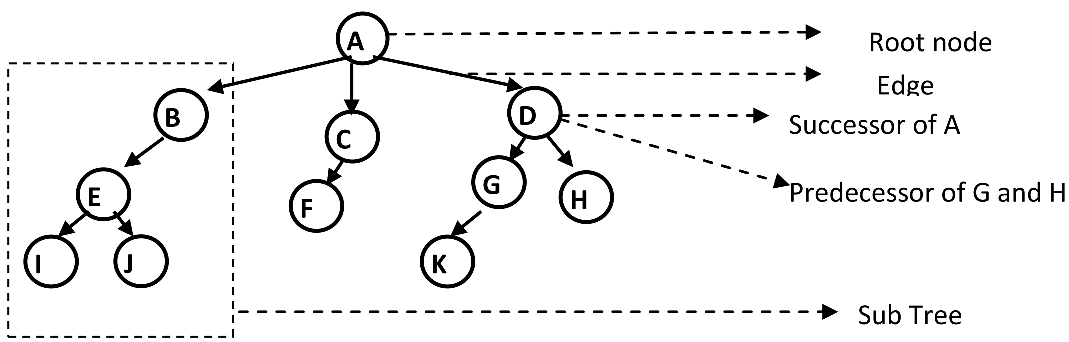
TREE

Tree is a non linear data structure that is used to represents the data in a hierarchical relationship. A node may have multiple successors (children nodes) but only one predecessor (parent node)

Definition: A tree is defined as a finite set of items called nodes and finite set of links, called branches or edges that connect two nodes.

- If the tree is non empty, then the first node is called the **root** node, A root node has no parent
- The tree is called **null tree or empty tree** if it has no node.



[Figure-1: The Tree]

1. Tree Terminology

Degree, In degree , Out degree:

Number of branches associated with a node is called **degree** of this node.

The degree of a node is in-degree + out-degree of this node.

Number of branches directed towards the node is called **in-degree** of this node. Number of branches directed away from the node is called out-degree of this node.

Parent node: A node which has one or more successor is called parent node.

Child node: The node with a predecessor is called child node. Any child node has 1 in-degree.

Sibling : All children of same parent are called siblings.

Leaf node: The node with no out-degree is called leaf node. I,J,F,K,H are leaf nodes in figure-1.

Non-leaf node: The node with at least one out degree is called non leaf node. A,B,C,D,E,G are leaf nodes in figure-1.

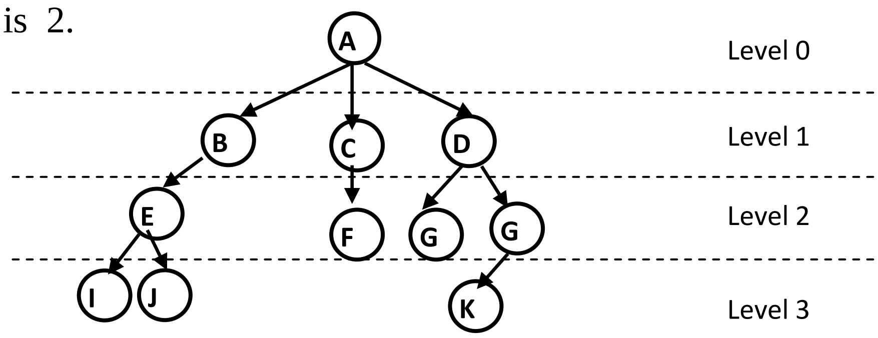
Internal node: The node other than leaf and root is called internal node. B,C,D,E,G

Path : The sequence of edges from source node to a destination node is called path. A path <A,D,G,K> is a path from node A to node K

Level : Level of a node is the distance from the root. The root is at level 0, the children of the root are at level 1 and so on.

Height: Height of a node P is the length (number of edges) of the longest path from the node P to a leaf. All the leaf nodes are in height 0.

Height of tree is the height of the root. Height of the following tree is 3. Height of the node D is 2.

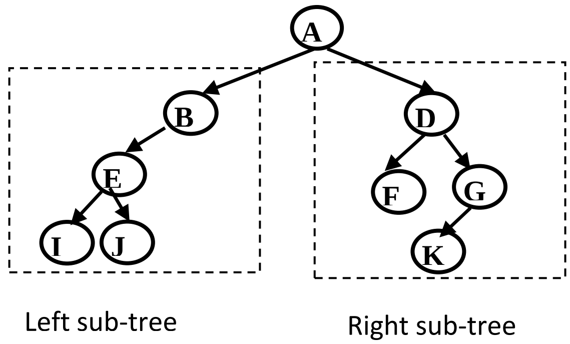


<u>Types of Tree</u>

1. Binary Tree (Complete Binary Tree, Strict Binary Tree etc.)
2. Binary search Tree
3. Height Balanced Tree
4. Expression Tree
5. General tree
6. Threaded Binary Tree
7. Multi- way search Tree: (B Tree , B + Tree , 2-3 Tree etc)

2. Binary Tree

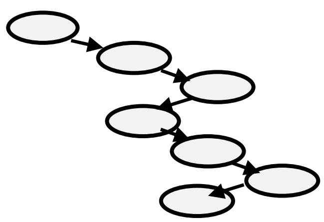
Binary tree is a tree, in which any node contains at most two sub-trees. In binary tree, maximum out-degree of any node is 2. That means, each node can have either 0,1, or 2 sub-trees.



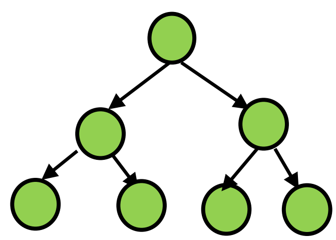
Height of binary Tree:

Maximum height of binary tree with n nodes = n–1. This is when each level contains one node.

Minimum height of tree with n nodes = $\lfloor \log_2 n \rfloor$. This is when each level contains maximum number of possible node.



[Figure 4 : Tree of 7 nodes has height 6]



[Figure 5 : Tree of 7 nodes has height 2]

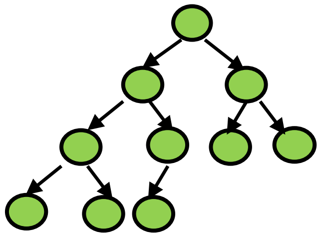
Notes: Number of binary tree possible with n number of node = $\frac{(2n)!}{(n+1)!*n!}$

Strict binary tree: A binary tree is called strict binary tree if any of its node has either 0 or 2 sub-trees. A strict binary tree with **L leaf nodes has 2*L -1** nodes.

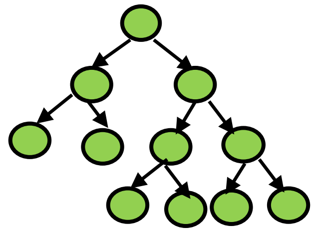
Complete Binary tree /Almost Complete Binary tree:

The binary tree is called **almost complete binary tr**ee if all its levels, except the last, have the maximum number of possible nodes, and at the last level, the nodes are appeared as left as possible. A **complete binary tree** has maximum number of nodes in every level including the last level.

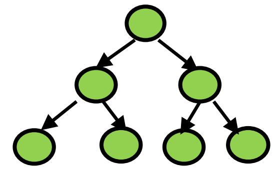
In most literature, almost complete binary tree is also called as complete binary tree.



[Complete Binary tree]



[Not a Complete Binary tree]



[Complete Binary tree]

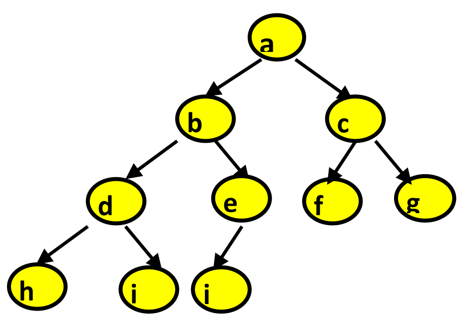
For a complete binary tree of height h,
Maximum number of nodes = $2^{h+1}-1$
Minimum number of nodes = 2^{h}
Minimum number of leaf nodes = 1
Maximum number of leaf nodes = 2^{h}

3. Representation of binary tree:

I. Using Array.   II. Using Linked list

3.1 : Array representation:

The root of the tree is in the 1st position of the array. The nodes in next level (level-1) are stored sequentially in next position in the array, and so on. See following tree and its array representation.



[1]	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**[10**
a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**

For a complete binary tree, for a node at index k

The index of its parent = $\left\lfloor \frac{k}{2} \right\rfloor$

The index of its left child = 2*k

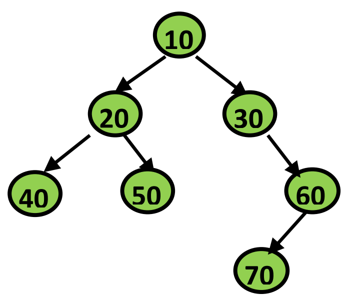
The index of its right child = 2*k + 1

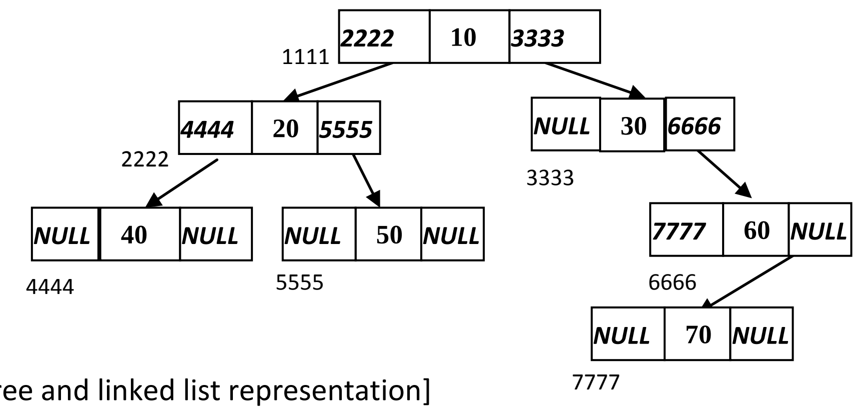
3.2: Linked list representation:

A node in a binary tree consists of least 3 parts.

1. Data : Contains actual information about the node
2. left : Contains a pointer that points to address of left child
3. right : Contains a pointer that points to address of right child

Pointer to left child	Data	Pointer to right child





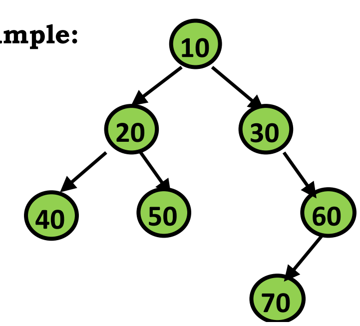
[Figure: The binary tree and linked list representation]

4. Binary tree traversal

There are three ways to traverse a tree

1. Pre order traversal:	**2. In order traversal:**	**3. Post order traversal:**
1. Visit the **Root**	1. Traverse the **Left sub tree**	1. Traverse the **Left sub tree**
2. Traverse the **Left sub tree**	2. Visit the **Root**	2. Traverse the **Right subtree**
3. Traverse the **Right subtree**	3. Traverse the **Right subtree**	3. Visit the **Root**

Example:



Pre order traversal	**:**	**10**	**20**	**40**	**50**	**30**	**60**	**70**
In order traversal	**:**	**40**	**20**	**50**	**10**	**30**	**70**	**60**
Post order traversal	**:**	**40**	**50**	**20**	**70**	**60**	**30**	**10**

[Figure: Binary tree and its Pre Order, In order and Post Order traversal]