Задача A. Адам и дерево

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Когда у Адама появляется корневое дерево (связный неориентированный граф без циклов), он сразу начинает его раскрашивать. Более формально, каждому ребру он сопоставляет некоторый цвет таким образом, чтобы выполнялись два условия:

- Не существует вершины, у которой больше двух инцидентных ребер покрашены в один цвет.
- Для любых двух вершин, у которых есть инцидентные ребра, покрашенные в один цвет (скажем, c), путь между ними содержит ребра только цвета c.

Не все раскраски дерева нравятся Адаму одинаково. Рассмотрим путь от некоторой вершины до корня. Количество различных цветов на этом пути назовем стоимостью вершины. Стоимостью раскраски дерева будем называть максимальную стоимость среди всех вершин дерева. Помогите Адаму определить минимальную стоимость раскраски дерева.

Изначально дерево Адама состоит из одной вершины, которая имеет номер один и является корнем. За один ход Адам подвешивает к уже существующей вершине новую, которая получает номер, равный наименьшему положительному целому не занятому числу. После каждой операции вам нужно сообщать минимальную стоимость раскраски получившегося дерева.

Формат входных данных

В первой строке задано целое число n ($1 \leqslant n \leqslant 10^6$) — количество добавлений новых вершин. Во второй строке записано n чисел p_i ($1 \leqslant p_i \leqslant i$) — номера вершин, к которым подвешивают очередную вершину.

Формат выходных данных

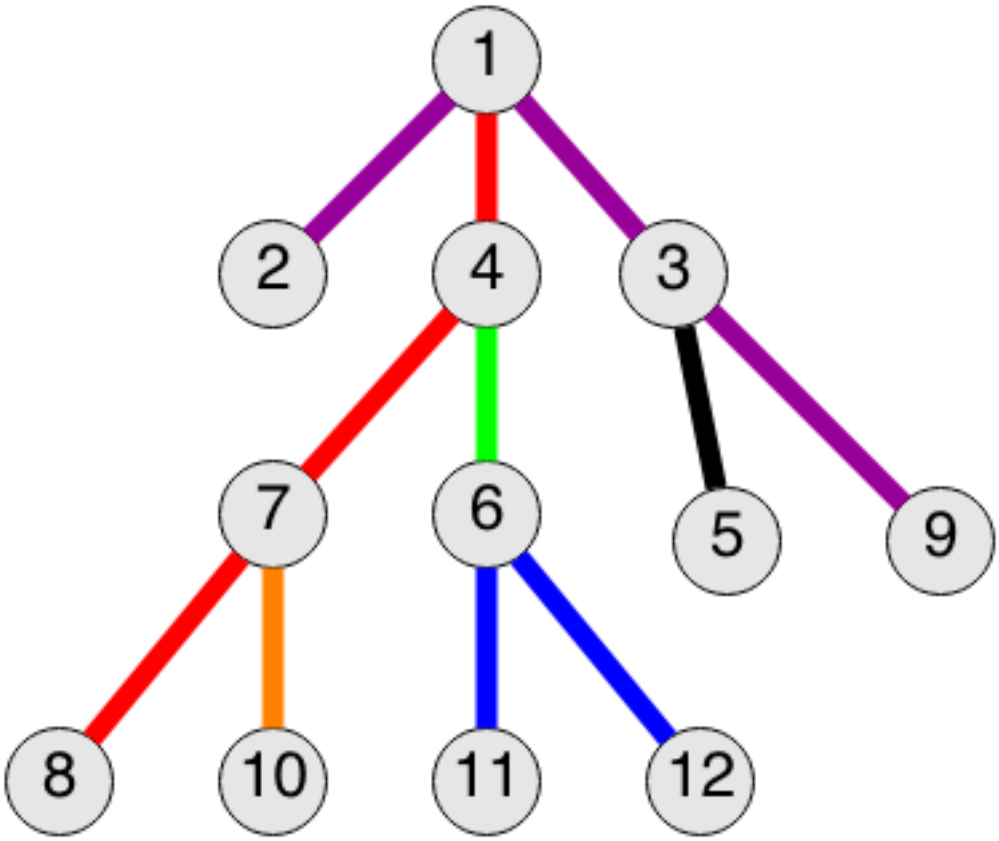
Выведите n целых чисел — минимальные стоимости раскраски дерева после каждого добавления.

Примеры

стандартный ввод	стандартный вывод
11 1 1 1 3 4 4 7 3 7 6 6	1 1 1 1 1 2 2 2 2 2 3

Замечание

На картинке изображен один из возможных вариантов раскраски дерева из примера в самый последний момент. Стоимость вершин с номерами 11 и 12 равна трем.


1
2
4
3
7
6
5
9
8
10
11
12

Задача B. Двоичное дерево поиска

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

Реализуйте сбалансированное двоичное дерево поиска.

Формат входных данных

Входной файл содержит описание операций с деревом, их количество не превышает 100000. В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x. Если ключ x в дереве уже есть, то ничего делать не надо.
- `delete` x — удалить из дерева ключ x. Если ключа x в дереве нет, то ничего делать не надо.
- `exists` x — если ключ x есть в дереве, выведите «`true`», иначе «`false`»
- `next` x — выведите минимальный элемент в дереве, строго больший x, или «`none`», если такого нет.
- `prev` x — выведите максимальный элемент в дереве, строго меньший x, или «`none`», если такого нет.

Все числа во входном файле целые и по модулю не превышают 10^9.

Формат выходных данных

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`. Следуйте формату выходного файла из примера.

Примеры

bst.in	bst.out
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4	true false 5 3 none 3

Задача C. 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 D. Динамический Лес

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

Вам нужно научиться обрабатывать 3 типа запросов:

1. Добавить ребро в граф (`link`).
2. Удалить ребро из графа (`cut`).
3. По двум вершинам a и b вернуть длину пути между ними (или -1, если они лежат в разных компонентах связности) (`get`).

Изначально граф пустой (содержит N вершин, не содержит ребер). Гарантируется, что в любой момент времени граф является лесом. При добавлении ребра гарантируется, что его сейчас в графе нет. При удалении ребра гарантируется, что оно уже добавлено.

Формат входных данных

Числа N и M ($1 \leqslant N \leqslant 10^5 + 1$, $1 \leqslant M \leqslant 10^5$) — количество вершин в дереве и, соответственно, запросов. Далее M строк, в каждой строке команда (`link` или `cut`, или `get`) и 2 числа от 1 до N — номера вершин в запросе.

Формат выходных данных

В выходной файл для каждого запроса `get` выведите одно число — расстояние между вершинами, или -1, если они лежат в разных компонентах связности.

Примеры

linkcut.in	linkcut.out
3 7 get 1 2 link 1 2 get 1 2 cut 1 2 get 1 2 link 1 2 get 1 2	-1 1 -1 1
5 10 link 1 2 link 2 3 link 4 3 cut 3 4 get 1 2 get 1 3 get 1 4 get 2 3 get 2 4 get 3 4	1 2 -1 1 -1 -1

Задача E. 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

Задача F. Кратчайший путь

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

В некотором царстве, в некотором государстве есть несколько городов, соединённых двусторонними дорогами. При этом страна разбита на провинции, внутри каждой провинции можно добраться от любого города до любого другого, а между провинциями пути не существует. Внутри каждой провинции есть один выделенный город, который называется столицей этой провинции.

Компания «AloeVera» занимается пассажирскими перевозками в этом государстве. Компания перевозит людей от одного города до другого по кратчайшему пути. Однако, дорожная сеть постоянно развивается, поэтому периодически в государстве появляются новые дороги. Строительство новой дороги — очень важное событие, поэтому при появлении новой дороги все стремятся проехать именно по ней. Более точно, при добавлении новой дороги между городами u и v существует две возможности:

1. Города u и v находятся в разных провинциях. Тогда после добавления этой дороги эти две провинции объединяются, при этом столицей становится столица первой провинции (той, в которой до добавления находилась вершина u).

2. Города u и v находятся в одной провинции. Тогда происходит перераспределение транспортных потоков. Услышав об открытии новой дороги, все продолжают добираться из столицы в другие города провинции по кратчайшему пути, но в случае, если таких путей существует несколько, то предпочтение отдается тому из них, на котором находится новая построенная дорога. После этого, если найдутся дороги, по которым никто ездить не будет, они навсегда удаляются из дорожной сети. Будьте внимательны: может так случиться, что по новой дороге никто ездить не будет, и тогда её не нужно добавлять вообще.

Иногда в государстве некоторые дороги приходят в негодность. При этом, если после удаления этой дороги некоторая часть провинции становится недостижимой из столицы, то она становится независимой провинцией, и её столицей назначается город, находящийся в этой провинции и являющийся концом удалённой дороги.

Основная же задача компании — сообщать людям о кратчайшем расстоянии между двумя какими-то городами. Ваша задача состоит в том, чтобы автоматизировать работу «AloeVera». При этом люди, сообщающие о разрушении дорог, иногда запаздывают или оперируют неверной информацией, и сообщают о разрушении дороги, которая уже была разрушена или вообще никогда не существовала.

Для решения поставленной задачи компания «AloeVera» использует следующий интерфейс:

- `1 u v` — запрос об удалении дороги между городами u и v. Будьте внимательны, эта дорога могла быть разрушена ранее, или вообще никогда не существовать!

- `2 u v w` — запрос о добавлении новой дороги между городами u и v, время проезда по которой будет равно w ($1 \leqslant w \leqslant 10^4$).

- `3 u v` — запрос кратчайшего расстояния от города u до города v. В случае, если города находятся в разных провинциях, длина полагается равной -1.

Во всех запросах номера городов корректны, то есть $1 \leqslant u, v \leqslant N$, где N обозначает общее число городов в государстве.

Формат входных данных

В первой строке входного файла содержится число N — количество городов в государстве. В следующей строке содержится число M — количество запросов. Далее в каждой строке содержится описание запроса. Описание соответствует условию.

$1 \leqslant N \leqslant 50\,000$, $1 \leqslant M \leqslant 100\,001$.

Формат выходных данных

Для каждого запроса типа 3 выведите единственное число: ответ на запрос.

Примеры

shortest.in	shortest.out
5 8 2 1 3 10 2 1 2 3 1 1 2 3 1 3 2 1 3 9 2 2 3 1 3 1 5 3 1 3	10 -1 9
5 10 2 2 3 2 2 3 1 2 2 1 5 2 2 5 4 2 2 2 3 2 2 1 5 2 1 3 1 2 3 1 1 2 3 1 2 3 3 1	1