



**CHANDIGARH
UNIVERSITY**
Discover. Learn. Empower.

**NAAC
GRADE A+**
Accredited University

UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE/IT-6th Sem)



Subject Name: Advanced Programming Lab - 2

Subject Code: 22CSP-351

Submitted to:

Mr. Vishal

Submitted by:

Name: Sanya Singh

UID: 23BCS14374

Section: FL_IOT_604

Group: A

ASSIGNMENT-3

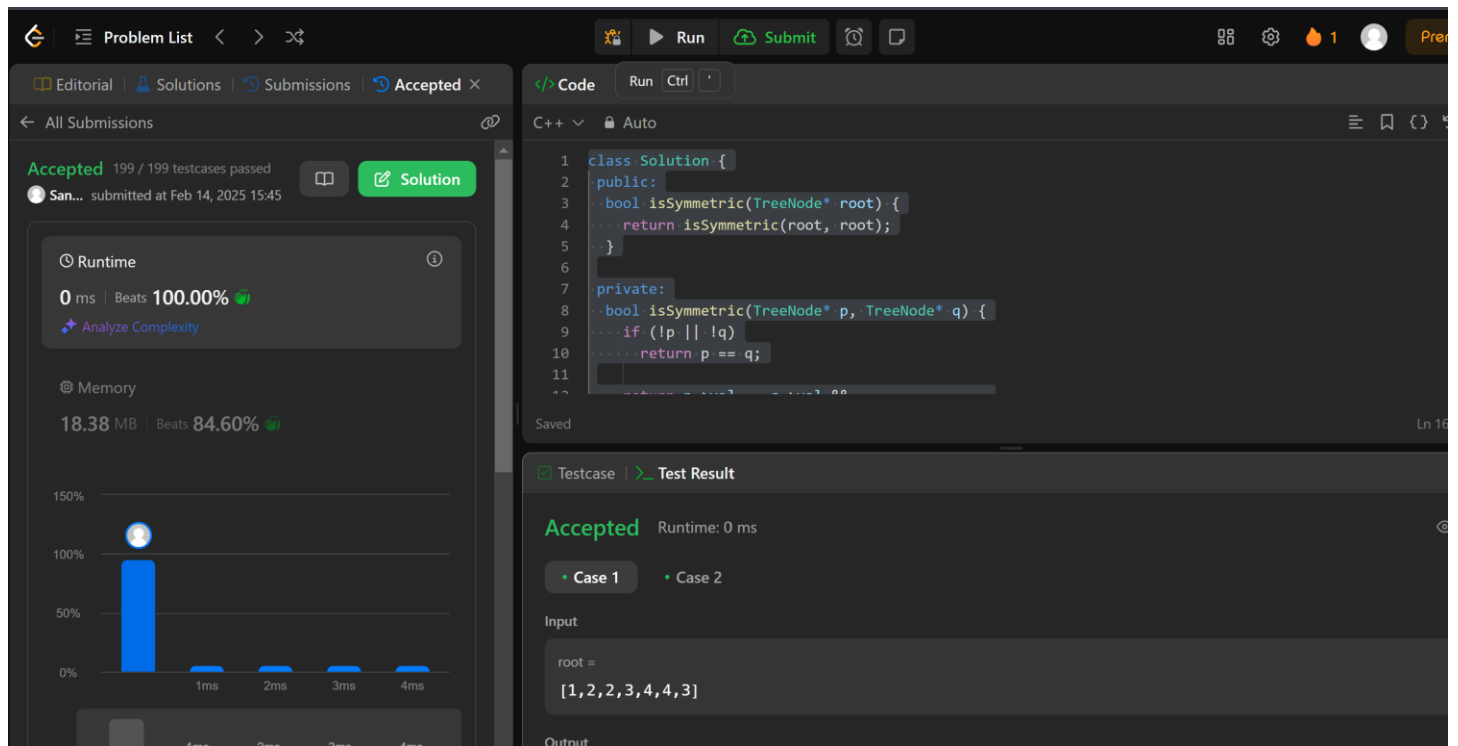
94. [Binary Tree Inorder Traversal](#)

```
class Solution {  
  
public:  
  
    void helper(TreeNode* root,vector<int> &ans){  
  
        if(root==NULL){  
  
            return;  
  
        }  
  
        helper(root->left,ans);  
  
        ans.push_back(root->val);  
  
        helper(root->right,ans);  
  
    }  
  
    vector<int> inorderTraversal(TreeNode* root) {  
  
        vector<int> ans;  
  
        if(root==NULL){  
  
            return ans;  
  
        }  
  
        helper(root,ans);  
  
        return ans;  
  
    }  
};
```

The screenshot displays a coding platform interface. On the left, the 'Problem List' shows the current problem is 'Accepted' with 71/71 testcases passed. Below this, a 'Runtime' panel indicates 0 ms execution time, beating 100.00% of other solutions. A 'Memory' panel shows 10.88 MB usage, beating 65.91%. A bar chart at the bottom left visualizes performance relative to other submissions. The main 'Code' editor shows the C++ solution for Binary Tree Inorder Traversal. On the right, the 'Test Result' panel shows the solution is 'Accepted' with a runtime of 0 ms. It lists four test cases, all of which passed. The input for Case 1 is 'root = [1,null,2,3]' and the output is '[1,3,2]', which matches the expected result.

101. [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);  
  
    }  
  
};
```



104. [Maximum Depth of Binary Tree](#)

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

The screenshot displays a coding platform interface with a dark theme. On the left, the 'All Submissions' panel shows the solution is 'Accepted' with 39/39 testcases passed, submitted at Jan 29, 2025 15:52. Performance metrics are shown: Runtime is 0 ms (Beats 100.00%) and Memory is 18.93 MB (Beats 75.60%). A bar chart compares the user's performance to others. The main editor shows the C++ code for the 'Maximum Depth of Binary Tree' problem. The code defines a 'TreeNode' struct and a 'Solution' class with a 'maxDepth' method. The 'Testcase' panel on the right shows 'Case 1' is 'Accepted' with a runtime of 0 ms. The input for Case 1 is 'root = [3,9,20,null,null,15,7]'.

98. [Validate Binary Search Tree](#)

```
class Solution {  
  
    bool isPossible(TreeNode* root, long long l, long long r){  
  
        if(root == nullptr) return true;  
  
        if(root->val < r and root->val > l)  
  
            return isPossible(root->left, l, root->val) and
```

```

        isPossible(root->right, root->val, r);

    else return false;

}

public:

    bool isValidBST(TreeNode* root) {

        long long int min = -1000000000000, max = 1000000000000;

        return isPossible(root, min, max);

    }

};

```

```

9  *   TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
10 * };
11 */
12 class Solution {
13
14 bool isPossible(TreeNode* root, long long l, long long r){
15     if(root == nullptr) return true;
16     if(root->val < l and root->val > r)
17         return isPossible(root->left, l, root->val) and
18                 isPossible(root->right, root->val, r);
19     else return false;
20 }
21
22 public:
23     bool isValidBST(TreeNode* root) {
24         long long int min = -1000000000000, max = 1000000000000;
25         return isPossible(root, min, max);
26     }
27 };

```

230. [Kth Smallest Element in a BST](#)

```

class Solution {

public:

    pair<int,int> kthSmall(TreeNode* root,int k){

        if(root == NULL) return make_pair(-1,0);

        pair<int,int> pr = kthSmall(root->left,k);

        if(pr.first!=-1) return pr;

        else k -= pr.second;
    }
};

```

```

        if(k==1) {
            pr.first = root->val;
            return pr;
        }

        pair<int,int> p2 = kthSmall(root->right,k-1);
        p2.second += 1 + pr.second;
        return p2;
    }

    int kthSmallest(TreeNode* root, int k) {
        return kthSmall(root,k).first;
    }
};

```

The screenshot displays a C++ IDE interface. On the left, the 'All Submissions' panel shows 'Accepted 93 / 93 testcases passed' and a runtime of '0 ms'. The main editor shows the C++ code for the kthSmallest function. The bottom right panel shows 'Test Result' with 'Accepted' and 'Runtime: 0 ms'.

```

class Solution {
public:
    pair<int,int> kthSmall(TreeNode* root,int k){
        if(root == NULL) return make_pair(-1,0);
        pair<int,int> pr = kthSmall(root->left,k);
        if(pr.first!=-1) return pr;
        else k -= pr.second;
        if(k==1) {
            pr.first = root->val;
            return pr;
        }
        pair<int,int> p2 = kthSmall(root->right,k-1);
        p2.second += 1 + pr.second;
        return p2;
    }
};

```

102. [Binary Tree Level Order Traversal](#)

```

class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> result;

        if(root==NULL){
            return result;
        }
    }
};

```

```

    }

    queue<TreeNode*> q;

    q.push(root);

    while(!q.empty()){

        int l=q.size();

        vector<int> ans;

        while(l>0){

            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);

            }

            l--;

        }

        result.push_back(ans);

    }

    return result;
}

```

The screenshot displays a C++ code editor with a submission for a problem. The code is a recursive function to serialize a binary tree. The submission is accepted, with 0ms runtime and 17.26 MB memory. A bar chart shows the runtime distribution. The test result shows the input root = [3, 9, 20, null, null, 15, 7] and the output [[3], [9, 20], [15, 7]].

Code:

```

28         if(temp->left){
29             q.push(temp->left);
30         }
31         if(temp->right){
32             q.push(temp->right);
33         }
34         l--;
35     }
36     result.push_back(ans);
37 }

```

Runtime: 0 ms | Beats 100.00%

Memory: 17.26 MB | Beats 16.68%

Testcase: Accepted Runtime: 0 ms

Case 1:

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

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

107. [Binary Tree Level Order Traversal II](#)

```
class Solution {  
  
public:  
  
    vector<vector<int>> levelOrderBottom(TreeNode* root) {  
  
        if (!root) return { };  
  
        vector<vector<int>> result;  
  
        queue<TreeNode*> q;  
  
        q.push(root);  
  
        while (!q.empty()) {  
  
            int size = q.size();  
  
            vector<int> level;  
  
            for (int i = 0; i < size; ++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);  
  
        }  
  
        reverse(result.begin(), result.end());  
  
        return result;  
  
    }  
  
};
```


Accepted 34 / 34 testcases passed
 submitted at Feb 14, 2025 22:28

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

Memory
 16.06 MB | Beats 25.38%

75%
 50%
 25%
 0%

1ms 2ms 3ms 4ms

```

1  /**
2   * Definition for a binary tree node.
3   * struct TreeNode {
4   *     int val;
5   *     TreeNode *left;
6   *     TreeNode *right;
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      vector<vector<int>> levelOrderBottom(TreeNode* root) {
15          if (!root) return {};

```

Testcase | Test Result
Accepted Runtime: 0 ms
 Case 1 Case 2 Case 3
 Input
 root =
 (3, 9, 20, null, null, 15, 7)

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

```

class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>> result;
        if(root==NULL){
            return result;
        }
        queue<TreeNode*> q;
        q.push(root);

        bool check=true;
        while(!q.empty()){
            int l=q.size();
            vector<int> ans(l);
            for(int i=0;i<l;i++){
                TreeNode* temp=q.front();
                q.pop();

```

```

int index;

if(check==true){

    index=i;

}

else{

    index=l-i-1;

}

ans[index]=temp->val;

if(temp->left){

    q.push(temp->left);

}

if(temp->right){

    q.push(temp->right);}}

check=!check;

result.push_back(ans);

}

return result;}}

```

The screenshot displays a code editor interface for a C++ solution. The code is a recursive function that traverses a binary tree and pushes node values into a queue. The interface includes a 'Problem List' tab, a 'Runtime' section showing 2 ms and 13.29% beats, a 'Memory' section showing 15.04 MB and 77.16% beats, and a 'Test Result' section showing 'Accepted' status for three test cases. The input for Case 1 is 'root = [3,9,20,null,null,15,7]'.

Code Editor:

```

40
41
42     if(temp->right){
43         q.push(temp->right);
44     }
45
46 }
47 check=!check;
48 result.push_back(ans);
49 }
50 return result;
51

```

Runtime: 2 ms | Beats 13.29%

Memory: 15.04 MB | Beats 77.16%

Test Result: Accepted Runtime: 0 ms

Case 1: Input: root = [3,9,20,null,null,15,7]

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

```
class Solution {  
  
public:  
  
    vector<int> rightSideView(TreeNode* root) {  
  
        vector<int> result;  
  
        if(root==NULL){  
            return result;  
        }  
  
        queue<TreeNode*> q;  
        q.push(root);  
        while(!q.empty()){  
            int n=q.size();  
  
            for(int i=0;i<n;i++){  
                TreeNode* temp = q.front();  
                q.pop();  
                if(i == n-1) {  
                    result.push_back(temp->val);  
                }  
                if(temp->left){  
                    q.push(temp->left);  
                }  
                if(temp->right){  
                    q.push(temp->right);  
                }  
            }  
  
        }  
  
        return result;  
    }  
}
```

};

The screenshot displays a coding platform interface. On the left, the 'Accepted' status is confirmed with 216/216 testcases passed. The runtime is 0 ms (100.00% beats) and memory usage is 14.44 MB (99.99% beats). The right panel shows the C++ code for a binary tree node definition and a rightSideView function. The test result panel shows 'Accepted' with 0 ms runtime and four test cases passed.

106. [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 displays a C++ IDE interface. On the left, a sidebar shows the problem status as 'Accepted' with a runtime of 3 ms and memory of 27.43 MB. The main editor shows the C++ code for a binary tree node and a solution class. The bottom right shows the test result as 'Accepted' with a runtime of 0 ms.

```

2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
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     TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {
15         unordered_map<int, int> inToIndex;
16         for (int i = 0; i < inorder.size(); ++i)

```

513. [Find Bottom Left Tree Value](#)

```

class Solution {
public:

    void tt(TreeNode* root, int level, vector<vector<int>>&nums){

        if(root==NULL){

            return;

        }

        if(nums.size()<=level){

            nums.push_back({});

        }

        nums[level].push_back(root->val);

```

```

        tt(root->right,level+1,nums);

        tt(root->left,level+1,nums);

    }

    int findBottomLeftValue(TreeNode* root) {

        vector<vector<int>>>nums;

        tt(root,0,nums);

        return nums.back().back();

    }

};

```

The screenshot shows a C++ solution on a coding platform. The left sidebar displays the submission status as 'Accepted' with a runtime of 13 ms (Beats 5.78%) and memory usage of 27.50 MB (Beats 8.86%). A bar chart below the memory usage shows the distribution of memory usage across different time intervals. The main editor shows the C++ code for a binary tree node definition and a solution class. The test result section shows 'Accepted' with a runtime of 0 ms for two test cases.

```

1  /**
2   * Definition for a binary tree node.
3   * struct TreeNode {
4   *     int val;
5   *     TreeNode *left;
6   *     TreeNode *right;
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      void tt(TreeNode* root, int level, vector<vector<int>>&nums){
15          if(root==NULL){

```

124. [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);  
  
}  
};
```

The screenshot displays a code editor interface. On the left, a sidebar shows the 'Problem List' and 'Accepted' status. The main editor area shows the C++ code for the function `maxPathSumDownFrom`. The code is as follows:

```
16 maxPathSumDownFrom(root, ans);  
17 return ans;  
18 }  
19  
20 private:  
21 int maxPathSumDownFrom(TreeNode* root, int& ans) {  
22     if (root == nullptr)  
23         return 0;  
24     const int l = max(0, maxPathSumDownFrom(root->left, ans));  
25     const int r = max(0, maxPathSumDownFrom(root->right, ans));  
26     ans = max(ans, root->val + l + r);  
27     return root->val + max(l, r);  
28 }  
29 };
```

Below the code editor, the 'Testcase' tab shows 'Accepted' status with a runtime of 0 ms. The 'Runtime' tab shows a performance of 15 ms and 5.10% beats. A memory usage graph is also visible, showing a peak of 27.93 MB and 50.69% beats.

987. [Vertical Order Traversal of a Binary Tree](#)

```
class Solution {  
  
public:  
  
    vector<vector<int>> verticalTraversal(TreeNode* root) {  
  
        vector<vector<int>> ans;  
  
  
        queue<pair<TreeNode*,int>> Q; // node and col  
  
        Q.push({root,0});  
  
        int depth=0;
```

```

while(!Q.empty()){
    int s=Q.size();
    while(s--){
        auto [node,col]=Q.front();
        Q.pop();
        ans.push_back({ col,depth,node->val });
        if(node->left!=nullptr) Q.push({ node->left,col-1 });
        if(node->right!=nullptr) Q.push({ node->right,col+1 });
    }
    depth++;
}

```

```

sort(ans.begin(),ans.end());

```

```

vector<vector<int>>> final;

```

```

vector<int> temp;

```

```

int curr=ans[0][0];

```

```

for(int i=0;i<ans.size();i++){
    if(ans[i][0]==curr) temp.push_back(ans[i][2]);
    else{
        final.push_back(temp);
        temp.clear();
        curr=ans[i][0];
        temp.push_back(ans[i][2]);
    }
}

```

```

if(!temp.empty()) final.push_back(temp);

```

```

return final;

```

```

}

```


};

Problem List

Description

Accepted

Editorial

Solutions

All Submissions

Accepted 34 / 34 testcases passed

San... submitted at Feb 14, 2025 22:28

Solution

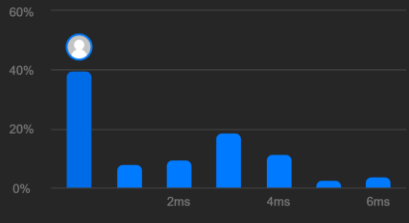
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

15.49 MB | Beats 92.56%



Runtime Range (ms)	Percentage
0 - 1	~45%
1 - 2	~10%
2 - 3	~10%
3 - 4	~18%
4 - 5	~12%
5 - 6	~5%

Code

C++

Auto

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
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     vector<vector<int>> verticalTraversal(TreeNode* root) {
15         vector<vector<int>> ans;
```

Saved

Ln 50

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

root =