Experiment 6

Student Name: Niraj Kumar **UID: 22BCS16736**
Branch: CSE **Section: 638/A**
Semester: 6th **DOP: 18-02-2014**
Subject: AP **Subject Code:22CSP-351**

Aim:

Problem-1: Convert Sorted Array to Binary Search Tree

Algorithm:

Algorithm for Convert Sorted Array to Binary Search Tree (BST):

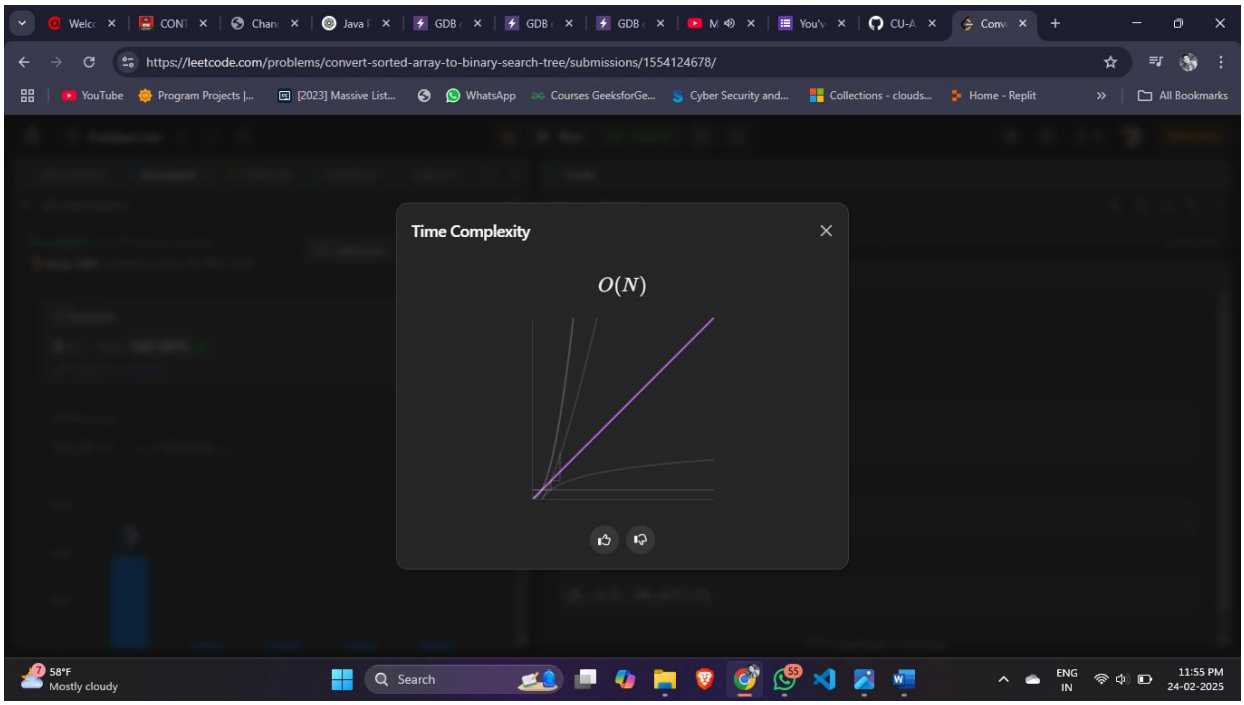
1. Define a TreeNode class with 'val', 'left', and 'right' attributes.
2. Implement a method sortedArrayToBST(int[] nums) that:
   a. Calls a helper method buildBST(nums, left, right) with initial bounds (0, nums.length - 1).
3. In buildBST:
   a. If left > right, return null (base case).
   b. Calculate mid as left + (right - left) / 2 to avoid overflow.
   c. Create a new TreeNode with nums[mid].
   d. Recursively build the left subtree with range (left, mid - 1).
   e. Recursively build the right subtree with range (mid + 1, right).
   f. Return the root node.
4. The recursion will ensure the tree is height-balanced.

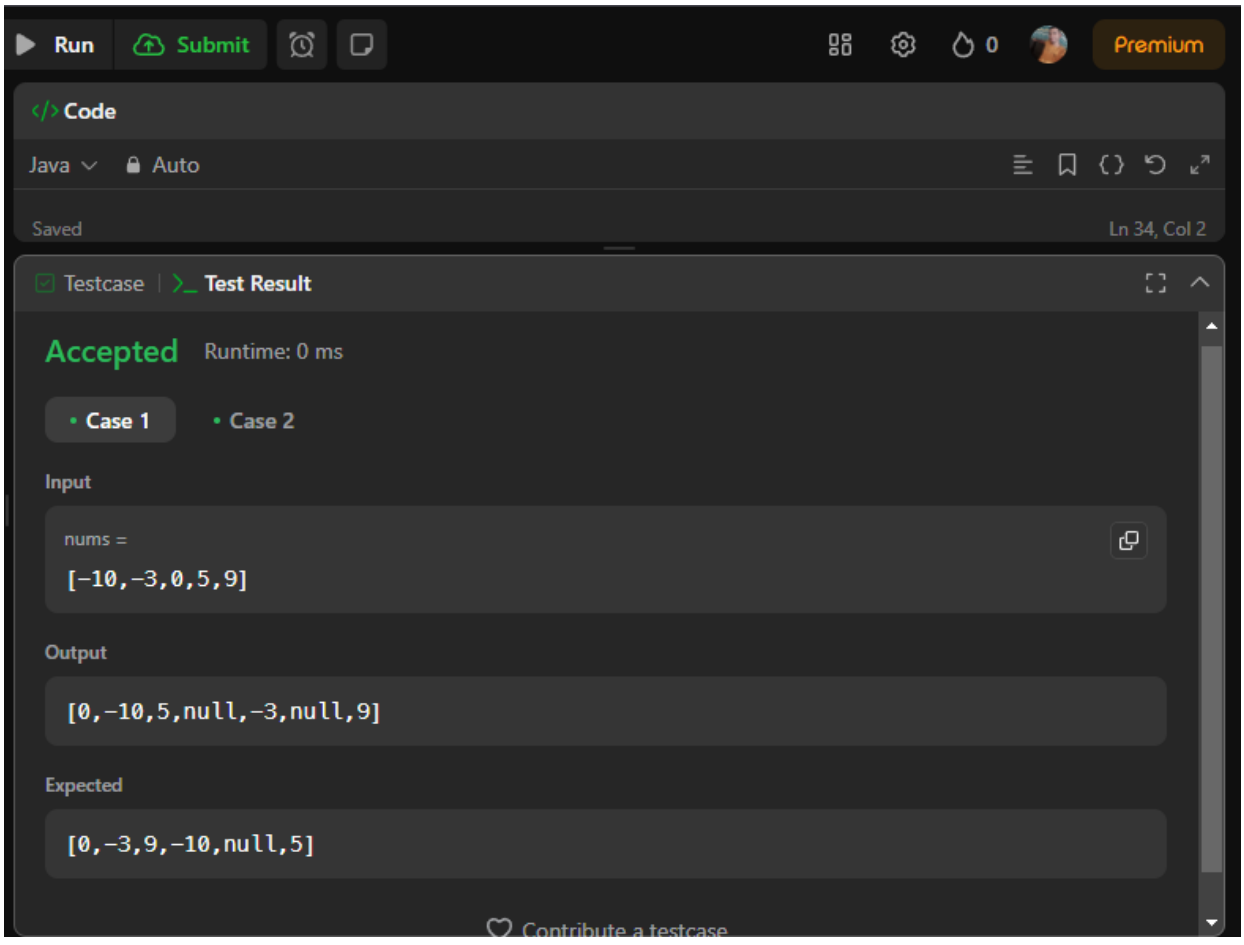
Code:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
```

```
 *     }
 * }
 */

class Solution {
    public TreeNode sortedArrayToBST(int[] nums) {
        return buildBST(nums, 0, nums.length - 1);
    }

    private TreeNode buildBST(int[] nums, int left, int right) {
        if (left > right) {
            return null;
        }

        int mid = left + (right - left) / 2; // to prevent overflow
        TreeNode root = new TreeNode(nums[mid]);
        root.left = buildBST(nums, left, mid - 1);
        root.right = buildBST(nums, mid + 1, right);

        return root;
    }
}
```

Output:


https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/submissions/1554124678/
Time Complexity
$O(N)$
58°F
Mostly cloudy
Search
ENG
IN
11:55 PM
24-02-2025


Run
Submit
Premium
Code
Java
Auto
Saved
Ln 34, Col 2
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
nums =
[-10,-3,0,5,9]
Output
[0,-10,5,null,-3,null,9]
Expected
[0,-3,9,-10,null,5]
Contribute a testcase

Aim:

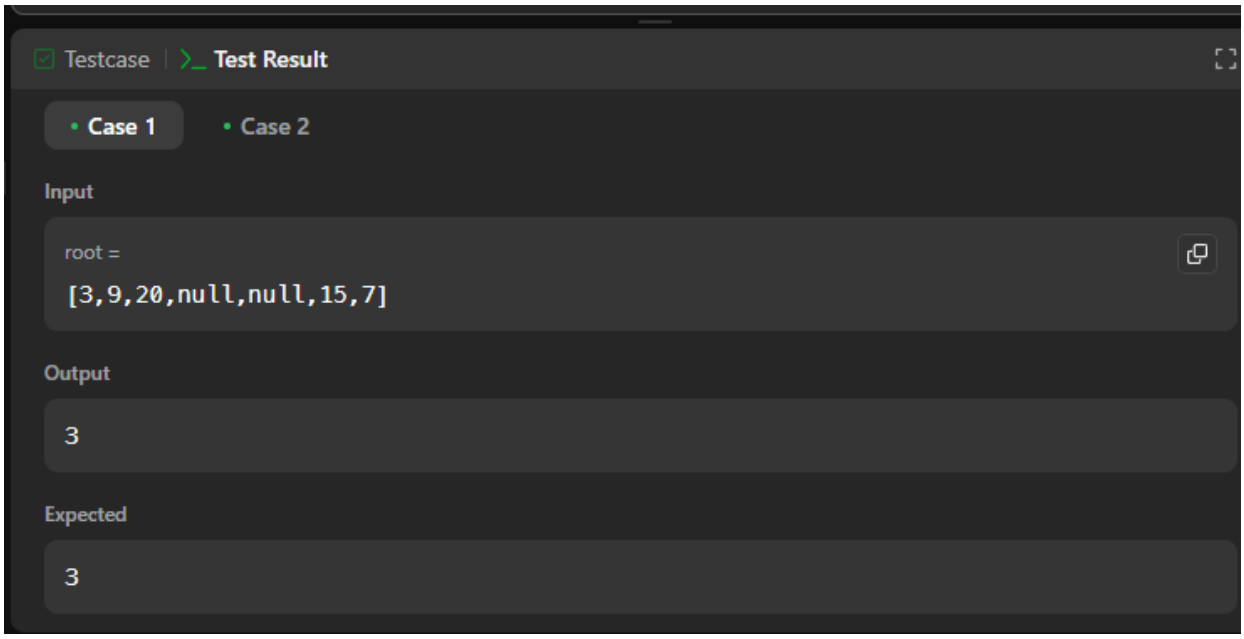
Problem-2: Maximum Depth of Binary Tree

Algorithm :

1. Define a TreeNode class with 'val', 'left', and 'right' attributes.
// 2. Implement a method maxDepth(TreeNode root) that:
// a. If root is null, return 0 (base case).
// b. Recursively find the depth of the left subtree using maxDepth(root.left).
// c. Recursively find the depth of the right subtree using maxDepth(root.right).
// d. Return 1 + maximum of left and right subtree depths.
// 3. The recursion will traverse all nodes, ensuring the longest path is found.

Code :

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
 public class Solution {
    public int maxDepth(TreeNode root) {
        if (root == null) {
            return 0;
        }
        int leftDepth = maxDepth(root.left);
        int rightDepth = maxDepth(root.right);
        return 1 + Math.max(leftDepth, rightDepth);
    }
}
```


Testcase
Test Result
Case 1
Case 2
Input
root =
[3,9,20,null,null,15,7]
Output
3
Expected
3