

ASSIGNMENT-3

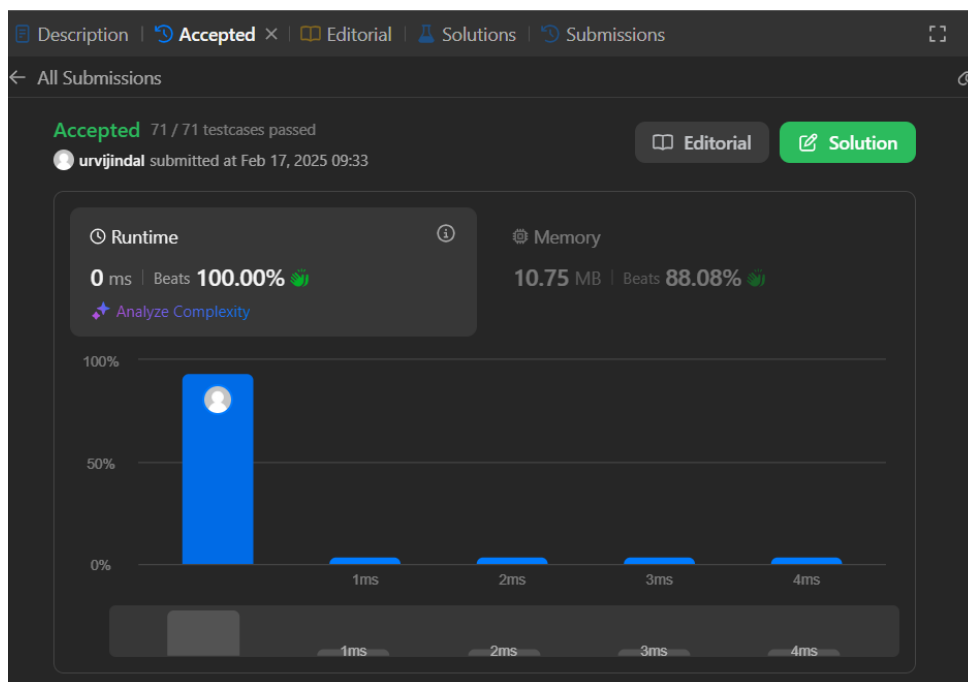
Name: Urvi Jindal

Section: FL_IOT-603/A

UID: 22BCS14860

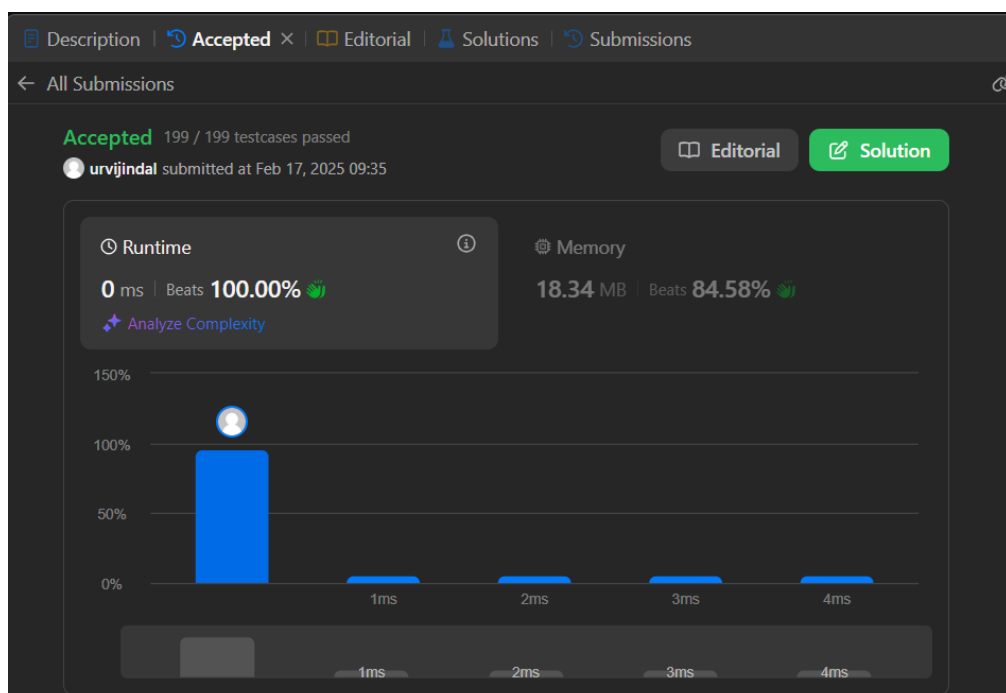
1) binary-tree-inorder-traversal

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



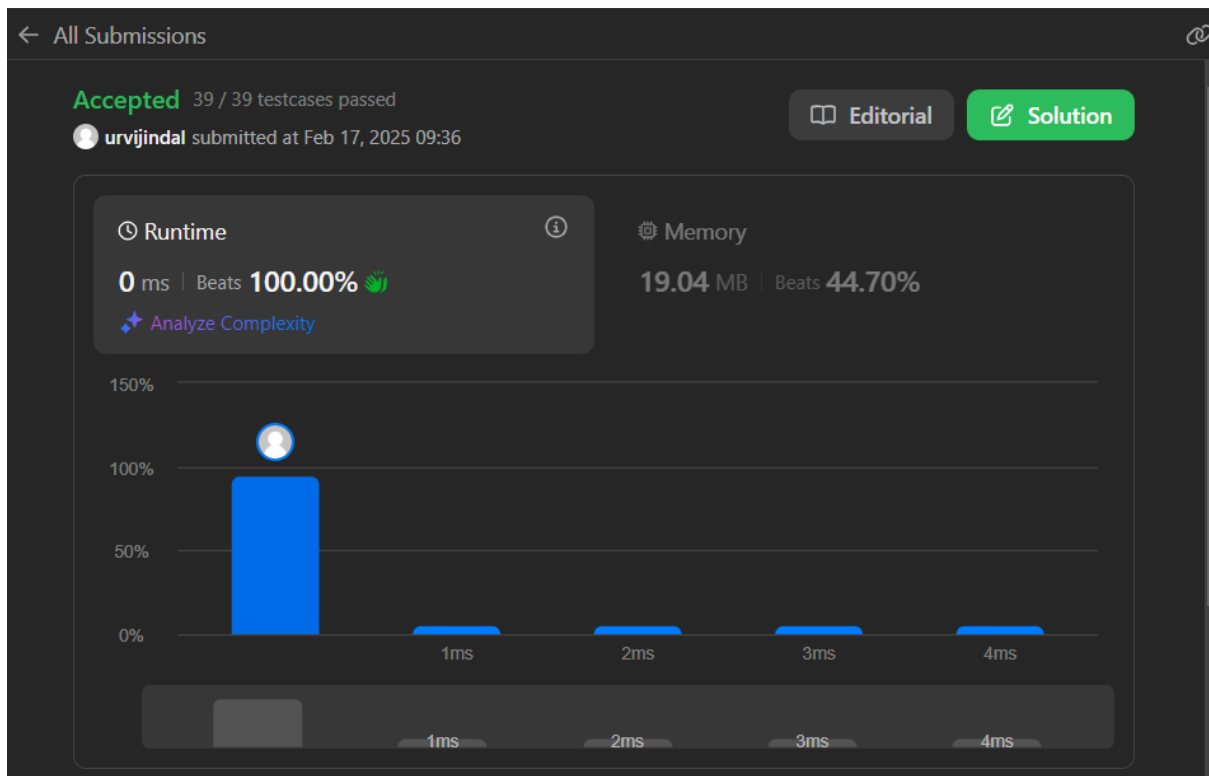
2) symmetric-tree

```
bool mirrorcheck(TreeNode* p, TreeNode* q) {  
    if (p == NULL && q == NULL) {  
        return true;  
    }  
    if (p == NULL || q == NULL) {  
        return false;  
    }  
    if (p->val != q->val) {  
        return false;  
    }  
    return mirrorcheck(p->left, q->right) && mirrorcheck(p->right, q->left);  
}  
  
bool isSymmetric(TreeNode* root) {  
    if (mirrorcheck(root->left, root->right) == true) {  
        return true;  
    } else {  
        return false;  
    }  
}
```



3) maximum-depth-of-binary-tree

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



4) validate-binary-search-tree

```
bool solve(TreeNode* root,long long int lb,long long int ub){  
    if(root==NULL){  
        return true;  
    }  
}
```

```

if(root->val>lb && root->val<ub){

    bool leftans=solve(root->left,lb,root->val);

    bool rightans=solve(root->right,root->val,ub);

    return leftans && rightans;

}

else{

    return false;

}

}

bool isValidBST(TreeNode* root) {

    long long int lowerbound=-4294967296;

    long long int upperbound=4294967296;

    bool ans= solve(root,lowerbound,upperbound);

    return ans;

}

```

The screenshot shows a C++ IDE with the following code in the editor:

```

14 bool solve(TreeNode* root, long long int lb, long long int ub){
15     if(root==NULL){
16         return true;
17     }
18
19     if(root->val>lb && root->val<ub){
20         bool leftans=solve(root->left, lb, root->val);
21         bool rightans=solve(root->right, root->val, ub);
22         return leftans && rightans;
23     }
24     else{
25         return false;
26     }
27 }

```

Below the code editor, the 'Test Result' tab is active, showing 'Accepted' with a runtime of 0 ms. There are two test cases listed: 'Case 1' and 'Case 2'.

5) kth-smallest-element-in-a-bst

```

int kthSmallest(TreeNode* root, int &k) {

    if(root==NULL){

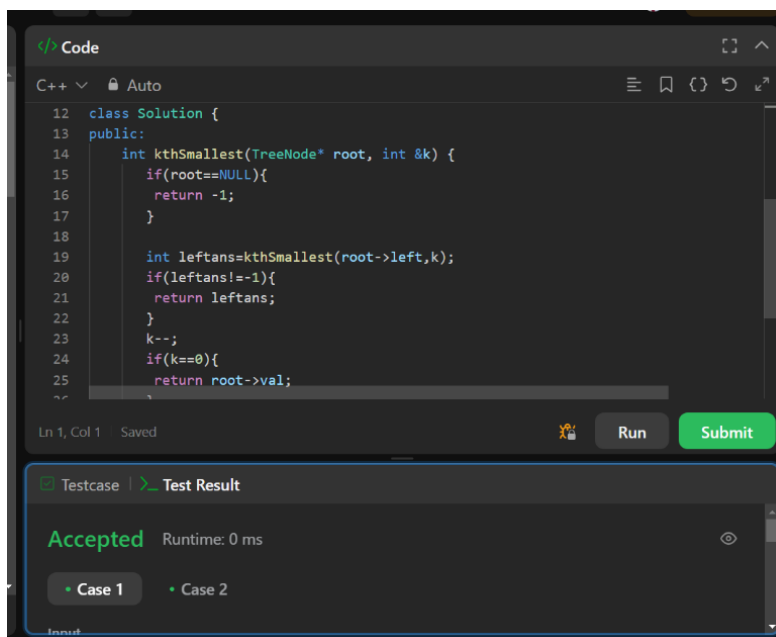
```

```

        return -1;
    }

    int leftans=kthSmallest(root->left,k);
    if(leftans!=-1){
        return leftans;
    }
    k--;
    if(k==0){
        return root->val;
    }
    int rightans=kthSmallest(root->right,k);
    return rightans;
}

```



```

12 class Solution {
13 public:
14     int kthSmallest(TreeNode* root, int &k) {
15         if(root==NULL){
16             return -1;
17         }
18
19         int leftans=kthSmallest(root->left,k);
20         if(leftans!=-1){
21             return leftans;
22         }
23         k--;
24         if(k==0){
25             return root->val;

```

Ln 1, Col 1 | Saved

Run Submit

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

6) binary-tree-level-order-traversal

```

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

    vector<vector<int>>ans;

    if(root==NULL){

        return ans;

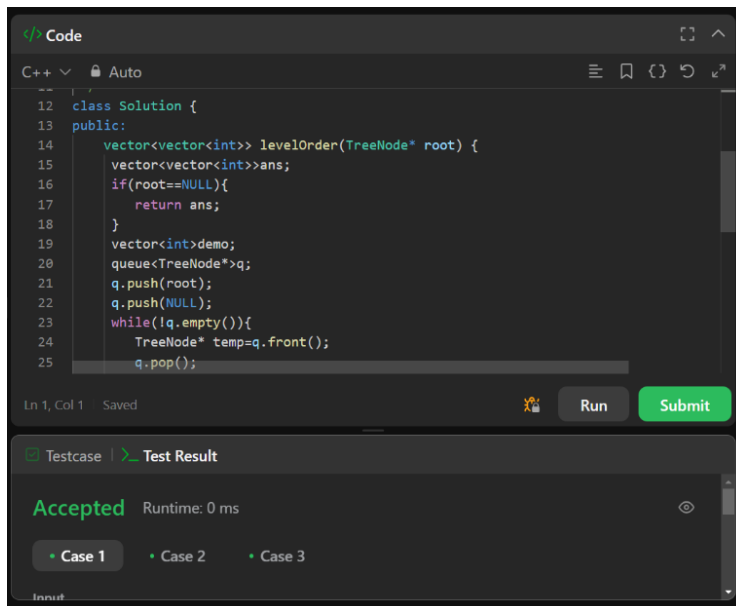
    }

```

```

vector<int>demo;
queue<TreeNode*>q;
q.push(root);
q.push(NULL);
while(!q.empty()){
    TreeNode* temp=q.front();
    q.pop();
    if(temp==NULL){
        ans.push_back(demo);
        demo.clear();
        if(!q.empty()){
            q.push(NULL);
        }
    }
    else{
        demo.push_back(temp->val);
        if(temp->left){
            q.push(temp->left);
        }
        if(temp->right){
            q.push(temp->right);
        }
    }
}
return ans;
}

```



```
12 class Solution {
13 public:
14     vector<vector<int>> levelOrder(TreeNode* root) {
15         vector<vector<int>>ans;
16         if(root==NULL){
17             return ans;
18         }
19         vector<int>demo;
20         queue<TreeNode*>q;
21         q.push(root);
22         q.push(NULL);
23         while(!q.empty()){
24             TreeNode* temp=q.front();
25             q.pop();
```

Ln 1, Col 1 Saved Run Submit

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

7) binary-tree-zigzag-level-order-traversal

```
vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
```

```
    vector<vector<int>>ans;
```

```
    if(root==NULL){
```

```
        return ans;
```

```
    }
```

```
    queue<TreeNode*>q;
```

```
    bool LtoR=true;
```

```
    q.push(root);
```

```
    while(!q.empty()){
```

```
        int width=q.size();
```

```
        vector<int>level(width);
```

```
        for(int i=0;i<width;i++){
```

```
            TreeNode* frontnode=q.front();
```

```
            q.pop();
```

```
            int index= LtoR? i: width-i-1;
```

```
            level[index]=frontnode->val;
```

```

        if(frontnode->left){
            q.push(frontnode->left);
        }
        if(frontnode->right){
            q.push(frontnode->right);
        }
    }
    LtoR=!LtoR;
    ans.push_back(level);
}

return ans;
}

```

The screenshot shows a C++ code editor with a dark theme. The code implements a zigzag level order traversal of a binary tree. It uses a queue to traverse the tree level by level, alternating the direction of node insertion into a result vector. The code is as follows:

```

12 class Solution {
13 public:
14     vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
15         vector<vector<int>>ans;
16         if(root==NULL){
17             return ans;
18         }
19         queue<TreeNode*>q;
20         bool LtoR=true;
21         q.push(root);
22         while(!q.empty()){
23             int width=q.size();
24             vector<int>level(width);
25             for(int i=0;i<width;i++){

```

Below the code editor, the 'Test Result' panel shows that the solution was 'Accepted' with a runtime of 0 ms. There are three test cases listed: Case 1, Case 2, and Case 3.

8) binary-tree-right-side-view

```

void RV(TreeNode* root,int level,vector<int>&ans){

    if(root==NULL){

        return;

    }

```



```

        if(level==ans.size()){
            ans.push_back(root->val);
        }

        RV(root->right,level+1,ans);
        RV(root->left,level+1,ans);
    }

vector<int> rightSideView(TreeNode* root) {
    vector<int>ans;

    int level=0;

    RV(root,level,ans);

    return ans;
}

```

```

Code
C++
1  struct TreeNode {
2      int val;
3      TreeNode *left;
4      TreeNode *right;
5      TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
6  };
7
8  class Solution {
9  public:
10     void RV(TreeNode* root,int level,vector<int>&ans){
11         if(root==NULL){
12             return;
13         }
14
15         if(level==ans.size()){
16             ans.push_back(root->val);
17         }
18
19         RV(root->right,level+1,ans);
20     }
21
22     vector<int> rightSideView(TreeNode* root) {
23         vector<int>ans;
24
25         int level=0;
26
27         RV(root,level,ans);
28
29         return ans;
30     }
31 };

```

Ln 1, Col 1 | Saved | Run | Submit

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4

9) construct-binary-tree-from-inorder-and-postorder-traversal

```

int findposition(int element,int size,vector<int>&inorder){
    for(int i=0;i<size;i++){
        if(element==inorder[i]){

```

```

        return i;
    }
}
return -1;
}

```

```

TreeNode* BT(vector<int>&inorder,vector<int>&postorder,int size,int &postindex,int
startinorder,int endinorder){

```

```

    if(postindex<0 || startinorder>endinorder){
        return NULL;
    }

```

```

    int element=postorder[postindex--];

```

```

    TreeNode* root= new TreeNode(element);

```

```

    int position =findposition(element,size,inorder);

```

```

    root->right=BT(inorder,postorder,size,postindex,position+1,endinorder);

```

```

    root->left=BT(inorder,postorder,size,postindex,startinorder,position-1);

```

```

    return root;

```

```

}

```

```

TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {

```

```

    int size=inorder.size();

```

```

    int postindex=size-1;

```

```

    int startinorder=0;

```

```

    int endinorder=size-1;

```

```

    TreeNode* root= BT(inorder,postorder,size,postindex,startinorder,endinorder);

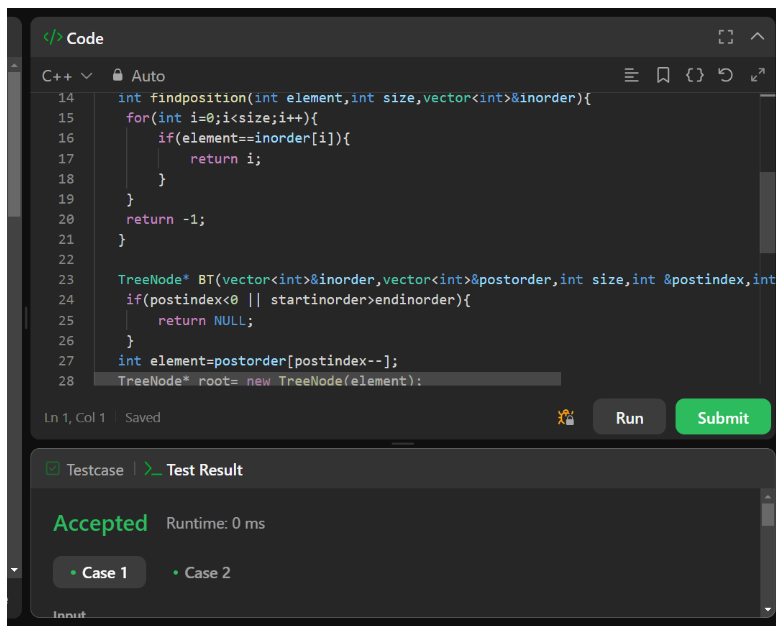
```

```

    return root;

```

}



```
Code
C++ Auto
14 int findposition(int element,int size,vector<int>&inorder){
15     for(int i=0;i<size;i++){
16         if(element==inorder[i]){
17             return i;
18         }
19     }
20     return -1;
21 }
22
23 TreeNode* BT(vector<int>&inorder,vector<int>&postorder,int size,int &postindex,int
24             &startinorder){
25     if(postindex<0 || startinorder>endinorder){
26         return NULL;
27     }
28     int element=postorder[postindex--];
29     TreeNode* root= new TreeNode(element);
```

Ln 1, Col 1 Saved Run Submit

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

10) vertical-order-traversal-of-a-binary-tree

```
vector<vector<int>>> verticalTraversal(TreeNode* root) {
    vector<vector<int>>>ans;
    queue<pair<TreeNode*,pair<int,int>>>>q;
    q.push({root,{0,0}});
    map<int,map<int,multiset<int>>>>mp;
    while(!q.empty()){
        auto temp=q.front();
        q.pop();
        TreeNode* node=temp.first;
        auto coordinate=temp.second;
        int row=coordinate.first;
        int col=coordinate.second;
        mp[col][row].insert(node->val);
        if(node->left){
            q.push({node->left,{row+1,col-1}});
        }
    }
}
```

```

        if(node->right){
            q.push({node->right,{row+1,col+1}});
        }
    }
}

```

```

for(auto i:mp){
    auto &map=i.second;
    vector<int>vline;
    for(auto j:map){
        auto &multiset=j.second;
        vline.insert(vline.end(),multiset.begin(),multiset.end());
    }
    ans.push_back(vline);
}

return ans;
}

```

```

Code
C++ v Auto
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(r
10 * };
11 */
12 class Solution {
13 public:
14     vector<vector<int>> verticalTraversal(TreeNode* root) {
15         vector<vector<int>>ans;
16         queue<pair<TreeNode*,pair<int,int>>>q;
17         q.push({root,{0,0}});
18         map<int,map<int,multiset<int>>>mp;
19         while(!q.empty()){
20             auto temp=q.front();
21             q.pop();

```

Ln 1, Col 1 Saved Run Submit

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3