива из 450 элементов для хранения всех разрядов числа $150!$ гарантированно хватит, даже если хранить по одному разряду в каждом элементе массива.

3 Задача «Выкройка»

7–9 класс

Задача. На квадратном клетчатом листе бумаги $N \times N$ клеток заштрихована часть клеток. Написать программу, которая определяет прямоугольник максимальной площади, не содержащий заштрихованных клеток. Предполагается, что такой прямоугольник единственный. В качестве ответа вывести площадь прямоугольника и координаты его двух противоположных вершин (левой верхней и правой нижней)³. Для приведенного примера: площадь 12, координаты вершин (3,2) и (6,4).

³Видимо, авторы задачи имеют в виду прямоугольник, стороны которого вертикальны или горизонтальны. Далее будем придерживаться этого допущения.

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							

Рис. 3.1: Пример к задаче «Выкройка» для 7–9 классов

Формат входных данных

Первая строка файла содержит число N — количество строк и колонок таблицы ($2 \leq N \leq 10$). Следующие N строк содержат по N значений элементов таблицы, разделенных пробелами (1 — заштрихованная клетка, 0 — нет).

Формат выходных данных

Первая строка выходного файла содержит значение максимальной площади. Две следующие строки — координаты левой верхней и правой нижней клеток такого прямоугольника, разделенных пробелами.

Пример входных и выходных файлов

Task3.in	Task3.out
7	12
0 1 0 0 1 1 1	3 2
0 1 1 0 0 0 0	6 4
1 0 0 0 0 1 0	
0 0 0 0 1 0 1	
1 0 0 0 0 1 0	
0 0 0 0 0 1 0	
0 1 0 0 0 0 0	

Решение

Рассмотрим пример матрицы.

$$\begin{array}{cccccc}
 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 1 & 1 \\
 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 1
 \end{array} \tag{3.1}$$

Пусть нам надо найти прямоугольник максимальной площади, состоящий из нулей и имеющий левым верхним углом элемент, стоящий на первой строке в первом столбце.

Случай 1

Для этого рассмотрим криволинейную трапецию⁴, основания которой есть первая строка матрицы и строка, расположенная выше первой единицы первого столбца (т.е. четвёртая строка). Левая боковая сторона этой трапеции — это первый столбец, а правая боковая сторона является кривой (или ломаной), проведённой через правые нули линеек нулей в строках, заключённых между основаниями (т.е. в строках номер 1, 2, 3 и 4 матрицы (3.1)). Чтобы выделить максимально большой прямоугольник из этой фигуры, найдём внутри неё самую короткую линейку нулей. Это линейка во второй строке длиной 3 нуля. Так получаем горизонтальную сторону прямоугольника. А вертикальная сторона — это левое основание фигуры. В нашем случае имеет размер 4. Итого площадь прямоугольника равна 12.

Казалось бы, для решения исходной задачи достаточно повторить приведённые выше рассуждения для каждого элемента матрицы, рассматривая его в качестве верхней левой вершины, а потом выбрать оптимальный ответ. Но это может привести к ошибке.

Случай 2

Рассмотрим ещё один пример.

$$\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{array} \quad (3.2)$$

Описанный выше подход дал бы ответ 9, тогда как правильный ответ — 10. Просто теперь надо рассматривать другую криволинейную трапецию⁵, повернутую на 90 градусов по часовой стрелке. Основаниями будут столбцы матрицы — самый первый столбец и первый столбец, расположенный левее первой единицы в первой строке (т.е. пятый столбец матрицы (3.2)). Одной боковой стороной будет первая строка матрицы, а другой — кривая, проведённая через нижние нули в столбиках нулей, заключённых между основаниями. Самые короткие столбики нулей матрицы (3.2) расположен в её четвёртом и пятом столбцах. Их длина 2. Это длина одной стороны прямоугольника. А длина другой стороны — это расстояние между основаниями. В нашем случае 5. Получается площадь прямоугольника 10.

Таким образом, для решения задачи достаточно перебрать все элементы матрицы, рассматривая их как верхний левый угол двух прямоугольников, рассмотренных в случаях 1 и 2. Из этих прямоугольников выберем прямоугольник наибольшей площади.

Замечания о реализации

Чтобы многократно не пересчитывать длины линеек нулей, рассмотренных в случае 1, их можно вычислить заранее и сохранить в массиве. Для этого каждую строку исходного массива достаточно просмотреть только один раз. То же самое можно сделать со столбиками нулей из случая 2. Если при этом минимальную длину линейки (столбика) нулей заново вычислять для каждого левого верхнего угла прямоугольника, то время работы алгоритма в худшем случае будет порядка N^3 . При заданных в задаче ограничениях это вполне допустимо.

4 Ещё одна «Выкройка»

10–11 класс

⁴Название не совсем корректно. В матрице (3.1) эта фигура выделена красным цветом

⁵В (3.2) выделена красным цветом

Задача. На квадратном клетчатом листе бумаги $N \times N$ клеток нарисованы фигуры, каждая из которых состоит только из целых заштрихованных соприкасающихся клеток. Фигуры не соприкасаются и не пересекаются. Написать программу, которая определяет фигуру максимальной площади. Предполагается, что такая фигура единственная. В качестве ответа вывести площадь фигуры. Для приведенного примера: площадь 7.

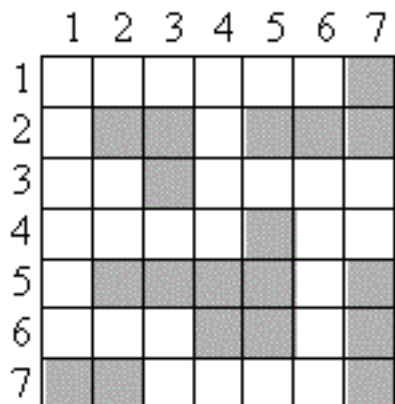


Рис. 4.1: Пример к задаче «Выкройка» для 10–11 класса

Формат входных данных

Первая строка файла содержит число N — количество строк и колонок таблицы ($2 \leq N \leq 10$). Следующие N строк содержат по N значений элементов таблицы, разделенных пробелами (1 — заштрихованная клетка, 0 — нет).

Формат выходных данных

Строка выходного файла содержит значение максимальной площади.

Пример входных и выходных файлов

Task3.in	Task3.out
<pre> 7 0 0 0 0 0 0 1 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 </pre>	<pre> 7 </pre>

Решение

Эту задачу можно решить рекурсивно. Пусть A — матрица (таблица), о которой идёт речь в условии задачи.

```
Function Calc(i,j : ShortInt) : ShortInt;
```



```

Begin
  If a[i,j]=0 Then
    Calc:=0
  Else Begin
    a[i,j]:=0;
    Calc:=1+Calc(i-1,j)+Calc(i+1,j)+Calc(i,j-1)+Calc(i,j+1);
  End
End;

```

Аргументами этой функции являются номер строки и номер столбца глобальной матрицы A . Если при вызове функции $A_{ij} = 0$, то площадь фигуры, которой принадлежит ячейка A_{ij} , равна нулю, иначе эта площадь равна единице плюс площади фигур, которым принадлежат примыкающие к A_{ij} ячейки. Из условия задачи не совсем ясно, считаются ли частью фигуры клетки, в которые из данной можно попасть только по диагонали. В приведённом решении предполагается, что не считаются. Однако, при необходимости функцию Calc легко дополнить ещё четырьмя рекурсивными вызовами.

Для решения задачи функцию Calc надо вызывать последовательно для каждой ячейки матрицы и выбрать максимальный результат. Однако, для ячеек, находящихся на границе матрицы, при вызове функции произойдёт выход за пределы массива. Например, при вызове из первой строки матрицы произойдёт обращение к нулевой строке. Чтобы не усложнять функцию Calc лишними проверками, можно просто окружить матрицу A «забором» из нулей, увеличив количество её строк, а также количество столбцов, на два.

Перед рекурсивными вызовами в функции Calc выполняется присваивание $a[i,j]:=0$. Это делается для того, чтобы алгоритм не зациклился. Ведь, например, вызов Calc для какой-нибудь ячейки с единицей приведёт к рекурсивному вызову для соседней с ней ячейки сверху. А этот, второй вызов, может привести к вызову для ячейки снизу, то есть для исходной ячейки. И если в обеих ячейках единицы, то алгоритм зацикливается. Надо ли после рекурсивных вызовов восстанавливать содержимое A_{ij} ? Это сделать можно, но нецелесообразно, потому что может значительно увеличить время работы программы. Ведь подсчитав площадь какой-то фигуры, мы затираем её нулями. И если потом из основной программы произойдёт вызов Calc для соседних ячеек, принадлежавших той же фигуре, то повторных вычислений той же площади удастся избежать.

5 Задача «Числа»

10–11 класс

Задача. Дано не более чем семизначное целое положительное число A . После удаления первой цифры это число уменьшилось в N раз. Вывести количество таких чисел.

Формат входных данных

Файл содержит число N . ($10 \leq A \leq 9999999$).

Формат выходных данных

Выходной файл содержит ответ на вопрос задачи⁶.

⁶В оригинале было: «Выходной файл содержит в отдельной строке такие числа». Это, конечно же, ошибка.

Пример входных и выходных файлов

Task1.in	Task1.out
5	16

Решение

Пусть мы умеем у любого натурального числа A удалять первую цифру и получать из него число k . Тогда перебираем все A от 10 до 9999999 включительно, находим соответствующие k и, если $A = k \cdot N$, то перед нами одно из искомым чисел. Значит, увеличиваем значение счётчика. Заметим, что проверять условие $\frac{A}{k} = N$ хуже, так как k может оказаться равным нулю.

Как удалить первую цифру числа A ? Пусть запись натурального числа A в десятичной системе счисления содержит c цифр ($c > 1$). Тогда для удаления первой цифры достаточно найти остаток от деления числа A на 10^{c-1} . Например, это можно реализовать так:

```
Function NewNumber(x : Integer) : Integer;  
Var p,c : Integer;  
Begin  
  p:=1;  
  c:=x;  
  While x>0 Do  
    Begin  
      x:=x div 10;  
      p:=p*10  
    End;  
    p:=p div 10;  
    NewNumber := c mod p  
  End;
```

Реализацию алгоритмов решения рассмотренных задач на FreePascal 2.4.4 смотрите в прилагаемых файлах. Для кодировки русских букв (в комментариях к программам) использована 866 кодовая страница.

Серeda А.Н.