Задача A. Снеговики

Имя входного файла:	`snowmen.in`
Имя выходного файла:	`snowmen.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	64 мегабайта

Зима. 2012 год. На фоне грядущего Апокалипсиса и конца света незамеченной прошла новость об очередном прорыве в областях клонирования и снеговиков: клонирования снеговиков. Вы конечно знаете, но мы вам напомним, что снеговик состоит из нуля или более вертикально поставленных друг на друга шаров, а клонирование — это процесс создания идентичной копии (клона).

В местечке Местячково учитель Андрей Сергеевич Учитель купил через интернет-магазин «Интернет-магазин аппаратов клонирования» аппарат для клонирования снеговиков. Теперь дети могут играть и даже играют во дворе в следующую игру. Время от времени один из них выбирает понравившегося снеговика, клонирует его и:

- либо добавляет ему сверху один шар;
- либо удаляет из него верхний шар (если снеговик не пустой).

Учитель Андрей Сергеевич Учитель записал последовательность действий и теперь хочет узнать суммарную массу всех построенных снеговиков.

Формат входных данных

Первая строка содержит количество действий n ($1 \leqslant n \leqslant 200\,000$). В строке номер $i + 1$ содержится описание действия i:

- `t m` — клонировать снеговика номер t ($0 \leqslant t < i$) и добавить сверху шар массой m ($0 < m \leqslant 1000$);
- `t 0` — клонировать снеговика номер t ($0 \leqslant t < i$) и удалить верхний шар. Гарантируется, что снеговик t не пустой.

В результате действия i, описанного в строке $i + 1$ создается снеговик номер i. Изначально имеется пустой снеговик с номером ноль.

Все числа во входном файле целые.

Формат выходных данных

Выведите суммарную массу построенных снеговиков.

Примеры

snowmen.in	snowmen.out
8 0 1 1 5 2 4 3 2 4 3 5 0 6 6 1 0	74

Задача B. Откат

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

Сергей работает системным администратором в очень крупной компании. Естественно, в круг его обязанностей входит резервное копирование информации, хранящейся на различных серверах и «откат» к предыдущей версии в случае возникновения проблем.

В данный момент Сергей борется с проблемой недостатка места для хранения информации для восстановления. Он решил перенести часть информации на новые сервера. К сожалению, если что-то случится во время переноса, он не сможет произвести откат, поэтому процедура переноса должна быть тщательно спланирована.

На данный момент у Сергея хранятся n точек восстановления различных серверов, пронумерованных от 1 до n. Точка восстановления с номером i позволяет произвести откат для сервера a_i. Сергей решил разбить перенос на этапы, при этом на каждом этапе в случае возникновения проблем будут доступны точки восстановления с номерами $l, l+1, \ldots, r$ для некоторых l и r.

Для того, чтобы спланировать перенос данных оптимальным образом, Сергею необходимо научиться отвечать на запросы: для заданного l, при каком минимальном r в процессе переноса будут доступны точки восстановления не менее чем k различных серверов.

Помогите Сергею.

Формат входных данных

Первая строка входного файла содержит два целых числа n и m, разделенные пробелами — количество точек восстановления и количество серверов ($1 \leqslant n, m \leqslant 100\,000$). Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ — номера серверов, которым соответствуют точки восстановления ($1 \leqslant a_i \leqslant m$).

Третья строка входного файла содержит q — количество запросов, которые необходимо обработать ($1 \leqslant q \leqslant 100\,000$). В процессе обработки запросов необходимо поддерживать число p, исходно оно равно 0. Каждый запрос задается парой чисел x_i и y_i, используйте их для получения данных запроса следующим образом: $l_i = ((x_i + p) \bmod n) + 1$, $k_i = ((y_i + p) \bmod m) + 1$ ($1 \leqslant l_i, x_i \leqslant n$, $1 \leqslant k_i, y_i \leqslant m$). Пусть ответ на i-й запрос равен r. После выполнения этого запроса, следует присвоить p значение r.

Формат выходных данных

На каждый запрос выведите одно число — искомое минимальное r, либо 0, если такого r не существует.

Примеры

rollback.in	rollback.out
7 3 1 2 1 3 1 2 1 4 7 3 7 1 7 1 2 2	1 4 0 6

Задача C. K-я порядковая статистика на отрезке

Имя входного файла:	`kth.in`
Имя выходного файла:	`kth.out`
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мегабайт

Дан массив из N неотрицательных чисел, строго меньших 10^9. Вам необходимо ответить на несколько запросов о величине k-й порядковой статистики на отрезке $[l, r]$.

Формат входных данных

Первая строка содержит число N ($1 \leqslant N \leqslant 450\,000$) — размер массива.

Вторая строка может быть использована для генерации a_i — начальных значений элементов массива. Она содержит три числа a_1, l и m ($0 \leqslant a_1, l, m < 10^9$); для i от 2 до N

$$a_i = (a_{i-1} \cdot l + m) \bmod 10^9.$$

В частности, $0 \leqslant a_i < 10^9$.

Третья строка содержит одно целое число B ($1 \leqslant B \leqslant 1000$) — количество групп запросов.

Следующие B строк описывают одну группу запросов. Каждая группа запросов описывается 10 числами. Первое число G обозначает количество запросов в группе. Далее следуют числа x_1, l_x и m_x, затем y_1, l_y и m_y, затем, k_1, l_k и m_k ($1 \leqslant x_1 \leqslant y_1 \leqslant N$, $1 \leqslant k_1 \leqslant y_1 - x_1 + 1$, $0 \leqslant l_x, m_x, l_y, m_y, l_k, m_k < 10^9$). Эти числа используются для генерации вспомогательных последовательностей x_g и y_g, а также параметров запросов i_g, j_g и k_g ($1 \leqslant g \leqslant G$)

$$\begin{array}{rcll}
x_g & = & ((i_{g-1} - 1) \cdot l_x + m_x) \bmod N) + 1, & 2 \leqslant g \leqslant G \\
y_g & = & ((j_{g-1} - 1) \cdot l_y + m_y) \bmod N) + 1, & 2 \leqslant g \leqslant G \\
i_g & = & \min(x_g, y_g), & 1 \leqslant g \leqslant G \\
j_g & = & \max(x_g, y_g), & 1 \leqslant g \leqslant G \\
k_g & = & (((k_{g-1} - 1) \cdot l_k + m_k) \bmod (j_g - i_g + 1)) + 1, & 2 \leqslant g \leqslant G
\end{array}$$

Сгенерированные последовательности описывают запросы, g-й запрос состоит в поиске k_g-го по величине числа среди элементов отрезка $[i_g, j_g]$.

Суммарное количество запросов не превосходит 600 000.

Формат выходных данных

Выведите единственное число — сумму ответов на запросы.

Примеры

kth.in	kth.out
5 1 1 1 5 1 1 0 0 3 0 0 2 0 0 1 2 0 0 5 0 0 3 0 0 1 1 0 0 5 0 0 5 0 0 1 3 0 0 3 0 0 1 0 0 1 1 0 0 4 0 0 1 0 0	15

Задача D. CHM

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

Ваша задача — реализовать **Persistent Disjoint-Set-Union**. Что это значит?
Про **Disjoint-Set-Union**:
Изначально у вас есть n элементов. Нужно научиться отвечать на 2 типа запросов.

- `+ a b` — объединить множества, в которых лежат вершины a и b
- `? a b` — сказать, лежат ли вершины a и b сейчас в одном множестве

Про **Persistent**:
Теперь у нас будет несколько копий (версий) структуры данных **Disjoint-Set-Union**.
Запросы будут выглядеть так:

- `+ i a b` — запрос к i-й структуре, объединить множества, в которых лежат вершины a и b. При этом i-я структура остается не изменной, создается новая версия, ей присваивается новый номер (какой? читайте дальше)
- `? i a b` — запрос к i-й структуре, сказать, лежат ли вершины a и b сейчас в одном множестве

Формат входных данных

На первой строке 2 числа N ($1 \leqslant N \leqslant 10^5$) и K ($0 \leqslant K \leqslant 10^5$) — число элементов и число запросов. Изначально все элементы находятся в различных множествах. Эта изначальная копия (версия) структуры имеет номер 0.

Далее следуют K строк, на каждой описание очередного запроса. Формат запросов описан выше. Запросы нумеруются целыми числами от 1 до K.

Пусть j-й из K запросов имеет вид «`+ i a b`». Тогда новая версия получит номер j. Запросы вида «`? i a b`» не порождают новых структур.

Формат выходных данных

Для каждого запроса вида `? i a b` на отдельной строке нужно вывести `YES` или `NO`.

Примеры

snm.in	snm.out
4 7 + 0 1 2 ? 0 1 2 ? 1 1 2 + 1 2 3 ? 4 3 1 ? 0 4 4 ? 4 1 4	NO YES YES YES NO

Задача E. НВП на дереве

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

ано дерево на N вершинах, у которого i-е ребро соединяет вершину u_i и вершину v_i. На вершине с номером i записано целое число a_i. Для каждого целого числа k от 1 до N решите следующую задачу:

Составим последовательность, выписав целые числа на вершинах, вдоль кратчайшего пути от вершины 1 до вершины k, в том порядке, в котором они появляются. Найдите длину наибольшей возрастающей подпоследовательности этой последовательности.

Формат входных данных

В первой строке вводится число n ($2 \leqslant n \leqslant 2 \cdot 10^5$) — количество вершин в дереве.

Во второй строке через пробел задаются числа a_i ($1 \leqslant a_i \leqslant 10^9$) — числа, записанные на вершинах.

В каждой из следующих $n - 1$-й строке вводятся пары v_i, u_i — рёбра дерева ($1 \leqslant v_i \neq u_i \leqslant n$). Гарантируется, чтоданный набор рёбер образует дерево.

Формат выходных данных

Выведите N строк. В k-й строке выведите длину yfb,jkmitq возрастающей подпоследовательности последовательности, полученной по кратчайшему пути из вершины 1 в вершину k.

Примеры

стандартный ввод	стандартный вывод
10 1 2 5 3 4 6 7 3 2 4 1 2 2 3 3 4 4 5 3 6 6 7 1 8 8 9 9 10	1 2 3 3 4 4 5 2 2 3

Задача F. Persistent List

Имя входного файла:	`plist.in`
Имя выходного файла:	`plist.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Даны N списков. Каждый состоит из одного элемента.

Нужно научитсья совершать следующие операции:

- `merge` — взять два каких-то уже существующих списка и породить новый, равный их конкатенации.
- `head` — взять какой-то уже существующий список L и породить два новых, в одном первый элемент L, во втором весь L кроме первого элемента.
- `tail` — взять какой-то уже существующий список L и породить два новых, в одном весь L кроме последнего элемента, во втором последний элемент L.

Для свежесозданных списков нужно говорить сумму элементов в них по модулю $10^9 + 7$.

Формат входных данных

Число N ($1 \leqslant N \leqslant 10^5$). Далее N целых чисел от 1 до 10^9 — элементы списков. Исходные списки имеют номера — $1, 2, \cdots, N$.

Затем число M ($1 \leqslant M \leqslant 10^5$) — количество операций. Далее даны операции в следующем формате:

- `merge i j`
- `head i`
- `tail i`

Где `i` и `j` — номера уже существующих списков. Если в текущий момент имеется K списков, новый список получает номер $K + 1$.

Для операций `head` и `tail` считается, что сперва порождается левая часть, затем правая (см. пример). Также вам гарантируется, что никогда не будут порождаться пустые списки.

Формат выходных данных

Для каждого нового списка нужно вывести сумму элементов по модулю $10^9 + 7$.

Примеры

plist.in	plist.out
4 1 2 3 4 7 merge 1 2 merge 3 4 merge 6 5 head 7 tail 9 merge 2 3 merge 1 1	3 7 10 3 7 5 2 5 2

Задача G. Менеджер памяти

Имя входного файла:	`memory.in`
Имя выходного файла:	`memory.out`
Ограничение по времени:	8 секунды
Ограничение по памяти:	512 мегабайт

Одно из главных нововведений новейшей операционной системы Indows 7 — новый менеджер памяти. Он работает с массивом длины N и позволяет выполнять три самые современные операции:

- `copy(a, b, l)` — скопировать отрезок длины $[a, a + l - 1]$ в $[b, b + l - 1]$
- `sum(l, r)` — посчитать сумму элементов массива на отрезке $[l, r]$
- `print(l, r)` — напечатать элементы с l по r, включительно

Вы являетесь разработчиком своей операционной системы, и Вы, безусловно, не можете обойтись без инновационных технологий. Вам необходимо реализовать точно такой же менеджер памяти.

Формат входных данных

Первая строка входного файла содержит целое число N ($1 \leqslant N \leqslant 1\,000\,000$) — размер массива, с которым будет работать Ваш менеджер памяти.

Во второй строке содержатся четыре числа $1 \leqslant X_1, A, B, M \leqslant 10^9 + 10$. С помощью них можно сгенерировать исходный массив чисел $X_1, X_2, \cdots, X_N$. $X_{i+1} = (A * X_i + B) \mod M$

Следующая строка входного файла содержит целое число K ($1 \leqslant K \leqslant 200\,000$) — количество запросов, которые необходимо выполнить Вашему менеджеру памяти.

Далее в K строках содержится описание запросов. Запросы заданы в формате:

- `cpy` a b l — для операции `copy`
- `sum` l r — для операции `sum` ($l \leqslant r$)
- `out` l r — для операции `print` ($l \leqslant r$)

Гарантируется, что суммарная длина запросов `print` не превышает 3 000. Также гарантируется, что все запросы корректны.

Формат выходных данных

Для каждого запроса *sum* или *print* выведите в выходной файл на отдельной строке результат запроса.

Примеры

memory.in	memory.out
6 1 4 5 7 7 out 1 6 cpy 1 3 2 out 1 6 sum 1 4 cpy 1 2 4 out 1 6 sum 1 6	1 2 6 1 2 6 1 2 1 2 2 6 6 1 1 2 1 2 6 13