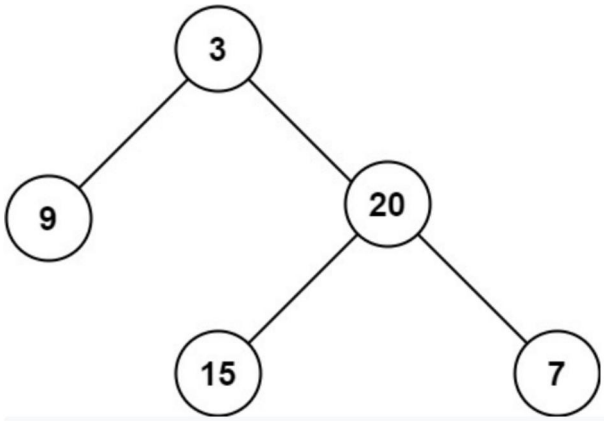
104. Maximum Depth of Binary Tree

14 February 2022 08:56 AM

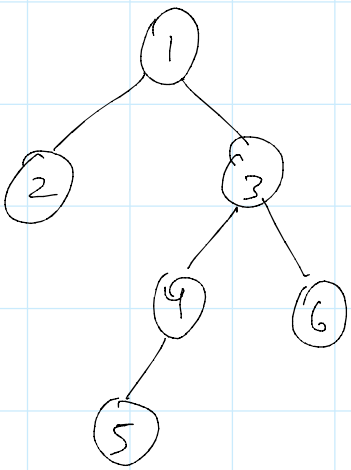
Given the `root` of a binary tree, return *its maximum depth*.

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.

Example 1:




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

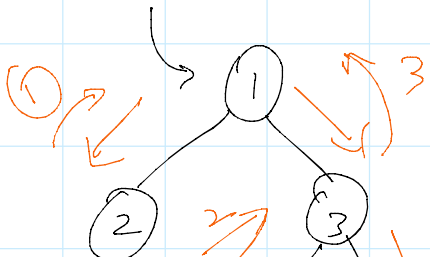


maxi depth / height = 4

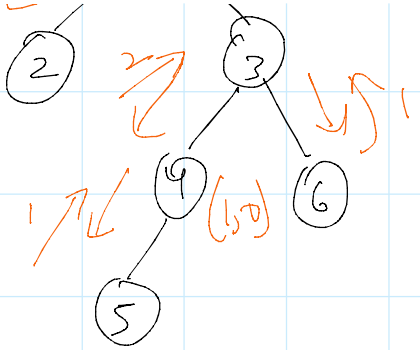
we can use Recursive / level Order.

↓ O(height)

↳ Queue data structure S.C O(n)



1 + max (l, r)



$1 + \max(l, r)$

first move to left and then right.

for ② ⇒ $1 + \max(0,0) = 1$

for ③ ⇒ $1 + \max(l, r)$

④ ⇒ $1 + \max(1, 0) = 1 + 1 = 2$

⑤ ⇒ $1 + \max(0,0) = 1$

⑥ ⇒ $1 + \max(0,0) = 0$

for ③ ⇒ $1 + \max(2,1)$

$= 3$

for ① ⇒ $1 + \max(3,1)$

$= 1 + 3 = 4$

T.C ⇒ $O(n)$ S.C ⇒ $O(n)$

```java
// Recursive Solution
class Solution {
    public int maxDepth(TreeNode root) {
        if(root == null) return 0;

        int lh = maxDepth(root.left);
        int rh = maxDepth(root.right);

        return 1 + Math.max(lh,rh);
    }
}
```

```java
// Level Order Traversal Solution
class Solution{
public int maxDepth(TreeNode root) {
        if(root == null) return 0 ;
        Queue<TreeNode> q = new LinkedList<>() ;
        q.add(root) ;
        int ans = 0 ;
        while(!q.isEmpty()) {
            ans ++ ;
            int n = q.size() ;
            for(int i = 0;  i< n ; i++) {
                TreeNode x = q.remove() ;
                if(x.left != null) q.add(x.left) ;
                if(x.right != null) q.add(x.right) ;
            }
        }
        return ans ;
    }

}
```