Задача A. 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 B. Декомпозиция

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

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 C. 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

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. Пути в дереве

Имя входного файла:	`tree-paths.in`
Имя выходного файла:	`tree-paths.out`
Ограничение по времени:	15 секунд
Ограничение по памяти:	512 мегабайт

Дано дерево из n вершин. Найдите для каждого d от 1 до $n-1$ число путей длины d.

Формат входных данных

Первая строка содержит n — число вершин дерева ($1 \leqslant n \leqslant 50000$).

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 $n-1$ число: i-е число — число путей длины i.

Примеры

tree-paths.in	tree-paths.out
3 1 2 2 3	2 1
9 3 2 4 2 1 2 5 1 1 6 7 6 6 8 8 9	8 10 10 6 2 0 0 0

Задача F. 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.
- `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 G. Двоичное дерево поиска (маленькое, 12,5 баллов)

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

Реализуйте двоичное дерево поиска.

Формат входных данных

Входной файл содержит описание операций с деревом, их количество не превышает 100. В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x.
- `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 H. Переворот

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

Дан массив. Надо научиться обрабатывать два типа запросов.

- `1 L R` - перевернуть отрезок [L, R]
- `2 L R` - найти минимум на отрезке [L, R]

Формат входных данных

Первая строка файла содержит два числа n, m. ($1 \leqslant n, m \leqslant 10^5$) Во второй строке находится n чисел a_i ($1 \leqslant a_i \leqslant 10^9$)- исходный массив. Остальные m строк содержат запросы, в формате описанном в условии. Для чисел L,R выполняется ограничение ($1 \leqslant L \leqslant R \leqslant n$).

Формат выходных данных

На каждый запрос типа 2, во входной файл выведите ответ на него, в отдельной строке.

Примеры

reverse.in	reverse.out
10 7 5 3 2 3 12 6 7 5 10 12 2 4 9 1 4 6 2 1 8 1 1 8 1 8 9 2 1 7 2 3 6	3 2 2 2