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:	2.5 секунд
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. 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 у двух вершин минимум среди стоимостей ребер пути между ними».

Формат входных данных

В первой строке файла записано одно числ — n (количество вершин).

В i-й из следующих $n-1$ строк записано два целых числа x и y ($x \leqslant i$, $|y| \leqslant 10^6$) — предок вершины i и стоимость ребра.

Далее следуют m ($0 \leqslant m \leqslant 5 \cdot 10^4$) запросов вида (x, y) — найти минимум на пути из x в y ($x \neq y$).

Формат выходных данных

Выведите m ответов на запросы.

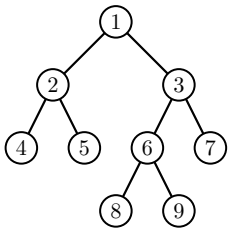
Примеры

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

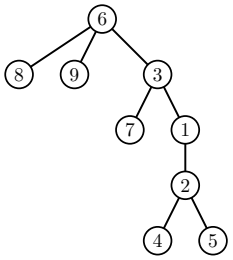
Задача D. Dynamic LCA

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. Перемножение матриц

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

Дана бинарная матрица. Вам необходимо возвести ее в квадрат по модулю 2.

Пусть имеется две матрицы, A и B. Тогда их произведение $C = AB$ определяется следующим образом: каждый элемент вычисляется по формуле

$$c_{ij} = \bigoplus_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Таким образом, c_{ij} — четность количества индексов k таких, что $a_{ik} = b_{kj} = 1$.

Формат входных данных

В первой строке входных данных содержится число n — размер матрицы ($1 \leqslant n \leqslant 4\,000$). Далее следуют n строк, каждая из которых состоит из n символов 0 или 1, описывающих соответствующие элементы матрицы.

Формат выходных данных

Выведите единственное число — количество единиц в результирующей матрице.

Примеры

matrixmul.in	matrixmul.out
3 011 100 101	5