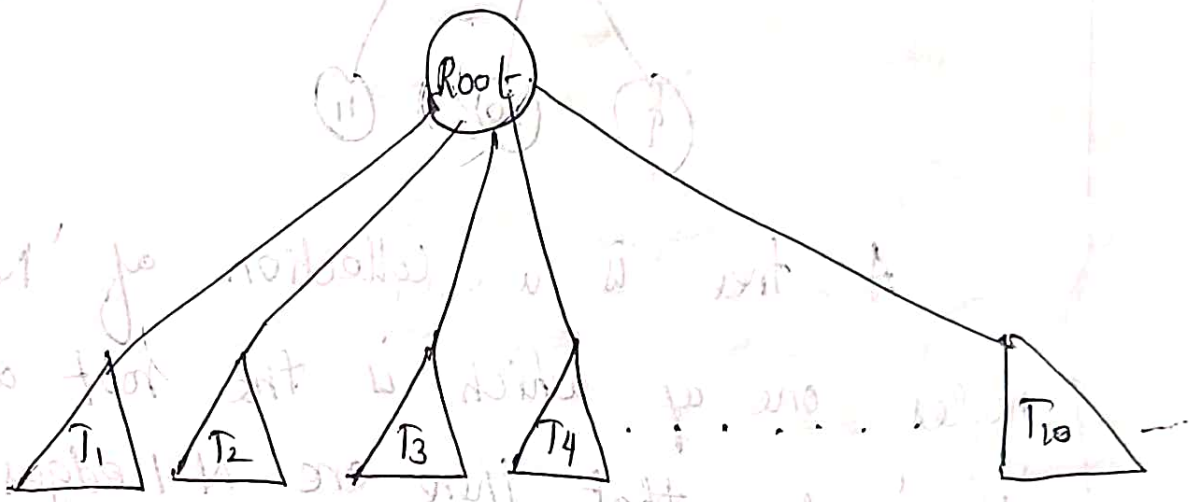
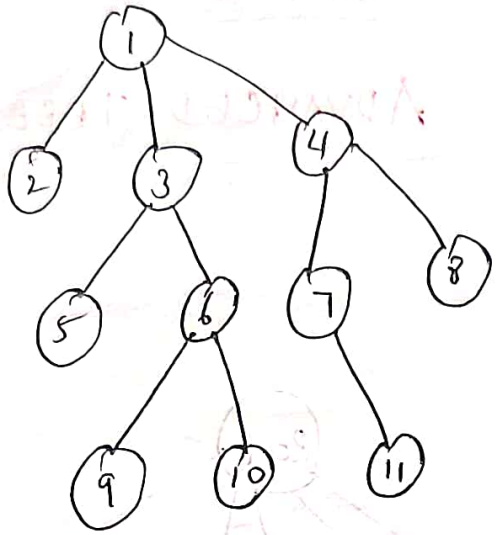
TREE



* A Tree is a collection of nodes, the collection can empty otherwise a tree consist of distinguished node 'R' called the Root.

* Zero or more non empty sub-trees $T_1, T_2 \ldots T_k$ each of whose roots are connected by directed edge from 'R', the root of each subtree is said to be child of 'R' and 'R' is parent of each sub-tree root.



A tree is a collection of 'N' nodes one of which is the root and 'N-1' edges that there are N-1 edges follow from the fact that each edge connects some node to its parents and every node except the root has one parent.

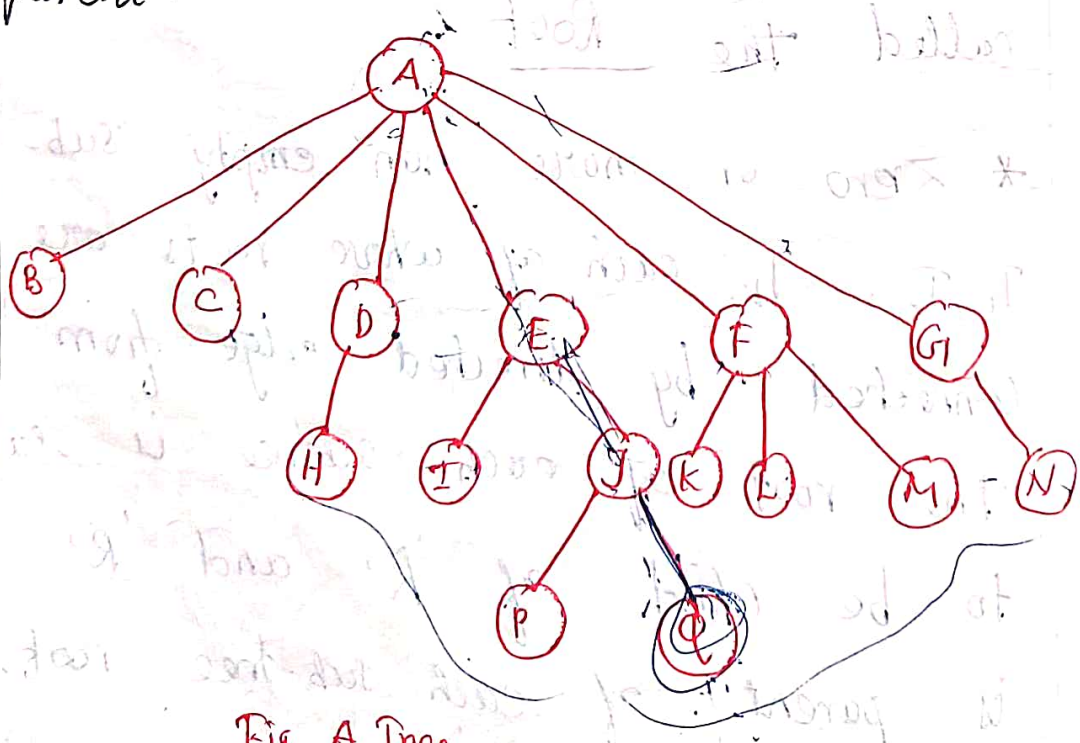


Fig. A Tree

Tree Terminology

NODE:-

Item of information is called as node

Eg. In above tree A, B, C. - Q

Root:-

Root doesn't have any parent node.

Eg. A is Root node

Leaves:-

The nodes doesn't have child nodes. Ex. B, C, H, I, P, Q, K, L, M, N

Siblings:-

Node with same parent are called siblings.

Ex. {B, C, D, E, F, G}, {I, J}, {K, L, M}, {P, Q}

Path:-

A path from node n_1 to n_k is defined as sequence of nodes $n_1, n_2 \ldots n_k$ such that n_i is the parent of n_{i+1} for $1 \leq i \leq k$

Ex. path of A to L is AFL

Length :-

The length of the path is number of edges on the path. There is a path of length 0 from every node to itself

Ex. A to H = 2 edges

A to P = 3 edges

Depth :-

For any node the depth of 'n_i' is the length of the unique path from the root to n_i. The root is at depth 0.

Ex. Depth of Q = 3

Height of leaf nodes = 0

Height of E = 2

Height :-

The height of 'n_i' is the length of longest path from n_i to a leaf. Thus all leaves are at height 0. The height of the tree is height of the root node.

Eg Depth of E = 1

Height of E = 2

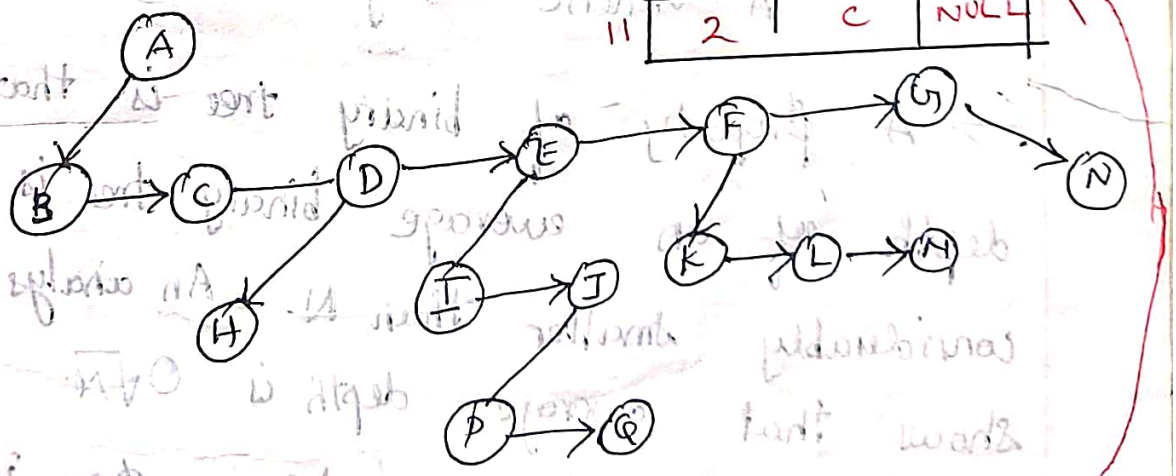
First child / Next siblings representation for general trees

Implementation of Trees

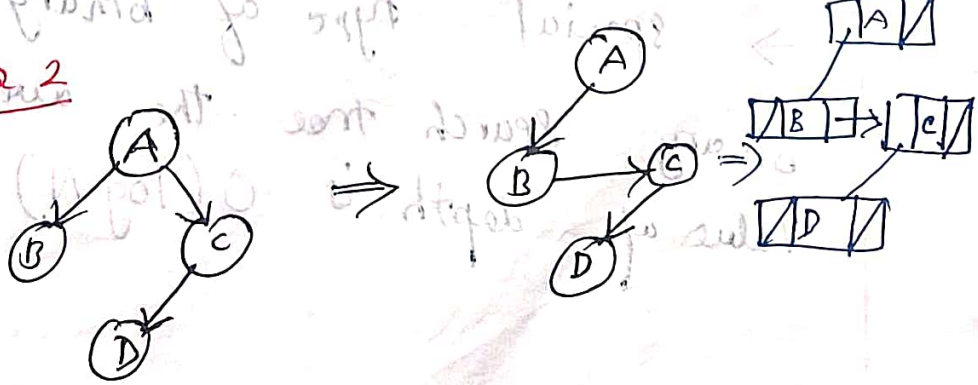
```
typedef struct TreeNode *ptrNode;
struct Treenode
{
    Element type Element;
    ptrNode firstchild;  // left child
    ptrNode Nextsibling; // right child
}
```

	Left	Label	Right
2	NULL	D	NULL
5	NULL	B	11
10	5	A	NULL
11	2	C	NULL

Example 1



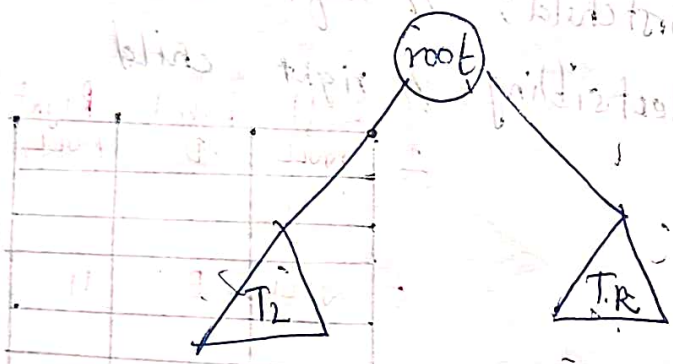
Example 2



Binary Tree ADT

→ A binary tree is a tree in which no node can have more than two children.

→ A binary tree consists of a root and two subtrees, T_L and T_R both of which could possibly be empty.



A Generic binary tree

→ A property of binary tree is that the depth of an average binary tree is considerably smaller than N. An analysis shows that average depth is $O\sqrt{N}$

→ special type of binary tree ie) Binary search tree the average value of depth is $O(\log N)$

→ Worst case of binary search tree depth can be as large as N−1

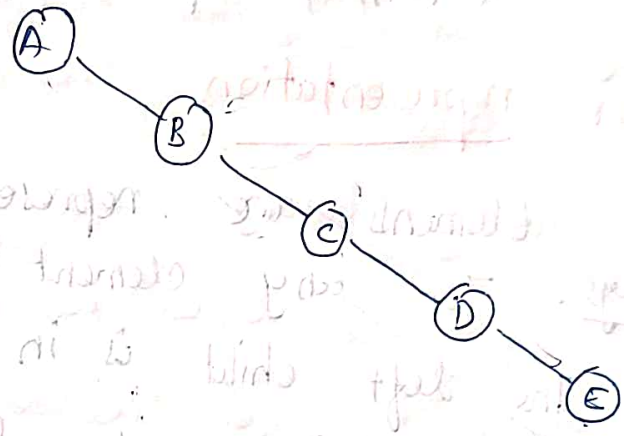
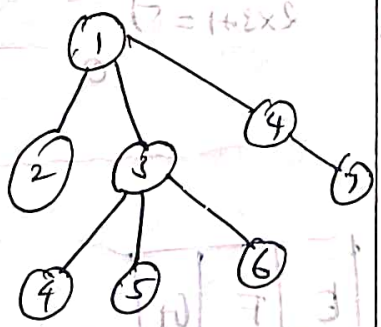
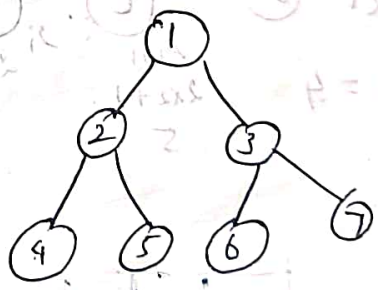
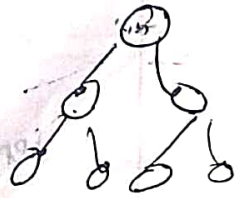


Fig - worst case binary Tree

General Tree	Binary Tree
* It has any no of children * Example 	* It doesn't have more than two children. * Example 

Node declaration of Binary Tree



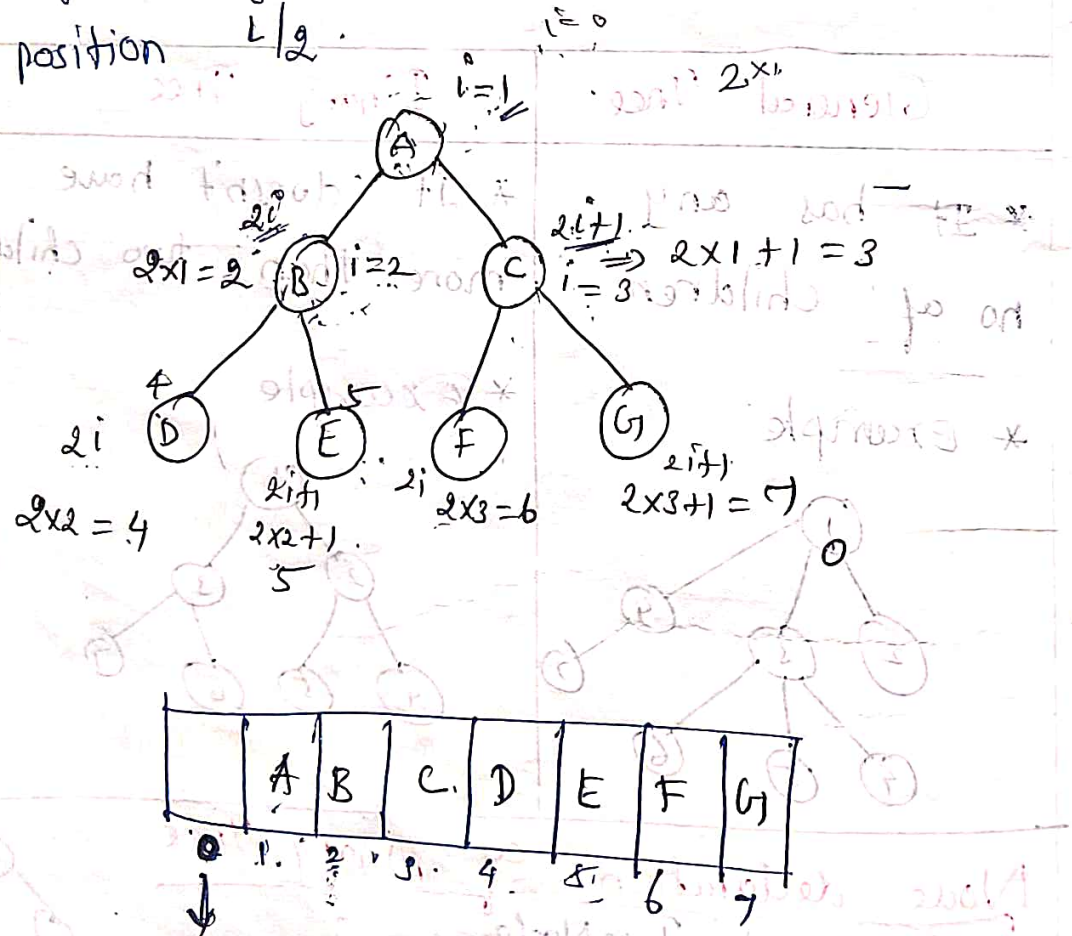
```
struct TreeNode
{
    int Element;
    struct TreeNode *left;
    struct TreeNode *right;
};
```

Representation of Binary Tree

* Linear representation
* Linked representation

Linear representation

Elements are represented using arrays. For any element in position 'i' The left child is in position $2i$ right child $2i+1$ and parent in the position $i/2$.



Sentinal node

Formula to find a parent node of given element $i/2$.

Example

Find parent node of E

E stored in the location 5 ie) i = 5

Parent of E = i/2 = 5/2 = 2.5

Parent of E stored in the location 2 so the parent is B

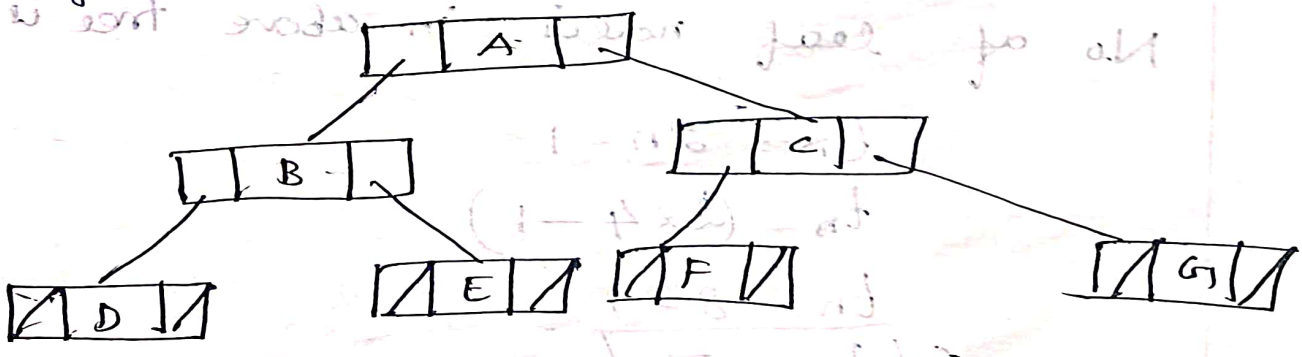
E is B

Linked representation :-

Elements are represented using pointers. Each node in linked representation has three fields.

* pointer to left sub tree
* Data field
* pointer to right sub tree

For leaf node both pointers are assigned as NULL



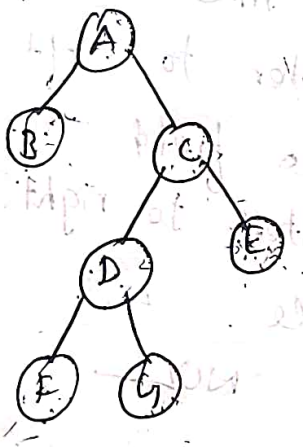
Strictly Binary Tree

If every non-leaf node in a binary tree has non empty left and right subtrees. The tree is termed as strictly binary tree.

To find how many number of nodes in strictly binary tree is

$$t_n = (2n-1)$$

n - no of leaves



No of leaf nodes in above tree is 4

$$t_n = 2n-1$$

$$t_n = (2 \times 4 - 1)$$

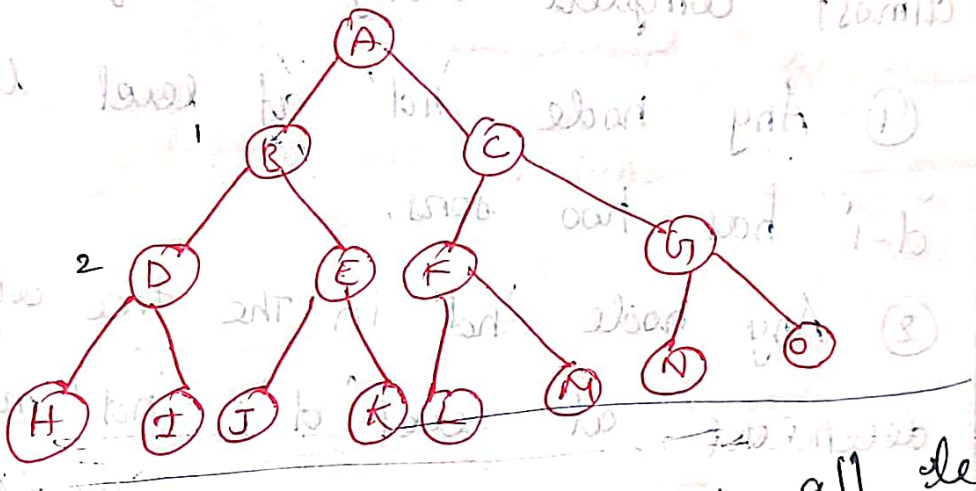
$$t_n = 8 - 1$$

$$\boxed{t_n = 7}$$

Total no of nodes in above tree is 7.

Complete Binary Tree

A complete binary tree of depth 'd' is strictly binary tree all whose leaves are at level 'd'.



In above tree depth of all leaves are same. So the depth of the tree is $d = 3$

Find total no of nodes in above tree depth (d) is given as 3

$$t_n = 2^{d+1} - 1 \quad (1)$$

$$t_n = 2^{3+1} - 1$$

$$t_n = 2^4 - 1$$

$$t_n = 16 - 1 = 15$$

$$\boxed{t_n = 15}$$