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.

Must have

Задача 10A. Простейшее BST [0.3 sec, 256 mb]

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

- `+ x` – добавить в дерево `x` (если `x` уже есть, ничего не делать).
- `> x` – вернуть минимальный элемент больше `x` или 0, если таких нет.

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.

Формат выходных данных

Для каждого запроса вида «`> x`» выведите в отдельной строке ответ.

Пример

stdin	stdout
+ 1 + 3 + 3 > 1 > 2 > 3 + 2 > 1	3 3 0 2

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

```
struct node {
    node *l, *r;
    int x;
};
```

Замечание

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

Задача 10B. Простейший неявный ключ [0.3 sec, 256 mb]

В этой задаче вам нужно написать BST по **не**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.

Замечание

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

Задача 10C. Двоичное дерево поиска [0.3 sec, 256 mb]

Воспользуйтесь любой стандартной структурой из `C++::STL`.
Или реализуйте сбалансированное двоичное дерево поиска.
Или попытайтесь запихать квадрат.

Формат входных данных

Входной файл содержит описание одной или нескольких операций с деревом.
Операций не больше 10^5. Все числа целые от -10^5 до 10^9.
В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x.
  Если ключ x в дереве уже есть, 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, «`true`», иначе «`false`»
- `next` x — минимальный элемент в дереве, $> x$, или «`none`», если такого нет.
- `prev` x — максимальный элемент в дереве, $< x$, или «`none`», если такого нет.

Формат выходных данных

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

Примеры

stdin	stdout
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4	true false 5 3 none 3

Подсказка по решению

`while (!seekEof()) { ... }`
`set`, умный квадрат, сбалансированное bst.

Замечание

`set/tree` использовать можно!

Обязательные задачи

Задача 10D. 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

Замечание

Напишите сюда, пожалуйста, именно **AVL дерево**.
Круто, если напишите честное удаление, но гораздо проще удалять лениво.

Подсказка по решению

Из лекции мы знаем, что вращений в добавлении будет $\mathcal{O}(1) \Rightarrow$ оптимизировать вращение не нужно. Из допдз мы также знаем, что даже при уадлении высота поменяется в среднем у $\mathcal{O}(1)$ вершин.

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

Задача 10E. И снова сумма... [2 sec, 256 mb]

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

- $add(i)$ — добавить в множество S число i, если i там уже есть, S не меняется;
- $sum(l, r)$ — вывести сумму всех элементов x из S: $l \leqslant x \leqslant r$.

Формат входных данных

Исходно множество S пусто. Первая строка входного файла содержит n — количество операций ($1 \leqslant n \leqslant 300\,000$). 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 во входном файле 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 $add((i + y) \bmod 10^9)$.

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

Формат выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

Пример

stdin	stdout
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7

Задача 10F. 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

Задача 10G. 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

Дополнительные задачи

Задача 10H. Перестановки [4 sec, 256 mb]

Рассмотрим циклический алфавит, состоящий из первых десяти букв английского алфавита. Циклическим он называется потому, что следующей буквой за ‘a’ является буква ‘b’, за ‘b’ — ‘c’ и так далее, и при этом следующей за буквой ‘j’ является буква ‘a’.

Вам дана строка S, состоящая из букв этого циклического алфавита. Вы должны обрабатывать запросы трёх типов:

- Развернуть подстроку строки S с L-го по R-й символ включительно.
- В подстроке S с L-го по R-й символ заменить каждый символ на D-й следующий символ в циклическом алфавите.
- Для подстроки S с L-го по R-й символ найти количество различных перестановок символов этой подстроки по модулю $10^9 + 7$.

Формат входных данных

Первая строка входного файла содержит одно число N ($1 \leqslant N \leqslant 10^5$) — длину строки S. Во второй строке содержится сама строка S, состоящая из строчных букв английского алфавита от ‘a’ до ‘j’. Третья строка содержит число M ($1 \leqslant M \leqslant 10^5$) — количество запросов. Затем идут M строк, содержащие описания запросов трех типов:

- `-1 L R` ($1 \leqslant L \leqslant R \leqslant N$) — разворот подстроки.
- `0 L R D` ($1 \leqslant L \leqslant R \leqslant N$, $0 < D \leqslant N$) — замена символов в подстроке.
- `1 L R` ($1 \leqslant L \leqslant R \leqslant N$) — количество различных перестановок подстроки $[L, R]$ строки S.

Формат выходных данных

На каждый запрос типа “`1 L R`” выведите ответ по модулю $10^9 + 7$ в отдельной строке.

Пример

stdin	stdout
6 abcabc 3 -1 1 6 0 1 3 1 1 1 6	180

Замечание

D-м следующим символом за x называется символ, получающийся путем D-кратного взятия следующего за x символа в циклическом алфавите.