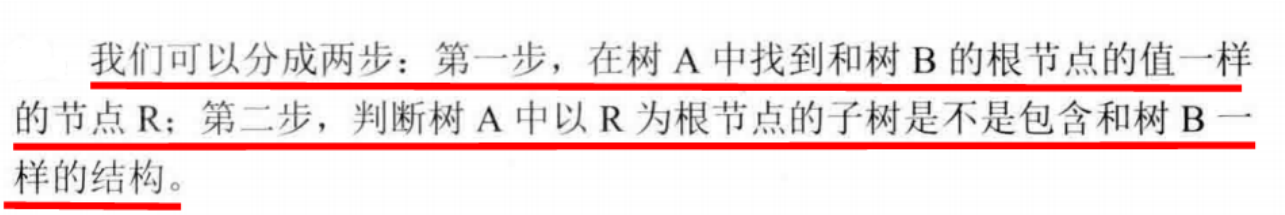
面试题26：树的子结构

题目描述

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）

解题思路



C++

```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};*/
class Solution {
public:
```

```
12      //第一步在树A中查找与根节点的值一样的节点，树的遍历，递归
13      bool HasSubtree(TreeNode* pRoot1, TreeNode* pRoot2)
14      {
15          bool result = false;
16          //判断树A和树B是否为空树
17          if(pRoot1 != nullptr && pRoot2 != nullptr){
18              //判断树A和树B值是否相等
19              if(pRoot1->val == pRoot2->val)
20              result = DoesTree1HasTree2(pRoot1, pRoot2);
21              //树A的左子树与树B比较
22              if(!result)
23                  result = HasSubtree(pRoot1->left, pRoot2);
24              //树A的右子树与树B比较
25              if(!result)
26                  result = HasSubtree(pRoot1->right, pRoot2);
27          }
28          return result;
29      }
30  private:
31      bool DoesTree1HasTree2(TreeNode* pRoot1, TreeNode* pRoot2){
32          //如果树B为空
```

```
        if(pRoot2 == nullptr)
            return true;
        //如果树A为空
        if(pRoot1 == nullptr)
            return false;
        //如果树A和树B的子树不相等
        if(pRoot1->val != pRoot2->val)
            return false;
        //返回树A和树B的左右子树相同
        return DoesTree1HasTree2(pRoot1->left,
pRoot2->left) && DoesTree1HasTree2(pRoot1-
>right, pRoot2->right);
    }
};
```

Java

```
/**
public class TreeNode {
    int val = 0;
    TreeNode left = null;
    TreeNode right = null;

    public TreeNode(int val) {
        this.val = val;

    }
```

```
11
12  }
13  */
14  public class Solution {
15      public boolean HasSubtree(TreeNode
    root1,TreeNode root2) {
16          boolean result = false;
17          //判断树A和树B是否为空树
18          if(root1 != null && root2 != null){
19              //判断树A和树B值是否相等
20              if(root1.val == root2.val)
21              result = DoesTree1HaveTree2(root1,
    root2);
22              //树A的左子树与树B比较
23              if(!result)
24                  result = HasSubtree(root1.left,
    root2);
25              //树A的右子树与树B比较
26              if(!result)
27                  result = HasSubtree(root1.right,
    root2);
28          }
29          return result;
30      }
31      public boolean DoesTree1HaveTree2(TreeNode
    root1, TreeNode root2){
32          //如果树B为空
33          if(root1 == null && root2 != null)
```

```
            return false;
        //如果树A为空
        if(root2 == null)
            return true;
        //如果树A和树B的子树不相等
        if(root1.val != root2.val)
            return false;
        //返回树A和树B的左右子树相同
        return DoesTree1HaveTree2(root1.left,
root2.left) && DoesTree1HaveTree2(root1.right,
root2.right);
    }
}
```