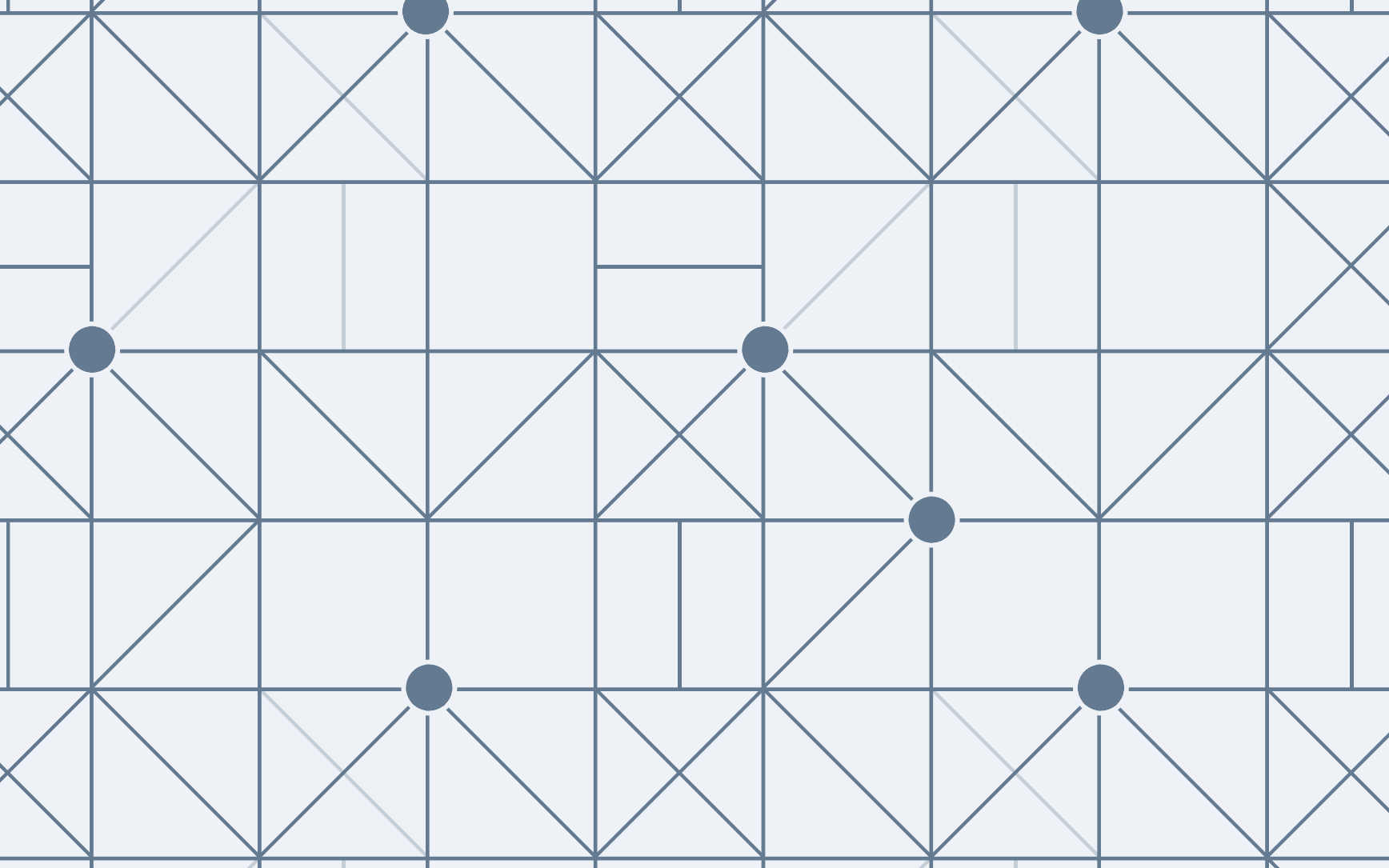
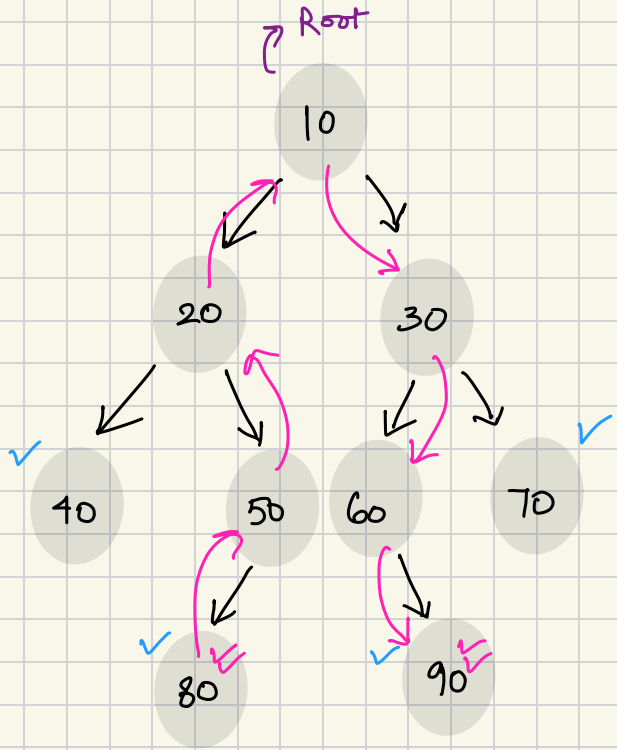


Diameter of a Binary Tree {Max^m distance b/w any two leaf Nodes}



{40, 80} → 4

{40, 90} → 6

{40, 70} → 5

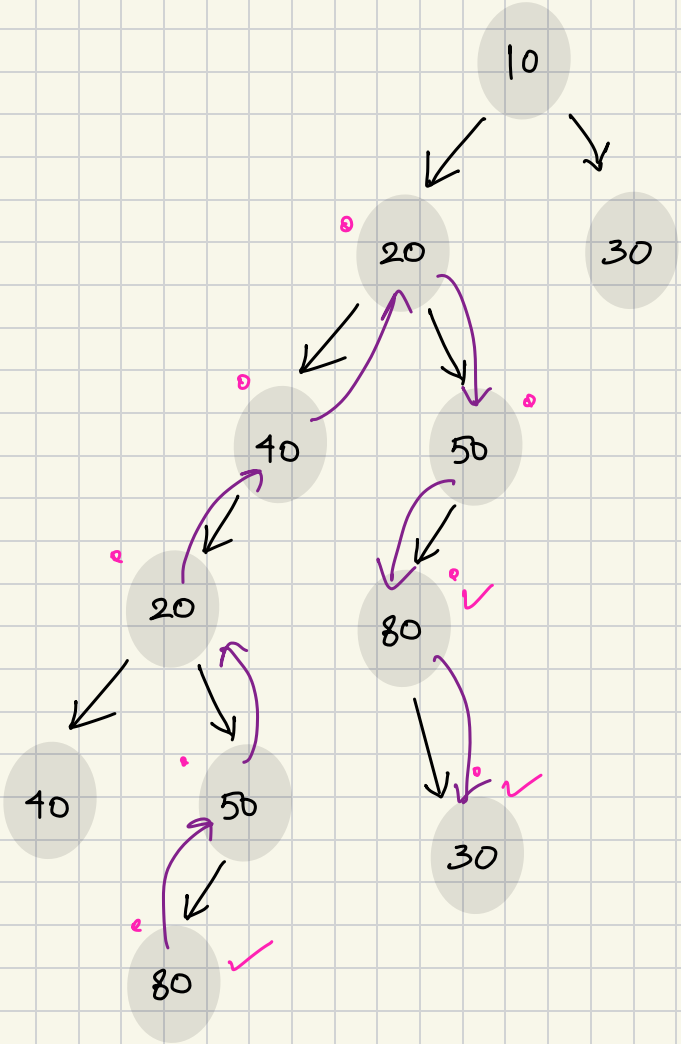
{80, 90} → 7

{80, 70} → 6

{90, 70} → 4

diameter = 7 units ✓

{~~diameter = h LST + h RST + 1~~} ✓ ~~Wrong~~.



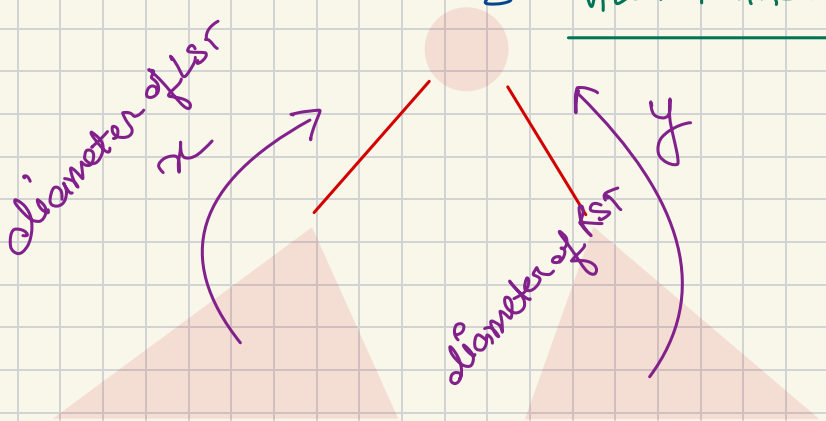
NOTE: Diameter not passes through root

faith: give the diameter of the tree starting from root.

```
int diameter(Node root)
{

}
```

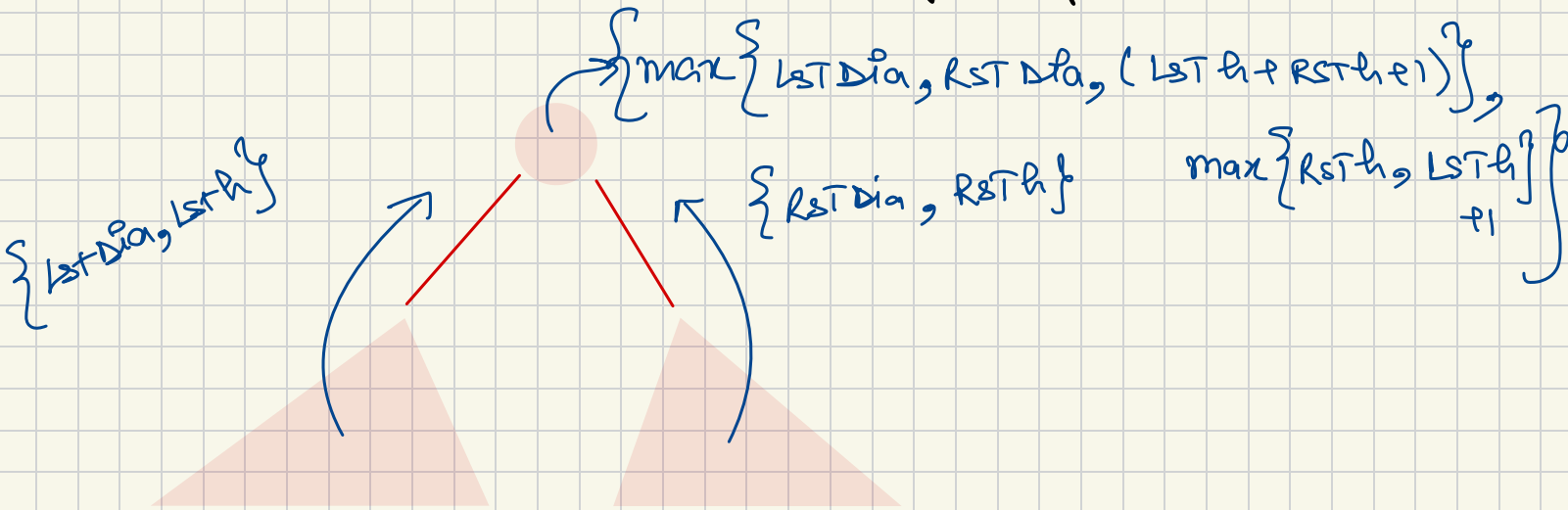
diameter passing through root.

$z = hLST + hRST + 1$



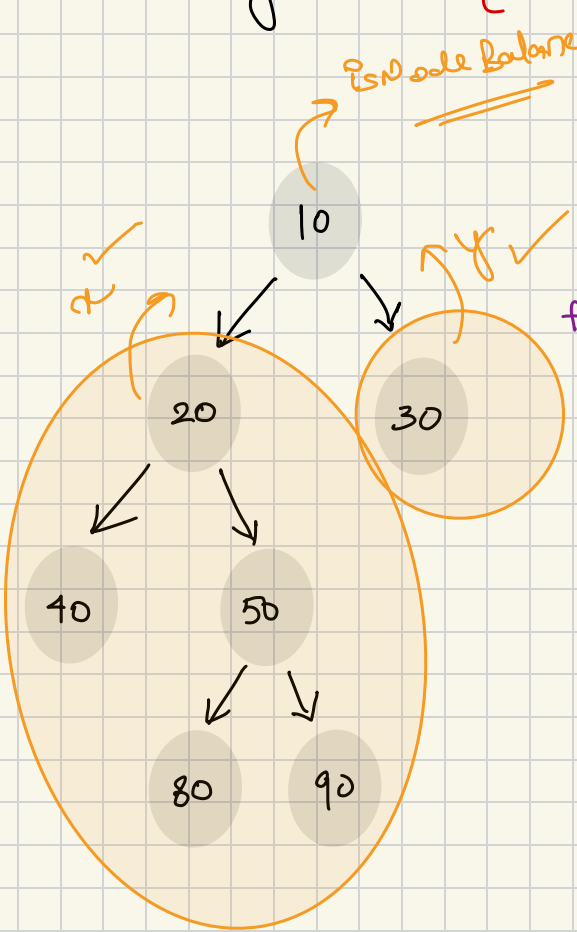
overall dia = $\max\{x, y, z\}$ ✓

faith: returns {diameter, height} of the given BT from root.



{diameter, height}

Balanced Binary Tree { if each node is balanced }



isNodeBalanced ✓

Balanced Node.

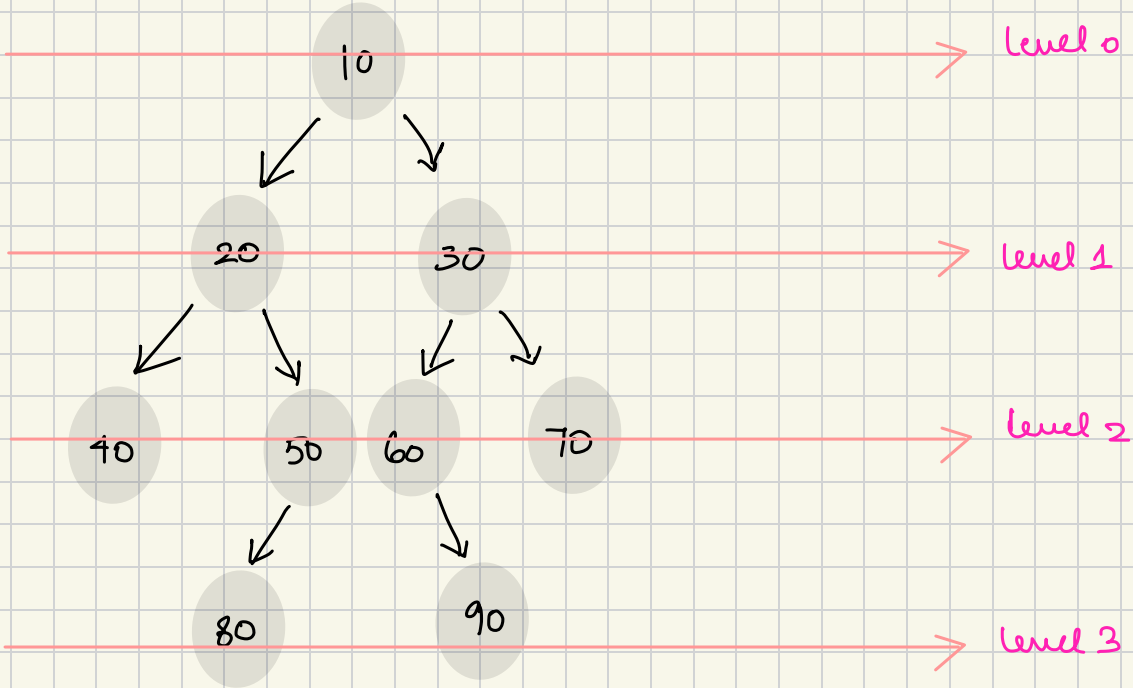
tree is balanced!

$|h_{LST} - h_{RST}| \leq 1$

faith: returns is tree balanced from a given root

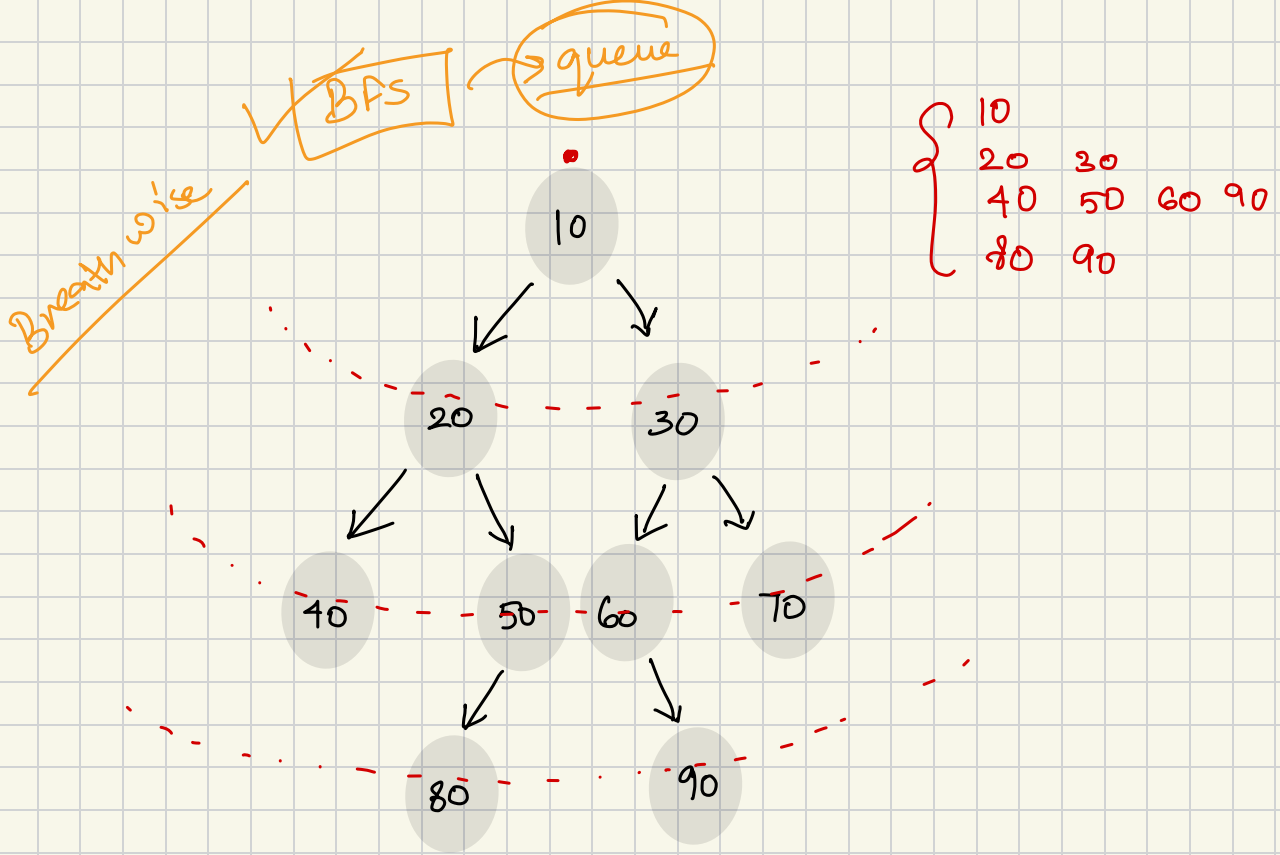
```
boolean isBalanced(Node root)
{

}
```

Level order traversal.

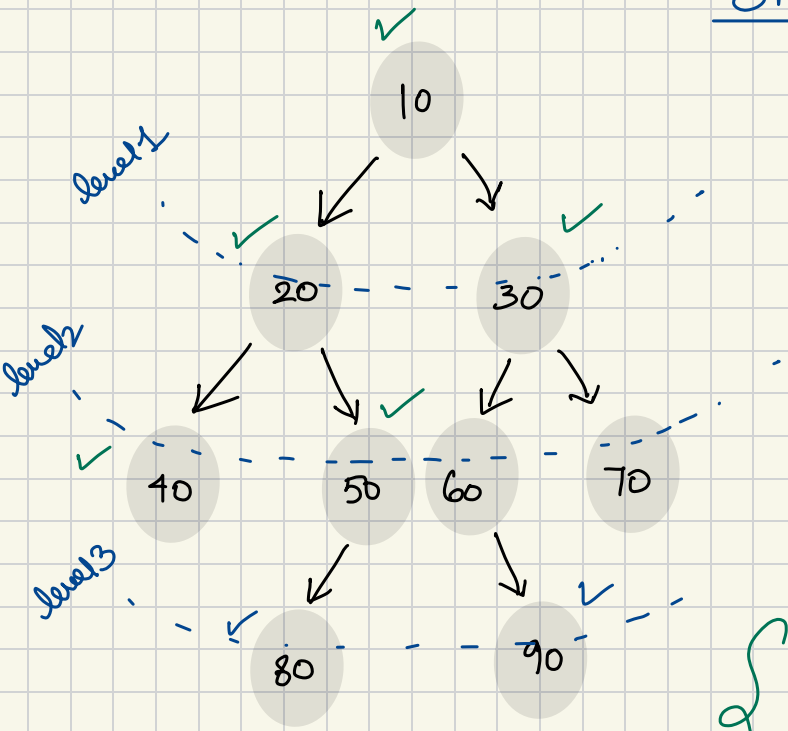


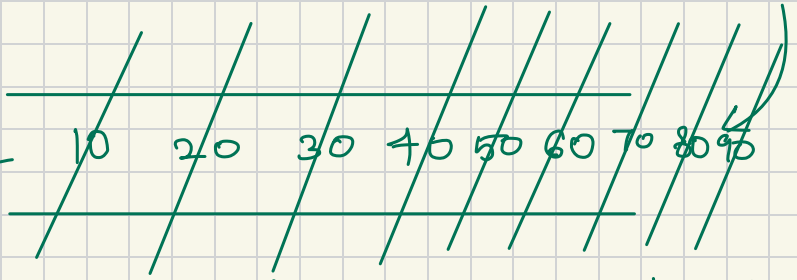
O/P

10
20 30
40 50 60 70
80 90


BFS
queue
Breadth wise
10
20
30
40
50
60
70
80
90
10
20 30
40 50 60 90
80 90

BFS Algorithm





size = ~~0~~

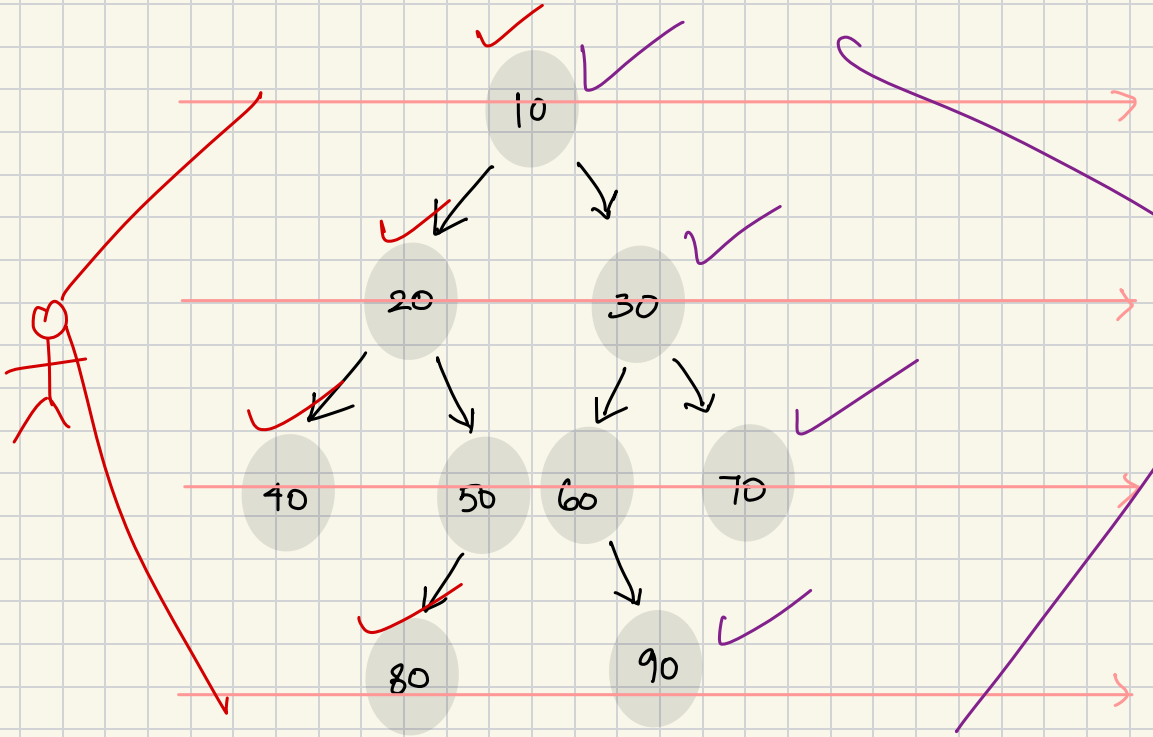
level = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~

~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~ ~~11~~ ~~12~~ ~~13~~ ~~14~~ ~~15~~

10
20 30
40 50 60 70
80 90

Left View.

Right View.



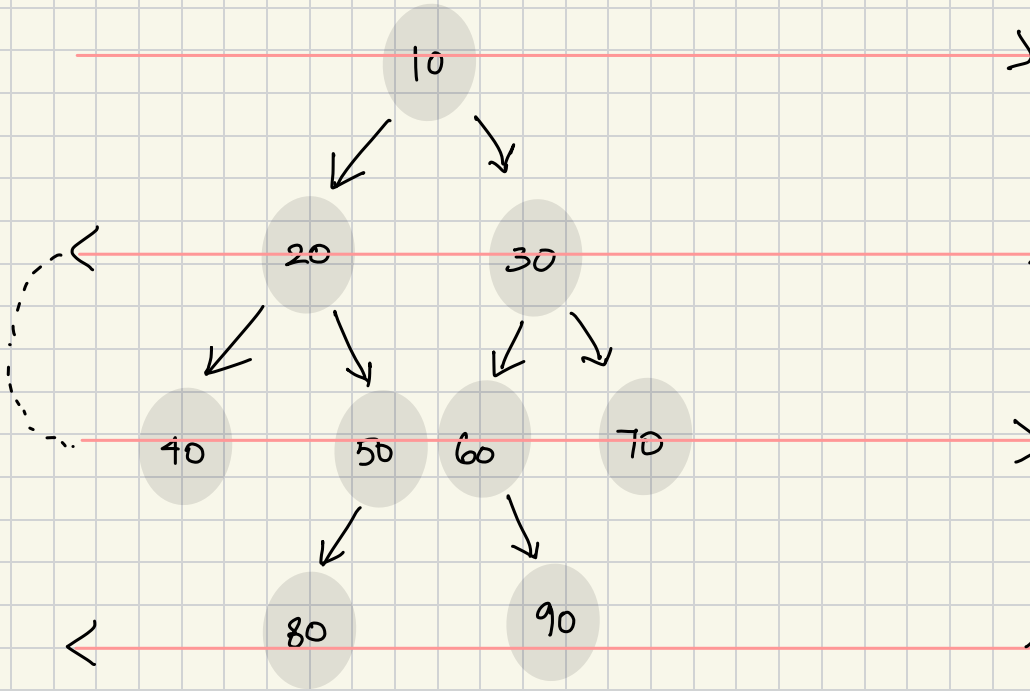
Left View

10, 20, 40, 80

RV

10, 30, 70, 90

Zig Zag traversal



O/P

→ 10

→ 30 20

→ 40 50 60 70

→ 90 80

if level is even:

print L → R curr level

if level is odd:

print R → L curr Level