1. Заведём стек, в котором кроме обычных функций есть ещё возможность обращения к элементу по индексу за O(1) (называется «вектор», но тут важно, что стек используется при dfs часто и что мы добавляем\удаляем только с конца и в середине ничего не меняем, поэтому назовём его стеком с фичей).

Будем хранить мапу, в которой по вершине и запросу можно в среднем за O(1) получить, изменить и добавить ответ.

Запустим обход в глубину от корня, используя наш стек. Когда посещается вершина v (все вершины предварительно наделены информацией о том, какие запросы им надо выдать), заполняются ячейки мапы по вершине v и всем запросам, связанным с ней. После этого продолжаем обычный DFS.

Асимптотика: O(n) из-за DFS, O(m) это суммарное количество обращений в стек и в мапу. Итого: O(n + m)

2. Пусть нам надо найти LCA вершин u и v. Заметим, что функция [LA(u, m) == LA(v, m)], зависящая от m монотонна: с ростом m функция убывает (пусть есть LCA(u, v) == LA (u, k), тогда при m <= k [LA(u, m) == LA (v, m)], а при m > k наоборот. Иначе нарушается определение LCA). Тогда будем перебирать m от -1 до max(depth(u), depth(v)) + 1 двоичным поиском (нужно найти наибольшее m, при котором [LA(u, m) == LA(v, m)] == 1). Глубина любой вершины дерева это O(n), поэтому поиск будет за O(logn). Ответом будет LA(u, m), где m – ответ в двоичном поиске.