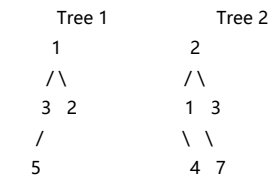
617 Merge Two Binary Trees

2018年4月16日 20:49

Given two binary trees and 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 them 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 new tree.

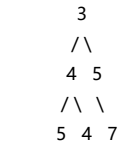
Example 1:

Input:



Output:

Merged tree:



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

来自 <https://leetcode.com/problems/merge-two-binary-trees/description/>
给定两个二叉树，想象当你将它们中的一个覆盖到另一个上时，两个二叉树的一些节点便会重叠。
你需要将他们合并为一个新的二叉树。合并的规则是如果两个节点重叠，那么将他们的值相加作为节点合并后的新值，否则**不为** NULL 的节点将直接作为新二叉树的节点。
注意: 合并必须从两个树的根节点开始。

Solution for Python3:

```python
class Solution1:
    def mergeTrees(self, t1, t2):
        """
        :type t1: TreeNode
        :type t2: TreeNode
        :rtype: TreeNode
        """
        if not t2:
           return t1
        if not t1:
           return t2
        t1.val += t2.val
        t1.left = self.mergeTrees(t1.left, t2.left)
        t1.right = self.mergeTrees(t1.right, t2.right)
        return t1

class Solution2:
    def mergeTrees(self, t1, t2):
        """
        :type t1: TreeNode
        :type t2: TreeNode
        :rtype: TreeNode
        """
        if t1 and t2:
           t1.val += t2.val
           t1.left = self.mergeTrees(t1.left, t2.left)
           t1.right = self.mergeTrees(t1.right, t2.right)
           return t1
        else:
```

```
            return t1 or t2

class Solution3:
    def mergeTrees(self, t1, t2):
        """
        :type t1: TreeNode
        :type t2: TreeNode
        :rtype: TreeNode
        """
        if not t1:
            return t2
        deq = Collections.deque([t1,t2])
        while deq:
            t = deq.pop()
            if not t[0] or not t[1]:
                continue
            t[0].val += t[1].val
            if not t[0].left:
                t[0].left = t[1].left
            else:
                deq.append([t[0].left, t[1].left])
            if not t[0].right:
                t[0].right = t[1].right

```

Solution for 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 Solution1 {
public:
    TreeNode* mergeTrees(TreeNode* t1, TreeNode* t2) {
        if (!t2)
            return t1;
        if (!t1)
            return t2;
        t1->val += t2->val;
        t1->left = mergeTrees(t1->left, t2->left);
        t1->right = mergeTrees(t1->right, t2->right);
        return t1;
    }
};

class Solution2 {
public:
    TreeNode* mergeTrees(TreeNode* t1, TreeNode* t2) {
        if (t1 && t2) {
            t1->val += t2->val;
            t1->left = mergeTrees(t1->left, t2->left);
            t1->right = mergeTrees(t1->right, t2->right);
            return t1;
        } else {
            return t1 ? t1 : t2;
```

```
34          }
35      }
36  };
37
38  class Solution3 {
39  public:
40      TreeNode* mergeTrees(TreeNode* t1, TreeNode* t2) {
41          if (!t1)
42              return t2;
43          stack<pair<TreeNode*, TreeNode*>> stk;
44          stk.push(pair<TreeNode*, TreeNode*>(t1, t2));
45          while (!stk.empty()) {
46              pair<TreeNode*,TreeNode*> t = stk.top();
47              stk.pop();
48              if(!t.first || !t.second)
49                  continue;
50              t.first->val += t.second->val;
51              if (!t.first->left)
52                  t.first->left = t.second->left;
53              else
54                  stk.push(pair<TreeNode*, TreeNode*>(t.first->left,t.second->left));
55              if (!t.first->right)
56                  t.first->right = t.second->right;
57              else
58                  stk.push(pair<TreeNode*, TreeNode*>(t.first->right, t.second->right));
59          }
60          return t1;
61      }
62  };
```