前序遍历--迭代实现

```
class Solution(object):
    def preorderTraversal(self, root):
        # 使用栈辅助进行迭代算法进行前序遍历
        if not root: return []
        stack, res = [], []

        while root or stack:
            # 如果根节点存在
            # 将根节点的值记录在res中
            # 将根节点押入栈中
            # 将左节点作为根节点继续搜索
            while root:
                res.append(root.val)
                stack.append(root)
                root = root.left
            # 如果左节点搜索完毕
            # 则将最后押入的节点释放
            # 取该节点的右节点进行搜索
            root = stack.pop()
            root = root.right
        return res
```

中序遍历--迭代算法

```
class Solution:
    def inorderTraversal(self, root: TreeNode) -> List[int]:
        # 使用栈辅助实现迭代算法进行中序遍历
        stack, res = [], []
        if not root: return []

        while stack or root:
            # 如果节点存在，则将该节点存入栈中
            # 搜索该节点的左节点
            while root:
                stack.append(root)
                root = root.left

            # 当左节点不存在时，取出最后押入栈中的节点
            root = stack.pop()
            res.append(root.val)
            root = root.right
        return res
```