Problem | Submissions | Comments

Count Number of Nodes in a Binary Tree

Easy | Accuracy: 92.67% | Submissions: 2K+ | Points: 2

You are given the root of a **complete** binary tree. Your task is to find the **count** of nodes.

A complete binary tree is a binary tree whose, all levels except the last one are completely filled, the last level may or may not be completely filled and Nodes in the last level are as left as possible.

Design an algorithm that runs better than O(n).

Example:

Input:
root = [1,2,3,4,5,6]
Output:
6
Explanation:
There are a total of 6 nodes in the given tree.

C++ (g++ 5.4) | Average Time: 20m | Start Timer

```
1  // } Driver Code Ends
26 // User function Template for C++
27
28 class Solution {
29   public:
30     int countNodes(Node* root) {
31         if(root == NULL){
32             return 0;
33         }
34         return countNodes(root->left) + countNodes(root->right) + 1;
35     }
36 };
37 // } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Unique Binary Tree Requirements

Easy Accuracy: 53.52% Submissions: 2K+ Points: 2

Geek wants to know the traversals required to construct a **unique binary tree**. Given a pair of traversal, return **true** if it is possible to construct unique binary tree from the given traversals otherwise return **false**.

Each traversal is represented with an integer: preorder - 1, inorder - 2, postorder - 3.

Example 1:

Input:
a = 1, b=2
Output: 1
Explanation: We can construct binary tree using inorder traversal and preorder traversal.

Example 2:

C++ (g++ 5.4) Average Time: 15m Start Timer

```
1  // } Driver Code Ends
9  //User function Template for C++
10
11 class Solution
12 {
13 public:
14     bool isPossible(int a,int b)
15     {
16         if( (a==2 || b==2) && (a!=b) )
17             return true;
18         else
19             return false;
20     }
21 };
22 // } Driver Code Ends
```

Custom Input Compile & Run Submit

Flatten binary tree to linked list

Medium Accuracy: 75.82% Submissions: 25K+ Points: 4

Given the root of a binary tree, flatten the tree into a "linked list":

- The "linked list" should use the same Node class where the right child pointer points to the next node in the list and the left child pointer is always null.
- The "linked list" should be in the same order as a pre-order traversal of the binary tree.

Example 1:

Input :

Output :

C++ (g++ 5.4) Start Timer

```
1  // } Driver Code Ends
97 //User function Template for C++
98
99 class Solution
100{
101    public:
102    void flatten(Node *root)
103    {
104        //code here
105        if(root==NULL)
106       return;
107
108       flatten(root->right);
109       flatten(root->left);
110
111       if(root->left!=NULL){
112
113           Node *a = root->right;
114
115           root->right=root->left;
116           root->left=NULL;
117           Node *b =root;
118
119           while(b->right!=NULL){
120               b=b->right;
121           }
122           b->right=a;
123
124       }
125    }
126};
127// } Driver Code Ends
```

Custom Input | Compile & Run | Submit