y2019-2-3. Запросы на деревьях

A. Двоичные подъемы

2 секунды, 256 мегабайт

Задано подвешенное дерево. Найдите для каждой вершины двоичные подъемы: предков, которые находятся от нее на расстоянии 2^k для какого-либо целого k.

Входные данные
В первой строке входа задано число n ($1 \le n \le 10^5$) — число вершин дерева. Во второй строке заданы n чисел p_i. Число p_i равно номеру вершины, являющейся предком вершины i (вершины нумеруются с 1) или нулю, если вершина i — корень дерева.

Выходные данные
Выведите n строк. В i-й строке выведите номер вершины i и далее после двоеточия список требуемых предков, в порядке увеличения расстояния от i.

входные данные
8 5 8 5 0 4 5 4 1

выходные данные
1: 5 4 2: 8 1 4 3: 5 4 4: 5: 4 6: 5 4 7: 4 8: 1 5

B. LCA

5 секунд, 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.

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

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

Выходные данные
Для каждого запроса выведите LCA двух вершин на отдельной строке.

входные данные
5 1 1 2 3 2 2 3 4 5

выходные данные
1 1

входные данные
5 1 1 2 2 3 4 5 4 2 3 5

выходные данные
2 2 1

C. Самое дешевое ребро

4 секунды, 256 мегабайт

Дано подвешенное дерево с корнем в первой вершине. Все ребра имеют веса (стоимости). Вам нужно ответить на M запросов вида "найти у двух вершин минимум среди стоимостей ребер пути между ними".

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

В следующих n - 1 строках записаны два целых числа x и y. Число x на строке i означает, что x — предок вершины i, y задает стоимость ребра ($x < i$; $|y| \le 10^6$).

Далее заданы m ($0 \le m \le 5 \cdot 10^5$) запросов вида (x, y) — найти минимум на пути из x в y ($1 \le x, y \le n; x \ne y$).

Выходные данные
Выведите ответы на запросы.

входные данные
5 1 2 1 3 2 5 3 2 2 2 3 4 5

выходные данные
2 2

входные данные
5 1 1 1 2 2 3 3 4 2 1 4 3 2

выходные данные
1 1

D. Опекуны карнотавров

2 секунды, 512 мегабайт

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.

Входные данные

Во входном файле записано число M, обозначающее количество запросов ($1 \le M \le 200000$). Далее на отдельных строках следуют M запросов, обозначающих следующие 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.

Выходные данные

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.

входные данные
11 + 1 + 1 + 2 ? 2 3 ? 1 3 ? 2 4 + 4 + 4 - 4 ? 5 6 ? 5 5

выходные данные
1 1 2 2 5

E. Трамваи

2 секунды, 256 мегабайт

Правительство небольшого города Мухоловска решило улучшить транспортную ситуацию в своем городе. Для этого была построена сеть трамвайных путей, соединяющая n трамвайных остановок. Для удобства пассажиров между каждой парой остановок можно было проехать на трамвае. С другой стороны, в целях экономии, проехать между двумя остановками можно было единственным образом. Формально говоря, трамвайная сеть представляет собой дерево с n вершинами. При этом вершины дерева соответствуют остановкам, а ребра — путям.

Изначально по каждому трамвайному пути проходил хотя бы один трамвайный маршрут. Однако со временем некоторые маршруты оказались отменены, а, следовательно, и некоторые трамвайные пути стали невостребованными. Путь считается невостребованным, если ни один трамвайный маршрут по нему не проходит. С целью экономии средств невостребованные трамвайные пути Мухоловска было решено разобрать.

Ваша задача — написать программу для определения числа невостребованных путей.

Входные данные

Первая строка содержит единственное число n — количество трамвайных остановок города ($2 \le n \le 100\,000$). Каждая из следующих $n - 1$ строк содержит описание одного трамвайного пути (ребра дерева). Описание состоит из двух чисел b и e — номеров остановок, соединенных соответствующим путем. Остановки пронумерованы целыми числами от 1 до n.

В следующей строке содержится число m — количество трамвайных маршрутов ($0 \le m \le 100\,000$). В каждой из следующих m строк содержится описание трамвайного маршрута. Описание состоит из двух чисел x и y — трамвайный маршрут имеет конечные остановки с номерами x и y и проходит по кратчайшему пути между ними ($x \ne y$).

Выходные данные

В выходной файл выведите количество невостребованных трамвайных путей Мухоловска.

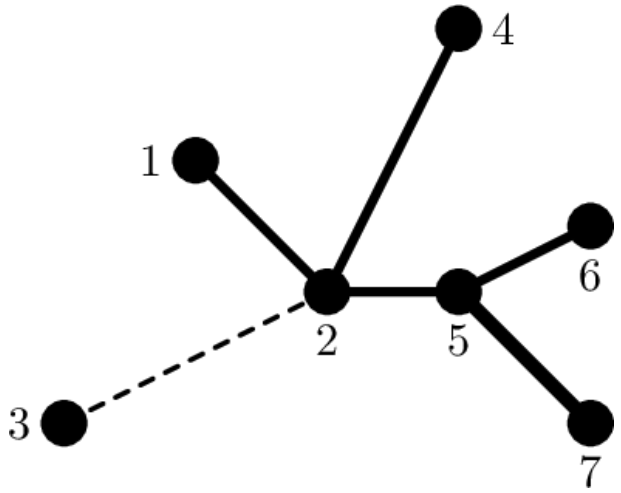
входные данные
4 1 2 1 3 1 4 0

выходные данные
3

входные данные
7 1 2 2 3 2 4 5 2 5 6 7 5 3 1 7 2 4 7 6

выходные данные
1

Иллюстрация ко второму примеру.



Пунктирной линией обозначен невостребованный путь.

F. Генеалогия

4 секунды, 256 мегабайт

Во время обсуждений в Парламенте лорды, с похожими взглядами на решение проблемы, обычно объединяются в группы. Как правило, результат обсуждения зависит от решения наиболее влиятельной группы лордов. Именно поэтому подсчёт влиятельности группы является наиболее важной задачей.

Естественно, каждый лорд дорожит древностью своего рода, поэтому влиятельность лорда равна древности его рода. Древность рода лорда — количество предков лорда: его отец, его дед, его прадед, и т.д. Чтобы посчитать влиятельность группы лордов, требуется посчитать количество лордов в группе вместе с их предками. Отметим, что если лорд является предком двух или более лордов в группе, то этот лорд должен быть посчитан только один раз.

Вам дано фамильное дерево лордов (удивительно, но все лорды произошли от одного пра-лорда) и список групп. Для каждой группы найдите её влиятельность.

Входные данные

Первая строка входного файла содержит число n — количество лордов ($1 \le n \le 100\,000$). Лорды нумеруются целыми числами от 1 до n. Следующая строка содержит n целых чисел $p_1, p_2, ..., p_n$, где p_i — отец лорда с номером i. Если лорд является основателем рода, то p_i равно -1. Гарантируется, что исходные данные формируют дерево. Третья строка входного файла содержит одно число g — количество групп ($1 \le g \le 3\,000\,000$). Следующие g строк содержат описания групп. j-ая строка содержит число k_j — размер j-ой группы, после которого следуют k_j различных чисел — номера лордов, состоящих в j-ой группе. Гарантируется, что сумма всех k_j во входном файле не превосходит $3\,000\,000$.

Выходные данные

В выходной файл выведите g строк. В j-ой строке выведите единственное число: влиятельность j-ой группы. Гарантируется, что размер выходного файла не превосходит шести мегабайт.

входные данные
4 -1 1 2 3 4 1 4 2 3 4 3 2 3 4 4 1 2 3 4

выходные данные
4 4 4 4

входные данные
5 2 -1 1 2 3 10 3 3 4 1 3 2 4 3 4 1 3 5 4 1 4 2 2 3 3 1 4 3 1 2 3 3 4 5 1 1 3 1 2 4

выходные данные
4 4 5 2 3 4 1 5 2 3

G. Прибавление на пути

2 секунды, 256 мегабайт

Задано дерево. В каждой вершине есть значение, изначально все значения равны нулю. Требуется обработать запрос прибавления на пути и запрос значения в вершине.

Входные данные

В первой строке задано целое число n — число вершин в дереве ($1 \le n \le 3 \cdot 10^5$).

В следующих $n - 1$ строках заданы ребра дерева: по два целых числа v и u в строке — номера вершин, соединенных ребром ($1 \le v, u \le n$).

В следующей строке задано целое число m — число запросов ($1 \le m \le 5 \cdot 10^5$).

Следующие m строк содержат запросы в одном из двух форматов:

- `+ v u d` — прибавить число d во все значения в вершинах на пути от v до u ($1 \le v, u \le n$; $1 \le d \le 10^9$);
- `? v` — вывести значение в вершине v ($1 \le v \le n$).

Выходные данные

Выведите ответы на все запросы.

входные данные
5 1 2 1 3 3 4 3 5 5 + 2 5 1 ? 3 + 1 1 2 ? 1 ? 3

выходные данные
1 3 1

H. Связность в дереве

2 секунды, 64 мегабайта

Есть граф из n вершин. Требуется обрабатывать следующие запросы:

- `link U V` — добавить ребро UV. Гарантируется, что до этого запроса вершины U и V были в разных компонентах связности.
- `cut U V` — удалить ребро UV. Гарантируется, что такое ребро существовало.
- `connected U V` — проверить, правда ли вершины U и V лежат в одной компоненте связности.

Входные данные
Первая строка содержит два числа n ($2 \le n \le 10^5$) и m ($1 \le m \le 10^5$) — число вершин и число операций. Следующие m строк содержат операции.

Выходные данные
Для каждой операции `connected V U` выведите `1`, если вершины в одной компоненте или `0` если в разных.

входные данные
5 10 link 2 5 link 1 5 connected 1 2 cut 2 5 connected 1 2 connected 5 1 link 2 3 link 2 4 link 3 5 connected 1 2
выходные данные
1 0 1 1

I. Размер компонент

2 секунды, 64 мегабайта

Есть граф из n вершин. Требуется обрабатывать следующие запросы:

- `link U V` — добавить ребро UV. Гарантируется, что до этого запроса вершины U и V были в разных компонентах связности.
- `cut U V` — удалить ребро UV. Гарантируется, что такое ребро существовало.
- `size V` — узнать размер компоненты связности вершины V.

Входные данные
Первая строка содержит два числа n ($2 \le n \le 10^5$) и m ($1 \le m \le 10^5$) — число вершин и число операций. Следующие m строк содержат операции.

Выходные данные
Для каждой операции `connected V U` выведите `1`, если вершины в одной компоненте или `0` если в разных.

входные данные
5 10 link 2 5 link 1 5 size 1 cut 2 5 size 1 size 2 link 2 3 link 2 4 link 3 5 size 1
выходные данные
3 2 1 5

J. Декомпозиция

2 секунды, 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: $S_1, S_2, ..., S_k$. Тогда корнем $D(T)$ будет вершина r, а детьми r в $D(T)$ будут $D(S_1), D(S_2), ..., D(S_k)$.

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.

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

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

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

входные данные
3 1 2 2 3
выходные данные
2 0 2

входные данные
9 3 2 4 2 1 2 5 1 1 6 7 6 6 8 8 9
выходные данные
0 1 2 2 1 1 6 6 8

K. Минимум в окрестности

4 секунды, 256 мегабайт

Рассмотрим дерево из n вершин. Требуется отвечать на запросы: найти минимальный номер вершины, находящейся на расстоянии не более d от вершины v.

Входные данные
Первая строка содержит n — число вершин дерева ($1 \le n \le 2 \cdot 10^5$) и m — число запросов ($1 \le m \le 10^5$) .

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

Следующие m строк содержат запросы, каждый вопрос задается двумя числами: номер вершины и расстояние.

Выходные данные
Для каждого запроса выведите ответ на него.

входные данные
3 3 1 2 2 3 3 1 2 0 2 2

выходные данные
2 2 1

входные данные
9 5 3 2 4 2 1 2 5 1 1 6 7 6 6 8 8 9 9 1 6 2 2 6 7 1 2 4

выходные данные
8 1 1 6 1

L. Черно-белое дерево

4 секунды, 256 мегабайт

Рассмотрим дерево из n вершин. Каждая вершина покрашена в черный или белый цвет. Изначально все вершины черные. Требуется отвечать на два типа запросов:

1. Поменять цвет вершины.
2. Найти сумму расстояний от заданной вершины до всех вершин того же цвета.

Входные данные
Первая строка содержит n — число вершин дерева ($1 \le n \le 2 \cdot 10^5$) и m — число запросов ($1 \le m \le 10^5$) .

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

Следующие m строк содержат запросы, каждый вопрос задается двумя числами: тип запроса (1 или 2) и номер вершины.

Выходные данные
Для каждого запроса второго типа выведите ответ на него.

входные данные
3 3 1 2 2 3 2 1 1 2 2 2

выходные данные
3 0

входные данные
9 5 3 2 4 2 1 2 5 1 1 6 7 6 6 8 8 9 2 1 1 2 2 6 1 5 2 2

выходные данные
14 13 2