Mastering Data Structures and Algorithms
Homework 5

Yiying Peng (Christine)

March 27, 2019

Problem

Create a function to test if a binary tree is symmetric. A tree is symmetric if its left subtree is a mirror image of the right subtree.

Sol. My Python algorithm implementation is as follows.

```
class TreeNode:
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None

class Solution:
    def _isSymmetric(self, lroot, rroot):
        if lroot is None and rroot is None:
            return True
        elif lroot is None or rroot is None:
            return False
        elif lroot.val == rroot.val:
            return self._isSymmetric(lroot.left, rroot.right) & self._isSymmetric(lroot.right, rroot.left)
        else:
            return False
    def isSymmetric(self, root):
        if root is None:
            return True
        else:
            return self._isSymmetric(root.left, root.right)


#test case
sol = Solution()
root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(2)
root.left.left = TreeNode(3)
root.left.right = TreeNode(4)
root.right.left = TreeNode(4)
root.right.right = TreeNode(3)
print(sol.is_symmetric(root))

root = TreeNode(1)
root.left = TreeNode(2)
```

```
root.right = TreeNode(2)
root.left.left = None
root.left.right = TreeNode(3)
root.right.left = None
root.right.right = TreeNode(3)
print(sol.is_symmetric(root))
```

Using recursive approach. Due to we traverse the entire tree, the time complexity is $O(n)$.

In the worse case, the tree is linear. Therefore, the space complexity due to recursive calls on the stack will be $O(n)$.

□