1 LCA

1.1 Постановка

LCA (Least common ancestor) двух вершин подвешенного дерева, называется самая глубокая вершина, которая является предком обеих вершин.

Несложно проверить, что это также самая близкая к корню вершина на единственном пути между ними.

1.2 Двоичные подъемы

Для каждой вершина предподсчитаем предка на уровне 2^k. Это делается динамикой $up[k][u] = up[k-1][up[k-1][u]]$. Предком корня считаем сам корень. Также предподсчитаем высоту каждой вершины.

Такой предподсчет поволяет подниматься на высоту h за время $O(\log n)$, разложив h в сумму степеней двойки.

Первым делом, выравнеем высоту. Теперь, если веришны совпадают, то это и есть LCA. Иначе будем подбирать на сколько подняться в виде суммы степеней двойки, начиная с больших. Поднимаемся, если попадаем в разные вершины. Таким образом получены вершины, которые являются сыновьями LCA в соответствующих поддеревьях.

Итого, задача решается за $O(n \log n)$ предподсчета и $O(\log n)$ на ответ на запрос.

1.3 Эйлеров обход

Пройдем по дереву обходом в глубину. При этом, каждый раз, когда входим в новую вершину, или возвращаемся из рекурсии, будем выписывать вершину. В результате получется список из $2 \cdot n$ веришн. Для каждой вершины найдем первое вхождение $first[u]$. Тогда $LCA(u, v)$ — вершина с минимальной высотой на отрезке $first[u]..first[v]$.

Если искать минимум на отрезке деревом отрезков, то получается решение за $O(n)$ предподсчета и $O(\log n)$ на ответ. Если использовать разреженные таблицы, то за $O(n \log n)$ и $O(1)$.

1.4 Идея 4-х русских

Рассмотрим Эйлеров обход. Разобьем его на блоки длиной $\frac{\log n}{2}$. В каждом блоке предподсчитаем минимумы на суффиксах и префиксах, а также минмум во всем блоке. Построим разреженную таблицу на блоках. Таким образом мы можем отвечать на все запросы, кроме запросов внутри

блоков за $O(n)$ предподсчета и $O(1)$ на запрос. Заметим, что высоты соседних вершин в блоке отличаются на ± 1. Значит различных возможных блоков $O(\sqrt{n})$. Тогда предподсчаем для каждого блока его тип, а для каждого типа ответы на все запросы. Тогда мы можем отвечать на любой запрос за $O(1)$, за $O(\sqrt{n}\log^2 n) = o(n)$.

1.5 Сведение задачи Static RMQ к LCA

Пусть мы хотим решать задачу минимума на отрезке, без запросов на изменение. Построим декартово дерево в котором в качестве ключей будет позиция элемента, а в качесве приоритета его значение. В таком случае, минимум на отрезке находится ровно в LCA этих двух веришин. Таким образом такая задача решается за $O(N)$ предподсчета и $O(1)$ на ответ, использую технику, описанную выше.

1.6 Оффлайн алгоритм Тарьяна

Рассмотрим оффлайн версию задачи. То есть, пусть все запросы известны заранее.

Тогда для каждой вершины будем хранить все запросы. Обойдем дерево обходом в глубину. Будем хранить вершины в системе непересекающихся множеств. При выходе из вершины, будем объединять ее множество с множеством отца. Тогда при заходе во вторую вершину из запроса, самая высокая вершина в множестве первой является их LCA. С помощью системы непересекающихся множеств это можно реализовать за $O((N+Q)\alpha(n))$.

2 LA

2.1 Постановка

По вершине деерва хотим найти предка на какой-то высоте. Высота также является параметром запроса.

2.2 Двоичные подъемы

Идея двоичных подъемов дословно переносится с LCA.

2.3 Long-path decomposition

Разобьем дерево на пути так, что из каждой вершины путь продолжается в самое глубокое поддерево. Удлиним каждый из полученных путей в два раза. Мы получили набор путей суммарной длины $2n$. Внутри пути мы можем подниматься на любую высоту за $O(1)$. Кроме того, у каждый вершины есть путь по которому мы можем подняться хотя бы в два раза. Тогда если каждый раз подниматься либо до верха этого пути, либо до ответа, нам понадобится $O(\log n)$ итераций.

2.4 Соединение идей

Предподсчитаем и двоичные подъемы и Long-path decomposition. Предподсчитаем для каждой высоты наибольшую степень 2 не превосходящую ее. Сделаем подъем на эту степень двойки. Тогда Long-path decomposition позволит за 1 шаг подняться на нужную высоту, т.к он позволяет подниматься в два раза.

Такое решение работает за $O(n \log n)$ предподсчета и $O(1)$ на ответ.

2.5 Оффлайн решение

Обойдем граф обходом в глубину, поддерживая в стеке путь до текущей вершины. В момент захода в вершину можно ответить на каждый из запросов про нее за $O(1)$.

2.6 Идея 4-х русских

Заметим, что двоичный подъем нужен не во всех вершинах. Достаточно, чтобы в поддереве была вершина, в которой он посчитан. Тогда мы можем спуститься на нужную высоту, а потом подняться из нее.

Предподсчитаем двоичные подъемы только для вершин, поддерво котрых имеет размер хотя бы $\frac{\log n}{4}$, а все сыновья имеют размер меньше.

Тогда мы умеем отвечать на все запросы из вершин с поддервом размера хотя бы $\frac{\log n}{4}$.

Кроме того, мы умеем отвечать на запросы из маленьких поддеревьев, если они выходят за это поддерево — запоминим корень, поднимемся до него, поднимемся на 1, мы попали в вершину с большим поддеревом, поднимемся из нее.

Осталось научиться отвечать на запросы внутри поддерева. Но! Различных поддеревьев размера $\frac{\log n}{4}$ $O(\sqrt{n})$. Поэтому можно для каждого типа дерева предподсчитать все ответы на все запросы.