

A. BST или нет?

1 секунда, 256 мегабайт

Вам дано какое-то двоичное дерево. Проверьте, является ли оно BST.

Входные данные

В первой строке дано целое число n — количество вершин в дереве ($1 \leq n \leq 100000$).

В следующих n строках даны описания вершин. Каждая строка содержит число x , содержащееся в вершине, затем два числа l и r — номера левого и правого ребёнка, или -1, если ребёнка нет. ($1 \leq x \leq 10^9$, $1 \leq l, r \leq n$).

В последней строке дан номер вершины, являющейся корнем.

Вершины нумеруются в порядке, в котором они перечислены во входных данных. Гарантируется, что заданная структура является бинарным деревом.

Выходные данные

Выведите «YES», если данное дерево является BST, иначе «NO»

входные данные
3 10 2 3 5 -1 -1 11 -1 -1 1
выходные данные
YES

входные данные
2 1 2 -1 2 -1 -1 1
выходные данные
NO

B. Постройте BST

1 секунда, 256 мегабайт

Входные данные

Вам даны n чисел. Постройте какое-нибудь BST с этими числами.

Выходные данные

Выведите BST в формате из предыдущей задачи.

входные данные
3 1 2 3
выходные данные
3 2 2 3 1 -1 -1 3 -1 -1 1

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

D. 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
<pre>insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4</pre>
выходные данные
<pre>true false 5 3 none 3</pre>

Е. И снова сумма

5 секунд, 512 мегабайт

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

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

Исходно множество S пусто.

Входные данные

Первая строка 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 `sum(l, r)`.

Если операция «+ i » идет в начале или после другой операции «+», то она задает операцию `add(i)`. Если же она идет после запроса «?», и результат этого запроса был y , то выполняется операция `add((i + y) mod ...)`.

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
<pre>6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4</pre>
выходные данные
<pre>3 7</pre>

К-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 s_i и k_i — тип и аргумент команды соответственно ($|k_i| \leq 10^9$).

Поддерживаемые команды:

- 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
<pre>11 1 5 1 3 1 7 0 1 0 2 0 3 -1 5 1 10 0 1 0 2 0 3</pre>
выходные данные
<pre>7 5 3 10 7 3</pre>

Г. Переместить в начало

3 с, 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. Добавление ключей

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

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 Insert , которое следует выполнить и M — максимальную позицию, которая используется в операциях Insert ($1 \leq N \leq 131\,072, 1 \leq M \leq 131\,072$).

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

Выходные данные

Выведите содержимое массива после выполнения всех сделанных операций 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

I. Развороты

3 с, 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

J. RSQ

3 с, 256 мегабайт

Входные данные

В первой строке находится число n — размер массива ($1 \leq n \leq 500\,000$).

Во второй строке находится n чисел a_i — элементы массива.

Далее содержится описание операций. Количество операций не превышает $1\,000\,000$.

В каждой строке находится одна из следующих операций:

- **set** $i\ x$ — присвоить элементу a_i значение x ;
- **sum** $i\ j$ — вывести значение суммы элементов на отрезке массива от i до j .

Гарантируется, что $1 \leq i \leq j \leq n$. Все числа во входных данных и результаты выполнения всех операций не превышают по модулю 10^{18} .

Выходные данные

Выведите последовательно результат выполнения всех операций **sum**. Следуйте формату выходных данных из примера.

входные данные
5 1 2 3 4 5 sum 2 5 sum 1 5 sum 1 4 sum 2 4 set 1 10 set 2 3 set 5 2 sum 2 5 sum 1 5 sum 1 4 sum 2 4
выходные данные
14 15 10 9 12 22 20 10

К. Максимум на подотрезках с добавлением на отрезке

2.5 с, 256 мегабайт

Реализуйте эффективную структуру данных для хранения массива и выполнения следующих операций: увеличение всех элементов данного интервала на одно и то же число; поиск максимума на интервале.

Входные данные
В первой строке вводится одно натуральное число N ($1 \leq N \leq 100000$) — количество чисел в массиве.

Во второй строке вводятся N чисел от 0 до 100000 — элементы массива.

В третьей строке вводится одно натуральное число M ($1 \leq M \leq 30000$) — количество запросов.

Каждая из следующих M строк представляет собой описание запроса. Сначала вводится одна буква, кодирующая вид запроса (m — найти максимум, a — увеличить все элементы на отрезке).

Следом за m вводятся два числа — левая и правая граница отрезка.

Следом за a вводятся три числа — левый и правый концы отрезка и число add , на которое нужно увеличить все элементы данного отрезка массива ($0 \leq add \leq 100000$).

Выходные данные
Выведите в одну строку через пробел ответы на каждый запрос m .

входные данные
5 2 4 3 1 5 5 m 1 3 a 2 4 100 m 1 3 a 5 5 10 m 1 5

выходные данные
4 104 104

L. RMQ2

8 с, 256 мегабайт

Входные данные
В первой строке находится число n — размер массива ($1 \leq n \leq 10^5$).

Во второй строке находится n чисел a_i — элементы массива.

Далее содержится описание операций. Количество операций не превышает $2 \cdot 10^5$.

В каждой строке находится одна из следующих операций:

- set** $i\ j\ x$ — присвоить значение x всем таким элементам a_k , что $i \leq k \leq j$;
- add** $i\ j\ x$ — прибавить значение x ко всем таким элементам a_k , что $i \leq k \leq j$;
- min** $i\ j$ — вывести значение минимального элемента на отрезке массива от i до j .

Гарантируется, что $1 \leq i \leq j \leq n$. Все числа во входных данных и результаты выполнения всех операций не превышают по модулю 10^{18} .

Выходные данные
Выведите последовательно результат выполнения всех операций **min**. Следуйте формату выходных данных из примера.

входные данные
5 1 2 3 4 5 min 2 5 min 1 5 min 1 4 min 2 4 set 1 3 10 add 2 4 4 min 2 5 min 1 5 min 1 4 min 2 4
выходные данные
2 1 1 2 5 5 8 8