剑指 Offer 28. 对称的二叉树

题目描述

请实现一个函数，用来判断一棵二叉树是不是对称的。如果一棵二叉树和它的镜像一样，那么它是对称的。

示例 1：

```
输入：root = [1,2,2,3,4,4,3]
输出：true
```

示例 2：

```
输入：root = [1,2,2,null,3,null,3]
输出：false
```

解题思路

- 二叉树是否对称，要比较小的是根节点的左子树和右子树是不是相互翻转
- 左子树的遍历顺序是左-右-根
- 右子树的遍历顺序是右-左-根
- 利用递归法解决
    1. 确定递归参数和返回值

       比较根节点两个子树是否相互翻转，则参数为两个，一个左子树，一个右子树;
    2. 确定终止条件

       左为空，右不为空，不对称，则返回false;

       左不为空，右为空，不对称，则返回false;

       左为空，右为空，对称，则返回true;

       左右都不为空，比较节点的数值，不相同就返回false;
    3. 确定单层递归

       比较二叉树外侧是否对称(左节点的左孩子，右节点的右孩子);

       比较二叉树内侧是否对称(左节点的右孩子，右节点的左孩子);

       如果都对称返回true，有一侧不对成返回false;

代码实现

```
class Solution {
public:
    bool compare(TreeNode* left,TreeNode* right)
    {
        if(left==NULL&&right!=NULL)
        {
            return false;
```

```
        }
        else if(left!=NULL&&right==NULL)
        {
            return false;
        }
        else if(left==NULL&&right==NULL)
        {
            return true;
        }
        else if(left->val!=right->val)
        {
            return false;
        }
        bool outside=compare(left->left,right->right);
        bool inside=compare(left->right,right->left);
        bool res=outside && inside;
        return res;
    }
    bool isSymmetric(TreeNode* root) {
        if(root==NULL)
        {
            return true;
        }
        return compare(root->left,root->right);
    }
};
```