CS & IT ENGINEERING

Data Structure

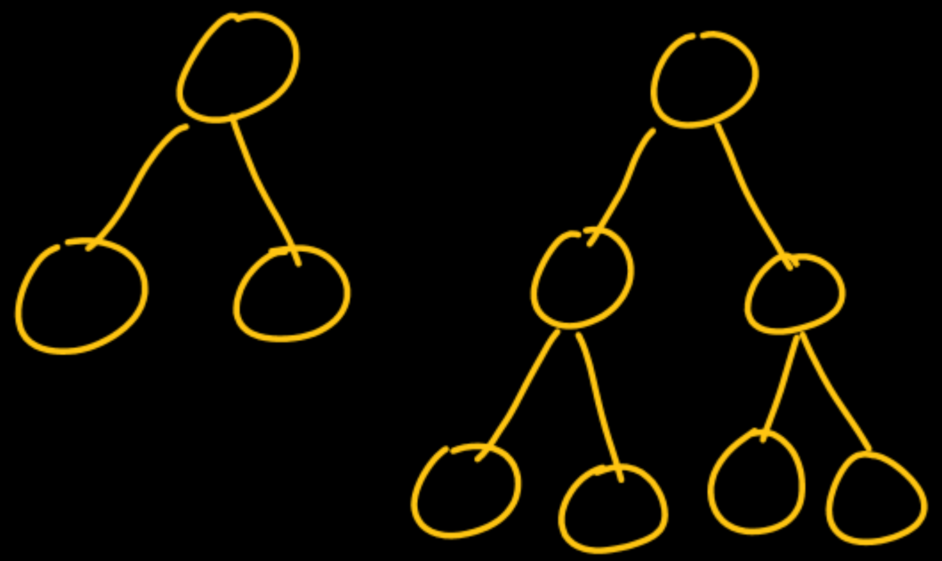
Tree
Chapter- 5
Lec- 02

By- Pankaj Sharma sir

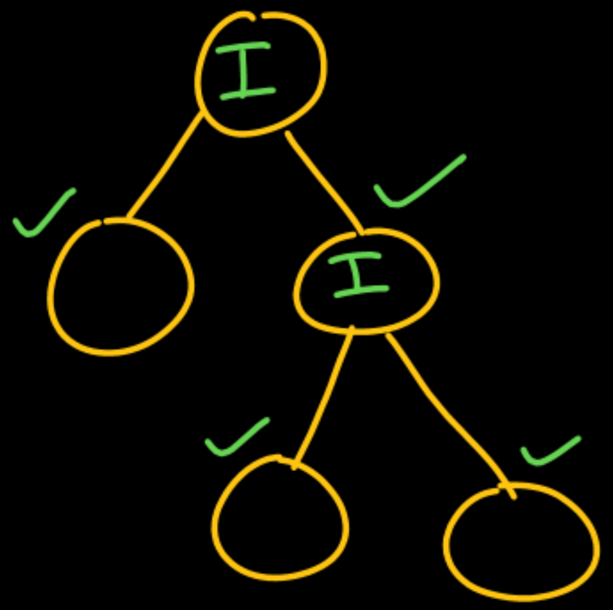
TOPICS TO BE COVERED

- Tree-II

Full binary tree :

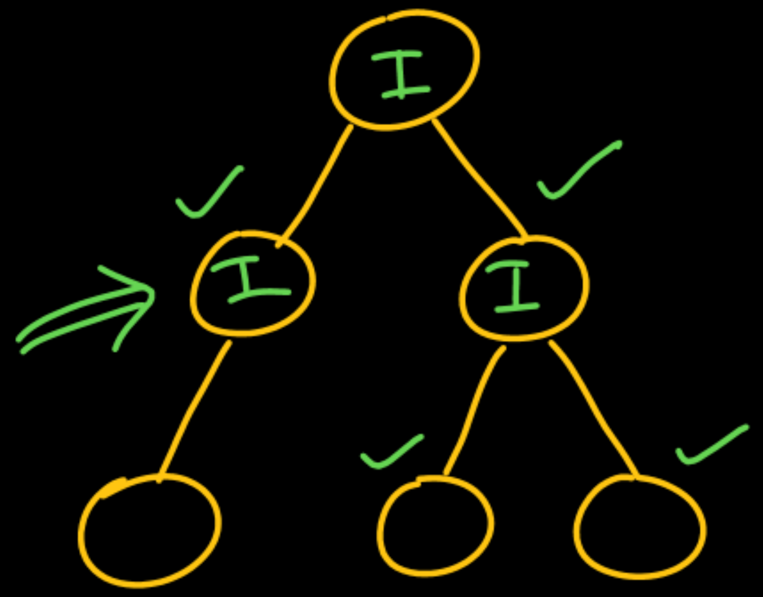


2-ary tree : A tree in which every internal node has exactly 2 childs.



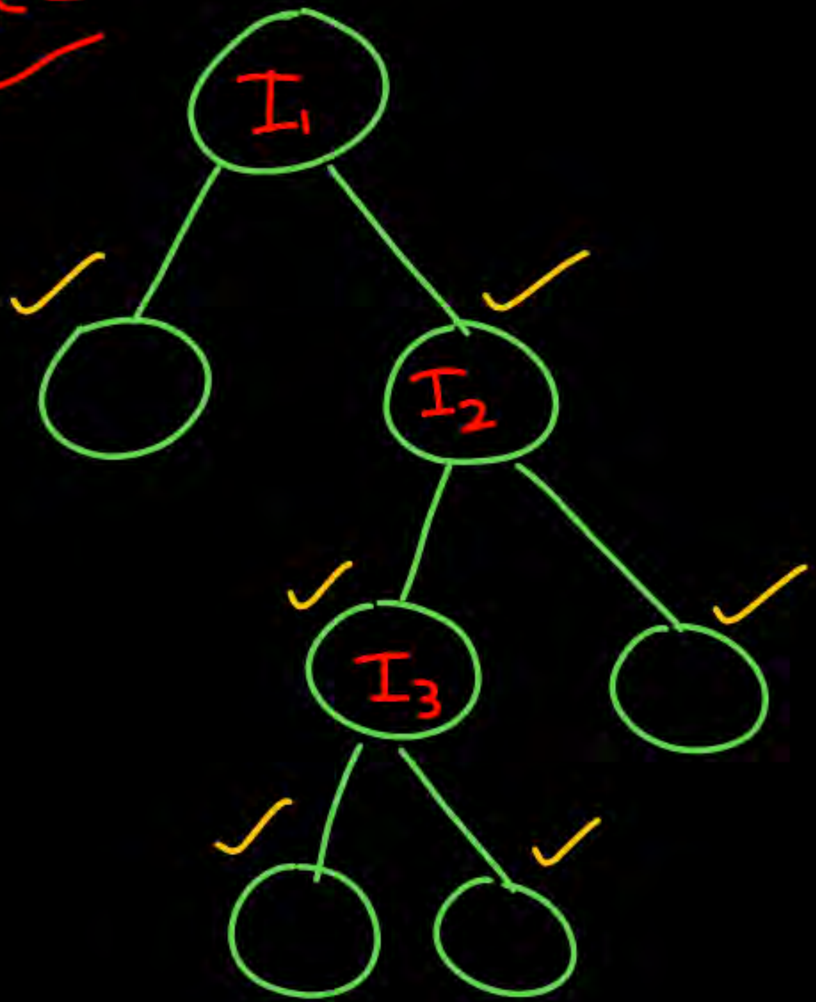
2-ary tree

binary tree



2-ary tree ✗

2-ary tree



3 nodes of degree 2

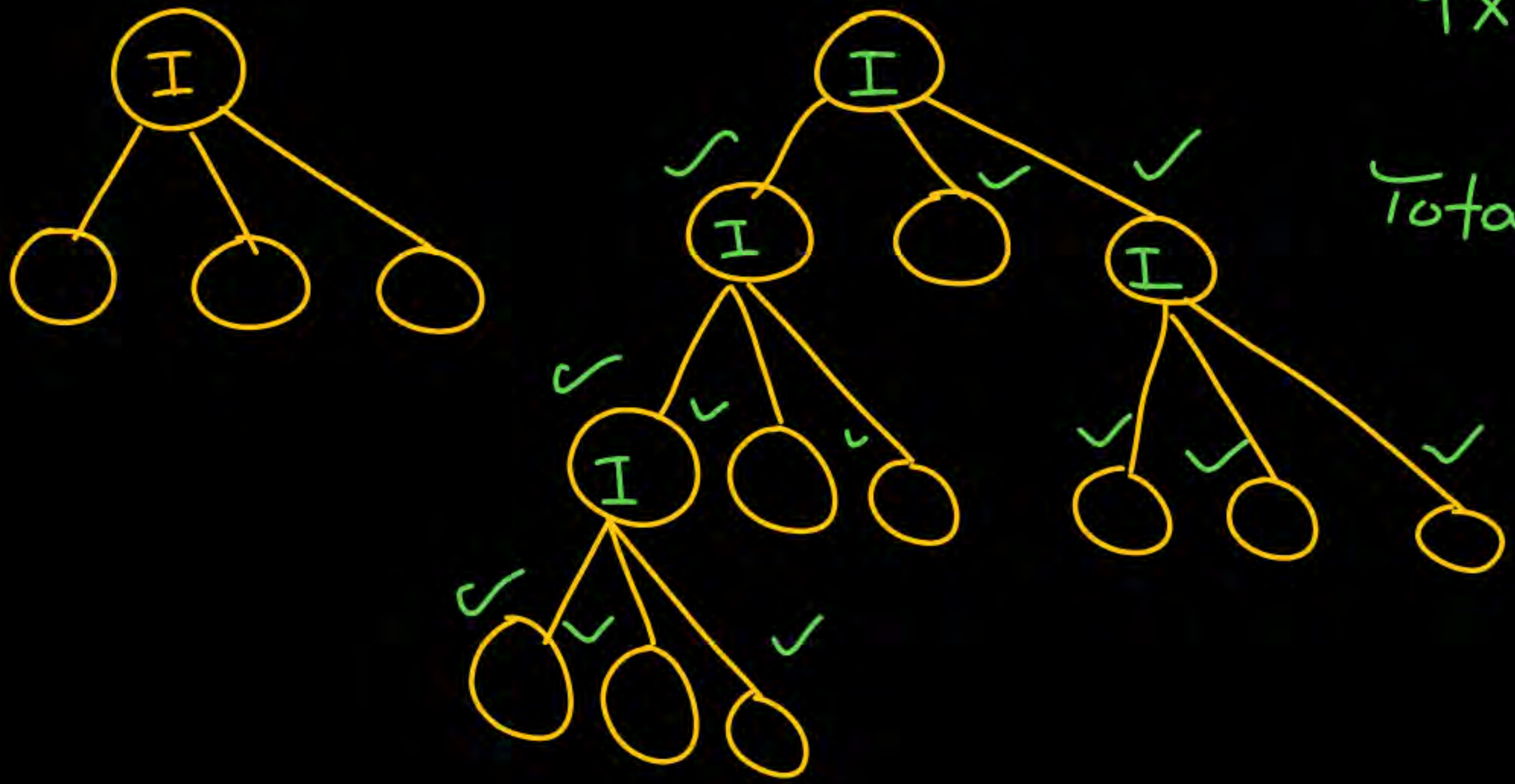
OR

3 internal nodes

$= 3 \times 2$

Total no. of nodes $= 3 \times 2 + \overset{\text{Root}}{1}$

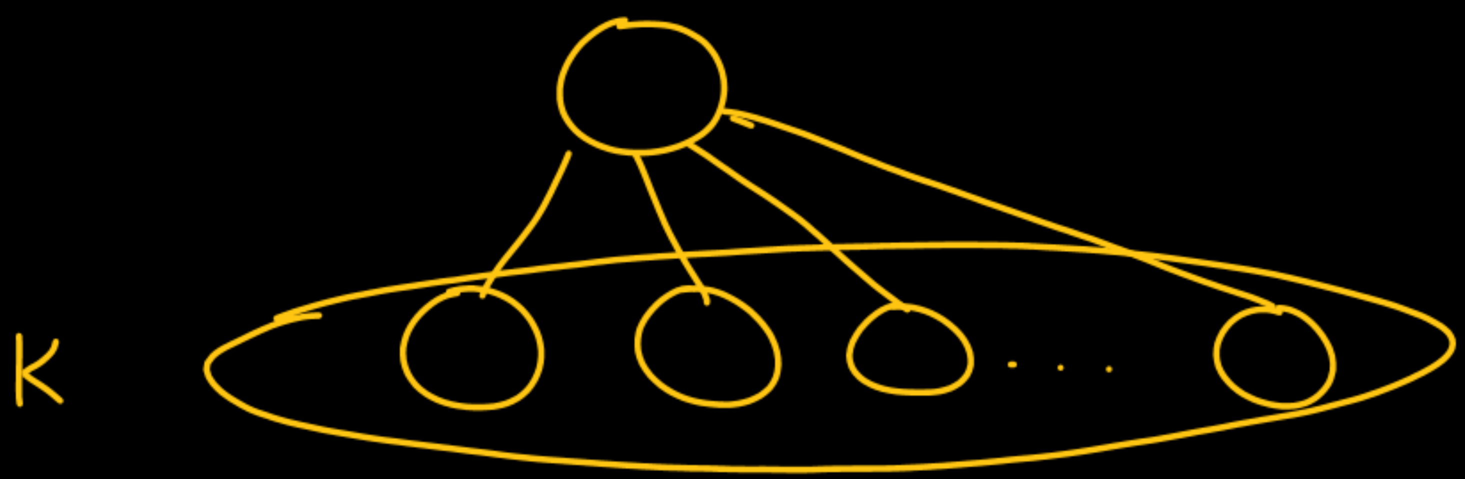
3-ary tree : A tree in which every internal node has exactly 3 childs



4×3 → no. of internal node → childs

Total $= 4 \times 3 + 1$ Root

K-ary tree : A tree in which every internal node has exactly K-childs.



let I be the no. of internal nodes.

Total = $I \times K + 1$

$$\boxed{n = K \cdot I + 1} \quad -(1)$$

$$\boxed{n = kI + 1}$$

#. of leaf nodes + #of internal nodes $= kI + 1$

$$L + I = k \cdot I + 1$$

$$L = kI - I + 1$$

$$\boxed{L = I(k-1) + 1} \quad -(2)$$

$$\boxed{n = KI + 1} \quad -(1)$$

$$\boxed{L = (K-1)\cdot I + 1} \quad -(2)$$

$$I = \frac{L-1}{K-1}$$

Don't learn any of these

$$n = K\left(\frac{L-1}{K-1}\right) + 1$$

$$= \frac{KL - \cancel{K} + \cancel{K} - 1}{K-1}$$

$$\boxed{n = \frac{KL-1}{K-1}}$$

Q A binary tree is having 6 nodes of degree 1, 12 nodes of degree 2

find the no. of leaf nodes.

$$\text{Total} = \overset{\text{Root}}{1} + 6 \times 1 + 12 \times 2$$

$$L + 6 + 12 = 31$$

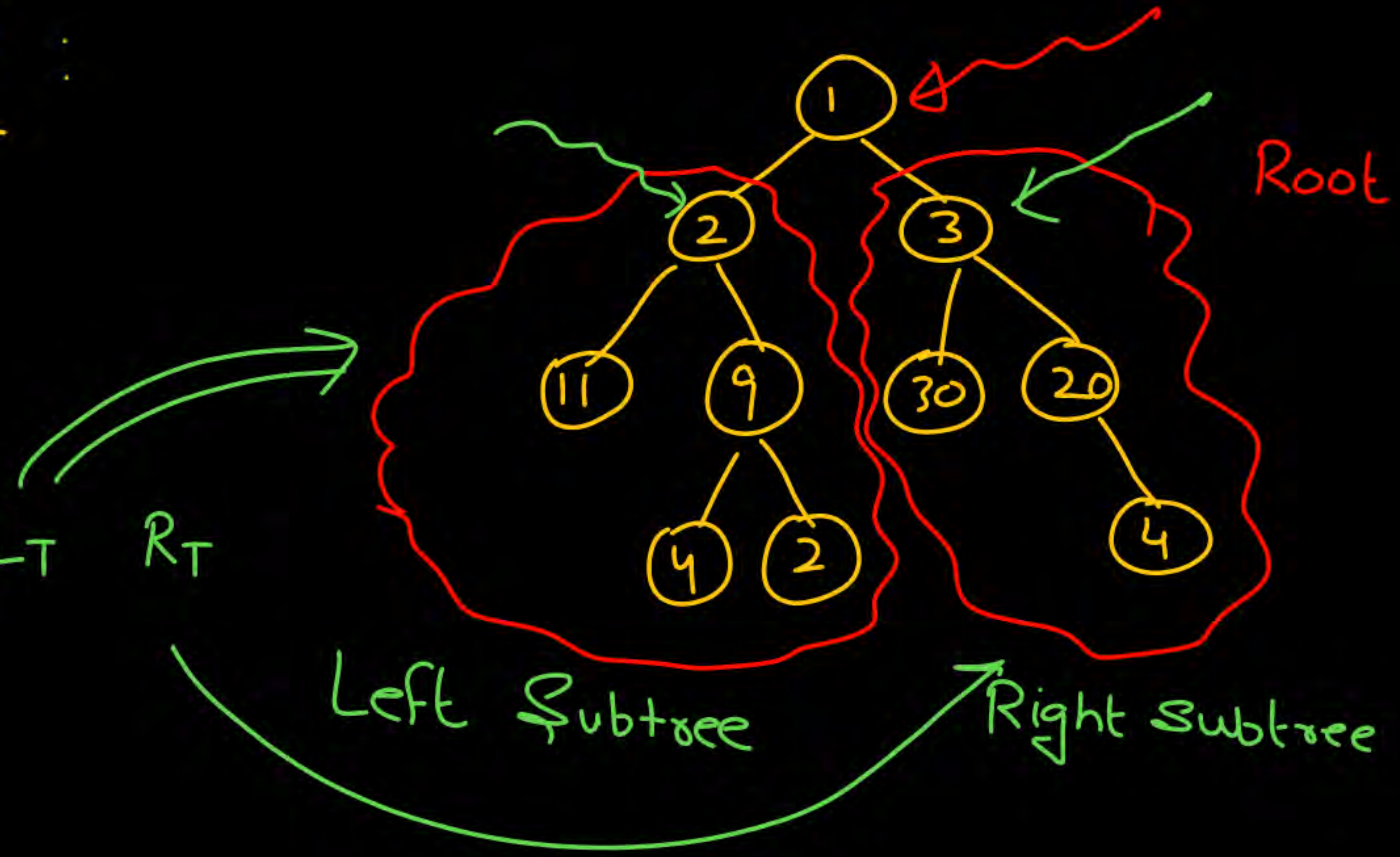
$$L + 18 = 31$$

$$\Rightarrow \boxed{L = 13}$$

Tree Traversal :

Root L_T R_T

1

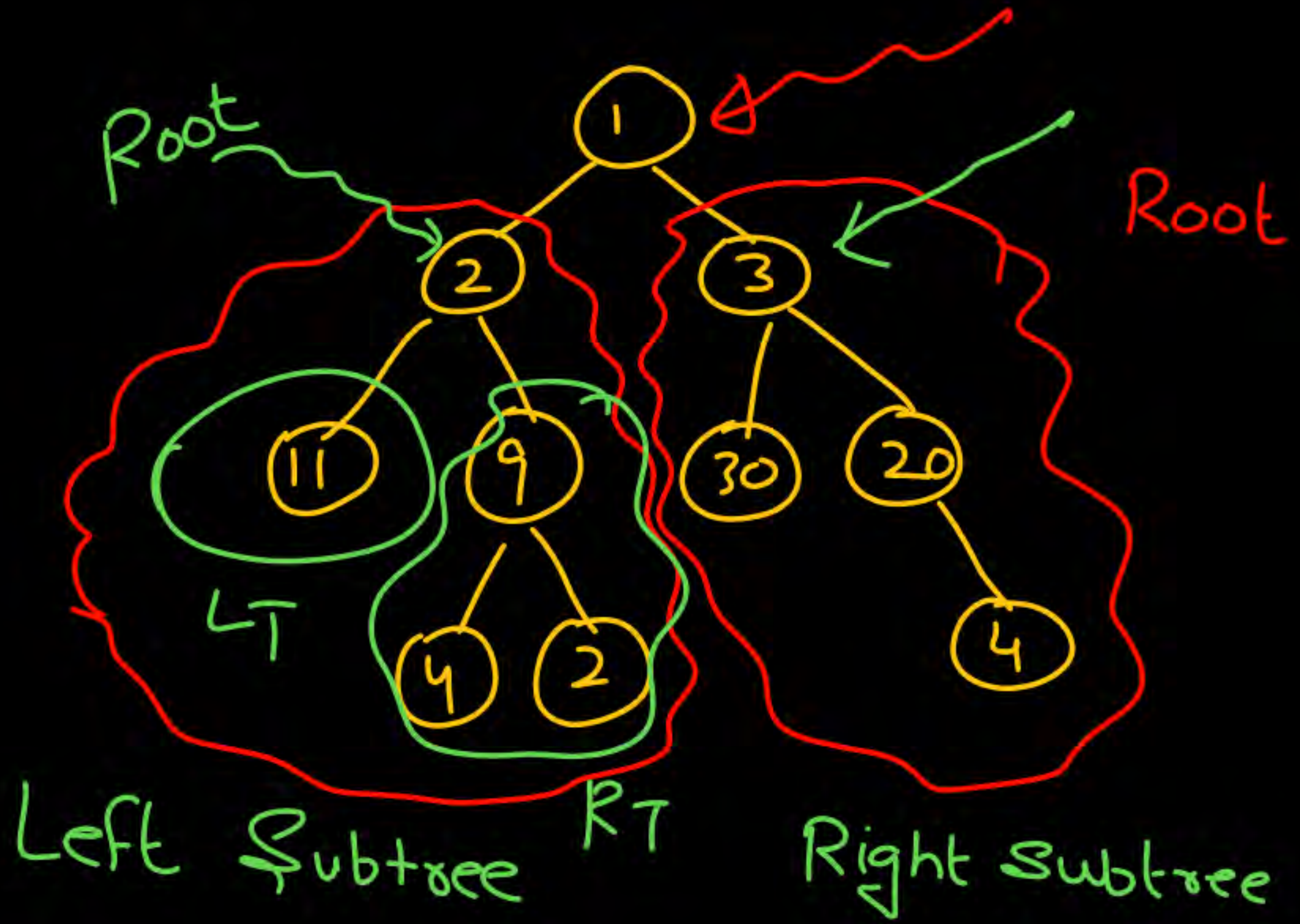


Tree Traversal :

1. Root

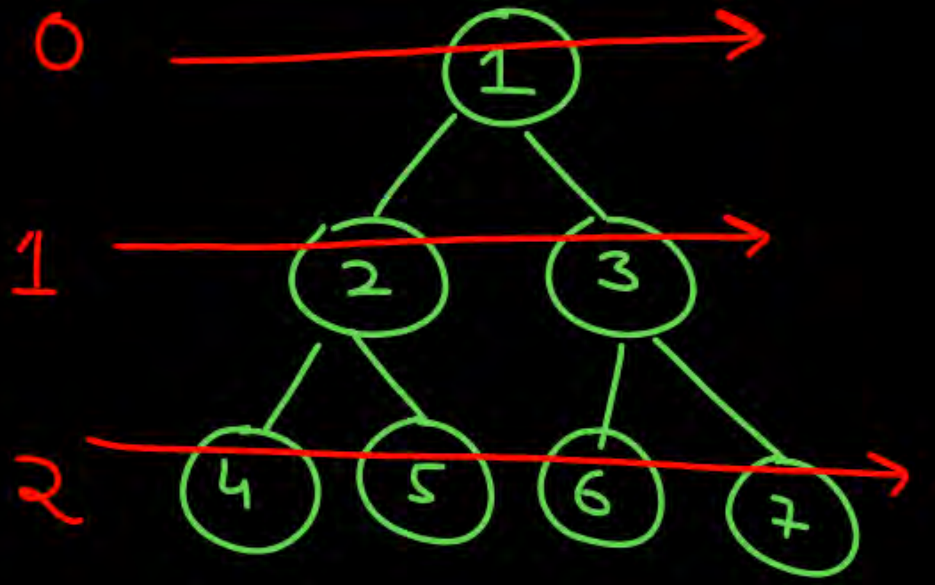
2. L_T

3. R_T



Traversal

Level order traversal



1 2 3 4 5 6 7

Depth order traversal

$R\ L_T\ R_T \Rightarrow 3! = 6$

1) $R\ \ L_T\ \ R_T$

2) $L_T\ \ R\ \ R_T$

3) $L_T\ \ R_T\ \ R$

✓ All these 3, L_T is traversed before R_T

4) $R\ \ R_T\ \ L_T$

5) $R_T\ \ R\ \ L_T$

6) $R_T\ \ L_T\ \ R$

1.] Root L_T R_T : Pre-order traversal

(i) Visit/Print the root node

(ii) Traverse the L_T (left subtree) of root node in Pre-order.

(iii) Traverse the R_T (Right subtree) of root node in Pre-order.

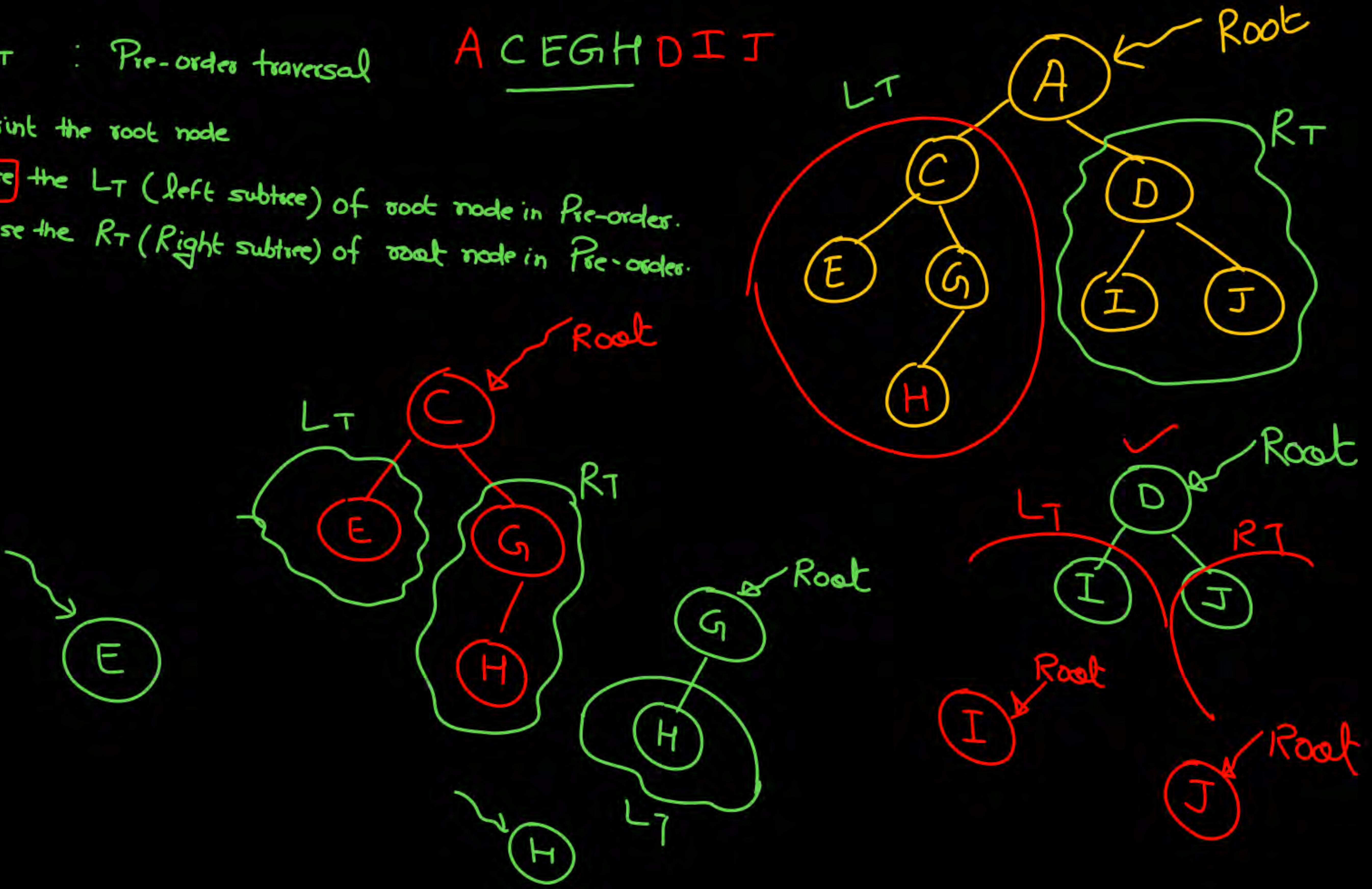
1] Root LT RT : Pre-order traversal

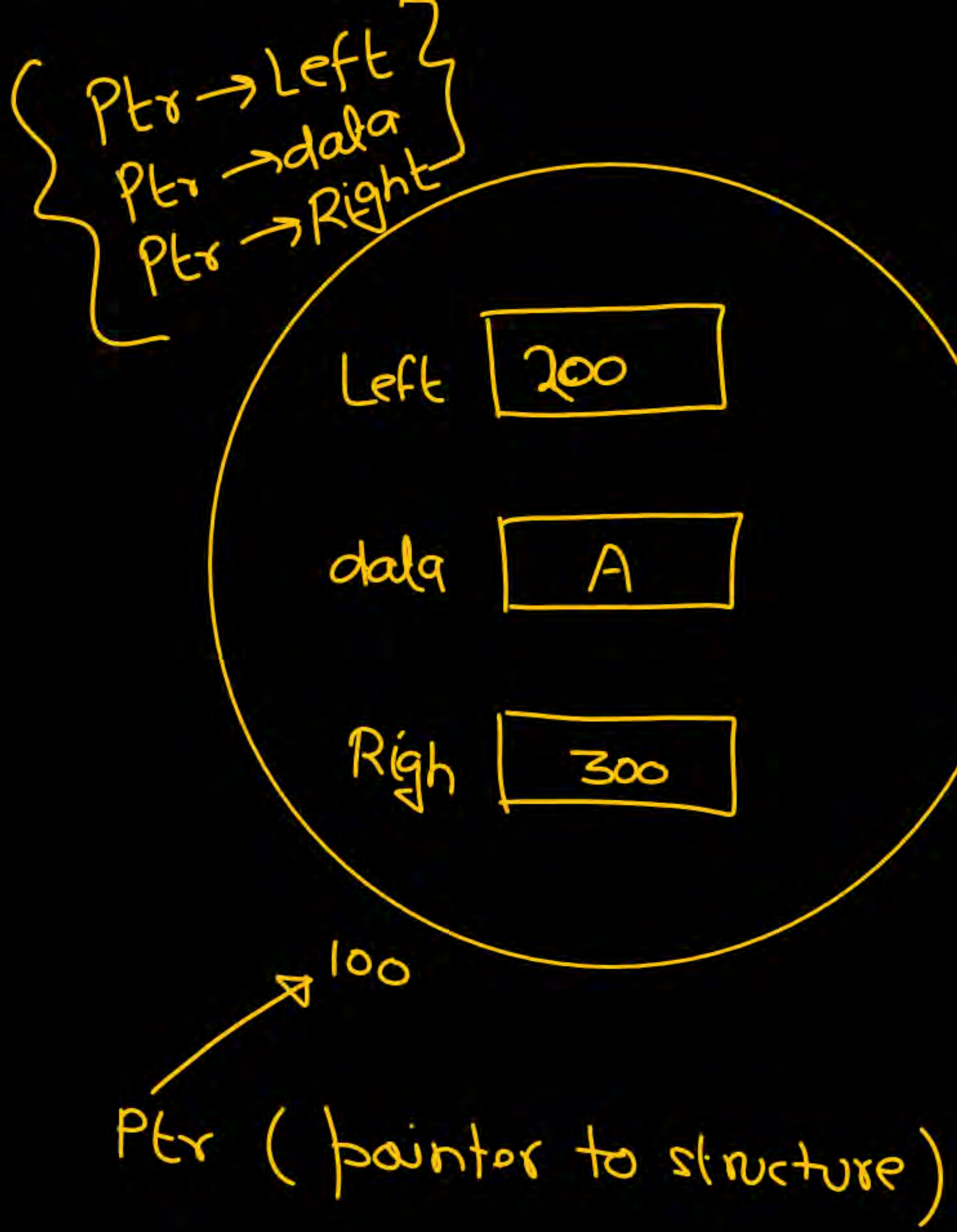
(i) Visit/Print the root node

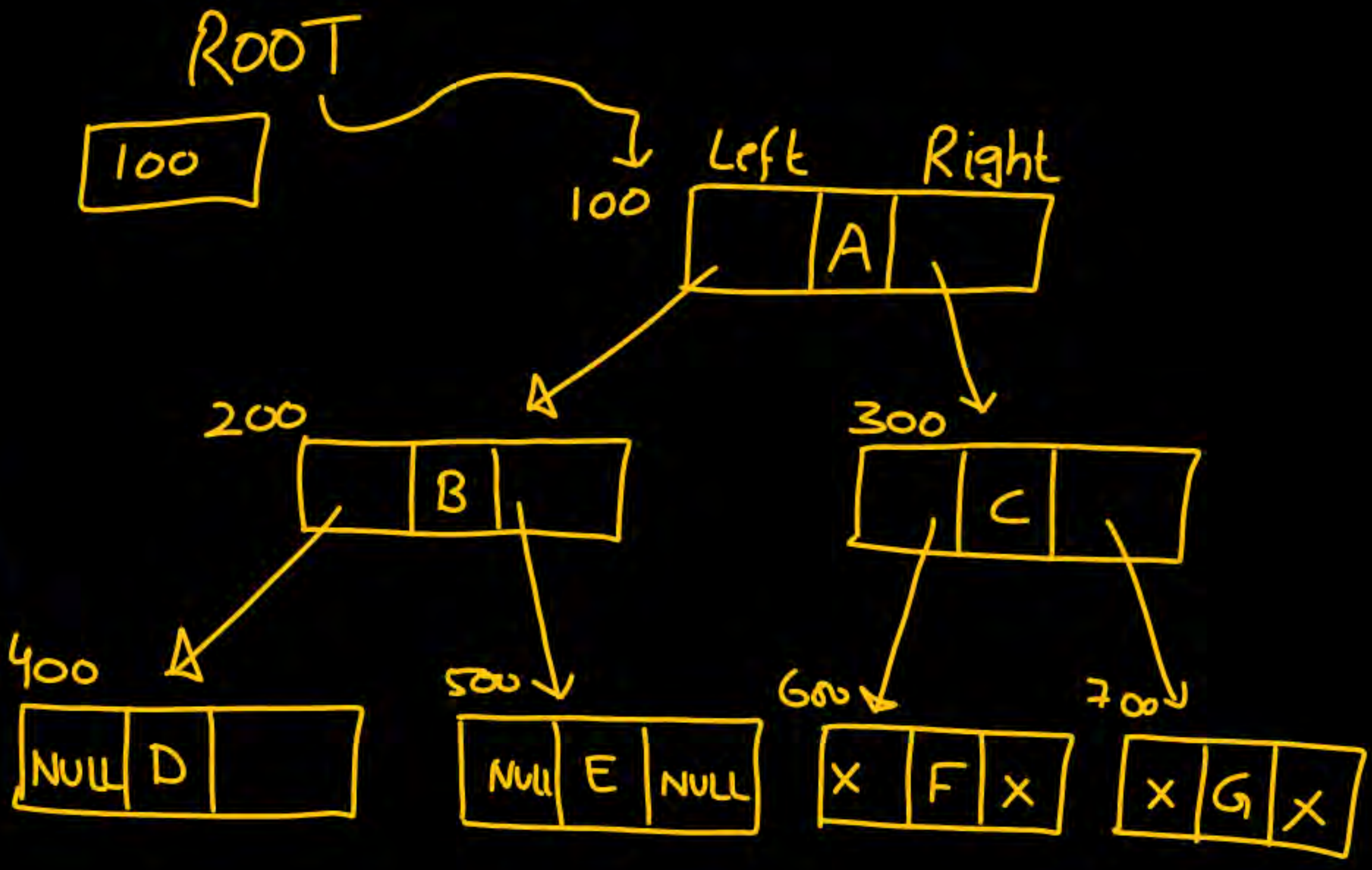
(ii) Traverse the LT (left subtree) of root node in Pre-order.

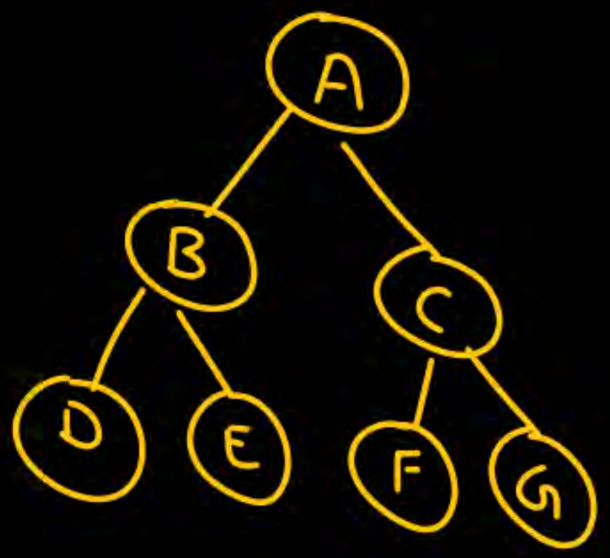
(iii) Traverse the RT (Right subtree) of root node in Pre-order.

A C E G H D I J

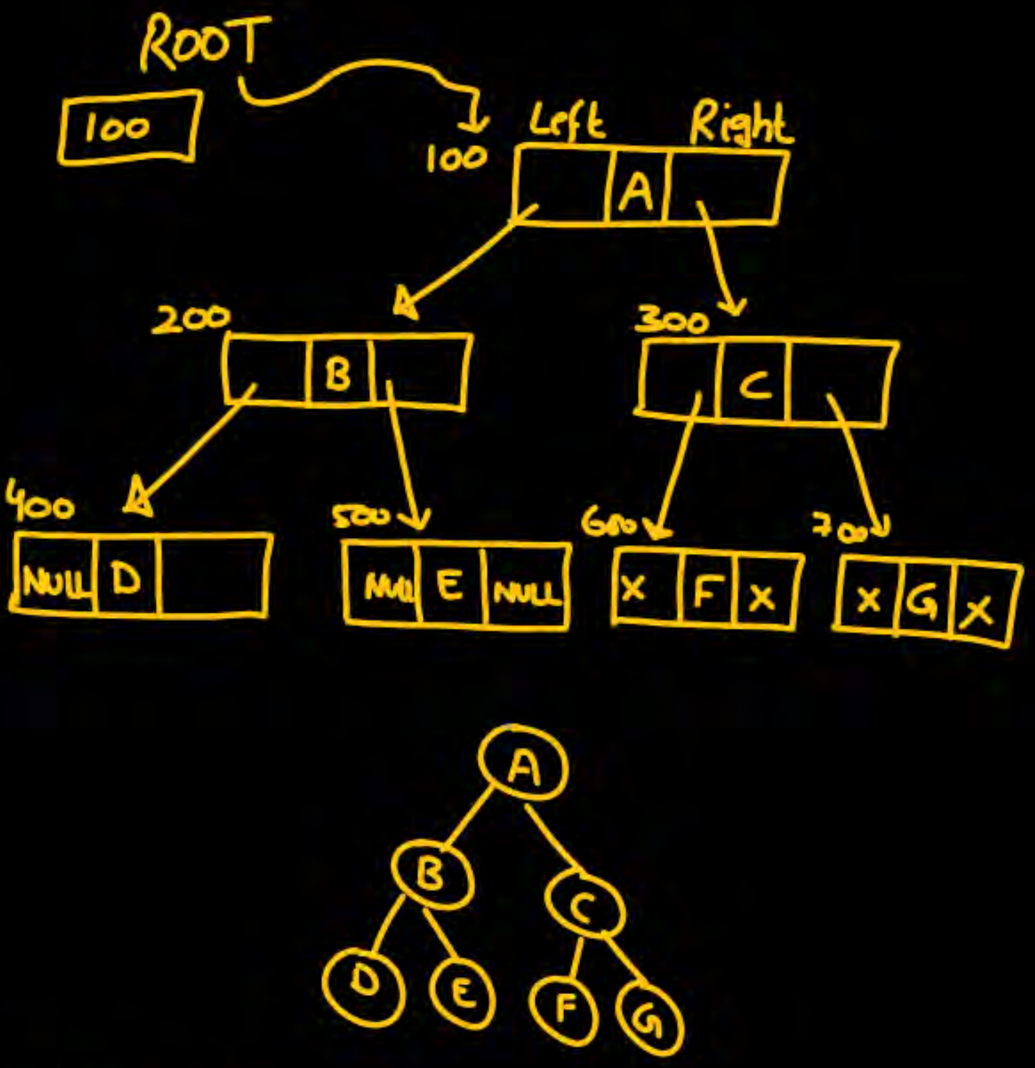



Ptr → Left
Ptr → data
Ptr → Right
Left
200
data
A
Righ
300
100
Ptr (pointer to structure)


ROOT
100
100
Left
A
Right
200
B
300
C
400
NULL
D
500
NULL
E
NULL
600
X
F
X
700
X
G
X


A
B
C
D
E
F
G

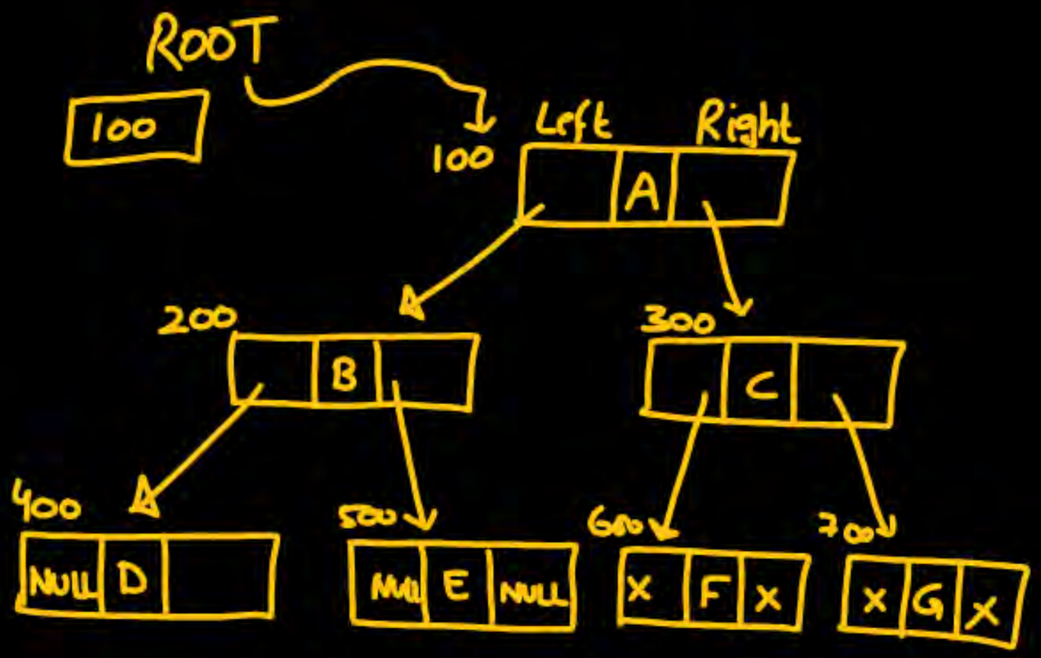
```
void Preorder( struct node *ptr)
  {



  }
```

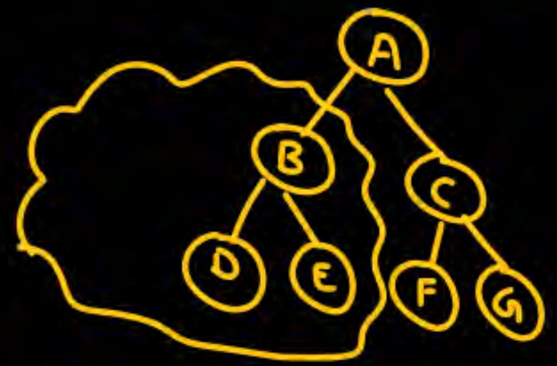


```
Void main()
  {
    =
  Preorder(ROOT);
    ―
    ―
    ―
  }
```

100 ⇐ Address of root node

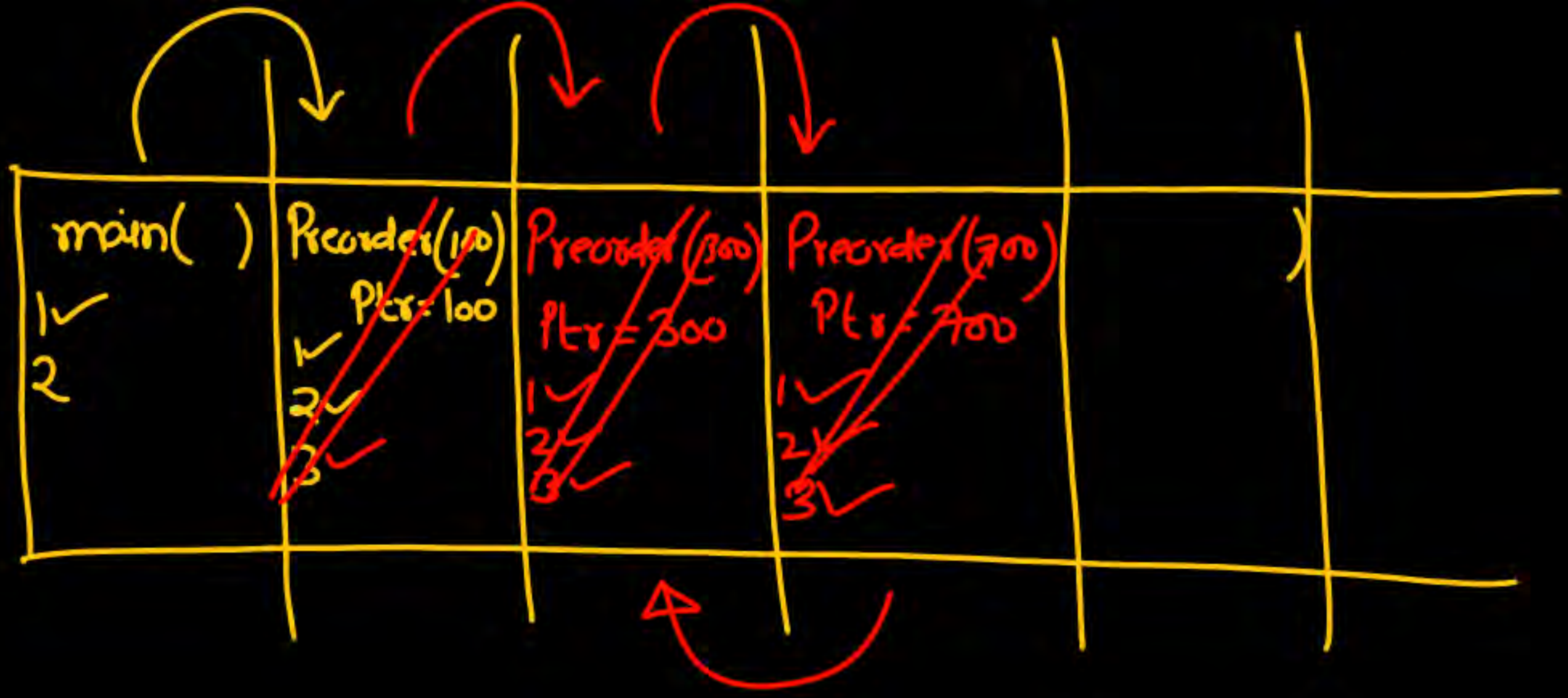
```
void Preorder( struct node *ptr)
    {
        if (Ptr = = NULL)
              return;

     1. printf(" %d", Ptr → data);
     2. Preorder( Ptr → Left   );
     3. Preorder ( Ptr → Right);

    }
```



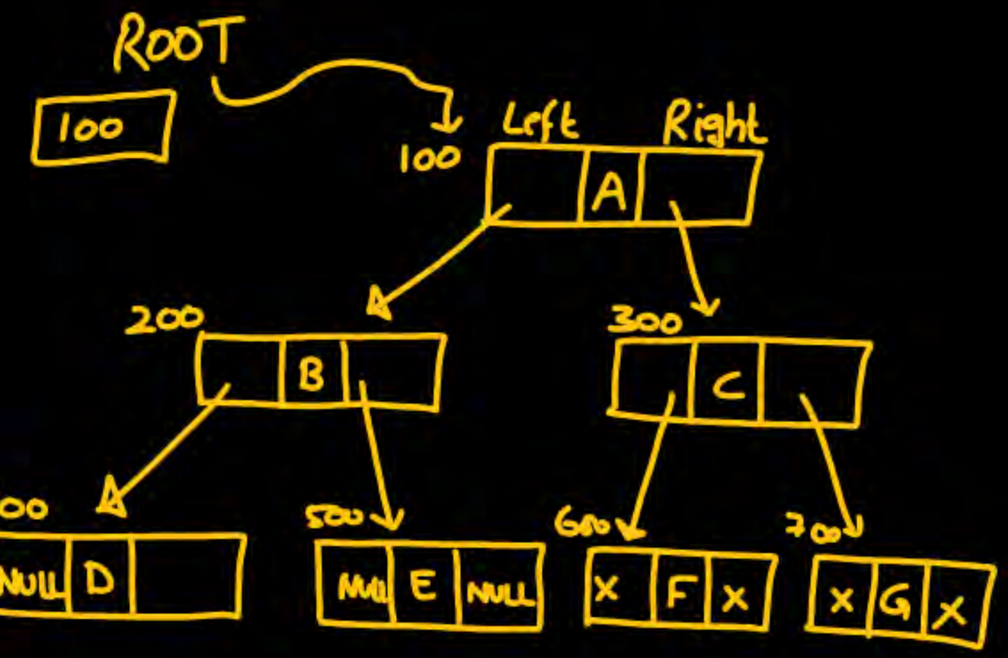


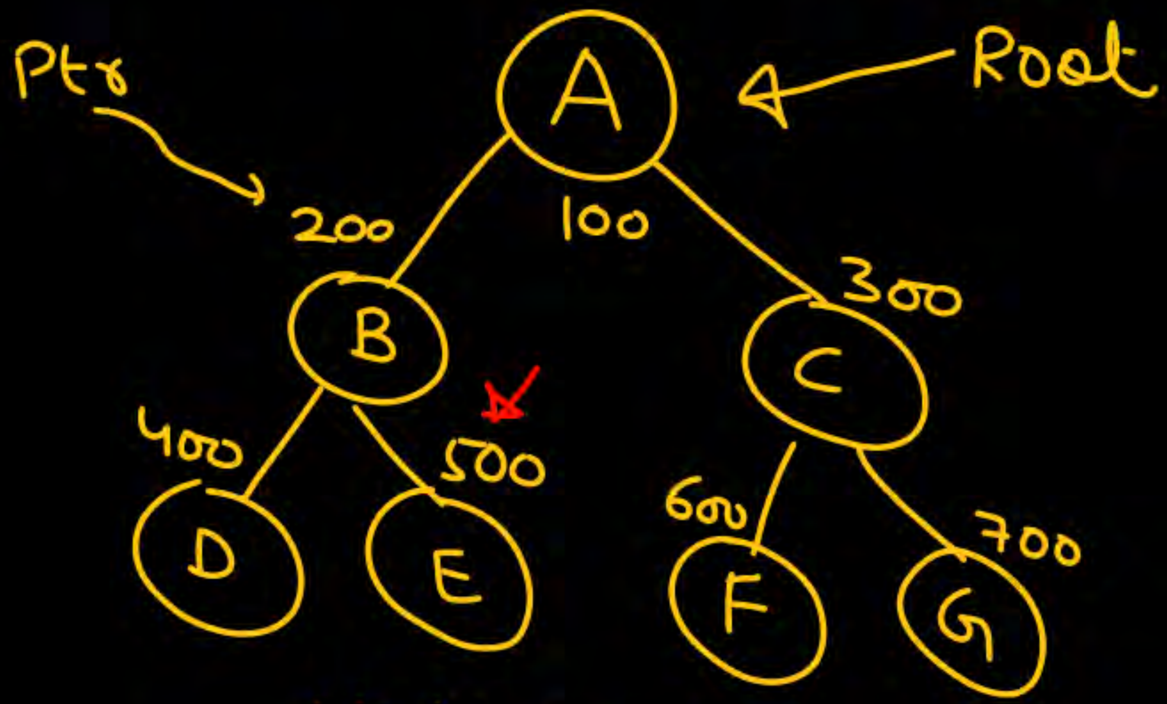
```
Void main( )
{
  =
  Preorder(ROOT);
  =
  =
  =
}
```

100 ⇐ Address of root node

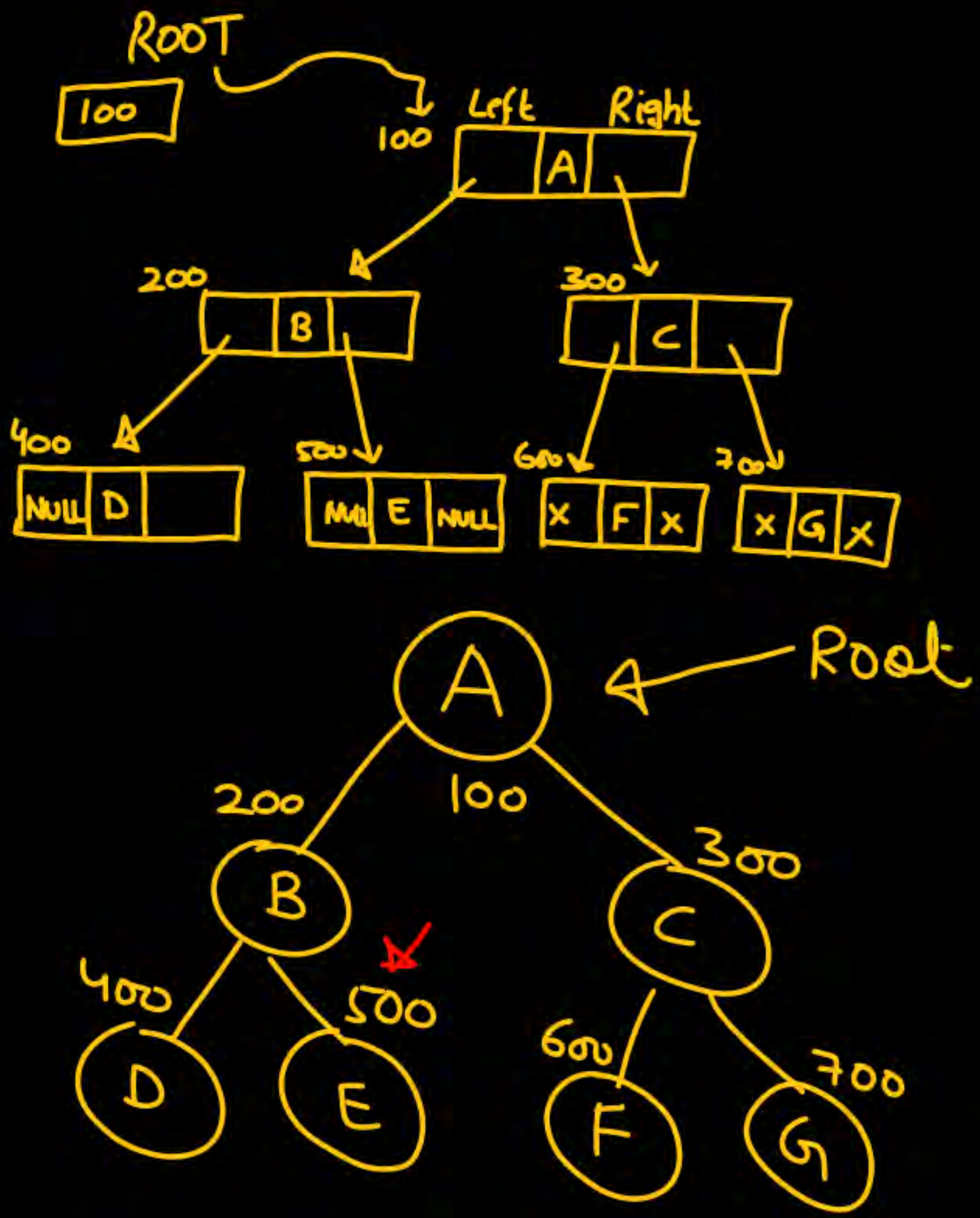


ABDECFG

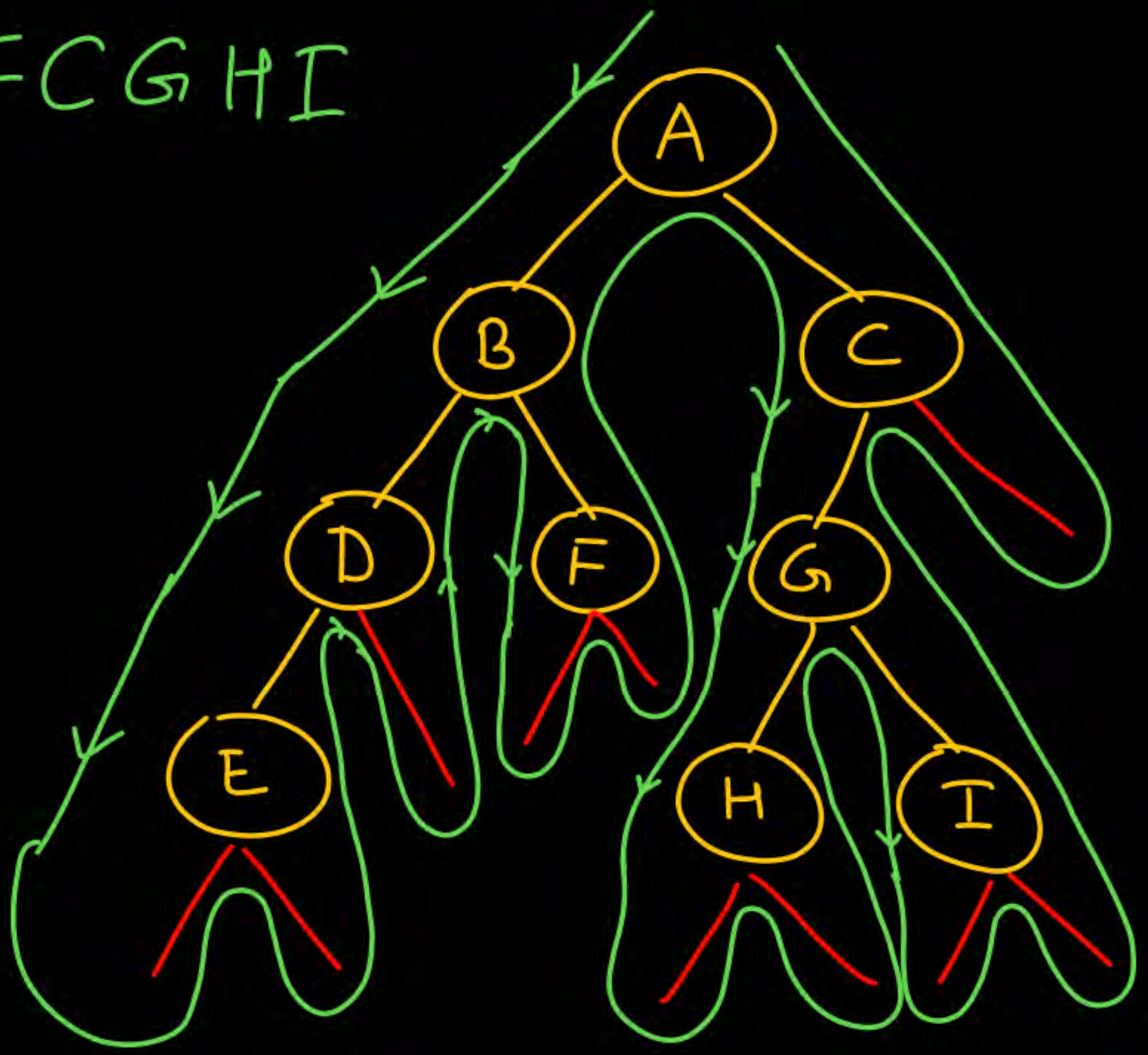




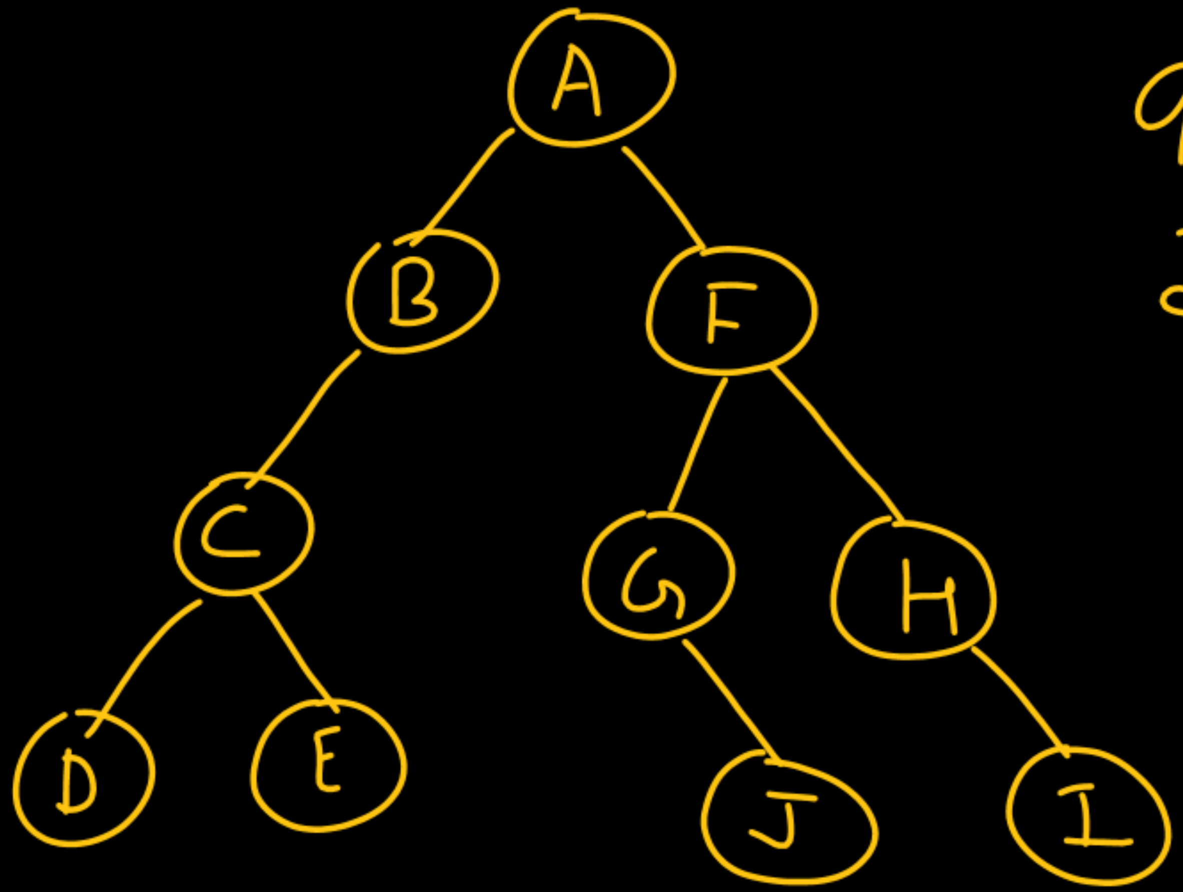
```
void main( )
{
1. Preorder(Root);
2. }
```


ROOT
100
100
Left
A
Right
200
B
300
C
400
NULL
D
500
NULL
E
NULL
600
x
F
x
700
x
G
x
A
Root
200
100
300
B
C
400
500
600
700
D
E
F
G

ABDEFCGHI



Pre :



97% ⇒ Coding

3% ⇒

A B C D E F G J H I

In-Order Traversal

$L_T \, R \, R_T$

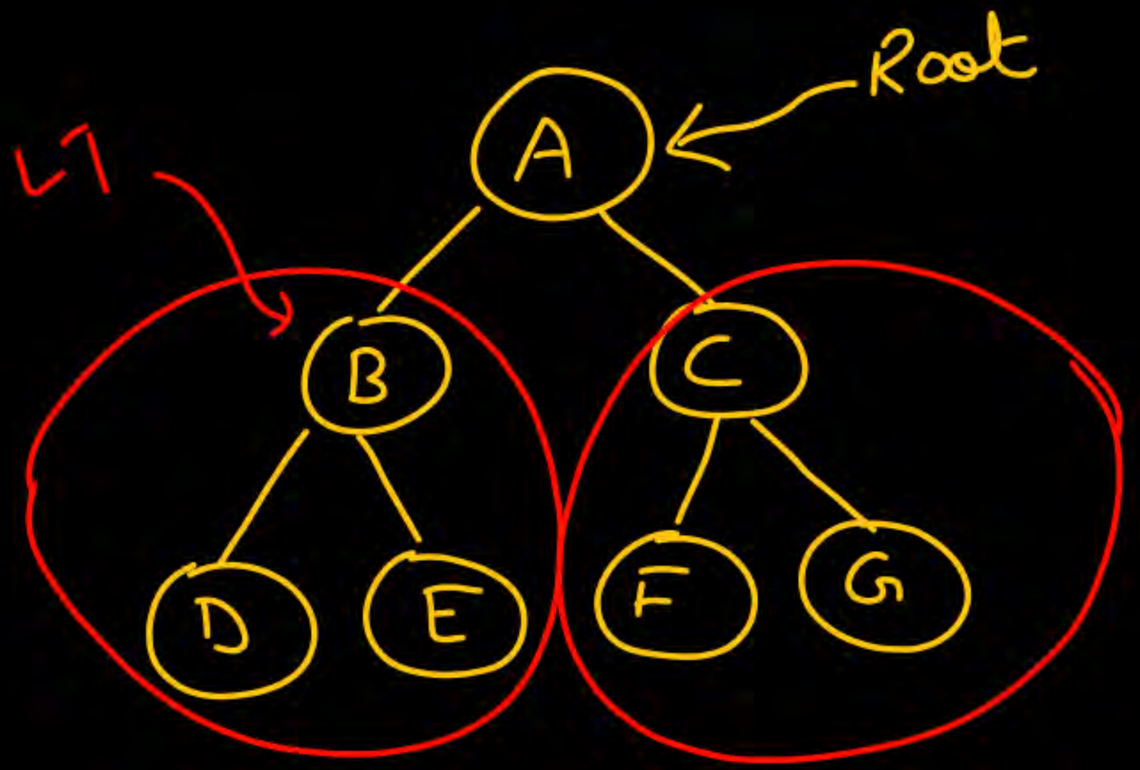
Inorder

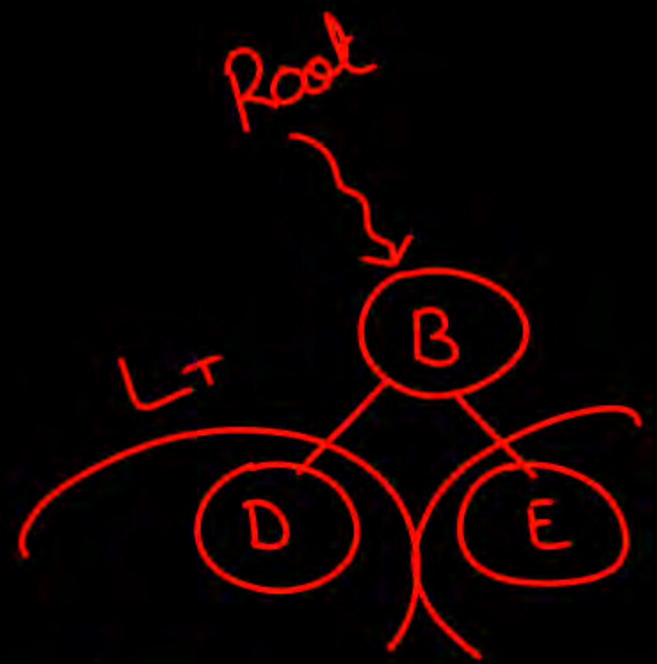
(i) Traverse the left subtree of root in In-order

(ii) Print/visit root node

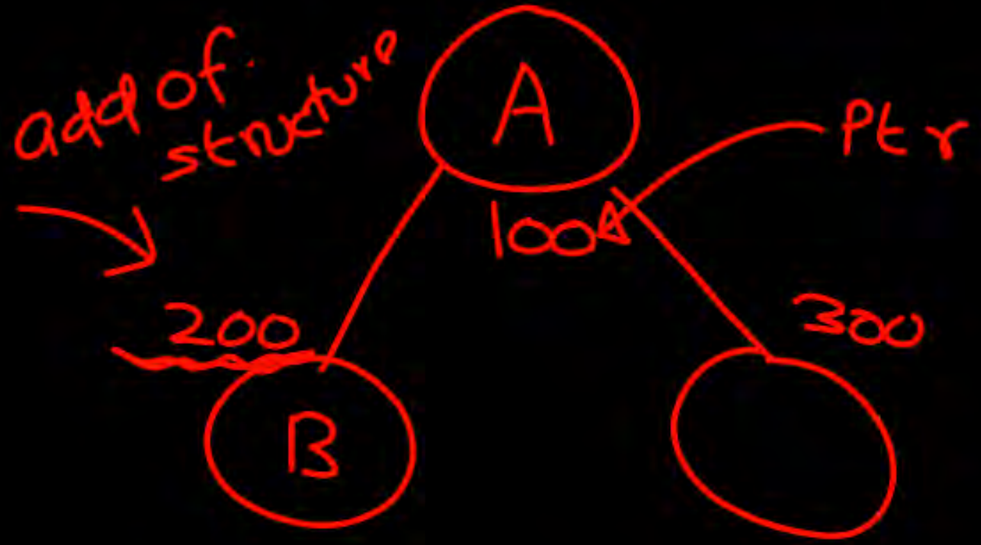
(iii) Traverse the right subtree of root in In-order.

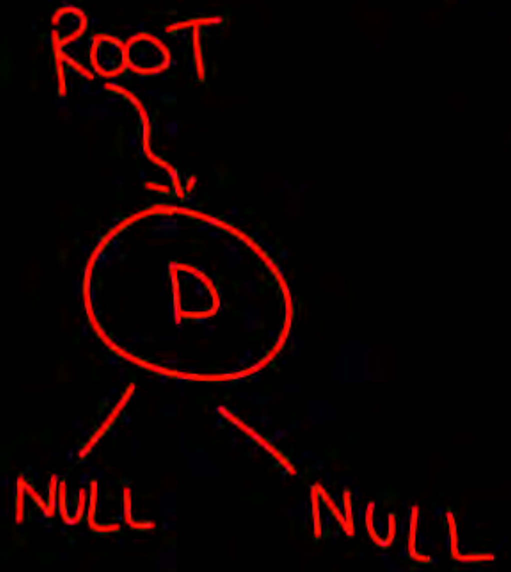
```
void  Inorder(struct node *Ptr)
   {
      if (Ptr = = NULL)
            return;

      Inorder(Ptr → Left);
      printf("%d", Ptr → data);
      Inorder(Ptr → Right);
   }
```



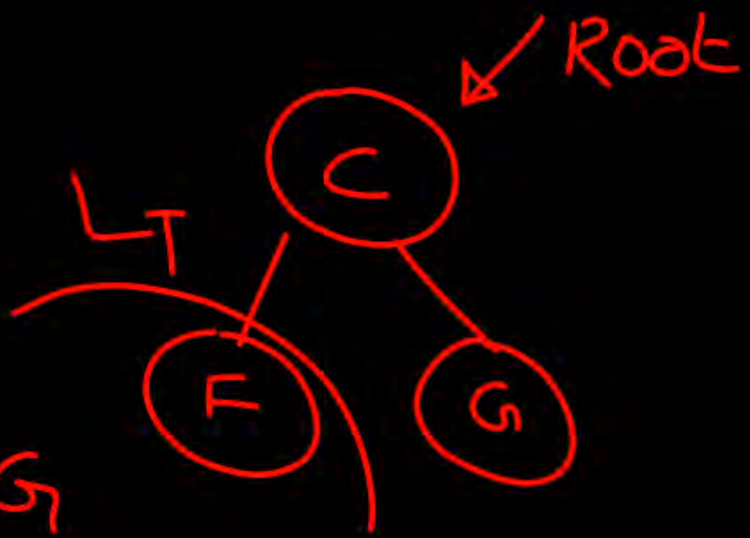


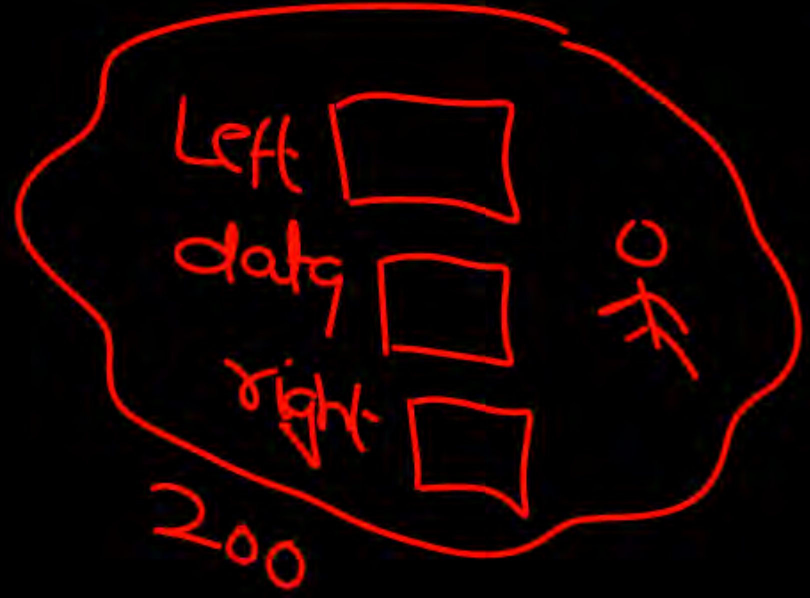
*(Ptr → left)

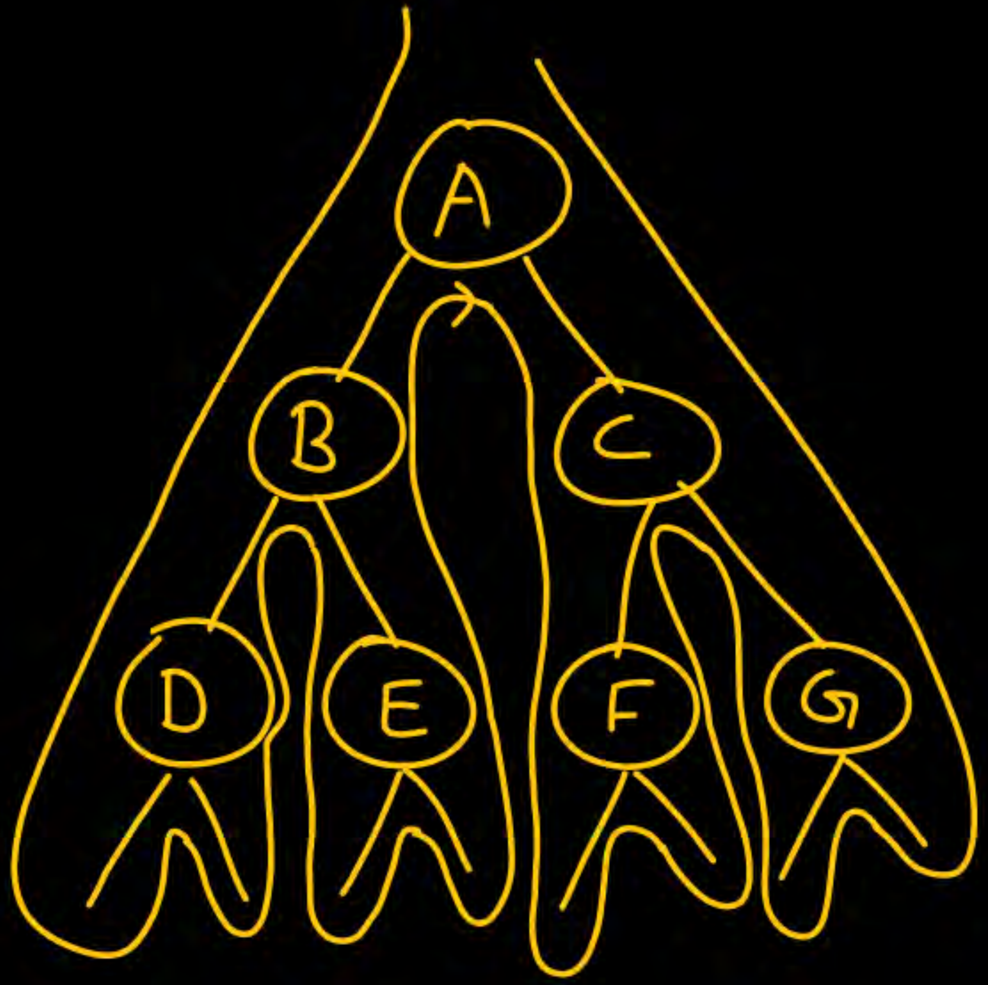


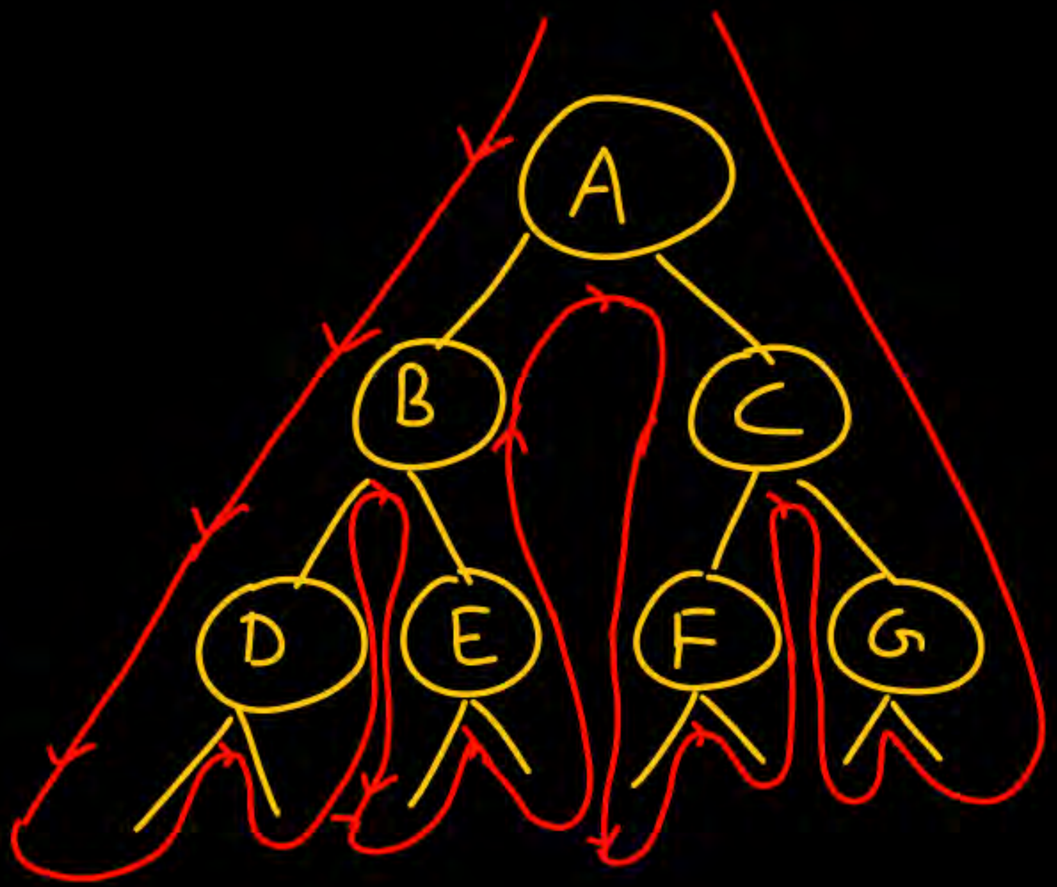


DBEAFCG









DBEAFCG

Post-Order traversal

$\overset{\checkmark}{L_T}, \overset{\checkmark}{R_T}$, Root

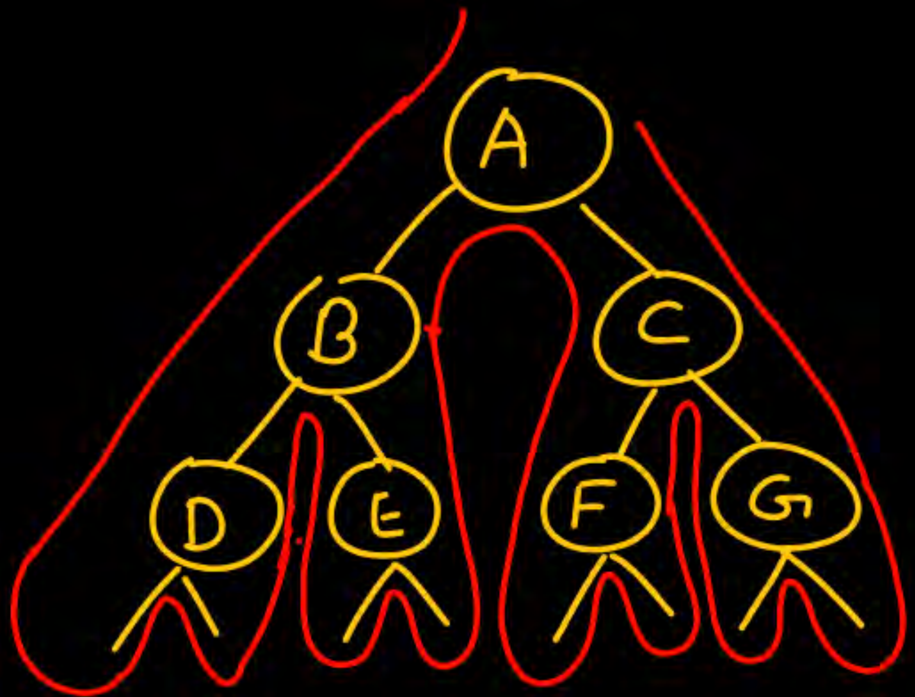
PostOrder

(i) Traverse L_T of root node in Postorder.

(ii) Traverse R_T of root node in Postorder.

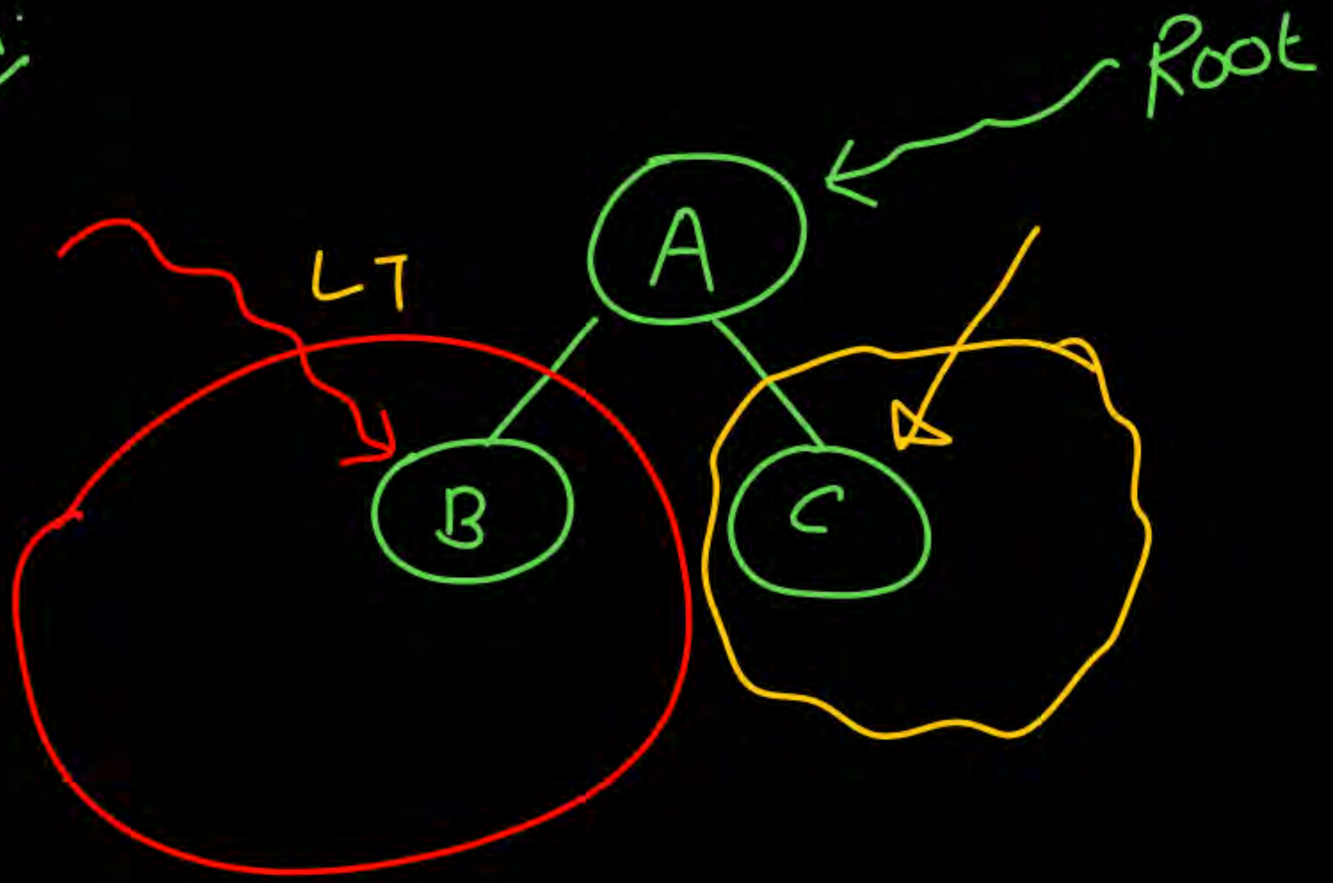
(iii) Print/visit the root node.

```
void Postorder(struct node *Ptr)
     {
        if (Ptr == NULL)
              return;
        Postorder( Ptr → Left );
        Postorder ( Ptr → Right);
        printf("%d", Ptr → data);
     }
```



DEBFGCA

Ex1.



ABC

THANK YOU GW SOLDIERS !