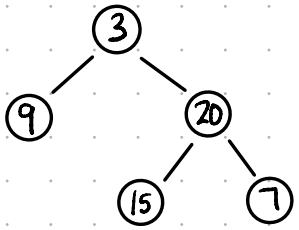
Maximum Depth of Binary Tree

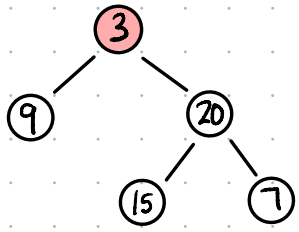
basically looking for height of tree

Will use Pre-Order DFS



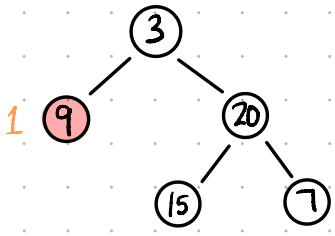

```
def maxDepth(root):
    # Base Case
    if not root:  # for None
        return 0
    # Recursive Case
    left = maxDepth(root.left)
    right = maxDepth(root.right)

    return 1 + max(left, right)
```

Process



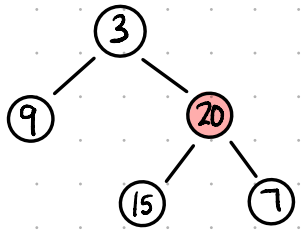
For Node 1:

- Call maxDepth(1), which calls maxDepth(9) and maxDepth(20)



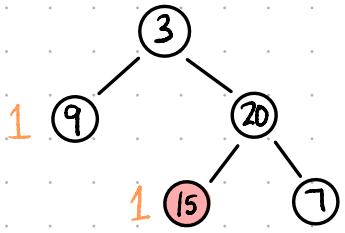
For Node 9:

- Call maxDepth(9), which checks its left and right children. Since (9) has no children, it calls maxDepth(None) for both left and right.
- maxDepth returns 0 because it's a leaf node
- Now, for (9), left = 0 and right = 0. So, the depth of (9) is calculated as 1 + max(0,0) = 1



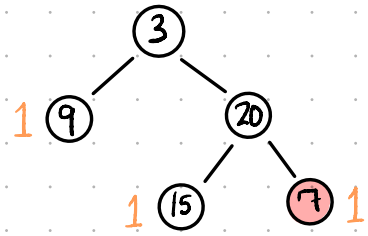
For Node 20:

- Call maxDepth(20), which calls maxDepth(15) & maxDepth(7)



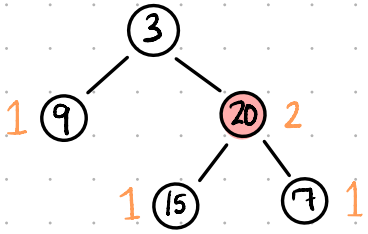
For Node 15:

- Call maxDepth(15), no children on left and right
- maxDepth returns 0
- For (15), 1 + max(0, 0) = 1



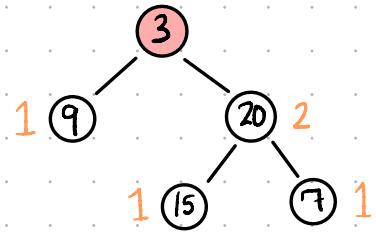
For Node 7:

- Call maxDepth(7), no children on left and right
- maxDepth returns 0
- For (7), 1 + max(0, 0) = 1



Back to Node 20:

- For (20), we have left = 1 and right = 1
- Depth of (20) is 1 + max(1 + 1) = 2



Back to Node 3:

- For (3), we have left = 1 and right = 2
- Depth of (3) is 1 + max(1, 2) = 3

Function returns 3