Содержание

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

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 `GNU C++` компиляторы с суффиксом `inc`, они позволяют пользоваться дополнительной библиотекой. Под ними можно сдать вот это.

Задачи здорового человека

Задача 2A. Простейший полуявный ключ [0.3 sec, 256 mb]

В этой задаче вам нужно написать BST по **явному** ключу и отвечать им на запросы:

- `+ x` – добавить в дерево `x` (если `x` уже есть, ничего не делать).
- `? k` – вернуть k-й по возрастанию элемент.

Формат входных данных

В каждой строке содержится один запрос.
Все x целые от 1 до 10^9, количество запросов от 1 до 300 000.
Гарантируется, что все x выбраны равномерным распределением.
В запросах «`? k`», число k от 1 до количества элементов в дереве.

Формат выходных данных

Для каждого запроса вида «`? k`» выведите в отдельной строке ответ.

Пример

stdin	stdout
+ 1 + 4 + 3 + 3 ? 1 ? 2 ? 3 + 2 ? 3	1 3 4 3

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

Вам нужно в каждой вершине поддерживать размер поддерева.

Замечание

`set/tree` использовать нельзя, все решения с ним будут баниться!

Задача 2B. K-ый максимум [0.2 sec, 256 mb]

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

Формат входных данных

Первая строка входного файла содержит натуральное число n — количество команд ($n \leqslant 100\,000$). Последующие n строк содержат по одной команде каждая. Команда записывается в виде двух чисел c_i и k_i — тип и аргумент команды соответственно ($|k_i| \leqslant 10^9$). 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.

Формат выходных данных

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

Пример

stdin	stdout
11 +1 5 +1 3 +1 7 0 1 0 2 0 3 -1 5 +1 10 0 1 0 2 0 3	7 5 3 10 7 3

Подсказка по решению

Псевдокод: `def rotate(v): return node(v.l.x, v.l.l, node(v.x, v.l.r, v.r))`

Напишите сюда, пожалуйста, **AVL-дерево** или **Treap**.

Без полноценного дерева с перебалансировками уже не обойтись.

Дерево нужно по **явному** ключу. Чтобы находить k-й элемент, понимать спускаться влево, или вправо, достаточно хранить размеры поддеревьев (и пересчитывать).

Замечание

AVL: круто, если напишите честное удаление, но гораздо проще удалять лениво (на практике обсудим). **Treap:** круто, если Insert/Delete будете делать через один Split/Merge (на практике обсудим), но через три тоже можно.

Задача 2C. Простейший неявный Ключ [0.2 sec, 256 mb]

Изначально есть пустой массив. Вам нужно обрабатывать запросы вида
`i x` — добавить после i-го элемента x $(0 \leqslant i \leqslant n)$, где n текущая длина массива.
В конце после всех добавлений нужно вывести полученный массив.

Формат входных данных

q $(1 \leqslant q \leqslant 100\,000)$ строк, на каждой запрос «`i x`» —
пара целых чисел $(0 \leqslant i \leqslant n,\ 1 \leqslant x < 100\,000)$.

Формат выходных данных

Выведите q целых чисел — полученный массив.

Примеры

stdin	stdout
0 1 0 2 0 3	3 2 1
0 1 1 2 2 3	1 2 3
0 1 1 2 1 3 0 4	4 1 3 2

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

Вам нужно в каждой вершине поддерживать размер поддерева.
В этой задаче вам нужно написать BST по **не**явному ключу.

Для искателей острых ощущений

Задача 2D. Неявный Ключ [0.6 sec, 256 mb]

Научитесь быстро делать две операции с массивом:

- `add i x` — добавить после i-го элемента x $(0 \leqslant i \leqslant n)$
- `del i` — удалить i-й элемент $(1 \leqslant i \leqslant n)$

Формат входных данных

На первой строке n_0 и m $(1 \leqslant n_0, m \leqslant 10^5)$ — 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.

Формат выходных данных

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

Примеры

stdin	stdout
3 4 1 2 3 del 3 add 0 9 add 3 8 del 2	3 9 2 8

Подсказка по решению

Сюда нужно писать ABL или Treap.

Задача 2E. Range Minimum Query [0.8 sec, 256 mb]

Компания *Giggle* открывает свой новый офис в Судиславле, и вы приглашены на собеседование. Ваша задача — решить поставленную задачу.

Вам нужно создать структуру данных, которая представляет из себя массив целых чисел. Изначально массив пуст. Вам нужно поддерживать две операции:

- запрос: «`? i j`» — возвращает минимальный элемент между i-ым и j-м, включительно;
- изменение: «`+ i x`» — добавить элемент x после i-го элемента списка.
  Если $i = 0$, то элемент добавляется в начало массива.

Конечно, эта структура должна быть достаточно хорошей.

Формат входных данных

Первая строка входного файла содержит единственное целое число n — число операций над массивом ($1 \leqslant n \leqslant 200\,000$). Следующие n строк описывают сами операции. Все операции добавления являются корректными. Все числа, хранящиеся в массиве, по модулю не превосходят 10^9.

Формат выходных данных

Для каждой операции в отдельной строке выведите её результат.

Примеры

stdin	stdout
8 + 0 5 + 1 3 + 1 4 ? 1 2 + 0 2 ? 2 4 + 4 1 ? 3 5	4 3 1

Подсказка по решению

Сюда нужно писать ABL или Treap. Удобнее Treap, чтобы были split/merge для ответа на запрос. Неявный ключ, конечно.