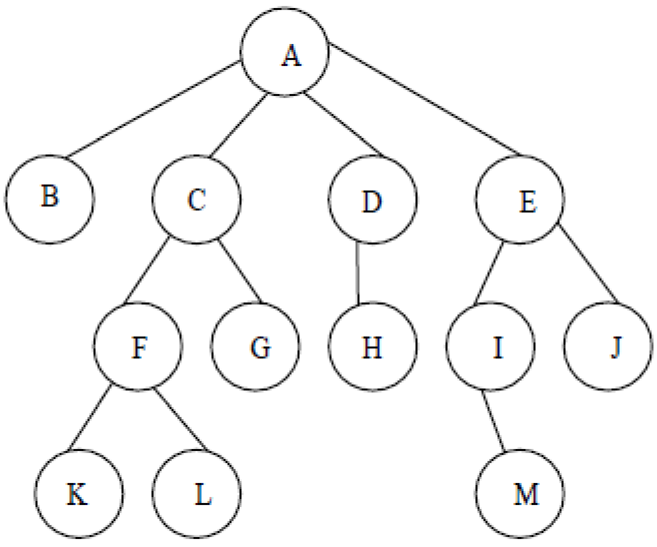
UNIT III – TREES

Trees - Tree Terminology – Binary Trees –Binary Tree Traversals – Expression Tree - Binary Search Trees – Priority Queues (Binary Heap). Use cases in biomedical imaging, power grid modeling, circuit routing.

3.1 TREE

A tree is a finite set of one or more nodes such that there is a specially designated node called the Root, and zero or more non empty sub trees T1, T2, … Tk, each of whose roots are connected by a directed edge from Root R.



2.1 TREE TERMINOLOGY

NODE : Item of Information.

ROOT : A node which does not have a parent. Here, Root is A.

LEAF : A node which does not have children is called leaf or Terminal node. Here B, K, L, G, H, M, J are leafs.

SIBLINGS : Children of the same parents are said to be siblings, Here B, C, D, E are siblings, F, G are siblings. Similarly I, J & K, L are siblings.

PATH : A path from node n1 to nk is defined as a sequence of nodes n1, n2,n3

......nk such that ni is the parent of ni+1. for 1 = i to k . There is exactly only one

path from each node to root.

Here, path from A to L is A, C, F, L. where A is the parent for C, C is the parent of F and F is the parent of L.

LENGTH :

The length is defined as the number of edges on the path. Here, the length for the path A to L is 3.

DEGREE :

The number of sub trees of a node is called its degree. Here Degree of A is 4 Degree of C is 2

The degree of the tree is the maximum degree of any node in the tree. Here, the degree of the tree is 4.

LEVEL :

The level of a node is defined by initially letting the root be at level one, if a node is at level L then its children are at level L + 1. Level of A is 1. Level of B, C, D, is 2. Level of F, G, H, I, J is 3 Level of K, L, M is 4.

DEPTH :

For any node n, the depth of n is the length of the unique path from root to n. The depth of the root is zero. Here, Depth of node F is 2. Depth of node L is 3. **HEIGHT :** For any node n, the height of the node n is the length of the longest path from n to the leaf. The height of the leaf is zero Here, Height of node F is 1. Height of L is 0.

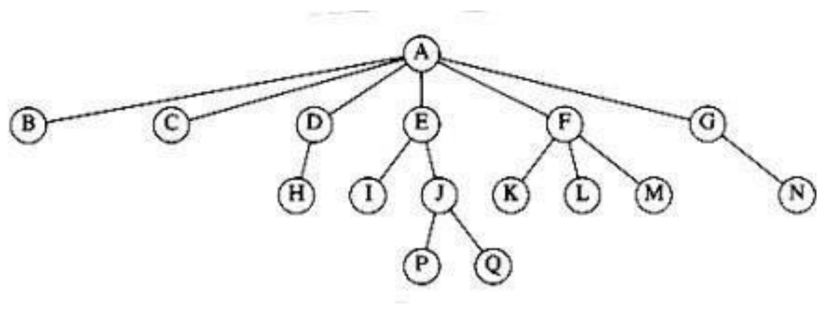
- The height of the tree is equal to the height of the root
- Depth of the tree is equal to the height of the tree.

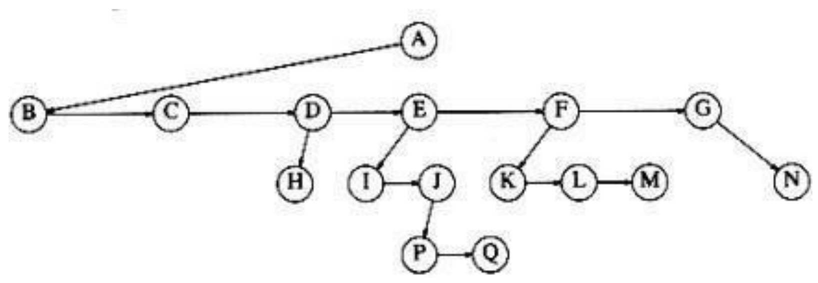
Implementation of Trees

One way to implement a tree is creating a node and in each node, store data, a pointer to each child of the node. Keep the children of each node in a linked list of tree nodes.

```
class Node:

Node* createNode(int data)

{

    Node* newNode = new Node;

    newNode->data = data;

    newNode->left = newNode->right = nullptr;

    return newNode;

}
```

First child/next sibling representation of the tree





Advantages of Trees

1. Represents **hierarchical data** efficiently
2. Provides **fast searching, insertion, and deletion** (especially in BST)
3. Data is stored in a **structured and organized manner**
4. Useful for **sorted data representation**
5. Supports **dynamic data** (size can grow or shrink)

Disadvantages of Trees

1. **Complex implementation** compared to arrays and lists
2. Requires **extra memory** for pointers
3. Can become **unbalanced**, reducing efficiency
4. Traversal is **more time-consuming** than linear structures
5. Difficult to implement and debug

Applications of Trees

1. **File system hierarchy** (folders and subfolders)
2. **Database indexing** (B-trees, B+ trees)
3. **Expression trees** in compilers
4. **Binary Search Trees** for fast searching
5. **Priority queues** using heaps
6. **Decision making systems** (decision trees)
7. **XML/HTML document representation**