Documentation: Binary Tree Preorder Traversal

Problem Statement

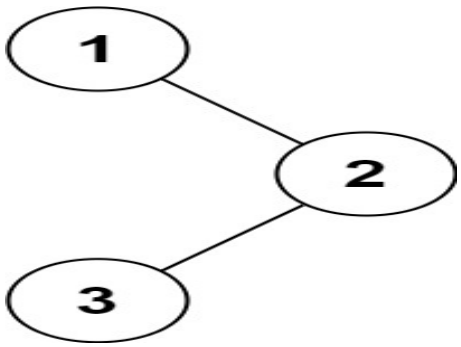
You are given the root of a binary tree. Your task is to return the preorder traversal of its nodes' values.

Preorder Traversal is a depth-first traversal method where nodes are recursively visited in the following order:

1. Visit the root node.
2. Traverse the left subtree.
3. Traverse the right subtree.

Example 1:

- **Input:** root = [1, null, 2, 3]
- **Output:** [1, 2, 3]
- **Explanation:** The binary tree represented is:



The preorder traversal visits nodes in the order: 1 (root), 2 (right child of 1), 3 (left child of 2).

Example 2:

- **Input:** root = []
- **Output:** []
- **Explanation:** The tree is empty, so the output is an empty list.

Example 3:

- **Input:** root = [1]
- **Output:** [1]
- **Explanation:** The tree has only one node. The output is simply the value of the root node.

Constraints

- The number of nodes in the tree is in the range [0, 100].
- Node values are within the range [-100, 100].

Follow-Up

The recursive solution for this problem is straightforward. However, an iterative approach is also possible and can be more efficient in some cases. The iterative solution uses a stack to simulate the recursive call stack and performs the traversal in a non-recursive manner.

Iterative Approach (Non-Recursive)

1. **Initialization:**
    - Start with a stack containing the root node.
    - Initialize an empty list to store the result.

2. **Traversal:**
    - *While the stack is not empty:*
        - Pop the top node from the stack.
        - *If the node is not null:*
            - ✓ Append the node's value to the result list.
            - ✓ Push the right child of the node to the stack (if it exists).
            - ✓ Push the left child of the node to the stack (if it exists).

3. **Return:**
    - After the traversal is complete, return the result list containing the preorder traversal of the tree.

This approach ensures that nodes are processed in the correct preorder sequence using an iterative method.