



# **SNS COLLEGE OF TECHNOLOGY**

**Coimbatore-35**

**An Autonomous Institution**

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

**COURSE NAME : 23ITT102\_Programming in C and Data Structure**

**I YEAR/ ISEMESTER**

**UNIT V**

**Topic: Tree ADT & Binary Tree**



# Tree Terminologies

A tree data structure is defined as a collection of objects or entities known as nodes that are linked together to represent or simulate hierarchy.

A tree data structure is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.

TREES

Node → Element of a Tree  
A, B, C, D, E, F, G, H

ROOT NODE → STARTING NODE OF TREE  
A is a ROOT NODE  
TREE will have only one Root

EDGE → Link (or) Connection between two nodes  
 $N - \text{nodes} \Rightarrow (N - 1) \text{ Edges}$   
 $N = 8 \text{ nodes}$   
 $E = 7 \text{ Edges}$

PARENT — Node with branches from top to bottom  
A, B, E are parent nodes

```
graph TD; A((A)) --- B((B)); A --- C((C)); B --- D((D)); B --- E((E)); E --- G((G)); E --- H((H)); B --- F((F));
```



# Tree Terminologies

TREES

CHILD — NODE with Edge from bottom to top (or) Branches of parent  
B, C, D, E, F, G, H — childs

SIBLINGS — CHILD nodes of same Parent node  
B, C — siblings  
D, E, F — siblings  
G, H — siblings

LEAF — Node without child node  
C, D, F, G, H

```
graph TD; A((A)) --- B((B)); A --- C((C)); B --- D((D)); B --- E((E)); B --- F((F)); E --- G((G)); E --- H((H));
```



# Tree Terminologies

Leaf nodes  
Node with child nodes

A, B, E — Internal nodes

```
graph TD; A((A)) --- B((B)); A --- C((C)); B --- D((D)); B --- E((E)); E --- G((G)); E --- H((H)); C --- F((F)); F --- Bottom[Bottom]
```

DEGREE — Number of child nodes represents degree of a node

Degree (A) — 2  
Degree (B) — 3

Maximum Degree among all nodes — Degree of Tree  
Degree of Tree — 3



# Tree Terminologies

TREES

HEIGHT - Longest path from Leaf node to that node is height.

Height(B) - 2  
Height(A) - 3

DEPTH - Longest path from Root node to that node

Depth(E) - 2    Depth(B) - 1  
Depth(G) - 3

```
graph TD; A((A)) --- B((B)); A --- C((C)); B --- D((D)); B --- E((E)); E --- G((G)); E --- H((H));
```



# Tree Terminologies

TREES

PATH - Sequence of nodes from ROOT to Leaf

Path (A to G) → A-B-E-G

SUBTREE - Node with child node forms subtree

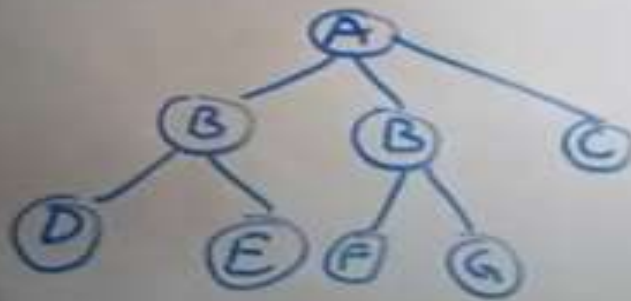


# Binary Tree

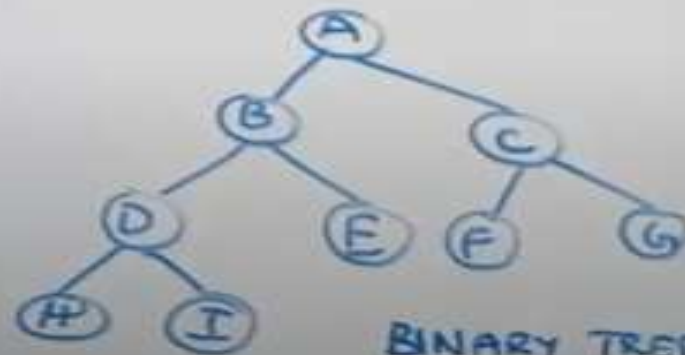
→ Every node in a tree should have atmost 2 children

Atmost 2 — 0 nodes  
1 nodes  
2 nodes

0, 1, 2



TREE



BINARY TREE



# *Binary Tree*

- Tree in which no node can have more than two children

## **Node declaration**

```
Struct TreeNode
```

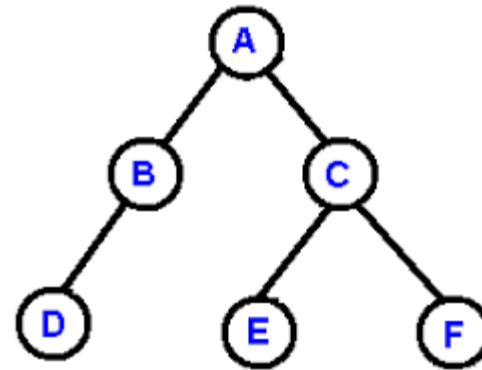
```
{
```

```
Int Element;
```

```
Struct TreeNode *Left;
```

```
Struct TreeNode *Right;
```

```
}
```







# *Binary Tree*

→ Every node in a tree should have at most 2 children

## Types of Binary Trees

→ Full Binary Tree / Strictly Binary Tree

→ Almost Complete Binary Tree / Incomplete Binary Tree

→ Complete Binary Tree / Perfect Binary Tree

→ Left Skewed Binary Tree

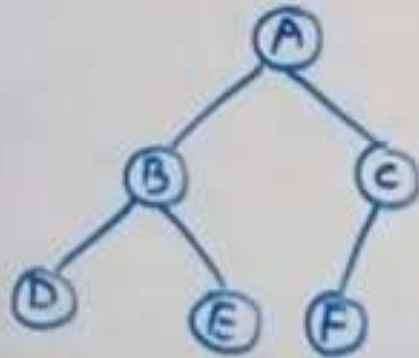
→ Right Skewed Binary Tree



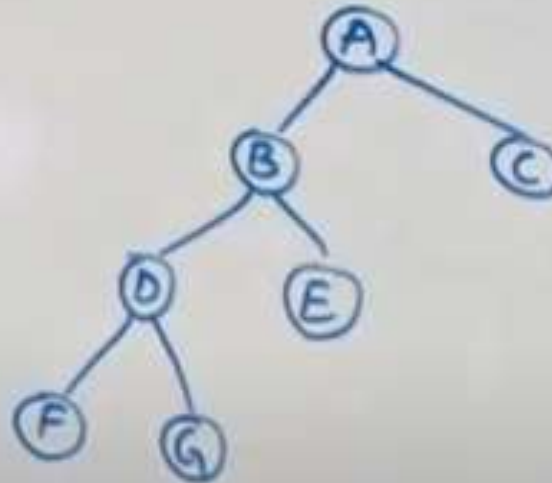
# Types of Binary Tree

→ Full Binary / Strictly Binary

Every node must have two children except the leaf nodes



Binary Tree

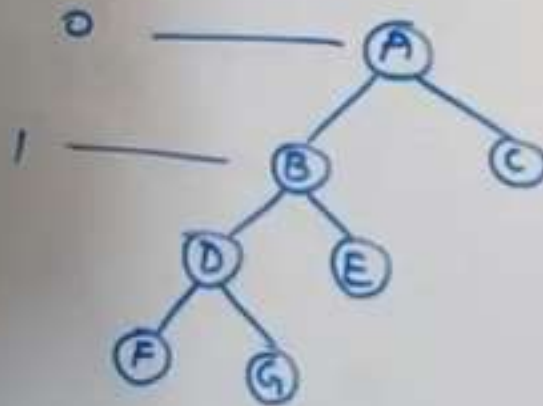




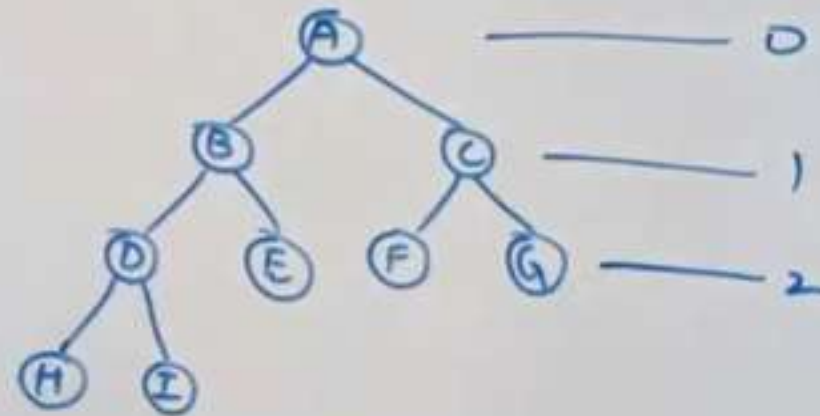
# Types of Binary Tree

→ Incomplete Binary / Almost Complete

Every node must have two children in all levels except in last level but filled from left to right



Strictly Binary



Incomplete

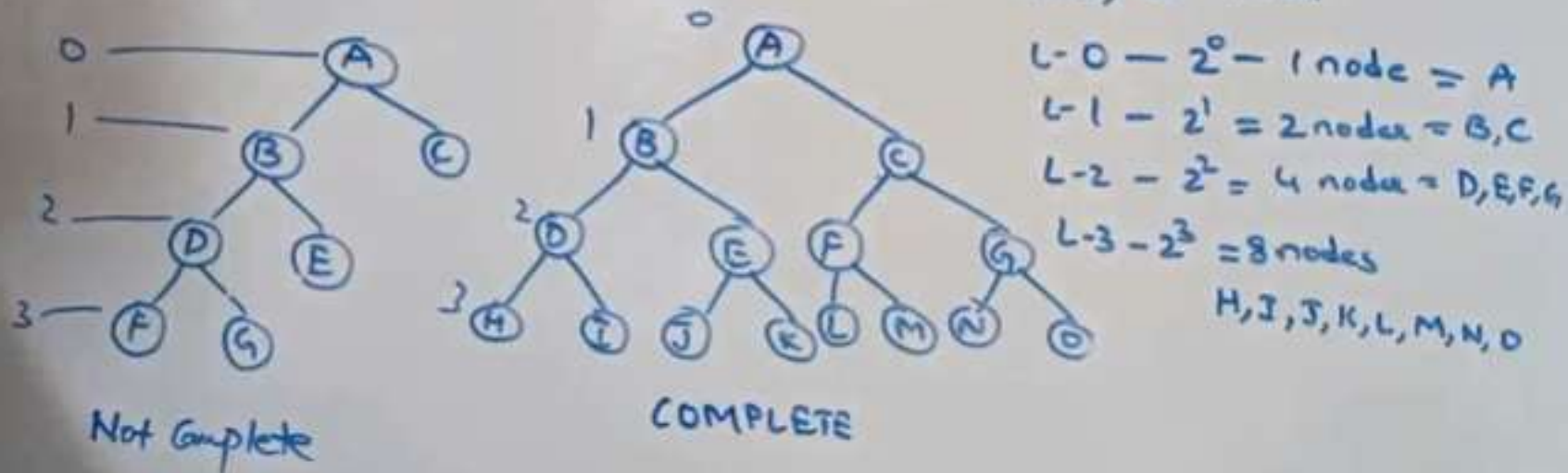


# Types of Binary Tree

→ Complete Binary / Perfect Binary

Every node must have two children in all the levels.

Each level there must be  $2^L$  nodes, L-level

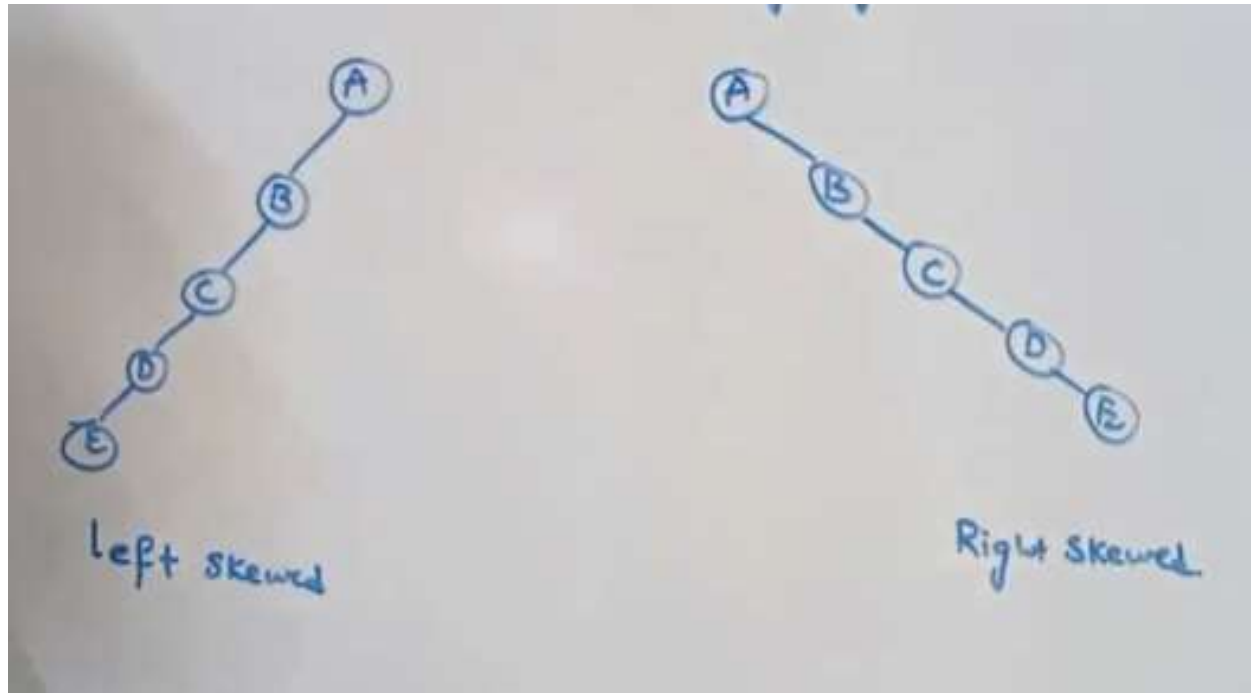




# Types of Binary Tree

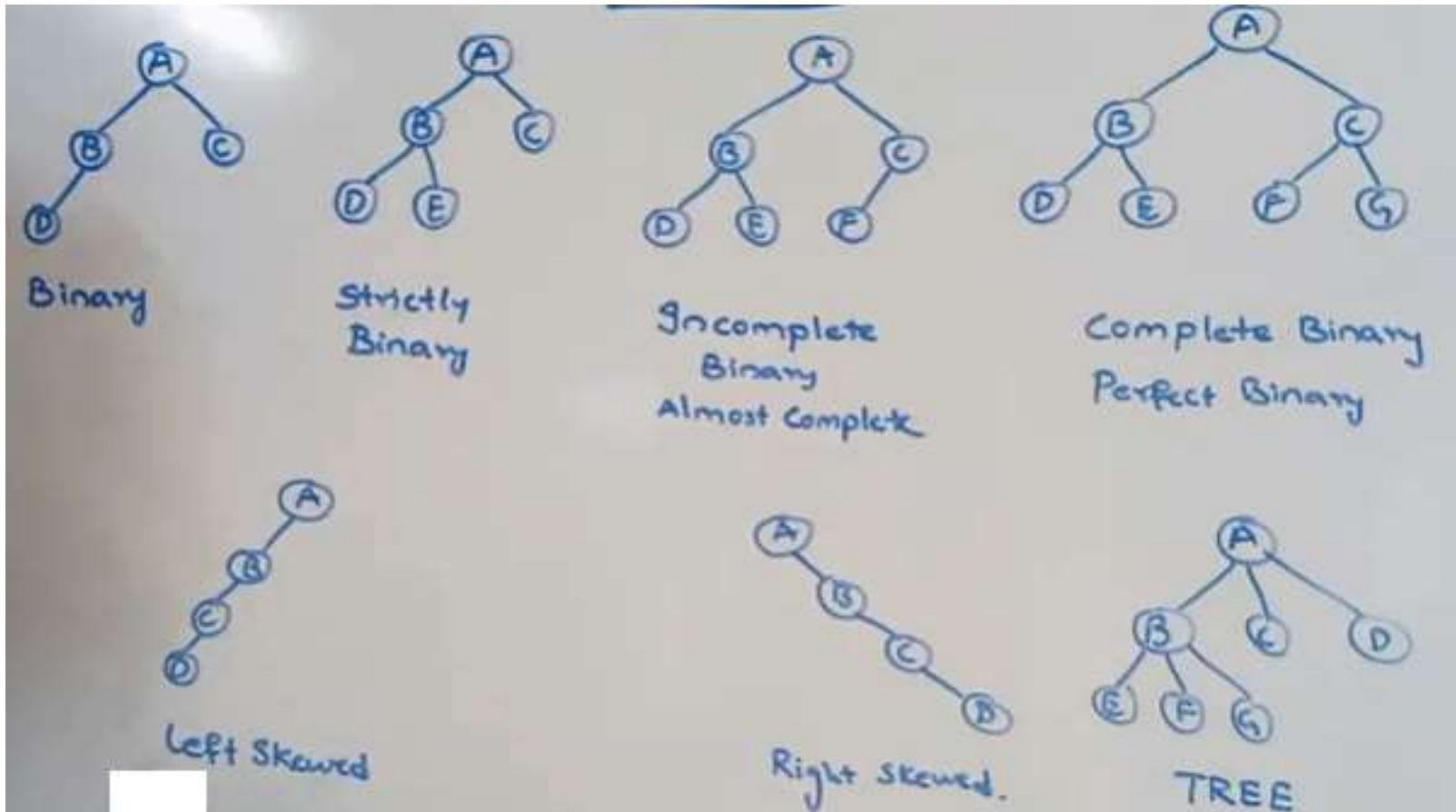
Left Skewed Tree: Every node should have only left Children

Right Skewed Tree : Every node should have only Right Children





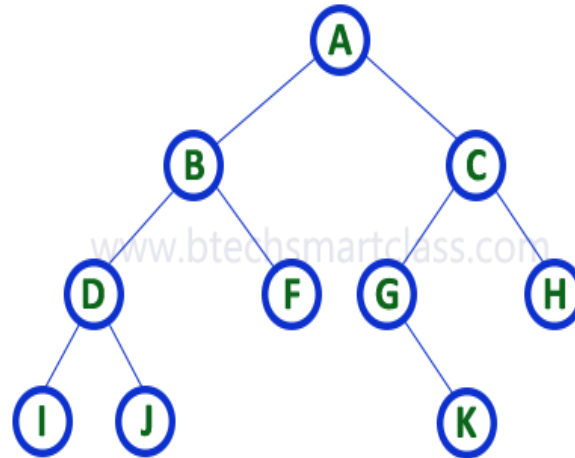
# Types of Binary Tree





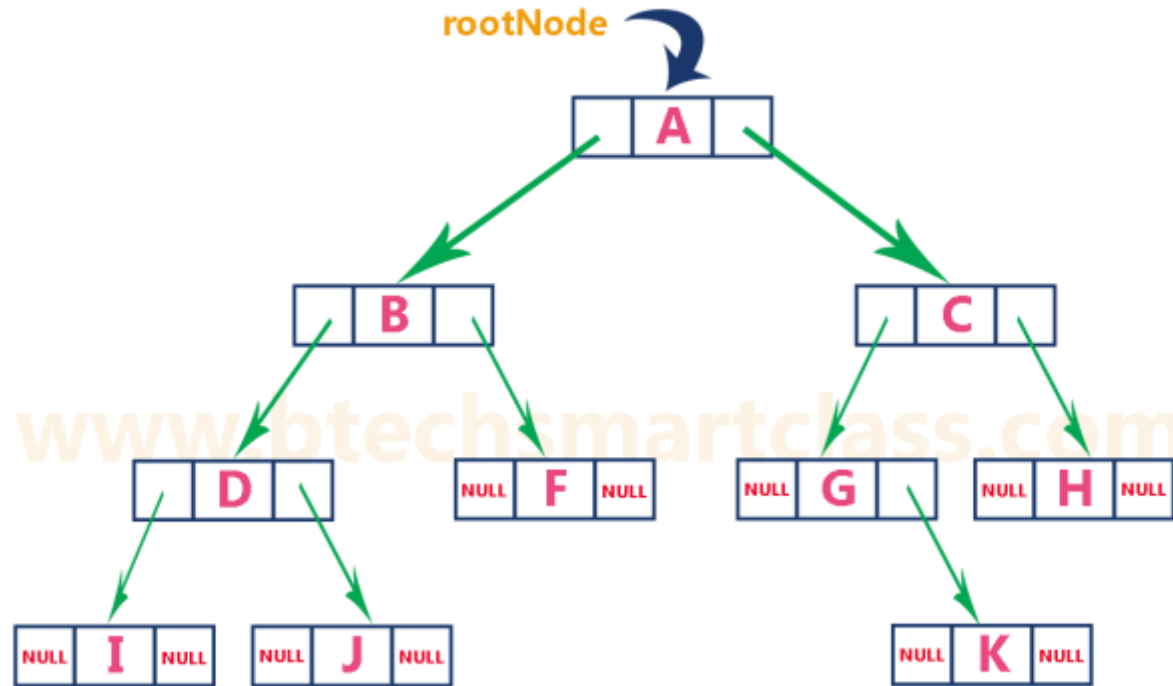
# Representation of Binary Tree -Linear Representation

- For any element in position  $i$ , the left child is in position  $2i$ , the right child is in position  $2i+1$  and the parent is in position  $(i/2)$





# Linked List Representation

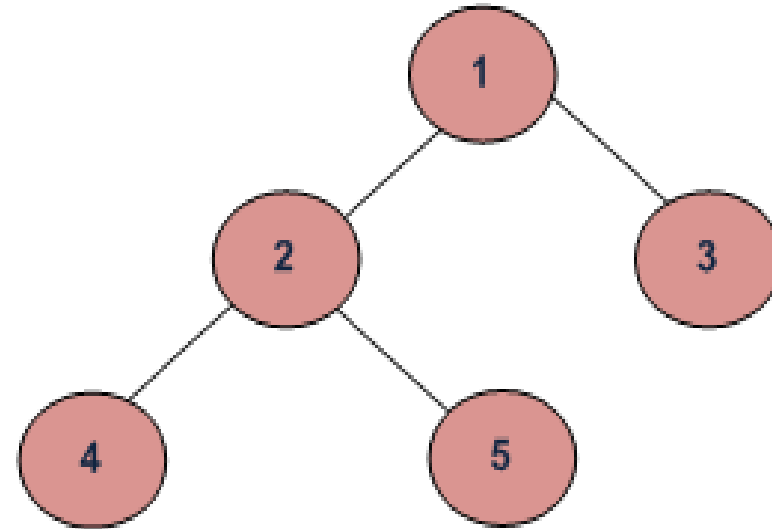






# *Tree Traversals*

- Visiting each node only once
- Three types of tree traversals
  - Inorder Traversal
  - Preorder Traversal
  - Postorder Traversal





# Tree Traversals

TREE TRAVERSALS

```
graph TD; A((A)) --- B((B)); A --- C((C));
```

① Inorder Traversal

LEFT CHILD — ROOT NODE — RIGHT CHILD

B A C

② Pre order Traversal

ROOT NODE — LEFT CHILD — RIGHT CHILD

A B C

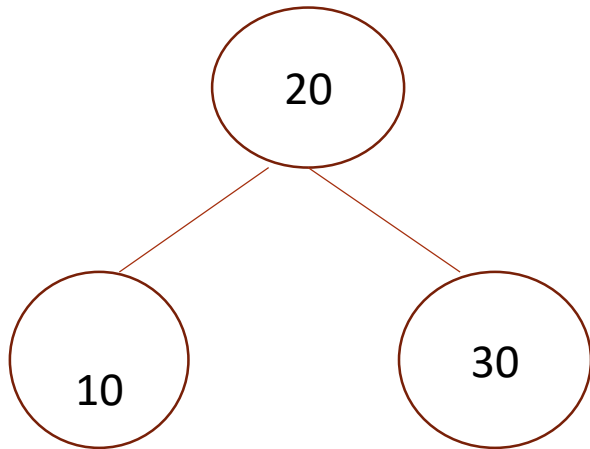
③ Post order Traversal

LEFT CHILD — RIGHT CHILD — ROOT NODE

B C A



## *Examples*



- Inorder 10 20 30
- Preorder 20 10 30
- Postorder 10 30 20



# *Inorder Traversal*

## **Routine**

Traverse the left subtree in inorder

Visit the root

Traverse the right subtree in inorder

```
Void Inorder(Tree T)
{
If(T!=NULL)
{
Inorder(T->left);
printElement((T->Element);
Inorder(T->right);
}
}
```



## *Preorder Traversals*

Visit the root

Traverse the left subtree in  
preorder

Traverse the right subtree in  
preorder

```
Void Preorder(Tree T)
{
If(T!=NULL)
{
printElement((T->Element);
Preorder(T->left);
Preorder(T->right);
}
}
```



## *Post order Traversals*

Traverse the left subtree in  
preorder

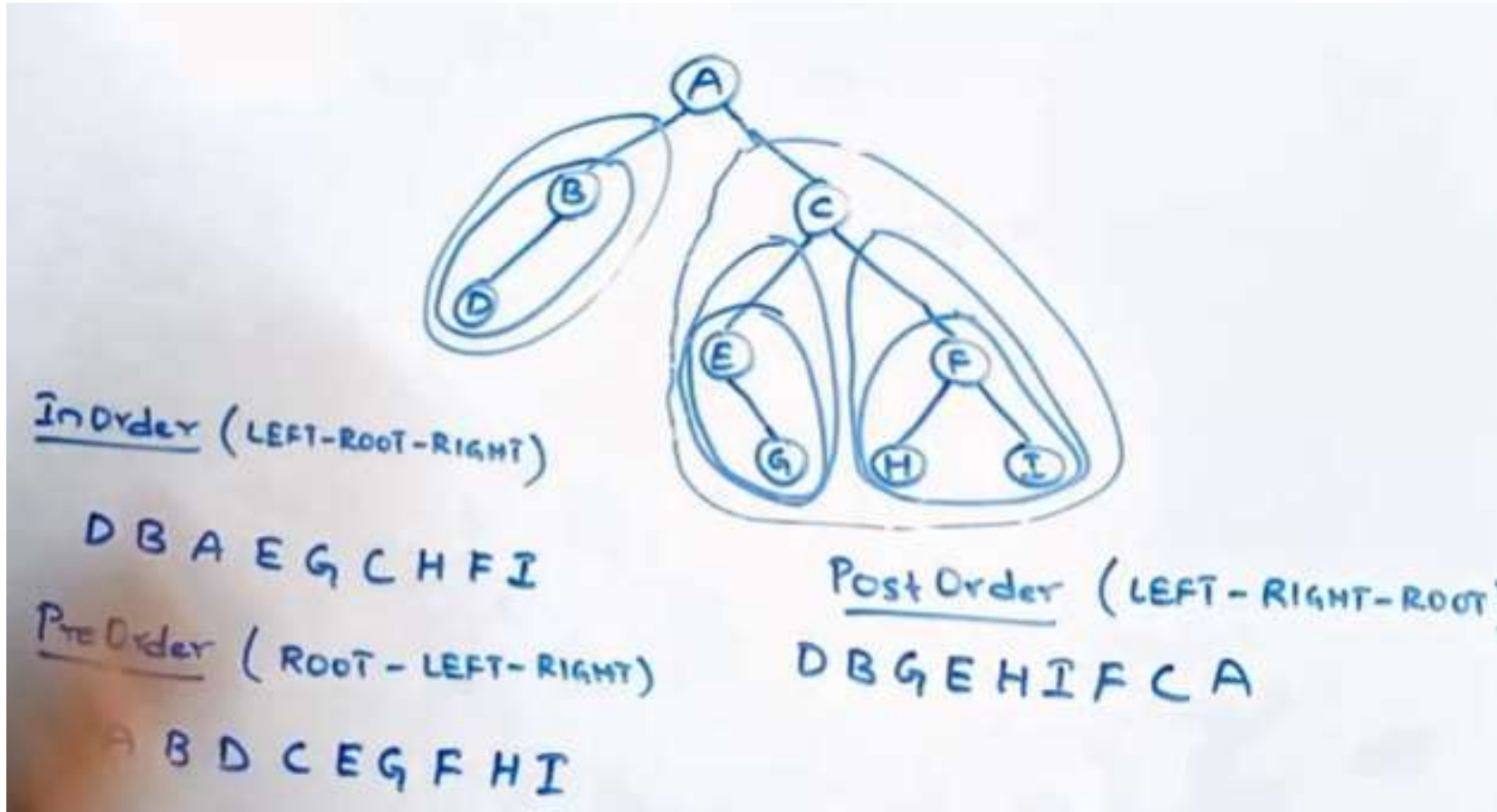
Traverse the right subtree in  
preorder

Visit the root

```
Void Postorder(Tree T)
{
If(T!=NULL)
{
Postorder(T->left);
Postorder(T->right);
printElement((T->Element));
}
}
```



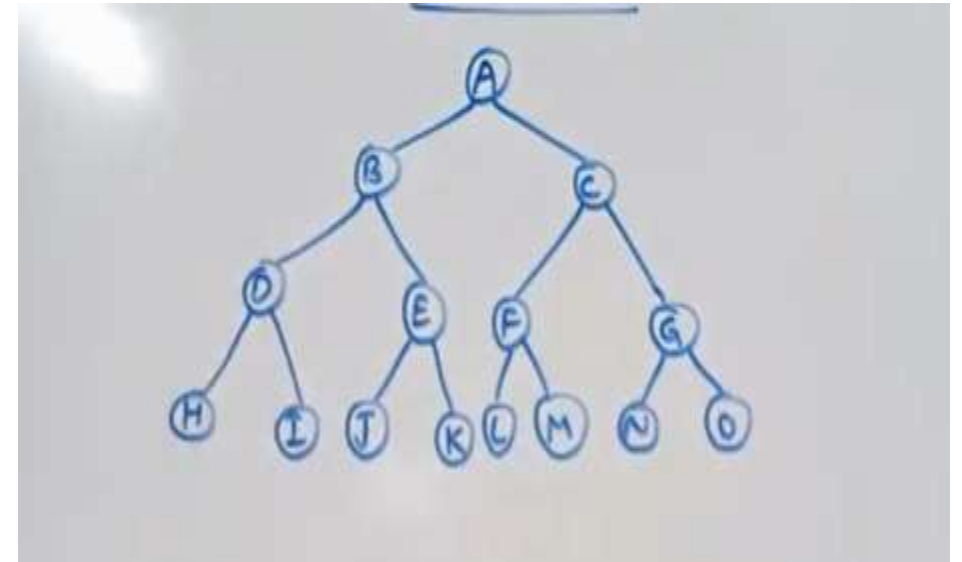
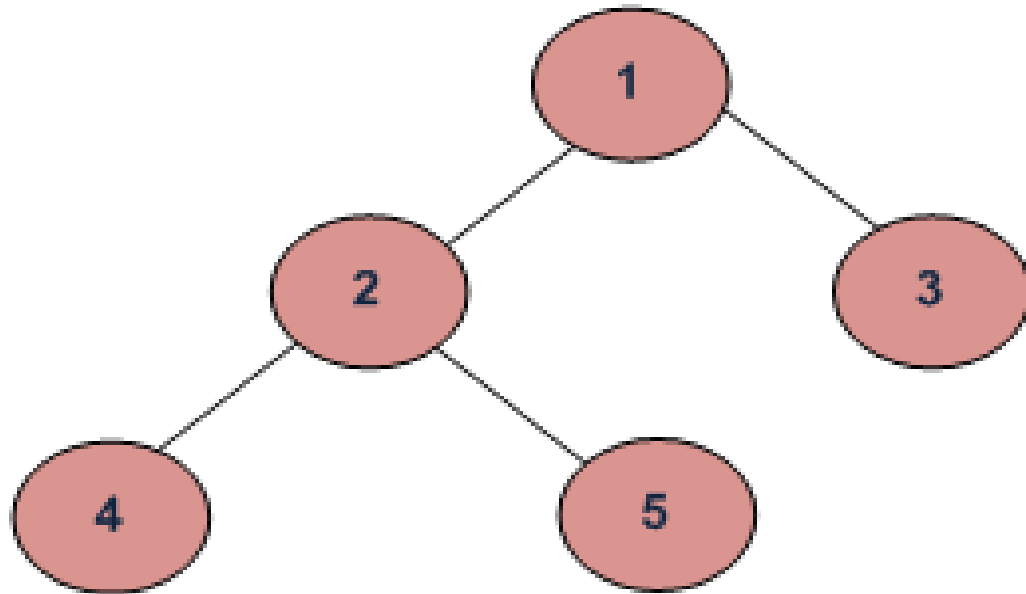
# Assessment





# Assessment

Find out the Tree Traversals for the following Tree







## *References*

1. M. A. Weiss, “Data Structures and Algorithm Analysis in C”, Pearson Education, 2<sup>nd</sup> Edition, 2002.
2. A. V. Aho, J. E. Hopcroft and J. D. Ullman, “Data Structures and Algorithms”, Pearson Education, 2<sup>nd</sup> Edition, 2007
3. Ashok Kamthane, " Data Structures Using C ", Pearson Education, 2<sup>nd</sup> Edition, 2012.
4. Sahni Horowitz, “Fundamentals of Data Structures in C”Universities Press; Second edition 2008



*Thank You*