Министерство образования
Московской области
Центр новых педагогических технологий
Московской области

Задачи районного и областного этапов XIX Всероссийской олимпиады по информатике в Московской области

2006 – 2007 учебный год

Под редакцией Чернова Ю.Г. и Шестимерова А.А.

Троицк
2007г.

В книге содержатся задачи районной и областной олимпиады по информатике для школьников Московской области, проходивших в 2006/07 учебном году. Все задачи снабжены подробными решениями. Кроме того, к большинству задач, при необходимости, приводятся полные тексты правильных программ.

Книга предназначена для школьников, учителей информатики, руководителей кружков и факультативов. Книга рекомендуется для подготовки к олимпиадам по информатике начальных уровней.

Задачи районного и областного этапов XIX Всероссийской олимпиады по информатике в Московской области

Под редакцией
Чернова Ю.Г. и Шестимерова А.А.

Троицк
2007 г.

Авторы задач и текстов решений — члены жюри и научного комитета Московской областной олимпиады по информатике:
Алексеев А.А., Попелышев И.И., Пучин С.А., Чернов Ю.Г., Шедов С.В., Шестимеров А.А.

Верстка Касабова М.Г.

Сдано в набор чч.чч.07. Подписано к печати чч.чч.06. Формат 60x84/16. Гарнитура “Таймс”. Печать офсетная. Тираж ччч экз. ЛР №071961 от 01.09.1999. Заказ № ччччч/ч

ЦНПТ, 142190, Московская обл., г. Троицк, Сиреневый б-р., 11.

Отпечатано с готового оригинал-макета в типографии издательства «Тровант», 142190, Московская обл. Троицк, ччччч.

ISBN

Введение

Олимпиады по информатике в последнее время пользуются все большей и большей популярностью. С целью оказания помощи учащимся при подготовке к олимпиадам по информатике ЦНПТ выпускает сборники задач и их решений.

Московская областная олимпиада является 3 этапом Всероссийской олимпиады по информатике. В этом году в ней приняли участие 91 школьника. По результатам олимпиады и учебно-тренировчных сборов была сформирована команда из 16 человек на Федеральный окружной этап. Команда Московской области завоевала 12 дипломов из 25 разыгрываемых призовых мест среди 16 областей. На XIX Всероссийской олимпиаде по информатике Московская область получила 6 дипломов.

В настоящем сборнике приводятся материалы XIX Московской областной олимпиады школьников по информатике (2006 – 2007 учебный год).

В первой части сборника содержатся пять задач районного тура. Каждая задача имеет подробное решение, а при необходимости — и программу.

Во второй части сборника содержатся все семь задач областного тура олимпиады. К каждой задаче приводится комментарий по ее решению, а также, при необходимости, полные листинги правильных программ.

В конце книги приводится списокк литературы, полезной для успешной подготовки.

Все замечания и предложения по содержанию сборника авторы просят направлять по адресу: olymp@informatics.ru. По этому же адресу можно направлять вопросы, касающиеся олимпиад по информатике в Московской области.

Условия задач и результаты Московских областных олимпиад по информатике можно найти на сайте www.bytic.ru и www.informatics.ru. Там же можно найти тесты и решения жюри, а так же, проверить свои решения.

В подготовке книги принимали участие члены жюри и научного комитета олимпиады. Мы благодарим за внимательное прочтение и критические замечания И. Кирова и В. Семипятного.

Часть I.
РАЙОННЫЙ ЭТАП ОЛИМПИАДЫ

Районный тур проходил во всех районах Московской области 11 ноября 2006 года. Продолжительность олимпиады составляла 4 астрономических часа.

Условия задач

Максимальное время работы решения задач на каждом тесте 2 секунды

Задача P-1. Календарь

Максимальная оценка за задачу: 10 баллов

Дни недели пронумерованы следующим образом: 1 — понедельник, 2 — вторник, ... , 6 — суббота, 7 — воскресенье. Дано целое число ***K***, лежащее в диапазоне 1–365, и целое число ***N***, лежащее в диапазоне 1–7.

Написать программу, которая определяет номер дня недели для ***K***-го дня года, если известно, что в этом году 1 января было днем недели с номером ***N***.

Формат входных данных

С клавиатуры вводится два числа через пробел — ***K*** ($0 < K < 366$) и ***N*** ($0 < N < 8$).

Формат выходных данных

Вывести на экран одно число – номер дня недели для ***K***-го дня года.

Примеры входных и выходных данных

Ввод	*Вывод*
5 4	1

Задача P-2. Круглые числа

Максимальная оценка за задачу: 15 баллов

Будем называть числа круглыми, если они содержат в своей записи только цифры 0 и 5.

Составим последовательность круглых чисел в порядке возрастания: 0, 5, 50, 55, 500, 505 и так далее.

Написать программу, которая находит ***K***-ое по порядку в этой последовательности круглое число.

Формат входных данных

С клавиатуры вводится натуральное число ***K*** – номер круглого числа в последовательности ($0 < K < 500$).

Формат выходных данных

Выведите на экран требуемое круглое число.

Примеры входных и выходных данных

Ввод	*Вывод*
2	5
6	505

Задача P-3. Морской бой

Максимальная оценка за задачу: 25 баллов

В известной игре «Морской бой» действие происходит на клетчатой бумаге размером ***M*** строк на ***N*** столбцов. На игровом поле могут располагаться различные корабли произвольной формы. При этом каждый корабль образует связную фигуру (фигура называется ***связной***, если из любой ее клетки можно добраться до любой другой, ходя только по клеткам фигуры и перемещаясь каждый раз в одну из 4 соседних по стороне клеток). Никакие два корабля не имеют общих точек, в том числе не касаются углами.

В процессе игры корабли могут принимать три состояния:

- если в корабль не было ни одного попадания, то он считается «живым»;
- если во все клетки, из которых состоит корабль, были попадания, то такой корабль считается «убитым»;
- в противном случае корабль считается «раненым».

Написать программу, которая по данному игровому полю определит количество «живых», «убитых» и «раненых» кораблей.

Формат входных данных

Ввод производится из файла **input.txt**

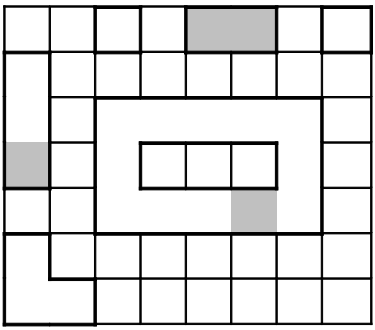
В первой строке входного файла вводятся количество строк и столбцов игрового поля через пробел — M и N ($0 < M, N < 50$). В последующих M строках содержится по N чисел в каждой. Эти числа могут принимать следующие значения:

- `0` – в этой клетке нет корабля (вода)
- `1` – это клетка корабля, попадания нет
- `-1` – это клетка корабля, в данную клетку было попадание

Формат выходных данных

Выведите на экран три числа через пробел – число «живых», «убитых» и «раненых» кораблей.

Пример входных и выходных данных



input.txt	*Вывод*
7 8 0 0 1 0 -1 -1 0 1 1 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 -1 0 1 0 0 0 1 0 0 0 1 1 1 -1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0	3 1 2

***Примечание**: будут отдельно оцениваться те случаи, когда корабли имеют только прямоугольную форму.*

Задача P-4. Остаток от деления

Максимальная оценка за задачу: 25 баллов

Написать программу, которая находит остаток от деления числа $2^{2^1} + 2^{2^2} + ... + 2^{2^{N-1}} + 2^{2^N}$ на 7.

Напомним, что $2N = \underbrace{2 * 2 * \ldots * 2}_{\text{N раз}}$, а $2^{2^N} = 2^{(2^N)}$

Формат входных данных

С клавиатуры вводится натуральное число $N < 100000$

Формат выходных данных

Вывести на экран одно число – остаток от деления.

Пример входных и выходных данных

Ввод	*Вывод*
2	6

Задача Р-5. Уравниловка

Максимальная оценка за задачу: 25 баллов

Дано N мешков с монетами. В первом мешке лежит одна монета. Во втором – две. И так далее, в мешке с номером K находится ровно K монет. За один шаг разрешается достать из любого количества мешков L строго одинаковое количество монет S из этих мешков (на каждом шаге числа L и S можно выбирать произвольными).

Написать программу, которая определяет, за какое минимальное количество шагов удастся уравнять количество монет в мешках?

Формат входных данных

С клавиатуры вводится число N – количество мешков ($0 < N < 10^9$).

Формат выходных данных

Вывести на экран одно число – минимальное количество шагов, за которое удастся добиться того, что в каждом мешке осталось ровно по одной монете

Пример входных и выходных данных

Ввод	*Вывод*
3	2

Тесты к задачам

Задача P-1. Календарь

№	Тест	Ответ	Баллы
1	6 1	6	2
2	100 3	4	4
3	181 7	5	4
		итого	10

Задача P-2. Круглые числа

№	Тест	Ответ	Баллы
1	8	555	3
2	99	5500050	3
3	302	500505505	3
4	359	505500550	3
5	497	555550000	3
		итого	15

Задача P-3. Морской бой

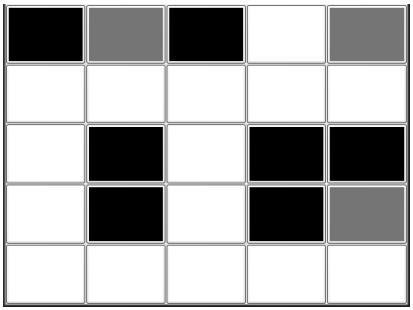
№	Тест	Ответ	Баллы
1	5 5 1 -1 1 0 -1 0 0 0 0 0 0 1 0 1 1 0 1 0 1 -1 0 0 0 0 0	1 1 2	3
2	10 10 0 1 -1 1 -1 1 -1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0	1 1 7	4

	0 1 1 0 1 1 0 1 -1 -1 0 1 1 0 -1 1 0 -1 -1 -1 0 1 1 0 0 0 0 0 0 0 0 1 -1 0 1 1 1 0 1 -1 0 0 0 0 1 1 1 0 1 1 -1 0 -1 0 -1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 1		
3	5 7 1 1 0 0 0 1 1 0 1 1 0 1 -1 0 0 0 -1 0 1 0 0 -1 0 0 0 0 0 1 -1 -1 0 1 0 -1 1	1 1 3	3
4	9 10 0 -1 0 0 0 0 0 0 1 0 -1 -1 -1 0 1 0 0 1 -1 1 0 1 -1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 -1 0 -1 0 1 1 1 1 0 -1 -1 0 1 1 1 0 0 0 0 -1 0 0 0 0 0	0 1 3	3
5	10 10 0 1 0 0 1 1 0 0 1 0 1 -1 1 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 -1 0 1 -1 1 0 0 1 0 1 1 0 1 0 1 0 -1 1 0 -1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 -1 0 -1 -1 -1 0 0 1 0 -1 1 -1 -1 0 -1 -1 0 1 0 0 -1	4 1 5	4
6	10 10 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1	0 0 1	4

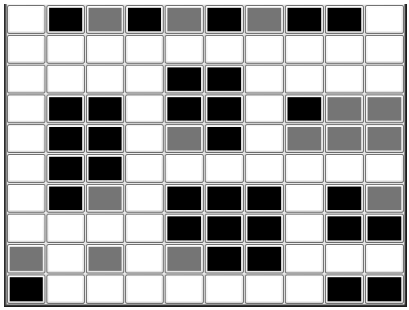
	1 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 1 0 1 0 -1 1 0 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1		
7	15 15 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 0 1 1 0 -1 0 0 0 0 0 0 0 0 -1 0 1 1 0 -1 0 1 1 1 1 1 1 1 0 -1 0 1 1 0 -1 0 1 0 0 0 0 0 1 0 -1 0 1 1 0 -1 0 1 0 -1 -1 -1 0 1 0 -1 0 1 1 0 -1 0 1 0 -1 0 -1 0 1 0 -1 0 1 1 0 -1 0 1 0 -1 -1 -1 0 1 0 -1 0 1 1 0 -1 0 1 0 0 0 0 0 1 0 -1 0 1 1 0 -1 0 1 1 1 1 1 1 -1 0 -1 0 1 1 0 -1 0 0 0 0 0 0 0 0 -1 0 1 1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	0 1 3	4
		итого	**25**

Примечание. *Ввод в данной задаче производится из файлов. Жюри должно один раз сделать файлы и использовать их для тестирования всех участников.*

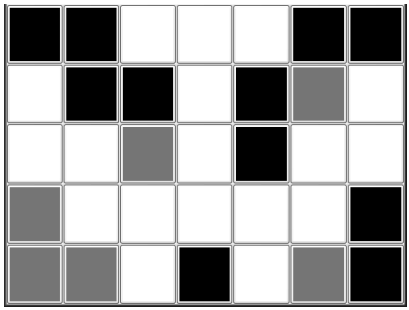
Визуализация тестов к задаче «Морской бой»



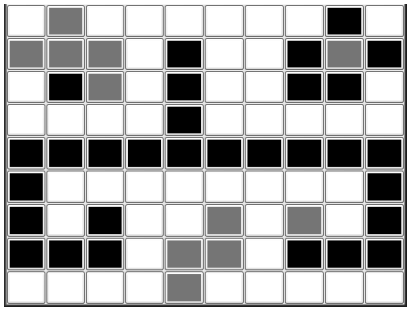
Тест № 1



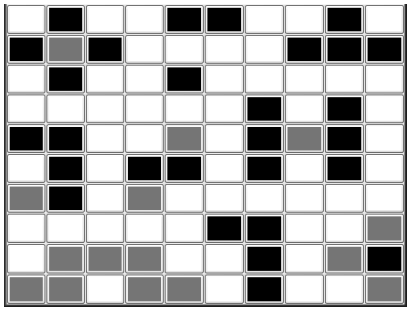
Тест № 2



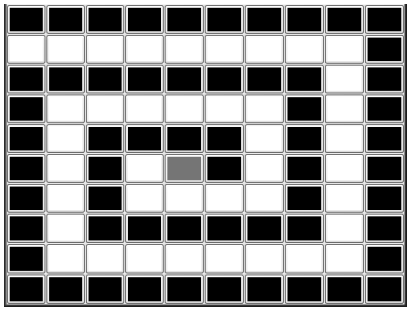
Тест № 3



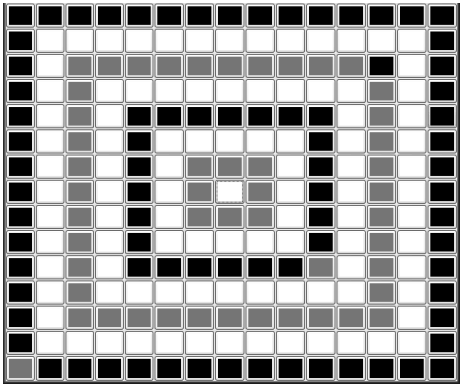
Тест № 4



Тест № 5



Тест № 6



Тест № 7

Задача P-4. Остаток от деления

№	Тест	Ответ	Баллы
1	3	3	4
2	4	5	4
3	10	2	4
4	11	6	4
5	99995	1	4
6	63482	4	5
		итого	25

Примечание. *Если программа участника выдает на все тесты один и тот же ответ — 0 баллов за все тесты. Жюри также имеет право ставить 0 баллов за программы вида* `writeln(random(7))`.

Задача Р-5. Уравниловка

№	Тест	Ответ	Баллы
1	1	0	3
2	2	1	3
3	3	2	3
4	15	4	4
5	346	9	4
6	57659	16	4
7	999999998	30	4
		итого	25

Примечание. *Если программа участника выдает на все тесты один и тот же ответ — 0 баллов за все тесты.*

Решения задач

Задача P-1. Календарь

Тема: разное, очень легкая.

Если первый день года – понедельник (***N***=1), то ***K***-ый день года будет ***M***-ым днем недели, где ***M***=(***K*** mod *7*) (если ***M***=0, то ***M***=7, т.е. – ***K***-ый день года — воскресенье). Сдвиг первого дня года со значения 1 до значения ***N*** (изменение на ***N***-1) приведет к изменению ***M*** на ту же величину ***N***–1, т.е.:

$$M=((K \bmod 7) + (N-1)) \bmod 7 = (K+N-1) \bmod 7$$

Что и требуется получить в задаче.

Задача P-2. Круглые числа

Тема: системы счисления, очень легкая.

Нетрудно заметить, что «круглые» числа – обычные двоичные числа, в которых цифра 1 заменена цифрой 5. ***K***-ое число в последовательности двоичных чисел отличается от «собственного» значения числа ***M*** на единицу, т.е. ***K***=***M***+1 (за счет первого нуля). Тогда программа должна выполнить следующее: по заданному ***K*** вычисляем ***M***=***K***-*1*, переводим ***M*** в двоичную систему счисления и заменяем единицы на пятерки. Отметим, что при заданных ограничениях результат помещается в 32-битный тип данных.

```
var
  res, st, m, k : longint;
begin
  read(k);
  m := k - 1;
  res := 0;
  st := 1;
  while (m <> 0) do begin
    res := res + 5*st*(m mod 2);
    st := st*10;
    m := m div 2;
  end;
  writeln(res);
end.
```

Задача Р-3. Морской бой

Тема: техника программирования, легкая

В задаче требуется выделить все фигуры, и для каждой найти некоторую характеристику (в данном случае число подбитых и не подбитых клеток). Существует несколько способов организации обхода игрового поля, но мы остоновимся на наиболее простом. Найдем какую-либо клетку, принадлежащую кораблю, и, используя её как начальную, запустим обход соседних клеток корабля. При обходе будем удалять просмотренные клетки, учитывая подбитые они или нет. Затем для каждой удаленной клетки так же рассматриваем соседнии. Обход клеток проще всего организовать с помощью рекурсии:

```
const
    // массив с направлениями соседних клеток
    dx : array [1..4] of integer = (-1,1,0 ,0);
    dy : array [1..4] of integer = (0 ,0,-1,1);
var
    // число подбитых и нет клеток области
    k1, k2 : integer;
procedure rec(x,y:integer);
  var
    i:integer;
  begin
   if (a[x,y] <> 0) then begin //клетка не
пуста
     // считаем тип клетки
     if (a[x,y]=1) then inc(k1) else inc(k2);
   a[x,y] := 0; //не просматриваем больше эту клетку
        for i:=1 to 4 do //рассматриваем 4
соседних
          rec(x+dx[i],y+dy[i]);
      end;
  end;
```

Основная логика программы выглядит так:

```
  const
    maxn = 50;
  var
    // размеры поля и ответ (целые, раненные,
убитые)
    m,n,t1,t2,t3 : integer;
    a : array [0..maxn+1, 0..maxn+1] of
integer; { размеры массива с каждой стороны на
1 больше, чтобы не проверять выход за границу
массива.}
    i,j : integer;
  begin
    assign(input,'input.txt');reset(input);
    read(n,m);
    for i:=1 to n do
      for j:=1 to m do read(a[i,j]);
    t1:=0;t2:=0;t3:=0;
    for i:=1 to n do begin
      for j:=1 to m do if a[i,j]<>0 then begin
        k1:=0;k2:=0;
        rec(i,j);
        if (k1=0) then inc(t1) else
          if (k2=0) then inc(t2) else
            inc(t3);
      end;
    end;
    write(t1,' ',t2,' ',t3);
    close(input);
  end.
```

Задача P-4. Остаток от деления

Тема: теория чисел, легкая

Заметим, что ограничения задачи таковы, что невозможно вычислить всю сумму и потом найти остаток от деления. Многие участники просто нашли закономерность в получаемых ответах. Мы же приведем конструктивное

решение. Рассмотрим остатки от деления степеней двойки на 7:

N	0	1	2	3	4	5
2^N	1	2	4	8	16	32
$2^N \bmod 7$	1	2	4	1	2	4

Остатки от деления на 7 зацикливаются, действительно из равенства (докажите его)

$$(ab \bmod c) = ((a \bmod c)(b \bmod c)) \bmod c$$

следует, что

$$2^N \bmod 7 = ((2^{N-1} \bmod 7)\cdot 2) \bmod 7$$

Каждый следующий остаток от деления зависит только от предыдущего. Так как число различных остатков равно 7 – они начнут повторяться, и возникнет цикл. В нашем случае он получился длины

$$3 — 2^N \bmod 7 = 2^{N+3} \bmod 7$$

Таким образом, исходная задача трансформируется в поиск остатка от деления на 7 выражения:

$$2^{(2^1) \bmod 3} + 2^{(2^2) \bmod 3} + \ldots + 2^{(2^{N-1}) \bmod 3} + 2^{(2^N) \bmod 3}$$

Для остатков делелия на 3 таким же образом получаем цикл длины два и выражение:

$$2^{(2^1 \bmod 2) \bmod 3} + 2^{(2^2 \bmod 2) \bmod 3} + \ldots + 2^{(2^{(N-1) \bmod 2}) \bmod 3} + 2^{(2^{N \bmod 2}) \bmod 3}$$

Это выражение уже можно вычислить в цикле, но заметив, что для четных номеров слагаемое равно 4, а для нечётных 2, получим ответ $(4N\text{-}2N \operatorname{div} 2) \bmod 7$

Задача P-5. Уравниловка

Тема: разное, легкая

Пусть на каком-то шаге у нас ***K*** мешков, из которых ***N*** различных. Понятно, что одинаковые мешки можно рассматривать как один – если на каком-то шаге из мешка

вынимают монеты, то столько же надо вынимать из равных ему.

Если есть ***K*** различных мешков, то на следующем шаге их не может быть меньше $\lceil K/2 \rceil$[1]. Действительно, если монеты достаются из ***L*** мешков, то число различных будет не меньше max(***K***-***L***,***L***). Таким образом, для исходного набора из ***N*** различных мешков от 1 до ***N*** монет, за один ход можно получить $\lceil N/2 \rceil$ различных (вытаскиваем монеты из всех мешков с номерами большими, чем ***N***/2). Эти ходы можно было промоделировать, но фактически ответ в задаче $\lceil \log N \rceil$

[1] $\lceil \ldots \rceil$ - означает округление в бóльшую сторону

Часть II
ОБЛАСТНОЙ ЭТАП ОЛИМПИАДЫ

На Московскую областную олимпиаду по информатике приглашались победители районных туров, а также школьники, показавшие наилучшие результаты в I Всероссийской заочной олимпиаде по информатике (www.olympiads.ru/zaoch).

Олимпиада проходила в два тура 11 и 18 февраля 2007 года. Продолжительность каждого тура 5 часов. Подробнее об олимпиаде можно узнать на сайтах www.informatics.ru и www.bytic.ru. Там же можно протестировоть свои решения.

Первый тур состоял из одной утешительной задачи, одной средней и сложной на теорию графов. В результате, три участника получили в первом туре полный балл. Во втором туре было уже 4 задачи — утешительная, на динамическое программирование, на реализацию и на теорию графов с элементами вычислительной геометрии.

Победителями олимпиады стали 11-классники:

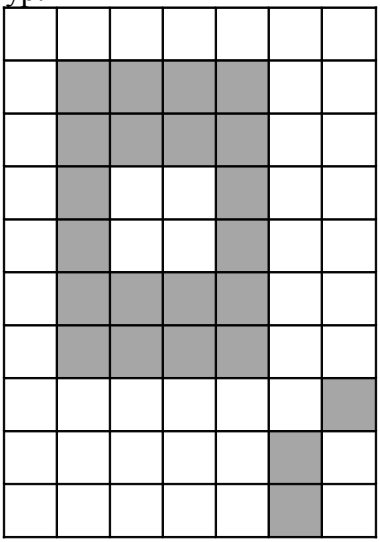
Семипятный Владислав	Мытищи	575 баллов
Киселев Павел	Раменское	518 баллов
Ющенко Павел	Краснознаменск	476 баллов

Условия задач
I ТУР

Задача О-1. Клетчатые фигуры

Имя входного файла:	figure.in
Имя выходного файла:	figure.out
Ограничения по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	100 баллов

На поле, состоящем из ***M*N*** белых квадратных клеток единичного размера, некоторые клетки покрасили в чёрный цвет, в результате чего образовалось одна или несколько закрашенных фигур.



Фигура называется ***связной***, если из любой ее клетки можно добраться до любой другой, ходя только по клеткам фигуры и перемещаясь каждый раз в одну из 4-х соседних по стороне клеток. Несвязные фигуры считаются различными. Например, на данном рисунке приведены 3 фигуры. Периметр фигуры — это сумма длин ее внешних и внутренних (при наличии) сторон. Периметр фигур,

изображенных на рисунке: 28, 6 и 4. Суммарный периметр фигур равен 38.

Написать программу, которая находит суммарный периметр фигур, получившихся на клетчатом поле.

Формат входных данных

Первая строка входного файла содержит два целых числа M и N ($0 < M, N \le 100$) — количество строк и столбцов, из которых состоит клетчатое поле. Во второй строке находится одно число K ($0 \le K \le M*N$) – количество клеток, закрашенных в черный цвет.

В последующих K строках содержатся координаты закрашенных клеток в формате:

<номер строки><пробел><номер столбца>.

Формат выходных данных

Выведите в выходной файл одно число — суммарный периметр всех фигур.

Пример

figure.in	**figure.out**
10 7 23 10 6 7 2 7 3 7 4 7 5 8 7 2 4 2 5 3 2 3 3 3 4 3 5 4 2	38

4 5 5 2 5 5 6 2 6 3 6 4 6 5 9 6 2 2 2 3	

Задача O-2. Алгоритм Евклида

Имя входного файла:	euclid.in
Имя выходного файла:	euclid.out
Ограничения по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	100 баллов

Андрей недавно начал изучать информатику. Одним из первых алгоритмов, который он изучил, был алгоритм Евклида для нахождения наибольшего общего делителя (НОД) двух чисел. Напомним, что наибольшим общим делителем двух чисел a и b называется наибольшее натуральное число x, такое, что и число a, и число b делится на него без остатка.

Алгоритм Евклида заключается в следующем:

1. Пусть a, b — числа, НОД которых надо найти.
2. Если $b = 0$, то число a — искомый НОД.
3. Если $b > a$, то необходимо поменять местами числа a и b.
4. Присвоить числу a значение $a - b$.
5. Вернуться к шагу 2.

Андрей достаточно быстро освоил алгоритм Евклида и вычислил с его помощью много наибольших общих делителей. Поняв, что надо дальше совершенствоваться, ему

пришла идея решить новую задачу. Пусть заданы числа ***a***, ***b***, ***c*** и ***d***. Требуется узнать, наступит ли в процессе реализации алгоритма Евклида для заданной пары чисел (***a***, ***b***) такой момент, когда перед исполнением шага 2 число ***a*** будет равно ***c***, а число ***b*** будет равно ***d***.

Написать программу, которая решает эту задачу.

Формат входных данных

Первая строка входного файла содержит количество наборов входных данных ***K*** ($1 \le K \le 100$). Далее идут описания этих наборов. Каждое описание состоит из двух строк. Первая из них содержит два целых числа: ***a***, ***b*** ($1 \le a, b \le 10^{18}$). Вторая строка – два целых числа: ***c***, ***d*** ($1 \le c, d \le 10^{18}$).

Все числа в строках разделены пробелом.

Формат выходных данных

Для каждого набора входных данных выведите в выходной файл слово «YES», если в процессе применения алгоритма Евклида к паре чисел (***a***, ***b***) в какой-то момент получается пара (***c***, ***d***). В противном случае выведите слово «NO».

Пример

euclid.in	**euclid.out**
2 20 10 10 10 10 7 2 4	YES NO

Задача О-3. Авторитетное сообщество

Имя входного файла:	meeting.in
Имя выходного файла:	meeting.out
Ограничения по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	100 баллов

На шахматный турнир в Нью-Васюках съехалось ***N*** игроков со всего света. Каждый игрок имеет свой шахматный рейтинг. Разумеется, на такой престижный турнир не допускались игроки с отрицательным рейтингом. В связи с разногласиями некоторых игроков по поводу регламента проведения матчей, после окончания турнира Председатель Шахматной Ассоциации решил собрать авторитетное сообщество шахматных игроков, для того чтобы внести изменения в регламент проведения будущих шахматных соревнований.

Авторитетность сообщества определяется суммарным рейтингом игроков, входящих в него. Но Председатель понимал, что нельзя приглашать на собрание всех игроков — иначе они увязнут в спорах, и никакого итогового решения принято не будет. Но чтобы соблюсти приличия, ему необходимо аргументировать свой выбор перед общественностью, а именно – это должно быть как можно более авторитетное (наибольшее) по рейтингу сообщество игроков. Кроме того, поскольку шахматисты — люди обидчивые, нельзя допустить и того, чтобы среди приглашенных игроков были проигравшие игроку, который приглашения не получил.

Написать программу, помогающую Председателю выбрать наиболее авторитетное сообщество, удовлетворяющее всем требованиям суровой шахматной политической жизни. Гарантируется, что такое сообщество всегда существует.

Формат входных данных

Первая строка входного файла содержит два целых числа: N ($0 < N \le 1000$) — число игроков, и M ($0 < M \le 10^6$) — число сыгранных на турнире партий. Следующие N строк содержат по одному целому неотрицательному числу A_i ($0 < A_i \le 10^6$) — рейтинг i-го игрока. Затем идет M строк с результатами партий (ничейные партии не приводятся, одни и те же игроки могли играть между собой несколько раз). Каждая строка состоит из номеров двух игроков через пробел: это значит, что в данной партии игрок, номер которого идет в строке первым, победил второго игрока. Все входные данные корректны.

Формат выходных данных

В первой строке выходного файла выведите количество игроков K ($K < N$) в наиболее авторитетном сообществе. В последующих K строках выведите номера игроков, входящих в это сообщество (в любом порядке, каждый игрок должен быть указан ровно один раз).

Примеры

meeting.in	meeting.out
4 3 6 3 7 2 1 2 2 1 4 3	3 1 2 4

meeting.in	meeting.out
4 3 6 3 10 2 1 2 2 1 4 3	2 4 3

Система оценки

Решения для ограничения $0 < M \leq 1000$ будут оцениваться до 70 баллов.

Условия задач
II ТУР

Задача О-4. Математическое казино

Имя входного файла:	casino.in
Имя выходного файла:	casino.out
Ограничения по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	80 баллов

Саша – страстный любитель компьютерных игр. Недавно он купил новейшую игру «Математическое казино». В этом казино играют на виртуальные деньги – мани, а каждый раунд игры состоит в решении интереснейшей задачи по математике. Перед началом игры у Саши ноль мани на счету, но программа в любой момент предоставляет ему неограниченный кредит.

Перед началом каждого раунда программа сообщает, на какую тему будет очередная математическая задача и Саша делает ставку на то, что он ее решит. В самом начале игры Саша всегда делает ставку в 1 «мани». Если Саша решает задачу правильно, то он выигрывает раунд и ставка плюсуется к его счету. Если он допускает ошибку в решении, то он проигрывает, и ставка вычитается из его счета. Саша очень самоуверенный и любое неверное решение задачи считает чистой случайностью, поэтому после проигрыша Саша всегда увеличивает ставку в 2 раза. Однако после выигрыша, дабы не вспугнуть удачу, Саша всегда снижает ставку до 1 «мани». Наконец, одолев очередную задачу, и выиграв этот раунд, Саша решает закончить игру.

Например, пусть Саша правильно решил первую задачу (выиграл начальную ставку в 1 «мани», поставил на следующий раунд 1 «мани»), затем неправильно решил вторую задачу (проиграл 1 «мани» и удвоил ставку), неправильно решил и третью задачу (проиграл 2 «мани» и снова удвоил ставку), но четвертую задачку ему все-таки

удалось решить правильно (выиграл 4 «мани», сбросил ставку на 1 «мани»). Затем он правильно решает и пятую задачу (выиграл 1 «мани») и заканчивает игру. Итого на его счету после игры: 1 – 1 – 2 + 4 + 1 = 3 «мани».

Написать программу, которая по имеющейся записи хронологии игры определяет, какое количество «мани» выиграл или проиграл Саша.

Формат входных данных

Первая строка входного файла содержит целое число N ($0 < N \le 2000$) — количество задач, которое решал Саша. Во второй строке располагаются N чисел 0 или 1 через пробел: 0, если Саша решил очередную задачку неправильно, и 1 – если правильно.

Формат выходных данных

Выведите в выходной файл одно целое число — выигрыш или проигрыш Саши (выигрыш определяется положительным числом, а проигрыш – отрицательным).

Пример

casino.in	casino.out
5 1 0 0 1 1	3

Задача O-5. Склад

```
Имя входного файла:              storage.in
Имя выходного файла:             storage.out
Ограничения по времени:          2 секунды
Ограничение по памяти:           64 мегабайта
Максимальная оценка за задачу:   80 баллов
```

На роботизированном складе имеется N отсеков, в которые робот может размещать грузы. Отсек с номером i имеет вместимость c_i. Груз с номером i имеет размер s_i,

поступает на склад в момент времени a_i и забирается со склада в момент времени d_i.

Когда груз с номером i поступает на склад, робот сначала пытается найти отсек, в котором достаточно свободного места для размещения этого груза. Свободное место в пустом отсеке совпадает с его вместимостью. Если в отсеке с вместимостью c находится несколько грузов с суммарным размером d, то свободное место в этом отсеке равно $c - d$.

Если отсеков, в которых достаточно свободного места, несколько, то робот помещает груз в тот из них, в котором свободного места меньше. Если и таких отсеков несколько, то робот выбирает отсек с минимальным номером.

Если отсеков с достаточным количеством свободного места нет, робот пытается переместить грузы, уже расположенные в отсеках. Для этого он пытается найти такой отсек и такой груз в нем, что перемещение его в другой отсек обеспечивает достаточное количество свободного места для размещения поступившего груза. Если таких вариантов перемещения грузов несколько, то выбирается тот вариант, в котором потребуется перемещение груза с минимальным размером. Если и таких вариантов несколько, то выбирается тот вариант перемещения, при котором в отсеке, из которого перемещается груз, после перемещения свободное место будет минимально, а при прочих равных — тот, при котором в отсеке, в который осуществляется перемещение, свободное место после перемещения будет минимально. Если и после этого остается более одного варианта, то выбирается тот вариант, при котором номер перемещаемого груза минимален, и номер отсека, в который он перемещается, – также минимален. Если варианта с перемещением одного груза найти не удалось, то груз не принимается на склад.

Написать программу, которая по списку грузов, поступающих для размещения на складе, выводит последовательность действий, выполняемых роботом.

Формат входных данных

Первая строка входного файла содержит два целых числа:

N — количество отсеков, и M — количество грузов ($1 \le N \le 10$, $1 \le M \le 100$). Вторая строка содержит N целых чисел c_i, определяющих вместимости отсеков ($1 \le c_i \le 10^9$).

Последующие M строк описывают грузы: каждый груз описывается тремя целыми числами: своим размером s_i, временем поступления на склад a_i и временем, когда его забирают со склада d_i ($1 \le s_i \le 10^9$, $1 \le a_i < d_i \le 1000$), все времена во входном файле различны, грузы упорядочены по возрастанию времени поступления на склад). Все числа в строках разделены пробелом.

Формат выходных данных

Выведите последовательность действий робота в том порядке, в котором они выполняются. Следуйте формату выходного файла, приведенного в примере.

Возможны следующие сообщения:

- `put cargo X to cell Y` — положить груз с номером *X* в отсек с номером *Y*;
- `move cargo X from cell Y to cell Z` — переложить груз с номером *X* из отсека с номером *Y* в отсек с номером *Z*;
- `take cargo X from cell Y` — достать груз с номером *X* из отсека с номером Y.
- `cargo X cannot be stored` — если груз с номером *X* разместить невозможно.

Примеры

storage.in	storage.out
3 5 3 2 10 1 1 6 3 2 8 9 3 5 2 4 9 12 7 10	put cargo 1 to cell 2 put cargo 2 to cell 1 put cargo 3 to cell 3 move cargo 1 from cell 2 to cell 3 put cargo 4 to cell 2 take cargo 3 from cell 3 take cargo 1 from cell 3 cargo 5 cannot be stored take cargo 2 from cell 1 take cargo 4 from cell 2

Задача О-6. Игра с шашками

Имя входного файла:	game.in
Имя выходного файла:	game.out
Ограничения по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	80 баллов

Поле для игры с шашками – длинная горизонтальная полоска, размеченная на клетки. Клетки пронумерованы от 1 до N ($2 < N < 10000$). На поле стоят две шашки. Позиция каждой из шашек определяется номером клетки, в которой она стоит.

Играют двое. Каждый игрок при своем ходе должен переместить любую шашку на одну, две или три клетки в сторону клетки 1 (сделать 1, 2 или 3 шага). Перепрыгивать через стоящую впереди шашку нельзя, но можно сдваивать шашки. На сдваивание шашек тратится два шага из трех доступных игроку (то есть сдваивать можно либо шашки, стоящие вплотную друг к другу, либо шашки, между которыми есть только одна пустая клетка). Если произошло сдваивание – ход передается другому игроку, который делает ход одной шашкой, оставив другую на месте.

Выигрывает тот, кто сдвоит шашку на клетке с номером 1.

Написать программу, реализующую алгоритм, обеспечивающий победу игроку, начинающему игру.

Формат входных данных

В первой строке входного файла содержится число K ($0 < K < 10$) – количество начальных позиций. В последующих K строках содержится по два целых числа ще 3 до 10000, разделенных пробелом – номера начальных позиций шашек на игровом поле.

Формат выходных данных

В выходной файл выводится K строчек – ответ на каждую начальную позицию.

Если при заданной начальной позиции шашек в игре не достигается выигрыш (при правильной игре противника) выводится слово **NO**.

Если выигрыш достижим, то в файл выводится первый ход начинающего игру, который приводит к его выигрышу независимо от того, как играет соперник. Ход описывается парой чисел ***i, j*** через пробел, означающих, что выигрышный ход игрока – это перемещение шашки из клетки с номером ***i*** в клетку с номером ***j***. Например, «4 3» означает, что игрок двигает шашку, стоящую в клетке 4, на одну клетку в сторону клетки 1.

Примеры

game.in	game.out
2 3 3 4 3	NO 4 3

Задача О-7. Радио «Байтик»

Имя входного файла:	radio.in
Имя выходного файла:	radio.out
Ограничения по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	80 баллов

Как известно, при распространении радиоволн возникает интерференция, поэтому если рядом расположены две радиопередающие станции, вещающие на одной и той же частоте, то качество радиопередач резко снижается.

Радиостанция «Байтик» планирует транслировать свои программы в стране Флатландия. Министерство связи Флатландии выдало радиостанции лицензию на вещание на двух различных частотах.

Владельцы радиостанции имеют возможность транслировать свои радиопрограммы с использованием N радиовышек, расположенных в различных точках страны. Для осуществления трансляции на каждой радиовышке требуется установить специальный передатчик – трансмиттер. Каждый передатчик можно настроить на одну из двух частот, выделенных радиостанции. Кроме частоты вещания, передатчик характеризуется также своей мощностью. Чем мощнее передатчик, тем на большее расстояние он распространяет радиоволны. Для простоты, предположим, что передатчик мощности R распространяет радиоволны на расстояние, равное R километрам.

Все передатчики, установленные на вышках, должны, согласно инструкции министерства, иметь одну и ту же мощность. Чтобы программы радиостанции могли приниматься на как можно большей территории, мощность передатчиков должна быть как можно большей. С другой стороны, необходимо, чтобы прием передач был качественным на всей территории Флатландии. Прием передач считается качественным, если не существует такого

участка ненулевой площади, на который радиоволны радиостанции «Байтик» приходят на одной частоте одновременно с двух вышек.

Написать программу, которая определяет, какую максимальную мощность можно было установить на всех передатчиках, позволяющую выбрать на каждом передатчике такую одну из двух частот передачи, чтобы прием был качественным на всей территории Флатландии.

Формат входных данных

Первая строка входного файла содержит число N — количество вышек ($3 \le N \le 1200$). Последующие N строк содержат по два целых числа — координаты вышек. Координаты заданы в километрах и не превышают 10^4 по модулю. Все точки, в которых расположены вышки, различны. Все числа в строках разделены пробелом.

Формат выходных данных

В первой строке выходного файла выводится вещественное число — искомая мощность передатчиков. Во второй строке выводятся N чисел, где i-е число должно быть равно 1, если соответствующий передатчик должен вещать на первой частоте, и 2, если на второй. Ответ должен быть выведен с точностью, не меньшей 10^{-8}.

Пример

radio.in	radio.out
4 0 0 0 1 1 0 1 1	0.70710678118654752 1 2 2 1

Разбор задач

Задача О-1. Клетчатые фигуры

Тема: задача на реализацию; легкая

Будем рассматривать заданное поле как совокупность белых и чёрных единичных клеток. Заметим, что суммарный периметр всех фигур равен количеству ребер, одна из прилежащих клеток которых белая, а другая чёрная (рис 1, исключение составляют клетки, примыкающие к краю поля – рис 2). Таким образом, решение задачи необходимо производить в два этапа:

1. По исходным данным построить двумерный массив размера ***M*** x ***N***, обозначив в нем некоторым образом чёрные и белые клетки (например, 0 - белая, 1 – чёрная).
2. Пройтись по всем клеткам этого массива и для каждой чёрной клетки посчитать количество смежных с ней белых. Кроме того, надо учитывать черные клетки, которые находятся на границе поля (для простоты реализации можно применить граничный метод — добавить в поле окантовку из белых клеток)

Рис. 1.
Граница черной и белой клеток

Рис 2.
Граница черной клетки и края поля

Приведем полное решение жюри на Delphi:

```
const
  maxn = 100;

var
  N, M, K, answer : integer;
  field: array [0..maxn+1, 0..maxn+1] of
integer;

procedure readdata;
var
  i, x, y: integer;
begin
  fillchar(field, sizeof(field), 0);
  read(N, M, K);
  for i:=1 to K do begin
    read(x, y);
    field[x, y]:=1;
  end;
end;

procedure solve;
var
  i, j: integer;
begin
  answer:=0;
  for i:=1 to N+1 do
    for j:=1 to M+1 do begin
      if field[i,j]<>field[i,j-1] then
        inc(answer);
      if field[i,j]<>field[i-1,j] then
        inc(answer);
    end;
end;
```

```
begin
  assign(input,'figure.in');reset(input);
  assign(output,'figure.out');rewrite(output);
  readdata;
  solve;
  writeln(answer);
  close(input);
  close(output);
end.
```

Авторы разбора
И.И. Попелышев, С.А. Пучин, С.В. Шедов

Задача О-2. Алгоритм Евклида

Тема: Математика; средняя

При решении данной задачи следует учитывать следующее. Поскольку ограничения на числа ***a***, ***b*** ,***c*** ,***d*** в этой задаче достаточно большие, простая симуляция описанного в условии алгоритма не будет укладываться в ограничения по времени. Однако, правильная реализация простой симуляции алгоритма набирает порядка 30 баллов

Более быстрый алгоритм таков. Разобьем процесс применения алгоритма Евклида к паре чисел (***a***, ***b)*** на фазы. Началом фазы будем считать обмен на шаге 3, а окончанием — момент перед следующим таким обменом (или окончание работы алгоритма — для последней фазы). Нетрудно видеть, что, если в начале фазы $a=a_0$, $b=b_0$, то в ее конце $a=a_0 \bmod b_0$, $b=b_0$. В то же время на каждом шаге внутри фазы из ***a*** вычиталось b_0, т.е. ***a*** последовательно было равно a_0, $a_0 - b_0$, $a_0 - 2b_0$, … , $a_0 \bmod b_0$.

Для того чтобы проверить, встречалась ли пара (c, d) во время фазы, надо проверить следующие условия:

1. $d=b_0$
2. c - $(a_0 \bmod b_0)$ делится на b_0
3. $c \le c_0$

Приведём пример реализации:

```
var
  a,b,c,d,n : int64;

procedure swap(var a,b:int64);
var
  t:integer;
begin
  t:=a;a:=b;b:=t;
end;

procedure solve;
begin
  while (b<>0) do begin
    // проверка интервала (a,b) и (a mod b,b)
    if (b = d) and (c < a)
       and ((c - a mod b) mod b = 0) then begin
      writeln('YES');
      exit;
    end;
    // переход к следующему интрервалу.
    a := a mod b;
    swap(a,b);
  end;
  writeln('NO');
end;

procedure init;
var
  i: integer;
begin
  assign(input, 'euclid.in');reset(input);
  assign(output, 'euclid.out');rewrite(output);
  readln(n);
  for i := 1 to n do begin
    read(a,b,c,d);
    solve;
```

```
  end;
end;

begin
  init;
  close(output);
end.
```

В завершении отметим, что временная сложность предложенного решения совпадает с сложностью работы алгоритма Евклида, которая равна $O(\log N)$. Доказательство и различные интерпретации алгоритма можно найти в статье С.Б.Гашкова "Алгоритм Евклида, цепные дроби, числа Фибоначчи и квадрирование прямоугольников"[6]

Задача О-3. Авторитетное сообщество

Тема: теория графов – обход, конденсация; сложная

Начнем решение с построения более удобной модели задачи. Представим турнир в виде ориентированного графа. В качестве вершин графа возьмем игроков, а в качестве ребер — результаты игр. Ребро направим от игрока-победителя к проигравшему игроку. Каждой вершине припишем рейтинг, который равен рейтингу игрока, соответствующего этой вершине.

Введем два типа множеств вершин:

Множество первого типа – множество вершин A, такое что ни в одну вершину этого множества не идет ребро из вершины не принадлежащей A.

Множество второго типа – множество вершин B, такое что ни из одной вершины множества не исходит ребро в вершину не принадлежащую B.

Если обозначить множество всех игроков за V, то для любого множества первого типа A — V/A является множеством второго типа (для множества 2-го типа B — V/B множеством первого типа). Тогда, чтобы решить нашу задачу требуется найти

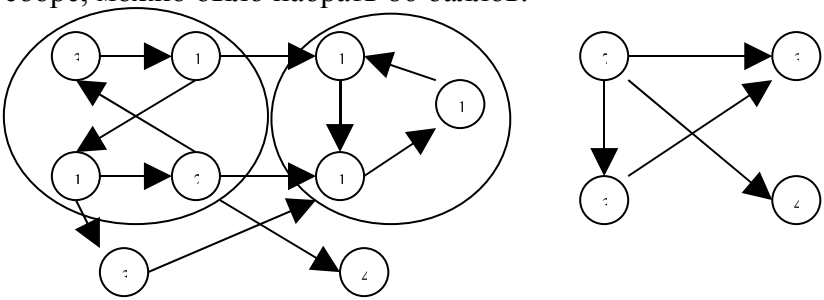
множество первого типа, с максимальным суммарным рейтингом. Из требований условия следует важное дополнение определений — множества первого и второго типов не могут совпадать со всеми вершинами графа. Заметим еще и то, что задача нахождения множества первого типа с максимальным рейтингом эквивалентна задаче построения множества второго типа с минимальным рейтингом. Далее иногда будем говорить минимальное или максимальное множество, подразумевая минимальное по рейтингу или максимальное по рейтингу множество.

Перейдем теперь к непосредственному решению. Будем решать несколько другую задачу, а именно строить множество второго типа $\boldsymbol{B}$ — не вошедших в авторитетное множество игроков. Построив множество $\boldsymbol{B}$ с минимальным суммарным рейтингом, мы можем взять оставшиеся вершины в качестве ответа к исходной задаче.

Заметим, что если мы *не* берем какую-то вершину $\boldsymbol{v}$ в наше авторитетное множество $\boldsymbol{A}$, то все вершины, которые достижимы из v так же не должны попасть в множество $\boldsymbol{A}$ (*Почему это утверждение верно? Докажите*). Таким образом, если мы берем вершину в множество $\boldsymbol{B}$, то все вершины, достижимые из этой вершины мы также должны отнести к множеству $\boldsymbol{B}$. Так как по условию авторитетное сообщество не может состоять из всех игроков, то в множестве $\boldsymbol{B}$ должна быть хотя бы одна вершина. Пусть мы взяли какую-то вершину $\boldsymbol{v}$ в множество $\boldsymbol{B}$. После положим в множество $\boldsymbol{B}$ все вершины достижимые из нашей вершины $\boldsymbol{v}$. Заметим, что такое множество является корректным множеством второго типа, причем минимальным среди множеств второго типа, включающих вершину $\boldsymbol{v}$ (*Почему? Докажите это*). Построив множество $\boldsymbol{B}$ для каждой вершины, и взяв минимальное из них, мы получаем минимальное множество $\boldsymbol{B}_{\mathbf{min}}$ второго типа. Ответом к задаче являются все вершины не вошедшие в минимальное множество $\boldsymbol{B}_{\mathbf{min}}$.

```
min_aft = +∞;
B_min = {}
for v=1 to N do begin
//Строим множество  достижимых вершин B, например,
поиском в глубину.
//Считаем суммарный авторитет aft(B) множества B.
      if min_aft > aft(B) then
        min_aft = aft(B)
        B_min = B;
      end
end;
//Вывести все вершины не вошедшие в множество B_min
```

Оценим сложность предложенного алгоритма. Для хранения графа мы можем использовать матрицу смежности или список смежности. В первом случае быстродействие поиска в глубину порядка N^2, а во втором случае порядка M, где M – число ребер. Наше решение N раз “запускает” поиск в глубину, а значит при хранении графа в виде матрицы смежности быстродействие решения порядка N^3, а при хранении списком смежности $N \cdot M$. При использовании последнего способа хранения графа предложенное решение набирало 70 баллов. Сделав отсечения в переборе, можно было набрать 86 баллов.



Исходный граф и его конденсация. Внутри вершин графов указан рейтинг.

Решение на 100 баллов несколько сложнее и требовало построения компонент сильной связанности. *Компонентой сильной связанности* ориентированного графа называется максимальное множество вершин, в котором существуют пути из любой вершины в любую другую вершину. Если выделить компоненты сильной связанности, то можно построить другой граф – *конденсацию исходного* (редуцированный граф). Вершинами графа – конденсации являются компоненты сильной связанности исходного. Если в исходном графе есть ребро между вершинами из разных компонент сильной связанности, то в графе–конденсации есть ребро между соответствующими компонентами, той же ориентации. Граф-конденсация всегда однозначно строится по исходному и *ациклично*.

Заметим, что если вершина входит в множество первого типа, то и компонента сильной связанности, которой она принадлежит, тоже должна входить в это множество. И наоборот если вершина входит в множество второго типа, то и соответствующая компонента тоже входит.

Построим для графа исходных данных конденсацию. При этом рейтинг вершин графа-конденсации будем считать как суммарный рейтинг вершин вошедших в соответствующую компоненту сильной связанности. Таким образом, мы получаем исходную задачу, но для графа – конденсации, а значит для ацикличного графа. Тогда в графе есть хотя бы один *сток* - вершина ориентированного графа, у которой нет исходящих ребер. А для такого графа задача решается просто, а именно решением будут все вершины, за исключением одной: вершины–стока (*Почему? Докажите. Будет ли утверждение верно, если рейтинг игроков может быть отрицательным?*). Если найден сток с минимальным рейтингом, то решением будут все вершины за исключением этого стока.

Для выделения компонент сильной связанности существует алгоритм со сложностью порядка M. А значит, и все наше решение имеет такую же сложность. Подробнее про данный

алгоритм, представление графа в памяти и поиск в ширину можно прочитать в [4] и [5].

Автор задачи и разбора — А. А. Алексеев

Задача О-4. Математическое казино

Тема: разное; очень легкая

Рассмотрим задачу, когда Саша после начала игры проиграл подряд k раундов, и на k+1-ом раунде выиграл. Заметим, что тогда счет Саши после проигрыша k раундов уменьшится на $1 + 2 + 4 + \ldots + 2^{k-2} + 2^{k-1} = 2^k - 1$ «мани», а затем, после выигрыша на k+1-ом раунде прибавится 2^k. Таким образом, к счету Саши после первых k+1 раунда прибавится 1. Далее, увеличиваем ответ на единицу и считаем, что игра возобновилась сначала. То есть, он снова проиграет некоторое число раундов, а потом выиграет. Если вспомнить, что Саша закончил играть после победы, то уже очевидно, что ответом к задаче является число выигранных раундов.

Таким образом, решением задачи является сумма всех «единичек» или просто сумма элементов введенной последовательности. Следует отметить, что многие участники релизовывали процедуры «длинной арифметики» и теряли на этом время.

Автор задачи И.И. Попелышев,
автор разбора А.А. Алексеев

Задача О-5. Склад

Тема: задача на реализацию; средняя

При решении данной задачи следует аккуратно формализовать указанные в условии действия. Особое внимание следует уделять вторичным критериям оптимизации.

В этой задаче частичные баллы набирают решения, которые не учитывают один или более критериев оптимизации. Приведём полное решение:

```
type
```

```
  TElem = record
    time : integer;
    box : integer;
  end;

const
  maxN = 10;
  maxM = 100;

var
  // свободное место в отсеках
  a : array [1..maxN] of integer;
  // номер отсека, где лежит груз и его размер
  b,s:array [1..maxM] of integer;
  // массив для сортировки событий добавления
  // извлечения по времени
  events: array [1..2*maxM] of TElem;
  n, m: integer;
procedure qsort(l,r:integer);
var
  i,j:integer;
  x,t: TElem;
begin
  i:=l;j:=r; x:=events[i+random(j-i)];
  repeat
    while events[i].time < x.time do inc(i);
    while events[j].time > x.time do dec(j);
    if (i<=j) then begin
      t:=events[i];
      events[i] := events[j];
      events[j] := t;
      inc(i);
      dec(j);
    end;
  until i>j;
  if i<r then qsort(i,r);
  if j>l then qsort(l,j);
end;

procedure init;
var
  i,st, en : integer;
```

```
begin
  assign(input,'storage.in');reset(input);
  fillchar(b,sizeof(b),0);
  read(n);read(m);
  for i:=1 to n do
    read(a[i]);
  for i:=1 to m do begin
    read(s[i],st,en);
    events[2*i].time:=st;
    events[2*i].box:=i;
    events[2*i-1].time:=en;
    events[2*i-1].box:=-i;
  end;
  close(input);
  qsort(1,2*m);
end;

procedure solve;
var
  i,j,k,j_,k_,mind,ss,sp1,sp2, f :integer;
begin
  for i:=1 to 2*m do begin
    f := events[i].box; // номер добавляемого груза
    if (f<0) then begin// удаляем груз
      if (b[-f] <> 0) then begin
        writeln('take cargo ', -f,' from cell ',b[-
f]);
        inc(a[b[-f]], s[-f]); // освобождаем место
        b[-f] := 0; // помечаем, где лежит груз
      end;
    end else begin
      k := 0; //лучший отсек для грузу без
перемещений
      mind := maxlongint; // и свободное место в
отсеке
      for j:=1 to n do // поиск отсека без
перемещений
        if ((a[j]>=s[f]) and (a[j]<mind)) then begin
          k := j;
          mind := a[j];
        end;
      if (k<>0) then begin // нашли пустое место
```

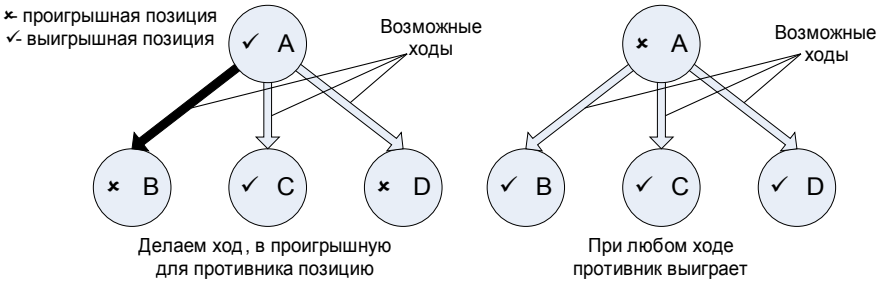
```
        dec(a[k], s[f]);
        b[f] := k;
        writeln('put cargo ', f, ' to cell ', k);
      end else begin
        j_ := 0; // ищем груз для перемещения
        k_ := 0; // и отсек, куда перемещать
        ss := maxlongint; // оптимальный груз
        sp1 := maxlongint;// оптимум места - откуда
        sp2 := maxlongint;// оптимум места – куда
        for j:=1 to m do
          for k:=1 to n do
            if (b[j]<>0) and (b[j]<>k) and
(s[j]+a[b[j]]>=s[f]) and (s[j]<=a[k]) then
            // можем переместить, учитываем критерии:
              if (ss>s[j]) or
                ((ss = s[j]) and (s[j]+a[b[j]]>sp1))
or
                ((ss = s[j]) and (s[j]+a[b[j]]=sp1)
and (a[k]<sp2)) then begin
                j_ := j;
                k_ := k;
                ss := s[j];
                sp1 := s[j]+a[b[j]];
                sp2 := a[k];
              end;
        if j_<>0 then begin // нашли что передвинуть
          writeln('move cargo ',j_,' from cell
',b[j_],' to cell ',k_);
          writeln('put cargo ',f,' to cell ',b[j_]);
          a[b[j_]] := a[b[j_]] + s[j_] - s[f];
          a[k_] := a[k_] - s[j_];
          b[f] := b[j_];
          b[j_] := k_;
        end else // не можем добавить груз
          writeln('cargo ',f,' cannot be stored')
      end;
    end;
  end;
end;

begin
  init;
```

```
  assign(output,'storage.out');rewrite(output);
  solve;
  close(output);
end.
```

Задача O-6. Игра с шашками

Тема: игры, динамическое программирование; легкая

Сначала отвлечёмся от нашей задачи и рассмотрим абстрактную игру. Пусть при нашем ходе игрок находится в позиции *A*, и может сделать ход в позицию *B, C, D*. И все последующие наши ходы и ходы противника правильные, то есть если в какой-то позиции игрок может выиграть в независимости от игры противника, то он побеждает. Тогда, находясь в позиции *A* мы всегда выиграем, если сделаем ход в какую-то из проигрышных состояний для противника. То есть, если какая-то из позиций *B, C, D* проигрышная. Если все из них выигрышные, то куда бы мы не пошли, противник выиграет.



Теперь, вернемся к нашей задаче. Текущее состояние игры определяется парой координат шашек (a,b). Заметим, что позиции (a,b) и (b,a) совпадают, поэтому можно рассматривать только позиции при $a \le b$. Игрок может сделать ход в позиции: (a-1,b), (a-2,b), (a-3,b), (a,b-1), (a,b-2), (a,b-3). Ходы в состояния (a,b-2), (a,b-3) возможны, если $b-1 \ne a$ и $b-2 \ne a$, соответственно.

Обозначим win[a,b] = 0, если (a,b) – проигрышная, и 1 – если выигрышная.

10										**1**
9									**1**	**1**
8								**1**	**1**	**1**
7							**1**	**1**	**1**	**0**
6						**1**	**1**	**1**	**0**	**1**
5					**1**	**1**	**1**	**0**	**1**	**1**
4				**1**	**1**	**1**	**0**	**1**	**1**	**1**
3			**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**
2		**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**
1	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**
a/b	1	2	3	4	5	6	7	8	9	10

В матрице `win` выделены рассматриваемое состояние и позиции, куда можно пойти и состояние (с нулём), за счёт которого (4,6) – выигрышная.

По заданному соотношению, поднимаясь от задач меньшей размерности к задачам большей, можно заполнить всю таблицу. И для заданного запроса определить выигрышность позиции и правильный ход, в случае выигрыша.

Но, построив, заданную таблицу, можно заметить закономерность в проигрышных позициях. Проигрышными являются такие `(a,b)`, что `(b-a) mod 4 = 3`. Исключения составляют (1,1), (2,2) и (3,3).

Более подробно про игровые задачи рекомендуем прочитать в брошюре А. Шеня “Игры и стратегии с точки зрения математики”[12].

Автор задачи Ю.Г. Чернов,
автор разбора А.А. Шестимеров

Задача О-7. Радио «Байтик»

Тема: вычислительная геометрия, графы; сложная

Будем использовать при решении этой задачи двоичный поиск по ответу. Мы ищем ответ на интервале $[0..L_{max}]$, где $L_{max} = ((X_{max}-X_{min})^2 + (Y_{max}-Y_{min})^2)/4 + 1$ — мощность, которую заведомо нельзя брать. Пусть текущий интервал поиска $[L, R]$, тогда

проверим можно ли оспользовать мощность $(L+R)/2$. Если можно, то ищем ответ на интервале $[(L+R)/2,R]$, иначе на $[L,(L+R)/2]$:

```
  while l < r - 1 do begin
    m := (l + r) div 2;
    if not test(m) then
      r := m
    else
      l := m;
  end;
```

Пусть текущая гипотеза равна $\boldsymbol{R}$. Требуется проверить, можно ли искомым образом сопоставить частоты вышкам.

С этой целью построим следующий граф. Вершинами графа будут являться вышки. Две вершины соединяем ребром, если расстояние между вышками строго меньше $2\boldsymbol{R}$. Тогда искомое сопоставление частот вышкам возможно, если построенный граф является двудольным. Граф называется двудольным, если множество вершин графа можно разделить на два непересекающихся множества $\boldsymbol{V_1}$ и $\boldsymbol{V_2}$, так что вершины из одного множества не соединены между собой ребром. Для проверки двудольности графа можно использовать обход в глубину. Нам необходимо найти максимальный $\boldsymbol{R}$ такой, что граф является двудольным.

Пример реализации процедуры проверки:

```
// d[i][j] – расстояние между двумя
передатчиками
// nnew – массив для раскраски графа в два цвета
// (соответсвующий долям графа)
  function test(r: longint) : boolean;
  var
    i:integer;
  begin
    fillchar(nnew, sizeof(nnew), 0);
    flag := false;
```

```
    // проверяем все компоненты графа
    for i := 1 to n do begin
      if (nnew [i] = 0) then
        dfs(i, R,1);
    end;
    test := flag;
  end;

  procedure dfs(i, r, c: longint);
  var
    j: longint;
  begin
    if nnew [i] = 0 then begin
      // не  просматривали вершину
      nnew[i] := c;
    end else begin
      if nnew [i] <> c then
        //вершину просмотрели, её цвет
отличается
        //- граф не двудольный
        flag := true;
      exit;
    end;
    for j := 1 to n do if i <> j then begin
      if (d[i][j] < 4*r) and not flag then begin
        dfs(j, r, 3 - c);
      end;
    end;
  end;
```

Нам остается только вывести соответствующую мощности L раскраску графа. Сложность этого решения $O(N^2 \log (X_{max}-X_{min}))$.

Приведенный выше алгоритм, достаточно очевиден, и не требует дополнительныз рассуждений для обоснования во время тура. При этом существует жадный алгоритм (п*опробуйте придумать и обосновать его*) меньшей сложности — $O(N^2)$.

Литература

1. Андреева Е. В, Фалина И. Н. Турбо-Паскаль в школе. — М.: Изд-во Бочкаревой Н.Ф., 1998.

2. Долинский М. С. Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач: Учебное пособие. — СПб.: Питер, 2005.

Книга содержит условия и разборы личных, командных и заочных Московских олимпиад, в качестве дополнения читатель найдет крайне полезную статью поиске в глубину в графе и его приложениях:

3. Андреева Е. В., Гуровец В.М., Матюхин В.А. Московские олимпиады по информатике. — М.: МЦМНО, 2006. (http://olympiads.ru/books/mskolymp/mosinformatolymp.pdf)

Лучший учебник по построению и анализу эффективных алгоритмов, содержит практически весь неободимый теоритический материал:.

4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. — М.: МЦНМО, 2000.

Ползная книга, для углубления знаний в теории графов:

5. Кристофедес Н. Теория графов. Алгоритмический подход. — М.: Мир, 1978.

Статья к задаче “Алгоритм Евклида”:

6. Гашкова С.Б. Алгоритм Евклида, цепные дроби, числа Фибоначчи и квадрирование прямоугольников — Сборник "Математическое Просвещение" Выпуск 6, 2002 (http://www.mccme.ru/free-books/matpros/i7093115.pdf.zip)

Комбинаторика и теория графов:

7. Липский В. Комбинаторика для программистов. — М.: Мир, 1988.

Следующие 4 книги содержат задачи, полезные для решения при подготовке к областным олимпиадам:

8. Овсянников А. П., Овсянникова Т. В., Марченко А. П., Прохоров Р. В. Избранные задачи олимпиад по информатике. — М.: Тровант, 1997.

9. Окулов С.М. Геометрические алгоритмы. “Информатика”, №15, 16, 17, 2000.

10. Окулов С.М. 100 задач по информатике. — Киров: изд-во ВГПУ, 2000.

11. Окулов С. М. Программирование в алгоритмах. — М.: БИНОМ, Лаборатория знаний, 2002.

К задаче “Игра с шашками”:

12. Шень А. Игры и стратегии с точки зрения математики — М.: МЦНМО, 2007.
(ftp://ftp.mccme.ru/users/shen/games.zip)

От задач без массивов до сбалансированных деревьев:

13. Шень А. Программирование: теоремы и задачи — М.: МЦНМО, 2004
(ftp://ftp.mccme.ru/users/shen/progbook2/progbookps.zip).

.

Оглавление