0226. 翻转二叉树

ITCharge 大约 1 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

- 0226. 翻转二叉树 - 力扣

题目大意

描述：给定一个二叉树的根节点 `root` 。

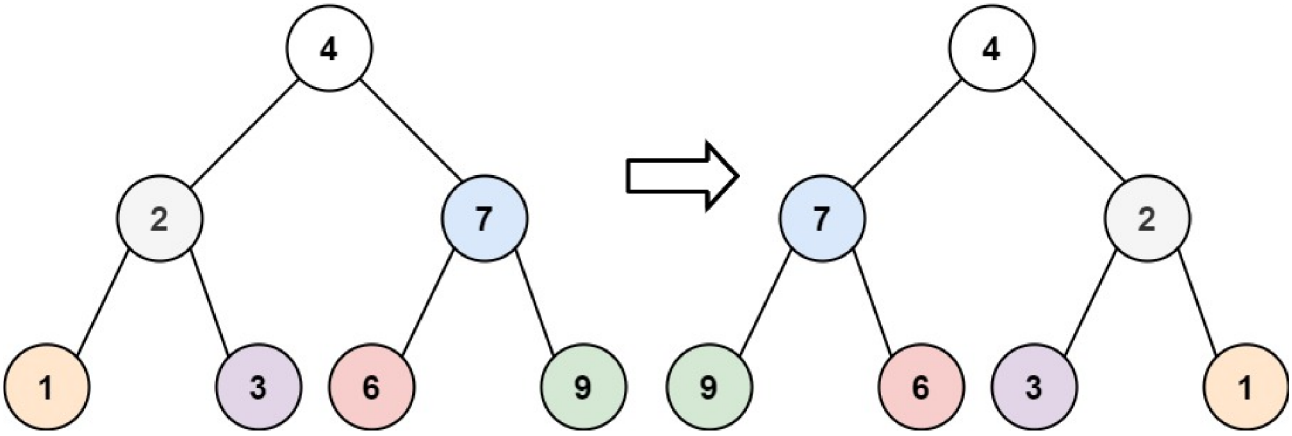
要求：将该二叉树进行左右翻转。

说明：

- 树中节点数目范围在 $[0, 100]$ 内。
- $-100 \leq Node.val \leq 100$。

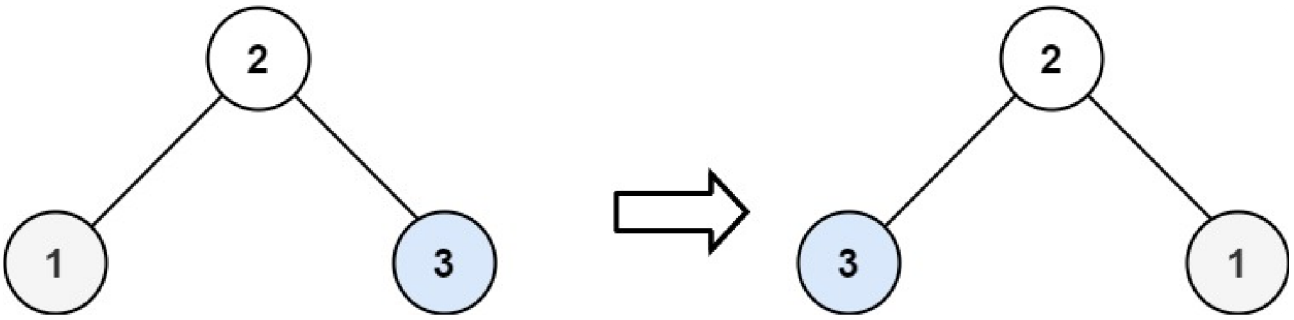
示例：

- 示例 1：




```py
输入：root = [4,2,7,1,3,6,9]
输出：[4,7,2,9,6,3,1]
```

- 示例 2：




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

解题思路

思路 1：递归遍历

根据我们的递推三步走策略，写出对应的递归代码。

1. 写出递推公式：
    1. 递归遍历翻转左子树。
    2. 递归遍历翻转右子树。
    3. 交换当前根节点 `root` 的左右子树。
2. 明确终止条件：当前节点 `root` 为 `None` 。
3. 翻译为递归代码：
    1. 定义递归函数： `invertTree(self, root)` 表示输入参数为二叉树的根节点 `root` ，返回结果为翻转后二叉树的根节点。
    2. 书写递归主体：

```py
left = self.invertTree(root.left)
right = self.invertTree(root.right)
root.left = right
root.right = left
return root
```

3. 明确递归终止条件： `if not root: return None`

4. 返回根节点 `root` 。

思路 1：代码

```py
class Solution:
    def invertTree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
        if not root:
            return None
        left = self.invertTree(root.left)
        right = self.invertTree(root.right)
        root.left = right
        root.right = left
        return root
```

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 [illegible] 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 $O(n)$。