543. Diameter of Binary Tree

Easy 12.4K 777

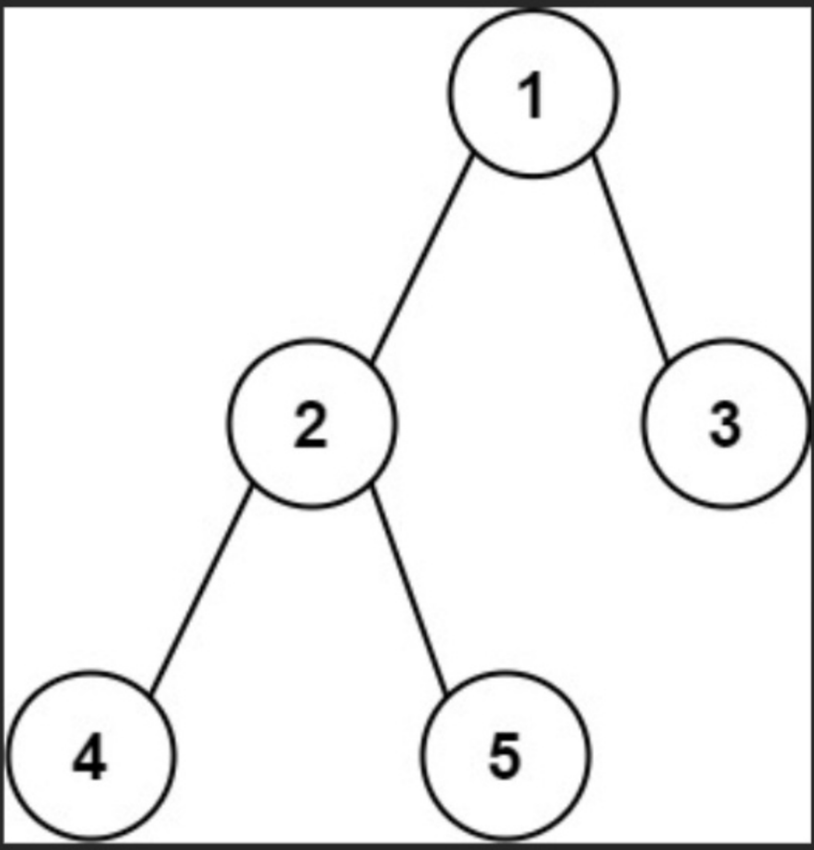
Companies

Given the `root` of a binary tree, return *the length of the **diameter** of the tree*.

The **diameter** of a binary tree is the **length** of the longest path between any two nodes in a tree. This path may or may not pass through the `root`.

The **length** of a path between two nodes is represented by the number of edges between them.

Example 1:




```
Input: root = [1,2,3,4,5]
Output: 3
Explanation: 3 is the length of the path [4,2,1,3] or [5,2,1,3].
```

Example 2:

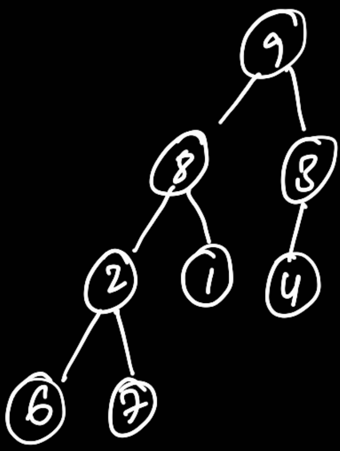
```
Input: root = [1,2]
Output: 1
```

Constraints:

- The number of nodes in the tree is in the range `[1, 10^4]`.
- `-100 <= Node.val <= 100`

Accepted 1.2M | Submissions 2M | Acceptance Rate 57.9%

Approach 1: Recursive implementation



diameter : The distance b/w two nodes n_1 and n_2 i.e. no. of edges.

here n_1 and n_2 can be the same node and they also can be root

at each node:

l = max no. of edges in LST
r = max no. of edges in RST
if ($l+r$ > currentmax)
  Currentmax = $l+r$

return max (l, r) + 1   bcoz we might get even more distance that passes through ancestors of current node.

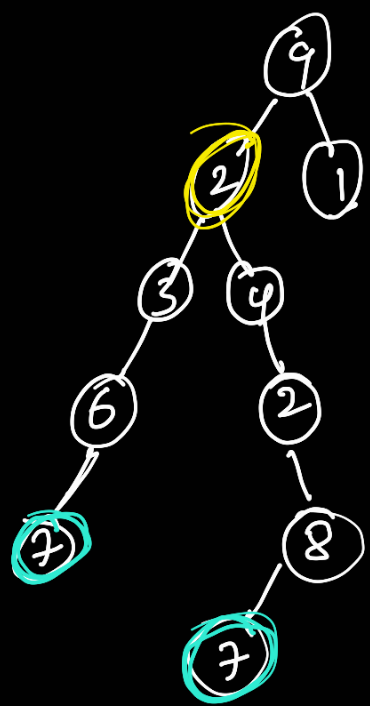
```
find ( node , int& d)
{
    if (node is null) return 0

    int l = find (node->left, d)
    int r = find(node->right, d)
```

```
    if( l+r > d)
        d = l+r

    return max(l, r) +1

}
```

T(n) : O(n)

Approach 2:

The diameter is basically the sum of max height of LST + max height of RST among all nodes



diameter = 3+4
= 7

```
for( every node in tree)
{
    int l = maxheight(node->left)
    int r = maxheight(node->right)
```

$$\text{if} (\; l + r > \text{currentmax})$$

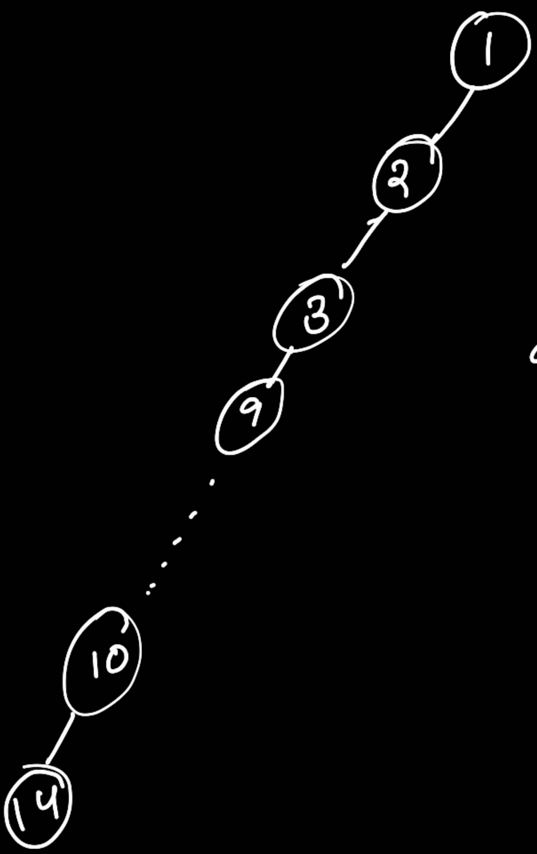
$$\text{currentmax} = l + r$$

}

$T(n) : O(n^2)$

Q: How $T(n)$ is $O(n^2)$?

Let there is a skewed Tree.



$T(n)$ of maxheight function of n nodes

at ③ at ⑩

$$n-1 + n-2 \; \cdots \; + 1$$

at ②

$$= \frac{(n-1)\,n}{2} = n^2$$