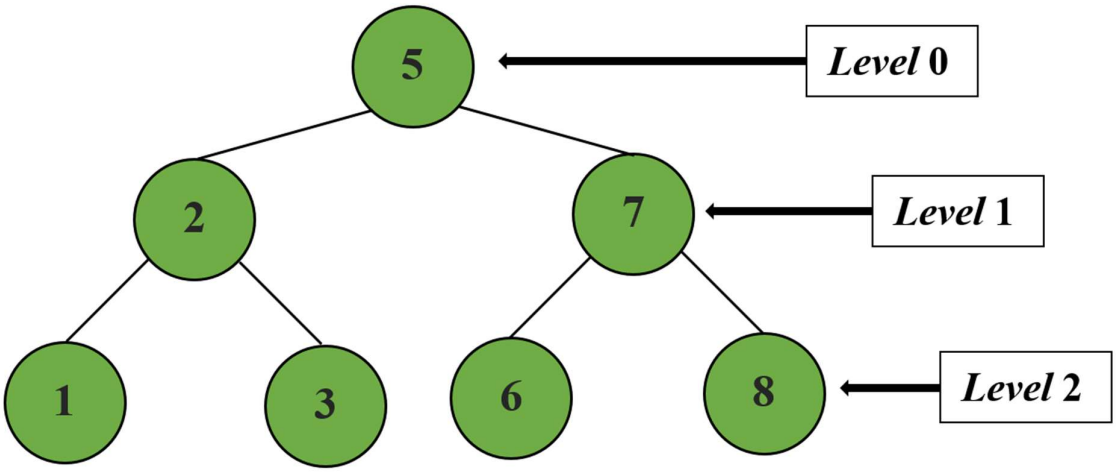
Level Order Traversal: -



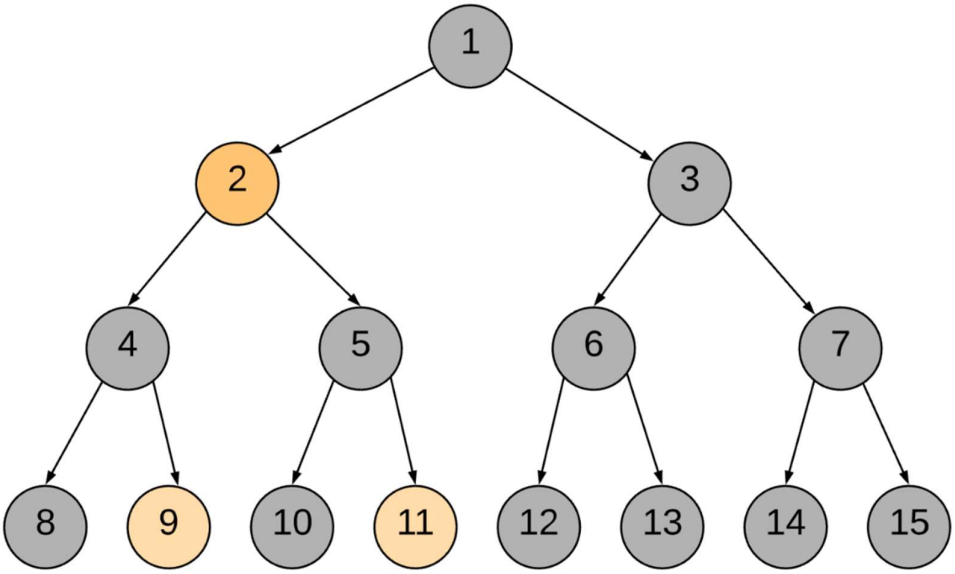

```
public class LevelOrderTraversal {
	public static void printLevel(Node root, int level) {
		if (root == null) {
			return;
		}
		if (level == 1) {
			System.out.print(root.data + " ");
		}
		printLevel(root.left, level - 1);
		printLevel(root.right, level - 1);
	}

	public static void main(String[] args) {
		Node root = new Node(1);
		root.left = new Node(2);
		root.right = new Node(3);
		root.right.left = new Node(4);
		root.right.right = new Node(5);
		int height = HeightBTree.height(root);
		for (int i = 1; i <= height; i++) {
			printLevel(root, i);
		}
	}
}
```

TC → O(n * h)

SC → O(h)

Lowest/Closest Common Ancestor



Lowest Common Ancestor for **Node 9** and **Node 11** is **Node 2**

```
public class LowestCommonAncestor {
	static Node lca(Node root, int node1, int node2) {
		if (root == null) {
			return null;
		}
		if (root.data == node1 || root.data == node2) {
			return root;
		}
		Node Llca = lca(root.left, node1, node2);
		Node Rlca = lca(root.right, node1, node2);
		if (Llca == null && Rlca == null) {
			return null;
		}
		if (Llca == null) {
			return Rlca;
		}
		if (Rlca == null) {
			return Llca;
		}
		return root;
	}
}
```

TC → O(n)

SC → O(h)

Mercy Technologies