y2018-2-2. Дерево поиска

Statement is not available on English language

A. 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

B. Balanced binary search tree

2 seconds, 512 megabytes

Implement a balanced binary search tree.

Input

The input contains several operations with the tree and their amount does not exceed 10^5. Every line contains one of the following operations:

- `insert` x — insert key x into the tree. If there is a key x in the tree already, do nothing;
- `delete` x — remove key x from the tree. If there is no key x in the tree, do nothing;
- `exists` x — if there is a key x in the tree, output "`true`", otherwise output "`false`";
- `next` x — output the smallest key in the tree that is strictly larger than x, or "`none`" if there's no such key;
- `prev` x — output the largest key in the tree that is strictly less than x, or "`none`" if there's no such key.

All keys are integers no greater than 10^9 by absolute value.

Output

Output results of all operations `exists`, `next`, `prev`.

input
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4
output
true false 5 3 none 3

C. Cartesian Tree

2 seconds, 256 megabytes

You are given pairs of integers (a_i, b_i). You have to build a Cartesian Tree, such that i-th node has keys (a_i, b_i), where the tree with keys a_i represents a binary search tree, and the tree with keys b_i represents a heap.

Input

The first line contains one integer n $(1 \le N \le 300\,000)$ — the number of pairs. Then, you are given n pairs (a_i, b_i), one per line. For all pairs two conditions hold: $|a_i|, |b_i| \le 1\,000\,000$, and $a_i \ne a_j$ and $b_i \ne b_j$ for all $i \ne j$.

Output

If it is possible to build a Cartesian Tree with such keys then the first line should contain "`YES`", otherwise, it should contain "`NO`". If the answer is "`YES`" the output n lines with the description of a corresponding node. The description consists of three numbers: an identifier of a parent, an identifier of the left child and an identifier of the right child. If the node does not have a parent or a child, output 0 instead of an identifier.

If there are several trees, output any of them.

input
7 5 4 2 2 3 9 0 5 1 3 6 6 4 11
output
YES 2 3 6 0 5 1 1 0 7 5 0 0 2 4 0 1 0 0 3 0 0

D. Key insertion

2 seconds, 256 megabytes

You are working in the company Macrohard. And you are asked to implement a data structure that will store a set of integer keys.

Suppose that keys are stored in an infinite array A, indexed from 1. Initially, all its elements are empty. The structure should have the following operation:

`Insert(`L`,` K`)`, where L is a position in the array, and K is some positive integer.

The operation should work as follows:

- If element $A[L]$ is empty, then assign $A[L] \geq ts\ K$.
- Otherwise, i.e., $A[L]$ is not empty, call `Insert(`$L+1$`,` $A[L]$`)` and then assign $A[L] \geq ts\ K$.

By given N integers $L_1, L_2, ..., L_N$ output an array after calling operations:

`Insert(`L_1`,` 1`)` `Insert(`L_2`,` 2`)` ... `Insert(`L_N`,` N`)`

Input
The first line contains two integers n and m ($1 \leq n \leq 131\,072$, $1 \leq m \leq 131\,072$) — the number of `Insert` operations and the maximum position which is used in one of insert operations `Insert`.

Next line contains n integers L_i ($1 \leq L_i \leq m$), which describes position of `Insert` operations, which you should call.

Output
Output an array after calling all operations. First line should contain integer W — largest index of a non-empty element in the array. Then output W integers — $A[1], A[2], ..., A[W]$. Output zeroes for empty elements.

input
5 4 3 3 4 1 3
output
6 4 0 5 2 3 1

E. Sum Problem

3 seconds, 256 megabytes

Implement a data structure that supports the set of S integers with which the following operations are allowed:

- $\mathrm{add}(i)$ — add the number i to the set S (if it is already there, the set does not change);
- $\mathrm{sum}(l, r)$ — output the sum of all elements x from S that satisfy the inequality $l \leq x \leq r$.

Input
Initially, the set S is empty. The first line of the input file contains n — the number of operations ($1 \leq n \leq 300\,000$). The next n lines contain operations. Each operation has the form either "`+` i" or "`?` l r". Operation "`?` l r" sets the query to $\mathrm{sum}(l, r)$.

If the operation "`+` i" is in the input file at the beginning or after another operation "+", then it defines the operation $\mathrm{add}(i)$. If it goes after the query "?", and the result of this query was y, then the operation $\mathrm{add}((i + y) \bmod 10^9)$ is performed.

In all queries and adding operations, the parameters are in the range from 0 to 10^9.

Output
For each request print one number — the response to the request.

input
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4
output
3 7

F. k-th

2 seconds, 512 megabytes

Write a program that implements a data structure that allows you to add and remove elements, and also find the k-th maximum.

Input
The first line of the input file contains a natural number n ($1 \leq n \leq 100\,000$) — the number of commands. Next n lines contain one command each. The command is written in the form of two numbers c_i and k_i ($|k_i| \leq 10^9$) — the type and the argument of the command, respectively. Supported Commands:

- `+1` (or just `1`): Add an item with the key k_i.
- `0`: Find and print the k_i-th maximum.
- `-1`: Delete the item with the key k_i.

It is guaranteed that the structure is not required to store elements with equal keys or to delete nonexistent elements. It is also guaranteed that when requesting the k_i-th maximum, it exists.

Output
Output the result of each command of the second type on a separate line.

input
11 +1 5 +1 3 +1 7 0 1 0 2 0 3 -1 5 +1 10 0 1 0 2 0 3
output
7 5 3 10 7 3

G. Move to front

6 seconds, 512 megabytes

You have an array $a_1 = 1, a_2 = 2, ..., a_n = n$ and a sequence of queries: move elements from l_i to r_i to front.

For example, if the array is $2, 3, 6, 1, 5, 4$, after the query $(2, 4)$ the new order of elements in the array is $3, 6, 1, 2, 5, 4$.

If, for example, the query $(3, 4)$ follows, the new order of elements is $1, 2, 3, 6, 5, 4$.

Print the final order of elements in the array.

Input

The first line of the input file contains two integer numbers n and m ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$) — the number of elements and the number of queries. The following m lines contain queries, each line contains two integer numbers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output

Output n integer numbers — the order of elements in the final array, after executing all queries.

input
6 3 2 4 3 5 2 2
output
1 4 5 2 3 6

Statement is not available on English language

Statement is not available on English language

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 ($1 \le n, m \le 100\,000$, $q \le 200\,000$). 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

Codeforces (c) Copyright 2010-2021 Mike Mirzayanov
The only programming contests Web 2.0 platform