94. Binary Tree Inorder Traversal
```
class Solution:
    def inorderTraversal(self, root: TreeNode) -> List[int]:
        result = []
        def inorderTraverse(node):
            if not node:
                return
            inorderTraverse(node.left)
            result.append(node.val)
            inorderTraverse(node.right)
        inorderTraverse(root)
        return result
```

94. Binary Tree Inorder Traversal
```
class Solution:
    def inorderTraversal(self, root: TreeNode) -> List[int]:
        result = []
        stack = []
        cur = root
        while cur or stack:
            while cur:
                stack.append(cur)
                cur = cur.left
            cur = stack.pop()
            result.append(cur.val)
            cur = cur.right
        return result
```

589. N-ary Tree Preorder Traversal

```
class Solution:
    def preorder(self, root: 'Node') -> List[int]:
        result = []
        if not root:
            return result
        def traverse(node):
            result.append(node.val)
            for n in node.children:
                traverse(n)
        traverse(root)
        return result
```

590. N-ary Tree Postorder Traversal

```
"""
# Definition for a Node.
class Node:
    def __init__(self, val=None, children=None):
        self.val = val
        self.children = children
"""
class Solution:
    def postorder(self, root: 'Node') -> List[int]:
        result = []
        if not root:
            return result
        def traverse(node):
            for n in node.children:
                traverse(n)
            result.append(node.val)
        traverse(root)
        return result
```

590. N-ary Tree Postorder Traversal

```
class Solution:
    def postorder(self, root: 'Node') -> List[int]:
        result = []
        if not root:
            return result
        stack = [root]
        while stack:
            cur = stack.pop()
            result.append(cur.val)
            for node in cur.children:
                stack.append(node)
        return reversed(result)
```