Задача A. Простое двоичное дерево поиска

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Реализуйте просто двоичное дерево поиска.

Формат входных данных

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.

Формат выходных данных

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

Пример

стандартный ввод	стандартный вывод
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

Задача B. Сбалансированное двоичное дерево поиска

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Реализуйте сбалансированное двоичное дерево поиска.

Формат входных данных

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.

Формат выходных данных

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

Пример

стандартный ввод	стандартный вывод
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

Задача C. Декартово дерево

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	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.

Формат входных данных

В первой строке записано число N — количество пар. Далее следует N ($1 \leqslant N \leqslant 300\,000$) пар (a_i, b_i). Для всех пар $|a_i|, |b_i| \leqslant 30\,000$. $a_i \neq a_j$ и $b_i \neq b_j$ для всех $i \neq j$.

Формат выходных данных

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.

Пример

стандартный ввод	стандартный вывод
7 5 4 2 2 3 9 0 5 1 3 6 6 4 11	YES 2 3 6 0 5 1 1 0 7 5 0 0 2 4 0 1 0 0 3 0 0

Задача D. Добавление ключей

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	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:

`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 `Insert(`$L+1$`, `$A[L]$`)` и затем присвоить $A[L] \leftarrow K$.

По заданным N целым числам $L_1, L_2, \ldots, L_N$ выведите массив после выполнения последовательности операций:

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

Формат входных данных

Первая строка входного файла содержит числа N — количество операций `Insert`, которое следует выполнить и M — максимальную позицию, которая используется в операциях `Insert` ($1 \leqslant N \leqslant 131\,072$, $1 \leqslant M \leqslant 131\,072$).

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

Формат выходных данных

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 — $A[1], A[2], \ldots, A[W]$. Выводите нули для пустых ячеек.

Пример

стандартный ввод	стандартный вывод
5 4 3 3 4 1 3	6 4 0 5 2 3 1

Задача E. И снова сумма

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

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

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

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

стандартный ввод	стандартный вывод
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7

Задача F. K-й максимум

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

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

стандартный ввод	стандартный вывод
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

Задача G. Переместить в начало

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мегабайт

Вам дан массив $a_1 = 1, a_2 = 2, \ldots, a_n = n$ 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.

Формат входных данных

В первой строке входного файла указаны числа n и m ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 100\,000$) — число элементов в массиве и число операций. Следующие m строк содержат операции в виде двух целых чисел: l_i и r_i ($1 \leqslant l_i \leqslant r_i \leqslant n$).

Формат выходных данных

Выведите n целых чисел — порядок элементов в массиве после применения всех операций.

Пример

стандартный ввод	стандартный вывод
6 3 2 4 3 5 2 2	1 4 5 2 3 6

Задача H. Различные буквы

Имя входного файла:	`log.in`
Имя выходного файла:	`log.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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

- **insert** $\langle index \rangle$ $\langle number \rangle$ $\langle letter \rangle$ — добавить $\langle number \rangle$ букв $\langle letter \rangle$ перед буквой с индексом $\langle index \rangle$.
- **remove** $\langle index \rangle$ $\langle number \rangle$ — удалить $\langle number \rangle$ букв, начиная с индекса $\langle index \rangle$.
- **query** $\langle index_1 \rangle$ $\langle index_2 \rangle$ — вывести количество различных букв на отрезке с $\langle index_1 \rangle$ до $\langle index_2 \rangle$ включительно.

Буквы нумеруются с 1.

Формат входных данных

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

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

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

$\langle number \rangle$ добавления, удаления не превышает 10 000.

Формат выходных данных

Для каждого запроса **query** выведите одно целое число — количество различных букв на отрезке $\langle index_1 \rangle$, $\langle index_2 \rangle$ включительно.

Пример

log.in	log.out
8 + 1 4 w + 3 3 o ? 2 3 - 2 2 ? 2 3 + 2 2 t ? 1 6 - 1 6	2 1 3

Замечание

Пояснение к примеру:

1. `wwww`
2. `wwoooww`
3. `w[wo]ooww` : 2 различные буквы
4. `wooww`
5. `w[oo]ww` : 1 буква
6. `wttooww`
7. `[wttoow]w` : 3 различные буквы
8. `w`

Задача I. Эх, дороги

Имя входного файла:	`roads.in`
Имя выходного файла:	`roads.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	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.

Формат входных данных

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

Формат выходных данных

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.

Пример

roads.in	roads.out
5 4 6 1 2 2 3 1 3 4 5 ? 1 2 ? 1 5 - 2 3 ? 2 3 + 2 4 ? 1 5	0 -1 1 2