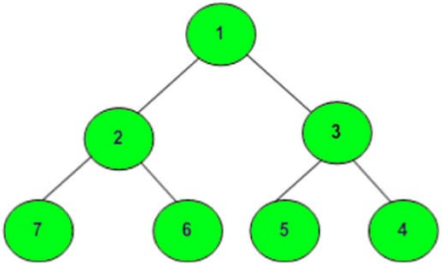
Problem | Editorial | Submissions | Comments

Level order traversal in spiral form

Easy | Accuracy: 36.43% | Submissions: 156K+ | Points: 2

Complete the function to find spiral order traversal of a tree. For below tree, function should return 1, 2, 3, 4, 5, 6, 7.



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

```
// } Driver Code Ends



//Function to return a list containing the level order traversal in spiral form.
vector<int> findSpiral(Node *root)
{
    //Your code here
    stack<Node *> s1,s2;
    vector<int> v;
    if(root == NULL) return v;
    s1.push(root);
    while(s1.empty() ==false || s2.empty() == false){
        while(!s1.empty()){
            Node *temp = s1.top();
            v.push_back(temp->data);
            s1.pop();
            if(temp->right != NULL) s2.push(temp->right);
            if(temp->left != NULL) s2.push(temp->left);
        }
        while(!s2.empty()){
            Node *temp = s2.top();
            v.push_back(temp->data);
            s2.pop();
            if(temp->left != NULL) s1.push(temp->left);
            if(temp->right != NULL) s1.push(temp->right);

        }
    }
    return v;
}
```

Custom Input | Compile & Run | Submit

Height of Binary Tree

Easy Accuracy: 78.58% Submissions: 195K+ Points: 2

Given a binary tree, find its height.

Example 1:

Input:

```
    1
   / \
  2   3
```

Output: 2

Example 2:

Input:

```
  2
   \
```

C++ (g++ 5.4) Average Time: 15m Start Timer

```
// } Driver Code Ends
//User function template for C++


class Solution{
    public:
    //Function to find the height of a binary tree.
    int height(struct Node* node){
        // code here
        if(node==NULL) return 0;
        int l = height(node->left);
        int r = height(node->right);

        return 1+ max(l,r);
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Check for Balanced Tree

Easy Accuracy: 43.15% Submissions: 209K+ Points: 2

Given a binary tree, find if it is height balanced or not.

A tree is height balanced if difference between heights of left and right subtrees is **not more than one** for all nodes of tree.

A height balanced tree

```
    1
  /   \
 10    39
/
5
```

An unbalanced tree

```
    1
  /
 10
/
```

```
90
91
92
93  class Solution{
94      public:
95      //Function to check whether a binary tree is balanced or not.
96      int height(Node* root)
97      {
98          if(root==NULL) return 0;
99
100         int lh=height(root->left);
101         int rh=height(root->right);
102         return max(lh,rh)+1;
103     }
104
105     bool isBalanced(Node *root)
106     {
107         //  Your Code here
108         if(root == NULL) return true;
109
110
111         if(isBalanced(root->left) == false) return false;
112
113         if(isBalanced(root->right) == false)  return false;
114
115
116         int left_h = height(root->left);
117         int right_h = height(root->right);
118
119         if(abs(left_h - right_h) <= 1) return true;
120
121         return false;
122     }
123 };
124
125 // } Driver Code Ends
```

Custom Input | Compile & Run | Submit