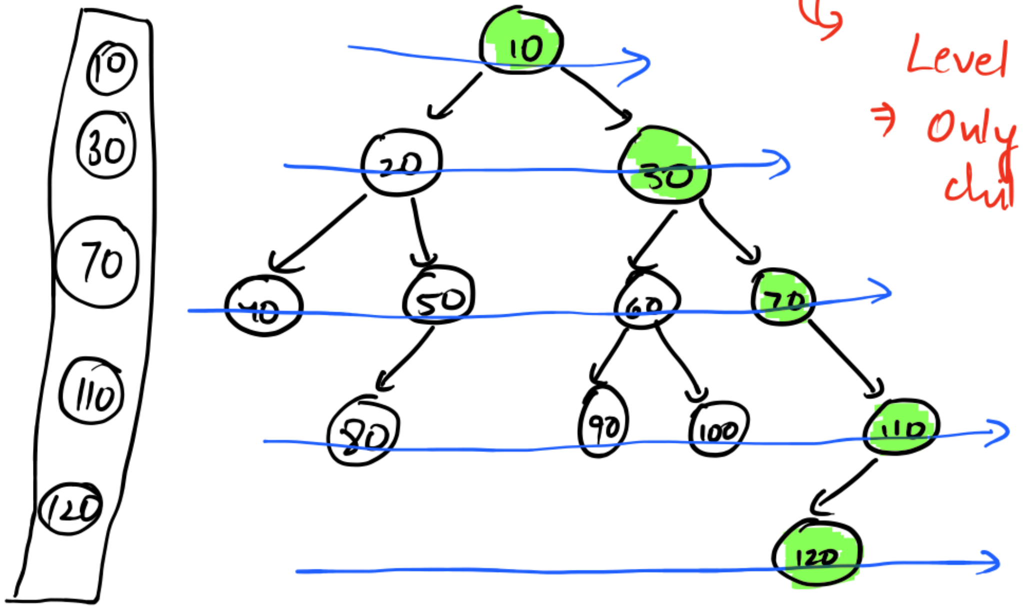
From: **Mihir Sood** mihirsood.mait@gmail.com
Subject:
Date: 3 August 2024 at 2:12 AM
To:

Friday, 2 August 2024
12:04AM

Right View of B.T HW



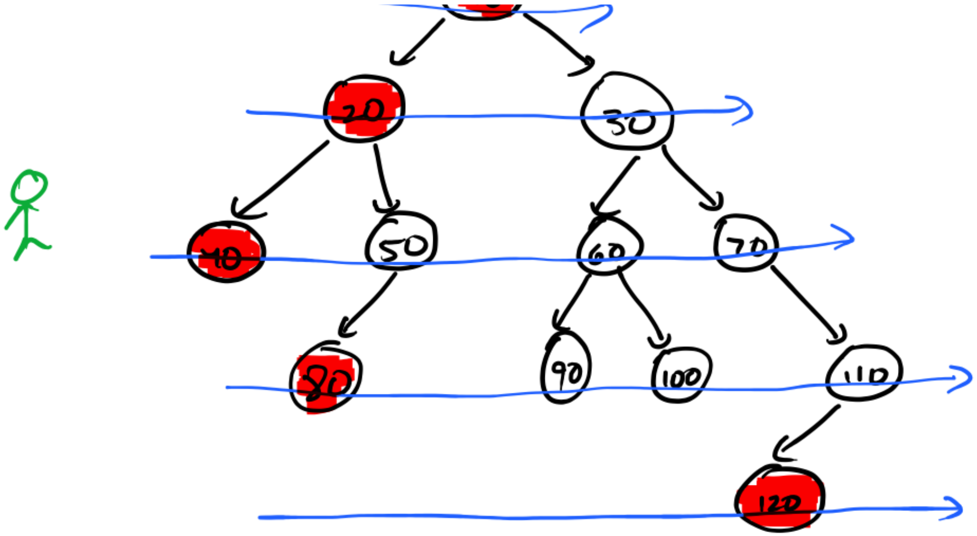
↳ Level Order traversal

⇒ Only add the last child of every level in arr.

O/P
↳ 10, 30, 70, 110, 120

Hint → BFS

Left View of B.T

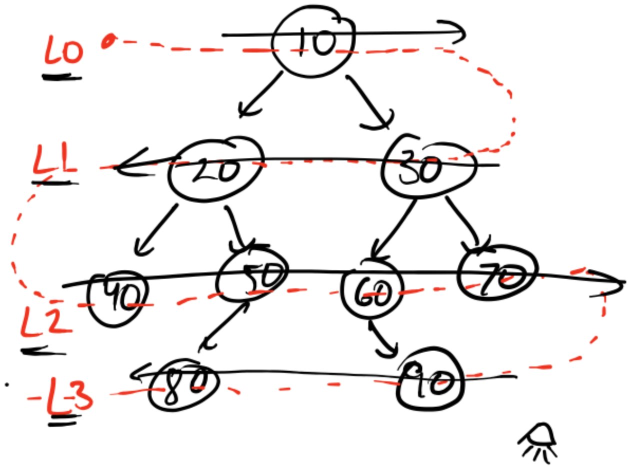


O/I

↳ 10, 20, 40, 80, 120

↳ Hint BFS

Zig-Zag Traversal



10

30,20

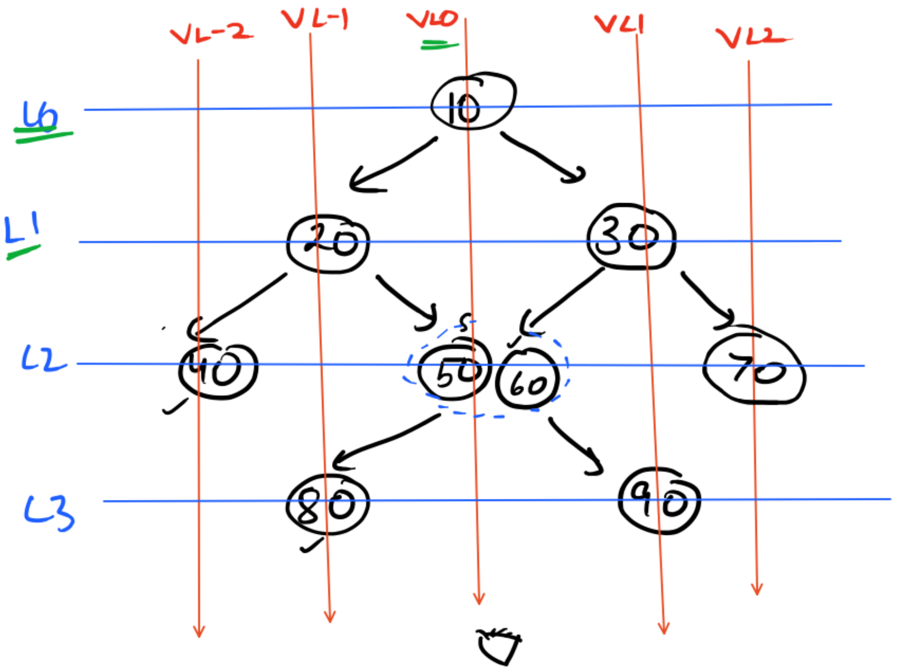
40,50,60,70

90,80

Even Levels:
Left to Right?

Odd Levels:
Right to Left?

Vertical Order Traversal



O/P

40

20, 80

10, 50, 60 → ArrayList
↳ Integer
Pair

30, 90

70

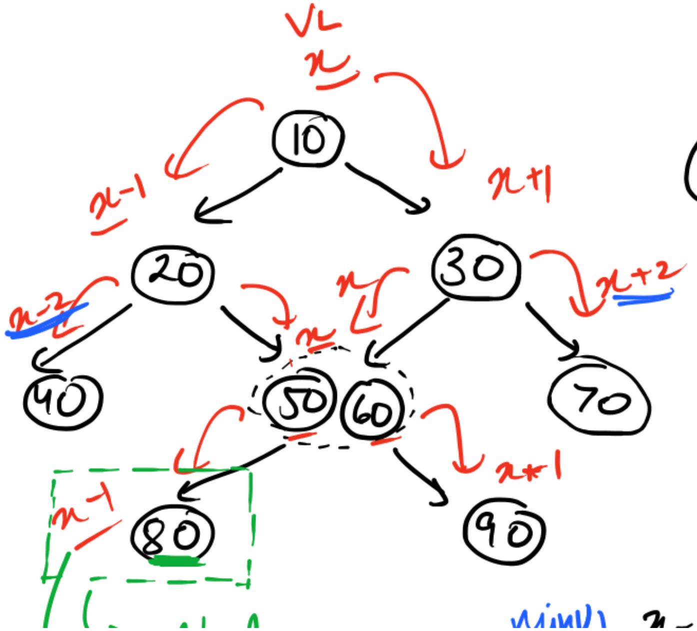
50, 60, 100 ✗

100, 50, 60

1

↳ MinVL

↳ MaxVL



~~(10, x)~~ ~~(20, x-1)~~ (30, x+1) (40, x-1-1)

(50, x-1+1)

↳ BFS Pair of (Node, VL)

Store in HashMap< >

x → 10 → ↳ VL vs list []

x-1 → 20

O/P

MinVL x-2 → 40

→ Node

↳ VL for the given node

$x-2 \Rightarrow 40$

$x-1 \Rightarrow 20, 80$

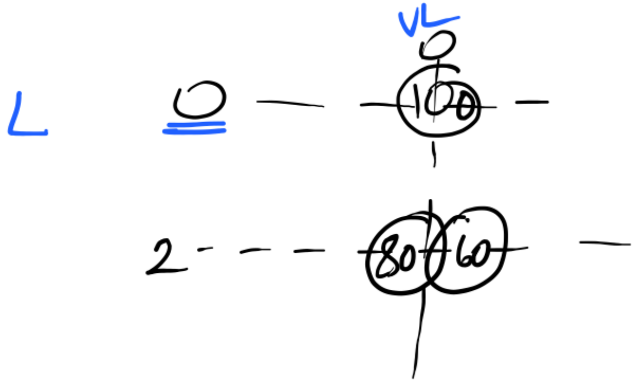
$x \Rightarrow 10, 50, 60$

$x+1 \Rightarrow 30, 90$

$x+2 \Rightarrow 70$

Max VL

} Vertical Order Traversal



→ 100, 60, 80

ArrayList ⇒ { 100, 80, 60 }

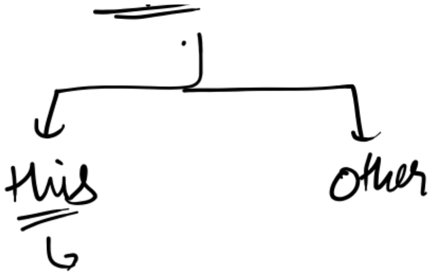
Sort ⇒ choose the one with smaller L

↳ if L same

↳ VL will be same

↳ choose smaller value.

compareTo ? ⇒ Compares a particular object with



another object of same class

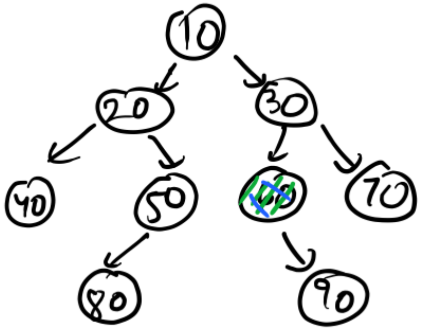
asc order
this - other

desc order
other - this

⇒ { Top View → First person in Vertical Order Traversal

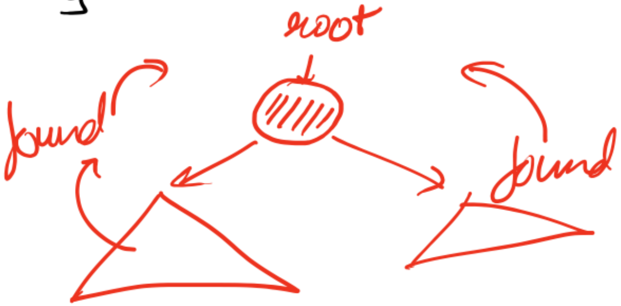
Bottom View → Last person in Vertical Order Traversal }

Find a given in a B.T

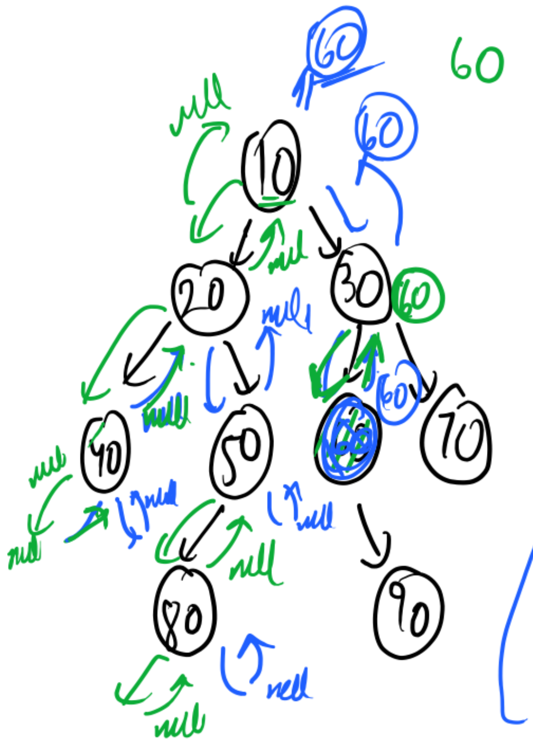


fⁿ returns address of Node with data = val

```
Node find (Node root, int val){
}
```



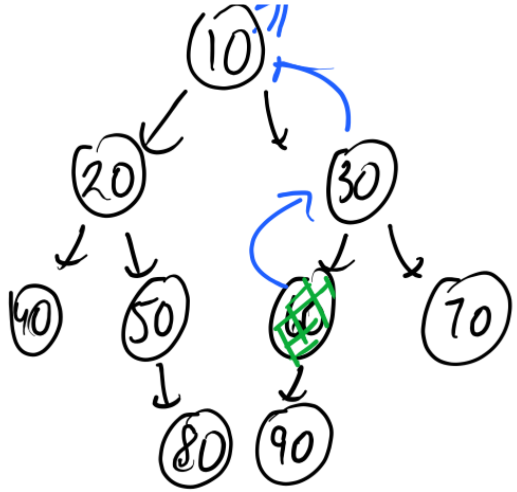
→ check root
→ check LST
→ check RST
→ If not found return null



```
Node find (Node root, int val){
    if(root ==null) return null;

        if(root.data == val){
            return root;
        }
        Node LST = find (root.left);
        if(LST != NULL) return LST;
        Node RST = find(root.right);
        return RST;
}
```

Node to Root Path



{60, 30, 10}

⇓

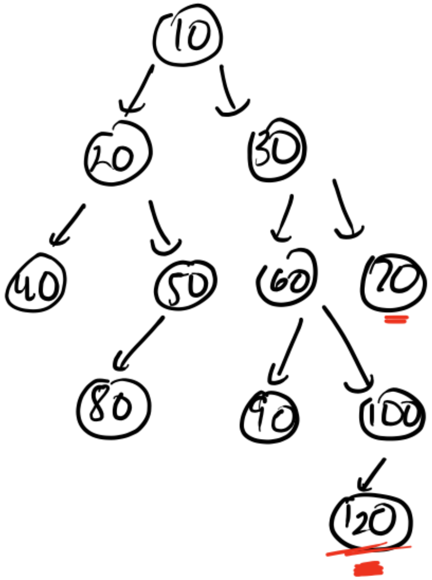
Reverse

⇓

{10, 30, 60}

Lowest Common Ancestor

Last common to Occur in Root to Node Path for Both



LCA(70, 120) ⇒ 30

LCA (90, 100) = 60

LCA (50, 100) = 10

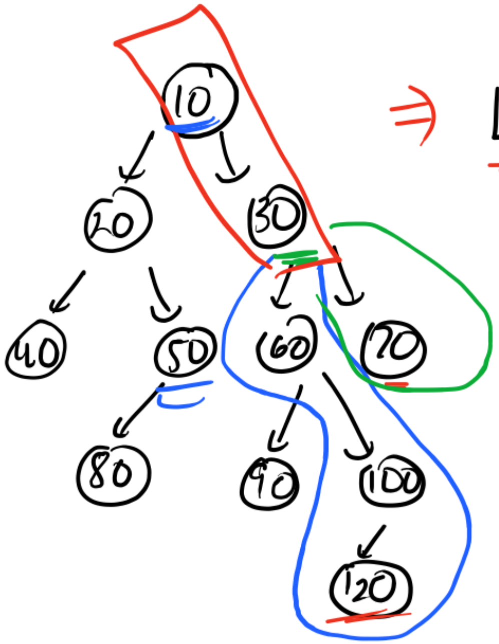
LCA (10, 60) = 10

Question 70, 120 ⇒ Approach.

70 → {10, 30, 70}

120 → {10, 30, 60, 100, 120}

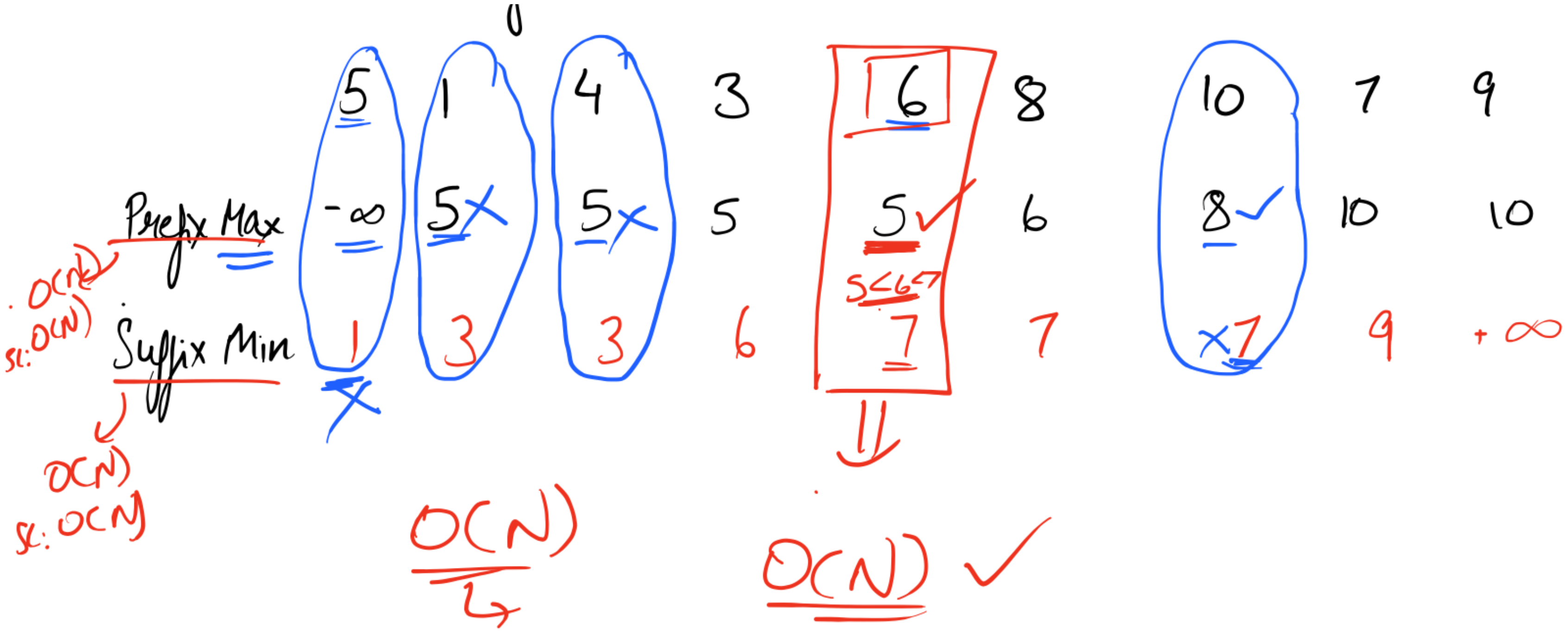
Last common Node in Root to Node path.



⇒ LCA is the intersecting pt

Meaning one node is present on the left side and one node is present on the right

— Find Hinged Element



	5	1	4	3	6	8	10	7	9
Prefix Max	-∞	5 ✗	5 ✗	5	5 ✓	6	8 ✓	10	10
Suffix Min	1	3	3	6	7	7	✗ 7	9	+∞