

Ap-Assignment 5

Name: Harmandeep Singh

UID:22BCS14975

Leetcode Profile: <https://leetcode.com/u/Harman001/>

Ques 1:Max depth of binary trees

```
class Solution {  
  
public:  
  
    int maxDepth(TreeNode* root) {  
  
        if (!root) return 0; // Base case: if the tree is empty  
  
  
        int leftDepth = maxDepth(root->left);  
  
        int rightDepth = maxDepth(root->right);  
  
  
        return max(leftDepth, rightDepth) + 1;  
  
    }  
  
};
```

The screenshot shows a LeetCode submission interface for the problem "Maximum Depth of Binary Tree". The submission is in C++ and has been accepted. The left sidebar displays performance metrics: Runtime is 0 ms (Beats 100.00%) and Memory is 19.03 MB (Beats 45.21%). A bar chart shows the runtime performance relative to other submissions. The main area displays the C++ code for the solution, which uses a recursive approach to calculate the maximum depth of the binary tree. The test case section shows the input as a binary tree with root 3, and the output is 3, which matches the expected result.

```
Accepted 39 / 39 testcases passed  
Har... submitted at Mar 10, 2025 20:09  
Solution  
Runtime  
0 ms | Beats 100.00%  
Analyze Complexity  
Memory  
19.03 MB | Beats 45.21%  
Code | C++  
/**  
 * Definition for a binary tree node.  
 * struct TreeNode {  
 *     int val;  
 *     TreeNode* left;  
 *     TreeNode* right;  
 * };  
 */  
class Solution {  
public:  
    int maxDepth(TreeNode* root) {  
        if (!root) return 0;  
        int leftDepth = maxDepth(root->left);  
        int rightDepth = maxDepth(root->right);  
        return max(leftDepth, rightDepth) + 1;  
    }  
};  
Testcase  
Accepted Runtime: 0 ms  
Case 1 Case 2  
Input  
root =  
[3,9,20,null,null,15,7]  
Output  
3  
Expected
```

Ques2: Validate Binary Search tree

```

class Solution {
public:
    bool isValidBST(TreeNode* root) {
        return validate(root, LONG_MIN, LONG_MAX);
    }

private:
    bool validate(TreeNode* node, long minVal, long maxVal) {
        if (!node) return true;
        if (node->val <= minVal || node->val >= maxVal) return false;
        return validate(node->left, minVal, node->val) && validate(node->right, node->val, maxVal);
    }
};

```

The screenshot displays a coding platform interface for a C++ solution. The top navigation bar includes links for Description, Accepted (86 / 86 testcases passed), Editorial, Solutions, and Submissions. The user's submission is identified as 'Harma...' submitted on Mar 10, 2025 at 20:14. The 'Runtime' section shows a performance of 0 ms, beating 100.00% of other solutions. The 'Memory' section shows 21.92 MB usage, beating 48.51% of other solutions. A bar chart visualizes the runtime performance across different test cases. The 'Code' editor shows the following C++ code:

```

12 class Solution {
13 public:
14     bool isValidBST(TreeNode* root) {
15         return validate(root, LONG_MIN, LONG_MAX);
16     }
17
18 private:
19     bool validate(TreeNode* node, long minVal, long maxVal) {
20         if (!node) return true;
21         if (node->val <= minVal || node->val >= maxVal) return false;
22         return validate(node->left, minVal, node->val) && validate(node->right, node->val, maxVal);
23     }
24 };

```

The 'Testcase' section shows a single test case labeled 'Case 1' with the input 'root = [2,1,3]' and the expected output 'true'. The test result is 'Accepted' with a runtime of 0 ms.

Ques 3: Symmetric tree

```

class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        if (!root) return true;
        return isMirror(root->left, root->right);
    }

private:
    bool isMirror(TreeNode* t1, TreeNode* t2) {
        if (!t1 && !t2) return true;
        if (!t1 || !t2) return false;
        return (t1->val == t2->val) &&
            isMirror(t1->left, t2->right) &&
            isMirror(t1->right, t2->left);
    }
};

```

```
}
};
```

Accepted 200 / 200 testcases passed
Harma... submitted at Mar 10, 2025 20:15

Runtime: 0 ms | Beats 100.00%
Memory: 18.54 MB | Beats 27.46%

```
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        if (!root) return true;
        return isMirror(root->left, root->right);
    }
private:
    bool isMirror(TreeNode* t1, TreeNode* t2) {
        if (!t1 && !t2) return true;
        if (!t1 || !t2) return false;
        return (t1->val == t2->val) &&
            isMirror(t1->left, t2->right) &&
            isMirror(t1->right, t2->left);
    }
};
```

Testcase: Case 1
Input: root = [1,2,2,3,4,4,3]
Output: true
Expected: true

Ques4: Binary tree Zigzag level order traversal

```
class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>> result;
        if (!root) return result;

        queue<TreeNode*> q;
        q.push(root);
        bool leftToRight = true;

        while (!q.empty()) {
            int size = q.size();
            vector<int> level(size);

            for (int i = 0; i < size; i++) {
                TreeNode* node = q.front();
                q.pop();

                int index = leftToRight ? i : (size - 1 - i);
                level[index] = node->val;

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

            result.push_back(level);
            leftToRight = !leftToRight;
        }

        return result;
    }
};
```

```

result.push_back(level);
leftToRight = !leftToRight;
}

return result;
}
};

```

Accepted 33 / 33 testcases passed

Harma... submitted at Mar 10, 2025 20:17

Runtime: 0 ms | Beats 100.00%

Memory: 15.16 MB | Beats 48.57%

Code | C++

```

/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 * };
 */
class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>> result;
        if (!root) return result;

        queue<TreeNode*> q;
        q.push(root);
        bool leftToRight = true;

        while (!q.empty()) {
            int size = q.size();

```

Testcase 1 Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

root =

[3, 9, 20, null, null, 15, 7]

Output

[[3], [20, 9], [15, 7]]

Expected

[[3], [20, 9], [15, 7]]

Ques 5: Lowest Common ancestors of Binary tree

```

class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if (!root || root == p || root == q) return root;
        TreeNode* left = lowestCommonAncestor(root->left, p, q);
        TreeNode* right = lowestCommonAncestor(root->right, p, q);
        if (left && right) return root;
        return left ? left : right;
    }
};

```

Description
Accepted
Editorial
Solutions
Submissions

Accepted 32 / 32 testcases passed

Harma... submitted at Mar 10, 2025 20:19

Editorial
Solution

Runtime
13 ms | Beats 46.27%

Memory
17.35 MB | Beats 67.78%

Code | C++

```

/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 * };

```

C++
Auto

```

9  */
10 class Solution {
11 public:
12     TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
13         if (!root || root == p || root == q) return root;
14
15         TreeNode* left = lowestCommonAncestor(root->left, p, q);
16         TreeNode* right = lowestCommonAncestor(root->right, p, q);
17
18         if (left && right) return root;
19         return left ? left : right;
20     }
21 };

```

Testcase | Test Result

Accepted Runtime: 4 ms

Case 1 Case 2 Case 3

Input

root =
[3,5,1,6,2,0,8,null,null,7,4]

p =
5

q =
1

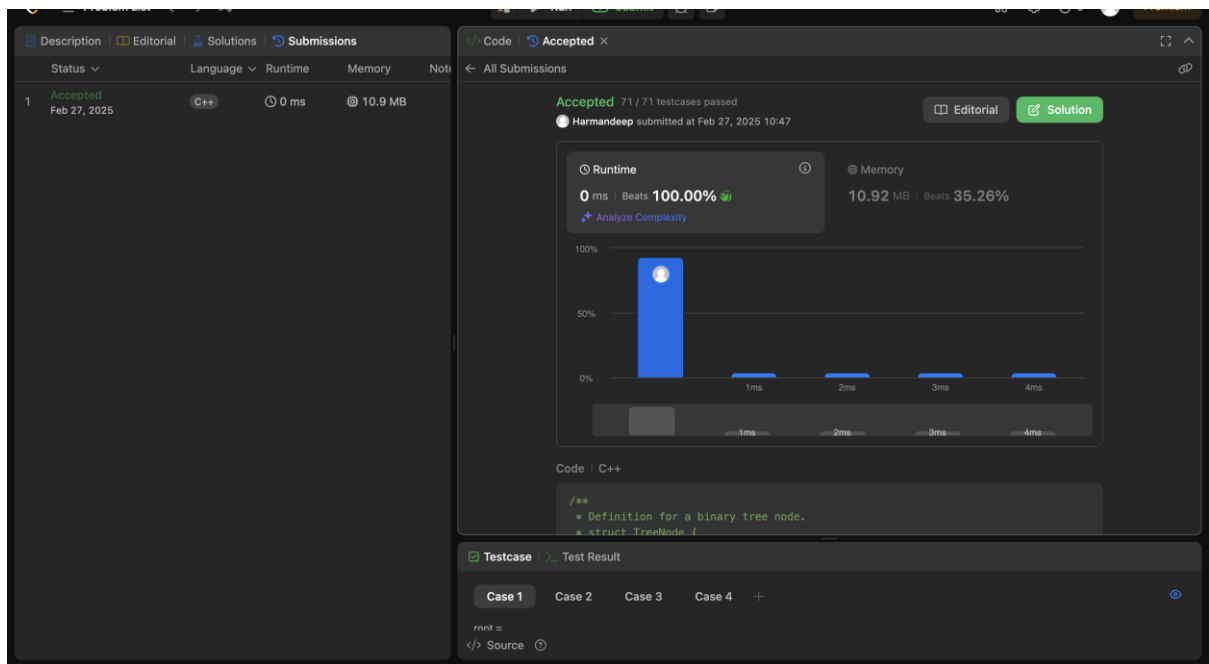
Ques: Binary tree Inorder traversal

```

class Solution {
public:
    void inorder(TreeNode* root, vector<int>& result) {
        if (root == nullptr) return;
        inorder(root->left, result);
        result.push_back(root->val);
        inorder(root->right, result);
    }

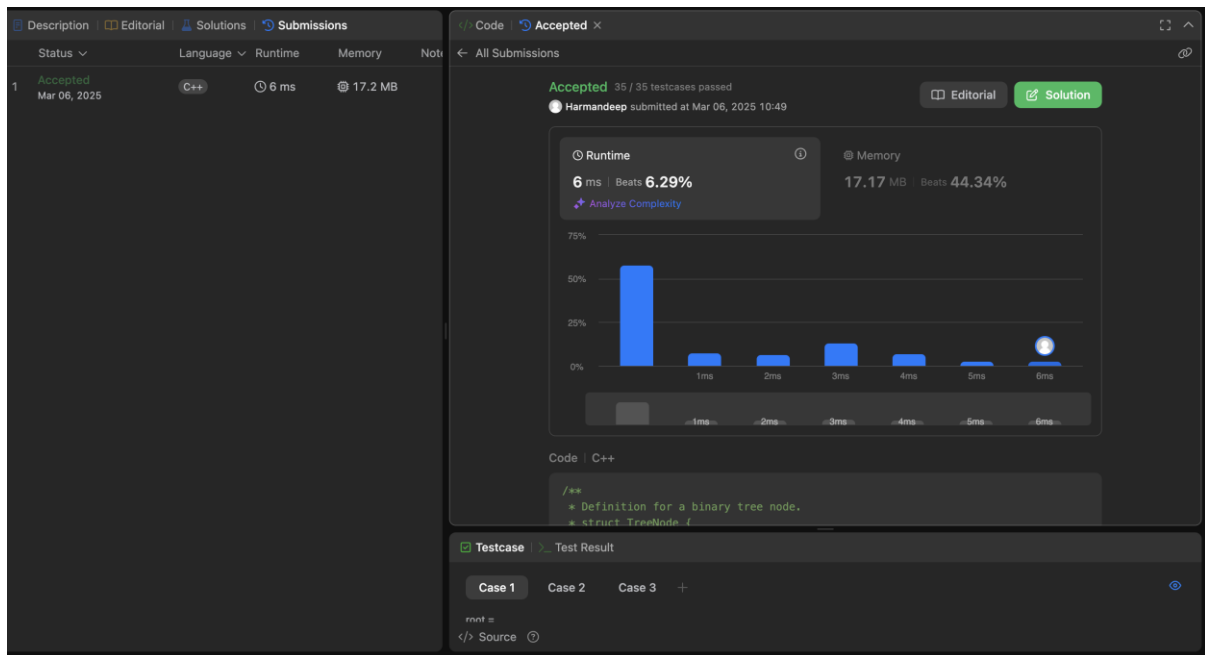
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int> result;
        inorder(root, result);
        return result;
    }
};

```



Ques : Binary tree Level order traversal

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> res;
        if(!root) return res;
        queue<TreeNode*> q;
        q.push(root);
        while(!q.empty()){
            int n=q.size();
            vector<int> ans;
            while(n--){
                TreeNode* temp=q.front();
                q.pop();
                ans.push_back(temp->val);
                if(temp->left) q.push(temp->left);
                if(temp->right) q.push(temp->right);
            }
            res.push_back(ans);
        }
        return res;
    }
};
```



Ques : Population next right Pointer in each node

```
class Solution {
public:
    Node* connect(Node* root) {
        if (!root) return nullptr;

        queue<Node*> q;
        q.push(root);

        while (!q.empty()) {
            int size = q.size();
            for (int i = 0; i < size; ++i) {
                Node* node = q.front();
                q.pop();

                if (i < size - 1) {
                    node->next = q.front();
                }

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

        return root;
    }
};
```

Accepted
59 / 59 testcases passed
Harma... submitted at Mar 10, 2025 20:26

Editorial
Solution

Runtime
11 ms | Beats 83.30%

Memory
19.37 MB | Beats 22.52%

Code | C++

```

/*
 * Definition for a Node.
 */

```

C++
Auto

```

22     if (!root) return nullptr;
23
24     queue<Node*> q;
25     q.push(root);
26
27     while (!q.empty()) {
28         int size = q.size();
29         for (int i = 0; i < size; ++i) {
30             Node* node = q.front();
31             q.pop();
32
33             if (i < size - 1) {
34                 node->next = q.front();
35             }
36
37             if (node->left) q.push(node->left);
38             if (node->right) q.push(node->right);

```

Testcase
Test Result

Accepted
Runtime: 0 ms

Case 1
Case 2

Input

```

root =
[1,2,3,4,5,6,7]

```

Ques :Sum of the left leaves

```

class Solution {
public:
    int sumOfLeftLeaves(TreeNode* root) {
        if (!root) return 0;
        int sum = 0;
        if (root->left && !root->left->left && !root->left->right) {
            sum += root->left->val;
        }
        return sum + sumOfLeftLeaves(root->left) + sumOfLeftLeaves(root->right);
    }
};

```

Accepted
100 / 100 testcases passed
Harma... submitted at Mar 10, 2025 20:28

Editorial
Solution

Runtime
0 ms | Beats 100.00%

Memory
16.13 MB | Beats 58.19%

Code | C++

```

/*
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;

```

C++
Auto

```

18     * };
19
20     class Solution {
21     public:
22         int sumOfLeftLeaves(TreeNode* root) {
23             if (!root) return 0;
24             int sum = 0;
25             if (root->left && !root->left->left && !root->left->right) {
26                 sum += root->left->val;
27             }
28             return sum + sumOfLeftLeaves(root->left) + sumOfLeftLeaves(root->right);
29         }
30     };

```

Testcase
Test Result

Accepted
Runtime: 0 ms

Case 1
Case 2

Input

```

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

```

Output

```

24

```

Expected

```

24

```