100. Heavy-light decomposition. Тяжёлые и лёгкие рёбра. Лемма о числе лёгких рёбер на пути между двумя вершинами. Решение задачи обновления на ребре и суммы на пути за $O(log^2(n))$ на запрос.

Heavy-light decomposition. Пулл задач: как на ДО, только в роли отрезков - пути в дереве. Числа написаны на каждом ребре.

Подвесим дерево за произвольную вершину r. Насчитаем размеры всех поддеревьев: subtree (см. предыдущие билеты на центроиды). Ребро из u в v - **тяжёлое**, если subtree[v] $\geqslant \frac{1}{2}$subtree[u]. Замечание: из каждой вершины выходит не более одного тяжёлого ребра. Вот эти тяжёлые рёбра можно склеить в пути; все остальные рёбра лёгкие.

Идея: разбили дерево на тяжёлые пути, между ними лёгкие рёбра. Тогда если мы хотим обновлять значение в ребре и находить сумму на пути между двумя вершинами, то мы хотим выстроить на тяжёлых путях деревья отрезков (котировать их как массивы, всё находить), а количество лёгких путей, которые надо будет доставить, будет немного.

Лемма (о числе лёгких рёбер на пути между двумя вершинами. На пути от u до v ($\forall u, v$ количество лёгких рёбер равно $O(log(n))$.

▲ Как устроен путь от u до v? Это путь от u до LCA и путь от LCA до v. Лёгкие рёбра, если мы проходим снизу вверх, увеличивают размер поддерева не менее, чем в два раза; отсюда на первом пути максимум логарифм лёгких рёбер; аналогичные рассуждения о пути вниз. ■

Следствие: путь от u до v пересекает $O(log(n))$ тяжёлых путей.

Тогда обновление значения в ребре либо изменяет ДО на тяжёлом пути (если ребро тяжёлое), либо ребро лёгкое и ничего не меняется; первое делается за $O(log(n))$, второе - за $O(1)$.

Сумма ищется за $O(log_2^2(n))$, так как у нас логарифм тяжёлых путей, на которых сумма ищется за логарифм.

101. Центроидная декомпозиция. Подсчёт числа объектов, обладающих заданным свойством.

Задача: найти число объектов со свойством α. Например, количество троек вершин, которые находятся на одном и том же расстоянии x друг от друга. Или количество путей, на которых сумма хорошая. Или на рёбрах написаны скобки, найти количество путей, образующих ПСП.

Решение в общем виде: Пусть C - центроид дерева T.

1.Находим его за линейное время. Подвесим дерево за центроид, все размеры не больше половины исходного. Найдём количество объектов, для которых свойство α пропадёт, если удалить вершинку С. Как именно это сделать? Зависит от задачи.

2. Если $T_1, \ldots, T_k$ - запускаемся рекурсивно от них (с удалённой вершинкой С). Таким образом, при запуске рекурсии мы минимум в два раза уменьшаем количество вершин в дереве на каждом шаге, значит, глубина рекурсии не превышает логарифм. Тогда если пункт 1 реализован за $O(n)$, то суммарная сложность - $O(nlog(n))$

102. Центроидная декомпозиция. Задача о перекрашивании синих вершин в красный цвет и поиска расстояния до ближайшей красной вершины.

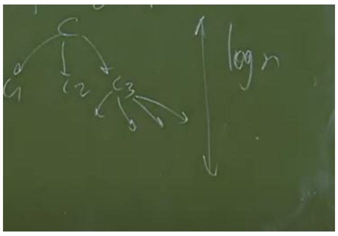

Рис. 1: Изображение дерева центроидов - 1.

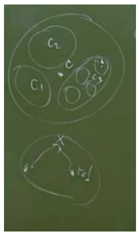

Рис. 2: Дерево центроидов 2 + поиск красной вершины.

Задача. Дано дерево. Каждая вершина либо синяя, либо красная. Дано два типа запросов:

- перекрасить синюю v в красный цвет. - для данной вершины u сообщить расстояние до ближайшей красной.

Дерево центроидов - корень дерева - исходный центроид, его дети - центроиды в поддеревьях, на которые мы разбились при рекурсии. Глубина дерева - максимум логарифм.

Запрос типа 2: искомая вершина лежит в одном из бОльших поддеревьев (наддеревьев?). Соотвественно, на произвольном шаге, где центроид x, хочется найти ближайшую к x красную вершину и сказать, что она же ближайшая к u, просто надо сначала дойти до x, потом до u. (см. рис. 2)

Доказательство: рассматрвиаем дерево, в котором они оба лежат, а дальше - не лежат, но тогда расстояние от u до красной - это расстояние до центроида + расстояние от центроида до красной.

Пусть T_v - поддерево с центроидом v. Для каждой v находим ближайшую красную в T_v: closest[v] $= min_{r \in T_V} dist(v, r)$, где r - красная. Тогда ответ на запрос 2 типа - min по всем v - предки в деревне центроидов от $dist(u, v) + closest[v]$. Первое слагаемое вычисляется за логарифм (потому что мы умеем искать LCA за логарифм), значит, всё в итоге делается за $O(log^2(n))$.

Для запроса первого типа надо обновить closest для тех вершин v, для которых r стало лежать в T_v: в точности все предки вершины r. По всем v - предкам r в дереве центроидов берём новый closest - минимум из старого и нового. Время: $O(log^2(n))$, т.к. для каждого предка за логарифм что-то считаем.