№ 13 (206) апрель 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИКА

Содержание

Всероссийская олимпиада школьников по информатике

В.М. КИРЮХИН

Рекомендации методической комиссии по проведению областных, территориальных и республиканских туров в 1999 году

Общие замечания

1. Традиционной формой организации всероссийских олимпиад по информатике является проведение двух компьютерных туров, длительность каждого из которых не более 5 часов. Методическая комиссия рекомендует придерживаться этого регламента, хотя он может быть изменен с учетом возможностей конкретного региона и существующих традиций при проведении олимпиад по информатике.
2. Количество предлагаемых задач на каждом туре может быть разным. На последних российских и международных олимпиадах на каждом туре предлагалось три задачи. Методическая комиссия считает, что этот вопрос может решаться организаторами олимпиад исходя из среднего уровня подготовки участников олимпиады в конкретном регионе, имеющегося в распоряжении местного жюри пакета своих задач и методики их проверки.
3. Результатом решения каждой из представленных на олимпиаде задач должна быть отлаженная программа в исполняемом виде. С целью достижения объективности в оценке полученных участниками решений рекомендуется при проверке программ использовать в основном тесты. Тесты должны в максимальной степени способствовать выявлению особенностей алгоритма решения задачи и давать возможность жюри дифференцировать полученные участниками решения по степени их правильности и эффективности. В условиях задач должны быть указаны рекомендуемые ограничения по времени тестирования для отдельного набора данных.
4. При подсчете результатов решения задач на каждом туре методическая комиссия рекомендует придерживаться 100-балльной системы. В случае нескольких задач на туре это количество баллов должно быть распределено между всеми задачами с учетом их сложности. Сложность каждой задачи определяется самими членами жюри исходя из конкретного состава участников олимпиады.
5. Для проведения областных, территориальных и республиканских олимпиад по информатике могут использоваться как задачи, представленные ниже, так и задачи, разработанные самими организаторами или членами жюри местных олимпиад. Основными критериями отбора олимпиадных задач должны быть их новизна и доступная для участников сложность их решения.
6. До начала проведения туров жюри олимпиады должно иметь в своем распоряжении отлаженный вариант программы для каждой предлагаемой задачи, набор тестов и результаты тестирования. С целью более тщательной подготовки к олимпиаде методическая комиссия специально минимизировала объем информации о каждой предлагаемой ей задаче, чтобы дать возможность организаторам олимпиад самим разобраться в сути каждой задачи. Более того, при решении олимпиадных задач по информатике достаточно часто бывает трудно представить однозначное решение, а тем более эталонное.

Задачи, предлагаемые методической комиссией для проведения областных, территориальных и республиканских туров

Задача 1. "Телефонный номер"

Клавиатура многих телефонов выглядит так, как показано на рисунке*.

Использование изображенных на клавишах букв позволяет представить номер телефона в виде легко запоминающихся слов, что часто бывает более удобным, чем традиционная запись номера телефона в виде последовательности цифр. Многие фирмы пользуются этим и стараются подобрать себе номер телефона так, чтобы он содержал как можно больше букв из названия фирмы.

Требуется написать программу, которая преобразует исходный цифровой номер телефона в соответствующую последовательность букв и цифр, содержащую как можно больше символов из названия фирмы. При этом буквы из названия фирмы должны быть указаны в полученном номере в той же последовательности, в

* Соответствие букв и цифр на вашем телефоне может несколько отличаться от приведенного.

которой они встречаются в названии фирмы. Например, если фирма называется IBM, а исходный номер телефона — 246, то замена его на BIM недопустима, тогда как замена его на 2IM или B4M является правильной.

Технические требования:
Входной файл: INPUT.TXT
Выходной файл: OUTPUT.TXT
Ограничение времени: 10 секунд

Формат входных данных:
Первая строка входного файла содержит название фирмы. Она состоит только из заглавных букв латинского алфавита, количество которых не превышает 80. Вторая строка содержит номер телефона в виде последовательности цифр. Цифр в номере телефона также не более 80.

Формат выходных данных:
В единственной строке выходного файла должен содержаться измененный номер телефона.

Пример файла входных данных:
```
IBM 246
```

Пример файла выходных данных (для приведенного выше входного файла):
```
2IM
```

Задача 2. "Разбиение на такты"

С точки зрения программиста, нотная запись состоит из последовательности нот, каждой из которых соответствует один звук музыкальной темы. Нота имеет высоту и длительность. Возможны следующие длительности нот: 1 — базовая длительность (зависит от темпа мелодии); 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 — являются соответствующими долями базовой длительности.

В нотной записи принято разделять последовательность нот на такты. Продолжительность такта определяется его размером. Например, размер 4/4 означает, что такт содержит четыре ноты длительности 1/4 или одну ноту длительности 1. Допустимы следующие размеры такта: 2/4, 3/4, 4/4, 5/4, 3/8, 6/8. Существуют и другие (довольно экзотические) размеры, но в этой задаче они не используются.

Требуется написать программу, которая по размеру такта и последовательности нот выводит эту последовательность, разбитую на такты, в соответствии с указанным размером. Если нота не умещается в такте, остаток размещается в следующем такте. В том случае, когда при разбиении получается нота нестандартной длительности, нужно разбить и ее на минимальное количество нот стандартной длительности в порядке возрастания длительности.

Длительность всей последовательности нот может быть не кратной размеру такта. В таком случае первый такт получается короче остальных (т.н. затакт), а последний обязан быть заполнен полностью в соответствии с размером такта.

Технические требования:
Входной файл: INPUT.TXT
Выходной файл: OUTPUT.TXT
Ограничение времени: 10 секунд

Формат входных данных:
Входной файл INPUT.TXT содержит последовательность строк. В первой строке — размер такта, для задания которого используется литерал, обозначающий размер: "2/4", "3/4" и т.д. Во второй и последующих строках — последовательность литералов, обозначающих ноты ("1/2", "1/4" и т.д.), разделенных пробелами и/или символами перевода строки. Длина исходной последовательности не превышает 1000 нот.

Формат выходных данных:
Выходные данные помещаются в файл OUTPUT.TXT, в каждой строке которого должна содержаться последовательность литералов, обозначающих ноты, соответствующие одному такту (по строке на такт).

Пример файла входных данных:
```
4/4
1/2 1/2 1/2 1/4
```

Пример файла выходных данных (для приведенного выше входного файла):
```
1/2 1/4
1/4 1/2  1/4
```

Задача 3. "Стопами Дирихле"

У главного животновода кролиководческого хозяйства "Стопами Дирихле" неожиданно возникла проблема. После долгих переговоров с западными партнерами наконец-то поступила партия из $2n$ кроликов k пород. Кроликов нужно рассадить в n имеющихся в наличии двухместных клеток, причем это нужно сделать так, чтобы в результате получилось как можно больше однопородных брачных пар (разнополых кроликов, сидящих в одной клетке), а разнопородных брачных пар не было вовсе.

Требуется написать программу, которая определяет возможность требуемого размещения кроликов в клетках и, в случае положительного ответа, количество образованных брачных пар.

Технические требования:
Входной файл: INPUT.TXT
Выходной файл: OUTPUT.TXT
Ограничение времени: 10 секунд

Формат входных данных:
Во входном файле INPUT.TXT содержится последовательность строк вида:

```
f  r
```

или

```
m  r
```

Символы `f` и `m` определяют пол кролика (`f` — самка, `m` — самец), `r` — целое число от 1 до 10, обозначающее номер породы, к которой принадлежит кролик. Количество кроликов не превышает 100 000.

Формат выходных данных:
Написанная вами программа должна поместить в выходной файл OUTPUT.TXT либо строку вида:

```
b
```

в случае, если кроликов можно разместить в соответствии с заданными требованиями (`b` — количество образованных брачных пар), либо строку вида:

```
No
```

в противном случае.

Пример файла входных данных:

```
f 2
m 3
m 2
f 3
```

Пример файла выходных данных (для приведенного выше входного файла):

```
2
```

Задача 4. "Умножение дроби"

Задана некоторая правильная периодическая дробь Q и натуральное число N. Q и N таковы, что количество цифр, используемых для их описания, не превосходит 100. При изображении дроби Q периодическая часть заключается в круглые скобки.

Требуется написать программу, которая определяет результат умножения Q на N, то есть непериодическую часть и минимальный период числа $Q \cdot N$. В случае получения результата умножения в виде конечной дроби скобки опускаются.

Технические требования:
Входной файл: INPUT.TXT
Выходной файл: OUTPUT.TXT

Формат входных данных:
Входной файл INPUT.TXT содержит две строки. В первой строке задается дробь Q, а во второй — число N.

Формат выходных данных:
В единственной строке выходного файла OUTPUT.TXT должен содержаться результат умножения Q на N.

Пример файла входных данных:

```
0.1(6)
3
```

Пример файла выходных данных (для приведенного выше входного файла):

```
0.5
```

Задача 5. "Арифметическая прогрессия"

Задана последовательность натуральных чисел из диапазона [1, 2147483647]. Количество чисел в этой последовательности не превышает 100 000. Необходимо определить, можно ли выстроить эти числа в отрезок арифметической прогрессии. При необходимости порядок чисел в последовательности можно изменять.

Требуется написать программу для решения вышеназванной задачи.

Технические требования:
Входной файл: INPUT.TXT
Выходной файл: OUTPUT.TXT
Ограничение времени: 15 секунд

Формат входных данных:
Входной файл INPUT.TXT содержит заданную последовательность натуральных чисел. Числа в файле разделены пробелами или символами перехода на новую строку.

Формат выходных данных:
Выходной файл OUTPUT.TXT должен содержать либо строку

```
Yes
```

в случае положительного ответа, либо строку

```
No
```
в противном случае.

Пример файла входных данных:
```
80 50 10 30 70 40 20 60 90
```

Пример файла выходных данных (для приведенного выше входного файла):
```
Yes
```

Задача 6. "Хамелеон"

Игра "Хамелеон" заключается в следующем. Задан квадрат размером 3 × 3, в клетках которого находятся 8 фишек с буквами этого слова, а одна из клеток пустая. За один ход разрешается одну из фишек переместить на соседнюю пустую клетку. Цель игры — достигнуть расположения фишек, указанного на *рис.* 1.

```
X   A   M
E   Л   E
O   H
```

Рис. 1

Требуется написать программу, которая определяет план достижения цели за минимально возможное число ходов либо сообщает, что цель достичь нельзя.

Технические требования:
Входной файл: INPUT.TXT
Выходной файл: OUTPUT.TXT
Ограничение времени: 30 секунд

Формат входных данных:
Во входном файле находится матрица 3 × 3, составленная из больших букв русского алфавита и одного пробела. Каждая строка матрицы располагается в соответствующей строке в файле.

Формат выходных данных:
Ваша программа должна вывести в первую строку выходного файла искомое число ходов, а в последующие — их список. Каждый ход задается координатами той фишки, которая перемещается. Если искомого плана не существует, выведите в выходной файл сообщение `Нет решения` в виде единственной строки.

Пример файла входных данных:
```
X A M
E E
O Л H
```

Пример файла выходных данных:
```
2
3 2
3 3
```

Задача 7. "Максимальный результат"

Задано алгебраическое выражение, составленное из неотрицательных вещественных чисел и знаков операций +, — и *. Необходимо расставить в этом выражении скобки так, чтобы его значение стало максимально возможным.

Требуется написать программу для решения поставленной задачи.

Технические требования:
Входной файл: INPUT.TXT
Выходной файл: OUTPUT.TXT
Ограничение времени: 10 секунд

Формат входных данных:
В первой строке входного файла INPUT.TXT записано исходное выражение длиной не более 250 символов. Внутри чисел пробелы не допускаются. Выражение содержит не более 50 чисел, каждое из которых лежит в диапазоне от 0 до 10^6.

Формат выходных данных:
Выходной файл OUTPUT.TXT должен содержать две строки. В первой строке должно находиться максимально возможное после расстановки скобок значение полученного выражения, а во второй строке — само это выражение. Если вариантов решения задачи несколько, нужно выдать любой из них.

Пример файла входных данных:
```
1 + 2 - 3.0*4
```

Пример файла выходных данных:
```
0
((1 + 2) - 3)*4
```

Комментарии к задачам для членов жюри

Задача 1. Это задача на метод динамического программирования. Будем последовательно (для k, пробегающего значения от 0 до длины названия фирмы) строить массив. В этом массиве в ячейке i будет записан телефонный номер из букв и цифр, совпадающий в цифровом выражении с k первыми цифрами номера фирмы, причем последняя буква, задействованная в номе-

ре, имеет позицию не правее i-й. Такой массив легко пересчитывать, увеличивая k на единицу. Когда k станет равным длине номера фирмы, в ячейке с номером k окажется ответ.

Задача 2. Эта задача требует аккуратного программирования и **внимательного** чтения условия. Никаких сложных алгоритмических проблем не содержит, однако не очень проста в реализации.

Задача 3. Суть решения задачи заключается в следующем. Посчитаем количество кроликов обоих полов (по отдельности) для каждой породы. Далее необходимо использовать только эти числа, а не работать с каждым кроликом индивидуально.

Задача 4. Задача имеет два решения. Например, можно перевести дробь в обыкновенную, потом числитель умножить на N, а затем снова делить, выделяя период. Но в длинной арифметике большинству школьников это технически сделать сложно. Проще разработать алгоритм умножения периода на число, а потом, если потребуется, период сократить, при этом еще нужно помнить, что $0,(9) = 1$, иначе период не будет минимальным.

Можно использовать следующее правило умножения периода правильной дроби на произвольное натуральное число:

1) если при умножении периода дроби на некоторое число количество цифр в произведении равно количеству цифр в периоде исходного числа, то период результата равен полученному произведению и переходим к (6), в противном случае — к (2);

2) "лишними" цифрами будем считать $n - k$ первых слева цифр результата, где n — количество цифр в результате, k — количество цифр в периоде исходной дроби;

3) сложим число, образованное "лишними" цифрами, с числом, образованным правыми k цифрами промежуточного результата;

4) если количество цифр в получившемся результате сложения больше, чем k, то процесс следует повторить с (2);

5) если количество цифр результата сложения стало равным количеству цифр периода исходной дроби, то период произведения равен последнему результату суммирования;

6) непериодическая (целая — для чисто периодических дробей) часть результата равна сумме чисел, образованных из "лишних" цифр каждого этапа. Если же у исходной дроби изначально была своя непериодическая часть, то умножить также следует и ее, а затем сложить с результатом умножения периода.

Рассмотрим два примера. Вначале выполним умножение $0,(7) \cdot 16$. Для этого умножим период дроби на 16: $7 \cdot 16 = 112$. В периоде дроби одна цифра, а в произведении — три цифры. Следовательно, "лишними" являются две левые цифры, образующие число 11. Прибавив 11 к 2, получим 13, что опять превышает исходную длину периода на одну цифру. Вновь сложим "лишнее" число 1 с правой цифрой результата, $1 + 3 = 4$. Количество цифр результата равно количеству цифр в периоде исходной дроби, следовательно, период произведения равен 4. Целая же часть результата умножения равна $11 + 1 = 12$ (в процессе ее формирования участвуют только "лишние" цифры). Таким образом, получается ответ: $0,(7) \cdot 16 = 12,(4)$.

В качестве второго примера выполним умножение $0,7(6) \cdot 12$. Для этого сначала умножим на 12 периодическую часть. $6 \cdot 12 = 72$. В получившемся произведении две цифры, а в исходном периоде — одна. Следовательно, "лишнее" число равно 7, результат сложения равен $2 + 7 = 9$. В получившемся результате количество цифр равно количеству цифр в периоде исходной дроби, т.е. период произведения равен 9. Непериодическая часть произведения равна единственному "лишнему" числу — 7. Таким образом, $0,0(6) \cdot 12 = 0,7(9) = 0,8$. Затем умножим на 12 непериодическую часть исходной дроби $0,7 \cdot 12 = 8,4$. В результате получаем: $0,7(6) \cdot 12 = 8,4 + 0,8 = 9,2$.

Задача 5. Один из алгоритмов решения этой задачи заключается в следующем. Найдем самое большое и самое маленькое числа в последовательности. Поделим разность между ними на количество чисел. Это и будет разностью прогрессии, которая может быть построена из этих чисел. Если разность прогрессии равна 0, то есть максимальный и минимальный элементы равны, то тогда такую прогрессию построить можно. В противном случае необходимо проверить, встречаются ли все элементы арифметической прогрессии ровно по одному разу в исходной последовательности. Для этого достаточно завести битовый массив в памяти и отмечать в нем номера встретившихся элементов.

Задачи 6 и 7. Эти задачи являются новыми вариациями известных задач. Тем не менее они могут выступать в качестве олимпиадных задач на региональном уровне, так как требуют для своего решения комплексного использования интересных в техническом отношении методов.

Методическая комиссия выражает надежду, что представленные материалы окажутся полезными при проведении областных, территориальных и республиканских олимпиад по информатике, и желает успехов организаторам в их проведении. В случае необходимости дополнительную информацию по представленным задачам можно получить по электронной почте, послав запрос по адресу: **VKiryukhin@elco.ru**

Задачи X международной олимпиады по информатике*

Решения подготовлены Е.В. АНДРЕЕВОЙ

Лампы для праздника

Для освещения заключительного вечера IOI'98 имеется N цветных ламп, пронумерованных от 1 до N. Четыре кнопки позволяют управлять лампами следующим образом:

кнопка 1 — изменяет состояние всех ламп: те, что были включены, становятся выключенными, те, что были выключены, — включаются,

кнопка 2 — изменяет состояние всех ламп, имеющих нечетные номера,

кнопка 3 — изменяет состояние всех ламп, имеющих четные номера,

кнопка 4 — изменяет состояние всех ламп, имеющих номера, вычисляемые по формуле $3k+1$ (где $k \geq 0$), т.е. 1, 4, 7, ...

Имеется счетчик C, который учитывает (хранит) суммарное число нажатий всех кнопок.

В начале вечера все лампы были включены, а счетчик C был установлен в нуль.

Задача. **Заданы значение счетчика C и информация о конечном состоянии некоторых ламп. Напишите программу для определения всех различных возможных конечных (окончательных) конфигураций N ламп, чтобы каждая конфигурация соответствовала заданной информации.**

Исходные данные

Файл, имеющий имя PARTY.IN, содержит четыре строки, задающих количество ламп N, конечное значение счетчика C нажатых клавиш и состояние некоторых ламп в окончательной конфигурации.

В первой строке содержится число N, во второй строке — конечное значение счетчика C. Третья строка содержит список номеров ламп, о которых известно, что в конечной конфигурации они включены. Номера ламп в строке отделены друг от друга одним пробелом, и список заканчивается числом -1. Четвертая строка содержит список номеров ламп, о которых известно, что в окончательной конфигурации они выключены. Номера ламп в строке отделены друг от друга одним пробелом, и список заканчивается числом -1.

Пример ввода:

```
10
1
-1
7 -1
```

Здесь число ламп — 10, и только одна кнопка была нажата. Лампа 7 — выключена в конечной конфигурации.

Выходные данные

Файл PARTY.OUT должен содержать все возможные различные окончательные конфигурации (без повторений) всех ламп. Каждая возможная конфигурация должна быть записана на отдельной строке. Конфигурации могут быть перечислены в произвольном порядке.

Каждая строка содержит N символов, где первый символ представляет состояние лампы номер 1, а последний символ представляет состояние лампы номер N. 0 (ноль) означает, что лампа выключена, а 1 (единица) означает, что лампа включена.

Пример вывода:

```
0000000000
0110110110
0101010101
```

Здесь видно, что имеются три возможные различные конечные конфигурации:

все лампы выключены;

или

лампы 1, 4, 7, 10 выключены, а лампы 2, 3, 5, 6, 8, 9 включены;

или

лампы 1, 3, 5, 7, 9 выключены, а лампы 2, 4, 6, 8, 10 включены.

Ограничения

Параметры N и C ограничены:

$10 \leq N \leq 100$

$1 \leq C \leq 10000$

Количество ламп, о которых известно, что в конечной конфигурации они включены, меньше или равно 2.

Количество ламп, о которых известно, что в конечной конфигурации они выключены, меньше или равно 2.

Гарантируется, что существует хотя бы одна конечная конфигурация.

* Решения задач X международной олимпиады были опубликованы также в № 11, 12/99.

Решение

Суммарное количество нажатий различных кнопок C настолько велико, что сразу же становится очевидной невозможность перебора всех различных вариантов нажатия кнопок. Однако легко показать, что этого делать и не нужно. Заметим, что если любая из кнопок была нажата дважды, то состояние всех ламп не изменится. Поэтому следует учитывать лишь кнопки, которые были нажаты нечетное количество раз. Общее число ламп также не влияет на сложность решения задачи, так как на самом деле все лампы можно разделить на четыре непересекающиеся группы: лампы с нечетными номерами, не попадающими во множество ламп с номерами $\{3 \cdot K+1,\ K \geq 0\}$, аналогичные лампы с четными номерами, лампы с нечетными номерами из множества $\{3 \cdot K+1,\ K \geq 0\}$ и лампы с четными номерами из этого же множества. Все лампы каждой из перечисленных групп одинаково реагируют на нажатие любых кнопок. А именно: лампы из первой группы меняют свое состояние лишь при нажатии кнопок 1 и 2, второй группы — кнопок 1 и 3, третьей группы — 1, 2 и 4, четвертой группы — 1, 3 и 4. Причем в конечном состоянии лампочка окажется выключенной, если сумма нажатий влияющих на нее кнопок окажется нечетной, и включенной в противном случае.

Опираясь на сказанное выше, будем решать задачу следующим образом. Переберем все возможные принципиальные варианты количества нажатий каждой из кнопок. Так как для каждой кнопки в отдельности таких вариантов всего 2: четное число нажатий, которое можно обозначить 0, и нечетное — 1, то общее количество вариантов равно $2^4=16$. Для каждой из 16 комбинаций нажатий кнопок определим, какие группы лампочек в этом случае окажутся включенными, а какие — выключенными. Полученную информацию сравним с входными данными, и если противоречия нет, то данные состояния лампочек можно печатать. При этом следует учитывать, что общее количество нажатий кнопок не должно превышать C и что полученное решение не должно совпадать с найденным ранее. Для реализации последней подзадачи все найденные уникальные решения, общее количество которых не может превосходить 16, будем запоминать, а каждое новое найденное решение будем сравнивать с уже запомненными и лишь в случае его уникальности также запоминать и печатать.

Реализацию описанного алгоритма на Паскале легко произвести с помощью такого типа данных, как множество, состоящее не более чем из четырех элементов, каждый из которых описывает одну группу лампочек. Так, например, если для одного из вариантов нажатия кнопок мы определили множество включенных ламп `s`, а из входных данных задачи известны множества действительно включенных ламп — `on` — и выключенных — `off`, то данный вариант нажатий допустим, если `(on<=s)and(off<=[1..4]-s)`. Здесь множество `[1..4]-s` обозначает выключенные при данной комбинации нажатий лампы. В случае использования для проверки подобных условий массивов решение задачи становится значительно более громоздким.

Ниже приведено полное решение задачи “Лампы для праздника”.

```
type t=set of 1..4;
var
   c,i1,i2,i3,i4,m,n,j,i,k:integer;
  on,off,s:t;
  contr:array[1..16] of t;
  flag:boolean;
procedure ReadData;
begin
  assign(input,'party.in');
  reset(input);
  readln(n,c);
  on:=[];
  read(j);
  while j<>-1 do{формируем множество включенных ламп}
    begin{определяем группу для номера j}
      if j mod 3<>1 then
          if odd(j) then on:=on+[1]
                    else on:=on+[2]
      else
          if odd(j) then on:=on+[3]
                    else on:=on+[4];
      read(j)
    end;
```

```
  off:=[];
  read(j);
  while j<>-1 do{формируем множество выключенных ламп}
    begin{определяем группу для номера j}
      if j mod 3<> 1 then
           if odd(j) then off:=off+[1]
                     else off:=off+[2]
      else
           if odd(j) then off:=off+[3]
                     else off:=off+[4];
      read(j)
    end
end;
procedure Solve;
begin
   assign(output,'party.out');
   rewrite(output);
   i:=0;{количество найденных решений}
   {перебор вариантов нажатий кнопок}
   for i1:=0 to 1 do
   for i2:=0 to 1 do
   for i3:=0 to 1 do
   for i4:=0 to 1 do
     begin
        k:=i1+i2+i3+i4;{минимальное число нажатий в данном варианте}
        if c>=k then
          begin
            s:=[1..4];{множество включенных ламп}
            for m:=1 to 4 do{удалим выключенные лампы}
              case m of{m определяет одну из 4 групп ламп}
                1:if odd(i1+i2)    then s:=s-[1];
                2:if odd(i1+i3)    then s:=s-[2];
                3:if odd(i1+i2+i4) then s:=s-[3];
                4:if odd(i1+i3+i4) then s:=s-[4];
              end;
            if (on<=s)and(off<=[1..4]-s) then{вариант подходит}
              begin
                flag:=true;
                for j:=1 to i do{проверяем решение на уникальность}
                  if s=contr[j] then flag:=false;
                if flag then
                  begin
                    inc(i); contr[i]:=s;{запоминаем решение}
                    for j:=1 to n do{печатаем решение}
                      if j mod 3<> 1 then{определяем группу для номера j}
                        if odd(j) then
                             if 1 in s then write(1) else write(0)
                        else if 2 in s then write(1) else write(0)
                      else
                        if odd(j) then
                             if 3 in s then write(1) else write(0)
                        else if 4 in s then write(1) else write(0);
                    writeln
                  end
              end
          end
     end
end;
begin
  ReadData;
  Solve
end.
```

Полигон

Существует игра для одного игрока, которая начинается с задания Полигона с N вершинами. Пример графического представления Полигона показан на *рис.* 1, где $N=4$. Для каждой вершины Полигона задается значение — целое число, а для каждого ребра — метка операции + (сложение) либо * (умножение). Ребра Полигона пронумерованы от 1 до N.

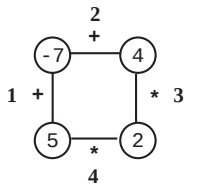


Рис. 1. Графическое представление Полигона

Первым ходом в игре удаляется одно из ребер.

Каждый последующий ход состоит из следующих шагов:

- выбирается ребро E и две вершины V_1 и V_2, которые соединены ребром E;
- ребро E и вершины V_1 и V_2 заменяются новой вершиной со значением, равным результату выполнения операции, определенной меткой ребра E, над значениями вершин V_1 и V_2.

Игра заканчивается, когда больше нет ни одного ребра. **Результат игры** — это число, равное значению оставшейся вершины.

Пример игры:

Рассмотрим Полигон на *рис.* 1. Игрок начал игру с удаления ребра 3 (см. *рис.* 2).

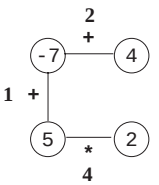


Рис. 2. Удаление ребра 3

После этого игрок выбирает ребро 1 (см. *рис.* 3)

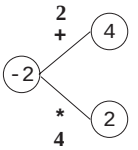


Рис. 3. Результат выбора ребра 1

затем — ребро 4 (см. *рис.* 4)

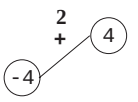


Рис. 4. Результат выбора ребра 4

и, наконец, ребро 2. Результатом игры будет число 0 (см. *рис.* 5).

Рис. 5. Результат выбора ребра 2

Задача. Напишите программу, которая по заданному Полигону вычисляет максимальное значение оставшейся вершины и выводит список всех тех ребер, удаление которых на первом ходе игры позволяет получить это значение.

Входные данные

Файл POLYGON.IN описывает Полигон с N вершинами. Файл содержит две строки. В первой строке записано число N. Вторая строка содержит метки ребер с номерами 1, ..., N, между которыми записаны через один пробел значения вершин (первое из них соответствует вершине, смежной ребрам 1 и 2, следующее — смежной ребрам 2 и 3, и так далее, последнее — смежной ребрам N и 1). Метка ребра — это буква t, соответствующая операции +, или буква x, соответствующая операции *.

Пример входного файла:

```
4
t -7 t 4 x 2 x 5
```

Данный входной файл описывает Полигон, изображенный на *рис.* 1.

Вторая строка начинается с метки ребра 1.

Выходные данные

В первой строке выходного файла POLYGON.OUT программа должна записать максимальное значение оставшейся вершины, которое может быть достигнуто для заданного Полигона. Во второй строке должен быть записан список всех тех ребер, удаление которых на первом ходе позволяет получить это значение. Номера ребер должны быть записаны в возрастающем порядке и отделяться друг от друга одним пробелом.

Пример выходного файла:

```
33
1 2
```

Таким должен быть выходной файл для Полигона, изображенного на *рис.* 1.

Ограничения

$3 \le N \le 50$

Для любой последовательности ходов значения вершин находятся в пределах $[-32768, 32767]$.

Решение

Данная игра относится к классу задач, решаемых методом динамического программирования. А именно: для получения искомого результата необходимо сначала подсчитать и запомнить в таблице значения игры для частей исходной замкнутой ломаной длины от 2-х

до n ребер, начинающихся в каждой из ее вершин (элементы таблицы для ломаных длины 1 совпадают со значениями вершин). Причем такую таблицу будем заполнять последовательно для всех частей сначала только длины 2, потом 3 и так далее до частей длины n, соответствующих всей ломаной, с одним исключенным ребром. При подсчете результатов игры для частей ломаной длины m будем использовать уже полученные результаты для элементов меньшего размера.

Делать это будем следующим образом. Пусть мы хотим подсчитать значение игры для части ломаной длины m, начинающейся с вершины с номером i. Тогда максимальное значение игры, которое можно получить для этой части ломаной, можно вычислить, испытав в качестве последнего арифметического действия в такой игре все операции, входящие в данную часть исходного полигона. Если мы предположили, что последним оказывается j-я из $m-1$ арифметической операции рассматриваемого элемента, то в случае, когда эта операция — сложение, значение игры равно сумме максимального значения игры на части ломаной длины j, начинающейся с i-й вершины, и максимального значения игры на части ломаной длины $m-j$, начинающейся с вершины $i+j$. С умножением ситуация несколько более сложная. Так как в вершинах полигона могут находиться и отрицательные числа, то для нахождения максимума из произведений пар чисел, являющихся различными значениями игры для двух смежных элементов ломаной, нужно знать не только максимальные, но и минимальные значения среди результатов игр на этих частях полигона, в том числе в силу того, что произведение двух отрицательных чисел есть положительное число. В общем же случае, если обозначить за `rmax[i,j]` — максимальное значение игры на элементе полигона длины `j`, начинающегося с вершины `i`, а `rmin[i,j]` — минимальное значение игры на этом же элементе и `j`-я операция является умножением, то значение игры будет равно

```
max{rmax[i,j]*rmax[i+j,m-j],
    rmin[i,j]*rmin[i+j,m-j],
    rmin[i,j]*rmax[i+j,m-j],
    rmax[i,j]*rmin[i+j,m-j]}.
```

Здесь `rmin[i+j,m-j]` и `rmax[i+j,m-j]` — минимальное и максимальное значение игры на части ломаной длины `m-j`, начинающейся с вершины `i+j`. В силу порядка построения таблицы `rmax` значения `rmax[i,j]` и `rmax[i+j,m-j]` уже известны. Очевидно, что параллельно на каждой части ломаной надо вычислять и минимальные значения игры, запоминая их в таблице `rmin`. Окончательно же значение элемента `rmax[i,m]` равно максимальному из значений игр, полученных при различных значениях `j` — от `1` до `m-1`.

После того как таблица `rmax` окажется полностью заполненной, то результатом всей игры в целом будет максимальное из значений таблицы, соответствующих ломаным длины `n`: `max{rmax[i,n], i=1,2,…,n}`.

Описанный алгоритм и реализован в приведенной ниже программе. Для того, чтобы программа стала прозрачной, для представления замкнутой ломаной, состоящей из n элементов, используем массивы длиной $2n$, продублировав информацию о ломаной (за вершиной с номером n поместим информацию о вершине с номером 1, затем 2 и так далее снова до N). Тогда, начиная с любой вершины ломаной, мы сможем отсчитать даже N звеньев, не упираясь в "конец полигона" — вершину с номером N, и проблемы с реализацией алгоритма исчезнут.

```
var n: integer;
    op:array [1..100] of char;  {двойной массив для знаков операций}
    x: array [1..50] of integer;{массив для операндов}
    rmax, rmin:array[1..100,1..50] of longint;{двойные таблицы}
    max_res,min_res:longint;
procedure ReadData;
var i: integer;
    c: char;
begin
   Assign(input,'POLYGON.IN');
  Reset(input);
  ReadLn(n);
  for i :=1 to n do
  begin
    read(op[i]);
    read(x[i]);
    read(c) {считываем разделительный пробел}
  end;
end;
function Min(a,b:longint):longint;
begin
  if a<b then Min:=a else Min:=b;
end;
```

```
function  Max(a,b:longint):longint;
begin
  if a>b then Max:=a else Max:=b;
end;
procedure Solve;
var i,m,j,rr: integer;
begin
  for i:=1 to n do
    begin
       rmax[i,1]:=x[i];rmax[n+i,1]:=x[i];
       rmin[i,1]:=x[i];rmin[n+i,1]:=x[i];
       op[n+i]:=op[i];{дублируем знаки операций}
    end;
  for m:=2 to n do {m - длина ломаной}
  for i:=1 to n do {i - начальное звено}
    begin
      max_res:=-maxlongint;
      min_res:= maxlongint;
      for j:=1 to m-1 do {цикл по номеру последнего действия}
        case op[i+j] of
        't':begin
              max_res:=Max(max_res,rmax[i,j]+rmax[i+j,m-j]);
              min_res:=Min(min_res,rmin[i,j]+rmin[i+j,m-j]);
            end;
        'x':begin
              max_res:=Max(max_res,
                       Max(rmax[i,j]*rmin[i+j,m-j],
                       Max(rmin[i,j]*rmax[i+j,m-j],
                       Max(rmax[i,j]*rmax[i+j,m-j],
                           rmin[i,j]*rmin[i+j,m-j]))));
              min_res:=Min(min_res,
                       Min(rmax[i,j]*rmin[i+j,m-j],
                       Min(rmin[i,j]*rmax[i+j,m-j],
                       Min(rmax[i,j]*rmax[i+j,m-j],
                           rmin[i,j]*rmin[i+j,m-j]))))
            end;
        end;
       rmax[i,m]:=max_res; rmax[i+n,m]:=max_res;
       rmin[i,m]:=min_res; rmin[i+n,m]:=min_res
    end;
  {ищем максимальное значение среди ломаных длины n}
  max_res:=rmax[1,n];
  for i:=2 to n do
    if rmax[i,n]>=max_res then max_res:=rmax[i,n];
  Assign(output,'POLYGON.OUT');
  Rewrite(output);
  WriteLn(max_res);
  for i:=1 to n do
    if rmax[i,n]=max_res then Write(i,' ')
end;
begin {основная программа}
  ReadData;
  Solve
end.
```

Картинки

На стене наклеено некоторое количество плакатов, фотографий и других картинок прямоугольной формы. Все их стороны либо вертикальны, либо горизонтальны. Каждый прямоугольник может быть частично или полностью перекрыт другими прямоугольниками. **Периметром** называется общая длина внутренних и внешних границ фигур, полученных путем наложения прямоугольников.

Задание. Написать программу, вычисляющую значение полученного периметра.

На *рис.* 1 показана фигура из 7 прямоугольников

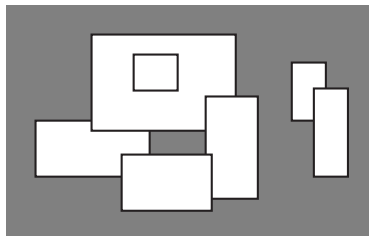


Рис. 1. Фигура из 7 прямоугольников

Соответствующие границы полученной фигуры показаны на *рис.* 2.

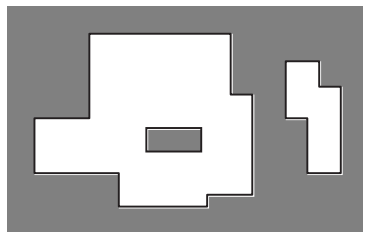


Рис. 2. Границы фигур из прямоугольников

Вершины всех прямоугольников имеют целочисленные координаты.

Входные данные

Первая строка входного файла PICTURE.IN содержит число прямоугольников, наклеенных на стену.

В каждой последующей строке приводятся целочисленные координаты нижней левой вершины и верхней правой вершины каждого прямоугольника. Значения этих координат даны как упорядоченные пары, состоящие из *X*-координаты и следующей за ней *Y*-координаты.

Пример входного файла:

```
7
-15 0 5 10
-5 8 20 25
15 -4 24 14
0 -6 16 4
2 15 10 22
30 10 36 20
34 0 40 16
```

Этот файл соответствует примеру, приведенному на *рис.* 1.

Выходные данные

Выходной файл должен содержать только одну строку с неотрицательным целым числом, которое является периметром для заданных прямоугольников.

Пример выходного файла:

```
228
```

Это соответствует выходному файлу для примера, приведенного выше.

Ограничения

$0 \le$ число прямоугольников < 5000

Все координаты находятся в пределах $[-10000, 10000]$, и каждый заданный прямоугольник имеет положительную площадь. Для хранения результата достаточно 32 битов.

Решение

Решение этой задачи технически является относительно сложным. То есть, даже придумав алгоритм решения задачи, не очень просто реализовать его без ошибок. Один из возможных путей решения заключается в построении всех замкнутых ломаных, образующих контур объединения прямоугольников. Для этого придется искать объединение контура первых i прямоугольников с $i+1$-м. Однако более простым в реализации представляется другой алгоритм. Заключается он в отдельном подсчете суммы длин всех горизонтальных и вертикальных отрезков, образующих контур пересечения всех прямоугольников. При этом замкнутые ломаные, его образующие, формировать не придется. Так как в условии задачи координаты границ картинок являются целочисленными, то в принципе задачу можно решать дискретно, то есть фактически просто подсчитывая количество точек в линиях контура. Однако при увеличении значений координат или при отказе от их целочисленности подобная процедура становится невозможной. Поэтому в предлагаемом ниже решении целочисленность координат практически не используется.

Рассмотрим общую идею алгоритма. Сначала при считывании исходных данных сформируем два отдельных множества отрезков: xSect — $2n$ отрезков, параллельных оси *OY*, и ySect — $2n$ отрезков, параллельных оси *OX*. Каждый отрезок описывается 4 числами: a — значение общей координаты вершин отрезка, $a1$ и $a2$ — меньшее и большее значения различных координат отрезка и f — признак начала (1) или конца (-1) прямоугольника. Так, для прямоугольника $(0,0)-(1,2)$ отрезок $(0,2)-(1,2)$ принадлежит множеству ySect и описывается числами $a=2$, $a1=0$, $a2=1$, $f=-1$. Затем каждое множество отрезков сортируется по возрастанию: сначала по значению a, а в случае равенства зна-

чений a — по $a1$. Теперь все отрезки, находящиеся на одном "уровне", расположены подряд. Рассматривая последовательно каждый уровень сначала по вертикали, затем по горизонтали и пересечение его отрезков с перпендикулярными отрезками, пересекающими его, подсчитаем вклад данного уровня в периметр (см. *рис.* 3).

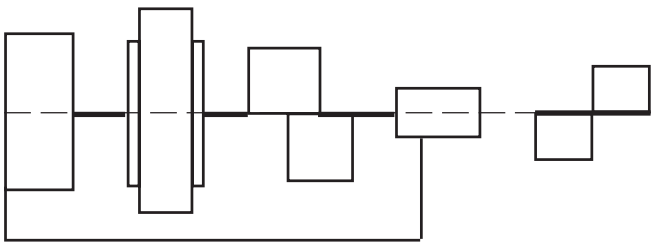


Рис. 3.

Последняя операция и является в программе основной. Для ее реализации сначала пересекающиеся отрезки одного уровня совмещаются в один. Затем из полученного совмещенного отрезка удаляются интервалы, принадлежащие прямоугольникам, расположенным по разные стороны от рассматриваемого уровня. В приведенной ниже программе это делается с помощью процедуры `Discrete` — дискретизации совмещенных отрезков, то есть в ней анализируются все точки интервала (другие способы решения данной проблемы возможны, но они существенно усложняют программу). Так, на приведенном рисунке отрезки, оставшиеся после двукратного выполнения данной процедуры допустимыми, обозначены пунктирными и утолщенными линиями. Вклад каждого из полученных отрезков в периметр считается следующим образом. Сначала подсчитывается, сколько прямоугольников, пересекающих наш уровень, началось и не закончилось до начала нашего отрезка. Для этого у отрезков, перпендикулярных нашему уровню, пересекающих его и расположенных до точки начала анализируемого отрезка включительно, суммируются значения f. Если сумма окажется больше 0, то начало отрезка находится внутри одного или нескольких прямоугольников. Далее рассматриваются пересечения уже непосредственно нашего отрезка с перпендикулярными ему. Пересечение отрезков, при котором указанная сумма оказалась равной 0, указывает на начало вхождения части отрезка в общий контур, а если сумма возросла от 0 до 1, то наш отрезок опять попадает внутрь прямоугольника и периметр следует увеличить на расстояние между этой точкой пересечения и запомненной ранее точки с нулевой суммой. Когда анализ отрезка закончен, то если сумма равна 0 (конец отрезка входит в контур объединения картинок), периметр также должен быть увеличен. Так, к началу первого отрезка, показанного на рисунке уровня, описанная сумма оказывается равной 1. После второго пересечения сумма уже равна 0, что и определяет начало контура. Описанная процедура аналогична проверке арифметических выражений со скобками, причем отрезки, помеченные "1", соответствуют открывающимся скобкам, а "—1"— закрывающимся, и интересуют нас лишь безскобочные части "выражения". В результате на рисунке утолщенными линиями показаны части отрезков, образующие контур объединения картинок, длина которых должна войти в периметр.

Алгоритм анализа всех отрезков произвольного уровня применим как к горизонтальным отрезкам, так и к вертикальным. Поэтому в приведенной ниже программе он описан как универсальный и применяется дважды. Множества отрезков xSect и ySect размещаются в динамической памяти в силу нехватки статической для хранения значений 10 000 отрезков.

```
type {тип для описания отрезка}
  rr=record
       a,a1,a2:integer;{общее a и различные a1 и a2 значения координат отрезка}
       f:-1..1;{индикатор начала f=1 или конца f=-1 прямоугольника}
     end;
  Arr=array[1..10000] of ^rr;
var xSect,ySect:^Arr;{отрезки, параллельные осям OY и OX соответственно}
    nSect{количество отрезков},n{количество прямоугольников}:integer;
    p:longint;{значение периметра}
    x:array[-10000..10000]of shortint;{массив для дискретизации одного уровня}
function less(var i,j:rr):boolean;
{определяет операцию '<' для отрезков параллельных прямых}
var b: boolean;
begin
 b:=i.a<j.a;
 if i.a=j.a then b:=i.a1<j.a1;
 less:=b
end;
```

```
procedure Sort(var s:arr;l, r: Integer);
{сортирует методом QuickSort отрезки параллельных прямых по возрастанию,
 в качестве отношения '<' между отрезками используя функцию less}
var
  i,j:integer;
  x,y:rr;
begin
  i:=l;j:=r;y:=s[(l+r)DIV 2]^;
  repeat
    while less(s[i]^,y) do i:=i+1;
    while less(y,s[j]^) do j:=j-1;
    if i<=j then
      begin
        x:=s[i]^; s[i]^:=s[j]^; s[j]^:=x;
        i:=i+1; j:=j-1;
      end
  until i>j;
  if l<j then Sort(s,l,j);
  if i<r then Sort(s,i,r);
end;
procedure Prepare;
{формирует и упорядочивает массивы отрезков xSect^ и ySect^}
var x1,y1,x2,y2,i:integer;
begin
  Assign(input,'PICTURE.IN');
  Reset(input);
  ReadLn(n);
  New(xSect); New(ySect);
  nSect:=2*n;{количество отрезков каждого вида}
  for i := 1 to n do
    begin
     read(x1,y1,x2,y2);
       new(xSect^[2*i-1]);new(xSect^[2*i]);new(ySect^[2*i-1]);new(ySect^[2*i]);
       xSect^[2*i-1]^.f:=1;xSect^[2*i]^.f:=-1;
       xSect^[2*i-1]^.a:=x1;xSect^[2*i-1]^.a1:=y1;xSect^[2*i-1]^.a2:=y2;
       xSect^[2*i  ]^.a:=x2;xSect^[2*i  ]^.a1:=y1;xSect^[2*i  ]^.a2:=y2;
       ySect^[2*i-1]^.f:=1;ySect^[2*i]^.f:=-1;
       ySect^[2*i-1]^.a:=y1;ySect^[2*i-1]^.a1:=x1;ySect^[2*i-1]^.a2:=x2;
       ySect^[2*i  ]^.a:=y2;ySect^[2*i  ]^.a1:=x1;ySect^[2*i  ]^.a2:=x2;
    end;
  Sort(xSect^,1,nSect);
  Sort(ySect^,1,nSect);
end;
procedure discrete(var s:arr;b,e:integer);
var i,j:integer;
begin
  fillchar(x,sizeof(x),10);{10 - метка нерассмотренной точки}
  for i:=b to e do{цикл по номерам совмещенных отрезков}
    for j:=s[i]^.a1 to s[i]^.a2-1 do{цикл по точкам i-го отрезка}
      if x[j]=10 then x[j]:=s[i]^.f
      else if x[j]<>s[i]^.f then x[j]:=0;
         {x[j]=0 - точка исключается из рассмотрения}
end;
procedure Solve(var s,t:arr;var p:longint);
{для отрезков массива s подсчитывает их вклад в периметр p,
 t - массив перпендикулярных отрезков}
```

```
var  i,j,k,l,m,q,e,b,f:longint;
begin
  i:=1;
  while i<=nSect do {цикл по отрезкам}
   begin
    j:=i;
    e:=s[j]^.a2;
    repeat {совмещаем пересекающиеся отрезки одной прямой}
      if s[i]^.a2>e then e:=s[i]^.a2;
      inc(i)
    until (i>nSect)or(s[i]^.a<>s[i-1]^.a)or(s[i]^.a1>e);
    q:=s[j]^.a1; f:=e; b:=q;{b и f - начало и конец совмещенных отрезков}
    if (j=1)or(s[j]^.a>s[j-1]^.a) then {начало уровня}
      begin m:=0;k:=1 end;{m - значение контрольной суммы}
    if j<i-1 then discrete(s,j,i-1);{проверяем несколько совмещенных отрезков}
    repeat
      if j<i-1 then {случай нескольких совмещенных отрезков}
       begin
         l:=q;
         while x[l]=0 do inc(l);
         q:=l;
         while (x[l]=-1)or(x[l]=1) do inc(l);
         e:=l
       end;{q и e - начало и конец допустимого отрезка}
      while (k<=nSect)and(t[k]^.a<=q) do{отрезки из t, расположенные до j-го}
       begin
          if (t[k]^.a1<s[j]^.a)and(t[k]^.a2>s[j]^.a) then
          {пересечение отрезков из t с нашим уровнем a} m:=m+t[k]^.f;
         inc(k)
       end;
      while (k<=nSect)and(t[k]^.a>b)and(t[k]^.a<=e) do
      {возможное пересечение отрезков из t с совмещенными отрезками j - i-1}
       begin
          if (t[k]^.a1<s[j]^.a)and(t[k]^.a2>s[j]^.a) then
          {отрезки пересекаются}
           begin
             m:=m+t[k]^.f;
              if (m=1)and(t[k]^.f>0) then{начинается новый прямоугольник}
                inc(p,t[k]^.a-q)
              else if m=0 then {пересекающиеся прямоугольники закончились}
                    q:=t[k]^.a
           end;
         inc(k)
       end;
      if m=0 then inc(p,e-q);{анализ конца отрезка}
      if j<i-1 then for l:=b to e-1 do x[l]:=0
    until e=f {обработаны все совмещенные отрезки}
   end
end;
begin
  Prepare;
  p:=0;
  Solve(xSect^,ySect^,p);{вклад в периметр отрезков, параллельных OY}
  Solve(ySect^,xSect^,p);{вклад в периметр отрезков, параллельных OX}
  Assign(output, 'PICTURE.OUT');
  rewrite(output);
  writeln(p);
end.
```

О целях, задачах и методике проведения олимпиад по информатике
(на примере олимпиад КБГУ)

В.М. КАЗИЕВ

Отмеченным в названии статьи проблемам посвящен ряд работ, например, [1—6, 13]. Появление еще одной, данной статьи вызвано, как представляется автору, необходимостью приведения олимпиад по информатике в более полное соответствие с все более частой и обоснованной трактовкой информатики как теоретической или математической информатики (см., например, [7—8]), а также желанием автора поделиться опытом их проведения в Кабардино-Балкарском госуниверситете. Естественно, что при этом олимпиады, турниры по информационным технологиям, программированию имеют полное право на самостоятельное существование — аналогично, как и методики их проведения, отличные от излагаемых ниже.

1. Цели и задачи олимпиады

Цели проведения олимпиады по информатике и преследуемые ею задачи достаточно разнообразны, и, на наш взгляд, можно выделить следующие основные цели:

- развитие способностей по самостоятельному приобретению знаний, умений, навыков, развитие интуиции и ускорение процесса перехода от обучения к научению, самообучению — наивысшей ступени образовательного процесса;
- отбор наиболее образованных, смекалистых, изобретательных, одаренных участников, нетривиально, правильно и логически мыслящих и обладающих высоким уровнем алгоритмического, системного мышления, умеющих находить оптимальные и верные решения, способных к индивидуальному соревнованию (а не только и не столько тех, кто овладел тем или иным набором — часто стандартным, приемов и методов программирования в той или иной среде, кто имел доступ к той или иной хорошей технике и среде, кто был "натаскан" под те или иные задачи и не имеет при этом познаний по ряду тем базового курса основ информатики);
- привлечение в науку (через задачи, через решения) талантливой молодежи и организация дальнейшей работы с победителями олимпиад по все более полному выявлению и развитию их способностей, знаний и умений, разработка программ работы с такой молодежью (любая программа, оторванная от участника и его наставника, нам кажется нежизнеспособной);
- профессиональная ориентация участников олимпиад и развитие интереса к своей будущей профессии;
- усиление межпредметных связей информатики и других дисциплин;
- проверка уровня преподавания информатики в учебных заведениях (представленных участниками олимпиады); это один из реальных и объективных хороших способов подтверждения квалификации преподавателя информатики, охвата школьной программы (т.е. "горизонтальная составляющая квалификации"), а также эрудиции и знаний из области внешкольной программы ("вертикальная составляющая");
- проверка роста знаний, умений, навыков за прошедший этап и формулировка ориентиров для последующего этапа их развития;
- проверка наличия у участников необходимого понятийного аппарата и инструментария для решения проблем информатики (математического, системного, информационно-логического и технологического);
- проведение своеобразного конкурса задач, формирование банка данных таких задач (решений), обмен задачами, опытом через этот банк;
- приобщение молодых преподавателей к олимпиадному движению и их профессиональный рост.

2. Методика подбора задач для проведения олимпиад

При подборе олимпиадных задач необходимо руководствоваться, на наш взгляд, следующими основными критериями и положениями:

- оригинальность задачи или идеи, на основе которой составлена задача, причем под оригинальностью понимается не только оригинальность формулировки, условия задачи (оригинальность "сообщения"), а оригинальность подхода к ее решению (т.е. оригинальность "информации"); это сложнореализуемое требование, но это необходимое требование, и именно в этом проявляются мастерство и квалификация составителя задачи;
- желательно присутствие (особенно для туров высокого ранга) как четко структурируемых и четко формализуемых постановок или решений задач,

так и нечетко формализуемых, нечетко структурируемых; при этом желательно формализовать решения всех задач, в том числе — реализуемых с помощью эвристических процедур, приемов;

— четкая формализуемость и структурируемость, независимо от постановки, решения задач или требование доказательства корректности решения, отсутствие двусмысленностей или же неоднозначностей в решениях задач; при этом очень желательно, чтобы задачи имели различные варианты решения (требование многовариантности решения), среди которых есть оптимальное решение (понятие оптимальности конкретизируется в условии задачи), ибо способность к нахождению оптимальных решений в заданной операционной обстановке — одна из основных целей обучения информатике;

— освобождение условия задачи от всего второстепенного и наносного для основной идеи, заложенной в условии или в решении задачи; при этом желательна постановка задачи в увлекательной форме для младших школьников;

— сужение в постановке задачи операционной среды для решения задачи до минимально необходимого (т.е. "ограничение ресурсов, средств" — в терминах [5]), что позволяет выявить как уровень знаний, так и владение этими ресурсами, умение их применения к нахождению оптимальных решений;

— наличие задач различного уровня сложности и одной-двух задач, которые доступны каждому участнику ("утешительного характера" задачи), а также наличие возможности выбора задач в задании;

— задание должно охватывать все основные разделы базового курса информатики: информация и сообщение; кодирование и декодирование; системы счисления; алгебра предикатов; алгоритмы и алгоритмизация; исполнители алгоритмов; архитектура ЭВМ; языки программирования и методы трансляции; программное обеспечение и программирование; модели и моделирование (включая и вычислительный эксперимент); информационные технологии; ясно, что это достаточно сложно реализуемое требование, но известно, что некоторые олимпиады, проводимые без соблюдения этого требования, иногда "получают" победителя олимпиады, который не знает ряда разделов базового курса информатики и владеет в основном лишь комбинаторными или "полнопереборными" методами решения задач (последние, к слову сказать, часто пригодны для показа, как не надо решать задачу);

— при решении задач желательно предусмотреть использование основных методов информатики (линеаризация, идентификация, агрегирование, декомпозиция, макетирование и др. [9—10]);

— желательно наличие одной задачи, поддерживающей весь цикл решения задач по информатике, — от содержательной постановки и формулировки гипотез до получения и анализа решения, выявления причинно-следственных связей (для выявления знания технологии решения задач);

— часто используемое требование "об одной большой, со многими пунктами задачи" нам представляется несущественным и более подходящим для научно-исследовательской работы, так как смена условий, методов, понятийного аппарата позволяет выявить глубину их осознания и использования; невозможно, как правило, сформулировать одну задачу, для решения которой необходимо было бы привлечение всех разделов базового курса информатики.

3. Методика проведения олимпиады

Отметим, не претендуя на универсальное их применение, следующие методические положения:

— желательно проведение олимпиады как теоретического или же теоретико-практического тура, выделяя достаточно времени для решения задач, в том числе, если это необходимо, на ЭВМ (не менее 5—6 часов, с паузами); этим мы ставим олимпиаду по информатике на один уровень с олимпиадами по математике, физике и другим дисциплинам, проводимыми, как правило, в один тур (компьютер и компьютерные технологии имеют такое же отношение к этим дисциплинам, как и к информатике, хотя к последней — более ярко, глубоко и широко выраженное);

— желательно, если это возможно, предоставление всем участникам одной и той же операционной среды с одними и теми же ограничениями ресурсов (например, стандартная операционная среда для разработки алгоритмов — школьный алгоритмический язык, базовые типы и структуры данных этого языка, а для разработки программ — P-166/16/SB/CD/Турбо Паскаль);

— учет того факта, что если раньше безкомпьютерный вариант проведения олимпиады по информатике был вызван отсутствием компьютеров в школах, то сейчас такой вариант проведения этой олимпиады (теоретический) может быть обоснован как более четким выявлением составляющих предмета информатики (как теоретической информатики), так и количеством компьютеров, уже достаточным для проведения олимпиад по компьютерным технологиям и системам, в том числе в дистанционном режиме (налицо виток спирали развития олимпиад по информатике);

— если позволяют условия и уровень олимпиады, то представляет интерес проведение командного тура соревнования (особенно для практического тура — в

случае его необходимости), что позволит выявить способности участников коллективно решать проблему и продемонстрирует способности к агрегированию, декомпозиции, тестированию, макетированию и т.д.

4. Методика оценки заданий олимпиады

Основные составляющие предлагаемой методики:

— задачи оцениваются в баллах, сами баллы назначаются жюри и методической комиссией экспертным образом после того, как завершился тур — исходя из распределения участников, решивших те или иные задачи задания и последующего выравнивания по набравшему максимальное количество баллов, например, по формуле:

$$V_i = K_i / K_{\max},$$

где K_i — баллы i-го участника, $K_{\max}$ — максимальные баллы, V_i — баллы i-го участника после выравнивания;

— система экспертно назначаемых "поощрительных" и "штрафных" баллов, например, добавление баллов за оригинальное правильное решение и снятие баллов за правильно, но не оптимальным способом решенную задачу;

— оценивание не только (или не столько) количества прошедших правильно тестов, но и стиля, структуры данных, логики алгоритма или программы: часто программа, у которой прошло большинство тестов, может оказаться менее интересной и "интеллектуальной", чем программа с интересным и оригинальным методом решения, алгоритмом, структурой данных, для которой по какой-либо оплошности или незначительной ошибке не прошел тест.

5. Олимпиада завершена. Что дальше?

Олимпиада завершилась и выявила группу победителей, с которыми необходима дальнейшая серьезная, профессиональная работа для подготовки к следующему туру, к интересной научно-исследовательской работе. Основу этой работы могут составить мероприятия для поддержки олимпиадного тонуса:

— лекции по избранным главам информатики (комбинаторные алгоритмы, методы оценки алгоритмов, методы сортировки, компьютерная алгебра, информационные процессы, синергетика, математическое и компьютерное моделирование и др.);

— практикум по решению избранных задач информатики или развивающих задач [11];

— летние и зимние сборы (школы) и соревнования;

— подготовка, выпуск сборников олимпиадных задач (с указаниями и/или решениями);

— повышение квалификации преподавателей информатики на различных курсах повышения квалификации (по составлению и решению олимпиадных задач);

— приобщение победителей к научно-исследовательской работе на кафедрах вузов и к работе с младшими школьниками в качестве тьюторов.

6. Примеры заданий

В качестве примера реализации ряда положений, высказанных выше, приведем некоторые задания олимпиад различного уровня в Кабардино-Балкарии и КБГУ, проведенных под руководством автора в 1992—1997 годах. Решения более сложных из этих и других задач опубликованы в цикле работ "Развивающие задачи" (см. [10—12]) и поэтому здесь не приводятся — желающие могут получить их по адресу e-mail: **kaziev@ns.kbsu.ru** (отметим, что сборник из более 400 таких задач будет вскоре сдан в печать). Приведены задания только для 10—11-х классов.

6.1. Задание олимпиады лицея КБГУ

1. Решить уравнение: 128^{x-3} (Кбайт) $= 64^x$ (бит).

2. Описать словесно и математически закон формирования чисел данного ряда (начиная с третьего): 1, 1, 2, 5, 29, 866, ...

3. Найти неизвестные десятичные цифры А, Б, В за минимум кратких, точных рассуждений — утвердительных высказываний о цифре разряда, если АВВ5 + + ВВАА = ББ922.

4. Найти шифр, по которому текст вида "КРАСНЫЙ-АРБУЗ" закодирован как "ЛСБТОЪКБСВФИ", и закодировать им текст "ИНФОРМАЦИЯ".

5. Найти число в системе с основанием p, совпадающее со своим обратным кодом. Что можно сказать о всех таких p, для которых имеется решение этой задачи?

6. Составить алгоритм проверки вхождения заданного слова в заданный текст. Составить набор тестов для его тестирования.

7. Исполнитель Робот может выполнять команды: `Положить`, `Взять`, `Вперед(x)`, `Направо(y)`, где x — в метрах, y — в градусах. Составить оптимальный алгоритм переноса грузов, расположенных в вершинах квадрата с центром в начале координат и со стороной a метров, в центр этого квадрата. Вначале Робот находится в точке $A(2; 2)$ и "смотрит" в направлении луча $x > 0$.

8. Принтер печатает 10 Кбайт за 1 с. Сколько времени нужно ему для печати одной страницы размером 50 строк по 40 символов в строке?

9. Охарактеризовать одну новую информационную технологию (5—10 предложений).

6.2. Задание олимпиады г. Нальчика

1. Записать словами и математически закон образования всех десятичных чисел ряда: 1, 10, 11, 100, 111, 1000, 1111, ...

2. Петя, Вася, Дима, Коля и Гена хотят по-отдельности пойти в кинотеатры, где они еще не были. Петя не был в "Победе", "Востоке", "Авроре", "Мире". Вася был в "Авроре", "Мире". Дима не был в "Авроре". Коля не был в "Победе" и "Мире". Гена был в "Востоке", "Мире" и "Юности". Найти за минимум рассуждений (кратких, простых утвердительных предложений), кто и где был.

3. Найти основание p системы счисления и цифру n, если $33m5n + 2n443 = 55424$.

Пример выполнен в p-й системе счисления, цифра m — максимальная в ней.

4. Можно ли логические функции

$$z = \bar{x} \vee \overline{x \vee y} \vee y,\ w = x \vee \overline{y \vee x} \vee \bar{y}$$

реализовать одним набором логических вентилей и почему? Каковы их минимальный состав и структура?

5. Определить функции $\mathrm{mod}(x, y)$, $\mathrm{int}(x)$, $\mathrm{abs}(x)$, $\max(x, y)$ через какие-то другие функции наиболее оптимально (рационально).

6. Составить эффективный по числу команд, операций, операндов алгоритм нахождения корней квадратного уравнения (если они существуют) с заданной точностью.

7. Задан квадрат со стороной n см. Около него описан и в него вписан квадраты. Вписанная окружность разделена на 4 равные части. Как можно раскрасить тремя цветами (сделать "заливку") все части большей окружности, чтобы никакие две соседние части разного цвета не имели бы протяженной общей границы? Можно ли ограничиться двумя цветами?

8. Даны натуральные m, p. Составить эффективный алгоритм или программу для вычисления всех чисел ряда $x[1], x[2], \ldots$ по правилу: $x[1]$ — остаток от деления m на p, $x[2]$ — остаток от деления $m \cdot x[1]$ на p, $x[3]$ — остаток от деления $m \cdot x[2]$ на p и т.д.

6.3. Задание республиканской олимпиады (двух туров)

1. Упростить, используя как можно меньше аксиом алгебры предикатов и за как можно меньшее число их применений:

$$z = \overline{x \vee y} \vee \bar{x} \wedge y \vee \bar{x}.$$

2. В факториальной системе счисления целые числа записывают как линейную комбинацию факториалов, например:

$2457 = 2 \cdot 3! + 4 \cdot 2! + 5 \cdot 1! + 7 \cdot 0!$

$(0! = 1,\ n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n)$

а) Позиционна ли эта система и почему? (Обосновать.)

б) Описать процедуры или формулы переводов из десятичной системы в факториальную и обратно.

в) Вычислить диапазон целых чисел, представимых в такой системе разрядности n.

г) Составить эффективные алгоритмы переводов из десятичной системы в факториальную и наоборот.

3. Спроектировать память условной трехадресной ЭВМ фон Неймана (найти минимально достаточную разрядность, найти структуру ячейки ОЗУ и регистров, сделать рисунок), если ячейки и все регистры ЭВМ одинаковой длины (т.е. память ЭВМ — однородна), ЭВМ реализует n различных машинных операций, ОЗУ ЭВМ равно m Мбайт, а адресуется всегда слово из двух байт (3 адреса — для двух операндов и результата операции).

4. Составить алгоритм нахождения количества всех различных двоичных сообщений (последовательностей) длины n битов, используя при этом (явно) только операции сложения. Обоснуйте (или докажите) правильность своего алгоритма.

5. Задан произвольный ряд вещественных чисел $x[1], x[2], \ldots, x[n]$. Составить алгоритм (программу) выдачи номеров всех этих чисел в ряде, который получается из данного ряда после его упорядочивания по возрастанию. Сортировать исходный ряд или вводить его копию нельзя. Составить минимальный и достаточный, на ваш взгляд, набор тестов. Оценить алгоритм.

6. Составить алгоритм (программу) нахождения количества слов заданной длины в заданном тексте, если разделителем слов в этом тексте может быть только один пробел, одна точка или одна запятая. Поставить эту задачу как корректную задачу для программирования на ЭВМ.

7. Составить программу удаления из заданного текста всех слов, стоящих на нечетных местах, если все слова разделены в тексте только одним пробелом. Составить минимальный и достаточный набор тестов для тестирования программы.

8. "Квадрат-палиндром" — это число, являющееся квадратом натурального числа и читаемое слева и справа одинаково, например, 9, 121, 484. Составить эффективную (по количеству команд, операций, операндов, времени, памяти) программу поиска всех таких чисел, не превосходящих заданного числа. Составить минимально достаточный набор тестов для тестирования программы.

Литература

1. *В.Прохоров.* Уроки олимпиады. ИНФО, № 2, 1990.

2. *В.Кирюхин.* Всесоюзные олимпиады по информатике — вчера, сегодня, завтра. ИНФО, № 3, 1991.

3. *В.Прохоров, С.Раков, В.Кирюхин.* Олимпиадные задачи по информатике. ИНФО, № 3, 1991.

4. *Н.Бадин, С.Волченков, Н.Дашниц, П.Корнилов.* Ярославские олимпиады по информатике. Ярославль, 1995.

5. *А.Бочкин, С.Кунцевич.* Метод ограничения средств при разработке олимпиадных задач. ИНФО, № 6, 1996.

6. Олимпиады: спецвыпуски. Информатика, № 13, 1996; № 12, 1997.

7. *А.Семенов.* Математическая информатика в школе. ИНФО, № 5, 1995.

8. *В.Казиев.* Системно-алгебраический подход к основам информатики. ИНФО, № 4, 1996.

9. *В.Казиев.* Информатика. Часть 1. Теория. Нальчик, 1997.

10. *В.Казиев.* Информатика. Часть 2. Задачи. Нальчик, 1997.

11. *В.Казиев.* Развивающие задачи. ИНФО, № 3, 1997; № 2, 1998.

12. *В.Казиев.* Развивающие задачи. Информатика в уроках и задачах (приложение к ИНФО), № 2, 1998.

13. *В.Казиев.* Олимпиада КБГУ по информатике. Информатика, № 47, 1996.

Восьмая олимпиада по математике и криптографии 1998 года

20 декабря 1998 г. состоялась VIII олимпиада по криптографии и математике для школьников 9–11-х классов. В олимпиаде приняло участие около 500 школьников г. Москвы, Подмосковья и других городов России. На решение задач отводилось 4 часа. Всех задач не решил никто, хотя каждая задача была решена. Подведение итогов и разбор задач состоялись 17 января 1999 г. Победители олимпиады награждены дипломами, грамотами и книгами по математике и криптографии. Адрес оргкомитета олимпиады: olymp@fssr.ru

Задачи

1. На рисунке изображена эмблема олимпиады. Она представляет собой замкнутую ленту, сложенную так, что образовавшиеся просветы являются одинаковыми равносторонними треугольниками. Если в некотором месте ленту разрезать перпендикулярно к ее краям и развернуть, то получится прямоугольник. Найдите отношение его сторон.

Рис. 1

2. Сообщение, составленное из нулей и единиц, шифруется двумя способами. При первом способе каждый нуль заменяется на последовательность из k_1 нулей и следующих за ними k_2 единиц, а каждая единица заменяется на последовательность из k_3 нулей. При втором способе шифрования каждая единица заменяется на последовательность из k_4 единиц и следующих за ними k_5 нулей, а каждый нуль заменяется на последовательность из k_6 нулей. При каких натуральных значениях k_i, $i = 1, 2, ..., 6$ найдется хотя бы одно сообщение, которое будет одинаково зашифровано обоими способами? Укажите общий вид таких сообщений.

3. Сообщение, подлежащее зашифрованию, представляет собой цифровую последовательность, составленную из дат рождения 6 членов оргкомитета олимпиады. Каждая дата представлена в виде последовательности из 8 цифр, первые две из которых обозначают день, следующие две — месяц, а остальные — год. Например, дата рождения великого русского математика Л.Эйлера, 4 апреля 1707 года, представляется в виде последовательности 04041707.

Для зашифрования сообщения строится ключевая последовательность длины 48. Для ее построения все нечетные простые числа, меньшие 100, выписываются через запятую в таком порядке, что модуль разности соседних чисел есть та или иная степень числа 2. При этом каждое простое число выписано ровно один раз, а числа 3, 5 и 7 написаны в виде 03, 05 и 07 соответственно. Удалив запятые из записи этой последовательности, получим ключевую последовательность.

При зашифровании цифровой последовательности, представляющей сообщение, ее цифры почленно складываются с цифрами ключевой последовательности и каждая полученная сумма заменяется ее остатком от деления на 10. В результате зашифрования сообщения получена последовательность:

150220454215266744305682533362327363924975709849

Определите даты рождения членов оргкомитета олимпиады.

4. Квадрат размера 13×13 разбит на клетки размера 1×1 (*рис.* 2). В начальный момент некоторые клетки окрашены в серый цвет, а остальные — в белый. По клеткам квадрата прыгает Криптоша. В момент попадания Криптоши в очередную клетку происходит изменение цвета на противоположный у всех тех клеток, расстояния от центров которых до центра клетки с Криптошей есть натуральные числа. После того, как Криптоша побывал в каждой клетке квадрата ровно 1999 раз, квадрат оказался раскрашенным так, как показано на рисунке. Восстановите цвет всех клеток квадрата в начальный момент.

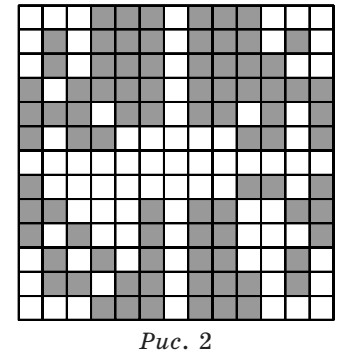


Рис. 2

5. Для всех действительных a, b решить уравнение:

$$\frac{a}{1 - bx} = \frac{b}{1 - ax}.$$

6. Разложить число $2^{30} + 1$ на простые множители.

Восьмая олимпиада по математике и криптографии 1998 года

Решения

№ 1

1. Проведем прямые, проходящие через точки пересечения границ сложенной ленты, параллельно ее краям. Тогда лента разобьется на равные равносторонние треугольники.
2. Отметим цифрой 0 все просветы, а цифрой 2 — все треугольники, которые получились наложением друг на друга двух треугольников в сложенной ленте. Выделим дополнительно ряд треугольников вне ленты, как показано на *рис.* 3. В полученной фигуре число удвоенных треугольников равно числу треугольников, отмеченных 0. Значит, площадь всей ленты равна площади четырехугольника *ABCD*, разбитого на треугольники.

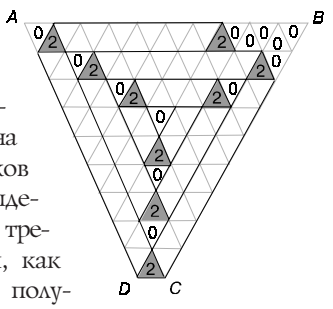


Рис. 3

3. Числа треугольников в горизонтальных рядах *ABCD* образуют 9 последовательных членов арифметической прогрессии с первым членом, равным 3 (нижний ряд), и разностью 2. Следовательно, общее число треугольников равно:

$$N = \frac{2 \cdot 3 + (9-1)\cdot 2}{2} \cdot 9 = 99.$$

4. Если h — ширина ленты, то площадь одного равностороннего треугольника с высотой h равна $S_0 = h^2 \operatorname{ctg} 60^\circ = \frac{h^2}{\sqrt{3}}$. Следовательно, площадь всей ленты равна $S = 99 \cdot S_0 = 33\sqrt{3} \cdot h^2$. С другой стороны, если длина прямоугольника, полученного после разрезания ленты, равна l, то площадь $S = l \cdot h$. Отсюда находим, что $\frac{l}{h} = 33\sqrt{3}$.

№ 2

Последовательность из k нулей или k единиц будем обозначать 0^k или 1^k. Тогда шифрование по I и II способу описывается в виде

I	II
$0 \to 0^{k_1}\ 1^{k_2}$	$1 \to 1^{k_4}\ 0^{k_5}$
$1 \to 0^{k_3}$	$0 \to 0^{k_6}$

(1)

1. В зашифрованном сообщении все серии из единиц имеют длину k_2 в первом случае и длину k_4 во втором случае, поэтому необходимо, чтобы

$$k_2 = k_4 \qquad (2)$$

Отсюда следует, что число нулей в сообщении равно числу единиц, так как число серий из единиц в шифрованном сообщении совпадает с числом нулей сообщения (при шифровании I способом) и с числом единиц сообщения (при шифровании II способом).

2. Учитывая, что число нулей и число единиц в сообщении одинаково, например n, посчитаем число нулей в зашифрованном сообщении I и II способами: $nk_1 + nk_3 = nk_5 + nk_6$.

Итак,

$$k_1 + k_3 = k_5 + k_6 \qquad (3)$$

3. Из (1) легко видеть, что сообщение должно начинаться с нуля и кончаться единицей.

Итак, пусть сообщение имеет вид $\underbrace{0 \ldots 0}_{a} 1$, где $a = 1, 2, \ldots$. Тогда шифрованное сообщение представляется в виде

$a = 1$

$$01 \to \begin{cases} 0^{k_1}\ 1^{k_2}\ 0^{k_3} & \text{I способ} \\ 0^{k_6}\ 1^{k_2}\ 0^{k_5} & \text{II способ} \end{cases}$$

$a > 1$

$$\underbrace{0 \ldots 0}_{a} 1 \to \begin{cases} 0^{k_1}\ 1^{k_2}\ 0^{k_1} \ldots 1^{k_2} 0^{k_3} & \text{I способ} \\ 0^{ak_6}\ 1^{k_2}\ 0^{k_5} & \text{II способ} \end{cases}$$

Случай $\boldsymbol{a = 1}$ сразу дает ответ. Необходимо, чтобы $k_1 = k_6$, а из (3) следует, что выполняется $k_3 = k_5$.

Случай $\boldsymbol{a > 1}$ приводит к условиям

$$k_1 = ak_6 \qquad a = 1, 2, \ldots$$
$$k_1 = k_5 + bk_6 \qquad b = 0, 1, 2, \ldots$$

Подставляя $k_1 = k_5 + bk_6$ в (3), получаем необходимость соотношения

$$k_3 = (1 - b)k_6$$

что при натуральных k_i и $b = 0, 1, 2, \ldots$ возможно лишь при $b = 0$, то есть $k_3 = k_6$, а из (3) следует, что $k_1 = k_5$. Итак, $k_2 = k_4$, $k_5 = k_1 = ak_6 = ak_3$.

При этих условиях добавление еще $(a - 1)$ единиц к сообщению $\underbrace{0 \ldots 0}_{a} 1$ приводит к одинаковым шифрованным сообщениям.

Ответ: При $k_2 = k_4$, $k_1 = k_6$, $k_3 = k_5$ сообщения вида 0101 ... 01 шифруются одинаково.

При $k_2 = k_4$, $k_1 = k_5 = ak_3 = ak_6$, $a = 2, 3, \ldots$ сообщения вида $\underbrace{0 \ldots 0}_{a} \underbrace{1 \ldots 1}_{a} \ldots \underbrace{0 \ldots 0}_{a} \underbrace{1 \ldots 1}_{a}$ шифруются одинаково.

Замечание. Первый ответ не является частным случаем второго при $a = 1$.

№ 3

Естественно предположить, что все члены оргкомитета родились в XX веке. Отсюда сразу находим, что на 3, 7, 11, 15, 19-м и 23 местах последовательности простых чисел стоят числа 11, 17, 47, 53, 83 и 89 соответственно.

Найдем, какие числа стоят рядом с указанными шестью. Для этого составим таблицу из возможных "соседей" (по условию, они должны быть простыми и отличаться на степень двойки):

11 — 13, 19, 43, 7, 3
47 — 79, 43, 31
53 — 61, 37
83 — 79, 67, 19
89 — 97, 73.

Учитывая, что первая цифра в номере месяца принимает значения только 0 или 1, построим следующую таблицу:

15	02	20	45	42	15	26	67	44	30	56	82	53	33	62	32	73	63	92	49	75	70	98	49
		19				19				19				19				19				19	
		11				17			31	47			37	53	61		67	83			73	89	97
	03		03		13		13				43								19				
	07		07		19		19				79								79				
			13																				
			19																				
			43																				

где в первой строке — шифрованное сообщение, вторая строка — известные участки исходного сообщения, третья строка — ставшие известными участки ключевой последовательности, остальные строки — возможные варианты ключевой последовательности в соответствующих позициях.

При составлении таблицы учитывалось, что каждое число должно встретиться ровно один раз. Числа 31, 37, 67, 73 расставились однозначно. Зная их расстановку, однозначно находятся места для 61 и 97.

Снова выпишем известные элементы последовательности и варианты для их соседей (первые две строки на этом шаге не представляют интереса, поэтому их пропускаем):

		11				17			31	47			37	53	61		67	83			73	89	97
	03		03		13		13				43								19				
	07		07		19		19				79								79				
			13																				
			19																				
			43																				

Возможные соседи для 61 — только 59 и 29, а для 67 — только 59 и 3. Поэтому между 61 и 67 может быть только число 59. Возможные соседи для 73 — 89, 71 и 41. Ни одно из этих чисел не может быть соседом для 19, а для 79 может быть только 71. Таким образом, однозначно определили положение чисел 71 и 79. Для числа 47 остался только один вариант продолжения — 43. Общим соседом для 43 и 37 может быть только 41. Перерисуем таблицу с учетом сделанных выводов:

		11				17			31	47	43	41	37	53	61	59	67	83	79	71	73	89	97
	03		03		13		13	29															
	07		07		19		19	23															
			13																				
			19																				

Участок 17 * * 31 имеет только два варианта доопределения: 17 19 23 31 и 17 13 29 31. Рассмотрим оба случая.

а) Выпишем фрагмент таблицы для первого случая

		11			13	17	19	23	31
	03		03						
	07		07						

Очевидно, что числа 3 и 7 должны обязательно быть соседями с числом 11. Число 29 еще не встречалось, значит, оно должно стоять либо на первом месте, либо на пятом. И то, и другое невозможно, так как в обоих позициях оно является соседом для числа 3 либо 7, что не соответствует условию (отличие соседних на степень двойки). Таким образом, рассматриваемый случай невозможен.

б) Выпишем фрагмент таблицы для второго случая

05		11		23	19	17	13	29	31
	03		03						
	07		07						

Очевидно, что числа 3 и 7 должны обязательно быть соседями числа 11. Число 5 может попасть только в первую ячейку (так как не может стоять рядом с 19). Значит, в пятой позиции — 23. Числа 3 и 7 теперь расставляются однозначно.

Таким образом, получили, что возможен всего один вариант ключевой последовательности. Составляем окончательный вариант таблицы и находим по нему ответ.

15	02	20	45	42	15	26	67	44	30	56	82	53	33	62	32	73	63	92	49	75	70	98	49
10	**09**	**19**	**48**	**29**	**04**	**19**	**54**	**25**	**09**	**19**	**49**	**12**	**06**	**19**	**71**	**24**	**06**	**19**	**70**	**04**	**07**	**19**	**52**
05	03	11	07	23	19	17	13	29	31	47	43	41	37	53	61	59	67	83	79	71	73	89	97

Ответ: 10.09.1948
29.04.1954
25.09.1949
12.06.1971
24.06.1970
04.07.1952

№ 4

Очевидно, что число 1999 является лишь напоминанием участникам олимпиады о наступлении нового 1999 года, а все дело лишь в нечетности числа попаданий Криптоши в каждую клетку.

На изменение цвета любой выбранной клетки не влияют клетки, расположенные в одном ряду с ней по вертикали и по горизонтали, так как их всего $2 \cdot 12 = 24$ — четное число. Кроме того, на изменение цвета выбранной клетки не влияют клетки, расположенные по диагонали от нее, так как расстояние между их центрами не натуральное число (а число вида $k\sqrt{2}$).

Исходя из этого и соображений симметрии, получаем, что клетки на осях симметрии квадрата своего цвета не изменят (отмечены "0" на *рис.* 4), а изменение цвета в клетках выделенного "треугольника" будет симметрично остальным 7 "треугольникам".

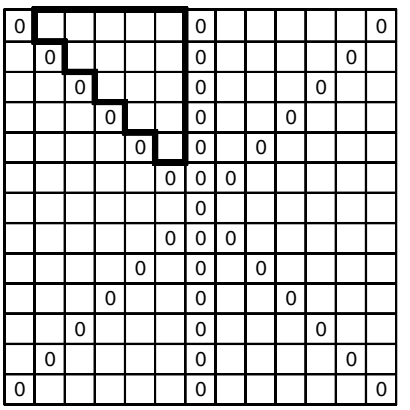


Рис. 4

Следующим шагом является выявление прямоугольных треугольников, стороны которых — натуральные числа, не превосходящие 12. Их всего четыре: с катета-

ми (3, 4), (6, 8), (5, 12), (9, 12). Определим, изменился или нет цвет каждой клетки выделенного "треугольника". В верхнюю клетку последнего столбца выделенного "треугольника" впишем цифру 1. Считая центр клетки за вершину острого угла прямоугольных треугольников с катетами (3, 4), (6, 8), (5, 12), (9, 12), отметим цифрой же 1 другую вершину острого угла. Таких клеток будет 7 — нечетное число, следовательно, цвет верхней клетки изменится.

Параллельным переносом получаем расстановку цифр 2, 3, 4, 5 для данных столбца, при этом появляются дополнительные цифры 4 и 5 в верхней части квадрата (*рис.* 5).

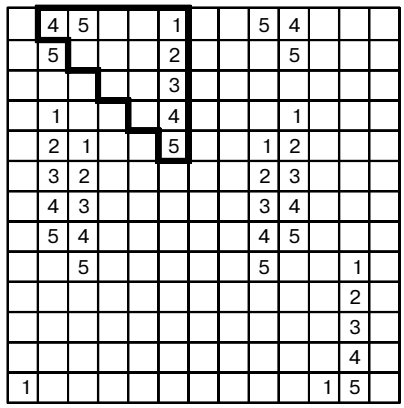


Рис. 5

Так же, как и для верхней клетки, делаем вывод о том, что цвет клеток последнего столбца выделенного "треугольника" изменится. Аналогичные рассуждения проводим для других столбцов выделенного "треугольника" (там некоторые клетки свой цвет не изменят). Учитывая симметрию на *рис.* 6, заштрихованы клетки, изменившие свой цвет. Окончательный ответ на *рис.* 7.

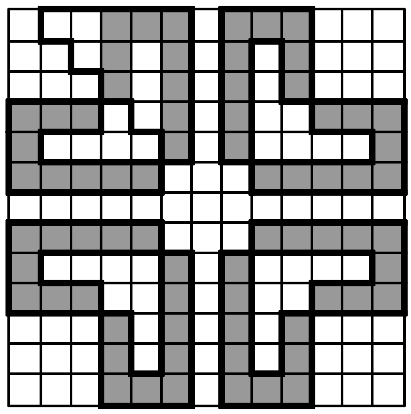


Рис. 6

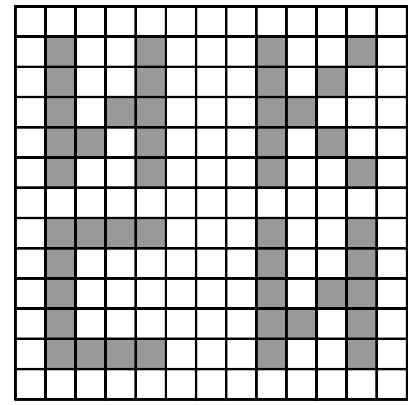


Рис. 7

№ 5

Если попытаться решать это уравнение, то видно, что придется рассматривать случаи $a \neq 0$, $b \neq 0$, $a - b \neq 0$, $a + b \neq 0$. Поэтому целесообразно выделить их сразу.

1. Пусть $a = 0$, $b = 0$.

Уравнение имеет вид $\frac{0}{1 - 0 \cdot x} = \frac{0}{1 - 0 \cdot x}$, x — любое число.

2. Пусть $a = 0$, $b \neq 0$.

Уравнение имеет вид $\frac{0}{1 - bx} = \frac{b}{1 - 0 \cdot x}$ или $0 = b$. Нет решений.

3. Аналогично, при $a \neq 0$, $b = 0$ нет решений.

4. Пусть $a \neq 0$, $b \neq 0$. Здесь рассмотрим три случая.

а) $a = b$

$$\frac{a}{1 - ax} = \frac{a}{1 - ax} \quad x \neq \frac{1}{a}$$

x — любое, кроме $\frac{1}{a}$.

б) $a = -b$

$$\frac{a}{1 + ax} = \frac{-a}{1 - ax} \quad x \neq \pm\frac{1}{a}$$

$$\frac{1}{a} + x = -\frac{1}{a} + x$$

$\frac{2}{a} = 0$ нет решений

в) $a \neq \pm b$

$$\frac{a}{1-bx} = \frac{b}{1-ax} \qquad x \neq \frac{1}{a} \quad x \neq \frac{1}{b}$$

$$\frac{1}{a} - \frac{b}{a}x = \frac{1}{b} - \frac{a}{b}x$$

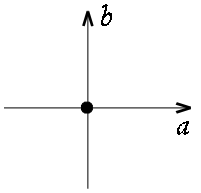
$$x\left(\frac{a}{b} - \frac{b}{a}\right) = \frac{1}{b} - \frac{1}{a}$$

$$x = \frac{(a-b)ab}{ab(a^2-b^2)} = \frac{1}{a+b}.$$

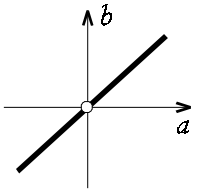
Ответ:

Ответ удобно изобразить графически на плоскости параметров a, b.

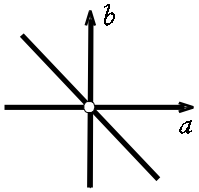
При $a = b = 0$ $\forall\ x \in R$



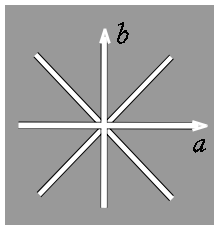
При $a = b \neq 0$ $x \in \left(-\infty; \frac{1}{a}\right) \cup \left(\frac{1}{a}; \infty\right)$



При $a = 0$, $b \neq 0$,
либо $a \neq 0$, $b = 0$,
либо $a = -b \neq 0$ нет решений



При $\begin{cases} a \neq b \\ a \neq -b \\ a \neq 0 \\ b \neq 0 \end{cases}$ $x = \frac{1}{a+b}$



№ 6

Выражение $2^{30} + 1$ можно представить как сумму кубов, сумму пятых степеней и можно выделить полный квадрат.

$$2^{30} + 1 = (2^{10} + 1) \cdot (2^{20} - 2^{10} + 1) =$$
$$= 41 \cdot 25 \cdot (2^{20} - 2^{10} + 1)$$
$$2^{30} + 1 = (2^6 + 1) \cdot (2^{6\cdot 4} - 2^{6\cdot 3} + 2^{6\cdot 2} - 2^6 + 1) =$$
$$= 13 \cdot 5 \cdot (2^{24} - 2^{18} + 2^{12} - 2^6 + 1) \qquad (*)$$
$$2^{30} + 1 = (2^{15} + 1)^2 - 2 \cdot 2^{15} =$$
$$= (2^{15} + 2^8 + 1) \cdot (2^{15} - 2^8 + 1) =$$
$$= 33025 \cdot 32513 = 25 \cdot 1321 \cdot 32513$$

Докажем, что 1321 — простое число. Для этого достаточно показать, что 1321 не делится ни на одно простое число, меньшее, чем 37 ($37^2 = 1369$). Это легко сделать таким приемом. Например,

$1321 = 1300 + 21 = 13 \cdot 100 + 21$ не делится на 13 и т.д.

Из соотношений $(*)$ видно, что делителями числа $2^{30} + 1$ являются 13 и 41. Непосредственной проверкой получаем

$32513 = 41 \cdot 793 = 41 \cdot 13 \cdot 61$.

Ответ:

$2^{30} + 1 = 5 \cdot 5 \cdot 13 \cdot 41 \cdot 61 \cdot 1321$

Задачи 11-й Кировской областной олимпиады школьников по информатике

С.М. ОКУЛОВ

Ребус

Написать программу поиска всех решений ребуса VOLVO+FIAT=MOTOR. Как обычно, разным буквам соответствуют разные цифры, одинаковым — одинаковые. Первая цифра числа не ноль.

Выходные данные

В каждой строке файла OUTPUT.TXT записывается одно решение ребуса — три числа через пробел(VOLVO FIAT MOTOR).

Решение

Задача носит утешительный характер. Семь циклов типа For и проверка ряда условий — вот и все решение.

```
Program Rebus;
Const S:Array[1..5]Of
Longint=(10,100,1000,10000,100000);
Var  A1,A2,A3:Longint;
V,O,L,F,I,A,T:Byte;
Ss:Set Of 0..9;
Procedure Prov;
Var  B:Longint;Pr:Byte;
 Begin
  B:=A3;Pr:=B Div S[4];
  If Pr in Ss Then Exit;
  B:=B Mod S[4];Pr:=B Div S[3];
  If Pr<>O Then Exit;
  B:=B Mod S[3];Pr:=B Div S[2];
  If Pr<>T Then Exit;
  B:=B Mod S[2];Pr:=B Div S[1];
  If Pr<>O Then Exit;
  Pr:=B Mod S[1];
  If Pr in Ss Then Exit;
  WriteLn(A1,' ',A2,' ',A3);
 End;
Begin
 Assign(Output,'Output.txt');
Rewrite(Output);
 For V:=1 To 9 Do Begin Ss:=[v];
  For O:=0 To 9 Do
   If Not(O in Ss) Then Begin Ss:=Ss+[O];
    For L:=0 To 9 Do
     If Not(L in Ss) Then Begin Ss:=Ss+[L];
      For F:=1 To 9 Do
       If Not(F in Ss) Then Begin Ss:=Ss+[F];
        For I:=0 To 9 Do
         If Not(I in Ss) Then Begin Ss:=Ss+[I];
           For A:=0 To 9 Do
          If Not(A in Ss) Then Begin Ss:=Ss+[A];
           For T:=0 To 9 Do
          If Not(T in Ss) Then Begin Ss:=Ss+[T];
         A1:=V*S[4]+O*S[3]+L*S[2]+V*S[1]+O;
        A2:=F*S[3]+I*S[2]+A*S[1]+T;A3:=A1+A2;
            If A3 Div S[5]=0 Then Prov;
            Ss:=Ss-[T];
           End;
          Ss:=Ss-[A];
          End;
        Ss:=Ss-[I];
        End;
       Ss:=Ss-[F];
End;
     Ss:=Ss-[L];
     End;
   Ss:=Ss-[O];
   End;
 End;
 Close(Output);
End.
```

Ответы

```
15615  9743  25358
15715  9643  25358
36736  9825  46561
36836  9725  46561
46346  9821  56167
46846  9321  56167
71571  9642  81213
71671  9542  81213
72472  9651  82123
72672  9451  82123
```

Делители

Написать программу для определения возможности разложения положительных чисел A и B на N положительных делителей $a_1, a_2, \ldots, a_N, b_1, b_2, \ldots, b_N$ таких, что произведения этих делителей совпадают, т.е. выполняются следующие условия: $a_1 \cdot a_2 \cdot \ldots \cdot a_N = A$, $b_1 \cdot b_2 \cdot \ldots \cdot b_N = B$ и $a_1 \cdot b_1 = a_2 \cdot b_2 = \ldots = a_N \cdot b_N$,

— где $a_1, a_2, \ldots, a_N$ — делители числа A,

$b_1, b_2, \ldots, b_N$ — делители числа B.

Число 1 считать делителем.

Входные данные

В файле INPUT.TXT через пробел записаны числа A, B и N. Произведение чисел A и B не превышает значения 2147483647, а значение N — 31.

Выходные данные

В первую строку файла OUTPUT.TXT через пробел записываются числа $a_1\ a_2 \ldots a_N$, а во вторую — $b_1\ b_2 \ldots b_N$ (достаточно вывести одно решение). Если решение отсутствует, то в выходной файл записывается сообщение NO.

Пример

Входные данные:	4	16	2
Выходные данные:	2	2	
	4	4	

Идея решения. Обозначим попарные произведения делителей чисел A и B через h. Пусть мы нашли число j такое, что h и A делятся нацело на j, а B нацело делится на h *div* j. В этом случае A следует разделить на j, а B — на h *div* j. После этого процесс поиска делителей следует продолжить. Если мы найдем N таких чисел j, то задача решена.

Данные программы

```
Const InputFile='Input.TxT';
OutputFile='Output.TxT';
MaxN=32;
Var  Res: Boolean;{рабочая переменная}
A, B: LongInt;{исходные данные}
N:Integer;
RsA, RsB: Array[1..MaxN] Of LongInt;
{массивы для хранения разбиения чисел}
h: LongInt;{рабочая переменная}
```

Пример

Если A равно 320, B равно 2278125 и N равно 6, то после вычислений получим h равное 30, а в массивах RsA, RsB будет записано соответственно (2, 2, 2, 2, 2, 10, 0,…) и (15, 15, 15, 15, 15, 3, 0, …).

Основная программа, процедуры ввода и вывода очевидны. Приведем ключевые фрагменты программы.

```
Procedure Done;
 Var i: Integer;
 Begin
  Assign(Output, OutputFile);
  Rewrite(Output);
  If Res Then Begin
{если решение найдено, то выводим результат}
   For i:=1 To N Do Write(RsA[i], ' ');
   WriteLn;
   For i:=1 To N Do Write(RsB[i], ' ');
   WriteLn;
   End
  Else WriteLn('NO');
  Close(Output);
 End;
Procedure Solve;
 Var z: Real;
     i:Integer;
 Begin
  Res:=False;
  z:=Exp(ln((A*1.0)*B)/N);
  h:=Round(z);
  {вычисляем произведение делителей}
  If (A*1.0)*B>MaxLongInt Then Exit;
  For i:=1 To N Do
  {нам необходимо найти N делителей}
   If Not CanSet(i) Then Exit;
   {если очередной делитель не найден,
    то обработка завершена}
  Res:=True;{разложение чисел на N
            делителей с одинаковым
            произведением найдено}
 End;
```

Функция поиска очередных делителей.

```
Function CanSet(Const k: Integer): Boolean;
 Var j: LongInt;
 Begin
  j:=1;
  CanSet:=False;
  While (j<=A) And (j<=h) Do Begin
  {делитель не превосходит значений A и h}
   If (h Mod j=0) And (A Mod j=0) And
   (B Mod (h Div j)=0) Then Begin
  {условие того, что j является делителем A
   и h Div j - делителем B выполнено}
  A:=A Div j; B:=B Div (h Div j);
  {изменяем значения A и B}
    RsA[k]:=j; RsB[k]:=h Div j;
   {запоминаем делители}
    CanSet:=True;
    Exit;
   End;
   Inc(j);
  End;
 End;
```

Троллейбусы

Троллейбусы одного маршрута проходят через остановку каждые k ($1 \le k \le 500$) минут. Известно время прихода каждого из N ($1 \le N \le 100000$) жителей Кирова на остановку (N, k — целые числа). Если человек приходит на остановку в момент прихода троллейбуса, то он также успевает войти в этот троллейбус. Необходимо написать программу определения времени прибытия первого троллейбуса на остановку (это число от 0 до k—1) такого, чтобы:

- суммарное время ожидания троллейбуса для всех граждан было минимально;
- максимальное время ожидания троллейбуса было минимально.

Входные данные

В первой строке файла INPUT.TXT записано число k, во второй — N, а затем N строк, в которых указано время прихода каждого человека на остановку (числа от 0 до 100 000).

Выходные данные

В файл OUTPUT.TXT записывается два числа, каждое в отдельной строке, являющиеся ответами на первый и второй вопросы задачи соответственно. Если вариантов решения задачи несколько, то достаточно выдать один из них. При решении только одного пункта задачи строка, соответствующая другому пункту, остается пустой.

Пример

Входные данные

```
100
5
0
210
99
551
99
```

Выходные данные

```
10
51
```

Идея решения. Достаточно подсчитать время прихода каждого из жителей по модулю k. После этого в каждый момент времени от 0 до k—1 имеем количество пришедших жителей. Для решения задачи достаточно проверить моменты прихода первого троллейбуса от 0 до k—1 и выбрать из них наилучший для первого и для второго пунктов.

Данные

```
ConstInputFile='Input.TxT';
OutputFile='Output.TxT';
MaxK=500;
{максимальное значение интервала времени}
Var  k, N: LongInt;
A: Array[0..MaxK-1] Of LongInt;
{время прихода каждого жителя считается
по модулю k}
ResA, ResB: LongInt;
{переменные для хранения результатов}
```

Процедуры ввода, вывода и основная программа.

```
Procedure Init;
  Var i, p: LongInt;
  Begin
   FillChar(A, SizeOf(A), 0);
   ResA:=0; ResB:=0;
   Assign(Input, InputFile);
   Reset(Input);Read(k, N);
   For i:=1 To N Do Begin
     Read(p);
     Inc(A[p Mod k]);
  End;
   Close(Input);
  End;

Procedure Done;
  Begin
    Assign(Output, OutputFile);
    Rewrite(Output);
    WriteLn(ResA);
    WriteLn(ResB);
    Close(Output);
  End;

Begin
 Init;
 SolvePA;
 SolvePB;
 Done;
End.
```

Процедура решения первого пункта задачи.

```
Procedure SolvePA;
 Var i, j: LongInt;
     sc, Min: LongInt;
 Begin
  Min:=MaxLongInt;
  For i:=0 To k-1 Do Begin
   {время прихода первого троллейбуса}
   sc:=0;{находим сумму времен ожидания
          троллейбусов для всех жителей}
   For j:=0 To k-1 Do
       sc:=sc + ((k - j + i) Mod k) * A[j];
```

```
    If sc<Min Then Begin
         ResA:=i;Min:=sc;End;{сравниваем}
                  End;
End;
```

Процедура решения второго пункта задачи.

```
Procedure SolvePB;
 Var i, j: LongInt;
     sc, Min: LongInt;
 Begin
  Min:=MaxLongInt;
  For i:=0 To k-1 Do Begin
    {время прихода первого троллейбуса}
   sc:=-MaxLongInt;
   For j:=0 To k-1 Do
    If (A[j]<>0) And (sc<(k-j+i) Mod k)
    Then sc:=(k-j+i) Mod k;
    {время ожидания равно (k-j+i) Mod k}
     If sc=-MaxLongInt Then sc:=MaxLongInt;
     {существенное замечание - максимальное
      время ожидания для данного значения i
      может не измениться}
   If sc<Min Then Begin ResB:=i;Min:=sc;End;
  End;
End;
```

Посылки

Весь воскресный день родители решили посвятить отправке посылок голодающему в Москве сыну-студенту. Для этого родители привозят посылки на вокзал и отправляют их на любом идущем в Москву поезде. Их сын в Москве приезжает к приходу этого поезда на вокзал, получает посылку и отвозит ее в общежитие. При этом родители не могут сразу принести на вокзал более одной посылки (очень тяжело!). Студент также не может носить по нескольку посылок сразу (он голодает!). Время, необходимое родителям, чтобы добраться от дома до вокзала, равно T (оно равно времени переезда от вокзала до дома). В Москве, как известно, поезда приходят на разные вокзалы. Для каждого вокзала известно время, необходимое студенту на дорогу от общежития до этого вокзала (оно равно времени на дорогу от этого вокзала до общежития). Дано расписание движения поездов из Кирова в Москву. Поезда в пути не обгоняют друг друга (т.е. поезд, который раньше отправляется из Кирова, раньше прибывает в Москву). Написать программу определения максимального количества посылок, которые родители смогут послать за воскресный день голодающему студенту (он все их должен встретить и отвезти в общежитие).

Входные данные

В файле INPUT.TXT заданы в следующем порядке целые числа:

T — время, которое затрачивают родители на дорогу от дома до вокзала в Кирове;

M — количество вокзалов в Москве ($0<M<20$);

M строк, в каждой из которых записано одно число — время переезда от соответствующего вокзала до общежития;

N — количество поездов ($0<N<1000$);

N строк, каждая из которых содержит 3 числа: время отправления поезда из Кирова, время в пути, номер вокзала в Москве, на который прибывает данный поезд.

Примечание. Список поездов дан в отсортированном по времени их отправления порядке. Время в задаче — число из интервала от 0 до 5000. В одну единицу времени отправляется не более одного поезда. Все данные — целые числа.

Выходные данные

В файле OUTPUT.TXT необходимо вывести одно число — максимальное количество посылок, которые родители смогут отправить за воскресенье.

Пример

Входные данные

```
70
2
40
50
4
60    1200  2
200   1120  1
1200  1440  2
1320  1440  2
```

Выходные данные

```
2
```

Решение

Предположим, что родители отправили посылку с первым поездом. Принимаем решение по второму поезду. Возможны два варианта. Если они успевают съездить за второй посылкой, то отправляют ее с поездом и количество отправленных посылок к этому моменту времени будет равно двум, а если нет, то первую посылку они могут отправить не с первым, а со вторым поездом.

Пусть для каждого поезда i ($1 \le i \le k-1$) найдено максимальное количество посылок a_i, которое можно отправить с поездами от 1 до i. Найдем максимальное количество посылок, которое можно отправить с поездом k. Во-первых, ясно, что уж одну-то посылку (первую) с поездом k всегда можно отправить. Во-вторых, можно увидеть, что $a_k = \max(1, ?\ (a_1+1), ?\ (a_2+1), ..., ?\ (a_i+1), ..., ?\ (a_{k-1}+1))$. Здесь всюду стоят знаки вопроса, которые означают, что необходимо проверить, успевают ли “обернуться” с очередной посылкой родители и студент между i-м и k-м поездами.

Запишем соответствующий алгоритм.

Структуры данных

```
Const InputFile='Input.TxT';
OutputFile='Output.TxT';
MaxM=20;
MaxN=1000;
Var  T, M, N: LongInt;
Train: Array[1..MaxN] Of Record St, HowGo,
WhCome: LongInt; End;
{для каждого поезда: время отправления
 поезда из Кирова, время в пути, номер
 вокзала в Москве}
Tm: Array[1..MaxM] Of LongInt;
{время проезда от общежития до вокзалов}
Res: Array[1..MaxN] Of LongInt;
{массив для хранения промежуточных данных}
ResP: LongInt;
{результат решения задачи}
```

Процедуры ввода, вывода и основная программа.

```
Procedure Init;
 Var i: Integer;
 Begin
  FillChar(Res,SizeOf(Res),0);
  FillChar(Train,SizeOf(Train),0);
  FillChar(Tm,SizeOf(Tm),0);
  Assign(Input, InputFile);Reset(Input);
  Read(T);Read(M);
  For i:=1 To M Do Read(Tm[i]);
  Read(N);
  For i:=1 To N Do
  Read(Train[i].St, Train[i].HowGo, Train[i].WhCome);
  Close(Input);
 End;

Procedure Done;
 Begin
  Assign(Output, OutputFile);
  Rewrite(Output);WriteLn(ResP);
  Close(Output);
 End;

Begin
 Init;
 Solve;
 Done;
End.
```

Напишем основную процедуру, отложив реализацию ключевой проверки на потом.

```
Function Max(Const a, b: LongInt): LongInt;
 Begin
  If a>b Then Max:=a Else Max:=b;
 End;

Procedure Solve;
 Var i, j: Integer;
 Begin
  ResP:=0;
  For i:=1 To N Do Begin
   {решаем задачу для поезда с номером i}
   Res[i]:=1;
   {мы всегда можем отправить первую посылку
    с этим поездом, "проспав" все предыдущие}
   For j:=1 To i-1 Do
    {согласуем отправку посылки с поездом
     i с отправками посылок на предыдущих
     поездах}
  If IfPossib(j,i) Then Res[i]:=Max(Res[j]+1,Res[i]);
    {если между поездами с номерами i и j
     родители успевают съездить домой,
     а студент — в общежитие, то сравниваем
     Res[j]+1 и Res[i]}
   ResP:=Max(Res[i], ResP);
  End
 End;
```

Ключевая проверка реализована в функции IfPossib.

```
 Function IfPossib(Const fp, tp:Integer):
Boolean;
  Begin
   IfPossib:=((Train[tp].St-Train[fp].St)>=(2*T))
    {родители успевают съездить домой между
     этими поездами}And
    ((Train[tp].St + Train[tp].HowGo-
      Train[fp].St - Train[fp].HowGo)
      {время в Москве между приходами
       поездов} >= (Tm[Train[tp].WhCome] +
       Tm[Train[fp].WhCome]));
       {время, необходимое студенту для
        поездки в общежитие от вокзалов,
        на которые прибывают поезда}
  End;
```

Журнал “КОМПЬЮТЕРНЫЕ ИНСТРУМЕНТЫ В ОБРАЗОВАНИИ”
совместно с САНКТ-ПЕТЕРБУРГСКИМ ГОСУДАРСТВЕННЫМ УНИВЕРСИТЕТОМ
начинает прием школьников в

ЗАОЧНУЮ ШКОЛУ СОВРЕМЕННОГО ПРОГРАММИРОВАНИЯ

С 1999 года в каждом номере журнала будут публиковаться статьи по проблемам современной информатики и предлагаться задачи олимпиадного характера. Особенность этих задач в том, что они конструктивны и доступны школьникам всех возрастов. Присланные решения будут оцениваться по указанным в формулировках задач критериям. Решения задач будут разбираться на страницах журнала. В школу принимаются учащихся 5—11-х классов.

Задачи имеют несколько уровней:

1. Первый уровень рассчитан на тех, кто не изучает алгоритмические языки;
решение этих заданий записывается в виде текстового файла в формате ASCII (точная форма записи ответа указывается в условии задачи).

2. Второй уровень предназначен для умеющих программировать;
решения заданий второго уровня представляются в виде исполнимых exe-файлов (точный формат входных и выходных данных указывается в условии задачи).

Решения задач принимаются на дискетах 3.5” (1.44 Мб) или по электронной почте.

К решению могут прилагаться документы (в формате ASCII или на бумаге), содержащие исходный текст программы (на языках Паскаль или Си/Си++), а также комментарии, поясняющие основные идеи и алгоритмы.

Стоимость обучения в школе — 80 руб. в год, для подписчиков журнала — бесплатно.

Для поступления в школу нужно прислать заявку **по адресу**:
191025, Санкт-Петербург, ул. Марата, д. 25
E-mail: pozdnkov@aec.neva.ru
Журнал “Компьютерные инструменты в образовании”
Школа современного программирования.
Телефон для справок: (812) 164-13-55.

Форма заявки

Прошу принять меня в Школу современного программирования.

О себе сообщаю следующее:

1. [Фамилия, имя, отчество. Возраст.]
2. [Место учебы: название и адрес учебного заведения, класс.]
3. [Фамилии, имена и отчества преподавателей математики и информатики.]
4. [Домашний адрес, телефон, адрес электронной почты (если есть).]
5. [Являюсь/не являюсь подписчиком журнала.]
6. [Личная подпись.]

Гл. редактор
Е.Б.Докшицкая
Зам. гл. редактора
С.Л.Островский
Редакция:
Л.Н.Картвелишвили,
Ю.А.Соколинский,
Н.Л.Беленькая,
Н.П.Медведева
Дизайн и компьютерная верстка:
Н.И.Пронская
Корректоры:
Е.Л.Володина,
С.М.Подберезина

©ИНФОРМАТИКА 1999
выходит четыре раза в месяц
При перепечатке ссылка на ИНФОРМАТИКУ обязательна, рукописи не возвращаются

121165, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

Internet: inf@1september.ru
Fidonet: 2:5020/69.32
WWW: http://www.1september.ru

Объединение педагогических изданий "ПЕРВОЕ СЕНТЯБРЯ"
Отпечатано в типографии “Пресса”
125865, Москва, ул. Правды, 24
Регистрационный номер 012868
Тираж 7000 экз.
Заказ №

Windows95
Start
тел./факс (095)249 3138
факс (095)249 3184
тел. (095)249 3386
24.03