Perfect Binary Tree

Problem | Submissions | Leaderboard | Discussions

Submitted 34 minutes 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

Submitted Code

Language: C++20 Open in editor

```cpp
#include <bits/stdc++.h>
using namespace std;

class Node
{
public:
    int val;
    Node *left;
    Node *right;
    Node(int val)
    {
        this->val = val;
        this->left = NULL;
        this->right = NULL;
    }
};
Node *inputTree();
void levelOrder(Node *root);
int countNodes(Node *root);
int maxHeight(Node *root);

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 levelOrder(Node *root)
{
    if (root == NULL)
    {
        return;
    }

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

    while (!q.empty())
    {
        // queue theke ber korlam
        Node *f = q.front();
        q.pop();

        // jabotiyo kaj

        cout << f->val << " ";


        // queue te push korlam
```

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

int countNodes(Node *root)
{
    if(root == NULL) return 0;

    int l = countNodes(root->left);
    int r = countNodes(root->right);
    return l+r+1;
}

int maxHeight(Node *root)
{
    if(root == NULL) return 0;
    int l = maxHeight(root->left);
    int r = maxHeight(root->right);
    return max(l,r)+1;
}


int main()
{
    Node *root = inputTree();
    // levelOrder(root);

    int noOfNodes = countNodes(root);
    // cout<<noOfNodes<<endl;


    int h = maxHeight(root);
    // cout<<h<<endl;
    double nodes = pow(2,h)-1;
    // cout<<nodes<<endl;

    if(noOfNodes == nodes) cout<<"YES"<<endl;
    else cout<<"NO"<<endl;
    return 0;
}
```