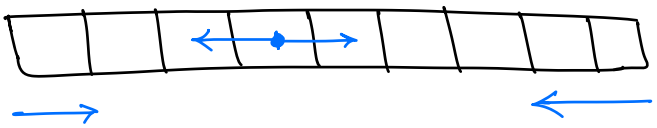
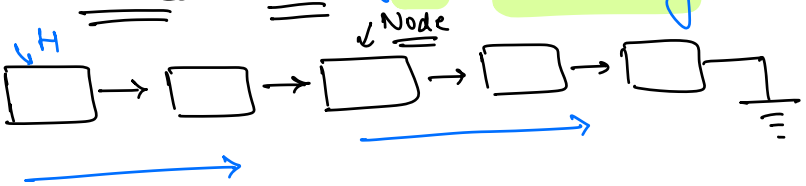
DS till now :-

1) Arrays : No hierarchy (Linear DS)



Random access ⇒ O(1)

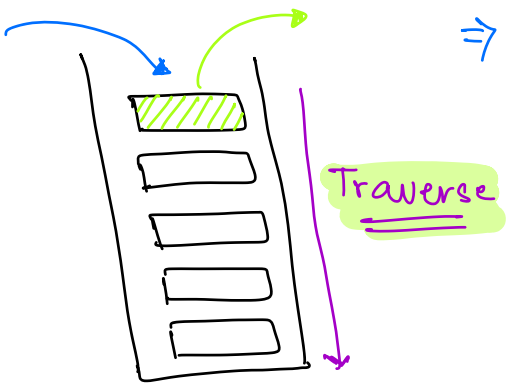
2) Linked List (No Hierarchy)



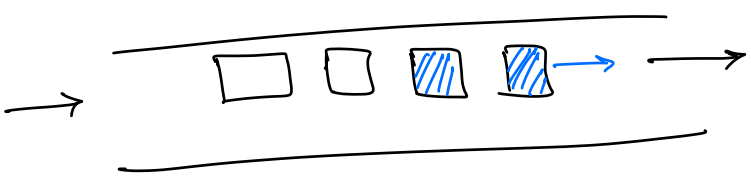
⇒ Linear DS

3) Stack (No Hierarchy)

⇒ Linear DS



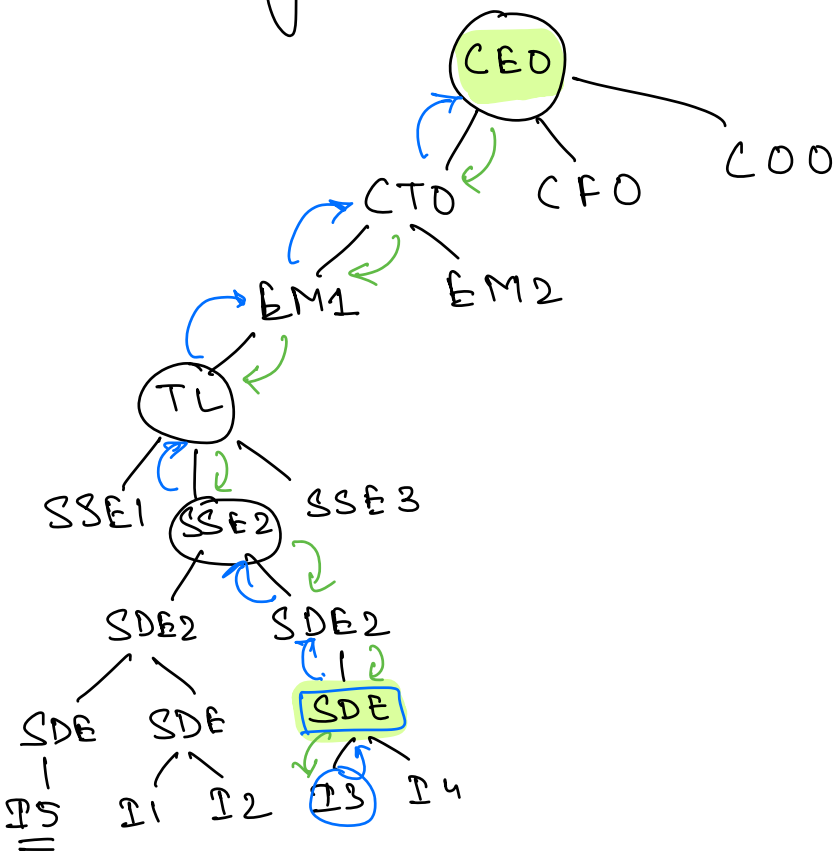
4) Queue (No Hierarchy)



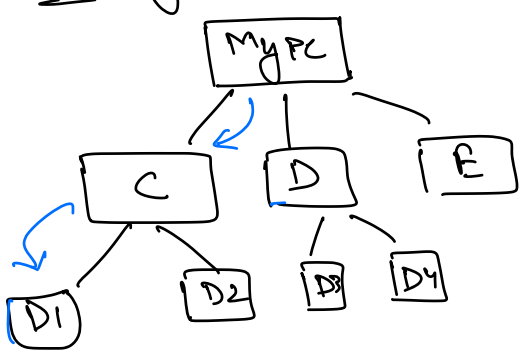
⇒ Linear DS

Hierarchical Data

1) Company Structure



2) Directory Structure



3) HTML Structure (Tags)

(XML)

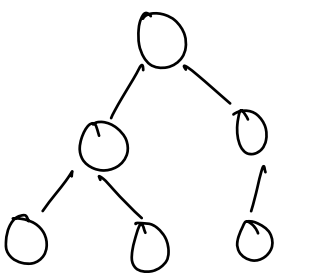
Tree DS

↳ Stores Hierarchical data

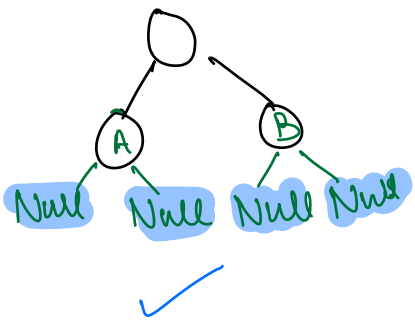
⇒ Binary Tree

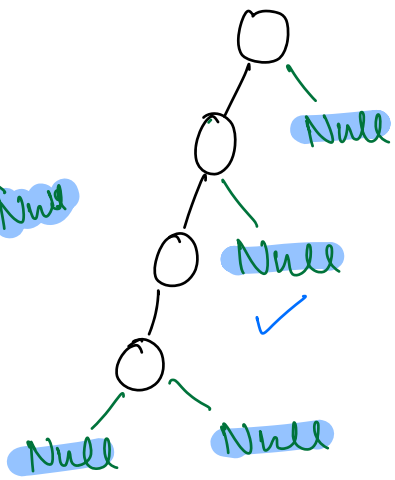
↳ Max 2 children of a node

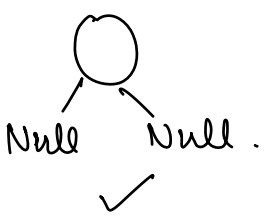
⇒ Every node can have 0 | 1 | 2 children

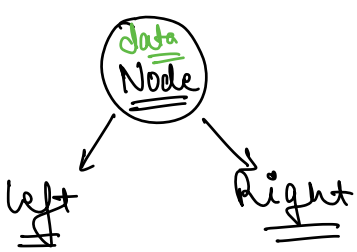


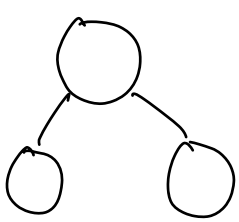
Binary Tree









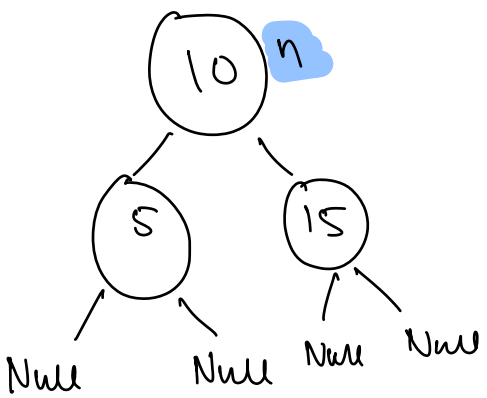


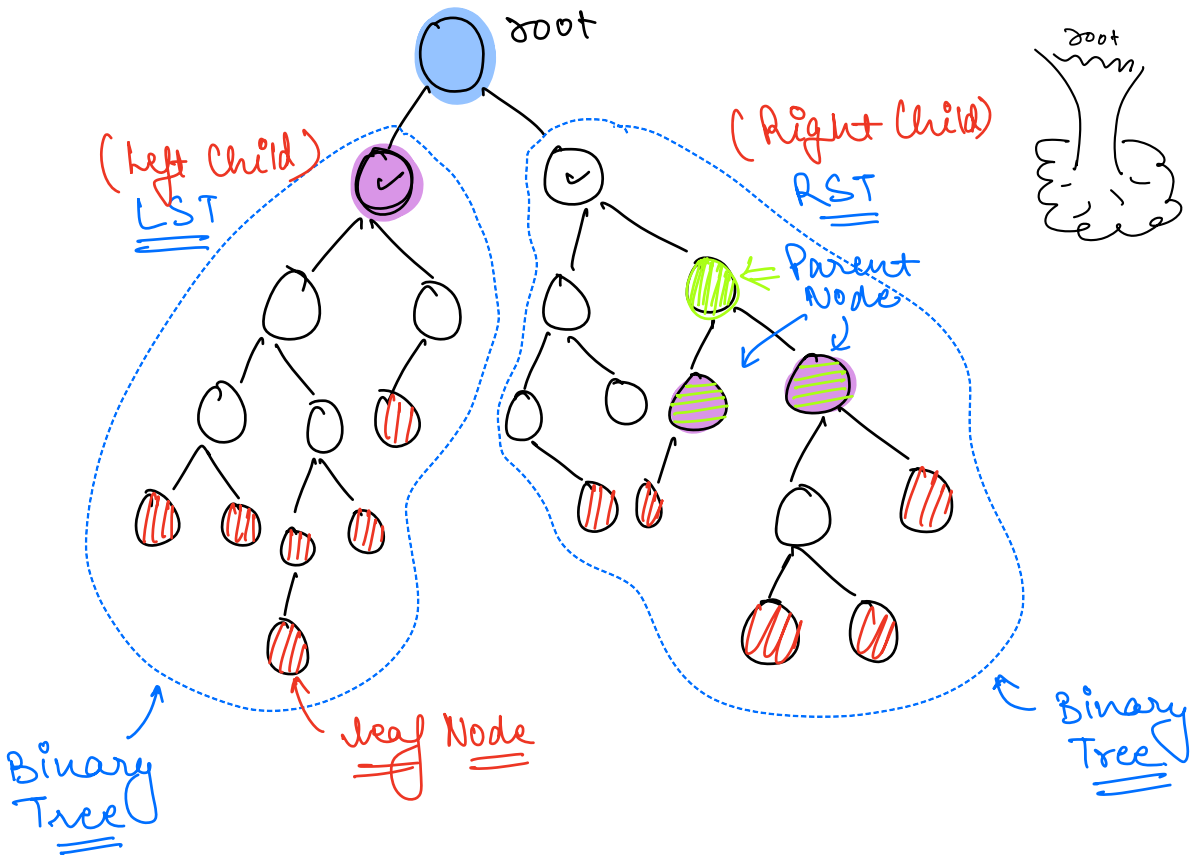
```
Class Node {
    int data;
    Node left;
    Node right;
    Node(int x) {
        this.data = x;
        this.left = Null;
        this.right = Null;
    }
}
```

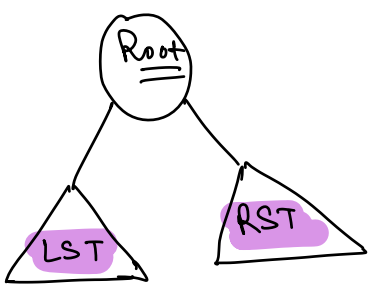
```
Node n = new Node();
```

n.data
n.left
n.right
} garbage values

```
Node n = new Node(10);
Node leftNode = new Node(5);
n.left = leftNode;
Node rightNode = new Node(15);
n.right = rightNode;
```







Binary Tree is a Recursive Structure.

1) LST is also a Binary Tree

2) RST " " " " "

Tree Traversal

1) Root value
   LST
   RST

2) ~~Root value~~
   RST
   LST

3) LST
   Root value
   RST

4) LST
   RST
   Root value

5) ~~RST~~
   Root value
   LST

6) ~~RST~~
   LST
   Root value

⇒ Order is very Imp in Trees.

⇒ Rule: LST before RST. ✓

1) Root value ✓
   LST ⇐
   RST ⇐
   Pre order

2) LST
   Root value ✓
   RST
   Inorder

3) LST
   RST
   Root value
   Post order

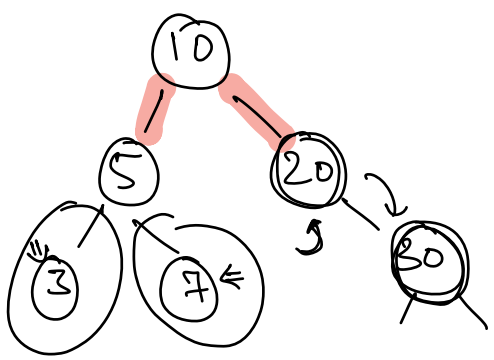
```
# void preOrder(root) {
    if(root == Null) return;

    print(root.data);
    preOrder(root.left);
    preOrder(root.right);
  }
```

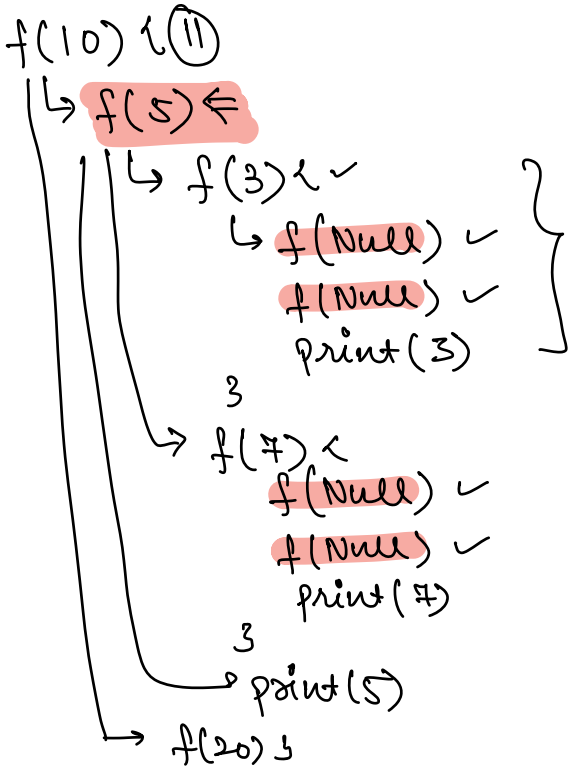
```
# Void inOrder (root) {
    if (root == Null) return;

    inOrder (root.left);
    print (root.data);
    inOrder (root.right);
}
```

```
# Void postOrder (root) {
    if (root == Null) return;
    postOrder (root.left);
    postOrder (root.right);
    print (root.data);
}
```

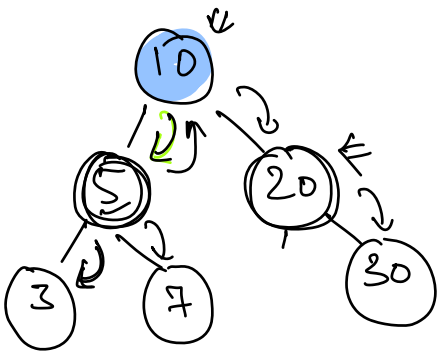
(10) root

Ex





3, 7, 5, 30, 20, 10



Pre Order

Root
LST
RST

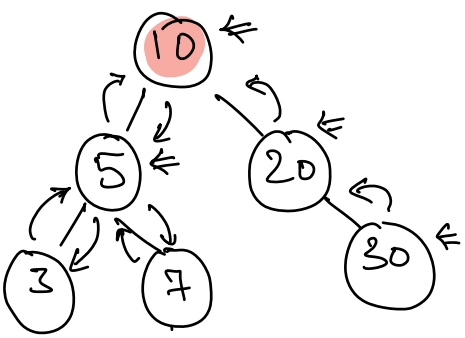
10, 5, 3, 7, 20, 30

f(10) ⇒ f(5), f(3), f(7),

f(Null) f(Null)

⇒ f(20) ✓

f(Null) f(30) ✓



Inorder

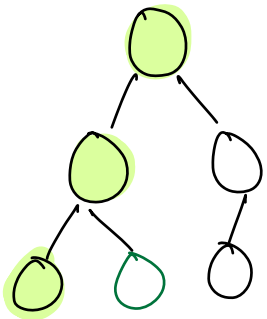
LST
Root
RST

3, 5, 7, 10, 20, 30

Q Given a Binary Tree, find the Height of it.

Root Node is given

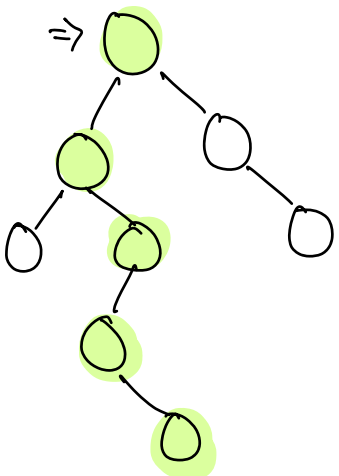
Height: No. of nodes / length of the largest path from root to leaf node.



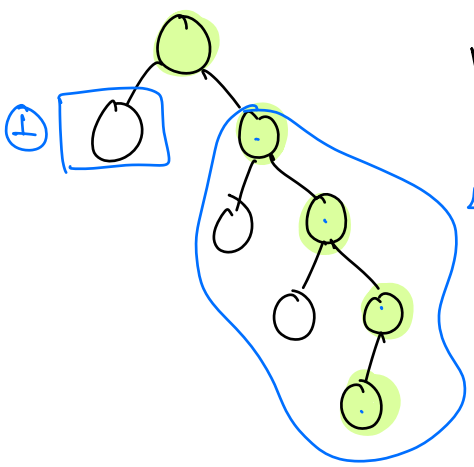
⇒ 3 | 2

3 ← No. of Nodes

2 ← No. of edges.



⇒ 5 = H

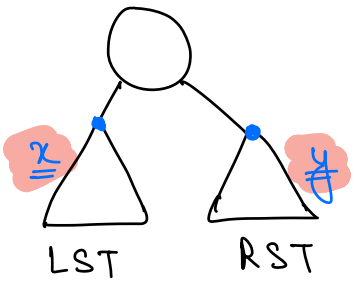


H = 5 ✓

H = max(1, 4) + 1 (Root Node)

= 4 + 1

= 5

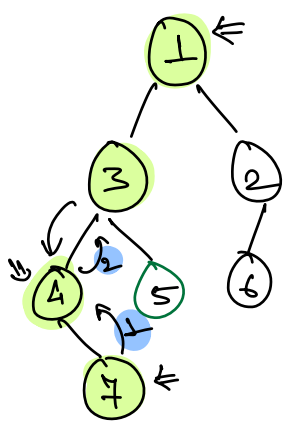


$H = \max(x, y) + 1$

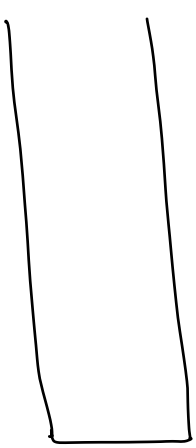
Height of a B.T = max (Height LST, Height RST) +1

Root Node

```
int height(root) {
    if(root == Null) return 0;
    leftHeight = height(root.left);
    rightHeight = height(root.right);
    return max(leftHeight, rightHeight)+1;
}
```



4



Post Order Traversal

Doubts

H = 1

(10)

H(10) {

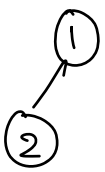
lh = H(Null) ⇐ 0

rh = H(Null) ⇐ 0

return max(0,0) + 1

}

↳ (1)



↑(2)

H(1) {

lh = 1

rh = 0

return max(1,0) + 1

}

max(a,b) —
- if a < b ⇒ b
- if a > b ⇒ a
- if a == b ⇒ a|b.