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. LCA

Имя входного файла:	`lca2.in`
Имя выходного файла:	`lca2.out`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 мегабайт

Дано подвешенное дерево с корнем в первой вершине. Вам нужно ответить на m запросов вида “найти LCA двух вершин“.

LCA вершин u и v в подвешенном дереве — это наиболее удалённая от корня дерева вершина, лежащая на обоих путях от u и v до корня.

Формат входных данных

В первой строке задано целое число n — число вершин в дереве ($1 \leqslant n \leqslant 2 \cdot 10^5$).

В следующих $n-1$ строках записано одно целое число x. Число x на строке i означает, что x — предок вершины $i (x < i)$.

Затем дано число m.

Далее заданы m ($0 \leqslant m \leqslant 5 \cdot 10^5$) запросов вида (u, v) — найти LCA двух вершин u и v ($1 \leqslant u, v \leqslant n$; $u \neq v$).

Формат выходных данных

Для каждого запроса выведите LCA двух вершин на отдельной строке.

Примеры

lca2.in	lca2.out
5 1 1 2 3 2 2 3 4 5	1 1
5 1 1 2 2 3 4 5 4 2 3 5	2 2 1

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 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 D. 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 E. LCA

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

Дан ориентированный граф. Подсчитайте, сколько пар вершин (i, j) имеют общего предка. Общим предком вершин i и j называется такая вершину k, что и i, и j достижимы из k.

Формат входных данных

В первой строке входного файла содержатся целые числа $1 \leqslant N \leqslant 10^4, 0 \leqslant M \leqslant 10^4$ — количество вершин и рёбер в графе. Далее следуют M строк по два числа от 1 до N. Пара чисел (a, b) означает, что из вершины a есть ребро в вершину b.

Формат выходных данных

Выведите одно целое число — количество пар.

Примеры

lca.in	lca.out
2 1 1 2	4
3 2 2 1 3 1	7

Задача F. Классическая задача про гориллу

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

Однажды огромный самец гориллы нашёл дерево из n вершин с корнем в вершине 1. Детальное исследование этого дерева показало, что в каждой вершине лежит какое-то количество бананов, а именно, в вершине с номером v находятся a_v бананов.

Совершенно внезапно, вам предстоит ответить на q запросов. Запросы бывают двух видов:

1. заменить количество бананов в вершине v на x;
2. вывести количество бананов в поддереве v.

Помогите горилле ответить на все запросы и спокойно насытиться бананами.

Формат входных данных

В первой строке даны целые числа n и q ($1 \leqslant n, q \leqslant 2 \cdot 10^5$) — количество вершин и запросов.

Во второй строке даны n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 10^9$) — количества бананов в вершинах.

Следующие $n - 1$ строк содержат описания рёбер. Каждое ребро задаётся парой вершин v и u ($1 \leqslant v, u \leqslant n$).

Следующие q строк содержат описания запросов. Запросы описываются в следующем формате:

- 1 v x — запрос первого типа;
- 2 v — запрос второго типа.

Формат выходных данных

Выведите ответ на каждый запрос второго типа в отдельной строке.

Пример

стандартный ввод	стандартный вывод
5 3 4 2 5 2 1 1 2 1 3 3 4 3 5 2 3 1 5 3 2 3	8 10

Замечание

Задача G. 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

Задача H. 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