Top View of Binary Tree

Medium Accuracy: 38.43% Submissions: 201K+ Points: 4

Given below is a binary tree. The task is to print the top view of binary tree. Top view of a binary tree is the set of nodes visible when the tree is viewed from the top. For the given below tree

```
    1
  /   \
 2     3
/ \   / \
4  5 6  7
```

Top view will be: 4 2 1 3 7

Note: Return nodes from **leftmost** node to **rightmost** node. Also if 2 nodes are outside the shadow of the tree and are at same position then consider the extreme ones only(i.e. leftmost and rightmost).

For ex - **1 2 3 N 4 5 N 6 N 7 N 8 N 9 N N N N N** will give **8 2 1 3** as answer. Here 8 and 9 are on the same position but 9 will get shadowed.

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

```cpp
// } Driver Code Ends


class Solution
{
    public:
    //Function to return a list of nodes visible from the top view
    //from left to right in Binary Tree.
    vector<int> topView(Node *root)
    {
        map<int,vector<int>> mp;
        vector<int> res;
        queue<pair<Node*,int>> q;

        q.push({root,0});
        while(!q.empty()){
            int n = q.size();
            while(n--){
                Node* temp = q.front().first;
                int level = q.front().second;
                mp[level].push_back(temp->data);
                q.pop();

                if(temp->left)
                q.push({temp->left,level-1});
                if(temp->right)
                q.push({temp->right,level+1});
            }
        }
        for(auto & it : mp){
            res.push_back(it.second[0]);
        }
        return res;
    }

};
```

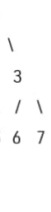
Custom Input | Compile & Run | Submit

Left View of Binary Tree

Easy Accuracy: 33.74% Submissions: 372K+ Points: 2

Given a Binary Tree, return Left view of it. Left view of a Binary Tree is set of nodes visible when tree is visited from Left side. The task is to complete the function **leftView()**, which accepts root of the tree as argument.

Left view of following tree is 1 2 4 8.



Example 1:

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

```
// } Driver Code Ends


//Function to return a list containing elements of left view of the binary tree.
vector<int> leftView(Node *root)
{
        vector<int>res;
        queue<Node*>q;
        if(root==0){
            return res;
        }
        q.push(root);
        while(!q.empty()){
            int s=q.size();
            res.push_back(q.front()->data);
            while(s--){
                Node *t=q.front();
                q.pop();
                if(t->left) q.push(t->left);
                if(t->right) q.push(t->right);
            }
        }
        return res;
}

```

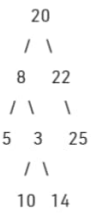
Custom Input | Compile & Run | Submit

Problem | Editorial | Submissions | Comments

Bottom View of Binary Tree

Medium Accuracy: 54.18% Submissions: 160K+ Points: 4

Given a binary tree, print the bottom view from left to right.
A node is included in bottom view if it can be seen when we look at the tree from bottom.



For the above tree, the bottom view is 5 10 3 14 25.
If there are **multiple** bottom-most nodes for a horizontal distance from root, then print the later one in level traversal. For example, in the below diagram, 3 and 4 are both the bottommost nodes at horizontal distance 0,

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

```cpp
// } Driver Code Ends
//Function to return a list containing the bottom view of the given tree.

class Solution {
  public:
    vector <int> bottomView(Node *root) {
        // Your Code Here
        vector<int>res;
        if(root==NULL) return {};
        queue<pair<Node*,int>> q;
        map<int,int> m;
        q.push({root,0});
        while(!q.empty()){
            auto temp=q.front();
            q.pop();
            Node* node=temp.first;
            int level=temp.second;
            m[level]=node->data;
            if(node->left) q.push({node->left,level-1});
            if(node->right) q.push({node->right,level+1});
        }
        for(auto i:m){
            res.push_back(i.second);
        }
        return res;
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit