Children Sum Parent

Easy Accuracy: 51.58% Submissions: 73K+ Points: 2

Given a Binary Tree. Check whether all of its nodes have the value equal to the sum of their child nodes.

Example 1:

Input:

```
      10
     /
   10
```

Output: 1

Explanation: Here, every node is sum of its left and right child.

Example 2:

C++ (g++ 5.4) | Average Time: 20m | Start Timer

```
// } Driver Code Ends


class Solution{
    public:
    //Function to check whether all nodes of a tree have the value
    //equal to the sum of their child nodes.
    int isSumProperty(Node *root)
    {
     // Add your code here
     if(root == NULL) return 1;
     if(root->left== NULL && root->right == NULL) return 1;
     int sum = 0;
     if(root->left != NULL){
         sum+=root->left->data;
     }
     if(root->right != NULL){
         sum+=root->right->data;
     }
     return (root->data == sum && isSumProperty(root->left) && isSumProperty(root->right));
    }
};
// } Driver Code Ends
```

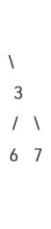
Custom Input | Compile & Run | Submit

Maximum Width of Tree

Easy Accuracy: 63.27% Submissions: 59K+ Points: 2

Given a Binary Tree, find the maximum width of it. **Maximum width** is defined as the maximum number of nodes at any level.

For example, the maximum width of the following tree is 4 as there are 4 nodes at the 3rd level.



Example 1:

Input:

1


```cpp
// } Driver Code Ends


class Solution {
  public:
    // Function to get the maximum width of a binary tree.
    int getMaxWidth(Node* root) {

        int res =0;
        queue<Node*>q;
        q.push(root);

        while(!q.empty()){
            int count = q.size();

            res = max(res,count);
            for(int i=0;i<count;i++){
                Node*curr = q.front();
                q.pop();

                if(curr->left!=NULL)
                q.push(curr->left);

                if(curr->right!=NULL)
                q.push(curr->right);
            }
        }
        return res;
    }
};

// } Driver Code Ends
```

Lowest Common Ancestor in a Binary Tree

Medium Accuracy: 52.85% Submissions: 137K+ Points: 4

Given a Binary Tree with all **unique** values and two nodes value, **n1** and **n2**. The task is to find the **lowest common ancestor** of the given two nodes. We may assume that either both n1 and n2 are present in the tree or none of them are present.

LCA: It is the first common ancestor of both the nodes n1 and n2 from bottom of tree.

Example 1:

Input:
n1 = 2 , n2 = 3

Output: 1
Explanation:

C++ (g++ 5.4) Average Time: 20m Start Timer

```cpp
// } Driver Code Ends


class Solution
{
    public:
    //Function to return the lowest common ancestor in a Binary Tree.
    Node* lca(Node* root ,int n1 ,int n2 )
    {
        if(root == NULL){
            return NULL;
        }
        if(root->data == n1 || root->data == n2){
            return root;
        }
        Node* left_path = lca(root->left, n1, n2);
        Node* right_path = lca(root->right, n1, n2);
        if(left_path == NULL){
            return right_path;
        }
        else if(right_path == NULL){
            return left_path;
        }
        return root;
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit