Mastering Data Structures and Algorithms
In Class Exercise 4

Yiying Peng (Christine)

March 27, 2019

Problem

Create a function that flattens a binary tree into a double linked list, Use the left and right pointers as next and previous pointers respectively. The function must return the head of the list.

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 _inorder(self, root):
        if root:
            l_lst = self._inorder(root.left)
            r_lst = self._inorder(root.right)
            if l_lst:
                l_lst[-1].left = root
                root.right = l_lst[-1]
            if r_lst:
                r_lst[0].right = root
                root.left = r_lst[0]
            return l_lst + [root] + r_lst
        else:
            return []

    def inorder(self, root):
        if root is None:
            return None
        else:
            return self._inorder(root)[0]


sol = Solution()
r = TreeNode(5)
r.left = TreeNode(3)
r.right = TreeNode(6)
r.left.left = TreeNode(1)
r.left.right = TreeNode(4)
r.right.left = None
```

```
r.right.right = TreeNode(9)
head = sol.inorder(r)
print(head.val, head.left.val, head.left.left.val, head.left.right.val)
```

The time complexity is $O(n)$, and the space complexity is also $O(n)$.

□