Задача A. Самое дешёвое ребро

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

Дано подвешенное дерево с корнем в первой вершине. Все ребра имеют веса (стоимости). Вам нужно ответить на m запросов вида «найти ребро минимального веса на пути между двумя заданными вершинами».

Формат входных данных

В первой строке файла записано одно число n — количество вершин в дереве ($1 \leqslant n \leqslant 50000$).

В i-й из следующих $n-1$ строк записано два целых числа p_{i+1} и w_{i+1} ($p_{i+1} < i+1$, $|w_{i+1}| \leqslant 10^6$) — предок вершины $i+1$ и вес ребра из вершины $i+1$ в её предка.

Следующая строка содержит число m ($0 \leqslant m \leqslant 5 \cdot 10^4$) — количество запросов. Следующие m строк содержат по одному запросу вида (x_i, y_i) — найти минимальный вес ребра на пути из вершины x_i в вершину y_i ($1 \leqslant x_i \leqslant n$, $1 \leqslant y_i \leqslant n$, $x_i \neq y_i$).

Формат выходных данных

Программа должна вывести m чисел — ответы на запросы.

Примеры

minonpath.in	minonpath.out
5 1 2 1 3 2 5 3 2 2 2 3 4 5	2 2

Задача B. 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 C. 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.

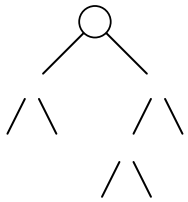
Примеры

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

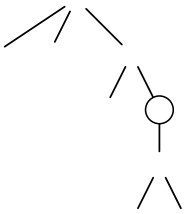
Задача D. Dynamic LCA

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, $\text{lca}(u, v)$ — вершина с максимальной глубиной, которая является предком и u, и v. Например, на картинке внизу $\text{lca}(8, 7)$ — вершина 3.



С помощью операции $\text{chroot}(u)$ 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 $\text{chroot}(6)$ на картинке сверху, $\text{lca}(8, 7)$ станет вершина 6. Получившееся дерево изображено внизу.



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

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 $\text{lca}(u, v)$, а строка `! u` — $\text{chroot}(u)$. 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 $\text{lca}(u, v)$. 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. 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 F. 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 G. 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 H. 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 I. Мультивселенные Павла

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2.5 секунд
Ограничение по памяти:	256 мегабайт

В последнее время Павел, как и обычно, занимался реализацией своих бизнес-планов. Последний его план содержал в себе инновационную идею. Павел решил обратиться к параллельным вселенным и построить бизнес-империю сразу во всех из них. В процессах эксперимента Павел почти добился успеха, но в критический момент допустил ошибку. Пока Павел ехал домой из своей секретной лаборатории на одном из рынков Москвы, мультивселенные смешались! Теперь в Москве оказалось сразу множество Павлов, и все они пытаются попасть домой.

Москву можно представить как n рынков и $n-1$ двунаправленных дорог между ними, i-я из которых соединяет между собой вершины a_i и b_i. От каждого рынка можно добраться до любого другого рынка, двигаясь по дорогам. Иными словами, Москва представляет собой дерево. Всего, в результате коллапса, в Москве в нашей вселенной оказалось c версий Павла. Павлы из разных мультивселенных живут и ведут разработки на разных рынках города. Если точнее, i-я версия Павла работает на рынке s_i и хочет вернуться домой на рынок f_i, перемещаясь по дорогам от рынка до рынка.

Предвидя негативные исходы, Павел заранее разобрался, что в случае смешения мультвселенных, все версии Павлов будут являться только спектральными следами и вернутся в свою вселенную в том случае, если доберутся до своего дома на f_i-м рынке. Во избежание пространственно-временного катаклизма ни в коем случае нельзя допустить, чтобы две версии Павла в некоторый момент времени встретились на каком-то рынке.

Все версии Павла в замешательстве и не начнут двигаться, если им не помочь. У Павла есть один агент, и он может послать его, чтобы тот сопроводил какую-то версию от ее лаборатории к её домашнему рынку. **Обратите внимание**, что агент может сопровождать за раз **только одну** версию и он **не может оставить ее** и перейти к другой версии, пока сопровождаемая версия Павла не окажется на рынке f_i. При этом, агент не сможет сопроводить версию, если на её пути окажутся еще не исчезнувшие версии Павла.

Теперь Павел хочет узнать, в каком порядке он должен отправить агентов для сопровождения версий или выяснить, что разрешить проблему мирового масштаба невозможно. Помогите ему в этом.

Формат входных данных

В первой строке входного файла содержится два целых числа n, c ($1 \leqslant c \leqslant n \leqslant 2 \times 10^5$) — количество рынков в Москве и количество версий Павла.

В следующих $n-1$ строке содержится описания дорог между рынками. В i-й из них находятся два целых числа a_i, b_i ($1 \leqslant a_i, b_i \leqslant n$) — концы i-й двунаправленной дороги.

После этого, в c следующих строках содержатся описания версий Павла. В i-й из строк находятся два целых числа s_i, f_i ($1 \leqslant s_i, f_i \leqslant n$) — расположение лаборатории и дома i-й версии Павла соответсвенно. Гарантируется, что никакие две различные версии Павла не имеют лабораторий на одном рынке. Иными словами, $s_i \neq s_j$ для $i \neq j$.

Формат выходных данных

Если существует порядок отправления агентов к версиям, при котором все версии отправляются домой и не происходит пространственно-временных коллапсов, то в первой строке выходного файла выведите `Yes`. Далее, во второй строке выведите перестановку из c натуральных чисел $p_1, p_2, \ldots p_c$ — порядок удаления версий Павла.

Иначе, выведите в единственной строке выходного файла `No`.

Группа	Баллы	Доп. ограничения	Необх. группы	Комментарий
0	0	–		Тесты из условия.
1	11	$c \leqslant 8, n \leqslant 100$		
2	18	$n \leqslant 100$	1	
3	24	$n \leqslant 4000$, длины путей $\leqslant 50$	1 – 2	
4	23	$n \leqslant 4000$		
5	24	–	1 – 4	**Полные ограничения.**

Примеры

стандартный ввод	стандартный вывод
5 4 1 2 2 3 2 4 4 5 5 3 4 3 3 2 1 3	Yes 3 2 1 4
4 2 1 2 2 3 3 4 2 4 3 1	No

Задача J. Дерево

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

Задано подвешенное дерево, содержащее n ($1 \leqslant n \leqslant 1\ 000\ 000$) вершин. Каждая вершина покрашена в один из n цветов. Требуется для каждой вершины v вычислить количество различных цветов, встречающихся в поддереве с корнем v.

Формат входных данных

В первой строке входного файла задано число n.

Следующие n строк описывают вершины, по одной в строке. Описание очередной вершины i имеет вид p_i c_i, где p_i — номер родителя вершины i, а c_i — цвет вершины i ($1 \leqslant c_i \leqslant n$). Для корня дерева $p_i = 0$.

Формат выходных данных

Выведите n чисел, обозначающих количества различных цветов в поддеревьях с корнями в вершинах $1, \ldots, n$.

Примеры

tree.in	tree.out
5 2 1 3 2 0 3 3 3 2 1	1 2 3 1 1