Задача A. Командир Ciel

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

Лиса Ciel становится командиром Древоземелья. В Древоземелье, как это следует из названия, есть n городов, соединенных $n-1$ ненаправленными дорогами, а между любыми двумя городами существует путь по дорогам Древоземелья.

Лиса Ciel должна назначить каждому городу офицера. У каждого офицера есть ранг — буква от '`A`' до '`Z`'. Таким образом, имеется 26 различных рангов, самый высокий — '`A`', самый низкий — '`Z`'.

У Ciel имеется достаточно офицеров каждого ранга. Но не все так просто, должно быть выполнено особое условие: если x, y — два различных города и у их офицеров одинаковые ранги, то на простом пути между x и y должен быть город z, имеющий офицера более высокого ранга. Таким образом, общение между офицерами одного ранга будет гарантированно проходить под присмотром офицера более высокого ранга.

Помогите Ciel составить подходящий план назначения офицеров городам. Если это невозможно, выведите «`Impossible!`».

Формат входных данных

В первой строке записано целое число n ($2 \leqslant n \leqslant 10^5$) — количество городов в Древоземелье.

В каждой из следующих $n-1$ строк записано два целых числа a и b ($1 \leqslant a, b \leqslant n, a \neq b$) — это значит, что существует дорога между городами a и b. Считайте, что города пронумерованы от 1 до n некоторым образом.

Гарантируется, что заданный граф будет деревом.

Формат выходных данных

Если подходящий план существует, выведите n символов, разделенных пробелами — i-ый символ обозначает ранг офицера в городе i.

В противном случае, выведите «`Impossible!`».

Примеры

commander.in	commander.out
4 1 2 1 3 1 4	A B B B

Замечание

В первом примере для любых двух офицеров ранга '`B`', офицер с рангом '`A`' будет на пути между ними. То есть, такое решение подходит.

Задача B. Почтовая реформа

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

В Флатландии идет пора реформ. Недавно была проведена реформа дорог, так что теперь по дорогам страны из любого города можно добраться в любой другой, причем только одним способом. Также была проведена реформа волшебников, так что в каждом городе остался ровно один волшебник. Теперь же началась реформа почтовой системы.

Недавно образованное почтовое агентство «Экс-Федя» предлагает уникальную услугу — коллективную посылку. Эта услуга позволяет отправлять посылки жителям всех городов на каком-либо пути по цене обычной посылки. Удивительно, но пользоваться такой услугой стали только волшебники Флатландии, которые стали в большом количестве отправлять друг другу магические кактусы. Агентство столкнулось с непредвиденной проблемой: как известно, все волшебники живут в башнях и мало того, что не строят в них лестницы, так еще время от времени меняют их высоту. Поэтому, чтобы доставить посылку волшебнику, который живет в башне высотой h, курьеру агентства требуется иметь с собой не менее h метров веревки.

Вам поручено руководить отделом логистики — по имеющимся данным о высотах башен и об их изменениях вам нужно определять минимальную длину веревки, которую нужно выдать курьеру, который доставляет посылки между городами i и j.

Формат входных данных

Первая строка входного файла содержит число n — количество городов в Флатландии ($1 \leqslant n \leqslant 50\,000$). Во второй строке находится n положительных чисел, не превосходящих 10^5 — высоты башен в городах. В следующих $n-1$ строках содержится по два числа u_i и v_i — описание i-й дороги, $1 \leqslant u_i, v_i \leqslant n, u_i \neq v_i$. В следующий строке содержится число k — количество запросов ($1 \leqslant k \leqslant 100\,000$). В следующих k строках содержатся описания запросов в следующем формате:

- Уведомление от волшебника из города i о том, что высота его башни стала равна h, имеет вид ! i h, $1 \leqslant i \leqslant n$, $1 \leqslant h \leqslant 10^5$.
- Запрос от курьера о выдаче веревки для доставки посылок во все города на пути от i до j включительно имеет вид ? i j, $1 \leqslant i, j \leqslant n$.

Формат выходных данных

Для каждого запроса доставки посылок выведите минимальную длину веревки, которую необходимо выдать курьеру.

Примеры

mail.in	mail.out
3 1 2 3 1 3 2 3 5 ? 1 2 ! 1 5 ? 2 3 ! 3 2 ? 1 2	3 3 5
1 100 5 ! 1 1 ? 1 1 ! 1 1000 ? 1 1 ! 1 1	1 1000

Задача C. На далекой Амазонке

Имя входного файла:	`treeeg.in`
Имя выходного файла:	`treeeg.out`
Ограничение по времени:	6.5 секунд
Ограничение по памяти:	256 мегабайт

В бассейне далёкой реки Амазонки расположены N городов, пронумерованных для удобства целыми числами от 1 до N. Всем известно, что местные леса непроходимы, и передвижение возможно только по рекам. Как следствие, схема соединения городов является деревом.

К несчастью, в этом году в бассейне далёкой Амазонки не на шутку разошлась эпидемия новой болезни — крабового гриппа. То и дело поступает информация о новых заболевших. Поначалу справляться с ней было легко, но вскоре почти все больницы были переполнены, и сейчас пациентов может принимать только госпиталь, находящийся в городе 1.

Для удобства граждан была открыта горячая линия, куда первым делом необходимо обратиться при появлении симптомов крабового (его ещё часто называют раковым) гриппа. Вам необходимо написать программу, которая будет отвечать на обращения пострадавших, учитывая при этом информацию о работающих больницах. Вам ещё повезло, что вы знаете все запросы заранее!

Более формально, поступают запросы трёх видов:

- «`+` v» — госпиталь города v снова может принимать больных. Гарантируется, что в момент перед этим запросом госпиталь города v не работал.
- «`-` v» — госпиталь города v не может больше принимать больных. Гарантируется, что в момент перед этим запросом госпиталь города v работал.
- «`?` v» — заболел человек в городе v, необходимо сообщить ему расстояние до ближайшего города с работающим госпиталем (в идеале неплохо бы ещё и сказать номер этого города, но этим пусть занимаются ваши коллеги). Гарантируется, что в момент такого запроса имеется хотя бы один работающий госпиталь.

Формат входных данных

В первой строке находится единственное число N — количество городов ($1 \leqslant N \leqslant 300\,000$). Следующие $N-1$ строк содержат информацию о соединениях между городами в формате «u v l», что означает соединение между городами u и v длиною l километров ($1 \leqslant u, v \leqslant N$, $1 \leqslant l \leqslant 1000$). Направлением течения можно пренебречь и считать, что время движения зависит только от расстояний.

Далее на отдельной строке записано число Q — количество запросов. Следующие Q строк содержат описание запросов в формате «c v», где c — это один из трёх символов «`+`», «`-`» и «`?`», а v — номер города ($1 \leqslant v \leqslant N$).

Формат выходных данных

Для каждого запроса вида «`?` v» выведите на отдельной строке одно число — расстояние в километрах до ближайшего города с работающим госпиталем.

Примеры

treeeg.in	treeeg.out
5 1 2 2 2 3 3 3 4 1 3 5 4 5 ? 4 + 5 ? 3 - 1 ? 2	6 4 7

Задача D. БДБД

Имя входного файла:	`lwdb.in`
Имя выходного файла:	`lwdb.out`
Ограничение по времени:	2.5 секунд
Ограничение по памяти:	256 мегабайт

Большая Древесная База Данных создана для того, чтобы в ней можно было надежно сохранить и раскрасить любое дерево. В новой версии БДБД запланирован новый функционал, для реализации которого потребуется вновь переосмыслить теорию графов.

В БДБД хранится взвешенное дерево. В языке запросов Системы Управления Большой Древесной Базы Данных (СУБДБД) предусмотрены два вида запросов:

1. «1 v d c» — покрасить все вершины, находящиеся на расстоянии не более d от вершины v, в цвет c. Все вершины изначально окрашены в цвет с номером 0.

2. «2 v» — вывести цвет вершины v.

Необходимо запрограммировать работу СУБДБД и ответить на все запросы пользователя.

Формат входных данных

В первой строке число N ($1 \leqslant N \leqslant 10^5$) — количество вершин дерева.

Следующие $N-1$ строк содержат описание ребер, по три числа в строке a_i, b_i, w_i ($1 \leqslant a_i, b_i \leqslant N$, $a_i \neq b_i$, $1 \leqslant w_i \leqslant 10^4$), где i-ое ребро имеет вес w_i и соединяет вершины a_i и b_i.

В следующей строке число Q ($1 \leqslant Q \leqslant 10^5$) — число запросов. В каждой из Q следующих строк запросы одного из двух видов:

1. Числа 1, v, d, c ($1 \leqslant v \leqslant N$, $0 \leqslant d \leqslant 10^9$, $0 \leqslant c \leqslant 10^9$).

2. Числа 2, v ($1 \leqslant v \leqslant N$).

Все числа во входных данных целые.

Формат выходных данных

Для каждого запроса второго типа необходимо вывести в отдельной строке цвет запрошенной вершины.

Примеры

lwdb.in	lwdb.out
5 1 2 30 1 3 50 3 4 70 3 5 60 8 1 3 72 6 2 5 1 4 60 5 2 3 2 2 1 2 144 7 2 4 2 5	6 6 0 5 7

Задача E. Наименьший общий предок

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

У Бобо есть корневое дерево из n вершин, удобно пронумерованных числами $1, 2, \ldots, n$. Вершина 1 — корень дерева, и i-я вершина имеет вес w_i.

Он хотел бы посчитать $f(2), f(3), \ldots, f(n)$ где

$$f(i) = \sum_{j=1}^{i-1} w_{\mathrm{LCA}(i,j)}.$$

Формат входных данных

Входные данные содержит ноль или более тестовых примеров и заканчиваются символом конца файла. Для каждого тестового примера:

Первая строка содержит число n ($2 \leqslant n \leqslant 2 \cdot 10^5$).

Вторая строка содержит n чисел $w_1, w_2, \ldots, w_n$ ($1 \leqslant w_i \leqslant 10^4$).

Третья строка содержит $(n-1)$ чисел $p_2, p_3, \ldots, p_n$, где p_i обозначает ребро из вершины p_i в вершину i ($1 \leqslant p_i \leqslant n$). Ребра образуют дерево.

Гарантируется, что сумма всех n не превосходит $2 \cdot 10^5$.

Формат выходных данных

Для каждого тестового примера, выведите $n-1$ чисел: $f(2), f(3), \ldots, f(n)$.

Примеры

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

Задача F. Близкие вершины

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

Дано дерево из n вершин. Каждое ребро имеет неотрицательный вес. Длиной пути между двумя вершинами называется количество ребер в пути. Весом пути называется суммарный вес всех входящих в него ребер.

Две вершины называются близкими, если существует путь между двумя этими вершинами длины не более l и также существует путь между ними веса не более w. Определите количество близких пар вершин.

Формат входных данных

В первой строке записаны три целых числа n, l и w ($1 \leqslant n \leqslant 10^5, 1 \leqslant l \leqslant n, 0 \leqslant w \leqslant 10^9$). Далее в $n-1$ строках дано описание ребер дерева. В i-той строке записано два целых числа p_i, w_i ($1 \leqslant p_i < (i+1), 0 \leqslant w_i \leqslant 10^4$), которые обозначают, что i-ое ребро соединяет вершину $(i+1)$ и p_i и имеет вес w_i.

Считайте, что вершины дерева пронумерованы от 1 до n некоторым образом.

Формат выходных данных

Выведите единственное целое число — количество близких пар.

Примеры

close-vertices.in	close-vertices.out

Задача G. Спички детям не игрушка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мегабайт

Лена играется со спичками. Естественный вопрос, посещающий любого школьника, играющего со спичками — а можно ли поджечь спичкой дерево?

Скажем, что дерево — это связный граф без циклов, вершины которого пронумерованы целыми числами $1, 2, \ldots, n$, в каждой вершине которого также записано некоторое целое число p_v, являющееся приоритетом вершины v. Все приоритеты различны.

Оказывается, что если поджечь дерево, то оно, как и можно было ожидать, сгорит целиком. Однако процесс этот не быстрый. Сначала у дерева сгорает лист (*листом* называется вершина, имеющая ровно одного соседа) с минимальным приоритетом, затем сгорает лист с минимальным приоритетом из оставшихся вершин дерева, и так далее. Таким образом, вершины превращаются в листья и сгорают до тех пор, пока от дерева не останется лишь одна вершина, после чего она тоже сгорает.

Лена приготовила дерево из n вершин и в каждой вершине записала приоритет $p_v = v$. Лене с одной стороны интересно посмотреть, как горит дерево, но с другой она понимает, что если дерево поджечь, оно исчезнет насовсем. Лена добрая девочка, и деревья ей жалко, так что она хочет ограничиться выяснением ответов на некоторые вопросы про процесс сгорания дерева в уме. Лена хочет ответить на q вопросов, каждый из которых относится к одному из трёх следующих видов:

- «`up` v», присвоить вершине v приоритет $1 + \max\{p_1, p_2, \ldots, p_n\}$;
- «`when` v», выяснить, какой по счёту сгорит вершина v, если дерево поджечь сейчас;
- «`compare` v u», выяснить, какая из вершин v и u сгорит раньше, если дерево поджечь сейчас.

Заметим, что если приоритеты всех вершин сейчас различны, то и после выполнения запроса «`up`» они тоже останутся различными. Исходно они различны, поэтому в любой момент времени порядок сгорания листьев определён однозначно.

Формат входных данных

Первая строка содержит два целых числа n и q ($2 \leqslant n \leqslant 200\,000$, $1 \leqslant q \leqslant 200\,000$) — количество вершин дерева и количество вопросов.

В i-й из следующих $n - 1$ строк находятся два целых числа v_i, u_i ($1 \leqslant v_i, u_i \leqslant n$), задающие концы i-го ребра дерева.

Каждая из оставшихся q строк содержит операцию одного из трёх типов.

- «`up` v» ($1 \leqslant v \leqslant n$) — присвоить новый приоритет вершине v;
- «`when` v» ($1 \leqslant v \leqslant n$) — определить момент сгорания вершины v для текущего дерева;
- «`compare` v u» ($1 \leqslant v, u \leqslant n$, $v \neq u$) — определить, какая из вершин v и u сгорит раньше для текущего дерева.

Гарантируется, что среди запросов хотя бы один имеет тип «`when`» или «`compare`».

Формат выходных данных

Для каждого запроса типа «`when`» нужно вывести одно целое число от 1 до n — момент времени, когда сгорит вершина v.

Для запроса типа «`compare`» выведите v или u, в зависимости от того, какая вершина сгорит раньше.

Примеры

стандартный ввод	стандартный вывод
5 7 1 5 1 2 1 3 4 3 when 1 when 2 when 3 when 4 when 5 compare 2 3 compare 3 4	4 1 3 2 5 2 4
5 5 1 5 1 2 1 3 4 3 up 1 compare 2 4 compare 4 3 compare 3 1 compare 1 5	2 4 3 5