75 Days of Code Day 38   236. Lowest Common Ancestor of a Binary Tree

Type : BST / dfs

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow a node to be a descendant of itself).”

Example 1:

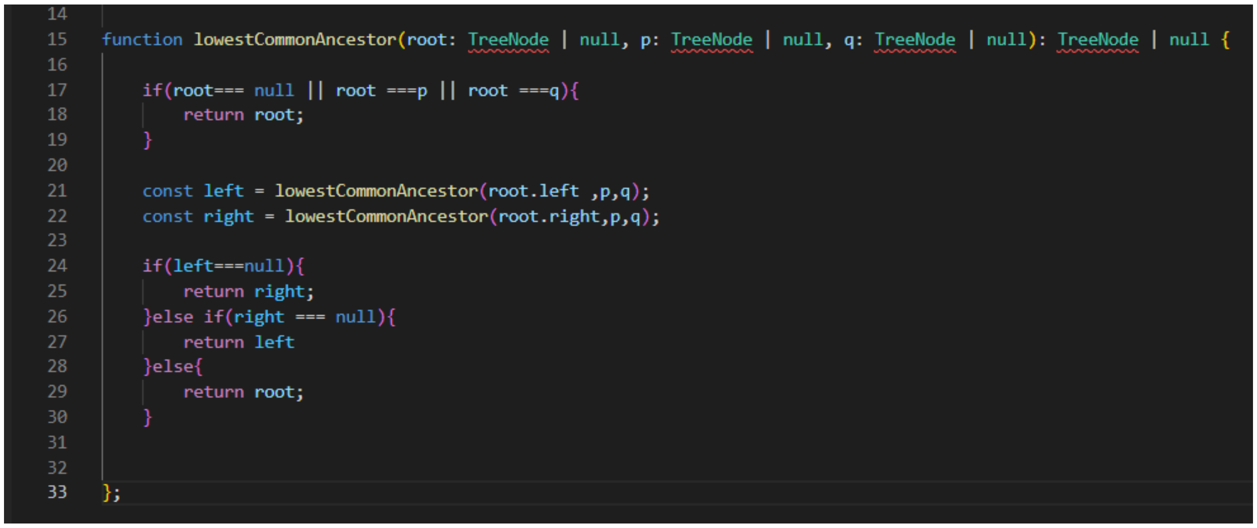
Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1
Output: 3
Explanation: The LCA of nodes 5 and 1 is 3.
Example 2:

Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 4
Output: 5
Explanation: The LCA of nodes 5 and 4 is 5, since a node can be a descendant of itself according to the LCA definition.
Example 3:

Input: root = [1,2], p = 1, q = 2
Output: 1

Solution using DFS

1. Traverse the list with recursion



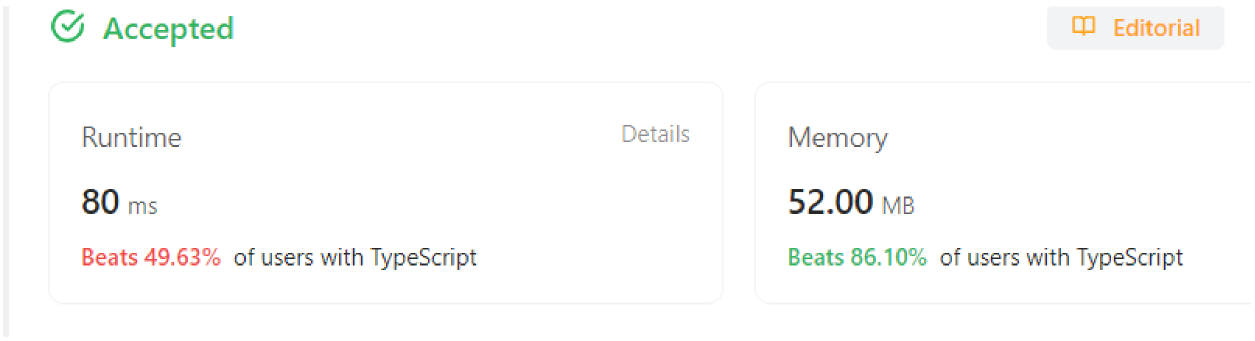
```
function lowestCommonAncestor(root: TreeNode | null, p: TreeNode | null, q: TreeNode | null): TreeNode | null {

    if(root=== null || root ===p || root ===q){
        return root;
    }

    const left = lowestCommonAncestor(root.left ,p,q);
    const right = lowestCommonAncestor(root.right,p,q);

    if(left===null){
        return right;
    }else if(right === null){
        return left
    }else{
        return root;
    }


};
```



Accepted

Runtime

80 ms

Beats 49.63% of users with TypeScript

Memory

52.00 MB

Beats 86.10% of users with TypeScript

Shubham Agrahari

Shubham Agrahari