101. Symmetric Tree

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

For example, this binary tree is symmetric:

1

/ \

2 2

/ \ / \

3 4 4 3

But the following is not:

1

/ \

2 2

\ \

3 3

**Note:**  
Bonus points if you could solve it both recursively and iteratively.

**我的版本 中序遍历看得到的序列是不是回文序列 但不能完全覆盖**

public class Solution {

public boolean isSymmetric(TreeNode root) {

if (root == null||root.left == null && root.right == null) return true;

else if (root.left != null && root.right != null&&root.left.val == root.right.val) {

ArrayList<Integer> res = (ArrayList<Integer>) inorderTraversal(root);

int flag = 0;

for (int i = 0; i < res.size()/2;i++){

if (res.get(i) != res.get(res.size()-i-1))

{

flag = 1;

break;

}

}

if (flag == 1) return false;

else return true;

}

else return false;

}

public List<Integer> inorderTraversal(TreeNode root) {

if (root.left == null && root.right == null) {

List<Integer> res = new ArrayList<Integer>();

res.add(root.val);

return res;

}

else if (root.left != null && root.right != null) {

List<Integer> res = new ArrayList<Integer>();

List<Integer> left = new ArrayList<Integer>();

List<Integer> right = new ArrayList<Integer>();

left = inorderTraversal(root.left);

right = inorderTraversal(root.right);

res.addAll(left);

res.add(root.val);

res.addAll(right);

return res;

}

else if (root.left != null && root.right == null) {

List<Integer> res = new ArrayList<Integer>();

List<Integer> left = new ArrayList<Integer>();

left = inorderTraversal(root.left);

res.addAll(left);

res.add(root.val);

return res;

}

else {

List<Integer> res = new ArrayList<Integer>();

List<Integer> right = new ArrayList<Integer>();

right = inorderTraversal(root.right);

res.add(root.val);

res.addAll(right);

return res;

}

}

}

大神简洁版：

**public** **class** **Solution** {

**public** **boolean** isSymmetric(TreeNode root) {

**return** isSymmetric(root, root);

}

**boolean** isSymmetric(TreeNode n1, TreeNode n2) {

**if**(n1 == **null** && n2 == **null**) **return** **true**;

**if**(n1 == **null** || n2 == **null**) **return** **false**;

**if**(n1.val != n2.val) **return** **false**;

**return** isSymmetric(n1.left, n2.right) && isSymmetric(n2.right, n1.left);

}

}