Data Structures – CST 201
Module ~ 4

Syllabus

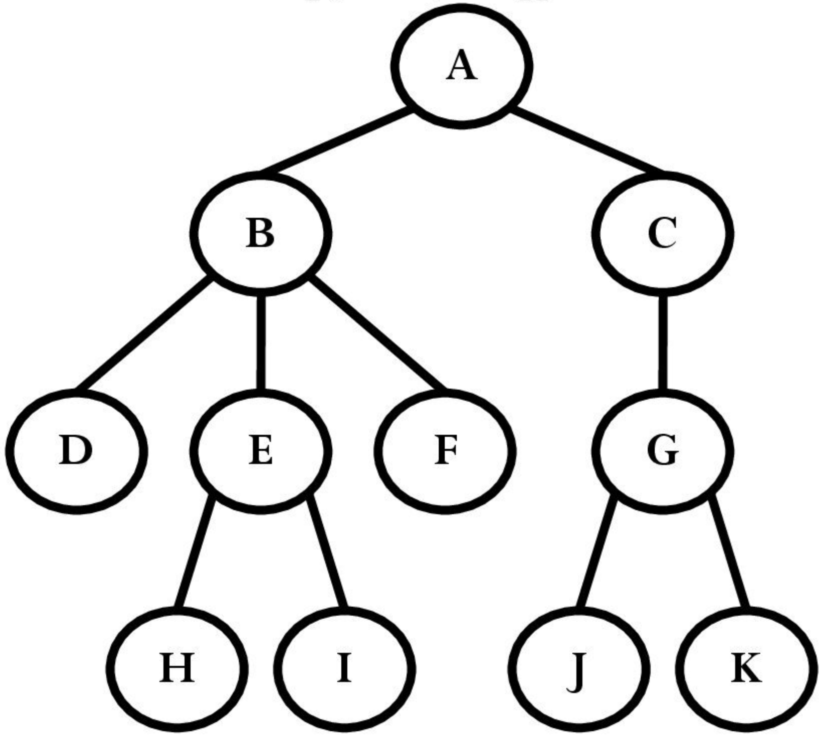
- **Trees and Graphs**
  - **Trees**
    - Binary Trees
      - Binary Tree Representation
      - Binary Tree Operations
      - Binary Tree Traversals
    - Binary Search Trees
      - Binary Search Tree Operations
  - Graphs
    - Representation of Graphs
    - Depth First Search and Breadth First Search on Graphs
    - Applications of Graphs

Tree

- Tree is a nonlinear data structure
- The elements appear in a non linear fashion, which require two dimensional representations.
- Using tree it is easy to organize hierarchical representation of objects.
- Tree is efficient for maintaining and manipulating data, where the hierarchy of relationship among the data is to be preserved.

Tree

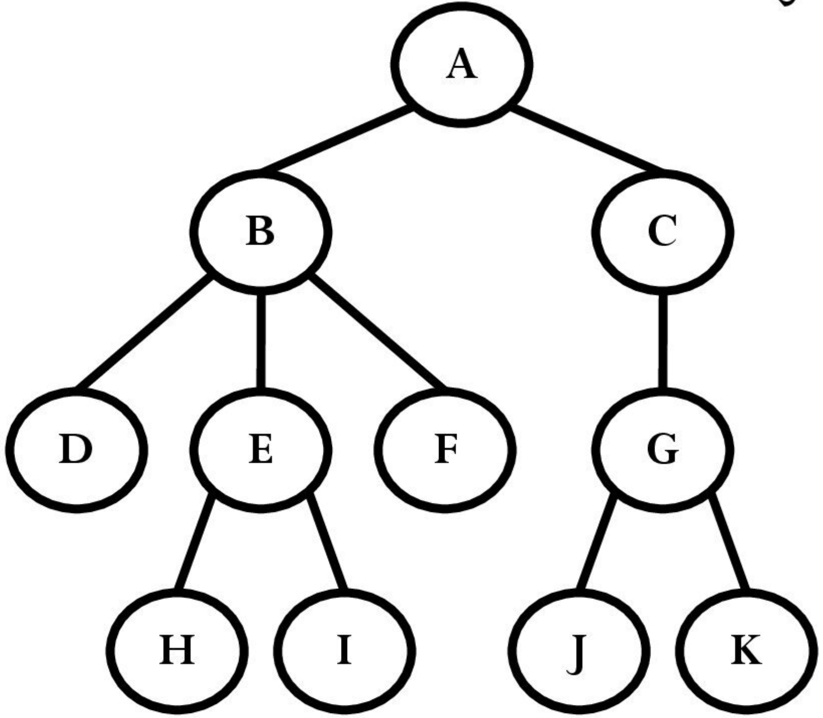
- **Definition:** A tree is a finite set of one or more nodes such that there is a specially designated node called the **root**. The remaining nodes are partitioned into n>=0 disjoint sets T_1, ..., T_n, where each of these sets is a tree. We call T_1, ..., T_n the subtrees of the root.



Tree ~ Terminologies

Tree ~ Terminologies

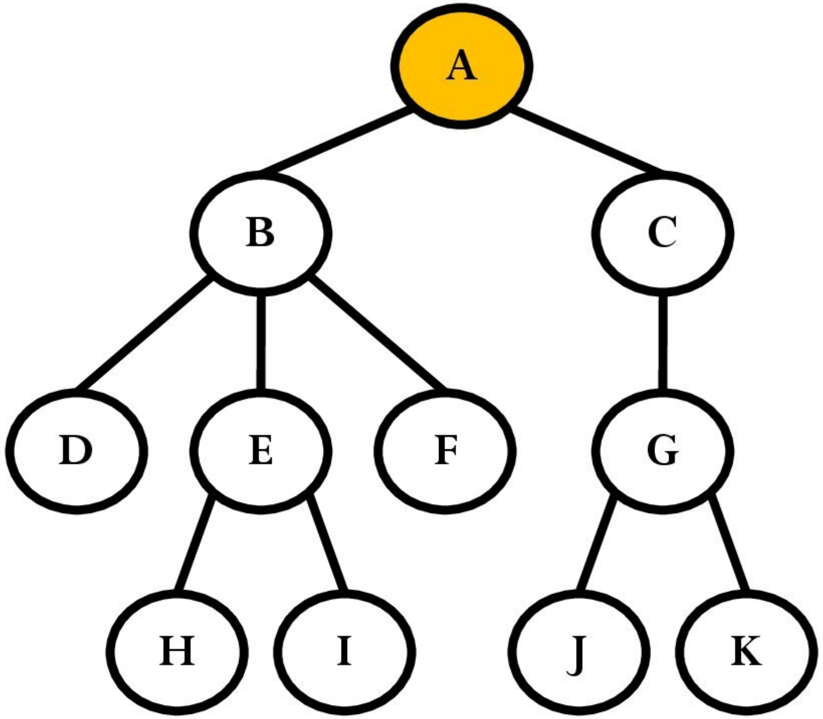
- **Node:** Every individual element in a tree is called a node
- **Edge/Links:** Nodes are connected by using edges



- Here there are 11 nodes and 10 edges
- In any tree with N nodes there will be N-1 edges

Tree ~ Terminologies

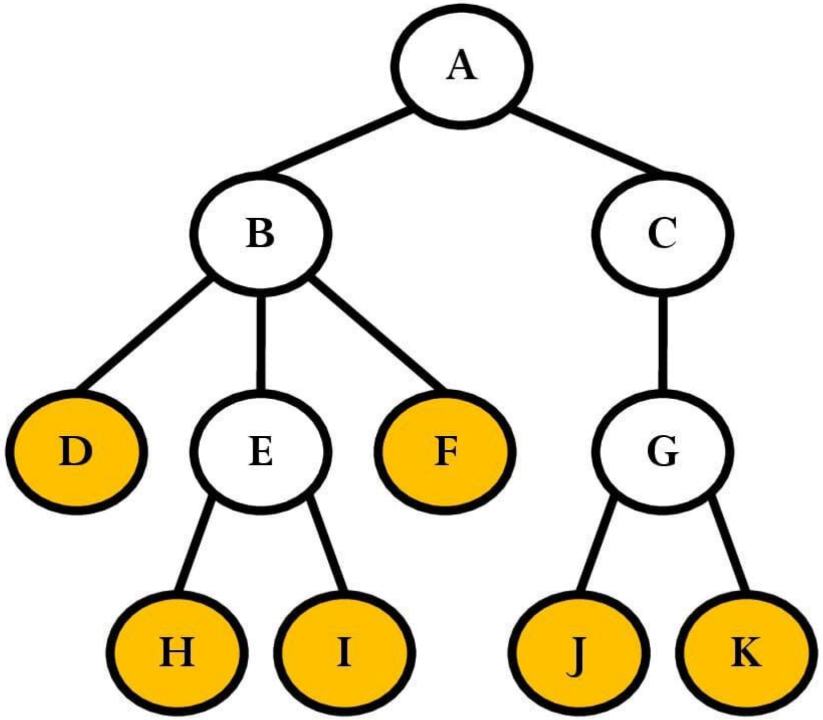
- **Root:** It is the origin of the tree data structure. In any tree, there must be only one root node.



- Here A is the root node
- In any tree the first node is called root node

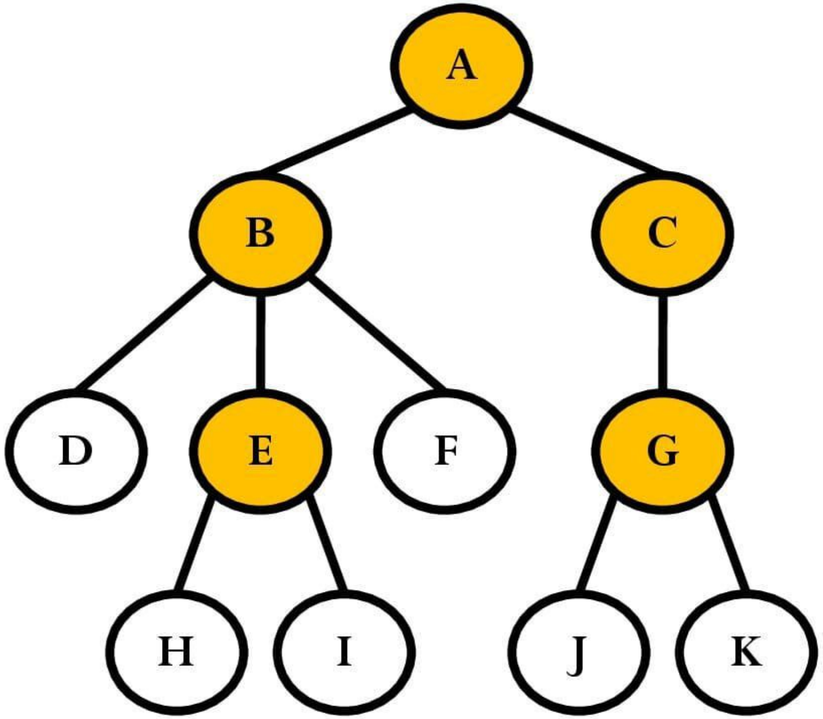
Tree ~ Terminologies

- **Leaf Node/External Node/Terminal Node:**
  - The node which does not have a child is called a leaf node.
  - A node without successor is called a leaf node.



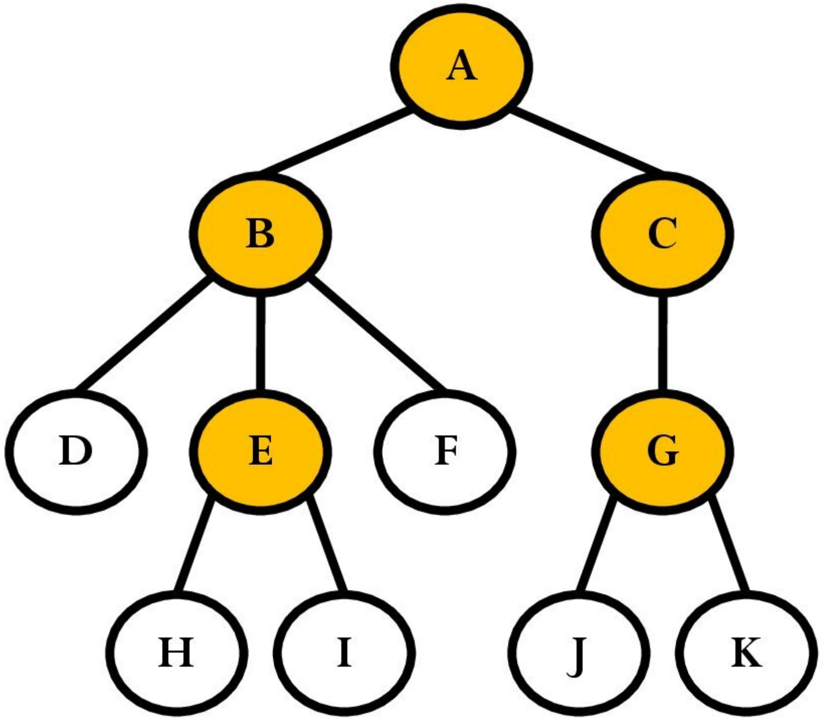
Tree ~ Terminologies

- **Internal Node/Non-terminal Node:**
  - An internal node is a node with atleast one child.
  - Every non leaf node is called Internal node



Tree ~ Terminologies

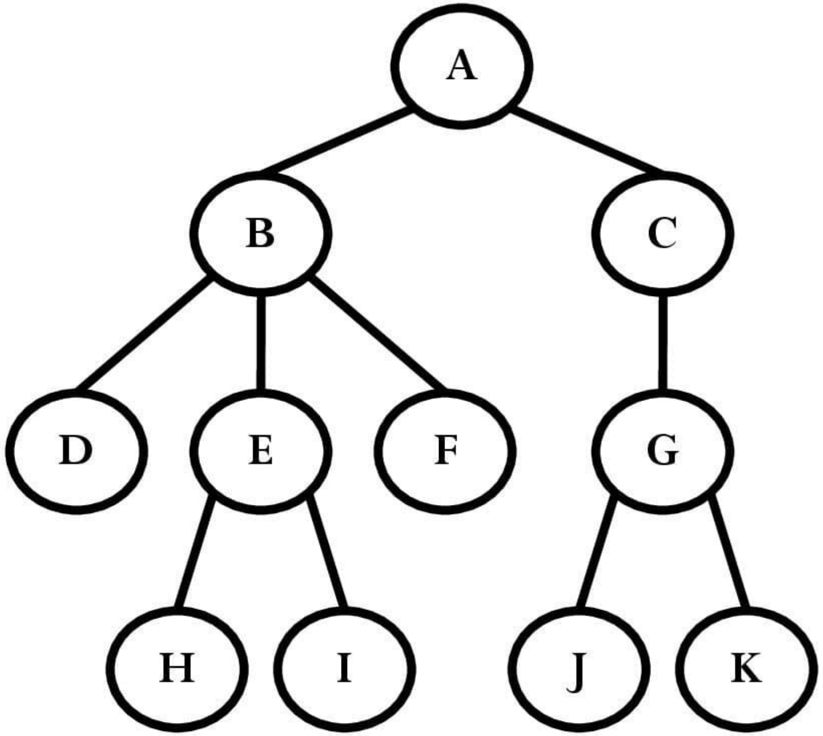
- **Parent:**
  - The node which has child/children.
  - A node which is predecessor of any other node.



- Here A is the parent of B & C. B is the parent of D, E & F.

Tree ~ Terminologies

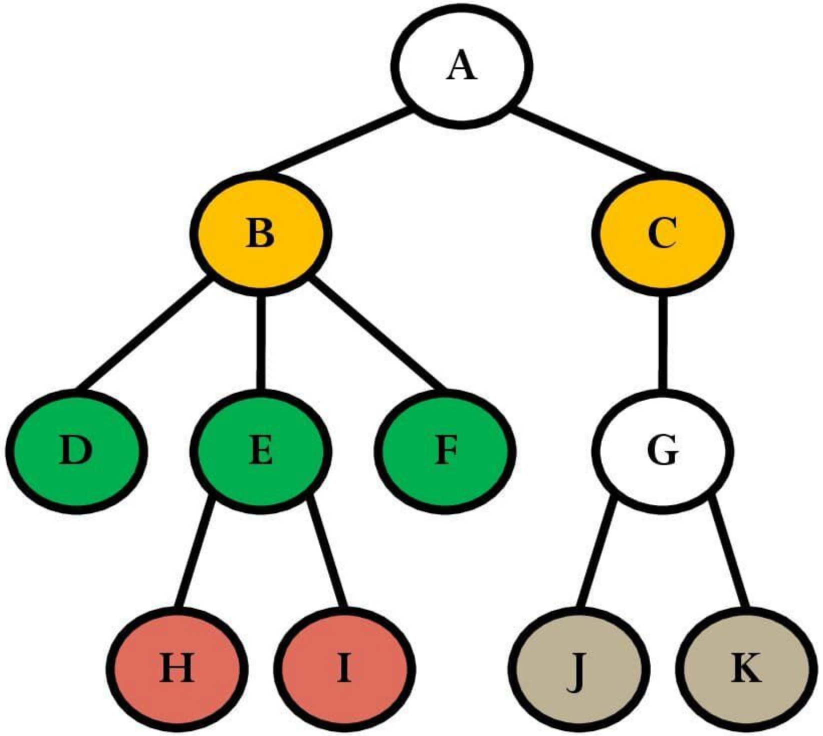
- **Child**: Descendant of any node is called a child node.



- Here children of A are B & C. Children of B are D, E & F.

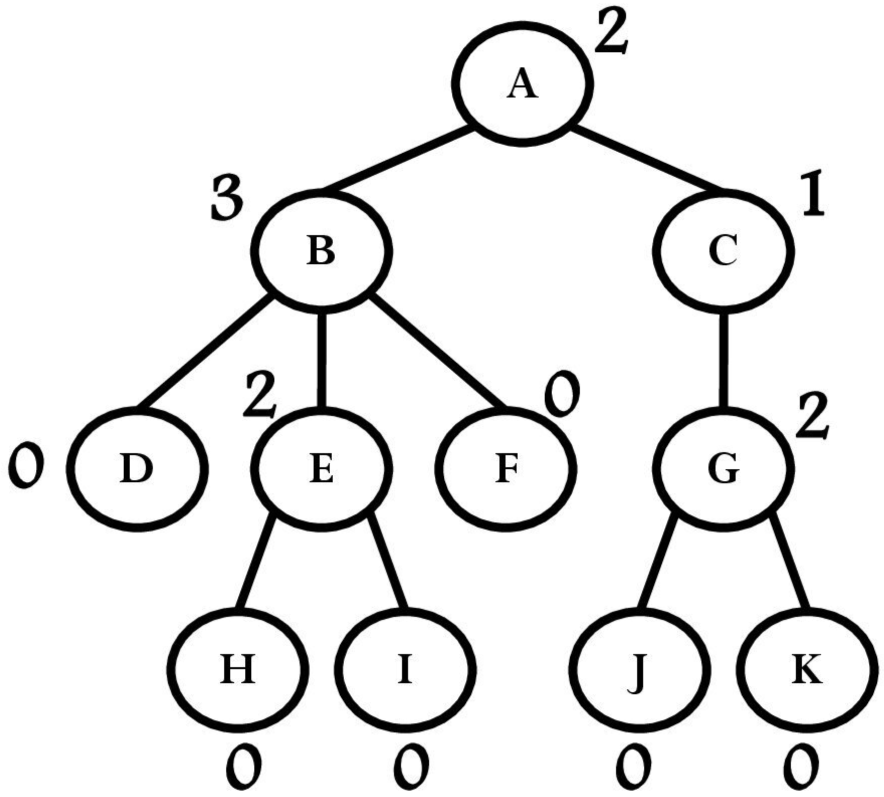
Tree ~ Terminologies

- **Sibling**: The nodes with the same parent are called Sibling nodes.



Tree ~ Terminologies

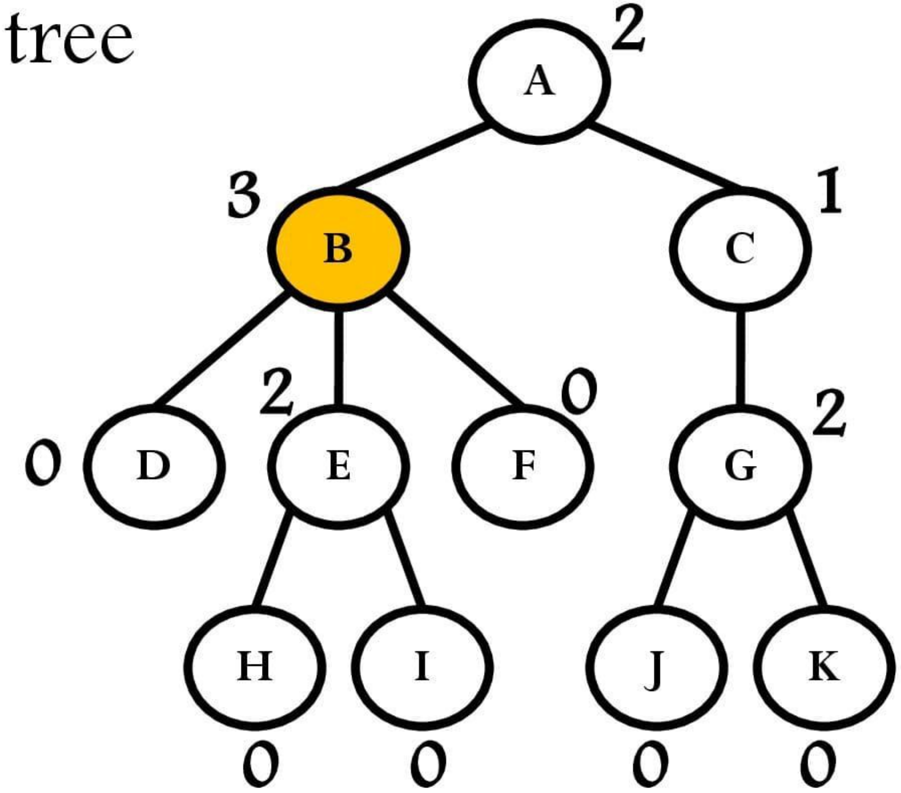
- **Degree of a Node**: Total number of the children of the given node



- Degree of the leaf node is 0.

Tree ~ Terminologies

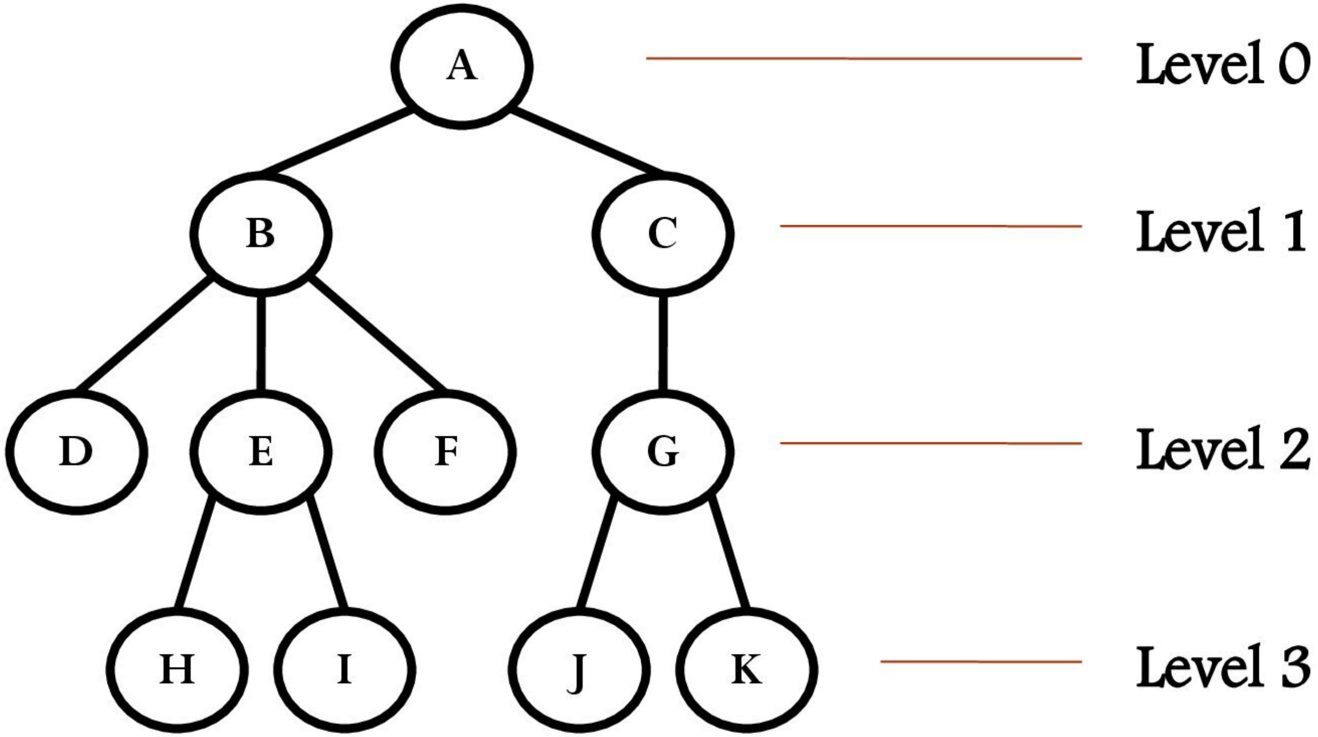
- **Degree of a Tree:** It is the maximum degree of nodes in a given tree



- Here maximum degree is for node B. Its degree is 3.
- So the degree of the given tree is 3

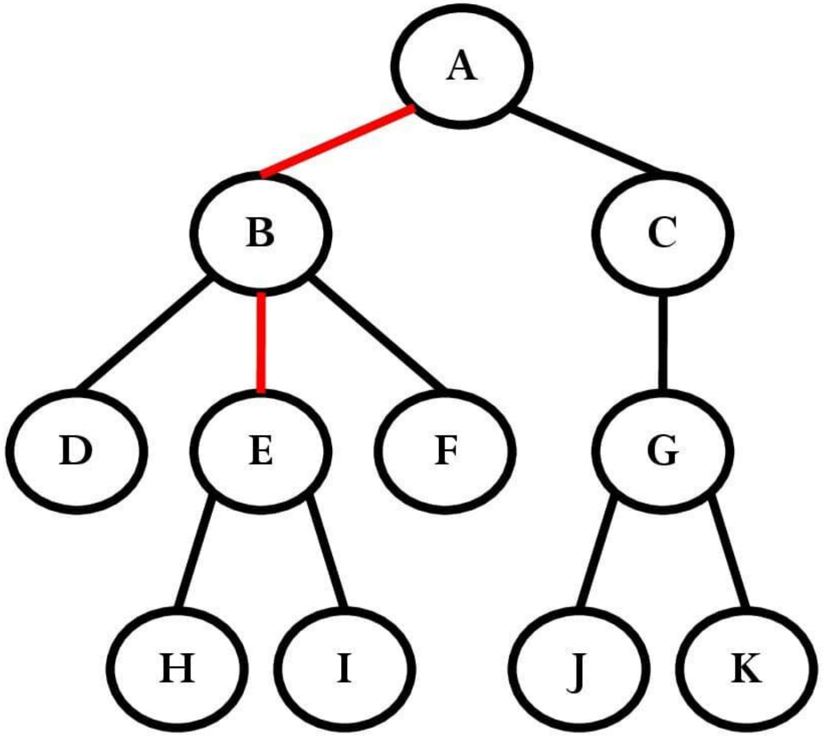
Tree ~ Terminologies

- **Level:** In a tree each step from top to bottom is called as a Level and the Level count starts with '0' and incremented by one at each level



Tree ~ Terminologies

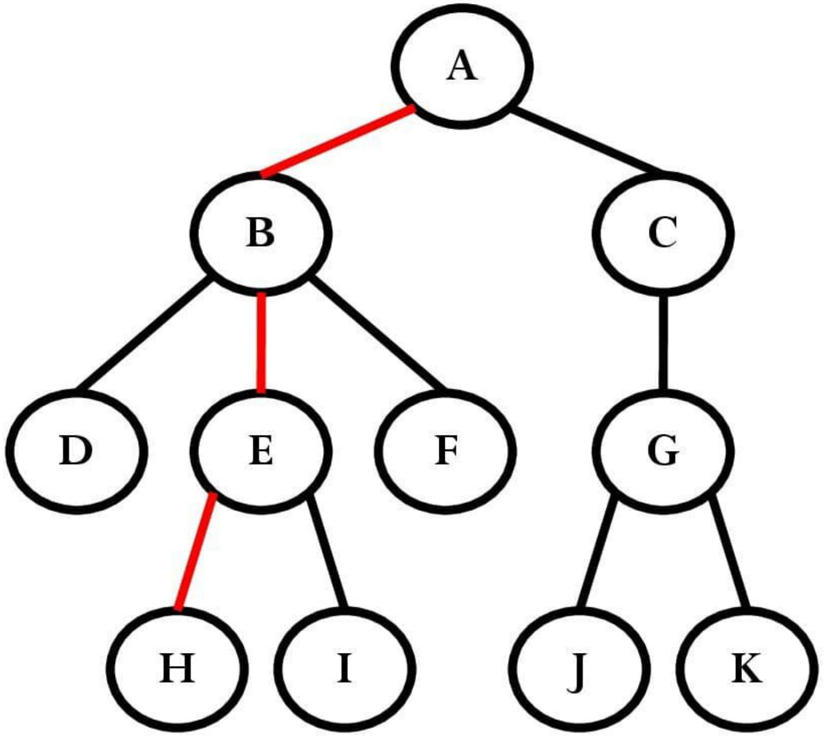
- **Depth of a node:** It is the total number of edges from root to that node.



- Depth of node E is 2

Tree ~ Terminologies

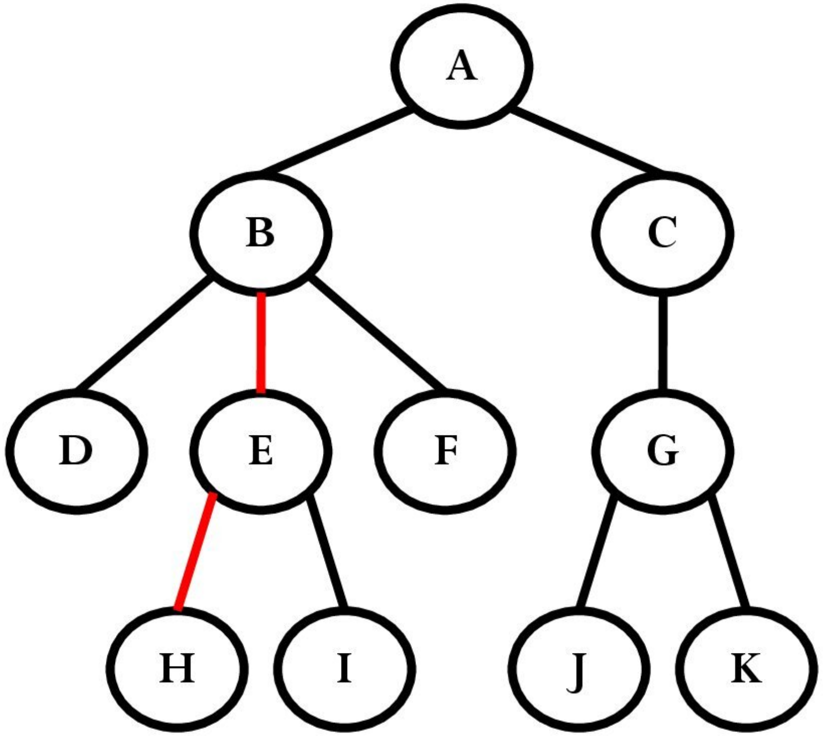
- **Depth of the tree:** It is the total number of edges from root to leaf in the longest path



- Depth of the tree = 3

Tree ~ Terminologies

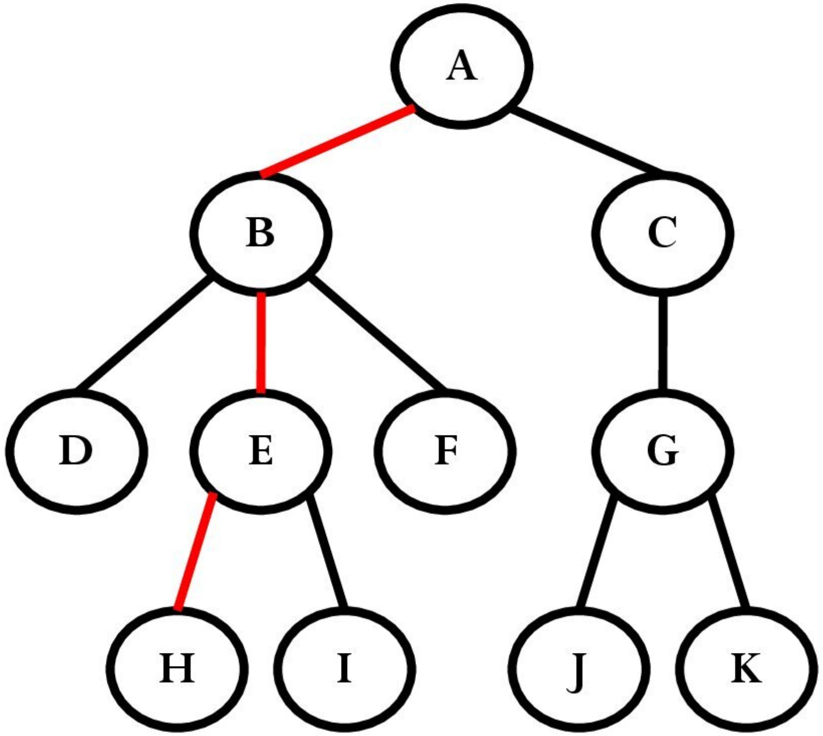
- **Height of a node:** It is the total number of edges from longest leaf node in its subtree to that node.



- Height of node B is 2

Tree ~ Terminologies

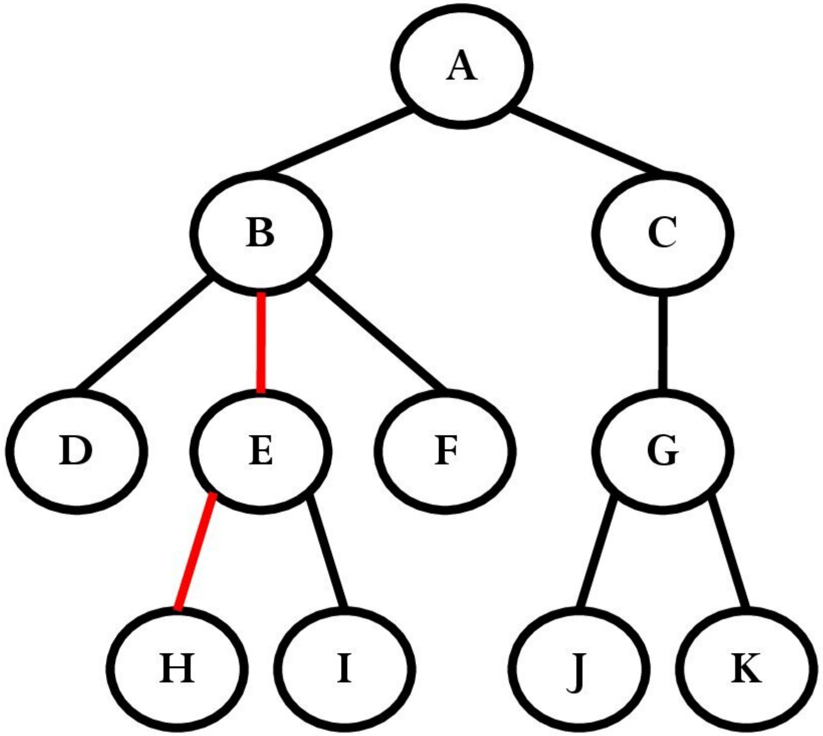
- **Height of the tree:** It is the total number of edges from longest path leaf node to root



- Height of the tree = 3

Tree - Terminologies

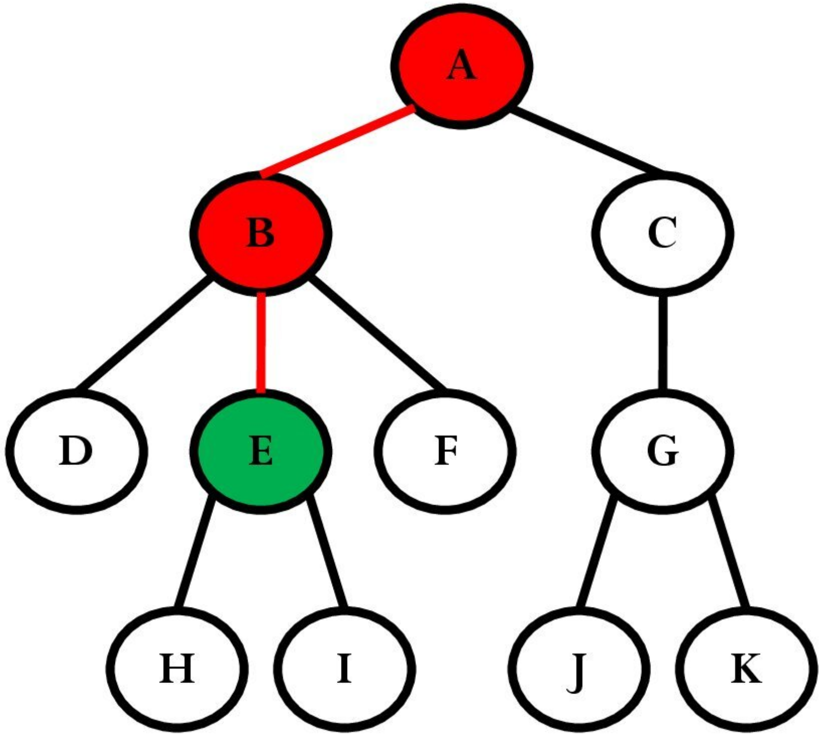
- **Path**: Sequence of nodes and edges between two nodes
- **Length of the path**: Total number of edges in the path



- Path between B and H is B-E-H. Its length is 2
- There is no path between B and G

Tree ~ Terminologies

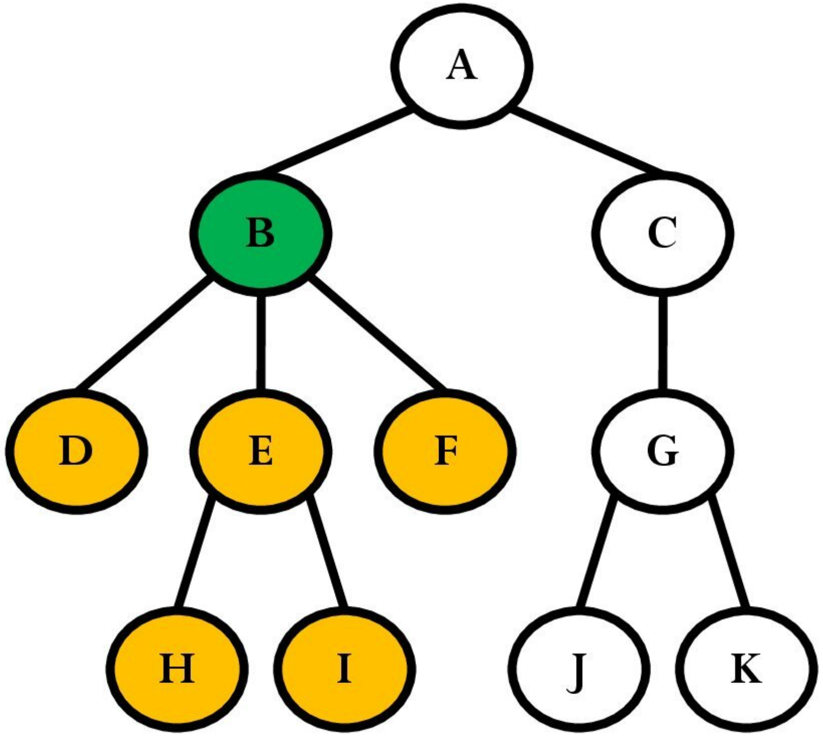
- The **ancestors** of a node are all the nodes along the path from the root to the node.
- The immediate ancestor of a node is the "parent" node



- Ancestors of E are B & A

Tree ~ Terminologies

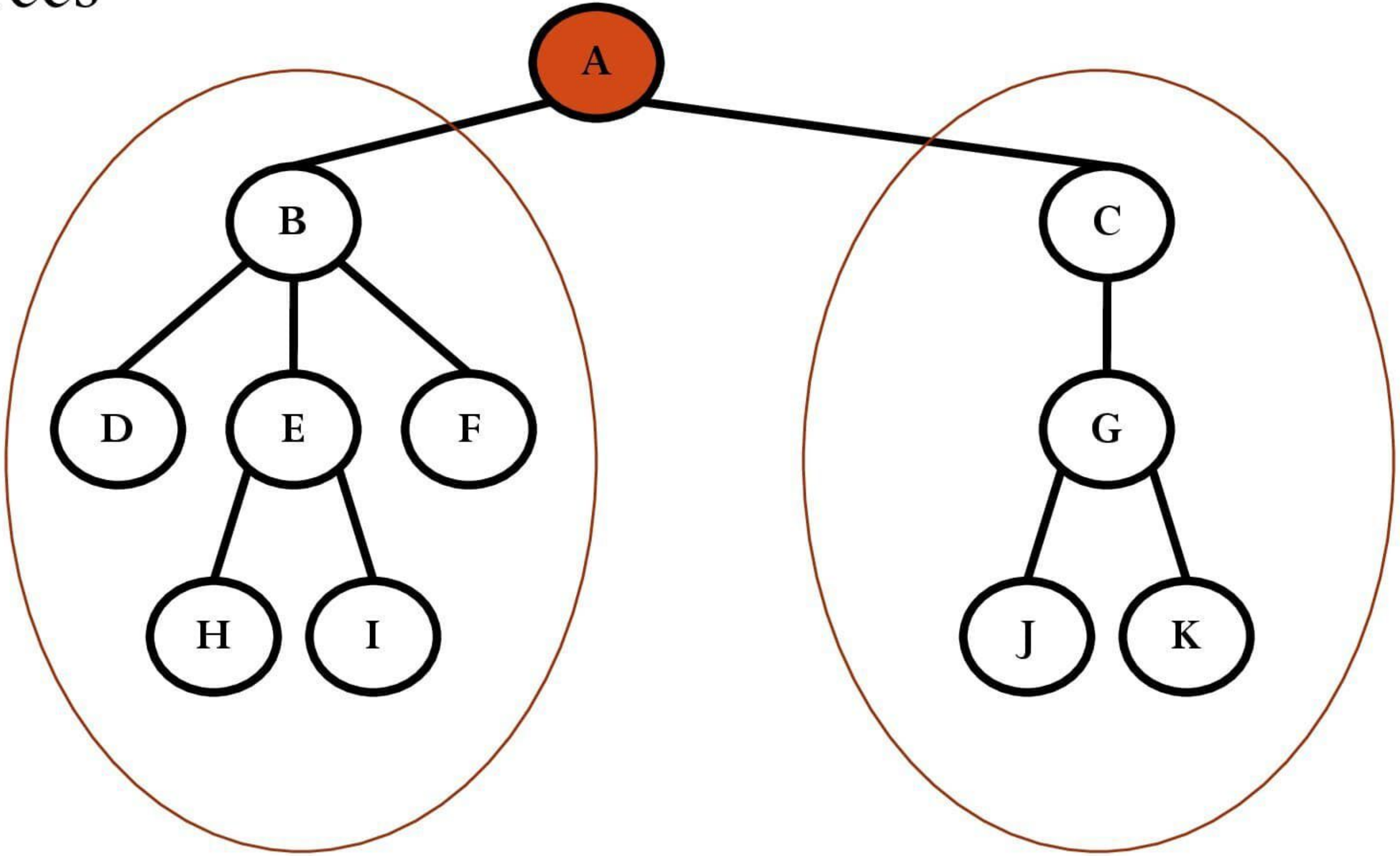
- A **descendant** node of a node is any node in the path from that node to the leaf node.
- The immediate descendant of a node is the “child” node.



- Descendant nodes of B are D, E, F, H & I

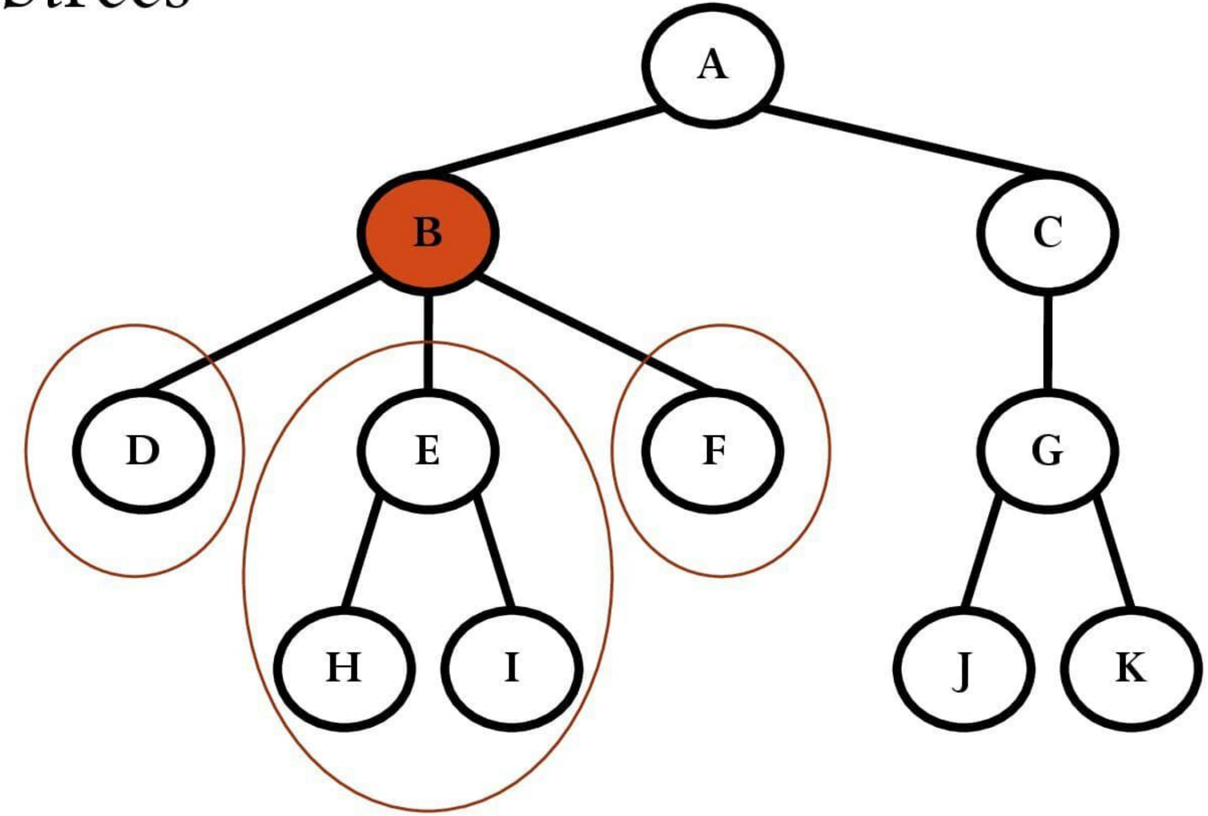
Tree ~ Terminologies

- Subtrees



Tree ~ Terminologies

- Subtrees



Tree Representations

1. List Representation
2. Left Child Right Sibling Representation
3. Representation as a degree 2 tree

List Representations

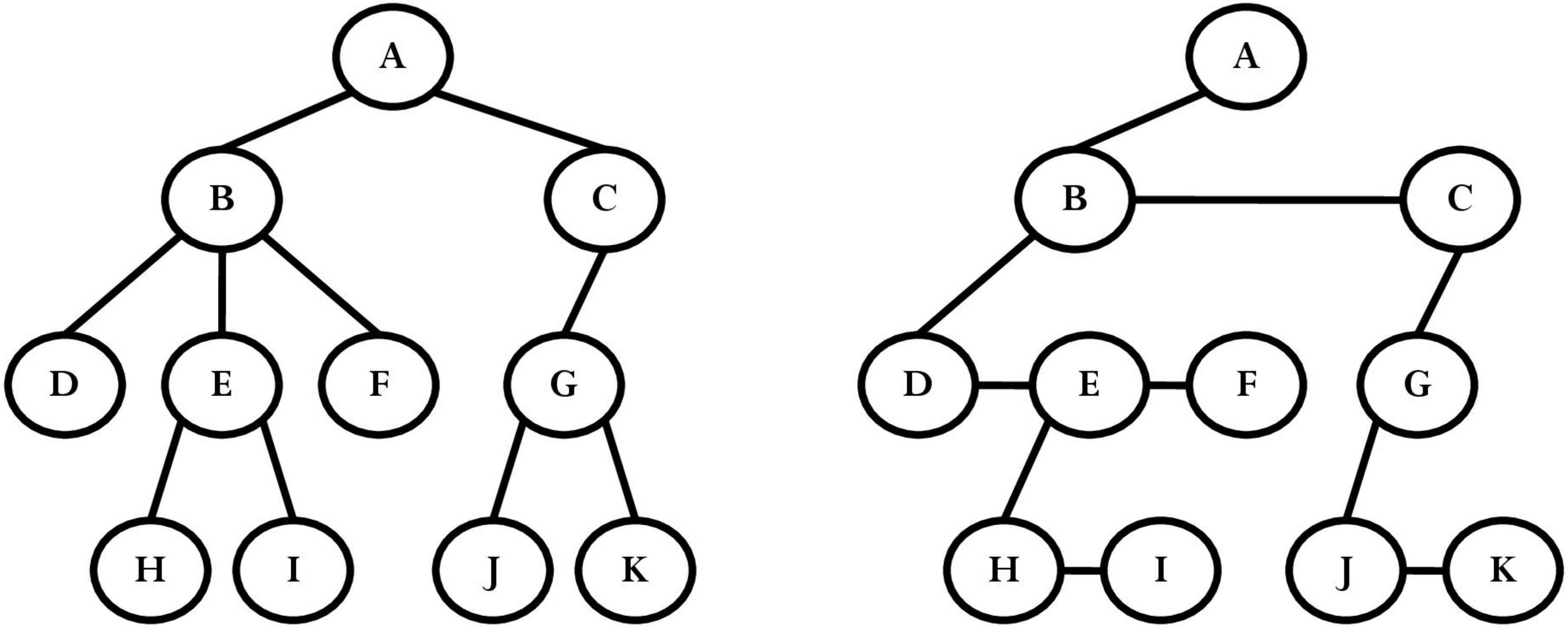
- Root comes first, followed by list of sub-trees

data	link 1	link 2	...	link n

Left child Right sibling Representations

- Fixed sized nodes
  - Easier to work
  - Two link/pointer fields per node

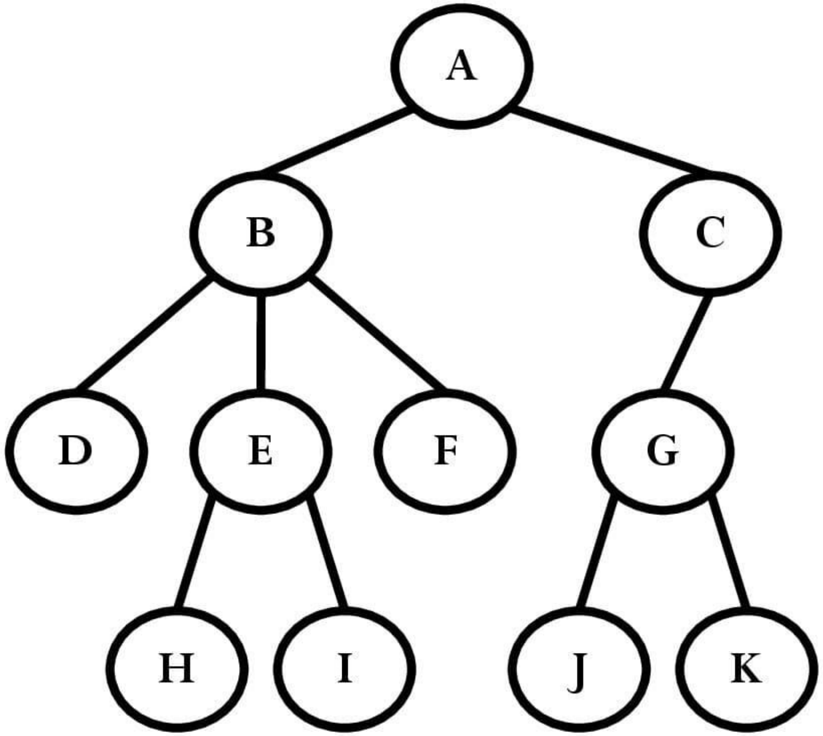
Left_child	data	Right_Sibling



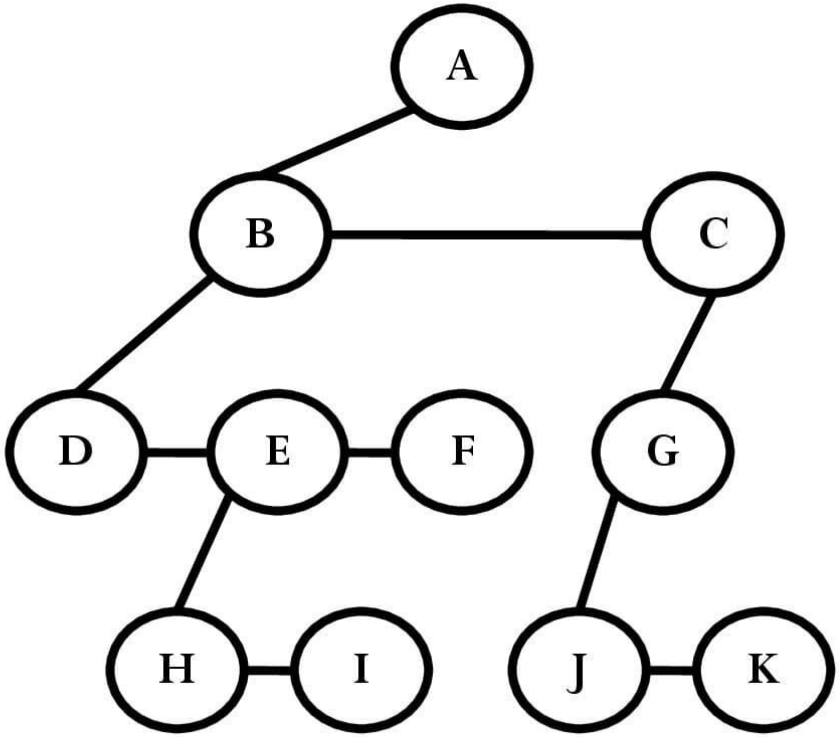
Representations as degree 2 tree

- Also known as Left child-Right child Tree/Degree - Two Tree/ Binary Tree
- Simply rotate the right sibling pointers in the Left-child Right-sibling tree clockwise by 45 degrees

Representations as degree 2 tree

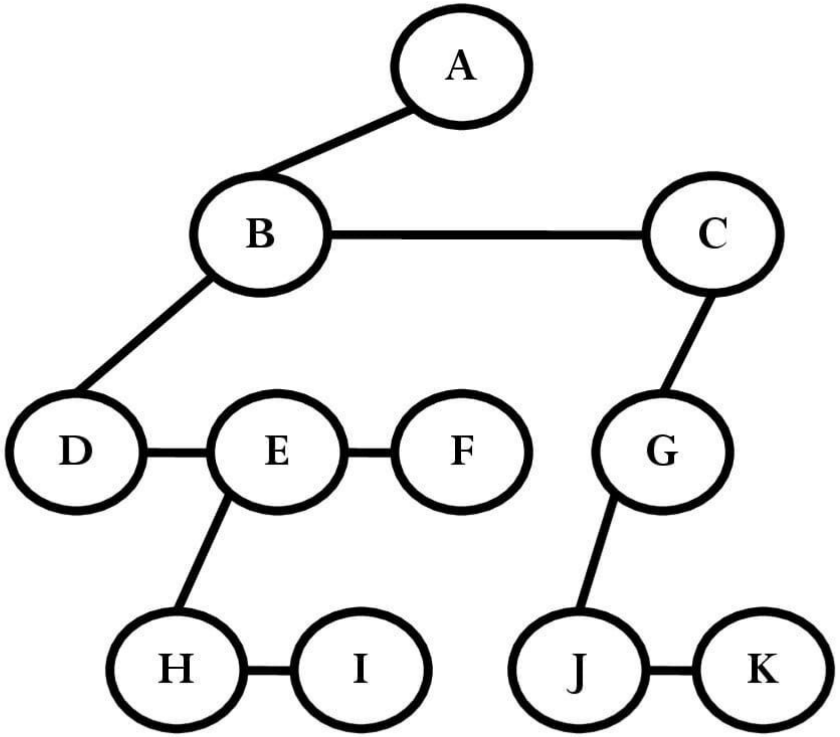


Tree

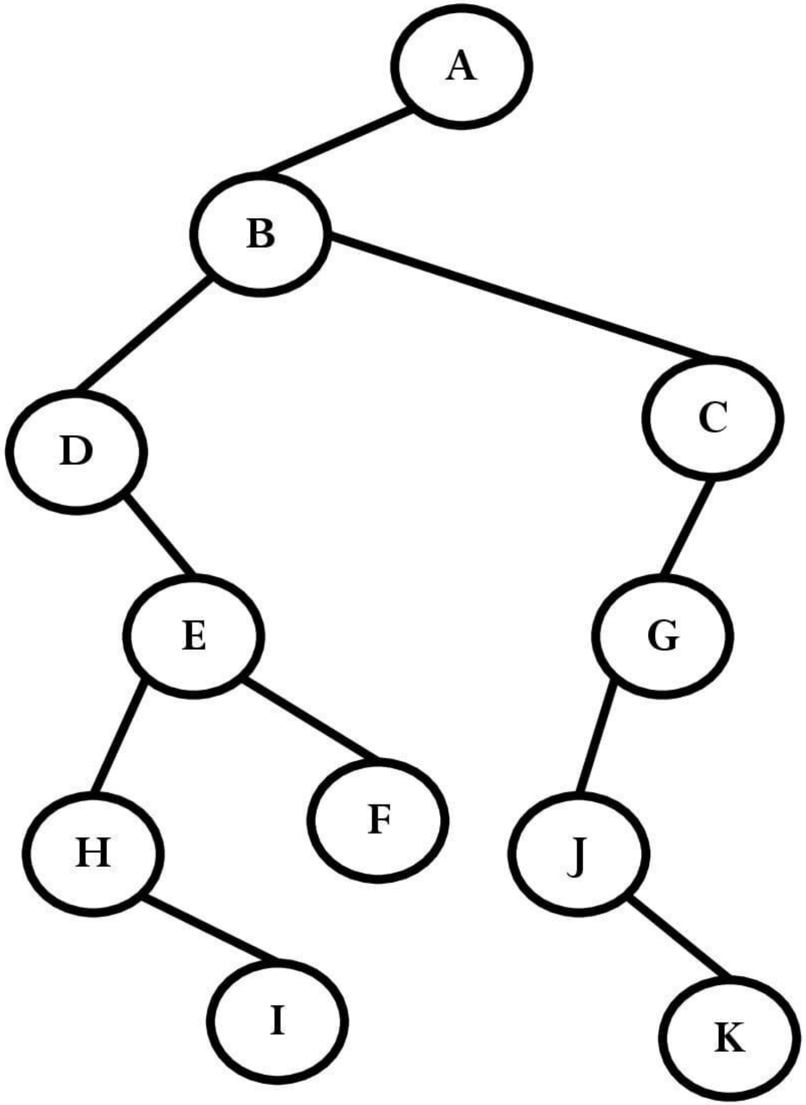


Left Child Right Sibling Representation

Representations as degree 2 tree



Left Child Right Sibling Representation



Degree 2 tree

Tree ~ Applications

- **Storing naturally hierarchical data**: Trees are used to store the data in the hierarchical structure. Example: Directory structure of a file system
- **Organize data**: It is used to organize data for efficient insertion, deletion and searching.
- Used in **compression algorithms**
- **Routing table**: The tree data structure is also used to store the data in routing tables in the routers.
- **To implement Heap data structure:** It is used to implement priority queues