0101. 对称二叉树

ITCharge 大约 2 分钟

- 标签：树、深度优先搜索、广度优先搜索、二叉树
- 难度：简单

题目链接

- 0101. 对称二叉树 - 力扣

题目大意

描述：给定一个二叉树的根节点 `root` 。

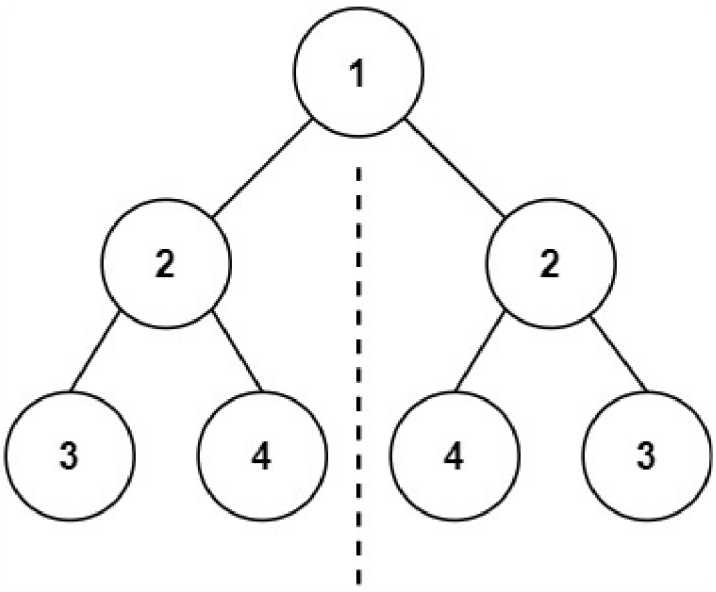
要求：判断该二叉树是否是左右对称的。

说明：

- 树中节点数目在范围 $[1, 1000]$ 内。
- $-100 \leq Node.val \leq 100$。

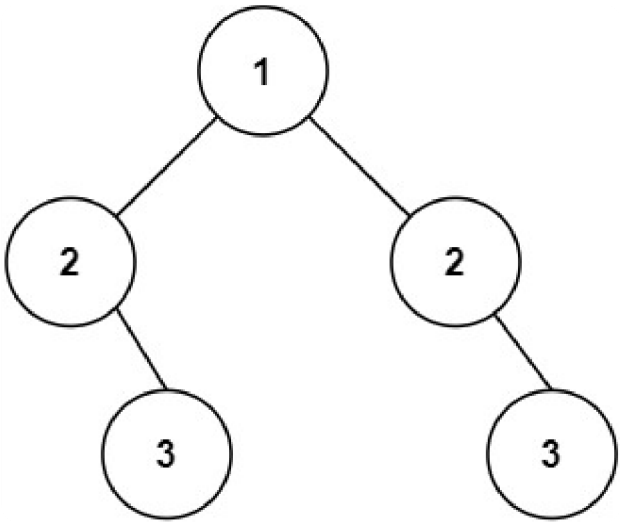
示例：

- 示例 1：



```py
输入：root = [1,2,2,3,4,4,3]
输出：true
```

- 示例 2：




```py
输入：root = [1,2,2,null,3,null,3]
输出：false
```

解题思路

思路 1：递归遍历

如果一棵二叉树是对称的，那么其左子树和右子树的外侧节点的节点值应当是相等的，并且其左子树和右子树的内侧节点的节点值也应当是相等的。

那么我们可以通过递归方式，检查其左子树与右子树外侧节点和内测节点是否相等。即递归检查左子树的左子节点值与右子树的右子节点值是否相等（外侧节点值是否相等），递归检查左子树的右子节点值与右子树的左子节点值是否相等（内测节点值是否相等）。

具体步骤如下：

1. 如果当前根节点为 `None` ，则直接返回 `True` 。
2. 如果当前根节点不为 `None` ，则调用 `check(left, right)` 方法递归检查其左右子树是否对称。
    1. 如果左子树节点为 `None` ，并且右子树节点也为 `None` ，则直接返回 `True` 。
    2. 如果左子树节点为 `None` ，并且右子树节点不为 `None` ，则直接返回 `False` 。

3. 如果左子树节点不为 `None` ，并且右子树节点为 `None` ，则直接返回 `False` 。
4. 如果左子树节点值不等于右子树节点值，则直接返回 `False` 。
5. 如果左子树节点不为 `None` ，并且右子树节点不为 `None` ，并且左子树节点值等于右子树节点值，则：
   1. 递归检测左右子树的外侧节点是否相等。
   2. 递归检测左右子树的内测节点是否相等。
   3. 如果左右子树的外侧节点、内测节点值相等，则返回 `True` 。

思路 1：代码

```py
class Solution:
    def isSymmetric(self, root: TreeNode) -> bool:
        if root == None:
            return True
        return self.check(root.left, root.right)

    def check(self, left: TreeNode, right: TreeNode):
        if left == None and right == None:
            return True
        elif left == None and right != None:
            return False
        elif left != None and ri... == None:
            return False
        elif left.val != right.val:
            return False

        return self.check(left.left, right.right) and self.check(left.right,
right.left)
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 是二叉树的节点数目。
- **空间复杂度**：$O(n)$。递归函数需要用到栈空间，栈空间取决于递归深度，最坏情况下递归深度为 n，所以空间复杂度为 $O(n)$。