0094. 二叉树的中序遍历

ITCharge 大约 2 分钟

- 标签：栈、树、深度优先搜索、二叉树
- 难度：简单

题目链接

- 0094. 二叉树的中序遍历 - 力扣

题目大意

描述：给定一个二叉树的根节点 `root` 。

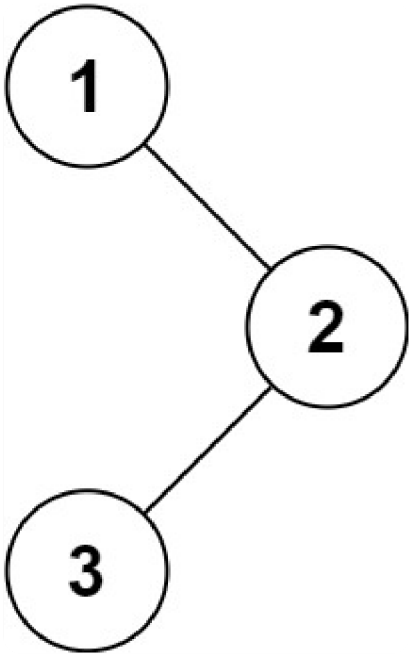
要求：返回该二叉树的中序遍历结果。

说明：

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

示例：

- 示例 1：



img

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

- 示例 2：

```py
输入：root = []
输出：[]
```

解题思路

思路 1：递归遍历

二叉树的前序遍历递归实现步骤为：

1. 判断二叉树是否为空，为空则直接返回。
2. 先访问根节点。
3. 然后递归遍历左子树。
4. 最后递归遍历右子树。

思路 1：代码

```py
class Solution:
    def inorderTraversal(self, root: TreeNode) -> List[int]:
        res = []
        def inorder(root):
            if not root:
                return
            inorder(root.left)
            res.append(root.val)
            inorder(root.right)

        inorder(root)
        return res
```

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**：$O(n)$。

思路 2：模拟栈迭代遍历

二叉树的前序遍历递归实现的过程，实际上就是调用系统栈的过程。我们也可以使用一个显式栈 `stack` 来模拟递归的过程。

前序遍历的顺序为：根节点 - 左子树 - 右子树，而根据栈的「先入后出」特点，所以入栈的顺序应该为：先放入右子树，再放入左子树。这样可以保证最终遍历顺序为前序遍历顺序。

二叉树的前序遍历显式栈实现步骤如下：

1. 判断二叉树是否为空，为空则直接返回。
2. 初始化维护一个栈，将根节点入栈。
3. 当栈不为空时：
    1. 弹出栈顶元素 `node`，并访问该节点。
    2. 如果 `node` 的右子树不为空，则将 `node` 的右子树入栈。
    3. 如果 `node` 的左子树不为空，则将 `node` 的左子树入栈。

思路 2：代码

```py
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        if not root:                # 二叉树为空直接返回
            return []

        res = []
        stack = []

        while root or stack:        # 根节点或栈不为空
            while root:
                stack.append(root)  # 将当前树的根节点入栈
                root = root.left    # 找到最左侧节点
```

```
            node = stack.pop()      # 遍历到最左侧，当前节点无左子树时，将最左侧节点弹出
            res.append(node.val)    # 访问该节点
            root = node.right       # 尝试访问该节点的右子树
        return res
```

思路 2：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**：$O(n)$。