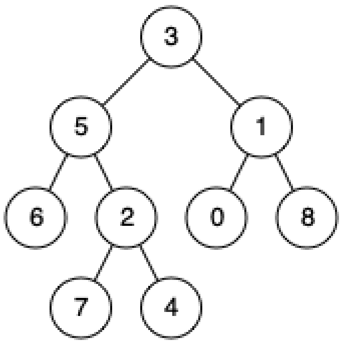
1644. Lowest Common Ancestor of a Binary Tree II

Given the **root** of a binary tree, return *the lowest common ancestor (LCA) of two given nodes, p and q*. If either node p or q **does not exist** in the tree, return **None**. All values of the nodes in the tree are **unique**.



Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1

Output: 3

Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 10

Output: null

Explanation: Node 10 does not exist in the tree, so return null.

```
class Solution:
    """
    @param: root: The root of the binary tree.
    @param: A: A TreeNode
    @param: B: A TreeNode
    @return: Return the LCA of the two nodes.
    """
    def lowestCommonAncestor3(self, root, A, B):
        # write your code here
        boolA = self.find(root,A.val)
        boolB = self.find(root,B.val)
        if boolA and boolB:
            return self.helper(root,A,B)
        else:
            return None
```

```
def find(self,root,val):
    if root is None:
        return False
    if root.val==val:
        return True
    return self.find(root.left,val) or self.find(root.right,val)

def helper(self,root,A,B):
    if root is None:
        return None
    if root.val==A.val or root.val==B.val:
        return root
    left = self.helper(root.left,A,B)
    right = self.helper(root.right,A,B)

    if left!=None and right!=None:
        return root
    else:
        return right if left is None else left
```