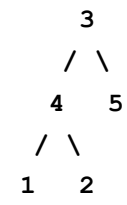
树的子结构

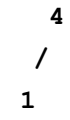
输入两棵二叉树 A 和 B，判断 B 是不是 A 的子结构。(约定空树不是任意一个树的子结构)

B 是 A 的子结构， 即 A 中有出现和 B 相同的结构和节点值。

例如:
给定的树 A:



给定的树 B：



返回 true，因为 B 与 A 的一个子树拥有相同的结构和节点值。

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    bool isSubStructure(TreeNode* A, TreeNode* B) {
        if(A==nullptr||B==nullptr) return false;//二者有一个为空则说明匹配失败
        bool res=false;//存储判断结构 初始为 false
        if(A->val==B->val)
        {
            res=isAhasB(A,B);
        }
        if(!res)//如果匹配不到子树结构 则先向左
            res=isSubStructure(A->left, B);
        if(!res)//还是匹配不到 则向右找
            res=isSubStructure(A->right,B);
        return res;
    }
    bool isAhasB(TreeNode* A, TreeNode* B)
    {
```

```
        if(B==nullptr)//说明 B 已经遍历结束  B 一定是 A 的子数
            return true;
        if(A==nullptr)//A 率先遍历结束 而 B 还没有遍历结束 说明 A 中不包含 B
            return false;
        if(A->val!=B->val)
            return false;
        return isAhasB(A->left, B->left)&&isAhasB(A->right, B->right);
    }
};
```