user: sakura

第26题——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:

3 /
4 5 /
1 2 给定的树 B:

4 / 1

> 返回 true，因为 B 与 A 的一个子树拥有相同的结构和节点值。

示例 1:

```
输入：A = [1,2,3], B = [3,1]
```

```
输出：false
```

示例 2:

```
输入：A = [3,4,5,1,2], B = [4,1]
```

```
输出：true
```

限制:

0 <= 节点个数 <= 10000

解题思路:

首先我们需要找到与B根节点匹配的结点

然后我们对该节点和B进行匹配(这里使用一个函数来递归)

```
/**
 * 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==NULL||B==NULL){
            return false;
        }
        if(dfs(A,B)){
            return true;
        }
        //递归左子树和右子树
        return isSubStructure(A->left, B) || isSubStructure(A->right, B);
    }
    bool dfs(TreeNode* A, TreeNode* B)
    {
        //如果B为空,表示已经遍历完了,返回true
        if(B==NULL)
            return true;
        //A结点为空或者值不相等,返回false
        if(A==NULL || A->val!=B->val)
            return false;
        //递归左子树和右子树
        return dfs(A->left,B->left)&&dfs(A->right,B->right);
    }
};
```