

## А. Простое двоичное дерево поиска

2 секунды, 512 мегабайт

Реализуйте просто двоичное дерево поиска.

### Входные данные

Входной файл содержит описание операций с деревом, их количество не превышает 100. В каждой строке находится одна из следующих операций:

- `insert  $x$`  — добавить в дерево ключ  $x$ . Если ключ  $x$  есть в дереве, то ничего делать не надо;
- `delete  $x$`  — удалить из дерева ключ  $x$ . Если ключа  $x$  в дереве нет, то ничего делать не надо;
- `exists  $x$`  — если ключ  $x$  есть в дереве выведите «true», если нет «false»;
- `next  $x$`  — выведите минимальный элемент в дереве, строго больший  $x$ , или «none» если такого нет;
- `prev  $x$`  — выведите максимальный элемент в дереве, строго меньший  $x$ , или «none» если такого нет.

В дерево помещаются и извлекаются только целые числа, не превышающие по модулю  $10^9$ .

### Выходные данные

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`. Следуйте формату выходного файла из примера.

входные данные
<code>insert 2</code> <code>insert 5</code> <code>insert 3</code> <code>exists 2</code> <code>exists 4</code> <code>next 4</code> <code>prev 4</code> <code>delete 5</code> <code>next 4</code> <code>prev 4</code>
выходные данные
<code>true</code> <code>false</code> <code>5</code> <code>3</code> <code>none</code> <code>3</code>

## В. Сбалансированное двоичное дерево поиска

2 секунды, 512 мегабайт

Реализуйте сбалансированное двоичное дерево поиска.

### Входные данные

Входной файл содержит описание операций с деревом, их количество не превышает  $10^5$ . В каждой строке находится одна из следующих операций:

- `insert  $x$`  — добавить в дерево ключ  $x$ . Если ключ  $x$  есть в дереве, то ничего делать не надо;
- `delete  $x$`  — удалить из дерева ключ  $x$ . Если ключа  $x$  в дереве нет, то ничего делать не надо;
- `exists  $x$`  — если ключ  $x$  есть в дереве выведите «true», если нет «false»;
- `next  $x$`  — выведите минимальный элемент в дереве, строго больший  $x$ , или «none» если такого нет;
- `prev  $x$`  — выведите максимальный элемент в дереве, строго меньший  $x$ , или «none» если такого нет.

В дерево помещаются и извлекаются только целые числа, не превышающие по модулю  $10^9$ .

### Выходные данные

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`. Следуйте формату выходного файла из примера.

#### ВХОДНЫЕ ДАННЫЕ

```
insert 2
insert 5
insert 3
exists 2
exists 4
next 4
prev 4
delete 5
next 4
prev 4
```

#### ВЫХОДНЫЕ ДАННЫЕ

```
true
false
5
3
none
3
```

## С. Добавление ключей

2 секунды, 256 мегабайт

Вы работаете в компании Макрохард и вас попросили реализовать структуру данных, которая будет хранить множество целых ключей.

Будем считать, что ключи хранятся в бесконечном массиве  $A$ , проиндексированном с 1, исходно все его ячейки пусты. Структура данных должна поддерживать следующую операцию:

$\text{Insert}(L, K)$ , где  $L$  — позиция в массиве, а  $K$  — некоторое положительное целое число.

Операция должна выполняться следующим образом:

- Если ячейка  $A[L]$  пуста, присвоить  $A[L] \geq K$ .
- Если  $A[L]$  непуста, выполнить  $\text{Insert}(L + 1, A[L])$  и затем присвоить  $A[L] \geq K$ .

По заданным  $N$  целым числам  $L_1, L_2, \dots, L_N$  выведите массив после выполнения последовательности операций:

$\text{Insert}(L_1, 1) \text{Insert}(L_2, 2) \dots \text{Insert}(L_N, N)$

### Входные данные

Первая строка входного файла содержит числа  $N$  — количество операций  $\text{Insert}$ , которое следует выполнить и  $M$  — максимальную позицию, которая используется в операциях  $\text{Insert}$  ( $1 \leq N \leq 131\,072$ ,  $1 \leq M \leq 131\,072$ ).

Следующая строка содержит  $N$  целых чисел  $L_i$ , которые описывают операции  $\text{Insert}$ , которые следует выполнить ( $1 \leq L_i \leq M$ ).

### Выходные данные

Выведите содержимое массива после выполнения всех сделанных операций  $\text{Insert}$ . На первой строке выведите  $W$  — номер максимальной непустой ячейки в массиве. Затем выведите  $W$  целых чисел —  $A[1], A[2], \dots, A[W]$ . Выводите нули для пустых ячеек.

входные данные
5 4 3 3 4 1 3
выходные данные
6 4 0 5 2 3 1

## D. И снова сумма

3 секунды, 256 мегабайт

Реализуйте структуру данных, которая поддерживает множество  $S$  целых чисел, с которым разрешается производить следующие операции:

- $\text{add}(i)$  — добавить в множество  $S$  число  $i$  (если он там уже есть, то множество не меняется);
- $\text{sum}(l, r)$  — вывести сумму всех элементов  $x$  из  $S$ , которые удовлетворяют неравенству  $l \leq x \leq r$ .

### Входные данные

Исходно множество  $S$  пусто. Первая строка входного файла содержит  $n$  — количество операций ( $1 \leq n \leq 300\,000$ ). Следующие  $n$  строк содержат операции. Каждая операция имеет вид либо «+  $i$ », либо «?  $l\ r$ ». Операция «?  $l\ r$ » задает запрос  $\text{sum}(l, r)$ .

Если операция «+  $i$ » идет во входном файле в начале или после другой операции «+», то она задает операцию  $\text{add}(i)$ . Если же она идет после запроса «?», и результат этого запроса был  $y$ , то выполняется операция  $\text{add}((i + y) \bmod 10^9)$ .

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до  $10^9$ .

### Выходные данные

Для каждого запроса выведите одно число — ответ на запрос.

входные данные
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4
выходные данные
3 7

# $K$ -й максимум

2 секунды, 512 мегабайт

Напишите программу, реализующую структуру данных, позволяющую добавлять и удалять элементы, а также находить  $k$ -й максимум.

## Входные данные

Первая строка входного файла содержит натуральное число  $n$  — количество команд ( $n \leq 100\,000$ ).

Последующие  $n$  строк содержат по одной команде каждая. Команда записывается в виде двух чисел  $c_i$  и  $k_i$  — тип и аргумент команды соответственно ( $|k_i| \leq 10^9$ ). Поддерживаемые команды:

- $+1$  (или просто  $1$ ): Добавить элемент с ключом  $k_i$ .
- $0$ : Найти и вывести  $k_i$ -й максимум.
- $-1$ : Удалить элемент с ключом  $k_i$ .

Гарантируется, что в процессе работы в структуре не требуется хранить элементы с равными ключами или удалять несуществующие элементы. Также гарантируется, что при запросе  $k_i$ -го максимума, он существует.

## Выходные данные

Для каждой команды нулевого типа в выходной файл должна быть выведена строка, содержащая единственное число —  $k_i$ -й максимум.

### входные данные

```
11
+1 5
+1 3
+1 7
0 1
0 2
0 3
-1 5
+1 10
0 1
0 2
0 3
```

### выходные данные

```
7
5
3
10
7
3
```

## Г. Неявный ключ

2 секунды, 256 мегабайт

Научитесь быстро делать две операции с массивом:  $\odot$  `add i x` — добавить после  $i$ -го элемента  $x$  ( $0 \leq i \leq n$ )  $\odot$  `del i` — удалить  $i$ -й элемент ( $1 \leq i \leq n$ )

### Входные данные

На первой строке  $n_0$  и  $m$  ( $1 \leq n_0, m \leq 10^5$ ) — длина исходного массива и количество запросов. На второй строке  $n_0$  целых чисел от 0 до  $10^9 - 1$  — исходный массив. Далее  $m$  строк, содержащие запросы. Гарантируется, что запросы корректны: например, если просят удалить  $i$ -й элемент, он точно есть.

### Выходные данные

Выведите конечное состояние массива. На первой строке количество элементов, на второй строке сам массив.

входные данные
<pre>3 4 1 2 3 del 3 add 0 9 add 3 8 del 2</pre>
выходные данные
<pre>3 9 2 8</pre>

## Г. Переместить в начало

6 секунд, 512 мегабайт

Вам дан массив  $a_1 = 1, a_2 = 2, \dots, a_n = n$  и последовательность операций: переместить элементы с  $l_i$  по  $r_i$  в начало массива. Например, для массива 2, 3, 6, 1, 5, 4, после операции (2, 4) новый порядок будет 3, 6, 1, 2, 5, 4. А после применения операции (3, 4) порядок элементов в массиве будет 1, 2, 3, 6, 5, 4.

Выведите порядок элементов в массиве после выполнения всех операций.

### Входные данные

В первой строке входного файла указаны числа  $n$  и  $m$  ( $2 \leq n \leq 100\,000$ ,  $1 \leq m \leq 100\,000$ ) — число элементов в массиве и число операций. Следующие  $m$  строк содержат операции в виде двух целых чисел:  $l_i$  и  $r_i$  ( $1 \leq l_i \leq r_i \leq n$ ).

### Выходные данные

Выведите  $n$  целых чисел — порядок элементов в массиве после применения всех операций.

входные данные
6 3 2 4 3 5 2 2
выходные данные
1 4 5 2 3 6

## Н. Развороты

1 секунда, 512 мегабайт

Вам дан массив  $a_1 = 1, a_2 = 2, \dots, a_n = n$  и последовательность операций: переставить элементы с  $l_i$  по  $r_i$  в обратном порядке. Например, для массива 1, 2, 3, 4, 5, после операции (2, 4) новый порядок будет 1, 4, 3, 2, 5. А после применения операции (3, 5) порядок элементов в массиве будет 1, 4, 5, 2, 3.

Выведите порядок элементов в массиве после выполнения всех операций.

### Входные данные

В первой строке входного файла указаны числа  $n$  и  $m$  ( $2 \leq n \leq 100\,000$ ,  $1 \leq m \leq 100\,000$ ) — число элементов в массиве и число операций. Следующие  $m$  строк содержат операции в виде двух целых чисел:  $l_i$  и  $r_i$  ( $1 \leq l_i \leq r_i \leq n$ ).

### Выходные данные

Выведите  $n$  целых чисел — порядок элементов в массиве после применения всех операций.

входные данные
5 3 2 4 3 5 2 2
выходные данные
1 4 5 2 3



# I. Эх, дороги

2 секунды, 256 мегабайт

В многострадальном Тридесятom государстве опять готовится дорожная реформа. Впрочем, надо признать, дороги в этом государстве находятся в довольно плачевном состоянии. Так что реформа не повредит. Одна проблема — дорожникам не развернуться, поскольку в стране действует жесткий закон — из каждого города должно вести не более двух дорог. Все дороги в государстве двусторонние, то есть по ним разрешено движение в обоих направлениях (разумеется, разметка отсутствует). В результате реформы некоторые дороги будут строиться, а некоторые другие закрываться на бессрочный ремонт.

Петя работает диспетчером в службе грузоперевозок на дальние расстояния. В связи с предстоящими реформами, ему необходимо оперативно определять оптимальные маршруты между городами в условиях постоянно меняющейся дорожной ситуации. В силу большого количества пробок и сотрудников дорожной полиции в городах, критерием оптимальности маршрута считается количество промежуточных городов, которые необходимо проехать.

Помогите Пете по заданной последовательности сообщений об изменении структуры дорог и запросам об оптимальном способе проезда из одного города в другой, оперативно отвечать на запросы.

## Входные данные

В первой строке входного файла заданы числа  $n$  — количество городов,  $m$  — количество дорог в начале реформы и  $q$  — количество сообщений об изменении дорожной структуры и запросов ( $1 \leq n, m \leq 100\,000$ ,  $q \leq 200\,000$ ). Следующие  $m$  строк содержат по два целых числа каждая — пары городов, соединенных дорогами перед реформой. Следующие  $q$  строк содержат по три элемента, разделенных пробелами. «+  $i$   $j$ » означает строительство дороги от города  $i$  до города  $j$ , «-  $i$   $j$ » означает закрытие дороги от города  $i$  до города  $j$ , «?  $i$   $j$ » означает запрос об оптимальном пути между городами  $i$  и  $j$ .

Гарантируется, что в начале и после каждого изменения никакие два города не соединены более чем одной дорогой, и из каждого города выходит не более двух дорог. Никакой город не соединяется дорогой сам с собой.

## Выходные данные

На каждый запрос вида «?  $i$   $j$ » выведите одно число — минимальное количество промежуточных городов на маршруте из города  $i$  в город  $j$ . Если проехать из  $i$  в  $j$  невозможно, выведите -1.

### входные данные

```
5 4 6
1 2
2 3
1 3
4 5
? 1 2
? 1 5
- 2 3
? 2 3
+ 2 4
? 1 5
```

### выходные данные

```
0
-1
1
2
```