Description | Editorial | Solutions (4.5K) | Submissions

222. Count Complete Tree Nodes

Easy 8K 439

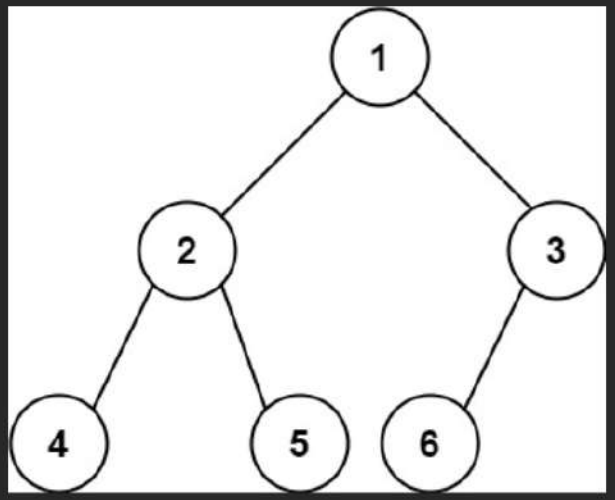
Companies

Given the `root` of a **complete** binary tree, return the number of the nodes in the tree.

According to **Wikipedia**, every level, except possibly the last, is completely filled in a complete binary tree, and all nodes in the last level are as far left as possible. It can have between `1` and 2^h nodes inclusive at the last level `h`.

Design an algorithm that runs in less than `O(n)` time complexity.

Example 1:



Input: root = [1,2,3,4,5,6]
Output: 6

Java | Auto

```
13  *     }
14  * }
15  */
16 class Solution {
17     //counter variable
18     int count = 0;
19     public int countNodes(TreeNode root) {
20
21         if(root != null){
22         countNodes(root.left);
23         count++;
24         countNodes(root.right);
25         }
26
27             return count;
28     }
29 }
```

Ln 29, Col 2

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

root =
[1,2,3,4,5,6]

Output

6

Console | Run | Submit

Description | Editorial | Solutions (3.8K) | Submissions

225. Implement Stack using Queues

Easy 5.4K 1.1K

Companies

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (`push`, `top`, `pop`, and `empty`).

Implement the `MyStack` class:

- `void push(int x)` Pushes element x to the top of the stack.
- `int pop()` Removes the element on the top of the stack and returns it.
- `int top()` Returns the element on the top of the stack.
- `boolean empty()` Returns `true` if the stack is empty, `false` otherwise.

Notes:

- You must use **only** standard operations of a queue, which means that only `push to back`, `peek/pop from front`, `size` and `is empty` operations are valid.
- Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations.

Example 1:

Input
```
["MyStack", "push", "push", "top", "pop", "empty"]
[[], [1], [2], [], [], []]
```
Output
```
[null, null, null, 2, 2, false]
```

Java | Auto

```java
public class MyStack {
    private Queue<Integer> q;

    public MyStack() {
        q = new LinkedList<>();
    }

    public void push(int x) {
        q.add(x);
        for (int i = 1; i < q.size(); i++) {
            q.add(q.remove());
        }
    }

    public int pop() {
        return q.remove();
    }
```

Ln 26, Col 2

Testcase | Result

Accepted Runtime: 0 ms

Case 1

Input

```
["MyStack","push","push","top","pop","empty"]
```

```
[[],[1],[2],[],[],[]]
```

Output

```
[null,null,null,2,2,false]
```

Console | Run | Submit