236. Lowest Common Ancestor of a Binary Tree

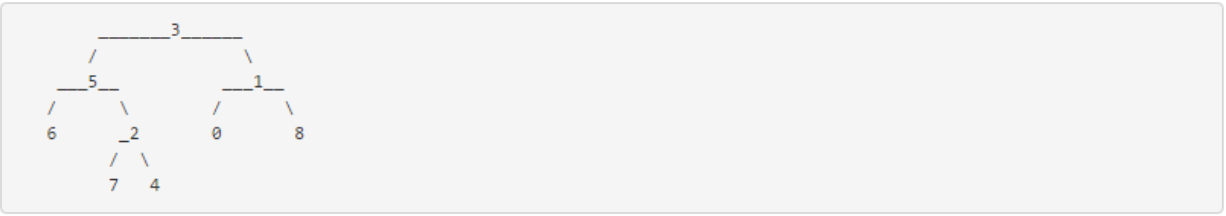
Description | Hints | Submissions | Discuss | Solution

Pick One

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**)."

Given the following binary tree: root = [3,5,1,6,2,0,8,null,null,7,4]



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 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.
```

Note:

- All of the nodes' values will be unique.
- p and q are different and both values will exist in the binary tree.

```java
public class L236 {
    public class TreeNode {
        int val;
        TreeNode left;
        TreeNode right;

        TreeNode(int x) {
            val = x;
        }
    }
/*
 * 这个题目有三种情况，第一个是两个及诶单是在公共祖先的左右两侧
 * 第二种情况的话是都在树的左侧，第三种情况是都在树的右侧。
 */
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
           if(root == null || root == p || root == q) {
               return root;
           }

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

           if(left != null && right != null) {
               return root;
           }else {
               return left != null ? left : right;
           }
    }
}
```