Олимпиадное программирование

ГРИГОРЬЕВА АНАСТАСИЯ

Дистанционное занятие №1

Задачи:

1. Тройки чисел
2. Представление чисел
3. Кубическое уравнение
4. Три сына
5. Очередь в банк

Тройки чисел

Напишите программу, находящую количество троек целых чисел a, b, p таких, что p — простое число, числа удовлетворяют равенству:

$$\sqrt{a} - \sqrt{b} = \sqrt{p}$$

и каждое из чисел a, b и p лежит в промежутке от N до M (то есть $N \le a \le M$, $N \le b \le M$, $N \le p \le M$).

Входные данные

Вводятся два целых числа N и M ($0 \le N \le M \le 100000$)

Выходные данные

Выведите искомое количество троек чисел a, b, p.

Тройки чисел. Примеры.

входные данные
1 8
выходные данные
1

входные данные
5 20
выходные данные
1

входные данные
1 7
выходные данные
0

Решение

Возвести в квадрат

Но не просто так, а перенеся корень из b в правую часть

$$a = b + 2\sqrt{bp} + p$$

Поскольку a,b и p – целые числа, то и sqrt(bp) – целое число, т.е. b равно произведению p и квадрата некоторого целого числа

$$b = pn^2,\ a = p(n-1)^2$$

Т.о. для каждого p из отрезка [N,M] требуется найти все такие n, что

$$N <= p(n-1)^2 < pn^2 <= M$$

Оптимизируем

Разделим на p все части и извлечем корень

$$Sqrt(N/p) <= n-1 < n <= Sqrt(M/p)$$

Т.о., количество решений на 1 меньше, чем количество целых чисел на отрезке [Sqrt(N/p);Sqrt(M/p)]

Проверяем на простоту каждое число от N до M и находим кол-во решений для каждого из найденных простых чисел указанным выше способом

Представление чисел

Дано натуральное число N. Требуется представить его в виде суммы двух натуральных чисел A и B таких, что НОД (наибольший общий делитель) чисел A и B — максимален.

Входные данные

Во входном файле записано натуральное число N ($2 \le N \le 10^9$)

Выходные данные

В выходной файл выведите два искомых числа A и B. Если решений несколько, выведите любое из них.

Примеры

входные данные
15
выходные данные
5 10

входные данные
16
выходные данные
8 8

Решение

Пусть A и B — искомые числа, а d — их наибольший общий делитель. Поскольку $N = A + B$ и каждое из чисел A и B делится на d, то и N делится на d. Таким образом, в качестве кандидата на роль d можно рассматривать наибольший делитель числа N, меньший N.

Тогда $e = N : d$ — наименьший делитель числа N, больший 1.

Представим число N в виде суммы: $N = (N : e) + (N - N : e)$. Оба слагаемых делятся на d, поэтому можно положить: $A = N : e$, $B = N - N : e$. Таким образом, исходная задача свелась к нахождению наименьшего делителя e числа N, большего единицы.

Заметим, что либо $e \leqslant \sqrt{N}$, либо N — простое число и $e = N$. Действительно, если N составное и $e > \sqrt{N}$, то тогда $\frac{N}{e} \geqslant e > \sqrt{N}$, так как $\frac{N}{e}$ — тоже делитель числа N, отличный от 1, а e — наименьший делитель, отличный от 1. Но тогда $N = e \cdot \frac{N}{e} > \sqrt{N} \cdot \sqrt{N} = N$, что невозможно.

Решение

Следовательно, для нахождения числа e достаточно перебрать все числа от 2 до $\sqrt{N}$, и если среди этих чисел делитель числа N найден не будет, то положить $e = N$.

Заметим, что перебор всех чисел от 2 до N требует времени, выходящего за рамки временны́х ограничений задачи. Также слишком много времени требует поиск наибольшего делителя числа N, меньшего N.

Как часто бывает, текст решения в данном случае существенно короче, чем его описание:

```
i:=2;
while (N mod i > 0) AND (i<=sqrt(N)) do
  Inc(i);
if (N mod i > 0) then
  write(N-1,' ',1)
else
  write(N div i,' ',N - N div i);
```

Оптимизация

Заметим, что если число N — простое, то НОД$(A, B) = 1$ для любых натуральных A и B, дающих в сумме N, поэтому можно вывести любые натуральные числа, сумма которых равна N. Пользуясь этим, можно еще сократить приведенную выше программу:

```
i:=2;
while (N mod i > 0) AND (i<=sqrt(N)) do
  Inc(i);
write(N div i,' ',N - N div i);
```

Совет: при такой реализации sqrt(N) вычисляется при каждой Итерации цикла while. Этого можно избежать, если заранее вычислить эту величину и использовать в условии цикла её значение.

Кубическое уравнение

Напишите программу, которая будет искать все целые X, удовлетворяющие уравнению

$AX^3 + BX^2 + CX + D = 0$,

где A, B, C, D — данные целые числа.

Входные данные

Во входном файле записаны четыре целых числа: A, B, C, D. Все числа по модулю не превышают $2 \cdot 10^9$.

Выходные данные

В выходной файл выведите сначала количество решений этого уравнения в целых числах, а затем сами корни в возрастающем порядке. Если уравнение имеет бесконечно много корней, выведите в выходной файл одно число −1 (минус один).

Примеры

входные данные
1 0 0 -27
выходные данные
1 3

Решение

Если целое число x — решение уравнения n-й степени с ненулевым свободным членом, то x является делителем этого свободного члена.

Это несложно доказать: сумма (в нашем случае $Ax^3 + Bx^2 + Cx + D$) равна нулю, а все слагаемые, кроме последнего, делятся на x, следовательно, и последнее слагаемое, т. е. свободный член, делится на x.

Таким образом, если D отлично от нуля, то нам достаточно перебрать все делители числа D (включая отрицательные) и, подставив каждый из них в уравнение, выбрать те, которые являются корнями уравнения. Если же $D = 0$, то можно «сократить» на x, не забыв при этом корень $x = 0$. Если в полученном уравнении свободный член опять окажется равным нулю, нужно еще раз сократить на x и т. д.

Случай, когда $A = B = C = D = 0$ — единственный случай, когда уравнение имеет бесконечно много решений, и потому требуется вывести в выходной файл -1. Во всех остальных случаях уравнение либо имеет не более трех корней, либо вовсе не имеет корней. Поэтому отсортировать полученные корни по возрастанию не составит труда.

Заметим, что приведенное решение не зависит от того, является ли данное уравнение кубическим, т. е. равно ли A нулю.

Три сына

Во владениях короля Флатландии находится прямая дорога длиной n километров, по одну сторону от которой расположен огромный лесной массив. Король Флатландии проникся идеями защиты природы и решил превратить свой лесной массив в заповедник. Но сыновья стали сопротивляться: ведь им хотелось получить эти земли в наследство.

У короля три сына: младший, средний и старший. Король решил, что в заповедник не войдут участки лесного массива, которые он оставит сыновьям в наследство. При составлении завещания король хочет, чтобы для участков выполнялись следующие условия:

- каждый участок должен иметь форму квадрата, длина стороны которого выражается целым положительным числом. Одна из сторон каждого квадрата должна лежать на дороге. Пусть участки имеют размеры $a \times a$, $b \times b$ и $c \times c$;
- стороны квадратов должны полностью покрывать дорогу: величина $a + b + c$ должна быть равна n;
- участок младшего сына должен быть строго меньше участка среднего сына, а участок среднего сына должен, в свою очередь, быть строго меньше участка старшего сына, то есть должно выполняться неравенство $a < b < c$;
- суммарная площадь участков $a^2 + b^2 + c^2$ должна быть минимальна.

Требуется написать программу, которая по заданной длине дороги определяет размеры участков, которые следует выделить сыновьям короля.

Формат входного файла

Входной файл содержит одно целое число n ($6 \le n \le 10^9$).

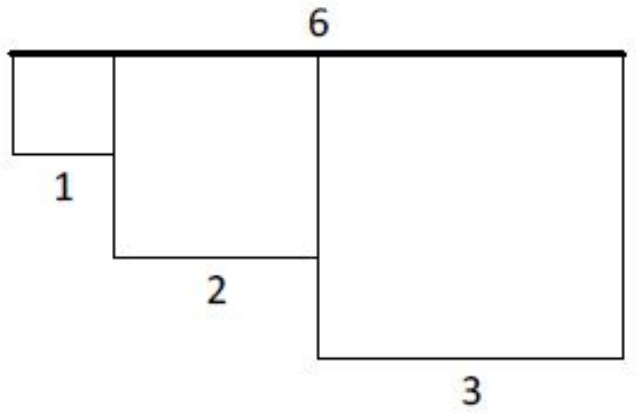
Формат выходного файла

Выходной файл должен содержать три целых положительных числа, разделенных пробелами: a, b и c – длины сторон участков, которые следует выделить младшему, среднему и старшему сыну, соответственно. Если оптимальных решений несколько, разрешается вывести любое.

Пример входных и выходных файлов

division.in	division.out
6	1 2 3

Пояснение к примеру



Решение

Заметим следующее: чтобы минимизировать сумму квадратов необходимо стараться выбрать числа a, b и c близкими к $n/3$. Формализуем это утверждение.

Докажем, сначала, вспомогательный факт про два числа. Пусть различные положительные целые $a < b$ таковы, что $a + b = m$. Тогда если сумма $a^2 + b^2$ минимальна, то $b - a \leq 2$. Действительно, пусть $b - a > 2$, тогда $a + 1 \neq b - 1$ и $(a + 1)^2 + (b - 1)^2 = a^2 + 2a + 1 + b^2 - 2b + 1 = a^2 + b^2 + 2(a - b) + 2 < a^2 + b^2 - 4 + 2 < a^2 + b^2$, следовательно взяв вместо a и b числа $a + 1$ и $b - 1$, мы получим два различных числа с такой же суммой, но меньшей суммой квадратов.

Рассмотрим теперь три числа a, b и c, которые требуется найти в задаче. Применив предыдущее утверждение для a и b при фиксированном c, а также для b и c при фиксированном a, получим, что все числа не более чем на 4 отстоят от значения $n / 3$.

Получаем следующее решение: переберем все тройки целых положительных чисел, где каждое число, не более чем на 4 отличается от $n / 3$ и выберем среди них тройку с минимальной суммой квадратов.

Приведем пример программы на Си++, которая решает поставленную задачу.

```
int n3 = n / 3;
int delta = 4;
int minv = max(n3 - delta, 1);
int maxv = min(n3 + delta, n);
long long best = (long long) n * n;
int ba, bb, bc = 0;
for (long long a = minv; a <= maxv; a++)
    for (long long b = a + 1; b <= maxv; b++)
        for (long long c = b + 1; c <= maxv; c++)
            if (a + b + c == n && a*a + b*b + c*c < best) {
                best = a * a + b * b + c * c;
                ba = a;
                bb = b;
                bc = c;
            }
cout << ba << " " << bb << " " << bc << endl;
```

Отметим, что вместо числа 4 можно использовать и любое большее значение. Главное, чтобы оно не было *слишком* большим, чтобы решение по прежнему проходило по времени.

Оптимизации

Частичные решения для подзадач 1, 2 и 3 позволяют получить частичные баллы в случае неполного развития этой идеи.

Для решения подзадачи 1 можно перебрать все возможные значения a, b и c и выбрать из них оптимальное. Время работы $O(n^3)$.

Для решения подзадачи 2 следует заметить, что можно перебрать лишь два значения, например, a и b, а значение c вычислить из равенства $a + b + c = n$. . Время работы $O(n^2)$.

Наконец, для решения подзадачи 3 можно, например, перебрать большее из трех значений: c, и заметить, что $a + b = n - c = m$, а для минимизации $a^2 + b^2$ следует взять $a = (m - 1) / 2$, $b = (m + 1) / 2$ при нечетном m и $a = m / 2 - 1$, $b = m / 2 + 1$ при четном m. Время работы $O(n)$.

Код на питоне

```
n = int (input())

b = n//3
a = b-1
if n%3 ==2
   b + = 1
c = n-a - b
print (a, b, c)
```

Очередь в банк

Недавно Скрудж устроился работать охранником в банке. Работа скучная, делать нечего, поэтому он начал следить за очередью. Исходно в очереди стоит n человек. Так как до этого несколько лет Скрудж работал психологом, он смог довольно точно оценить настроение каждого человека в очереди. Скрудж пронумеровал людей в очереди по порядку, начиная с нуля, таким образом получилось, что номер человека в очереди равен числу людей, которое стоит в очереди перед ним. Настроение i-го человека он описал целым неотрицательным числом a_i. Скрудж считает, что у человека хорошее настроение, если оно не меньше x. Если это не так, то настроение у человека плохое.

Люди приходят в очередь, уходят из нее. Если в очередь приходит новый человек, Скрудж мгновенно оценивает его настроение, и с течением времени оно не меняется.

Теперь Скрудж придумал себе следующее занятие: в некоторые моменты времени он выбирает одного человека из очереди и считает, сколько перед ним стоит человек с хорошим настроением. Это занятие уже показалось ему интересным, и он решил придумать, как его можно автоматизировать. Так как сам Скрудж не силен в программировании, помощи в решении этой задачи он попросил у вас. Помогите ему!

Входные данные

В первой строке входного файла даны два числа n, x ($1 \le n \le 100\,000, 0 \le x \le 10^9$) — начальное количество человек в очереди и нижняя граница хорошего настроения.

В следующей строке даны n чисел a_i — настроения людей в очереди ($0 \le a_i \le 10^9$).

В третьей строке входного файла дано число m ($1 \le m \le 100\,000$) — количество событий, которые происходили с очередью. В следующих m строках дано описание событий. Событие описывается одним из трех способов:

- 1 a ($0 \le a \le 10^9$) — в конец очереди приходит человек с настроением, равным a.
- 2 — из очереди уходит человек, перед которым никого не стоит (в нумерации Скруджа он имеет номер 0). После этого Скрудж мысленно уменьшает номера людей в очереди на 1.
- 3 i — Скрудж хочет узнать, сколько людей с хорошим настроением стоит перед человеком, перед которым в очереди в этот момент стоит i человек.

Гарантируется, что все запросы корректны: если в очереди никого нет, то операция второго типа не выполняется, а количество человек в очереди всегда будет строго больше i в запросе третьего типа.

Выходные данные

На каждый запрос третьего типа в отдельной строке выходного файла выведите одно число — количество человек с хорошим настроением, которые стоят перед человеком с данным номером.

Тесты

longqueue.in

```
1 2
3
5
1 2
1 1
3 0
3 1
3 2
```

longqueue.out

```
0
1
2
```

longqueue.in

```
2 2
1 2
7
3 0
3 1
2
3 0
1 3
3 0
3 1
```

longqueue.out

```
0
0
0
0
1
```

Упрощенное условие

- Дана очередь из людей, у каждого человека какое-то настроение
- Настроение считается хорошим, если оно не меньше x
- В очередь приходят и уходят люди
- Надо научиться отвечать на запрос «Сколько человек с хорошим настроением стоит в очереди перед данным»

Решение на 40 баллов

- Будем хранить очередь в массиве
- Добавление человека в конец очереди и удаление из начала — $O(1)$.
- Ответ на запрос — посмотреть на всех людей в очереди перед данным и посчитать, сколько из них в хорошим настроением
- Ответ на запрос в худшем случае за $O(n)$.

Решение на 100 баллов

- Поддерживаем префиксные суммы массива, где в i-ом элементе записано 1, если у i-го человека хорошее настроение, и 0, если плохое
- $prefixSum[maxIndex]$ — сколько человек с хорошим настроением стоит перед человеком с номером $maxIndex$
- Поддерживаем человека, который стоит первым в очереди

Решение на 100 баллов

- Добавление в конец очереди — подсчет новой префиксной суммы
- Удаление из начала очереди — увеличение номера первого человека в очереди на 1
- Запрос — разность двух префиксных сумм:

$$answer = prefixSum[queryId] - prefixSum[firstManIndex]$$

- Все операции выполняются за O(1)
- Итоговая асимптотика — O(m)

Код на питоне

```
input = open('longqueue.in', 'r')
output = open('longqueue.out', 'w')
n,x=input.readline().split()
x=x.rstrip()
a=input.readline().split()
n=int(n)
a[n-1]=a[n-1].rstrip()
x=int(x)
b=[0]
for i in range(1,n):
    if int(a[i-1])>=x:
        b.append(b[i-1]+1)
    else:
        b.append(b[i-1])
k=int(input.readline().rstrip())
smesh=0
for i in range(k):
    rab=input.readline()
    rab=rab.rstrip()
    if rab[0]=='2':
        smesh+=1
    elif rab[0]=='1':
        e=int(rab[2:])
        if int(a[-1])>=x:
            b.append(b[-1]+1)
        else:
            b.append(b[-1])
        a.append(e)
    else:

otv=b[int(rab[2:])+smesh]-b[smesh]
        output.write(str(otv)+'\n')
```

```
f = open ("longqueue.in", "r")
n, x = [int(x) for x in f.readline().split()]
bb = [int(x) for x in f.readline().split()]

d = 0
a = [0]
for b in bb:
    if b >= x:
        d += 1
    a.append ( d)
m_count = int (f.readline())
result = []

h = 0  # head person position

for i in range (m_count):
    m = f.readline().split()
    if m[0] == '1':
        if int (m[1]) >= x:
            a.append(a[-1] + 1)
        else:
            a.append(a[-1])
    elif m[0] == '2':
        h  = h + 1

    elif m[0] == '3':
        r = a[int (m[1]) + h]  - a[h]
        result.append(r)
        print (r)

f.close

with open("longqueue.out", "w") as f:
    for r in result:
        f.write(str(r)+'\n')
```

Код на C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <fstream>
using namespace std;

int main(){
    ifstream fin("longqueue.in");
    ofstream fout("longqueue.out");

    int n, x; fin >> n >> x;
    vector <pair <int, int>> v; // f - mood, s - count
    for (int i = 0; i != n; i++){
        int a; fin >> a;
        v.push_back({a, 0});
        if (i != 0){
            if (v[i - 1].first >= x){
                v[i].second = v[i - 1].second + 1;
            }
            else{
                v[i].second = v[i - 1].second;
            }
        }
    }
    int r = 0;
    int m; fin >> m;
    for (int i = 0; i != m; i++){
        if (v.size() == 0){
            r = 0;
        }
        int a; fin >> a;
        if (a == 1){
            int u; fin >> u;
            v.push_back({u, 0});
            if (v.size() != 1 && v[v.size() - 2].first >= x){
                v[v.size() - 1].second = v[v.size() - 2].second + 1;
            }
            else{
                if (v.size() != 1){
                    v[v.size() - 1].second = v[v.size() - 2].second;
                }
            }
        }
        else if (a == 2){
            int y = v.begin()->first;
            if (y >= x){
                r++;
            }
            v.erase(v.begin());

        }
        else if (a == 3){
            int u; fin >> u;
            fout << v[u].second - r << endl;
        }
    }
}
```