



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment-2.1

Student Name: Nabha Varshney

UID: 20BCS4995

Branch: CSE

Section/Group: 20BCS-DM-704 (A)

Semester: 6<sup>th</sup>

Date of Performance: 05<sup>th</sup> Apr 2023

Subject Name: Competitive Coding II

Subject Code: 20CSP- 351

---

**Aim** – To demonstrate the concept of Trees

### **Objective-**

- ♦ The objective is to build problem solving capability and to learn the basic concepts of data structures.
- ♦ The implementation of balanced binary tree which shows and brushes up the concept of Trees and can be solved through various approaches.
- ♦ The implementation of path sum problem in C++.

### **1) Balanced Binary Tree**

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

### **Code –**

```
class Solution {
public boolean isBalanced(TreeNode root)
{
    return height(root)!=-1;
}
public int height(TreeNode node){
    if(node==null)
    {
        return 0;
    }
    int leftHeight=height(node.left);
```

```

    if(leftHeight== -1)
        return -1;
    int rightHeight=height(node.right);

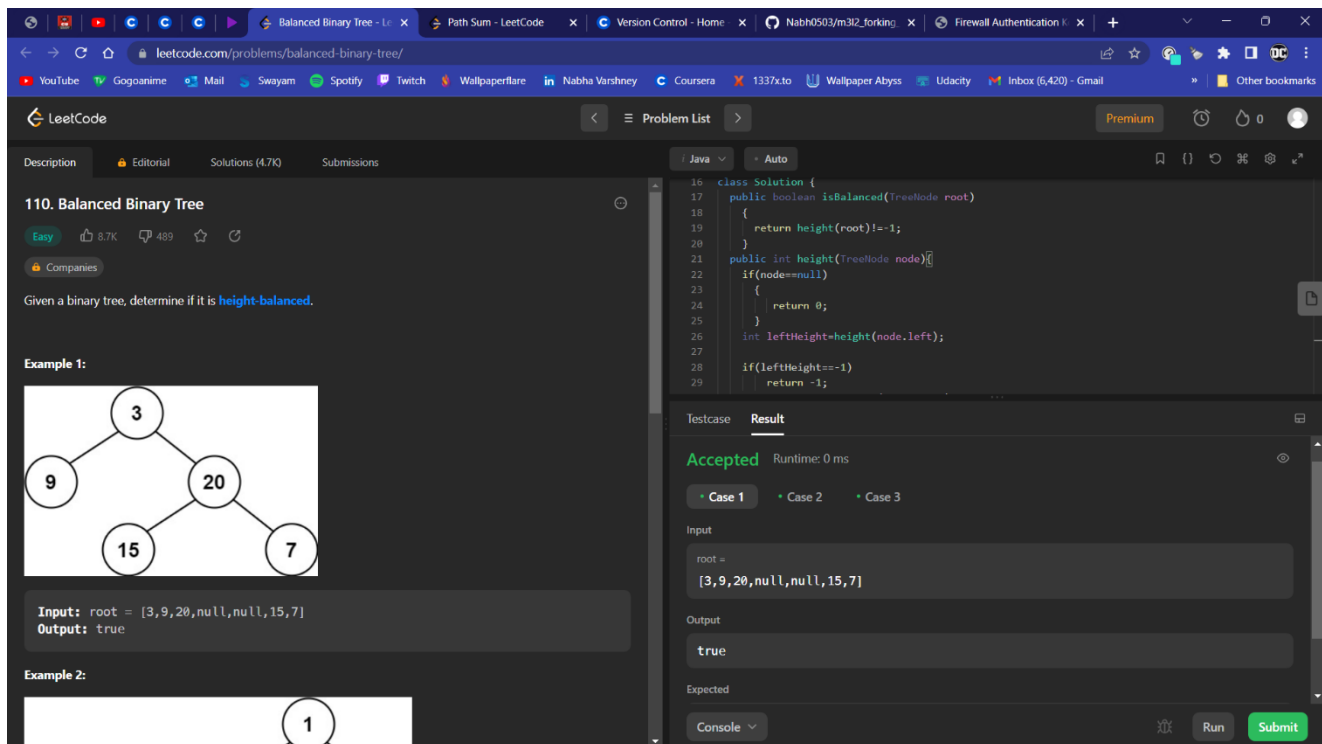
    if(rightHeight== -1)
        return -1;

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

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

```

## Output -



**110. Balanced Binary Tree**

Easy 8.7K 489

Companies

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

**Example 1:**

```

graph TD
    3((3)) --- 9((9))
    3 --- 20((20))
    20 --- 15((15))
    20 --- 7((7))

```

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

**Example 2:**

```

graph TD
    1((1))

```

```

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

```

Testcase Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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

Output

true

Expected

Console Run Submit

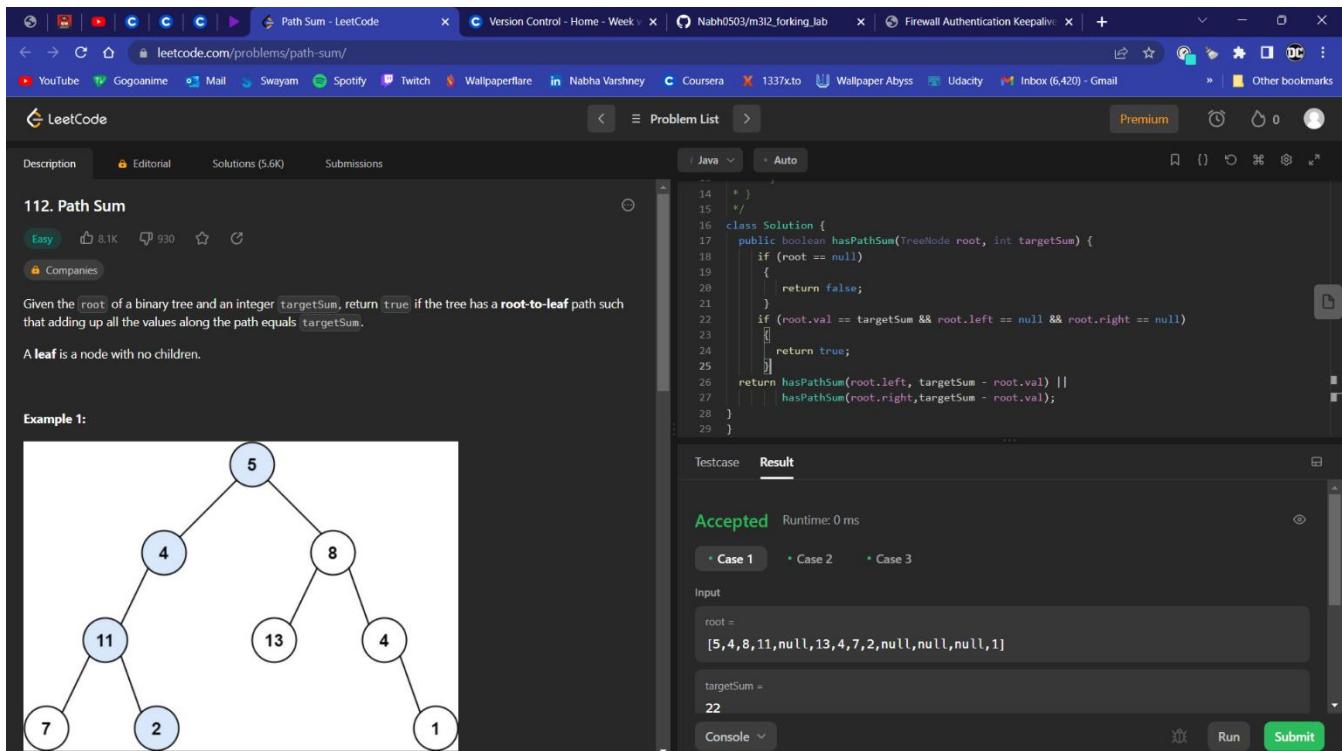
## 2) Path Sum

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

### 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);
    }
}
```

### Output –



**112. Path Sum**

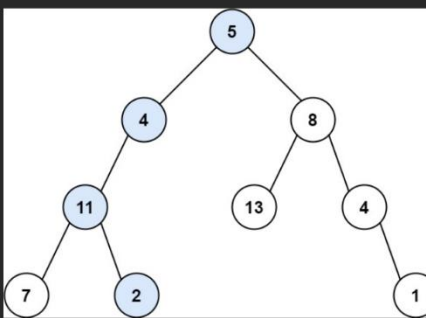
Easy 8.1K 930

Companies

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:**



```
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);
    }
}
```

Testcase Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

root =  
[5,4,8,11,null,13,4,7,2,null,null,null,1]

targetSum =  
22

Console Run Submit