Trees

Data Structures and Algorithm

Trees

Siblings

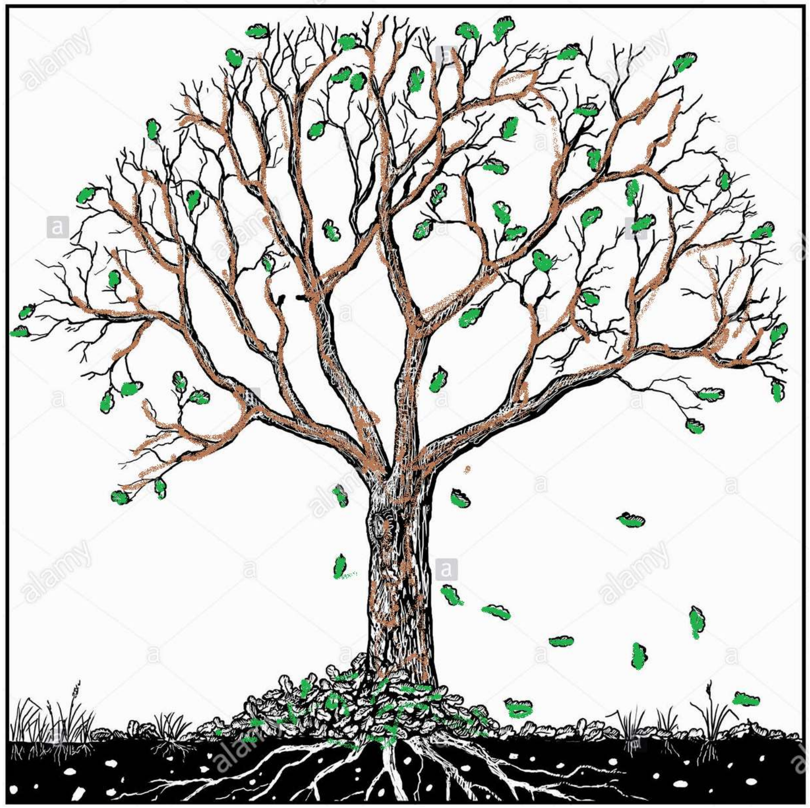
Degree of node

Successor node

Children

Parent node

Root



Depth

Internal node

Leaf node

Height

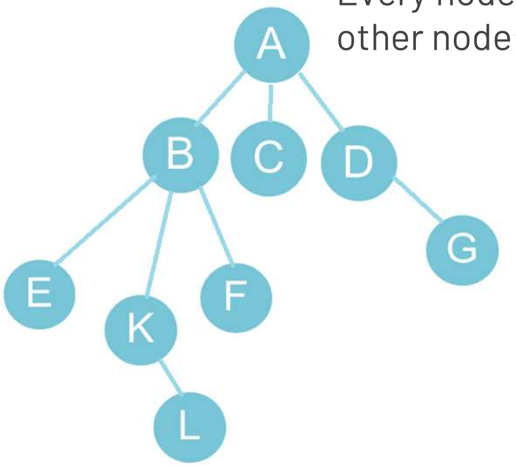
Path

Ancestor

Descendant

Trees

- Extend the concept of linked data structure to a structure that may have multiple relations among its nodes Such a structure is called a **tree**.
- A tree is a collection of nodes connected by directed (or undirected) edges.
- A tree can be empty with no nodes or a tree is a structure consisting of one node called the **root** and zero or one or more sub trees. A tree has following general properties:
  - One node is distinguished as a **root**;
  - Every node (exclude a root) is connected by a directed edge *from* exactly one other node; A direction is: *parent -> children*



A is a parent of B, C, D,
B is called a child of A.
on the other hand, B is a parent of E, F, K
In the given picture, the root has 3 subtrees.

Advantages of Trees

Trees are so useful and frequently used, because they have some very serious advantages:

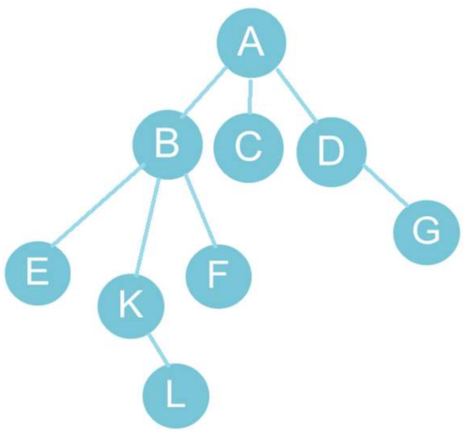
- Trees reflect structural relationships in the data.
- Trees are used to represent hierarchies.
- Trees provide an efficient insertion and searching.
- Trees are very flexible data, allowing to move subtrees around with minumum effort.

Trees

- 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
    - All nodes will have zero or more **child nodes** or *children*
    - For all nodes other than the root node, there is one parent node

Trees

- Each node can have *arbitrary* number of children. Nodes with no children are called **leaves**, or **external** nodes. In the above picture, C, E, F, L, G are leaves. Nodes, which are not leaves, are called **internal** nodes. Internal nodes have at least one child.
- Nodes with the same parent are called **siblings**. In the picture, B, C, D are called siblings. The **depth of a node** is the number of edges from the root to the node. The depth of K is 2. The **height of a node** is the number of edges from the node to the deepest leaf. The height of B is 2. The **height of a tree** is a height of a root from deepest node.



A is a parent of B, C, D,
B is called a child of A.
on the other hand, B is a parent of E, F, K

Trees

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

- I has three children: J, K and L

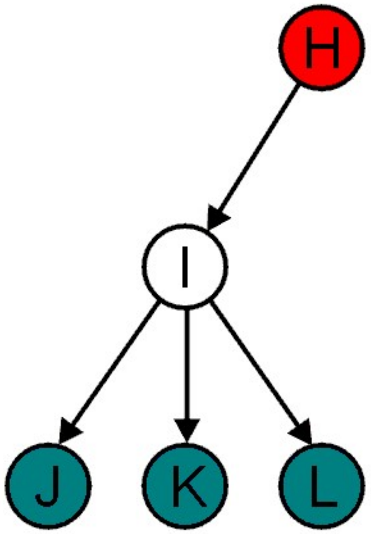
For all nodes other than the root node, there is one parent noc

- H is the parent

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

Nodes with the same parent are *siblings*

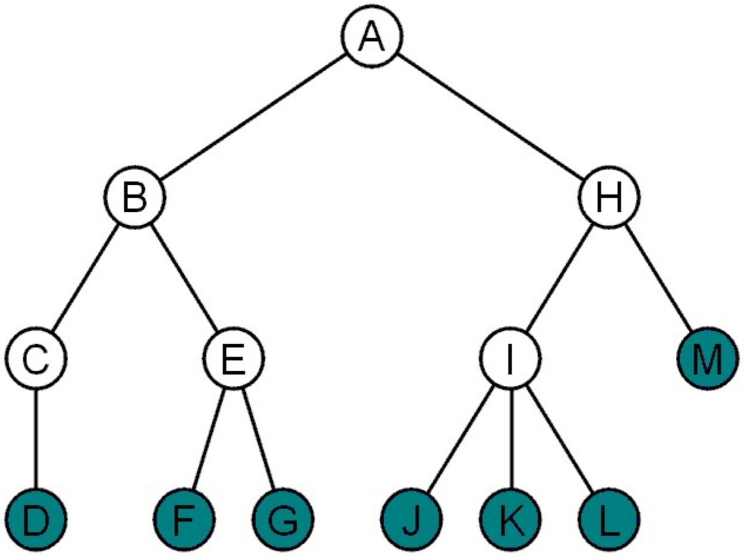
- J, K, and L are siblings



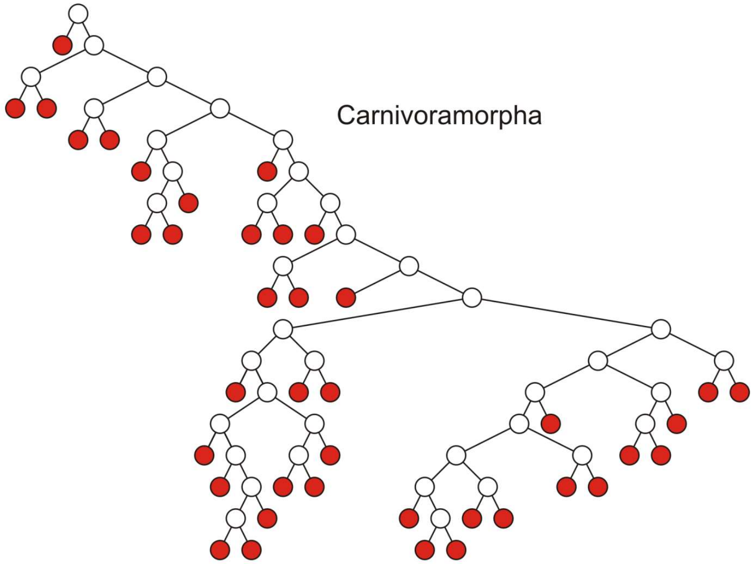
Trees

Nodes with degree zero are also called *leaf nodes*

All other nodes are said to be *internal nodes*, that is, they are internal to the tree

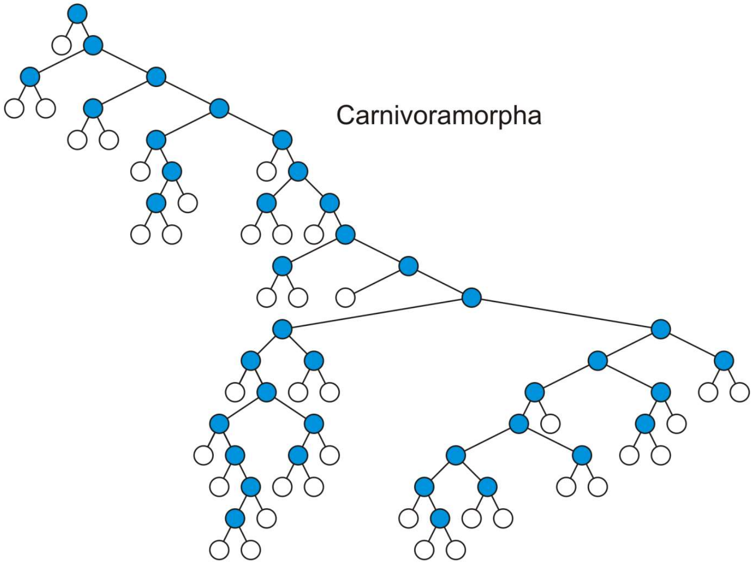


Tree



Wesley-Hunt, G. D.; Flynn, J. J. "Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of 'Miacoidea'

Leaf nodes:

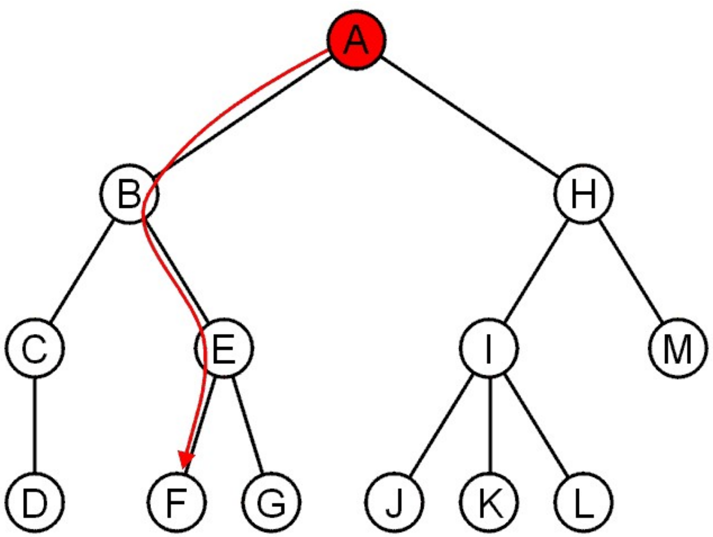


Wesley-Hunt, G. D.; Flynn, J. J. "Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of 'Miacoidea'

Internal nodes:

Trees

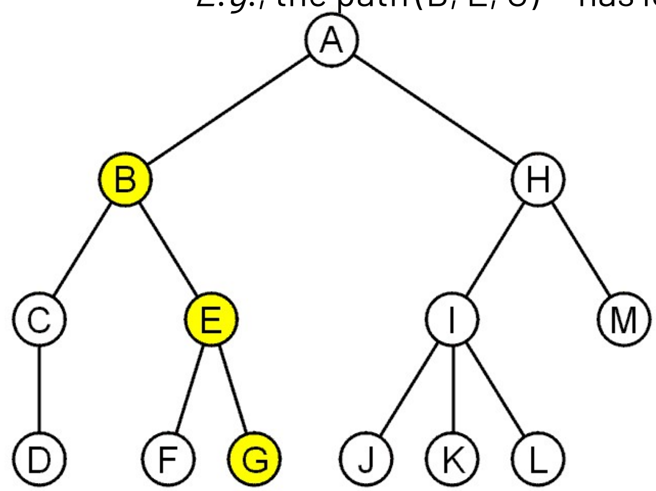
- The shape of a rooted tree gives a natural flow from the *root node*, or just *root*



A path is a sequence of nodes
$(a_0, a_1, \ldots, a_n)$
where a_{k+1} is a child of a_k is

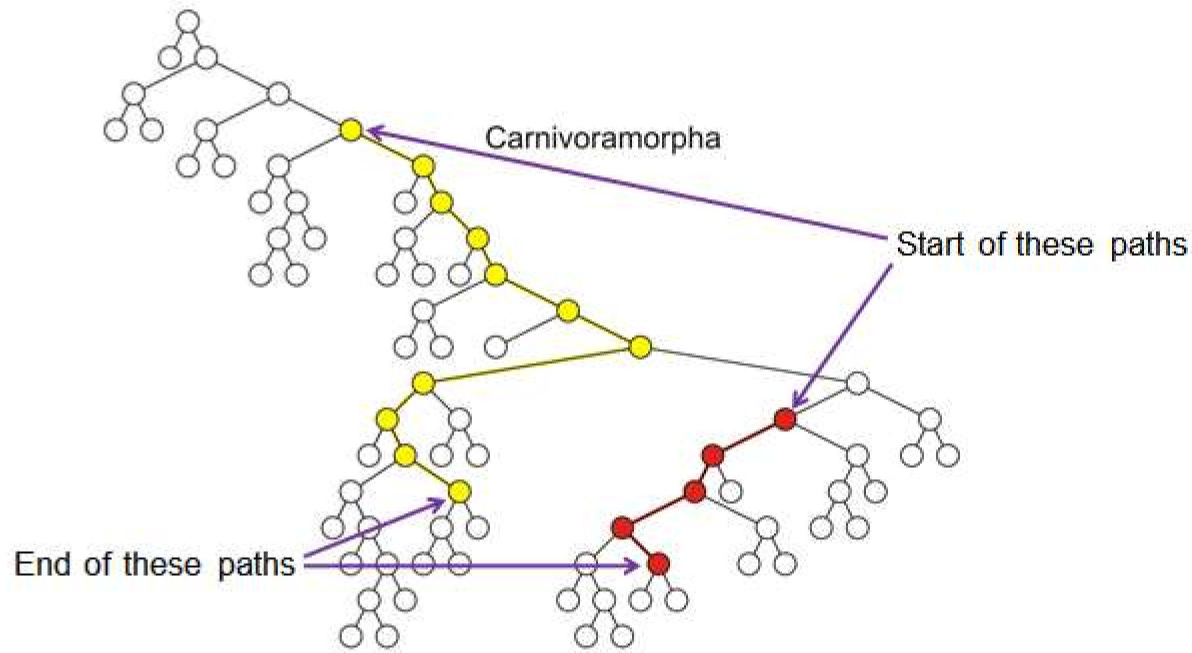
The length of this path is n

E.g., the path (B, E, G) has length 2



Trees

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

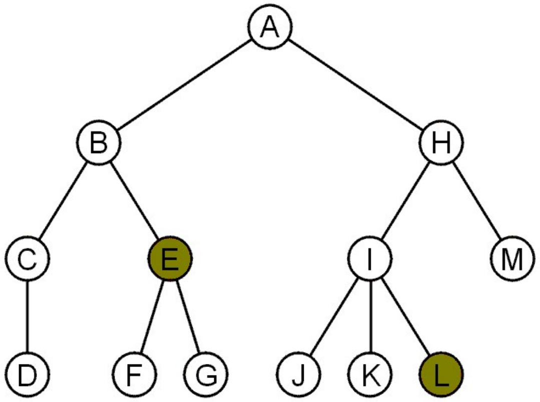


Trees

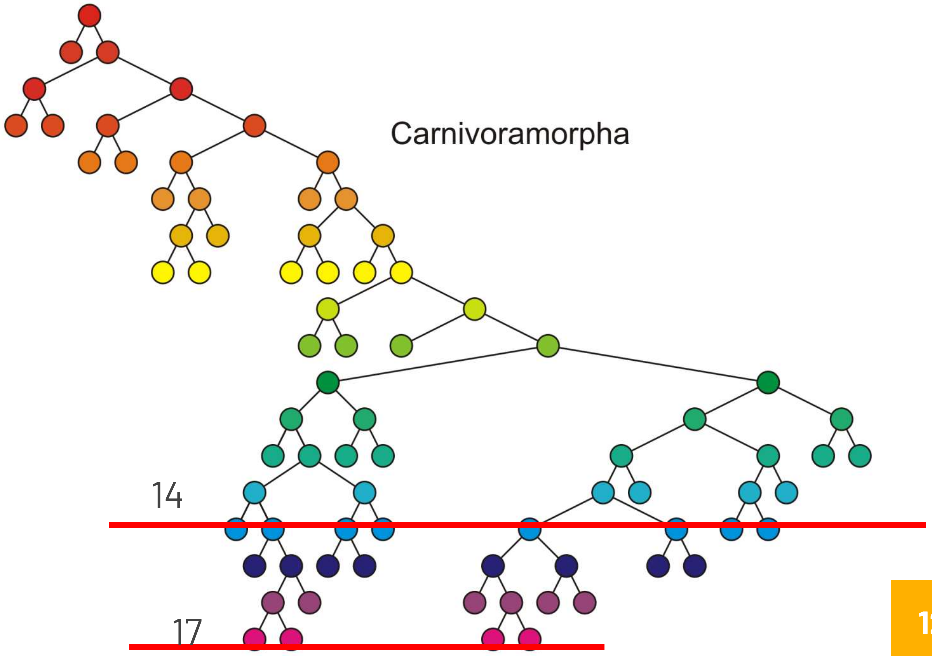
For each node in a tree, there exists a unique path from the root node to that node

The length of this path is the *depth* of the node, *e.g.*,

- E has depth 2
- L has depth 3



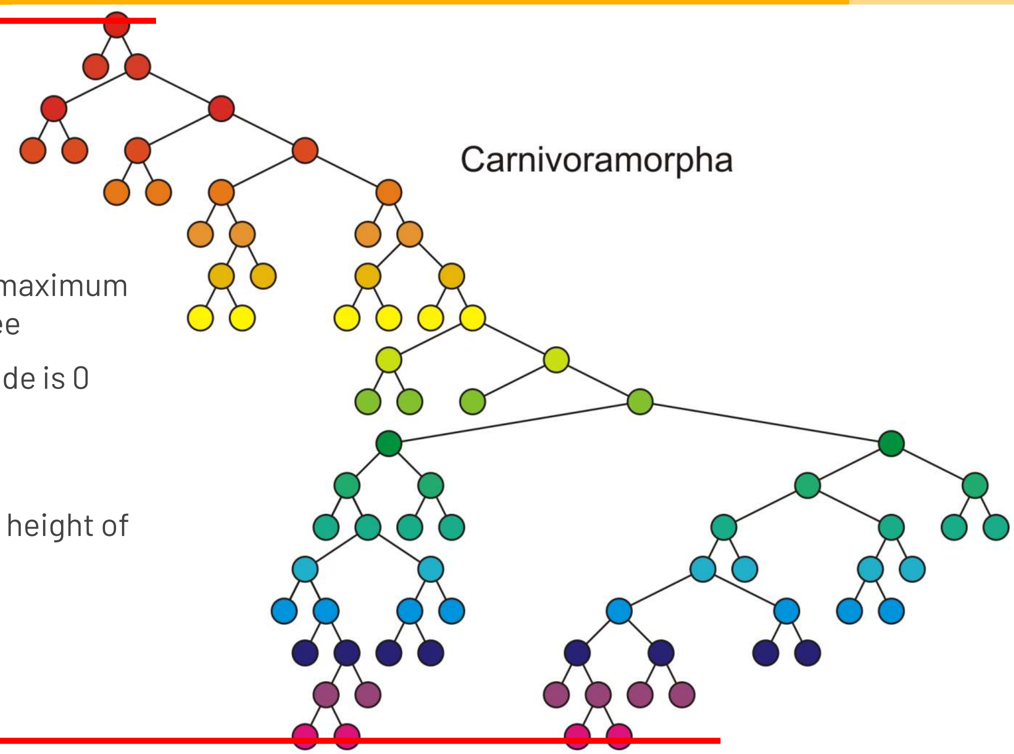
Nodes of depth up to 17



Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Trees

The height of this tree is 17



The *height* of a tree is defined as the maximum depth of any node within the tree

- 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

Wesley-Hunt, G. D.; Flynn, J. J. "Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of 'Miacoidea'

Trees

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

Trees

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

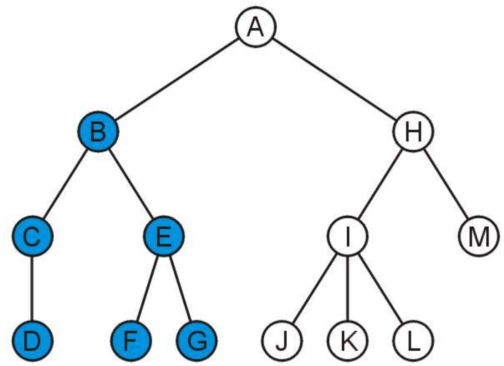
- *a* is an *ancestor* of *b*
- *b* is a *descendent* of *a*

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

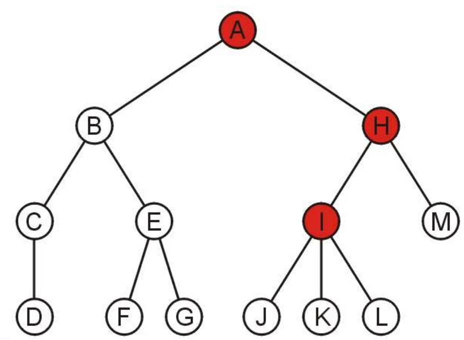
- We can add the adjective *strict* to exclude equality: *a* is a *strict descendent* of *b* if *a* is a descendant of *b* but $a \neq b$

The root node is an ancestor of all nodes

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

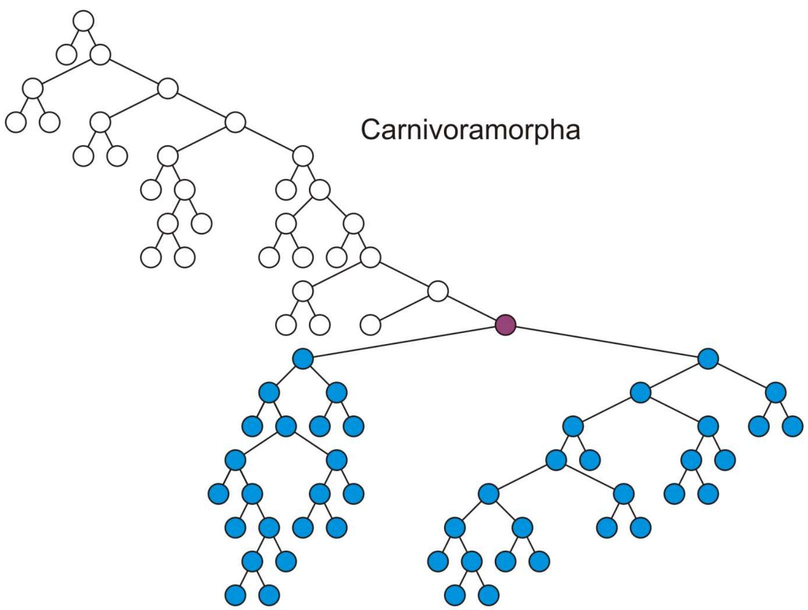


The ancestors of node I are I, H, and A:



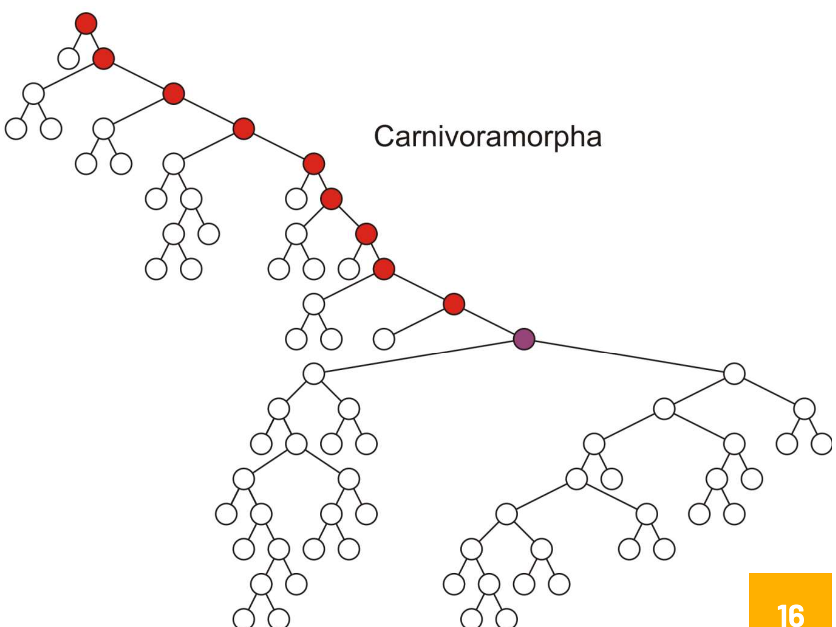
Trees

- All descendants (including itself) of the indicated node



Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

All ancestors (including itself) of the indicated node



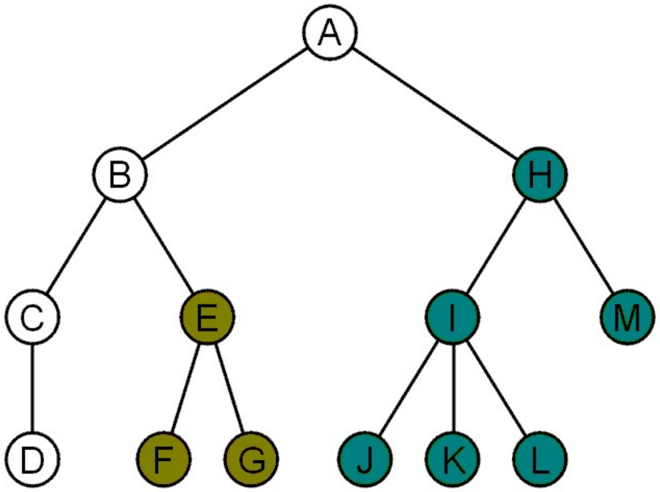
Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relation... among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Trees

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 *subtree of the tree with root a*



Trees

Examples

Example: XHTML

The XML of XHTML has a tree structure

Cascading Style Sheets (CSS) use the tree structure to modify the display of HTML

Consider the following XHTML document

<html>

<head>

<title>Hello World!</title>

</head>

<body>

<h1>This is a <u>Heading</u></h1>

<p>This is a paragraph with some

<u>underlined</u> text.</p>

</body>

</html>

title

heading

body of page

underlining

paragraph

Consider the following XHTML document

<html>

<head>

<title>Hello World!</title>

</head>

<body>

<h1>This is a <u>Heading</u></h1>

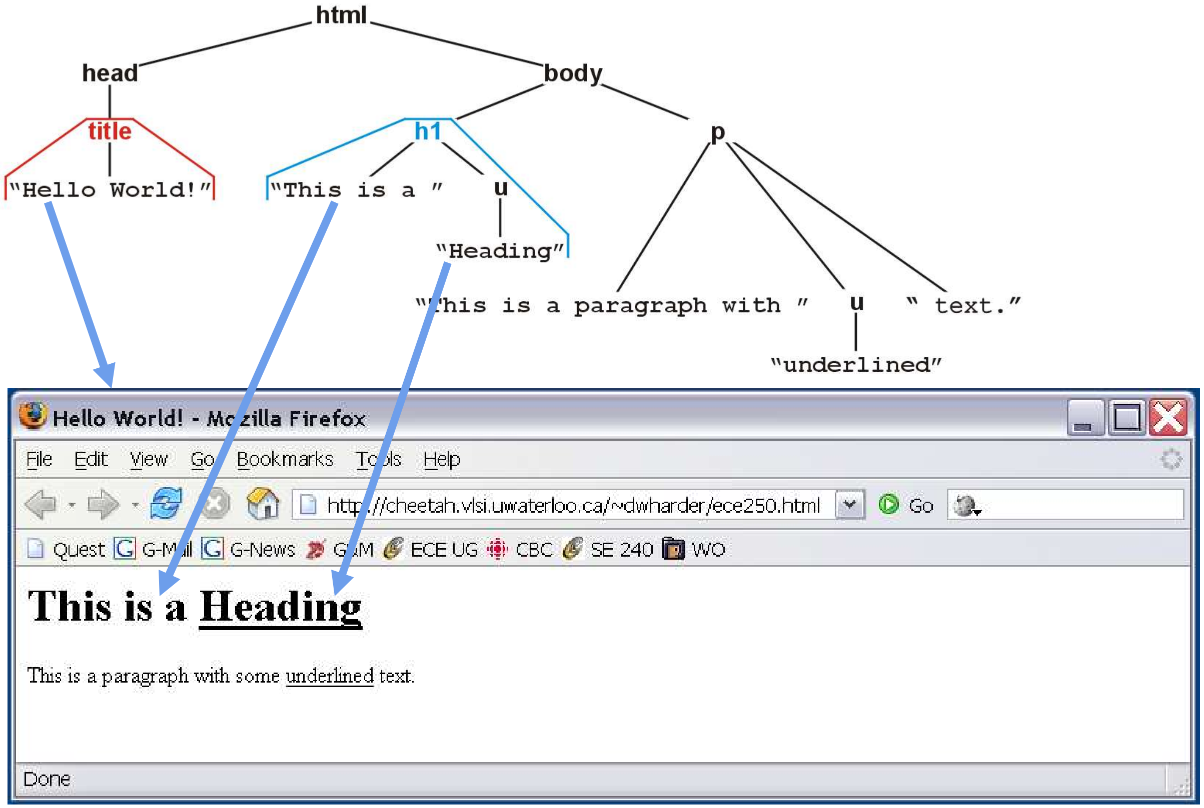
<p>This is a paragraph with some

<u>underlined</u> text.</p>

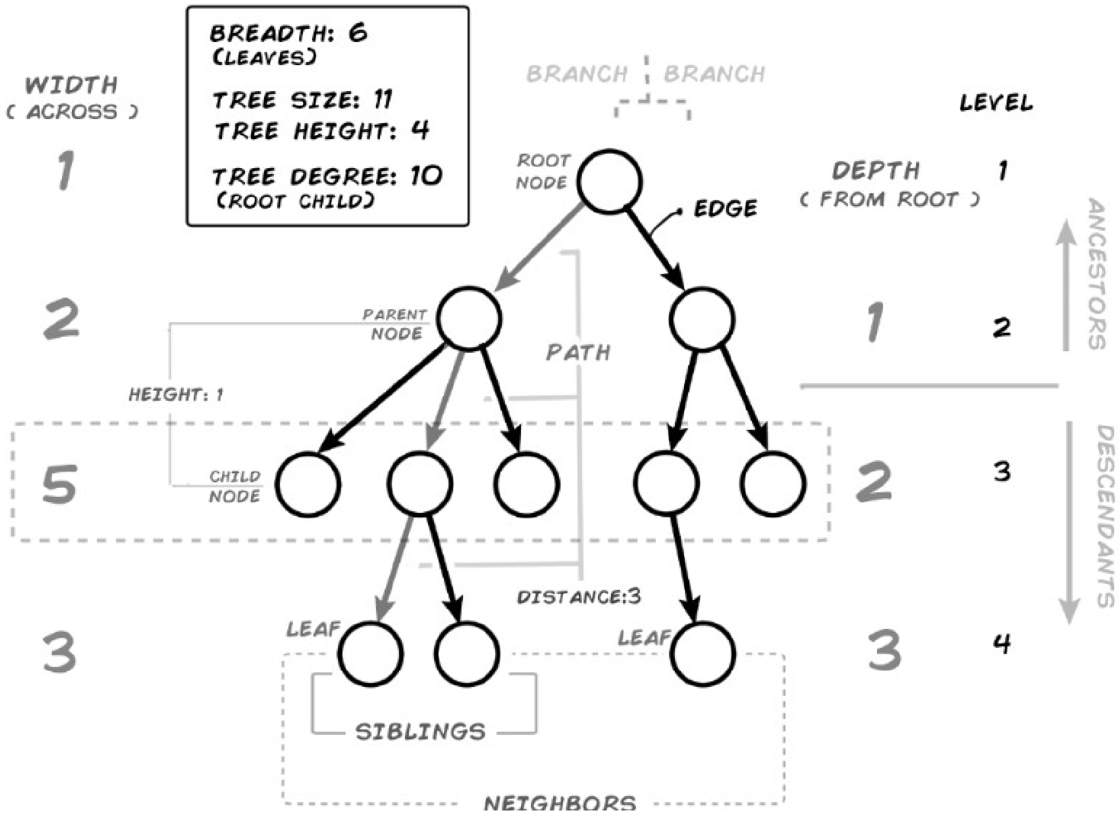
</body>

</html>

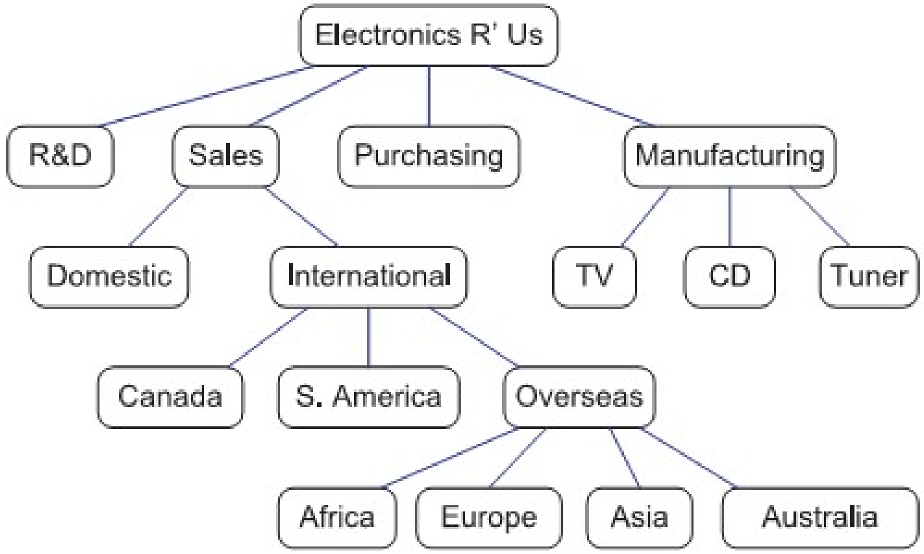
Example: XHTML

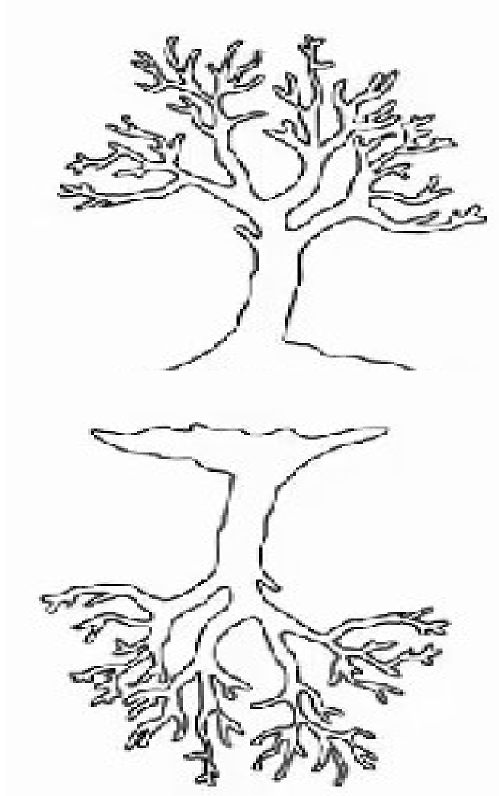


Trees


BREADTH: 6
(LEAVES)
TREE SIZE: 11
TREE HEIGHT: 4
TREE DEGREE: 10
(ROOT CHILD)
WIDTH
(ACROSS)
1
2
5
3
BRANCH
BRANCH
ROOT NODE
EDGE
PARENT NODE
PATH
HEIGHT: 1
CHILD NODE
DISTANCE:3
LEAF
LEAF
SIBLINGS
NEIGHBORS
LEVEL
DEPTH
(FROM ROOT)
1
2
3
4
1
2
3
ANCESTORS
DESCENDANTS

Trees


Electronics R' Us
R&D
Sales
Purchasing
Manufacturing
Domestic
International
TV
CD
Tuner
Canada
S. America
Overseas
Africa
Europe
Asia
Australia



Trees

Traversal

Trees

- **List of a tree's nodes** is called a traversal if it lists each tree node exactly once.
- Traversal is a process to visit all the nodes of a tree and may print their values.
- A traversal of a tree T is a systematic way of accessing, or “visiting,” all the nodes
- The three most commonly used traversal orders are recursively described as:
  - **Inorder:** traverse left subtree, visit current node, traverse right subtree
  - **Postorder:** traverse left subtree, traverse right subtree, visit current node
  - **Preorder:** visit current node, traverse left subtree, traverse right subtree