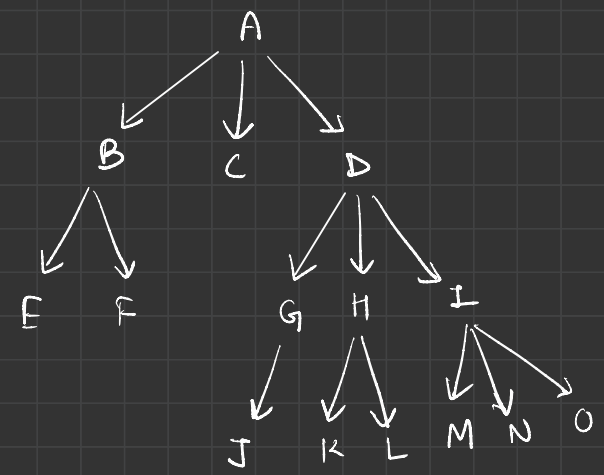
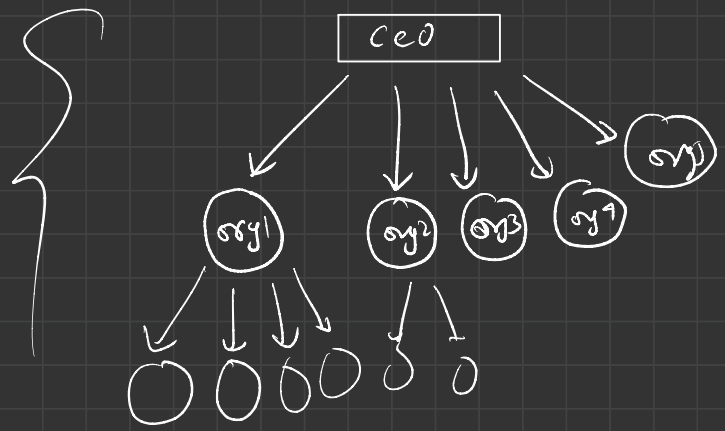
Binary trees

Store your family members!

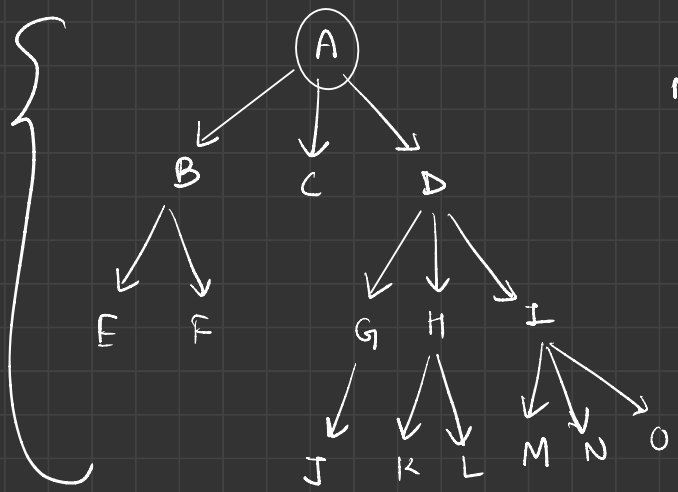
data structure





---

data structure : trees



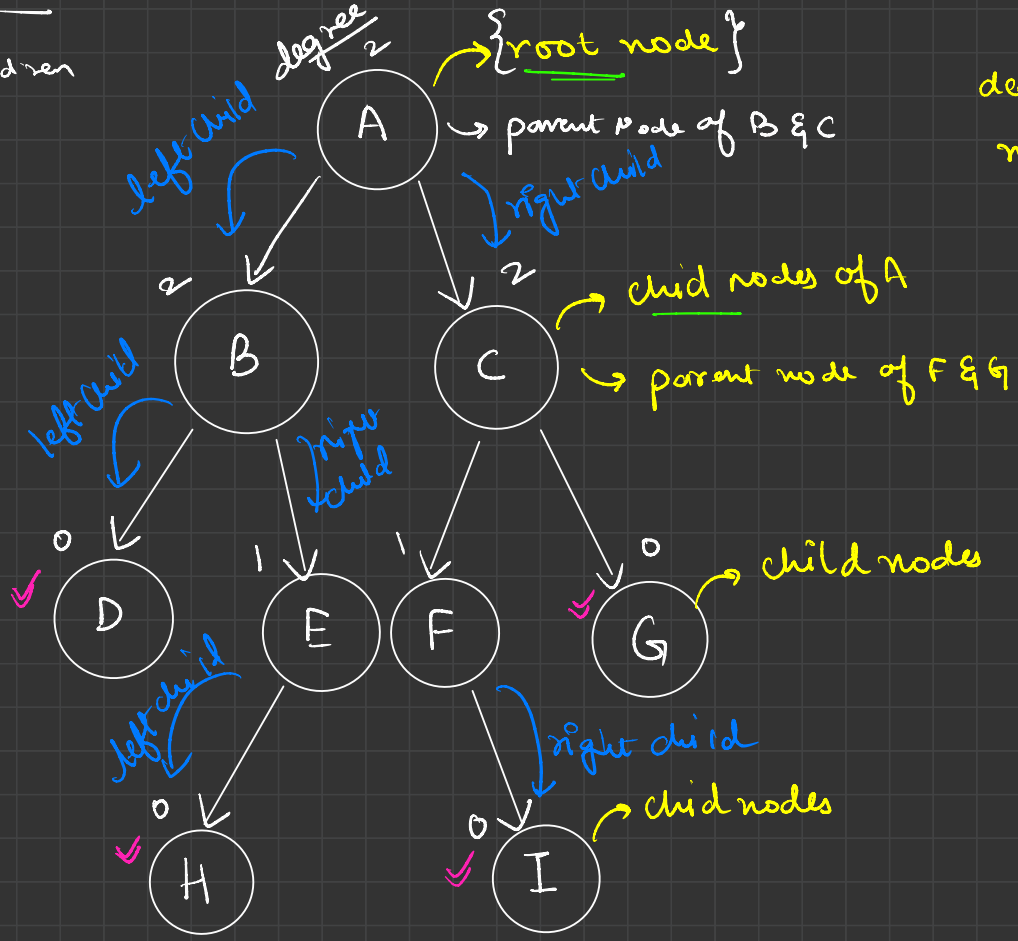

```
int val;
Node[] childrens;
```

Binary trees

atmost 2 children

siblings

- B, C
- D, E
- F, G

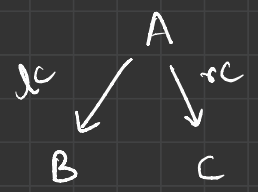


degree of node

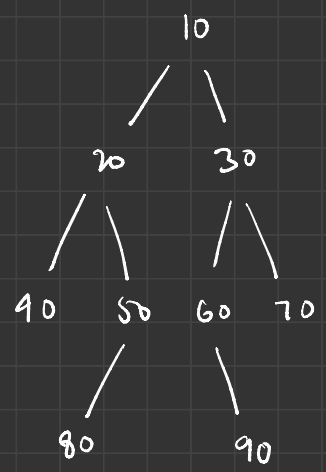
no. of childs that Node has

leaf Nodes

degree = 0

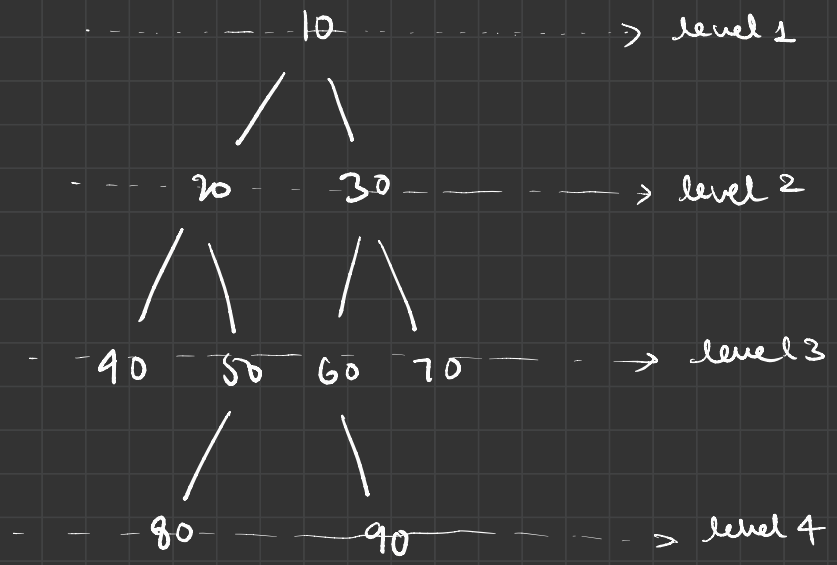



```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
}
```

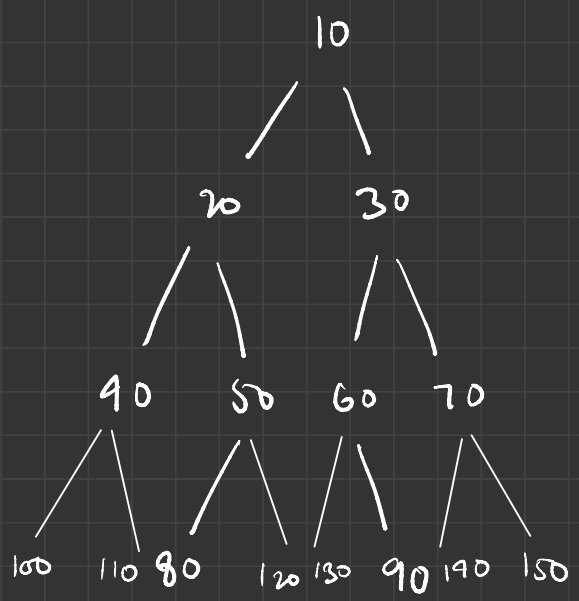


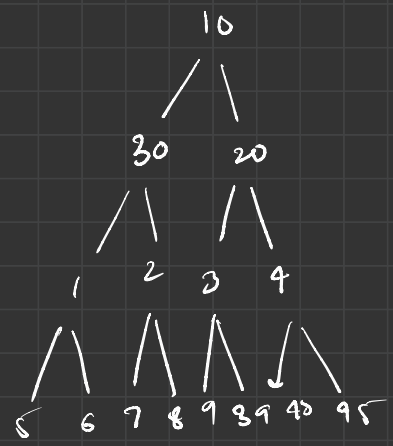
height of this tree: 3

dist b/w root and lowest leaf Node in term of edges


10
level 1
20
30
level 2
40
50
60
70
level 3
80
90
level 4

perfect Binary tree





level 1 : 1 node

level 2 : 3 nodes

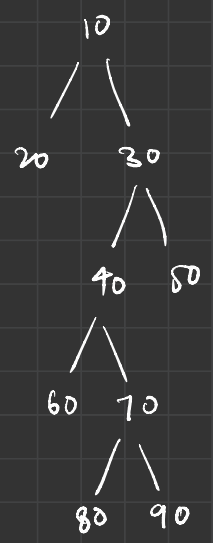
level 3 : 7 nodes

level 4 : 15 nodes

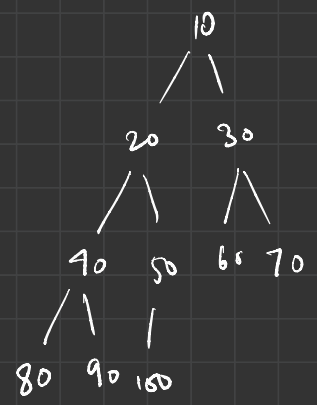
⋮

level h : $2^h - 1$

Full Binary Tree : a full Binary tree is a tree where, each node either have zero or two childrens.

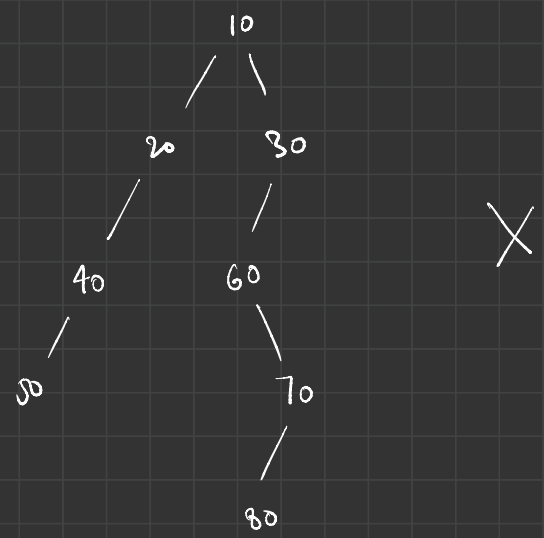


<u>Complete Binary tree</u> : where each level is completely filled except last level, In last level nodes are as left as possible.



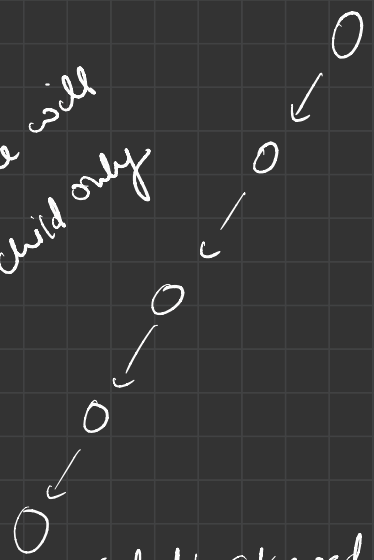
Balanced Binary trees

$$|\text{height of left subtree} - \text{height of right subtree}| \leq 1$$



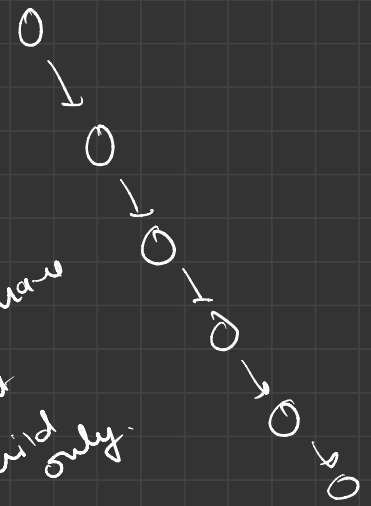
Skew tree

each node will left child only



left skewed tree

each node will have right child only.

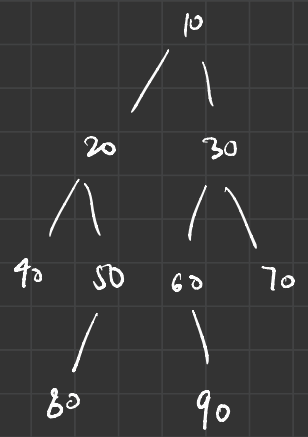


right skewed tree

Questions

traversal over a tree

① pre order traversal

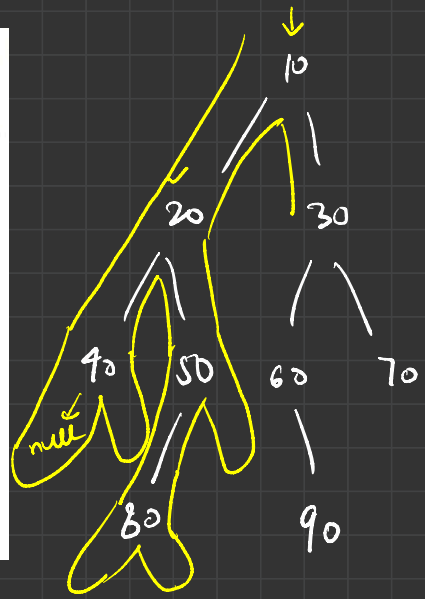


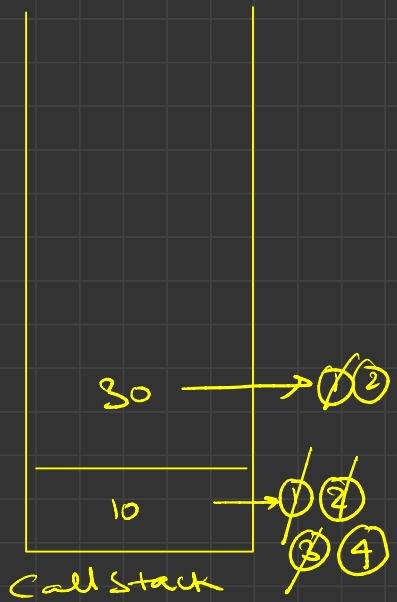
pre order traversal of a tree

10, 20, 40, 50, 80, 30, 60, 90, 70

root left right

```java
public void preTraversal (TreeNode node) {
    if (node == null) {
        return;
    }

    // print this new Node
    System.out.print (node.val);

    // call my left subtree
    preTraversal (node.left);

    // call my right subtree
    preTraversal (node.right);
}
```





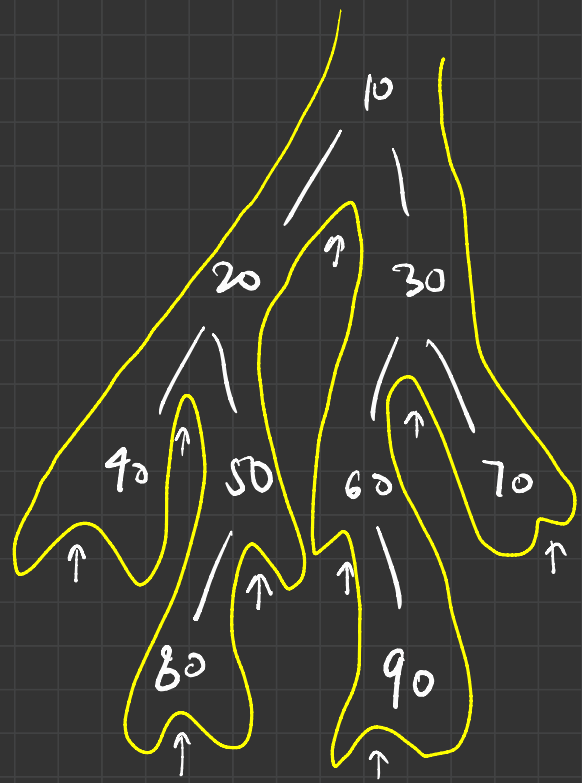
10 , 20 , 40, 50 , 80, 30

TC : O(N)
SC : O(h)

In order traversal

TC: O(N)

SC: O(h)



$\boxed{40, 20, 80, 50}$, 10, $\boxed{60, 90, 30, 70}$

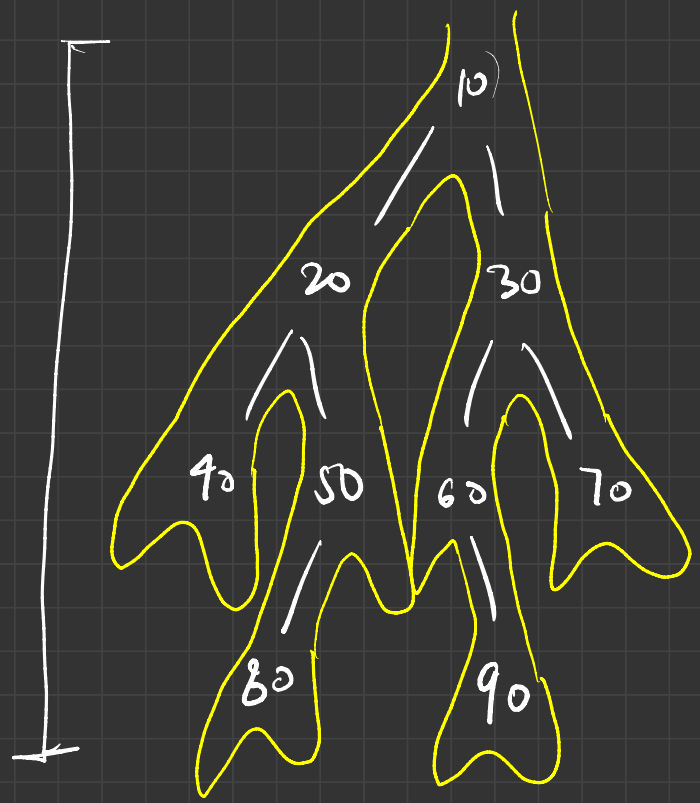
left root right

call (left)

print(node)

call (right)

post order traversal



TC: O(N)

SC: O(h)

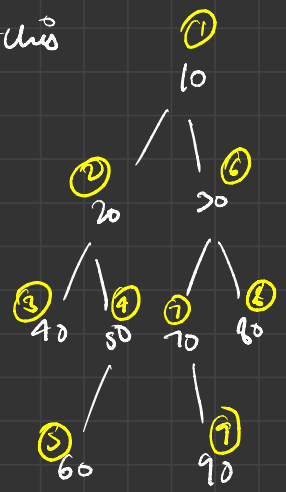
$\boxed{40, 80, 50, 20}$, $\boxed{90, 60, 70, 30}$, 10

left, right, root

call(left)
call(right)
print(node)

Size of a Binary tree

↳ No. of Nodes in this tree



✓ Size = 9

```java
// TC: O(N), SC: O(H)
public int sizeOfTree(Node root) {
    // base case
    // if root == null then size = 0
    if (root == null) {
        return 0;
    }

    // get size of left sub-tree
    int sizeOfLeftSubTree = sizeOfTree(root.left);

    // get size of right sub-tree
    int sizeOfRightSubTree = sizeOfTree(root.right);

    // size of tree = size of left sub-tree + 1 for my node + size of right sub-tree
    int size = sizeOfLeftSubTree + 1 + sizeOfRightSubTree;

    return size;
}
```

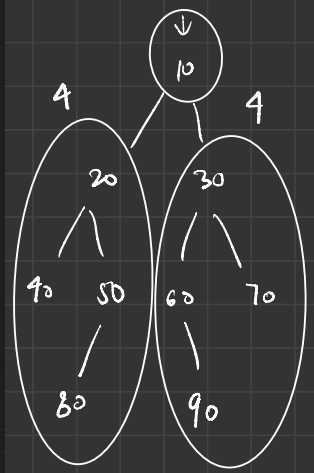
1

2

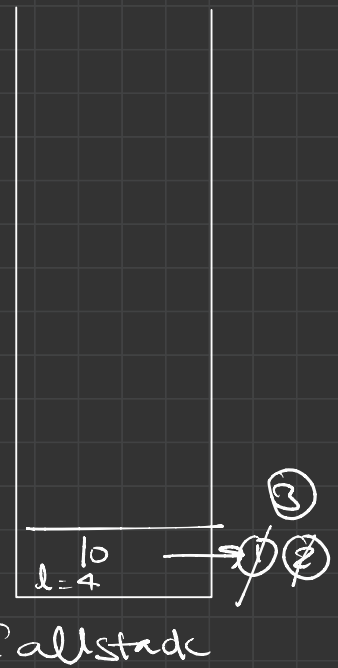
3

4

$4+1+4=\underline{\underline{9}}$

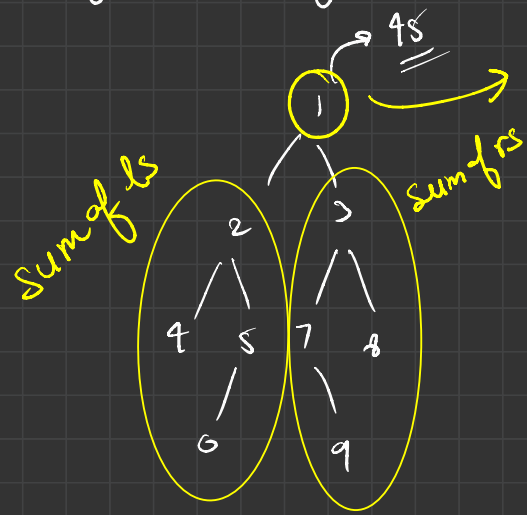


$l+1+r$



Sum of tree

Sum of values of each node in a tree.

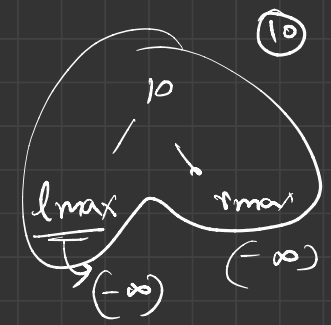


sum ls + root.val + sum rs

sum
45

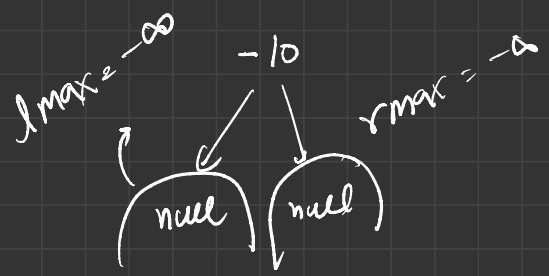
maximum of a tree

↳ max^m node val present in this tree.



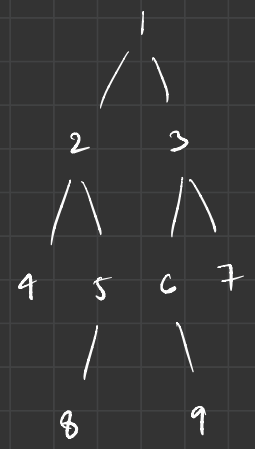
$\max(lmax, rmax, root.data)$

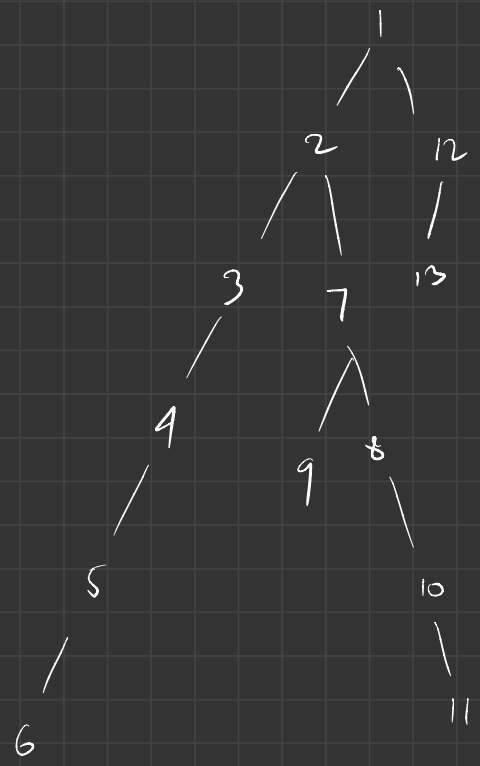
$= -10$



diameter of tree.

(maximum dist. b/w any two leaf node)

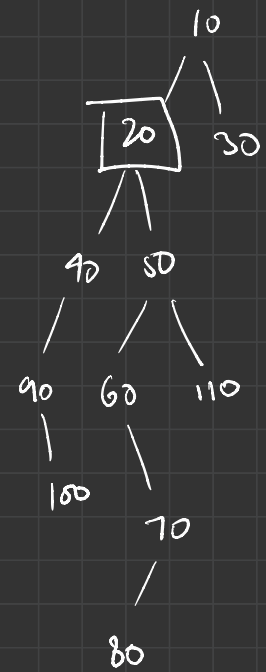


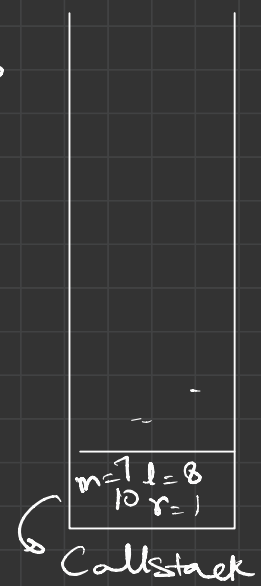

1
2
12
3
7
13
4
9
8
5
10
6
11

```java
public static int diameter(Node root) {
    // Your code here
    if (root == null) {
        return 0;
    }

    // get maximum diamter present in left-subtree
    int maxDiaInLeft = diameter(root.left);

    // get maximum diamter present in right-subtree
    int maxDiaInRight = diameter(root.right);

    // my diamter = height of left sub-tree + 1 + height of right sub-tree
    int myDia = heightOfTree(root.left) + 1 + heightOfTree(root.right);

    // diameter of the tree, max of diameter from left sub-tree, diameter pa
    return Math.max(myDia, Math.max(maxDiaInLeft, maxDiaInRight));
}
```

1 2 3 4 5

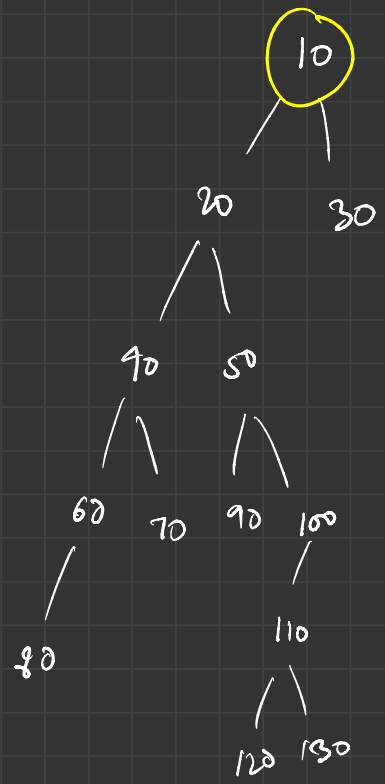
O(N)



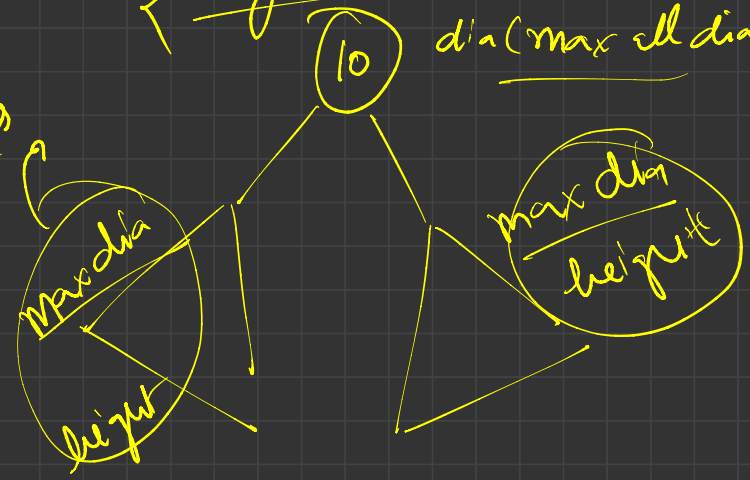


TC: $O(N^2)$

SC: $O(h)$



class

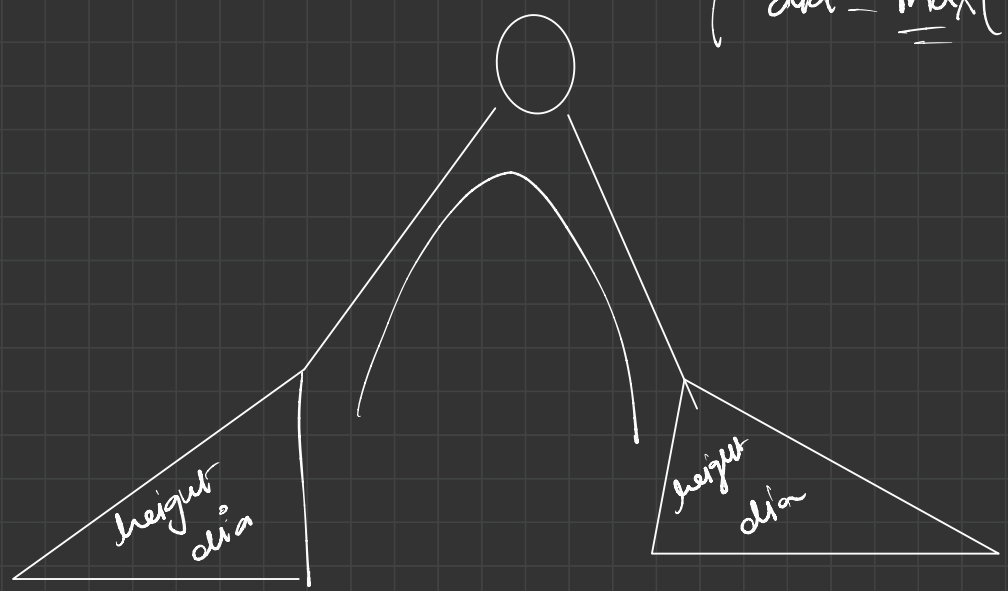


my dia = lh+1+rh

dia (max all dia)

Pair

```
height = max(left.h, right.h) + 1;
dia = max(l.dia, r.dia, (l.h + 1 + r.h))
```



Balanced Binary tree

T

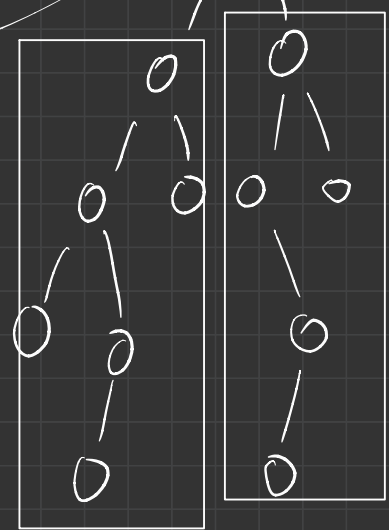
isleftBalanced

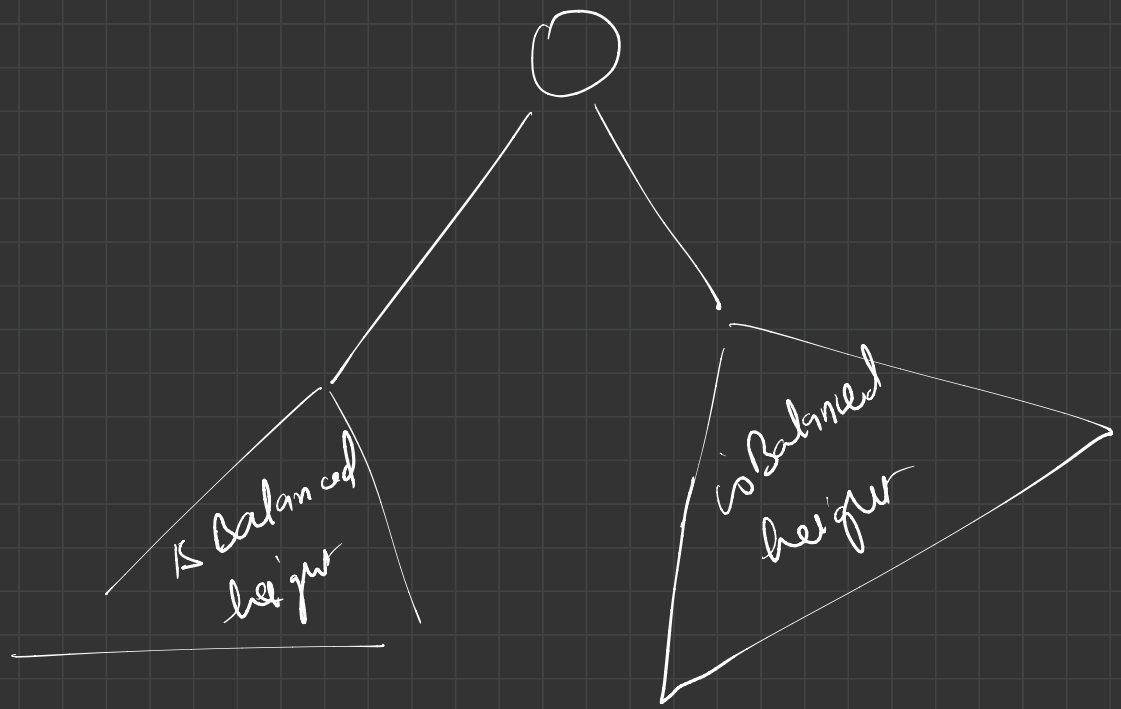
(lh-rh) ≤ 1

T

isright Balanced

T




is Balanced height
is Balanced height