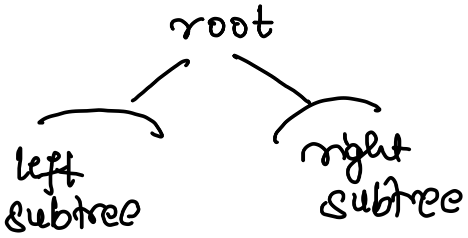
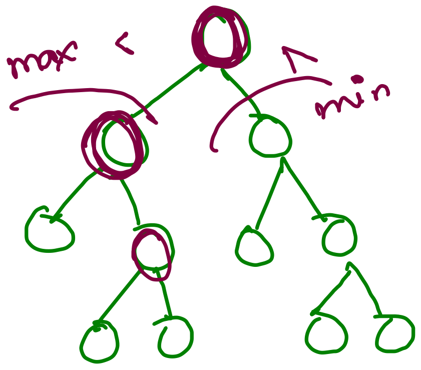


for BST

→ # all nodes in left subtree is smaller than root.data

→ # all nodes in right subtree is greater than root.data

→ these are valid for all node in BST.



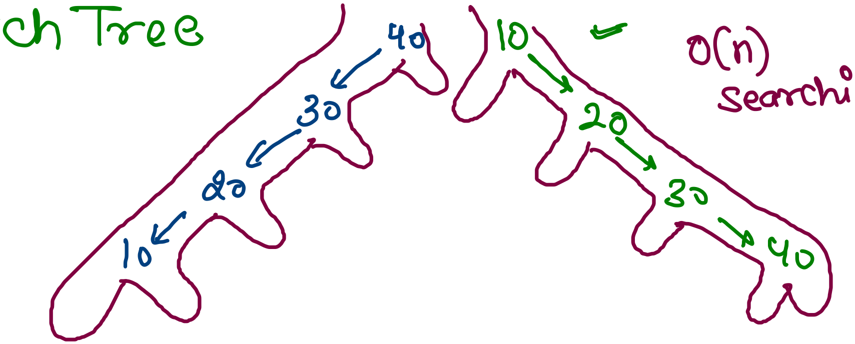
Benefits. → * searching. ] Based on searching.

* store

* value based problem.

BST → Binary Search Tree

AVL → Balanced BST



O(n) searching

Construction.

Inorder → Sorted

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 12 | 25 | 30 | 37 | 50 | 62 | 70 | 75 | 87 |
| ↑<br>lo | | | | | | | | ↑<br>hi |

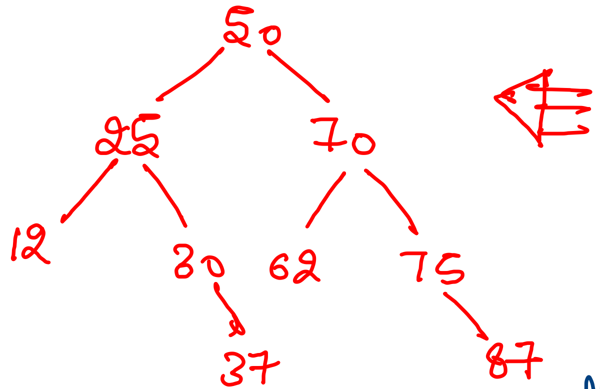
# Construction.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 12 | 25 | 30 | 37 | 50 | 62 | 70 | 75 | 87 |
| lo | | | | | | | | hi |

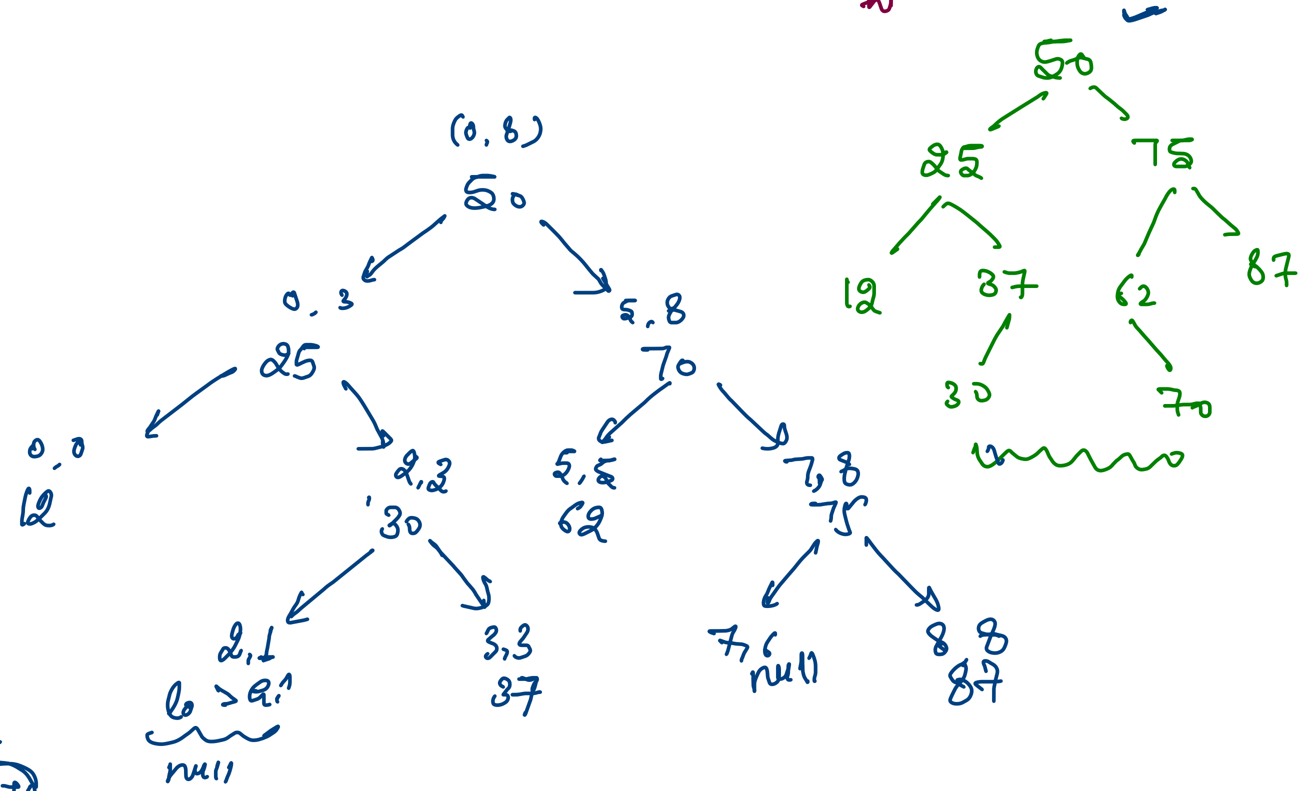
lo = 0

hi = 8

$$mid = \frac{lo + hi}{2} = \frac{0+8}{2} = \boxed{4}$$



Balance || BST

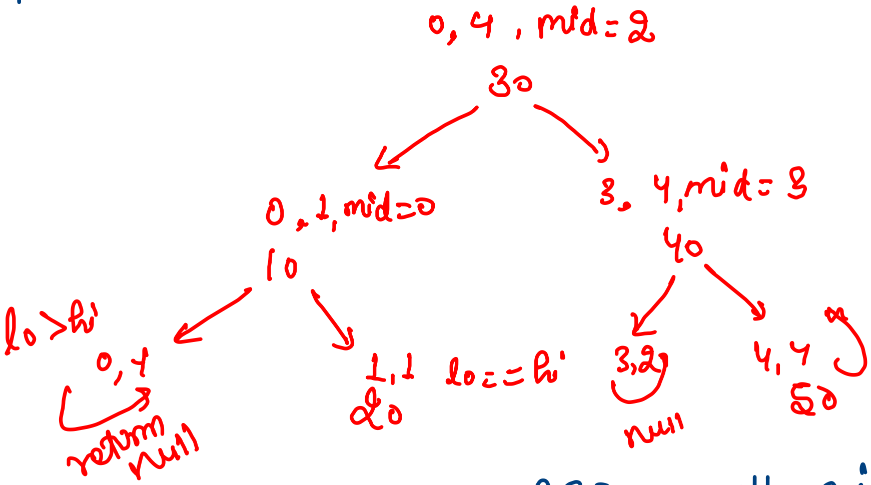


In Order → 12 25 30 37 50 62 70 75 87

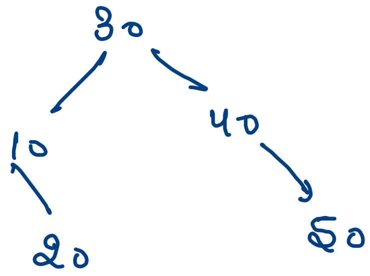
data →

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 |

Construction →



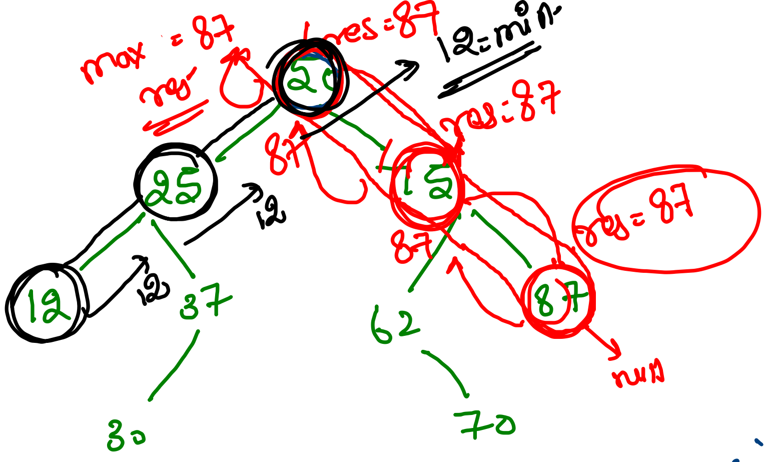
=



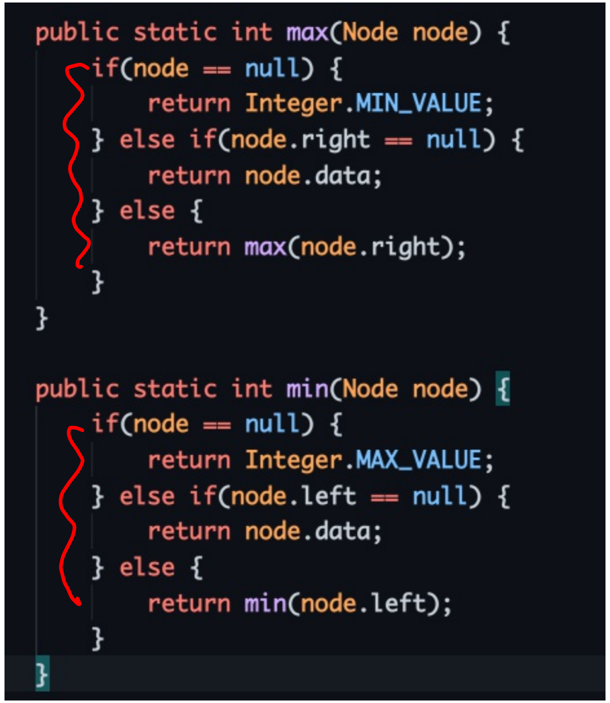
Same in BST as well as in Binary Tree

Structured based → → Size
→ height
→ Diameter
→ Sum

Value based problem → min, max, find ]



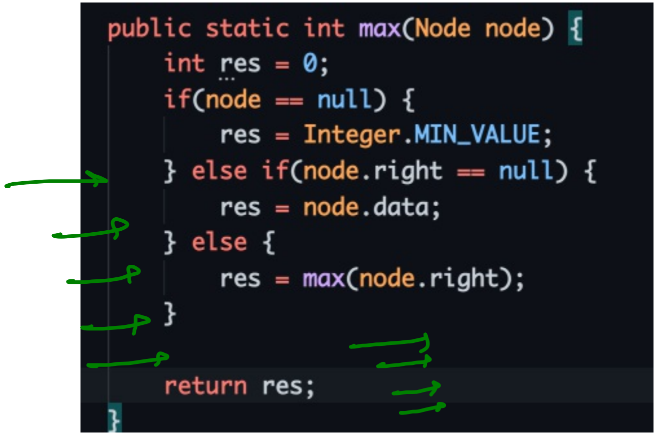
guided traversal



```java
public static int max(Node node) {
    if(node == null) {
        return Integer.MIN_VALUE;
    } else if(node.right == null) {
        return node.data;
    } else {
        return max(node.right);
    }
}

public static int min(Node node) {
    if(node == null) {
        return Integer.MAX_VALUE;
    } else if(node.left == null) {
        return node.data;
    } else {
        return min(node.left);
    }
}
```

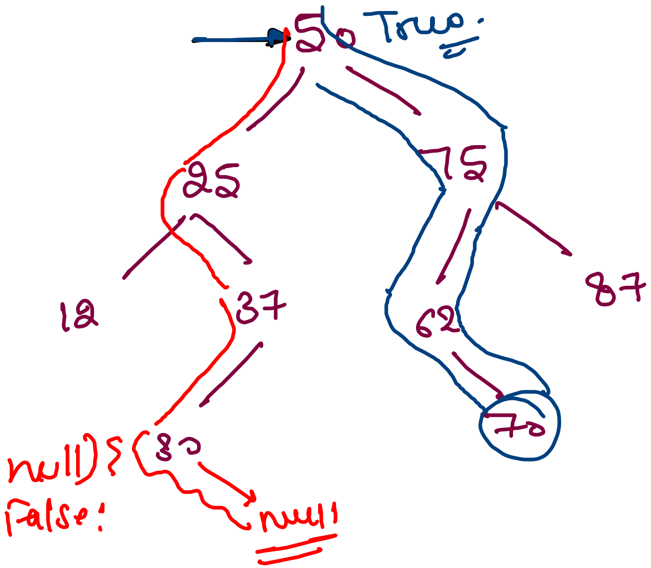
node==null } if root is null



```java
public static int max(Node node) {
    int res = 0;
    if(node == null) {
        res = Integer.MIN_VALUE;
    } else if(node.right == null) {
        res = node.data;
    } else {
        res = max(node.right);
    }

    return res;
}
```

find



dtf = 70

dtf = 35.

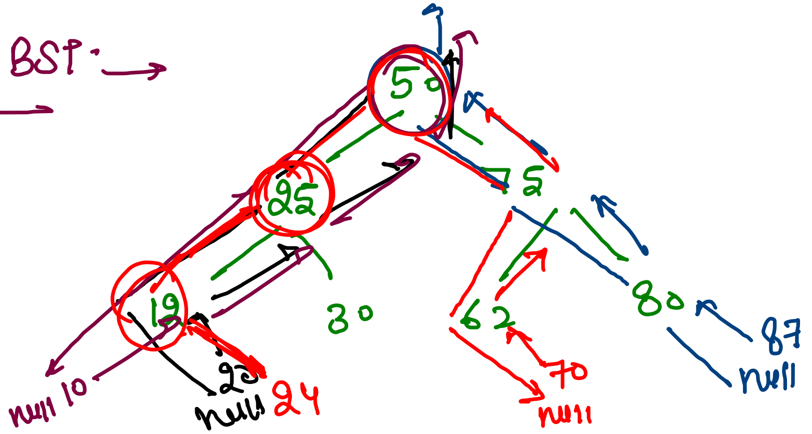
```
if(node == null){
    return false;
}
```

```
if( data > root.data) {
    // Right side.
    return find(root.right, dtf);
} else if (data < root.data) {
    // left side
    return find(root.left, dtf);
} else { // data == root.data
    //data found
    return true;
}
```

guided Recursion.

#Recursion is going toward target.

# Add node in BST →



✓ add → 87

✓ add → 70

✓ add → 20

✓ add → 10

add → 24

addNode is Similar to find