Write a Program to view a tree from left View.

Mr. Vikas Kumar
Assistant Professor
Industry Embedded Program

Lesson Plan

Subject/Course	Competitive Coding
Lesson Title	Write a Program to view a tree from left View.

Lesson Objectives
Understand the concept of tree views, especially the Left View of a binary tree.
Learn how to identify and display nodes visible from the left side of a tree.
mplement the Left View logic using Level-Order (BFS) and Depth-First (DFS) traversals.
Analyze the time and space complexity of both approaches for efficiency comparison.

Problem Statement:

The Left View of a binary tree consists of nodes visible when the tree is viewed from the left side.

The task is to traverse the tree and print the first node at each level.

Concept

A Binary Tree can be viewed from different angles — left, right, top, bottom.
The Left View includes only the leftmost nodes at each level.

We can find it using:

1. Level-Order Traversal (BFS) — using a queue.

2. Depth-First Search (DFS) — using recursion and level tracking.

Algorithm/Logic

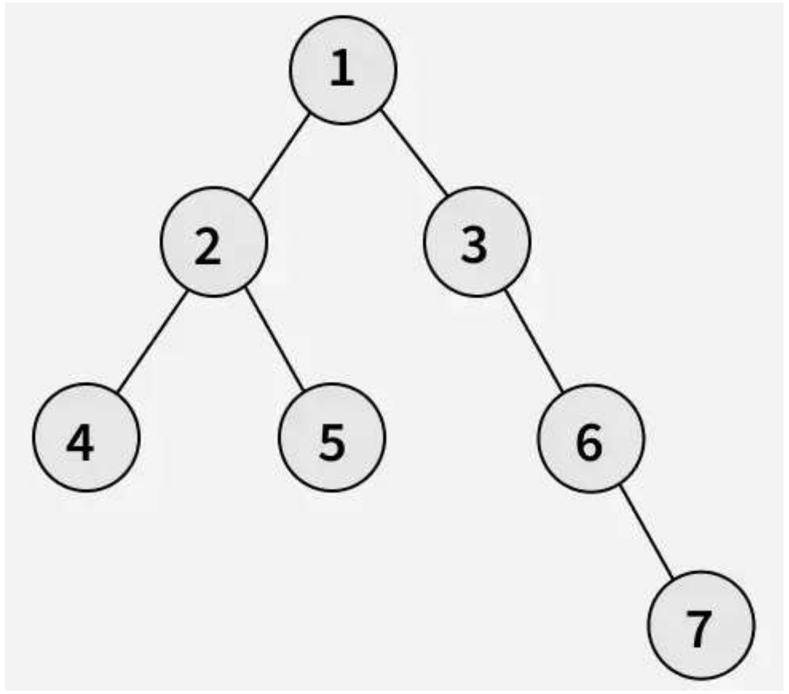
Approach 1 — BFS (Level Order):

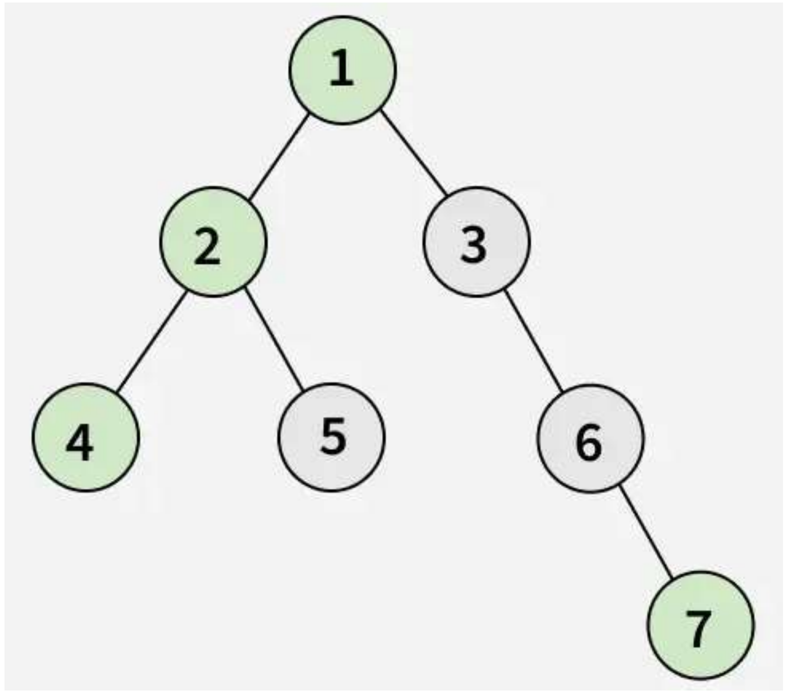
1. Initialize a queue and push the root node.
2. For each level, process all nodes — print the first node of that level.
3. Enqueue left and right children of each node.

Approach 2 — DFS (Recursive):

1. Start from the root with level = 1 and maxLevel = 0.
2. If current level > maxLevel, print the node and update maxLevel.
3. Recur left, then right.

Visualization





Output: [1, 2, 4, 7]
Explanation: From the left side of the tree, only the nodes 1, 2, 4 and 7 are visible.

Code Implementation

```
import java.util.*;

class Node {
    int data;
    Node left, right;

    Node(int data) {
        this.data = data;
        left = right = null;
    }
}

public class LeftViewBinaryTree {

    // Function to print the Left View
    public static void printLeftView(Node root) {
        if (root == null)
            return;
Queue<Node> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int levelSize = queue.size();  // Number of nodes at current level

            for (int i = 0; i < levelSize; i++) {
                Node current = queue.poll();

                // Print the first node of each level
                if (i == 0)
                    System.out.print(current.data + " ");

                if (current.left != null)
                    queue.add(current.left);
                if (current.right != null)
                    queue.add(current.right);
            }
        }
    }
```

```
public static void main(String[] args) {
        // Constructing a sample binary tree
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.left = new Node(4);
        root.left.right = new Node(5);
        root.right.left = new Node(6);
        root.right.right = new Node(7);
        root.left.left.left = new Node(8);

        System.out.print("Left View of Binary Tree: ");
        printLeftView(root);
    }
}
```

Time & Space Complexity

Approach	Time Complexity	Space Complexity
BFS	O(n)	O(width of tree)
DFS	O(n)	O(height of tree)

Summary

- The Left View of a binary tree shows all nodes visible from the left side.
- Can be implemented using BFS (iterative) or DFS (recursive) traversal.
- Both have O(n) time complexity.
- Commonly used in visualization, tree analysis, and real-world rendering algorithms.

Practice Questions:

1 Left View of Binary Tree — LeetCode #199

🔗 https://leetcode.com/problems/binary-tree-right-side-view/

(Use the same concept for Left View.)

Concept:

Use BFS or DFS traversal to print the visible nodes from the left side of the tree.

Why Practice:

Strengthens understanding of tree traversal and level-wise processing.

Thanks