Trees

COMP 2210 – Dr. Hendrix

AUBURN UNIVERSITY

SAMUEL GINN
COLLEGE OF ENGINEERING

Trees

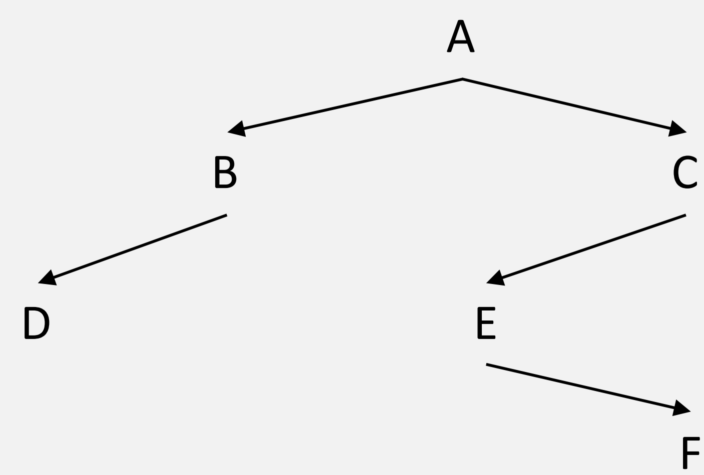
A tree is a collection in which the elements are arranged in a hierarchy.

A **list** is a *one dimensional* structure because it defines *linear relationships* between elements: **predecessor, successor**

A → B → C → D → E → F

successor(B) == C   predecessor(C) == B

A **tree** is a *two dimensional* structure because it defines *hierarchical relationships* among elements: **parent, child**

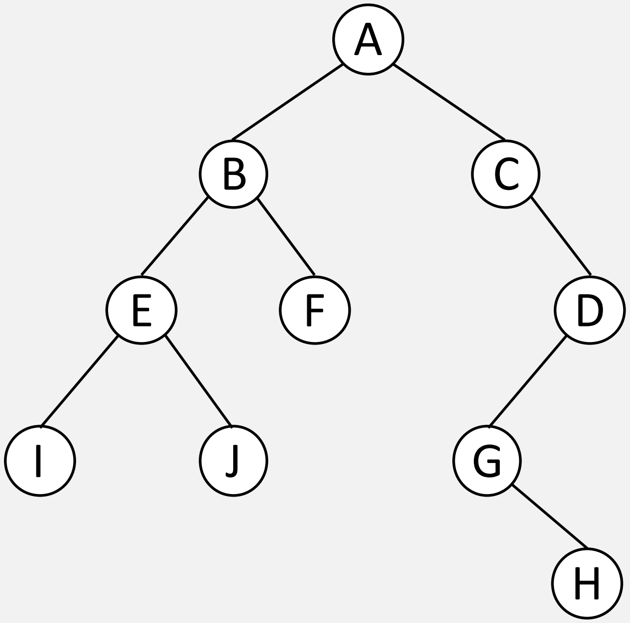


parent(B) == A

parent(C) == A

child(A) == B, C

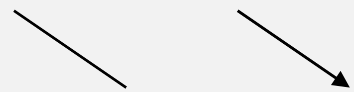
Tree terminology



A tree is composed of nodes and branches.

Node – places in the tree where the elements are stored

Branches – connections between nodes, from parent to child. Also called **edges**.

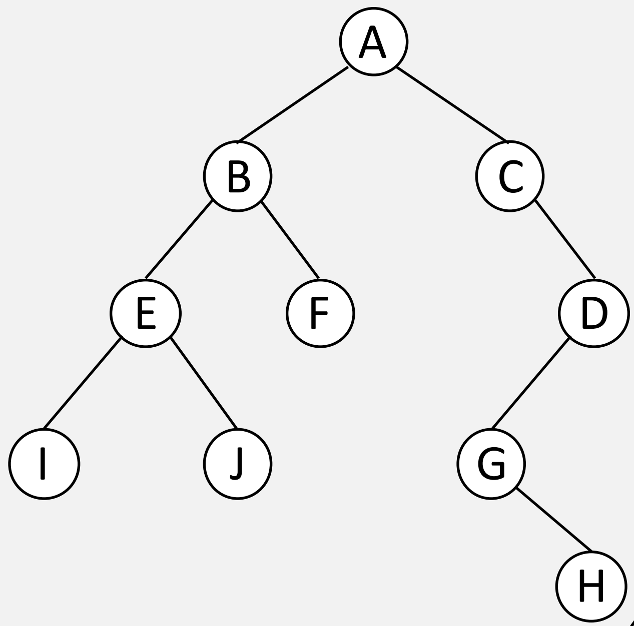


!

The terms "nodes" and "branches" are abstract and do not imply a particular implementation.

That is, we could implement a tree with either arrays or (physical) nodes and pointers.

Tree terminology



A **parent node** has one or more **children**.

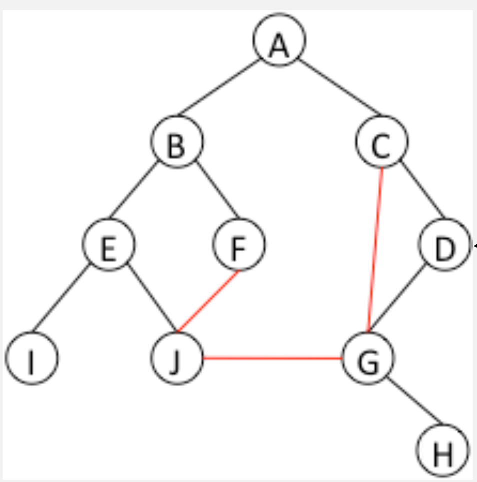
A, B, C, E, D, and G are parents.

A **leaf node** has no children.

I, J, F, and H are leaves.

A **child node** has exactly one parent.

B, C, E, F, D, I, J, G, H are children.



*Makes structures like this **not** a tree.*

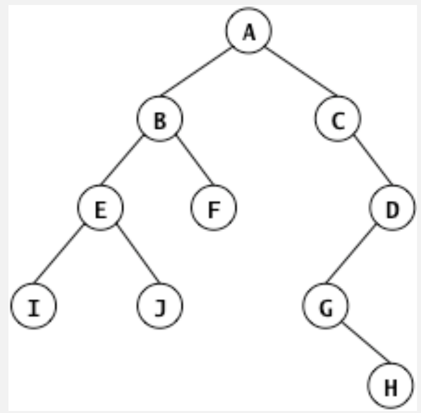
The **root node** has no parent

A is the root.

Tree terminology

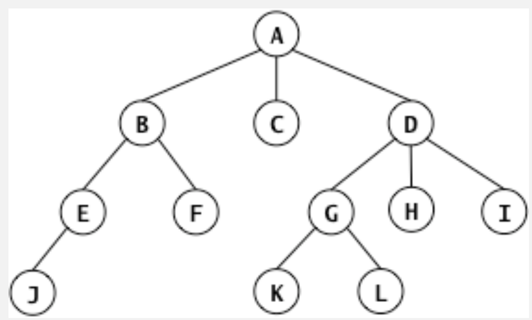
The **order** of a tree is an integer ≥ 2 that represents the upper limit on the number of children that any node can have.

Order = 2 Binary Tree



Each node can have at most 2 children.

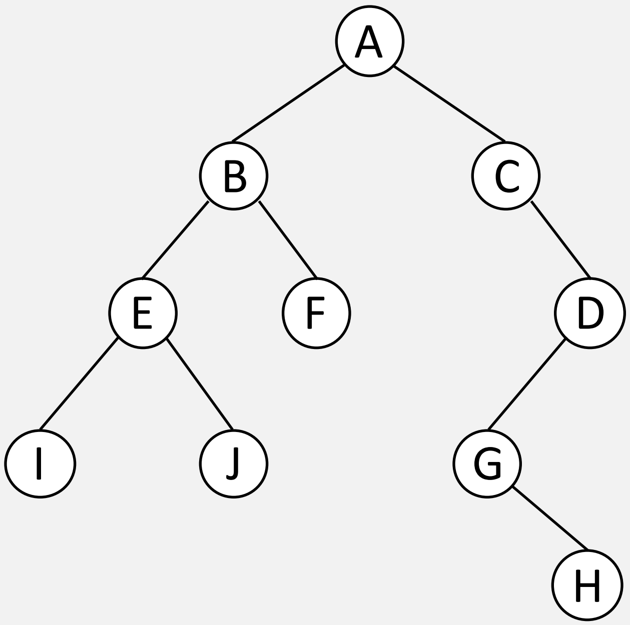
Order = 3 Ternary Tree



Each node can have at most 3 children.

General tree = a tree with no specified order.

Tree terminology



Path – a sequence of nodes from one node to another node, going from parent to child

Path from A to J = A-B-E-J

There is no path from J to A.

Path length – the number of nodes on the path

Path from A to J has length 4

Ancestor – Node X is an ancestor of node Y iff there is a path from X to Y

Descendent– Node X is an descendent of node Y iff there is a path from Y to X.

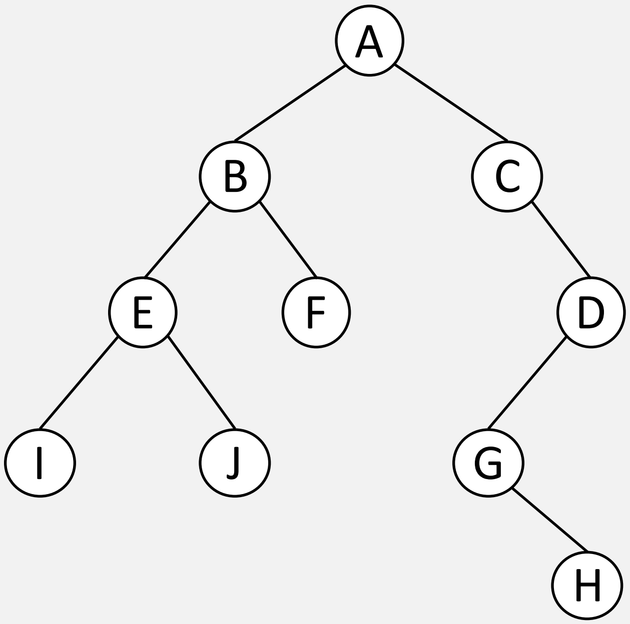
!

A path is sometimes defined as a sequence of edges instead of nodes.

!

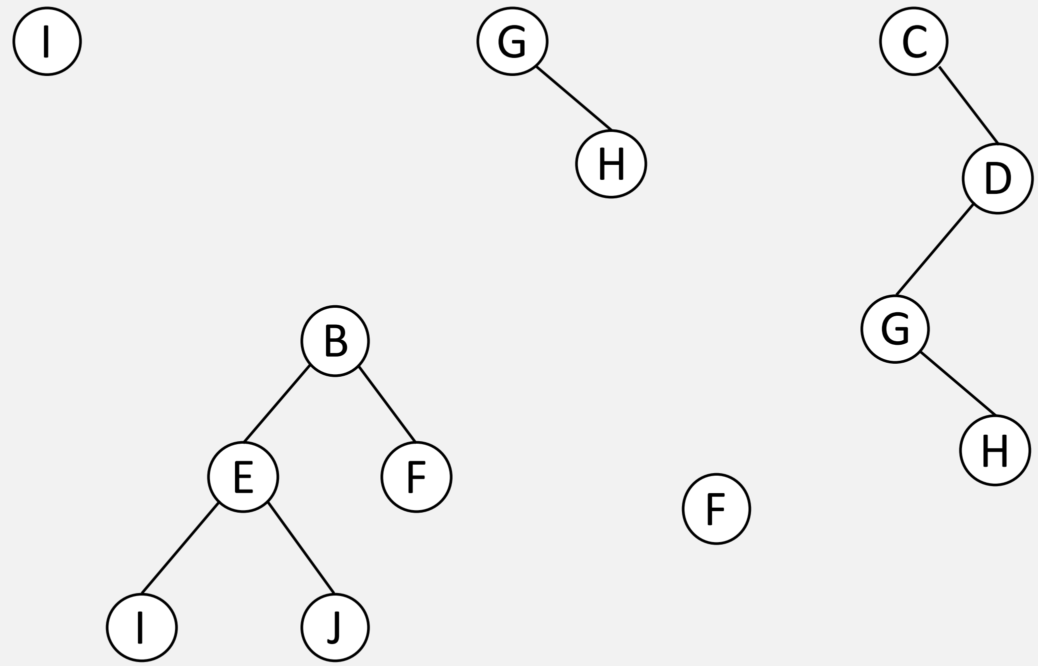
So, path length is sometimes counted differently.

Tree terminology



Subtree – A tree within a larger tree, rooted at a given node X. The subtree consists of X and all descendents of X.

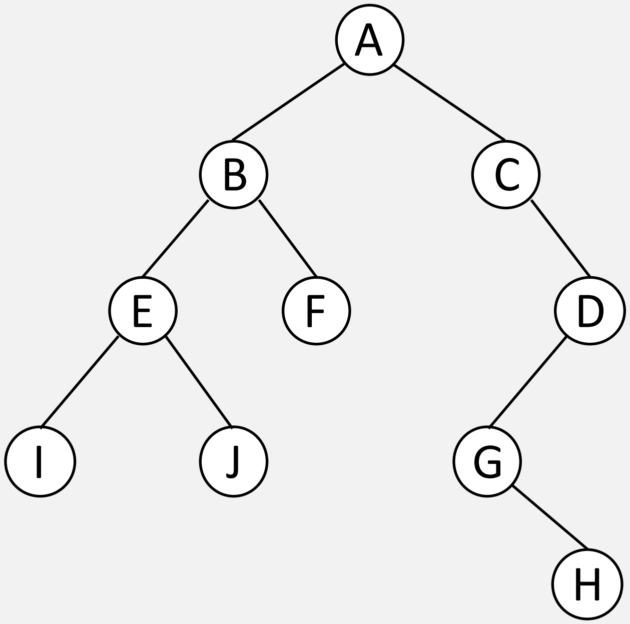
Example subtrees:



There are as many subtrees are there are nodes in the tree.

The tree itself is a subtree.

Tree terminology – height



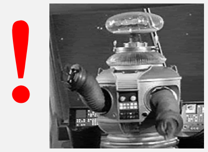
Height is a metric that is defined in terms of a given node, but is typically used to describe a tree or subtree.

When height is applied a tree or subtree, it refers to the height of its root.

Height measures the distance of a given node from the "bottom" of the tree.

Height = length of the longest path from a given node to a descendent leaf

Height depends on how path and path length are defined.

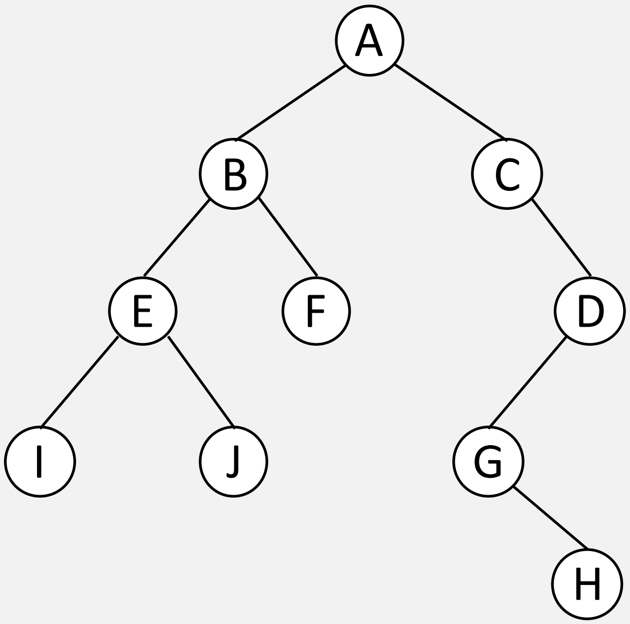


You will be off by one from the text.

Height of A = 5 ← *height of the tree*

Height of B = 3 Height of J = 1 Height of H = 1

Tree terminology – depth



Depth measures the distance of a given node from the “top” of the tree.

Depth is the same concept as “level” in the text.

Depth = length of the path from the root of the tree to a given node.

Depth of A = 1

Depth of B = 2   Depth of J = 4   Depth of H = 5

Depth of a leaf on the lowest level is the same as the height of the tree.

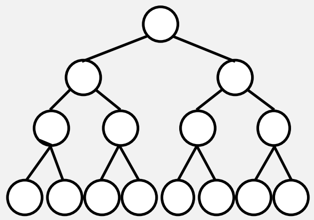
!


Depth depends on how path and path length are defined.

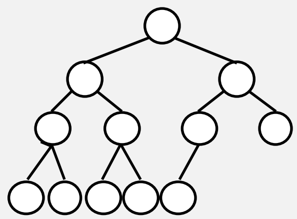
!


You will be off by one from the text.

Tree terminology

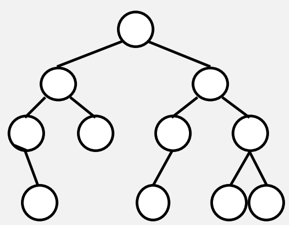


Full – A tree is full if all leaves have the same depth and every parent node has the maximum number of children.



Complete – A tree is complete if it is full to the next-to-last level, and the leaves on the lowest level are "left justified".

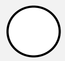
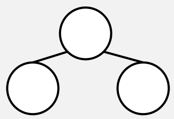
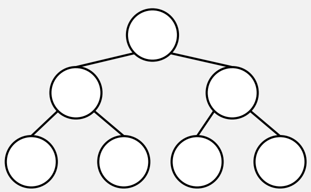
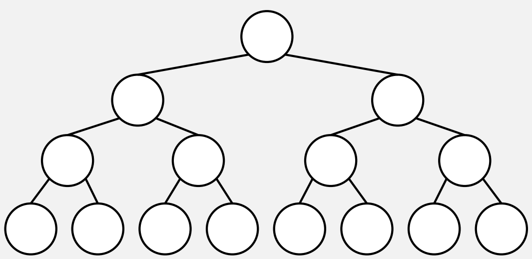
A full or complete tree is the shortest possible tree (minimum height) that could store N nodes.



Balanced – A tree is balanced if for each node, its subtrees have similar heights. The term "similar" is intentionally vague since different balancing schemes exist.

A balanced tree will have near-optimal height for storing N nodes.

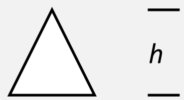
Capacity v. Height

Full Binary Tree	#Nodes (n)	Tree Height (h)
	1	1
	3	2
	7	3
	15	4

$$h = \lfloor \log_2 n \rfloor + 1 \qquad n = 2^h - 1$$

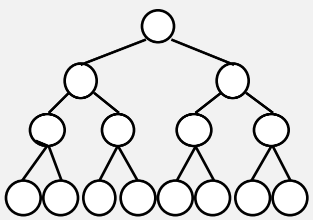
Shapes and height

height

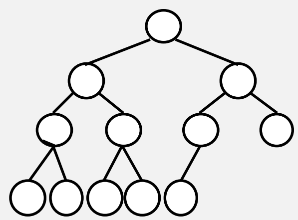


Many tree algorithms are dependent to some extent on the tree's height.

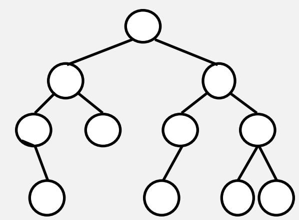
full



complete



balanced



$h = \log_2(n+1)$

Height is O(logn)