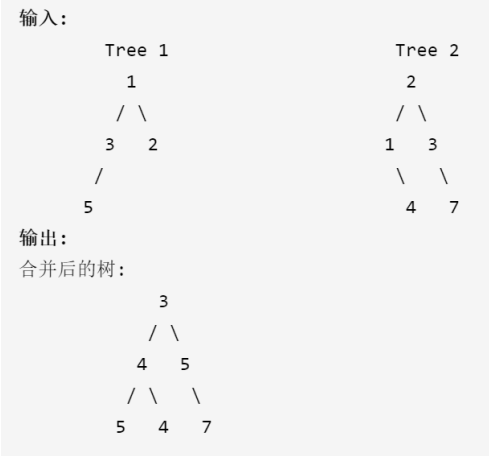
617 合并二叉树

Label：二叉树
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。



- 递归

```
class Solution {
    public TreeNode mergeTrees(TreeNode root1, TreeNode root2) {
        TreeNode dummy = new TreeNode();
        if (root1 == null && root2 == null) {
            return null;
        }else if (root1 != null && root2 == null) {
            return new TreeNode(root1.val,root1.left,root1.right);
        }else if (root2 != null && root1 == null) {
            return new TreeNode(root2.val,root2.left,root2.right);
        }else {  // root1 != null && root2 != null
            dummy.val = root1.val + root2.val;
            dummy.left = mergeTrees(root1.left,root2.left);
            dummy.right = mergeTrees(root1.right,root2.right);
            return dummy;
        }
    }
}
```

- 先序遍历（深度优先）

```
class Solution {
    public TreeNode mergeTrees(TreeNode t1, TreeNode t2) {
        if (t1 == null) return t2;
        if (t2 == null) return t1;
        TreeNode merged = new TreeNode(t1.val + t2.val);   // 先序遍历
        merged.left = mergeTrees(t1.left, t2.left);
        merged.right = mergeTrees(t1.right, t2.right);
        return merged;
    }
}
```

- 广度优先

```
class Solution {
    public TreeNode mergeTrees(TreeNode t1, TreeNode t2) {
         // 特殊判断
        if (t1 == null) return t2;
        if (t2 == null) return t1;

        TreeNode merged = new TreeNode(t1.val + t2.val);

        Queue<TreeNode> queue = new LinkedList<TreeNode>();
        Queue<TreeNode> queue1 = new LinkedList<TreeNode>(); // 存储t1
        Queue<TreeNode> queue2 = new LinkedList<TreeNode>(); // 存储t2

        queue.offer(merged);
        queue1.offer(t1);
        queue2.offer(t2);

        while (!queue1.isEmpty() && !queue2.isEmpty()) {
            TreeNode node = queue.poll(), node1 = queue1.poll(), node2 =
queue2.poll();
            TreeNode left1 = node1.left, left2 = node2.left, right1 =
node1.right, right2 = node2.right;

            // left
            if (left1 != null || left2 != null) {
                if (left1 != null && left2 != null) {
                    TreeNode left = new TreeNode(left1.val + left2.val);
                    node.left = left;
                    queue.offer(left);
                    queue1.offer(left1);
                    queue2.offer(left2);
                } else if (left1 != null) {  // left2 == null，后续直接移接就行
                    node.left = left1;
                } else if (left2 != null) {   // left1 == null，后续直接移接就行
                    node.left = left2;
                }
            }
            // right
            if (right1 != null || right2 != null) {
                if (right1 != null && right2 != null) {
                    TreeNode right = new TreeNode(right1.val + right2.val);
                    node.right = right;
                    queue.offer(right);
                    queue1.offer(right1);
                    queue2.offer(right2);
                } else if (right1 != null) {
                    node.right = right1;
                } else {
                    node.right = right2;
                }
            }
        }
        return merged;
    }
}
```