题目 树的子结构

考点 代码的鲁棒性 热点指数 64057 通过率 23.05%

具体题目

输入两棵二叉树A，B，判断B是不是A的子结构。（ps：我们约定空树不是任意一个树的子结构）

结题步骤

```
public class Solution {
    public static boolean HasSubtree(TreeNode root1, TreeNode root2) {
        boolean result = false;
        //当Tree1和Tree2都不为零的时候，才进行比较。否则直接返回false
        if (root2 != null && root1 != null) {
            //如果找到了对应Tree2的根节点的点
            if(root1.val == root2.val){
                //以这个根节点为为起点判断是否包含Tree2
                result = doesTree1HaveTree2(root1,root2);
            }
            //如果找不到，那么就再去root的左儿子当作起点，去判断时候包含Tree2
            if (!result) {
                result = HasSubtree(root1.left,root2);
            }


            //如果还找不到，那么就再去root的右儿子当作起点，去判断时候包含Tree2
            if (!result) {
                result = HasSubtree(root1.right,root2);
               }
            }
            //返回结果
        return result;
    }
    public static boolean doesTree1HaveTree2(TreeNode node1, TreeNode node2) {
        //如果Tree2已经遍历完了都能对应的上，返回true
        if (node2 == null) {
            return true;
        }
        //如果Tree2还没有遍历完，Tree1却遍历完了。返回false
        if (node1 == null) {
            return false;
        }
        //如果其中有一个点没有对应上，返回false
        if (node1.val != node2.val) {
                return false;
        }

        //如果根节点对应的上，那么就分别去子节点里面匹配
        return doesTree1HaveTree2(node1.left,node2.left) &&
doesTree1HaveTree2(node1.right,node2.right);
    }

/*思路：参考剑指offer
1、首先设置标志位result = false，因为一旦匹配成功result就设为true，
剩下的代码不会执行，如果匹配不成功，默认返回false
2、递归思想，如果根节点相同则递归调用DoesTree1HaveTree2（），
如果根节点不相同，则判断tree1的左子树和tree2是否相同，
再判断右子树和tree2是否相同
```

```
3、注意null的条件，HasSubTree中，如果两棵树都不为空才进行判断，
DoesTree1HasTree2中，如果Tree2为空，则说明第二棵树遍历完了，即匹配成功，
tree1为空有两种情况（1）如果tree1为空&&tree2不为空说明不匹配，
（2）如果tree1为空，tree2为空，说明匹配。
*/
public class Solution {
    public boolean HasSubtree(TreeNode root1,TreeNode root2) {
        boolean result = false;
            if(root1 != null && root2 != null){
                if(root1.val == root2.val){
                    result = DoesTree1HaveTree2(root1,root2);
                }
                if(!result){result = HasSubtree(root1.left, root2);}
                if(!result){result = HasSubtree(root1.right, root2);}
            }
            return result;
    }
    public boolean DoesTree1HaveTree2(TreeNode root1,TreeNode root2){
            if(root1 == null && root2 != null) return false;
            if(root2 == null) return true;
            if(root1.val != root2.val) return false;
            return DoesTree1HaveTree2(root1.left, root2.left) &&
DoesTree1HaveTree2(root1.right, root2.right);
        }
}
```

其中需要注意的是：

1. 测试用例如果pRoot2为空的话，返回的false而不是我们认为的空树应该是所有树的子树
2. 再判断是否子树的过程中，应该先判断pRoot2是否为空，为空则表明子树的所有节点都比较完了，应该是子树返回True
3. 要养成一个习惯，对任何一个树节点进行访问时，一定要提前检测该节点是否为空