

Problem 1644: Lowest Common Ancestor of a Binary Tree II

Problem Information

Difficulty: Medium

Acceptance Rate: 69.22%

Paid Only: Yes

Tags: Tree, Depth-First Search, Binary Tree

Problem Description

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 `null`. All values of the nodes in the tree are **unique**.

According to the **[definition of LCA on Wikipedia](https://en.wikipedia.org/wiki/Lowest_common_ancestor)**: "The lowest common ancestor of two nodes `p` and `q` in a binary tree `T` is the lowest node that has both `p` and `q` as **descendants** (where we allow **a node to be a descendant of itself**). A **descendant** of a node `x` is a node `y` that is on the path from node `x` to some leaf node.

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. A node can be a descendant of itself according to the definition of LCA.

****Example 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.

****Constraints:****

* The number of nodes in the tree is in the range `[1, 104]`. * `-109 <= Node.val <= 109` * All `Node.val` are ****unique****. * `p != q`

****Follow up:**** Can you find the LCA traversing the tree, without checking nodes existence?

Code Snippets

C++:

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {

    }
};
```

Java:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
```

```

* TreeNode left;
* TreeNode right;
* TreeNode(int x) { val = x; }
* }
*/
class Solution {
public:
    TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {

    }
}

```

Python3:

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    def lowestCommonAncestor(self, root: 'TreeNode', p: 'TreeNode', q:
'TreeNode') -> 'TreeNode':

```