1 Обязательная часть

Строим centroid decomposition. С этой фразы начинается каждое решение. Так что вынесем ее отдельно.

1. ***Путь с нулевым ксором***

   Перебираем все центроиды (то есть в итоге все вершины). Рассматриваем компоненту выбранного центра, запускаем в ней дфс. Будем поддерживать счетчик того, что уже наксорили по пути, изначально на нем вес x - нашего центроида. Каждый раз, когда приходим в новую вершину - делаем ксор нашего счетчика и веса этой вершины. Запоминаем каждый результат на счетчике.

   Как понять, что мы нашли путь? Первый вариант: Мы получили 0 на счетчике. То есть мы нашли путь: он начинается в рассматриваемом центроиде и заканчивается в данной вершине. И есть второй вариант. Вдруг путь не начинается в данном центроиде, а он где-то "внутри"пути. Но тогда его по центроиду можно разбить на два, которые мы обязательно обойдем в процессе дфса, но по-отдельности. И тут нет никакой проблемы, ведь мы запоминаем все счетчики. Если в какой-то момент окажется, что текущий счетчик, проксоренный с весом центроида, мы уже встречали, значит, мы нашли составной путь. Зачем ксорить с весом центра? Иначе центр будет учтен в пути дважды, а мы этого не хотим, от одного надо избавиться. А обратное к числу при ксоре - само число.

   Итак, получили условие:

   if (xored_path == 0 || (xored_path **xor** w[x]) in map) cout « "we found way!";

2. ***Число путей***

   Делаем небольшой предподсчет дфсом. Заводим два массива num[depth] и d[x][v]. Для каждого центроида запускаем дфс внутри компоненты, в d[x][v] записываем расстояние от центроида x до рассматриваемой вершины v, а в num[depth] записываем количество вершин на этой глубине дфса. То есть каждый раз, когда приходим в новую вершину такой глубины, увеличиваем соответствуютщее значение на 1, изначально там 0 хранится. Заметим, что глубина не может быть больше размера компоненты, то есть $depth \in [0, |C(x)|]$

   Посчитаем число путей от L до R для центроидов по отдельности, а потом просто просуммируем получившиеся значения: $Ans = \sum_{x=0}^{n} ans[x]$. Как мы это сделаем. Будем склеивать пути, проходящие через центр x:

   $$tmp = \sum_{i=0}^{|C(x)|} (num[i] \cdot \sum_{j=L-i}^{R-i} num[j]).$$

   Но в такой сумме мы считаем не только простые пути, но и не очень. Как от них избавиться? Вычесть! Будем работать сначала с центроидами нижних уровней, постепенно поднимаясь. Таким образом, на момент подсчета ans[x] у нас уже будут посчитаны ans[v] его детей, так что их можно будет просто взять ивычесть. Собственно, все. Все это дело будет работать за $n \log n$, если заранее посчитать префиксные суммы для той формулы наверху, чтобы ее можно было искать за линию.

3. ***Ближайшая черная вершина***

One more time предподсчет. Он нужен нам, чтобы заполнить массивчик d[v][x] - расстояние от вершины v до центроида x. Это делается обычным дфсом за $n \log n$, как всегда. Берем центроид, в его компоненте запускаем дфс, радуемся жизни. На каждом уровне в сумме не больше n вершин, уровней логарифм. Предподсчет окончен. Для решения еще понадобится хранить массив dblack[x] - расстояние от центроида до ближайшей черной в его компоненте. Изначально в нем бесконечности.

Как работать. Когда приходит запрос покрасить белую в черный - просто идем по ее "декомпозиционным"предкам вверх и обновляем для них значение dblack[][], если надо. Когда просят сказать ближайшую к вершине черную - опять ходим по ее предкам. В начале прохода заводим переменную, которая будет отвечать за расстояние до ближайшей черной на данный момент, обновляем ее по мере прохода:

relax(min_dist, dblack[x] + d[v][x]), где x - рассматриваемый сейчас предок.