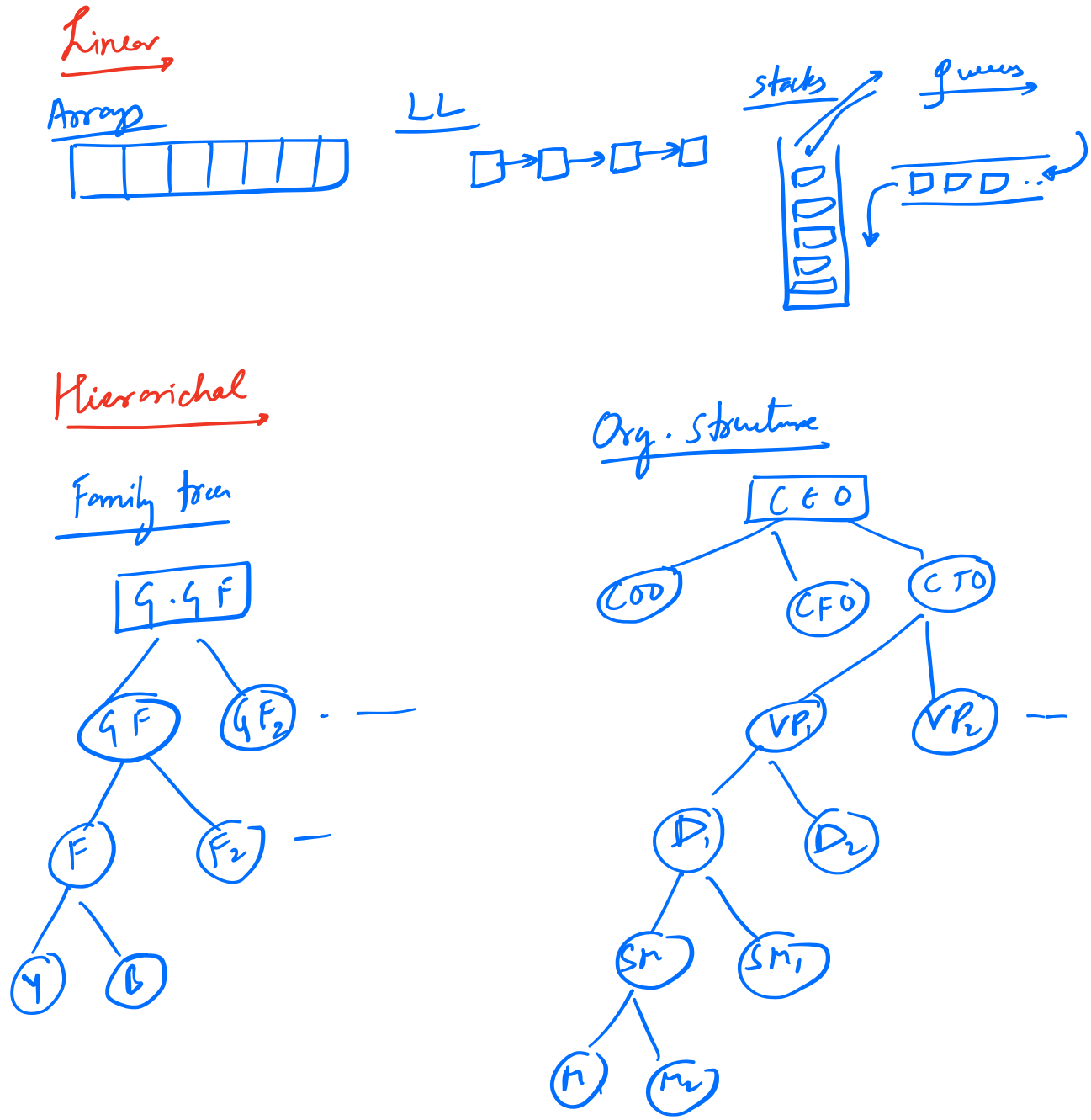
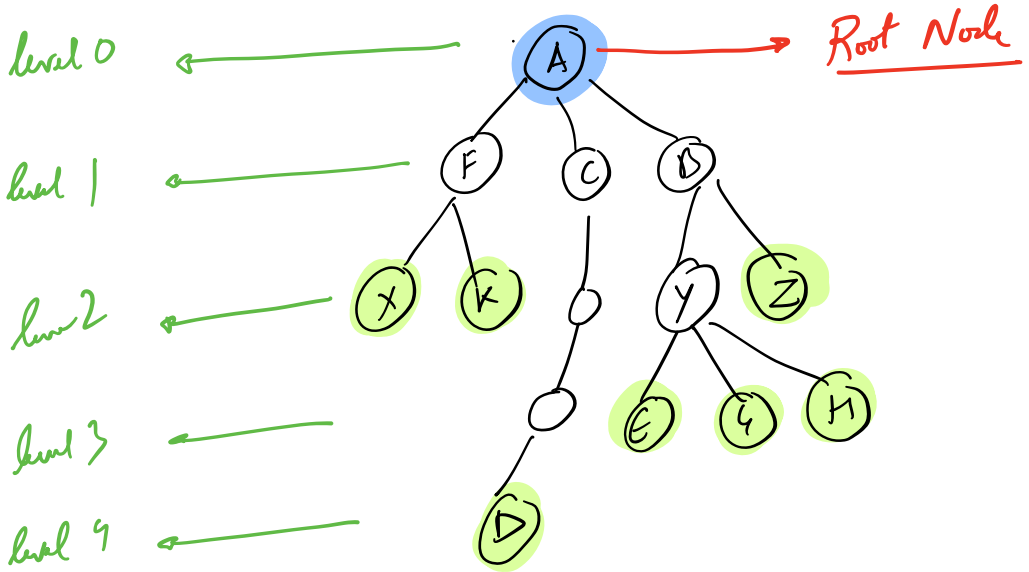

Linear
Array
LL
stacks
queues
DDD..
Hierarchal
Family tree
G.GF
GF
GF2
F
F2
Y
B
Org. Structure
CEO
COO
CFO
CTO
VP1
VP2
D1
D2
SM
SM1
M
M2
UPSIDE DOWN !



A is parent of B

B is child of A

F, C, B are siblings

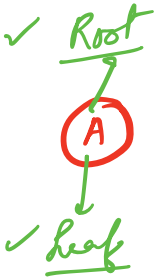
Ancestors of H : Y, B, A.

Descendants of B : Y, Z, E, G, H

Leaf Nodes: X, K, D, E, G, H, Z

→ Node with NO child.

→ having the SAME PARENT!

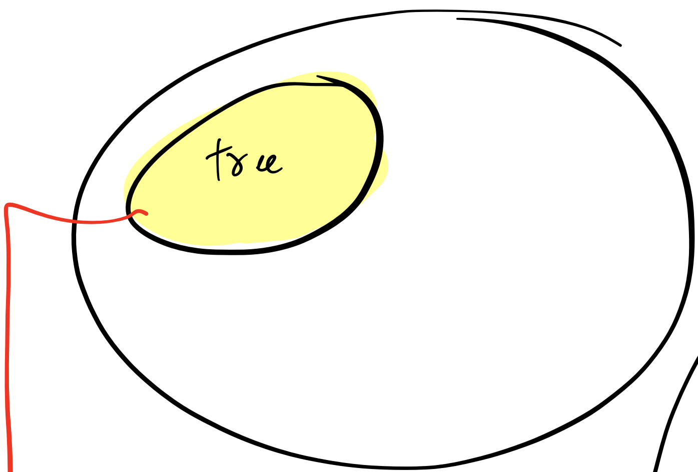


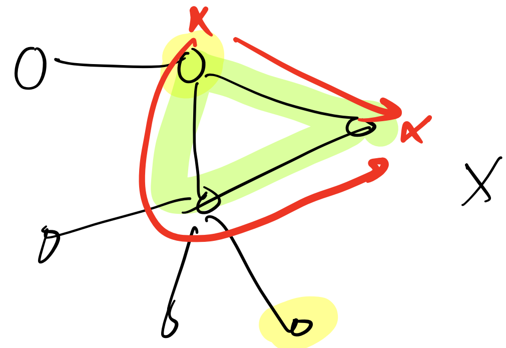
Tree : Hierarchical DS.

AT MAX 1 root!

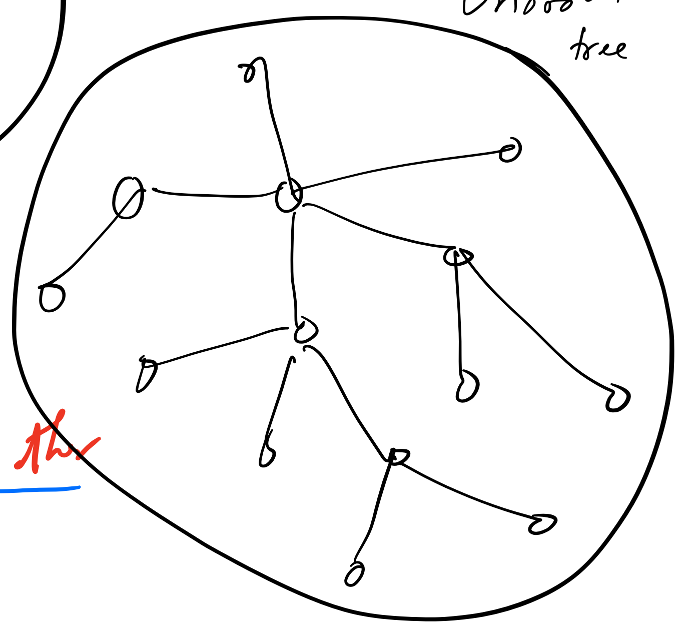
Every node → exactly 1 parent!
except for the root node!

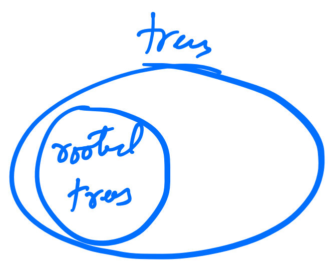
Graph



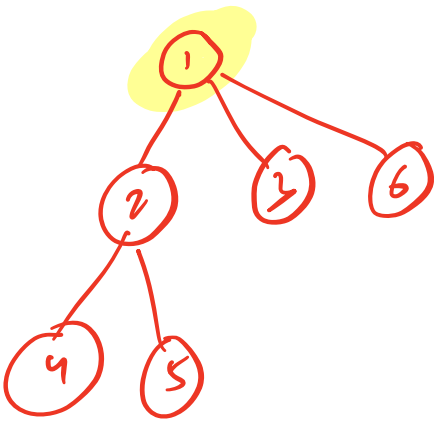


exactly 1 path from any node to any other node!





Q If a tree has N nodes.
How many edges does it have?

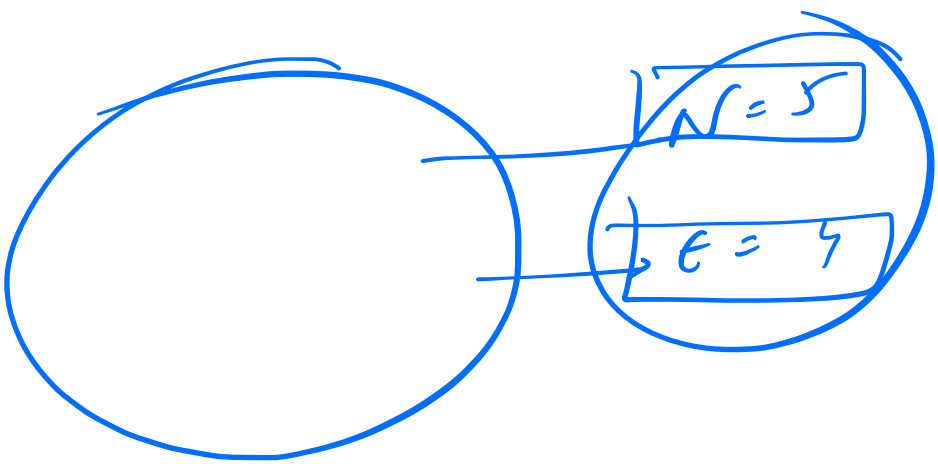


x

Nodes	Edges
1	0
2	1
3	2
4	3
5	4
6	5

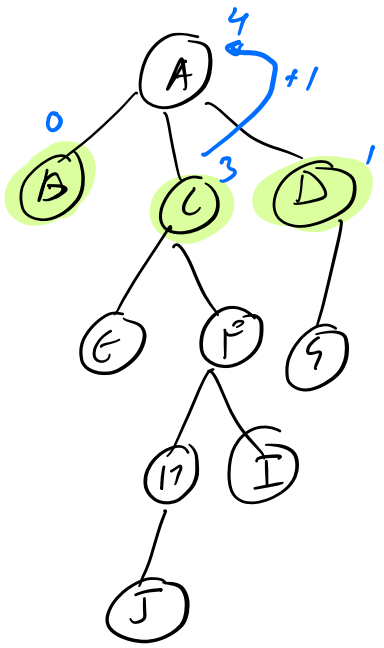
+1 +1 +1

Edges = Nodes − 1



Height of a Node :

Length of the longest path from that node to any of it's descendants

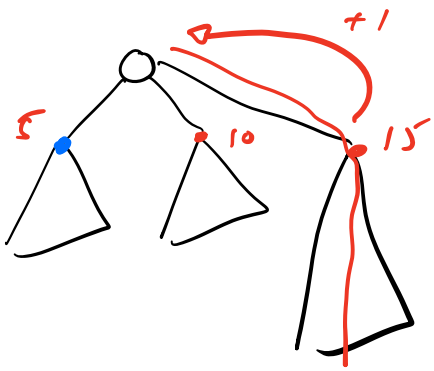


$h(C) = 3$

$h(B) = 0$

$h(D) = 1$

$h(A) = 4$



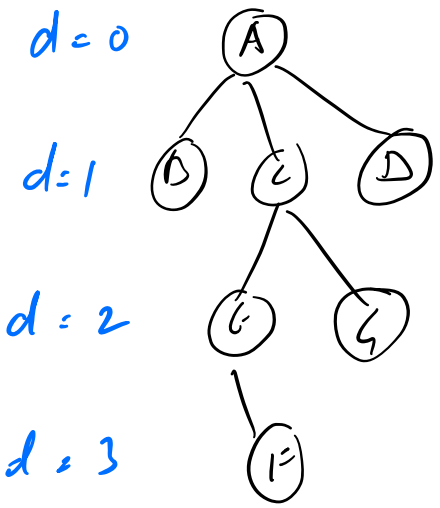
$$h(A) = 1 + \max(h(B), h(D), h(C))$$

$h(\text{node}) = 1 + \max$ height of it's children !

① Depth of a Node

[Distance from the root]

Depth ≡ Level

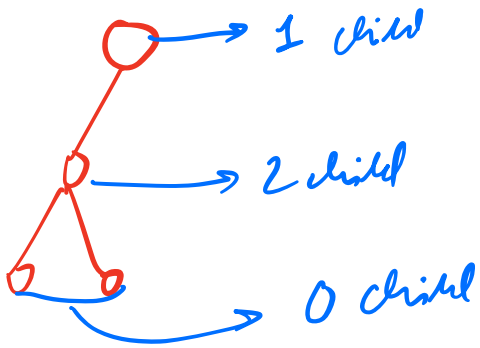


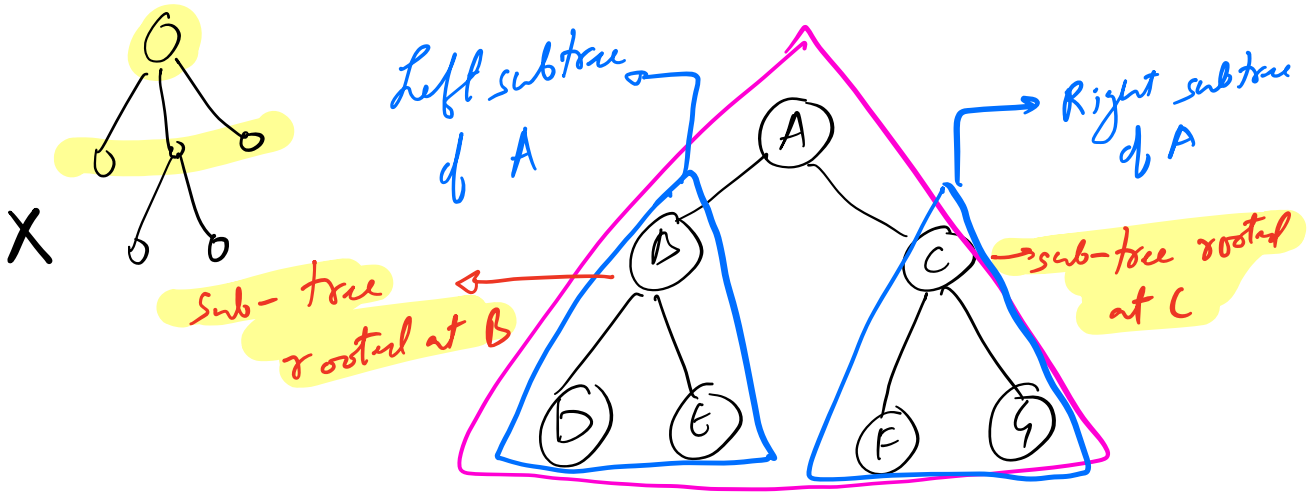
---

② Height of a tree ?

= height (root)

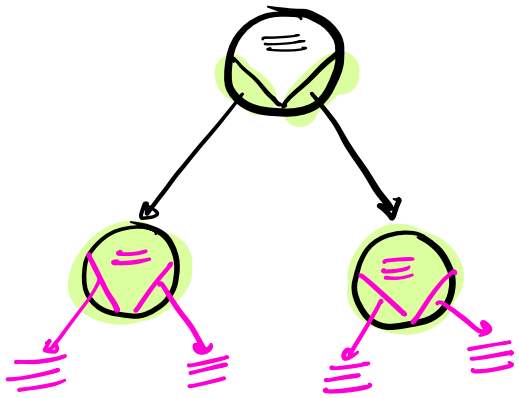
Binary Tree → [Every node would have AT MAX 2 children]

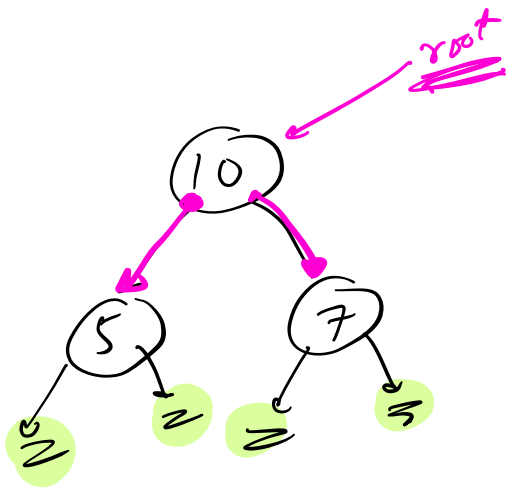




(*) subtree rooted at a node is: that node & all it's descendants!

```
class Node {
    int data;
    Node left;
    Node right;
    Node(val) {
        data = val;
        left = NULL;
        right = NULL;
    }
}
```






```
Node root = new Node(10);
root.left = new Node(5);
root.right = new Node(7);
```

Tree traversal →

PREORDER	INORDER	POSTORDER
D L R	L D R	L R D

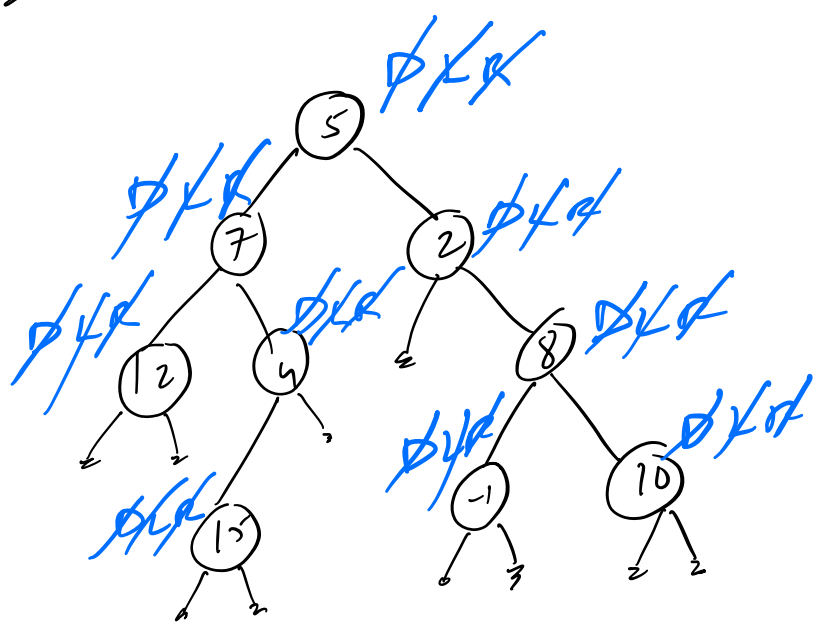
① Preorder D L R

1. process the data of the current node!
2. traverse the left subtree of the current node in preorder fashion
3. ——— right ———

& print in preorder fashion!

5, 7, 12, 4, 15

2, 8, -1, 10

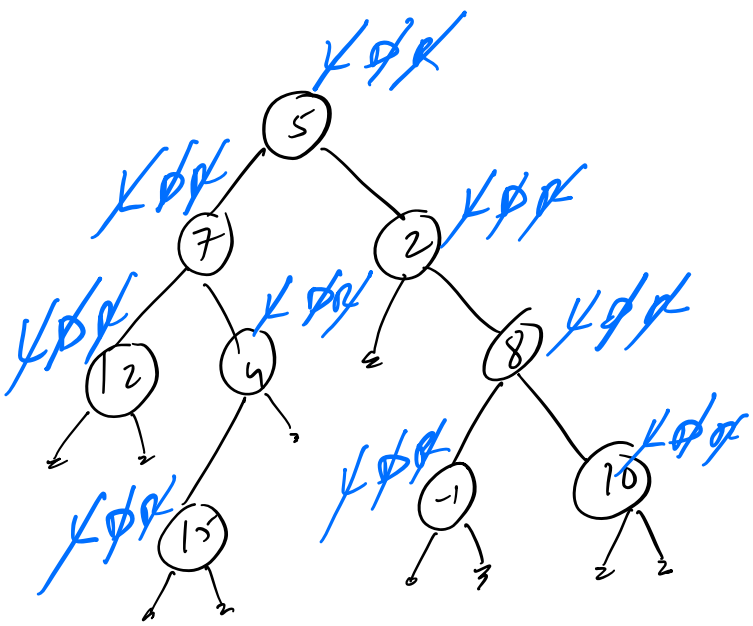


INORDER → L D R

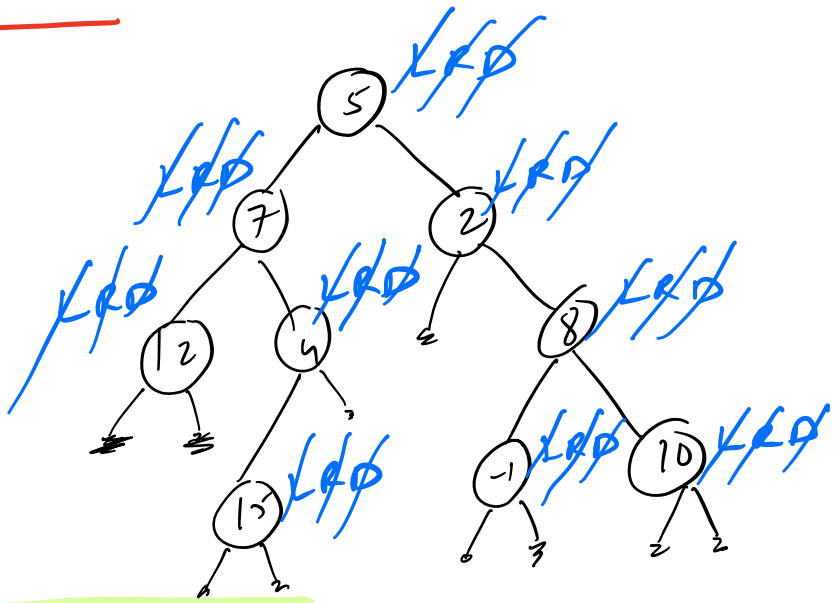
1. traverse the Left subtree of the current node in inorder fashion
2. process the data of the current node!
3. ——— right ———

12, 7, 15, 4, 5

2, -1, 8, 10,

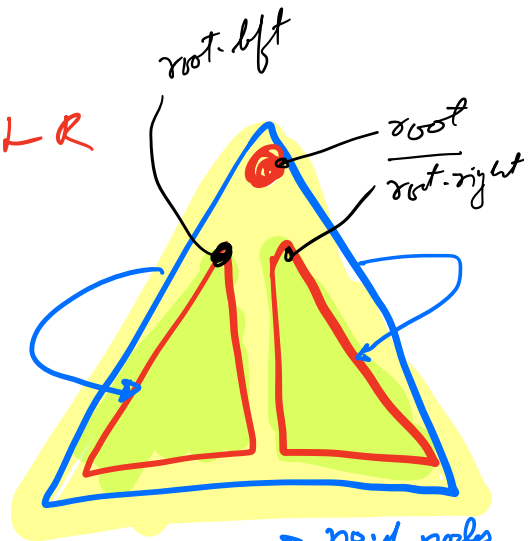


POST ORDER [L R D]



12, 15, 4, 7, -1, 10, 8, 2, 5

CODE : PREORDER DLR

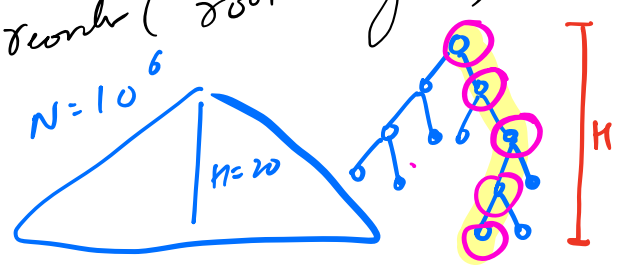



```
void preorder (Node root) {
// Ass: print the tree rooted at root in preorder fashion
if (root == NULL) ret;
print (root.data);
preorder (root.left);
preorder (root.right);
}
```

no. of nodes

TC = N × O(1)

TC = O(N)

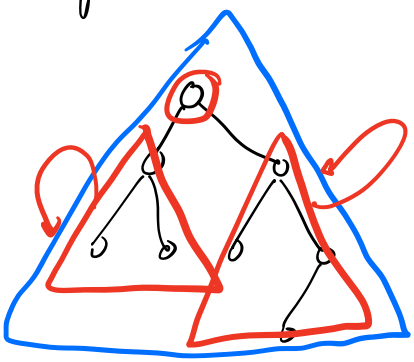


SC = O(H)

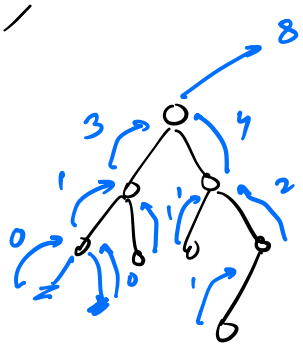
H → height of tree!

HW Code for INORDER & POSTORDER!

Q Given a B.T. Find the no. of nodes!

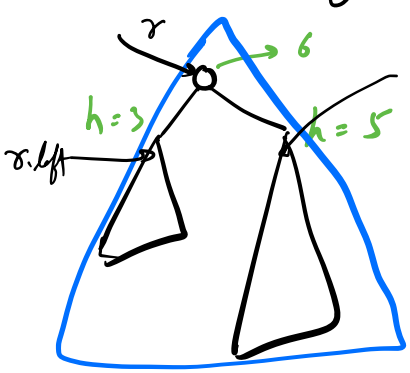


```
int cnt(Node r) {
// Ass: ret the # nodes in
       tree rooted at 'r'
  if (r == NULL) ret 0;
  ret cnt(r.left) + cnt(r.right) + 1;
}
```

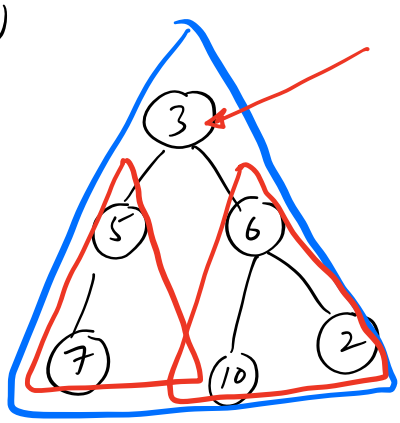


Q Given a BT. Find the height!

```
int h(Node r) {
  if (r == NULL) ret -1;
  ret 1 + max{ h(r.left), h(r.right) };
}
```



Q Given a B.T. Each node has a val.
Find the sum of values of all nodes!




```
int sum(Node r) {
  if(r==NULL) ret 0;

  ret (sum(r.left)
    + sum(r.right)
    + r.val );
}
```

TC = O(N)

SC = O(H)