/*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.*/

-
- 思想:
- (1) 比较简单，增加递归辅助函数，用于判断两个节点分别形成的子树是否对称：判断标准是：子树的根节点值相同 且 左根节点的左子树与右根节点的右子树 对称 且 左根节点的右子树与右根节点的左子树 对称 *
- (2) 一个巧妙的判断左/右节点可能为空的方法是

```
   if (left == null || right == null) {
        return left == right;
    }
```

```
public boolean isSymmetric(TreeNode root) {

    if (root == null) {
        return true;
    }

    return isSymmetricHelper(root.left, root.right);
}

private boolean isSymmetricHelper(TreeNode left, TreeNode right) {

    if (left == null || right == null) {
        return left == right;
    }

    if (left.val != right.val) {
        return false;
    }

    return isSymmetricHelper(left.left, right.right) && isSymmetricHelper(left.right,right.left);
}
```