Задача A. Декомпозиция

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

Рассмотрим дерево T. Назовем *деревом декомпозиции* корневое дерево $D(T)$.

Выберем любую из вершин дерева T, назовем ее r. Рассмотрим все компоненты связности дерева T, после удаления вершины r: $S_1, S_2, \ldots, S_k$. Тогда корнем $D(T)$ будет вершина r, а детьми r в $D(T)$ будут $D(S_1), D(S_2), \ldots, D(S_k)$.

Вам задано T. Найдите дерево декомпозиции, высота которого не более 20. Высотой дерева называется максимальное число вершин, которые может содержать простой путь начинающийся в корне.

Формат входных данных

Первая строка содержит n — число вершин дерева T ($1 \leqslant n \leqslant 2 \cdot 10^5$).

Следующие $n-1$ строк содержат ребра дерева. Каждое ребро описывается парой чисел v_i, u_i — концы ребра ($1 \leqslant v_i, u_i \leqslant n$).

Формат выходных данных

Выведите n чисел: i-е число — родитель вершины i в дереве декомпозиции, если вершина является корнем, выведите 0.

Примеры

decomposition.in	decomposition.out
3 1 2 2 3	2 0 2
9 3 2 4 2 1 2 5 1 1 6 7 6 6 8 8 9	0 1 2 2 1 1 6 6 8

Задача B. 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
10 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10	D C B A D C B D C D

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 C. На далекой Амазонке

Имя входного файла:	`treeeg.in`
Имя выходного файла:	`treeeg.out`
Ограничение по времени:	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. Близкие вершины

Имя входного файла:	`close-vertices.in`
Имя выходного файла:	`close-vertices.out`
Ограничение по времени:	5 секунд
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
4 4 6 1 3 1 4 1 3	4
6 2 17 1 3 2 5 2 13 1 6 5 9	9

Задача E. БДБД

Имя входного файла:	`lwdb.in`
Имя выходного файла:	`lwdb.out`
Ограничение по времени:	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-е 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 F. Сигнализация

Имя входного файла:	`alarm.in`
Имя выходного файла:	`alarm.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 МБ

Подземный бункер состоит из n комнат, соединённых $n-1$ коридорами. Каждый коридор соединяет две различные комнаты и имеет определённую длину. Бункер устроен таким образом, что из любой комнаты i можно дойти в любую другую комнату j. Заметим, что существует единственный такой путь, не проходящий по одному и тому же коридору дважды. Сумма длин коридоров, составляющих этот путь, называется расстоянием между комнатами i и j и обозначается $\rho(i, j)$.

Каждая комната бункера оборудована звуковой сигнализацией, состоящей из сирены и датчика звука, который её включает. Сирена, включённая в комнате i, активирует датчик звука в каждой комнате, расстояние до которой не превосходит расстояние d_i, определяемое мощностью этой сирены. Другими словами, включение сирены в комнате i автоматически включает сирену во всех комнатах j, таких что $\rho(i, j) \leqslant d_i$. Эта сирена, в свою очередь, может вызвать автоматическое включение других сирен и так далее.

В случае возникновения чрезвычайной ситуации некоторые сирены необходимо включить вручную, после чего звук от них автоматически включит сирены в других комнатах. Правила безопасности предписывают выбор такого набора сирен для ручного включения, который в конце концов приведёт к автоматическому включению сирен во всех комнатах.

Требуется написать программу, которая определяет минимальное количество сирен в наборе, удовлетворяющем правилам безопасности.

Формат входных данных

Первая строка входных данных содержит единственное число n — количество комнат.

Вторая строка содержит последовательность из n целых чисел d_i, i-е из них равно максимальному расстоянию, на котором расположенная в комнате i сирена активирует датчики ($0 \leqslant d_i \leqslant 10^9$).

Последующие $n-1$ строк описывают коридоры бункера. В i-й из них находятся три целых числа: u_i, v_i, l_i, где u_i, v_i — номера различных комнат, соединённых коридором i, а l_i — длина этого коридора ($1 \leqslant u_i, v_i \leqslant n$; $1 \leqslant l_i \leqslant 10^9$).

Формат выходных данных

Выходные данные должны состоять из единственного числа — минимального количества сирен, которые необходимо включить вручную.

Примеры

alarm.in	alarm.out
10 1 2 2 2 6 3 4 5 4 3 1 2 5 2 3 1 2 4 5 4 5 2 4 6 4 4 7 3 1 8 1 8 9 5 8 10 4	3

Замечания

В тесте из примера сирена в комнате 4 включает сирену в комнате 5, которая, в свою очередь, включает сирены в комнатах 6 и 7. Сирена в комнате 2 включает сирену в комнате 3. Сирена в комнате 8 включает сирены в комнатах 1, 9 и 10.

1

2

3

4

5