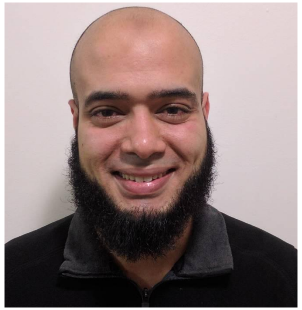
Data Structures

Binary Tree Homework 1

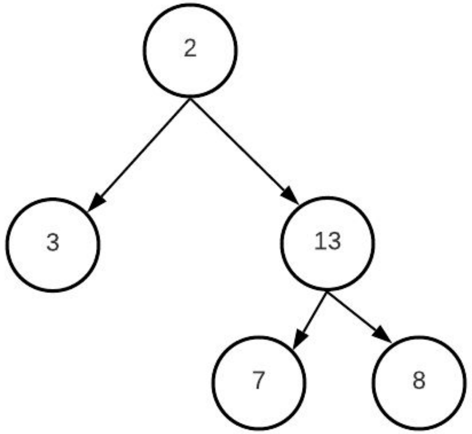
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

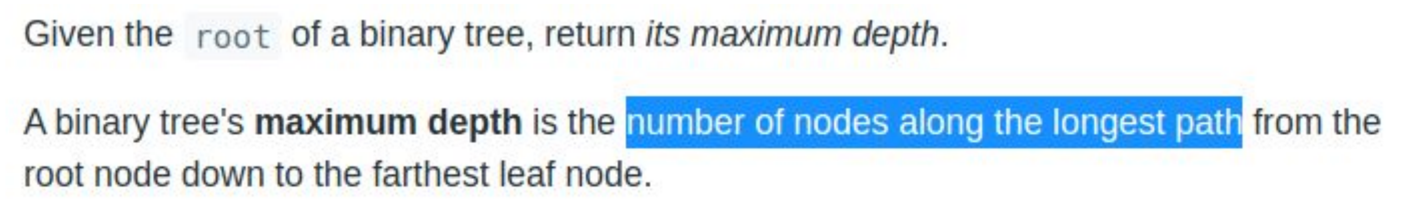


Problem #1: Tree Max

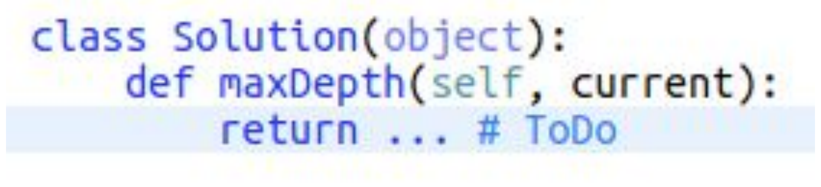
- `def tree_max(self):`
- Inside the BinaryTree class, add this function. It returns the **maximum value in the whole tree**
    - The function should be recursive
    - i.e. similar to the pre-order traversal
- In 'this' tree, the max value is 13
- Create several trees using the add functions we learned and test your code

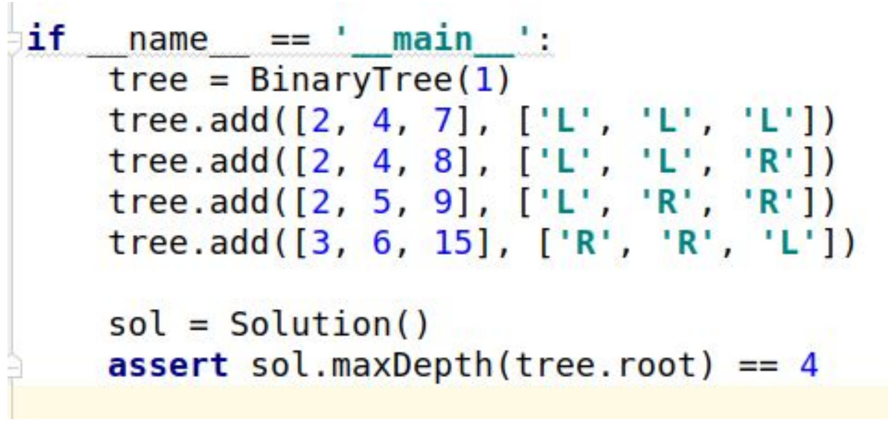


Problem #2: LeetCode 104 - Maximum Depth of Binary Tree


Given the `root` of a binary tree, return *its maximum depth*.

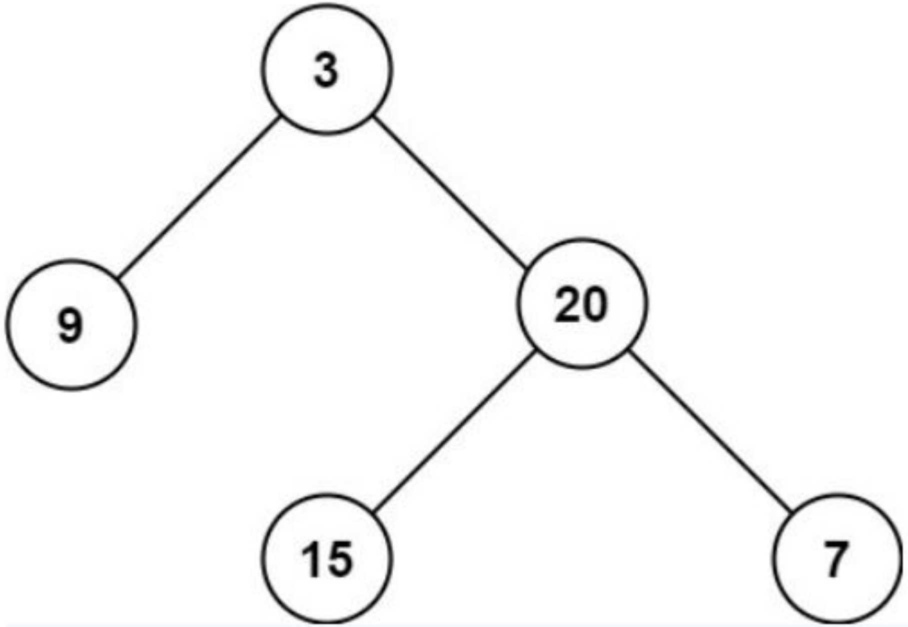
A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.


```python
class Solution(object):
    def maxDepth(self, current):
        return ... # ToDo
```


```python
if __name__ == '__main__':
    tree = BinaryTree(1)
    tree.add([2, 4, 7], ['L', 'L', 'L'])
    tree.add([2, 4, 8], ['L', 'L', 'R'])
    tree.add([2, 5, 9], ['L', 'R', 'R'])
    tree.add([3, 6, 15], ['R', 'R', 'L'])

    sol = Solution()
    assert sol.maxDepth(tree.root) == 4
```

- The tree nodes similar to our code:
    - Attributes: val, left and right



```
Input: root = [3,9,20,null,null,15,7]
Output: 3
```

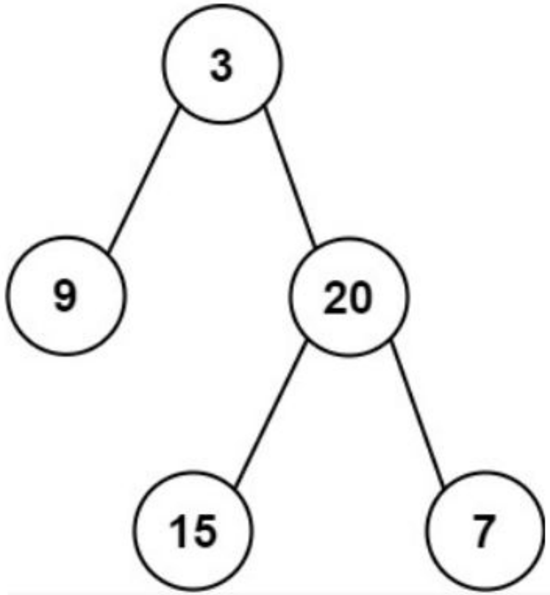
Example 2:

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

Problem #3: LeetCode 404 - Sum of Left Leaves

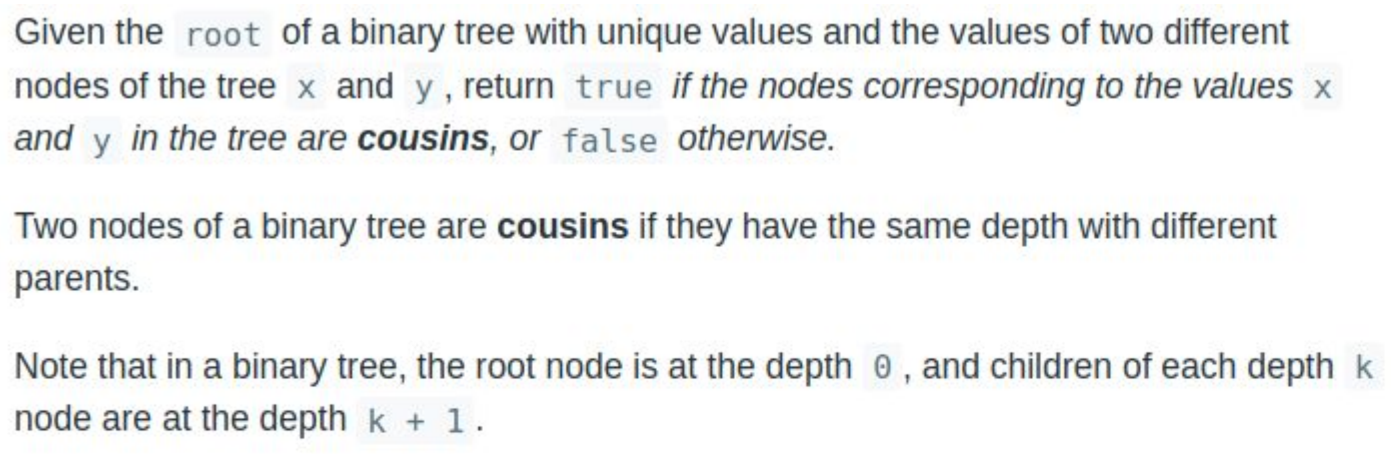
Given the `root` of a binary tree, return the sum of all left leaves.

Example 1:



```
Input: root = [3,9,20,null,null,15,7]
Output: 24
Explanation: There are two left leaves in the binary tree, with
values 9 and 15 respectively.
```

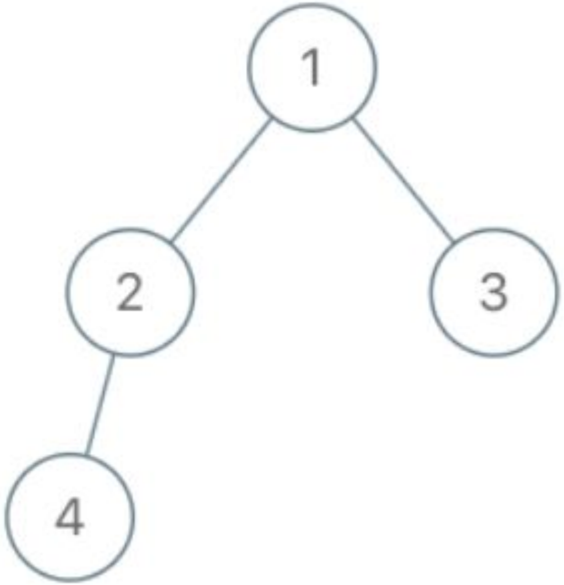
Problem #4: LeetCode 993 - Cousins in Binary Tree



Given the `root` of a binary tree with unique values and the values of two different nodes of the tree `x` and `y`, return `true` *if the nodes corresponding to the values* `x` *and* `y` *in the tree are* ***cousins****, or* `false` *otherwise.*

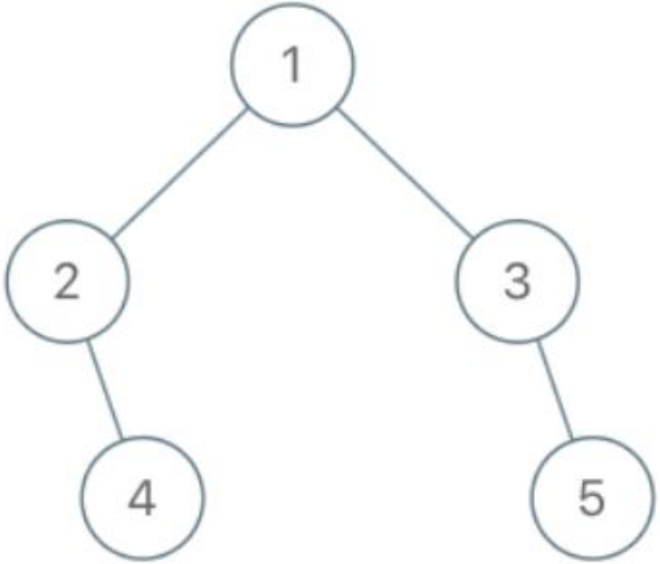
Two nodes of a binary tree are **cousins** if they have the same depth with different parents.

Note that in a binary tree, the root node is at the depth `0`, and children of each depth `k` node are at the depth `k + 1`.

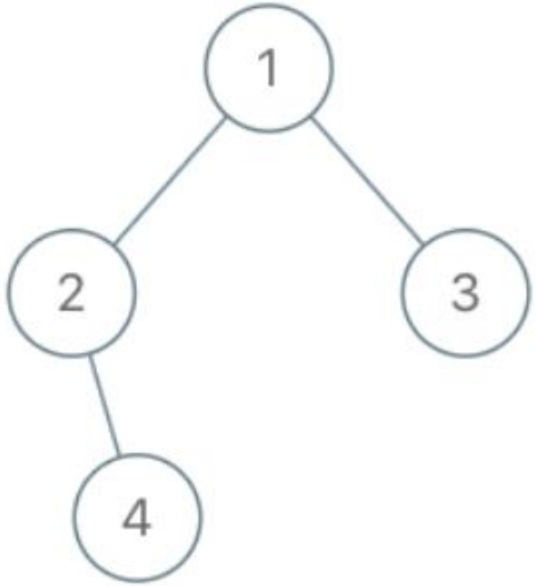


```
Input: root = [1,2,3,4], x = 4, y = 3
Output: false
```

Example 2:



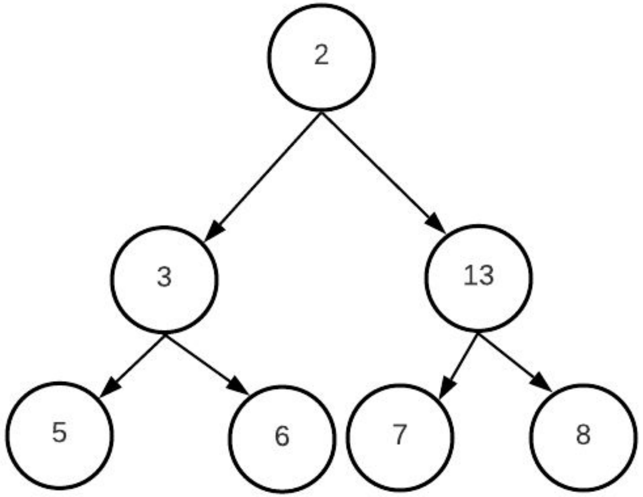
```
Input: root = [1,2,3,null,4,null,5], x = 5, y = 4
Output: true
```



```
Input: root = [1,2,3,null,4], x = 2, y = 3
Output: false
```

Problem #5: Is Perfect Tree

- **def** is_perfect(self)
- It returns **True** if the tree is perfect, False otherwise
- Develop it in 2 ways
    - Recursive way
    - A formula-based way



“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”