关于二叉树轴对称问题

2022年8月15日 15:29

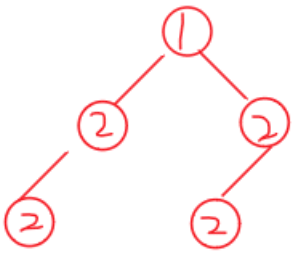
无法通过中序遍历+遍历结果轴对称解决

错误代码：

```
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        deque<int> ans;
        ans=this->inorderTraversal(root);
        if(!ans.size()%2){
            return false;
        }
        while(ans.size()!=1){
            if(ans.front()!=ans.back()){
                return false;
            }
            ans.pop_back();
            ans.pop_front();
        }
        return true;
    }
```

原因是当以下类型树出现时，无法识别

遍历结果22122关于轴对称，但树并不是轴对称的



使用递归解决该问题

```cpp
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        if(!root) return true;
        return judge(root->left,root->right);
    }
    bool judge(TreeNode* rl,TreeNode* rr){
        if(!rl&&!rr)
            return true;
        if(!rl||!rr)
            return false;
        return rl->val==rr->val&&judge(rl->left,rr->right)&&judge(rl->right,rr->left);
    }
};
```

一个很棒的解释

递归的难点在于：找到可以递归的点 为什么很多人觉得递归一看就会，一写就废。 或者说是自己写无法写出来，关键就是你对递归理解的深不深。

对于此题： 递归的点怎么找？从拿到题的第一时间开始，思路如下：

1.怎么判断一棵树是不是对称二叉树？ 答案：如果所给根节点，为空， 那么是对称。如果不为空的话，当他的左子树与右子树对称时， 他对称

2.那么怎么知道左子树与右子树对不对称呢？在这我直接叫为左树和右树 答案：如果左树的左孩子与右树的右孩子对称，左树的右孩子与右树的左孩子对称，那么这个左树和右树就对称。

仔细读这句话，是不是有点绕？怎么感觉有一个功能A我想实现，但我去实现A的时候又要用到A实现后的功能呢？

当你思考到这里的时候，递归点已经出现了： 递归点：我在尝试判断左树与右树对称的条件时，发现其跟两树的孩子的对称情况有关系。

想到这里，你不必有太多疑问，上手去按思路写代码： 函数A（左树，右树）功能是返回是否对称

def 函数A（左树，右树）： 左树节点值等于右树节点值 且 函数A（左树的左子树，右树的右子树），函数A（左树的右子树，右树的左子树）均为真 才返回真

实现完毕。。。

写着写着。。。你就发现你写出来了。。。。。。