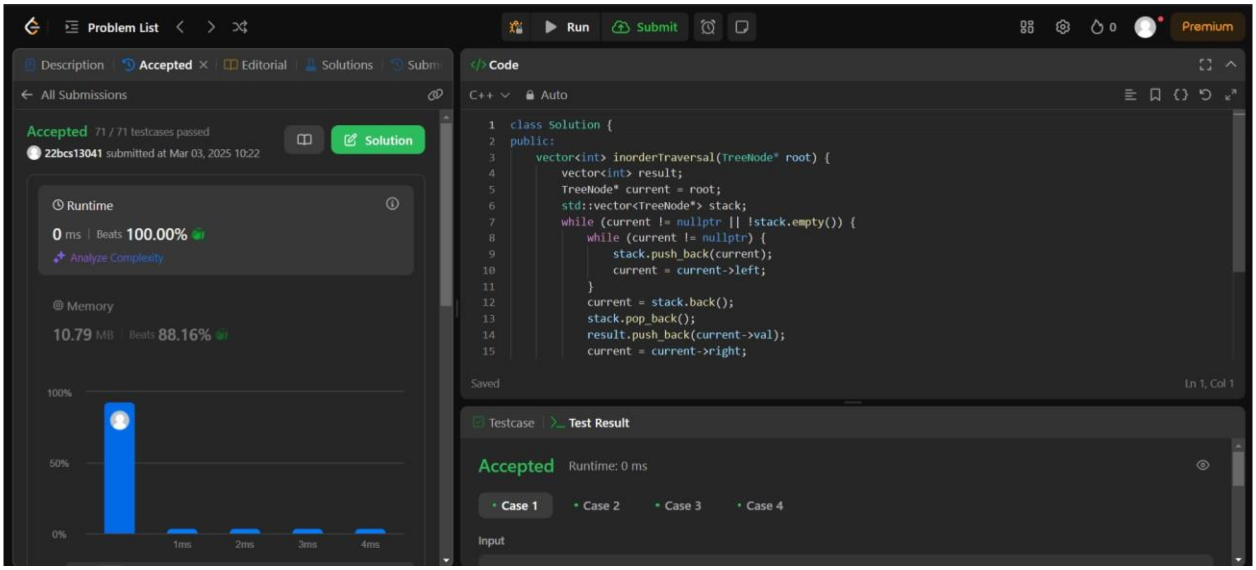
Name: Akshita Sood

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;
  }
};
```



Accepted 71 / 71 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:22

Runtime: 0 ms | Beats 100.00%

Memory: 10.79 MB | Beats 88.16%

```
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int> result;
        TreeNode* current = root;
        std::vector<TreeNode*> stack;
        while (current != nullptr || !stack.empty()) {
            while (current != nullptr) {
                stack.push_back(current);
                current = current->left;
            }
            current = stack.back();
            stack.pop_back();
            result.push_back(current->val);
            current = current->right;
```

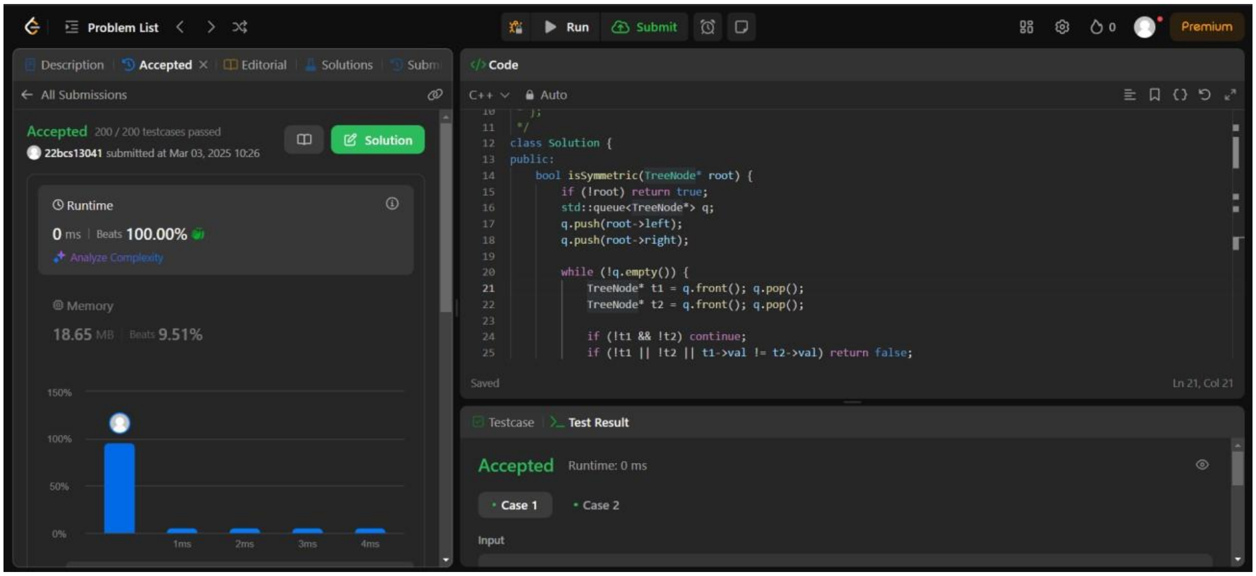
Saved

Ln 1, Col 1

Accepted Runtime: 0 ms

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);
  }
};
```



Accepted 200 / 200 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:26

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

18.65 MB | Beats 9.51%

```
 };
 */
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        if (!root) return true;
        std::queue<TreeNode*> q;
        q.push(root->left);
        q.push(root->right);

        while (!q.empty()) {
            TreeNode* t1 = q.front(); q.pop();
            TreeNode* t2 = q.front(); q.pop();

            if (!t1 && !t2) continue;
            if (!t1 || !t2 || t1->val != t2->val) return false;
```

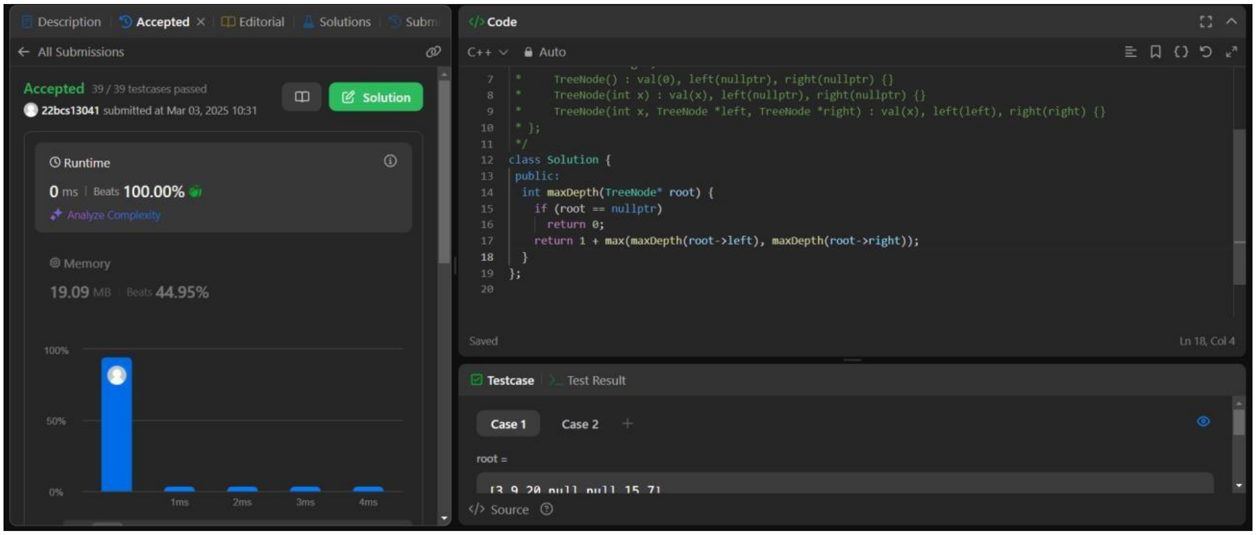
Saved

Ln 21, Col 21

Accepted Runtime: 0 ms

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));
};
```



Accepted 39 / 39 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:31

Runtime

0 ms | Beats 100.00%

Memory

19.09 MB | Beats 44.95%

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

Saved

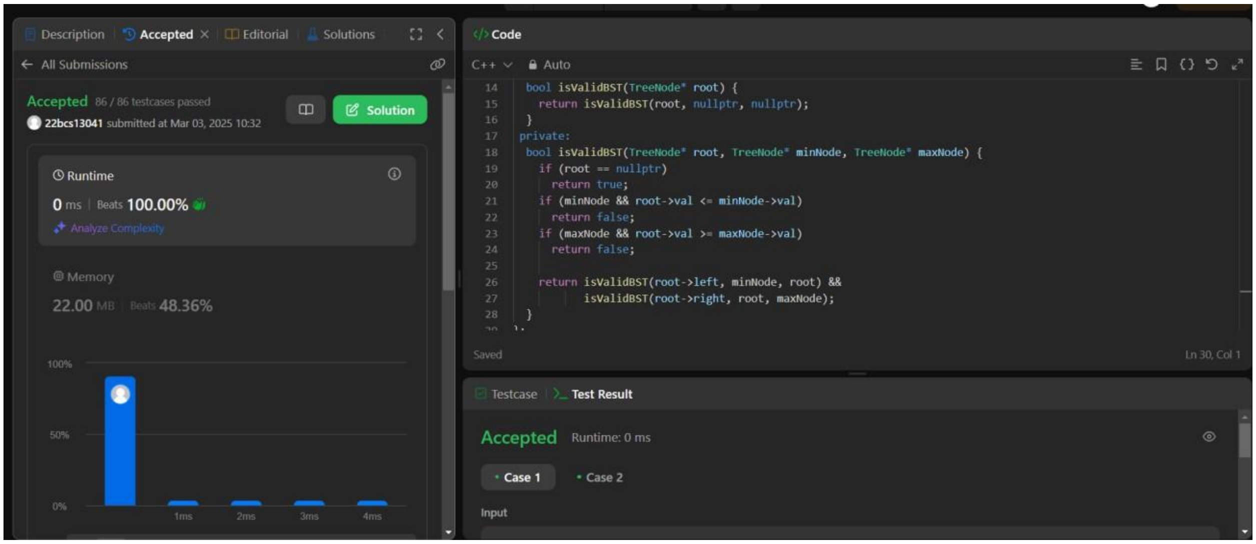
Ln 18, Col 4

root =

```
[3,9,20,null,null,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

22bcs13041 submitted at Mar 03, 2025 10:32

0 ms | Beats 100.00%

22.00 MB | Beats 48.36%

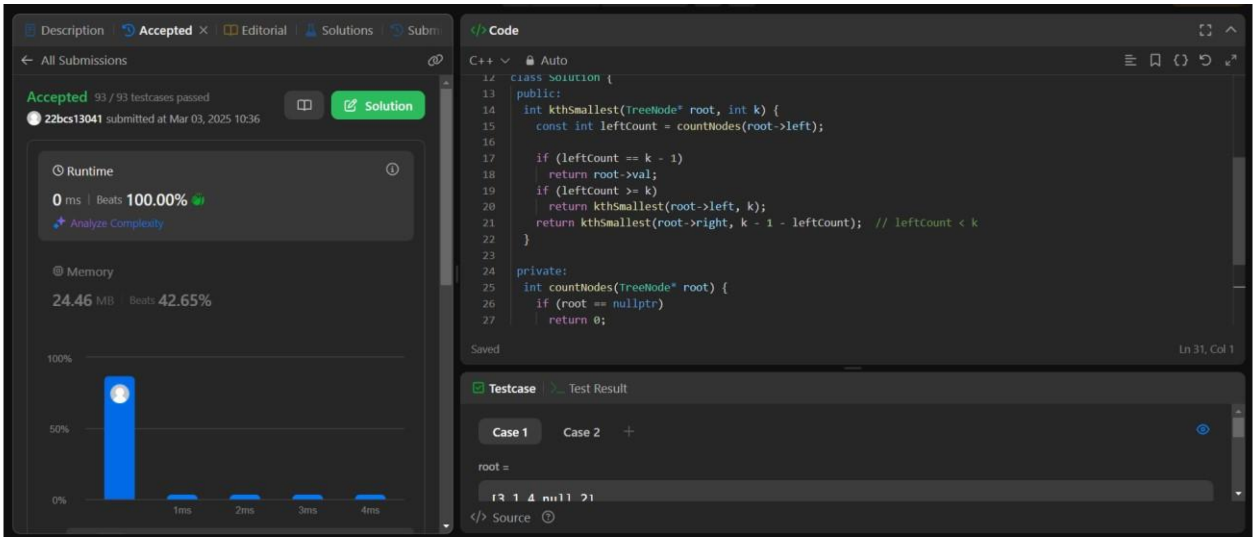
```
  bool isValidBST(TreeNode* root) {
    return isValidBST(root, nullptr, nullptr);
  }
 private:
  bool isValidBST(TreeNode* root, TreeNode* minNode, TreeNode* maxNode) {
    if (root == nullptr)
      return true;
    if (minNode && root->val <= minNode->val)
      return false;
    if (maxNode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

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

Saved   Ln 30, Col 1

Accepted   Runtime: 0 ms

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

0 ms | Beats 100.00%

24.46 MB | Beats 42.65%

```
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;
```

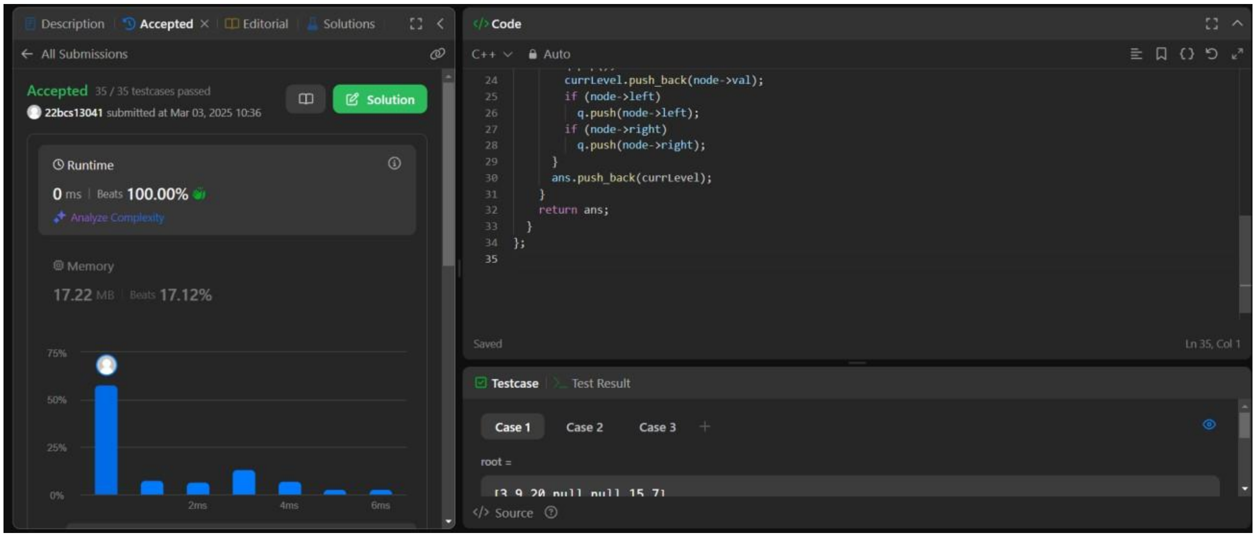
Saved   Ln 31, Col 1

root =

[3,1,4,null,2]

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;
 }
};
```



Accepted 35 / 35 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:36

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

17.22 MB | Beats 17.12%

```
          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;
    }
};
```

Saved

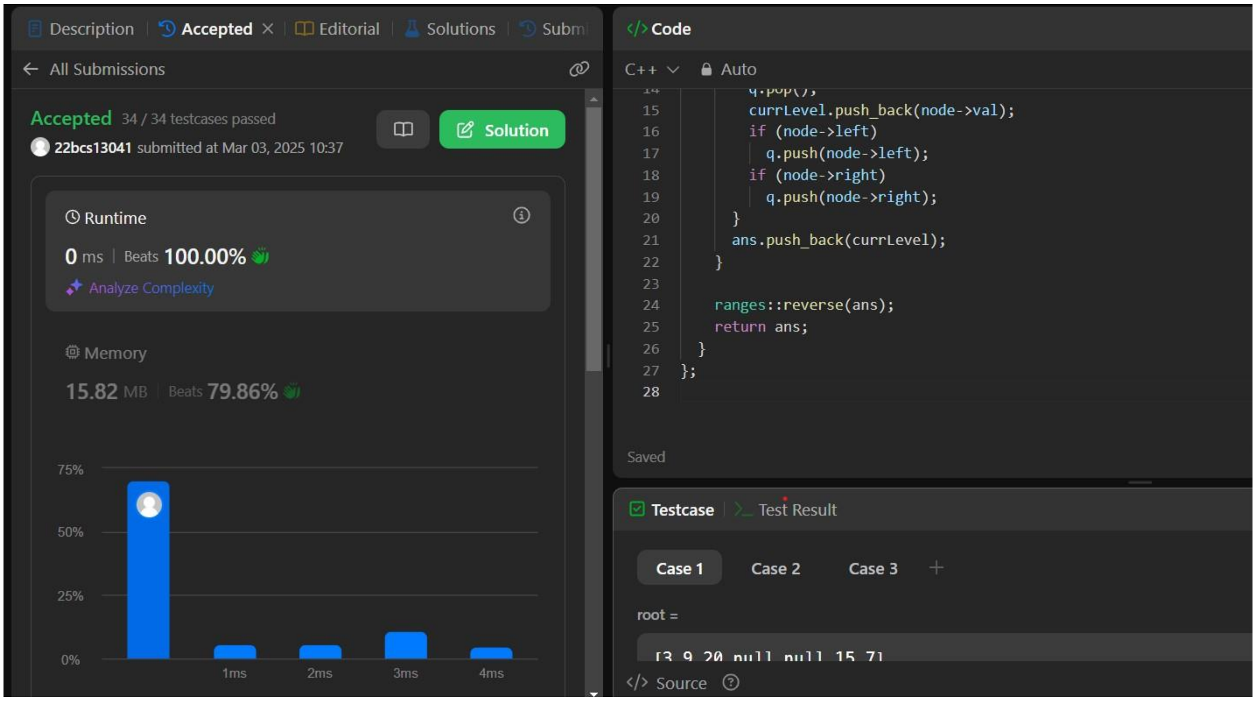
Ln 35, Col 1

root =

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

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

22bcs13041 submitted at Mar 03, 2025 10:37

Runtime

0 ms | Beats 100.00%

Memory

15.82 MB | Beats 79.86%

```
            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;
      }
};
```

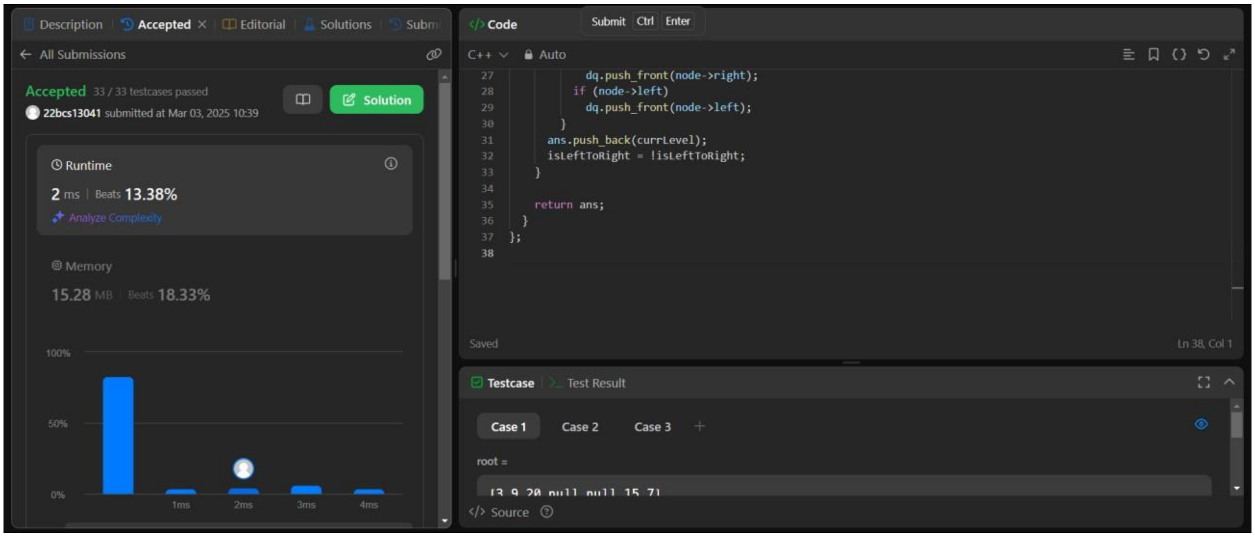
root =

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

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;
  }
};
```



Accepted 33 / 33 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:39

Runtime

2 ms | Beats 13.38%

Memory

15.28 MB | Beats 18.33%

```
            dq.push_front(node->right);
          if (node->left)
            dq.push_front(node->left);
        }
      ans.push_back(currLevel);
      isLeftToRight = !isLeftToRight;
    }

    return ans;
  }
};
```

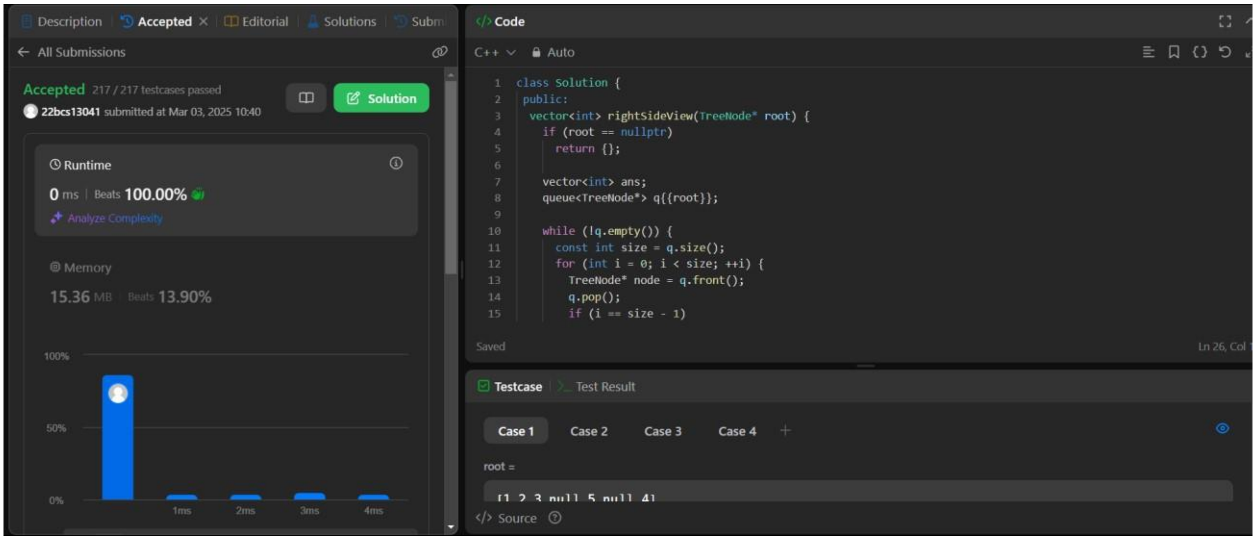
root =

```
[3 9 20 null null 15 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

0 ms | Beats 100.00%

15.36 MB | Beats 13.90%

```
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)
```

Saved

Ln 26, Col 1

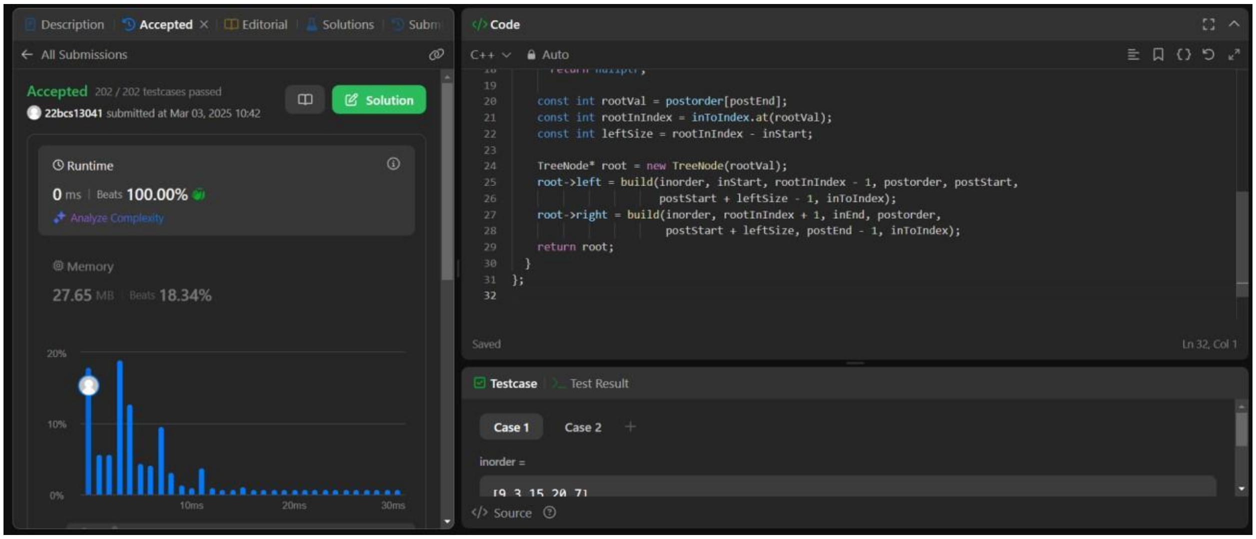
root =

[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;
  }
};
```



Accepted 202 / 202 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:42

0 ms | Beats 100.00%

27.65 MB | Beats 18.34%

```
      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;
  }
};
```

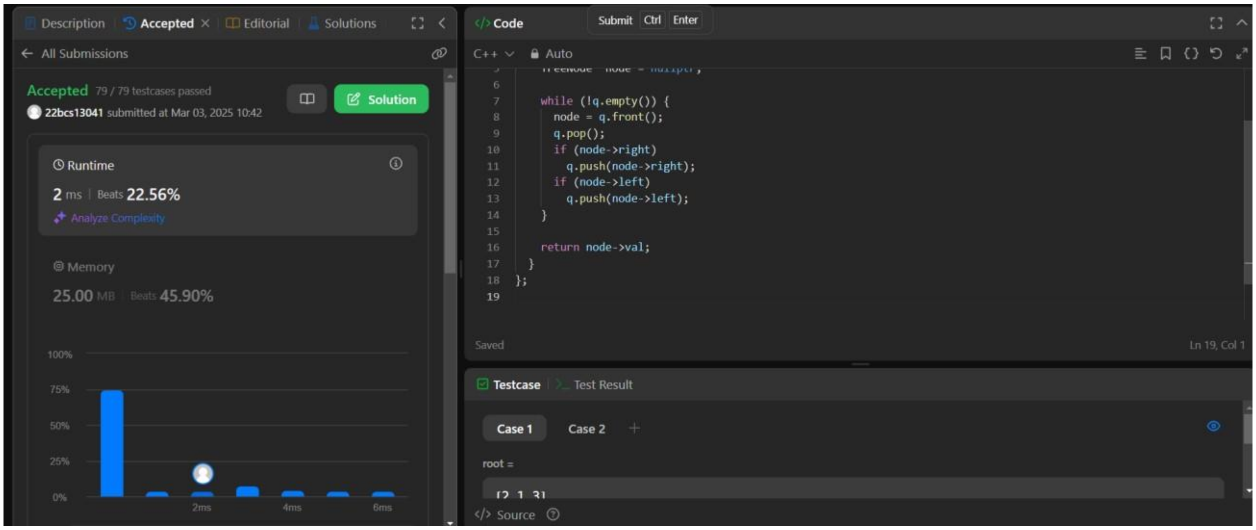
inorder =

[9,3,15,20,7]

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;
 }
};
```



Accepted 79 / 79 testcases passed
22bcs13041 submitted at Mar 03, 2025 10:42

Runtime
2 ms | Beats 22.56%

Memory
25.00 MB | Beats 45.90%

```
    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;
  }
};
```

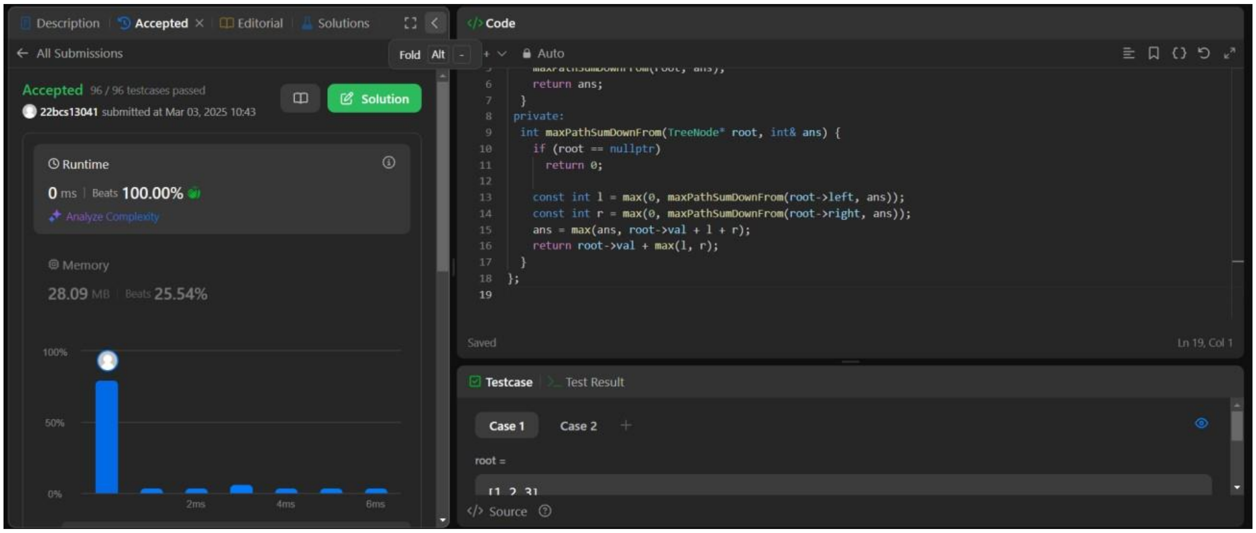
Saved
Ln 19, Col 1

root =
[2,1,3]

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

0 ms | Beats 100.00%

28.09 MB | Beats 25.54%

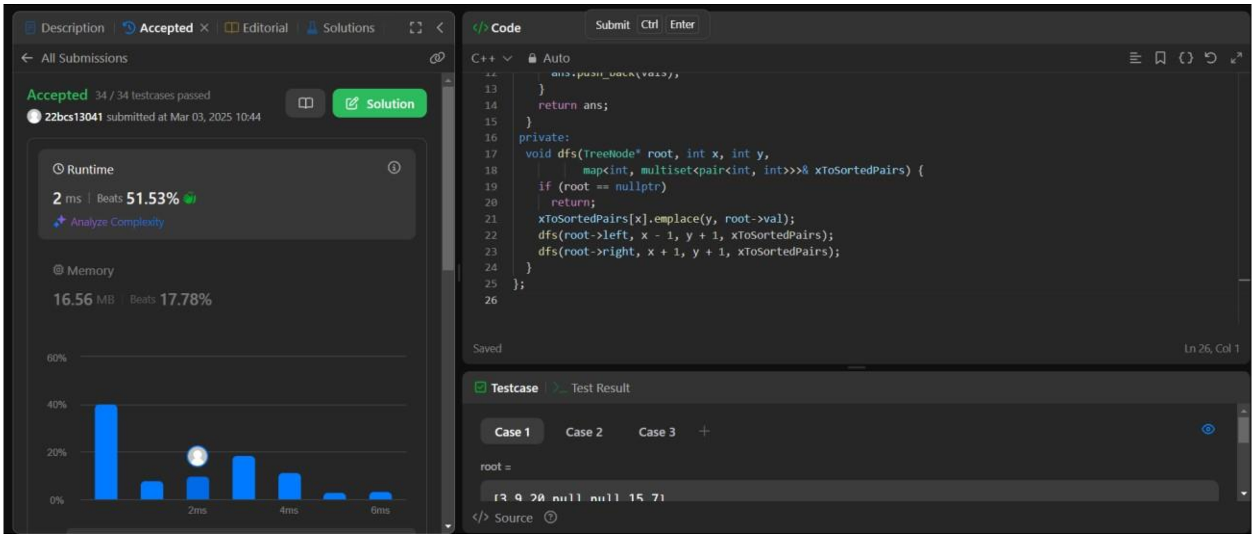
```
    maxPathSumDownFrom(root, ans);
    return ans;
  }
private:
 int maxPathSumDownFrom(TreeNode* root, int& ans) {
   if (root == nullptr)
     return 0;

   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);
 }
};
```

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);
  } };
```



Accepted 34 / 34 testcases passed

22bcs13041 submitted at Mar 03, 2025 10:44

Runtime

2 ms | Beats 51.53%

Memory

16.56 MB | Beats 17.78%

```
      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);
  }
};

```

Saved

Ln 26, Col 1

root =

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