**ASSIGNMENT – 3**

**Name – Ritvik Jindal**

**UID – 22BCS10235**

**CLASS – FL\_IOT\_604-A**

**94.**[**Binary Tree Inorder Traversal**](https://leetcode.com/problems/binary-tree-inorder-traversal/)

class Solution {

public:

vector<int> inorderTraversal(TreeNode\* root) {

vector<int> ans;

traverse(root,ans);

return ans;

}

vector<int> traverse(TreeNode \*root, vector<int> &ans){

if(!root){

return ans;

}

traverse(root->left,ans);

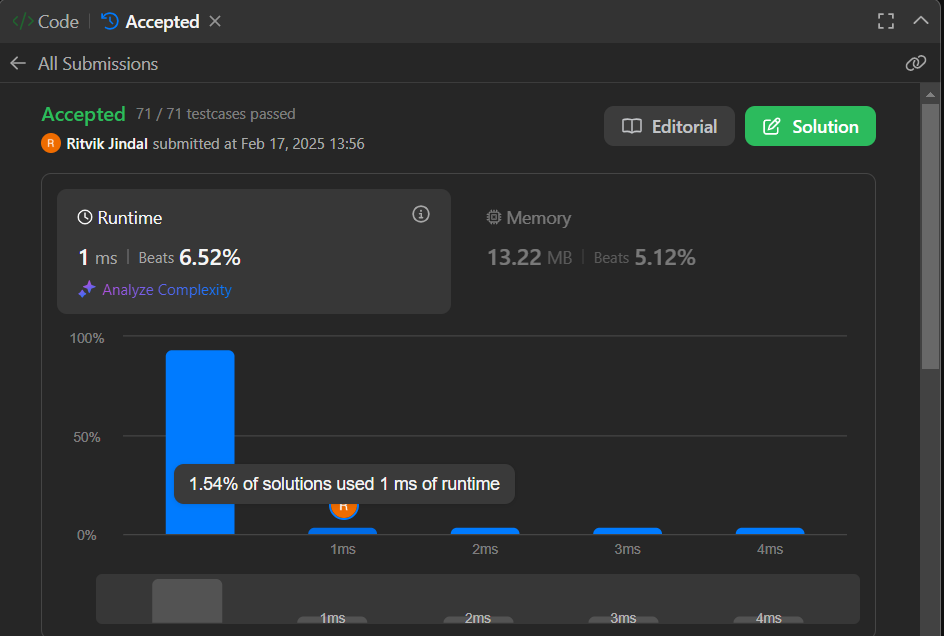
ans.push\_back(root->val);

traverse(root->right,ans);

return ans;

}

};



**104.**[**Maximum Depth of Binary Tree**](https://leetcode.com/problems/maximum-depth-of-binary-tree/description/)

class Solution {

public:

int maxDepth(TreeNode\* root) {

int l =0,r=0;

if(!root){

return 0;

}

if(root->left == NULL && root->right == NULL){

return 1;

}

if(root->right){

r = maxDepth(root->right);

}

if(root->left){

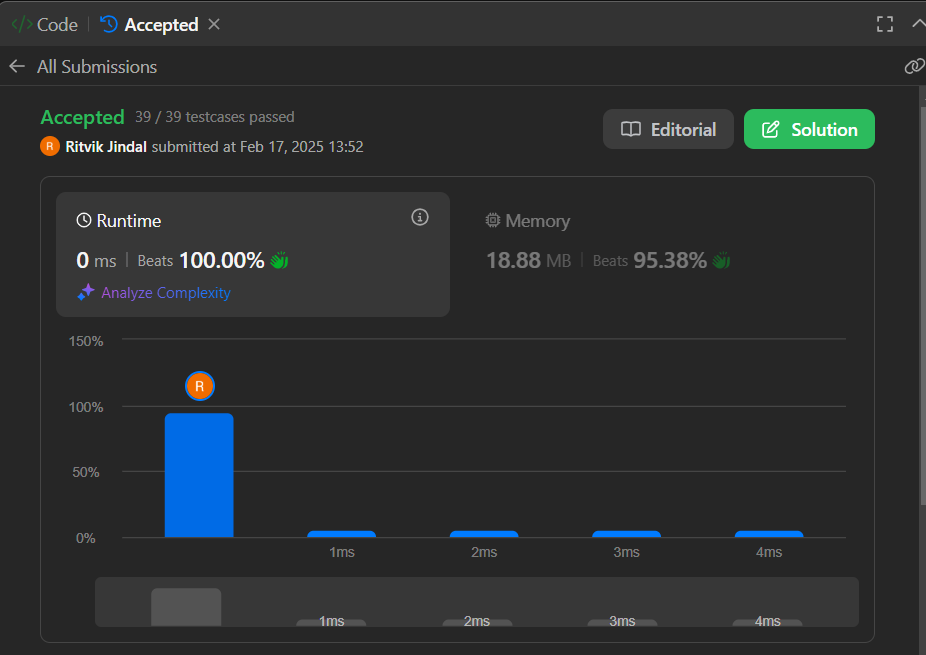
l = maxDepth(root->left);

}

return (max(l,r)+1);

}

};



# **98.**[**Validate Binary Search Tree**](https://leetcode.com/problems/validate-binary-search-tree/description/)

class Solution {

public:

bool isValidBST(TreeNode\* root) {

if(root == NULL){

return true;

}

if(root->right){

if(root->right->val > root->val){

isValidBST(root->right);

}

else{

return false;

}

}

if(root->left){

if(root->left->val < root->val){

isValidBST(root->left);

}

else{

return false;

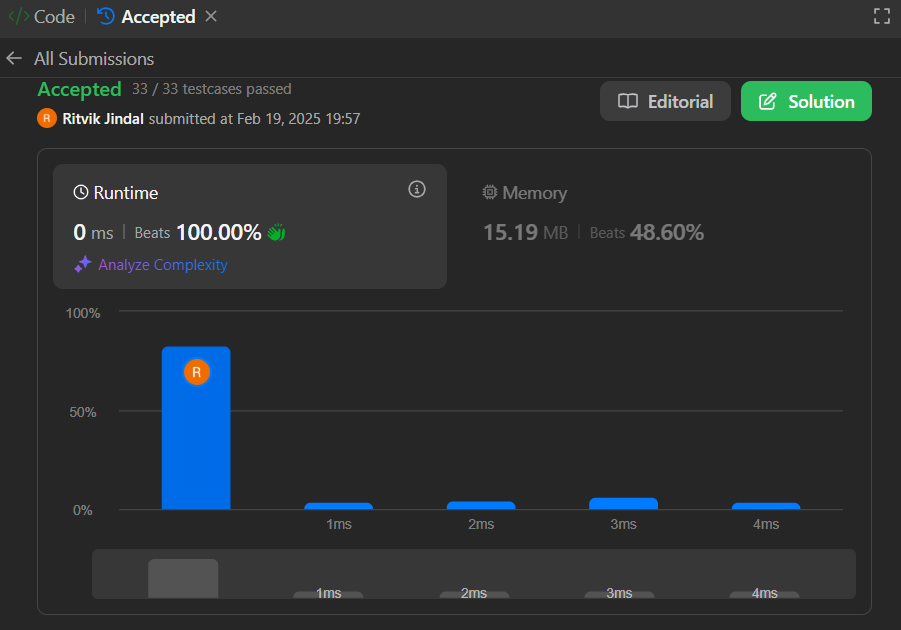
}

}

return true;

}

};



**102.**[**Binary Tree Level Order Traversal**](https://leetcode.com/problems/binary-tree-level-order-traversal/description/)

class Solution {

public:

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

vector<vector<int>> res;

if(root == NULL){

return res;

}

queue<TreeNode\*> q;

q.push(root);

while (!q.empty()) {

vector<int> ans;

int n = q.size();

while (n) {

TreeNode\* temp = q.front();

q.pop();

if (temp != NULL && temp->left) {

q.push(temp->left);

}

if (temp != NULL && temp->right ) {

q.push(temp->right);

}

if(temp!= NULL){

ans.push\_back(temp->val);

}

n--;

}

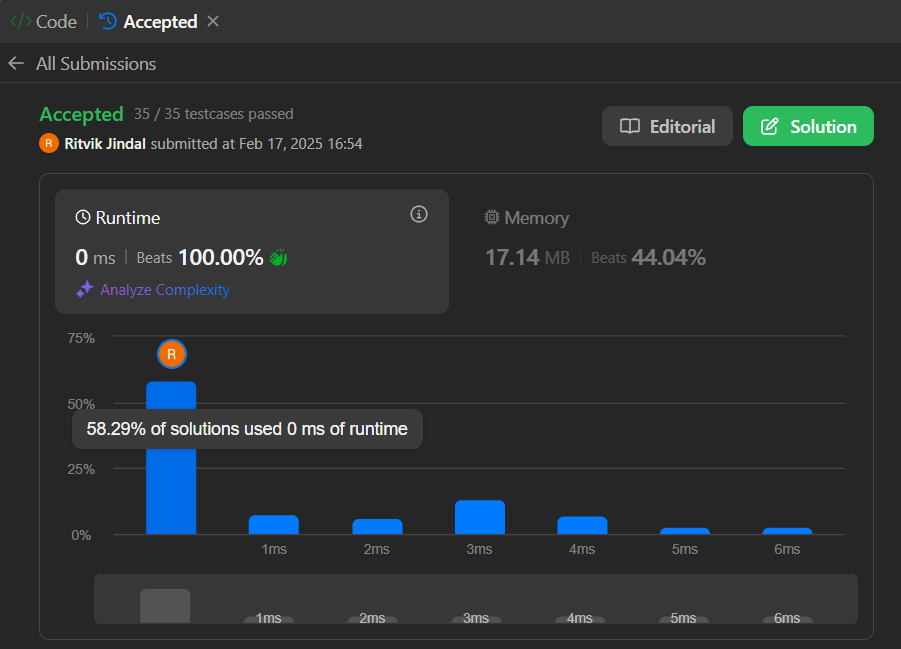
res.push\_back(ans);

}

return res;

}

};



**107.**[**Binary Tree Level Order Traversal II**](https://leetcode.com/problems/binary-tree-level-order-traversal-ii/description/)

class Solution {

public:

vector<vector<int>> levelOrderBottom(TreeNode\* root) {

vector<vector<int>> res;

if(root == NULL){

return res;

}

queue<TreeNode \*> q;

q.push(root);

while(!q.empty()){

vector<int> ans;

int n = q.size();

while(n--){

TreeNode\* temp = q.front();

q.pop();

if(temp!= NULL && temp->left){

q.push(temp->left);

}

if(temp!= NULL && temp->right){

q.push(temp->right);

}

if(temp!= NULL){

ans.push\_back(temp->val);

}

}

res.push\_back(ans);

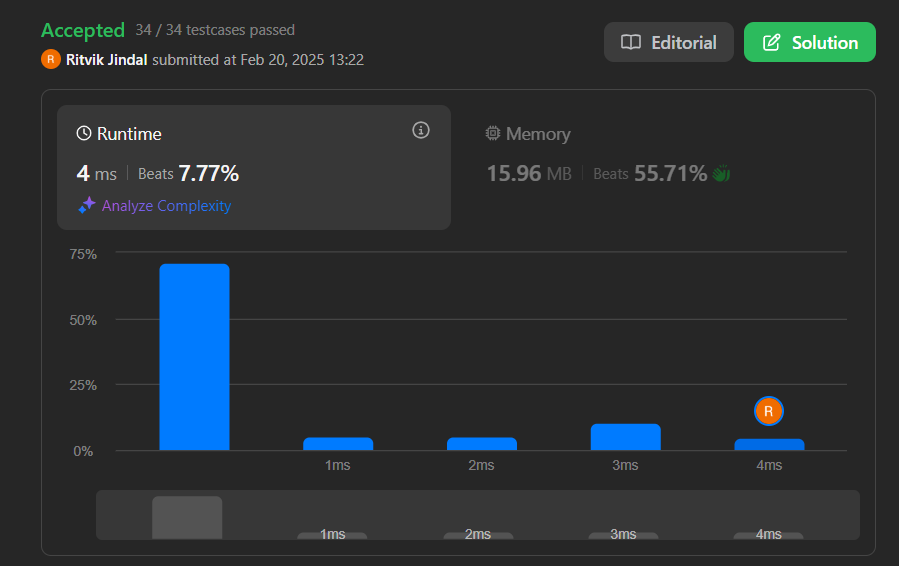
}

reverse(res.begin(),res.end());

return res;

}

};



**103.**[**Binary Tree Zigzag Level Order Traversal**](https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/description/)

class Solution {

public:

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

vector<vector<int>> res;

queue<TreeNode\*> q;

if(root == NULL){

return res;

}

q.push(root);

int c = 0;

while (!q.empty()) {

int n = q.size();

vector<int> ans;

while (n--) {

TreeNode\* temp = q.front();

q.pop();

if (temp != NULL) {

ans.push\_back(temp->val);

if (temp->left) {

q.push(temp->left);

}

if (temp->right) {

q.push(temp->right);

}

}

}

if (c % 2 != 0) {

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

}

res.push\_back(ans);

c++;

}

return res;

}

};

