

# 101. Symmetric Tree

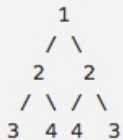
## 101 Symmetric Tree

- Depth-first Search + tree
- Breadth-first Search + tree

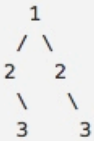
### Description

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

For example, this binary tree `[1,2,2,3,4,4,3]` is symmetric:



But the following `[1,2,2,null,3,null,3]` is not:



#### Note:

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

### 1. Thought line

### 2. Breadth-first Search + tree

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8  * };
9  */
10 class Solution {
11 private:
12     bool BreadthFirstSearchSymmetric(TreeNode* leftNode, TreeNode* rightNode){
13         if (!leftNode && !rightNode) return true;
14         if (leftNode && rightNode){
15             return ((leftNode->val == rightNode->val) && BreadthFirstSearchSymmetric(leftNode->left, rightNode->right) &&
16 BreadthFirstSearchSymmetric(leftNode->right, rightNode->left));
17         }
18         return false;
19     }
20 public:
21     bool isSymmetric(TreeNode* root) {
22         if (root==nullptr) return true;
23         return BreadthFirstSearchSymmetric(root->left, root->right);
24     };
25 }
```

