Problem 101: Symmetric Tree

Problem Information

Difficulty: Easy
Acceptance Rate: 60.23%
Paid Only: No
Tags: Tree, Depth-First Search, Breadth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, _check whether it is a mirror of itself_ (i.e., symmetric around its center).

Example 1:

![](https://assets.leetcode.com/uploads/2021/02/19/symtree1.jpg)

Input: root = [1,2,2,3,4,4,3] **Output:** true

Example 2:

![](https://assets.leetcode.com/uploads/2021/02/19/symtree2.jpg)

Input: root = [1,2,2,null,3,null,3] **Output:** false

Constraints:

* The number of nodes in the tree is in the range `[1, 1000]`. * `-100 <= Node.val <= 100`

Follow up: Could you solve it both recursively and iteratively?

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
bool isSymmetric(TreeNode* root) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public boolean isSymmetric(TreeNode root) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def isSymmetric(self, root: Optional[TreeNode]) -> bool:
```