

# 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

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Node
5 {
6 public:
7     int val;
8     Node *left;
9     Node *right;
10    Node(int val)
11    {
12        this->val = val;
13        this->left = NULL;
14        this->right = NULL;
15    }
16 };
17 Node *inputTree();
18 void levelOrder(Node *root);
19 int countNodes(Node *root);
20 int maxHeight(Node *root);
21
22 Node *inputTree()
```

```
23 {
24     int val;
25     cin >> val;
26     Node *root;
27     if (val == -1)
28         root = NULL;
29     else
30         root = new Node(val);
31
32     queue<Node *> q;
33     if (root)
34         q.push(root);
35
36     while (!q.empty())
37     {
38         Node *f = q.front();
39         q.pop();
40
41         int l, r;
42         cin >> l >> r;
43         Node *myLeft, *myRight;
44
45         if (l == -1)
46             myLeft = NULL;
47         else
48             myLeft = new Node(l);
49
50         if (r == -1)
51             myRight = NULL;
52         else
53             myRight = new Node(r);
54
55         f->left = myLeft;
56         f->right = myRight;
57
58         if (f->left)
59             q.push(f->left);
60         if (f->right)
61             q.push(f->right);
62     }
63     return root;
64 }
65
66 void levelOrder(Node *root)
67 {
68     if (root == NULL)
69     {
70         return;
71     }
72
73     queue<Node*> q;
74     if (root)
75         q.push(root);
76
77     while (!q.empty())
78     {
79         // queue theke ber korlam
80         Node *f = q.front();
81         q.pop();
82
83         // jabotiyo kaj
84
85         cout << f->val << " ";
86
87
88         // queue te push korlam
```

```
89         if (f->left)
90             q.push(f->left);
91         if (f->right)
92             q.push(f->right);
93     }
94 }
95
96 int countNodes(Node *root)
97 {
98     if(root == NULL) return 0;
99
100    int l = countNodes(root->left);
101    int r = countNodes(root->right);
102    return l+r+1;
103 }
104
105 int maxHeight(Node *root)
106 {
107     if(root == NULL) return 0;
108     int l = maxHeight(root->left);
109     int r = maxHeight(root->right);
110     return max(l,r)+1;
111 }
112
113
114 int main()
115 {
116     Node *root = inputTree();
117     // levelOrder(root);
118
119     int noOfNodes = countNodes(root);
120     // cout<<noOfNodes<<endl;
121
122
123     int h = maxHeight(root);
124     // cout<<h<<endl;
125     double nodes = pow(2,h)-1;
126     // cout<<nodes<<endl;
127
128     if(noOfNodes == nodes) cout<<"YES"<<endl;
129     else cout<<"NO"<<endl;
130     return 0;
131 }
```