

Задача А. Персистентный стек

Имя входного файла: `stack.in`
Имя выходного файла: `stack.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Реализуйте персистентный стек.

Формат входных данных

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

- $t\ m$ — добавить в конец стека номер t ($0 \leq t < i$) число m ($0 < m \leq 1000$);
- $t\ 0$ — удалить последний элемент стека номер t ($0 \leq t < i$). Гарантируется, что стек t не пустой.

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

Все числа во входном файле целые.

Формат выходных данных

Для каждой операции удаления выведите удаленный элемент на отдельной строке.

Примеры

stack.in	stack.out
8	3
0 1	1
1 5	
2 4	
3 2	
4 3	
5 0	
6 6	
1 0	

Задача В. Persistent Array

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

Дан массив (вернее, первая, начальная его версия).

Нужно уметь отвечать на два запроса:

- $a_i[j] = x$ — создать из i -й версии новую, в которой j -й элемент равен x , а остальные элементы такие же, как в i -й версии.
- `get $a_i[j]$` — сказать, чему равен j -й элемент в i -й версии.

Формат входных данных

Количество чисел в массиве N ($1 \leq N \leq 10^5$) и N элементов массива. Далее количество запросов M ($1 \leq M \leq 10^5$) и M запросов. Формат описания запросов можно посмотреть в примере. Если уже существует K версий, новая версия получает номер $K + 1$. И исходные, и новые элементы массива — целые числа от 0 до 10^9 . Элементы в массиве нумеруются числами от 1 до N .

Формат выходных данных

На каждый запрос типа `get` вывести соответствующий элемент нужного массива.

Примеры

<code>parray.in</code>	<code>parray.out</code>
6	6
1 2 3 4 5 6	5
11	10
create 1 6 10	5
create 2 5 8	10
create 1 5 30	8
get 1 6	6
get 1 5	30
get 2 6	
get 2 5	
get 3 6	
get 3 5	
get 4 6	
get 4 5	

Задача С. Откат

Имя входного файла: `rollback.in`
Имя выходного файла: `rollback.out`
Ограничение по времени: 1.5 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 $l, l + 1, \dots, r$ 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 ($1 \leq n, m \leq 100\,000$). Вторая строка содержит n целых чисел a_1, a_2, \dots, a_n — номера серверов, которым соответствуют точки восстановления ($1 \leq a_i \leq m$).

Третья строка входного файла содержит q — количество запросов, которые необходимо обработать ($1 \leq q \leq 100\,000$). 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: $l_i = ((x_i + p) \bmod n) + 1$, $k_i = ((y_i + p) \bmod m) + 1$ ($1 \leq l_i, x_i \leq n$, $1 \leq k_i, y_i \leq m$). 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

<code>rollback.in</code>	<code>rollback.out</code>
7 3	1
1 2 1 3 1 2 1	4
4	0
7 3	6
7 1	
7 1	
2 2	

Задача D. K -я порядковая статистика на отрезке

Имя входного файла: `kth.in`
Имя выходного файла: `kth.out`
Ограничение по времени: 4 секунды
Ограничение по памяти: 256 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 ($1 \leq N \leq 450\,000$) — размер массива.

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

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

В частности, $0 \leq a_i < 10^9$.

Третья строка содержит одно целое число B ($1 \leq B \leq 1000$) — 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 ($1 \leq x_1 \leq y_1 \leq N$, $1 \leq k_1 \leq y_1 - x_1 + 1$, $0 \leq l_x, m_x, l_y, m_y, l_k, m_k < 10^9$). Эти числа используются для генерации вспомогательных последовательностей x_g и y_g , а также параметров запросов i_g, j_g и k_g ($1 \leq g \leq G$)

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

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

<code>kth.in</code>	<code>kth.out</code>
5	15
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	

Задача Е. Урны и шары

Имя входного файла: `balls.in`
Имя выходного файла: `balls.out`
Ограничение по времени: 4 секунды
Ограничение по памяти: 256 мегабайт

Пусть у вас есть n урн, в каждой из которых лежит по одному шару. Урна с номером i содержит шарик под номером i . У вас есть специальное устройство, которое позволяет перемещать шарики. Им чрезвычайно просто пользоваться: сначала вы выбираете некоторый отрезок последовательных урн. После этого вы выбираете некоторый другой отрезок последовательных урн такой же длины, как и исходный, и затем шарики из урн первого отрезка перемещаются в соответствующие урны второго отрезка.

Дана последовательность перемещений. Установите, в какой урне окажется каждый шарик.

Формат входных данных

Первая строка входных данных содержит два числа n и m — число урн и число перемещений, соответственно ($1 \leq n \leq 100\,000$, $1 \leq m \leq 50\,000$). Каждая из следующих m строк содержит три числа $count_i$, $from_i$ и to_i , которые означают одновременное перемещение всех шариков из урны $from_i$ в урну to_i , всех шариков из урны $from_i + 1$ в урну $to_i + 1$, ..., всех шариков из урны $from_i + count_i - 1$ в урну $to_i + count_i - 1$ ($1 \leq count_i, from_i, to_i \leq n$, $\max(from_i, to_i) + count_i \leq n + 1$).

Формат выходных данных

Выведите n чисел — итоговые позиции каждого шарика.

Примеры

<code>balls.in</code>	<code>balls.out</code>
2 3	1 1
1 1 2	
1 2 1	
1 2 1	

Задача F. Разрезание графа

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

Разбейте множество вершин заданного графа на два непустых подмножества A и B так, чтобы количество рёбер между вершинами различных подмножеств было минимально.

Формат входных данных

В первой строке входного файла записано целое число n ($2 \leq n \leq 50$) — число вершин в графе. Каждая из следующих n строк содержит по n символов. i -ый символ j -ой из этих строк равен “1”, если между вершинами i и j есть ребро, и “0” в противном случае. Заданная таким образом матрица смежности графа является антирефлексивной (на главной диагонали стоят нули) и симметричной (относительно главной диагонали).

Формат выходных данных

Выведите в выходной файл две строки. На первой выведите номера вершин, попавших во множество A , через пробел, а на второй — номера вершин, попавших во множество B , также через пробел. Номера вершин можно выводить в любом порядке.

Примеры

<code>mincut.in</code>	<code>mincut.out</code>
4	2
0111	1 3 4
1001	
1001	
1110	