

Problem 1650: Lowest Common Ancestor of a Binary Tree III

Problem Information

Difficulty: Medium

Acceptance Rate: 82.84%

Paid Only: Yes

Tags: Hash Table, Two Pointers, Tree, Binary Tree

Problem Description

Given two nodes of a binary tree `p` and `q`, return their lowest common ancestor (LCA).

Each node will have a reference to its parent node. The definition for `Node` is below:

```
class Node { public int val; public Node left; public Node right; public Node parent; }
```

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 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)."

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

****Constraints:****

* The number of nodes in the tree is in the range `[2, 105]`. * `-109 <= Node.val <= 109` * All `Node.val` are ****unique****. * `p != q` * `p` and `q` exist in the tree.

Code Snippets

C++:

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    Node* left;
    Node* right;
    Node* parent;
};
*/

class Solution {
public:
    Node* lowestCommonAncestor(Node* p, Node * q) {

    }
};
```

Java:

```
/*
// Definition for a Node.
class Node {
public int val;
public Node left;
```

```

public Node right;
public Node parent;
};
*/

class Solution {
public Node lowestCommonAncestor(Node p, Node q) {

}

}

```

Python3:

```

"""
# Definition for a Node.
class Node:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None
        self.parent = None
"""

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

```