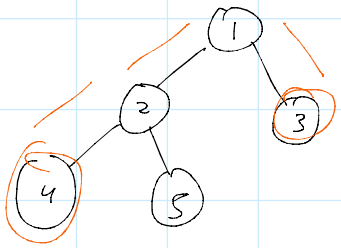
543. Diameter of Binary Tree

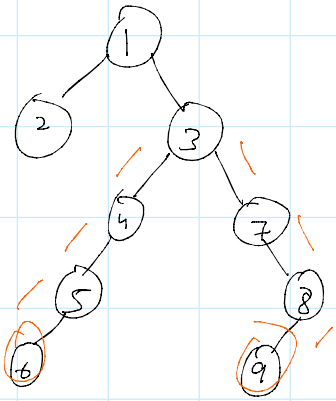
18 February 2022 07:49 PM

In gfg : diameter is no.of nodes on longest path
So diameter = lh+rh+1
In leetcode :- length of longest path
Diameters =lh+rh

* It's the longest path between any 2 nodes.
* Path doesn't need to pass via root.

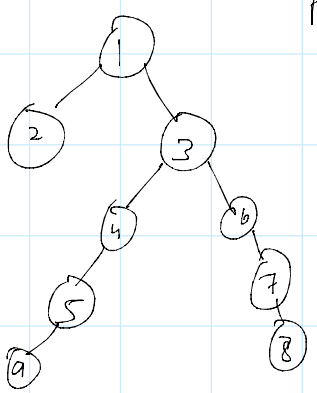


d = 3



d = 6

Brute force



for node (1)
lh + rh = 1 + 4 = 5

for node (2) = 0

for node (3) = 3 + 3 = 6

for node (4) = 2 + 0 = 2

for node (5) = 1 + 0 = 1

for node (6) = 0 + 1 = 2

" " (7) = 0 + 1 = 1

" " (8) = 0 + 0 = 0

Trying to take the longest on every node considering that node as the curve point

for node (5) = 1 + 0 = 1 ,, ,, (8) = 0 + 0 = 0

for node (9) = 0 + 0 = 0

```
        ↓ root
find max (node)
{ if (root == null)
        return

   lh = findleft(node.left)
   rh = findRight(node.right)
 maxi = max (maxi, lh + rh);
findMax (node.left)
} findMax (node.right)
```

T.C ⇒ $O(n^2)$

S.C ⇒ $O(N)$

Better Approach

maxi = 0

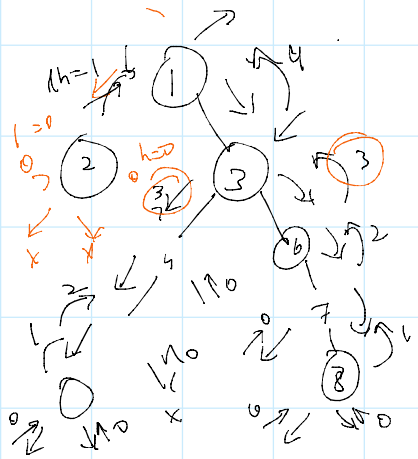
```
int findMax(node, maxi)
{
    if (node == null) return;

    lh = findMax(node.left, maxi)
    rh = findMax (node.right, maxi)
    maxi = max(maxi, lh + rh)
    return 1 + max (lh, rh)
}
```

Dry Run

initialise

maxi = 0



"nil" ie

for node (2)

maxi = max (0, 0)

return 1 + 0 = 1

So for node (9)

lh 1+0 =

So for node (5)

lh = 1 rh = 0 maxi = max (0, 1)

return 1 + 1 = 2

for node (3)

3+3=6

maxi = max (maxi, lh + rh)

return 1 + 3 = 4

for node (1)

maxi = 5 not greater than node (3) ≥ 6

(1+4)

Leet Code

```
class Solution {
    public int diameterOfBinaryTree(TreeNode root) {
        int[] diameter = new int[1];
        height(root, diameter);
        return diameter[0];
    }

    private int height(TreeNode root, int[] diameter){
        if(root == null) return 0;

        int lh = height(root.left, diameter);
        int rh = height(root.right, diameter);
        diameter[0] = Math.max(diameter[0], lh + rh);
        return 1 + Math.max(lh, rh);
    }
}
```

T. C ⇒ O(N)

S. C ⇒ O(N)

```java
class Solution {
    int maxi = 0;
    int diameter(Node root) {
        height(root);
        return maxi;
    }
    private int height(Node root){
        if(root == null) return 0;

        int lh = height(root.left);
        int rh = height(root.right);
        maxi = Math.max(maxi, lh + rh + 1);
        return 1 + Math.max(lh, rh);
    }

}
```

GfG