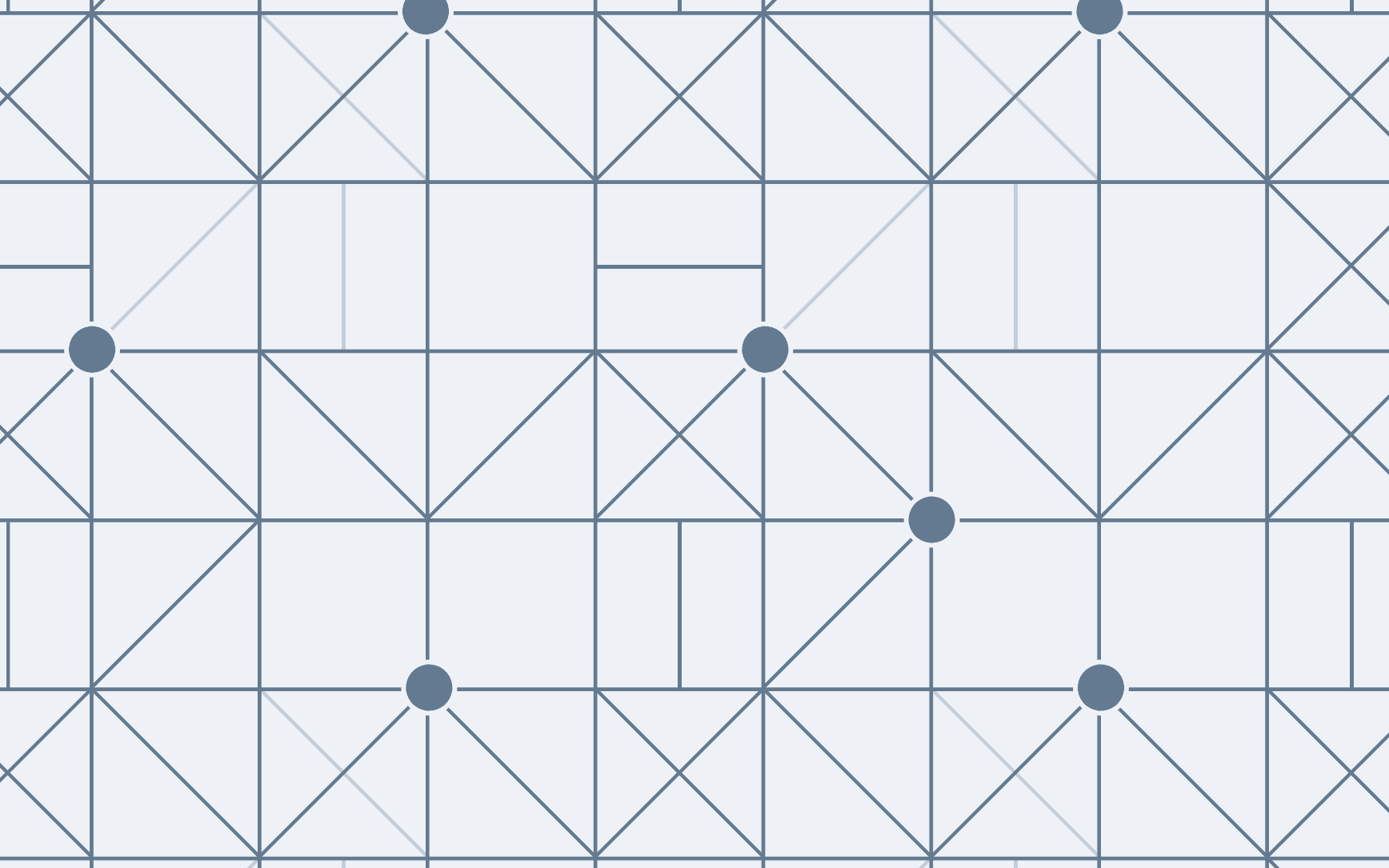
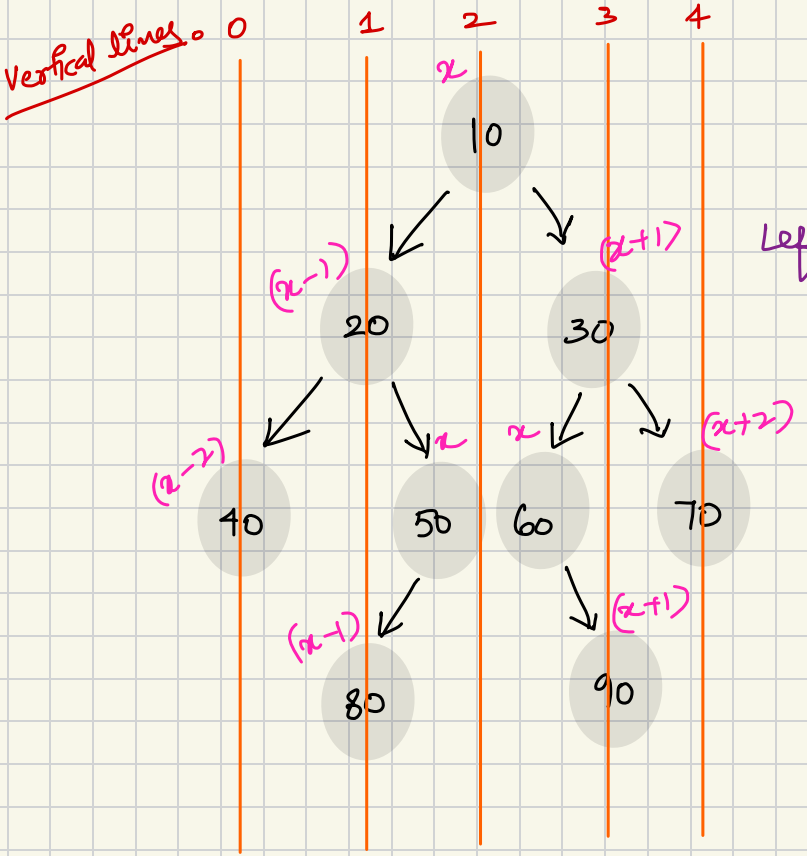
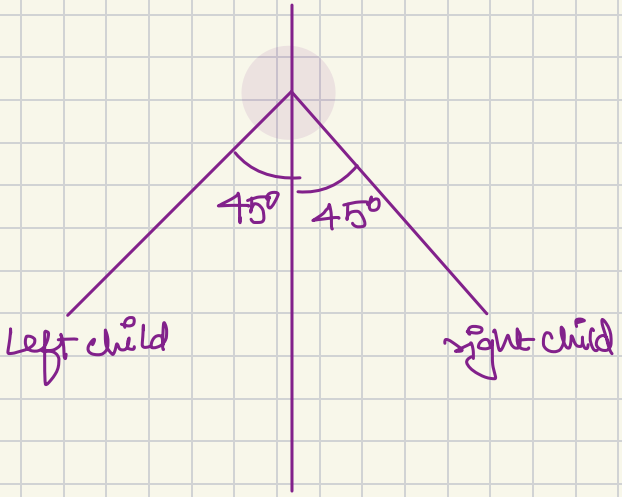


Vertical Order traversal





vertical line

0 → 40

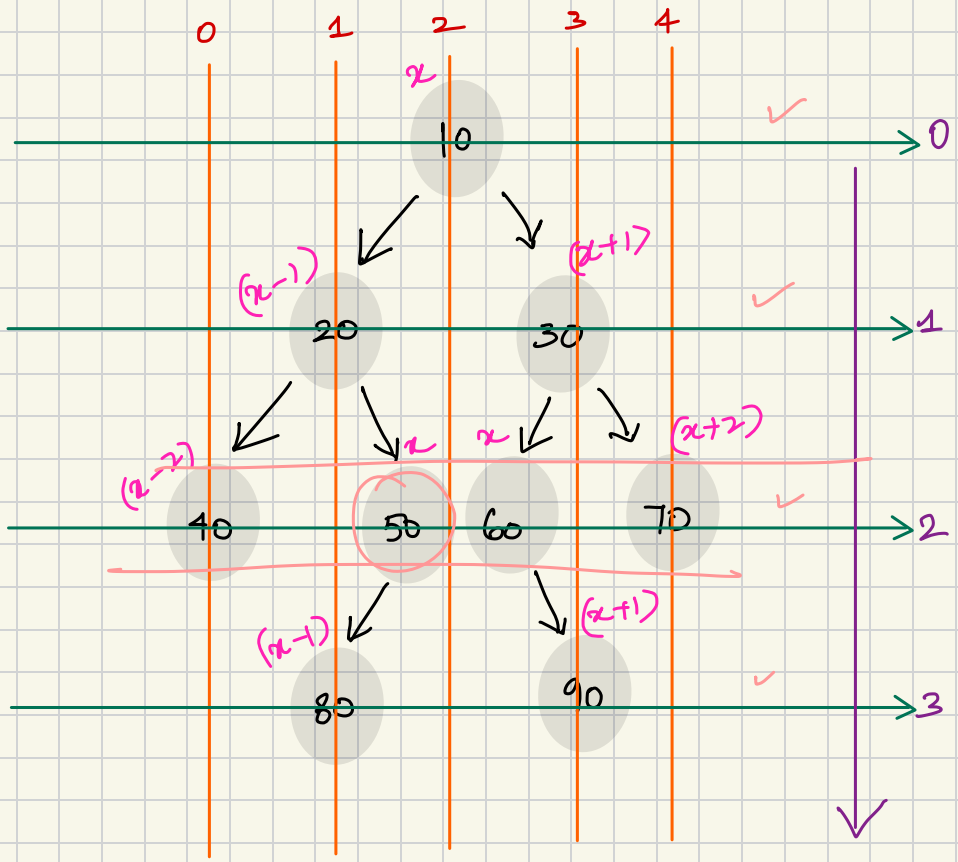
1 → 20 80

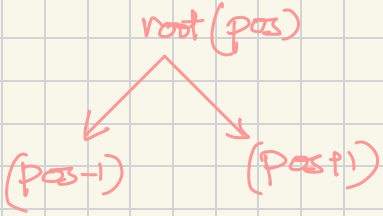
2 → 10 50 60

3 → 30 90

4 → 70

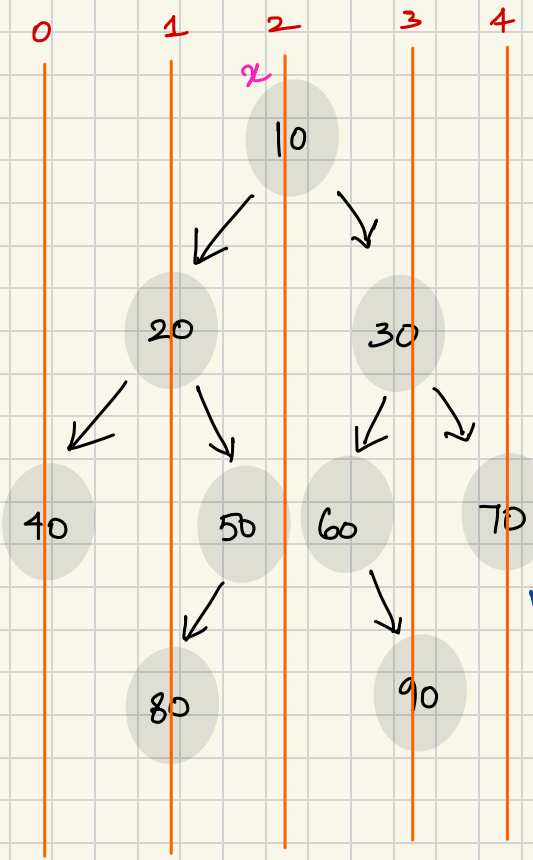
Vertical Order traversal



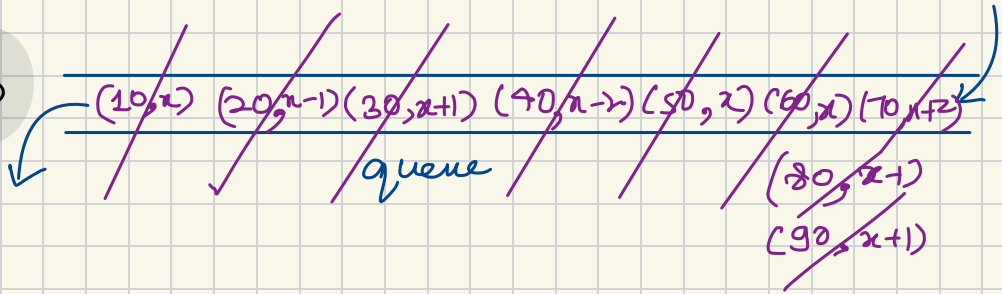


0 → (x-2) → 40
1 → (x-1) → 20 80
2 → (x) → 10 50 60
3 → (x+1) → 30 90
4 → (x+2) → 70

pos → top to bottom ✓

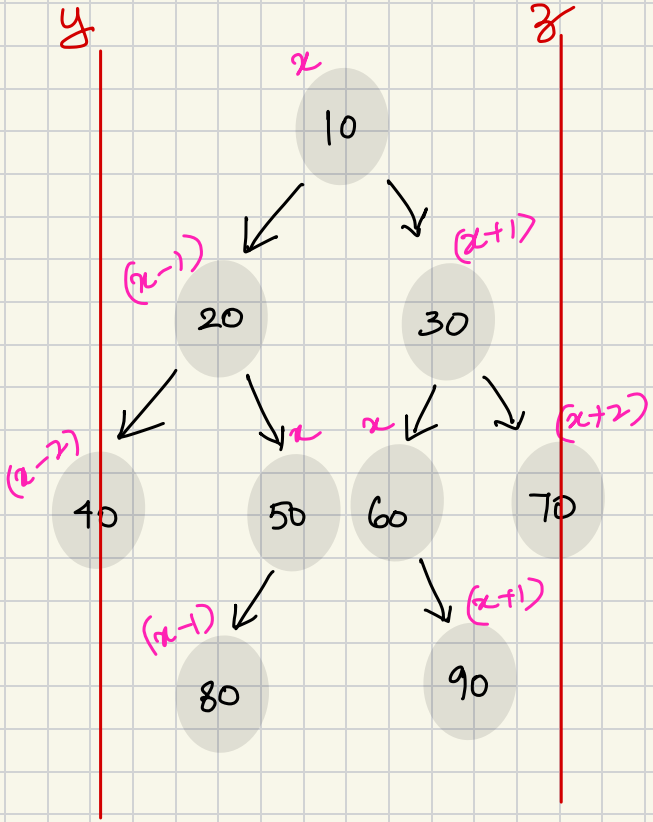


0 (x-2) ⇝ 40
1 (x-1) ⇝ 20, 80
2 (x) ⇝ 10, 50, 60
3 (x+1) ⇝ 80, 90
4 (x+2) ⇝ 70



✓ : x {vertical level of root}

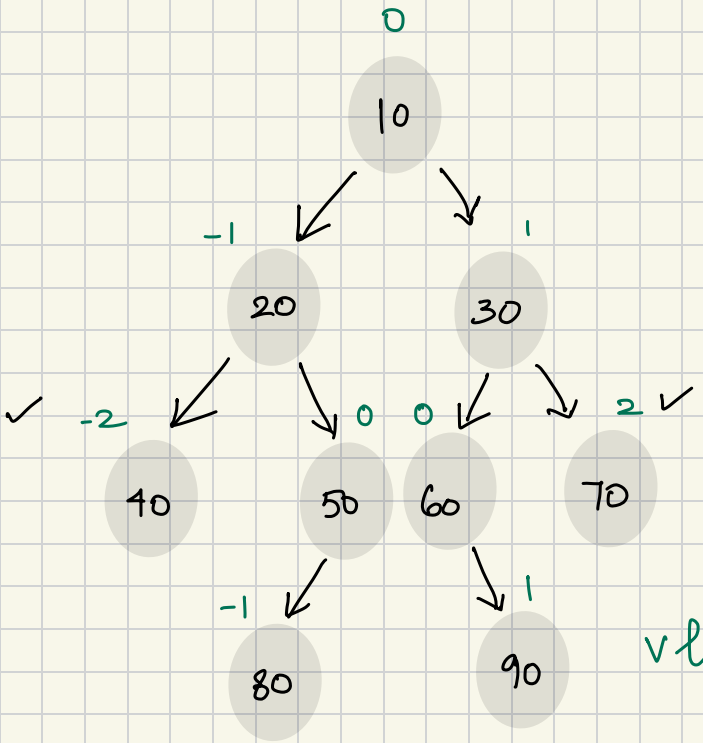
✓ : no. vertical levels



✓ no. of vlevels

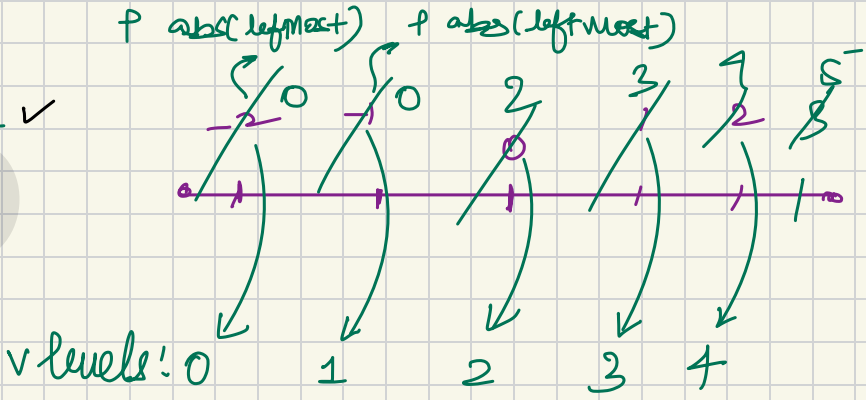
= {right most pos - left most pos}

+ 1



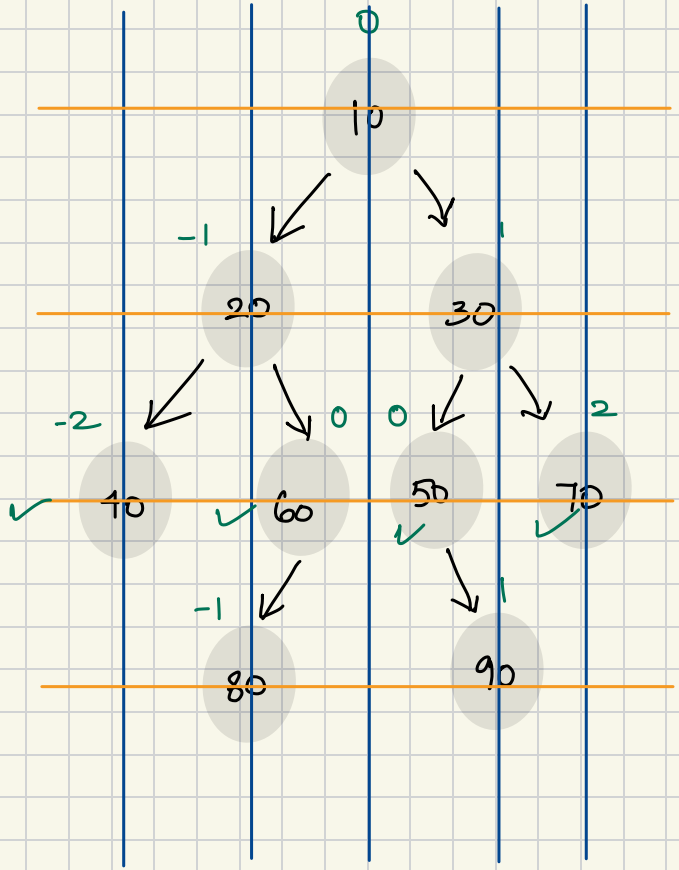
vlevel of root = abs(leftMost)

no. of vertical levels = 2 − (−2) + 1
= 5



Step 1 Calc. Level of root, and no of level

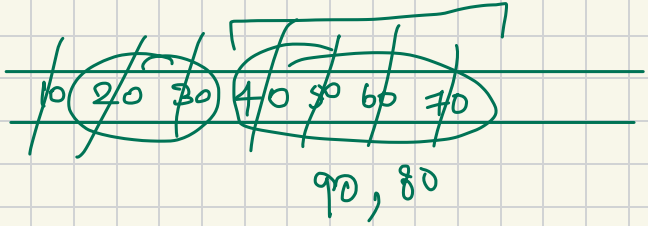
Step 2 : do a level order traversal



Vo

0 → 40
1 → 20, 80
2 → 10, ~~60, 50~~ {50, 60}
3 → 30, 90
4 → 70

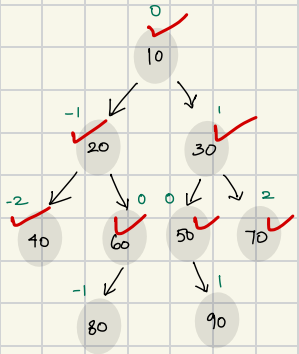
Note: If nodes are at same level and that level then they should be in a sorted order



priority queue { for each level }

sort in inc. vlevel

if two people have same vlevel, sort them in inc. order



pPq

cPq

(80, -1) (90, 1)

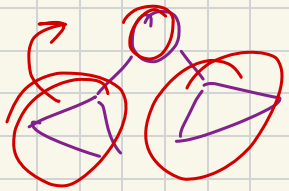
Bottom view

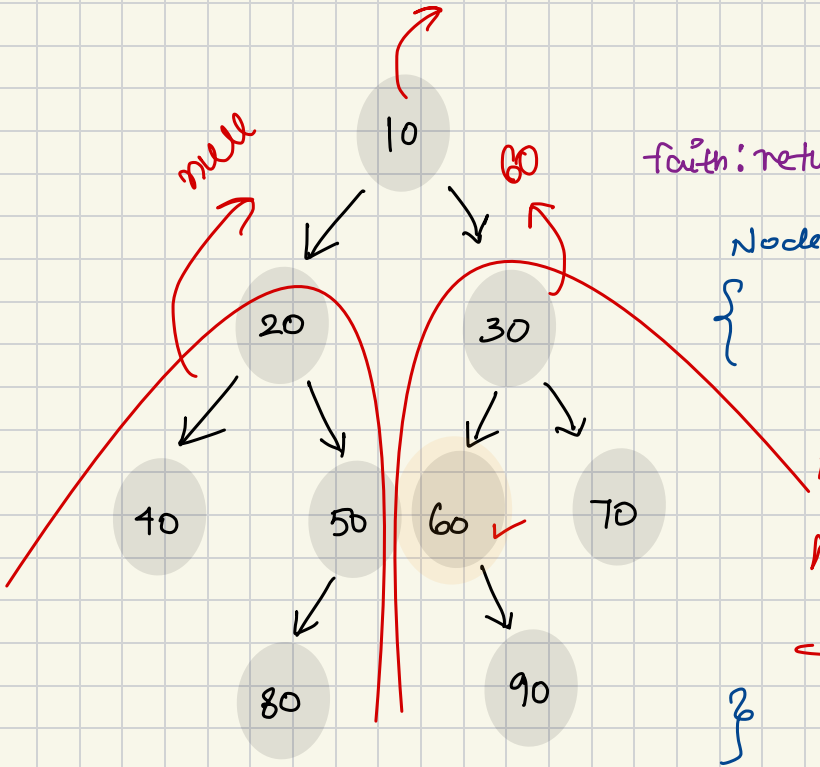
↳ last entity of each vlevel

top view

↳ first entity of each vlevel

Find a given Node in a BT

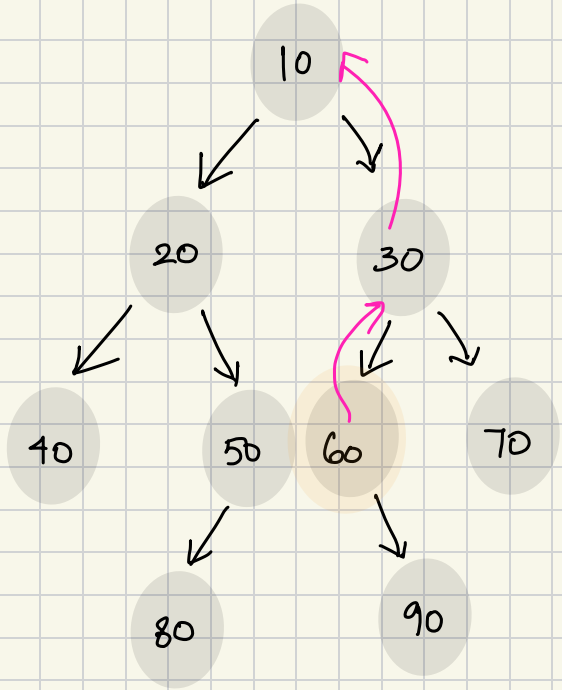




faith: returns the add. of target if found in BT

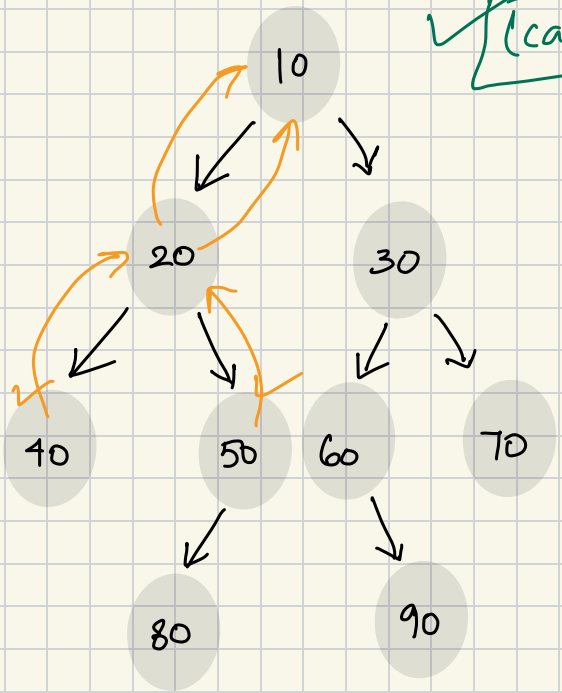
```
Node find (Node root, int tar)
{
    Node fole = ____
    Node fisc
    ____
}
```

Node to Root Path.



path: 60 → 30 → 10

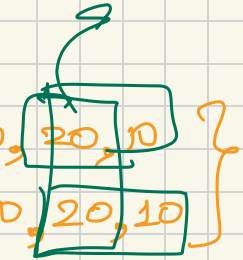
LCA { lowest Common Ancestor }

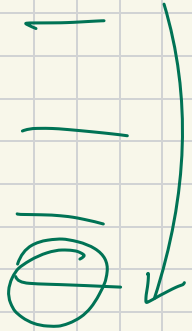


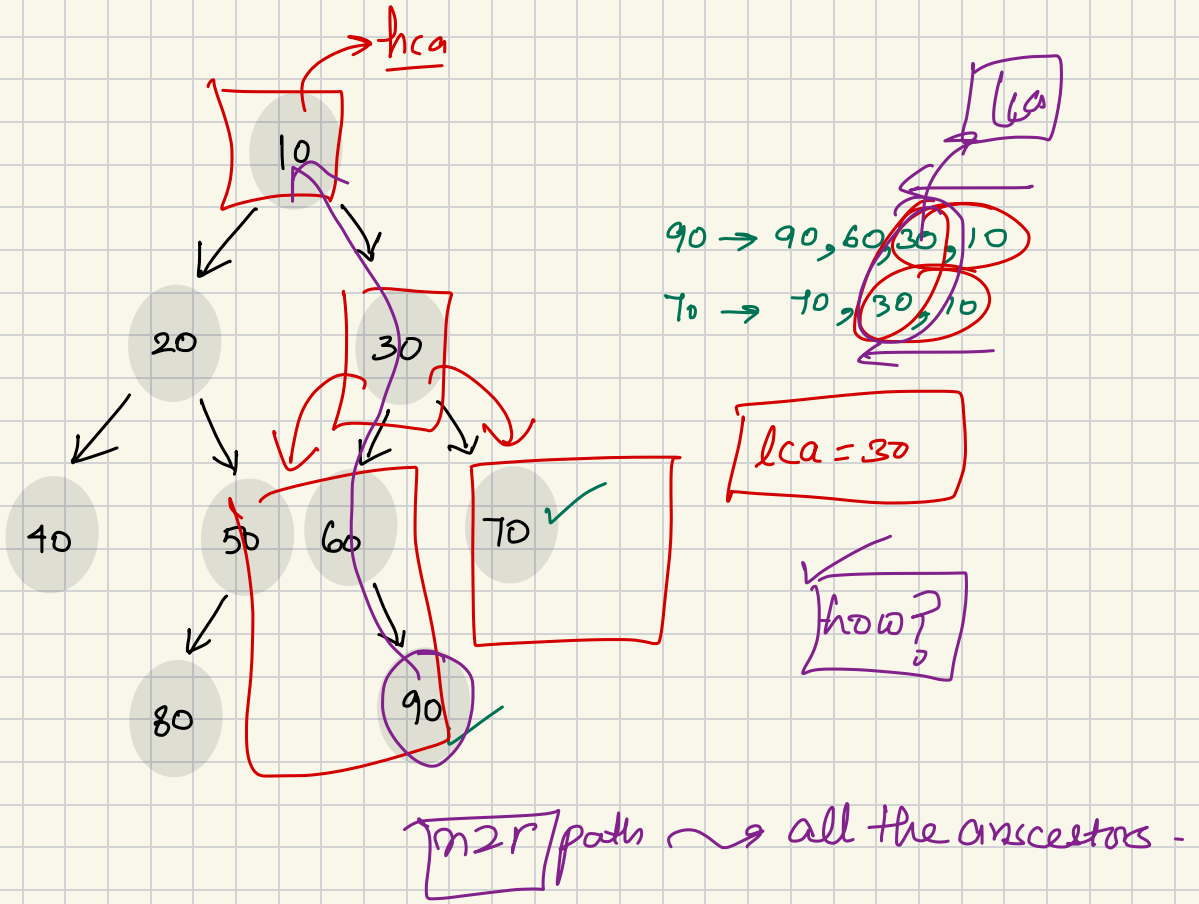
lca = 20

40 → 40, 20, 10

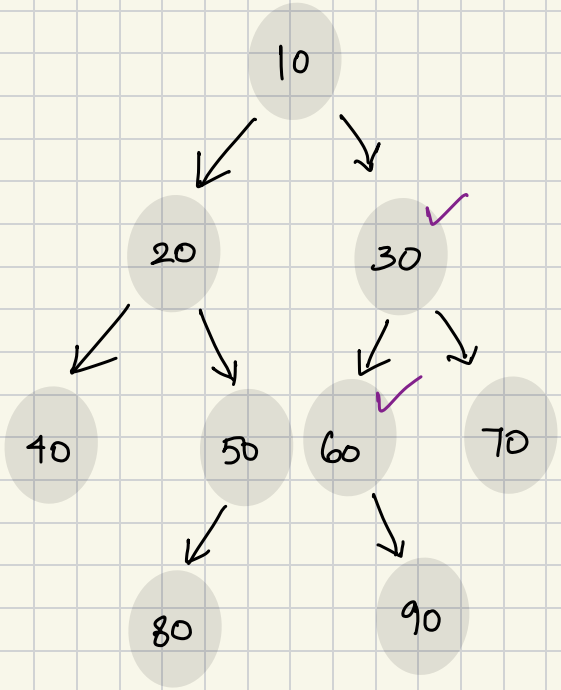
50 → 50, 20, 10





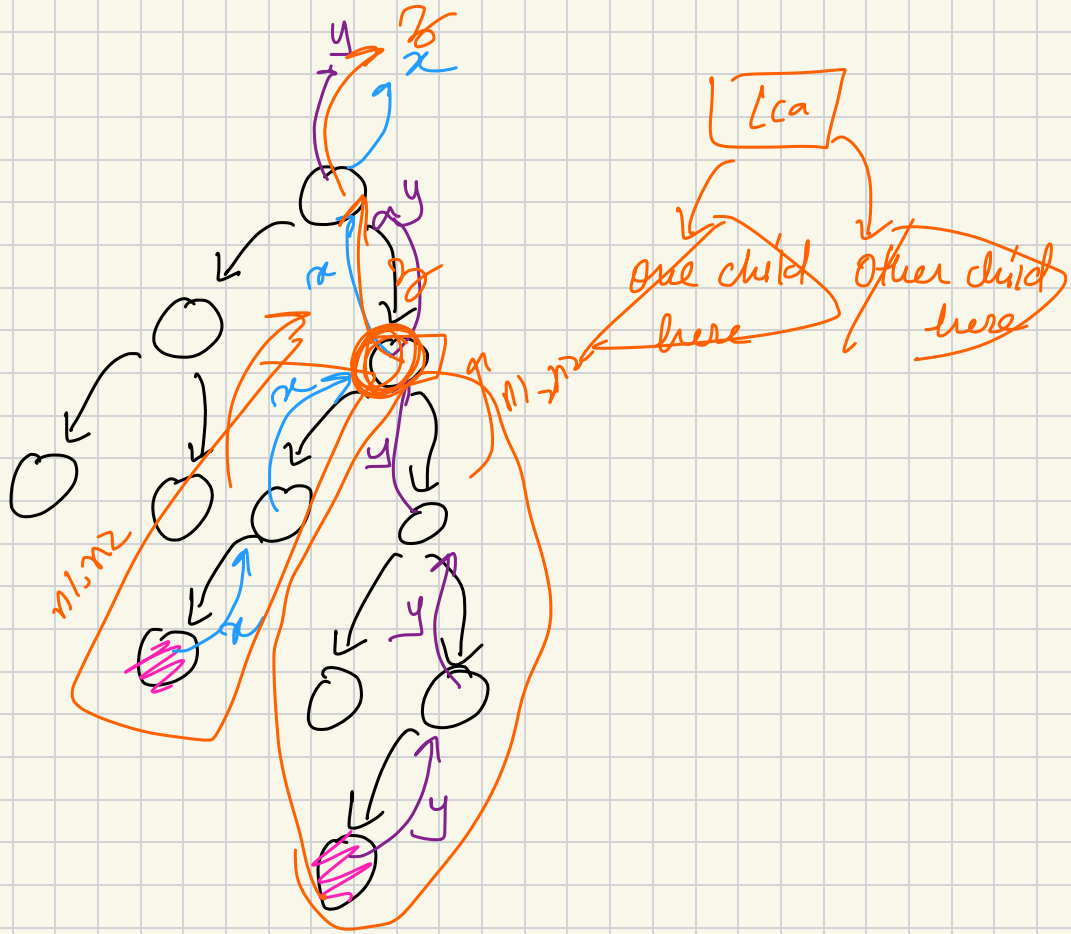


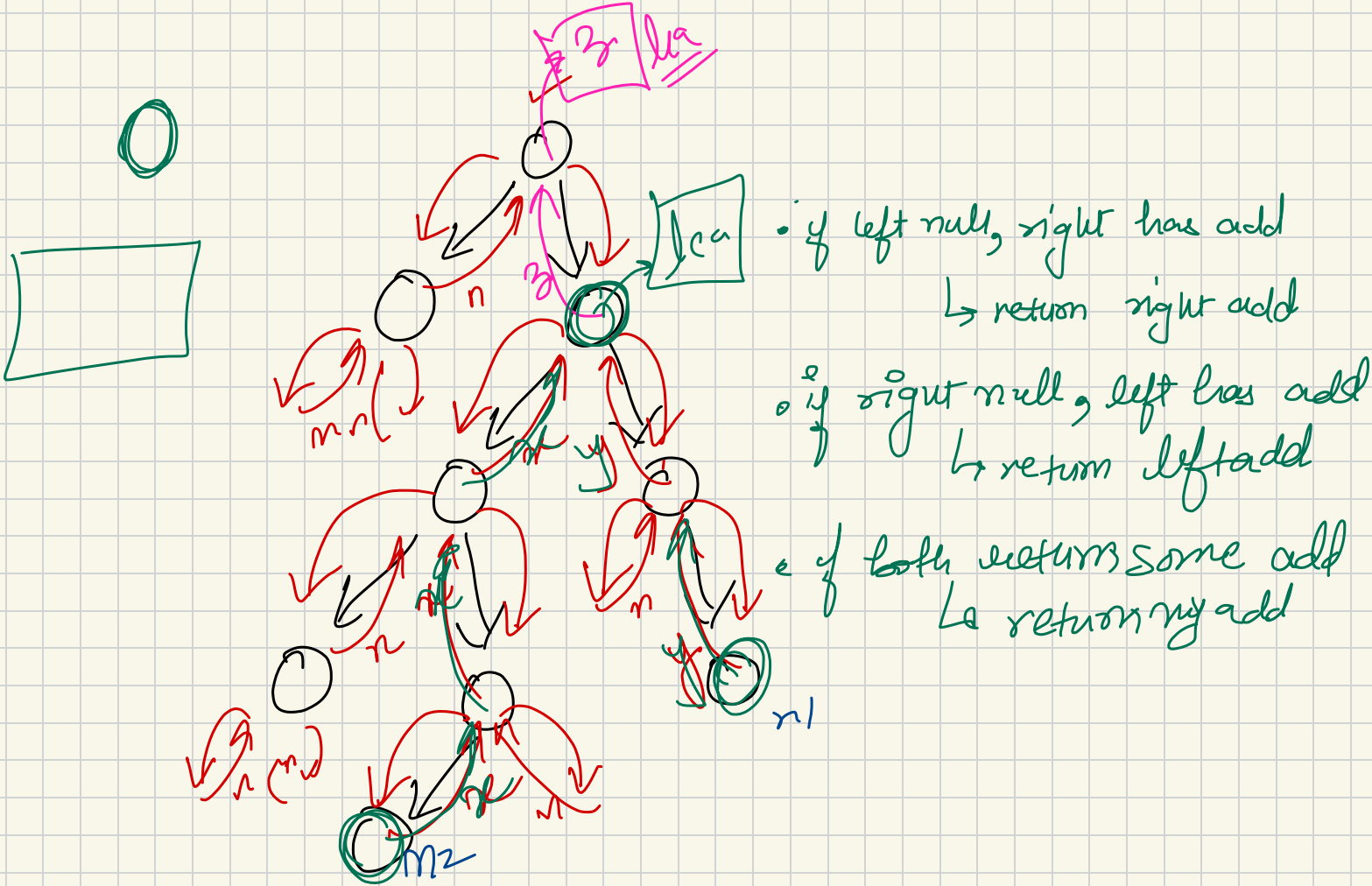
m2r/path ~> all the ancestors.



60 → 60 30 10

30 → 30 10


y
z
x
y
x
y
x
y
y
y
y
lca
one child here
other child here



- if left null, right has add
  - ↳ return right add
- if right null, left has add
  - ↳ return left add
- if both return some add
  - ↳ return my add