Standardized Tree Traversal Algorithms

Algorithm	Best Case	Average Case	Worst Case
Inorder (Recursive & Iterative)	O(1)	O(N)	O(N)
Preorder (Recursive & Iterative)	O(1)	O(N)	O(N)
Postorder (Recursive & Iterative)	O(1)	O(N)	O(N)

Python Code for Standardized Tree Traversals

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

# Inorder Traversal - Recursive
def inorder_recursive(root):
    if root:
        inorder_recursive(root.left)
        print(root.val, end=" ")
        inorder_recursive(root.right)

# Inorder Traversal - Iterative
def inorder_iterative(root):
    stack, current = [], root
    while stack or current:
        if current:
            stack.append(current)
            current = current.left
        else:
            current = stack.pop()
            print(current.val, end=" ")
            current = current.right

# Preorder Traversal - Recursive
def preorder_recursive(root):
    if root:
        print(root.val, end=" ")
        preorder_recursive(root.left)
        preorder_recursive(root.right)

# Preorder Traversal - Iterative
def preorder_iterative(root):
    stack = [root] if root else []
    while stack:
        node = stack.pop()
        print(node.val, end=" ")
        if node.right:
            stack.append(node.right)
        if node.left:
            stack.append(node.left)

# Postorder Traversal - Recursive
def postorder_recursive(root):
    if root:
        postorder_recursive(root.left)
        postorder_recursive(root.right)
        print(root.val, end=" ")

# Postorder Traversal - Iterative
def postorder_iterative(root):
    if not root:
        return
    stack, output = [root], []
    while stack:
        node = stack.pop()
        output.append(node.val)
        if node.left:
            stack.append(node.left)
        if node.right:
            stack.append(node.right)
    print(" ".join(map(str, output[::-1])))
```