Департамент образования администрации города Нижнего Новгорода
Городской ресурсный центр физико-математического образования

Десятая нижегородская городская олимпиада школьников по информатике

30 января 2014 г.

Нижний Новгород
2014

Результаты, архивы и другие материалы олимпиады
можно найти на сайте `http://olympiads.nnov.ru`

Оригинал-макет подготовлен в системе LaTeX 2$_\varepsilon$
с использованием набора шрифтов LH.

Десятая городская олимпиада по информатике

30 января 2014 г. Городской ресурсный центр физико-математического образования в лице учредителей: Департамента образования администрации города Нижнего Новгорода и МБОУ Лицей № 40, в партнёрстве с Нижегородским государственным техническим университетом им. Р. Е. Алексеева проводит десятую городскую олимпиаду по информатике среди учащихся 6–11 классов образовательных учреждений города Нижнего Новгорода. Целью проведения олимпиады является поиск талантливой молодёжи, привлечение её в науку, повышения уровня преподавания предметов физико-математического цикла.

Генеральным спонсором городской олимпиады школьников по информатике является компания «Мера НН».

Десятая городская олимпиада проводится в НГТУ (корпус 6, Казанское шоссе, д.12).

Олимпиада проводится в соответствии с Положением о городской олимпиаде по информатике (Приказ № 1453 от 07.11.2008 Департамента образования и СПЗД администрации г. Нижнего Новгорода).

Разрешённые среды программирования — Borland Pascal, Free Pascal, Borland C, Borland C++, GNU C (MinGW), GNU C++ (MinGW), Free Basic, Python 3. Ввод и вывод данных в программах осуществляется через файлы.

Регламент проведения олимпиады

9:30—10:00	Регистрация участников
10:00—10:30	Открытие олимпиады, информация от жюри
10:30—15:30	Решение задач олимпиады
15:30—16:00	Обед
16:00—16:30	Открытое тестирование
16:30—18:00	Приветствие участников олимпиады, выступления учредителей, спонсоров. Подведение итогов олимпиады. Поздравление победителей и призёров.

Состав оргкомитета олимпиады

Сидоркина С. Л., первый заместитель директора департамента образования администрации города Нижнего Новгорода;
Цветков М. И., начальник отдела общего среднего образования департамента образования администрации города Нижнего Новгорода;
Бовкун И. Л., главный специалист отдела общего среднего образования департамента образования администрации города Нижнего Новгорода;
Ивашкин Е. Г., проректор НГТУ им. Р.Е. Алексеева;
Бушуева М. Е., декан ФДПиДОУ НГТУ им. Р.Е. Алексеева;
Кулагина Л. В., зам. директора по учебной работе ИНЭУ НГТУ им. Р.Е. Алексеева;
Умнова Н. С., директор муниципального бюджетного образовательного учреждения лицей № 40;
Евстратова Л. П., заместитель директора по учебно-воспитательной работе МБОУ лицей № 40;
Гашпар И. Л., куратор классов НОЦ ИПФ РАН;

Состав предметной комиссии (жюри)

Председатель — Калинин П. А., старший инженер по разработке программного обеспечения, ЗАО «Интел А/О», к.ф.-м.н.;

Члены комиссии:
Евстратова Л. П., заместитель директора МОУ лицей № 40, ответственный секретарь комиссии;
Лазарев Е. А., старший инженер по разработке программного обеспечения, ЗАО «Интел А/О», ведущий программист кафедры «Прикладная математика» ИРИТ НГТУ, к.т.н.;
Разенштейн И. П., аспирант второго года EECS MIT;
Тимин А. А., студент 3 курса ФИВТ МФТИ;
Тимушев Р. И., старший инженер-программист, Grid Dynamics;
Шмелёв А. С., инженер-программист, НПП «ПРИМА».

Ниже приведены примеры текстов программ, написанных на разрешённых языках программирования. Программы считывают данные (два числа в одной строке) из файла с именем `example.in` и выводят в файл с именем `example.out` их сумму:

Pascal

```
var buf,bufo:text;
    a,b:integer;
begin
  assign(buf,'example.in');
  reset(buf);
  read(buf,a,b);
  assign(bufo,'example.out');
  rewrite(bufo);
  writeln(bufo,a+b);
  close(buf);
  close(bufo);
end.
```

C/C++

```
#include <stdio.h>
int main() {
  FILE *buf,*bufo;
  int a,b;
  buf=fopen("example.in","r");
  fscanf(buf,"%d %d",&a,&b);
  fclose(buf);
  bufo=fopen("example.out","w");
  fprintf(bufo,"%d\n",a+b);
  fclose(bufo);
  return 0;
}
```

Basic

```
OPEN "example.in" FOR INPUT AS #1
OPEN "example.out" FOR OUTPUT AS #2
INPUT #1, A, B
PRINT #2, A + B
CLOSE #2, #1
```

Python 3

```
inf = open("example.in", "r")
a,b = map(int, inf.read().split())
ouf = open("example.out", "w")
ouf.write(str(a+b))
```

X Городская олимпиада школьников по информатике 30 января 2014 г.

Задача 1. Классики

Входной файл	`classics.in`
Выходной файл	`classics.out`
Ограничение по времени	1 с
Ограничение по памяти	256 МиБ
Максимальный балл за задачу	100

Маленькие девочки Оля и Катя любят гулять в парке рядом со своим домом. В этом парке есть асфальтовая дорожка, часть которой вымощена большими одинаковыми бетонными плитами. Ширина каждой плиты равна ширине дорожки, таким образом, дорожка образована просто последовательно лежащими плитами. До и после участка плит идёт асфальт.

Конечно, Катя и Оля всегда, когда идут по этому участку, прыгают по плитам. Катя за один прыжок прыгает ровно на A плит вперёд (т.е. если она стояла на второй плите, то за один прыжок она окажется на $(2 + A)$-й), Оля — на B (например, со второй плиты на $(2 + B)$-ю); назад девочки не прыгают. Первый прыжок обе девочки делают с асфальта на одну из первых плит, при этом Катя может прыгнуть сразу на любую из первых A плит, а Оля — на любую из первых B. Аналогично, когда Катя оказывается на одной из последних A плит, она одним прыжком перепрыгивает на продолжение асфальта; Оля так же поступает, когда оказывается на одной из B последних плит. В частности, если плит меньше A, то Катя может перепрыгнуть их все за один прыжок (сразу с асфальта на асфальт), а может затратить и два прыжка; Оля может поступить так же, если плит меньше B.

Однажды девочки заспорили: сколько всего плит на этом участке? Катя помнит, что последний раз, когда они гуляли, она сделала N прыжков, а Оля — M. Помогите им определить, сколько всего могло быть плит.

Формат входных данных

Во входном файле находятся четыре натуральных числа: A, B, N и M. Все числа не превосходят 10^9.

Формат выходных данных

В выходной файл выведите два натуральных числа: минимальное и максимальное ненулевое количество плит, которые могли быть на этом участке. Если ни одно ненулевое количество плит не соответствует условию, выведите два раза число 0.

Пример

Входной файл	*Выходной файл*
2 3 2 2	1 3
2 3 4 3	5 7
2 3 100 3	0 0
4 6 3 2	5 11

Примечание: Тесты к этой задаче будут устроены следующим образом:

- тесты суммарной стоимостью 25 баллов будут иметь $A = 2$, $B = 3$, $N \leqslant 1000$, $M \leqslant 1000$;
- тесты суммарной стоимостью 25 баллов будут иметь $A = 2$, $B = 3$, но или N, или M, или оба значения будут больше 1000;
- тесты суммарной стоимостью 25 баллов будут иметь $N \leqslant 1000$, $M \leqslant 1000$, но или $A \neq 2$, или $B \neq 3$, или и то, и другое;
- тесты суммарной стоимостью 25 баллов не будут попадать ни под одно из указанных выше условий.

Примечание: В первом примере количество плит может быть от 1 до 3. Действительно, если плита была только одна, то обе девочки могли совершить два прыжка: первым прыжком прыгнуть с асфальта на плиту, вторым прыжком — с плиты на асфальт. Если плит было две, то обе девочки могли прыгнуть сначала, например, на первую плиту, а с неё сразу на асфальт после плит. Если плит было три, то обе девочки могли прыгнуть сразу на среднюю плиту, и с неё сразу на асфальт после плит. А если бы плит было четыре, то Катя уже не смогла бы их перепрыгнуть за два прыжка.

Решение

Это довольно простая задача, и основной сложностью здесь было не запутаться при написании решения. Для этого надо придумать, как бы организовать код как можно проще, чтобы обойтись минимальным количеством случаев и формул. Один из наиболее простых способов сделать это состоял в следующем.

Рассмотрим по отдельности информацию, которую мы знаем о прыжках Кати, и информацию о прыжках Оли. Катя за один прыжок прыгает на A плит вперёд, и тратит N прыжков, чтобы преодолеть весь участок. Определим, какое минимальное и максимальное количество плит могло быть на этом участке, исходя лишь из этих данных.

Первый и последний прыжки Катя делает с асфальта и на асфальт. Рассмотрим остальные $N - 2$ прыжка. За один прыжок Катя продвигается вперёд на A плит, за $N - 2$ прыжка — на $(N - 2) \cdot A$ плит. Учитывая ту плиту, на которую первый раз приземлилась Катя, получаем $(N - 2) \cdot A + 1$ плит между той плитой, куда Катя приземлилась первый раз, и той, куда она приземлилась в последний раз, включительно.

Несложно видеть, что это и есть минимальное количество плит, которые могли быть на этом участке. Действительно, меньше их, очевидно, не могло быть (иначе $N - 2$ «полных» прыжка не поместились бы), а столько плит могло быть: Катя просто первым прыжком прыгнула бы на первую из плит, и последним прыжком прыгнула бы с последней плиты.

Определим теперь, сколько максимально плит могло быть. Первым прыжком Катя могла перепрыгнуть не более $A - 1$ плит, т.е. в начале дорожки до первого приземления Кати могла быть максимум $A - 1$ плита. Аналогично в конце дорожки после последнего приземления Кати тоже могла быть максимум $A - 1$ плита. Итого получаем, что максимальное количество плит равно $(N - 2) \cdot A + 1 + 2A - 2$. Также несложно видеть, что любое промежуточное количество плит также возможно.

Итого, исходя лишь из данных о Катиных прыжках, получаем, что искомое количество плит может находиться в пределах

$$\text{от } L_{Kate} = (N-2)\cdot A + 1 \qquad \text{до } R_{Kate} = L_{Kate} + 2A - 2.$$

Аналогично рассматривая прыжки Оли, получаем, что количество плит находится в пределах

$$\text{от } L_{Olga} = (M-2)\cdot B + 1 \qquad \text{до } R_{Olga} = L_{Olga} + 2B - 2.$$

Очевидно, что, чтобы выполнялись оба условия, количество плит должно находиться в пределах

$$\text{от } L = \max(L_{Kate}, L_{Olga}) \qquad \text{до } R = \min(R_{Kate}, R_{Olga}).$$

Если $L > R$, то решения нет, иначе плит могло быть произвольное количество от L до R. Правда, есть особый случай: если получилось $L \leqslant 0$ (это может быть, если $N = M = 1$), то надо явно заменить L на 1. Проще всего это сделать, установив $L = 1$ перед вычислением максимума.

Кроме того, отметим, что удобнее написать процедуру, обрабатывающую только одну пару чисел, а потом вызвать её два раза, передав ей сначала пару (A, N), а потом (B, M) — именно так и сделано в примере программы.

Пример правильной программы

```
var n,m,a,b:int64;
    minans,maxans:int64;
    f:text;

procedure correct(a,n:int64);
var cmin,cmax:int64;
begin
cmin:=a*(n-2)+1;
cmax:=cmin+(a-1)*2;
if minans<cmin then minans:=cmin;
if maxans>cmax then maxans:=cmax;
end;
```

```
begin
assign(f,'classics.in');reset(f);
read(f,a,b,n,m);
close(f);
maxans:=2000000000*2000000000;
minans:=1;
correct(a,n);
correct(b,m);
writeln(minans,' ',maxans);
assign(f,'classics.out');rewrite(F);
if minans>maxans then writeln(f,'0 0')
else writeln(f,minans,' ',maxans);
close(f);
end.
```

Задача 2. Лотерея

Входной файл	`lottery.in`
Выходной файл	`lottery.out`
Ограничение по времени	1 с
Ограничение по памяти	256 МиБ
Максимальный балл за задачу	100

В стране Грустьландии на железных дорогах все продаваемые билеты нумеруются последовательными натуральными числами. На этих железных дорогах проводится лотерея по номеру билета.

А именно, пассажир, покупая билет, может также приобрести за дополнительные деньги «сертификат участия в лотерее», в котором указывается номер купленного пассажиром билета; если пассажир не купил сертификат, то он не участвует

в лотерее. В начале каждого месяца проводится розыгрыш главного приза за прошедший месяц. Розыгрыш происходит следующим образом.

Пусть за прошедший месяц были приобретены билеты с номерами от L до R включительно. Тогда выбирается случайное целое число X на отрезке $[L, R]$. После этого среди всех сертификатов участия, купленных за этот месяц, выбирается сертификат, номер билета которого находится как можно ближе к X, — и обладатель этого сертификата и получает приз. Если таких сертификатов два, то главный приз делится поровну между обладателями этих сертификатов. (В частности, если к билету с номером X был куплен сертификат, то его обладатель и получит приз.)

Некий человек получил доступ к базе данных продаж сертификатов за прошедший месяц и знает, с какими билетами покупался сертификат, а с какими нет. Он понимает, что пассажиры, которые не покупали сертификат, не будут следить за результатами лотереи, и хочет воспользоваться этим, чтобы выиграть главный приз.

А именно, он может выбрать некоторые номера билетов, на которые не были куплены сертификаты, и подделать соответствующие сертификаты (в частности, внеся соответствующие изменения в базу проданных сертификатов). После этого его поддельные сертификаты будут участвовать в розыгрыше на общих условиях. Естественно, подделывать сертификаты на те билеты, на которые был куплен честный сертификат, невозможно.

Определите, какое минимальное количество сертификатов он должен подделать, чтобы максимизировать шансы и величину выигрыша. Обратите внимание, что если у него есть два способа действий, различающихся лишь тем, что при некоторых исходах лотереи в первом случае ему придётся делить выигрыш с кем-то, а во втором случае весь выигрыш достанется ему, то второй способ действий предпочтительнее, даже если в нем надо подделывать больше сертификатов.

Формат входных данных

В первой строке входного файла находятся три целых числа — L, R и N ($1 \leqslant L \leqslant R \leqslant 10^9$, $0 \leqslant N \leqslant 10^5$) — минимальный и максимальный номера проданных билетов и количество проданных сертификатов соответственно.

Во второй строке находятся N целых чисел a_i — номера билетов, для которых были проданы сертификаты ($L \leqslant a_i \leqslant R$). Номера билетов расположены строго по возрастанию.

Формат выходных данных

В первой строке выходного файла выведите минимальное количество сертификатов, которые надо подделать, чтобы максимизировать шансы выиграть.

Во второй строке выведите номера билетов, сертификаты к которым необходимо подделать, упорядоченные по возрастанию.

Пример

Входной файл	*Выходной файл*
5 20 3 10 16 17	4 9 11 15 18
1 4 2 1 4	2 2 3

Примечание: Тесты к этой задаче будут устроены следующим образом:

- тесты суммарной стоимостью 20 баллов будут иметь $L, R, N \leqslant 1000$;
- тесты суммарной стоимостью 30 баллов будут иметь $L, R, N \leqslant 100\,000$, при этом как минимум одно из значений будет больше 1000;
- тесты суммарной стоимостью 50 баллов не будут попадать ни под одно из указанных выше условий.

Решение

Чтобы решить данную задачу, необходимо было догадаться до простой идеи, что чтобы максимизировать выигрыш, необходимо подделать сертификаты таким образом, чтобы «окружить» блоки из подряд идущих номеров билетов, для которых были куплены сертификаты. Т.е. для каждого купленного сертификата требуется подделать сертификат на билет с номером на 1 больше и на 1 меньше, если только сертификат на такой билет ещё не был куплен. Докажем данный факт.

Пусть для некоторого билета с номером i был куплен сертификат, а для одного из соседних (для определённости пусть это будет билет с номером $i+1$) сертификат не был куплен и злоумышленник сертификат для него не подделал. В этом случае, если в ходе проведения лотереи выпало число $i + 1$, то злоумышленник вообще ничего не получит (если не был подделан сертификат для билета с номером $i+2$, или билета с таким номером не существует), либо поделит приз с обладателем билета i (если был подделан сертификат для билета с номером $i + 2$). Таким образом, доказана необходимость подделывать сертификаты для всех билетов, соседних с теми, для которых сертификаты были куплены.

Покажем, что других сертификатов подделывать не надо. Пусть в ходе лотереи выпало число i. Тогда возможны 3 варианта:

- если для билета с номером i сертификат был куплен, то победителем является купивший его человек, и злоумышленник повлиять на это никак не может;
- если для билета с номером i сертификат был подделан, то победителем является злоумышленник;
- если не выполнено ни одно из перечисленных выше условий, то ближайшим билетом, для которого есть сертификат, является подделанный билет (так как купленные сертификаты «окружены» подделанными) и победителем является злоумышленник.

Таким образом, злоумышленник во всех трёх случаях либо оказывается единоличным победителем, либо повлиять на выбор победителя не может. Оптимальная стратегия выбора сертификатов для подделывания доказана. Остаётся её реализовать.

Проще всего это сделать следующим образом. Будем хранить два последовательных номера билетов, для которых были куплены сертификаты: «предыдущий» pr и «текущий» cur. Если номера являются соседними, т.е $pr + 1 = cur$, то подделки никакого сертификата не требуется. В противном случае, необходимо подделать сертификат для билета с номером $cur - 1$. Если, при этом, $pr + 1 \neq cur - 1$, то необходимо в ответ добавить ещё билет с номером $pr + 1$. Таким алгоритмом будут найдены все номера, для которых необходимо подделать сертификаты, лежащие между парами номеров с купленными сертификатами. Остаётся добавить номер

на единицу меньше первого купленного сертификата и номер на единицу больше последнего купленного сертификата, если они не выходят за границу купленных номеров.

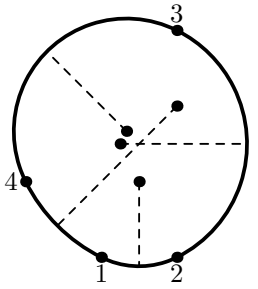
Остаётся полученные номера отсортировать, либо изначально добавлять в ответ номера в порядке возрастания, что делается элементарно.

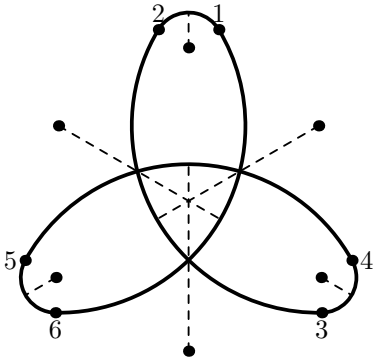
Пример правильной программы

```
const maxn=100000;
var f:text;
    l, r, n, i, pr, ans_len, cur:integer;
    ans:array[1..2 * maxn] of integer;

begin
  assign(f,'lottery.in'); reset(f);
  read(f, l, r, n);

  read(f, pr);
  ans_len := 0;
  if pr - 1 >= l then begin
    ans_len := ans_len + 1;
    ans[ans_len] := pr - 1;
  end;
  for i := 1 to n - 1 do begin
    read(f, cur);

    if pr + 1 <> cur then begin
      if pr + 1 <> cur - 1 then begin
        ans_len := ans_len + 1;
        ans[ans_len] := pr + 1;
      end;
      ans_len := ans_len + 1;
      ans[ans_len] := cur - 1;
    end;
    pr := cur;
  end;
  if pr <> r then begin
    ans_len := ans_len + 1;
    ans[ans_len] := pr + 1;
  end;
  close(f);

  assign(f, 'lottery.out'); rewrite(f);
  writeln(f, ans_len);
  for i := 1 to ans_len do begin
    writeln(f, ans[i], ' ');
  end;
  close(f);
end.
```

Задача 3. Рисование кругами

Входной файл	circles.in
Выходной файл	circles.out
Ограничение по времени	1 с
Ограничение по памяти	256 МиБ
Максимальный балл за задачу	100





Маленькая девочка Оля очень любит пони. И очень любит рисовать. Поэтому она была очень рада, когда родители подарили ей на день рождения раскраску с пони. Причём, поскольку Оля уже большая девочка и недавно научилась считать, то эта раскраска необычная — в ней на каждой странице просто отмечены несколько точек, занумерованных последовательными натуральными числами. Прежде чем начать раскрашивать, надо эти точки последовательно соединить: первую со второй, вторую с третьей и так далее, предпоследнюю с последней, а последнюю — с первой.

Оля принялась за работу. Но, поскольку бабушка подарила Оле на тот же день рождения набор для рисования, и Оле там очень понравился циркуль, то она хочет соединять точки

с помощью циркуля. А циркулем, как известно, можно нарисовать только дуги окружностей. Поэтому Оля хочет нарисовать одну дугу окружности от первой точки ко второй, ещё одну — от второй к третьей, и т.д.

Оля уже начала рисовать, но обнаружила проблему: в местах стыка двух последовательных дуг окружностей получаются острые углы, а Оля очень не любит углы. Поэтому она хочет, чтобы окружности стыковались как можно плавнее. Строго говоря (хотя Оля и не знает таких слов), она хочет, чтобы у каждой пары последовательных дуг окружностей в их общем конце (в точке, изначально отмеченной в раскраске) была общая касательная; и более того, чтобы ни в какой точке не происходило изменения направления движения на противоположное (т.е. не допустимы фрагменты типа такого: $\prec$).

Помогите ей: определите, какие точки Оля должна использовать в качестве центров дуг.

Формат входных данных

В первой строке входного файла находится одно число n — количество точек на очередной странице раскраски. Далее следуют n строк, i-я из которых содержит два вещественных числа x_i и y_i — координаты очередной точки.

Гарантируется, что $3 \leqslant n \leqslant 50\,000$, и что координаты точек не превосходят по модулю 10 000. Гарантируется, что расстояние между любыми двумя последовательными точками не меньше 1, и что расстояние между n-ой и первой точкой также не меньше 1.

Формат выходных данных

Если решение существует, то в первую строку выходного файла выведите одно слово "`yes`" (без кавычек), а далее выведите n строк. На i-ую из них выведите два числа: координаты центра дуги, соединяющей i-ую и $(i+1)$-ую точку (n-ая строка должна содержать координаты центра дуги, соединяющей n-ую и первую точки), а затем один символ: "`+`", если дугу от i-ой точки до $(i+1)$-ой (для последней строки — от n-ой до первой) следует рисовать против часовой стрелки, и "`-`", если по часовой стрелке. Разделяйте числа между собой, а также отделяйте их от символа пробелом.

Если решения не существует, то выведите в выходной файл одно слово "`no`" (без кавычек).

Если существует несколько решений, выведите любое.

Гарантируется, что если решение существует, то существует решение, в котором координаты центров окружностей не превосходят по модулю 1 000 000. (Но вы не обязаны искать именно такое решение.) Также гарантируется, что если решения не существует, то при достаточно малом сдвиге данных точек оно не появляется.

При проверке вашего ответа все сравнения будут проверяться с абсолютной точностью 0.01.

Обратите внимание, что решение вполне может быть самопересекающимся.

Пример

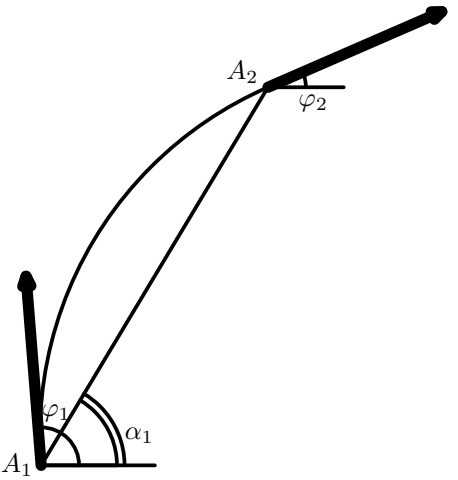
Входной файл	*Выходной файл*
4 0 0 2 0 2 6 -2 2	yes 1 2 + 0.5 3 + 0.66666 3.33334 + 2 4 +
6 0.69458 3.93921 -0.69458 3.93921 3.06415 -2.57117 3.75879 -1.3681 -3.75879 -1.3681 -3.06415 -2.57117	yes 0 3.52246 + 2.99188 1.72736 + 3.05054 -1.76123 + 0 -3.4544 + -3.05054 -1.76123 + -2.99188 1.72736 +

Примечание: Примеры соответствуют рисункам. На каждом рисунке точками с цифрами обозначены изначально нарисованные в раскраске точки. Точками без цифр обозначены центры дуг окружностей; кроме того, из каждого центра пунктиром проведён радиус до некоторой точки соответствующей дуги, чтобы было понятно, какой центр к какой дуге относится.

Примечание: Среди тестов будут, в том числе, тесты с маленьким n, а именно, с $n = 3, 4, 5, 6$. Для каждого из этих значений n суммарная стоимость соответствующих тестов будет 15 баллов.

Решение

Будем называть полярным углом вектора угол между направлением оси x и направлением этого вектора. Углы будем считать ориентированными: если от направления оси x к направлению вектора вращение происходит против часовой стрелки, то угол будем считать положительным, иначе отрицательным. Полярный угол будем считать определённым с точностью до слагаемого, кратного 2π (здесь и далее все углы измеряем в радианах), т.е., например, будем считать полярные углы $\pi/4 - 2\pi$, $\pi/4$ и $\pi/4 + 2\pi$ совпадающими.



Обозначим данные нам точки A_1, A_2, ..., A_n. Обозначим полярный угол вектора $\overrightarrow{A_1A_2}$ как α_1 (см. рис), полярный угол вектора $\overrightarrow{A_2A_3}$ как α_2 и т.д.

Пусть у нас имеется некоторое решение задачи. Построим в каждой точке A_i вектор, касательный к окружностям, и

направленный по направлению движения от точки A_i к A_{i+1} (см. рис.); обозначим полярный угол этого вектора как φ_i.

Из несложных геометрических соображений можно получить, что

$$\varphi_{i+1} = 2\alpha_i - \varphi_i.$$

Тогда выразим все углы последовательно через φ_1:

$$\begin{aligned}
\varphi_2 &= 2\alpha_1 - \varphi_1, \\
\varphi_3 &= 2\alpha_2 - 2\alpha_1 + \varphi_1, \\
&\dots, \\
\varphi_{n+1} &= 2(\alpha_n - \alpha_{n-1} + \dots \mp \alpha_1) \pm \varphi_1
\end{aligned}$$

в последней строке выбор знака $+$ или $-$ зависит от четности n, а φ_{n+1} обозначает полярный угол касательной к последней окружности в точке A_1.

Конечно, должно выполняться $\varphi_{n+1} = \varphi_1$, откуда получаем уравнение на φ_1:

$$\varphi_1 = 2(\alpha_n - \alpha_{n-1} + \dots \mp \alpha_1) \pm \varphi_1.$$

В случае нечетного n в правой части перед φ_1 будет стоять знак $-$, откуда получаем

$$2\varphi_1 = 2(\alpha_n - \alpha_{n-1} + \dots + \alpha_1).$$

С учетом того, что все углы у нас определены с точностью до слагаемого, кратного 2π, это уравнение имеет два решения:

$$\varphi_1 = \alpha_n - \alpha_{n-1} + \dots + \alpha_1 \qquad \text{и} \quad \varphi_1 = \alpha_n - \alpha_{n-1} + \dots + \alpha_1 + \pi.$$

Несложно показать, что оба этих решения дают решение задачи. А именно, исходя из φ_1 и α_1 несложно вычисляется радиус дуги окружности, соединяющей A_1 и A_2 и дающей нужное значение φ_1, после чего эта дуга легко строится. Далее аналогично строится вторая дуга, и т.д.

Таким образом, при нечетном n всегда существуют два решения (точнее, не совсем всегда, см. последний абзац разбора). Несложно заметить, что одно решение получается из другого заменой каждой дуги на ее дополнение до полной окружности.

В частности, при $n = 3$ одно из решений — описанная вокруг треугольника окружность.

В случае же четного n в правой части уравнения на φ_1 будет стоять знак $+$. В итоге, φ_1 уничтожится и останется равенство

$$2(\alpha_n - \alpha_{n-1} + \dots - \alpha_1) = 0.$$

Очевидно, это условие существования решения: решение существует тогда и только тогда, когда равенство выполняется. При этом не следует забывать, что все углы у нас определены с точностью до слагаемого, кратного 2π, т.е. это равенство следует понимать как условие, что знакочередующаяся сумма $\alpha_n - \alpha_{n-1} + \dots - \alpha_1$ кратна π.

(В частности, из этого условия следует, что при $n = 4$ решение существует тогда и только тогда, когда данные четыре точки лежат на одной окружности.)

Более того, если это условие выполняется, то в качестве решения можно взять любое φ_1, и исходя из него построить решение аналогично описанному выше процессу.

Тонкости реализации алгоритма в этом разборе рассматриваться не будут. Отметим лишь, что если φ_1 будет выбрано неудачно, то при некотором i окажется, что $\varphi_i = \alpha_i$ или $\varphi_i = \alpha_i + \pi$, и соответствующей окружности не будет существовать (она выродится в прямую). При четном n это исправляется выбором другого φ_1 (а точнее, можно выбрать некоторое случайное φ_1 и с крайне высокой вероятностью этот выбор будет хорошим); а при нечетном n такая ситуация невозможна по условию: решение здесь не существует, но при небольшом сдвиге точек появляется.

Пример правильной программы

```
uses math;
var x,y:array[1..100001] of extended;
    xx,yy:extended;
    nx,ny:extended;
    n:integer;
    f:text;
    angle,angle_i,a:extended;
    i:integer;
    d:extended;

function vect(x1,y1,x2,y2:extended):extended;
begin
vect:=x1*y2-x2*y1;
end;

begin
assign(f,'circles.in');reset(f);
read(f,n);
for i:=1 to n do read(f,x[i],y[i]);
x[n+1]:=x[1];
y[n+1]:=y[1];
close(f);
angle:=0;
for i:=1 to n do begin
    angle_i:=arctan2(y[i+1]-y[i], x[i+1]-x[i]);
    if i mod 2 =1 then
       angle:=angle+angle_i
    else angle:=angle-angle_i;
end;
if n mod 2=0 then begin
   while angle>pi/2 do
         angle:=angle-pi;
   while angle<-pi/2 do
         angle:=angle+pi;
   if abs(angle)>1e-6 then begin
      assign(f,'circles.out');rewrite(f);
      writeln(f,'no');
      close(f);
      halt;
   end;
   angle:=137; // chosen by fair dice roll
end;
// if n mod 2 = 1, we should start
// directly from calculated angle
assign(f,'circles.out');rewrite(f);
writeln(f,'yes');
for i:=1 to n do begin
    angle_i:=arctan2(y[i+1]-y[i],x[i+1]-x[i]);
    a:=angle_i-(angle-pi/2);
    d:=sqrt(sqr(x[i+1]-x[i])+sqr(y[i+1]-y[i]));

    xx:=(x[i+1]+x[i])/2;
    yy:=(y[i+1]+y[i])/2;

    nx:=(y[i+1]-y[i])/d;
    ny:=-(x[i+1]-x[i])/d;

    d:=d*tan(a)/2;
    xx:=xx+nx*d;
    yy:=yy+ny*d;
    write(f,xx:20:20,' ',yy:20:20);
    if (vect(x[i]-xx, y[i]-yy,
           cos(angle), sin(angle)) > 0) then
        writeln(f,' +')
    else writeln(f,' -');

    angle:=2*angle_i-angle;
end;
close(f);
end.
```

Задача 4. Кастрюли и крышки

Входной файл	`pots.in`
Выходной файл	`pots.out`
Ограничение по времени	1 с
Ограничение по памяти	256 МиБ
Максимальный балл за задачу	100

Страшный беспорядок произошёл сегодня утром в столовой школы, где вы учитесь. Мария Ивановна, уборщица, подметая полы, уронила один из шкафов, и вся находившаяся в нем посуда разлетелась по всей столовой. К счастью, в этом шкафу хранились лишь металлические кастрюли с крышками. Однако, при падении некоторые из них погнулись, и их пришлось выбросить.

Теперь директор школы хочет подсчитать ущерб, и понять, сколько же новых кастрюль или крышек придётся закупать. Но сначала она хочет выяснить, а сколько же уцелевших кастрюль можно накрыть оставшимися крышками?

Кастрюли и крышки в сечении являются окружностями. Крышкой можно накрыть кастрюлю тогда и только тогда, когда радиус крышки больше либо равен радиуса кастрюли.

Формат входных данных

В первой строке входного файла находятся два целых числа n и m ($1 \leqslant n \leqslant 1000$, $1 \leqslant m \leqslant 1000$) — количество оставшихся кастрюль и крышек соответственно.

Во второй строке находятся n целых чисел a_i — радиусы оставшихся кастрюль ($1 \leqslant a_i \leqslant 1000$).

В третьей строке находятся m целых чисел b_i — радиусы оставшихся крышек ($1 \leqslant b_i \leqslant 1000$).

Формат выходных данных

Выведите единственное число — максимальное количество кастрюль, которые можно накрыть имеющимися крышками.

Пример

Входной файл	*Выходной файл*
5 5 4 8 1 2 5 7 2 4 6 5	4

Решение

Данная задача решается с помощью «жадного» алгоритма. Очевидно, что если одну кастрюлю можно накрыть двумя разными по размеру крышками, то выгоднее использовать меньшую из них, так как большей можно попытаться накрыть кастрюлю большего диаметра.

Используя эту идею, задачу легко решить алгоритмом, работающим за время, пропорциональное произведению количества кастрюль и крышек. А именно, можно последовательно перебирать все кастрюли, и для каждой из них искать крышку наименьшего радиуса, которая её покрывает и ещё не была использована. Такое

решение пройдёт все тесты и наберёт максимальный балл, так как ограничения на количество кастрюль и крышек в задаче небольшие.

Однако, существует и более быстрый алгоритм. Если сначала отсортировать крышки и кастрюли в порядке увеличения радиуса, то потом можно последовательно перебирать кастрюли и соответствующие им крышки. Причём, если для кастрюли с номером i подошла крышка с номером j, то перебор крышек для кастрюли с номером $i+1$ надо начинать с крышки с номером $j+1$, так как в силу упорядоченности по возрастанию меньшие точно не подойдут. Поэтому, достаточно хранить указатели на текущую рассматриваемую кастрюлю и крышку, и тогда время поиска ответа будет пропорционально максимуму из количества кастрюль и крышек.

Если для сортировки использовать алгоритм быстрой сортировки или, например, сортировки кучей, которые работают за время $O(n \log n)$, то такое решение будет работать даже если количество крышек и кастрюль порядка 100 000 и более. Именно такое решение и приведено ниже.

Пример правильной программы

```
const MAX = 1000;
type TArray = array[1..MAX] of integer;
procedure sort(var a:TArray; l,r:integer);
var i, j, m, t:integer;
begin
  i:=l; j:=r; m:=a[l+random(r-l+1)];
  repeat
    while a[i]<m do inc(i);
    while a[j]>m do dec(j);
    if (i<=j) then begin
      t:=a[i]; a[i]:=a[j]; a[j]:=t;
      inc(i); dec(j);
    end;
  until i>j;
  if i<r then sort(a,i,r);
  if l<j then sort(a,l,j);
end;

var
    f: text;
    n, m, i, ans, p_ind, c_ind: integer;
    pots, covers: TArray;
begin
  assign(f, 'pots.in');reset(f);
  readln(f, n, m);
  randseed:=n+m;
  for i := 1 to n do
      read(f, pots[i]);
  for i := 1 to m do
      read(f, covers[i]);
  sort(pots, 1, n);
  sort(covers, 1, m);
  p_ind := 1; c_ind := 1; ans := 0;
  while (p_ind <= n) and (c_ind <= m) do
  begin
    if covers[c_ind] >= pots[p_ind] then
    begin
      inc(ans);
      inc(p_ind);
      inc(c_ind);
    end
    else begin
      inc(c_ind);
    end;
  end;
  assign(f, 'pots.out'); rewrite(f);
  write(f, ans); close(f);
end.
```

Задача 5. Сумма степеней

Входной файл	powersum.in
Выходной файл	powersum.out
Ограничение по времени	1 с
Ограничение по памяти	256 МиБ
Максимальный балл за задачу	100

Ваш маленький брат уже учится в девятом классе, и вовсю интересуется арифметикой и алгеброй. Например, он давно знает, как вычислить сумму

$$1^2 + 2^2 + \ldots + n^2.$$

Ему теперь хочется научиться вычислять аналогичную сумму для произвольного показателя степени:

$$S_{nk} = 1^k + 2^k + \ldots + n^k.$$

Правда, он не любит возиться с большими числами, зато ему очень нравится операция взятия остатка от деления одного числа на другое. Напишите программу, которая по данным n, m и k вычислит остаток от деления S_{nk} на m.

Формат входных данных

В первой строке входного файла находятся три целых числа — n, m и k ($1 \leqslant n \leqslant 10^9$, $1 \leqslant m \leqslant 10^9$, $1 \leqslant k \leqslant 20$).

Формат выходных данных

Выведите единственное число — остаток от деления S_{nk} на m.

Пример

Входной файл	*Выходной файл*
3 1000 5	276
3 4 5	0

Примечание: Среди тестов будут тесты с $n \leqslant 10^5$ общей стоимостью 30 баллов, и тесты с $n > 10^5$, но $m \leqslant 50\,000$, общей стоимостью ещё 30 баллов.

Решение

Эту задачу можно решать разными способами, опишем один из них.

Разобьём все слагаемые в S_{nk} на чётные и нечётные. Рассмотрим сумму чётных слагаемых. Выделив в каждом слагаемом явно множитель 2, сведём эту сумму к более простой задаче:

$$2^k + 4^k + \ldots = 2^k \cdot (1^k + 2^k + \ldots) = 2^k \cdot S_{\lfloor \frac{n}{2} \rfloor, k},$$

где $\lfloor x \rfloor$ обозначает целую часть x.

Сумму нечётных слагаемых преобразуем чуть более сложно. А именно, разложим каждое слагаемое $(2p+1)^k$ по формуле бинома Ньютона:

$$(2p+1)^k = (2p)^k + C_k^1 \cdot (2p)^{k-1} + C_k^2 \cdot (2p)^{k-2} + \ldots + C_k^{k-1} \cdot 2p + 1;$$

здесь C_n^m — число сочетаний из n по m (биномиальный коэффициент).

Сумма всех таких слагаемых теперь вычисляется легко. Обозначая для сокращения записи $x = \lfloor (n-1)/2 \rfloor$ — максимальное значение p при данном n, — имеем:

$$1^k + 3^k + 5^k + \ldots + (2x+1)^k = 2^k \cdot S_{xk} + C_k^1 \cdot 2^{k-1} S_{x,k-1} + \ldots + C_k^{k-1} \cdot 2 S_{x,1} + x.$$

Эти две формулы позволяют легко свести задачу с данными n и k к нескольким задачам с меньшими параметрам. Эти задачи также можно свести к ещё более мелким и т.д., т.е. можно написать рекурсивный алгоритм, который будет решать задачу. Для получения полного решения необходимо ещё учесть, что среди тех задач, которые такой алгоритм будет решать, будет много повторяющихся, поэтому

надо запоминать уже решённые задачи и не решать одну и ту же задачу два раза. Естественно, все вычисления следует производить по модулю m.

Отметим также, что в этой задаче можно было написать частичные решения. Во-первых, при маленьком n можно было требуемую сумму посчитать простым циклом. Во-вторых, при достаточно маленьком m можно было воспользоваться тем, что у чисел, различающихся на m, остатки степеней по модулю совпадают: $(i+m)^k$ и i^k имеют одинаковый остаток по модулю m для любых i и k. Поэтому при маленьких m можно было вычислить только первые m слагаемых суммы, а дальше использовать периодичность.

Пример правильной программы

```
const maxans=1000;
var n,k,m:integer;
    c:array[0..20,-1..20] of integer;
    f:text;
    i,j:integer;
    ans:array[1..maxans] of record n,k:integer;
                                   ans:integer; end;
    nans:integer;

function pow(x:int64; k:int64; m:int64):int64;
var i:integer;
begin
result:=1;
for i:=1 to k do result:=(result*x) mod m;
end;

function calc(n,m,k:int64):int64;
var n0,i:integer;
    corr:integer;
begin
if n=0 then exit(0);
if n=1 then exit(1 mod m);
if k=0 then exit(n mod m);
result:=0;
corr:=0;
n0:=n;
if (n mod 2 =0) then begin
   inc(n);
   corr:=m-pow(n,k,m);
end;
for i:=1 to nans do
    if (ans[i].n=n)and(ans[i].k=k) then begin
       result:=(corr+ans[i].ans) mod m;
       exit;
    end;
// now n div 2 = (n-1) div 2
result:=(result +
   (2 * pow(2,k,m) * calc(n div 2, m, k))
   mod m) mod m;
for i:=k-1 downto 0 do
    result:=(result +
      (c[k,i] * (pow(2,i,m) *
      calc((n-1) div 2, m, i)
      mod m) mod m)) mod m;
result:=(result+1) mod m;
inc(nans);
ans[nans].n:=n;
ans[nans].k:=k;
ans[nans].ans:=result;
result:=(result+corr) mod m;
end;

begin
assign(f,'powersum.in');reset(f);
read(f,n,m,k);
close(f);
fillchar(c,sizeof(c),0);
c[0,0]:=1;
for i:=1 to k do
    for j:=0 to k do
        c[i,j]:=(c[i-1,j-1]+c[i-1,j]) mod m;
nans:=0;
assign(f,'powersum.out');rewrite(f);
writeln(f,calc(n,m,k));
close(f);
end.
```

Задача 6. Расписание поездов

Входной файл	trains.in
Выходной файл	trains.out
Ограничение по времени	1 с
Ограничение по памяти	256 МиБ
Максимальный балл за задачу	100

Ваш друг все ещё работает главным диспетчером в компании, владеющей сетью маршрутных такси. Благодаря вашей помощи он сумел весьма эффективно оптимизировать расписание маршруток, и на проходившем недавно бизнес-форуме он выступал с докладом, посвящённым как раз этой оптимизации. После доклада к

нему подошли представители одной из железнодорожных компаний и попросили помочь с оптимизацией расписания движения поездов их компании.

Их компания обслуживает несколько железнодорожных маршрутов. По каждому маршруту каждый день должен следовать поезд, содержащий определённое количество вагонов: а именно, в одно и то же время каждый день поезд должен отправляться с начальной станции и в определённое время в этот или на следующий день прибывать на конечную станцию. Времена отправления и прибытия давно согласованы с администрацией железнодорожных станций, да и вообще с клиентами этой компании, поэтому они жёстко фиксированы. Количество вагонов в каждом поезде также жёстко фиксировано.

Компания перевозит в основном однотипный груз, и поэтому все вагоны, которые у неё имеются, абсолютно одинаковы: любой вагон можно поставить в любой поезд. Более того, когда некоторый поезд прибывает на конечную станцию, этот поезд можно расформировать и соответствующие вагоны использовать в других поездах, отправляющихся с этой станции в тот же момент или позже (в этот же день или на следующий день).

Соответственно, перед компанией стоит задача оптимизации вагонного парка: они хотят оставить у себя минимальное количество вагонов, которых будет достаточно, чтобы обслуживать все маршруты.

Помогите им и вашему другу определить это минимальное количество.

Формат входных данных

Первая строка входного файла содержит два натуральных числа N и M: количество станций на железной дороге и количество маршрутов, которые обслуживает компания, соответственно. Гарантируется, что оба числа не превосходят 100 000.

Далее следуют M строк, каждая из которых содержит описание очередного маршрута. А именно, i-я строка содержит сначала три натуральных числа s_i, f_i и n_i: номера начальной и конечной станции этого маршрута ($1 \leqslant s_i, f_i \leqslant N$) и количество вагонов, которые должны быть в поездах этого маршрута ($1 \leqslant n_i \leqslant 100\,000$). Далее в той же строке в формате `hh:mm` заданы времена отправления и прибытия поездов этого маршрута (где `hh` — количество часов, а `mm` — количество минут). Если первый момент времени меньше второго, то поезд должен прибывать в тот же день, если же второй момент меньше первого, то это обозначает, что поезд должен прибывать на следующий день в указанное время.

Гарантируется, что для каждого поезда заданные времена отправления и прибытия не совпадают. Гарантируется, что для каждой станции количество вагонов, прибывающих на эту станцию за день, в точности равно количеству вагонов, отправляющихся с этой станции за день.

Считайте, что вагоны, прибывшие на станцию в некоторый момент, можно отправлять с поездами, которые отправляются в этот же момент времени или позже.

Формат выходных данных

В выходной файл выведите одно число — минимальное количество вагонов, которые надо иметь железнодорожной компании, чтобы иметь возможность обслуживать все маршруты.

Пример

Входной файл	*Выходной файл*
2 2 1 2 5 06:00 13:00 2 1 5 16:00 17:00	5
3 4 3 1 3 23:59 00:01 1 2 15 06:00 13:00 2 1 12 16:00 04:00 2 3 3 17:00 23:59	15

Примечание: В первом примере достаточно иметь пять вагонов, которые каждый день будут сначала отправляться с первой станции на вторую с поездом первого маршрута, а потом возвращаться с поездом второго маршрута.

Во втором примере достаточно иметь 15 вагонов. Каждый день в 6 утра все пятнадцать вагонов в составе поезда второго маршрута будут отправляться с первой станции на вторую. Далее в 16:00 двенадцать из этих вагонов возвращаются обратно на первую станцию с поездом третьего маршрута. Три оставшиеся вагона в 17:00 отправляются на третью станцию с поездом четвёртого маршрута, куда прибывают в 23:59, и тут же отправляются с поездом первого маршрута обратно на первую станцию.

Решение

Данная задача решалась весьма легко, главное было подойти к ней с правильной стороны.

А именно, определим для каждой станции количество вагонов, которые надо иметь на этой станции в полночь (точнее, за мгновение до полуночи). Для этого отсортируем все поезда, приходящие на эту станцию и отправляющиеся с неё, по времени; при равных временах будем прибывающие поезда учитывать до отправляющихся. После этого пробежимся по отсортированному массиву и будем вычислять для каждого момента времени следующую величину: суммарный «баланс» вагонов, т.е. общее количество прибывших вагонов минус общее количество убывших вагонов. Очевидно, что если в какой-то момент этот баланс становится отрицательным, то искомое количество вагонов, которые нужны на станции до полуночи, равно максимальному модулю среди отрицательных балансов. Если же балансы все время остаются положительными, то это количество равно нулю.

После того, как мы для каждой станции определили нужное количество вагонов, их остаётся просуммировать и добавить те вагоны, которые в полночь находятся в пути. Полученное значение и будет ответом.

Отметим, что, т.к. минут в сутках всего 1440, то вагоны можно сортировать подсчётом, т.е. просто для каждого момента времени вычислять, сколько в этот момент прибывает вагонов на станцию, и сколько убывает. Именно такой вариант и реализован в примере решения.

Пример правильной программы

```
const max_time=24*60-1;
type ttrain=record s,f,t1,t2,n:integer; end;
     ttrainArr=array of ttrain;
var f:text;
    n,m:integer;
    train:array[1..100000] of ttrain;
    arrives, departs:array [0..max_time] of
                                  ttrainArr;
    min,bal:array[1..100000] of int64;
    ans:int64;
    i,j:integer;
    curs:integer;

function readnum:integer;
var ch:char;
begin
repeat
  read(f,ch);
until ch in ['0'..'9'];
result:=0;
while ch in ['0'..'9'] do begin
      result:=result*10+ord(ch)-ord('0');
      read(f,ch);
end;
end;

function readmom:integer;
begin
result:=readnum*60+readnum;
end;

procedure push(var a:ttrainArr;const t:ttrain);
begin
setlength(a, length(a)+1);
a[length(a)-1]:=t;
end;

begin
assign(f,'trains.in');reset(f);
read(f,n,m);
ans:=0;
for i:=1 to m do begin
    read(f,train[i].s,train[i].f,train[i].n);
    train[i].t1:=readmom;
    train[i].t2:=readmom;
    if train[i].t2<train[i].t1 then
       ans:=ans+train[i].n;
    push(arrives[train[i].t2], train[i]);
    push(departs[train[i].t1], train[i]);
end;
close(f);
fillchar(bal,sizeof(bal),0);
fillchar(min,sizeof(min),0);
for i:=0 to max_time do begin
    for j:=0 to length(arrives[i])-1 do
        inc(bal[arrives[i][j].f],
            arrives[i][j].n);
    for j:=0 to length(departs[i])-1 do begin
        curs:=departs[i][j].s;
        dec(bal[curs], departs[i][j].n);
        if bal[curs]<min[curs] then
           min[curs]:=bal[curs];
    end;
end;
for i:=1 to n do begin
    if min[i]<0 then
       inc(ans, -min[i]);
end;
assign(f,'trains.out');rewrite(f);
writeln(f,ans);
close(f);
end.
```

Содержание