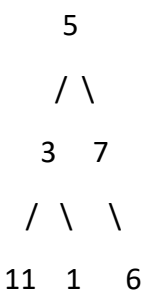
PreOrder Traversal of Binary Tree LeetCode

Pre-order traversal: current node, Left subtree, right subtree

Example:



Output: **[5, 3, 11, 1, 7, 6]**

Approach 1: Perform an pre-order traversal of the binary tree using recursion

- **Explanation:** The recursive approach starts at the root node and follows the order: current node, left subtree, right subtree. It pushes the value of the current node into a vector and then recursively traverses its left and right subtrees.
- **Time Complexity: The time complexity of this approach is O(N), where N is the number of nodes in the binary tree.** This is because we visit each node once.
- **Space Complexity**: **The space complexity is O(H), where H is the height of the binary tree. In the worst case, for a skewed tree, the space complexity can be O(N), as the recursion stack may contain all nodes.**

Approach 2: Perform an pre-order traversal of the binary tree using an iterative approach

- **Explanation:** The iterative approach uses a stack to simulate the recursive process. It starts at the root node and follows the order: current node, left subtree, right subtree. It maintains a stack to keep track of nodes to be processed. It pushes the value of the current node into a vector, processes the left subtree, and then moves to the right subtree.
- **Time Complexity: The time complexity of this approach is also O(N) because it visits each node once.**
- **Space Complexity: The space complexity is O(H), where H is the height of the binary tree, due to the space required for the stack. In the worst case, for a skewed tree, the space complexity can be O(N).**