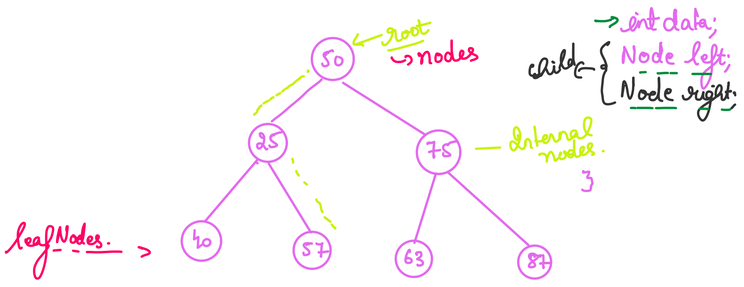
Lecture 16

Saturday, 29 February 2020 2:21 PM

B.S.T → Binary tree. → (2).

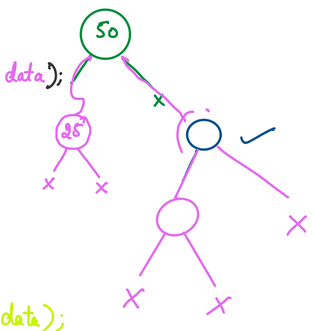
- Trees. → non-linear ds.
  - → heirarchical.




```
class Node {
    int data;
    Node left;
    Node right;
}
```

40 and 57 → Siblings.
40 and 63 → Cousins.
Size → 7
Max → 87
min → 25
height → 2.

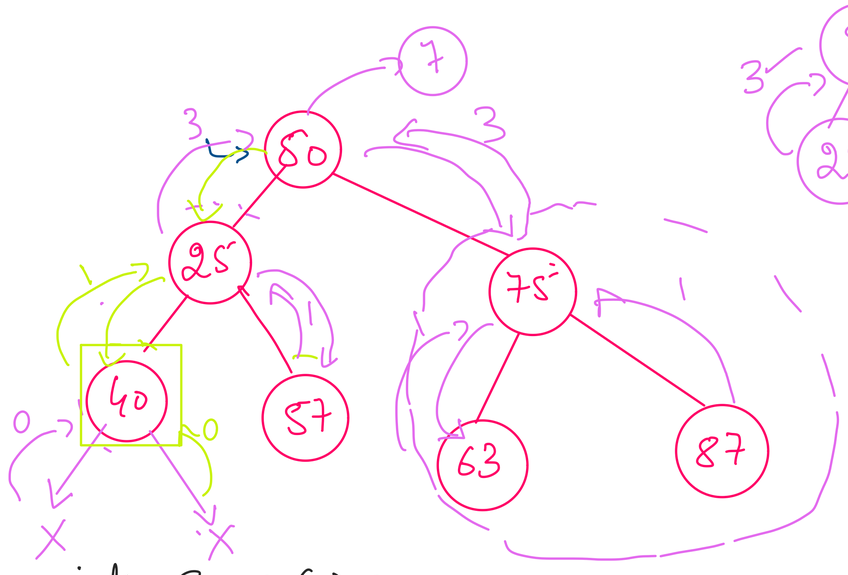
```
class Tree {
    class Node {
        int data;
        Node left;
        Node right;
        Node(int data, Node left, Node right) {
            this.data = data;
            this.left = left;
            this.right = right;
        }
    }
    Node root;
    int size;
    Tree() {
        Scanner s = new Scanner(System.in);
        this.root = takeInput(s, null, false);
    }
    private Node takeInput(Scanner s, Node parent, boolean isleftOrright) {
        if (parent == null) {
            Syso("enter data for root node");
        } else {
            if (isleftOrright) {
                Syso("enter data for left child of " + parent.data);
            } else {
                Syso("enter data for right child of " + parent.data);
            }
        }
        int data = s.nextInt();
        Node node = new Node(data, null, null);
        this.size++;
        boolean choice = false;
        Syso("Do you have left child of " + node.data);
        choice = s.nextBoolean();
        if (choice) {
            node.left = this.takeInput(s, node, true);
        }
        choice = false;
        Syso("Do you have right child of " + node.data);
        choice = s.nextBoolean();
        if (choice) {
            node.right = this.takeInput(s, node, false);
        }
        return node;
    }
```

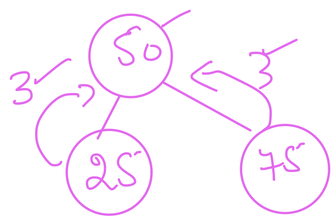


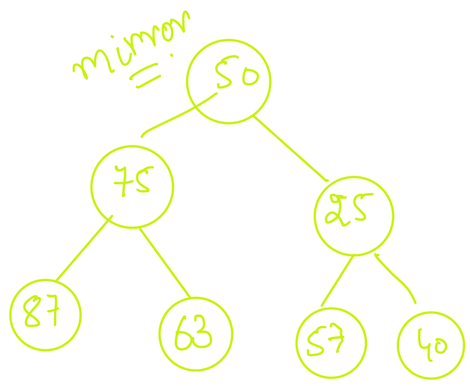

```
    display();
```

```
display() {
    25 => 50 <= 75
    40 => 25 <= 57
    null => 40 <= null
    — => 57 <= —
    63 => 75 <= 87
    null => 63 <= null
    null => 87 <= null.
}
```

Q => Size2:

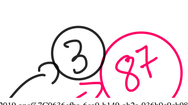


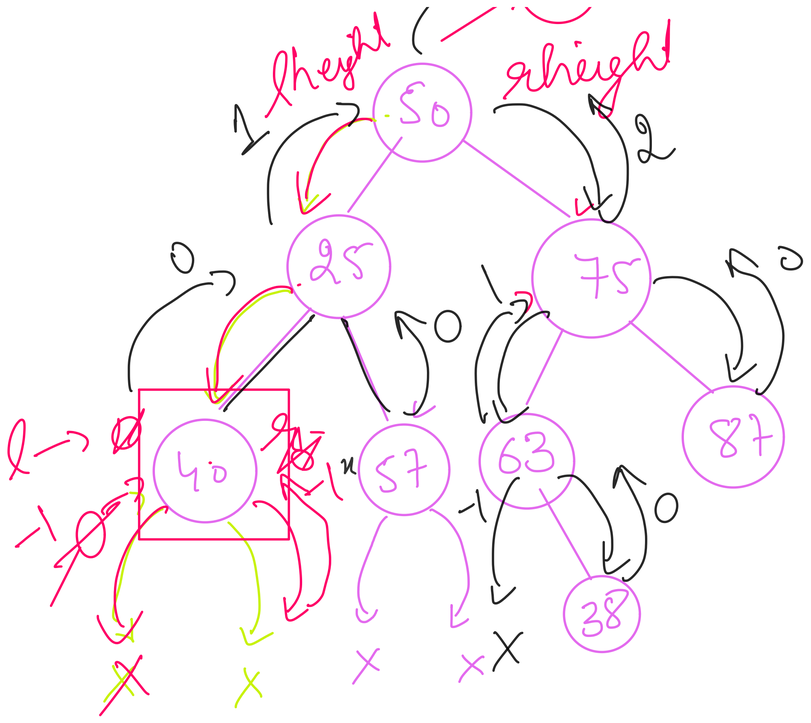





```
public int Size2() {
    Syso(this.Size2(this.root));
}
private int Size2(Node node) {
    if (node == null) {
        return 0;
    }
    int lsize = Size2(node.left);
    int rsize = Size2(node.right);
    int mysize = lsize + rsize + 1;
    return mysize;
}
```

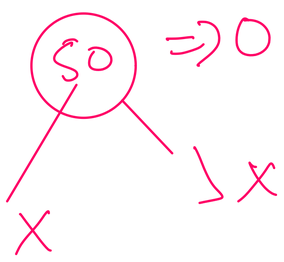
Q = Size2.
Q = Max → 87
Q = Min → 25
Q = height
Q = find(100);
Q = mirror();



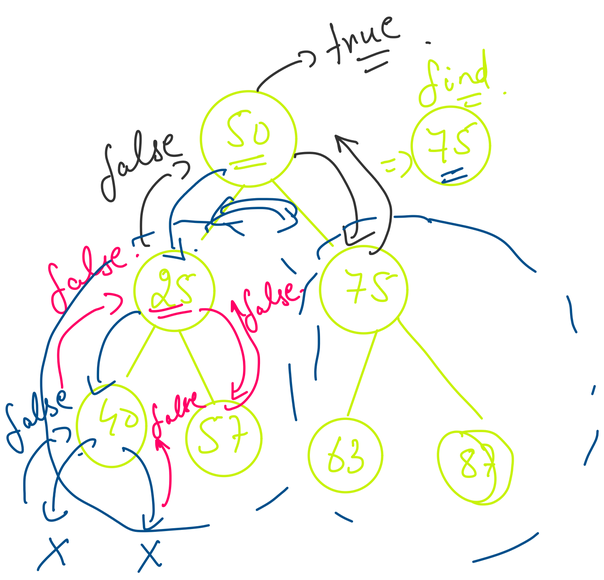


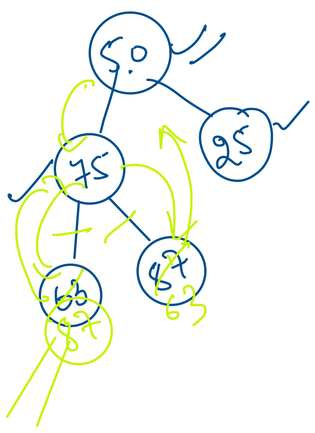

```
Max(Node node){
  if(node == null){
    return -∞
  }
  int lmax = Max(node.left);
  int rmax = Max(node.right);
  int mymax = Math.max(Max(lmax, rmax), node.data);
}
```

```
height(Node node){
  if(node==null){
    return 0;-1.
  }
  int lheight = height(node.left);
  int rheight = height(node.right);
  myheight = Math.max(lheight, rheight) + 1;
  return myheight;
}
```

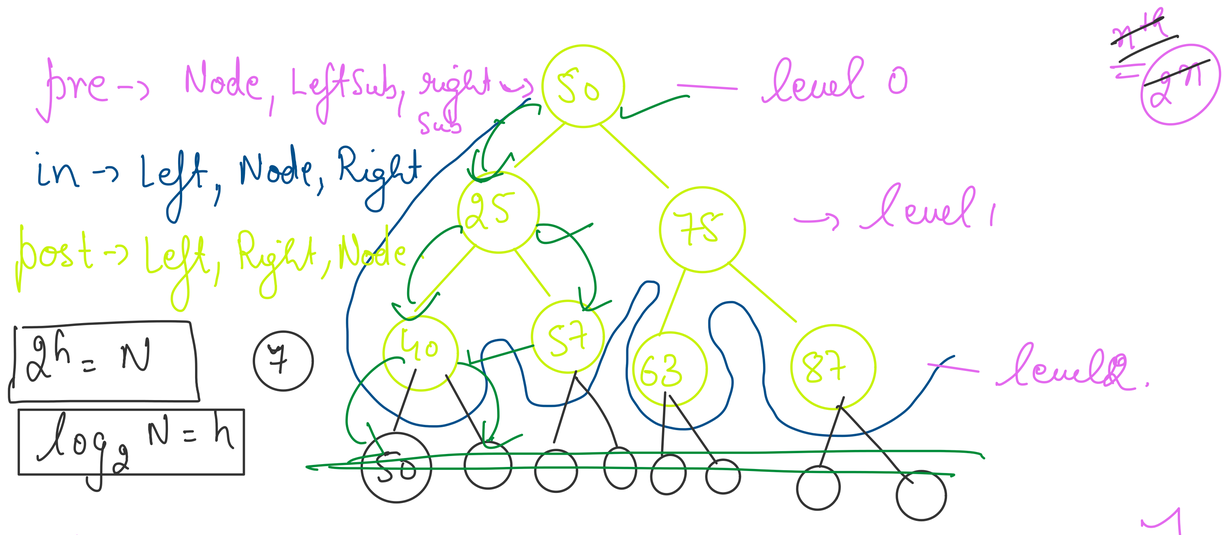



```
find(Node node, int data){
  if(node == null){
    return false
  }
  if(node.data == data){
    return true
  } else if(find(node.left, data)){
    return true;
  } else if(find(node.right, data)){
    return true;
  } else{
    return false;
  }
}
```






```
void mirror(Node node){
  if(node == null){
    return;
  }
  Node temp = node.left;
  node.left = node.right;
  ___.right = temp;
  mirror(node.left);
  mi___(___.right);
}
```



pre → 50, 25, 40, 57, 75, 63, 87.

in → 40, 25, 57, 50, 63, 75, 87.

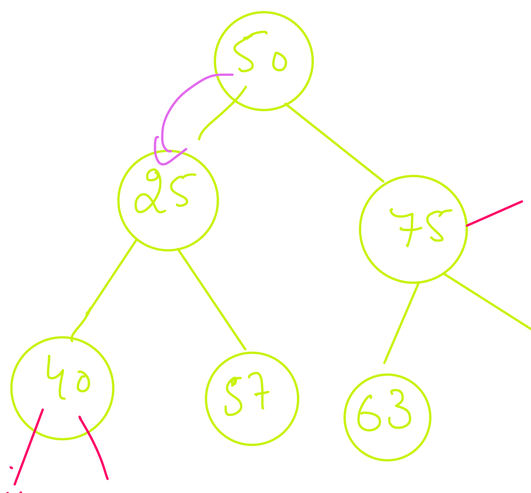
↳ Recursive.

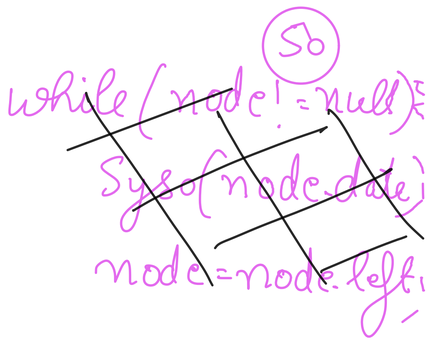
post → 40, 57, 25, 63, 87, 75, 50.

level → 50, 25, 75, 40, 57, 63, 87 } Iterative.

```
public void preOrder( ){
        preOrder(this.root);
}
private void preOrder(Node node){
        if(node==null){
            return;
        }
        Syso(node.data);
        preOrder(node.left);
        preOrder(node.right);
}
```

50





Console

50 25 75 40 57 63 87 →

Queue => 50 ~~25~~ ~~75~~ ~~40~~ ~~57~~ ~~63~~ ~~87~~

```
public void levelOrder( ){
    Queue.add(root.data);
→   while(!queue.isEmpty()){
        Node node = queue.remove();
        Syso(node.data);
        if(node.left!=null){
            q.add(node.left);
        }
        if(node.right!=null){
            q.add(node.right);
        }
    }
}
```