## NAME: Siddharth Rastogi UID: 22BET10095

**Branch: BE – IT SECTION/GROUP: 22BET\_IOT – 703 (B)**

**th**

# SEMESTER: 6

## AIM: Maximum Depth of Binary Tree CODE:

class Solution

{ public:

int maxDepth(TreeNode\* root)

{ if (!root) return 0;

# SUBJECT CODE: 22ITP – 351

## Problem 1

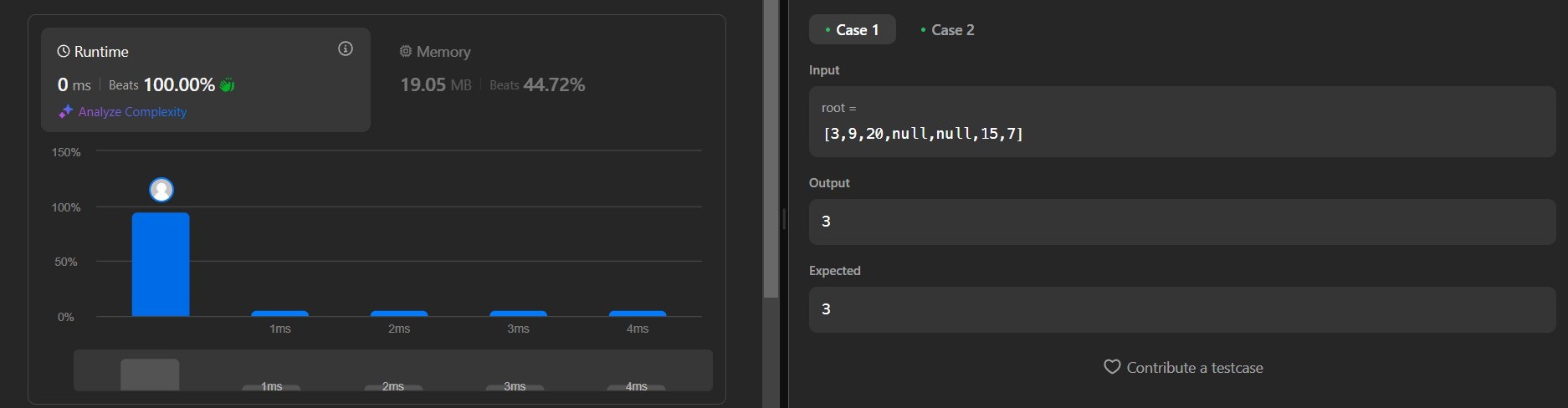
int leftDepth = maxDepth(root->left); int rightDepth = maxDepth(root->right);

return max(leftDepth, rightDepth) + 1;

}

};

# OUTPUT:

****

## Problem 2

**AIM: Validate Binary Search Tree CODE:**

class Solution

{ public:

bool isValidBST(TreeNode\* root, long long minVal = LLONG\_MIN, long long maxVal = LLONG\_MAX) {

if (!root) return true;

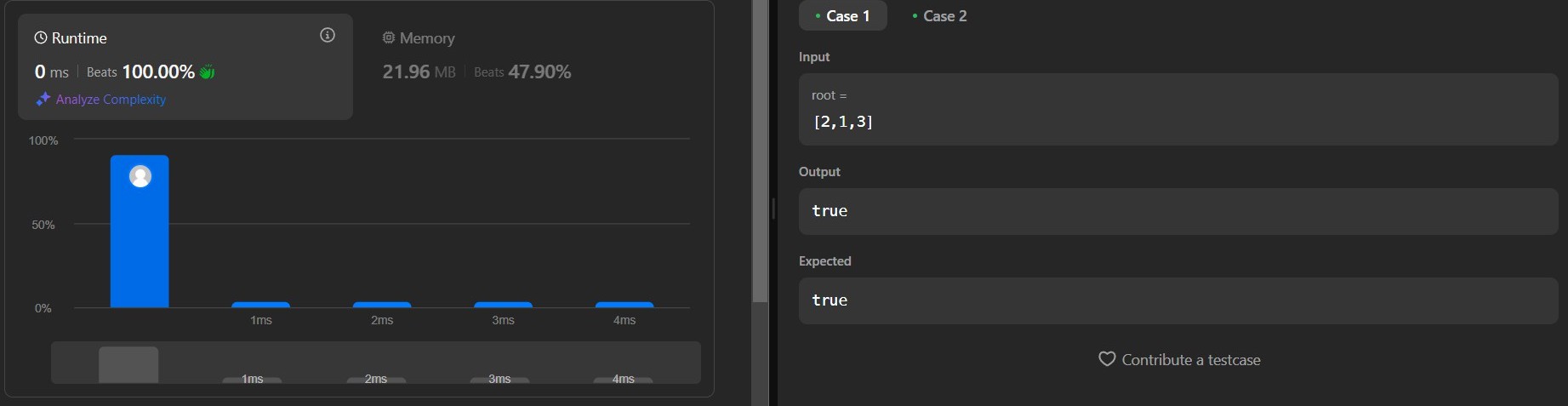
if (root->val <= minVal || root->val >= maxVal) return false;

return isValidBST(root->left, minVal, root->val) && isValidBST(root->right, root->val, maxVal);

}

};

# OUTPUT:

****

## AIM: Symmetric Tree CODE:

class Solution

{ public:

bool isMirrored(TreeNode\* root1,TreeNode\* root2)

{ if(root1==NULL && root2==NULL)return true; else if(root1==NULL || root2==NULL)return false;

return (root1->val==root2->val)&&isMirrored(root1->left,root2-

>right)&&isMirrored(root1->right,root2->left);

}

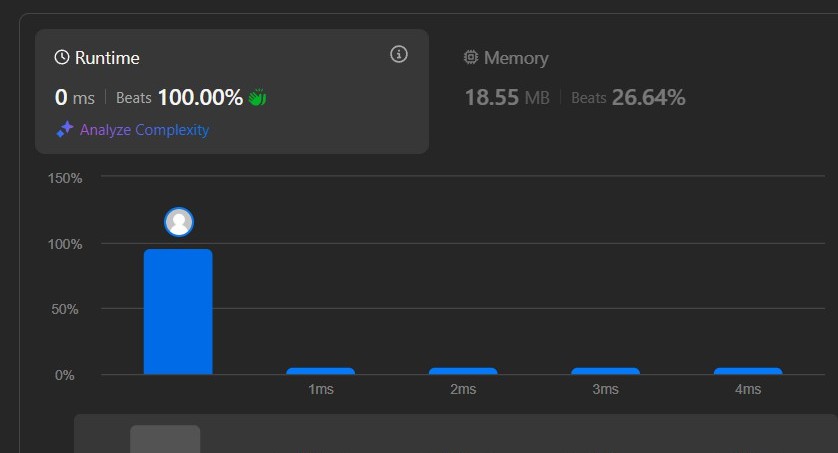
bool isSymmetric(TreeNode\* root)

{ return isMirrored(root,root);

}

};

# OUTPUT:

****

## Problem 4 AIM: Binary Tree Level Order Traversal CODE:

class Solution

{ public:

vector<vector<int>> levelOrder(TreeNode\* root)

{ vector<vector<int>> result; if (!root) return result;

queue<TreeNode\*> q; q.push(root);

while (!q.empty()) {

int levelSize = q.size(); vector<int> level;

for (int i = 0; i < levelSize; ++i)

{ TreeNode\* node = q.front(); q.pop();

level.push\_back(node->val);

if (node->left) q.push(node->left);

if (node->right) q.push(node->right);

}

result.push\_back(level);

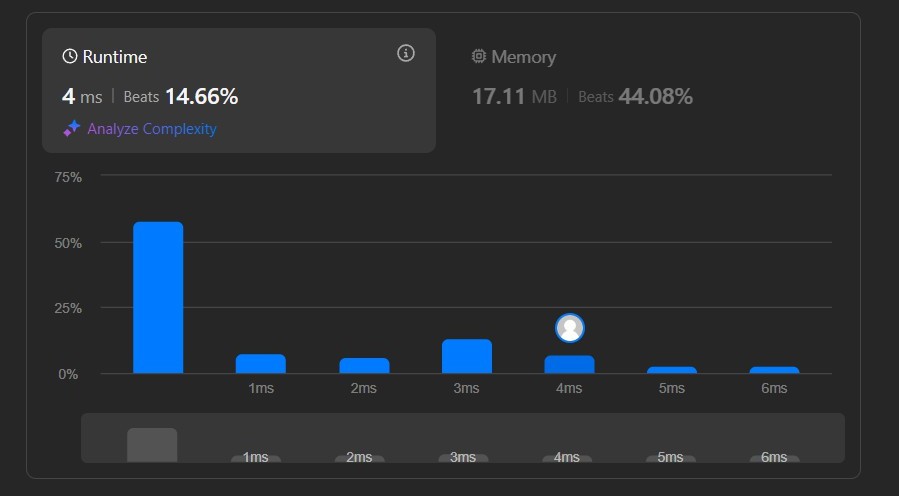
}

return result;

}

};

# OUTPUT:

****

## AIM: Convert Sorted Array to Binary Search Tree CODE:

class Solution

{ public:

TreeNode\* sortedArrayToBST(vector<int>& nums, int left, int right)

{ if (left > right) return nullptr;

int mid = left + (right - left) / 2;

TreeNode\* node = new TreeNode(nums[mid]);

node->left = sortedArrayToBST(nums, left, mid - 1); node->right = sortedArrayToBST(nums, mid + 1, right);

return node;

}

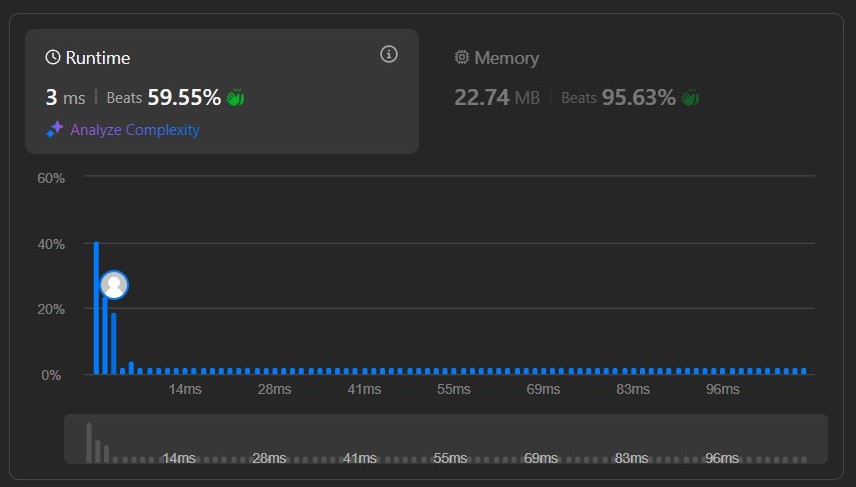
TreeNode\* sortedArrayToBST(vector<int>& nums)

{ return sortedArrayToBST(nums, 0, nums.size() - 1);

}

};

# OUTPUT:

****

## AIM: Binary Tree Inorder Traversal CODE:

class Solution

{ public:

void inorderTraversalHelper(TreeNode\* root, vector<int>& result)

{ if (!root) return; inorderTraversalHelper(root->left, result); result.push\_back(root->val); inorderTraversalHelper(root->right, result);

}

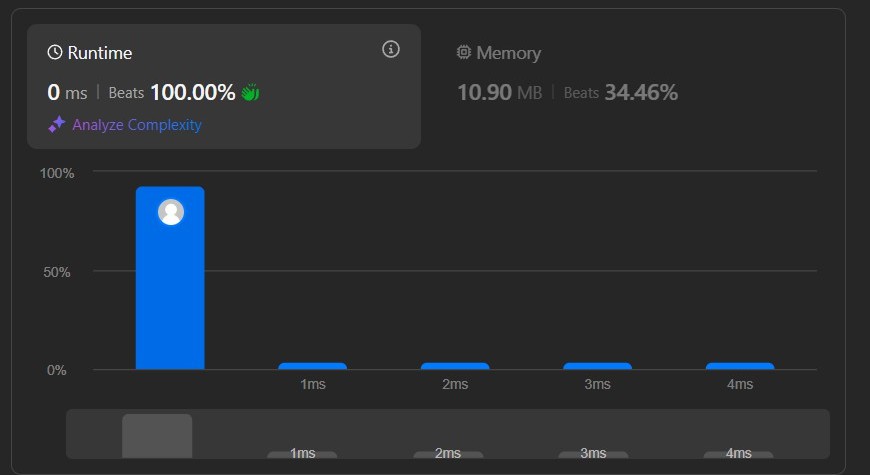
vector<int> inorderTraversal(TreeNode\* root)

{ vector<int> result; inorderTraversalHelper(root, result); return result;

}

};

**OUTPUT:**

****