EXPERIMENT - 5

Student Name: Sandeep Kumar **UID: 20BCS4885**

Branch: CSE **Section/Group: 603/A**

Semester: 6th semester **Subject: Competitive Coding**

Aim: To demonstrate the concept of Trees.

Objective:

a) Balanced binary tree
b) Path sum

Problem 1: Balanced binary tree

Solution code:

```
class Solution {
public:
    bool isBalanced(TreeNode* root) {
        if (root == NULL)  return true;
                if (Height(root) == -1)  return false;
                return true;
        }
        int Height(TreeNode* root) {
                if (root == NULL)  return 0;
                int leftHeight = Height(root->left);
                int rightHight = Height(root->right);
                if (leftHeight == -1 || rightHight == -1 || abs(leftHeight - rightHight) > 1)  return -1;
                return max(leftHeight, rightHight) + 1;
    }
};
```

Explanation of code:

The solution uses a recursive approach to calculate the height of the left and right subtrees of each node, and then check if their difference is greater than 1. If it is, the function returns -1, which is used as a flag to indicate that the tree is not balanced. If at any point in the recursion, the function returns -1, the isBalanced function will return false.

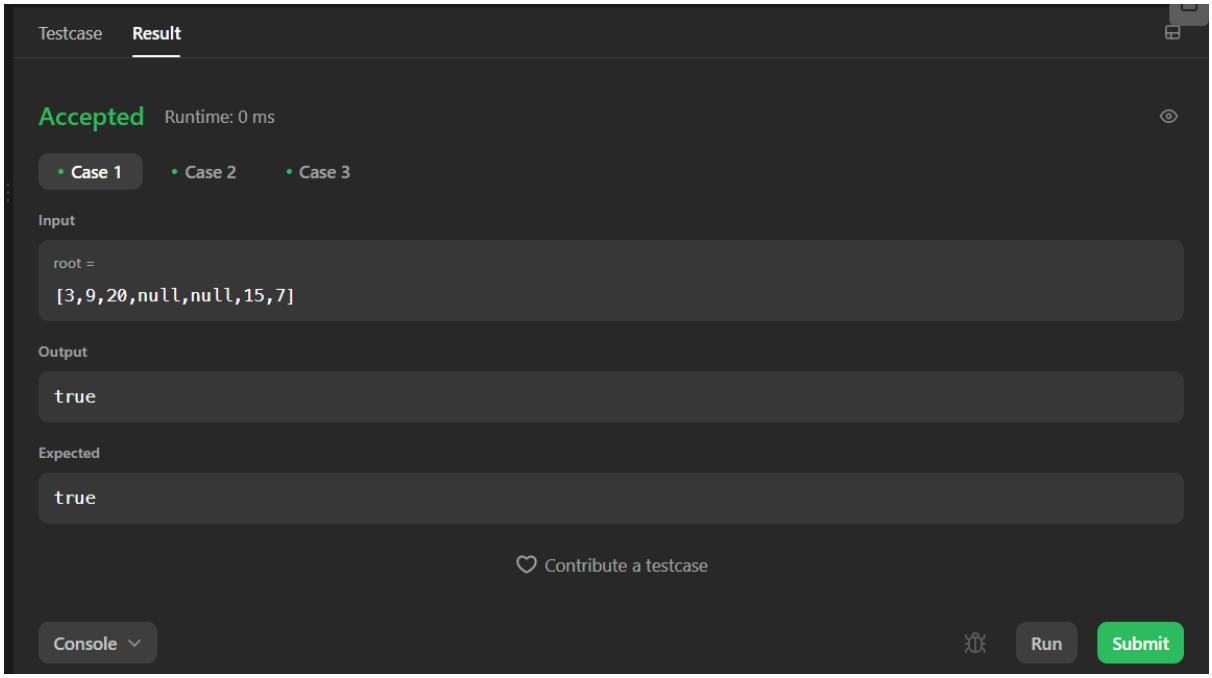
If the recursion reaches the bottom of the tree (i.e., a leaf node is reached), the function returns 0, and the height is calculated as the maximum height of the left and right subtrees, plus 1. If at the end of the recursion, no node has a height difference greater than 1, the function returns true.

Complexity:

Time Complexity: O(n)

Space Complexity: O(1)

Output:



Problem 2: Path Sum

Input Code:

```
class Solution {
public:
   bool hasPathSum(TreeNode* root, int targetSum) {
      if (!root)
         return false;

      if (root->val == targetSum and !root->left and !root->right)
         return true;

      return hasPathSum(root->left, targetSum - root->val) or hasPathSum(root->right, targetSum
- root->val);
   }
};
```

Explanation of code:

- The given code defines a class Solution and a member function hasPathSum that takes a pointer to a binary tree node and an integer targetSum as input and returns a boolean value indicating whether there exists a root-to-leaf path in the binary tree such that the sum of values along the path equals the targetSum.

- The function first checks if the root is null. If it is null, it returns false as there is no path to traverse. If the current node's value equals the targetSum and it is a leaf node (i.e., both the left and right child nodes are null), then it returns true indicating that there exists a root-to-leaf path that sums up to targetSum.

- If the above conditions are not met, the function recursively checks whether there is a path in the left subtree or the right subtree of the current node that sums up to (targetSum - current node's value). The function uses the logical OR operator to combine the results of the left and right subtree checks.

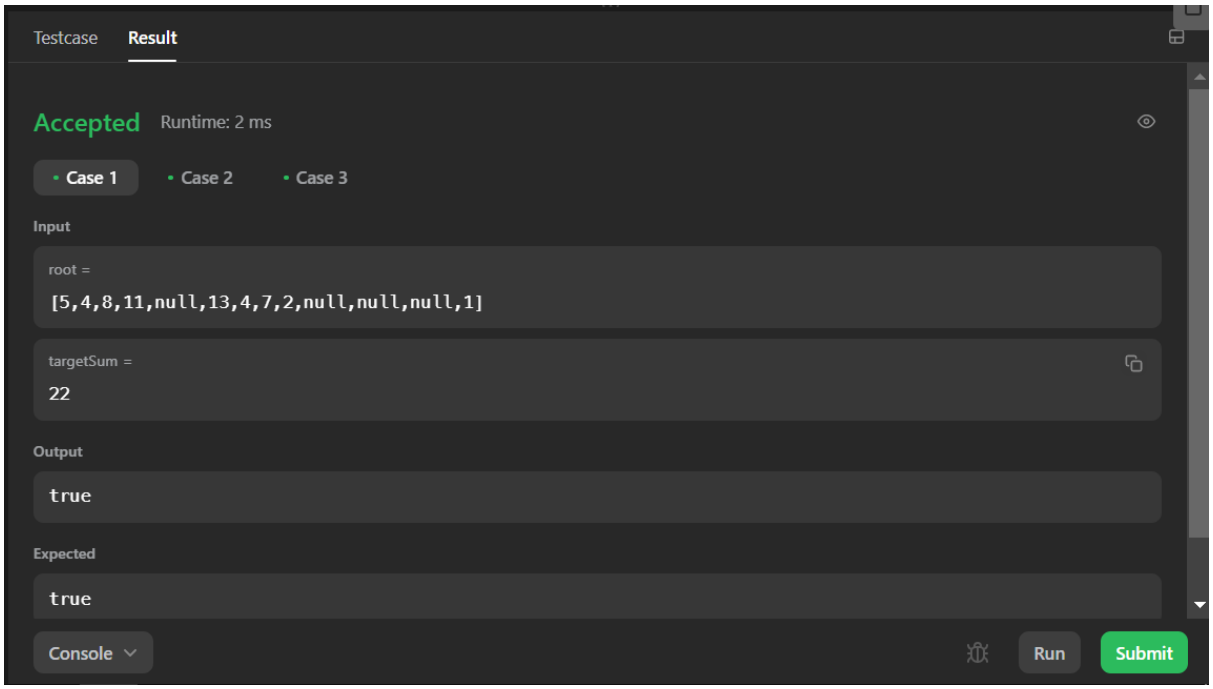
Overall, the function implements a depth-first search (DFS) algorithm to traverse the binary tree in a recursive manner and check for the existence of a root-to-leaf path that sums up to the given targetSum.

Complexity:

Time Complexity: O(n) [worst case : Traverse all n nodes]

Space Complexity: O(1)

Output:


Testcase
Result
Accepted
Runtime: 2 ms
Case 1
Case 2
Case 3
Input
root =
[5,4,8,11,null,13,4,7,2,null,null,null,1]
targetSum =
22
Output
true
Expected
true
Console
Run
Submit