Департамент образования администрации города Нижнего Новгорода
Городской ресурсный центр физико-математического образования
Нижегородский государственный университет им. Н. И. Лобачевского

Тринадцатая нижегородская городская олимпиада школьников по информатике имени В. Д. Лелюха

18 февраля 2017 г.

Нижний Новгород
2017

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.

Тринадцатая городская олимпиада по информатике им. В. Д. Лелюха

18 февраля 2017 г. Городской ресурсный центр физико-математического образования в лице учредителей: Департамента образования администрации города Нижнего Новгорода, Нижегородского государственного университета им. Н.И. Лобачевского и МБОУ Лицей № 40 проводит тринадцатую городскую олимпиаду по информатике им. В.Д. Лелюха среди учащихся 8–11 классов образовательных учреждений города Нижнего Новгорода.

Целью проведения олимпиады является поиск талантливой молодёжи, привлечение её в науку, повышение уровня преподавания предметов физико-математического цикла.

Спонсорами городской олимпиады школьников по информатике являются НПП «Прима», компания «Яндекс», ННГУ им. Н.И. Лобачевского.

Тринадцатая городская олимпиада проводится в ННГУ им. Лобачевского на базе Института информационных технологий, математики и механики (корпус 2).

Олимпиада проводится в соответствии с Положением о городской олимпиаде по информатике (Приказ № 633 от 06.09.2016 Департамента образования администрации г. Нижнего Новгорода).

Для участия в городской олимпиаде приглашаются талантливые школьники из Нижегородской области.

Регламент проведения олимпиады

9:30—10:00	Регистрация участников
10:00—10:30	Открытие олимпиады, информация от жюри
10:45—15:45	Решение задач олимпиады
15:45—17:00	Обед
16:00—17:00	Работа жюри
17:00—18:30	Приветствие участников олимпиады, выступления учредителей, спонсоров. Подведение итогов олимпиады. Поздравление победителей и призёров.

Состав оргкомитета олимпиады

Сидоркина С. Л., заместитель директора департамента образования администрации города Нижнего Новгорода;
Авралев Н. В., проректор по связям с общественностью ННГУ им. Н.И. Лобачевского;
Умнова Н. С., директор муниципального бюджетного образовательного учреждения лицей № 40;
Смирнов А. А., заместитель директора МБОУ лицей № 40;
Тинькова Е. В., консультант отдела общего среднего образования департамента образования администрации города Нижнего Новгорода;
Гашпар И. Л., куратор классов НОЦ ИПФ РАН.

Состав предметной комиссии (жюри)

Председатель — Калинин П. А., старший разработчик, ООО «Яндекс», к.ф.-м.н.;

Члены комиссии:
Борисов Н. А., доцент кафедры программной инженерии ИТММ ННГУ, к.т.н;
Жидков Н. В., студент 2 курса СПб АУ;
Калинин Н. А., студент 3 курса ВШОПФ ННГУ;
Кривоносов М. И., студент 1 курса магистратуры ИИТММ ННГУ;
Кузьмичев Д. А., студент 4 курса МФТИ;
Матросов М. В., инженер-программист, НПП «АВИАКОМ»;
Сорокин А. А., студент 3 курса ВШОПФ ННГУ;
Шмелёв А. С., инженер-программист, НПП «ПРИМА».

Иллюстрации в условиях: Анна Тендитная, студентка 3 курс ВШОПФ ННГУ.

Ниже приведены примеры текстов программ, написанных на разрешённых языках программирования. Программы считывают данные (два числа в одной строке) с клавиатуры и выводят на экран их сумму:

Pascal

```
var a,b:integer;
begin
  read(a,b);
  writeln(a+b);
end.
```

C/C++

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
    cin >> a >> b;
    cout << a + b << endl;
    return 0;
}
```

C#

```
using System;
class Program {
static void Main(string[] args) {
    string[] s = Console.ReadLine().
        Split();
    int n = Int32.Parse(s[0]);
    int m = Int32.Parse(s[1]);
    Console.Write(n + m);
}
}
```

Python 3

```
a, b = map(int, input().split())
print(a + b)
```

XIII Городская олимпиада школьников по информатике 18 февраля 2017 г.

Задача 1. A+B

Ввод	`aplusb.in` или стандартный ввод
Вывод	`aplusb.out` или стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт
Максимальный балл за задачу	200

Напишите программу, которая складывает два числа.

Формат входных данных

Во входном файле находятся два числа, A и B. Числа целые и не превосходят по модулю 1000.

Формат выходных данных

Выведите в выходной файл одно число — сумму двух данных чисел.

Пример

Ввод	*Вывод*
2 2	4
-4 2	-2

Решение

Данная задача не представляла участникам никаких сложностей; примеры решения этой задачи были приведены в памятке участника.

Задача 2. Кефир

Ввод	`kefir.in` или стандартный ввод
Вывод	`kefir.out` или стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт
Максимальный балл за задачу	100

Маленькая девочка Оля очень любит кефир. Каждый день она выпивает ровно k пакетов кефира, если их имеется хотя бы k, в противном случае — все, что есть. Но есть одна проблема: сроки годности. На каждом пакете написан день, позже которого этот пакет пить нельзя (ровно в этот день еще можно). Поэтому если в холодильнике у Оли в какой-то момент оказывается просроченный пакет, она его выкидывает.

Оля очень не любит выкидывать пакеты с кефиром, поэтому она применяет следующую стратегию: каждый раз, когда надо выпить очередной пакет, она выбирает тот, у которого срок годности заканчивается раньше всего. Несложно видеть, что эта стратегия минимизирует количество выкинутых пакетов и, в частности, позволяет обойтись без выкидывания, если это вообще возможно.

Но основная проблема, с которой столкнулась Оля, — это покупка нового кефира. Сейчас у Оли в холодильнике есть n пакетов кефира, про каждый известен его срок годности (через сколько дней он заканчивается). Оля пришла в магазин, и там есть m пакетов кефира, про каждый тоже известен его срок годности.

Определите, какое максимальное количество пакетов может купить Оля так, чтобы потом не пришлось ничего выбрасывать. Считайте, что Оля сегодня еще не пила ни одного пакета кефира.

Формат входных данных

В первой строке находятся три целых числа n, m, k ($1 \leqslant n, m \leqslant 10^6$, $1 \leqslant k \leqslant n + m$) — число пакетов кефира у Оли в холодильнике, число пакетов кефира в магазине и сколько пакетов выпивает Оля каждый день.

Во второй строке находятся n целых чисел $f_1, f_2, \ldots, f_n$ ($0 \leqslant f_i \leqslant 10^7$) — сроки годности пакетов кефира, которые уже есть у Оли в холодильнике. Срок годности выражается числом дней, на которые еще можно отложить употребление этого пакета. Таким образом, срок годности 0 означает, что пакет нужно выпить сегодня, 1 — что не позже завтра, и так далее.

Наконец, в третьей строке находятся m целых чисел $s_1, s_2, \ldots, s_m$ ($0 \leqslant s_i \leqslant 10^7$) — сроки годности пакетов, которые есть в магазине, в аналогичном формате.

Формат выходных данных

Если в любом случае Оля не сможет выпить даже уже имеющиеся у нее пакеты, выведите ровно одно число `-1`.

Иначе в первой строке выведите максимальное количество пакетов x, которые Оля может купить так, чтобы ничего не выбрасывать. В следующей строке выведите ровно x чисел — номера пакетов, которые следует взять (пакеты нумеруются в том порядке, в котором они заданы во входном файле, начинается с 1). Естественно,

числа не должны повторяться, однако могут идти в произвольном порядке. Если существует несколько подходящих наборов, разрешается вывести любой.

Система оценивания

В тестах суммарной стоимостью не менее 40 баллов все числа во входных данных не будут превосходить 10^4.

Пример

Ввод	*Вывод*
3 6 2 1 0 1 2 0 2 0 0 2	3 1 2 3
3 1 2 0 0 0 1	-1
2 1 2 0 1 0	1 1

Примечание: В первом примере из условия $k = 2$ и дома у Оли есть три пакета со сроками годности 0, 1 и 1 (т. е. истекающими сегодня, завтра и завтра), а в магазине есть 3 пакета со сроком годности 0 и 3 пакета со сроком годности 2. Оля может купить три пакета, например, один со сроком годности 0 и два со сроком годности 2.

Во втором примере все три уже имеющихся пакета имеют срок годности, оканчивающийся сегодня, и поэтому Оля не успеет их выпить вне зависимости от того, возьмет ли она еще один пакет в магазине или нет.

В третьем примере сегодня Оля выпьет $k = 2$ пакета (один уже имеющийся и один из магазина), а завтра — оставшийся один пакет.

Решение

Обозначим за t максимальный срок годности, который встречается во входных данных. Сначала приведем решение за $O(tm)$.

Ключевое наблюдение, которое нужно сделать: если мы зафиксируем число взятых пакетов x, то, если мы можем взять некоторые x пакетов из магазина и ничего не выбросить, то точно можно взять x пакетов с наибольшими сроками годности из имеющихся, и тоже выбрасывать ничего не придется. Это так, потому что если увеличить у некоторых взятых пакетов срок годности при фиксированном распределении пакетов по дням, то это распределение останется корректным.

Теперь для произвольного числа x научимся проверять, правда ли, что взяв x пакетов из магазина с наибольшими сроками годности, можно будет составить корректное распределение пакетов по дням. Для этого достаточно для каждого дня i от 0 до t проверить, что количество пакетов со сроками годности $\leqslant i$ (как исходных, так и взятых из магазина) не больше $(i + 1)k$.

Переберем число x от m до 0, как только для текущего x проверка проходит успешно, выводим ответ из x пакетов с наибольшими сроками годности. Заметим, что для определения номеров этих пакетов не нужно сортировать s_i, а достаточно

использовать сортировку подсчетом, т.к. по условию $s_i \leqslant 10^7$. Таким образом, у нас получилось решение за $O(tm)$, т.к. в худшем случае мы сделаем $m+1$ проверку за $O(t)$.

Чтобы получить полное решение задачи, остается заменить перебор x на бинарный поиск. Если z — ответ на задачу, то для всех x от 0 до z проверка будет пройдена успешно, а для $x > z$ нет; этой монотонности достаточно, чтобы бинарный поиск по значению x работал корректно. Получаем решение за $O(t \log m)$ на полный балл.

Задача 3. Спасение принцессы

Ввод	стандартный ввод
Вывод	стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт
Максимальный балл за задачу	100

Каждый год тысячам рыцарей приходится спасать принцесс, и наш герой Гарольд — не исключение. Его принцесса находится в самом центре замка, охраняемого злым драконом Русланом. Как и в любом нормальном замке, вокруг покоев нашей принцессы есть ров с водой. Будем считать, что он представляет собой окружность, причем ширина рва равна 0. Цель Гарольда — попасть в покои принцессы и снять с нее древние чары (другими словами, попасть в центр окружности, образованной рвом).

К сожалению, наш рыцарь слеп, поэтому самостоятельно добраться до принцессы он не в состоянии. Но у него есть верный конь Константин, который всегда выручает своего хозяина. Правда, лошади вечно хотят пить, поэтому все, что может Константин, это сбегать до рва с водой и рассказать, сколько он пробежал (то есть расстояние от местоположения Гарольда до ближайшей точки рва). После этого у Константина вырастают крылья, и он может перенести хозяина в любую точку карты.

В данной задаче вы играете роль Гарольда. Вы можете приказывать коню, куда лететь, а в ответ получать минимальное расстояние от данной точки до точек рва. Ваша задача — добраться до центра замка.

Формат взаимодействия с проверяющей программой

Это интерактивная задача. В процессе тестирования ваша программа будет с использованием стандартных потоков ввода/вывода (вывод «на экран»/ввод «с клавиатуры») взаимодействовать с программой жюри, которая моделирует работу коня Константина.

При каждом запуске вашей программы ей необходимо будет решить несколько тестовых случаев. В начале работы ваша программа должна считать одно натуральное число T ($1 \leqslant T \leqslant 10^4$) — количество тестовых случаев.

Далее, для каждого тестового случая, ваша программа должна следовать следующему протоколу:

- Для приказания коню перенести рыцаря ваша программа должна вывести в стандартный поток вывода («на экран») запрос в формате «`? x y`», где x, y — вещественные числа, координаты точки перемещения. Они не должны превосходить 10^4 по абсолютному значению. После этого ваша программа должна считать из стандартного потока ввода («с клавиатуры») одно вещественное число с ровно 10 знаками после десятичной точки — расстояние от точки (x, y) до ближайшей точки рва. Конечно, из-за конечного числа знаков это число может не быть равно настоящему расстоянию, однако гарантируется, что данное расстояние отличается от настоящего не более, чем на 10^{-10}.

- Когда ваша программа найдет центр замка, она должна вывести его координаты (x, y) в формате «`A x y`», и перейти к следующему тестовому случаю или завершить работу, если этот тестовый случай был последним.

Координаты центра замка — вещественные числа, не превосходящие по абсолютному значению 10^4. Радиус рва — неотрицательное вещественное число, не превосходящее 10^4.

Ваша программа может сделать не более 5 запросов в каждом тестовом случае всего (считая как запросы на перемещение, так и вывод ответа).

Ваш ответ будет считаться правильным, если его абсолютная или относительная ошибка не будет превосходить 10^{-6} для каждой координаты. А именно: пусть ваш ответ равен (x, y), а загаданный центр (x_1, y_1). Проверяющая программа будет считать ваш ответ правильным, если $\frac{|x-x_1|}{max(1,|x_1|)} \leqslant 10^{-6}$ и $\frac{|y-y_1|}{max(1,|y_1|)} \leqslant 10^{-6}$.

Все расстояния в данной задаче измеряются в метрах.

Запросы вашей программы должны завершаться переводом строки и сбросом буфера потока вывода. Для этого используйте flush(output) в Pascal/Delphi; fflush(stdout) или cout.flush() в C/C++, sys.stdout.flush() в Python.

В случае возникновения каких-либо технических проблем с этой задачей обращайтесь к представителю жюри.

Система оценивания

В этой задаче 33 теста. Верное прохождение одного из тестов оценивается в 4 балла, каждого из остальных тестов — в три балла.

При этом ваша программа также будет получать частичные баллы, если ошибка составила больше 10^{-6}. А именно, если во всех T тестовых случаях одного теста абсолютная или относительная ошибка не будет превосходить 10^{-2} для каждой координаты, то ваша программа получит 1 балл за этот тест. Если во всех T тестовых случаях абсолютная или относительная ошибка не будет превосходить 10^{-4} для каждой координаты, то ваша программа получит 2 балла за этот тест.

Пример

Ввод	*Вывод*
1 2.0000000 1.0000000 0.1715729	 ? 0.000 0.000 ? 3.000 0.000 ? 5.000 6.000 A 3.000 4.000

Примечание: В примере радиус рва равен 3 метрам.

Решение

Пусть искомые координаты центра окружности-рва есть X, Y, а ее радиус — R. Пусть i-й запрос у нашей программы был x_i, y_i, и в ответ программа получила число r_i. В таком случае несложно видеть, что

$$r_i = \left| R - \sqrt{(X - x_i)^2 + (Y - y_i)^2} \right|$$

В этом уравнении мы знаем x_i, y_i и r_i, и не знаем X, Y и R. Поэтому, сделав три запроса, мы получим три уравнения на три неизвестных, из которых можно будет найти искомые величины. Тем не менее, система из таких трех уравнений может иметь более одного решения, поэтому трех запросов недостаточно, потребуется сделать четыре запроса.

После этого остается только решить полученную систему из 4 уравнений с 3 неизвестными. Это можно делать многими способами, получая решения различной надежности и точности. Ниже описывается один из возможных вариантов решения. Отметим, что в данной задаче практически отсутствуют какие-либо особые случаи, а поэтому эффективность и точность выбранного метода решения в этой задаче достаточно просто оценить с помощью так называемого стресс-тестирования: можно просто в программе генерировать случайные параметры окружности (X, Y и R), запускать решение и проверять, насколько точно оно определило эти параметры и насколько часто ошибка определения превышает некоторый порог. В частности, изложенное ниже решение определяет координаты центра окружности с точностью хуже, чем требовавшаяся по условию 10^{-6}, лишь примерно в 1.5 запусках из 10^8. В этой задаче только 33 теста по 10^4 тестовых случаев, поэтому такой надежности безусловно хватает.

Итак, во-первых, в уравнениях присутствуют модули. Каждый модуль можно раскрыть двумя способами, для четырех уравнений получаем $2^4 = 16$ способов раскрытия модулей. Переберем все 16 вариантов. Получаем уравнения вида

$$\alpha_i r_i = R - \sqrt{(X - x_i)^2 + (Y - y_i)^2},$$

где $\alpha_i = \pm 1$ в зависимости от того, как мы раскрыли модуль.

Преобразованиями получаем

$$x_i^2 - 2x_i X + X^2 + y_i^2 - 2y_i Y + Y^2 - r_i^2 + 2\alpha_i r_i R - R^2 = 0$$

(здесь учтено, что $\alpha_i^2 = 1$).

Вычитая из i-го уравнения j-е, после преобразований имеем

$$2x_i X + 2y_i Y - 2\alpha_i r_i R - x_i^2 - y_i^2 + r_i^2 = 2x_j X + 2y_j Y - 2\alpha_j r_j R - x_j^2 - y_j^2 + r_j^2,$$

что является линейным уравнением относительно X, Y и R.

Выбирая различным способом i и j, можно получить 12 уравнений на X и Y. Несложно понять, что не более 3 из них независимы; с другой стороны, если x_i и y_i выбирать более-менее случайно, то с крайне высокой вероятностью независимых уравнений будет именно три, а не меньше. Тогда полученная система из линейных уравнений с тремя неизвестными будет иметь ровно одно решение.

Но для простоты уравнений можно пойти чуть дальше и обратить внимание на то, что в предыдущем уравнении левая часть не зависит от j, а правая — от i. Тогда очевидно, что на самом деле что левая, что правая часть уравнения равны некоторой величине C:

$$2x_i X + 2y_i Y - 2\alpha_i r_i R - x_i^2 - y_i^2 + r_i^2 = C,$$

и C одна для всех i.

В итоге получаем систему из четырех линейных уравнений (для $i = 1 \dots 4$) с четырьмя неизвестными (X, Y, R и C). Решая ее, находим искомые величины.

Вспомним, что у нас на самом деле было 16 вариантов выбора значений α_i (что соответствует 16 вариантам раскрытия модулей в исходных уравнениях). Поэтому мы на самом деле получаем 16 вариантов искомых координат. Подставляя их в исходные уравнения, выберем настоящий ответ, который и выведем. В особенно неудачном случае может оказаться, что несколько решений удовлетворяют исходным уравнениям, но такое крайне маловероятно, если мы выбираем x_i и y_i более-менее случайно. Кроме того, может оказаться, что погрешность ответа слишком большая, но, как показывает экспериментирование, это возникает обычно в случае, когда точки x_i и y_i расположены слишком близко к началу координат. Для определения ответа с максимальной точностью стоит брать точки достаточно удаленные от начала координат.

Как описано выше, можно провести стресс-тестирование полученного решения и убедиться в том, что оно находит неправильный ответ лишь в примерно 1.5 случаях из 100 000 000, чего вполне достаточно в данной задаче.

Задача 4. Снековик

Ввод	`snack.in` или стандартный ввод
Вывод	`snack.out` или стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт
Максимальный балл за задачу	100

Согласно древней легенде, давным-давно жители Анк-Морпорка провинились перед Госпожой Удачей, и та прокляла их. Она сказала, что однажды на город упадут n снеков разных размеров, а жители должны будут составить их в один большой Снековик. При этом, естественно, внизу должны будут быть самые большие снеки, а наверху — самые маленькие.

Прошли годы, и однажды на город стали действительно падать самые разнообразные снеки — от огромных Кендер-сюр до маленьких Что-по-чёмс. И жители города принялись строить из них Снековика.

Правда, их поджидала одна неприятность. Каждый день на город выпадал один снек, но падали они в каком-то странном порядке. Поэтому жители не всегда могли водрузить очередной снек на вершину Снековика; иногда им приходилось выпавший только что снек откладывать до тех пор, пока не выпадут все снеки больше его. Конечно, чтобы не разозлить Госпожу Удачу, жители все-таки устанавливали каждый снек, как только для того появлялась возможность.

Напишите программу, которая будет моделировать деятельность жителей города по постройке Снековика.

Формат входных данных

В первой строке находится одно натуральное число n ($1 \leqslant n \leqslant 100\,000$) — общее количество снеков, которые выпадут в городе.

Во второй строке находятся n чисел, i-ое из которых равно размеру снека, выпадающего в i-ый день. Все размеры — различные натуральные числа от 1 до n.

Формат выходных данных

Выведите в выходной файл n строк. На i-й из них выведите размеры всех снеков, которые будут установлены на снековик в i-й день, в том порядке, в котором они будут установлены. Если в какой-то день не будет установлен ни один снек, оставьте соответствующую строку выходного файла пустой.

Пример

Ввод	*Вывод*
3 3 1 2	3 2 1

Примечание: В примере в первый день выпадает снек размера 3, и его сразу можно устанавливать. Во второй день выпадает снек размера 1, его устанавливать еще нельзя, т.к. снек размера 2 пока отсутствует. В третий день выпадает снек размера 2, его тут же устанавливают, после чего устанавливают выпавший ранее снек размера 1.

Решение

В данной задаче было достаточно просто сделать то, что сказано в условии. Для этого удобно хранить массив has, в котором отмечать, какие снеки уже поступили, а какие еще нет, и в отдельной переменной $next$ хранить номер следующего снека, который надо поставить на вершину. Теперь обрабатываем все входные числа по очереди. Считав очередное число, отмечаем его в массиве has, после чего идем по массиву has начиная со снека $next$ до первого снека, которого еще нет, и все промежуточные снеки выводим и двигаем значение $next$.

Задача 5. Поищите без сдачи!

Ввод	`nochange.in` или стандартный ввод
Вывод	`nochange.out` или стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт
Максимальный балл за задачу	100

Студент Арсений любит планировать свои действия ровно на n дней вперед. Он каждый день ходит в столовую обедать, и поэтому он уже определился, что он будет заказывать в каждый из этих n дней. Цены в столовой не меняются, поэтому в i-й из этих дней Арсений пообедает ровно на c_i рублей.

В ходу монеты номиналом 1 рубль и купюры номиналом 100 рублей. У Арсения сейчас имеется m монет и достаточно много купюр (вы можете считать, что бесконечно много). Арсений любит современные технологии, поэтому везде, кроме столовой, он расплачивается карточкой, а в столовой карточки не принимают, и ему приходится расплачиваться наличными.

Кассир всегда просит студента расплатиться без сдачи. К сожалению, это не всегда возможно, но Арсений старается минимизировать *недовольство* кассира. Недовольство кассира в каждый из дней определяется суммарным количеством монет и купюр в сдаче Арсению в этот день, а именно, если кассир сдал x купюр и монет в день i, то его недовольство в этот день равно $x \cdot w_i$. Кассир всегда выдает сдачу наименьшим возможным числом купюр и монет, у него достаточно монет и купюр для этого.

Арсений хочет расплачиваться так, чтобы минимизировать суммарное недовольство кассира за n дней. Помогите ему определить, как ему стоит расплачиваться в каждый из n дней!

Заметьте, что Арсению всегда хватит денег, чтобы расплатиться, так как у него бесконечно много купюр. Арсений может использовать монеты и купюры, полученные в сдаче, в любой из следующих дней.

Формат входных данных

В первой строке находятся два целых числа n и m ($1 \leqslant n \leqslant 10^5$, $0 \leqslant m \leqslant 10^9$) — число дней, на которые Арсений распланировал свои действия, и число имеющихся у него монет в данный момент.

Во второй строке находится последовательность целых чисел $c_1, c_2, \ldots, c_n$ ($1 \leqslant c_i \leqslant 10^5$) — суммы, которые Арсений собирается потратить в каждый из дней.

В третьей строке находится последовательность целых чисел $w_1, w_2, \ldots, w_n$ ($1 \leqslant w_i \leqslant 10^5$) — коэффициенты недовольства кассира в каждый из дней.

Формат выходных данных

В первой строке выведите одно целое число — минимальное суммарное недовольство кассира, которого может достичь Арсений.

Далее выведите n строк. В i-й из этих строк выведите два числа — число купюр и число монет, которыми Арсений должен расплатиться в i-й день.

Естественно, в любой из дней сумма, которую даст Арсений, должна быть не меньше суммы, на которую он собирается пообедать. Также эта сумма не должна превышать 10^6 рублей: Арсений никогда не носит большие суммы с собой.

Если оптимальных ответов несколько, выведите любой.

Система оценивания

В тестах суммарной стоимостью не менее 20 баллов будут выполняться ограничения $1 \leqslant n \leqslant 20$, $0 \leqslant m \leqslant 500$.

В тестах суммарной стоимостью не менее 40 баллов будут выполняться ограничения $1 \leqslant n \leqslant 500$, $0 \leqslant m \leqslant 500$.

В тестах суммарной стоимостью не менее 50 баллов будут выполняться ограничения $1 \leqslant n \leqslant 500$.

Пример

Ввод	*Вывод*
5 42 117 71 150 243 200 1 1 1 1 1	79 1 17 1 0 2 0 2 43 2 0
3 0 100 50 50 1 3 2	150 1 0 1 0 0 50
5 42 117 71 150 243 200 5 4 3 2 1	230 1 17 1 0 1 50 3 0 2 0

Решение

Первое, что нужно заметить в этой задаче — это то, что в день i имеет смысл платить либо c_i div 100 банкнот и $c_i \bmod 100$ монет (тогда недовольство кассира будет равно 0), либо только c_i div $100 + 1$ банкнот (в таком случае недовольство кассира равно $\big(100 - (c_i \bmod 100)\big)w_i$). Более того, второй случай невозможен, если $c_i \bmod 100 = 0$, в таком случае надо просто заплатить необходимое число банкнот. (Здесь mod обозначает остаток от деления, а div — неполное частное.)

Пользуясь этим соображением, нетрудно придумать решение, имеющее асимптотику $O(2^n \cdot n)$: переберем для каждого дня, каким из этих двух способов Арсений будет расплачиваться, проверим, что у него всегда будет достаточно монет, и посчитаем суммарное недовольство кассира. Среди всех подходящих вариантов выберем тот, в котором суммарное недовольство минимально. Такое решение набирает 20 баллов.

Чтобы ускорить решение, воспользуемся методом динамического программирования. Заметим, что действия Арсения после некоторого дня зависят только от количества оставшихся у него монет после этого дня. Поэтому для каждого дня i

и каждого числа монет j посчитаем, какое минимальное суммарное недовольство кассира Арсений может достичь в дни после i-го, если после этого дня у него будет ровно j монет. Возможных переходов всего два, они соответствуют возможным вариантам, как Арсений расплатится в день $i+1$. Для полноценного решения осталось выяснить один момент: какое максимальное число монет может быть у Арсения после какого-то дня, иными словами, до какого максимального значения изменяется параметр j? Несложно заметить, что, т. к. сдача в каждый день меньше 100 монет, то у Арсения никогда не будет больше, чем $m + 100n$ монет. Таким образом, такое решение работает за $O(n \cdot (m + 100n))$ и набирает 40 баллов.

Чтобы набрать 50 баллов, достаточно заметить, что Арсению никогда не понадобится больше чем $100n$ монет, а значит, можно уменьшить m до такого значения, если изначально число монет было больше. Теперь динамическое программирование из предыдущего абзаца будет иметь асимптотику $O(n^2)$, правда, с довольно большой константой, и такое решение набирает 50 баллов.

Для полного решения надо заметить еще один факт. Предположим, Арсений расплатился в i-й день без сдачи, дав кассиру $c_i \bmod 100$ монет. Тогда, если мы изменим способ расплатиться в этот день, то количество монет у Арсения увеличится ровно на 100 во все последующие дни независимо от c_i! Действительно, Арсений в таком случае не отдаст эти $c_i \bmod 100$ монет, а еще он получит в сдаче ровно $100 - (c_i \bmod 100)$ монет, что в сумме дает 100 монет.

Будем строить оптимальное решение по дням начиная с первого, каждый раз стараясь расплачиваться без сдачи, чтобы минимизировать недовольство кассира. Пусть в первый раз Арсений не сможет расплатиться без сдачи в день i. Это значит, что среди дней с первого по i-й Арсений должен хотя бы один раз расплатиться со сдачей. Но, независимо от того, какой это будет день, после i-го дня у него будет одно и то же число монет! Значит, выгодно расплатиться со сдачей в тот день, в который недовольство кассира будет минимально. Далее, продолжим опять расплачиваться без сдачи, пока можем. Если в день j Арсений опять не может расплатиться без сдачи, то среди дней с первого по j-й должен быть еще один день, в который Арсений должен получить сдачу. По аналогичным соображениям можно выбрать такой день, в который недовольство кассира будет минимально (кроме дня, выбранного в первый раз). Продолжим повторять такие операции, пока не обработаем все дни.

Наивная реализация такого процесса имеет асимптотику $O(n^2)$ и не укладывается в ограничение по времени, хотя и набирает больше баллов, чем предыдущее решение. Однако, используя некоторую структуру данных, которая позволяет добавлять элемент, находить минимум и удалять его за $O(\log n)$, например, кучу или двоичное дерево поиска, мы можем в процессе хранить в этой структуре все прошедшие дни, в которых мы расплатились без сдачи, и находить день с минимальным недовольством кассира быстрее, чем за $O(n)$. Итоговая асимптотика такого решения $O(n \log n)$, что позволяет получить полный балл.

Задача 6. Очередь

Ввод	`queue.in` или стандартный ввод
Вывод	`queue.out` или стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт
Максимальный балл за задачу	100

Наконец-то! Васе исполнилось 14 лет, а значит, пора получать паспорт! Для этого ему необходимо обратиться в паспортный стол. Но не все так просто. В паспортном столе работает только одно окно по приему документов, а очередь в него иногда занимают задолго до его открытия. Вася хочет подать документы завтра.

Он знает, что окно начинает работать спустя t_s минут после полуночи, а закрывается спустя t_f после полуночи (то есть $(t_f - 1)$ — последняя минута, когда окно еще работает). На прием документов от одного человека тратится ровно t минут. Если очередного клиента не успеют обслужить до закрытия окна (т.е. если его очередь наступает менее чем за t минут до закрытия), то его не обслуживают вообще.

Ещё Вася знает, что завтра придет ровно n посетителей. Для каждого посетителя Вася знает время, в которое он придет. Каждый приходящий посетитель занимает очередь в окно, и не уходит, пока его не обслужат (или пока окно по приему документов не закроется). Если в момент прихода посетителя окно свободно (в частности, если в этот же момент закончили обслуживать предыдущего посетителя), то пришедшего посетителя начинают сразу же обслуживать.

Времена приходов всех посетителей положительны, Вася, если надо, может прийти и в нулевой момент времени (т. е. ровно в полночь), однако он не может прийти в момент времени, выражающийся нецелым числом минут после полуночи. Если Вася приходит одновременно с несколькими посетителями, то он пропускает их вперед и встает в конец очереди.

Вася, безусловно, хочет прийти так, чтобы успеть подать документы до закрытия окна. Но из всех таких вариантов он хочет выбрать такой, чтобы стоять в очереди как можно меньше. Помогите ему.

Формат входных данных

В первой строке входных данных находятся три положительных целых числа: время открытия окна t_s, время закрытия окна t_f и время обслуживания одного человека t. В следующей строке дано одно целое число n — количество посетителей ($0 \leqslant n \leqslant 100\,000$). В третьей строке перечислены положительные целые числа в порядке неубывания — времена, в которые посетители приходят в паспортный стол.

Все времена заданы в минутах и не превосходят 10^{12}; гарантируется, что $t_s < t_f$. Также гарантируется, что Вася может прийти так, чтобы успеть подать документы до закрытия окна.

Формат выходных данных

В выходной файл выведите одно целое неотрицательное число — время, когда должен прийти Вася. Если Вася приходит одновременно с несколькими посетителями, то он пропускает их вперед и встает в конец очереди. Если оптимальных решений несколько, выведите любое.

Пример

Ввод	*Вывод*
10 15 2 2 10 13	12
8 17 3 4 3 4 5 8	2

Примечание: В первом примере первый посетитель приходит точно к открытию окна, и его обслуживают две минуты. В момент времени 12 минут окно освобождается, и, если Вася подойдет к этому моменту, то его обслужат без очереди, т.к. следующий посетитель подойдет только в момент времени 13 минут.

Во втором примере, чтобы успеть подать документы, Васе надо прийти раньше всех.

Решение

Посчитаем, в какое время начнут обслуживать каждого посетителя. Пусть массив a содержит времена прихода посетителей. Первого посетителя начнут обслуживать в момент открытия окна t_s, если он пришел раньше этого времени, либо, в противном случае, в момент его прихода. Аналогично, можно посчитать время начала обслуживания i-ого посетителя, зная время начала обслуживания $(i-1)$-ого. Предположим, $(i-1)$-ого начали обслуживать в минуту b_{i-1}, тогда i-ого могут обслужить не раньше, чем через t минут, т.е. в момент времени $b_{i-1}+t$. В это время его начнут обслуживать, если i-ый посетитель пришел раньше ($a_i \leqslant b_{i-1} + t$). Если он пришел позднее, то его начнут обслуживать в момент его прихода. Таким образом, для каждого посетителя мы узнаем время b_i, когда его начнут обслуживать.

Если кто-то из посетителей пришел позже, чем закончили обслуживать предыдущего посетителя ($a_i > b_{i-1} + t$), то окно простаивало. Значит, Вася, придя в подходящий момент (например, в момент $b_{i-1}+t$), подаст документы без очереди. Если таких посетителей нет, то для того, чтобы обслужиться i-ым, Вася должен прийти не позднее времени $(a_i - 1)$. Тогда ждать ему придется минимум $b_i - (a_i - 1)$ минут. Из всех этих вариантов нам необходимо найти момент с минимальным временем ожидания.

При этом, следует учесть следующие случаи:

1. Если i-ый и $(i-1)$-ый посетители пришли в одно время, то Вася не сможет прийти между ними.

2. Некоторых посетителей могут не обслужить, если время начала их обслуживания больше или равно T_f. А значит, Васе не имеет смысла приходить после них.

3. Вася может обслужиться последним, если после обслуживания последнего посетителя до закрытия окна остается хотя бы t минут.

4. Вася может обслужиться первым, только если придет раньше всех посетителей.

Также, с учетом ограничений на времена до 10^{12}, для всех вычислений необходимо использовать 64-битный тип данных.

Таким образом, задача сводится к аккуратному подсчету времен обслуживания клиентов (сложность $O(n)$), с учетом перечисленных выше крайних случаев.

Задача 7. Гирлянда

Ввод	`garland.in` или стандартный ввод
Вывод	`garland.out` или стандартный вывод
Ограничение по времени	1.5 секунды
Ограничение по памяти	256 мегабайт
Максимальный балл за задачу	100

Однажды под Новый Год Диме приснилось, что ему подарили сказочную гирлянду. Гирлянда — это набор лампочек, некоторые пары которых соединены проводами. Дима запомнил, что она представляла из себя единое целое, то есть любая пара лампочек была связана некоторой последовательностью проводов. Еще он заметил, что проводов было ровно на один меньше, чем лампочек.

Необычность гирлянды состояла в том, что яркость каждой лампочки зависела от температуры этой лампочки, которая могла быть как отрицательной, так и положительной! У Димы есть два друга, и он захотел поделиться с ними сказочным подарком. Для этого он планирует разрезать два разных провода так, чтобы гирлянда распалась на три части. Дима хочет, чтобы все три части светились одинаково, то есть чтобы в каждой из них суммарная температура лампочек совпадала. Конечно же, каждая из частей должна быть непустой, то есть в ней должна быть хотя бы одна лампочка.

Помогите ему найти способ, как это сделать, либо определите, что такого не существует.

Осматривая гирлянду, Дима поднял ее, взяв за какую-то лампочку. Таким образом, каждая лампочка, кроме той, за которую он взял, оказалась висящей на некотором проводе. Поэтому в качестве ответа вам нужно вывести номера двух различных лампочек, что будет означать, что Диме нужно разрезать провода, за которые они подвешены. Соответственно, в ответ не может входить лампочка, за которую Дима взял гирлянду.

Формат входных данных

В первой строке находится целое число n ($3 \leqslant n \leqslant 10^6$) — количество лампочек в гирлянде.

Затем следует n строк: в i-ой из них находится информация об i-ой лампочке, а именно, два целых числа — номер лампочки a_i, к которой она подвешена (если такой нет, то число 0), и температура t_i ($-100 \leqslant t_i \leqslant 100$). Лампочки пронумерованы от 1 до n.

Формат выходных данных

Если решения не существует, выведите ровно одно число -1.

В противном случае выведите одну строку из двух различных чисел — номеров искомых лампочек. Если правильных ответов несколько, выведите любой.

Система оценивания

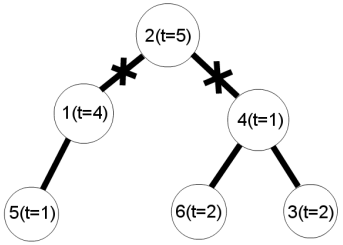
В тестах суммарной стоимостью не менее 50 баллов будут выполняться ограничения $1 \leqslant n \leqslant 2 \cdot 10^4$.

В тестах суммарной стоимостью не менее 50 баллов будут выполняться ограничения $1 \leqslant t_i \leqslant 100$.

Пример

Ввод	*Вывод*
6 2 4 0 5 4 2 2 1 1 1 4 2	1 4
6 2 4 0 6 4 2 2 1 1 1 4 2	-1

Примечание: Схема гирлянды и разрезаний к первому примеру из условия:



Решение

Заметим, что описанную в условии гирлянду можно представить как граф специфического вида — дерево. Приведем решение за $O(n)$. Сделаем один обход алгоритмом поиска в глубину (dfs) из корня.

Нам понадобится насчитать суммы t_i в каждом поддереве, обозначим такую сумму s_v для вершины v. Чтобы вычислить s_v, для каждого потомка v рекурсивно запустим dfs и сложим полученные суммы, прибавив еще t_v.

Обозначим сумму t_i во всем дереве как x. Если x не делится на три, то решения не существует. Иначе есть ровно два принципиально разных способа получить три поддерева с одинаковой суммой $(x/3)$ после разрезания двух ребер (обозначим вершины, ребра от которых к предкам будут удалены, как v_1 и v_2):

1. Одна из вершин является предком другой (без ограничения общности, v_2 предок v_1), и $s_{v_2} = 2x/3$, $s_{v_1} = x/3$.

2. Ни одна из вершин v_1 и v_2 не является предком другой, и $s_{v_1} = s_{v_2} = x/3$

Чтобы обнаружить первый вариант, достачно с помощью все того же обхода отслеживать, есть ли в поддереве текущей вершины v хотя бы одна вершина u с суммой $s_u = x/3$. Если такая вершина есть, и $s_v = 2x/3$, то имеет место быть первый вариант с $v_2 = v, v_1 = u$. С точки зрения реализации, можно возвращать номер нужной вершины u или знак об отсутствии таковой (например, -1) как результат работы функции dfs.

Чтобы отследить второй вариант, будем запоминать все вершины v, такие что $s_v = x/3$ и в поддереве v нет ни одной вершины u с $s_u = x/3$. Заметим, что наличие/отсутствие такой u для всех v мы уже научились определять в предыдущем пункте. Итак, если нашлось хотя бы две вершины v_1 и v_2, удовлетворяющие условию выше, то они и составляют ответ.

Содержание