

## Содержание

<b>Must have</b>	<b>2</b>
<b>Задача 13А. Менеджер памяти [2 sec, 512 mb]</b>	<b>2</b>
<b>Обязательные задачи</b>	<b>3</b>
<b>Задача 13В. Менеджер памяти [4 sec, 256 mb]</b>	<b>3</b>
<b>Задача 13С. Фабризаторы-дизайнеры [2.5 sec, 256 mb]</b>	<b>4</b>
<b>Дополнительные задачи</b>	<b>5</b>
<b>Задача 13D. Опять k-я статистика [3 sec, 256 mb]</b>	<b>5</b>
<b>Задача 13Е. Формула-1 [4.5 sec, 256 mb]</b>	<b>6</b>
<b>Задача 13F. Точки в полуплоскости [2 sec, 256 mb]</b>	<b>7</b>

---

Вы не умеете читать/выводить данные, открывать файлы? Воспользуйтесь **примерами**.

В некоторых задачах большой ввод и вывод. Пользуйтесь **быстрым вводом-выводом**.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) **переопределение стандартного аллокатора** ускорит вашу программу.

Обратите внимание на компилятор GNU C++11 5.1.0 (TDM-GCC-64) `inc`, который позволяет пользоваться **дополнительной библиотекой**. Под ним можно сдать **вот это**.

## Must have

### Задача 13А. Менеджер памяти [2 sec, 512 mb]

Одно из главных нововведений новейшей операционной системы Indows 7 — новый менеджер памяти. Он работает с массивом длины  $N$  и позволяет выполнять три самые современные операции:

- `copy(a, b, l)` — скопировать отрезок длины  $[a, a+l-1]$  в  $[b, b+l-1]$
- `sum(l, r)` — посчитать сумму элементов массива на отрезке  $[l, r]$
- `print(l, r)` — напечатать элементы с  $l$  по  $r$ , включительно

Вы являетесь разработчиком своей операционной системы, и Вы, безусловно, не можете обойтись без инновационных технологий. Вам необходимо реализовать точно такой же менеджер памяти.

#### Формат входных данных

Первая строка входного файла содержит целое число  $N$  ( $1 \leq N \leq 1\,000\,000$ ) — размер массива, с которым будет работать Ваш менеджер памяти.

Во второй строке содержатся четыре числа  $1 \leq X_1, A, B, M \leq 10^9 + 10$ . С помощью них можно сгенерировать исходный массив чисел  $X_1, X_2, \dots, X_N$ .  $X_{i+1} = (A \cdot X_i + B) \bmod M$

Следующая строка входного файла содержит целое число  $K$  ( $1 \leq K \leq 10\,000$ ) — количество запросов, которые необходимо выполнить Вашему менеджеру памяти.

Далее в  $K$  строках содержится описание запросов. Запросы заданы в формате:

- `cpy a b l` — для операции `copy`
- `sum l r` — для операции `sum` ( $l \leq r$ )
- `out l r` — для операции `print` ( $l \leq r$ )

Гарантируется, что суммарная длина запросов `print` не превышает 3000. Также гарантируется, что все запросы корректны.

#### Формат выходных данных

Для каждого запроса `sum` или `print` выведите в выходной файл на отдельной строке результат запроса.

#### Пример

stdin	stdout
6	1 2 6 1 2 6
1 4 5 7	18
8	1 2
out 1 6	3
sum 1 6	1 1 2 1 2 6
cpy 1 3 2	13
out 3 4	
sum 3 4	
cpy 1 2 4	
out 1 6	
sum 1 6	

#### Замечание

Эту задачу вы умеете решать персистентным деревом и корневой.

Предполагается, что Вы напишите персистентное дерево.

## Обязательные задачи

### Задача 13В. Менеджер памяти [4 sec, 256 mb]

Абсолютно такая же, как предыдущая.  
Только ограничения жестче:  $K \leq 200\,000$ .

#### Замечание

Добавьте сборку мусора в своё персистентное дерево (ссылочный garbage collector).  
Любители `shared_ptr`, возможно, получают TL или ML.

К сожалению, скорее всего получить ОК без своего “списочного аллокатора” не получится:  
нужно заранее выделить список из  $10^6 + 1000$  вершин; переопределить `new/delete`.

Альтернативная идея аллокатора: стек + если память закончилась, собрать мусор.

### Задача 13С. Фаброзавры-дизайнеры [2.5 sec, 256 mb]

Фаброзавры известны своим тонким художественным вкусом и увлечением ландшафт-ным дизайном. Они живут около очень живописной реки и то и дело перестраивают тропинку, идущую вдоль реки: либо насыпают дополнительной земли, либо срывают то, что есть. Для того, чтобы упростить эти работы, они поделили всю тропинку на горизонтальные участки, пронумерованные от 1 до  $N$ , и их переделки устроены всегда одинаково: они выбирают часть дороги от  $L$ -ого до  $R$ -ого участка (включительно) и изменяют (увеличивают или уменьшают) высоту на всех этих участках на одну и ту же величину (если до начала переделки высоты были разными, то и после переделки они останутся разными).

Поскольку, как уже говорилось, у фаброзавров тонкий художественный вкус, каждый из них считает, что их река лучше всего выглядит с определенной высоты. Поэтому им хочется знать, есть ли поблизости от их дома место на тропинке, где высота на их взгляд оптимальна. Помогите им в этом разобраться.

#### Формат входных данных

Первая строка входного файла содержит два числа  $N$  и  $M$  — длину дороги и количество запросов соответственно ( $1 \leq N, M \leq 10^5$ ). На второй строке содержатся  $N$  чисел, разделенных пробелами — начальные высоты соответствующих частей дороги; высоты не превосходят  $10^4$  по модулю. В следующих  $M$  строках содержатся запросы по одному на строке.

Запрос  $+ L R X$  означает, что высоту частей дороги от  $L$ -ой до  $R$ -ой (включительно) нужно изменить на  $X$ . При этом  $1 \leq L \leq R \leq N$ , а  $|X| \leq 10^4$ .

Запрос  $? L R X$  означает, что нужно проверить, есть ли между  $L$ -ым и  $R$ -ым участками (включая эти участки) участок, где дорога проходит точно на высоте  $X$ . Гарантируется, что  $1 \leq L \leq R \leq N$ , а  $|X| \leq 10^9$ .

#### Формат выходных данных

На каждый запрос второго типа нужно вывести в выходной файл на отдельной строке одно слово «YES» (без кавычек), если нужный участок существует, и «NO» в противном случае.

#### Примеры

stdin	stdout
10 5	NO
0 1 1 3 3 3 2 0 0 1	YES
? 3 5 2	YES
+ 1 4 1	
? 3 5 2	
+ 7 10 2	
? 9 10 3	

#### Замечание

Классическая корневая...

Авторское решение – корневая без `split`, статичное разбиение на куски.

Что нужно хранить в куске? Можно и проще сортированный массив, можно свою хеш-таблицу (именно свою).

## Дополнительные задачи

### Задача 13D. Опять k-я статистика [3 сек, 256 mb]

Изначально вам дан массив целых чисел.

Нужно уметь отвечать на три запроса:

- $+ i x$  — Вставить на  $i$ -ю позицию число  $x$  (размер массива увеличивается на 1)
- $- i$  — Удалить число на  $i$ -й позиции (размер массива уменьшается на 1)
- $? L R x$  — Сказать, сколько чисел  $y$  на позициях  $L \leq i \leq R$  таких, что  $y \leq x$  ( $|x| \leq 10^9$ )

Все индексы  $i$ ,  $L$ ,  $R$  нумеруются с нуля. Все числа в запросах целые. Все запросы корректны. Пример запроса: “+ 0 x” означает “добавление  $x$  в начало массива”. Исходное число элементов в массиве —  $0 \leq N \leq 10^5$ , числа в массиве по модулю не превышают  $10^9$ . Число запросов —  $1 \leq K \leq 10^5$ .

#### Пример

stdin	stdout
10	1
455184306 359222813 948543704	2
914773487 861885581 253523	2
770029097 193773919 581789266	0
457415808	2
- 1	
? 2 5 527021001	
? 0 5 490779085	
? 0 5 722862778	
+ 9 448694272	
- 5	
? 1 2 285404014	
- 4	
? 3 4 993634734	
+ 0 414639071	

### Задача 13Е. Формула-1 [4.5 sec, 256 mb]

Фотокорреспондент планирует съёмки соревнований “Формулы-1”. Для этого он собирается пройти вдоль самой длинной прямой трека и сделать снимки последовательно с каждой из  $n$  запланированных точек обзора. Для каждой точки обзора известны расстояния  $a$  и  $b$  от данной точки до каждого из концов отрезка, по которому болиды проезжают мимо неё, находясь в прямой видимости.

У корреспондента есть  $m$  объективов, для каждого из которых заданы два параметра: минимальное расстояние  $c$  и максимальное расстояние  $d$  до болида, необходимое для получения качественных снимков с помощью этого объектива. Так как смена объектива — трудоёмкое занятие, то фотокорреспондент хочет для каждого объектива вычислить, каково максимальное число **последовательных** точек обзора, на которых данный объектив может быть полезен. Объектив с параметрами  $[c, d]$  является полезным в некоторой точке обзора, если болиды могут проехать мимо этой точки на некотором расстоянии  $x$  таком, что  $c \leq x \leq d$ .

Подсчитайте интересующее фотографа число для каждого из имеющихся в наличии объективов.

#### Формат входных данных

В первой строке входного файла заданы два целых числа  $n$  и  $m$  ( $1 \leq n \leq 50\,000, 1 \leq m \leq 200\,000$ ) — количество точек обзора и количество объективов у фотокорреспондента. В последующих  $n$  строках заданы параметры точек обзора в порядке их обхода корреспондентом. Для  $i$ -й точки заданы два целых числа  $a_i, b_i$  ( $1 \leq a_i \leq b_i \leq 10^9$ ), обозначающие, что болиды могут проезжать на расстоянии от  $a_i$  до  $b_i$  от данной точки, включительно. Далее следуют  $m$  строк, задающих параметры объективов. Каждый объектив задан двумя целыми числами  $c_j, d_j$  ( $1 \leq c_j \leq d_j \leq 10^9$ ) — минимальным и максимальным расстоянием до болида, требуемым для получения качественных снимков.

#### Формат выходных данных

Для каждого из  $m$  объективов в порядке их перечисления во входном файле выведите одно число — наибольшее количество идущих подряд точек обзора, на которых данный объектив является полезным.

#### Примеры

stdin	stdout
3 4	2
2 5	3
1 3	0
6 6	1
3 5	
1 10	
7 9	
5 6	

### Задача 13F. Точки в полуплоскости [2 sec, 256 mb]

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 входных данных

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 выходных данных

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

#### Пример

stdin	stdout
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	