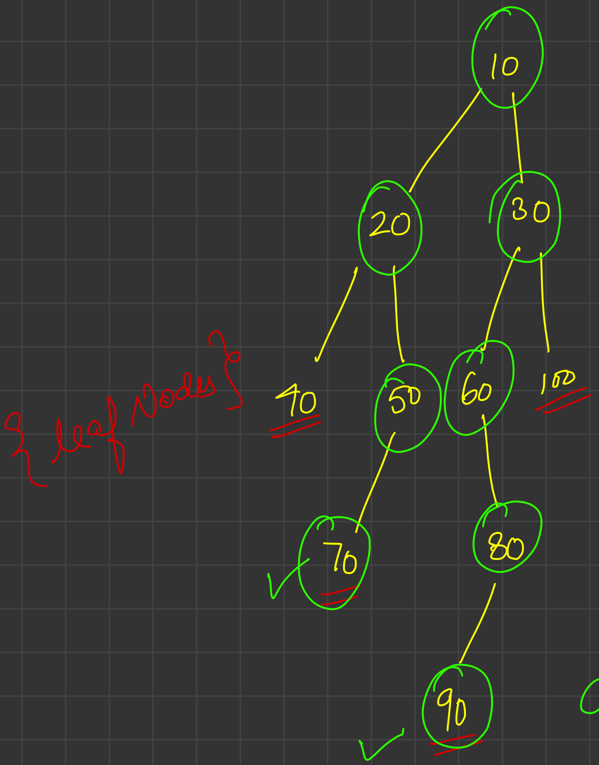
# Diameter of a Tree.

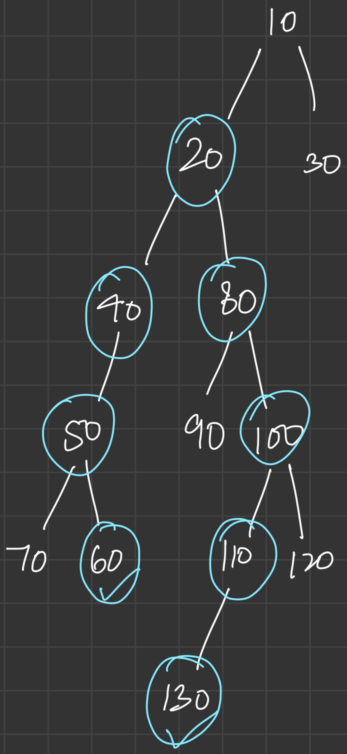
{maxm distance b/w any two leaf Nodes}



$$\boxed{{}^4C_2}$$ → No. of diff. possible diameter.

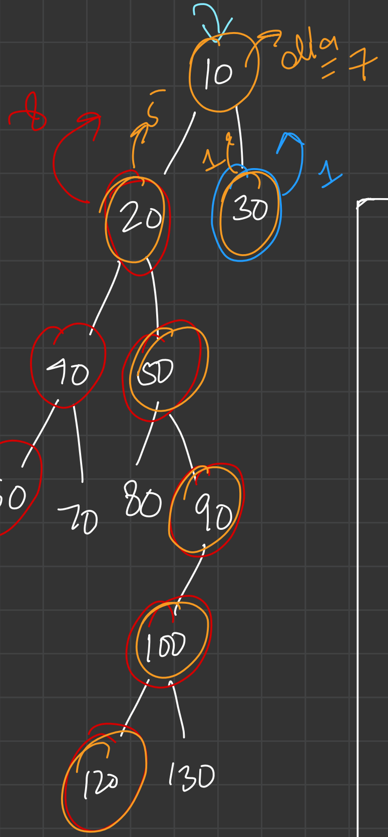
{diameter of the tree}

diameter = 8 ✓



{dia = 8}

TC: $O(N^2)$
SC: $O(H)$



faith: returns diameter of the tree starting from root

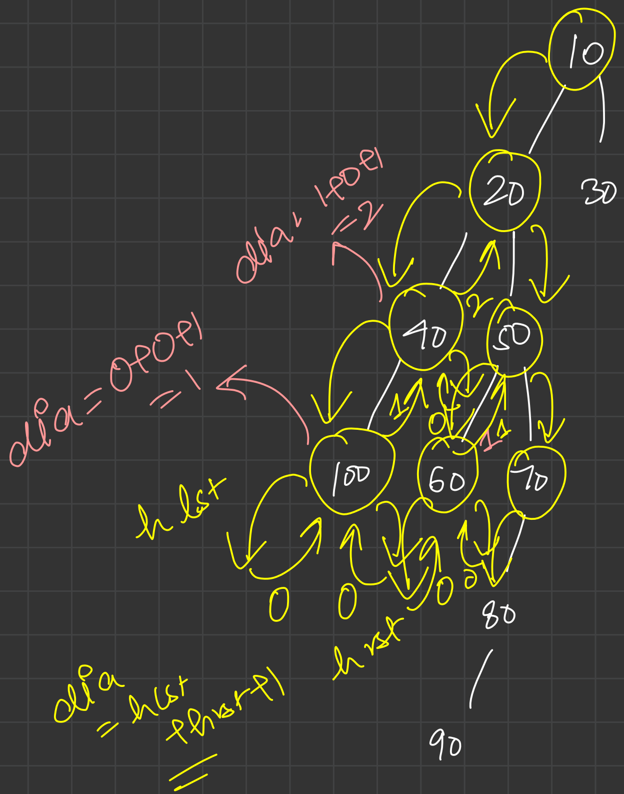
```
int diameter(Node root)
{
    Base Case   root == null : return 0;

    int diaLST = diameter(root.left);
    int diaRST = diameter(root.right);

    Calc. dia. passing through root.
        → h LST
        → h RST
        → diaThroughRoot = hLST + hRST + 1;

    diaTree = max(diaLST, diaRST,
                  passingRoot);
}
```

# diameter in O(N)?

✓height

dia = -~~∞~~ ~~1~~ 2

TC: O(N), SC: O(H)



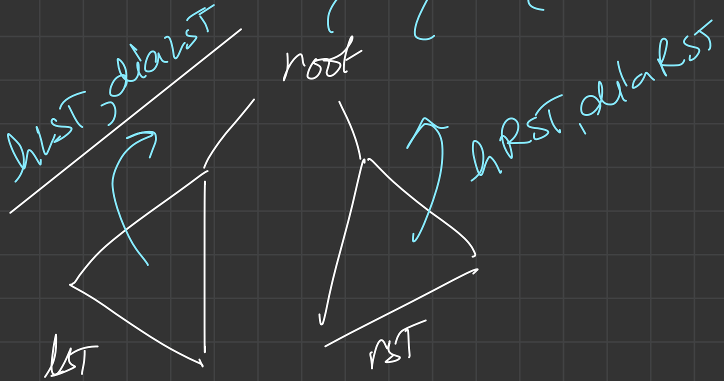
```
int dia = -∞;

int height(Node root)
{   if (root == null) return 0;

    int hLST = height(root.left);
    int hRST = height(root.right);
    dia = max{dia, hLST + hRST + 1};
    return max{hLST, hRST} + 1;
}
```

NOT ALLOWED!

in Interview

# diameter of the tree →

faith: returns dia & Height of the tree !



root → $\{\max(hLST, hRST)+1, \max(diaLST, diaRST, hLST+hRST+1)\}$

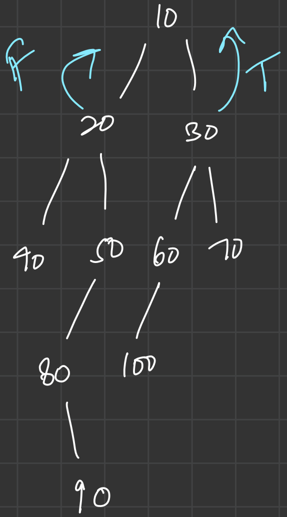
```
class Pair
{ int h;
  int d;
}
```

# Is Tree Balanced ?

↳ when each Node is Balanced !

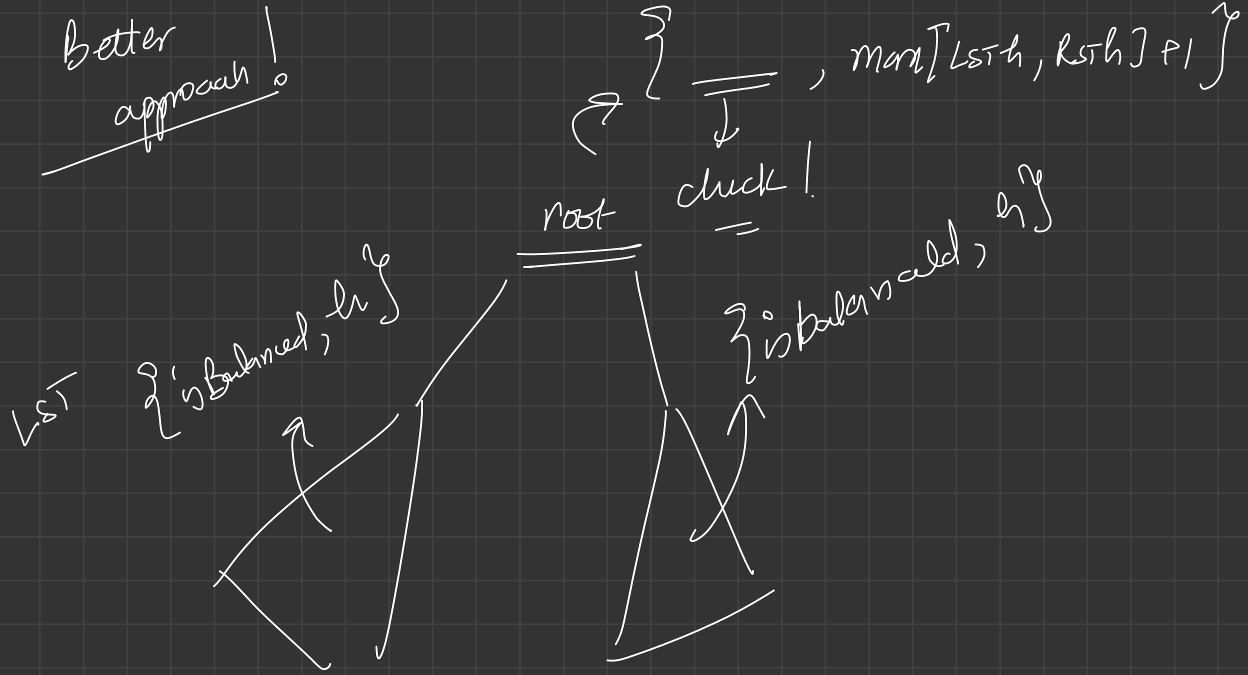
✓ $| hLST - hRST | \le 1$

faith: returns a tree is balanced or not - !



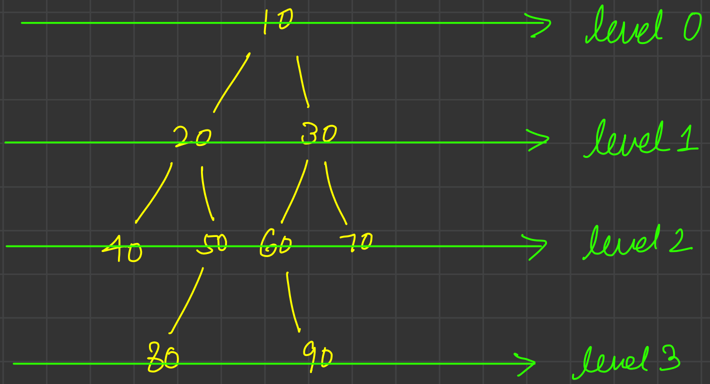
```
boolean isBalanced ( Node root)
{   if (root ==NULL) return T;
    boolean LSTB = isBalanced(root.left);
    boolean RSTB = isBalanced(root.right);
    Check root Balanced or Not,
        ↳ get height and abs diff .
    if (LSTB==True && RSTB==True &&
                     rootisBal==True)
            returnT;
} else   return F;
```

Better approach!

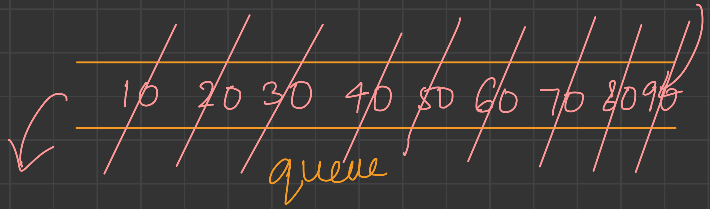


faith! returns is Tree Balanced & height

# Level Order Traversal.



BFS.



80
90

size = ~~2~~ ~~1~~ 0

o/p

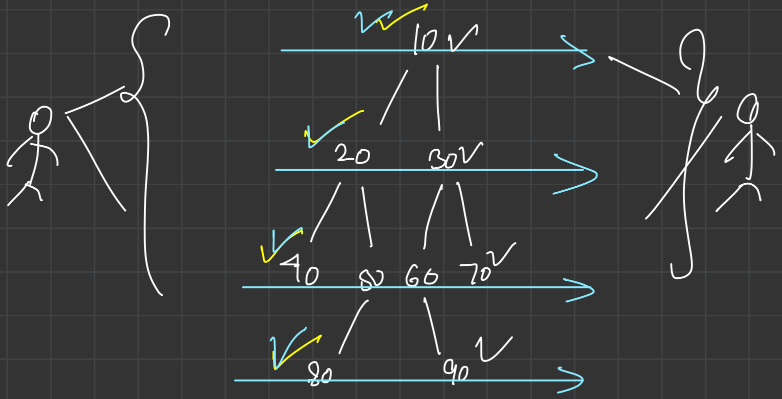
{
↳ 10
↳ 20, 30
↳ 40, 50, 60, 70
↳ 80, 90

o/p

10
↳ 20, 30
↳ 40, 50, 60, 70
↳ 80, 90

## Left View

{first Node of each Level}



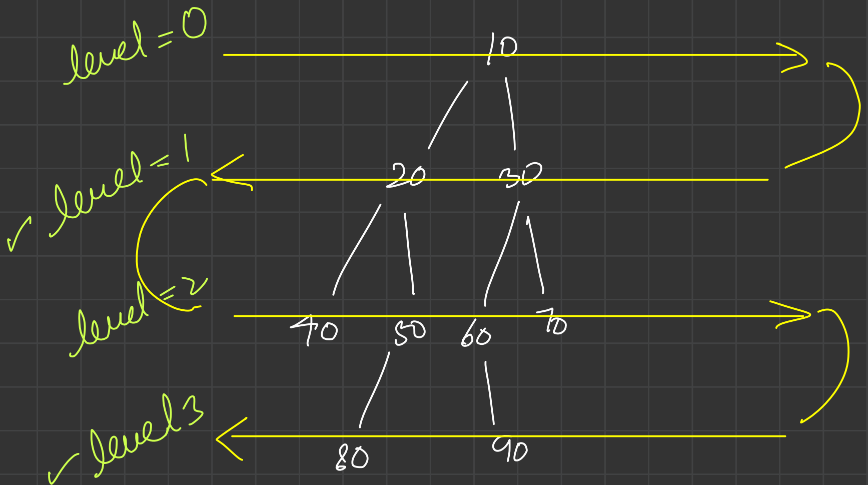
## Right View

{Last Node of each Level}

o/p 10,30,70,90.

o/p 10,20,40,80.

# Zig Zag Traversal

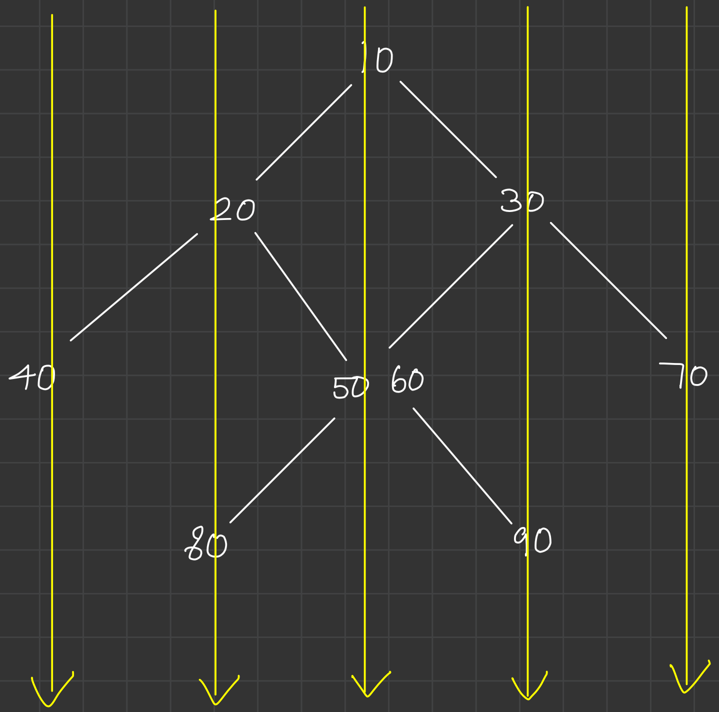


odd level No. (R→L)
= reverse

even level No.
= (L→R)

o/p {10, 30, 20, 40, 50, 60, 70, 90, 80}

# Vertical Order Traversal



o/p

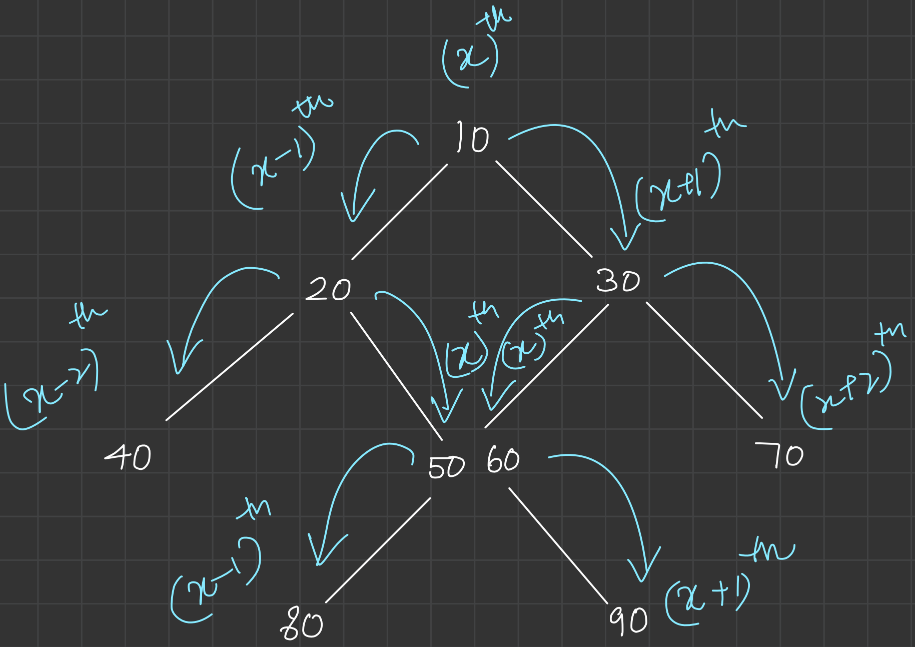
40

20, 80

10, 50, 60

30, 90

70



(60, $x$) (70, $x+2$) (80, $x-1$) (90, $x+1$)

$x-2 \longrightarrow 40$

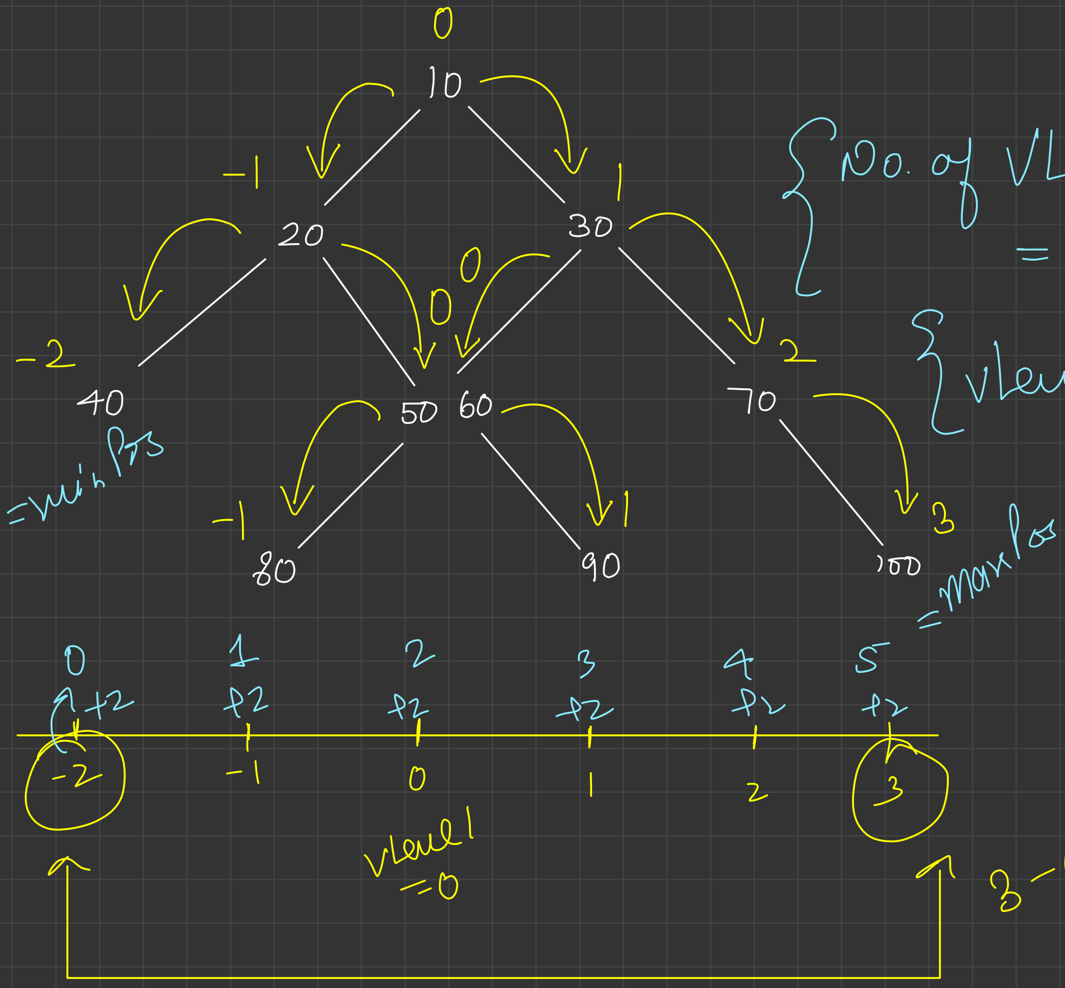
$x-1 \longrightarrow 20, 80$

$x \longrightarrow 10, 50, 60$

$x+1 \longrightarrow 30, 90$

$x+2 \longrightarrow 70$

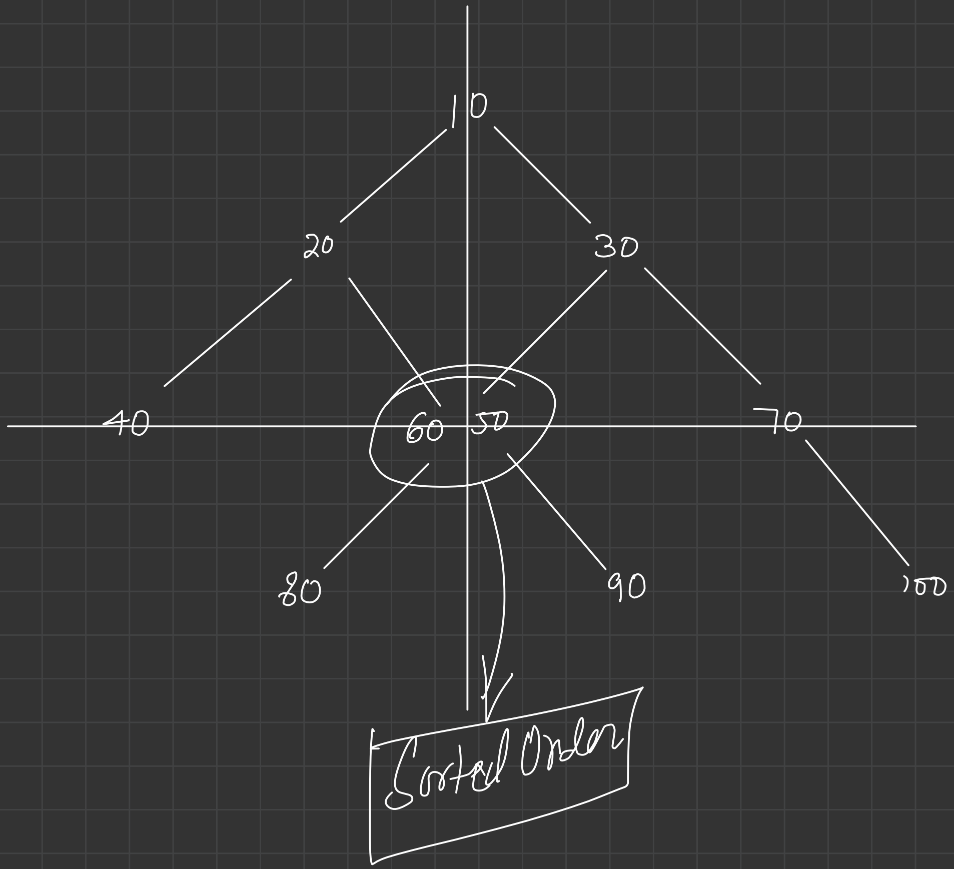
- * vLevel of root
- * No. of vLevel



{No. of VLevel = maxPos - minPos + 1

{vLevel of root = - minPos;

$3-(-2)+1$
$= 6$


10
20
30
40
60
50
70
80
90
100
Sorted Order

{ PriorityQueue

↳ own Priority!