Agenda

- Vertical Order traversal
- find a node
- path to a given node
- LCA

Lambda function.

- Integer [] arr = {5, 2, 7, 8, 3, 1, 4, 6}

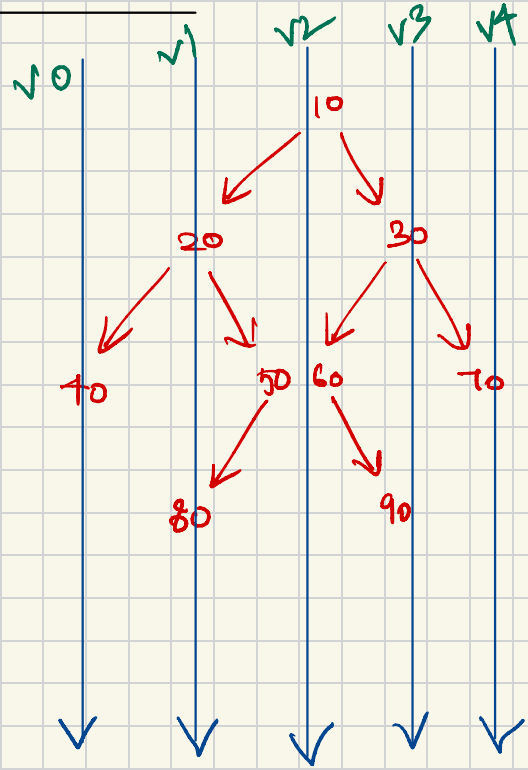
a) sort in inc order?

```
Arrays.sort(arr);
```

b) sort in dec. order?

```
Arrays.sort(arr, (a, b) -> {
        return b - a;
});
```

Vertical Order Traversal



Vtraversal

0

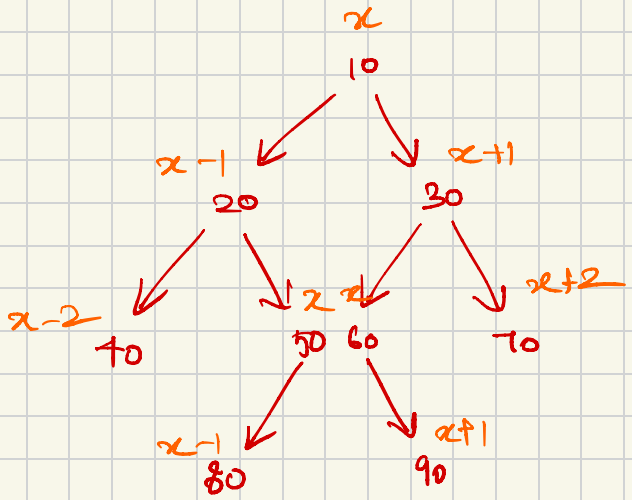
40

20 80

10 50 60

30 90

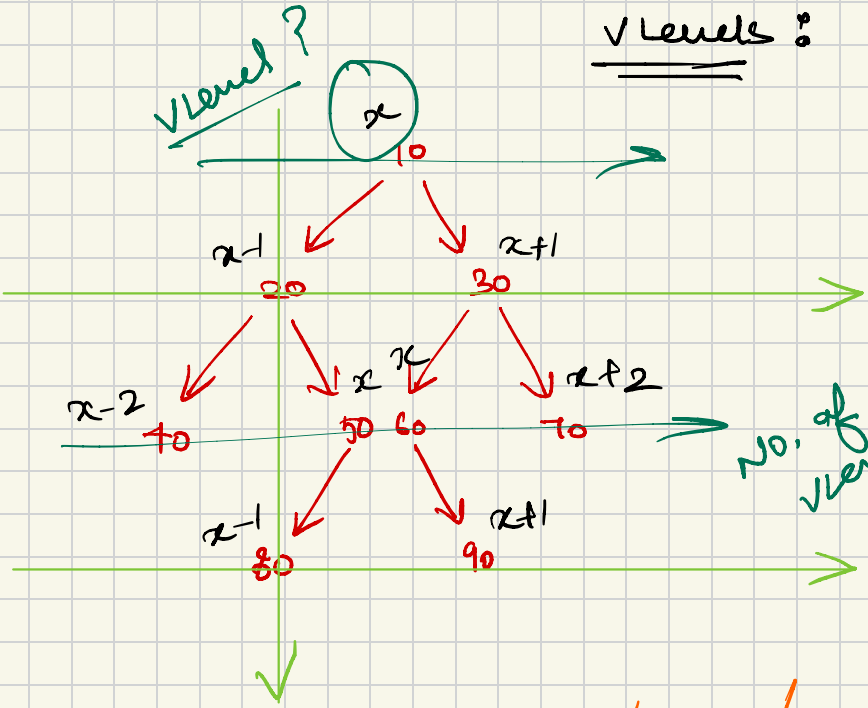
70



- when moving left
  vlevel decreases by 1
- when moving right
  vlevel increases by 1

wrt. the parent node.

V Levels ?



No. of Vlevel ?

x-2

x-1 : 20

x : 10

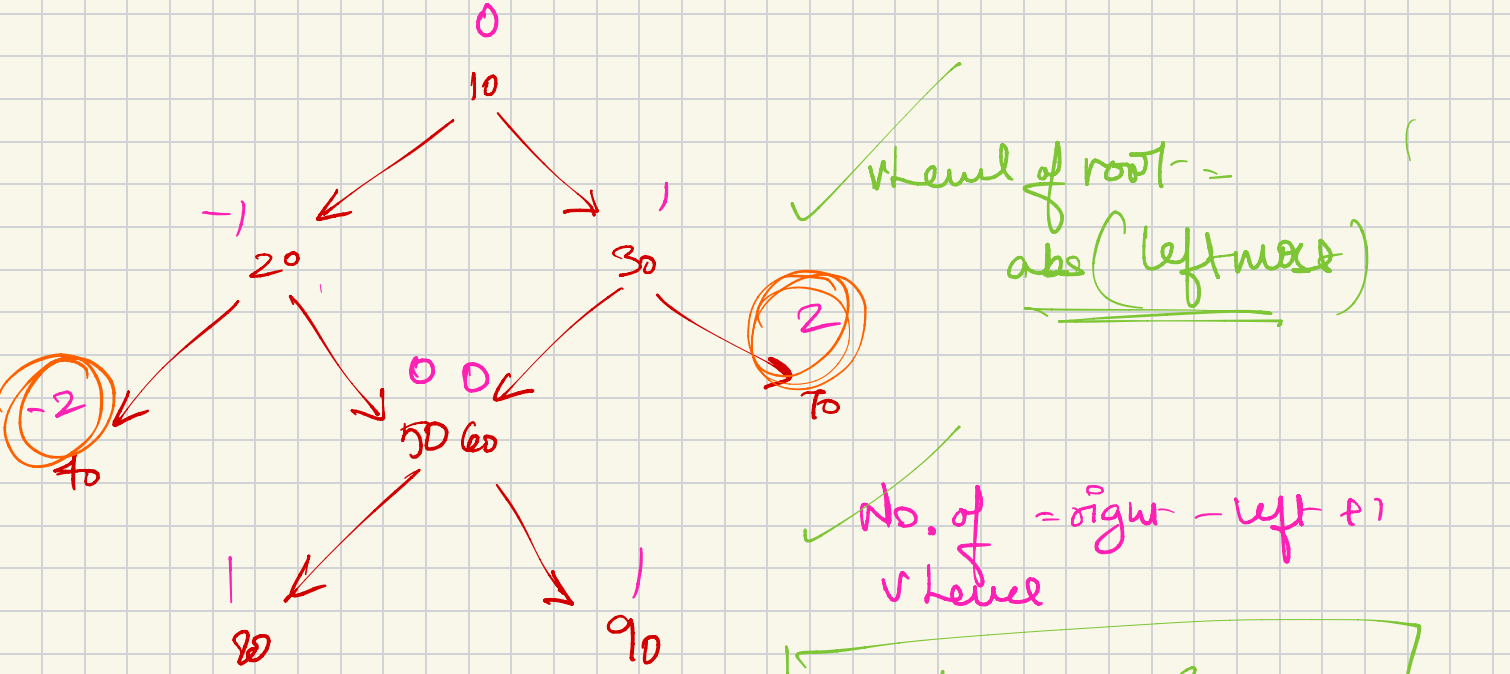
x+1 : 30

x+2

level order traversal

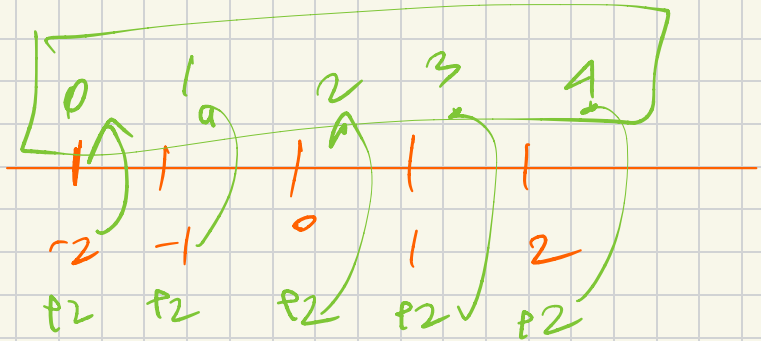
(10, x) (20, x-1) (30, x+1) (40, x-2) (50, x) (60, x) (70, x+2)

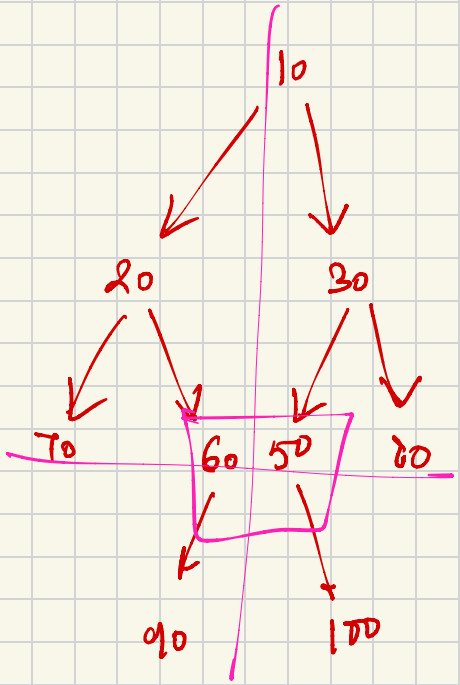
pair



✓ Level of root =
abs(leftmost)

✓ No. of = right - left + 1
✓ Leaves





70

20 90

10 50 60

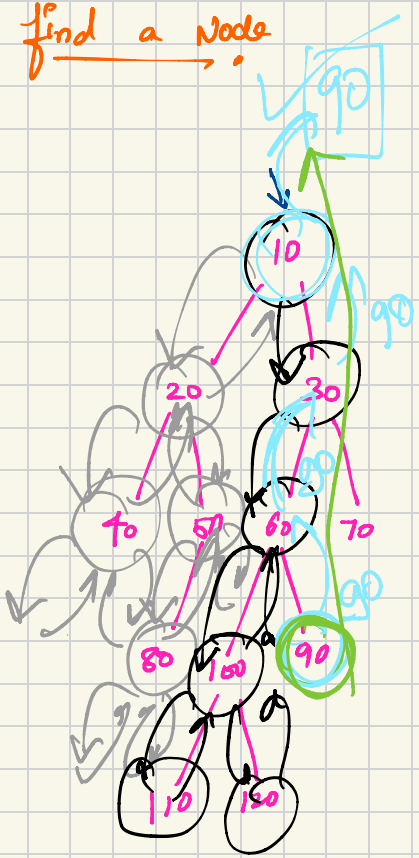
20 100

80

(70,0) (60,2) (50,2) (80,3)

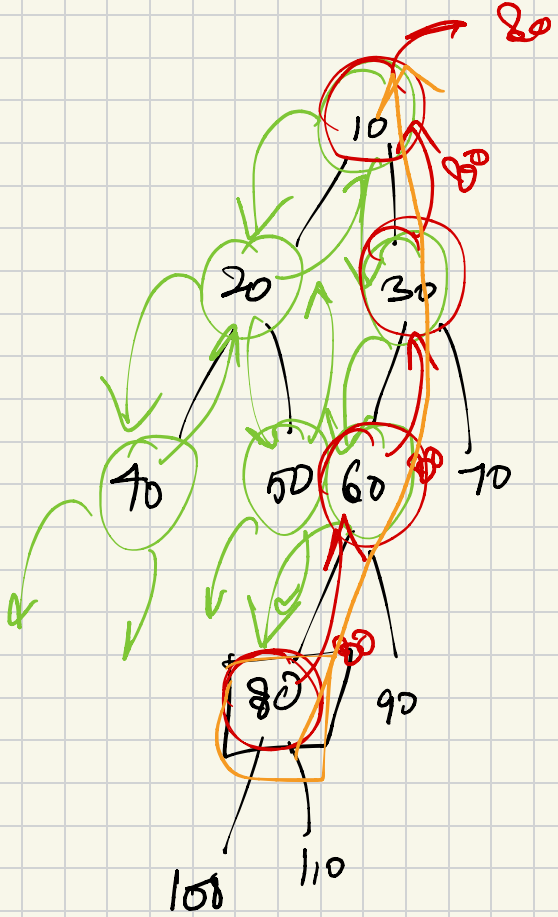
find a Node → !

{ add of the Node with value target }



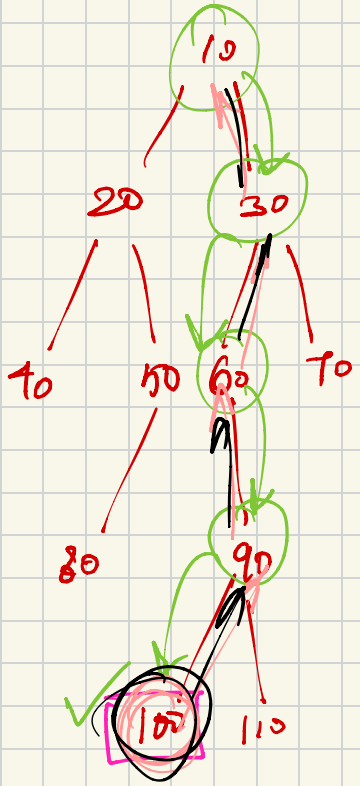
faith! Returns add of target Node if found.

```
Node find (Node root, int target)
(1) { if (root == null) return null;
(2)   if (root.data == target)
          return root;
(3) Node filc = find (root.left, target);
    if (filc != null)
        return filc;
(4) Node firc = find (root.right, target);
    if (firc != null) return firc;
  }
  return null;
```



traced back to root

Path to a given Node



Paths {10 → 30 → 60 → 90 → 100}

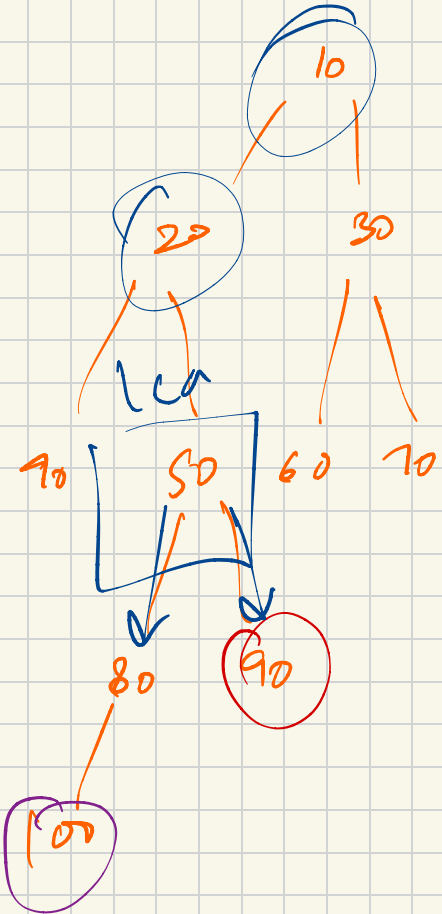
root to node path

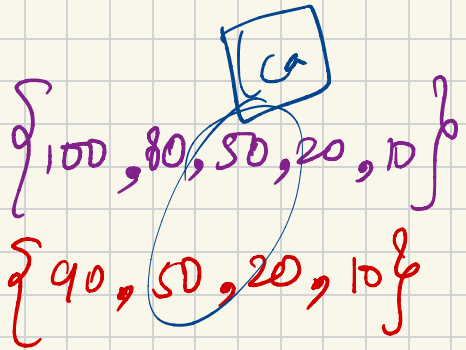
find

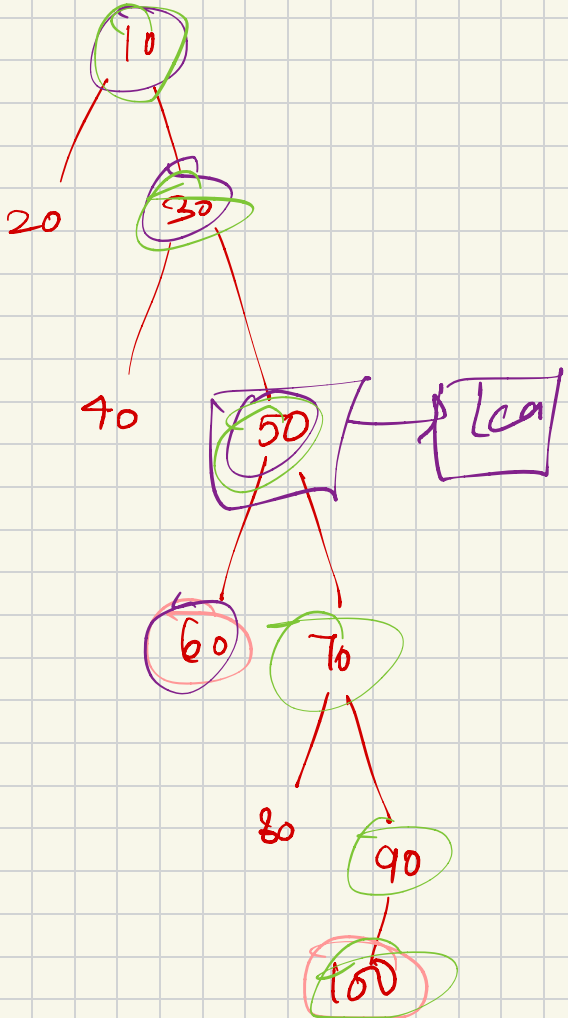
{100, 90, 60, 30, 10}

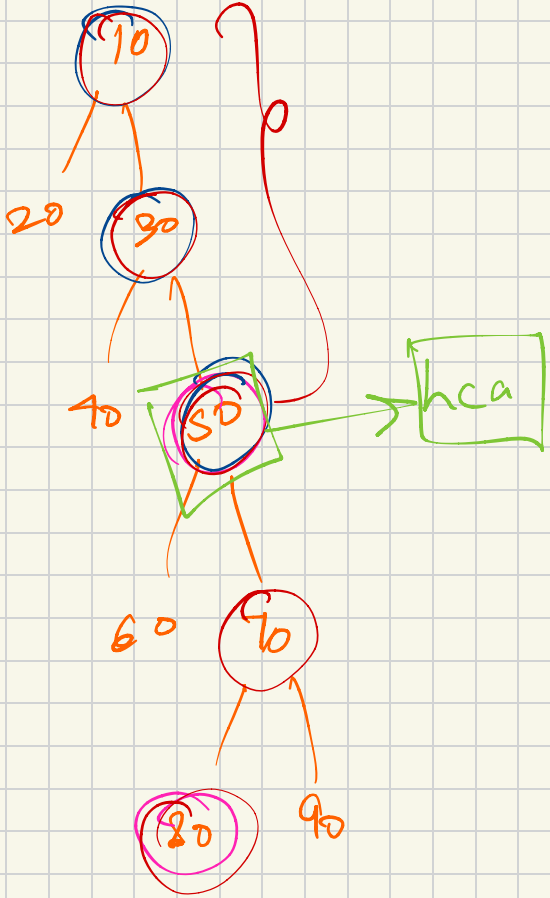
reverse

Lowest Common Ancestor

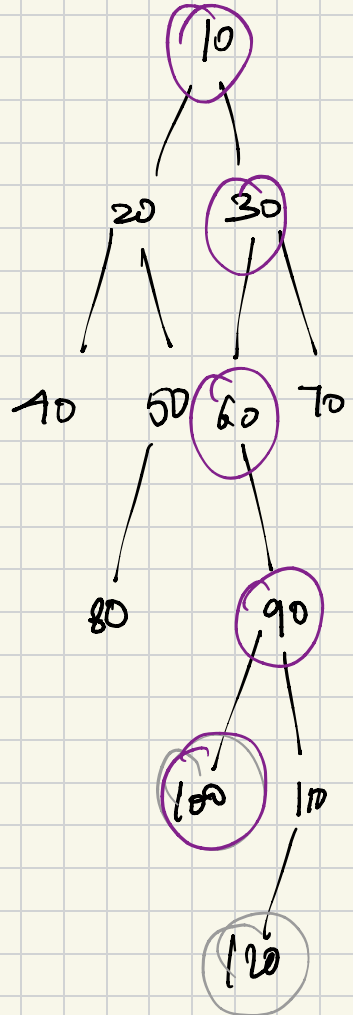





10
20
30
40
50
lca
60
70
80
90
100



Lowest Common Ancestor



[100, 120] → LCA

N2R path ancestors

100, 90, 60, 30, 10

120, 110, 90, 60, 30, 10

LCA = null 10 30 60 90

LCA