Problem | Editorial | Submissions | Comments

Symmetric Tree

Easy Accuracy: **44.96%** Submissions: **97K+** Points: **2**

Given a Binary Tree. Check whether it is Symmetric or not, i.e. whether the binary tree is a **Mirror image of itself** or not.

Example 1:

Input:

Output: True

Explanation: Tree is mirror image of itself i.e. tree is symmetric

Example 2:

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

```
// } Driver Code Ends


class Solution{
    public:
    // return true/false denoting whether the tree is Symmetric or not
    bool check(Node* r1, Node* r2){

        if(r1 == NULL && r2 == NULL)
            return true;
        if(r1 == NULL || r2 == NULL)
            return false;
        if(r1->data != r2->data)
            return false;

        if(check(r1->left, r2->right) == false || check(r1->right, r2->left) == false)
            return false;
        return true;

    }

    bool isSymmetric(struct Node* root){
        if(root == NULL)
            return true;

        return check(root->left, root->right);
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Vertical Traversal of Binary Tree

Medium Accuracy: 32.87% Submissions: 147K+ Points: 4

Given a Binary Tree, find the vertical traversal of it starting from the leftmost level to the rightmost level.
If there are multiple nodes passing through a vertical line, then they should be printed as they appear in **level order** traversal of the tree.

Example 1:

Input:

Output:

C++ (g++ 5.4) Average Time: 45m Start Timer

```cpp
// } Driver Code Ends
class Solution
{
    public:
    //Function to find the vertical order traversal of Binary Tree.
    vector<int> verticalOrder(Node *root)
    {
        vector<int> res;
        if(root == NULL)
            return {};
        queue< pair<Node*, int> > q;
        map<int, vector<int> > mp;

        q.push({root, 0});

        while( !q.empty() ) {
            root = q.front().first;
            int col = q.front().second;
            q.pop();

            mp[col].emplace_back(root->data);

            if(root->left)
                q.push({root->left, col - 1});
            if(root->right)
                q.push({root->right, col + 1});
        }

        for(auto &i : mp) {
            res.insert(res.end(), i.second.begin(), i.second.end() );
        }
        return res;
    }
};
```

Custom Input | Compile & Run | Submit