Left View of Binary Tree GFG

Given a Binary Tree, return Left view of it. Left view of a Binary Tree is set of nodes visible when tree is visited from Left side.

Example:

```
            5
          /    \
        3       7
      /   \      \
   11      1       6
  /   \          /   \
9     12      13    15
```

Output: The Left View of Binary Tree: 5, 3, 11, 9

Approach 1: Function to print Left View of a binary tree using a recursive approach.

- The recursive approach involves starting from the root and traversing the tree while maintaining the current level.
- It keeps track of the leftmost node encountered at each level, adding it to the result vector.
- The traversal begins from the root and proceeds first to the left child and then to the right child.
- The result vector contains the left view nodes.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once.**

Space Complexity:

- **The space complexity is O(H), where H is the height of the binary tree, due to the function call stack.**

Approach 2: Function to print the Left View of a binary tree using an iterative approach

- The iterative approach uses level-order traversal (BFS) to traverse the tree.
- It maintains a queue with a pair of nodes and their levels.

- While traversing, it checks if the size of the result vector is equal to the current level. If so, it adds the leftmost node at that level to the result vector.
- This ensures that the left view nodes are added in left-to-right order.
- Finally, it returns the result vector.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once during level-order traversal.**

Space Complexity:

- **The space complexity is O(N) in the worst case, primarily due to the queue used for level-order traversal and the map used to store bottom view nodes.**

Conclusion:

Both the recursive and iterative approaches effectively print the left view of a binary tree. The recursive approach uses a depth-first traversal, while the iterative approach employs a breadth-first traversal. Both approaches have a time complexity of O(N) and provide the expected left view nodes.

The choice between the two approaches depends on the specific requirements and constraints of the problem.