Задача A. Предок

Имя входного файла:	`ancestor.in`
Имя выходного файла:	`ancestor.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входных данных

Первая строка входного файла содержит целое число n ($1 \leqslant n \leqslant 100\,000$) — количество вершин в дереве. Во второй строке находятся n чисел, i-е из которых определяет номер непосредственного родителя вершины с номером i. Если это число равно нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leqslant m \leqslant 100\,000$) — количество запросов.

Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

Формат выходных данных

Для каждого из m запросов выведите на отдельной строке число `1`, если вершина a является одним из предков вершины b, и `0` в противном случае.

Примеры

ancestor.in	ancestor.out
6 0 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

Задача B. 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 C. 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 D. Гонки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

Наряду с IOI в Паттайе проходят международные олимпийские гонки (IOR) 2011. Принимающей стороне требуется найти наиболее подходящую трассу для гонок.

В регионе Паттайя-Чонбури находятся N городов, соединённых сетью из $(N-1)$ магистралей. Каждая магистраль — двусторонняя, соединяет два различных города, и для нее известна длина в километрах — целое число. Известно, что между каждой парой городов существует ровно один возможный путь, соединяющий эти города. Таким образом, для любой пары городов существует ровно одна последовательность различных магистралей, по которой можно проехать из одного города в другой, не посещая никакой город дважды.

По требованиям организаторов IOR трасса должна являться путём суммарной длины *ровно* K километров, начинающимся и заканчивающимся в различных городах. Естественно, никакая магистраль и, поэтому, никакой город не могут быть использованы дважды при выборе трассы, иначе возможны столкновения. Чтобы минимизировать влияние гонок на трафик движения в регионе, необходимо выбрать для трассы путь из *наименьшего возможного количества магистралей.*

Формат входных данных

В первой строке через пробел записаны пары чисел N и K — количество городов и требуемая длина трассы в километрах ($1 \leqslant N \leqslant 200\,000$, $1 \leqslant K \leqslant 1\,000\,000$). В следующих $N-1$ строках через пробел три целых числа u_i, v_i и c_i — номера городов, соединенных магистралью, и длина этой магистрали, соответственно ($0 \leqslant u_i, v_i \leqslant N-1$, $0 \leqslant c_i \leqslant 1\,000\,000$).

Формат выходных данных

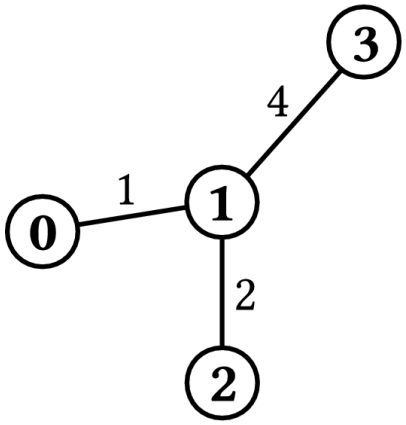
Выведите единственное число — минимальное возможное количество магистралей на допустимой трассе, имеющей длину, равную K. Если такой трассы не существует, выведите -1.

Примеры

стандартный ввод	стандартный вывод
4 3 0 1 1 1 2 2 1 3 4	2
3 3 0 1 1 1 2 1	-1
11 12 0 1 3 0 2 4 2 3 5 3 4 4 4 5 6 0 6 3 6 7 2 6 8 5 8 9 6 8 10 7	2

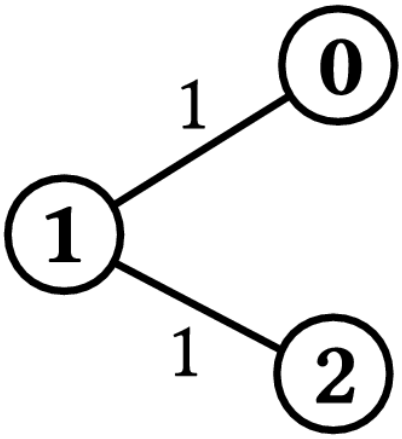
Замечание

Пример 1



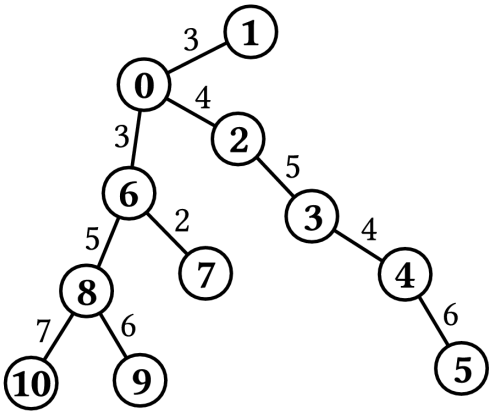
Единственная допустимая трасса начинается в городе 0, проходит через город 1 и заканчивается в городе 2.

Пример 2



В этом примере допустимой трассы не существует.

Пример 3

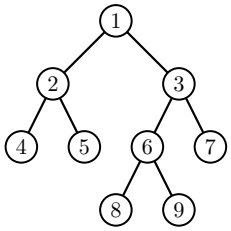


Одна из допустимых трасс состоит из 3 магистралей: она идёт из города с номером 6 через города с номерами 0 и 2 в город с номером 3. Другая трасса начинается в городе с номером 10 и идёт через город с номером 8 в город с номером 6. Вторая из них оптимальна, так как не существует подходящей трассы из одной магистрали.

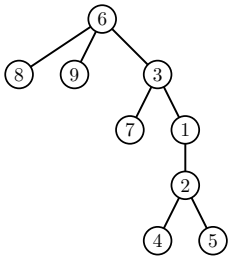
Задача E. Dynamic LCA

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 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. Цветные волшебники

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

Сказочная страна представляет собой множество городов, соединенных дорогами с двухсторонним движением. Причем из любого города страны можно добраться в любой другой город либо непосредственно, либо через другие города. Известно, что в сказочной стране не существует дорог, соединяющих город сам с собой и между любыми двумя разными городами, существует не более одной дороги.

В сказочной стране живут желтый и синий волшебники. Желтый волшебник, пройдя по дороге, перекрашивает ее в желтый цвет, синий — в синий. Как известно, при наложении желтой краски на синюю, либо синей краски на желтую, краски смешиваются и превращаются в краску зеленого цвета, который является самым нелюбимым цветом обоих волшебников.

В этом году в столице страны (городе f) проводится конференция волшебников. Поэтому желтый и синий волшебники хотят узнать, какое минимальное количество дорог им придется перекрасить в зеленый цвет, чтобы добраться в столицу. Изначально все дороги не покрашены.

Начальное положение желтого и синего волшебников заранее не известно. Поэтому необходимо решить данную задачу для k возможных случаев их начальных расположений.

Формат входных данных

Первая строка входного файла содержит целые числа: n ($1 \leqslant n \leqslant 100\,000$) и m ($1 \leqslant m \leqslant 500\,000$) — количество городов и дорог в волшебной стране соответственно.

Вторая строка содержит одно целое число f ($1 \leqslant f \leqslant n$) — номер города, являющегося столицей сказочной страны. В следующих m строках, находится описание дорог страны. В этих m строк записано по два целых числа a_i и b_i,

означающих, что существует дорога, соединяющая города a_i и b_i. Следующая строка содержит целое число k ($1 \leqslant k \leqslant 100\,000$) — количество возможных начальных расположений волшебников. Далее следуют k строк, каждая из которых содержит два целых числа — номера городов, в которых изначально находится желтый и синий волшебники соответственно.

Формат выходных данных

Для каждого из k случаев, выведите минимальное количество дорог, которое придется покрасить в зеленый цвет волшебникам для того, чтобы добраться в столицу.

Примеры

`magic.in`	`magic.out`
6 6 1 1 2 2 3 3 4 4 2 4 5 3 6 2 5 6 6 6	1 2

Задача H. Разреженные таблицы

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

Дан массив из n чисел. Требуется написать программу, которая будет отвечать на запросы следующего вида: найти минимум на отрезке между u и v включительно.

Формат входных данных

В первой строке входного файла даны три натуральных числа n, m ($1 \leqslant n \leqslant 10^5$, $1 \leqslant m \leqslant 10^7$) и a_1 ($0 \leqslant a_1 < 16\,714\,589$) — количество элементов в массиве, количество запросов и первый элемент массива соответственно. Вторая строка содержит два натуральных числа u_1 и v_1 ($1 \leqslant u_1, v_1 \leqslant n$) — первый запрос.

Элементы $a_2, a_3, \ldots, a_n$ задаются следующей формулой:

$$a_{i+1} = (23 \cdot a_i + 21563) \bmod 16714589.$$

Например, при $n = 10$, $a_1 = 12345$ получается следующий массив: $a = (12345, 305498, 7048017, 11694653, 1565158, 2591019, 9471233, 570265, 13137658, 1325095)$.

Запросы генерируются следующим образом:

$$u_{i+1} = \big((17 \cdot u_i + 751 + ans_i + 2i) \bmod n\big) + 1,$$
$$v_{i+1} = \big((13 \cdot v_i + 593 + ans_i + 5i) \bmod n\big) + 1,$$

где ans_i — ответ на запрос номер i.

Обратите внимание, что u_i может быть больше, чем v_i.

Формат выходных данных

В выходной файл выведите u_m, v_m и ans_m (последний запрос и ответ на него).

Примеры

sparse.in	sparse.out
10 8 12345 3 9	5 3 1565158

Замечание

Пояснение к тесту из примера: запросы и результаты.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
12345	305498	7048017	11694653	1565158	2591019	9471233	570265	13137658	1325095

#	u	v	ans
1	3	9	570265
2	10	1	12345
3	1	2	12345
4	10	10	1325095
5	5	9	570265
6	2	1	12345
7	3	2	305498
8	5	3	1565158