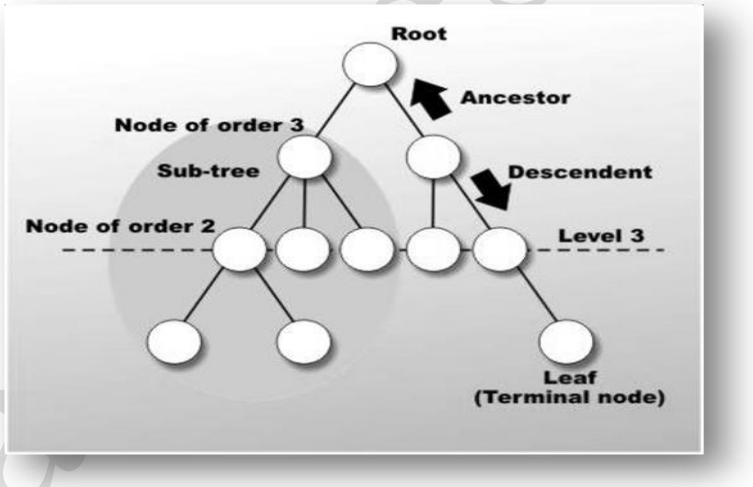
# TREE

## Trees Basic Concepts:

A **tree** is a non-empty set one element of which is designated the root of the tree while the remaining elements are partitioned into non-empty sets each of which is a sub-tree of the root. A tree T is a set of nodes storing elements such that the nodes have a parent-child relationship that satisfies the following

- If T is not empty, T has a special tree called the root that has no parent.
- Each node v of T different than the root has a unique parent node w; each node with parent w is a child of w.

Tree nodes have many useful properties. The **depth** of a node is the length of the path (or the number of edges) from the root to that node. The **height** of a node is the longest path from that node to its leaves. The height of a tree is the height of the root. A **leaf node** has no children -- its only path is up to its parent.




## Tree Terminology:

### Leaf node

A node with no children is called a leaf (or external node). A node which is not a leaf is called an internal node.

**Path:** A sequence of nodes $n_1, n_2, \ldots, n_k$, such that $n_i$ is the parent of $n_{i+1}$ for i = 1, 2,. . ., k - 1. The length of a path is 1 less than the number of nodes on the path. Thus there is a path of length zero from a node to itself.

**Siblings:** The children of the same parent are called siblings.

**Ancestor and Descendent** If there is a path from node A to node B, then A is called an ancestor of B and *B* is called a descendent of A.

**Subtree:** Any node of a tree, with all of its descendants is a subtree.

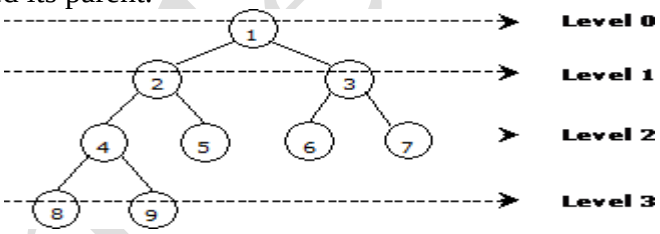
**Level:** The level of the node refers to its distance from the root. The root of the tree has level O, and the level of any other node in the tree is one more than the level of its parent.

**The maximum number of nodes at any level is $2^n$.**

**Height**: The maximum level in a tree determines its height. The height of a node in a tree is the length of a longest path from the node to a leaf. The term depth is also used to denote height of the tree.

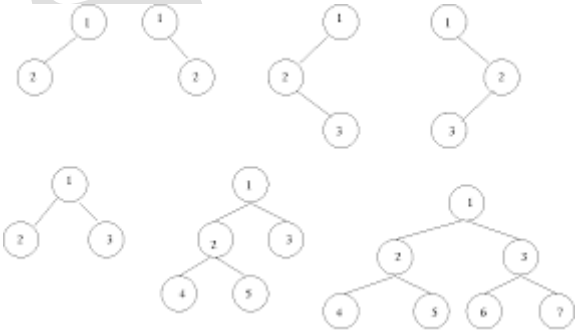
**Depth:**The depth of a node is the number of nodes along the path from the root to that node.

**Assigning level numbers and Numbering of nodes for a binary tree:** The nodes of a binary tree can be numbered in a natural way, level by level, left to right. The nodes of a complete binary tree can be numbered so that the root is assigned the number 1, a left child is assigned twice the number assigned its parent, and a right child is assigned one more than twice the number assigned its parent.




**Binary Tree:**

In a binary tree, each node can have at most two children. A binary tree is either **empty** or consists of a node called the **root** together with two binary trees called the **left subtree** and the **right subtree**.



(Different cases of binary trees)

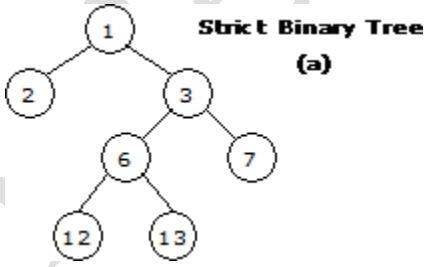
**Properties of Binary Trees:**

Some of the important properties of a binary tree are as follows:

1. If *h* = height of a binary tree, then
    a. Maximum number of leaves = $2^h$
    b. Maximum number of nodes = $2^{h+1} - 1$
2. If a binary tree contains m nodes at level l, it contains at most 2m nodes at level l + 1.
3. Since a binary tree can contain at most one node at level 0 (the root), it can contain at most $2^l$ node at level l.
4. The total number of edges in a full binary tree with n node is n - 1.

**Strictly Binary tree:**

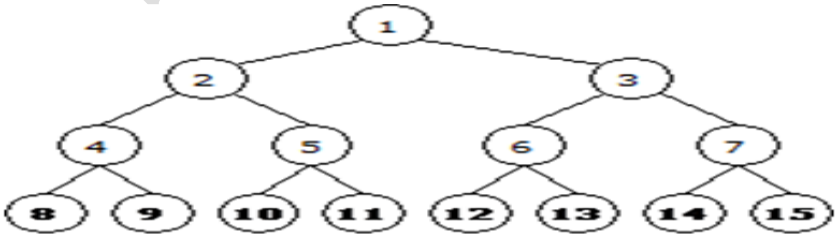
If every non-leaf node in a binary tree has nonempty left and right subtrees, the tree is termed a strictly binary tree. Thus the tree is strictly binary. A strictly binary tree with n leaves always contains 2n - 1 node.




**Full Binary Tree:**

A full binary tree of height h has all its leaves at level h. Alternatively all non leaf nodes of a full binary tree have two children, and the leaf nodes have no children.

A full binary tree with height *h* has $2^{h+1} - 1$ node. A full binary tree of height h is a *strictly binary tree* all of whose leaves are at level h.




For example, a full binary tree of height 3 contains $2^{3+1} - 1 = 15$ nodes.

**Perfect Binary Tree**:

A Binary tree is Perfect Binary Tree in which all internal nodes have two children and all leaves are at same level.
Following are examples of Perfect Binary Trees.

```
        18
      /    \
    15      30
   /  \     /  \
  40   50  100  40
```

or

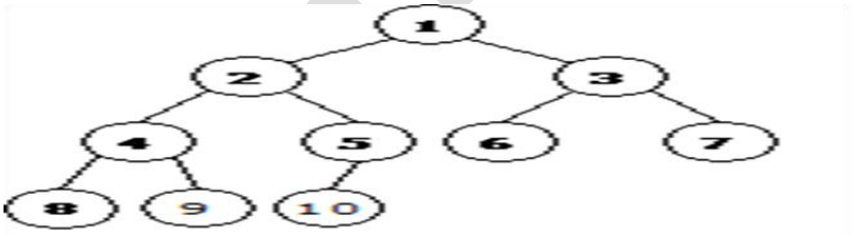
```
     18
   /    \
 15      30
```

A Perfect Binary Tree of height h (where height is number of nodes on path from root to leaf) has $2^h - 1$ node.

Example of Perfect binary tree is ancestors in family. Keep a person at root, parents as children, parents of parents as their children.

**Almost Complete Binary Tree:**

A binary tree with *n* nodes is said to be **almost complete** if it contains all the first *n* nodes of the above numbering scheme.
A almost complete binary tree of height h looks like a full binary tree down to level h-1, and the level h is filled from left to right.




**Balanced Binary Tree:**

A binary tree is balanced if height of the tree is O(Log n) where n is number of nodes. For Example, AVL tree maintain O(Log n) height by making sure that the difference between heights of left and right subtrees is 1. Red-Black trees maintain O(Log n) height by making sure that the number of Black nodes on every root to leaf paths are same and there are no adjacent red nodes. Balanced Binary Search trees are performance wise good as they provide O(log n) time for search, insert and delete.

## Memory Representation of Binary Trees:

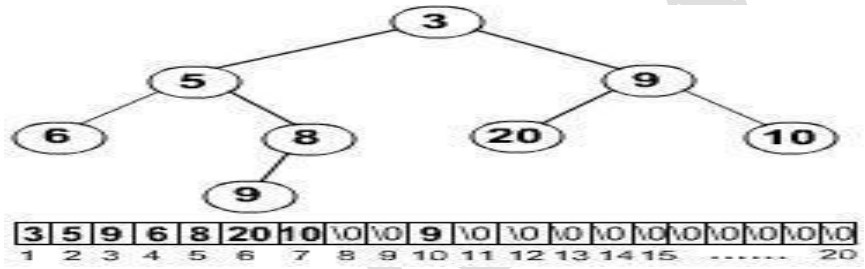
1. Array Representation of Binary Tree
2. Pointer-based(Linked List Based)

### Array Representation of Binary Tree:

A single array can be used to represent a binary tree.

For these nodes are numbered / indexed according to a scheme giving 0 to root. Then all the nodes are numbered from left to right level by level from top to bottom. Empty nodes are also numbered. Then each node having an index i is put into the array as its $i^{th}$ element.

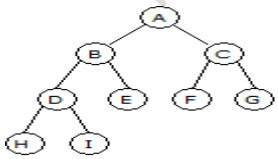
In the figure shown below the nodes of binary tree are numbered according to the given scheme.

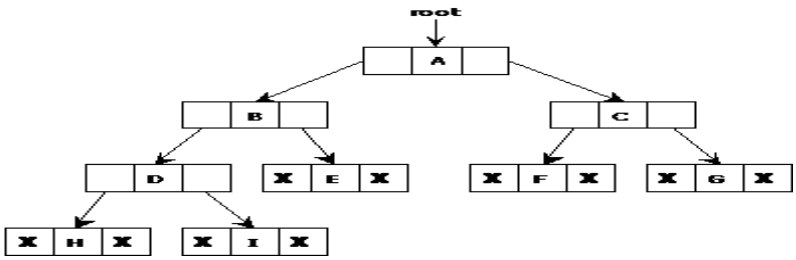



The figure shows how a binary tree is represented as an array. The root 3 is the $0^{th}$ element while its left child 5 is the 1st element of the array. Node 6 does not have any child so its children i.e. $7^{th}$ and $8^{th}$ element of the array are shown as a Null value.
It is found that if n is the number or index of a node, then its left child occurs at $(2n + 1)^{th}$ position and right child at $(2n + 2)^{th}$ position of the array. If any node does not have any of its child, then null value is stored at the corresponding index of the array.
**NB: Elements can be stored from $0^{th}$ index of array like we used to store using array but the index of left child and right child calculation will be different.**

### Linked List Representation of Binary Tree (Pointer based):

Binary trees can be represented by links where each node contains the address of the left child and the right child. If any node has its left or right child empty then it will have in its respective link field, a null value. A leaf node has null value in both of its links.



**Properties of a binary tree:**

1. In a complete binary tree, the number of nodes at depth d is $2^d$.

   **Proof:**
   there are $2^0$ nodes at depth 0. The root node is the single node present at level 0. So at level 0 the number of node is 1.
   if there are $2^d$ nodes at depth d, then there are $2^{d+1}$ nodes at depth d+1. Each node at depth d has 2 children, so there are $2*2^d$ nodes at d+1, and $2*2^d = 2^{d+1}$.

2. In a complete binary tree the **height of the tree is log(n+1).**

   **Proof:**
   The number of nodes in the tree is the sum of the number of nodes at each level:
   $n = 2^h+2^{h-1}+2^{h-2}+...+2^1+2^0 = 2^{h+1} - 1$
   $n = 2^{h+1} - 1$
   $n+1 = 2^{h+1}$
   $\log(n+1) = \log(2^{h+1})$
   $\log(n+1) = h+1$
   **$h = \log(n+1) - 1.$**

   **As the height of a tree is same as its depth. So $d = \log(n+1) - 1.$**

   Where d is the depth of a tree.

**Programming is a skill best acquired by practice and example rather than from books.**

With Regards
**Yours Sam Sir**