

## Задача А. Добавление и удаление точек

Имя входного файла: a.in  
Имя выходного файла: a.out  
Ограничение по времени: 1 с  
Ограничение по памяти: 256 Мб

Не все  $N^2 \log N$  одинаковы полезны...

Какая-то лекция

У вас в каждый момент есть мультимножество  $A$  точек на плоскости.

Нужно научиться обрабатывать запросы трех типов:

- Добавить точку в мультимножество  $A$
- Удалить точку из мультимножества  $A$
- Вычислить  $\sum_{p \in A} \max_{q \in A} distance(p, q)$ .

### Формат входного файла

Число запросов  $N$  ( $1 \leq N \leq 3000$ ). Далее  $N$  строк, описывающие запросы, точный формат смотрите в примере. Координаты точек — целые число от 0 до 3000. Точки могут совпадать. Запрос *удалить точку* должен удалять ровно одну точку (гарантируется, что такая точка в мультимножестве на момент запроса есть).

### Формат выходного файла

После каждой операции с множеством выводите текущую сумму максимальных расстояний. Абсолютная погрешность не должна превышать  $10^{-6}$ .

### Примеры

a.in	a.out
6	0.00000000000000000000
+ 0 0	14.14213562373095100000
+ 5 5	19.14213562373095100000
+ 5 0	10.00000000000000000000
- 5 5	0.00000000000000000000
- 5 0	0.00000000000000000000
- 0 0	

## Задача В. Сбалансированное дерево

Имя входного файла:           b.in  
Имя выходного файла:       b.out  
Ограничение по времени:    0.5 с  
Ограничение по памяти:     256 Мб

Дано подвешенное за вершину с номером 1 дерево из  $N$  вершин. Вершины дерева нумеруются от 1 до  $N$  и покрашены в красный и черный цвета.

Дерево называется красно-черным, если:

- У каждой вершины количество детей или 2 или 0.
- Все листья черные
- На пути от корня до любого из листьев одинаковое количество черных вершин
- Отцом красной вершины не может быть красная вершина

Ваша задача — проверить, является ли данное вам дерево красно-черным. Если же является, то нужно построить изоморфное ему 2-3-4-дерево.

### Формат входного файла

Число  $N$  ( $1 \leq N \leq 10^5$ ), далее  $N$  пар чисел.  $i$ -я пара чисел — отец  $i$ -й вершины и ее цвет (буква R = red, или буква B = black). Отец корня — вершина с номером  $-1$ .

### Формат выходного файла

Выведите YES или NO. Если ответ YES, выведите 2-3-4 дерево изоморфное исходному. Дерево выведите в следующем формате: количество ребер и ребра. Нумерация вершин должна быть такая же, как и в исходном дереве.

## Примеры

b.in	b.out
1 -1 R	NO
3 -1 R 1 B 1 B	YES 2 1 2 1 3
2 -1 B 1 B	NO
5 -1 R 1 B 1 R 3 B 3 B	NO
17 -1 B 1 R 1 B 2 B 2 B 4 B 4 B 5 B 5 R 9 B 9 B 3 R 3 R 12 B 12 B 13 B 13 B	YES 12 1 3 1 4 1 5 4 6 4 7 5 8 5 10 5 11 3 14 3 15 3 16 3 17
4 -1 B 1 B 1 B 1 B	NO

## Примечание

2-3-4-дерево обладает двумя свойствами — во-первых, расстояния от корня до листьев одинаковы, во-вторых, число детей любого не листа — 2, 3 или 4.

Имеется в виду изоморфизм, обсуждавшийся на лекции. Если лекция не помогла, предлагается изучить тестовые примеры.

## Задача C. Count Offline

Имя входного файла: `c.in`  
 Имя выходного файла: `c.out`  
 Ограничение по времени: 2 с  
 Ограничение по памяти: 256 Мб

Вам дано множество точек на плоскости.

Нужно уметь отвечать на два типа запросов:

- `+ x y` — добавить в множество точку  $(x, y)$ .
- `? x1 y1 x2 y2` — сказать, сколько точек лежит в прямоугольнике  $[x_1..x_2] \times [y_1..y_2]$ . Точки на границе и в углах тоже считаются.  $x_1 \leq x_2, y_1 \leq y_2$ .

### Формат входного файла

Число точек  $N$  ( $1 \leq N \leq 50\,000$ ). Далее  $N$  точек. Число запросов  $Q$  ( $1 \leq Q \leq 100\,000$ ). Далее  $Q$  запросов. Все координаты от 0 до  $10^9$ .

### Формат выходного файла

Для каждого запроса GET одно целое число — количество точек внутри прямоугольника.

### Пример

<code>c.in</code>	<code>c.out</code>
4	2
0 0	4
1 0	1
0 1	
1 1	
5	
? 0 1 1 2	
+ 1 2	
+ 2 2	
? 1 0 2 2	
? 0 0 0 0	

### Примечание

Автор задачи уже написал некоторый код, который вы можете скачать по адресу (<http://ejudge.kture.kharkov.ua/buffer/Kopeliovich/C/lib.cpp> или <http://ejudge.kture.kharkov.ua/buffer/Kopeliovich/C/lib.java>) и использовать. Имеется код на языках C++, Java.

Написанная часть умеет делать три вещи:

- `Build( множество точек на плоскости и их начальные значения )`.
- `ChangeValue( индекс точки в множестве, ее новое значение )`.
- `GetSum( прямоугольник )`.

Время работы: `Build` за  $O(N \log N)$ , `ChangeValue` за  $O(\log^2 N)$ , `Get` за  $O(\log^2 N)$ .

## Задача D. Count Online

Имя входного файла: `d.in`  
 Имя выходного файла: `d.out`  
 Ограничение по времени: 3 с  
 Ограничение по памяти: 256 Мб

Вам дано множество точек на плоскости.

Нужно уметь отвечать на два типа запросов: Вам дано множество точек на плоскости.

Нужно уметь отвечать на два типа запросов:

- `? x1 y1 x2 y2` — сказать, сколько точек лежит в прямоугольнике  $[x_1..x_2] \times [y_1..y_2]$ . Точки на границе и в углах тоже считаются.  $x_1 \leq x_2$ ,  $y_1 \leq y_2$ .
- `+ x y` — добавить в множество точку  $(x + \text{res} \% 100, y + \text{res} \% 101)$ . Где `res` — ответ на последний запрос вида `?`, а `%` — операция взятия по модулю.

### Формат входного файла

Число точек  $N$  ( $1 \leq N \leq 50\,000$ ). Далее  $N$  точек. Число запросов  $Q$  ( $1 \leq Q \leq 100\,000$ ). Далее  $Q$  запросов. Все координаты от 0 до  $10^9$ .

### Формат выходного файла

Для каждого запроса GET одно целое число — количество точек внутри прямоугольника.

### Пример

d.in	d.out
5	3
0 0	3
1 0	1
0 1	0
1 1	0
1 1	3
9	
? 0 1 1 2	
+ 1 2	
+ 2 2	
? 1 0 2 2	
? 0 0 0 0	
+ 3 3	
? 3 3 3 3	
? 4 3 4 3	
? 4 4 5 5	

### Примечание

На самом деле добавлялись точки  $(4, 5)$ ,  $(5, 5)$ ,  $(4, 4)$ .

## Задача F. Самая дальняя

Имя входного файла: `f.in`  
Имя выходного файла: `f.out`  
Ограничение по времени: 1 с  
Ограничение по памяти: 256 Мб

Даны  $N$  точек на плоскости, нужно уметь обрабатывать следующие запросы:

- `get a b` — возвращает максимум по всем точкам величины  $ax + by$ .
- `add x y` — добавить точку в множество.

### Формат входного файла

Число  $N$  ( $1 \leq N \leq 10^5$ ) и  $N$  точек. Далее число  $M$  ( $1 \leq M \leq 10^5$  — количество запросов и собственно запросы. Формат запросов можно посмотреть в примере. Все координаты точек и числа  $a$ ,  $b$  — целые числа, по модулю не превосходящие  $10^9$ .

### Формат выходного файла

На каждый запрос вида `get` выведите одно целое число — максимум величины  $ax + by$ .

### Примеры

<code>f.in</code>	<code>f.out</code>
3	1
0 0	0
1 0	1
0 1	1
10	4
get 1 1	4
get -1 -1	1
get 1 -1	1
get -1 1	
add 2 2	
add -2 -2	
get 1 1	
get -1 -1	
get 1 -1	
get -1 1	

### Примечание

Автор задачи уже написал некоторый код, который вы можете скачать по адресу (<http://ejudge.kture.kharkov.ua/buffer/Kopeliovich/F/lib.cpp> или <http://ejudge.kture.kharkov.ua/buffer/Kopeliovich/F/lib.java>) и использовать. Имеется код на языках C++, Java.

Написанная часть умеет делать две вещи:

- `Build( множество точек на плоскости )`.
- `GetMax(a, b)`.

Время работы: `Build` за  $O(N \log N)$ , `GetMax` за  $O(\log N)$ .

## Задача G. Persistent Array

Имя входного файла: `g.in`  
 Имя выходного файла: `g.out`  
 Ограничение по времени: 0.5 с  
 Ограничение по памяти: 256 Мб

Дан массив (вернее, первая, начальная его версия).

Нужно уметь отвечать на два запроса:

- $a_i[j] = x$  — создать из  $i$ -й версии новую, в которой  $j$ -й элемент равен  $x$ , а остальные элементы такие же, как в  $i$ -й версии.
- `get  $a_i[j]$`  — сказать, чему равен  $j$ -й элемент в  $i$ -й версии.

### Формат входного файла

Количество чисел в массиве  $N$  ( $1 \leq N \leq 10^5$ ) и  $N$  элементов массива. Далее количество запросов  $M$  ( $1 \leq M \leq 10^5$ ) и  $M$  запросов. Формат описания запросов можно посмотреть в примере. Если уже существует  $K$  версий, новая версия получает номер  $K + 1$ . И исходные, и новые элементы массива — целые числа от 0 до  $10^9$ . Элементы в массиве нумеруются числами от 1 до  $N$ .

### Формат выходного файла

На каждый запрос типа `get` вывести соответствующий элемент нужного массива.

### Примеры

<code>g.in</code>	<code>g.out</code>
6	6
1 2 3 4 5 6	5
11	10
create 1 6 10	5
create 2 5 8	10
create 1 5 30	8
get 1 6	6
get 1 5	30
get 2 6	
get 2 5	
get 3 6	
get 3 5	
get 4 6	
get 4 5	

## Задача Н. Перестановки **strike back**

Имя входного файла: `h.in`  
Имя выходного файла: `h.out`  
Ограничение по времени: 1.5 с  
Ограничение по памяти: 256 Мб

Вася выписал на доске в каком-то порядке все числа от 1 по  $N$ , каждое число ровно по одному разу. Иногда он стирает какое-то число и записывает на его место другое. Количество чисел, выписанных Васей, оказалось довольно большим, поэтому Вася не может окинуть взглядом все числа. Однако ему надо всё-таки представлять эту последовательность, поэтому он написал программу, которая в любой момент отвечает на вопрос — сколько среди чисел, стоящих на позициях с  $x$  по  $y$ , по величине лежат в интервале от  $k$  до  $l$ . Сделайте то же самое.

### Формат входного файла

В первой строке лежит два натуральных числа —  $1 \leq N \leq 100\,000$  — количество чисел, которые выписал Вася и  $1 \leq M \leq 100\,000$  — суммарное количество вопросов и изменений сделанных Васей. Во второй строке дано  $N$  чисел — последовательность чисел, выписанных Васей. Далее в  $M$  строках находятся описания вопросов. Каждый запрос на изменение числа в некоторой позиции начинается со слова **SET** и имеет вид **SET a b** ( $1 \leq a \leq N$ ,  $1 \leq b \leq N$ ). Это означает, что Вася изменил число, записанное в позиции  $a$  на число  $b$ . Каждый Васин вопрос начинается со слова **GET** и имеет вид **GET x y k l** ( $1 \leq x \leq y \leq N$ ,  $1 \leq k \leq l \leq N$ ).

### Формат выходного файла

Для каждого Васиного вопроса выведите единственное число — ответ на Васиный вопрос.

### Примеры

<code>h.in</code>	<code>h.out</code>
4 4	1
1 2 3 4	3
GET 1 2 2 3	2
GET 1 3 1 3	
SET 1 4	
GET 1 3 1 3	



## Задача J. Проекция в $R^3$

Имя входного файла: `j.in`  
Имя выходного файла: `j.out`  
Ограничение по времени: 1 с  
Ограничение по памяти: 256 Мб

Даны  $N$  трехмерных точек. Нужно для каждой найти любую ближайшую точку. Расстояние между точками равно  $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ .

### Формат входного файла

Число точек  $N$  ( $2 \leq N \leq 3 \cdot 10^4$ ) и  $N$  точек. Каждая точка задается тремя координатами  $x$ ,  $y$ ,  $z$ . Все координаты — целые числа от 0 до  $10^9$ .

### Формат выходного файла

Выведите  $N$  чисел — для каждой точки номер ближайшей к ней точки (от 1 до  $N$ ).

### Пример

<code>j.in</code>	<code>j.out</code>
6 0 0 0 2 0 0 2 2 0 0 2 0 1 1 0 0 4 0	5 5 5 5 3 4
3 0 0 0 0 0 0 1 1 1	2 1 2

## Задача К. Прямоугольные запросы

Имя входного файла: k.in  
 Имя выходного файла: k.out  
 Ограничение по времени: 4 с  
 Ограничение по памяти: 256 Мб

Даны  $N$  точек на плоскости, у каждой точки есть ценность. Нужно быстро обрабатывать запросы двух типов:

- Присвоить всем точкам в области  $[x_1..x_2] \times [y_1..y_2]$  ценность  $K$ .
- Найти точку с минимальной ценностью в области  $[x_1..x_2] \times [y_1..y_2]$ .

### Формат входного файла

Число точек  $N$  ( $1 \leq N \leq 262144$ ) и  $N$  точек. Каждая точка задается тремя числами —  $x$ ,  $y$ , начальная ценность.

Число запросов  $M$  ( $1 \leq M \leq 10^4$ ) и  $M$  запросов в формате =  $x_1 \ y_1 \ x_2 \ y_2 \ value$  для присваивания и ?  $x_1 \ y_1 \ x_2 \ y_2$  для взятия минимума.

Все координаты от  $-10^9$  до  $10^9$ . Все ценности от 0 до  $10^9$ .

### Формат выходного файла

На каждый запрос ? выведите минимальную ценность точек в прямоугольнике. Если в прямоугольнике нет ни одной точки, выведите NO.

### Пример

k.in	k.out
4	2
1 1 1	1
-1 1 1	NO
-1 -1 1	0
1 -1 1	
7	
= 0 0 3 3 2	
= -3 -3 0 0 2	
? 0 0 3 3	
? -3 -3 3 3	
= -1 -1 1 1 0	
? 0 0 0 0	
? -1000 -1000 1000 1000	

## Задача L. Точки в полуплоскости

Имя входного файла: 1.in  
Имя выходного файла: 1.out  
Ограничение по времени: 0.75 с  
Ограничение по памяти: 256 Мб

Есть  $N$  точек на плоскости. Точки равномерно распределены внутри квадрата  $[0..C] \times [0..C]$ . Вам нужно научиться отвечать на запрос “сколько точек лежит в полуплоскости”?

### Формат входного файла

Число точек  $N$  ( $1 \leq N \leq 5 \cdot 10^4$ ), число запросов  $M$  ( $1 \leq M \leq 5 \cdot 10^4$ ), константа  $C$  (целое число от 1 до  $10^4$ ). Далее  $N$  точек  $(X, Y)$  с целочисленными координатами. Далее  $M$  полуплоскостей  $(a, b, c)$ . Числа  $a$ ,  $b$ ,  $c$  — целые, по модулю не превосходят  $10^4$ .  $a^2 + b^2 \neq 0$ . Считается, что точка лежит в полуплоскости тогда и только тогда, когда  $ax + by + c \geq 0$ .

### Формат выходного файла

Для каждого из  $M$  запросов одно целое число — количество точек в полуплоскости.

### Пример

1.in	1.out
3 4 10	2
5 5	2
1 7	1
7 4	0
1 1 -9	
1 1 -10	
1 1 -11	
1 1 -12	

## Задача N. Подстроки со сдвигом

Имя входного файла: n.in  
Имя выходного файла: n.out  
Ограничение по времени: 2 с  
Ограничение по памяти: 256 Мб

Вам даны  $K$  текстов. Все тексты имеют одинаковую длину.

Ваша задача — научиться искать подстроку со сдвигом. Подстрока  $S$  со сдвигом  $a_1, a_2, \dots, a_K$  входит в набор из  $K$  текстов  $T_1, T_2, \dots, T_K$ , если существует такое число  $x$ , что для всех  $i$   $LCP(T_i + a_i + x, S) \geq |S|$ . Где  $LCP$  — длина наибольшего общего префикса,  $(T_i + j)$  —  $j$ -й суффикс строки  $T_i$ ,  $|S|$  — длина строки  $S$ .

### Формат входного файла

Число  $K$  от 1 до 10 и  $K$  текстов (длины текстов одинаковы и лежат от 1 до  $10^5$ ). Далее  $M$  от 1 до  $10^5$  — число запросов и сами запросы. Каждый запрос это строка и  $K$  чисел от  $-10^9$  до  $10^9$ . Суммарная длина всех строк в запросах не более  $10^5$ . Все строки и тексты состоят только из маленьких символов английского алфавита. **Все строки  $S$  по всем запросам различны.**

### Формат выходного файла

Для каждого запроса выведите NO или YES **x** ( $x$  — величина из условия).

### Пример

n.in	n.out
3	YES 0
abacabaa	YES 5
ababbbbaa	NO
aababbbb	YES 0
4	
a 0 0 1	
b 0 0 0	
ba 0 0 -1	
aa 6 6 0	

## Задача О. Антитест

Имя входного файла: o.in  
 Имя выходного файла: o.out  
 Ограничение по времени: 0.3 с  
 Ограничение по памяти: 256 Мб

Вам дана реализация декартова дерева. Ошибок тут нет, не волнуйтесь. Есть одна особенность: вместо у-ков в операции **Merge** используется `rnd.next(2)` (функция возвращает случайное число от 0 до 1).

Утверждается, что построенное таким образом дерево может иметь слишком большую глубину. Ваша задача — найти последовательность из не более чем 3000 операций, создающую дерево глубины не менее 1000 (дерево из одной вершины имеет глубину 1).

```
1 typedef struct tree * ptree;
2 struct tree { ptree l, r; int x; };
3 ptree root, a, b, c;
4
5 // comment: l = {y < x}, r = {y >= x}
6 void Split( ptree t, ptree &l, ptree &r, int x ) {
7     if (t == 0)         l = r = 0;
8     else if (t->x >= x) Split(t->l, l, t->l, x), r = t;
9     else                Split(t->r, t->r, r, x), l = t;
10 }
11 void Merge( ptree &t, ptree l, ptree r ) {
12     if (l == 0)         t = r;
13     else if (r == 0)     t = l;
14     else if (rnd.next(2)) t = l, Merge(t->r, t->r, r);
15     else                t = r, Merge(t->l, l, t->l);
16 }
17 void Add( int x ) {
18     b = new tree(x);
19     Split(root, a, c, x);
20     Merge(a, a, b);
21     Merge(root, a, c);
22 }
23 void Del( int x ) {
24     Split(root, a, b, x);
25     Split(b, b, c, x + 1);
26     Merge(root, a, c);
27 }
```

### Формат входного файла

Вам предоставлен абсолютно пустой файл. Дерево изначально также пусто.

### Формат выходного файла

Последовательность операций вида **ADD x** и **DEL x**.

**ADD x** вызывает функцию **Add(x)** (см. реализацию)

**DEL x** вызывает функцию **Del(x)** (см. реализацию)

### Пример

o.in	o.out
	ADD 1
	ADD 2
	DEL 3
	DEL 2
	DEL 1

### Примечание

В чекере используется **Random** не зависящий от времени. Т.е. одинаковые решения всегда получают одинаковый результат. Если вы получили **РЕ**, ваш вывод не является корректной последовательностью из не более чем 3000 операций. Если вы получили **WA**, последовательность операций корректна, а получившееся дерево имеет глубину меньше 1000.