Бабушкин А.

1 Обязательные задачи

1. Построим двоичные подъёмы только на вершинах, глубина которых делится на округлённый вверх $\log n$. Теперь когда ищем lca сначала поднимаемся от v глупо до ближайшей такой вершины, от неё скачем двоичными подъёмами до самой верхней, которая не предок u, и там снова глупо поднимаемся. Каждая фаза работает за лог.

2. Если вершины в одной компоненте рёберной двусвязности, то мы не можем не пройти только по ним самим, иначе нас интересует путь по таким компонентам между ними. За dfs находим компоненты, на дереве этих компонент строим что-нибудь для поиска lca.

3.

4. dp[k][v] – min сумма последовательности из k элементов с концом в элементе v. Пересчитать dp[k][v] – взять минимум на отрезке dp[k - 1]. Можно rmq, а можно держать минимум на очереди и получить халявный O(nL).

5. a) Будем в ДД держать эйлеров обход дерева, причём для каждой вершины две копии – как будто если бы мы делали push_back в вектор когда входили в вершину и когда выходили. Тогда между первым и вторым вхождением у нас как раз поддерево. Добавить лист – это вставить две новые вершинки в правильно место в ДД.
   b) Выпишем сразу эйлеров обход для итогового дерева, запихнём его в ДО, изначально заполненное нулями, и будут запросы "сделать элемент единичкой" и "сумма на отрезке".

2 Дополнительные задачи

1.