```
public class HeightBTree {
	static int height(Node root) {
		// Base Condition
		if (root == null) {
			return -1;
		}

		int leftHeight = height(root.left);
		int rightHeight = height(root.right);
		if (leftHeight > rightHeight) {
			return leftHeight + 1;
		} else {
			return rightHeight + 1;
		}
	}
}
```

TC → O(n)

SC → O(h)

```
public class NumberOfNodesInBinaryTree {
	static int getNodesCount(Node root) {
		if (root == null) {
			return 0;
		}
		int leftCount = getNodesCount(root.left);
		int rightCount = getNodesCount(root.right);
		return leftCount + rightCount + 1;
	}

	public static void main(String[] args) {
		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);
		System.out.println(getNodesCount(root));
	}
}
```

TC → O(n)

SC → O(h)

```java
public class MaximumElementInBinaryTree {
	static int getMaxElement(Node root) {
		// Base Condition
		if (root == null) {
			return Integer.MIN_VALUE;
		}
		int leftMax = getMaxElement(root.left);
		int rightMax = getMaxElement(root.right);
		return Math.max(root.data, Math.max(leftMax, rightMax));
	}

	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);
		System.out.println(getMaxElement(root));
	}
}
```

TC → O(n)

SC → O(h)

```java
public class NumberOfLeafNodes {
	static int getNumberOfLeafNodes(Node root) {
		if (root == null) {
			return 0;
		}
		if (root.left == null && root.right == null) {
			return 1;
		}
		int leftLeaves = getNumberOfLeafNodes(root.left);
		int rightLeaves = getNumberOfLeafNodes(root.right);
		return leftLeaves + rightLeaves;
	}

	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);
		System.out.println(getNumberOfLeafNodes(root));
	}
}
```

TC → O(n)

SC → O(h)

```java
public class SearchForElementInBinaryTree {

	static Node search(Node root, int x) {
		if (root == null) {
			return null;
		}
		if (root.data == x) {
			return root;
		}
		Node leftSearch = search(root.left, x);
		if (leftSearch != null) {
			return leftSearch;
		}
		Node rightSearch = search(root.right, x);
		if (rightSearch != null) {
			return rightSearch;
		}
		return null;
	}

	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);
		System.out.println(search(root, 2).data);
	}
}
```

TC → O(n)

SC → O(h)