

Problem 617: Merge Two Binary Trees

Problem Information

Difficulty: Easy

Acceptance Rate: 78.89%

Paid Only: No

Tags: Tree, Depth-First Search, Breadth-First Search, Binary Tree

Problem Description

You are given two binary trees `root1` and `root2`.

Imagine that when you put one of them to cover the other, some nodes of the two trees are overlapped while the others are not. You need to merge the two trees into a new binary tree. The merge rule is that if two nodes overlap, then sum node values up as the new value of the merged node. Otherwise, the NOT null node will be used as the node of the new tree.

Return _the merged tree_.

****Note:**** The merging process must start from the root nodes of both trees.

****Example 1:****

****Input:**** root1 = [1,3,2,5], root2 = [2,1,3,null,4,null,7] ****Output:**** [3,4,5,5,4,null,7]

****Example 2:****

****Input:**** root1 = [1], root2 = [1,2] ****Output:**** [2,2]

****Constraints:****

* The number of nodes in both trees is in the range `[0, 2000]`. * `-104 <= Node.val <= 104`

Code Snippets

C++:

```
/*
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
 *     right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
        }
    };
}
```

Java:

```
/*
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
```

```
public TreeNode mergeTrees(TreeNode root1, TreeNode root2) {  
    if (root1 == null) return root2;  
    if (root2 == null) return root1;  
    root1.val = root1.val + root2.val;  
    root1.left = mergeTrees(root1.left, root2.left);  
    root1.right = mergeTrees(root1.right, root2.right);  
    return root1;  
}
```

Python3:

```
# Definition for a binary tree node.  
# class TreeNode:  
#     def __init__(self, val=0, left=None, right=None):  
#         self.val = val  
#         self.left = left  
#         self.right = right  
class Solution:  
    def mergeTrees(self, root1: Optional[TreeNode], root2: Optional[TreeNode]) ->  
        Optional[TreeNode]:  
        if not root1:  
            return root2  
        if not root2:  
            return root1  
        root1.val += root2.val  
        root1.left = self.mergeTrees(root1.left, root2.left)  
        root1.right = self.mergeTrees(root1.right, root2.right)  
        return root1
```