given tree.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(N).

Constraints:
1 <= Number of nodes <= 10^5
1 <= Data of a node <= 10^5

Note: The **Input/Output** format and **Example** given are used for the system's internal purpose, and should be used by a user for **Expected Output** only. As it is a function problem, hence a user should not read any input from the stdin/console. The task is to complete the function specified, and not to write the full code.

View Bookmarked Problems (https://practice.geeksforgeeks.org/explore/?problemType=bookmark)

Company Tags

Topic Tags

Related Courses

Related Interview Experiences

We are replacing the old Disqus forum with the new Discussions section given below. Click here (https://practice.geeksforgeeks.org/comments/bottom-view-of-binary-tree/1/?rel=https://practice.geeksforgeeks.org/problems/bottom-view-of-binary-tree/1) to view old Disqus comments.

Discussions

```
98      void solve(Node* root, map<int, pair<int, int>>& levelTraverse, int hLevel, int vLevel){
99          if(root == NULL)
100             return;
101
102         if(levelTraverse.find(hLevel) == levelTraverse.end())
103             levelTraverse[hLevel] = make_pair(root->data, vLevel);
104         else if(vLevel >= levelTraverse[hLevel].second)
105             levelTraverse[hLevel] = make_pair(root->data, vLevel);
106
107         solve(root->left, levelTraverse, hLevel - 1, vLevel + 1);
108         solve(root->right, levelTraverse, hLevel + 1, vLevel + 1);
109
110     }
111     vector <int> bottomView(Node *root) {
112         // Your Code Here
113         vector<int> ans;
114         map<int, pair<int, int>> levelTraverse;
115         solve(root, levelTraverse, 0, 0);
116         for(auto node: levelTraverse){
117             ans.push_back(node.second.first);
118         }
119
120
```

archit23200 8 hours ago

0

JAVA Solution Based on **Striver**'s approach:-

```
class ds{
    Node node;
    int verticalIndex;
    ds(Node node,int verticalIndex){
        this.node=node;
        this.verticalIndex=verticalIndex;
    }
}
class Solution
{
    //Function to return a list containing t
    public ArrayList <Integer> bottomView(No
    {
        // Code here
        Queue<ds> q=new LinkedList<>();
        ArrayList<Integer> ans=new ArrayList
        Map<Integer,Integer> m=new TreeMap<>
        q.offer(new ds(root,0));
        while(q.isEmpty()==false){
            ds d=q.poll();
            if(m.containsKey(d.verticalIndex
                m.put(d.verticalIndex,d.node
            }
            m.put(d.verticalIndex,d.node.dat
            if(d.node.left!=null) q.offer(ne
            if(d.node.right!=null) q.offer(n
        }
        for(Integer i:m.values()){
            ans.add(i);
        }
        return ans;
    }
}
```

Output Window

Problem Solved Successfully

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: **60** / 60

Total Time Taken: **0.23**/1.98

Correct Submission Count: **5**

Attempts No.: **11**

Reply Open Externally

rmn5124 2 days ago

0

```
class Solution {
  public:
    vector <int> bottomView(Node *root) {
        // Your Code Here
        queue<pair<Node*,int>>q;//node,level
        map<int,int>m; //level,verticle
        q.push({root,0});
        while(!q.empty()){
            auto f=q.front();
```

Average Time: 45m
Your Time: 17m 3s

Compile & Run Submit