## 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：

1. 先判断两树是否为空，不空的条件下进行判断
2. 先判断根节点是否相同，如果相同在分别左右子树是否相同，（通过遍历两棵树进行比较，定义一个递归函数即可）
3. 如果不同，则分别判断A树的左节点和右节点是否与B的根节点相同，然后重复上述过程
4. 判断的递归函数：
5. 若A的某节点和B的根节点相同，进行判断递归函数的调用。
6. 停止条件：遍历过程中，B的节点为空，说明是子树

A为空，说明不是子树， AB节点值不同则说明不是子树

## 算法描述：

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};\*/

class Solution {

public:

bool HasSubtree(TreeNode\* pRoot1, TreeNode\* pRoot2)

{

bool result = false;

if (pRoot1 != NULL && pRoot2 != NULL)

{

if (pRoot1->val == pRoot2->val)

result = doesTree1HasTree2(pRoot1,pRoot2);

if(!result)

result = HasSubtree(pRoot1->left,pRoot2);

if(!result)

result = HasSubtree(pRoot1->right,pRoot2);

}

return result;

}

bool doesTree1HasTree2(TreeNode \* pRoot1, TreeNode\* pRoot2)

{

if (pRoot2 == NULL)

return true;

if (pRoot1 == NULL)

return false;

if (pRoot1->val != pRoot2->val)

return false;

return doesTree1HasTree2(pRoot1->left,pRoot2->left) && doesTree1HasTree2(pRoot1->right, pRoot2->right);

}

};