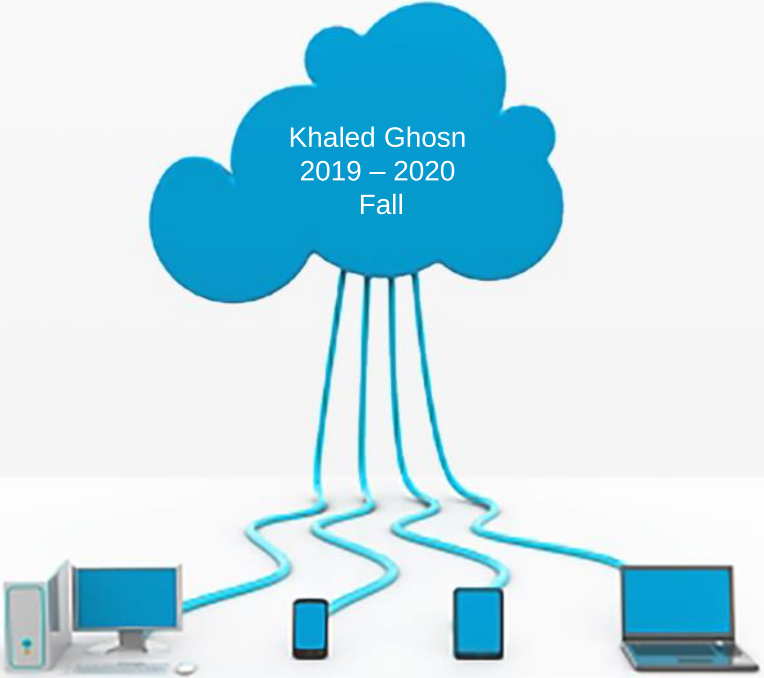
Trees

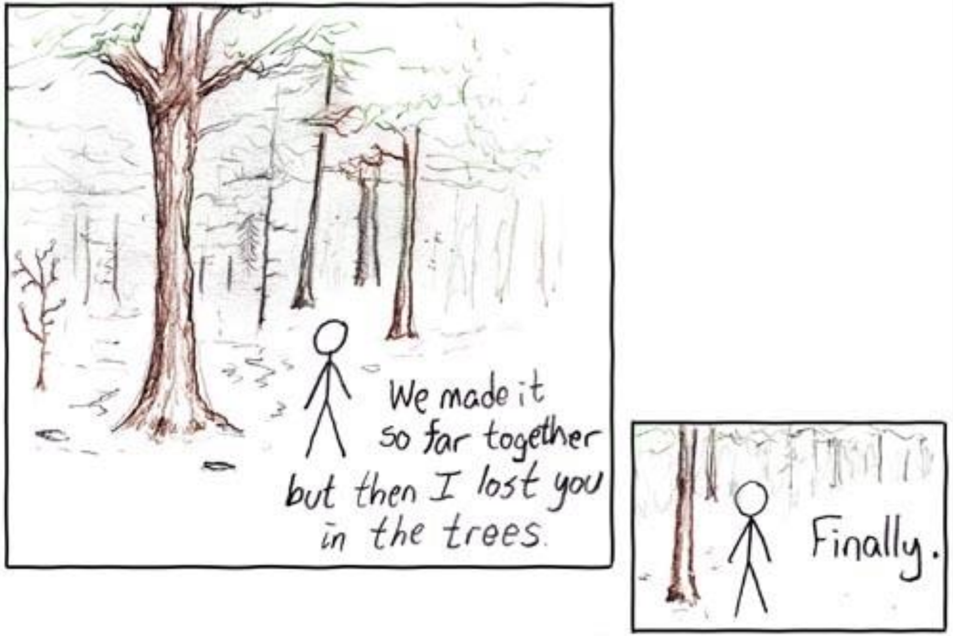
Definitions & Terminologies



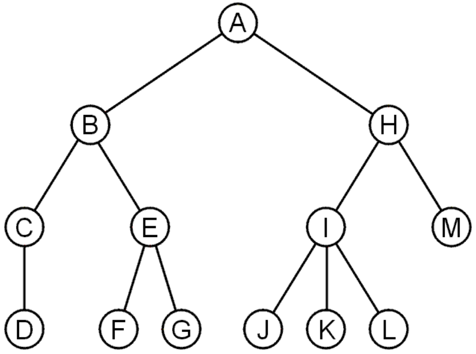
Khaled Ghosn
2019 – 2020
Fall

Tree

Trees are the first data structure different from what you've seen in previous courses

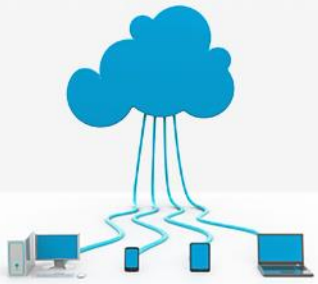


Tree

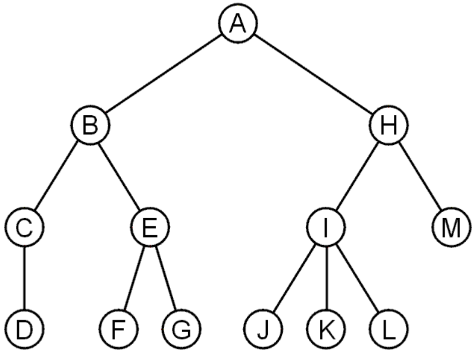


Definition

- ✓ Node – based data structure
    - A rooted tree data structure stores information in nodes
- ✓ Similar to linked lists:
    - There is a first node, or root
    - Each node has variable number of references to successors
    - Each node, other than the root, has exactly one node pointing to it
- ✓ Dynamic data structures
    - Size limited by the amount of free memory in O.S.
- ✓ Hierarchical (or non-linear) data structure
    - Unlike Array and Linked List, which are linear data structures



Tree



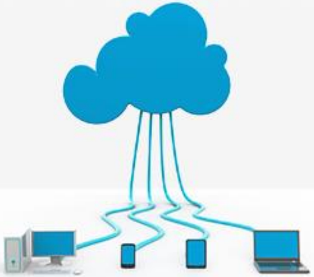
Definition

- ✓ are useful for hierarchically ordered data
  - unlike the Linear lists which are useful for serially ordered data
- ✓ is a data structure which allows associate a parent-child relationship between various pieces of data and thus allows us to arrange our records, data and files in a hierarchical fashion
- ✓ is a data structure that simulates a hierarchical tree structure, with a root value and sub-trees of children, represented as set of linked nodes
- ✓ can also be seen as collection of nodes, where each node is a data structure consisting of a value
- ✓ has a root, branches, and leaves

Tree

Definition

- ✓ A ***tree*** **T** is a set of ***nodes*** storing elements such that the nodes have a ***parent-child*** relationship that satisfies the following properties:
  - If **T** is nonempty, it has a special node, called the ***root*** of **T**, that has no parent.
  - Each node **v** of **T** different from the root has a unique ***parent*** node **w**; every node with parent w is a ***child*** of **w**.
- ✓ A tree can be empty, meaning that it does not have any nodes.



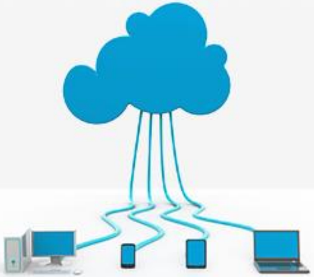
Tree

Definition

✓ Trees are used in many areas of computer science, such as:

- operating systems (e.g. file system)
- graphics
- database systems
- computer networking
- XHTML

…

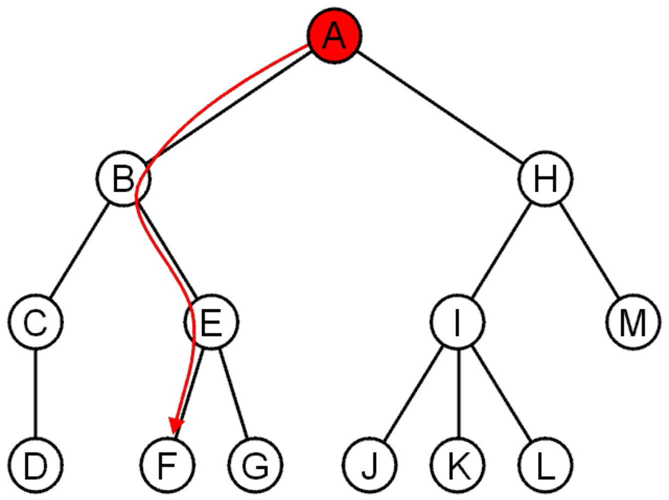


Tree

Terminologies

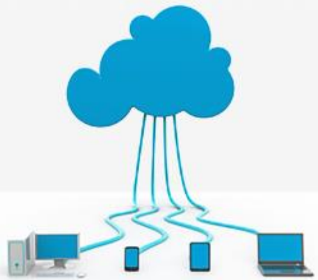
✓ ***Empty*** or ***Null Tree***

Tree having no nodes

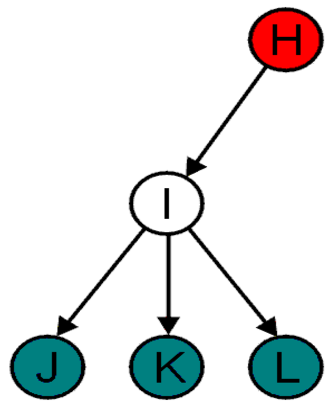


✓ ***Root***

- The top node in a tree
- The shape of a rooted tree gives a natural flow from the *root node*
  - A is the root



Tree



Terminologies

✓ ***Child Node***

All nodes will have zero or more child nodes or *children*

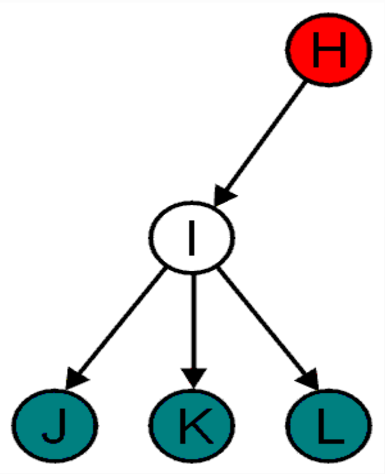
- Four children: I, J, K and L

✓ ***Parent Node***

For all nodes other than the root node, there is one parent node, the opposite notion of child

- H is the parent for I
- I is the parent for J, K, L

Tree



Terminologies

✓ ***Siblings***

Nodes with the same parent (brothers)

- J, K, and L are siblings

✓ ***Degree** of a Node*

The degree of a node is defined as the number of its children

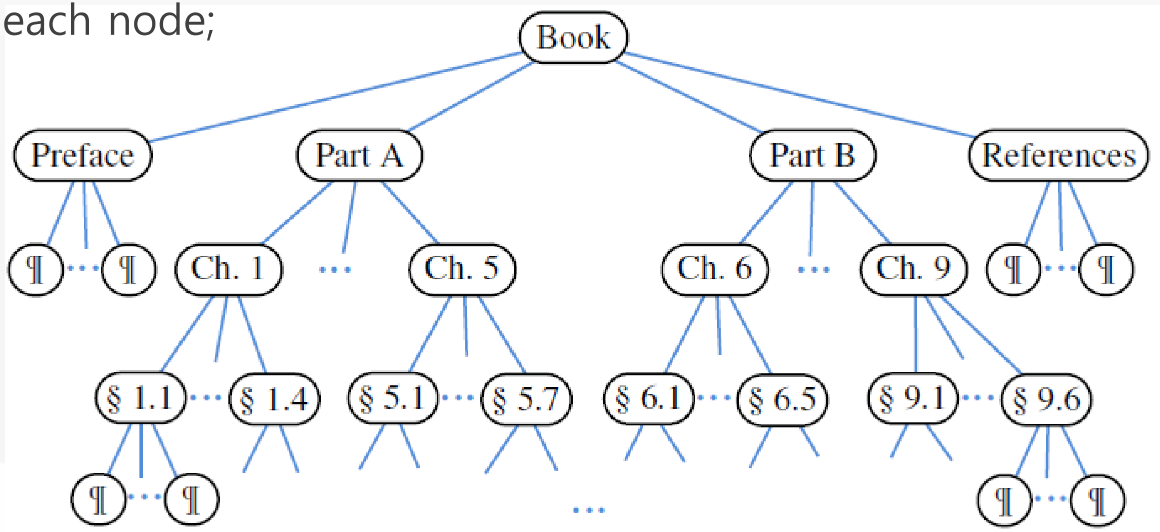
- number of sub trees of a node
- deg (I) = 3

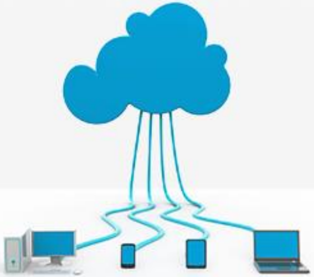
Tree

Terminologies

✓ *Ordered Trees*

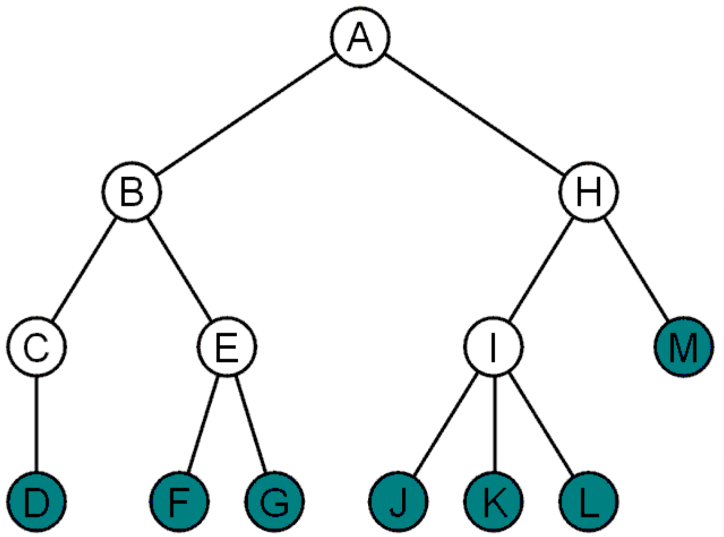
A tree is ***ordered*** if there is a meaningful linear order among the children of each node;





Tree

Terminologies



- ✓ ***Leaf Node***

  - Nodes with no child nodes

  - Nodes with degree zero

  - Also known as ***External Nodes***

    - D, F, G, J, K, L, M

- ✓ ***Internal Nodes***

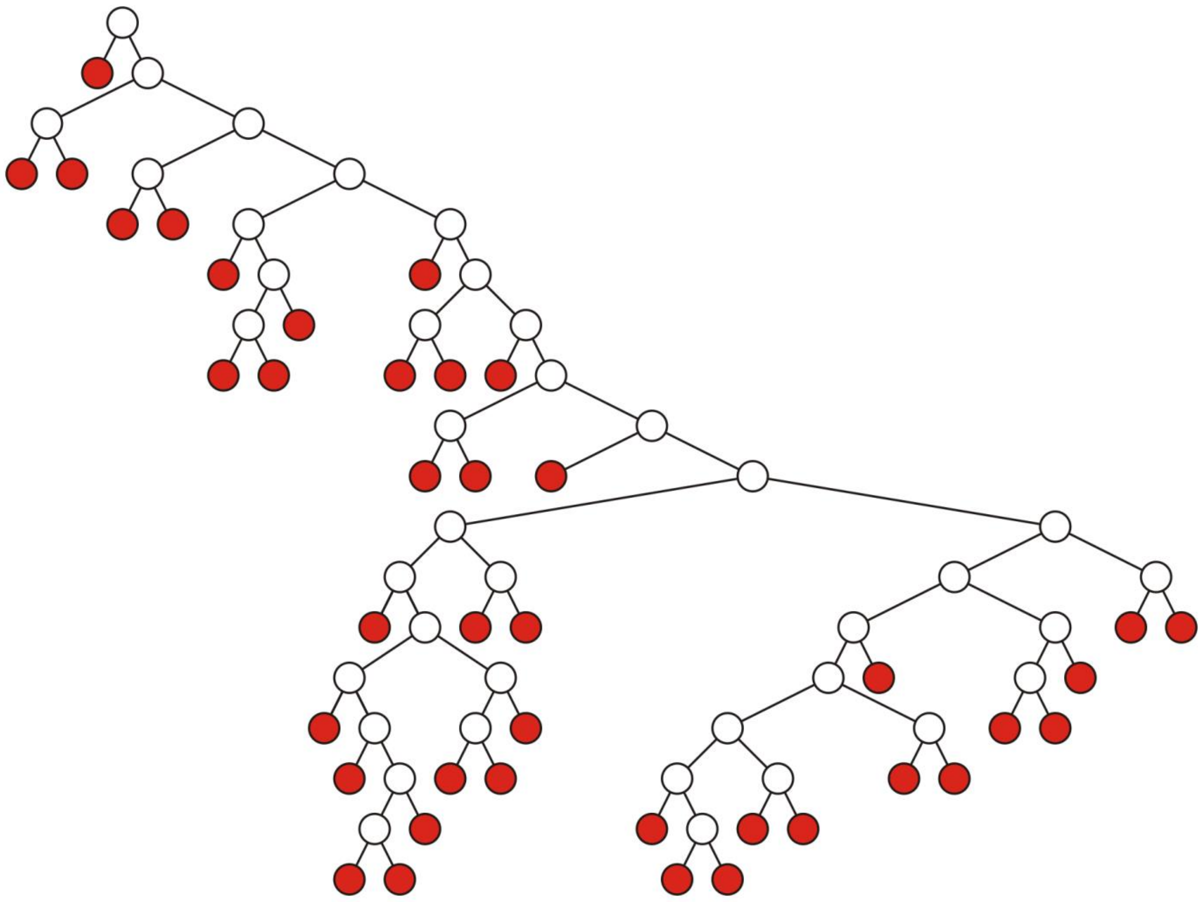
  - All other nodes (other than leaves) are said to be internal nodes, that is, they are internal to the tree

  - Nodes with at least one child

    - A, B, C, E, H, I

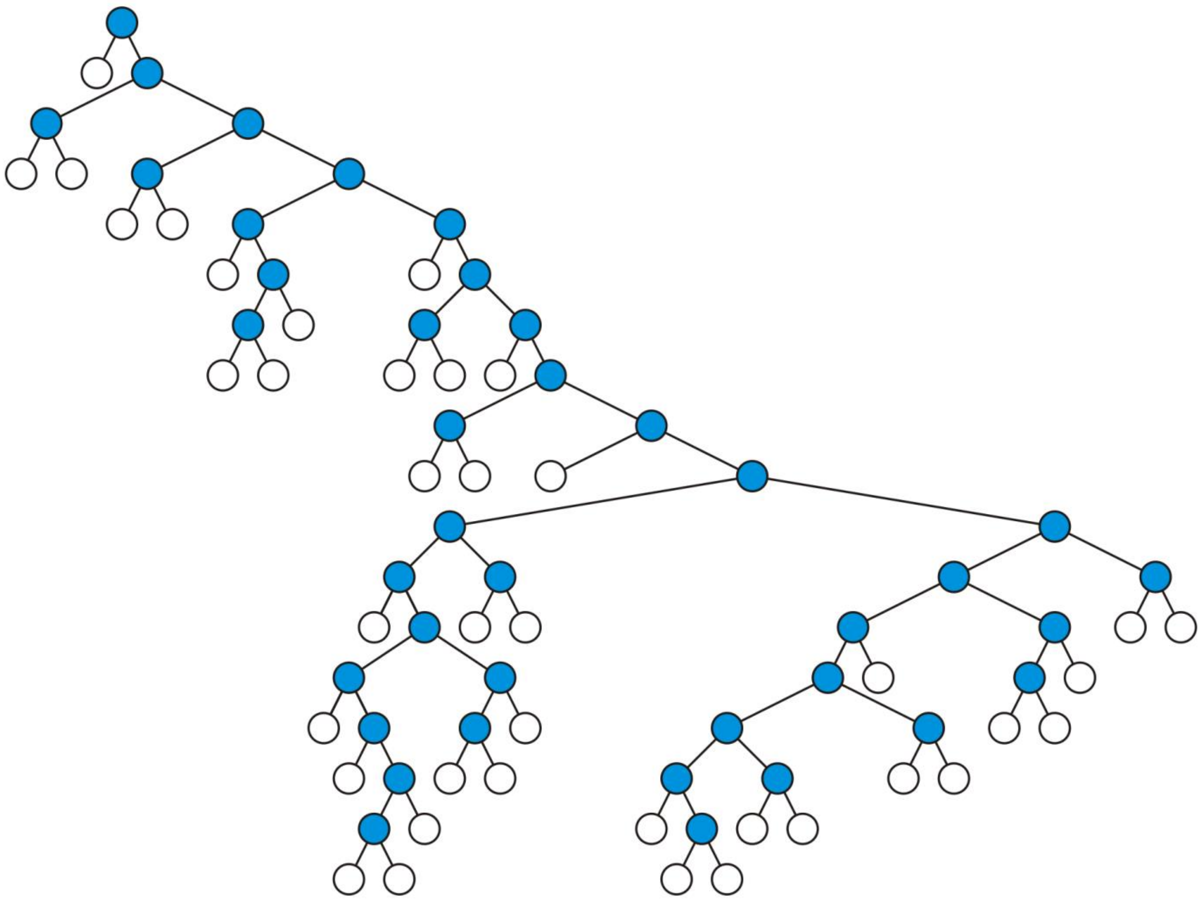
Tree

Leaf Nodes



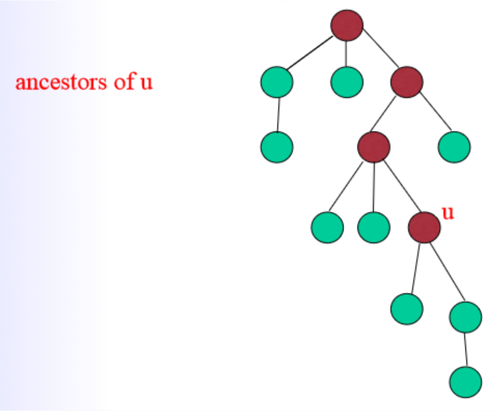
Tree

Internal Nodes



Tree

Terminologies



✓ ***Descendant***

- Child (inheritor)
- A node reachable by repeated proceeding from parent to child

✓ ***Ancestor***

- Grandparent
- A node reachable by repeated proceeding from child to parent

Tree

Terminologies

If a path exists from node *a* to node *b* :

- *a* is an *ancestor* of *b*
- *b* is a *descendent* of *a*

Thus, a node is both an ancestor and a descendant of itself

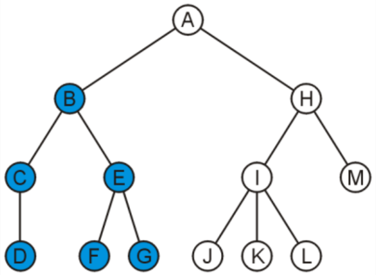
- We can add the adjective *strict* to exclude equality: *a* is a *strict descendent* of *b* if *a* is a descendant of *b* but $a \neq b$

The **Root** node is an ancestor of all nodes

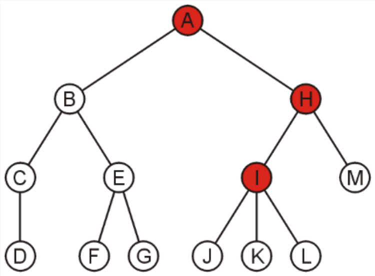
Tree

Terminologies

The descendants of node B are B, C, D, E, F, and G:



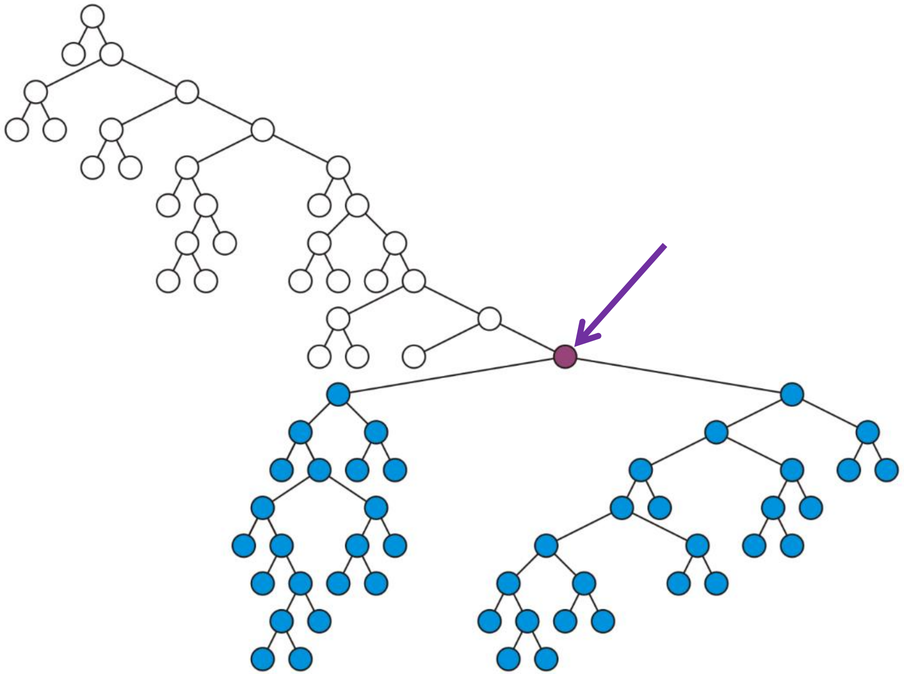
The ancestors of node I are I, H, and A:



Tree

Terminologies

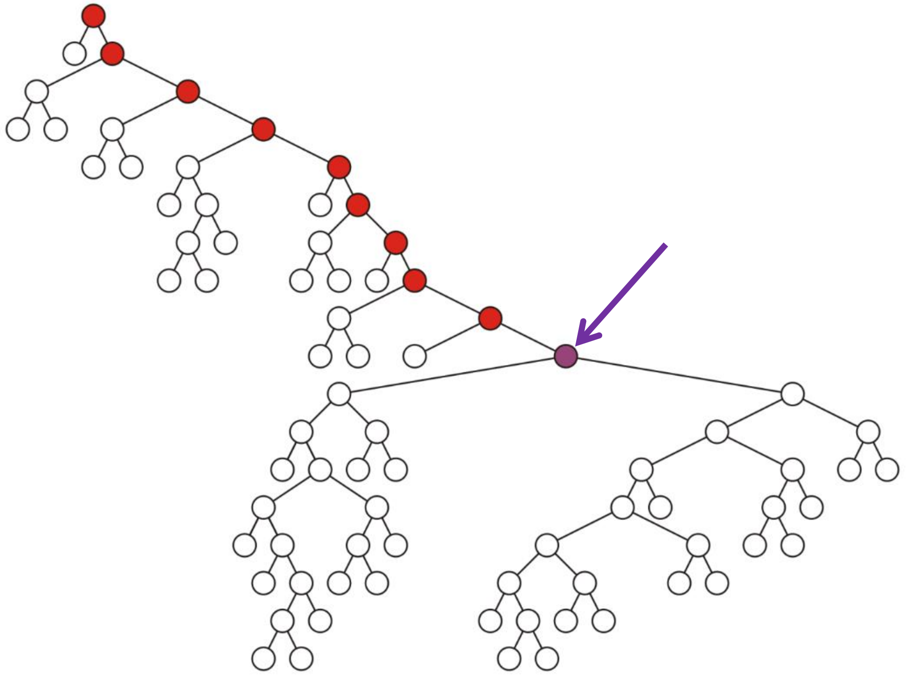
All descendants (including itself) of the indicated node



Tree

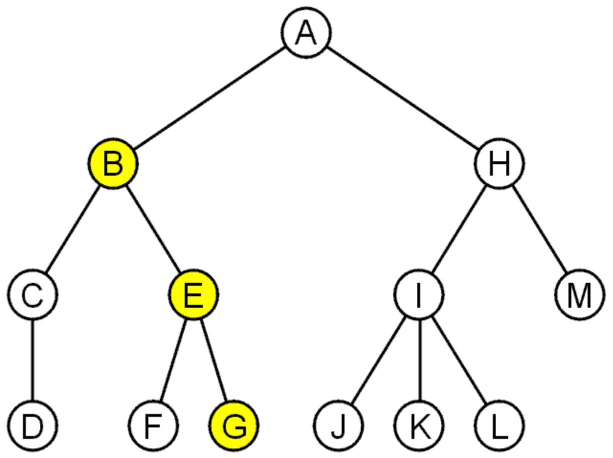
Terminologies

All ancestors (including itself) of the indicated node



Tree

Terminologies



✓ ***Edge***

Connection between one node to another

✓ ***Path***

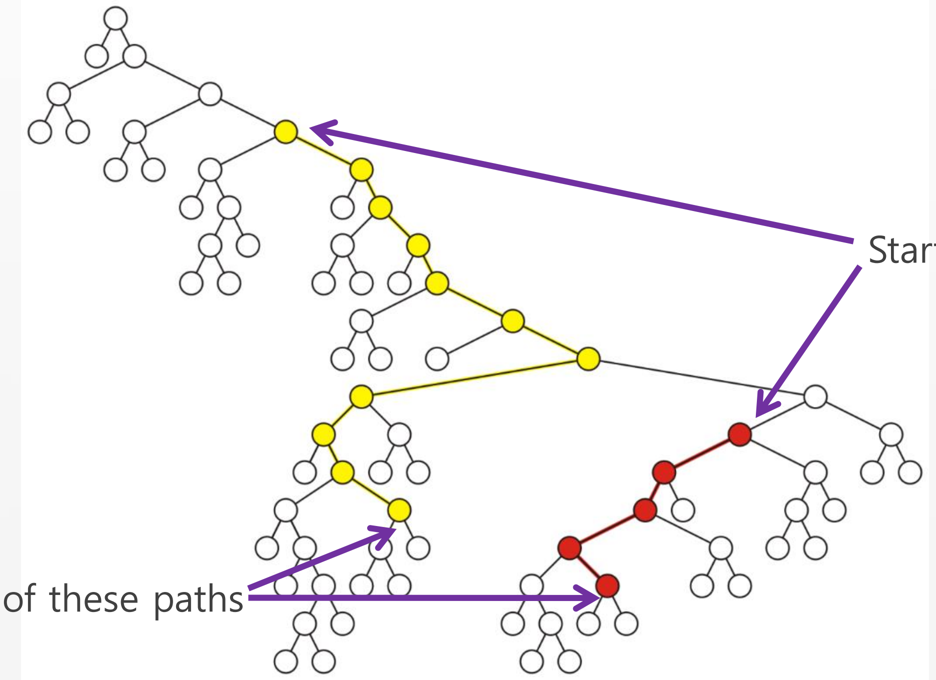
- A path is a sequence of nodes (and edges) connecting a node with a descendant

$(a_0 , a_1 , \ldots a_n)$ where $a_k + 1$ is a child of a_k

- For each node in a tree, there exists a unique path from the root node to that node

Tree

Paths of length 10 (11 nodes) and 4 (5 nodes)

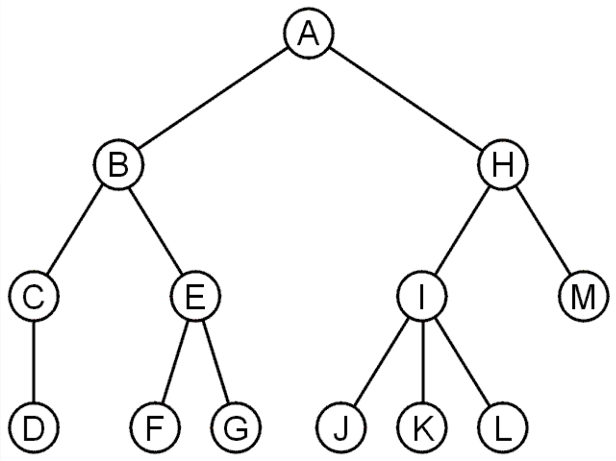


Start of these paths

End of these paths

Tree

Terminologies



✓ ***Length***

Number of edges on the path

- the path (B, E, G) has length 2

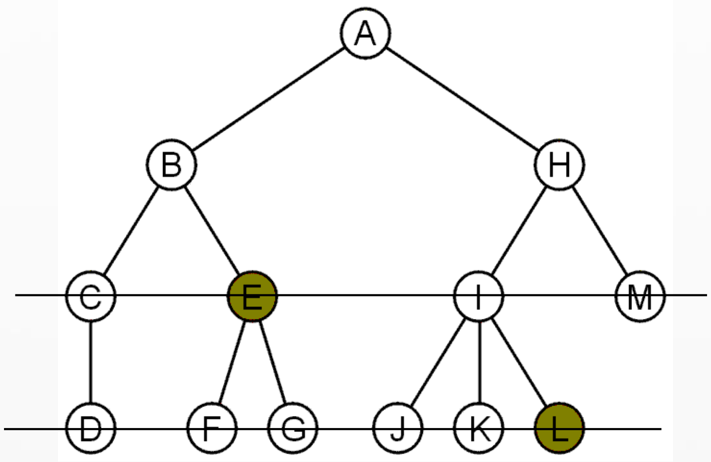
✓ ***Level***

The level of a node is defined by 1 + the number of edges (connections) between the node and the **Root**

- Level of G is: 1 + 3 (connections) = 4

Tree

Terminologies

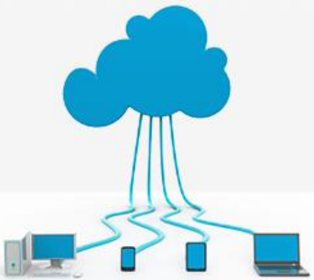


✓ *Depth*

- Length of the unique path from the **Root** to that node

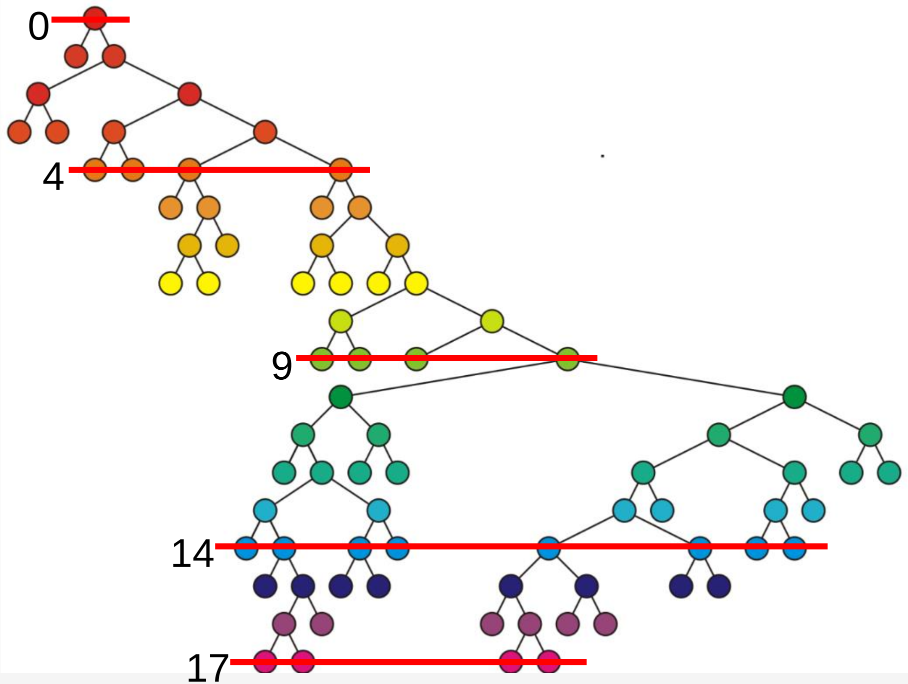
- Number of edges from the node to the tree's **Root** node

- E has depth 2
- L has depth 3



Tree

Nodes of depth up to 17



Tree

Terminologies

✓ ***Height of Tree***

- The maximum depth of any node within the tree

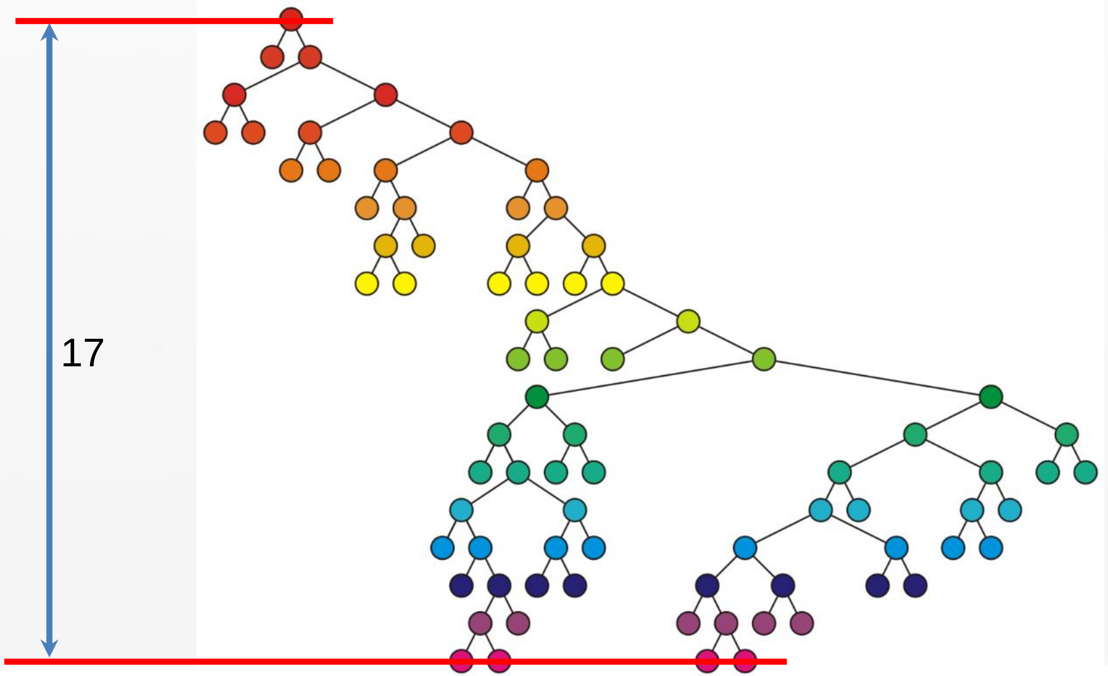
- Number of edges on the longest downward path between the **Root** and a leaf

- The height of a tree with one node is 0
    - Just the root node
- For convenience, we define the height of the empty tree to be –1

Tree

Terminologies

The height of this tree is 17



Tree

Terminologies

- ✓ ***Height of Node***
  - Length of the longest path from that node to a leaf
  - Number of edges on the longest downward path between that node and a leaf

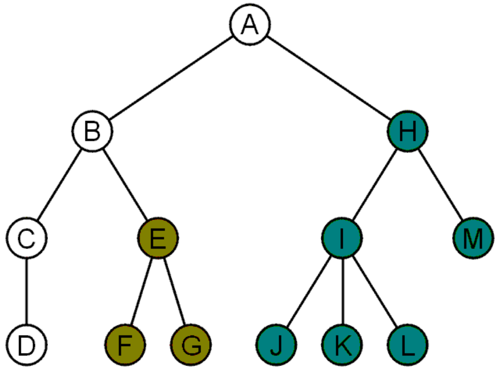
Tree

Terminologies

Another approach to a tree is to define the tree recursively:

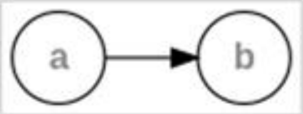
- A degree-0 node is a tree
- A node with degree *n* is a tree if it has *n* children and all of its children are disjoint trees (*i.e.*, with no intersecting nodes)

Given any node (a) within a tree with root (r), the collection of (a) and all of its descendants is said to be a *sub-tree of the tree with root* (a)



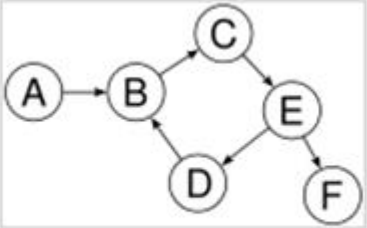
Tree

Not a Tree

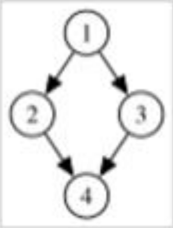


Each linear list is trivially a tree

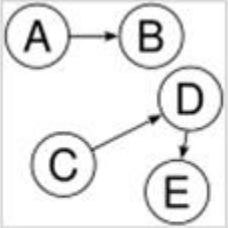
Not a tree: cycle A→A



Not a tree: cycle B→C→E→D→B



Not a tree: undirected cycle 1-2-4-3

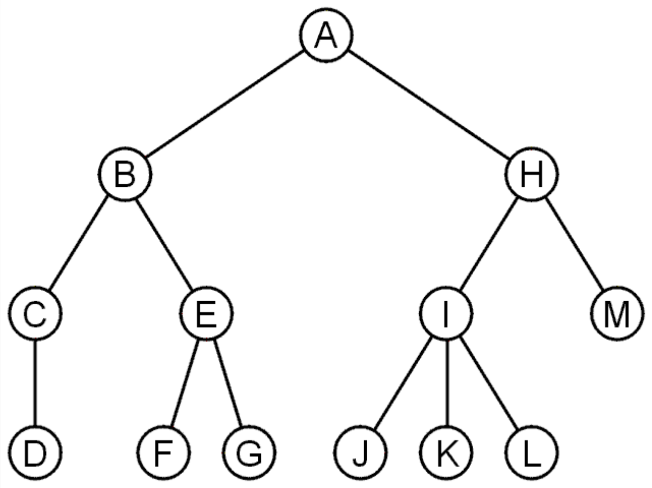


Not a tree: two non-connected parts, A→B and C→D→E

Abstract Tree

- ✓ An abstract tree does not restrict the number of nodes
  - In this tree, the degrees vary:

Degree	Nodes
0	D, F, G, J, K, L, M
1	C
2	B, E, H
3	I



- ✓ We can implement a general tree by using a class which:
  - Stores an element
  - Stores the children in a linked-list

Abstract Tree

The tree with six nodes would be stored as follows: