Experiment 6:

Name: Himanshu **UID: 22BCS50162**
Branch:BE/CSE **Section/Group: 634/B**
Semester: 6th **Date of performance:28-01-25**
Subject Name:APLab-2 **Subject Code: 22CSP-351**

1) Aim:

Problem 1: Binary Tree Inorder Traversal

- Problem Statement: Given the root of a binary tree, return the inorder traversal of its nodes' values.

Problem 2: Maximum Depth of Binary Tree

- Problem Statement: Given the root of a binary tree, return its maximum depth. 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.

Problem 3: Validate Binary Search Tree

- Problem Statement: Given the root of a binary tree, determine if it is a valid binary search tree (BST). 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.

2) Objective:

Given the root of a binary tree, perform inorder traversal to return its nodes' values, calculate its maximum depth, and determine if it is a valid binary search tree (BST).

3) Code:

Problem 1: Binary Tree Inorder Traversal

```
class Solution {
   ArrayList<Integer> ans = new ArrayList<>();
   public List<Integer> inorderTraversal(TreeNode root) {
      inorder(root);
      return ans;
   }
   public void inorder(TreeNode root){
```

```
        if(root == null ){
            return;
        }
        inorder(root.left);
        ans.add(root.val);
        inorder(root.right);
    }
}
```

Problem 2: Maximum Depth of Binary Tree

```
class Solution {
    public int maxDepth(TreeNode root) {
        if(root==null){
            return 0;
            }
        int left=maxDepth(root.left);
        int right=maxDepth(root.right);
        int ans=Math.max(left,right)+1;
        return ans;
}
}
```

Problem 3: Validate Binary Search Tree

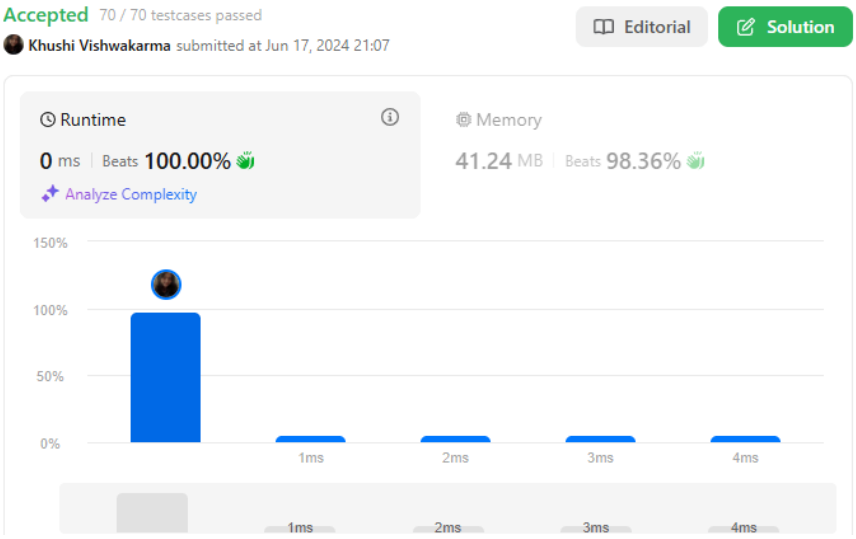
```
class Solution {
    private long minVal = Long.MIN_VALUE;
    public boolean isValidBST(TreeNode root) {
        if (root == null) return true;
        if (!isValidBST(root.left)) return false;

        if (minVal >= root.val) return false;
```

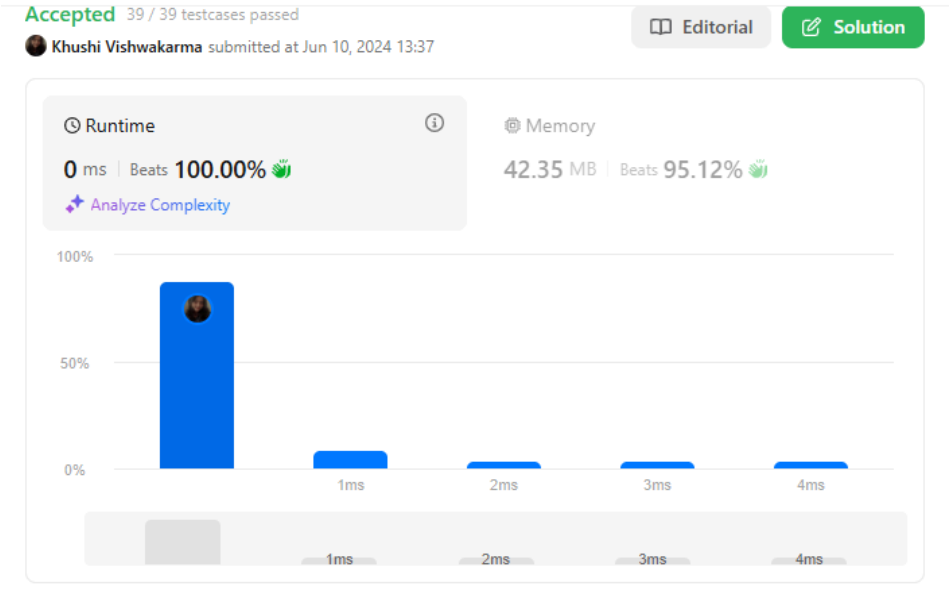
```
        minVal = root.val;
        if (!isValidBST(root.right)) return false;
        return true;
    }
}
```

4) Result:

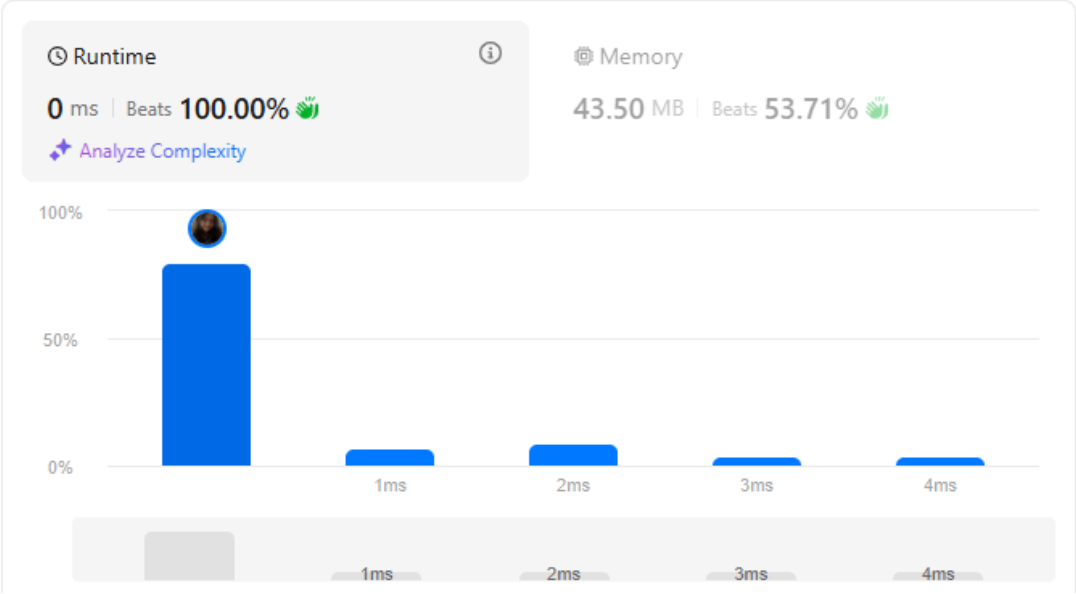
Problem 1: Binary Tree Inorder Traversal



Problem 2:Maximum Depth of Binary Tree



Problem 3: Validate Binary Search Tree



5) Learning Outcome:

1. **Inorder Traversal**: Learn to traverse binary trees in inorder to visit nodes in a specific sequence.
2. **Recursive Depth Calculation**: Use recursion to calculate the maximum depth of a binary tree.
3. **BST Validation**: Understand how to check if a binary tree follows the binary search tree properties.
4. **Tree Operations Optimization**: Optimize tree traversal and analysis techniques for efficiency.
5. **Problem-Solving with Trees**: Enhance problem-solving skills with various tree operations and structures.