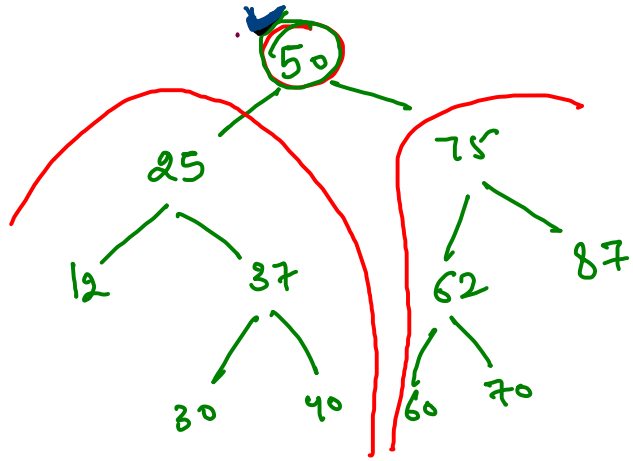
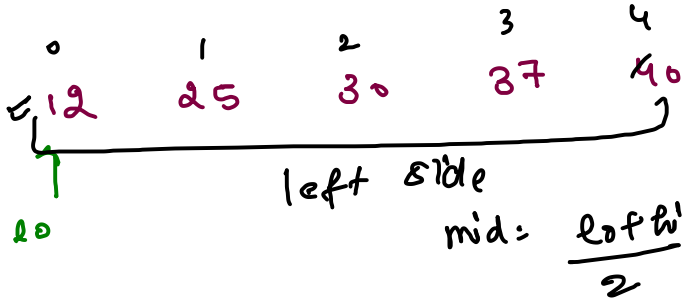
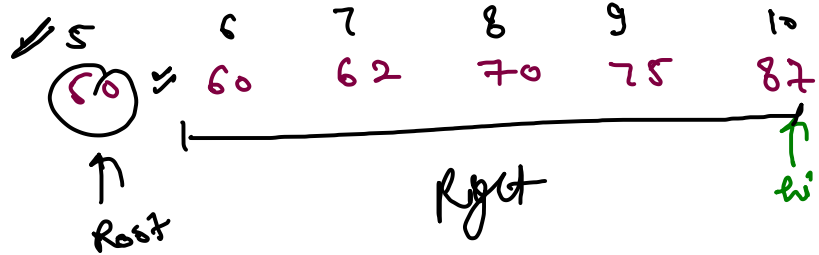


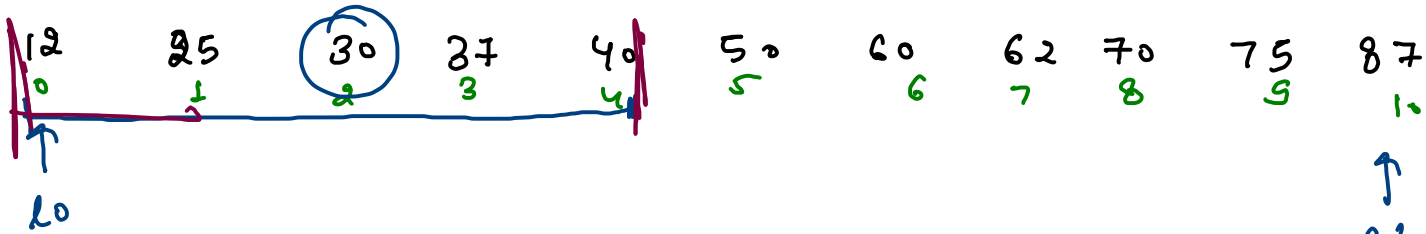
* Max of left subtree is smaller than root data
* Min from Right subtree is greater than root data.

→ Inorder → sorted.

Inorder →

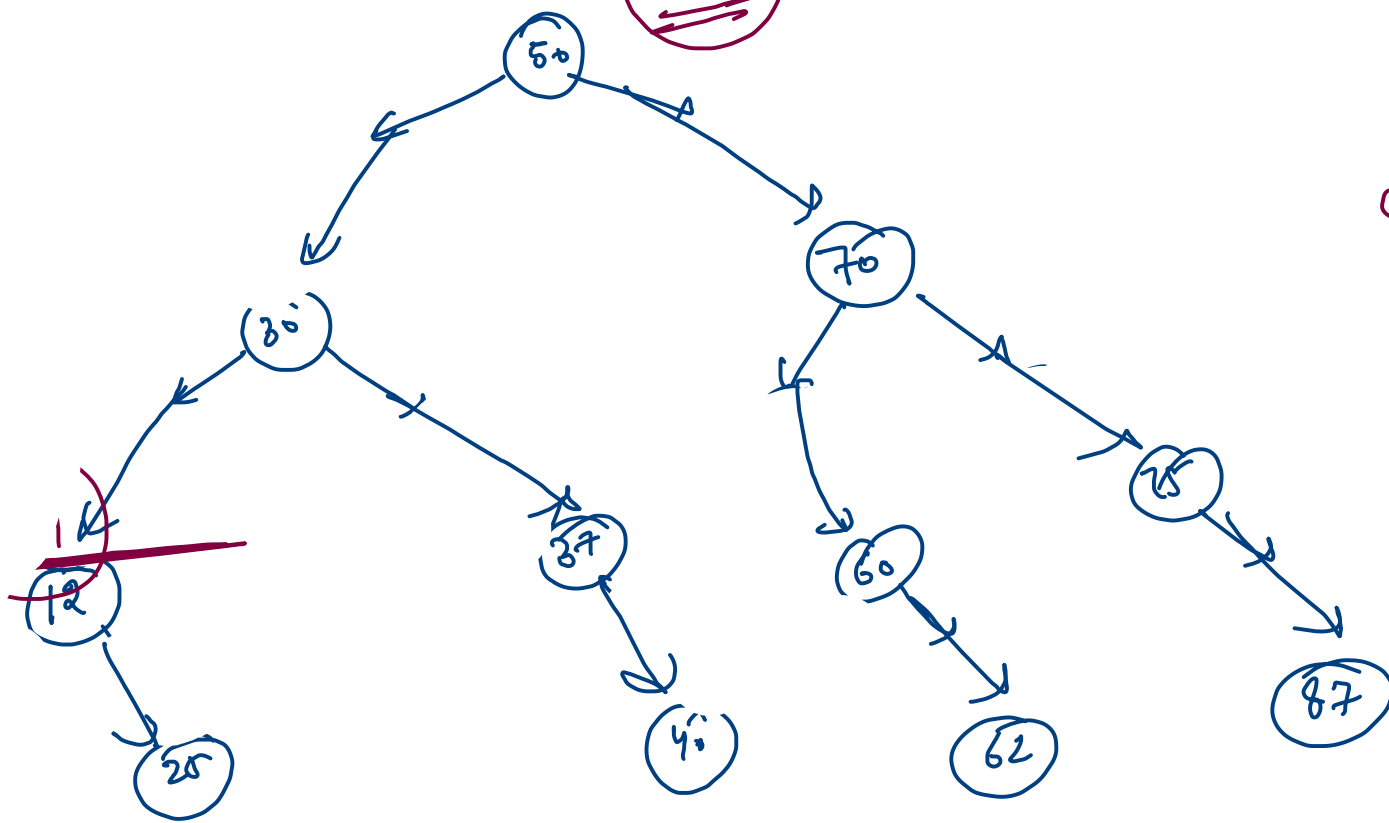






mid = $\frac{lo+hi}{2}$

BST



Node
→ data
→ left
→ Right

Construct(int data, lo, hi)

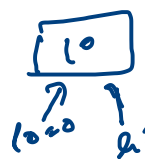
① `int mid = (lo + hi) / 2;`

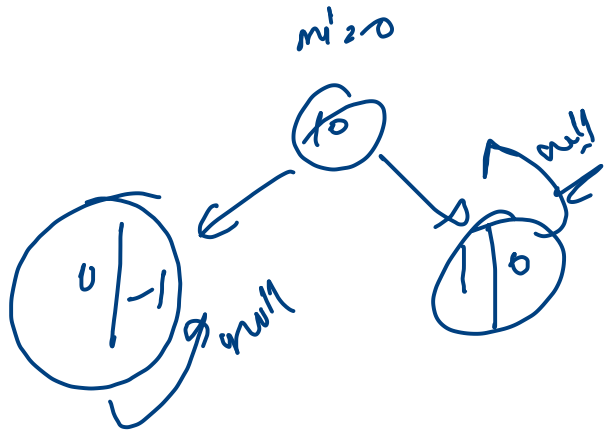
② `int mid = lo + (hi - lo) / 2;`

① mid = (lo + hi) / 2

lo + hi ≥ Integer.MAX_VALUE

(lo + hi) / 2 → valid integer.





② mid = lo + (hi - lo) / 2.

= lo + hi/2 - lo/2.

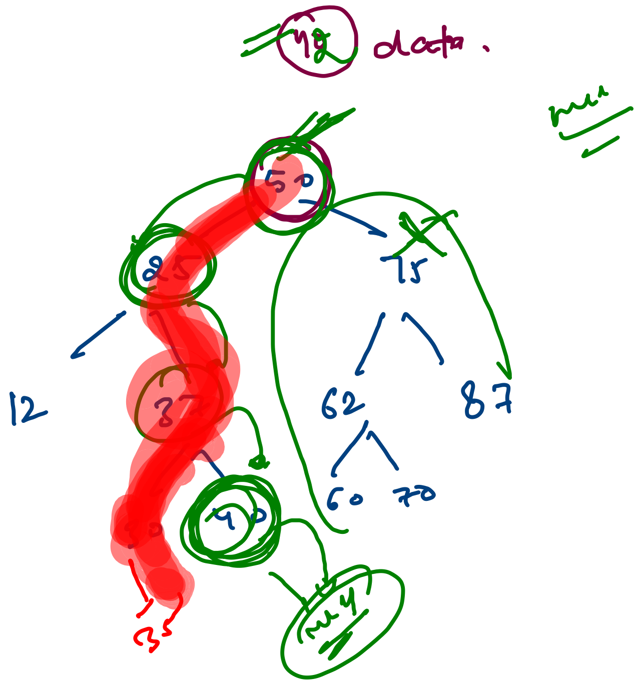
= lo/2 + hi/2.

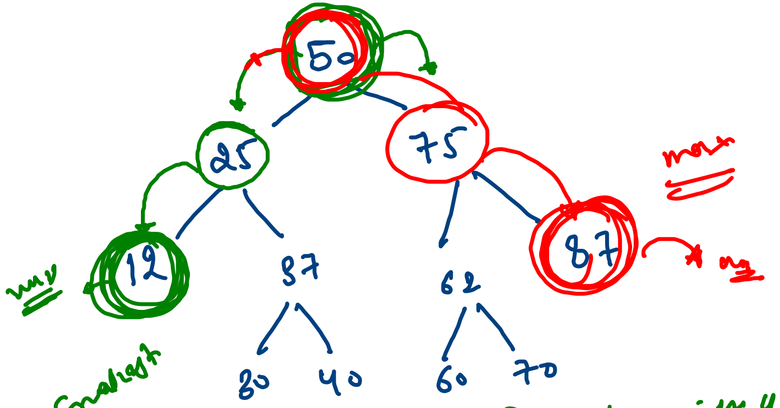
= (lo + hi) / 2

Min

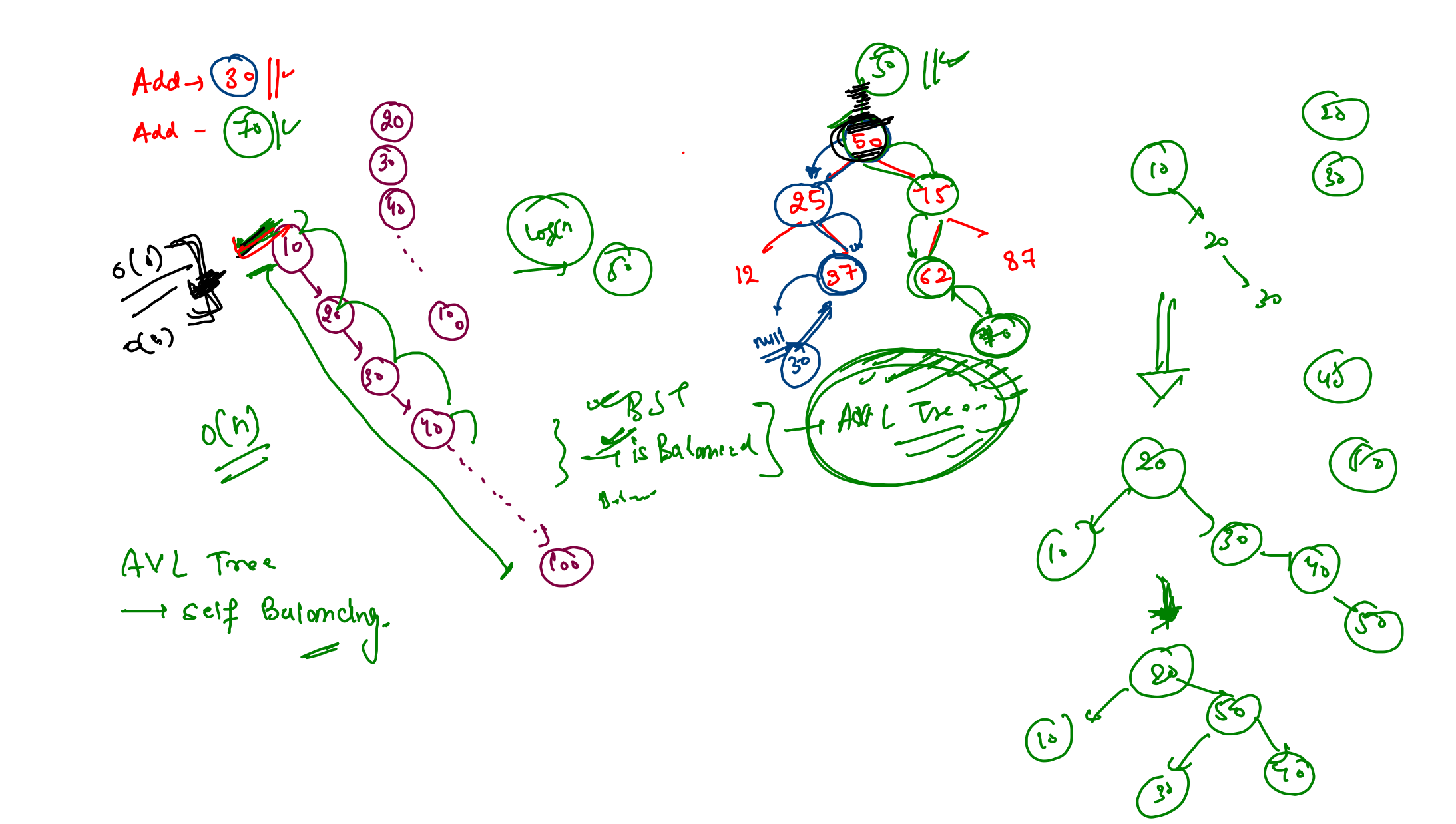
Max

Find.






```
if(root == null) return false
if( root.data == data){
    return True
} else if (root.data > data){
    search in left side.
} else { data > root.data
    search in Right side
}
```


Add → 30
Add - 70
O(1)
O(n)
AVL Tree
→ self Balancing.
BST
is Balanced
AVL Tree
null
12
87

Binary Tree

Height	$h = O(n)$ ★
Find	$O(n)$
Max	$O(n)$
Min	$O(n)$
Sum	$O(n)$ ★
Diameter	$O(n)$ ★

B.S.T.

★ → Structural problem.

$h = O(n)$ ★

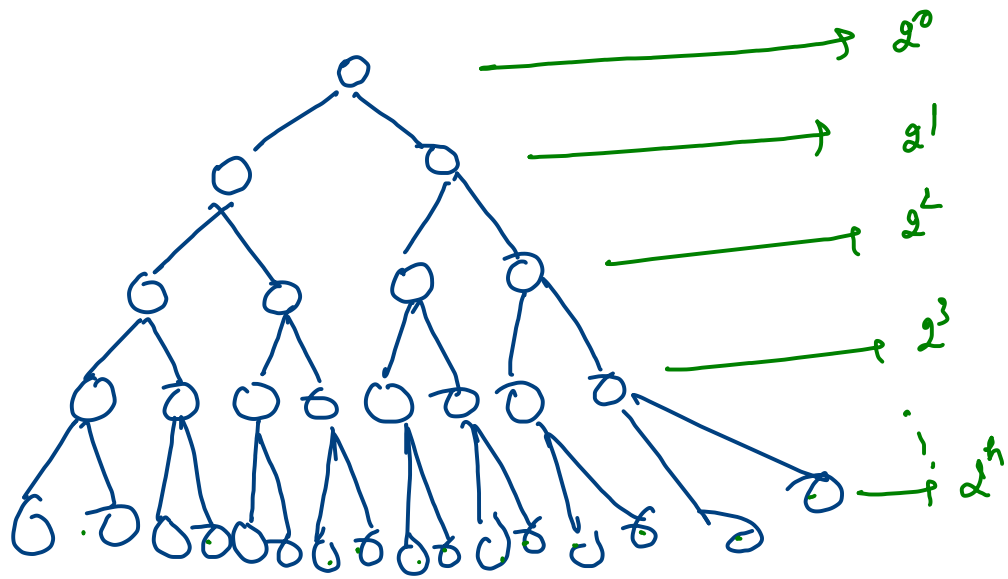
$O(\log n)$ $O(h)$ $h = \log n$.

$O(h) \rightarrow O(\log n)$

$O(h) \rightarrow O(\log n)$

$O(n)$ ★

$O(n)$ ★



$$\frac{a(r^n-1)}{r-1}$$

$a=1$

$r=2$

$n=h$

no. of nodes $= 2^0 + 2^1 + 2^2 + \dots + 2^h$

$n = 2^0 + 2^1 + 2^2 + \dots + 2^h$

$n = \frac{1(2^h-1)}{2-1} = (2^h-1)$

$(n+1) = 2^h$

$\log_2(n+1) = \log_2 2^h$

$\Rightarrow$

$n = 2^h - 1 \quad h = \log_2(n+1)$

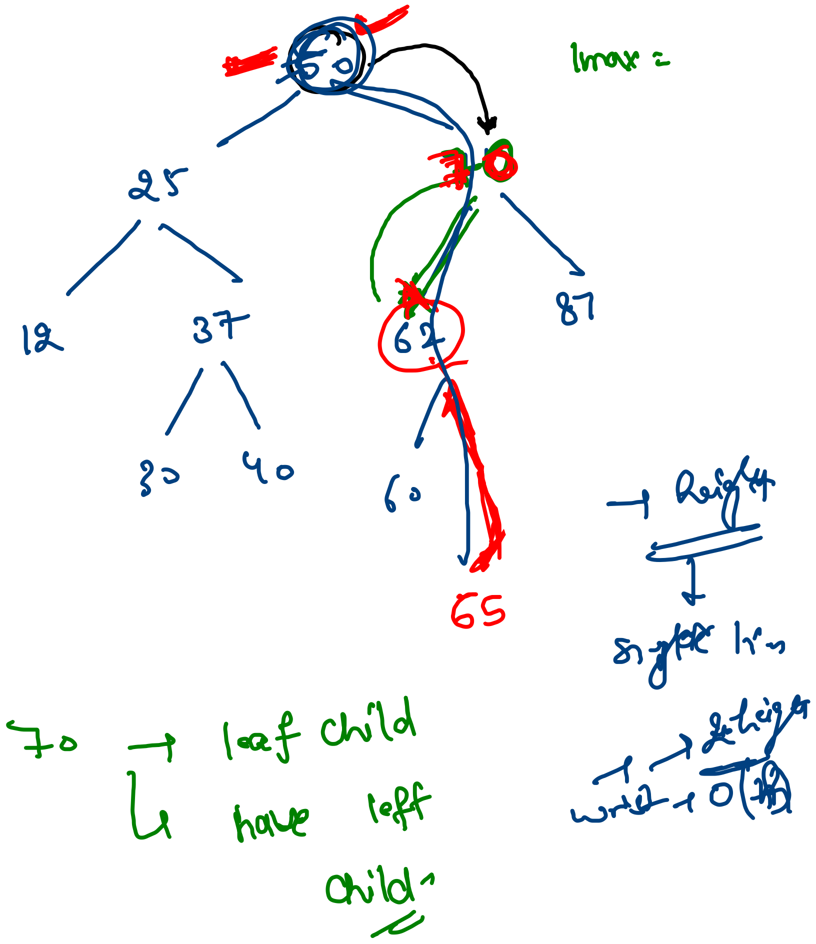
$h = O(\log n)$

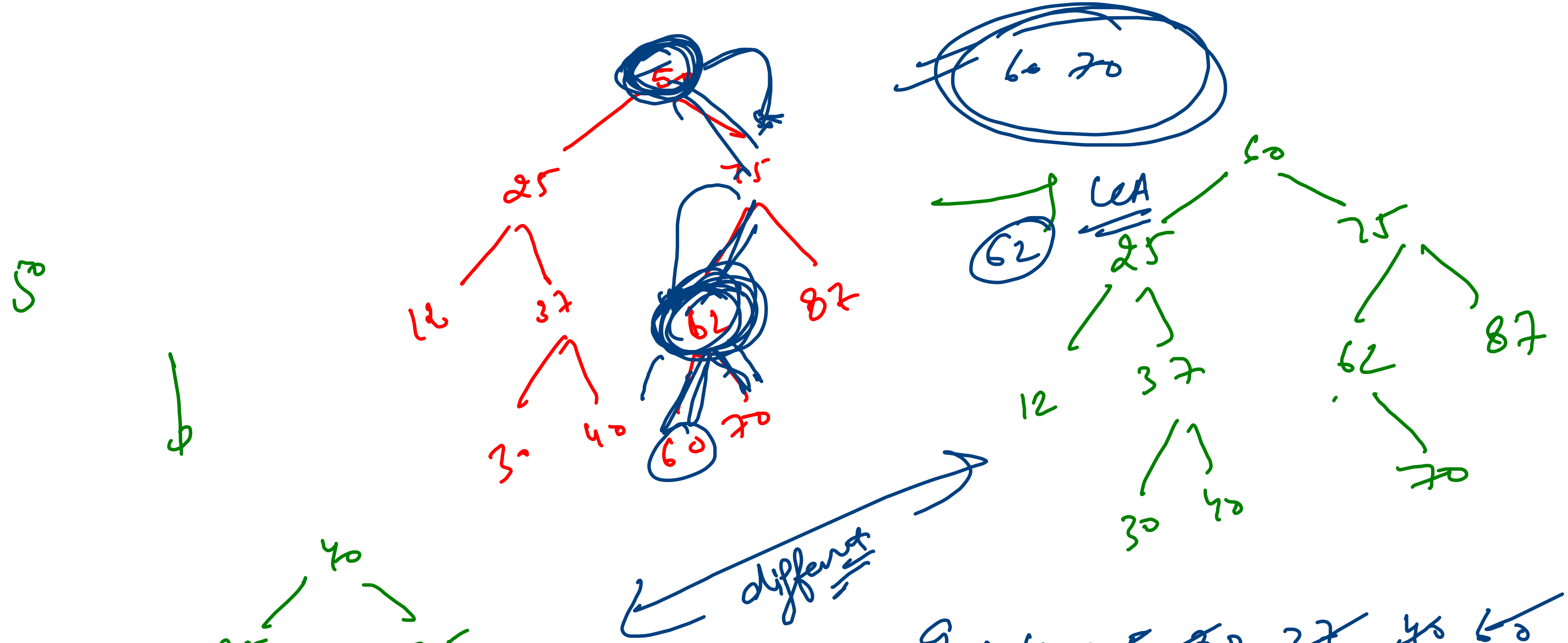
```java
public static Node remove(Node node, int data) {
    // write your code here
    if(node.data == data) {
        if(node.left != null && node.right != null) {
            // set max from left subtree
            int lmax = max(node.left);
            node.data = lmax;
            node.left = remove(node.left, lmax);
            return node;
            // remove max from left subtree
        } else if(node.left != null) {
            return node.left;
        } else if(node.right != null) {
            return node.right;
        } else {
            return null;
        }
    } else if(node.data > data) {
        // removal from left
        node.left = remove(node.left, data);
    } else {
        // removal from right side
        node.right = remove(node.right, data);
    }
    return node;
}
```

Remove : 75

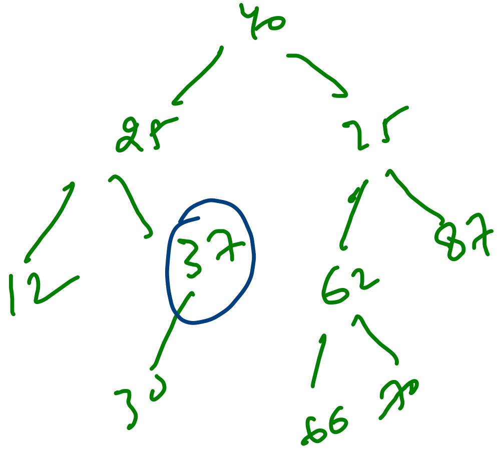
O(n)

O(logn)





Inord 12 25 30 37 40 60 62
70 75 87



hint → calls_split

InOrd - 12 25 30 37 40 66 62 70 75 87