```
/*
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。

示例 1:

输入:
    Tree 1                     Tree 2
          1                         2
         / \                       / \
        3   2                     1   3
       /                           \   \
      5                             4   7
输出:
合并后的树:
         3
        / \
       4   5
      / \   \
     5   4   7

注意: 合并必须从两个树的根节点开始。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/merge-two-binary-trees
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 方法一:(不破坏传入的两个二叉树)
  - 同时递归两个二叉树，若两个二叉树的同一个位置都NULL，则该节点也为NULL；若两个二叉树的同一个位置都不为NULL，则把对应值相加赋值给新的节点，若只有一个不为NULL，就直接复制其中一个即可.

方法一:C++_不破坏原二叉树

```
/**
 * 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
{
    private:
```

```
        TreeNode* __mergeTrees(TreeNode* t1, TreeNode* t2)
        {

            TreeNode*   temp              =   NULL;
            TreeNode*   temp_t1_left      =   NULL;
            TreeNode*   temp_t1_right     =   NULL;
            TreeNode*   temp_t2_left      =   NULL;
            TreeNode*   temp_t2_right     =   NULL;

            if(   (t1==NULL)
                &&(t2==NULL)
            )
            {
                return NULL;
            }

            temp = new struct TreeNode(0);

            if(t1!=NULL)
            {
                temp -> val     +=  t1->val;
                temp_t1_left    =   t1 -> left;
                temp_t1_right   =   t1 -> right;

            }

            if(t2!=NULL)
            {
                temp -> val += t2->val;
                temp_t2_left    =   t2 -> left;
                temp_t2_right   =   t2 -> right;

            }

            temp -> left  = __mergeTrees(temp_t1_left,temp_t2_left);
            temp -> right = __mergeTrees(temp_t1_right,temp_t2_right);
            return temp;
        }

    public:
        TreeNode* mergeTrees(TreeNode* t1, TreeNode* t2)
        {
            return __mergeTrees(t1,t2);
        }
};

/*
执行结果:
通过
显示详情
执行用时 :92 ms, 在所有 C++ 提交中击败了16.64% 的用户
内存消耗 :22.4 MB, 在所有 C++ 提交中击败了15.45%的用户
*/
```

AlimyBreak
2019.08.30