题目描述

101. Symmetric Tree

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center). 判断是否是镜像树（沿中心轴对称）

示例

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
```

解题思路

一开始看到这道题想的是用栈，按照深度优先搜索（DFS）的顺序。因为是第一次用栈存二叉树，各种情况又没有考虑很周全，很多次都是wrong answer。。这道题大家可以尝试用三种不同的方法写一下（栈/队列和递归），练习熟了这样接下来比如102的题就会简单很多。

代码（栈）

```
class Solution:
    def isSymmetric(self, root: TreeNode) -> bool:
        if not root:
            return True

        stack = collections.deque() # collections module很好用，单独用list也可以
        stack.append(root.left)
        stack.append(root.right)

        while stack:
            leftNode = stack.pop()
            rightNode = stack.pop()
            if not leftNode and not rightNode:
                continue # wrong when I first thought it should return False
            if not leftNode or not rightNode:
                return False
            if leftNode.val != rightNode.val:
                return False
            stack.append(leftNode.left) # careful of the four sequence
            stack.append(rightNode.right)
```

```
            stack.append(leftNode.right)
            stack.append(rightNode.left)
        return True
```

代码（队列）

和栈差不多，主要是将popleft改成pop

```
class Solution:
    def isSymmetric(self, root: TreeNode) -> bool:
        if not root:
            return True

        queue = collections.deque()
        queue.append(root.left)
        queue.append(root.right)

        while queue:
            leftNode = queue.popleft()
            rightNode = queue.popleft()
            if not leftNode and not rightNode:
                continue # wrong when I first thought it should return False
            if not leftNode or not rightNode:
                return False
            if leftNode.val != rightNode.val:
                return False
            queue.append(leftNode.left)
            queue.append(rightNode.right)
            queue.append(leftNode.right)
            queue.append(rightNode.left)
        return True
```

代码（递归）

递归的思路很简单，但思路没有打开，一开始没往这边想

```
class Solution:
    def mirrorTrees(self, root):
      return mirror(root,root)

    def mirror(root1,root2):
      if not root1 and not root2:
        return True
      if root1 and root2 and root1.val == root2.val and
self.mirror(root1.left,root2.right):
        return True
      return False
```

小结

灵活用栈/队列/递归

```python
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution(object):
    def isSymmetric(self, root):
        """
        :type root: TreeNode
        :rtype: bool
        """
        if not root:
            return 1
        else:
            return self.symmetric(root.left, root.right)

    def symmetric(self, node1, node2):
        if not node1 and not node2:
            return 1
        if not node1 or not node2:
            return 0
        if node1.val != node2.val:
            return 0
        else:
            result = self.symmetric(node1.left, node2.right) and self.symmetric(node1.right, node2.left)
            return result

```