

## Лабораторная работа 1-5. Дерево поиска

### 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

```
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
```

### 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
<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>

С. Декартово дерево

2 секунды, 256 мегабайт

Вам даны пары чисел  $(a_i, b_i)$ . Необходимо построить декартово дерево, такое что  $i$ -я вершина имеет ключи  $(a_i, b_i)$ , вершины с ключом  $a_i$  образуют бинарное дерево поиска, а вершины с ключом  $b_i$  образуют кучу.

**Входные данные**

В первой строке записано число  $N$  — количество пар. Далее следует  $N$  ( $1 \leq N \leq 300\,000$ ) пар  $(a_i, b_i)$ . Для всех пар  $|a_i|, |b_i| \leq 1\,000\,000$ .  $a_i \neq a_j$  и  $b_i \neq b_j$  для всех  $i \neq j$ .

**Выходные данные**

Если декартово дерево с таким набором ключей построить возможно, выведите в первой строке «YES», в противном случае выведите «NO». В случае ответа «YES» выведите  $N$  строк, каждая из которых должна описывать вершину. Описание вершины состоит из трёх чисел: номера предка, номера левого сына и номера правого сына. Если у вершины отсутствует предок или какой либо из сыновей, выведите на его месте число 0.

Если подходящих деревьев несколько, выведите любое.

входные данные
<code>7</code> <code>5 4</code> <code>2 2</code> <code>3 9</code> <code>0 5</code> <code>1 3</code> <code>6 6</code> <code>4 11</code>
выходные данные
<code>YES</code> <code>2 3 6</code> <code>0 5 1</code> <code>1 0 7</code> <code>5 0 0</code> <code>2 4 0</code> <code>1 0 0</code> <code>3 0 0</code>

Условие  
недоступно  
на

## Е. 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.

<b>f. <math>k</math>-й максимум</b>
11 +1 5 +1 3 +1 7 0 1 0 2 0 3 -1 5 +1 10 0 1 0 2 0 3
<b>выходные данные</b>
7 5 3 10 7 3

## G. 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.

<b>входные данные</b>
6 3 2 4 3 5 2 2
<b>выходные данные</b>
1 4 5 2 3 6

## H. Различные буквы

2 секунды, 256 мегабайт

Вы работаете со списком из строчных латинских букв. Изначально список пуст. Вы должны поддерживать следующие операции:

- **insert** index number letter — добавить number букв letter перед буквой с индексом index.
- **remove** index number — удалить number букв, начиная с индекса index.
- **query** index\_1 index\_2 — вывести количество различных букв на отрезке с index\_1 до index\_2 включительно.

Буквы нумеруются с 1.

### Входные данные

В первой строке входного файла содержится единственное целое число  $n$  — количество операций ( $1 \leq n \leq 30\,000$ ). Следующие по  $n$  строк содержат описание операций.

Описание операции начинается с типа операции: '+' для добавления, '-' для удаления и '?' для запроса. Далее следует аргументы запроса, описанные в условиях выше.

Все запросы корректны, элементы с такими индексами существуют, нет запросов на удаление несуществующих элементов.

number добавления, удаления не превышает 10 000.

Выходные данные

Для каждого запроса query выведите одно целое число — количество различных букв на отрезке index\_1, index\_2 включительно.

входные данные
8 + 1 4 w + 3 3 o ? 2 3 - 2 2 ? 2 3 + 2 2 t ? 1 6 - 1 6
выходные данные
2 1 3

Пояснение к примеру:

- 1. wwww
- 2. wwooooww
- 3. w[wo]oooww : 2 различные буквы
- 4. wooww
- 5. w[oo]ww : 1 буква
- 6. wttooww
- 7. [wttoow]w : 3 различные буквы
- 8. w

I. Эх, дороги

2 секунды, 256 мегабайт

В многострадальном Тридесятом 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