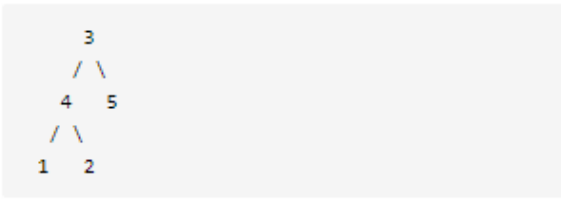
另一个树的子树

给定两个非空二叉树 **s** 和 **t**，检验 **s** 中是否包含和 **t** 具有相同结构和节点值的子树。**s** 的一个子树包括 **s** 的一个节点和这个节点的所有子孙。**s** 也可以看做它自身的一棵子树。

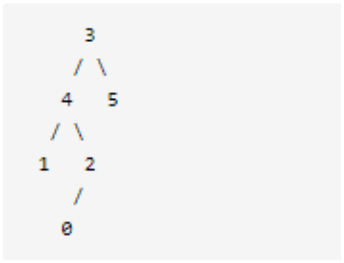
示例 1:
给定的树 s:



给定的树 t:

返回 **true**，因为 t 与 s 的一个子树拥有相同的结构和节点值。

示例 2:
给定的树 s:



给定的树 t:

返回 **false**。

解题思路:

借助相同树的判断，判断是否存在相同子树

需要注意代码最后一行的返回值,采用或逻辑并行,即使当前节点不相同可以找左子树或右子树判断是否为相同树，或逻辑十分关键

同时需要判断s是否为空,如果s为空,且isSameTree(s,t)为假,则后续的执行将出现bug,当前节点为空，则无法找到当前的左右节点，所以s为空，t不为空 返回false

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
bool isSameTree(struct TreeNode* p, struct TreeNode* q){
    if(p==NULL&&q==NULL)//同时为空，返回 true;
        return true;
    if(p==NULL||q==NULL)//不同时为空，返回 false
        return false;
    return p->val==q->val
    && isSameTree(p->left,q->left)
    && isSameTree(p->right,q->right);
}

bool isSubtree(struct TreeNode* s, struct TreeNode* t){
```

```
    if(t==NULL)
        return true;
    if(s==NULL)
        return false;
    return isSameTree(s,t)||isSubtree(s->left,t)||isSubtree(s->right,t);
}
```