# 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:

```
    1
   / \
  2   2
 / \ / \
3  4 4  3
```

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

```
    1
   / \
  2   2
   \   \
   3    3
```

**Note:**

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

### 1. Thought line

### 2. Breadth-first Search + tree

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
private:
    bool BreadthFirstSearchSymmetric(TreeNode* leftNode, TreeNode* rightNode){
        if (!leftNode && !rightNode) return true;
        if (leftNode && rightNode){
            return ((leftNode->val == rightNode->val) && BreadthFirstSearchSymmetric(leftNode->left, rightNode->right) &&
BreadthFirstSearchSymmetric(leftNode->right, rightNode->left));
        }
        return false;
    }
public:
    bool isSymmetric(TreeNode* root) {
```

```
        if (root==nullptr) return true;
        return BreadthFirstSearchSymmetric(root->left, root->right);
    }
};
```