1 Обязательная часть

1. ***Компактные двоичные подъемы***

   Для начала сделаем небольшой предподсчет дфсом - времена входа и выхода в вершины. Это нужно, чтобы потом за единицу узнавать, находятся ли сейчас вершины в одном поддереве. Теперь найдем такую недоглубину, что вершин на всех уровнях с глубиной по модулю $\log n$ равной нашей недоглубине наименьшее количество. Заметим, что вершин на этой недоглубине будет не больше $\frac{n}{\log n}$.

   Сделаем теперь двоичные подъемы в недодереве — будем видеть в нашем дереве только вершины, находящиеся на недоглубине. Так как вершин $\mathcal{O}(\frac{n}{\log n})$, то это произойдет за линию. Найдем недоLCA в таком недодереве. Заметим, что настоящий LCA не больше чем на логарифм ниже, чем недоLCA, и точно не выше. Ну поползем до него вниз. Как раз тут предподсчет и понадобится.

2. ***Нельзя не пройти***

   Через какие точки мы обязательно должны пройти? Через точки сочленения. А все точки, что находятся в одной компоненте вершинной двусвязности, всегда можно заменить друг другом, так что они нас не интересуют. Ну и давайте сожмем каждуй компоненту. В итоге получили новый граф, состоящий только из точек сочления и компонент двусвязности. Два интересных факта: эти вершины чередуются (ведь мы выделили точки сочленения отдельно) и полученный граф - дерево (если будет цикл, то из одной компоненты можно попасть в другую двумя путями).

   Итак, у нас есть дерево. На пути из a в b мы обязательно пройдем только через точки сочления на пути в этом дереве. Есть два варианта, как можно отвечать на запрос. Первый — ответ на запрос за линию — подниматься до lca(a, b) и считать количество точек сочленения. Чтобы за единицу отвечать на запрос, давайте предподсчитаем эти количества дфсом. Тогда ответ будет $ans[a] + ans[b] - 2 \cdot ans[lca(a, b)]$.

3. ***Помечающиеся вершины***

   Делаем все как всегда, но теперь поддерживаем для каждой вершины ближайшего непомеченного предка. Например, в СНМ. Теперь после того как по обычному алгоритму получили общего предка a и b делаем запрос к СНМ — get(lca).

4. ***Вспомним прошлое***

   Динамика + ФКБ. Динамика:

$$\text{база: } sum[1][pos] = mass[pos]$$
$$sum[l][pos] = min_{i \in [pos-k;i-1]} sum[l-1][i] + mass[pos]$$

   Чтобы за единицу искать минимум в конце обработки каждого уровня строим на нем ФКБ за линию. Минимумы мы как раз ищем на отрезках, так что RMQ при помощи ФКБ отвечаем за единицу. Получили динамику, которая для каждой возможной длины вплоть до L обрабатывает за единицу n элементов строки, а затем за n строит ФКБ для этого уровня. Итого $\mathcal{O}(nL)$.

5. ***Размер меняющегося дерева***

Онлайн. Поддерживаем Эйлеров обход, в котором записываем вершины при входе и выходе. Размер поддерева - разность индексов в Эйлеровом обходе. Чтобы быстро добавлять нужно ДД. Храним ссылку из вершины обычного дерева в две вершины ДД. Добавление = сплит этих вершин и вставка между ними миниДД из двух одинаковых новых вершин.

Оффлайн. Сначала считываем все запросики и строим по итоговому дереву Эйлеров обход как в онлайне. Теперь строим на нем ДО по суммам, в котором 1 соответствует вершине, что была изначально, а 0 добавленной позже. Заново бежим по запросам. При добавлении новой вершины меняем два соответствующих 0 на 1. Запрос размера дерева - сумма на отрезке.