

94.Binary Tree Inorder Traversal

```
Code
Java Auto
1 import java.util.*;
2
3 class TreeNode {
4     int val;
5     TreeNode left;
6     TreeNode right;
7     TreeNode(int x) { val = x; }
8 }
9
10 class Solution {
11     public List<Integer> inorderTraversal(TreeNode root) {
12         List<Integer> result = new ArrayList<>();
13         Stack<TreeNode> stack = new Stack<>();
14         while (root != null || !stack.isEmpty()) {
15             while (root != null) {
16                 stack.push(root);
17                 root = root.left;
18             }
19             root = stack.pop();
20             result.add(root.val);
21             root = root.right;
22         }
23         return result;
24     }
25 }
--
Saved Ln 1, Col 1
```

Code Accepted x

All Submissions

Accepted 71 / 71 testcases passed

krishkansal0 submitted at Feb 17, 2025 21:30

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

41.71 MB | Beats 43.94%

Analyze Complexity

150%
100%
50%
0%

1ms 2ms 3ms 4ms

Testcase Test Result

101. [Symmetric Tree](#)

```
Code
Java Auto
1 import java.util.*;
2
3 class TreeNode {
4     int val;
5     TreeNode left;
6     TreeNode right;
7     TreeNode(int x) { val = x; }
8 }
9
10 class Solution {
11     public boolean isSymmetric(TreeNode root) {
12         if (root == null) return true;
13         return isMirror(root.left, root.right);
14     }
15
16     private boolean isMirror(TreeNode t1, TreeNode t2) {
17         if (t1 == null && t2 == null) return true;
18         if (t1 == null || t2 == null) return false;
19         return (t1.val == t2.val) && isMirror(t1.left, t2.right) && isMirror
20 (t1.right, t2.left);
21 }
22 }
```

Saved Ln 21, Col 2

Description | Accepted x | Editorial | Solutions | Submissions | Submit

← All Submissions

Accepted 199 / 199 testcases passed

krishkansal0 submitted at Feb 17, 2025 21:33

Editorial Solution

Runtime ⓘ

0 ms | Beats 100.00% 🌿

[Analyze Complexity](#)

Memory ⓘ

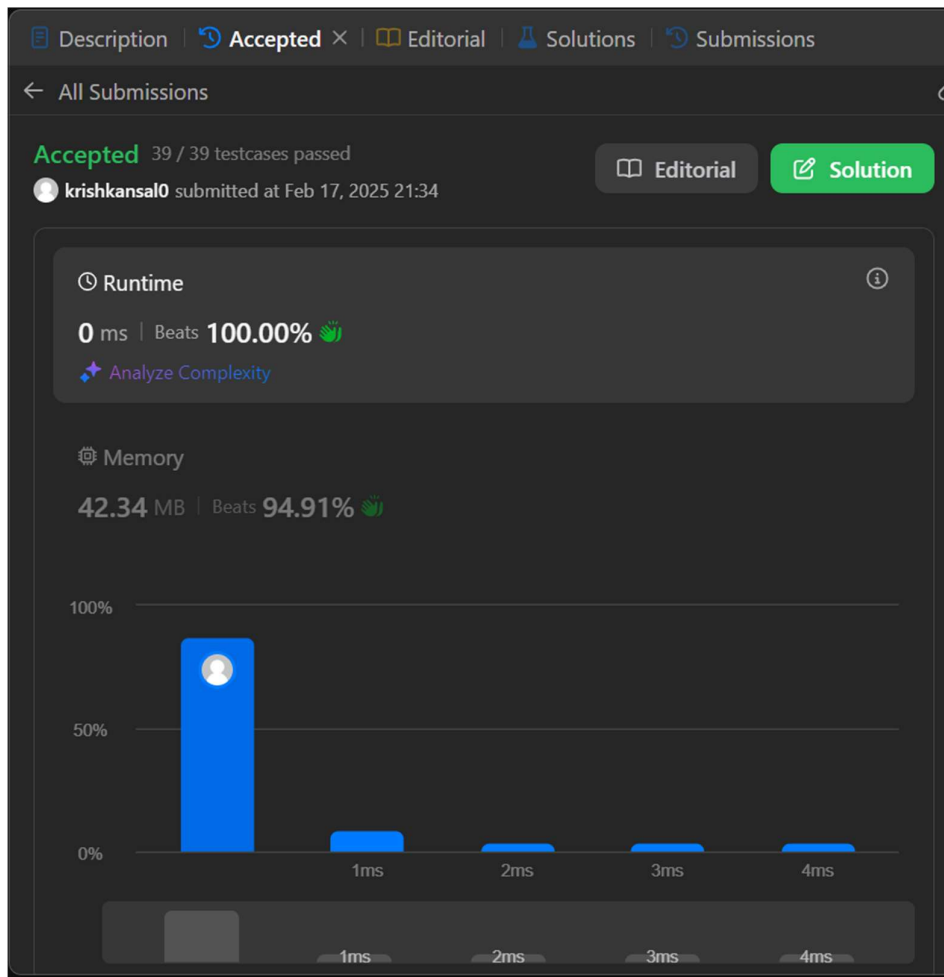
41.76 MB | Beats 66.84% 🌿

Submission	Runtime (ms)
User (krishkansal0)	0
Other Submissions	1, 2, 3, 4

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

```
</> Code
Java Auto

1  import java.util.*;
2
3  class TreeNode {
4      int val;
5      TreeNode left;
6      TreeNode right;
7      TreeNode(int x) { val = x; }
8  }
9
10 class Solution {
11     public int maxDepth(TreeNode root) {
12         if (root == null) return 0;
13         return 1 + Math.max(maxDepth(root.left), maxDepth(root.right));
14     }
15 }
```



98. [Validate Binary Search Tree](#)

```
Code
Java Auto
1 import java.util.*;
2
3 class TreeNode {
4     int val;
5     TreeNode left;
6     TreeNode right;
7     TreeNode(int x) { val = x; }
8 }
9
10 class Solution {
11     public boolean isValidBST(TreeNode root) {
12         return validate(root, null, null);
13     }
14
15     private boolean validate(TreeNode node, Integer low, Integer high) {
16         if (node == null) return true;
17         if ((low != null && node.val <= low) || (high != null && node.val >=
high)) return false;
18         return validate(node.left, low, node.val) && validate(node.right,
node.val, high);
19     }
20 }
```

Saved Ln 1, Col 1

Description | Accepted x | Editorial | Solutions | Submissions | Submit

← All Submissions

Accepted 86 / 86 testcases passed

krishkansal0 submitted at Feb 17, 2025 21:36

Editorial Solution

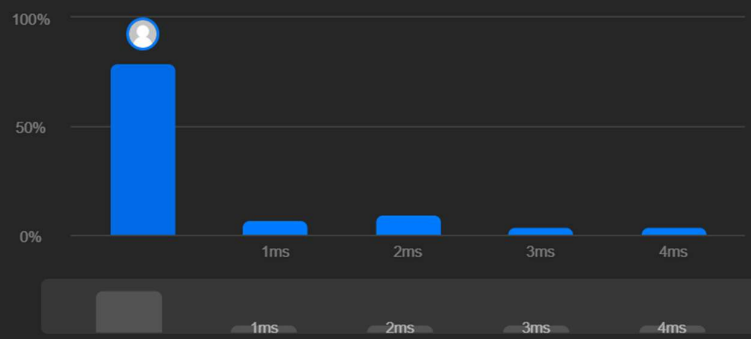
Runtime ⓘ

0 ms | Beats 100.00% 🌿

Analyze Complexity

Memory ⓘ

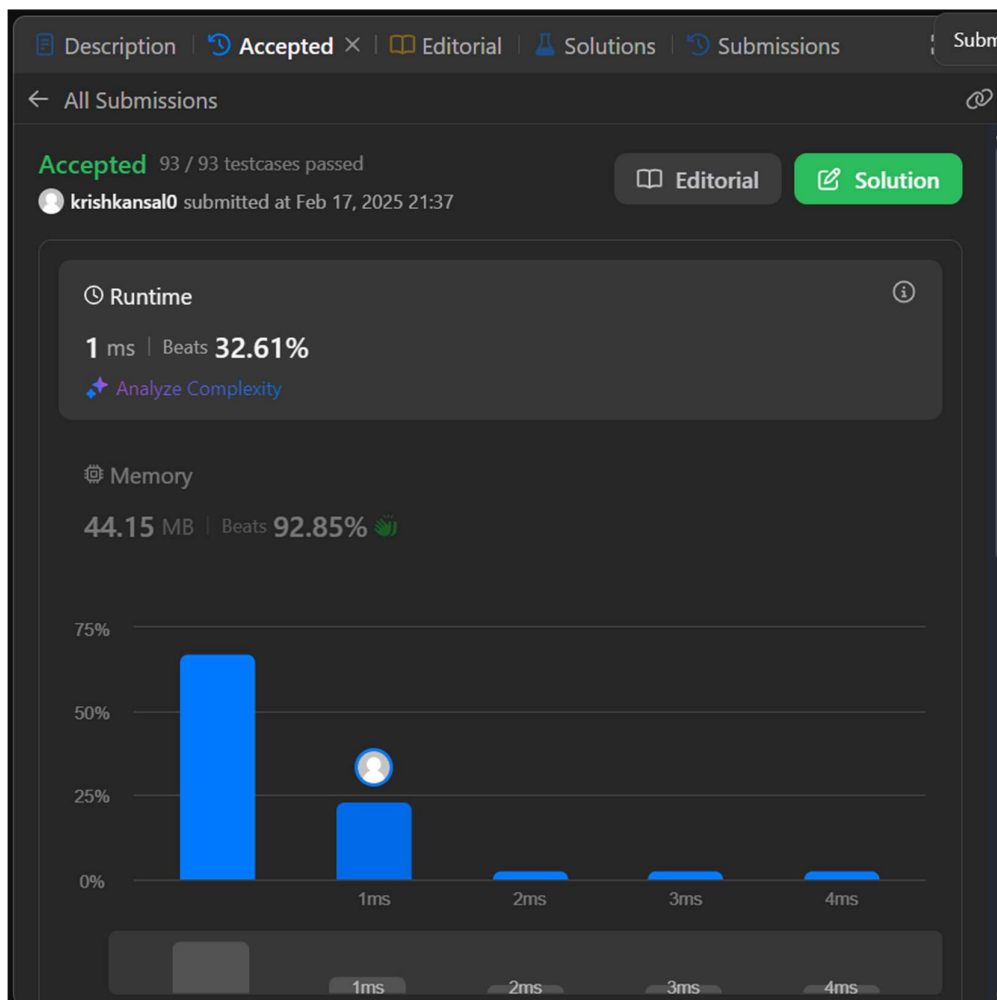
44.35 MB | Beats 28.60%



Runtime (ms)	Percentage of Submissions
0	100%
1	~10%
2	~10%
3	~10%
4	~10%

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

```
</> Code
Java Auto
1 class Solution {
2     public int kthSmallest(TreeNode root, int k) {
3         List<Integer> elements = new ArrayList<>();
4         inorder(root, elements);
5         return elements.get(k - 1);
6     }
7     static void inorder(TreeNode node, List<Integer> elements) {
8         if (node == null) return;
9         inorder(node.left, elements);
10        elements.add(node.val);
11        inorder(node.right, elements);
12    }
13 }
```



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

```
Code
Java Auto
1 class Solution {
2     public List<List<Integer>> levelOrder(TreeNode root) {
3         List<List<Integer>> result = new ArrayList<>();
4         if (root == null) return result;
5         Queue<TreeNode> queue = new LinkedList<>();
6         queue.offer(root);
7         while (!queue.isEmpty()) {
8             int size = queue.size();
9             List<Integer> level = new ArrayList<>();
10            for (int i = 0; i < size; i++) {
11                TreeNode node = queue.poll();
12                level.add(node.val);
13                if (node.left != null) queue.offer(node.left);
14                if (node.right != null) queue.offer(node.right);
15            }
16            result.add(level);
17        }
18        return result;
19    }
20 }
21
```

Saved Ln 21, Col 1

Description | Accepted × | Editorial | Solutions | Submissions | Submit

← All Submissions

Accepted 35 / 35 testcases passed

krishkansal0 submitted at Feb 17, 2025 21:38

Editorial Solution

Runtime ⓘ

1 ms | Beats 89.88% 🏆

Analyze Complexity

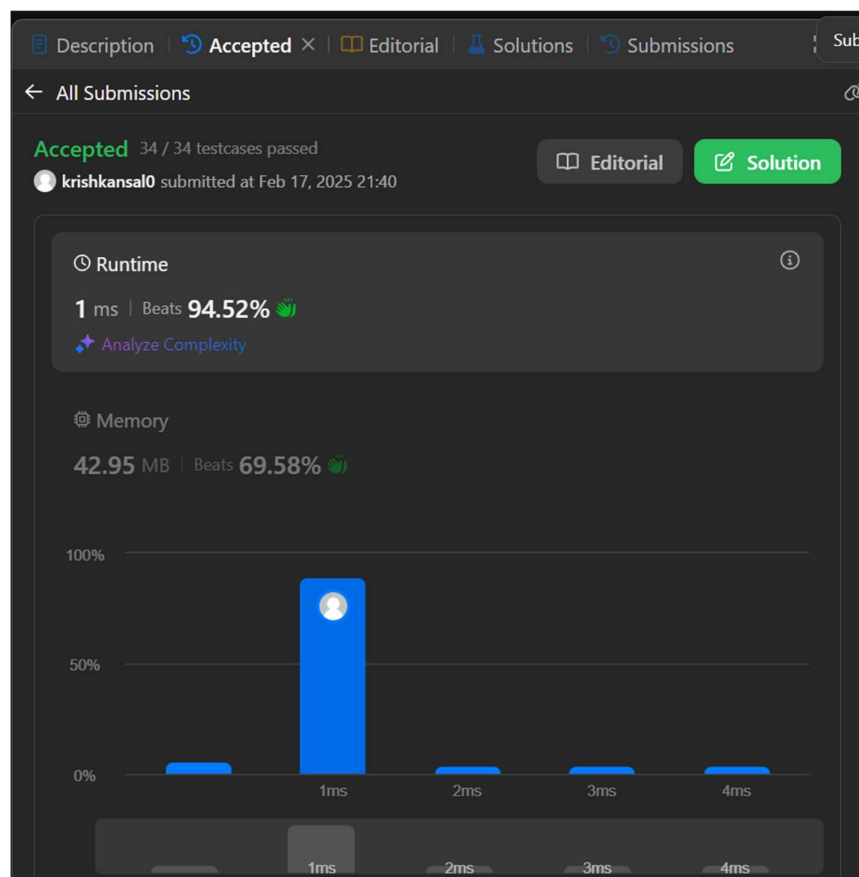
Memory ⓘ

45.39 MB | Beats 11.87%

Runtime (ms)	Beats (%)
1ms	89.88%
2ms	~1%
3ms	~1%
4ms	~1%

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

```
Code
Java Auto
1 class Solution {
2     public List<List<Integer>> levelOrderBottom(TreeNode root) {
3         List<List<Integer>> result = new LinkedList<>();
4         if (root == null) return result;
5         Queue<TreeNode> queue = new LinkedList<>();
6         queue.offer(root);
7         while (!queue.isEmpty()) {
8             int size = queue.size();
9             List<Integer> level = new ArrayList<>();
10            for (int i = 0; i < size; i++) {
11                TreeNode node = queue.poll();
12                level.add(node.val);
13                if (node.left != null) queue.offer(node.left);
14                if (node.right != null) queue.offer(node.right);
15            }
16            result.add(0, level);
17        }
18        return result;
19    }
20 }
21
```



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

```
Code
Java Auto
1 import java.util.*;
2
3 class TreeNode {
4     int val;
5     TreeNode left;
6     TreeNode right;
7     TreeNode(int x) { val = x; }
8 }
9
10 class Solution {
11     public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
12         List<List<Integer>> result = new ArrayList<>();
13         if (root == null) return result;
14         Queue<TreeNode> queue = new LinkedList<>();
15         queue.offer(root);
16         boolean leftToRight = true;
17         while (!queue.isEmpty()) {
18             int size = queue.size();
19             LinkedList<Integer> level = new LinkedList<>();
20             for (int i = 0; i < size; i++) {
21                 TreeNode node = queue.poll();
22                 if (leftToRight) {
23                     level.addLast(node.val);
24                 } else {
25                     level.addFirst(node.val);
26                 }
27             }
28             result.add(level);
29             leftToRight = !leftToRight;
30             while (!queue.isEmpty()) queue.poll();
31         }
32         return result;
33     }
34 }
```

Saved Ln 36, Col 1

Description | Accepted x | Editorial | Solutions | Submissions | Submit

← All Submissions

Accepted 33 / 33 testcases passed

krishkansal0 submitted at Feb 17, 2025 21:41

Editorial Solution

Runtime 1 ms | Beats 69.59%

Analyze Complexity

Memory 42.00 MB | Beats 88.36%

Runtime	Beats
1ms	69.59%
2ms	
3ms	
4ms	

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

```
Code
Java Auto
1 class Solution {
2     public List<Integer> rightSideView(TreeNode root) {
3         List<Integer> result = new ArrayList<>();
4         if (root == null) return result;
5         Queue<TreeNode> queue = new LinkedList<>();
6         queue.offer(root);
7         while (!queue.isEmpty()) {
8             int size = queue.size();
9             for (int i = 0; i < size; i++) {
10                TreeNode node = queue.poll();
11                if (i == size - 1) {
12                    result.add(node.val);
13                }
14                if (node.left != null) queue.offer(node.left);
15                if (node.right != null) queue.offer(node.right);
16            }
17        }
18        return result;
19    }
20 }
```

Description | Accepted × | Editorial | Solutions | Submissions

← All Submissions

Accepted 217 / 217 testcases passed

krishkansal0 submitted at Feb 17, 2025 21:42

Editorial Solution

Runtime 1 ms | Beats 72.44% 🌱

[Analyze Complexity](#)

Memory 42.50 MB | Beats 16.10%

Runtime	Percentage
1ms	~65%
2ms	~2%
3ms	~2%
4ms	~2%

106. [Construct Binary Tree from Inorder and Postorder Traversal](#)

```
Code
Java Auto
1 class Solution {
2     private int postIndex;
3
4     public TreeNode buildTree(int[] inorder, int[] postorder) {
5         postIndex = postorder.length - 1;
6         return build(inorder, postorder, 0, inorder.length - 1);
7     }
8
9     private TreeNode build(int[] inorder, int[] postorder, int inStart, int
inEnd) {
10        if (inStart > inEnd) return null;
11
12        TreeNode root = new TreeNode(postorder[postIndex--]);
13        int inIndex = findIndex(inorder, inStart, inEnd, root.val);
14
15        root.right = build(inorder, postorder, inIndex + 1, inEnd);
16        root.left = build(inorder, postorder, inStart, inIndex - 1);
17
18        return root;
19    }
20
21    private int findIndex(int[] inorder, int start, int end, int value) {
22        for (int i = start; i <= end; i++) {
23            if (inorder[i] == value) return i;
24        }
25    }
26 }
```

