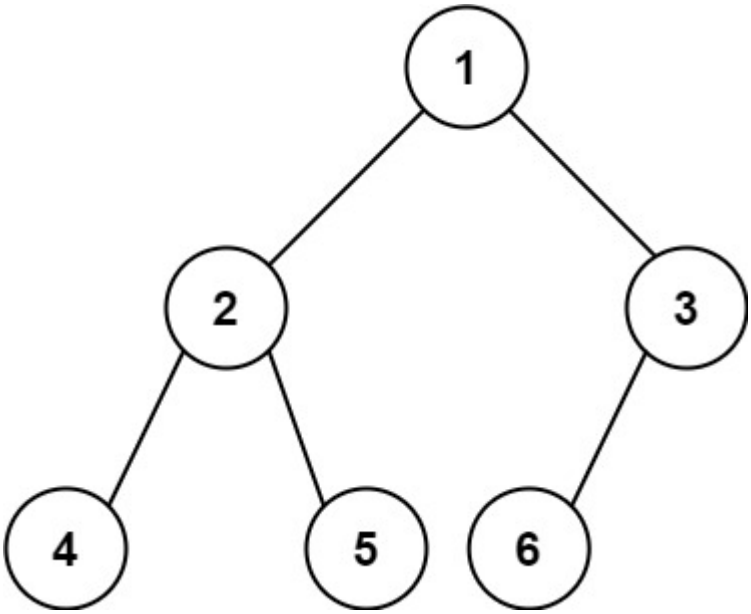
222. Count Complete Tree Nodes

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 `2h` 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

Example 2:

Input: root = []
Output: 0

Example 3:

Input: root = [1]
Output: 1

```
def countNodes(self, root):
        if not root:
            return 0
```

```
        def depthLeft(node):
            d = 0
            while node:
                d += 1
                node = node.left
            return d

        def depthRight(node):
            d = 0
            while node:
                d += 1
                node = node.right
            return d

        ld = depthLeft(root.left)
        rd = depthRight(root.right)

        if ld == rd:
            return 2**(ld + 1) - 1
        else:
            return 1 + self.countNodes(root.left) +
self.countNodes(root.right)
```

TC = O(logN^2)