Data Structures and Algorithms
CS-206

Trees

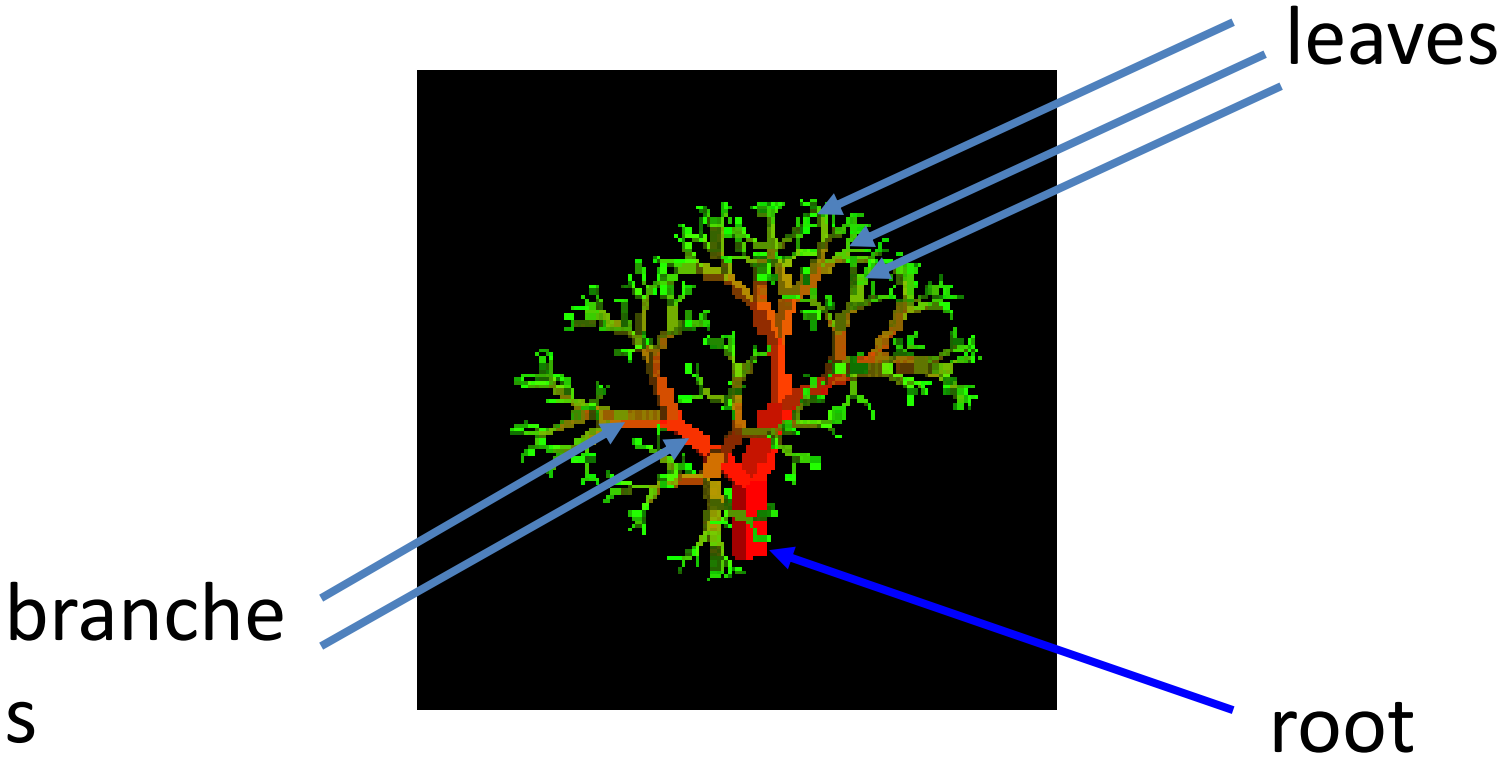
Instructor
Dr. Maria Anjum
Assistant Professor

Department of Computer Science
Lahore College for Women University

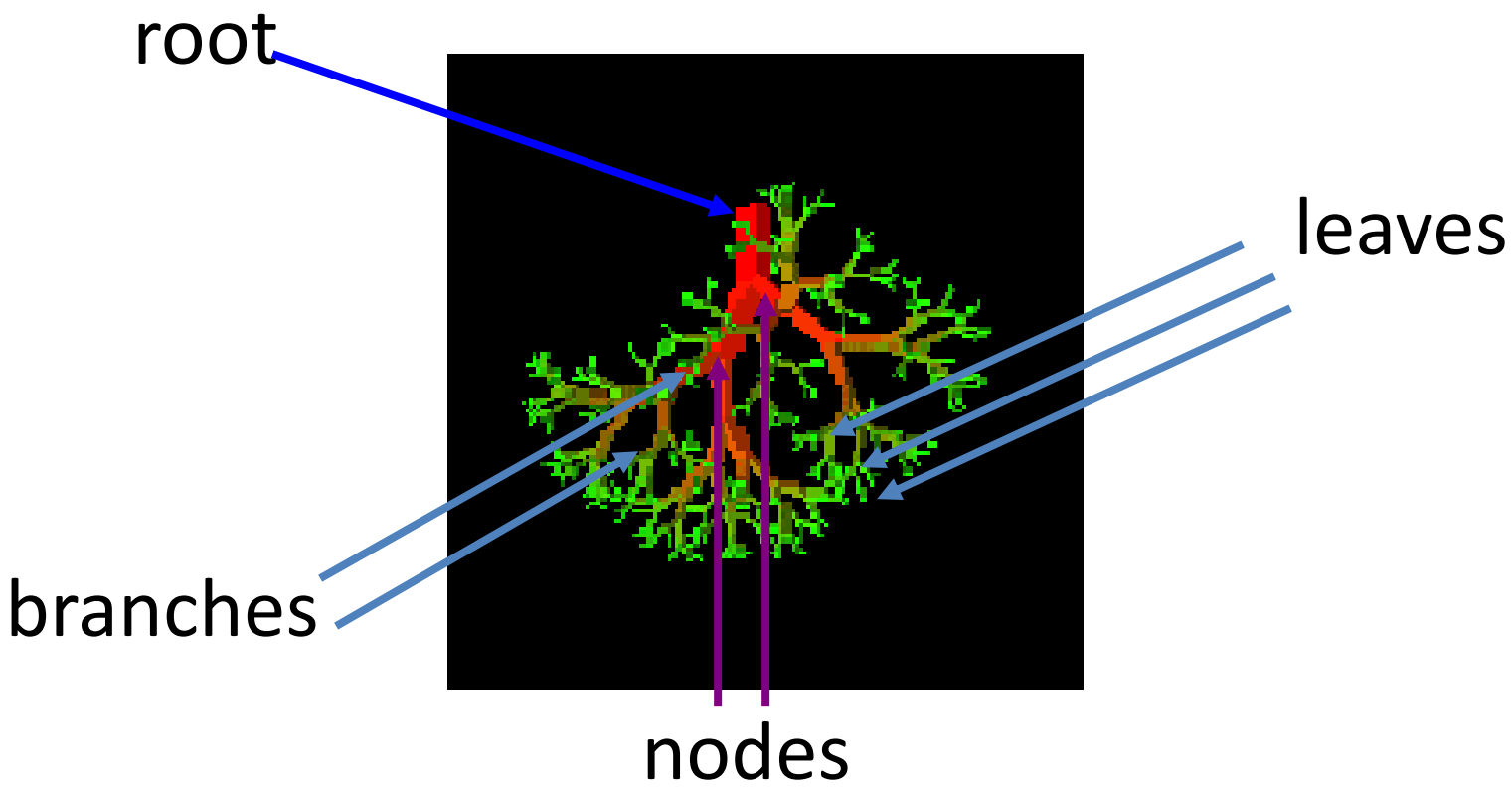
Topics Covered

- **Trees**
- Binary Tree
- Balanced Trees
- AVL Trees
- Spanning Trees

Nature View of a Tree



Computer Scientist's View

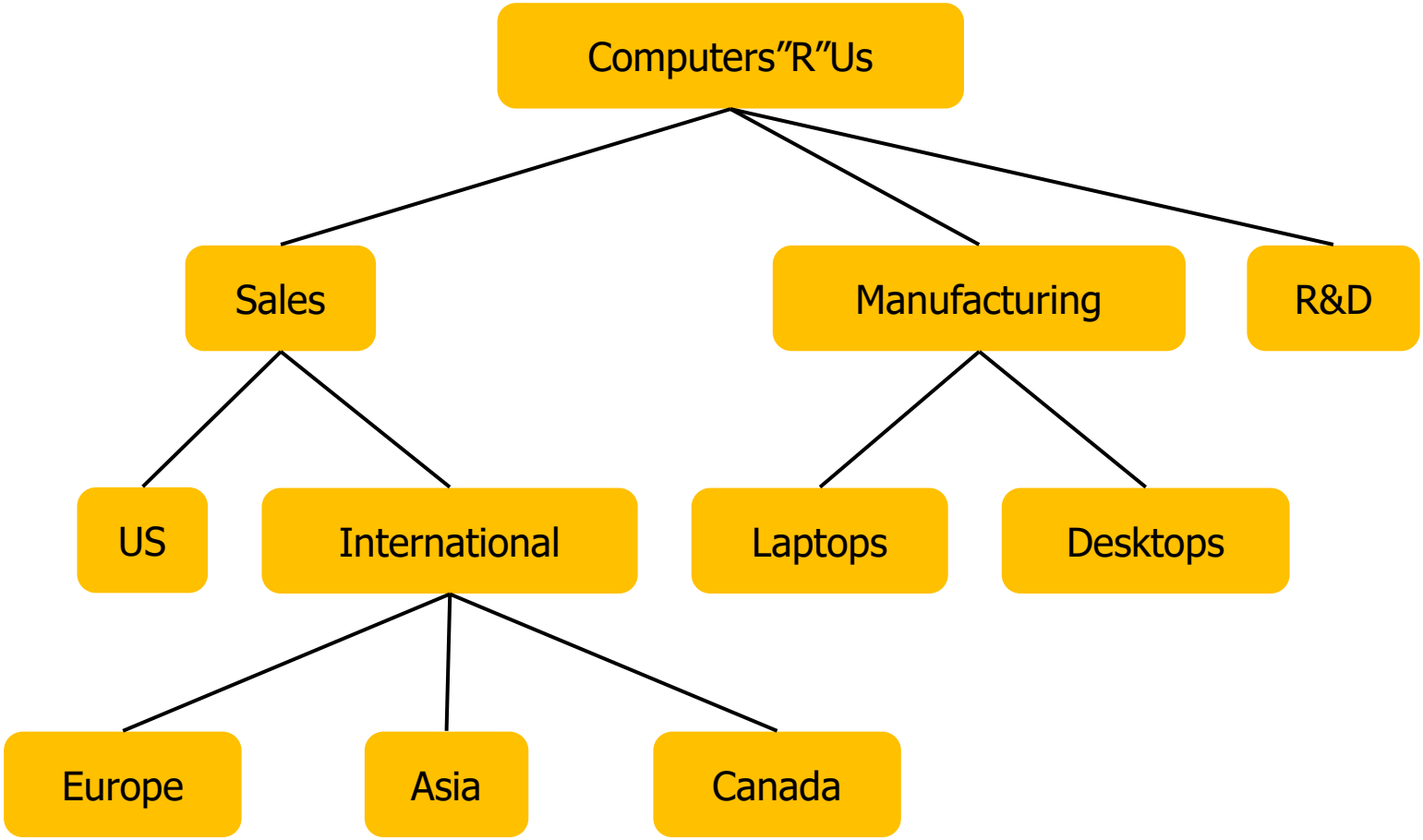


Tree --- Applictions

- Trees are used in many areas of computer science, including
  - Operating systems,
  - Graphics,
  - Programming languages,
  - File systems
  - Database systems, and
  - Computer networking etc.

  What about organizational charts.

Example


Computers"R"Us
Sales
Manufacturing
R&D
US
International
Laptops
Desktops
Europe
Asia
Canada

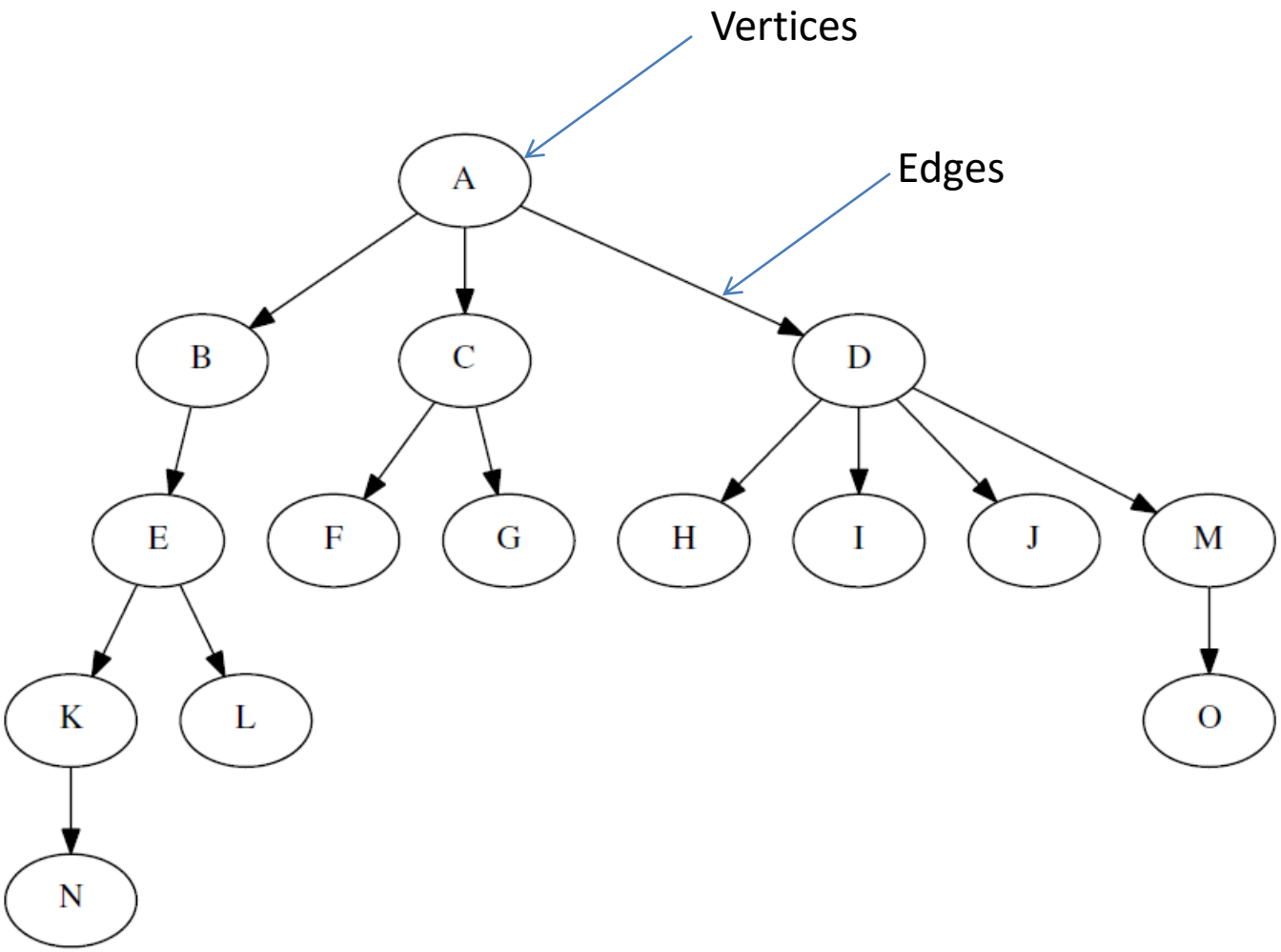
Tree

A tree is a structure in which each node can have multiple successors

A **tree** is a non-empty collection of **vertices** (nodes) and **edges** (lines) that satisfy certain requirements.

- Contrary to arrays, stacks, queues and sequences all of which are one-dimensional data structures, trees are **two-dimensional data structures with hierarchical relationship between data items.**

Tree --- Example



Tree Elements ---

- **Root**
  - The first node in a tree is called a root, it is often called the top level node.
  - The root of the tree is the only node in the tree that has no incoming edges.
- **Children**
  - Given a node in a tree, its successors (nodes connected to it in a level below) are called its **children**.

Tree Elements ---

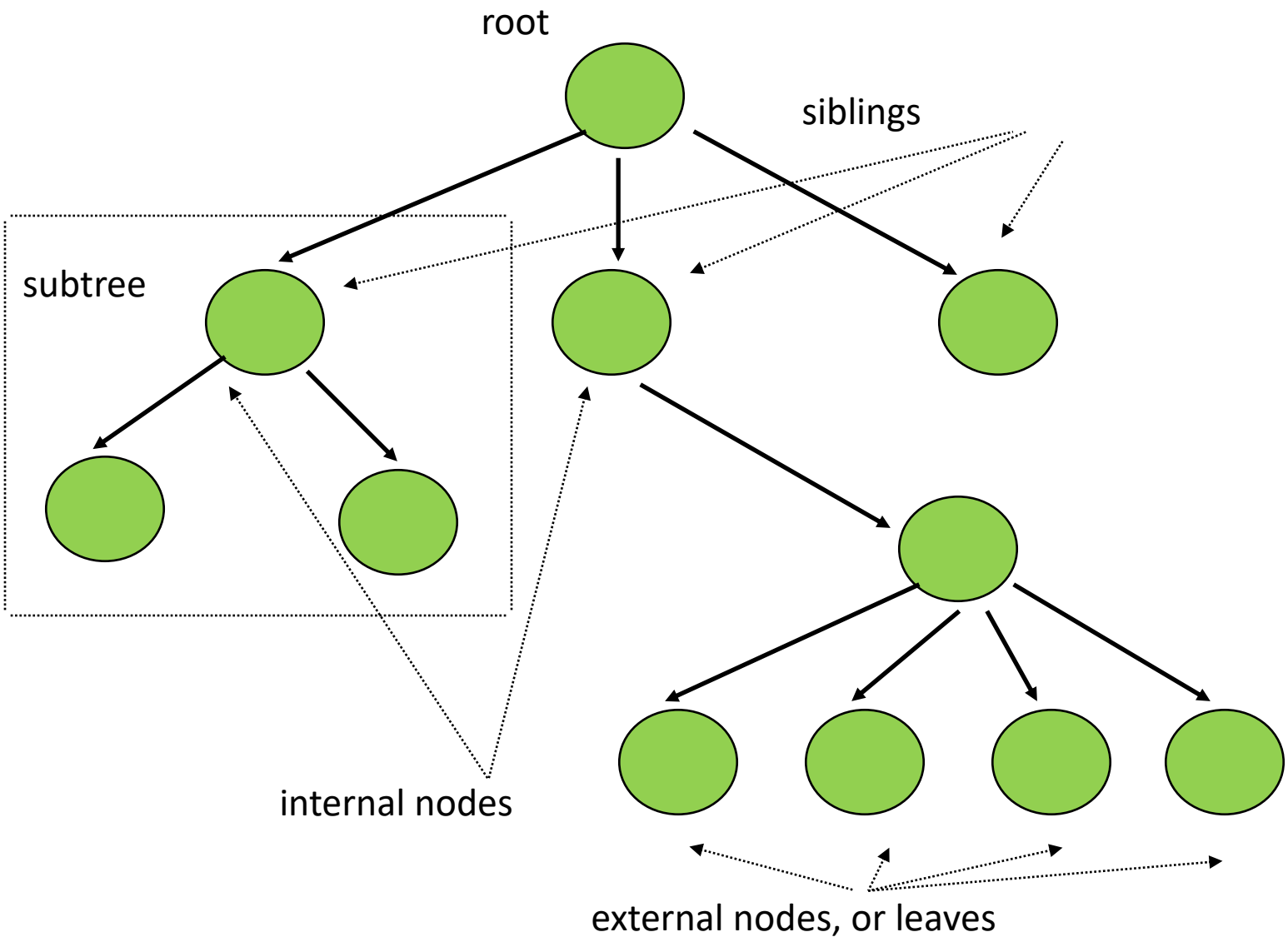
- **Parent**
  - Given a node in a tree, its predecessor (node that connects to it in a level above - there is only one such node) is called its **parent.**
  - A node is the parent of all the nodes only if it connects to with outgoing edges.

- **Siblings**
  - Nodes in the tree that are children of the same parent are said to be siblings.

Tree Elements ---

- **Subtree**
  - Any node within a tree can be viewed as a root of its own subtree - just take any node, cut of the branch that connects above to the rest of a tree, and it becomes a root with a smaller tree of its own.
  - A subtree is a set of nodes and edges comprised of a parent and all the descendants of that parent.
- **Leaves**
  - The nodes at the end of each path without children are called **leaves or end nodes or leaf nodes.**
  - A leaf node has no children.

Example


root
siblings
subtree
internal nodes
external nodes, or leaves

Tree Elements --- Path

- A **path** in a tree is a list of distinct vertices in which successive vertices are connected by edges in the tree.

- A path is an ordered list of nodes that are connected by edges.

  - In a tree, there is always a unique path from the root of a tree to every other node in a tree – this has an important consequence: there are no cycles in a tree.

  - One node in the tree is designated as the **root**. Each tree has exactly one path between the root and each of the other nodes. If there is more than one path between the root and some node, or no path at all, we have a **graph**.

Warning!

- The length of the path is the number of edges in the path. (Warning: Some texts use the number of **nodes** rather than the number of edges).

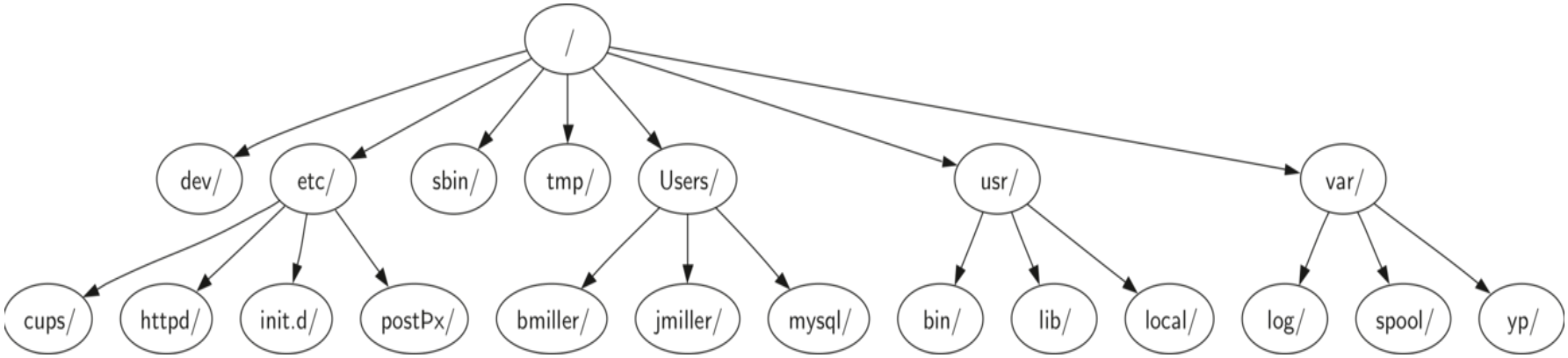
Tree Elements ---

- **Level**
  - The level of a node n is the number of edges on the path from the root node to n.
  - By definition, the level of the root node is zero.
- **Height**
  - The height of a tree is equal to the maximum level of any node in the tree.

Tree Important Considerations

- Every node (excluding a root) in a tree is connected by a directed edge from exactly one other node. This node is called a parent.
- On the other hand, each node can be connected to arbitrary number of nodes, called children.
- Nodes with no children are called leaves, or external nodes.
- Nodes which are not leaves are called internal nodes.
- Nodes with the same parent are called siblings.

Tree Example --- Unix file system hierarchy



Tree Example --- Unix file system hierarchy

- You can follow a path from the root to any directory. That path will **uniquely identify** that subdirectory (and all the files in it).

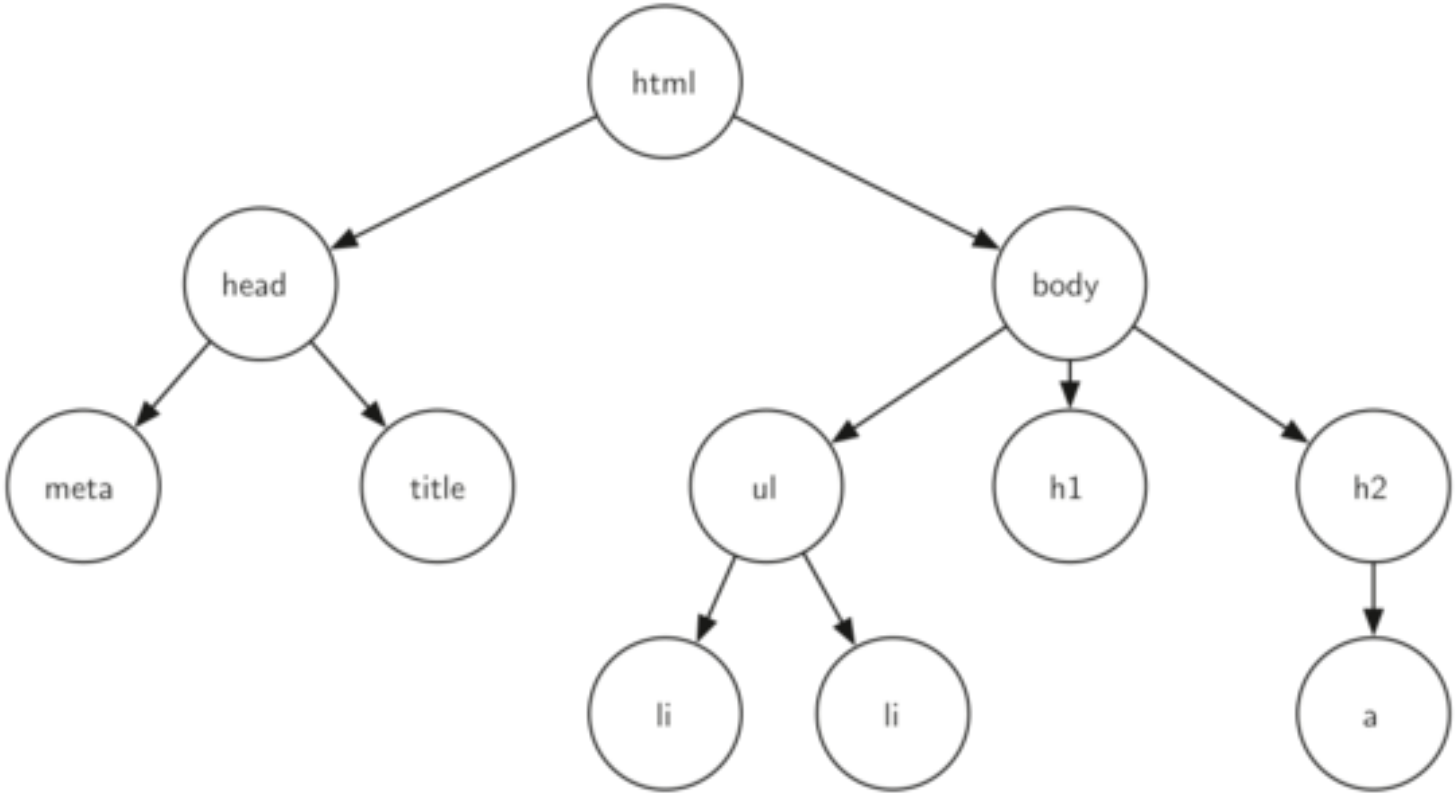
- You can move entire sections of a tree (called a **subtree**) to a different position in the tree without affecting the lower levels of the hierarchy.

- For example, take the entire subtree staring with **/etc/,** detach **etc/** from the root and reattach it under usr/. This would change the unique pathname to httpd from **/etc/httpd** to **/usr/etc/httpd,** but would not affect the contents or any children of the httpd directory.

Tree Example --- Web page

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-     8" />
<title>simple</title>
</head>
<body>
<h1>A simple web page</h1>
<ul>
            <li>List item one</li>
            <li>List item two</li>
</ul>
<h2><a href="http://www.cs.luther.edu">Luther CS </a><h2>
</body>
</html>
```

Tree Example --- Web page



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

Vocabulary of Tree

Height

Vertices

Root

Edges

Node

Subtree

Path

Parent

Siblings

Children

Internal Nodes

External Nodes

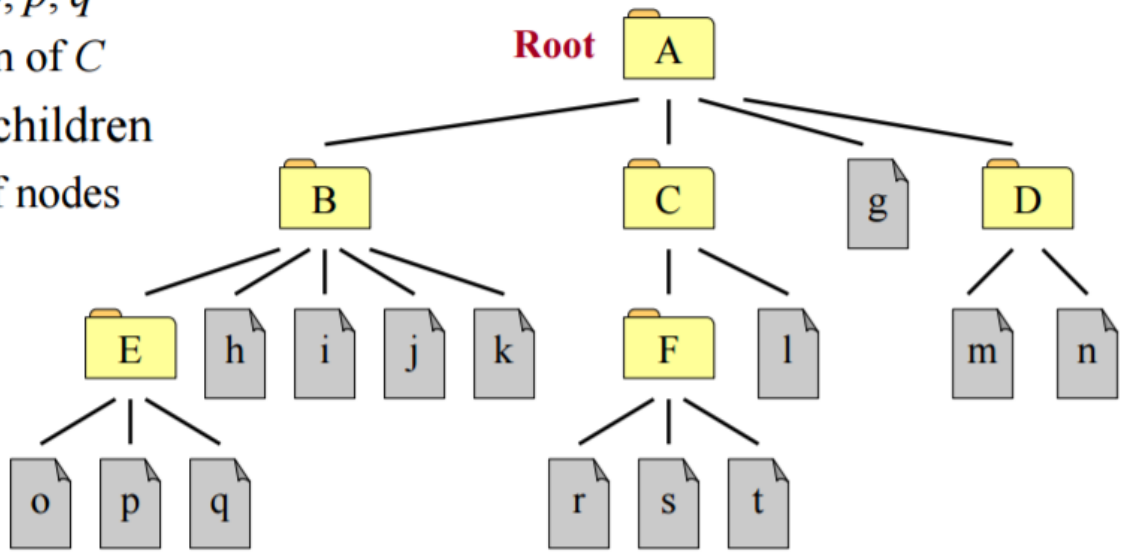
Leaves

Level

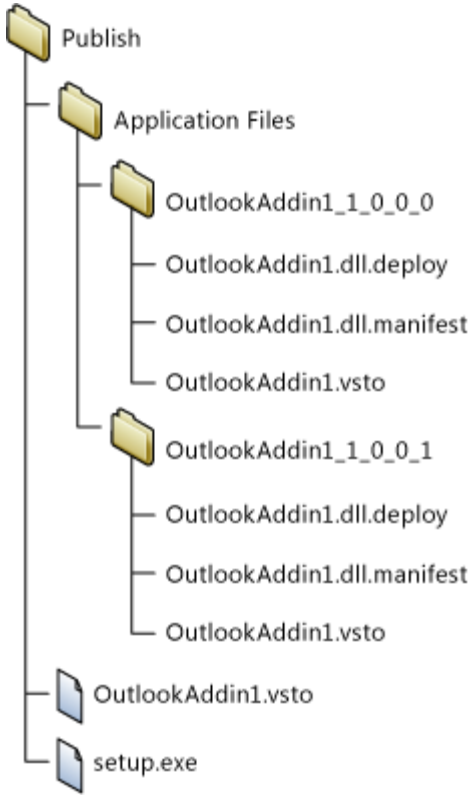
Tree Concepts --- Root, Parent, Child and Leaf

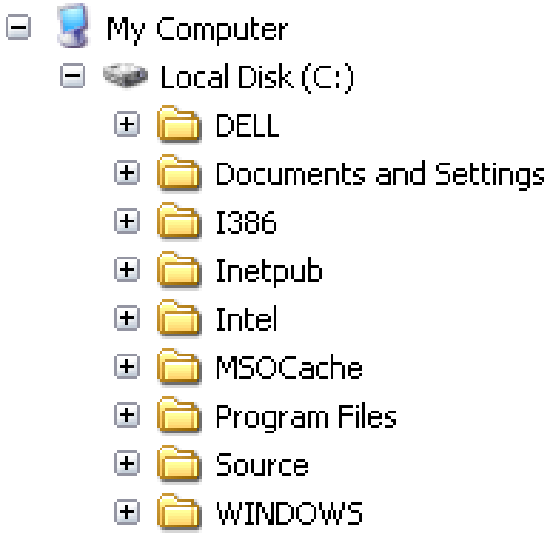
- A tree provides a way of storing **hierarchical data**
  - A **node** or a **vertex** is a container of data
  - The top node of the tree is identified as the **root node**
  - An **edge** is a link from a **parent node** to a successor node, called **child node**
    - *B* is the parent of *E*, *h*, *i*, *j*, and *k*
    - *E* is the parent of *o*, *p*, *q*
    - *F* and *l* are children of *C*
  - A **leaf node** has no children
    - *g* through *t* are leaf nodes

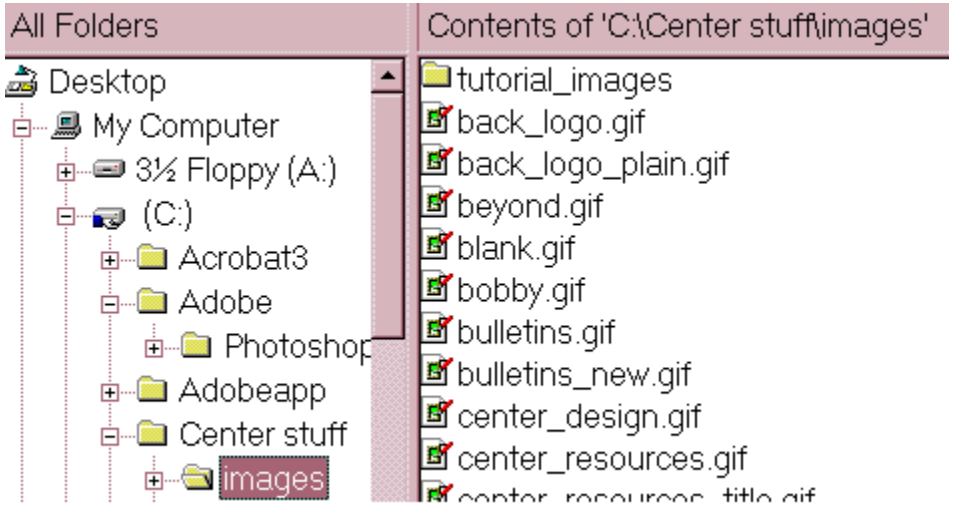
An example of a tree is a **hierarchical file system** consisting of **directories** and **files**



Examples

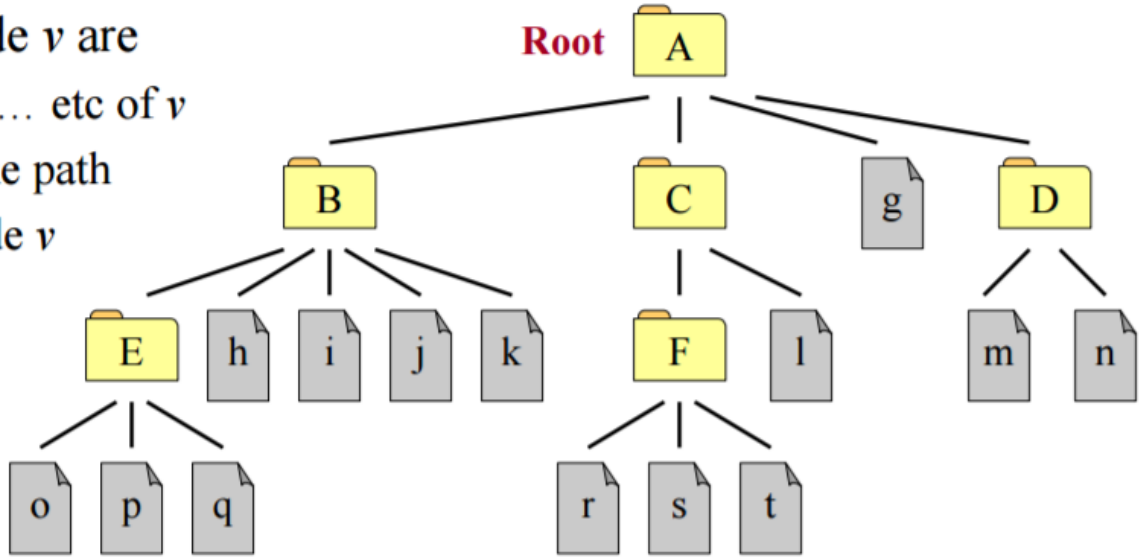

Publish
Application Files
OutlookAddin1_1_0_0_0
OutlookAddin1.dll.deploy
OutlookAddin1.dll.manifest
OutlookAddin1.vsto
OutlookAddin1_1_0_0_1
OutlookAddin1.dll.deploy
OutlookAddin1.dll.manifest
OutlookAddin1.vsto
OutlookAddin1.vsto
setup.exe


My Computer
Local Disk (C:)
DELL
Documents and Settings
I386
Inetpub
Intel
MSOCache
Program Files
Source
WINDOWS


All Folders
Contents of 'C:\Center stuff\images'
Desktop
My Computer
3½ Floppy (A:)
(C:)
Acrobat3
Adobe
Photoshop
Adobeapp
Center stuff
images
tutorial_images
back_logo.gif
back_logo_plain.gif
beyond.gif
blank.gif
bobby.gif
bulletins.gif
bulletins_new.gif
center_design.gif
center_resources.gif

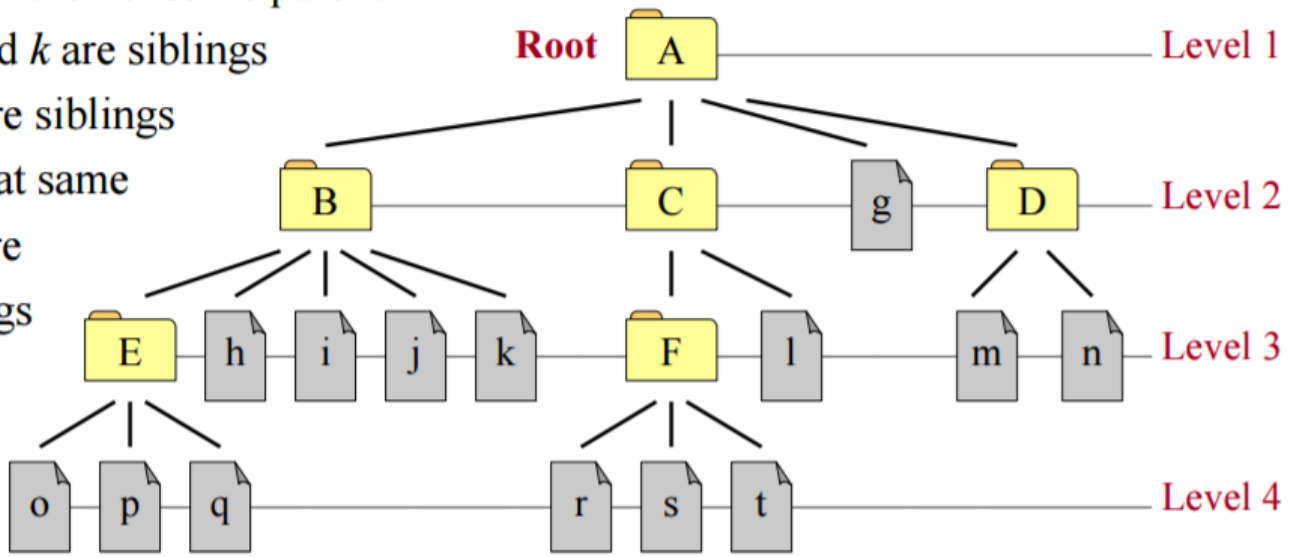
Tree Concepts – Path, Descendants, Ancestors

- A **path** from node v_0 to v_n is a sequence of nodes
  - $v_0, v_1, v_2, \ldots, v_{n-1}, v_n$, where there is an edge from one node to the next
  - Path from A to r is: A, C, F, r
- The **descendants** of a node v are
  - Children, grand children, grand grand children, … etc of v
  - All nodes reached by a path from node v to the leaf nodes
  - The descendants of C are: F, l, r, s, and t
- The **ancestors** of a node v are
  - Parent, grand parent, … etc of v
  - All nodes found on the path from root node to node v
  - Ancestors of p are E, B, and A



Tree Concepts – Level, Height, Siblings

- The **level** of a node *v* is the number of vertices in the path from root to *v*
  - Root node *A* is at level 1
  - Leaf nodes *o*, *p*, *q*, *r*, *s*, and *t* are at level 4
- The **height** of a tree is the maximum level
  - An empty tree has height 0
- Nodes are called **siblings** iff
  - They are children of same parent
  - *E*, *h*, *i*, *j*, and *k* are siblings
  - *r*, *s*, and *t* are siblings
  - *q* and *r* are at same level, but are NOT siblings



Special Trees

- An empty tree has a height of zero.
- A single node tree is a tree of height 1.
  - This is the only case where a node is both a root and a leaf.

References and Reading Material

- http://cs.nyu.edu/~joannakl/cs102_f14/notes/lecture06_TreesAndBST.pdf
- https://www.cs.cmu.edu/~adamchik/15-121/lectures/Trees/trees.html
- http://interactivepython.org/runestone/static/pythonds/Trees/ExamplesofTrees.html#fig-filetree
- https://dvanderboom.wordpress.com/2008/03/15/treet-implementing-a-non-binary-tree-in-c/
- http://web.york.cuny.edu/~kzaman/cs291/trees.pdf