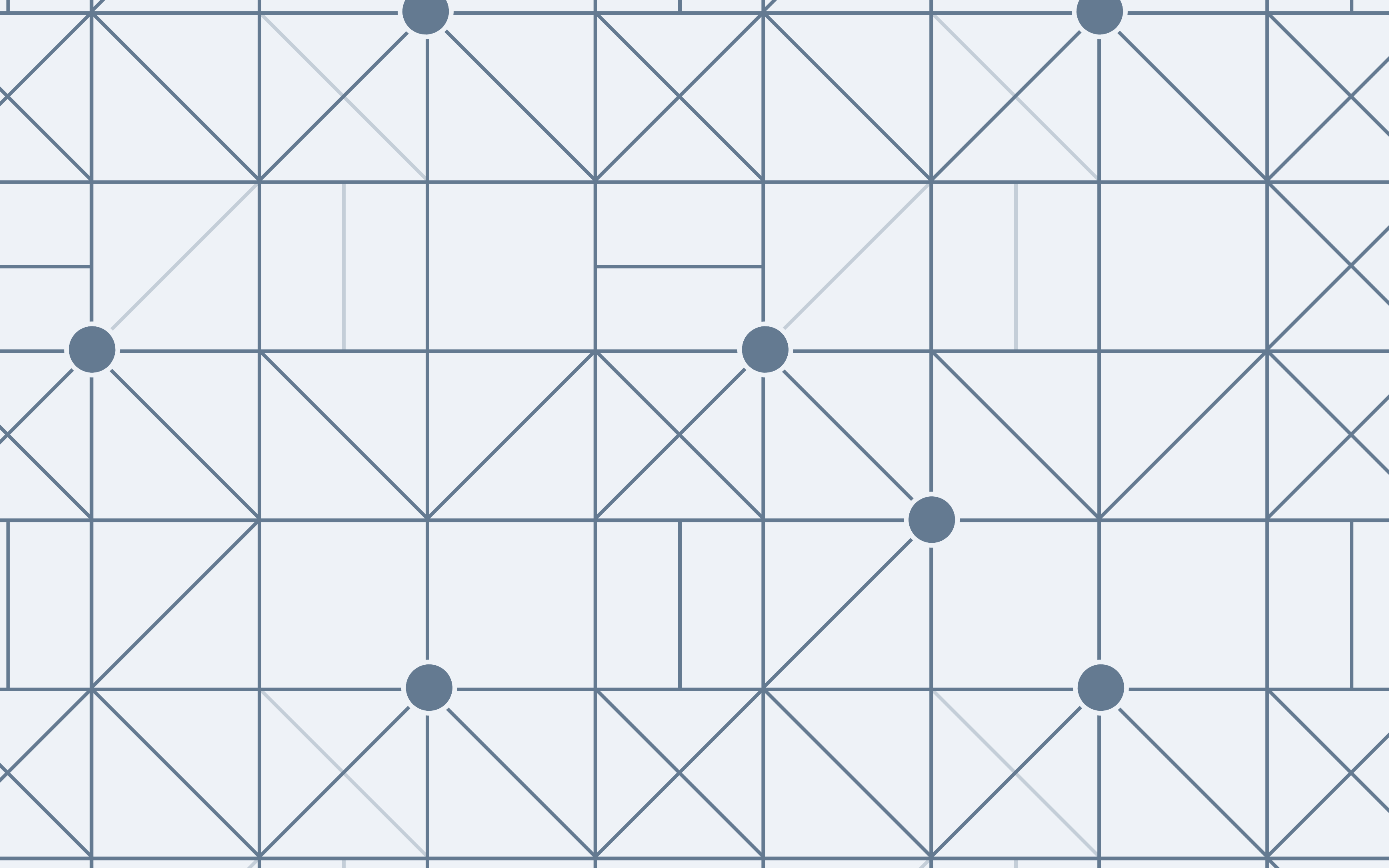
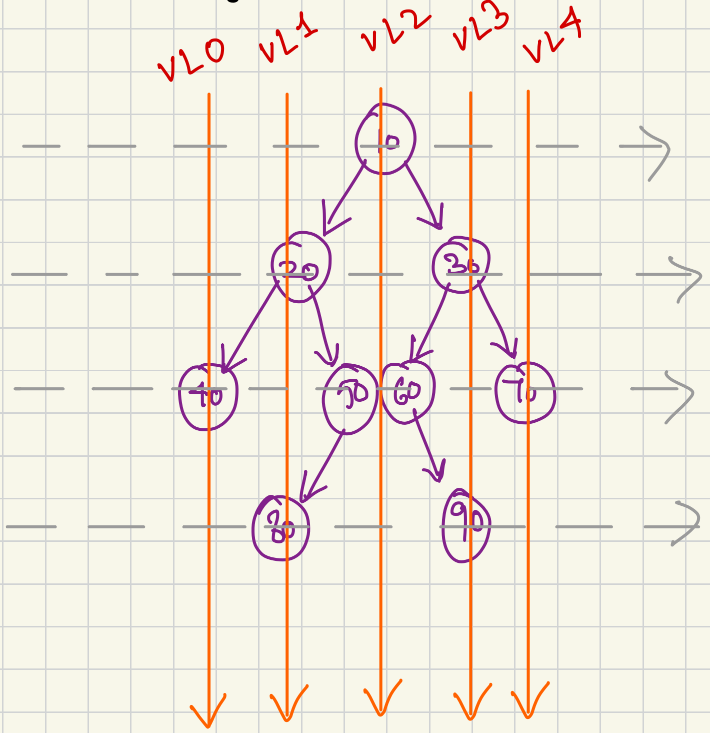
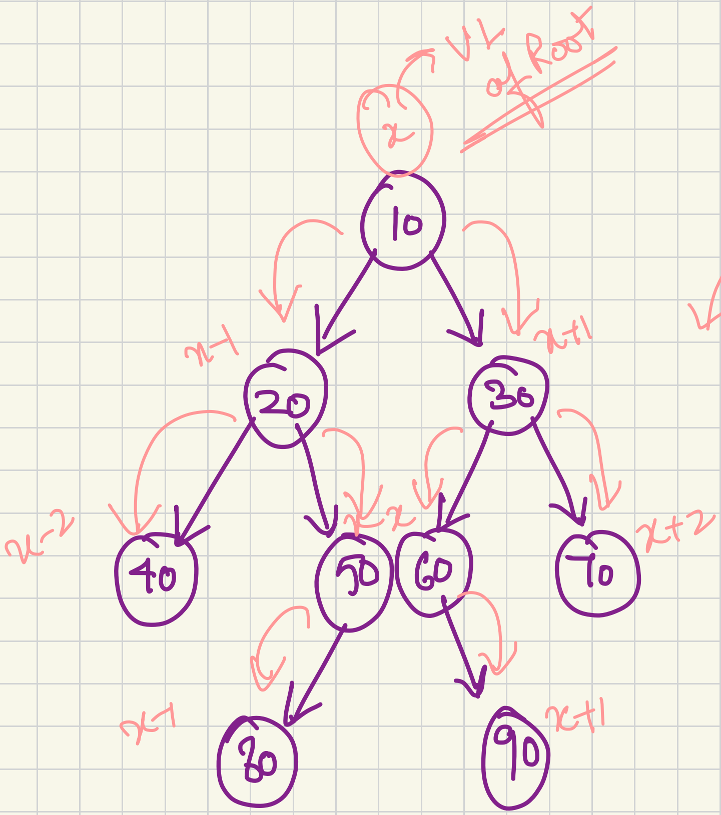


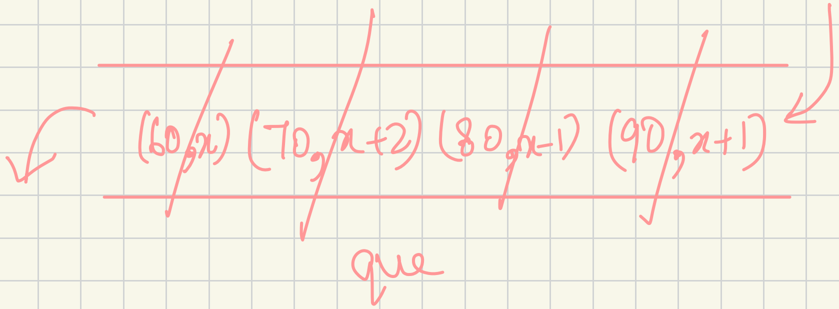
Vertical Order Traversal.



o/p

40
20 80
10 50 60
30 90
70



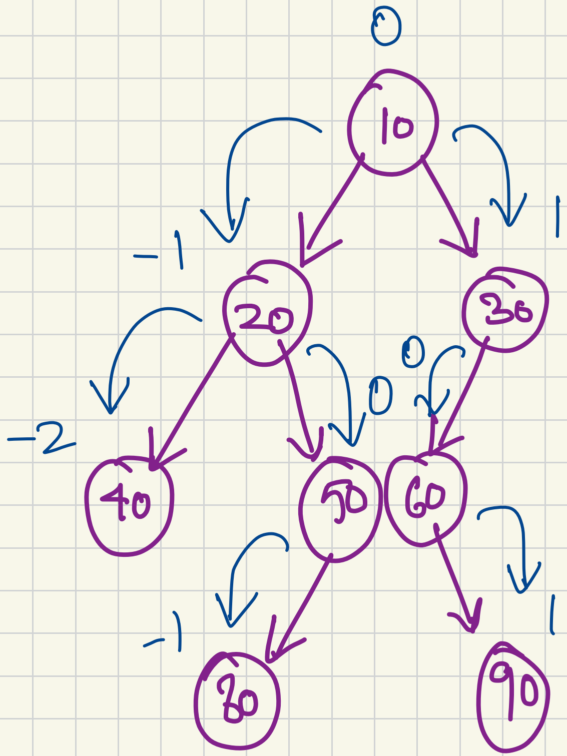


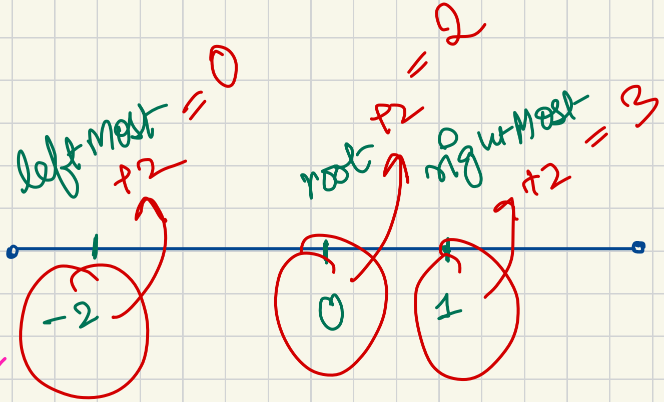
O/P

x-2	→	40
x-1	→	20, 80
x	→	10, 50, 60
x+1	→	30, 90
x+2	→	70

} vertical order traversal

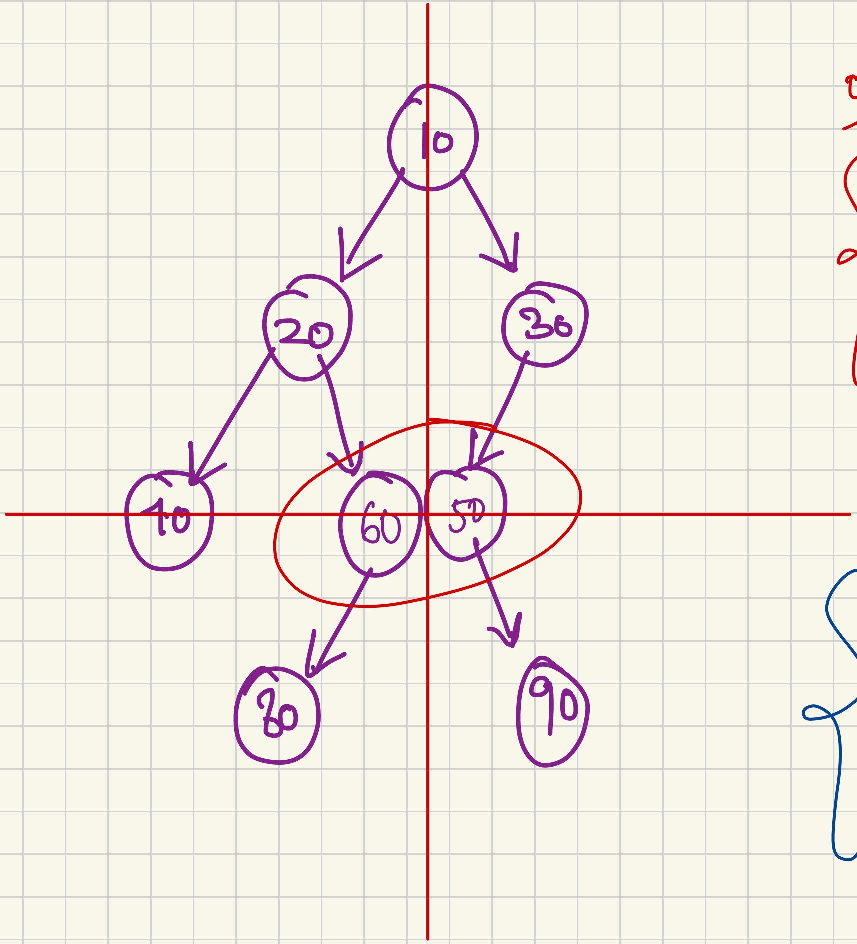
1. Vertical Level of root.
2. No. of vertical Levels





No. of VLevels = right - left + 1
= 1 - (-2) + 1
= 4

Vertical Level of root = abs(left)



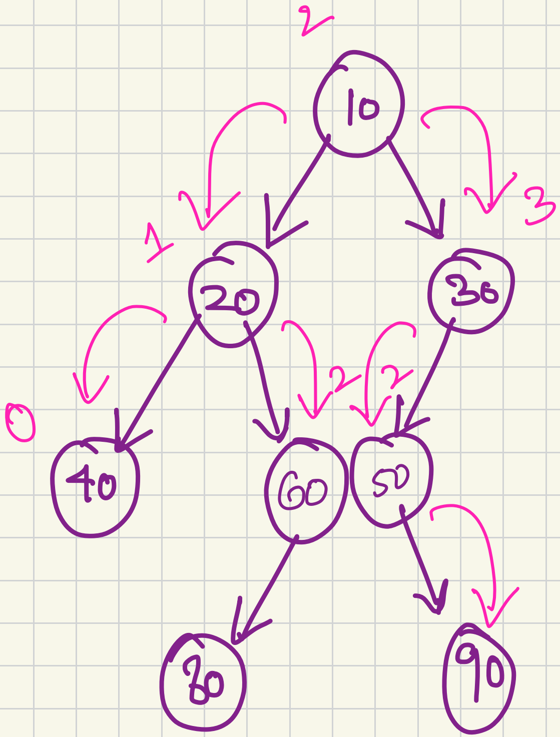
o/p

40
20 80
10 50 60
30 90

✓

Custom logic {People of same level}

- remove left most Person
- if same v Level - then remove smallest data among them



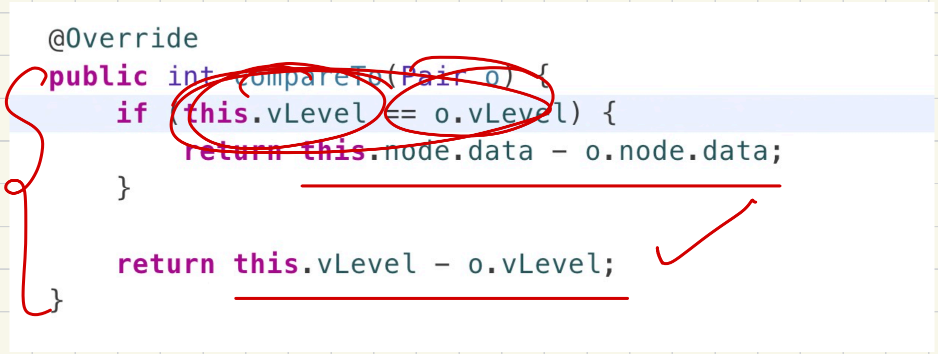
mQue

~~(90, 3)~~ ~~(80, 1)~~

CQue

0 ⟶ {40}
1 ⟶ {20, 80}
2 ⟶ {10, 50, 60}
3 ⟶ {30, 90}

✓



```java
@Override
public int compareTo(Pair o) {
    if (this.vLevel == o.vLevel) {
        return this.node.data - o.node.data;
    }

    return this.vLevel - o.vLevel;
}
```

By default Behaviour of Sorting

asc. order

$\{ \underline{\qquad\qquad\qquad} \} \rightarrow$ (this, other)

✓ this − other → asc

other − this → des

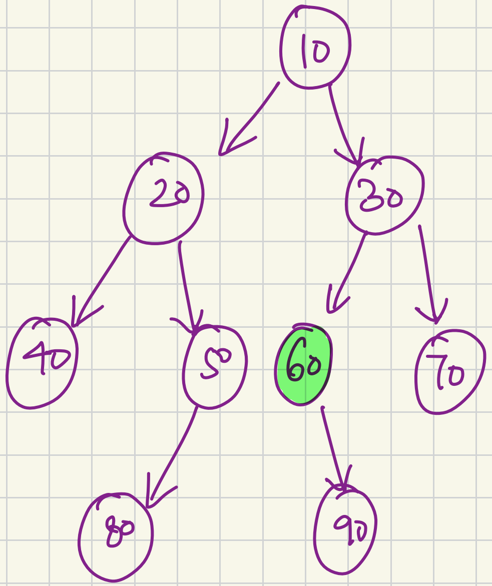
Bottom View

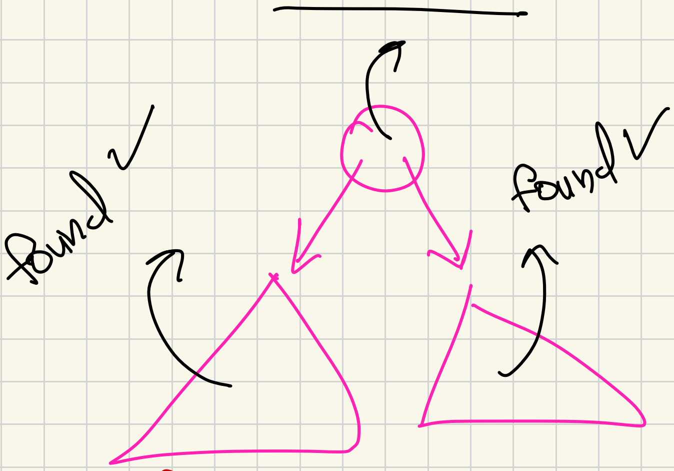
↳ Last Person of Each V Level

Top View

↳ first Person of Each V Level

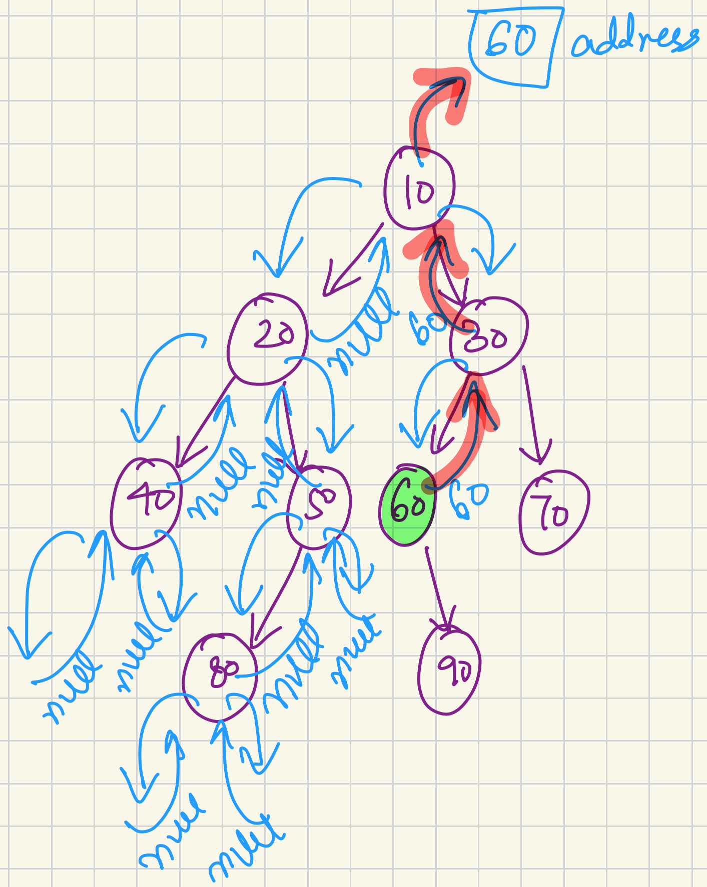
Find a given Node in a BT





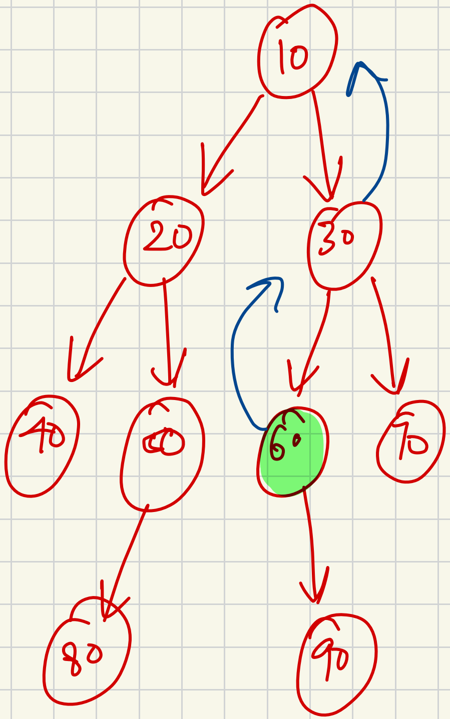
return! returns add target found.

Node find (Node root, int tar)



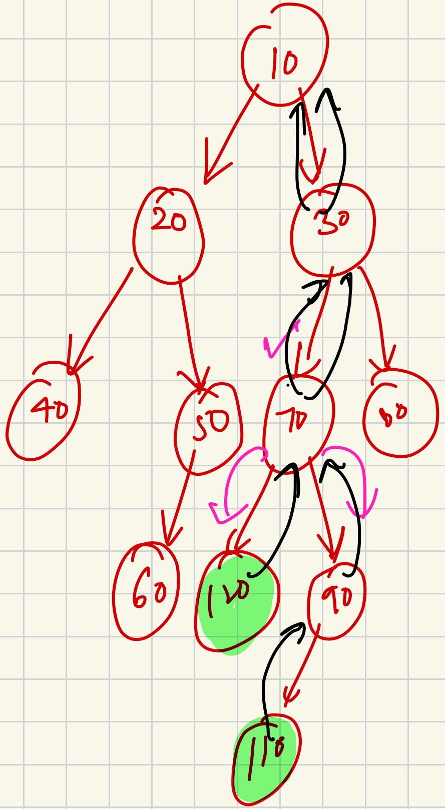
```
Node find( Node root, int tar)
{
    if (root = = null)
        return null;

    if (root.data = = tar)
        return root;

    Node filc = find(root.left, tar)
    if ( filc ! = null)
        return filc;

    Node firc = find(root.right, tar)
    if (firc ! = null)
        return firc;
    return null;
}
```

Node to Root Path



$\{60, 30, 10\}$

Lowest Common Anscestor



120 → 120 70 30 10 → n2R

110 → 110 90 70 30 10 → n2R

LCA = ~~10~~ ~~30~~ $\boxed{70}$ ✓