Задача A. LCA - 2

Имя входного файла:	`lca2.in`
Имя выходного файла:	`lca2.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Задано подвешенное дерево, содержащее n вершин, пронумерованных от 0 до $n-1$. Требуется ответить на m запросов о наименьшем общем предке для пары вершин.

Запросы генерируются следующим образом. Заданы числа a_1, a_2 и числа x, y и z.

Числа $a_3, \ldots, a_{2m}$ генерируются следующим образом: $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. Первый запрос имеет вид $\langle a_1, a_2 \rangle$. Если ответ на $i-1$-й запрос равен v, то i-й запрос имеет вид $\langle (a_{2i-1}+v) \bmod n, a_{2i} \rangle$.

Формат входных данных

Первая строка содержит два числа: n ($1 \leqslant n \leqslant 100\,000$) и m ($1 \leqslant m \leqslant 10\,000\,000$). Корень дерева имеет номер 0. Вторая строка содержит $n-1$ целых чисел, i-е из этих чисел это предок вершины i

Третья строка содержит целые числа a_1 и a_2 ($0 \leqslant a_i \leqslant n-1$).

Четвёртая строка содержит три целых числа: x, y и z ($0 \leqslant x, y, z \leqslant 10^9$)

Формат выходных данных

Выведите в выходной файл сумму номеров вершин — ответов на все запросы.

Примеры

lca2.in	lca2.out
3 2 0 1 2 1 1 1 0	2
1 2 0 0 1 1 1	0

Задача B. Учиться!

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

Каждый год огромное количество выпускников, сдавшие ЕГЭ, выбирают, куда же они пойдут учиться. Не удивительно, что многие из них предпочитают перебраться поближе к столице. Транспортная инфраструктура страны переживает не лучшие времена, и в приемлемом качестве поддерживается минимально возможное число городов, необходимое для того, чтобы от любого города можно было добраться до любого другого.

Каждый выпускник оценивает свои результаты сдачи экзаменов, и решает, насколько далеко от своего родного города в сторону столицы он сможет уехать.

Выпускников настолько много, что вам не требуется выводить для каждого из них, до какого города он сможет доехать. Достаточно вывести сумму ответов для каждого выпускника.

Запросы генерируются следующим образом. Заданы числа a_1, a_2 и числа x, y и z. Числа $a_3, \ldots, a_{2m}$ генерируются следующим образом: $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. Первый запрос имеет вид $\langle a_1, a_2 \rangle$. Если ответ на $i-1$-й запрос равен v, то i-й запрос имеет вид $\langle (a_{2i-1} + v) \bmod n, a_{2i} \rangle$. В i-м запросе первое число соответствует городу, в котором окончил школу i-й выпускник, а второе — насколько далеко от родного города он может уехать. Все выпускники стараются перебраться как можно ближе к столице.

Формат входных данных

Первая строка содержит два числа: n ($1 \leqslant n \leqslant 100\,000$) и m ($1 \leqslant m \leqslant 10\,000\,000$). Столица имеет номер 0. Вторая строка содержит $n-1$ целых чисел, i-е из этих чисел равно номеру следующего за городом i на пути к столице. Третья строка содержит два целых числа в диапазоне от 0 до $n-1$: a_1 и a_2. Четвертая строка содержит три целых числа: x, y и z, эти числа неотрицательны и не превосходят 10^9.

Формат выходных данных

Выведите в выходной файл сумму номеров городов — ответов на все запросы.

Примеры

moscow.in	moscow.out
3 2 0 1 2 1 1 1 0	1
1 2 0 0 1 1 1	0

Задача C. RMQ

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

Есть массив из N целых чисел и M запросов вида: найдите минимум на отрезке с концами l_i, r_i.

Формат входных данных

Входной файл содержит T наборов тестовых данных. Каждый набор тестовых данных задаётся числами N, M, A, B ($1 \leqslant N \leqslant 25\,000, 1 \leqslant A, B \leqslant 10^9$), где N — размер массива, M — число запросов.

Массив и запросы нужно получить следующим образом: выпишем последовательность чисел $C_i = (A \cdot i + B) \bmod 2^{32}$.

Элементы последовательности с номерами от 1 до N — элементы массива. Элементы последовательности с номерами от $N + 1$ до $N + 2 \cdot M$ взятые по модулю N образуют M пар чисел, которые являются границами отрезков запросов. Ввод заканчивается числами 0 0 0 0. Массив индексируется с нуля.

Сумма N по всем наборам тестовых данных не превосходит 10^8. Сумма M по всем наборам тестовых данных не превосходит $2 \cdot 10^7$.

Формат выходных данных

Для каждого набора тестовых данных выведите сумму по всем запросам.

Примеры

стандартный ввод	стандартный вывод
10 10 955379886 619166003 0 0 0 0	7671393960

Замечание

Массив:

```
1574545889 2529925775 3485305661 145718251 1101098137 2056478023 3011857909
3967237795 627650385 1583030271
```

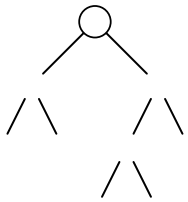
Запросы:

```
7 3
3 9
5 1
7 7
3 9
5 5
1 7
3 9
9 5
1 7
```

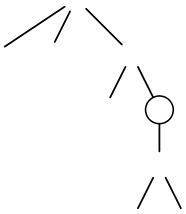
Задача D. Dynamic LCA

Имя входного файла:	`dynamic.in`
Имя выходного файла:	`dynamic.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Постановка задачи о *наименьшем общем предке* такова: дано дерево T с выделенным корнем и две вершины u и v, $\mathrm{lca}(u, v)$ — вершина с максимальной глубиной, которая является предком и u, и v. Например, на картинке внизу $\mathrm{lca}(8, 7)$ — вершина 3.



С помощью операции $\mathrm{chroot}(u)$ мы можем менять корень дерева, достаточно отметить u, как новый корень, и направить ребра вдоль пути от корня. Наименьшие общие предки вершин поменяются соответствующе. Например, если мы сделаем $\mathrm{chroot}(6)$ на картинке сверху, $\mathrm{lca}(8, 7)$ станет вершина 6. Получившееся дерево изображено внизу.



Вам дано дерево T. Изначально корень этого дерева — вершина 1. Напишите программу, которая поддерживает эти две операции: $\mathrm{lca}(u, v)$ и $\mathrm{chroot}(u)$.

Формат входных данных

Входной файл состоит из нескольких тестов.

Первая строка каждого теста содержит натуральное число n — количество вершин в дереве ($1 \leqslant n \leqslant 100\,000$). Следующие $n - 1$ строк содержат по 2 натуральных числа и описывают ребра дерева. Далее идет строка с единственным натуральным числом m — число операций. Следующие m строк содержат операции. Строка `? u v` означает операцию $\mathrm{lca}(u, v)$, а строка `! u` — $\mathrm{chroot}(u)$. Последняя строка содержит число 0.

Сумма n для всех тестов не превосходит 100 000. Сумма m для всех тестов не превосходит 200 000.

Формат выходных данных

Для каждой операции `? u v` выведите значение $\mathrm{lca}(u, v)$. Числа разделяйте переводами строк.

Примеры

dynamic.in	dynamic.out
9 1 2 1 3 2 4 2 5 3 6 3 7 6 8 6 9 10 ? 4 5 ? 5 6 ? 8 7 ! 6 ? 8 7 ? 4 5 ? 4 7 ? 5 9 ! 2 ? 4 3 0	2 1 3 6 2 3 6 2

Задача E. Опекуны карнотавров

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

Карнотавры очень внимательно относятся к заботе о своем потомстве. У каждого динозавра обязательно есть старший динозавр, который его опекает. В случае, если опекуна съедают (к сожалению, в юрский период такое не было редкостью), забота о его подопечных ложится на плечи того, кто опекал съеденного динамозавра. Карнотавры — смертоносные хищники, поэтому их обычаи строго запрещают им драться между собой. Если у них возникает какой-то конфликт, то, чтобы решить его, они обращаются к кому-то из старших, которому доверяют, а доверяют они только тем, кто является их опекуном или опекуном их опекуна и так далее (назовем таких динозавров суперопекунами). Поэтому для того, чтобы решить спор двух карнотавров, нужно найти такого динозавра, который является суперопекуном для них обоих. Разумеется, беспокоить старших по пустякам не стоит, поэтому спорщики стараются найти самого младшего из динозавров, который удовлетворяет этому условию. Если у динозавра возник конфликт с его суперопекуном, то этот суперопекун сам решит проблему. Если у динозавра нелады с самим собой, он должен разобраться с этим самостоятельно, не беспокоя старших. Помогите динозаврам разрешить их споры.

Формат входных данных

В первой строке содержит целое число M ($1 \leqslant M \leqslant 200\,000$) — количество запросов.
Далее следуют M запросов, описывающие события:

- `+ v` — родился новый динозавр и опекунство над ним взял динозавр с номером v. Родившемуся динозавру нужно присвоить наименьший натуральный номер, который до этого еще никогда не встречался.
- `- v` — динозавра номер v съели.
- `? u v` — у динозавров с номерами u и v возник конфликт и вам надо найти им третейского судью.

Изначально есть один прадинозавр номер 1. Гарантируется, что он никогда не будет съеден.

Формат выходных данных

Для каждого запроса типа «?» в выходной файл нужно вывести на отдельной строке одно число — номер самого молодого динозавра, который может выступить в роли третейского судьи.

Примеры

carno.in	carno.out
11 + 1 + 1 + 2 ? 2 3 ? 1 3 ? 2 4 + 4 + 4 - 4 ? 5 6 ? 5 5	1 1 2 2 5

Задача F. Поездка на каникулах

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

Железная дорога Флатландии представляет собой прямую, вдоль которой расположены n станций. Будем называть участок железной дороги от некоторой станции до следующей перегоном.

Поезд следует от станции 1 до станции n, делая остановку на каждой станции. В поезде k мест, пронумерованных от 1 до k. На поезд продаются билеты, каждый билет характеризуется тремя числами: s, t и a. Такой билет позволяет проехать от станции s до станции t на месте a.

Вася планирует в один из дней летних каникул проехать на поезде от одной станции до другой. Он выяснил, что на поезд в этот день уже продано m билетов, и возможно уже нет мест, свободных на всех перегонах между интересующими его станциями. Билет от одной станции до другой на определенное место можно купить, только если это место свободно на всех перегонах между этими станциями.

Вася сообразил, что иногда все равно можно проехать от одной станции до другой, купив несколько билетов и пересаживаясь с одного места на другое на некоторых промежуточных станциях. Разумеется, пересаживаться с места на место неудобно, поэтому Вася хочет купить минимальное количество билетов, чтобы на каждом перегоне у него было свое место.

Вася еще не решил, от какой станции и до какой он поедет. Он записал q вариантов поездки, и для каждого из них хочет узнать, какое минимальное число билетов ему придется купить, если он выберет этот вариант.

Требуется написать программу, которая по заданному описанию уже проданных билетов и вариантов поездки Васи определяет для каждого варианта, какое минимальное количество билетов необходимо купить, чтобы совершить такую поездку.

Формат входных данных

Первая строка входного файла содержит числа n, m и k ($2 \leqslant n \leqslant 200\,000$, $0 \leqslant m \leqslant 200\,000$, $1 \leqslant k \leqslant 200\,000$) — количество станций, количество уже проданных билетов и количество мест в поезде. Последующие m строк содержат информацию о проданных билетах. Каждая строка содержит три числа: s_i, t_i и a_i — номер станции, от которой куплен билет, номер станции, до которой куплен билет, и номер места, на которое куплен билет($1 \leqslant s_i < t_i \leqslant n$, $1 \leqslant a_i \leqslant k$). Гарантируется, что все билеты куплены таким образом, что ни на каком перегоне ни на какое место нет более одного билета.

Далее идет строка, которая содержит число q ($1 \leqslant q \leqslant 200\,000$). Последующие q строк содержат описания вариантов поездки. Каждая строка содержат два числа: f_j, d_j — номер станции, от которой Вася хочет поехать в этом варианте, и номер станции, до которой он хочет поехать ($1 \leqslant f_j < d_j \leqslant n$).

Формат выходных данных

Выходной файл должен содержать q чисел: для каждого варианта поездки требуется вывести минимальное количество билетов, которое необходимо купить Васе, чтобы совершить соответствующую поездку. Если поездку совершить невозможно, то для этого варианта требуется вывести `-1`.

Примеры

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

Задача G. LCA-3

Имя входного файла:	`lca3.in`
Имя выходного файла:	`lca3.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

> *Подвешенное дерево* —это ориентированный граф без циклов, в котором в каждую вершину, кроме одной, называемой *корнем* ориентированного дерева, входит одно ребро. В корень ориентированного дерева не входит ни одного ребра. *Отцом* вершины называется вершина, ребро из которой входит в данную.

(по материалам Wikipedia)

Дан набор подвешенных деревьев. Требуется выполнять следующие операции:

1. 0 u v Для двух заданных вершин u и v выяснить, лежат ли они в одном дереве. Если это так, вывести вершину, являющуюся их наименьшим общим предком, иначе вывести 0.
2. 1 u v Для корня u одного из деревьев и произвольной вершины v другого дерева добавить ребро (v, u). В результате эти два дерева соединятся в одно.

Вам необходимо выполнять все операции online, т.е. вы сможете узнать следующий запрос только выполнив предыдущий.

Формат входных данных

На первой строке входного файла находится число n — суммарное количество вершин в рассматриваемых деревьях, $1 \leqslant n \leqslant 50000$. На следующей строке расположено n чисел — предок каждой вершины в начальной конфигурации, или 0, если соответстующая вершина является корнем. Затем следует число k — количество запросов к вашей программе, $1 \leqslant k \leqslant 100000$. Каждая из следующих строк содержит по три целых числа: вид запроса (0 — для поиска LCA или 1 — для добавления ребра) и два числа x, y. Вершины, участвующие в запросе можно вычислить по формуле: $u = (x - 1 + ans) \bmod n + 1$, $v = (y - 1 + ans) \bmod n + 1$, где ans - ответ на последний запрос типа 0 ($ans = 0$ для первого запроса).

Формат выходных данных

Для каждого запроса типа 0, выведите в выходной файл одно число на отдельной строке — ответ за этот запрос.

Примеры

lca3.in	lca3.out
5 0 0 0 0 0 12 1 5 3 0 2 5 1 4 2 1 1 5 0 1 5 1 3 4 0 1 5 0 3 1 0 4 2 0 1 4 0 5 2 0 4 1	0 5 5 3 2 3 3 2

Задача H. Кварум

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	7.5 секунды
Ограничение по памяти:	512 мегабайт

Второе августа 1702 года, среда. На болоте, ныне называемом Санкт-Петербург, собрались лягушки со всех ближайших водоемов. Для каждой из жабок Великим Уквакнителем была приготовлена определенная кувшинка. Оказалось, что лягушки могут общаться только с теми лягушками, кто сидит на лягушке похожего цвета. Давайте соединим такие кувшинки ребрами. Оказалось, что полученный граф является подвешенным деревом на n вершинах (в корне дерева сидит сам Великий Уквакнитель). Все кувшинки пронумерованы от 0 до $n-1$.

Так как лягушки собрались, чтобы обсудить разрешение людям на строительство на их любимом болоте города, то некоторые пары лягушек хотят обмениваться своими аргументами друг с другом. Для ускорения процесса они просят вас найти их наименьшего общего предка.

Номера лягушек, которые в момент времени k ($1 \leqslant k \leqslant m$) хотят получить от вас их наименьшего предка, это a_{2k-1} и a_{2k} генерируются следующим образом:

- Изначально даны a_1, a_2 и числа x, y, z.
- Числа $a_3, \ldots, a_{2m}$ генерируются следующим образом: $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. Первый запрос имеет вид $\langle a_1, a_2 \rangle$. i-й запрос имеет вид $\langle a_{2i-1}, a_{2i} \rangle$.

Формат входных данных

Первая строка содержит два числа: n $\left(1 \leqslant n \leqslant 10^6\right)$ и m $\left(1 \leqslant m \leqslant 10^7\right)$.
Великий Уквакнитель сидит на кувшинке с номером 0.
Вторая строка содержит $n-1$ целых чисел, i-е из этих чисел это предок вершины i
Третья строка содержит целые числа a_1 и a_2 ($0 \leqslant a_i \leqslant n-1$).
Четвёртая строка содержит три целых числа: x, y и z ($0 \leqslant x, y, z \leqslant 10^9$)

Формат выходных данных

Выведите в выходной файл сумму номеров вершин — ответов на все запросы.

Примеры

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

Замечание

Сюда надо сдать Тарьяна. Остальные решения получат реджект

Задача I. Метрополитен в Омске (сложная версия)

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

Это сложная версия задачи. Единственное отличие между простой и сложной версией состоит в том, что в этой версии u может принимать любые возможные значения.

Как известно, Омск — столица Берляндии. Как и в любой столице, в Омске есть хорошо развитая система метрополитена. Метрополитен Омска представляет из себя некоторое количество станций, соединённых туннелями, причём между любыми двумя станциями есть ровно один путь, проходящий по каждому из туннелей не более одного раза. Иными словами, метрополитен представляет собой дерево.

Для развития метрополитена и привлечения жителей в Омске используется следующая система. Каждая станция имеет свой вес $x \in \{-1, 1\}$. Если станция имеет вес -1, то при посещении станции с жителя Омска взимается плата в 1 бурль. Если же вес станции равен 1, то житель Омска вознаграждается 1-м бурлем.

Пока что есть только одна станция с номером 1 и весом $x = 1$. Каждый день происходит одно из следующих событий:

- *К станции с номером v_i присоединяется новая станция с весом x, и ей присваивается номер, на единицу больший количества уже существующих станций.*
- *Лёша, живущий в Омске, задаётся вопросом: существует ли подотрезок (возможно, пустой) пути между вершинами u и v такой, что, проехав по нему, можно заработать ровно k бурлей (если $k < 0$, то это означает, что на проезд придётся потратить k бурлей). Иначе говоря, Лёшу интересует, существует ли такой подотрезок пути, что сумма весов вершин в нем равна k. Обратите внимание, что подотрезок может быть пустым, и тогда сумма равна 0.*

Вы являетесь другом Лёши, поэтому Ваша задача — ответить на вопросы Лёши.

Формат входных данных

В первой строке находится число t ($1 \leqslant t \leqslant 10^4$) — количество наборов входных данных.

В первой строке каждого набора входных данных дано число n ($1 \leqslant n \leqslant 2 \cdot 10^5$) — количество событий.

Далее следует n строк, описывающих события. В i-й строке возможен один из вариантов:

- Сначала идёт символ «+» (без кавычек), затем два числа v_i и x_i ($x_i \in \{-1, 1\}$, также гарантируется, что вершина с номером v_i существует). В этом случае к станции с номером v_i присоединяется новая станция с весом x_i.
- Сначала идёт символ «?» (без кавычек), а затем три числа u_i, v_i и k_i ($-n \leqslant k_i \leqslant n$). Гарантируется, что вершины с номерами u_i и v_i существуют. В этом случае необходимо определить, существует ли подотрезок (возможно, пустой) пути между станциями u_i и v_i с суммой весов ровно k_i.

Гарантируется, что сумма n по всем наборам входных данных не превосходит $2 \cdot 10^5$.

Формат выходных данных

Для каждого вопроса Лёши выведите «`Yes`» (без кавычек), если описанный в условии подотрезок существует, иначе выведите «`No`» (без кавычек).

Вы можете выводить ответ в любом регистре (например, строки «`yEs`», «`yes`», «`Yes`» и «`YES`» будут распознаны как положительный ответ).

Примеры

стандартный ввод	стандартный вывод
1 8 + 1 -1 ? 1 1 2 ? 1 2 1 + 1 1 ? 1 3 -1 ? 1 1 1 ? 1 3 2 ? 1 1 0	NO YES NO YES YES YES
1 7 + 1 -1 + 2 -1 + 2 1 + 3 -1 ? 5 2 2 ? 3 1 -1 ? 5 4 -3	NO YES YES

Замечание

Пояснение к первому примеру.

Ответ на второй вопрос «Yes», так как существует путь 1.

В четвертом вопросе мы можем снова выбрать путь 1.

В пятом запросе ответ «Yes», так как есть путь $1 - 3$.

В шестом запросе мы можем выбрать пустой путь, так как сумма весов на нем равна 0.

Нетрудно показать, что нет путей, удовлетворяющих первому и третьему запросу.