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 натуральное 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 дешевое 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 следующих $n-1$ строках записаны два числа — x и y. Число x на строке i означает, что x — предок вершины i, y означает стоимость ребра.

$x < i,\ |y| \leqslant 10^6$.

Далее m запросов вида (x, y) — найти минимум на пути из x в y ($x \neq y$).

Ограничения: $2 \leqslant n \leqslant 5 \cdot 10^4$, $0 \leqslant m \leqslant 5 \cdot 10^4$.

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:	5 секунды
Ограничение по памяти:	256 мегабайт

Задано подвешенное дерево, содержащее n ($1 \leqslant n \leqslant 100\,000$) вершин, пронумерованных от 0 до $n-1$. Требуется ответить на m ($1 \leqslant m \leqslant 10\,000\,000$) 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 и m. Корень дерева имеет номер 0. Вторая строка содержит $n-1$ целых чисел, i-е из этих чисел равно номеру родителя вершины i. 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 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. Зигмунд Фрейд и Карл Юнг

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

Известный психиатр Зигмунд Фрейд в своей книге "Толкование сновидений"подробно описал, что ему снится, когда в его генеологическое дерево добавляется новый лист. В своём более позднем труде "Я и оно"он также описал ощущения человека, видевшего сон про удаление вершины из генеологичекого дерева. Несколькими годами позже молодой Карл Юнг — будущий не менее известный психиатр, изучая работы своего знаменитого предшественника, не мог пройти мимо тех работ и стал готовить грандиозный эксперимент, основанный на строго задокументированных показаниях о более, чем ста тысячах опрошенных. Для завершения эксперимента не хватает совсем немногого — быстро находить наименьшего общего предка двух вершин.

Несмотря на то, что Юнг при жизни так и не закончил эксперимент, мы уверены, что он будет Вам безмерно благодарен, если Вы довершите его гениальную задумку.

Формат входных данных

Во входном файле записано число q, обозначающее количество запросов ($1 \leqslant q \leqslant 200\,000$). Далее на отдельных строках следуют q запросов, обозначающих следующие события:

- `+ v` — добавился новый лист, его предком стала вершина с номером v. Добавившейся вершине нужно присвоить наименьший натуральный номер, который до этого еще никогда не встречался.
- `- v` — вершина с номером v удалилась из дерева, предком её детей становится её предок.
- `? u v` — Карл Юнг интересуется наименьшим общим предком вершин u и v.

Изначально есть одна вершина с номером 1, гарантируется, что она никогда не будет удалена.

Формат выходных данных

Для каждого запроса типа «?» в выходной файл нужно вывести на отдельной строке одно число — номер вершины интересующей Юнга

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 E. 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. Последующие 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

Задача F. 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. Третья 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, ваша программа должна вывести в выходной 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 G. Дорешивание

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

В Летней Компьютерной Школе есть n параллелей, каждая из которых живёт в своём домике. Все параллели пронумерованы от 1 до n от младших к старшим. Периодически школьник, дорешивающий прошедшие практики у себя в домике, не справляется с задачей и идёт за помощью к товарищам из более старшей параллели.

Некоторые пары домиков соединены тропинками, всего есть $n-1$ такая тропинка. Все тропинки имеют одинаковую длину, по тропинке можно ходить между двумя домиками, которые она соединяет, и только между ними. От любого домика можно дойти до любого другого домика, используя только данные тропинки.

Если у школьника из параллели k не получается решить задачу, он из своего домика с номером k идёт просить помощи до какого-нибудь домика с номером, большим k. Поскольку ему не хочется тратить ни секунды драгоценного времени, он выбирает ближайший подходящий домик. Школьники из параллели n всегда решают свои задачи сами, так как им не к кому обратиться.

Вам дано описание тропинок между домиками. Для каждого k от 1 до $n-1$ определите минимальное расстояние, которое школьник из параллели k пройдёт в случае проблем с решением задачи.

Формат входных данных

Первая строка входных данных содержит целое число n ($1 \leqslant n \leqslant 200\,000$) — количество параллелей в ЛКШ.

В i-й из следующих $n-1$ строк содержатся два целых числа a_i и b_i ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$) — номера домиков, которые соединяет i-я тропинка.

Гарантируется, что каждую пару домиков соединяет не более одной тропинки, и что из любого домика можно дойти до любого другого.

Формат выходных данных

Выведите $n-1$ строку, i-я из них должна содержать целое число d_i — расстояние до ближайшего домика с номером, большим i, от домика параллели i.

Примеры

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