Unit IV : Tree

- Introduction to Binary Trees, Representation of Binary trees in memory.
- Terminology of Binary tree
- Types of Binary Tree
- Traversing of Binary Tree (Pre-order, In-order, Post-order)
- Header Nodes, Threads
- General Tree Introduction

Introduction and Representation of Binary trees in memory :

Tree is a non-linear data structure in which each data element is called as 'Node' represented using hierarchical relationship.

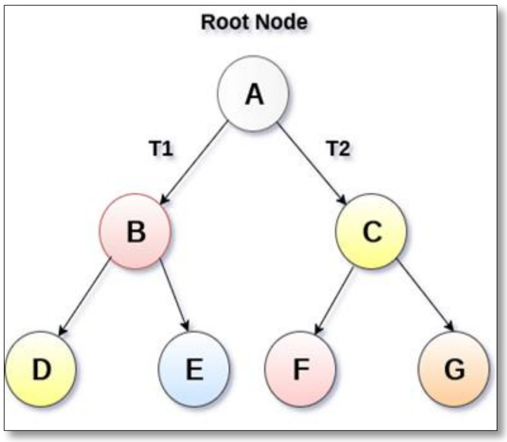
For e.g. Index, Family tree, etc.

Trees can be of multiple types, the most common use of tree is in the form of 'Binary Tree'. A tree in which each node can contain only two sub-nodes is called 'Binary Tree'.

A binary tree is a hierarchical data structure in which each node has at most two children generally referred as left child and right child.

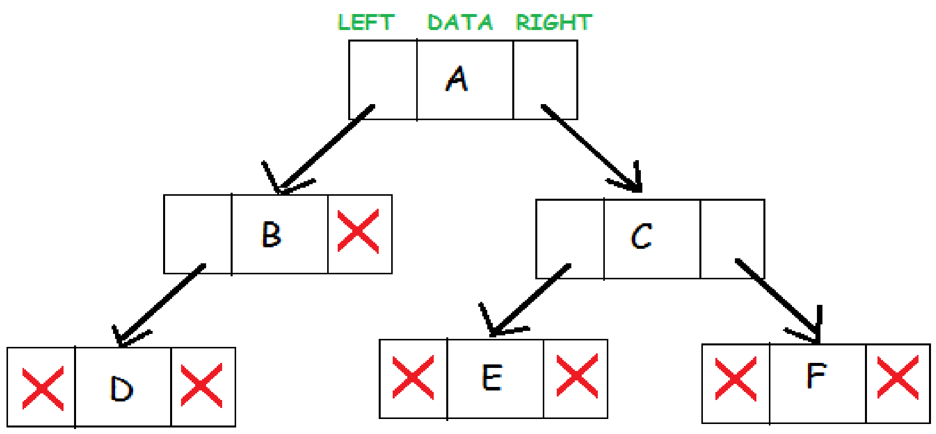
Each node contains three components :

- Data element
- Pointer to left subtree
- Pointer to right subtree



The topmost node in the tree is called the root. An empty tree is represented by **NULL** pointer.

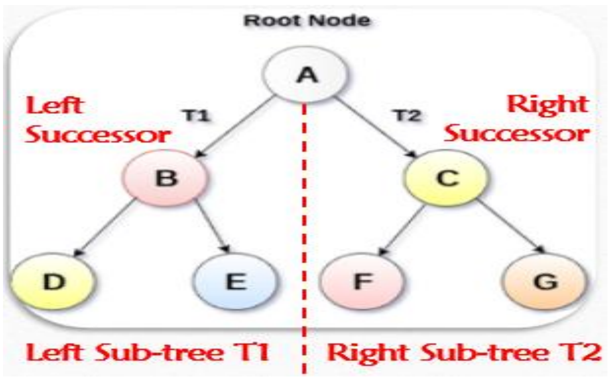
A representation of binary tree can be shown as :



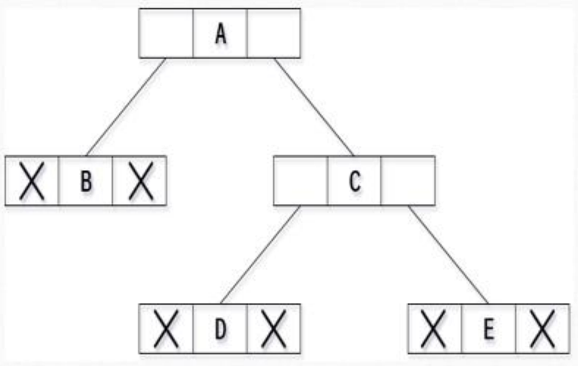
A binary tree T with root node & other nodes forms two sub-trees T1 & T2, as :

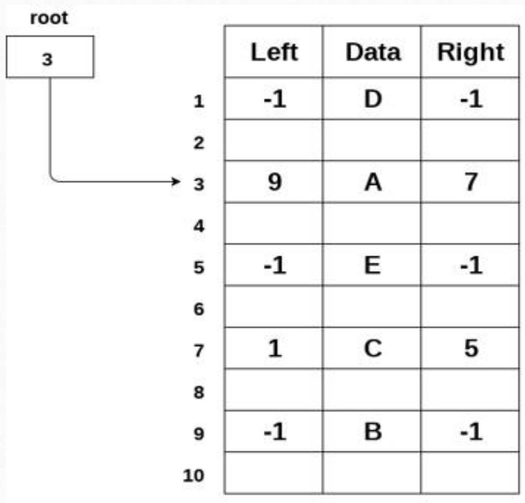
The given binary tree shows following characteristics :

1. Tree contains total 7 nodes represented using alphabets from A to G.
2. The topmost node A represents root node.
3. B is the left successor & C is the right successor of root node A.
4. The left sub-tree consists of B D E & the right sub-tree consists of C F G.



Linked representation of Binary tree in memory :





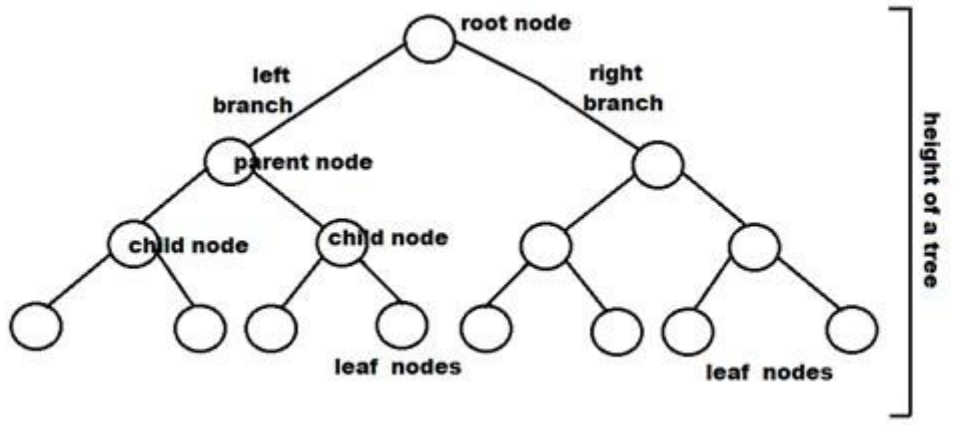
	Left	Data	Right
1	-1	D	-1
2			
3	9	A	7
4			
5	-1	E	-1
6			
7	1	C	5
8			
9	-1	B	-1
10			

Introduction to Binary Tree and its Terminology

A binary tree is a tree-type non-linear data structure with a maximum of two children for each parent. Every node in a binary tree has a left and right reference along with the data element.

The node at the top of the hierarchy of a tree is called the root node. The nodes that hold other sub-nodes are the parent nodes.

A parent node has two child nodes: the left child and right child.



- **Node :** It represents a termination point in a tree.
- **Root :** A tree's topmost node.
- **Parent :** Each node (apart from the root) in a tree that has at least one sub-node of its own is called a parent node. It is also called as Predecessor.
- **Child :** A new node which is added after the existing node becomes child . It is also called as Successor.
- **Leaf Node :** These are external nodes. They are the nodes that have no child.
- **Internal Node :** As the name suggests, these are inner nodes with at least one child.
- **Level Number :** The tree starting with root node is assigned an integer value i.e. Zero. With each successive level, this integer is incremented by 1.
- **Depth of a Tree :** It represents maximum level of given tree i.e., last level + 1.
- **Height of a Tree :** It is the number of edges from the node to the deepest leaf. The tree height is also considered the root height.

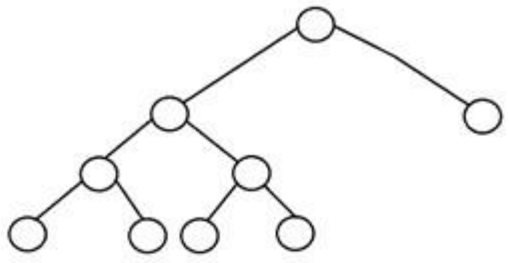
Types of Binary Tree :

There are various **types of binary trees**, and each of these **binary tree types** has unique characteristics.

1. **Full Binary Tree**

Full Binary Tree is a Binary Tree in which every node has 0 or 2 children. In other words, a full binary tree is a unique binary tree where every node except the external node has two children.

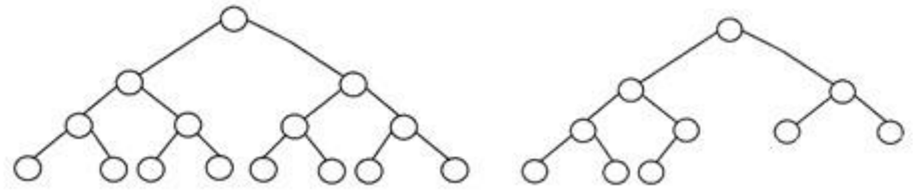
Here is the structure of a full binary tree :



2. **Complete Binary Tree**

Complete Binary Tree has all levels completely filled with nodes except the last level and in the last level, all the nodes are as left side as possible. Also, in the last or the lowest level of this binary tree, every node should possibly reside on the left side.

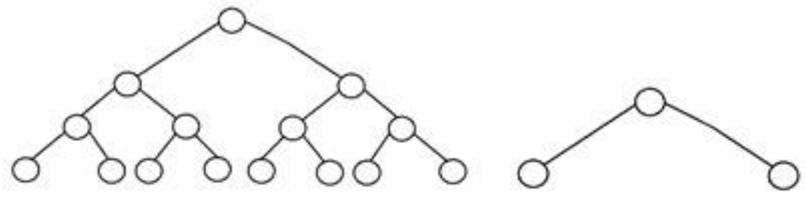
Here is the structure of a complete binary tree :



3. **Perfect Binary Tree**

Perfect Binary Tree is a Binary Tree in which all internal nodes have 2 children and all the leaf nodes are at the same depth or same level. A binary tree is said to be 'perfect' if all the internal nodes have strictly two children, and every external or leaf node is at the same level or same depth within a tree.

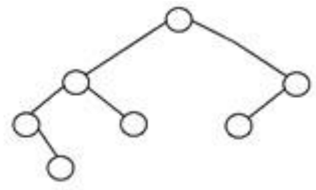
Here is the structure of a perfect binary tree :



4. **Balanced Binary Tree**

Balanced Binary Tree is a Binary tree in which height of the left and the right sub-trees of every node may differ by at most 1.

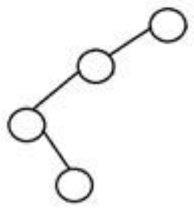
Here is the structure of a balanced binary tree :



5. **Degenerate Binary Tree**

Degenerate Binary Tree is a Binary Tree where every parent node has only one child node. A binary tree is said to be a degenerate binary tree or pathological binary tree if every internal node has only a single child.

Here is the structure of a degenerate binary tree :



Traversing Binary Tree :

Traversing is one of the most fundamental operations performed on any data structure. Traversing refers to processing or visiting each node in tree atleast once.

A binary tree is a non-linear data structure in which nodes are hierarchically organized on which traversing can be performed in three ways :

- Pre-order Traversal
- In-order Traversal
- Post-order Traversal

1. **Pre-order Traversal :** In this traversing mode, the sequence of traversing is root – left – right. It follows following rules :
   a. Process root node.
   b. Traverse left sub-tree using pre-order traversal i.e root – left – right.
   c. Traverse right sub-tree using pre-order traversal i.e root – left – right.

   The sequence becomes : ABDECFG

2. **In-order Traversal :** In this traversing mode, the sequence of traversing is left – root – right. In-order traversing follows following rules :

a. Process leftmost node of left sub-tree.
b. Process root node of left sub-tree.
c. Process rightmost node of left sub-tree.
d. Process root node of binary tree.
e. Process leftmost of right sub-tree.
f. Process root node of right sub-tree.
g. Process rightmost node of right sub-tree.

The sequence becomes : DBEAFCG

3. **Post-order Traversal :** In this traversing mode, the sequence of traversing is left – right – root. Post-order traversing follows following rules :
   a. Process leftmost node of left sub-tree.
   b. Process rightmost node of left sub-tree.
   c. Process root node of left sub-tree.
   d. Process leftmost node of right sub-tree.
   e. Process rightmost of right sub-tree.
   f. Process root node of right sub-tree.
   g. Process root node of binary tree.

The sequence becomes : DEBFGCA

- **Pre-order Traversal : root - left – right**

  Algorithm :

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
 Step 2: Write TREE DATA
 Step 3: PREORDER(TREE LEFT)
 Step 4: PREORDER(TREE RIGHT) [END OF LOOP]
 Step 5: END
```

- **In-order Traversal : left – root – right**

  Algoithm :

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
 Step 2: INORDER(TREE LEFT)
 Step 3: Write TREE DATA
 Step 4: INORDER(TREE RIGHT) [END OF LOOP]
 Step 5: END
```

- **Post-order Traversal : left – right – root**

  Algorithm :

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
    Step 2: POSTORDER(TREE LEFT)
    Step 3: POSTORDER(TREE RIGHT)
    Step 4: Write TREE DATA [END OF LOOP]
Step 5: END
```

Header Node :

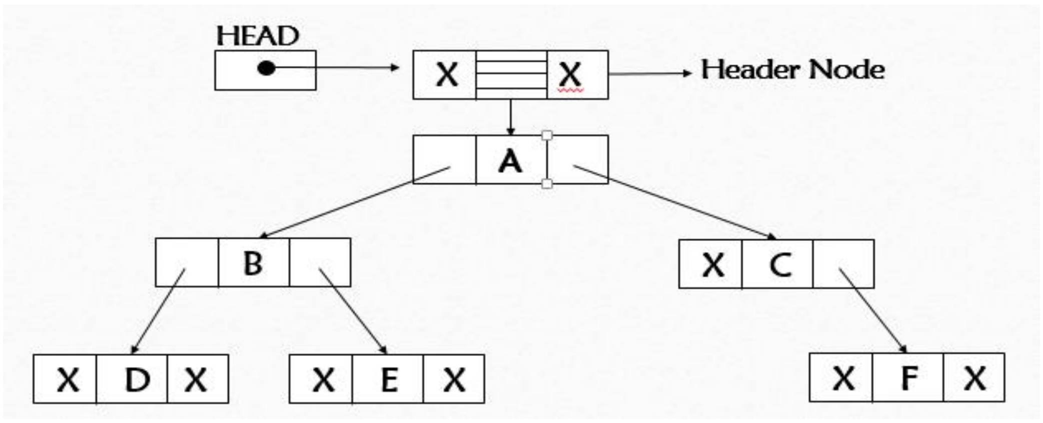
A binary tree in a computer program can be represented in memory using linked list. The specialized form of linked list in which each node has one data part & two address parts, known as double linked list.

When a double linked list is used to represent a binary tree, the root node of double linked list shows following properties :

- The left address part of root node points to left sub-tree.
- The right address part of root node points to right sub-tree.

When a binary tree is represented using double linked list, a special node is assigned to point root node of binary tree, called as Header node.

There is one more node used to point header node which is called Head Node. If T is a given binary tree, its double linked list representation can be given as :



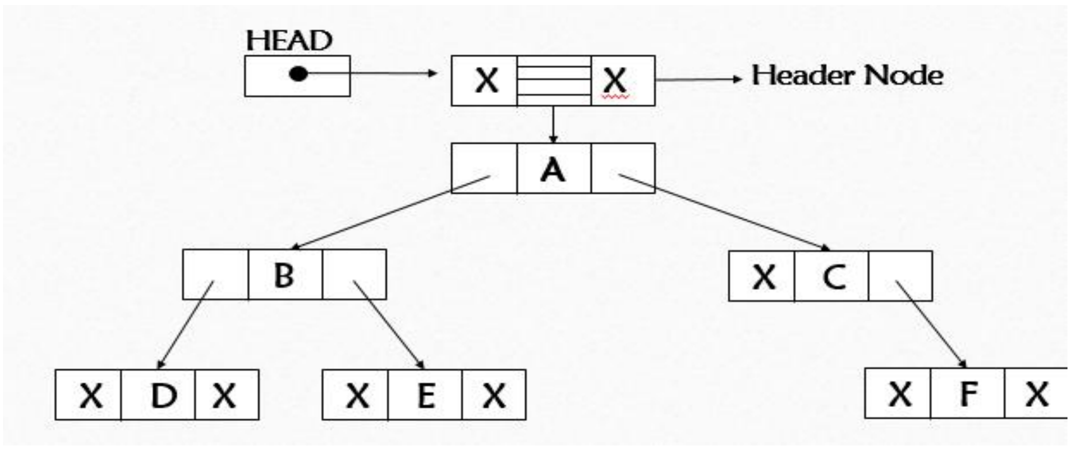
The double linked list representation provides following conclusions on binary tree :

1. If ROOT[left] = NULL, it means binary tree does not have left sub-tree.
2. If ROOT[right] = NULL, it means binary tree does not have right sub-tree.
3. If ROOT[left] = NULL & ROOT[right] = NULL, it means binary tree contains only root node.
4. If Header = NULL, it means binary tree is empty.

Threads :

In a double linked list representation of binary tree, there are so many address parts that are wasted & very few parts are used, as :

- Total address parts : 12
- Total used parts : 5
- Total unused parts : 7



To avoid such memory wastage, a computer program supports use of threads. **The pointer which points to the higher level node or the parent node is called thread.**

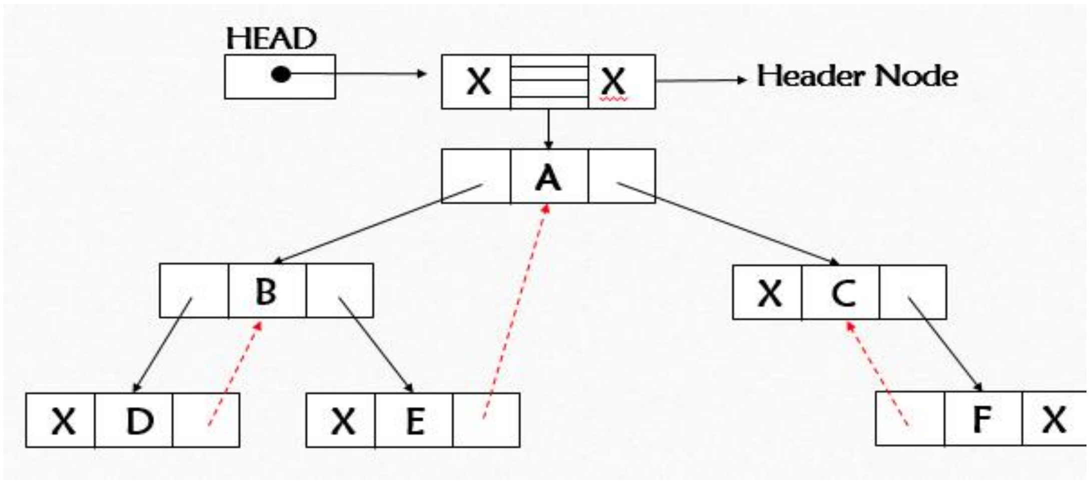
The binary tree with such thread is called threaded tree.

In threaded tree, forward link i.e. from higher level to lower level (Parent to child) is shown by a solid line (   ) and reverse link i.e. from lower level to higher level (Child node to parent node) is shown by a dotted line (------).

Such threaded tree is very useful while performing In-order traversing **(left-root-right)**.

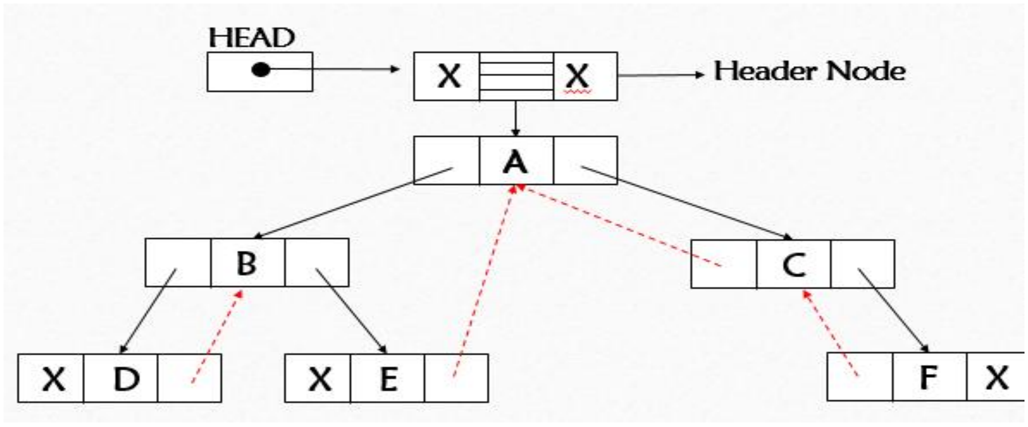
1. **One way In-order threading :** In this type of threaded tree, each higher level node receives atleast one reverse link from its sub-node.

   Total address parts : 12
   Total used parts : 8
   Total unused parts : 4



2. **Two way In-order threading :** In this type of threaded tree, each higher level node receives multiple reverse links from its sub-node.

   Total address parts : 12
   Total used parts : 9
   Total unused parts : 3



General Tree Introduction :

A general tree is defined to be a non empty finite set T of elements, called nodes such that :

i.) T contains a distinguished element R called the Root of the tree T.

ii.) The remaining elements of T form an ordered collection of Zero or more disjoint trees T1, T2, ---, Tm.

The trees T1, T2, ---, Tm are called sub trees of the root R and the roots of T1, T2, --- , Tm are called successor of R.

A binary tree T' is not a special case of a general tree T. Binary trees and general trees are different objects. The two basic differences are :

i.) A binary tree T' may be empty, but a general tree T is non empty.

ii.) Suppose a node N has only one child, then the child is distinguished as a left child or right child in a binary tree T' , but no such distinction exists in a general tree.