Level Nodes

Problem | Submissions | Leaderboard | Discussions

Submitted 3 hours ago • Score: 20.00 Status: Accepted

✔ Test Case #0	✔ Test Case #1	✔ Test Case #2
✔ Test Case #3	✔ Test Case #4	✔ Test Case #5
✔ Test Case #6	✔ Test Case #7	✔ Test Case #8
✔ Test Case #9	✔ Test Case #10	✔ Test Case #11
✔ Test Case #12	✔ Test Case #13	✔ Test Case #14
✔ Test Case #15	✔ Test Case #16	✔ Test Case #17
✔ Test Case #18	✔ Test Case #19	✔ Test Case #20
✔ Test Case #21	✔ Test Case #22	✔ Test Case #23
✔ Test Case #24	✔ Test Case #25	✔ Test Case #26
✔ Test Case #27	✔ Test Case #28	✔ Test Case #29
✔ Test Case #30	✔ Test Case #31	✔ Test Case #32
✔ Test Case #33	✔ Test Case #34	✔ Test Case #35
✔ Test Case #36	✔ Test Case #37	✔ Test Case #38
✔ Test Case #39	✔ Test Case #40	✔ Test Case #41
✔ Test Case #42	✔ Test Case #43	✔ Test Case #44
✔ Test Case #45	✔ Test Case #46	✔ Test Case #47
✔ Test Case #48	✔ Test Case #49	✔ Test Case #50
✔ Test Case #51	✔ Test Case #52	✔ Test Case #53
✔ Test Case #54	✔ Test Case #55	✔ Test Case #56
✔ Test Case #57	✔ Test Case #58	✔ Test Case #59
✔ Test Case #60		

Submitted Code

Language: C++20 Open in editor

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Node
```

```
{
public:
    int val;
    Node *left;
    Node *right;
    Node(int val)
    {
        this->val = val;
        this->left = NULL;
        this->right = NULL;
    }
};

stack<int> st;
void noOfLevels(Node *root)
{
    if (root == NULL)
    {
        return;
    }

    queue<pair<Node *, int>> q;
    if (root)
        q.push({root, 0});

    while (!q.empty())
    {
        // queue theke ber korlam
        pair<Node *, int> pr = q.front();
        Node *f = pr.first;
        int level = pr.second;
        q.pop();

        // jabotiyo kaj
        st.push(level);


        // queue te push korlam
        if (f->left)
            q.push({f->left, level + 1});
        if (f->right)
            q.push({f->right, level + 1});
    }
}

Node *inputTree()
{
    int val;
    cin >> val;
    Node *root;
    if (val == -1)
        root = NULL;
    else
        root = new Node(val);

    queue<Node *> q;
    if (root)
        q.push(root);

    while (!q.empty())
    {
        Node *f = q.front();
        q.pop();

        int l, r;
        cin >> l >> r;
```

```
        Node *myLeft, *myRight;

        if (l == -1)
            myLeft = NULL;
        else
            myLeft = new Node(l);

        if (r == -1)
            myRight = NULL;
        else
            myRight = new Node(r);

        f->left = myLeft;
        f->right = myRight;

        if (f->left)
            q.push(f->left);
        if (f->right)
            q.push(f->right);
    }
    return root;
}

void levelNodes(Node *root, int lvl,int mxLevel) // 10 3
{
    if (root == NULL)
    {
        return;
    }

    queue<pair<Node *, int>> q;
    if (root)
        q.push({root, 0});

    while (!q.empty())
    {
        // queue theke ber korlam
        pair<Node *, int> pr = q.front();
        Node *f = pr.first;
        int level = pr.second;
        q.pop();

        // jabotiyo kaj
        if(lvl > mxLevel)
        {
            cout<<"Invalid"<<endl;
            return;
        }
        else if (level == lvl )
        {
            cout << f->val << " ";
        }

        // queue te push korlam
        if (f->left)
            q.push({f->left, level + 1});
        if (f->right)
            q.push({f->right, level + 1});
    }
}

/*
10 20 30 40 50 -1 60
-1 -1 -1 -1 -1 -1
*/
int main()
```

```
{
    Node *root = inputTree();
    int l;
    cin >> l;

    noOfLevels(root);
    // cout<<st.top()<<endl;
    int highestLevel = st.top();
    levelNodes(root, l,highestLevel);


    return 0;
}
```