

Experiment No. - 5

Student Name: Vivek Kumar

Branch: BE-CSE(LEET)

Semester: 6th

Subject Name: Competitive coding - II

UID: 21BCS8129

Section/Group: 20BCS-ST-801/B

Date of Performance: 07/03/2023

Subject Code: 20CSP-351

1. Aim/Overview of the practical:

Q.1 Balance Binary Tree.

<https://leetcode.com/problems/balanced-binary-tree/>

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

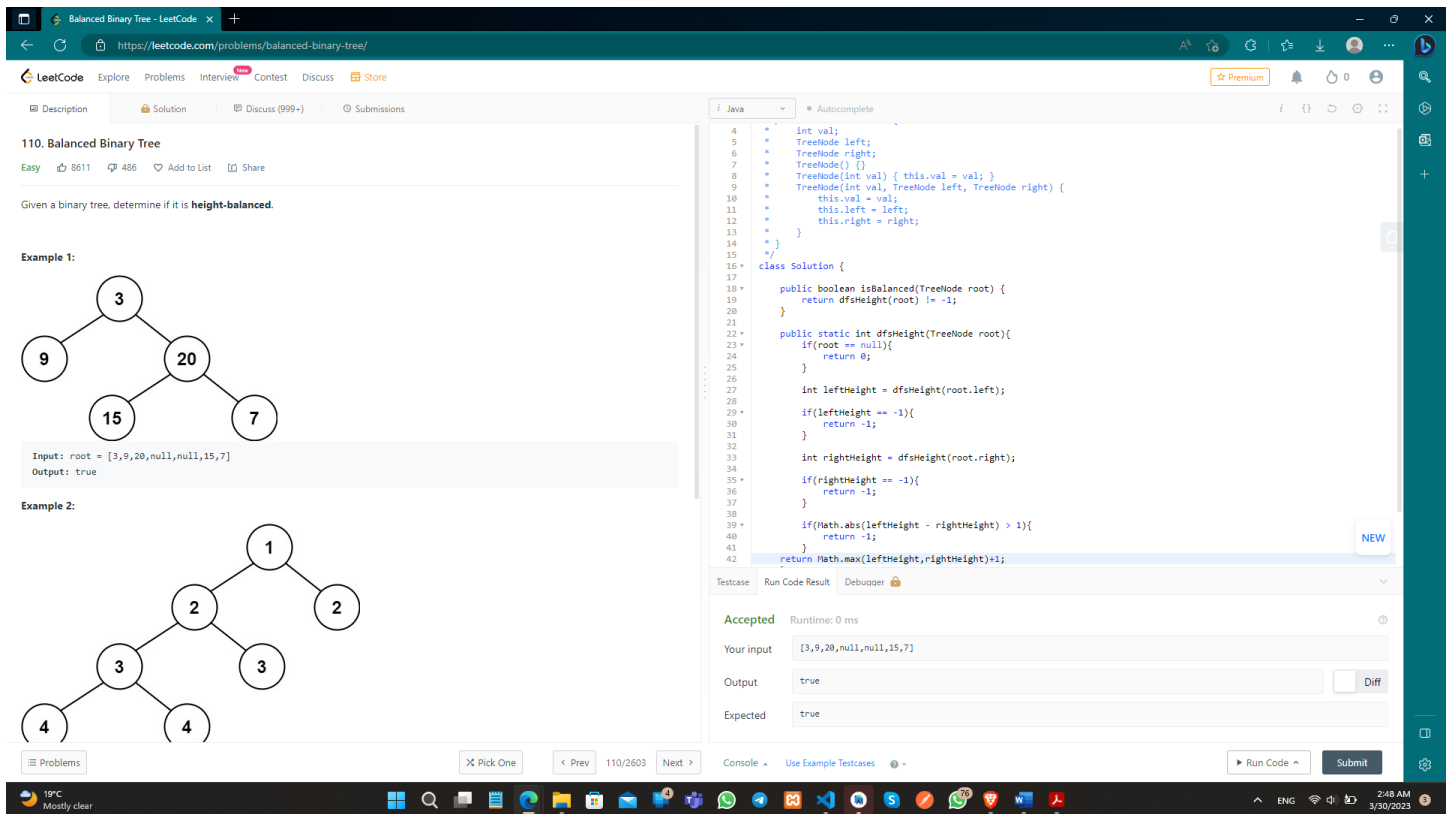
- To understand the concept of Tree
- To implement the concept of Balance Binary Tree.

4. Code:

```
class Solution {  
  
    public boolean isBalanced(TreeNode root) {  
        return dfsHeight(root) != -1;  
    }  
  
    public static int dfsHeight(TreeNode root){  
        if(root == null){  
            return 0;  
        }  
  
        int leftHeight = dfsHeight(root.left);  
  
        if(leftHeight == -1){  
            return -1;  
        }  
  
        int rightHeight = dfsHeight(root.right);  
  
        if(rightHeight == -1){  
            return -1;  
        }  
  
        if(Math.abs(leftHeight - rightHeight) > 1){  
            return -1;  
        }  
        return Math.max(leftHeight, rightHeight) + 1;  
    }  
}
```

Submitted By: Vivek Kumar

5. Result/Output/Writing Summary:



110. Balanced Binary Tree

Easy 8611 486 Add to List Share

Given a binary tree, determine if it is **height-balanced**.

Example 1:

```

    3
   / \
  9  20
     / \
    15  7
  
```

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

Example 2:

```

    1
   / \
  2   2
 / \ / \
3  3 3  3
/ \ / \
4  4 4  4
  
```

```

class Solution {
    public boolean isBalanced(TreeNode root) {
        return dfsHeight(root) != -1;
    }

    public static int dfsHeight(TreeNode root) {
        if (root == null) {
            return 0;
        }

        int leftHeight = dfsHeight(root.left);
        if (leftHeight == -1) {
            return -1;
        }

        int rightHeight = dfsHeight(root.right);
        if (rightHeight == -1) {
            return -1;
        }

        if (Math.abs(leftHeight - rightHeight) > 1) {
            return -1;
        }

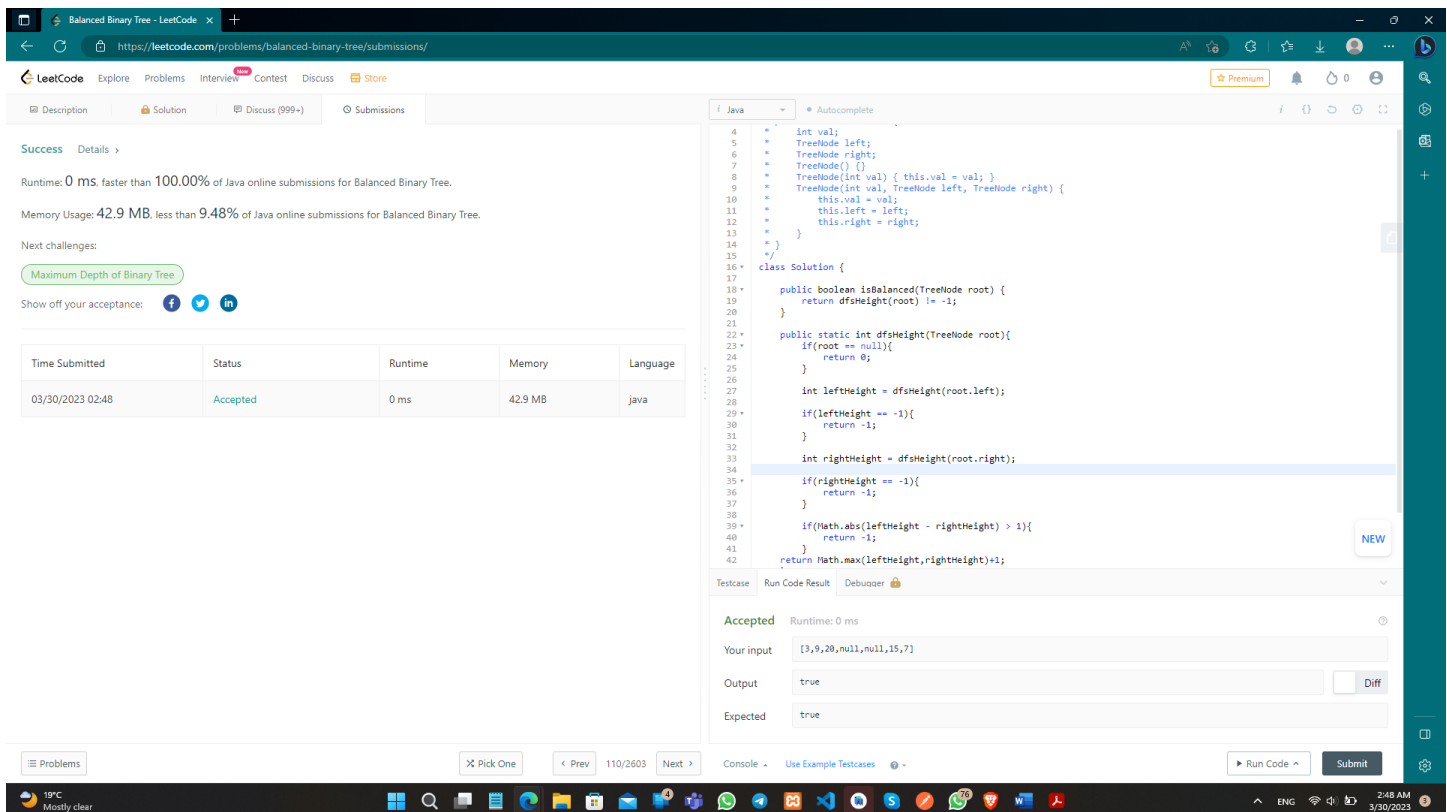
        return Math.max(leftHeight, rightHeight) + 1;
    }
}
  
```

Accepted Runtime: 0 ms

Your input: [3,9,20,null,null,15,7]

Output: true

Expected: true



Success Details

Runtime: 0 ms, faster than 100.00% of Java online submissions for Balanced Binary Tree.

Memory Usage: 42.9 MB, less than 9.48% of Java online submissions for Balanced Binary Tree.

Next challenges: [Maximum Depth of Binary Tree](#)

Show off your acceptance: [Facebook](#) [Twitter](#) [LinkedIn](#)

Time Submitted	Status	Runtime	Memory	Language
03/30/2023 02:48	Accepted	0 ms	42.9 MB	java

```

class Solution {
    public boolean isBalanced(TreeNode root) {
        return dfsHeight(root) != -1;
    }

    public static int dfsHeight(TreeNode root) {
        if (root == null) {
            return 0;
        }

        int leftHeight = dfsHeight(root.left);
        if (leftHeight == -1) {
            return -1;
        }

        int rightHeight = dfsHeight(root.right);
        if (rightHeight == -1) {
            return -1;
        }

        if (Math.abs(leftHeight - rightHeight) > 1) {
            return -1;
        }

        return Math.max(leftHeight, rightHeight) + 1;
    }
}
  
```

Accepted Runtime: 0 ms

Your input: [3,9,20,null,null,15,7]

Output: true

Expected: true

1. Aim/Overview of the practical:

Q.2 Path Sum

<https://leetcode.com/problems/path-sum/>

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Tree traversal.
- To implement the concept of calculate the path sum.

4. Code:

```
class Solution {  
    public boolean hasPathSum(TreeNode root, int targetSum) {  
        if (root == null) {  
            return false;  
        }  
        if (root.val == targetSum && root.left == null && root.right == null) {  
            return true;  
        }  
        return hasPathSum(root.left, targetSum - root.val) || hasPathSum(root.right, targetSum - root.val);  
    }  
}
```

5. Result/Output/Writing Summary:

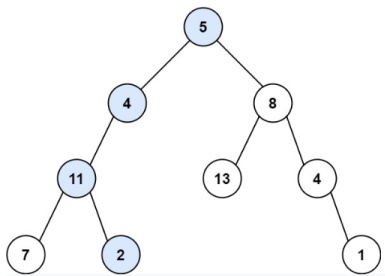
112. Path Sum

Easy 8105 928 Add to List Share

Given the `root` of a binary tree and an integer `targetSum`, return `true` if the tree has a **root-to-leaf** path such that adding up all the values along the path equals `targetSum`.


A **leaf** is a node with no children.

Example 1:



Input: `root = [5,4,8,11,null,13,4,7,2,null,null,1]`, `targetSum = 22`
Output: `true`
Explanation: The root-to-leaf path with the target sum is shown.

Example 2:



```

1  /**
2   * Definition for a binary tree node.
3   * public class TreeNode {
4   *     int val;
5   *     TreeNode left;
6   *     TreeNode right;
7   *     TreeNode() {}
8   *     TreeNode(int val) { this.val = val; }
9   *     TreeNode(int val, TreeNode left, TreeNode right) {
10  *         this.val = val;
11  *         this.left = left;
12  *         this.right = right;
13  *     }
14  * }
15  */
16  class Solution {
17  *     public boolean hasPathSum(TreeNode root, int targetSum) {
18  *         if (root == null) {
19  *             return false;
20  *         }
21  *         if (root.val == targetSum && root.left == null && root.right == null) {
22  *             return true;
23  *         }
24  *         return hasPathSum(root.left, targetSum - root.val) || hasPathSum(root.right, targetSum - root.val);
25  *     }
26  }

```

Testcase Run Code Result Debbuger

Accepted Runtime: 0 ms

Your input: `[5,4,8,11,null,13,4,7,2,null,null,1]`
22

Output: `true` ☐ Diff

Expected: `true`

Console Use Example Testcases

Run Code Submit

Success Details

Runtime: 0 ms. faster than 100.00% of Java online submissions for Path Sum.

Memory Usage: 42.5 MB. less than 35.59% of Java online submissions for Path Sum.

Next challenges:

Path Sum II Binary Tree Maximum Path Sum

Sum Root to Leaf Numbers Path Sum III Path Sum IV

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
03/30/2023 02:53	Accepted	0 ms	42.5 MB	java

```

1  /**
2   * Definition for a binary tree node.
3   * public class TreeNode {
4   *     int val;
5   *     TreeNode left;
6   *     TreeNode right;
7   *     TreeNode() {}
8   *     TreeNode(int val) { this.val = val; }
9   *     TreeNode(int val, TreeNode left, TreeNode right) {
10  *         this.val = val;
11  *         this.left = left;
12  *         this.right = right;
13  *     }
14  * }
15  */
16  class Solution {
17  *     public boolean hasPathSum(TreeNode root, int targetSum) {
18  *         if (root == null) {
19  *             return false;
20  *         }
21  *         if (root.val == targetSum && root.left == null && root.right == null) {
22  *             return true;
23  *         }
24  *         return hasPathSum(root.left, targetSum - root.val) || hasPathSum(root.right, targetSum - root.val);
25  *     }
26  }

```

Testcase Run Code Result Debbuger

Accepted Runtime: 0 ms

Your input: `[5,4,8,11,null,13,4,7,2,null,null,1]`
22

Output: `true` ☐ Diff

Expected: `true`

Console Use Example Testcases

Run Code Submit

Learning outcomes (What I have learnt):

- Learned the concept of Balanced Binary Tree.
- Learnt about Tree and Path Sum.