

Name: Akshita Sood

UID: 22BCS14993

Batch: FL_IoT 601 'A'

1. Binary Tree Inorder Traversal

```
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root)
    { vector<int> ans; stack<TreeNode*> stack;

        while (root != nullptr || !stack.empty()) {
            while (root != nullptr) {
                stack.push(root);
                root = root->left;
            }
            root = stack.top(), stack.pop();
            ans.push_back(root->val);
            root = root->right;
        }

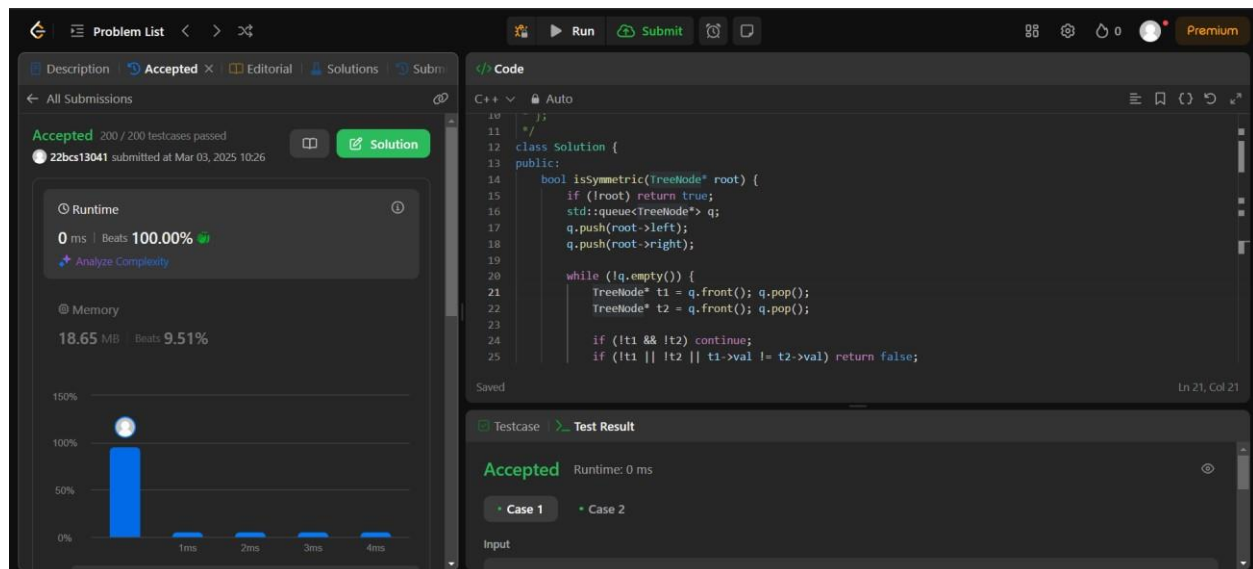
        return ans;
    }
};
```

The screenshot displays a code editor interface for a C++ solution. The left sidebar shows the problem status as 'Accepted' with 71/71 testcases passed, submitted by user 22bcs13041 on Mar 03, 2025 at 10:22. Performance metrics indicate a runtime of 0 ms (Beats 100.00%) and memory usage of 10.79 MB (Beats 88.16%). A bar chart shows the runtime performance across different cases. The main editor displays the C++ code for the inorder traversal using a stack. The bottom right panel shows the test results, confirming the solution is 'Accepted' with a runtime of 0 ms.

```
1 class Solution {
2 public:
3     vector<int> inorderTraversal(TreeNode* root) {
4         vector<int> result;
5         TreeNode* current = root;
6         std::vector<TreeNode*> stack;
7         while (current != nullptr || !stack.empty()) {
8             while (current != nullptr) {
9                 stack.push_back(current);
10                current = current->left;
11            }
12            current = stack.back();
13            stack.pop_back();
14            result.push_back(current->val);
15            current = current->right;
16        }
17        return result;
18    }
19 }
```

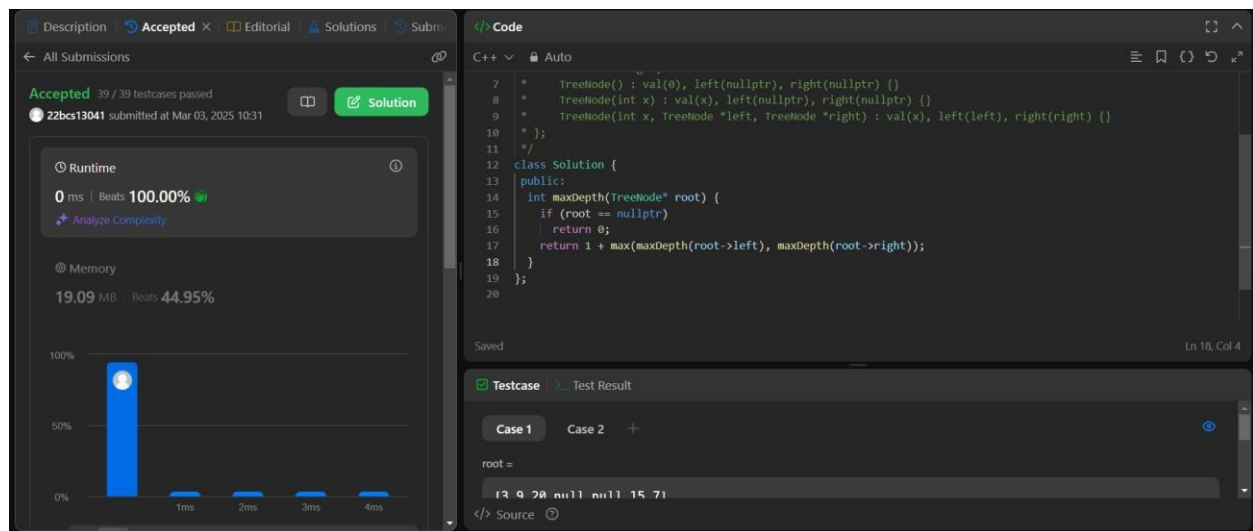
2. Symmetric Tree

```
class Solution { public: bool
isSymmetric(TreeNode* root) {
return isSymmetric(root, root);
} private: bool isSymmetric(TreeNode* p,
TreeNode* q) { if (!p || !q) return
p == q;
return p->val == q->val &&
// isSymmetric(p->left, q->right) &&
// isSymmetric(p->right, q->left);
}
};
```



3. Maximum Depth of Binary Tree

```
class Solution {
public:
    int maxDepth(TreeNode* root)
    { if (root == nullptr) return
0;
return 1 + max(maxDepth(root->left), maxDepth(root->right));
};
```



4. [Validate Binary Search Tree](#)

```
class Solution {
public:
    bool isValidBST(TreeNode* root) {
        return isValidBST(root, nullptr, nullptr);
    }
private:
    bool isValidBST(TreeNode* root, TreeNode* minNode, TreeNode* maxNode) {
        if (root == nullptr)
            return true;
        if (minNode && root->val <= minNode->val)
            return false;
        if (maxNode && root->val >= maxNode->val)
            return false;

        return isValidBST(root->left, minNode, root) &&
            isValidBST(root->right, root, maxNode);
    }
};
```

Accepted

86 / 86 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:32

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

22.00 MB | Beats 48.36%

Code

```

14 bool isValidBST(TreeNode* root) {
15     return isValidBST(root, nullptr, nullptr);
16 }
17 private:
18 bool isValidBST(TreeNode* root, TreeNode* minNode, TreeNode* maxNode) {
19     if (root == nullptr)
20         return true;
21     if (minNode && root->val <= minNode->val)
22         return false;
23     if (maxNode && root->val >= maxNode->val)
24         return false;
25
26     return isValidBST(root->left, minNode, root) &&
27         isValidBST(root->right, root, maxNode);
28 }

```

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Input

5. Kth Smallest Element in a BST

```

class Solution { public:
    int kthSmallest(TreeNode* root, int k) {
const int leftCount = countNodes(root->left);
        if (leftCount == k - 1) return root->val;    if (leftCount >=
k) return kthSmallest(root->left, k);    return kthSmallest(root-
>right, k - 1 - leftCount); // leftCount < k    } private:
    int countNodes(TreeNode* root) {    if (root == nullptr)
return 0;    return 1 + countNodes(root->left) +
countNodes(root->right);
    }
};

```

Accepted

93 / 93 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:36

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

24.46 MB | Beats 42.65%

Code

```

12 class Solution {
13 public:
14     int kthSmallest(TreeNode* root, int k) {
15         const int leftCount = countNodes(root->left);
16
17         if (leftCount == k - 1)
18             return root->val;
19         if (leftCount >= k)
20             return kthSmallest(root->left, k);
21         return kthSmallest(root->right, k - 1 - leftCount); // leftCount < k
22     }
23
24 private:
25     int countNodes(TreeNode* root) {
26         if (root == nullptr)
27             return 0;

```

Testcase

Test Result

Case 1

Case 2

+

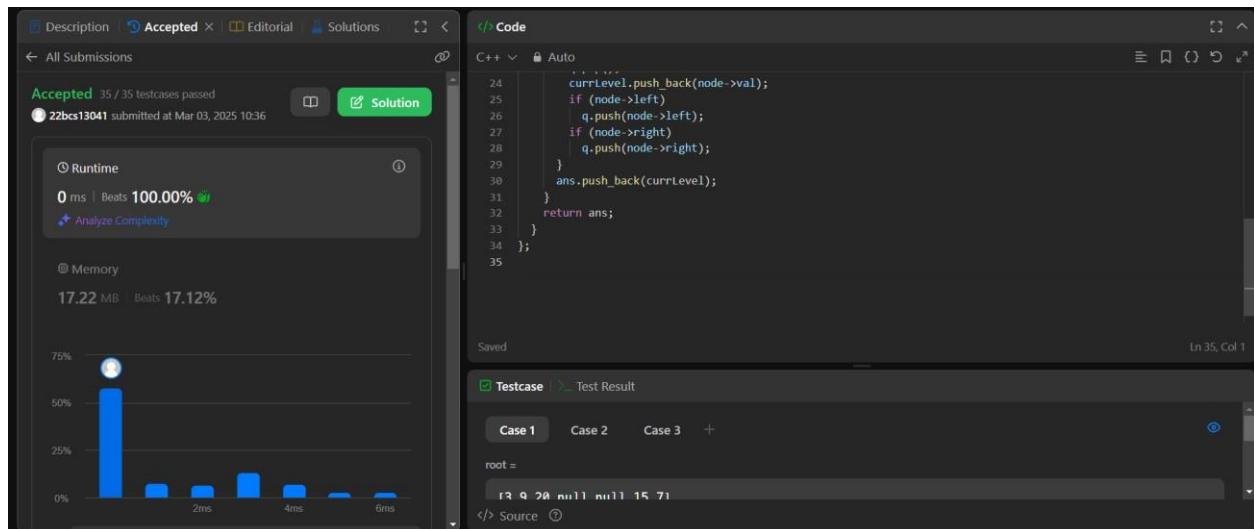
root =

1 3 1 4 null 2 1

Source

6. Binary Tree Level Order Traversal

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
if (root == nullptr)
    return {};
    vector<vector<int>> ans;
    queue<TreeNode*> q{{root}};
    while (!q.empty()) {
        vector<int> currLevel;
        for (int sz = q.size(); sz > 0; --sz) {
            TreeNode* node = q.front();
            q.pop();
            currLevel.push_back(node->val);
            if (node->left)
                q.push(node->left);
            if (node->right)
                q.push(node->right);
        }
        ans.push_back(currLevel);
    }
    return ans;
};
```



7. Binary Tree Level Order Traversal II

```
class Solution { public:
    vector<vector<int>> levelOrderBottom(TreeNode* root) {
    if (root == nullptr)
        return {};

    vector<vector<int>> ans;
    queue<TreeNode*> q{{root}};

    while (!q.empty()) {
    vector<int> currLevel;
        for (int sz = q.size(); sz > 0; --sz) {
    TreeNode* node = q.front();
    q.pop();
            currLevel.push_back(node->val);
    if (node->left)
                q.push(node->left);
    if (node->right)
                q.push(node->right);
        }
        ans.push_back(currLevel);
    }

    ranges::reverse(ans);
    return ans;
    }
};
```

Description
Accepted
Editorial
Solutions
Submissions

All Submissions

Accepted 34 / 34 testcases passed
22bcs13041 submitted at Mar 03, 2025 10:37

Solution

Runtime
0 ms | Beats 100.00%

Memory
15.82 MB | Beats 79.86%

C++
Auto

```

14 q.pop();
15 currLevel.push_back(node->val);
16 if (node->left)
17     q.push(node->left);
18 if (node->right)
19     q.push(node->right);
20 }
21 ans.push_back(currLevel);
22 }
23
24 ranges::reverse(ans);
25 return ans;
26 }
27 };
28

```

Saved

Testcase
Test Result

Case 1 Case 2 Case 3 +

root =

1 3 9 20 null null 15 71

Source

8. [Binary Tree Zigzag Level Order Traversal](#)

```

class Solution { public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        if (root == nullptr)
            return {};

        vector<vector<int>> ans;
        deque<TreeNode*> dq{{root}};
        bool isLeftToRight = true;

        while (!dq.empty()) {
            vector<int> currLevel;    for (int sz
            = dq.size(); sz > 0; --sz)    if
            (isLeftToRight) {
                TreeNode* node = dq.front();
                dq.pop_front();
                currLevel.push_back(node->val);
                if (node->left)
                    dq.push_back(node->left);
                if (node->right)
                    dq.push_back(node->right);
            } else {

```

```

TreeNode* node = dq.back();
dq.pop_back();
currLevel.push_back(node->val);
if (node->right)
    dq.push_front(node->right);
if (node->left)
    dq.push_front(node->left);
}
ans.push_back(currLevel);
isLeftToRight = !isLeftToRight;
}

return ans;
}
};

```

The screenshot displays a code editor interface with a dark theme. On the left, a sidebar shows submission statistics: 'Accepted 33 / 33 testcases passed', '22bcs13041 submitted at Mar 03, 2025 10:39', 'Runtime: 2 ms | Beats: 13.38%', and 'Memory: 15.28 MB | Beats: 18.33%'. The main editor area shows a C++ code snippet. The bottom right panel shows a test case result for 'Case 1' with input 'root = [1, 2, 3, 4, 5, 6, 7]' and output '[1, 2, 3, 4, 5, 6, 7]'.

9. [Binary Tree Right Side View](#)

```

class Solution {
public:
    vector<int> rightSideView(TreeNode* root) {
        if (root == nullptr)
            return {};

        vector<int> ans;
        queue<TreeNode*> q{{root}};

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

```



```

        for (int i = 0; i < size; ++i) {
            TreeNode* node = q.front();
            q.pop();
            if (i == size - 1)
                ans.push_back(node->val);
            if (node->left)
                q.push(node->left);
            if (node->right)
                q.push(node->right);
        }
    }
    return ans;
}
};

```

Accepted 217 / 217 testcases passed
22bcs13041 submitted at Mar 03, 2025 10:40

Runtime
0 ms | Beats 100.00%

Memory
15.36 MB | Beats 13.90%

Code

```

1 class Solution {
2 public:
3     vector<int> rightSideView(TreeNode* root) {
4         if (root == nullptr)
5             return {};
6
7         vector<int> ans;
8         queue<TreeNode*> q{{root}};
9
10        while (!q.empty()) {
11            const int size = q.size();
12            for (int i = 0; i < size; ++i) {
13                TreeNode* node = q.front();
14                q.pop();
15                if (i == size - 1)

```

Testcase Test Result

Case 1 Case 2 Case 3 Case 4 +

root = [1,2,3,null,4,null,5]

10. Construct Binary Tree from Inorder and Postorder Traversal

```

class Solution { public:
    TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {
        unordered_map<int, int> inToIndex;

        for (int i = 0; i < inorder.size(); ++i)
            inToIndex[inorder[i]] = i;

```

```

return build(inorder, 0, inorder.size() - 1, postorder, 0,
            postorder.size() - 1, inToIndex);
}

```

private:

```

TreeNode* build(const vector<int>& inorder, int inStart, int
inEnd,          const vector<int>& postorder, int postStart, int
postEnd,        const unordered_map<int, int>& inToIndex) {
if (inStart > inEnd)
return nullptr;

```

```

const int rootVal = postorder[postEnd];
const int rootInIndex = inToIndex.at(rootVal);
const int leftSize = rootInIndex - inStart;

```

```

TreeNode* root = new TreeNode(rootVal);
root->left = build(inorder, inStart, rootInIndex - 1, postorder, postStart,
                  postStart + leftSize - 1, inToIndex); root->right =
build(inorder, rootInIndex + 1, inEnd, postorder,
      postStart + leftSize, postEnd - 1, inToIndex); return root;
}
};

```

The screenshot shows a code editor with a C++ solution. The code is a recursive function to build a binary tree from inorder and postorder traversals. The left sidebar shows the submission status as 'Accepted' with 202/202 testcases passed. The runtime is 0 ms, beating 100.00% of other solutions. The memory usage is 27.65 MB, beating 18.34% of other solutions. A runtime graph is shown below the memory usage. The right sidebar shows the test results for Case 1, with the input 'inorder = 1 2 3 4 5 6 7' and the output '1 2 3 4 5 6 7'.

11. Find Bottom Left Tree Value

```

class Solution {

```

```

public:

```

```

int findBottomLeftValue(TreeNode* root) {
    queue<TreeNode*> q{{root}};

```

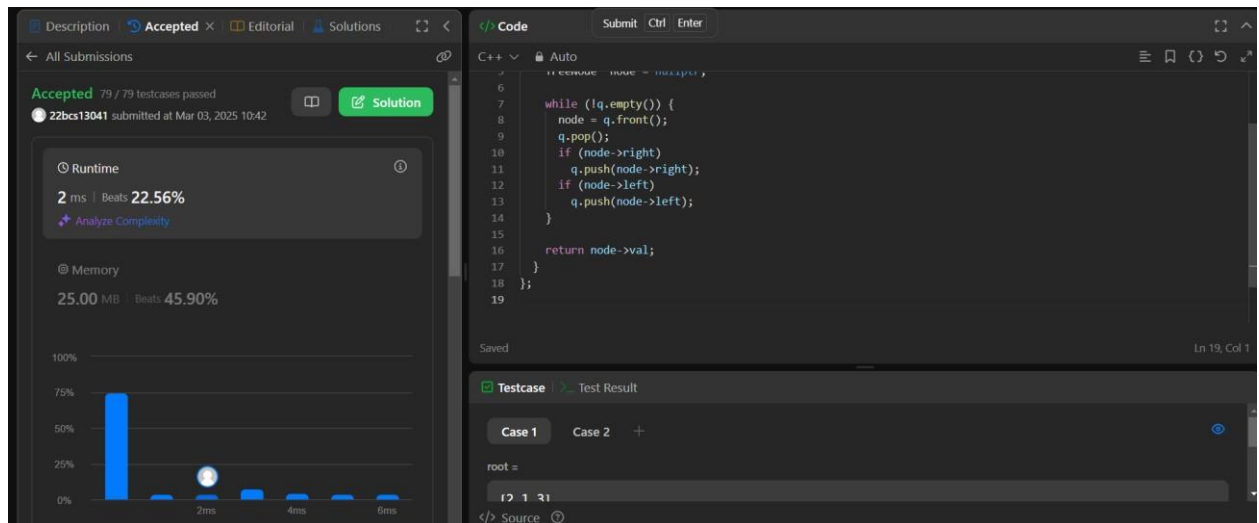
```

TreeNode* node = nullptr;

while (!q.empty()) {
node = q.front();
q.pop();    if (node-
>right)
    q.push(node->right);
if (node->left)
    q.push(node->left);
}

return node->val;
}
};

```



12. Binary Tree Maximum Path Sum

```

class Solution {
public:
    int maxPathSum(TreeNode* root) {
int ans = INT_MIN;
maxPathSumDownFrom(root, ans);
return ans;
}
private:
    int maxPathSumDownFrom(TreeNode* root, int& ans) {
if (root == nullptr) return 0;
const int l = max(0, maxPathSumDownFrom(root->left, ans));
const int r = max(0, maxPathSumDownFrom(root->right, ans));
ans = max(ans, root->val + l + r); return root->val +
max(l, r);
}
}

```

```

} };

```

Accepted 96 / 96 testcases passed
22bcs13041 submitted at Mar 03, 2025 10:43

Runtime: 0 ms | Beats 100.00%
Memory: 28.09 MB | Beats 25.54%

```

5  return ans;
6  }
7  private:
8  int maxPathSumDownFrom(TreeNode* root, int& ans) {
9      if (root == nullptr)
10         return 0;
11
12     const int l = max(0, maxPathSumDownFrom(root->left, ans));
13     const int r = max(0, maxPathSumDownFrom(root->right, ans));
14     ans = max(ans, root->val + l + r);
15     return root->val + max(l, r);
16 }
17 };
18
19

```

Testcase Test Result

Case 1 Case 2 +

root =

1 2 3

13. Vertical Order Traversal of a Binary Tree

```

class Solution {
public:
    vector<vector<int>> verticalTraversal(TreeNode* root) {
        vector<vector<int>> ans;      map<int, multiset<pair<int,
int>>> xToSortedPairs;
        dfs(root, 0, 0, xToSortedPairs);    for
        (const auto& [_ , pairs] : xToSortedPairs) {
            vector<int> vals;          for (const pair<int, int>&
pair : pairs)          vals.push_back(pair.second);
            ans.push_back(vals);
        }
        return ans;
    }
private:
    void dfs(TreeNode* root, int x, int y,          map<int,
multiset<pair<int, int>>>& xToSortedPairs) {    if (root ==
nullptr)          return;
        xToSortedPairs[x].emplace(y, root->val);
        dfs(root->left, x - 1, y + 1, xToSortedPairs);
        dfs(root->right, x + 1, y + 1, xToSortedPairs);
    } };

```

DescriptionAccepted xEditorialSolutions

All Submissions

Accepted 34 / 34 testcases passed
22bcs13041 submitted at Mar 03, 2025 10:44

Runtime

2 ms | Beats 51.53%

Analyze Complexity

Memory

16.56 MB | Beats 17.78%

Time Interval	Percentage
0ms	40%
1ms	10%
2ms	20%
3ms	15%
4ms	10%
5ms	5%
6ms	2%

Code

SubmitCtrlEnter

C++Auto

```
12 auto pair_stack = vector<>();
13 }
14 return ans;
15 }
16 private:
17 void dfs(TreeNode* root, int x, int y,
18         map<int, multiset<pair<int, int>>>& xToSortedPairs) {
19     if (root == nullptr)
20         return;
21     xToSortedPairs[x].emplace(y, root->val);
22     dfs(root->left, x - 1, y + 1, xToSortedPairs);
23     dfs(root->right, x + 1, y + 1, xToSortedPairs);
24 }
25 };
26
```

SavedLn 26, Col 1

TestcaseTest Result

Case 1Case 2Case 3+

root =

1 3 0 7 0 null null 15 71

</> Source