

Name: Akash Pandey

UID: 22BCS11135

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 List' and 'All Submissions' tabs. The main editor area shows the C++ code for the 'inorderTraversal' function. The code is accepted, with a runtime of 0 ms and a memory usage of 10.79 MB. The test result section shows 'Accepted' for Case 1.

**Runtime:** 0 ms | Beats 100.00%

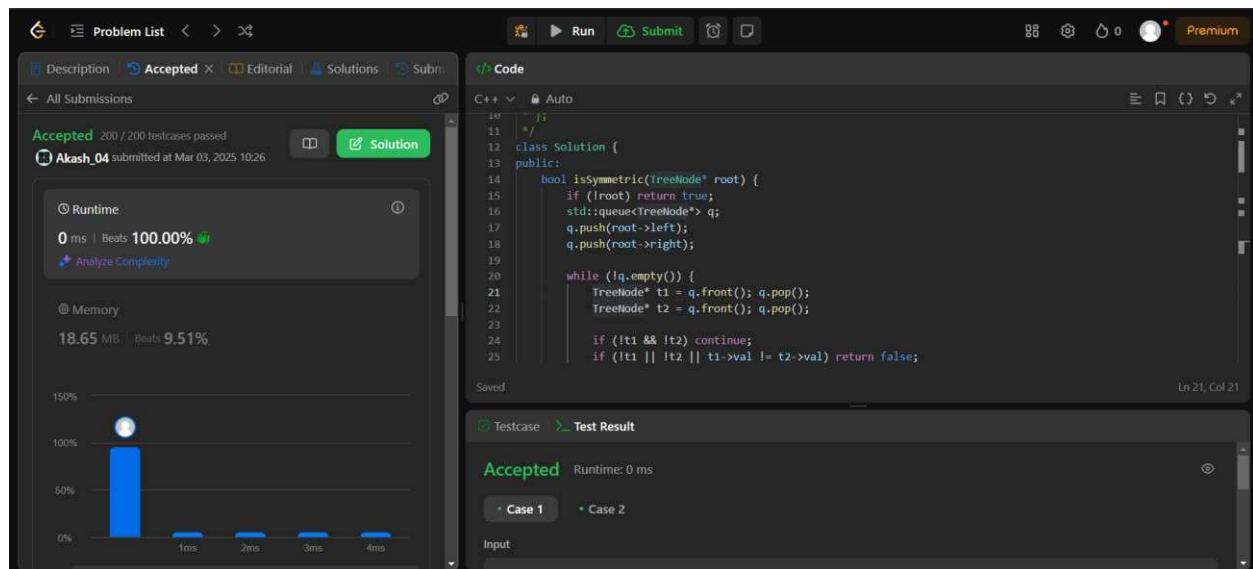
**Memory:** 10.79 MB | Beats 88.16%

**Testcase:** Accepted Runtime: 0 ms

**Case 1:** Input

## 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));
    }
};
```

The screenshot displays a LeetCode submission page for the problem "Maximum Depth of Binary Tree". The submission is by user "Akash\_04" and is marked as "Accepted". The runtime is 0 ms, beating 100.00% of submissions, and the memory usage is 19.09 MB, beating 44.95%. A bar chart shows the runtime performance across different test cases. The code editor on the right shows the following C++ code:

```

7  *   TreeNode() : val(0), left(nullptr), right(nullptr) {}
8  *   TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
9  *   TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
10 *};
11
12 class Solution {
13 public:
14     int maxDepth(TreeNode* root) {
15         if (root == nullptr)
16             return 0;
17         return 1 + max(maxDepth(root->left), maxDepth(root->right));
18     }
19 };
20

```

Below the code editor, the "Testcase" section shows "Test Result" for "Case 1". The input is "root = [3,9,20,null,null,15,7]" and the output is "15 7".

#### 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  
Akash\_04 submitted at Mar 03, 2025 10:32

Runtime: 0 ms | Beats 100.00%  
Memory: 22.00 MB | Beats 48.36%

```

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     return isValidBST(root->left, minNode, root) &&
26            isValidBST(root->right, root, maxNode);
27 }

```

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  
Akash\_04 submitted at Mar 03, 2025 10:36

Runtime: 0 ms | Beats 100.00%  
Memory: 24.46 MB | Beats 42.65%

```

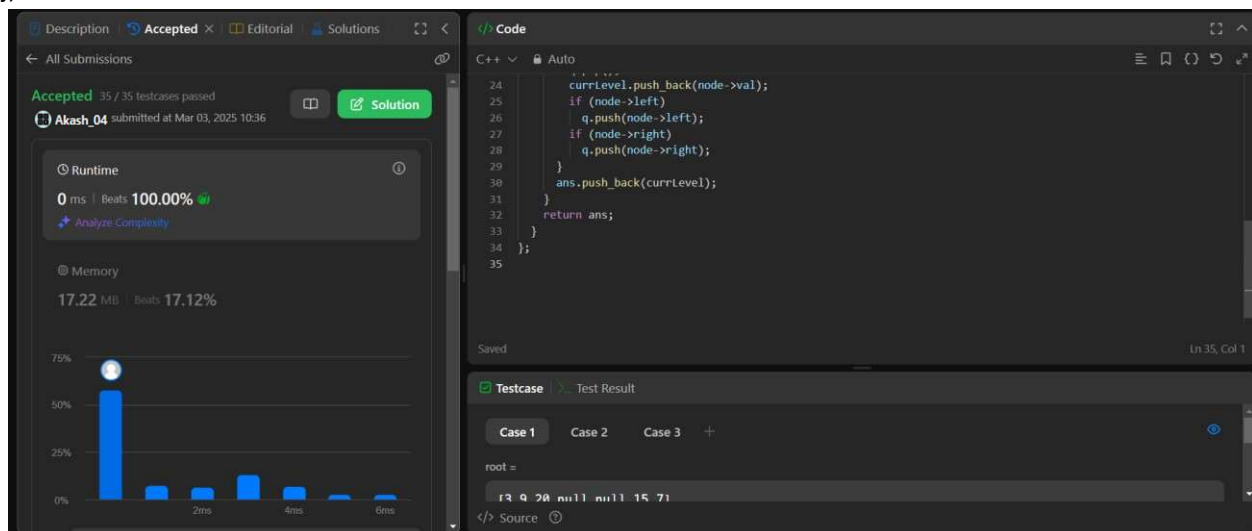
12 class Solution {
13 public:
14     int kthSmallest(TreeNode* root, int k) {
15         const int leftCount = countNodes(root->left);
16
17         if (leftCount == k - 1)
18             return root->val;
19         if (leftCount >= k)
20             return kthSmallest(root->left, k);
21         return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

```

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;
    }
};
```

Accepted 34 / 34 testcases passed  
Akash\_04 submitted at Mar 03, 2025 10:37

**Runtime**  
0 ms | Beats 100.00%  
[Analyze Complexity](#)

**Memory**  
15.82 MB | Beats 79.86%

75%  
50%  
25%  
0%

1ms 2ms 3ms 4ms

**Code**  
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 =

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

</> 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 {

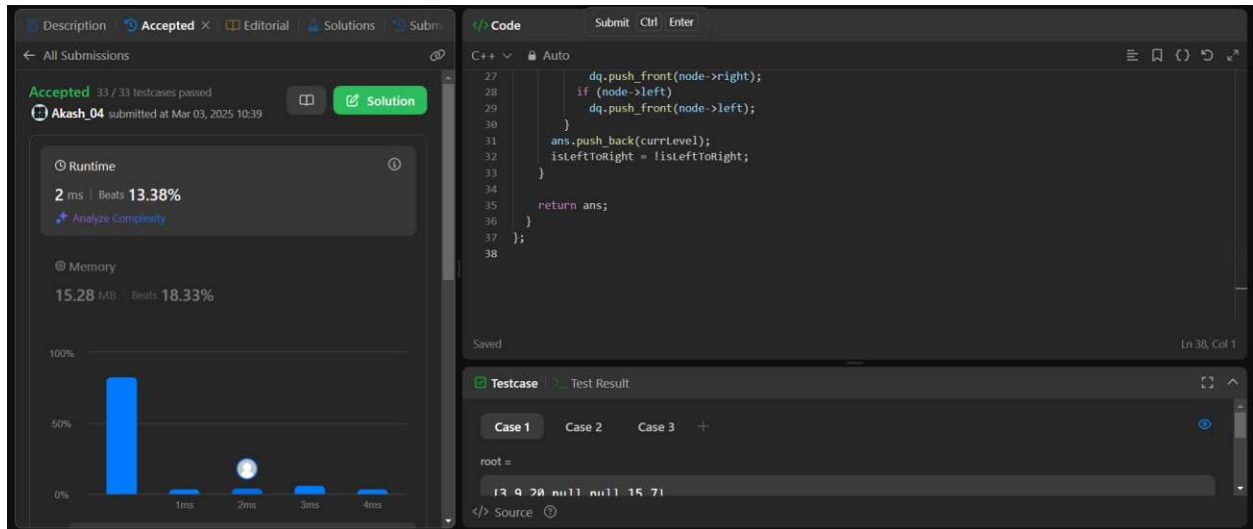
```

```

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;
}
};

```



## 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;
}
};

```

The screenshot displays a code editor interface for a C++ solution. The left sidebar provides submission details: 'Accepted 217 / 217 testcases passed', 'Akash\_04 submitted at Mar 03, 2025 10:40', 'Runtime: 0 ms | Beats 100.00%', and 'Memory: 15.36 MB | Beats 13.90%'. The main editor shows the C++ code for the 'rightSideView' function. The bottom panel shows a 'Testcase' section with 'Case 1' selected, displaying the input 'root = [1, 2, 3, null, 5, null, 4]'.

## 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;
    }
};

```

```

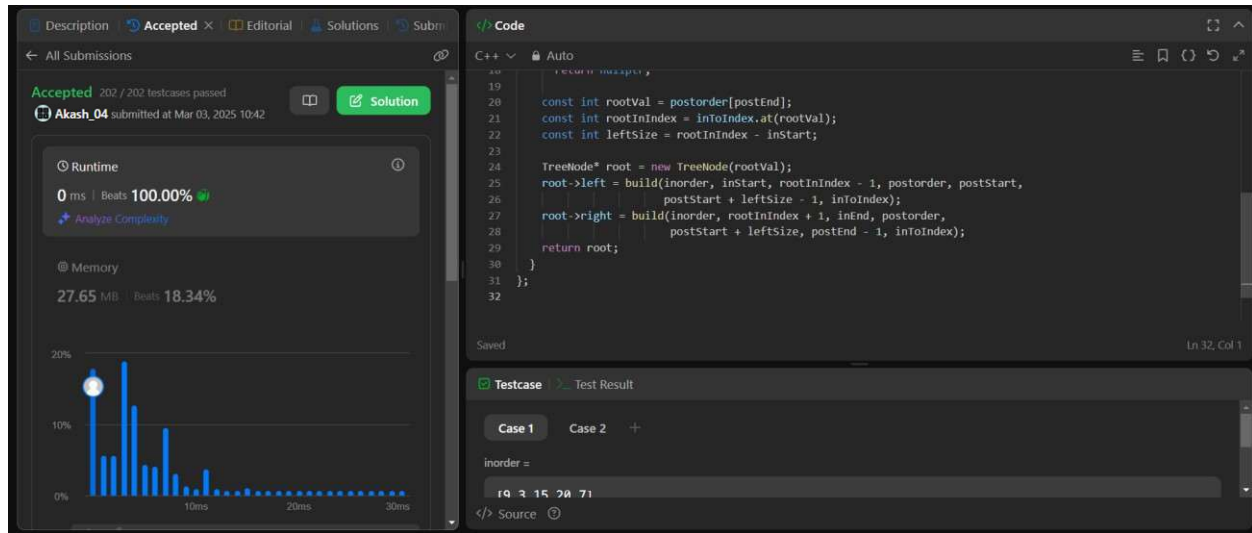
        return build(inorder, 0, inorder.size() - 1, postorder, 0,
                    postorder.size() - 1, inToIndex);
    }

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;
    }
};

```



## 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;
}
};

```

Accepted 79 / 79 testcases passed  
Akash\_04 submitted at Mar 03, 2025 10:42  
Runtime: 2 ms | Beats 22.56%  
Memory: 25.00 MB | Beats 45.90%

```

C++
1 // C++ program to find the maximum path sum in a binary tree
2
3 #include <iostream>
4 #include <queue>
5
6 using namespace std;
7
8 struct TreeNode {
9     int val;
10    TreeNode* left;
11    TreeNode* right;
12 };
13
14 int maxPathSum(TreeNode* root) {
15     if (root == nullptr) return 0;
16     queue<TreeNode*> q;
17     q.push(root);
18     while (!q.empty()) {
19         node = q.front();
20         q.pop();
21         if (node->right)
22             q.push(node->right);
23         if (node->left)
24             q.push(node->left);
25     }
26     return node->val;
27 }
28
29 int main() {
30     // Example usage
31     // ...
32 }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Testcase | Test Result

Case 1 Case 2

root =

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

## 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  
Akash\_04 submitted at Mar 03, 2025 10:43

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

```

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

```

Testcase Test Result

Case 1 Case 2 +

root =

11 2 31

### 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  
Akash\_04 submitted at Mar 03, 2025 10:44


Runtime

2 ms | Beats 51.53%

Analyze Complexity

Memory

16.56 MB | Beats 17.78%



Code

SubmitCtrlEnter

```
12 auto print_ans(ans);
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

Testcase

Test Result

Case 1Case 2Case 3+

root =

1 3 0 7 0 null null 15 71

</> Source