TREE Data Structure

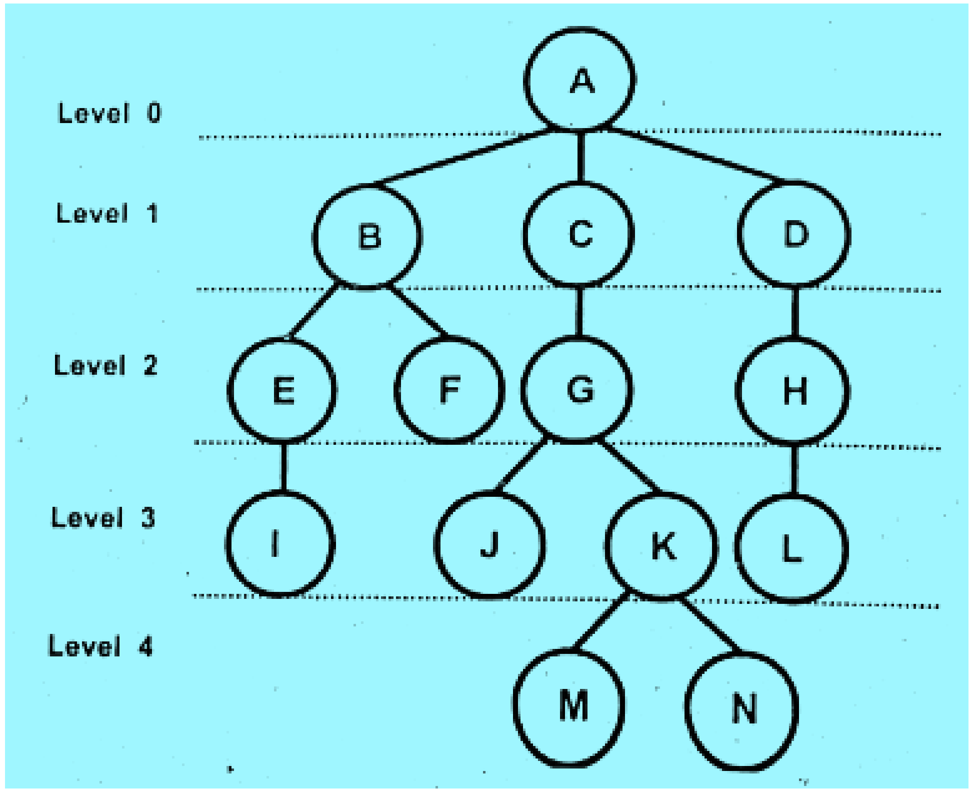
Instructor: Javeria Naz

Introduction

- There are number of applications where linear data structure are not applicable.
- In such cases there is a need of a non-linear data structure such a tree.
- A tree is a non-linear data structure in which data is organized in hierarchical structure.
- In tree data structure every individual data element is called a node.
- Nodes are connected by edges (the connection line between two nodes is called edge).
- Examples: Company’s organizational chart and family tree.

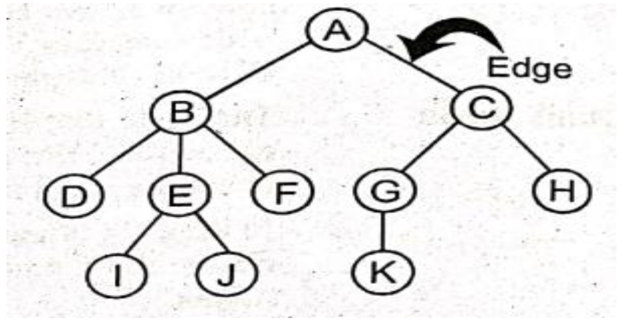
Cont.

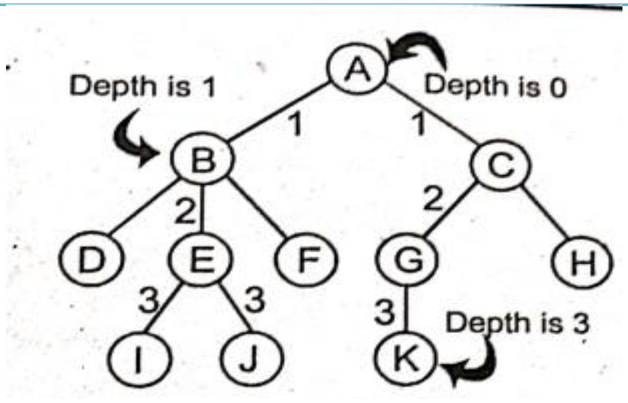
- In computer applications, tree ia a hierarchical collection of finite (limited) nodes.



Important terminologies

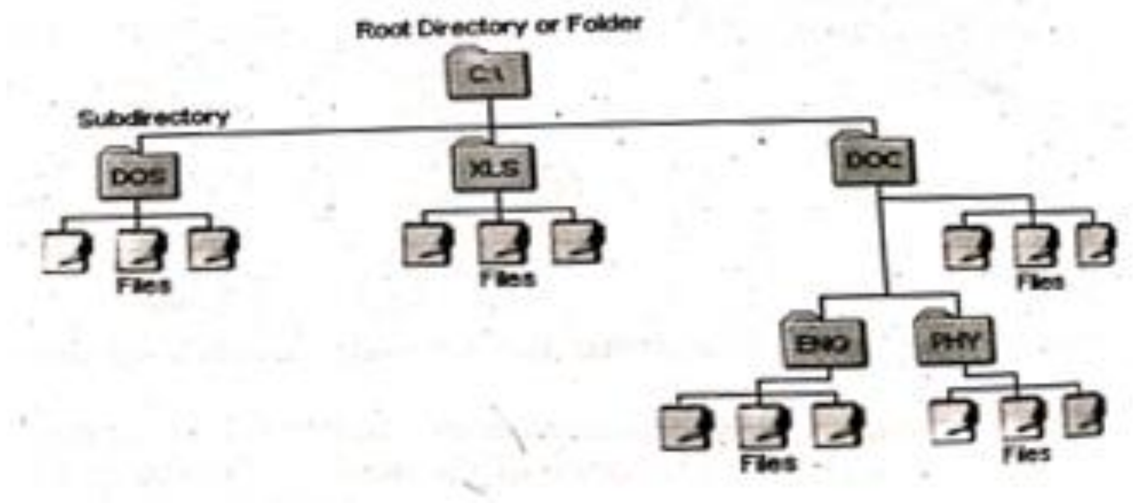
node	In a tree data structure every individual element is called as node. It holds the data and links to child nodes. In a tree all small circles represent the nodes
Root node	The top most node in the tree that has no parent node is called the Root node. It is the origin of the tree, in a tree there must be a single root node. A tree can never have multiple root nodes.
Parent node	A tree that has one or more child nodes is called parent node.
Child node	The node that is directly connected to the parent node called child node. All the nodes except root node are child nodes.
Subtree	A child node of the Root node that has its own child nodes is called Subtree
Terminal node	A node having no successor or child node is called terminal node. Also called Leaf node or external node.
Internal node	A node which have at least one child node is called internal nodes. Nodes other than leaf are called internal nodes. Internal nodes are also called non-Terminal nodes.

Level of a Node	Level represents the rank of hierarchy in a tree. The root node is said to be at Level 0 and the children of root node are at Level 1 and the children of the nodes which are at Level 1 will be at Level 2 and so on. In simple words, in a tree each step from top to bottom is called as a Level and the Level count starts with 'O' and is incremented by one at each level (Step)
siblings nodes	The nodes which have the same parent are called siblings.
edge	 A connection from one node to another node is known as edge. The nodes are connected by edges. In a tree with 'N' number of nodes there will be a maxim 'N-1' number of edges.
Path & Length of Path	A path is a sequence of nodes and edges between two specific nodes. In the above figure, path between A and J is A-B-E-J. Similarly, path between C and K is C - G - K. In a tree, there is exactly one path from the root to each node. Length of a path is the total number of edges on the path. The length of path from every node to itself is zero. In the figure, path between A and J is A-B-E-J and has length 3.

Depth & Depth of Tree	 Depth is 1 Depth is 0 Depth is 3 A B C D E F G H I J K 1 1 2 2 3 3 3	In a tree data structure, the total number of edges. length of path) from root node to a particular no called as depth of that node. The depth of the root is '0'. In a tree, the total number of edges from root node to the leaf node that exists on the longest path is the depth of the tree.
Empty Tree	A tree that has no 'node is called an empty tree. Its height is - 1.	
Degree of Node	The total number of children (child nodes) of a node is called degree of the node. In the figure, degree of the node E is one and the degree of node K is two.	
Singleton Tree	A tree whose root is its only node is called singleton tree. tree in which a node may have no child, one child or any number of children is called general tree. A tree whose all-internal nodes have the same degree and all its terminals are at the same level is called full tree.	

Applications of tree data structure

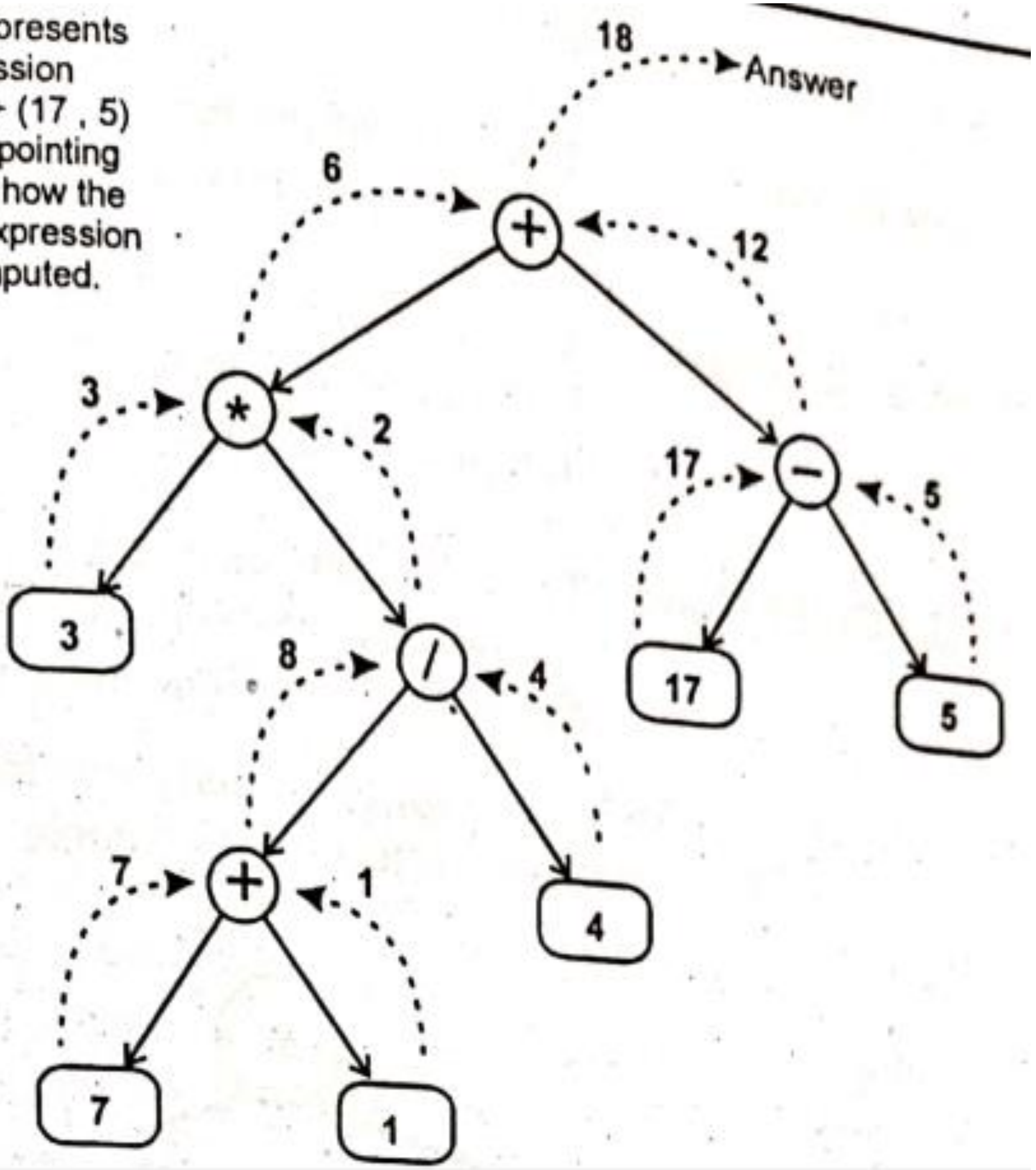
- Tree is very important data structure.
- In computer sciences, tree data is used for many purposes. Some of these are as follows:
- **File Systems:** It is used to implement hierarchical file systems of popular operating systems such as UNIX and DOS.



Cont.

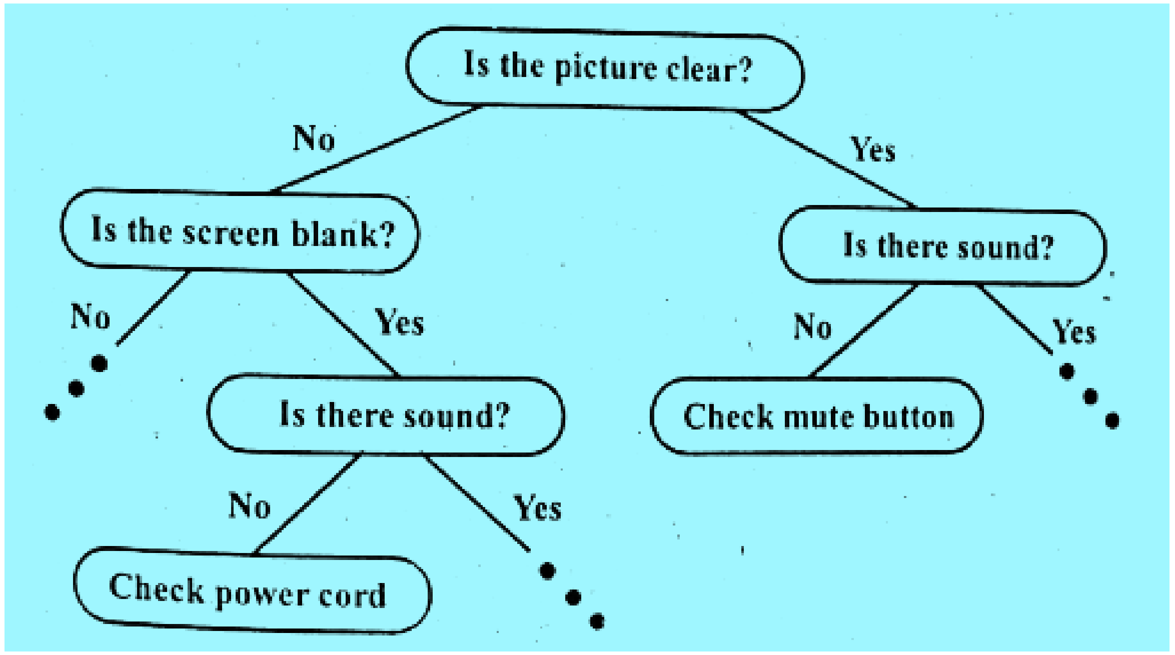
- **Data Storage:** It is used to organize large volume of data structure. Data stored in this manner can be accessed and searched efficiently.
- **Expression Evaluation:** It can be used to evaluate arithmetic expression. When a parenthesized expression is represented by a tree becomes a leaf node and each operator becomes and each operator becomes an internal node.

A tree that represents the expression 3' ((7 * 1)/4) + (17 , 5) The upward pointing arrows show how the value of the expression can be computed.



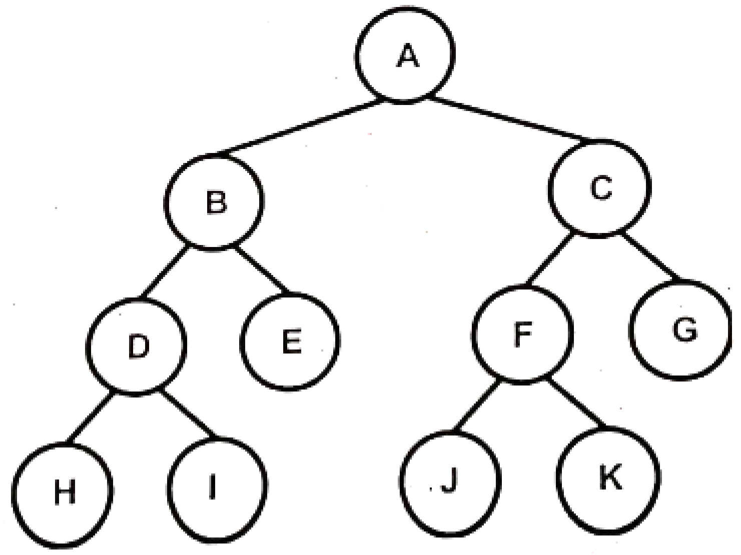
Cont.

- **Making Decision:** A tree data structure can be used in a complicated decision making process. For example, following tree structure can be used to find out (making decision) the fault of a television set;



Binary Tree

- A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.



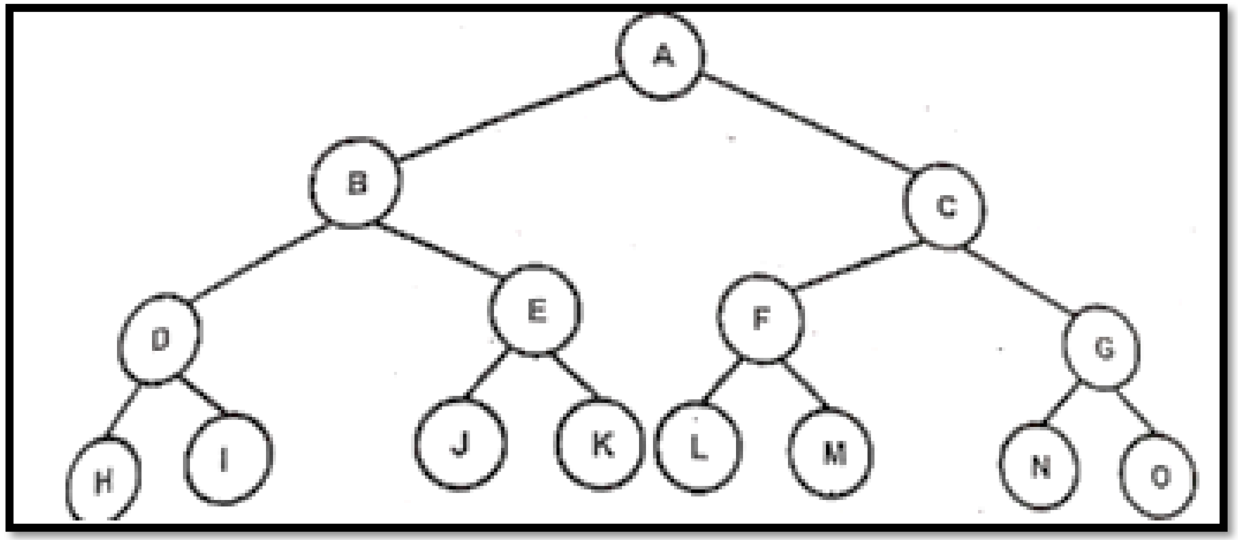
Cont.

- A non-empty binary tree has the following three parts:
- **Root Node:** The node of the binary tree that has no parent is called the root node. It is the starting node of any binary tree.
- **Left Child Node:** The node that lies on the left tree (or root of subtree) is called left child node.
- **Right Child Node:** The node that lies on the right side of root of binary tree (or root of subtree) is called right child node.

- In this figure, A is root of the binary tree and B & C are left and right children respectively, of the root A. The left child B and right child C also have their own child nodes.
- Binary tree is a very useful data structure. It is used to keep data that ne be searched quickly, or that needs to be maintained in a sorted order with insertions or deletions.

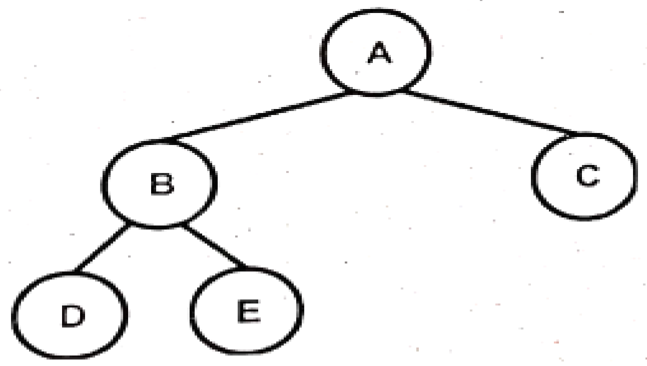
Full Binary Tree

- A tree in which each node must have two children and all terminals same level is called full binary tree.
- A typical full binary tree is show following figure: terminals must be at Is shown in the



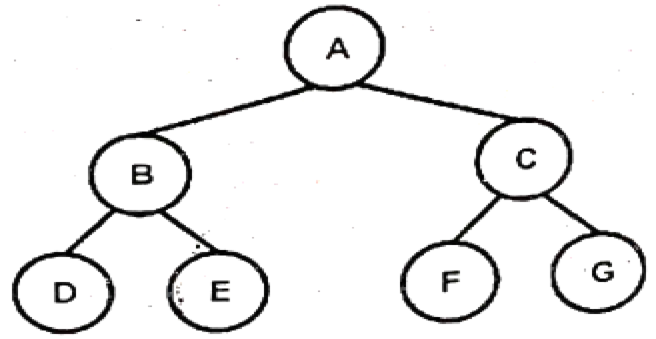
Extended Binary Tree (2_Tree)

- A binary tree in which an extended binary tree.
- It is also called internal node and the node in which each node has either no child or two children is called **Extended Binary Tree.** It is also called 2-Tree.
- The node that has two children is called internal node
- the node that has no child is called external node.



Complete Binary Tree

- **Complete Binary Tree:**
- binary tree in which every internal node has exactly two children and all nodes are at same level is called complete binary tree.
- The complete binary tree A binary tree in is shown in the following figure:

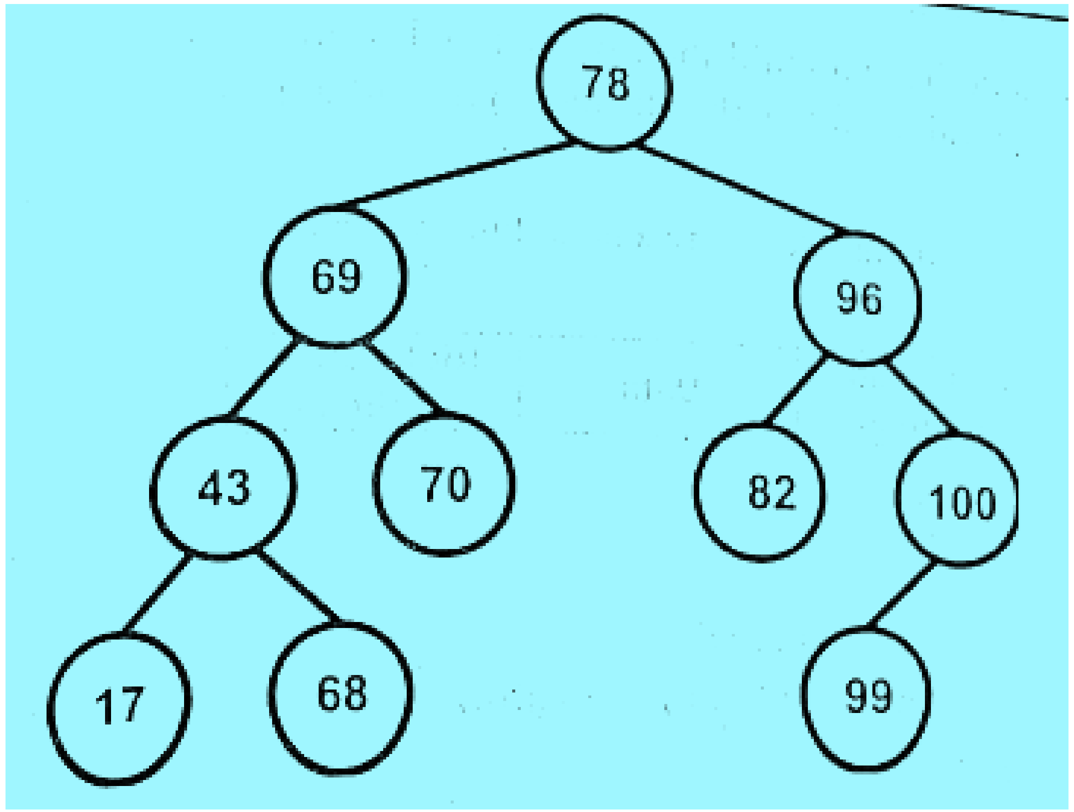


In a complete binary tree there must be 2^{level} number of nodes. For example at level 2 there must be $2^2 = 4$ nodes and at level 3 there must be $2^3 = 8$ nodes.

Binary search tree (BST)

- A Binary search tree has the property that the value in any left child node is less than the value in its parent node, and the value in any right child node is greater than the value in its parent node. The duplication of values in binary search tree is not allowed.
- The main advantage of binary search tree is that it is easy to perform operations like creating new tree, insertion, searching and deletion of data.
- Data: 78, 69, 43, 70, 17, 96, 68, 82, 100,99

Cont.



Representation of a BST in Memory

- Different techniques can be used to represent a binary search tree in the computer memory.
- The most commonly used techniques are as follows:

    i) Linked Storage Technique

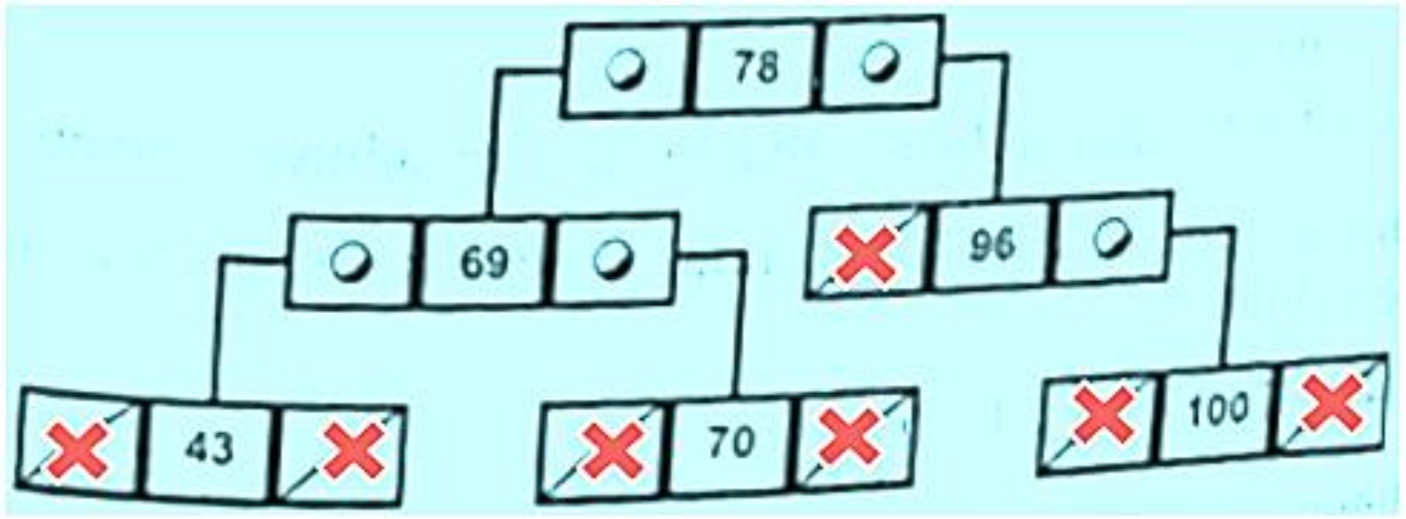
    ii) Sequential storage technique

i) Linked Storage Technique

- Linked storage technique is the most commonly used method for representing the binary tree inside the computer memory.
- In this Technique double linked listis used to represent a binary tree.
- Each node of the binary tree is consists of three fields: one data field and two pointer/link fields.
- The data field is used to store the value of node.
- The pointer fields are used to store the memory addresses of the left child node and right child node.
- Linked storage is shown in the figure:

Cont.

- If any node has no child then pointer fields are assigned with the value NULL.
- The pointer fields for the leaf nodes are always NULL.



Sequential storage technique

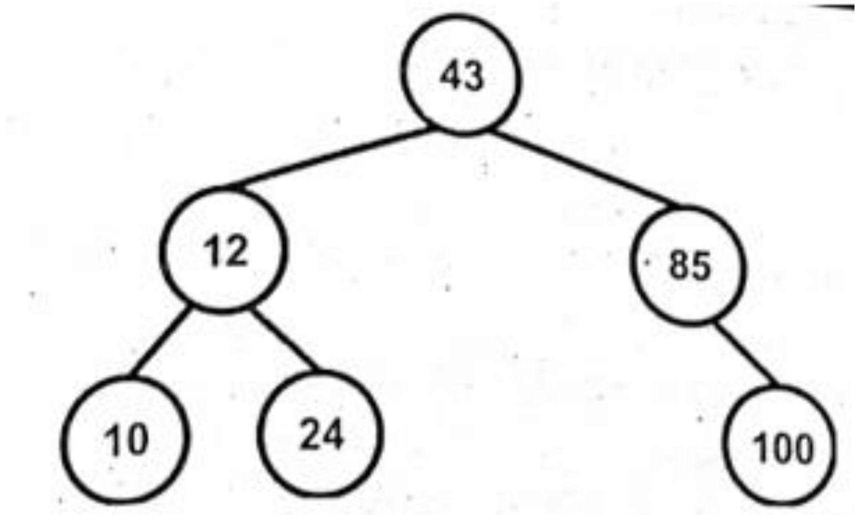
- The sequential storage technique is very simple to represent the tree inside the computer memory.
- In this technique a linear array is used to represent the tree inside the memory.
- This technique is efficient if structure of the tree does not change frequently during its existence in the memory.
- It is best used to represent full binary tree. However it is static in nature.

Sequential storage technique (Cont.)

- Suppose the root is represented by “R” and it is stored in the first position of array (index is zero and position/element is one).
- The value of the left child of the node “K” is stored in the element “ 2*K” of the array.
- The value of the right child of the node “K” is stored in the element “ 2*K+1” of the array.
- Null is used to represent the empty.

Sequential storage technique (Cont.)

- Suppose the data values are: 43, 12, 85, 10, 24, 100



Index	0	1	2	3	4	5	6
Value	43	12	85	10	24		100
Position	1	2	3	4	5	6	7

Tree Traversal

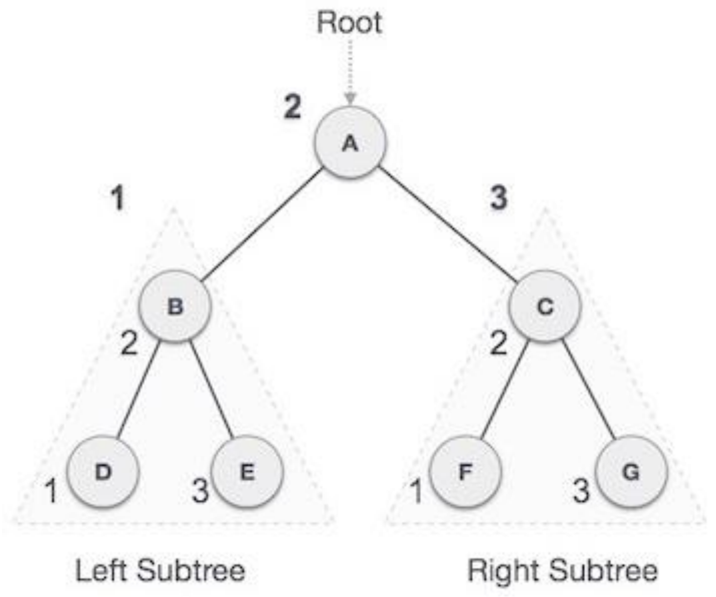
- Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from the root (head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree :

1. In-order Traversal
2. Pre-order Traversal
3. Post-order Traversal

- Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains.

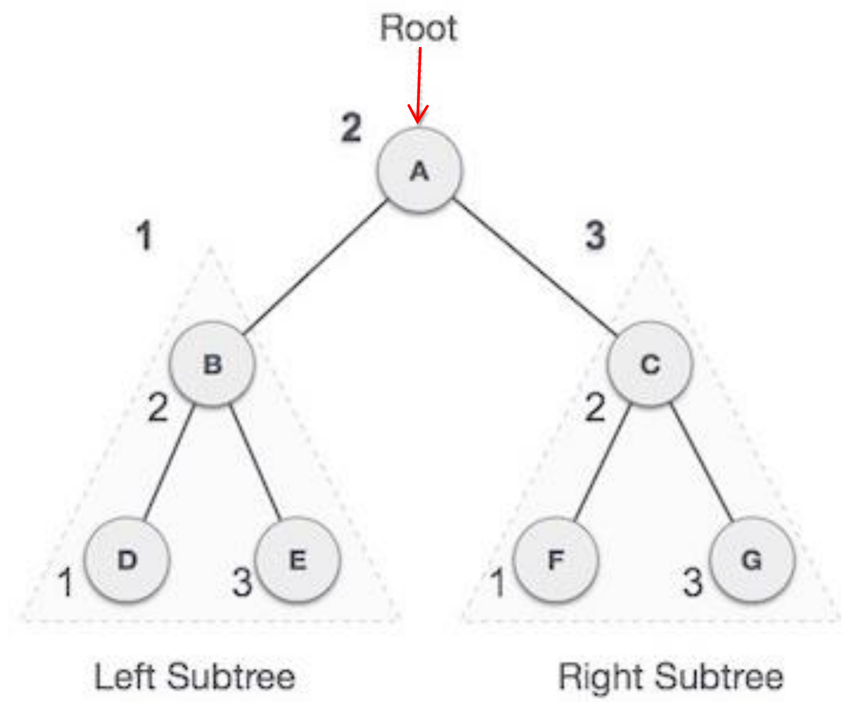
In-order Traversal

- In this traversal method, the left subtree is visited first, then the root and later the right sub-tree.
    1. Visit the left sub-tree in in-order.
    2. Visit the root.
    3. Visit the right sub-tree in in-eorder.
- We should always remember that every node may represent a subtree itself.
- If a binary tree is traversed **in-order**, the output will produce sorted key values in an ascending order.



In-order Traversal (Cont.)

- We start from **A**, and following in-order traversal, we move to its left subtree **B**.
- **B** is also traversed in-order. The process goes on until all the nodes are visited.
- The output of inorder traversal of this tree will be −
- ***D → B → E → A → F → C → G***

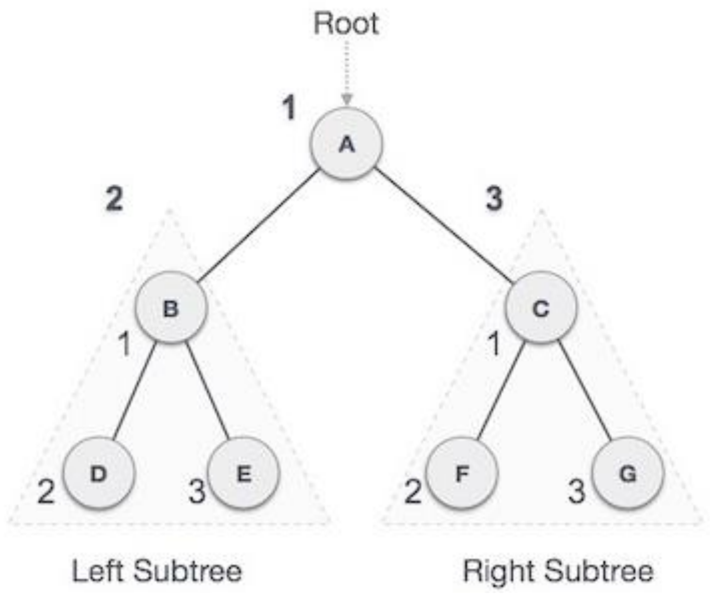


Pre-Order Traversal

In this traversal method, the root node is visited first, then the left subtree and finally the right subtree.

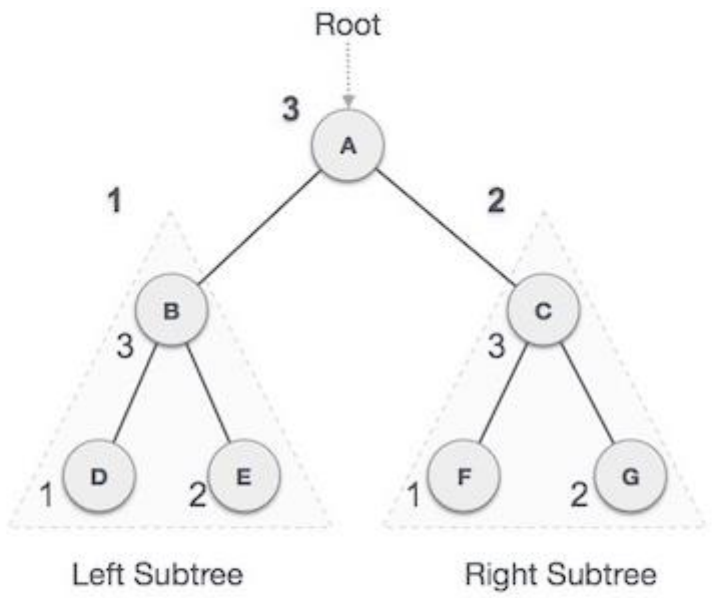
1. Visit the root.
2. Visit the left sub-tree in preorder.
3. Visit the right sub-tree in preorder.

- We start from **A**, and following pre-order traversal, we first visit **A** itself and then move to its left subtree **B**. **B** is also traversed pre-order. The process goes on until all the nodes are visited. The output of pre-order traversal of this tree will be −
- ***A → B → D → E → C → F → G***



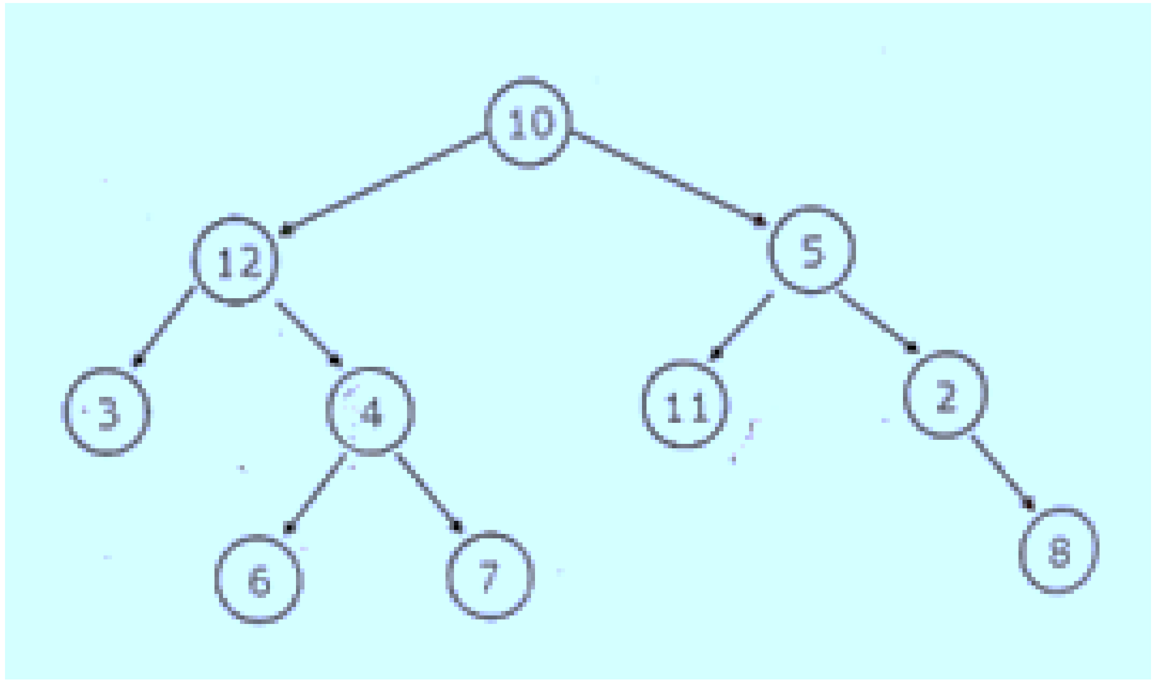
Post-order Traversal

- In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node.
  1. Visit the left sub-tree in Post-orde.
  2. Visit the right sub-tree in Post-orde.
  3. Visit the root.



- We start from **A**, and following Post-order traversal, we first visit the left subtree **B**. **B** is also traversed post-order. The process goes on until all the nodes are visited. The output of post-order traversal of this tree will be −
- ***D → E → B → F → G → C → A***

Task


10
12
5
3
4
11
2
6
7
8