树的子结构\_剑指Offer\_26

# 树的子结构\_剑指Offer\_26

## 题目描述

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*剑指Offer26\_树的子结构\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* 问题描述：

\* 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）

## 思路分析

\* 思路分析： 根据先序遍历的顺序，首先判断根节点是否与子树相同，然后左子树，最后右子树。

\* 具体判断过程：若根节点不相同，直接返回false；否则，只有左子树和右子树都相同才返回true。

## Java代码

public boolean **HasSubtree**(TreeNode root1,TreeNode root2) {

**if(root2 == null) return false;//**默认空树不是任意一个树的子结构

if(root1 == null) return false;

boolean result = false;

if(root1.val == root2.val)//若根节点值相同，则判断左子树和右子树是否包含root2的左子树和右子树

result = **doesTreeHasTree2(root1.left,root2.left)&&doesTreeHasTree2(root1.right,root2.right);**

if(!result)//若没有找到，则递归判断左子树中是否包含root2子树

result = HasSubtree(root1.left,root2);

if(!result)//若还是没有找到，则递归判断右子树中是否包含root2子树

result = HasSubtree(root1.right,root2);

return result;

}

/\*\*

\* 判断root1中是否包含root2

\*/

public boolean doesTreeHasTree2(TreeNode root1,TreeNode root2){

**if(root2 == null) return true;//**这个必须返回true，递归结束的标识；与约定有区别

if(root1 == null) return false;

if(root1.val != root2.val) return false;

return **doesTreeHasTree2(root1.left,root2.left)&&doesTreeHasTree2(root1.right,root2.right);**

}