DATA STRUCTURES

LECTURE-12

TREE

Dr. Sumitra Kisan

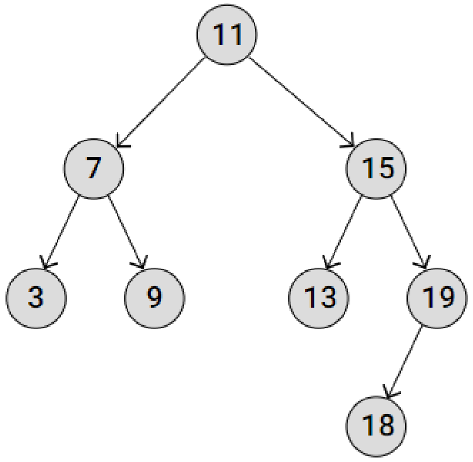
Types of Binary Trees

- A **balanced** Binary Tree has at most 1 in difference between its left and right subtree heights, for each node in the tree.
- A **complete** Binary Tree has all levels full of nodes, except the last level, which is can also be full, or filled from left to right. The properties of a complete Binary Tree means it is also balanced.
- A **full** Binary Tree is a kind of tree where each node has either 0 or 2 child nodes.
- A **perfect** Binary Tree has all leaf nodes on the same level, which means that all levels are full of nodes, and all internal nodes have two child nodes. The properties of a perfect Binary Tree means it is also full, balanced, and complete.
- **Skewed Binary Tree-** A skewed binary tree is a binary tree that satisfies the following 2 properties-

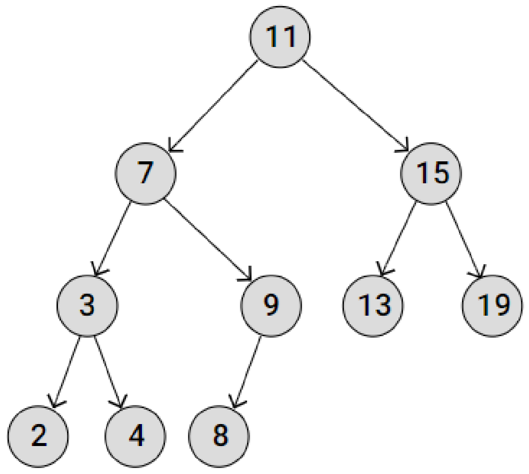
  All the nodes except one node has one and only one child.

  The remaining node has no child.

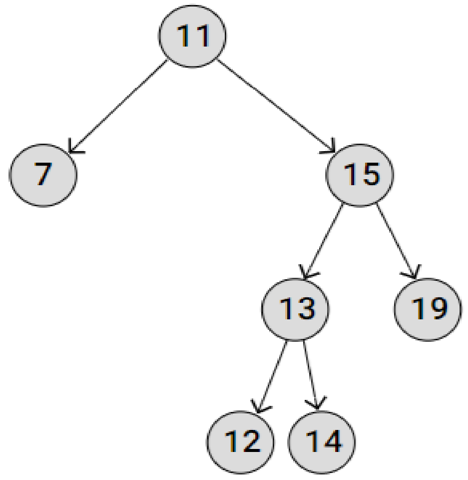
  A skewed binary tree is a binary tree of n nodes such that its depth is (n-1).



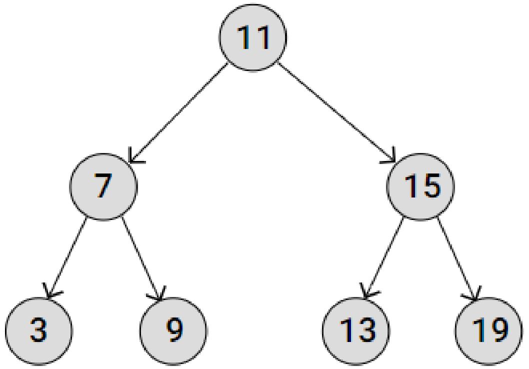
Balanced



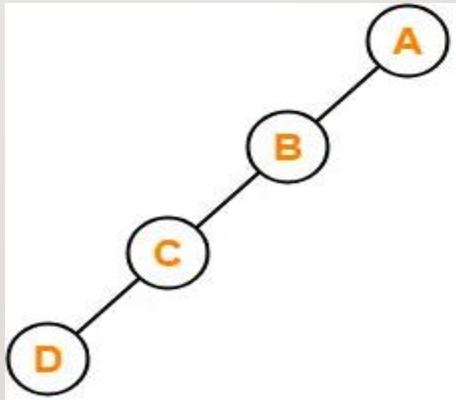
Complete and balanced



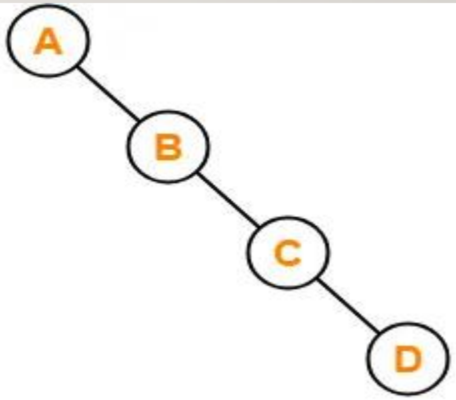
Full



Perfect, full, balanced and complete



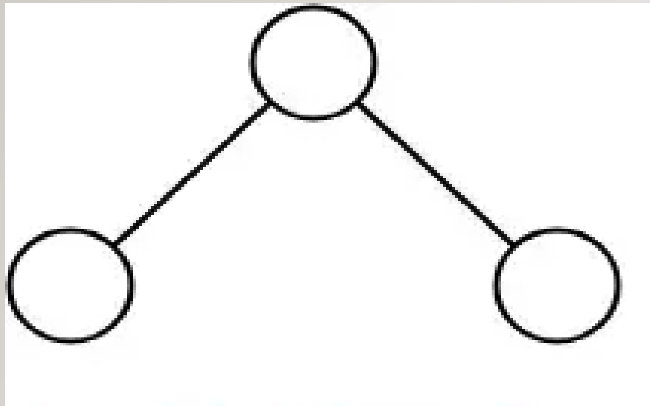
Left Skewed Binary Tree



Right Skewed Binary Tree

Unlabeled Binary Tree

Binary tree is unlabeled if its nodes are not assigned any label.



Unlabeled Binary Tree

$$\text{Number of different Binary Trees possible with 'n' unlabeled nodes} = \frac{{}^{2n}C_n}{n+1}$$

Example-

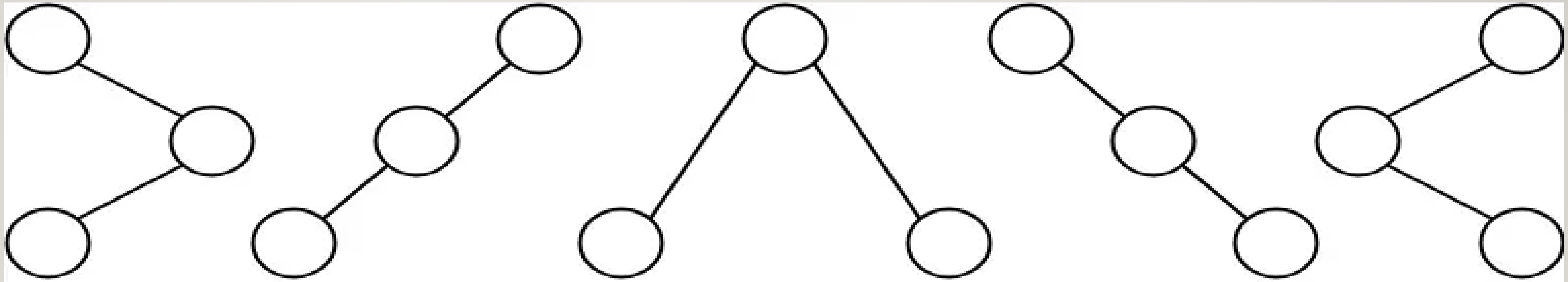
Consider, we want to draw all the binary trees possible with 3 unlabeled nodes.

Number of binary trees possible with 3 unlabeled nodes

$= 2x3_{C}3 \ / (3 + 1)$

= 6C3 / 4

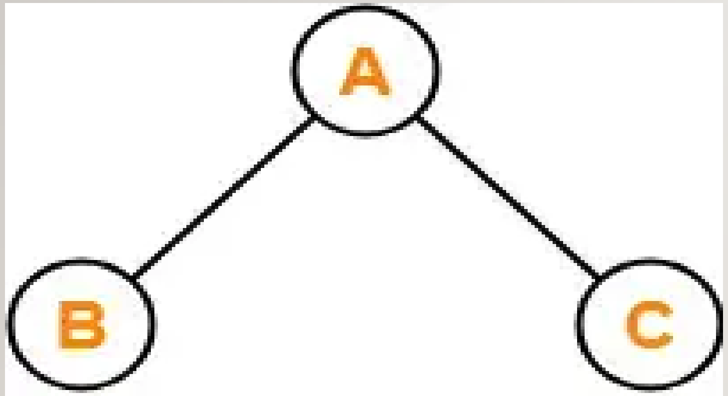
= 5



Binary Trees Possible With 3 Unlabeled Nodes

Labeled Binary Tree

A binary tree is labeled if all its nodes are assigned a label.



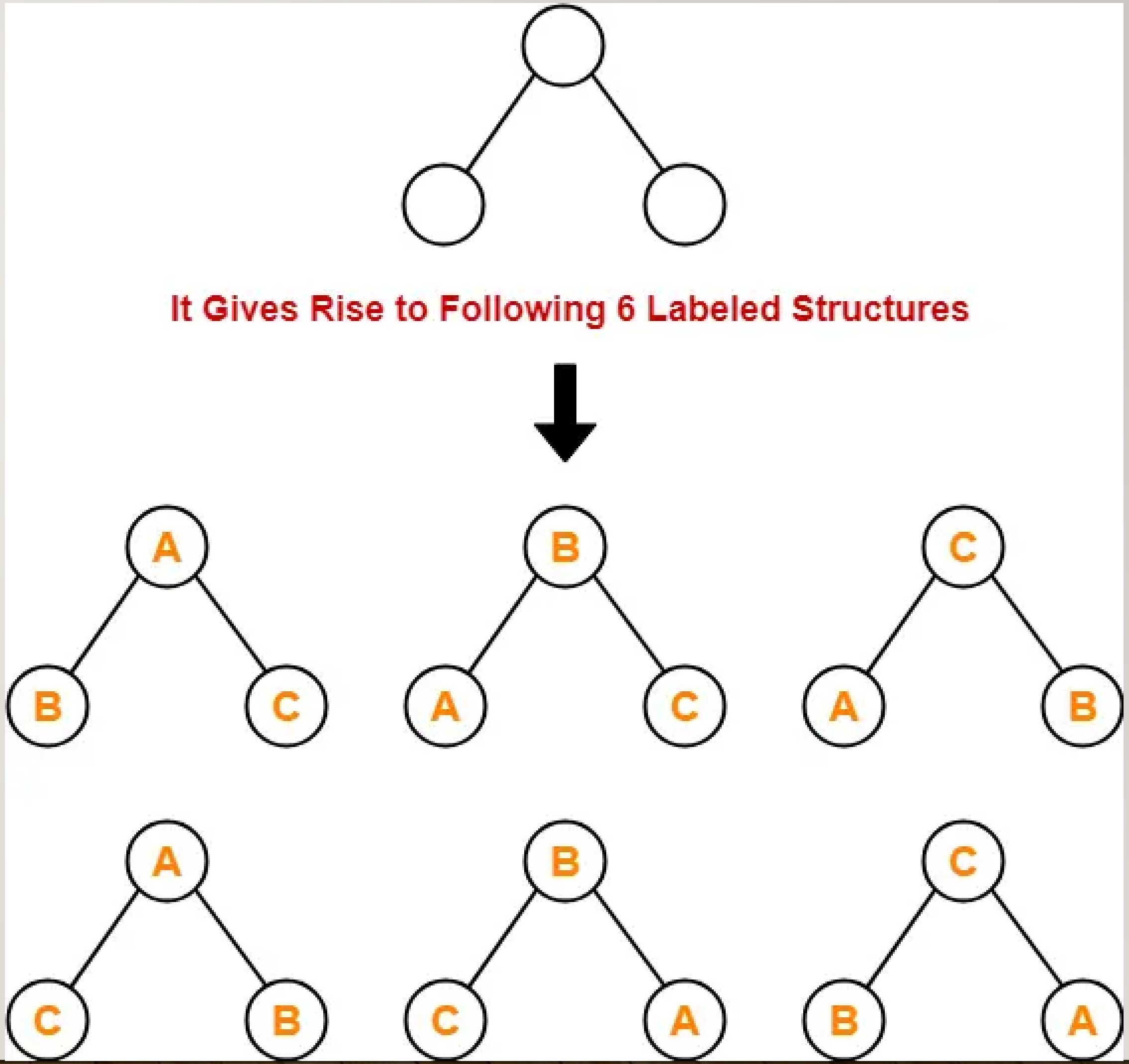
Labeled Binary Tree

Number of different Binary Trees possible with 'n' labeled nodes $= \dfrac{^{2n}C_n}{n+1} \times n!$

Example

Consider, we want to draw all the binary trees possible with 3 labeled nodes.
Using the above formula, we have-

Number of binary trees possible with 3 labeled nodes
$= \{ {}^{2 \times 3}C_3 / (3 + 1) \} \times 3!$
$= \{ {}^{6}C_3 / 4 \} \times 6$
$= 5 \times 6$
$= 30$

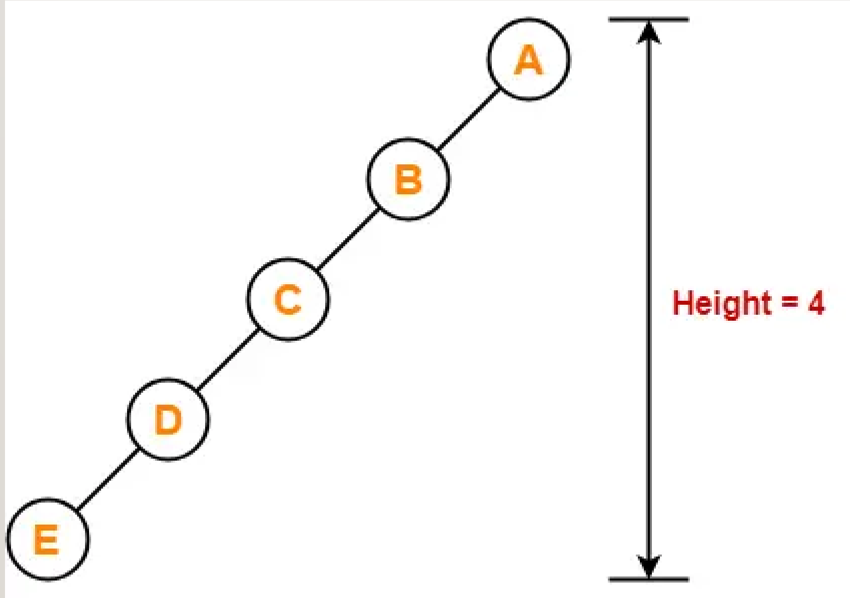


Similarly,
- Every other unlabeled structure gives rise to 5 different labeled structures.
- Thus, in total 30 different labeled binary trees are possible.

Binary Tree Properties

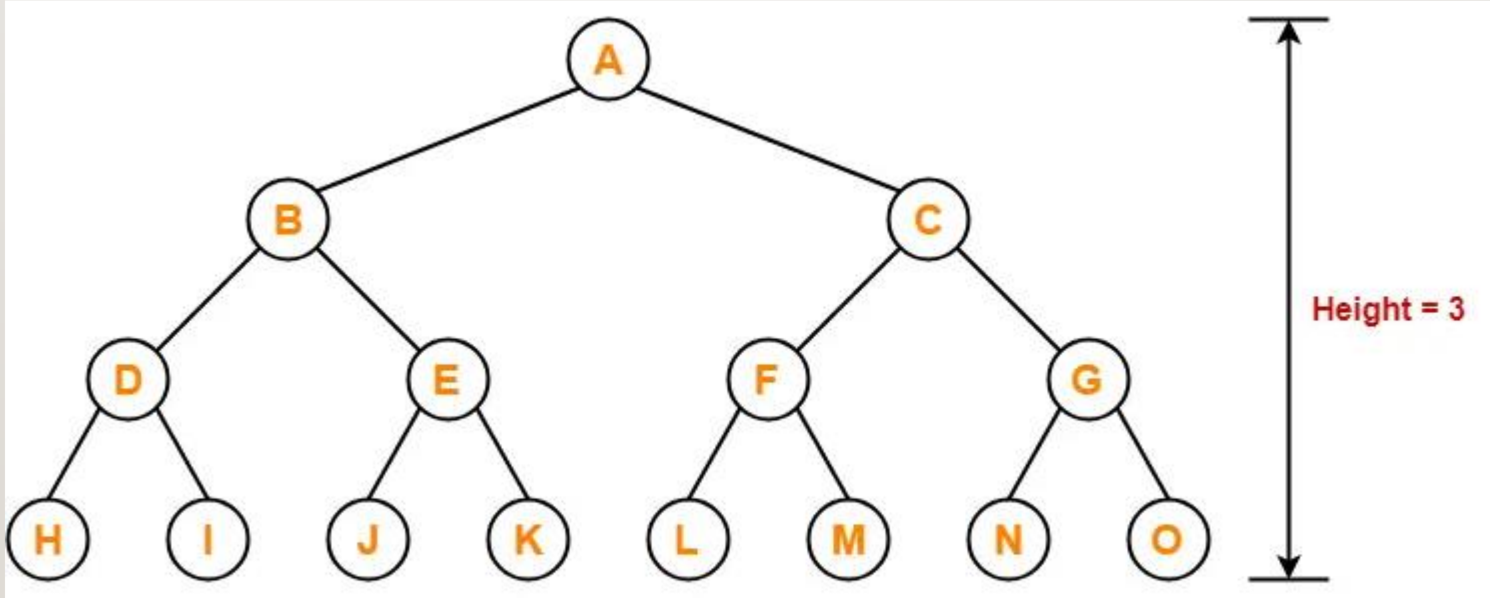
1. ***Minimum number of nodes in a binary tree of height H = H +*** 1

To construct a binary tree of height = 4, we need at least 4 + 1 = 5 nodes



2. *Maximum number of nodes in a binary tree of height H* $= 2^{H+1} - 1$

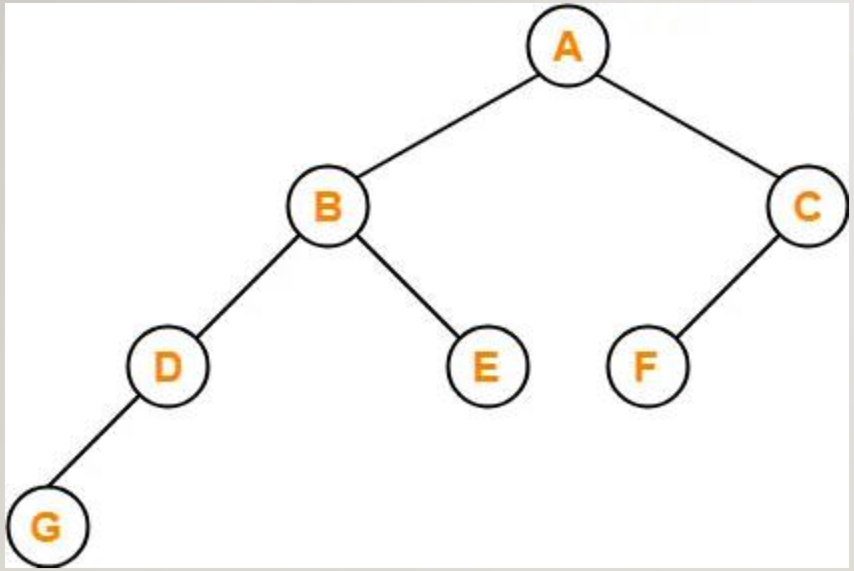
Example:
Maximum number of nodes in a binary tree of height 3
$= 2^{3+1} - 1$
$= 16 - 1$
= 15 nodes
Thus, in a binary tree of height = 3, maximum number of nodes that can be inserted = 15.



3. *Total Number of leaf nodes in a Binary Tree*
= Total Number of nodes with 2 children + 1

Example-

Consider the following binary tree-



Here,

- Number of leaf nodes = 3
- Number of nodes with 2 children = 2

Clearly, number of leaf nodes is one greater than number of nodes with 2 children.

4. Maximum number of nodes at any level 'L' in a binary tree= 2^L

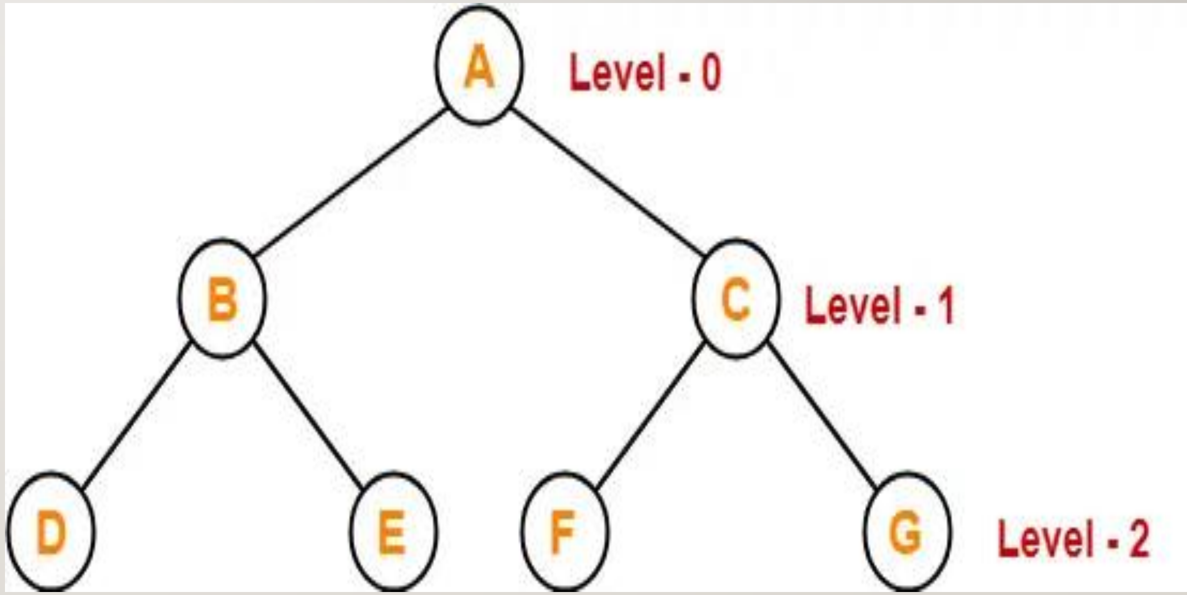
Example-

Maximum number of nodes at level-2 in a binary tree

$= 2^2$

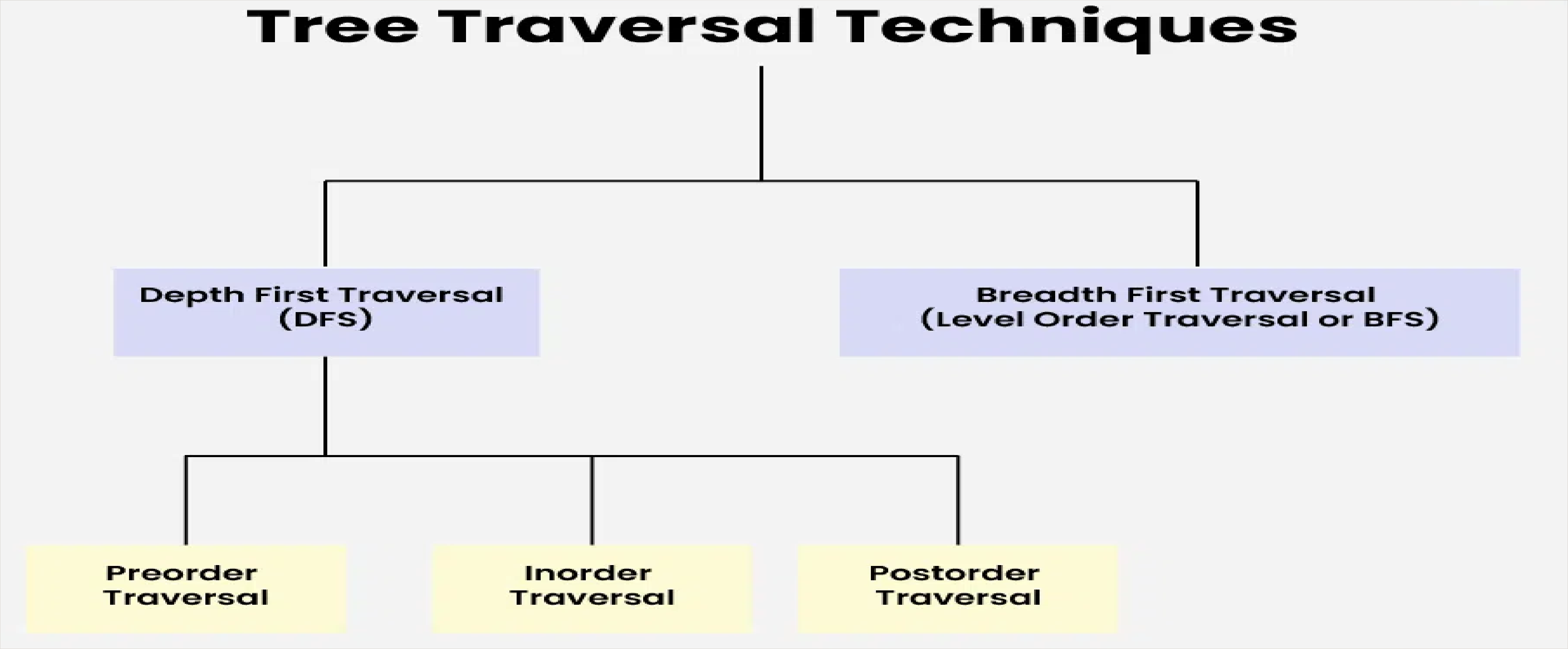
$= 4$

Thus, in a binary tree, maximum number of nodes that can be present at level-2 = 4.



Tree Traversal Techniques

Tree Traversal refers to the process of visiting or accessing each node of the tree exactly once in a certain order. Tree traversal algorithms help us to visit and process all the nodes of the tree.



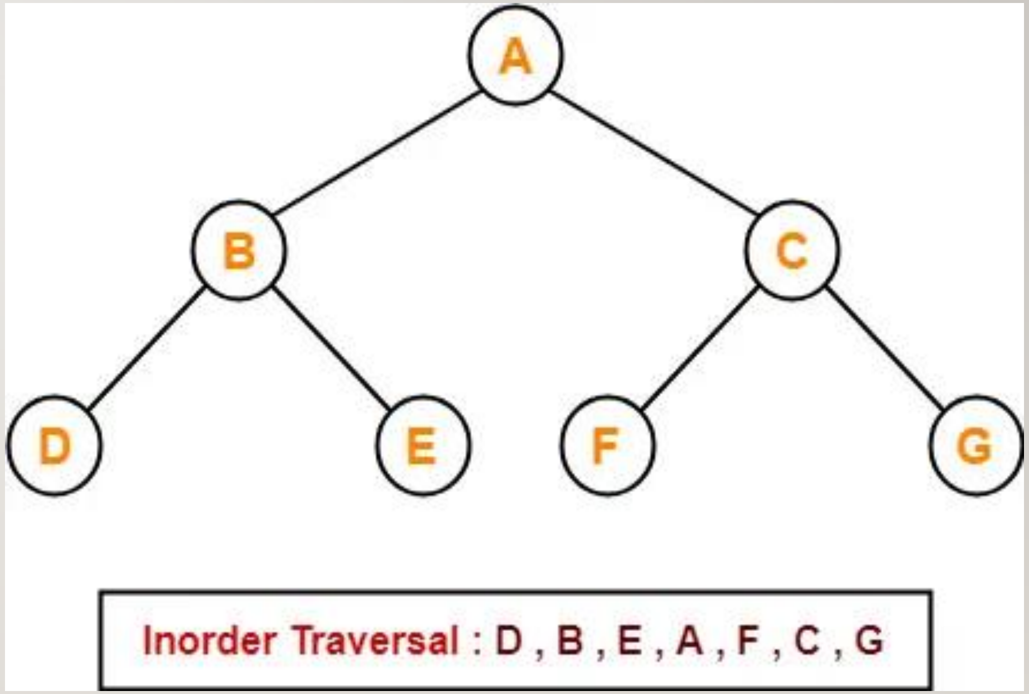
Inorder Traversal

Inorder traversal visits the node in the order: **Left -> Root -> Right**

Algorithm:

Inorder(tree)

- Traverse the left subtree, i.e., call Inorder(left->subtree)
- Visit the root.
- Traverse the right subtree, i.e., call Inorder(right->subtree)



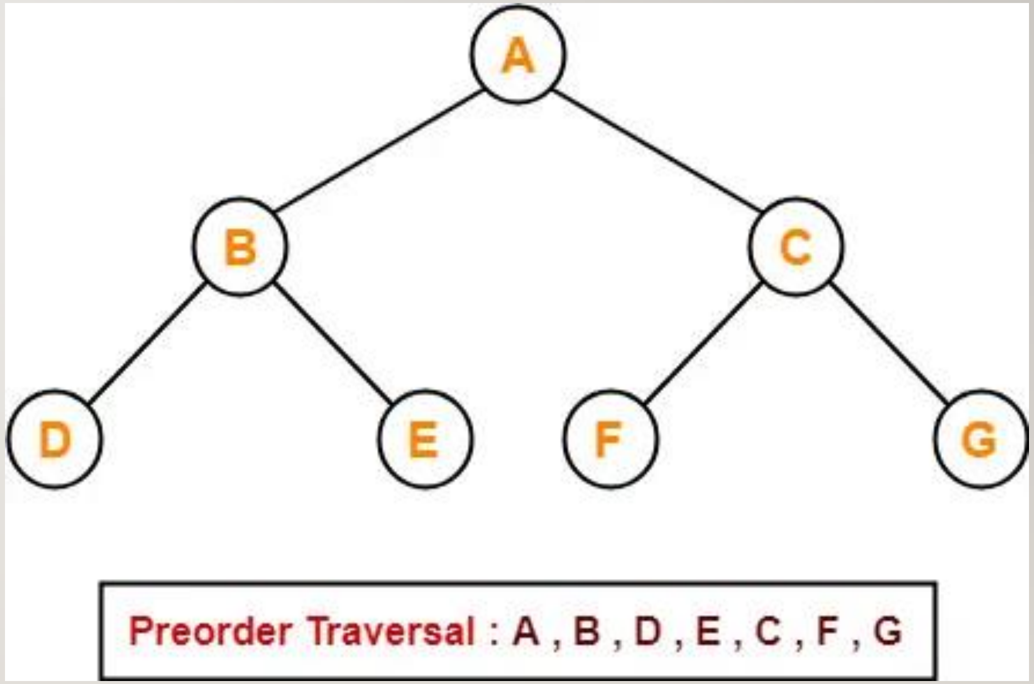
Preorder Traversal

Preorder traversal visits the node in the order: **Root -> Left -> Right**

Algorithm

Preorder(tree)

- Visit the root.
- Traverse the left subtree, i.e., call Preorder(left->subtree)
- Traverse the right subtree, i.e., call Preorder(right->subtree)



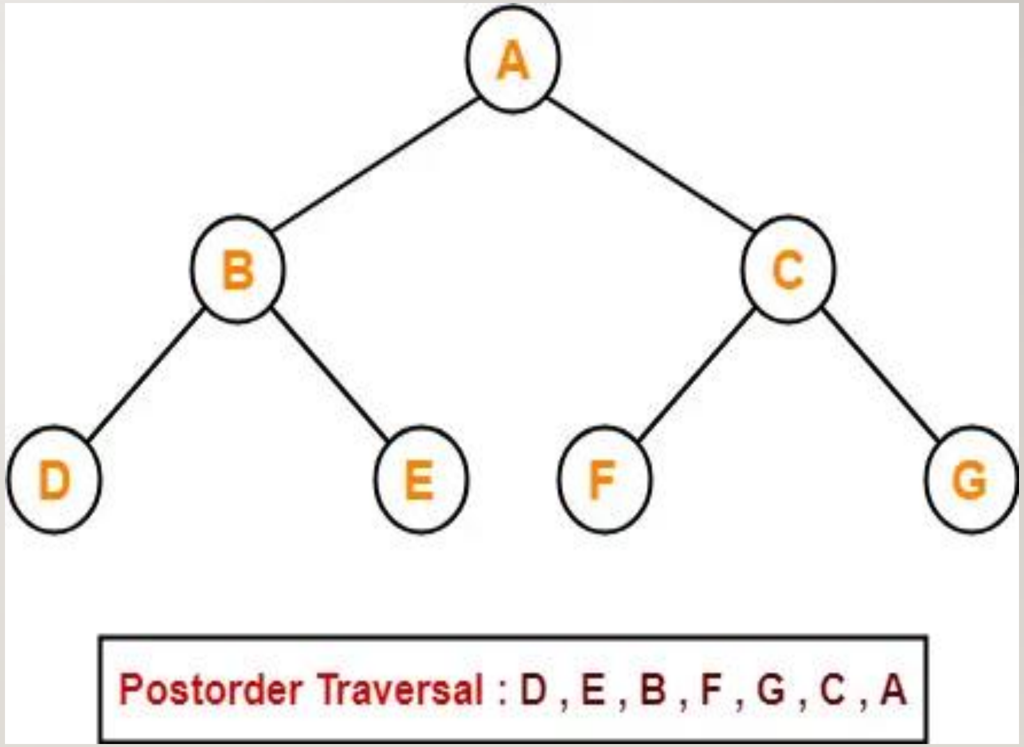
Postorder Traversal

Postorder traversal visits the node in the order: **Left -> Right -> Root**

Algorithm

Postorder(tree)

- Traverse the left subtree, i.e., call Postorder(left->subtree)
- Traverse the right subtree, i.e., call Postorder(right->subtree)
- Visit the root



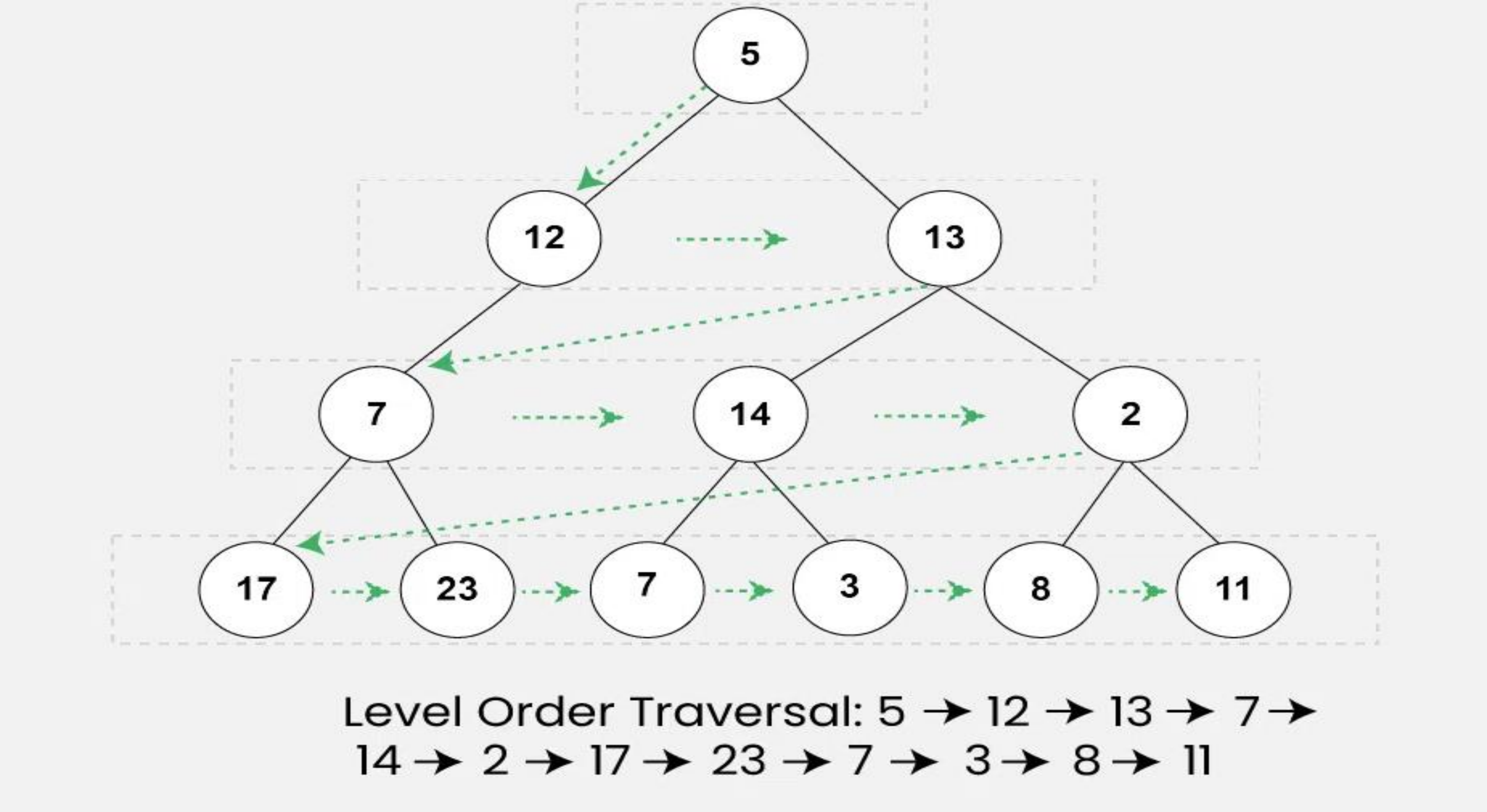
Breadth First Traversal

- Breadth First Traversal of a tree prints all the nodes of a tree level by level.
- Breadth First Traversal is also called as **Level Order Traversal**.
- **Level Order Traversal** visits all nodes present in the same level completely before visiting the next level

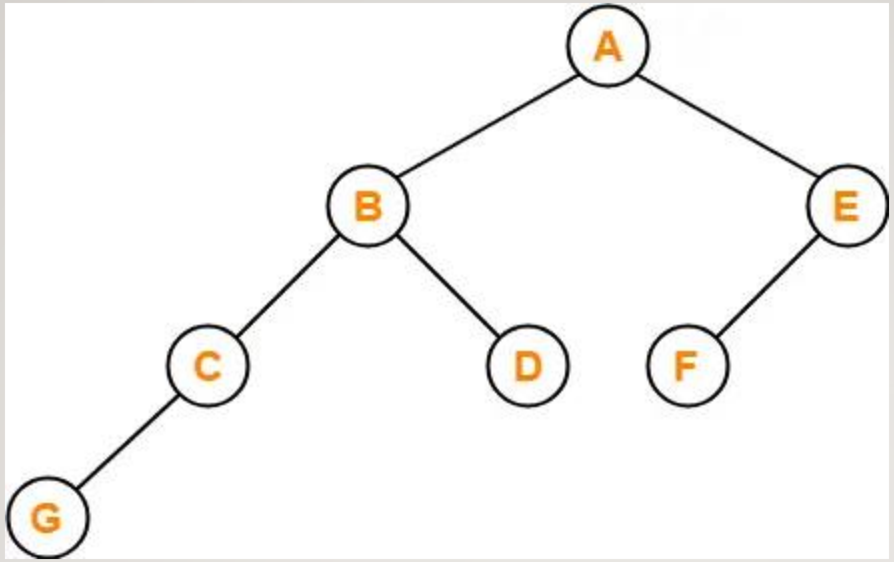
Algorithm:

LevelOrder(tree)
- *Create an empty queue Q*
- *Enqueue the root node of the tree to Q*
- *Loop while Q is not empty*
  - *Dequeue a node from Q and visit it*
  - *Enqueue the left child of the dequeued node if it exists*
  - *Enqueue the right child of the dequeued node if it exists*

Example:



5
12
13
7
14
2
17
23
7
3
8
11
Level Order Traversal: 5 ➔ 12 ➔ 13 ➔ 7 ➔
14 ➔ 2 ➔ 17 ➔ 23 ➔ 7 ➔ 3 ➔ 8 ➔ 11



If the binary tree in figure is traversed in inorder, then the order in which the nodes will be visited is ____?

Preorder:
Postorder: