Experiment-5

Student Name: Anupreet Kaur **UID: 22BCS50071**

Branch: BE-CSE **Section/Group: _NTPP_IOT-602-A**

Semester: 6th **Date of Performance: 17/02/2025**

Subject Name: AP Lab **Subject Code: 22CSP-351**

1. Aim: Tree.

- ❖ Problem 1.2.1: Maximum Depth of Binary Tree
- ❖ Problem 1.2.2: Symmetric Tree
- ❖ Problem 1.2.2: Validate Binary Search Tree

2. Objective:

To understand and implement Tree algorithms for problem-solving.

3. Theory:

A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.
The problem requires checking if a binary tree is symmetric, meaning it is a mirror of itself around the center.
A valid BST is defined as follows:
The left subtree of a node contains only nodes with keys less than the node's key.
The right subtree of a node contains only nodes with keys greater than the node's key.
Both the left and right subtrees must also be binary search trees.

4. Code:

Maximum Depth of Binary Tree

```
class Solution {
  public int maxDepth(TreeNode root) {
    if (root == null) {
      return 0;
    }

    int leftDepth = maxDepth(root.left);
    int rightDepth = maxDepth(root.right);

    return Math.max(leftDepth, rightDepth) + 1;
  }
}
```

Symmetric Tree

```
class Solution {
   public boolean isSymmetric(TreeNode root) {
      if (root == null) return true;
      return isMirror(root.left, root.right);
   }

   private boolean isMirror(TreeNode t1, TreeNode t2) {
      if (t1 == null && t2 == null) return true;  // Both null -> symmetric
      if (t1 == null || t2 == null) return false; // One null, one not -> not symmetric
      if (t1.val != t2.val) return false;         // Values must be equal

      // Check mirrored children (t1.left vs t2.right, t1.right vs t2.left)
      return isMirror(t1.left, t2.right) && isMirror(t1.right, t2.left);
   }
  }
```

Validate Binary Search Tree

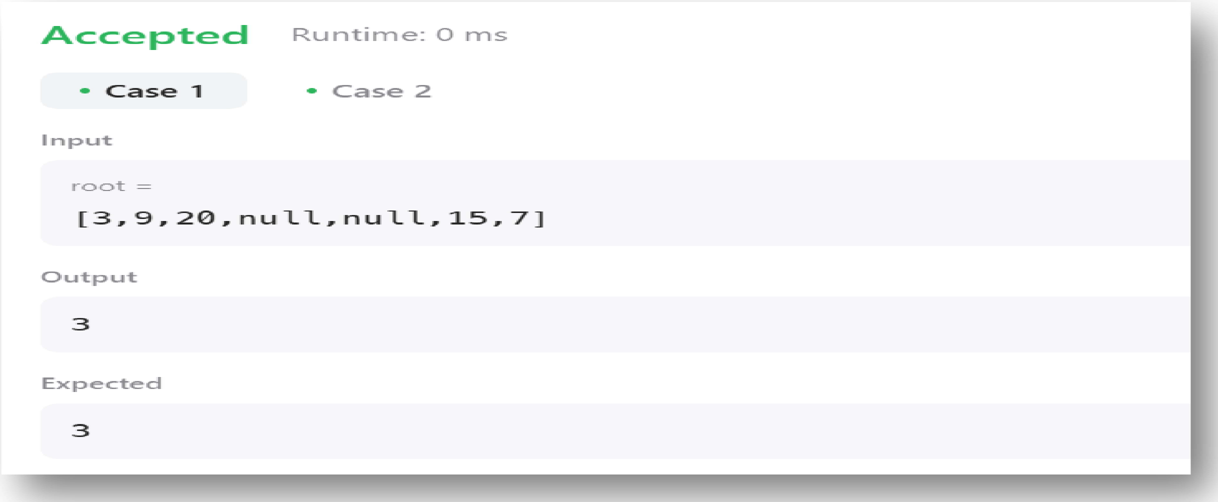
```
class Solution {
   public boolean isValidBST(TreeNode root) {
      return isValidBSTHelper(root, Long.MIN_VALUE, Long.MAX_VALUE);
   }

   private boolean isValidBSTHelper(TreeNode node, long min, long max) {
      if (node == null) return true;

      if (node.val <= min || node.val >= max) return false;

      return isValidBSTHelper(node.left, min, node.val) &&
           isValidBSTHelper(node.right, node.val, max);
   }
}
```

6. Output:



Input

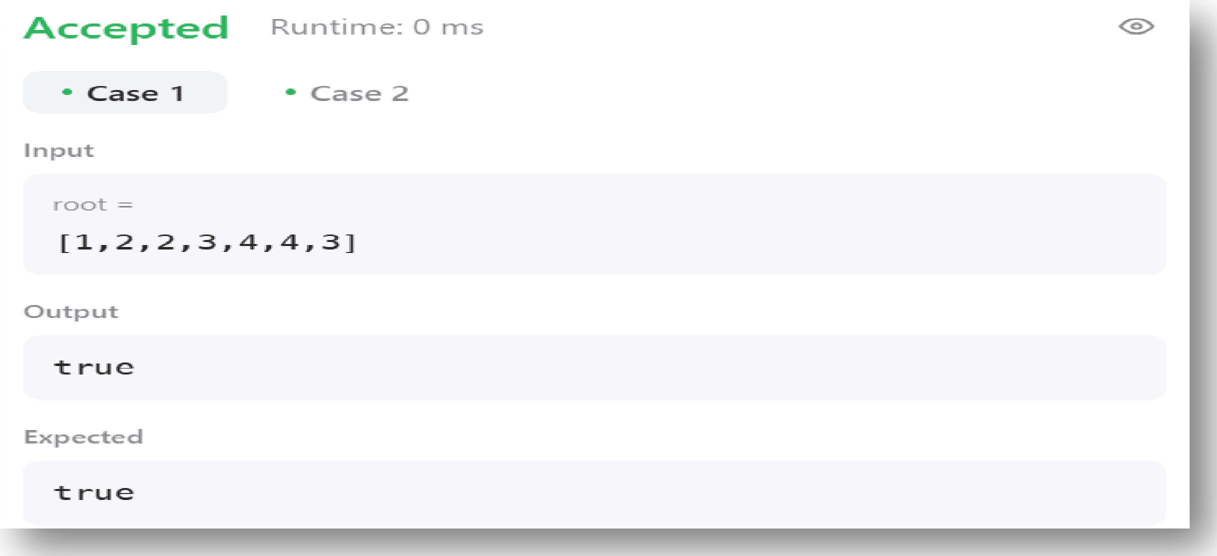
```
root =
[3,9,20,null,null,15,7]
```

Output

```
3
```

Expected

```
3
```



Input

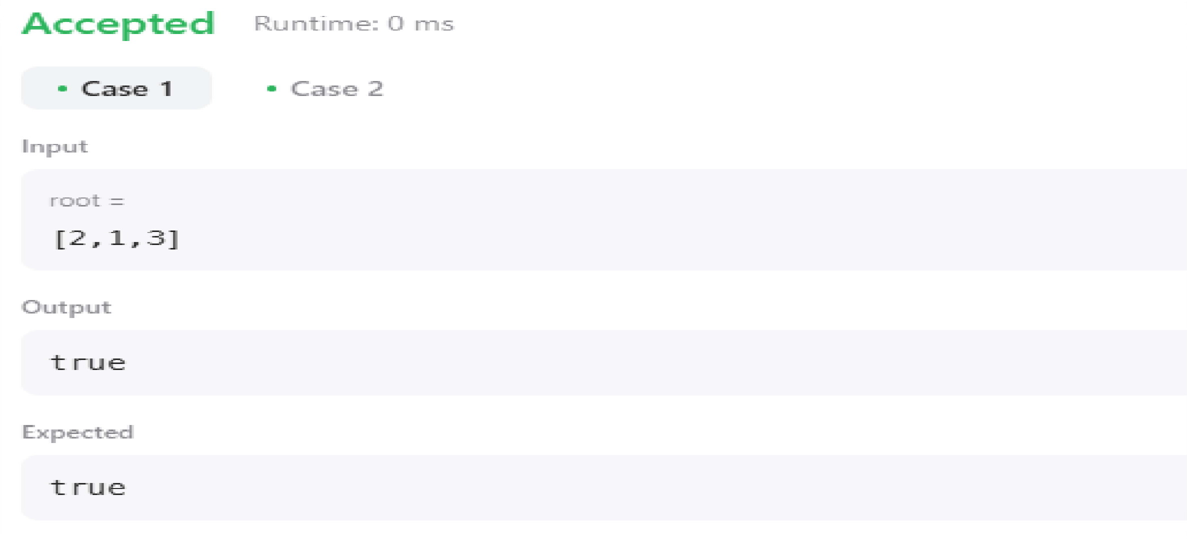
```
root =
[1,2,2,3,4,4,3]
```

Output

```
true
```

Expected

```
true
```



Input

```
root =
[2,1,3]
```

Output

```
true
```

Expected

```
true
```

7. Learning Outcomes:

- Understand the definition and properties of a binary search tree (BST).
- Apply recursion to check if each node in the tree satisfies the BST property, where left children are less and right children are greater.
- Analyze how boundary values (Long.MIN_VALUE, Long.MAX_VALUE) help ensure correct comparison when validating node values.
- Evaluate the time complexity of the solution, recognizing that it processes each node once (O(n)).