

Advanced Programming LAB II

ASSIGNMENT - 3

Submitted by,

Jiya | 22BCS14856

22BCS_FL_IOT-601 (A)

94. Binary Tree Inorder Traversal

<https://leetcode.com/problems/binary-tree-inorder-traversal/description/>

```
class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> res = new ArrayList<>();

        inorder(root, res);
        return res;
    }

    private void inorder(TreeNode node, List<Integer> res) {
        if (node == null) {
            return;
        }
        inorder(node.left, res);
        res.add(node.val);
        inorder(node.right, res);
    }
}
```

The screenshot shows a LeetCode submission page for the problem 'Binary Tree Inorder Traversal'. The submission is for a Java solution and is marked as 'Accepted'. The runtime is 0 ms, and the memory usage is 41.63 MB. A bar chart shows the performance relative to other submissions, with the user's submission being the fastest. The test cases are all passed, and the expected output for the first case is [1, 3, 2].

Problem List < > > > Run Submit

Description | Accepted x | Editorial | Solutions | Note x | Submissions

All Submissions

Accepted 71 / 71 testcases passed
Jiya submitted at Mar 03, 2025 16:39

Editorial Solution

Runtime 0 ms Beats 100.00%
Memory 41.63 MB Beats 64.45%

Analyze Complexity

Code | Java

```
class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> res = new ArrayList<>();

        inorder(root, res);
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input

root = [1,null,2,3]

Output

[1,3,2]

Expected

[1,3,2]

Testcase

Case 1 Case 2 Case 3 Case 4 +

root = [1,null,2,3]

Source

101. Symmetric Tree

<https://leetcode.com/problems/symmetric-tree/description/>

```
class Solution {
    public boolean isSymmetric(TreeNode root) {
        return root == null || isMirror(root.left, root.right);
    }

    private boolean isMirror(TreeNode a, TreeNode b) {
        if (a == null && b == null) return true;
        if (a == null || b == null || a.val != b.val) return false;
        return isMirror(a.left, b.right) && isMirror(a.right, b.left);
    }
}
```

The screenshot displays the LeetCode submission interface for the problem "101. Symmetric Tree". The top navigation bar includes "Problem List", "Run", "Submit", and "Test Result". The main content area is divided into three sections:

- Accepted:** Shows "200 / 200 testcases passed" and "Jiya submitted at Mar 03, 2025 16:43".
- Runtime:** A bar chart showing the runtime performance. The first bar (Case 1) is significantly higher than the others, indicating it is the slowest. The runtime is 0 ms, and the memory is 41.71 MB.
- Testcase:** Shows the input and output for Case 1. The input is "root = [1,2,2,3,4,4,3]" and the output is "true".

The bottom section shows the code for the solution in Java:

```
class Solution {
    public boolean isSymmetric(TreeNode root) {
        return root == null || isMirror(root.left, root.right);
    }
}
```

104. Maximum Depth of Binary Tree

<https://leetcode.com/problems/maximum-depth-of-binary-tree/description/>

```
class Solution {
    public int maxDepth(TreeNode root) {

        if (root == null) return 0;
        int leftDepth = maxDepth(root.left);
        int rightDepth = maxDepth(root.right);

        return 1 + Math.max(leftDepth, rightDepth);
    }
}
```

Problem List < > Run Submit

Description Accepted X Editorial Solutions Note X Submissions Run Ctrl

All Submissions

Accepted 39 / 39 testcases passed
Jiya submitted at Mar 03, 2025 16:40

Editorial Solution

Runtime 0 ms | Beats 100.00%
Memory 42.57 MB | Beats 76.31%

Analyze Complexity

100%
50%
0%
1ms 2ms 3ms 4ms

Code | Java

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;

```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

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

Output

3

Expected

3

Testcase

Case 1 Case 2 +

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

Source

Trending videos Watch the emoti...

Search

ENG IN 4:41 PM 3/3/2025

98. Validate Binary Search Tree

<https://leetcode.com/problems/validate-binary-search-tree/description/>

```
class Solution {
    public boolean isValidBST(TreeNode root) {
        return check(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }

    private boolean check(TreeNode node, long min, long max) {
        if (node == null) return true;
        if (node.val <= min || node.val >= max) return false;
        return check(node.left, min, node.val) && check(node.right, node.val, max);
    }
}
```

The screenshot shows a LeetCode submission page for the problem "98. Validate Binary Search Tree". The submission is for user "Jiya" and is marked as "Accepted". The runtime is 0 ms, and the memory usage is 43.55 MB. The submission is compared against 86 test cases, all of which are passed. The code is written in Java and is as follows:

```
class Solution {
    public boolean isValidBST(TreeNode root) {
        return check(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }
}
```

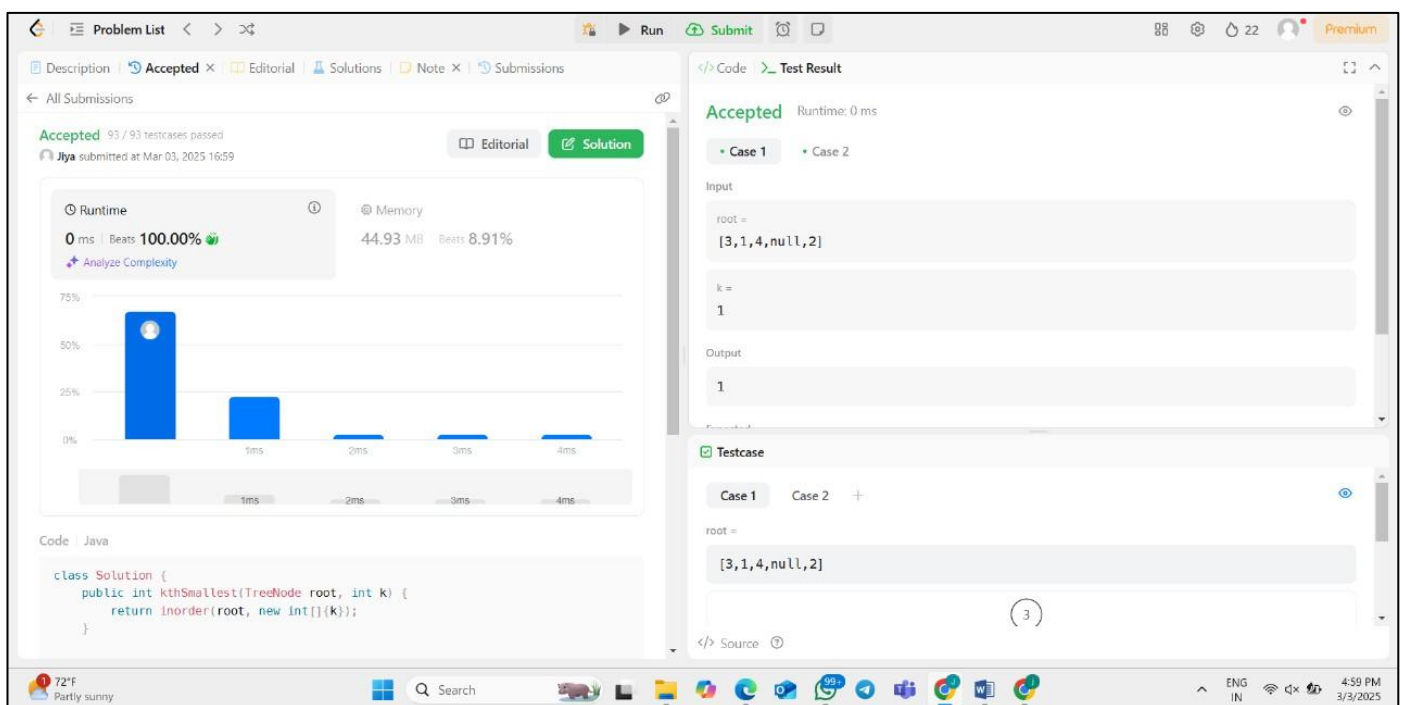
The "Testcase" section shows two cases. Case 1 has an input of "root = [2,1,3]" and an expected output of "true". Case 2 is also shown with the same input and expected output. The submission is successful, and the user is ranked 1st among all submissions for this problem.

230. Kth Smallest Element in a BST

<https://leetcode.com/problems/kth-smallest-element-in-a-bst/description/>

```
class Solution {
    public int kthSmallest(TreeNode root, int k) {
        return inorder(root, new int[]{k});
    }

    private int inorder(TreeNode node, int[] k) {
        if (node == null) return -1;
        int left = inorder(node.left, k);
        if (left != -1) return left;
        if (--k[0] == 0) return node.val;
        return inorder(node.right, k);
    }
}
```



102. Binary Tree Level Order Traversal

<https://leetcode.com/problems/binary-tree-level-order-traversal/description/>

```
class Solution {
    public List<List<Integer>> levelOrder(TreeNode root)
    {
        List<List<Integer>> al=new ArrayList<>();
        pre(root,0,al);
        return al;
    }
    public static void pre(TreeNode root,int l,List<List<Integer>>al)
    {
        if(root==null)
            return;
        if(al.size()==l)
        {
            List<Integer>li=new ArrayList<>();
            li.add(root.val);
            al.add(li);
        }
        else
            al.get(l).add(root.val);
        pre(root.left,l+1,al);
        pre(root.right,l+1,al);
    }
}
```

The screenshot displays the LeetCode submission interface for the problem 'Binary Tree Level Order Traversal'. The submission is in Java and has been accepted. The left panel shows performance metrics: Runtime is 0 ms (beats 100.00%) and Memory is 45.04 MB (beats 53.30%). A bar chart illustrates the runtime performance across different test cases. The right panel shows the test results for Case 1, where the input is a binary tree with root 3, left child 9, and right child 20, and the expected output is a list of lists representing the level order traversal: [[3], [9, 20], [15, 7]].

Accepted 35 / 35 testcases passed
Jiya submitted at Mar 03, 2025 17:05

Runtime 0 ms | Beats 100.00%
Memory 45.04 MB | Beats 53.30%

Testcase
Case 1 Case 2 Case 3 +
root =
[3, 9, 20, null, null, 15, 7]
Output
[[3], [9, 20], [15, 7]]
Expected
[[3], [9, 20], [15, 7]]

102. Binary Tree Level Order Traversal II

<https://leetcode.com/problems/binary-tree-level-order-traversal-ii/description/>

```
import java.util.*;

class Solution {
    public List<List<Integer>> levelOrderBottom(TreeNode root) {
        List<List<Integer>> result = new LinkedList<>();
        if (root == null) return result;

        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);

        while (!q.isEmpty()) {
            int size = q.size();
            List<Integer> level = new ArrayList<>();
            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                level.add(node.val);
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
            result.add(0, level);
        }
        return result;
    }
}
```

leetcode.com/problems/binary-tree-level-order-traversal-ii/submissions/1561361183/

Problem List < > </>

Description | Accepted x | Editorial | Solutions | Note x | Submissions

All Submissions

Accepted 34 / 34 testcases passed
Jiya submitted at Mar 03, 2025 17:07

Editorial Solution

Runtime 1 ms | Beats 94.02%
Memory 43.22 MB | Beats 19.83%

Analyze Complexity

100%
50%
0%

1ms 2ms 3ms 4ms

Code: Java

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 * }
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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

Output

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

Expected

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

Testcase

Case 1 Case 2 Case 3 +

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

Source

Weather alert In effect

Search

ENG IN

5:07 PM 3/3/2025

103. Binary Tree Zigzag Level Order Traversal

<https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/description/>

```
import java.util.*;

class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
        List<List<Integer>> result = new ArrayList<>();
        if (root == null) return result;

        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);
        boolean leftToRight = true;

        while (!q.isEmpty()) {
            int size = q.size();
            LinkedList<Integer> level = new LinkedList<>();
            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                if (leftToRight) level.addLast(node.val);
                else level.addFirst(node.val);
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
            result.add(level);
            leftToRight = !leftToRight;
        }
        return result;
    }
}
```

leetcode.com/problems/binary-tree-zigzag-level-order-traversal/submissions/1561362202/

Problem List < > ↺

Description Accepted Editorial Solutions Note Submissions

All Submissions

Accepted 33 / 33 testcases passed

Jiya submitted at Mar 03, 2025 17:09

Editorial Solution

Runtime 0 ms | Beats 100.00% Memory 42.35 MB | Beats 38.84%

Analyze Complexity

Code Java

```

/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;

```

Code Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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

Output

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

Expected

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

Testcase

Case 1 Case 2 Case 3 +

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

Source

199. Binary Tree Right Side View

<https://leetcode.com/problems/binary-tree-right-side-view/description/>

```
import java.util.*;

class Solution {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> result = new ArrayList<>();
        if (root == null) return result;

        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);

        while (!q.isEmpty()) {
            int size = q.size();
            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                if (i == size - 1) result.add(node.val);
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
        }
        return result;
    }
}
```

leetcode.com/problems/binary-tree-right-side-view/submissions/1561363309/

Problem List < > Run Submit

Description Accepted Editorial Solutions Note Submissions

All Submissions

Accepted 217 / 217 testcases passed
Jiya submitted at Mar 03, 2025 17:10

Runtime 1 ms Beats 72.03%
Memory 42.20 MB Beats 42.64%

75%
50%
25%
0%

1ms 2ms 3ms 4ms

Code Java

```
import java.util.*;

class Solution {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> result = new ArrayList<>();
        if (root == null) return result;

        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);

        while (!q.isEmpty()) {
            int size = q.size();
            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                if (i == size - 1) result.add(node.val);
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
        }
        return result;
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input

root = [1,2,3,null,5,null,4]

Output

[1,3,4]

Expected

[1,3,4]

Testcase

Case 1 Case 2 Case 3 Case 4

root = [1,2,3,null,5,null,4]

Source

71°F Partly sunny Search 5:11 PM 3/3/2025

106. Construct Binary Tree from Inorder and Postorder Traversal

<https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/description/>

```
import java.util.*;

class Solution {
    private int postIndex;
    private Map<Integer, Integer> inorderMap;

    public TreeNode buildTree(int[] inorder, int[] postorder) {
        postIndex = postorder.length - 1;
        inorderMap = new HashMap<>();
        for (int i = 0; i < inorder.length; i++) {
            inorderMap.put(inorder[i], i);
        }
        return build(inorder, postorder, 0, inorder.length - 1);
    }

    private TreeNode build(int[] inorder, int[] postorder, int left, int right) {
        if (left > right) return null;

        int rootVal = postorder[postIndex--];
        TreeNode root = new TreeNode(rootVal);
        int index = inorderMap.get(rootVal);

        root.right = build(inorder, postorder, index + 1, right);
        root.left = build(inorder, postorder, left, index - 1);

        return root;
    }
}
```

The screenshot displays the LeetCode submission interface for problem 106. The top section shows the problem title and a link to the description. Below this, the submission status is 'Accepted' with a runtime of 0 ms. The 'Testcase' section shows the input and output for Case 1. The input is inorder = [9, 3, 15, 20, 7] and postorder = [9, 15, 7, 20, 3]. The output is [3, 9, 20, null, null, 15, 7]. The 'Runtime' section shows a bar chart with a peak at 1 ms, indicating a 97.00% beat rate. The 'Code' section shows the Java code for the solution.

Accepted Runtime: 0 ms

Case 1

Input

inorder = [9, 3, 15, 20, 7]

postorder = [9, 15, 7, 20, 3]

Output

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

Testcase

Case 1 Case 2

inorder = [9, 3, 15, 20, 7]

postorder = [9, 15, 7, 20, 3]

Code Java

```
import java.util.*;

class Solution {
    private int postIndex;
    private Map<Integer, Integer> inorderMap;

    public TreeNode buildTree(int[] inorder, int[] postorder) {
        postIndex = postorder.length - 1;
        inorderMap = new HashMap<>();
        for (int i = 0; i < inorder.length; i++) {
            inorderMap.put(inorder[i], i);
        }
        return build(inorder, postorder, 0, inorder.length - 1);
    }

    private TreeNode build(int[] inorder, int[] postorder, int left, int right) {
        if (left > right) return null;

        int rootVal = postorder[postIndex--];
        TreeNode root = new TreeNode(rootVal);
        int index = inorderMap.get(rootVal);

        root.right = build(inorder, postorder, index + 1, right);
        root.left = build(inorder, postorder, left, index - 1);

        return root;
    }
}
```

513. Find Bottom Left Tree Value

<https://leetcode.com/problems/find-bottom-left-tree-value/description/>

```
import java.util.*;

class Solution {
    public int findBottomLeftValue(TreeNode root) {
        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);
        int leftmost = root.val;

        while (!q.isEmpty()) {
            int size = q.size();
            leftmost = q.peek().val;

            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
        }
        return leftmost;
    }
}
```

The screenshot displays the LeetCode submission page for problem 513. The top section shows the problem title and a link to the description. Below this, the submission status is 'Accepted' with 79/79 testcases passed. The user 'Jiya' submitted the solution on Mar 03, 2025 at 17:14. The performance metrics show a runtime of 3 ms, beating 56.19% of solutions, and a memory usage of 45.31 MB, beating 11.41% of solutions. A bar chart visualizes the runtime performance across different time intervals. The test results section shows Case 1 with input [2,1,3], output 1, and expected 1. The code editor at the bottom shows the Java solution.

Runtime Performance:

Time Interval	Percentage of Solutions Beaten
1ms	~20%
2ms	~10%
3ms	~40%
4ms	~5%

Testcase Results:

Case	Input	Output	Expected
Case 1	[2,1,3]	1	1

124. Binary Tree Maximum Path Sum

<https://leetcode.com/problems/binary-tree-maximum-path-sum/description/>

```
class Solution {
    private int maxSum = Integer.MIN_VALUE;

    public int maxPathSum(TreeNode root) {
        dfs(root);
        return maxSum;
    }

    private int dfs(TreeNode node) {
        if (node == null) return 0;

        int left = Math.max(0, dfs(node.left));
        int right = Math.max(0, dfs(node.right));

        maxSum = Math.max(maxSum, left + right + node.val);

        return Math.max(left, right) + node.val;
    }
}
```

The screenshot displays the LeetCode submission interface for problem 124. The submission is marked as 'Accepted' with 96/96 testcases passed. The runtime is 0 ms, which is faster than 100.00% of other submissions. The memory usage is 44.44 MB, which is less than 49.33% of other submissions. The test case input is 'root = [1,2,3]' and the output is '6'. The code is written in Java and is shown in the 'Code' tab. The submission was made by 'Jiya' on March 03, 2025, at 17:16.

Accepted 96 / 96 testcases passed
Jiya submitted at Mar 03, 2025 17:16

Runtime: 0 ms | Beats 100.00%
Memory: 44.44 MB | Beats 49.33%

Testcase: Case 1 Case 2 +
root = [1,2,3]
Output: 6
Expected: 6

```
class Solution {
    private int maxSum = Integer.MIN_VALUE;

    public int maxPathSum(TreeNode root) {
        dfs(root);
    }

    private int dfs(TreeNode node) {
        if (node == null) return 0;

        int left = Math.max(0, dfs(node.left));
        int right = Math.max(0, dfs(node.right));

        maxSum = Math.max(maxSum, left + right + node.val);

        return Math.max(left, right) + node.val;
    }
}
```

987. Vertical Order Traversal of a Binary Tree

<https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/description/>

```
import java.util.*;
class Solution {
    public List<List<Integer>> verticalTraversal(TreeNode root) {
        TreeMap<Integer, TreeMap<Integer, PriorityQueue<Integer>>> map = new
        TreeMap<>();
        dfs(root, 0, 0, map);
        List<List<Integer>> result = new ArrayList<>();
        for (TreeMap<Integer, PriorityQueue<Integer>> ys : map.values()) {
            List<Integer> column = new ArrayList<>();
            for (PriorityQueue<Integer> nodes : ys.values()) {
                while (!nodes.isEmpty()) {
                    column.add(nodes.poll());
                }
            }
            result.add(column);
        }
        return result;
    }
    private void dfs(TreeNode node, int x, int y, TreeMap<Integer, TreeMap<Integer,
    PriorityQueue<Integer>>> map) {
        if (node == null) return;

        map.putIfAbsent(x, new TreeMap<>());
        map.get(x).putIfAbsent(y, new PriorityQueue<>());
        map.get(x).get(y).offer(node.val);

        dfs(node.left, x - 1, y + 1, map);
        dfs(node.right, x + 1, y + 1, map);
    }
}
```

The screenshot displays the LeetCode submission interface for problem 987. The left sidebar shows the problem description and submission details, including a bar chart of runtime performance. The main area shows the test case result, which is 'Accepted'. The input is a binary tree with root [3, 9, 20, null, null, 15, 7] and the output is a list of lists representing the vertical order traversal: [[9], [3, 15], [20], [7]].

Runtime Performance:

Runtime	Beats
2 ms	99.62%

Test Case Result:

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

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

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

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

Testcase: Case 1 Case 2 Case 3

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

Source