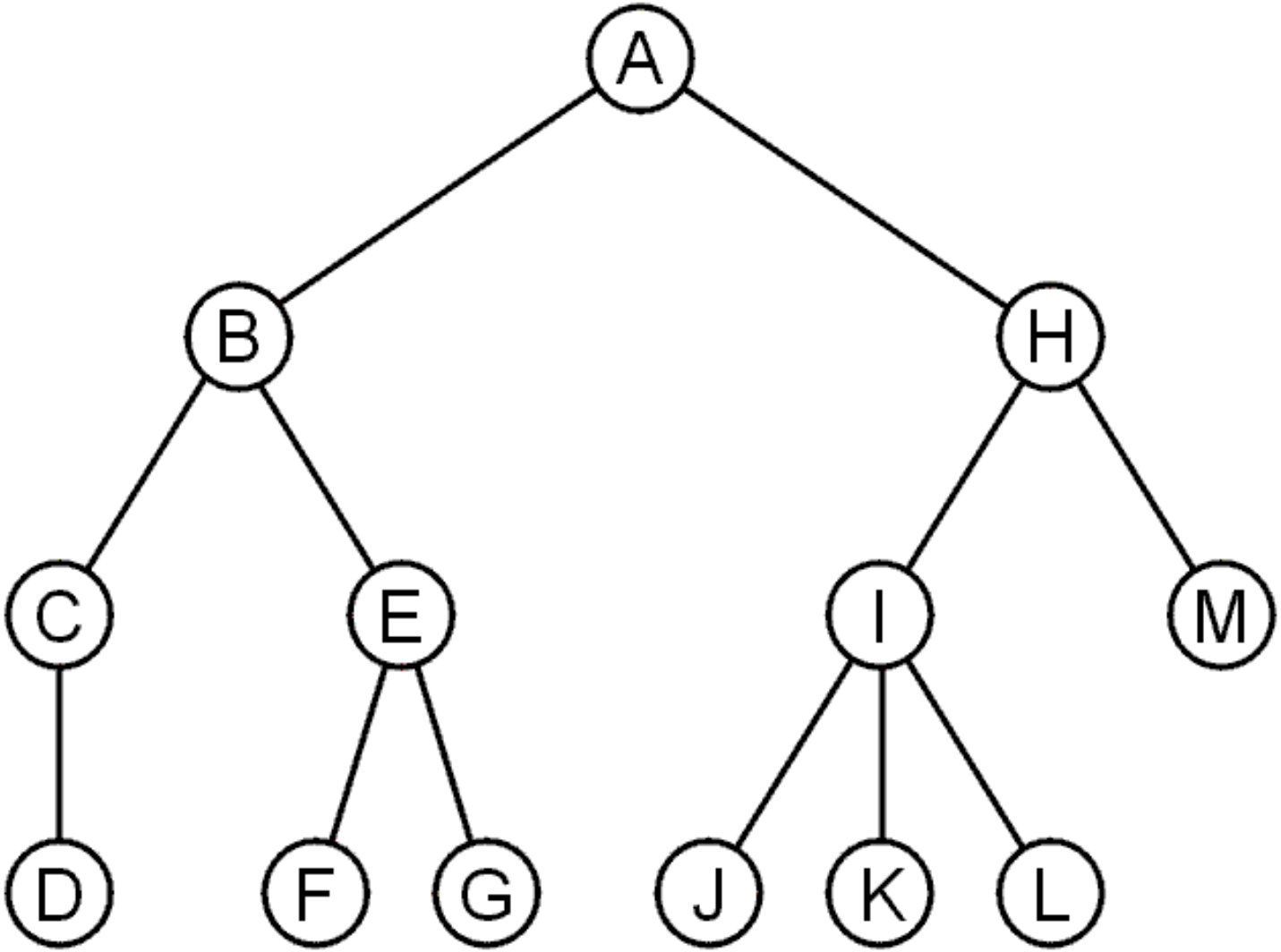
Trees

CSCI-2270

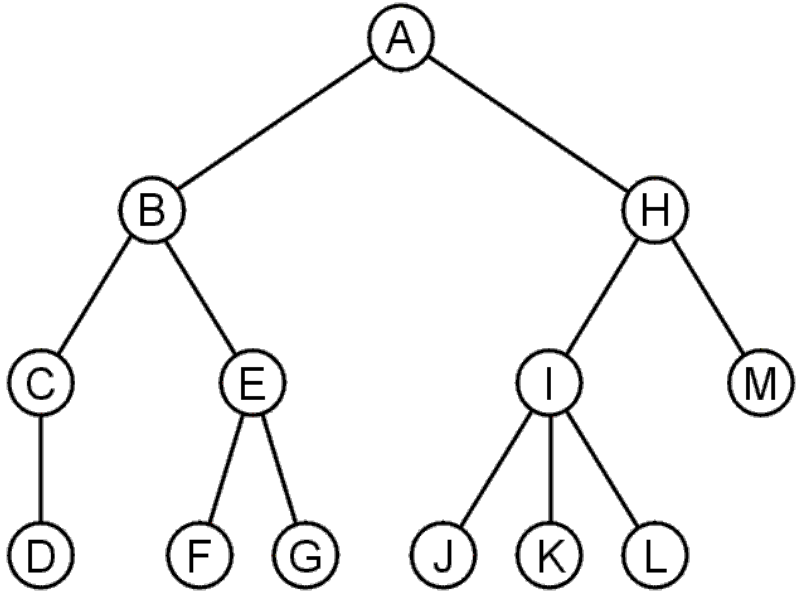
Elizabeth Boese

Trees



Trees

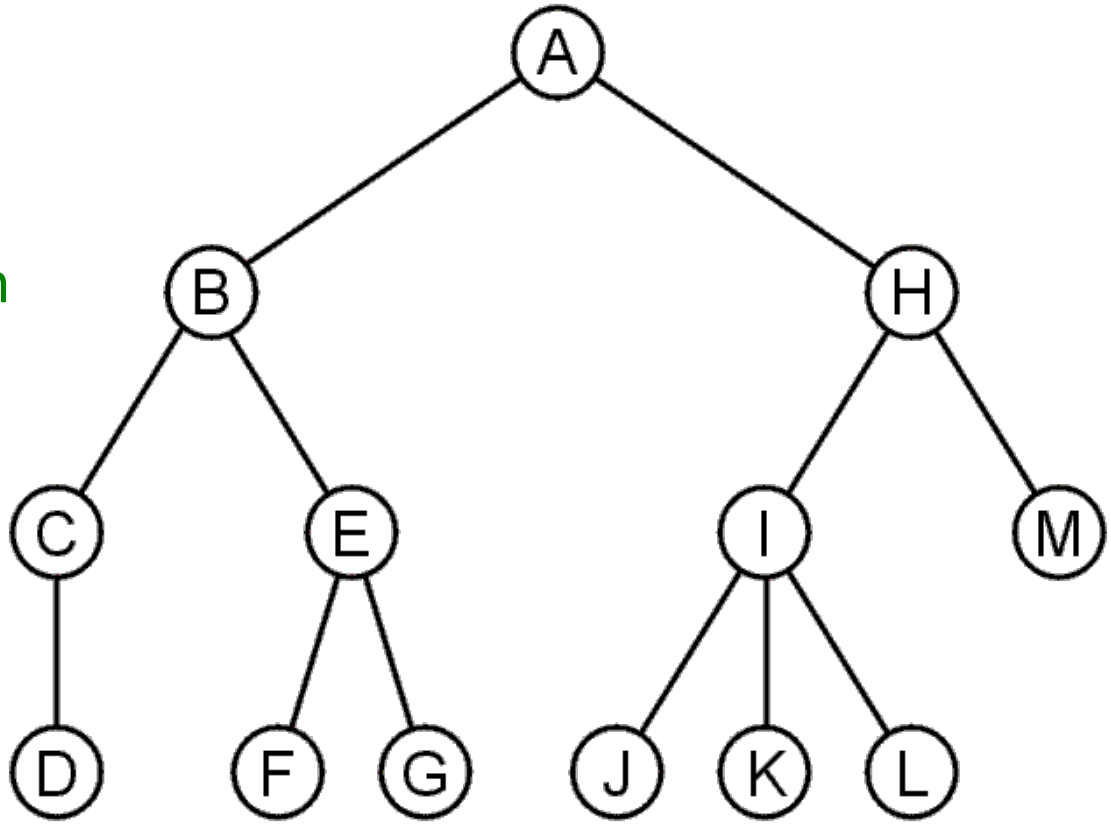
- Nodes
  - root
  - child nodes, children
  - Has exactly one node pointing to it *(except root)*
    - *Variation: pointer to parent*



Trees

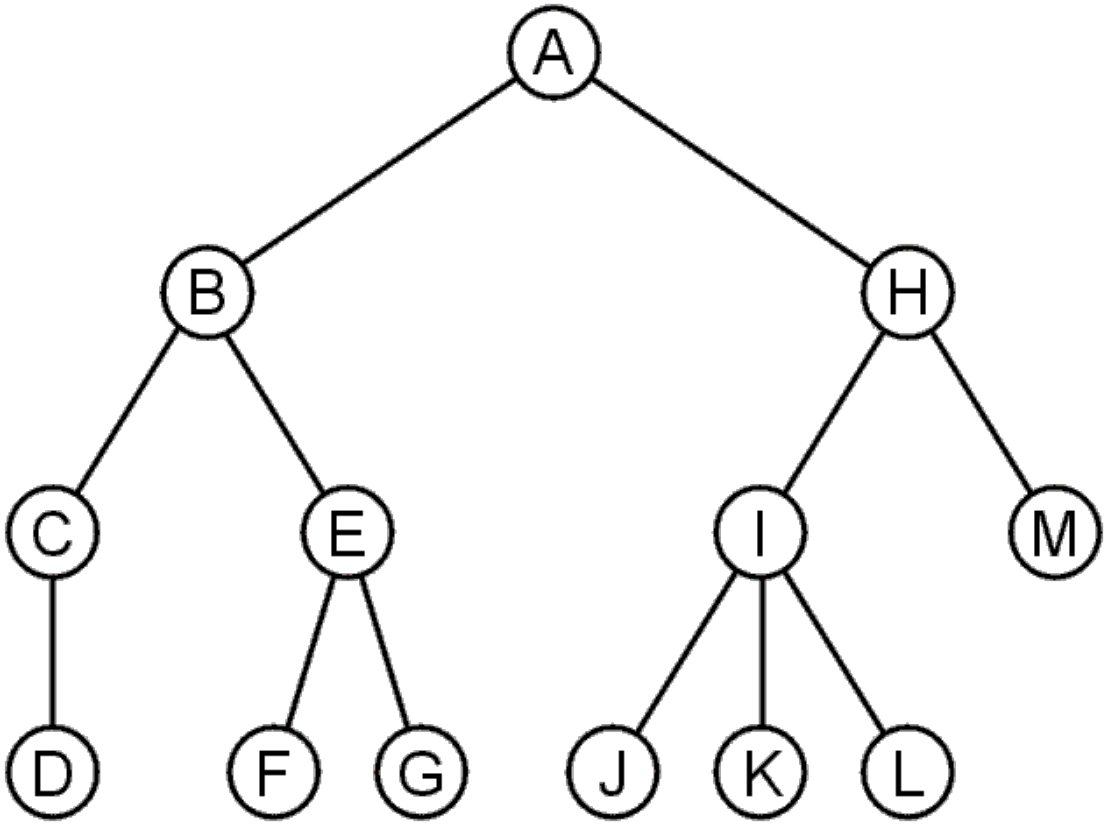
Terminology

- Node
  - (a.k.a. vertex) a data element in the tree
- Edge
  - (a.k.a. branch, or link) a connection between two nodes
- Empty tree
  - has zero nodes
- Size
  - the size of a tree is the number of nodes



Trees

- Root
- Parent
- Children
- Siblings
- Leaf nodes
- Internal nodes
- Degree
  - # of children of a node
  - deg(node)



Trees

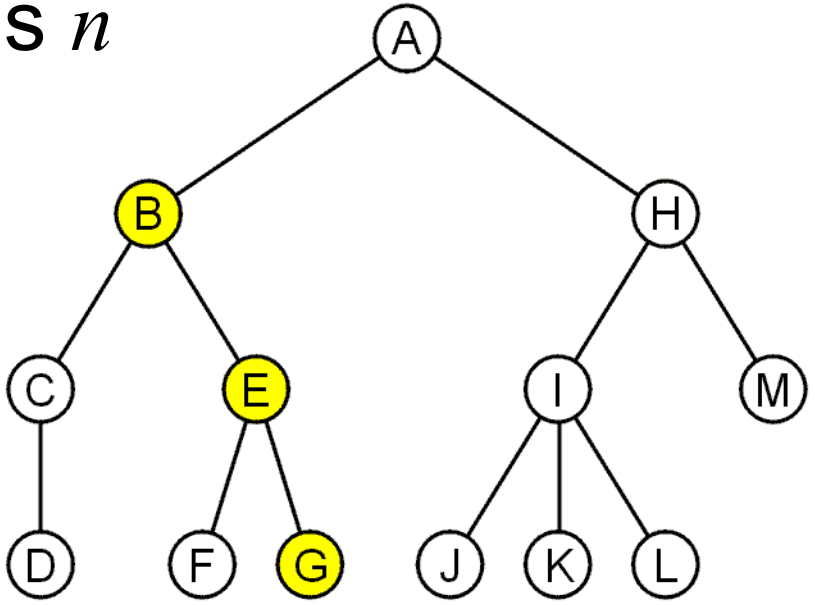
A path is a sequence of nodes

$$(a_0, a_1, ..., a_n)$$

where a_{k+1} is a child of a_k is

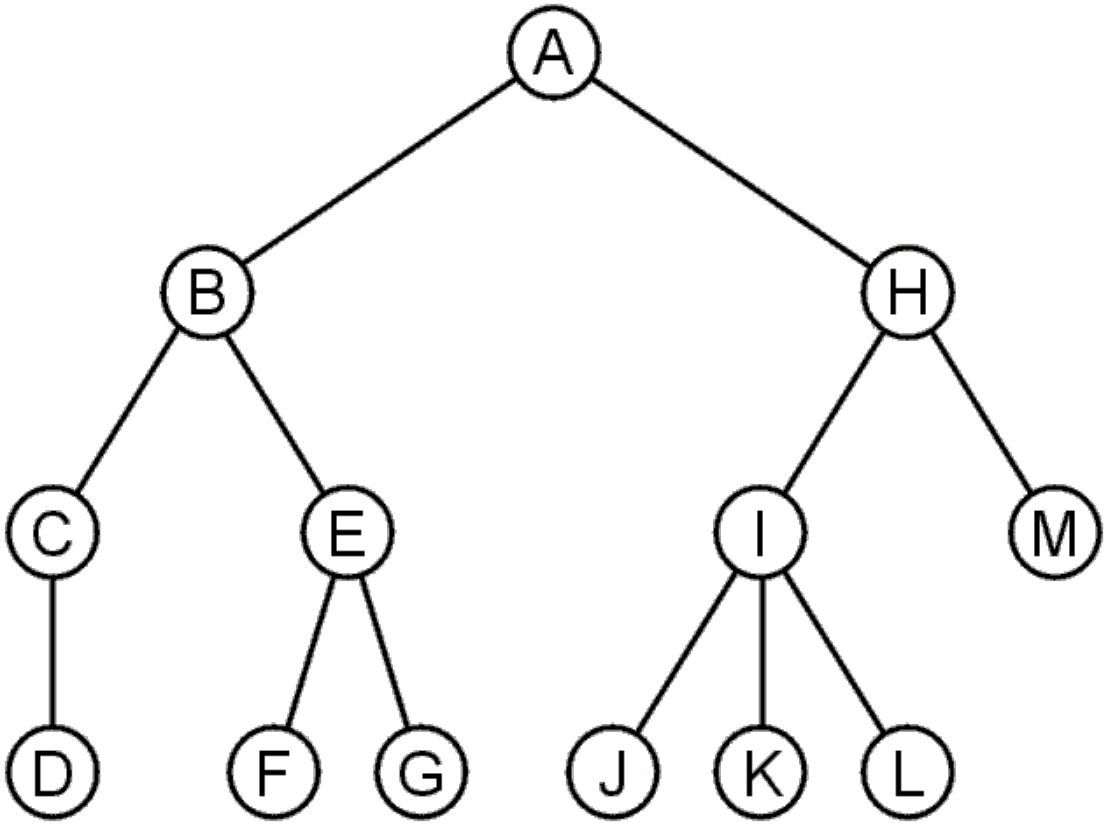
The length of this path is n

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



Trees

- Path
  - a sequence $n_0e_1n_1e_2n_2...e_kn_k$ where $k \geq 0$ and e_i connects n_{i-1} and n_i
- Start(p)
  - n_0
- End(p)
  - n_k
- Length(p)
  - k
- Simple Path
  - a path in which all nodes are distinct

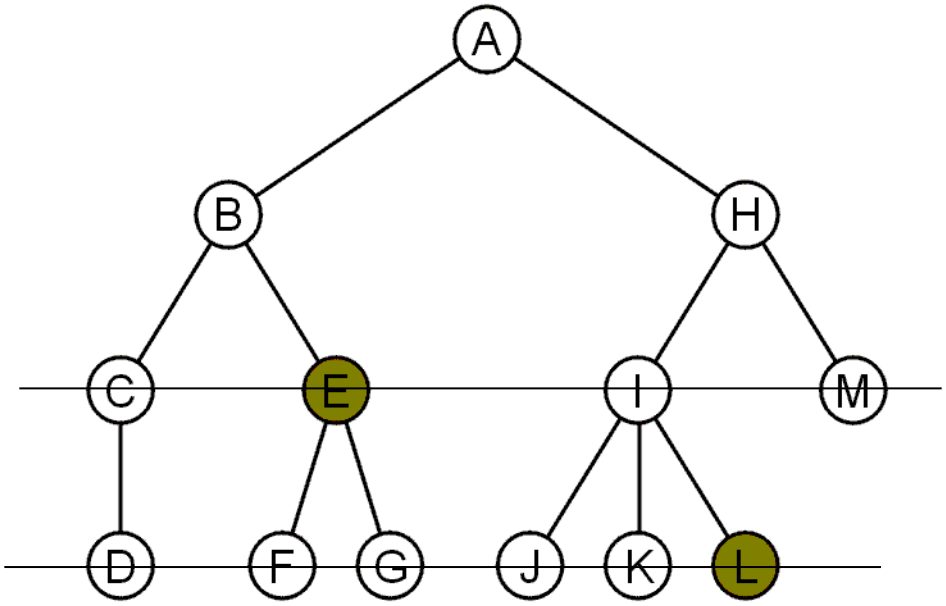


Trees

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



Terminology

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

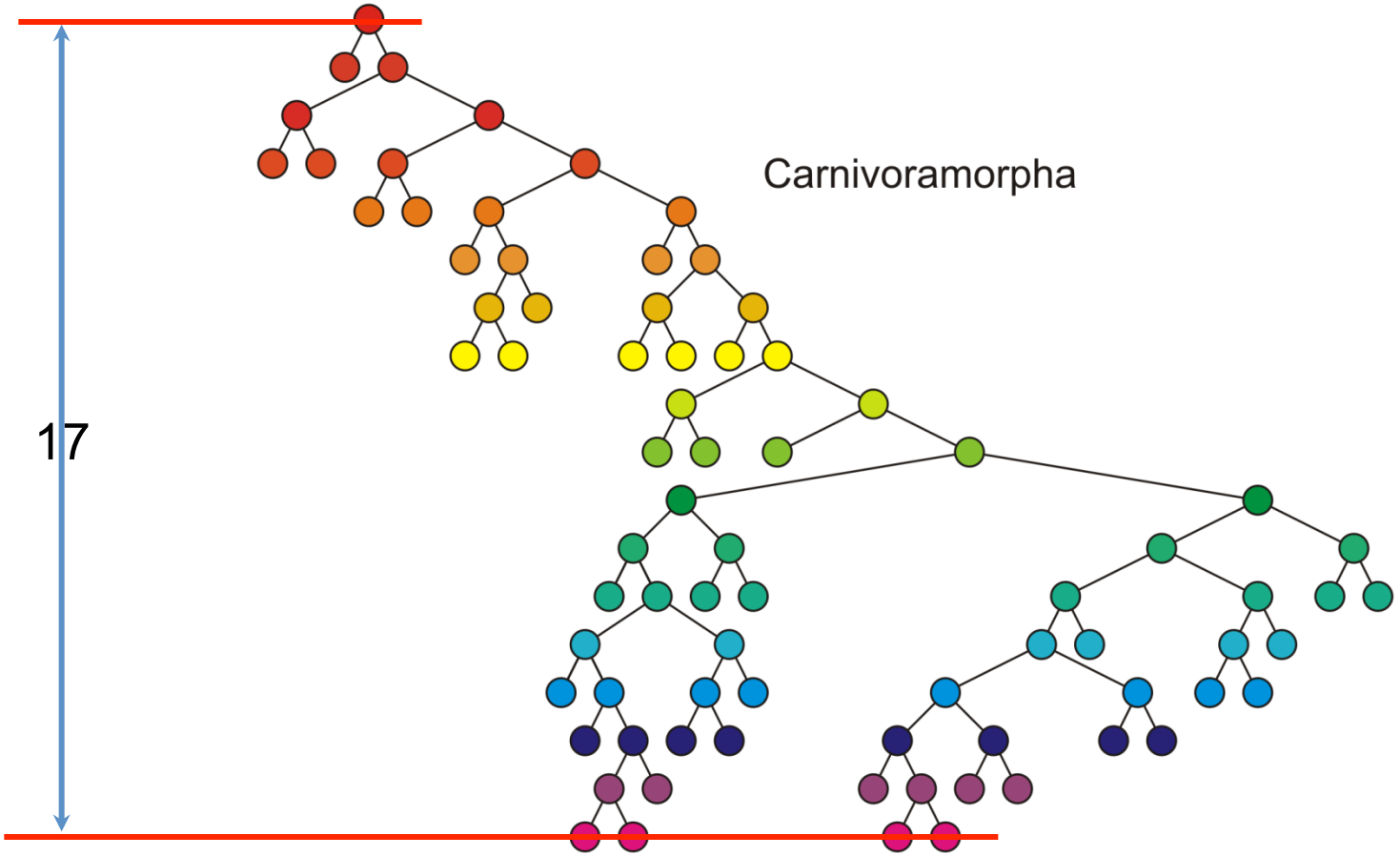
The height of a tree with one node is 0

- Just the root node

For convenience, we define the height of the empty tree to be -1

Trees

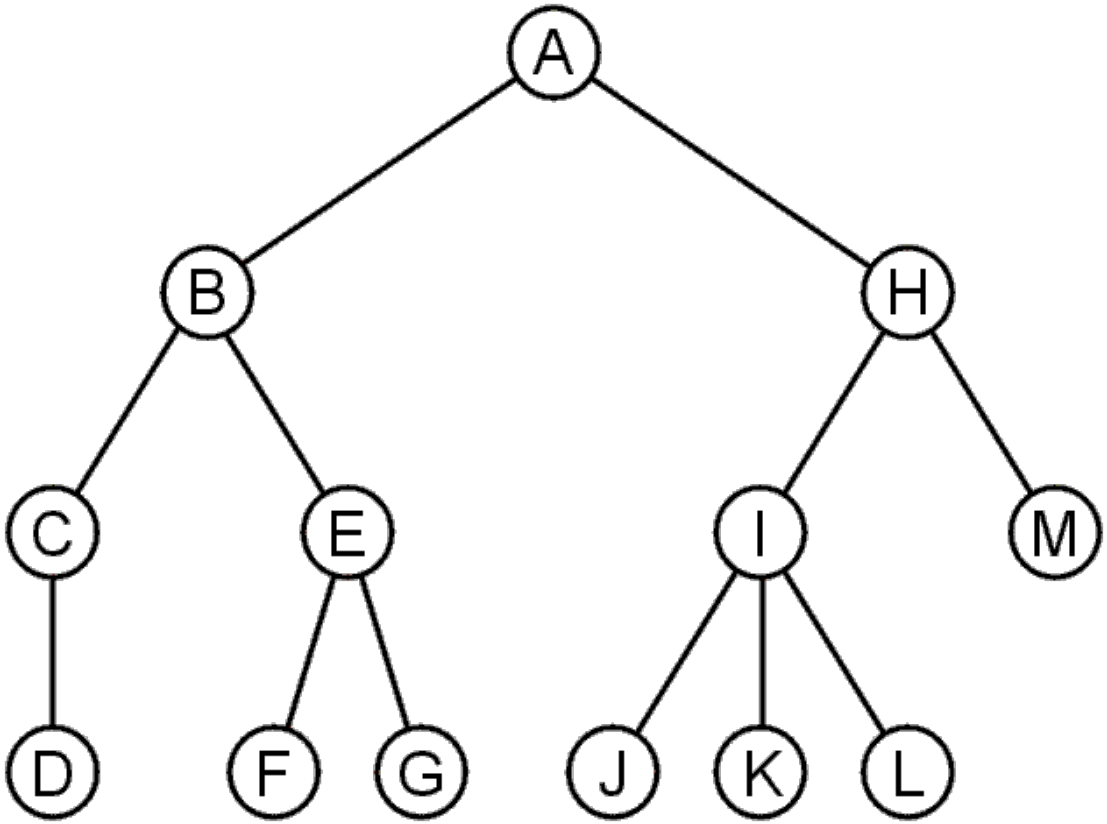
The height of this tree is 17



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

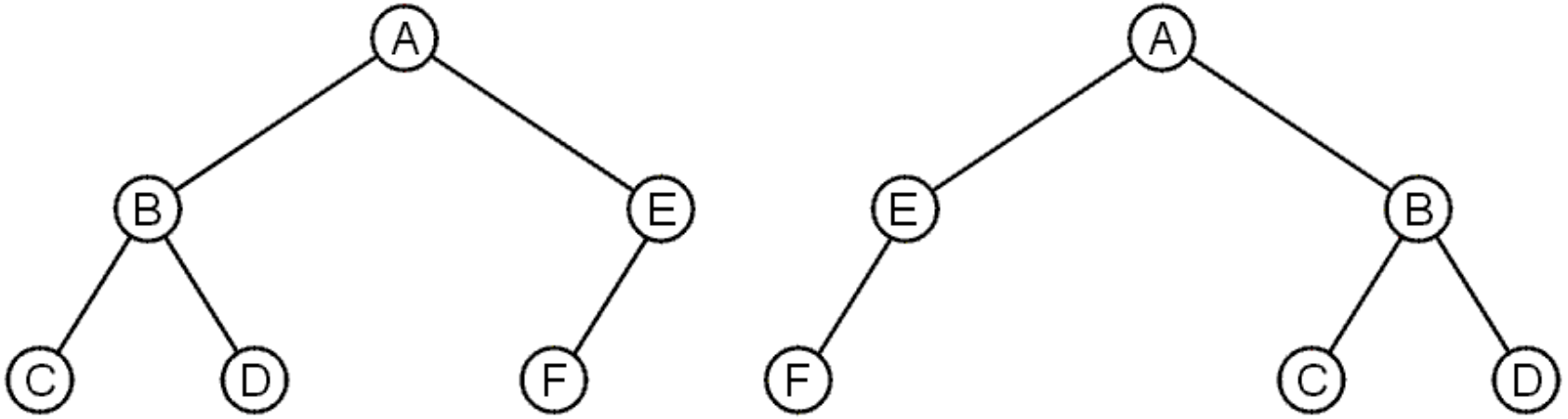
Trees

- Ancestors
- Descendents
- Subtree



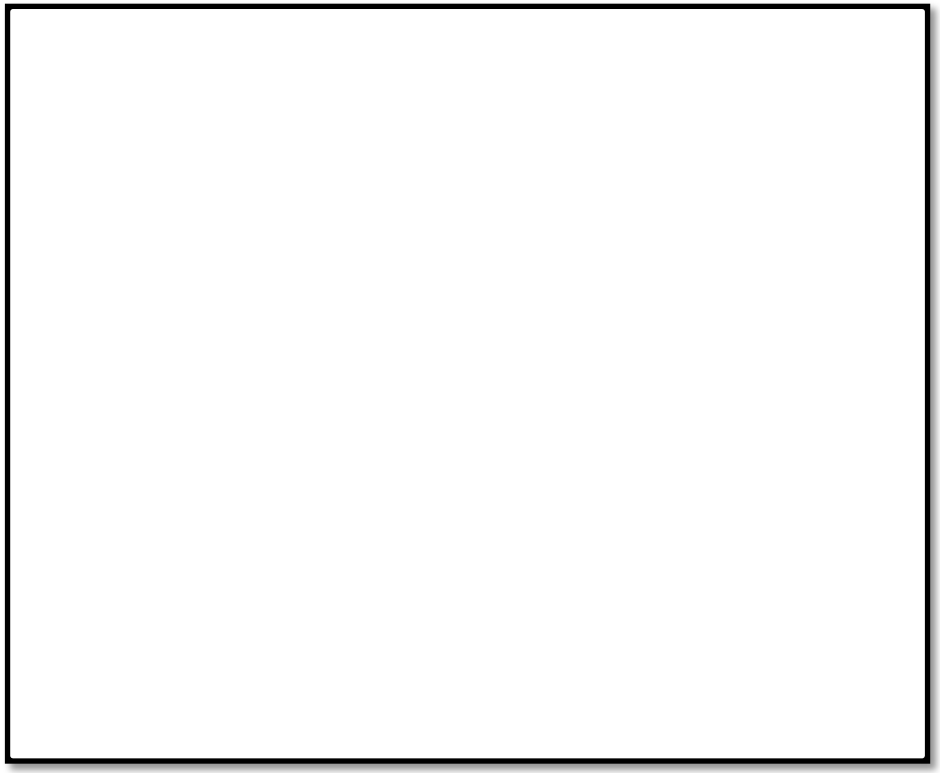
Trees

- Ordered vs Unordered Trees



Trees

- Associations are
  (denoted as pair {parent,child}):
- – {T, E}, {P, D}, {G, H}, {P, S}, {T, A}, {P, T}, {G, P}
- Draw the tree



DEFINING

Defining a Tree

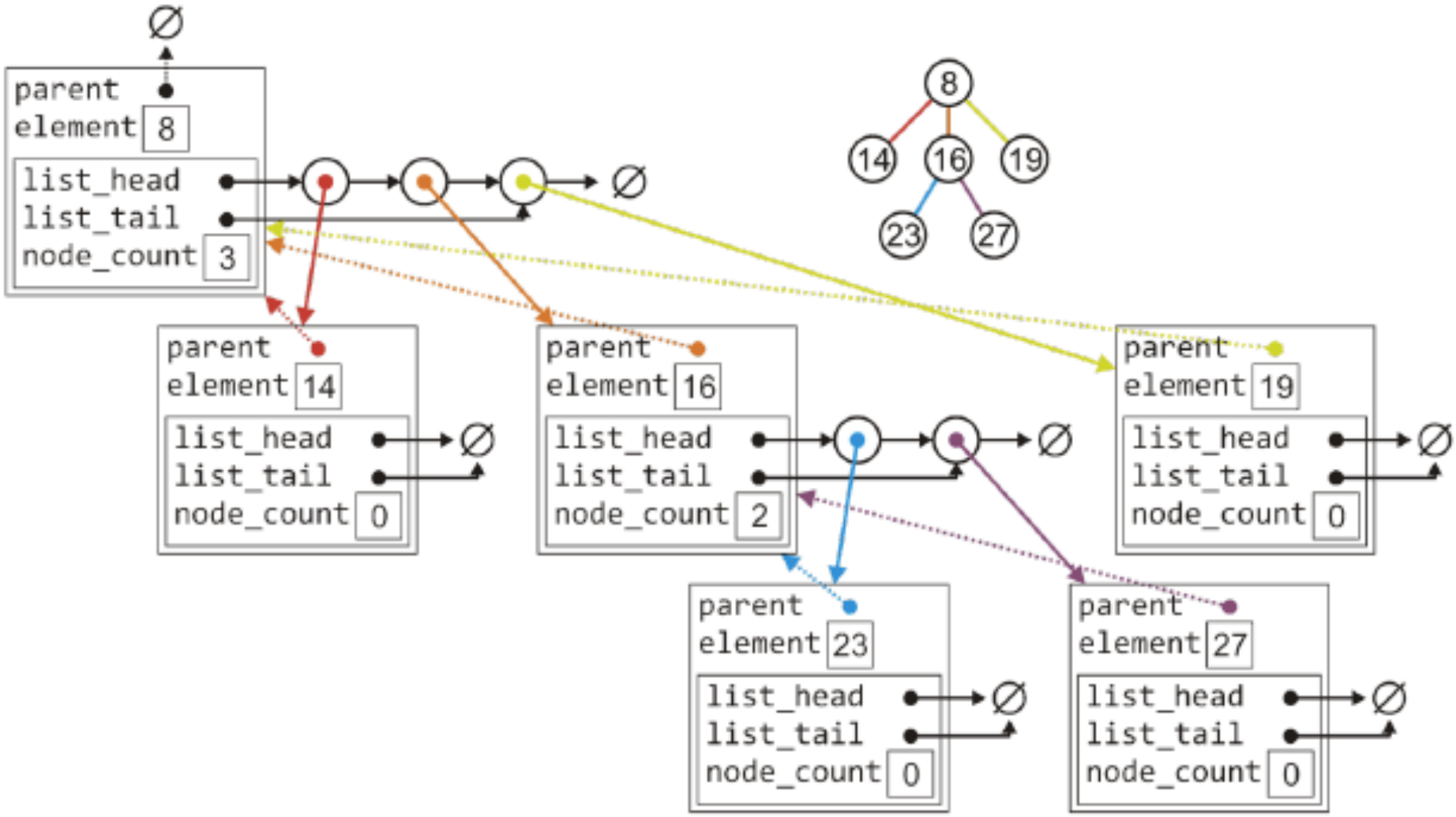
- A class Node to represent nodes.
- A class Tree that will own a collection of Node objects, and will have, among others, a pointer to the root node.

```
Type element;
Node<Type> * parent;
std::list<Node<T> *> children;
    // or std::vector – could store them as an array
```

Trees

```
template <typename Type>
class Node
{
    Type element;
    Node<Type> * parent;
    std::list<Node<T> *> children;
        // or std::vector – could store them as an array
  public:
    Node (Type & obj, Node<Type> * parent);
    Type retrieve();
    Node<Type> * parent();
    int degree();
    bool isRoot();
    bool isLeaf();
    Node<Type> * getChild (int n);
    int getHeight();
    void insert (Type & obj);
};
```

Trees


parent
element 8
list_head
list_tail
node_count 3
parent
element 14
list_head
list_tail
node_count 0
parent
element 16
list_head
list_tail
node_count 2
parent
element 19
list_head
list_tail
node_count 0
parent
element 23
list_head
list_tail
node_count 0
parent
element 27
list_head
list_tail
node_count 0
8
14
16
19
23
27

- isRoot()
  - if parent is NULL
- degree()
  - size of children list
- isLeaf()
  - degree() == 0

Trees

- insert(Type &obj)

Trees

- size() – Recursion! *(and iterators)*

- where's the end of the recursion? *(i.e., shouldn't a recursive call always be inside a conditional statement?)*

Iterators

```
#include <iostream>
#include <list>
using namespace std;

int main ()
{
  int myints[] = {75,23,65,42,13};
  list<int> mylist (myints, myints+5);

  cout << "mylist contains:";
  for (list<int>::iterator it = mylist.begin();
          it != mylist.end();
          ++it)
    cout << ' ' << *it;

  cout << '\n';

  return 0;
}
```

http://www.cplusplus.com/reference/list/list/begin/

Iterators

```
for (list<Node<Type> *>::iterator ch = d_children.begin();
        ch != d_children.end(); ++ch)
```

is same as:

```
for (list_iterator ch = d_children.begin();
    !ch.at_end(); ch.advance())
```

- The method `advance()` is called `operator++()` and it's invoked when using ++

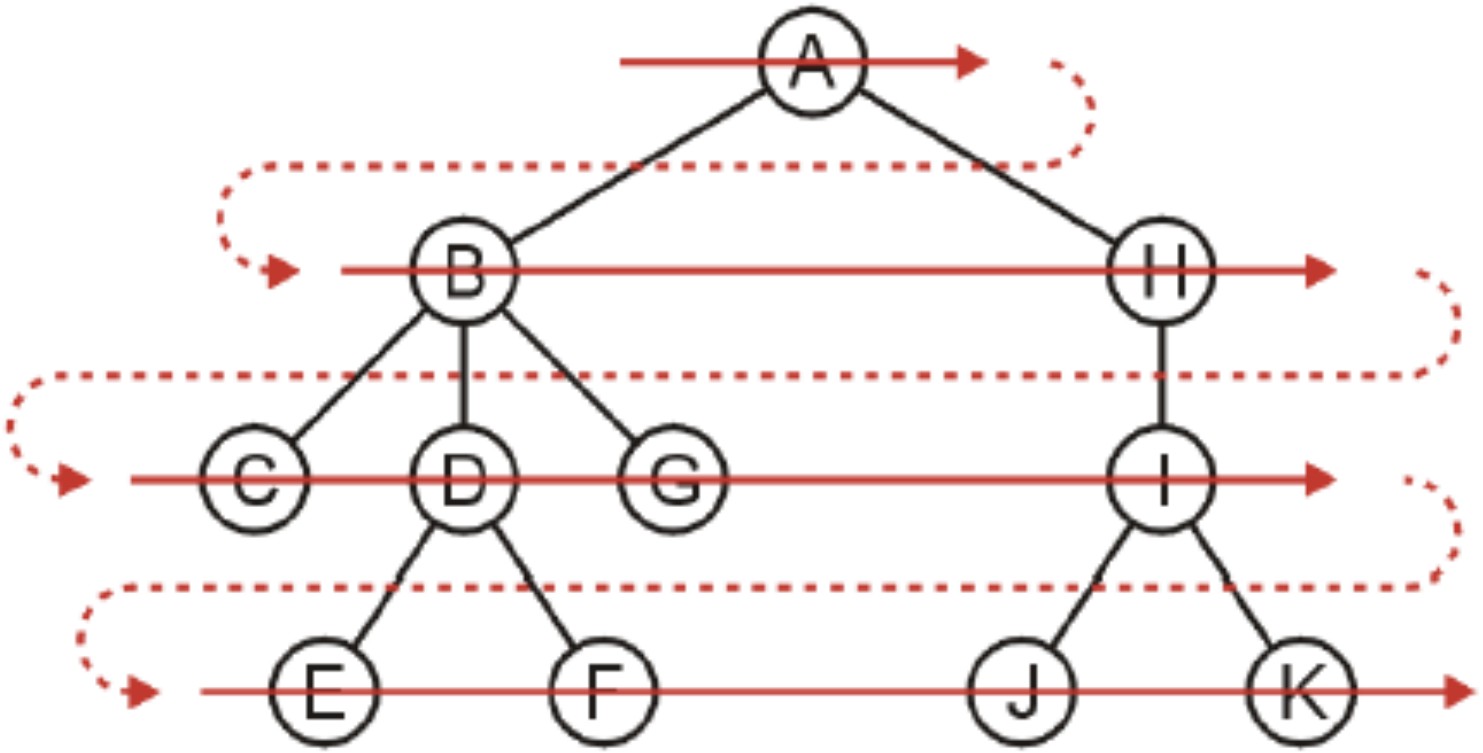
Trees

- getHeight()
  - iterate – get max height of all children

TRAVERSAL

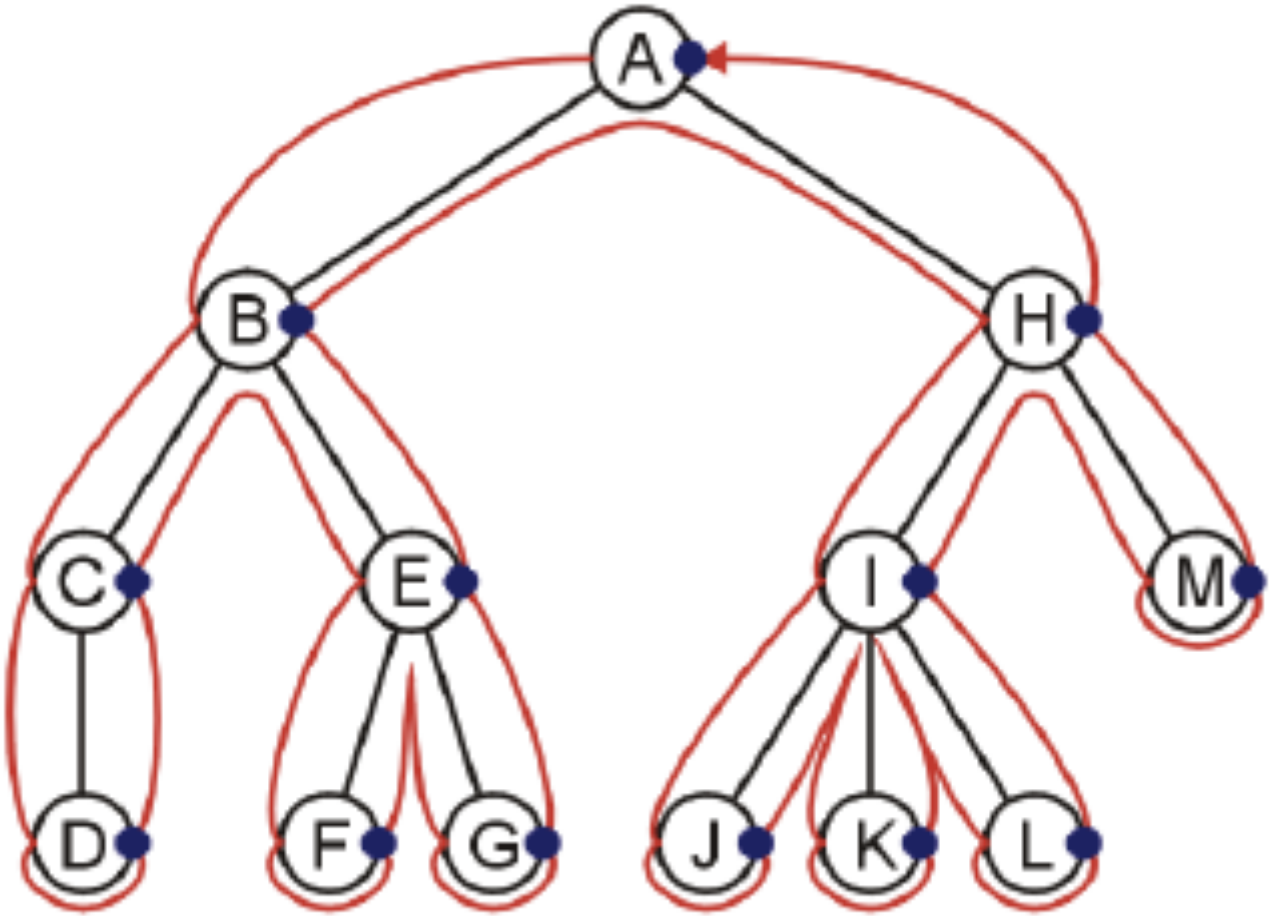
Breadth-First Traversal

- Explore at each depth thoroughly before moving to next level



Depth-First Traversal

- Each node is visited more than once



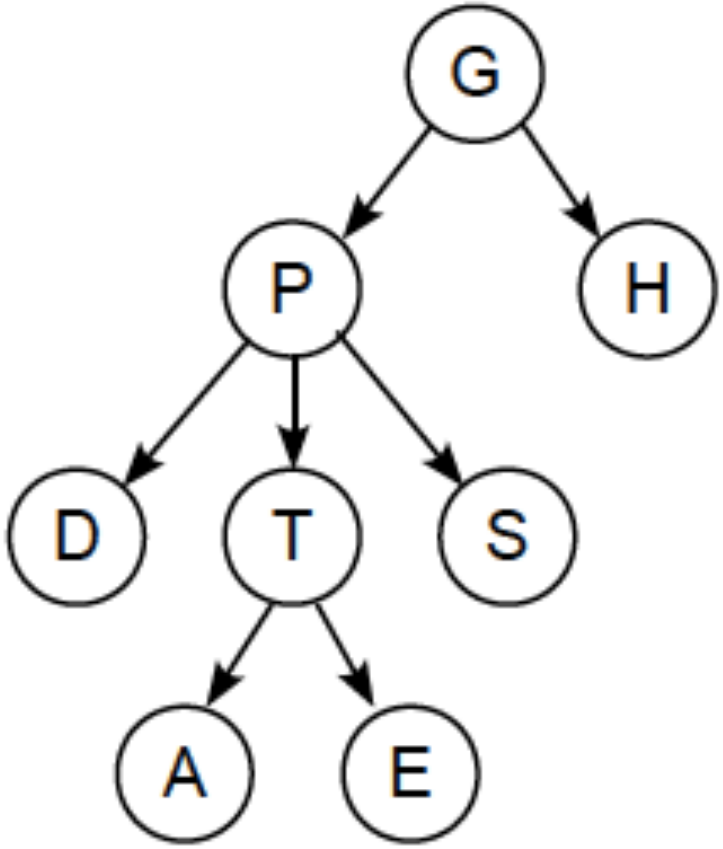
Traversals

- Breadth-first traversal can be implemented with a queue (a FIFO data structure)
- Depth-first requires a stack (LIFO)
  - Huh?? Didn't we already see some depth-first examples? We didn't use a stack — did we??

Breadth-First Traversal

Queue

- Enqueue the root node (or ptr to)
  - While the queue is not empty:
    - Dequeue an element
    - Enqueue all of the children of the just dequeued node



Queue:

Depth-First Traversal

- Each node visited more than once
- What if we require some processing for the present node?
  When would we do it?
  - On our way down, or
  - On our way back?

  .... Depends...

Depth-First Traversal

- Pre-order
  - Process current node
  - Process children
- Post-order
  - Process children
  - Process current node
- In-order

Depth-First Traversal

Example:

- Tree containing a directory structure in memory
  - Each node is a directory
  - Containing a list of files as the node's data
  - List of subdirectories being the child nodes

1. Print hierarchy of directories
2. Compute the total disk space used by all the files in the directory and all subdirectories below

EXAMPLE

Example: XHTML and CSS

The XML of XHTML has a tree structure

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

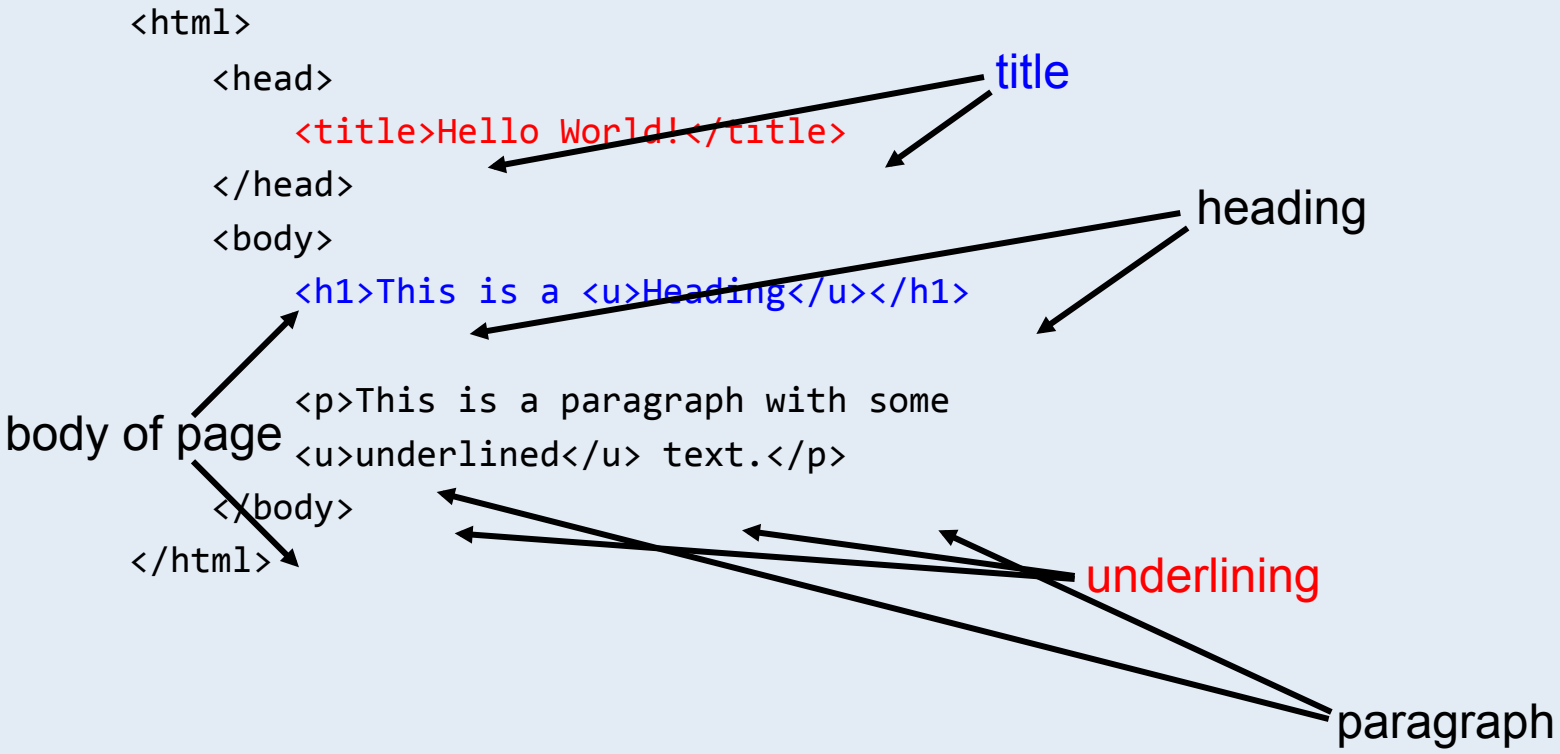
Example: XHTML and CSS

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 and CSS

Consider the following XHTML document

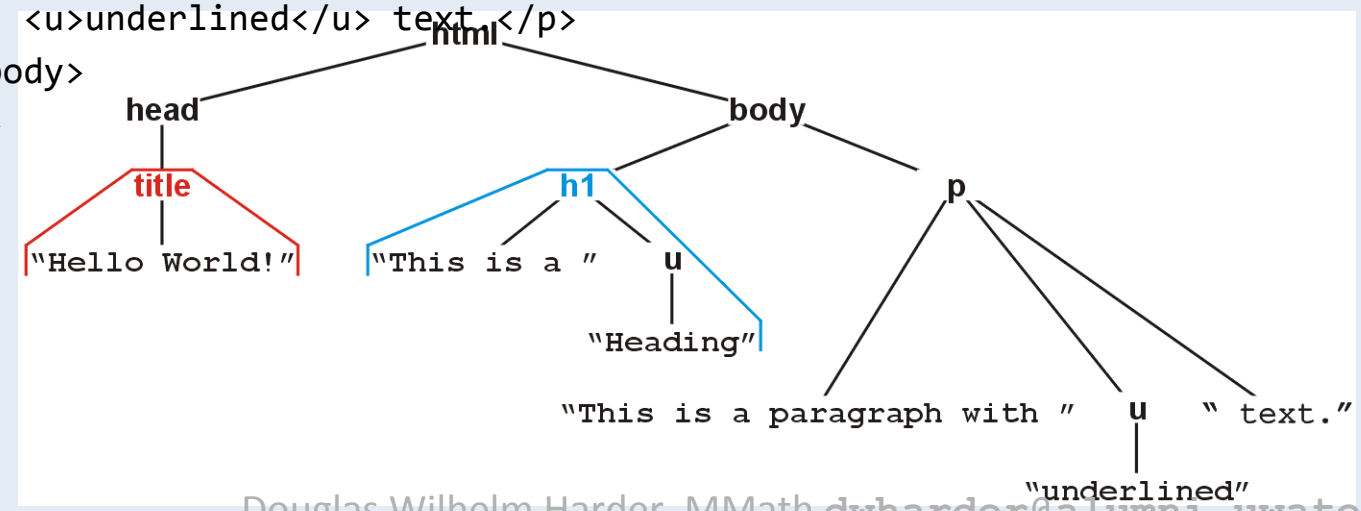



```
<html>
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <h1>This is a <u>Heading</u></h1>

        <p>This is a paragraph with some
        <u>underlined</u> text.</p>
    </body>
</html>
```

Example: XHTML and CSS

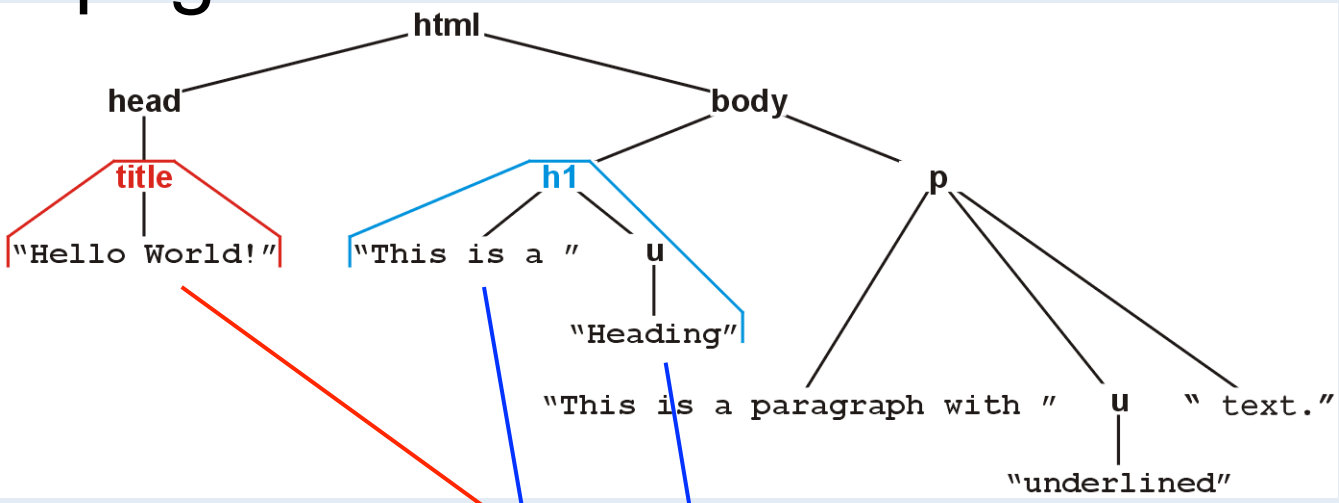
The nested tags define a tree rooted at the HTML tag

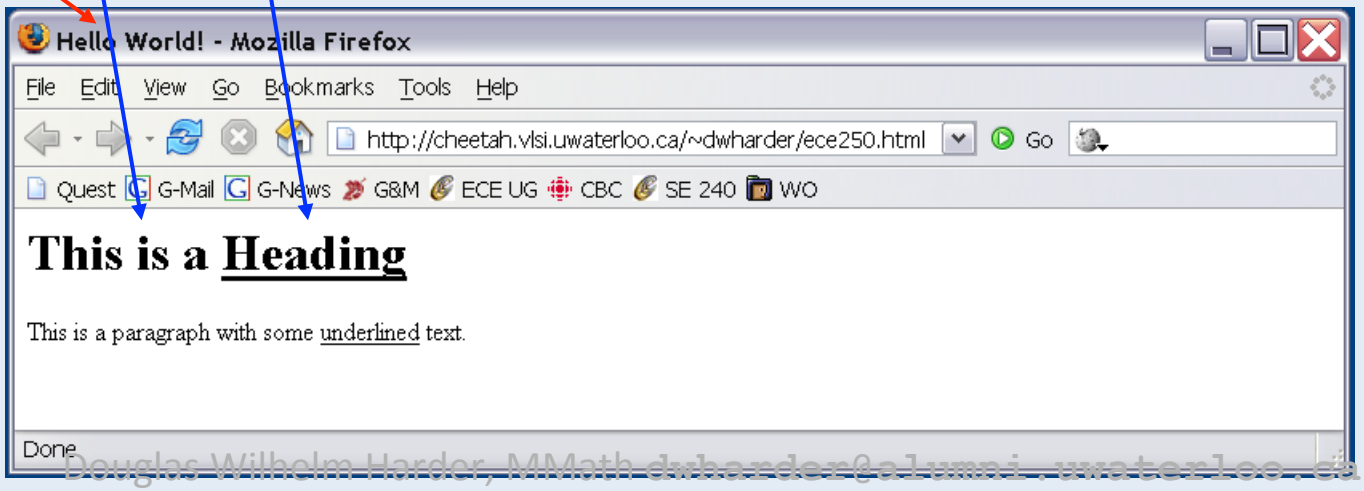
```
<html>
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <h1>This is a <u>Heading</u></h1>

        <p>This is a paragraph with some
        <u>underlined</u> text.</p>
    </body>
</html>
```



Example: XHTML and CSS

Web browsers render this tree as a web page





Example: XHTML and CSS

XML tags `<tag>...</tag>` must be nested

For example, to get the following effect:

1 2 3 **4 5 6** **7 8 9**

you may use

```
<u>1 2 3 <b>4 5 6</b></u><b> 7 8 9</b>
```

You may not use:

```
<u>1 2 3 <b>4 5 6</u> 7 8 9</b>
```

Example: XHTML and CSS

Cascading Style Sheets (CSS) make use of this tree structure to describe how HTML should be displayed

- For example:

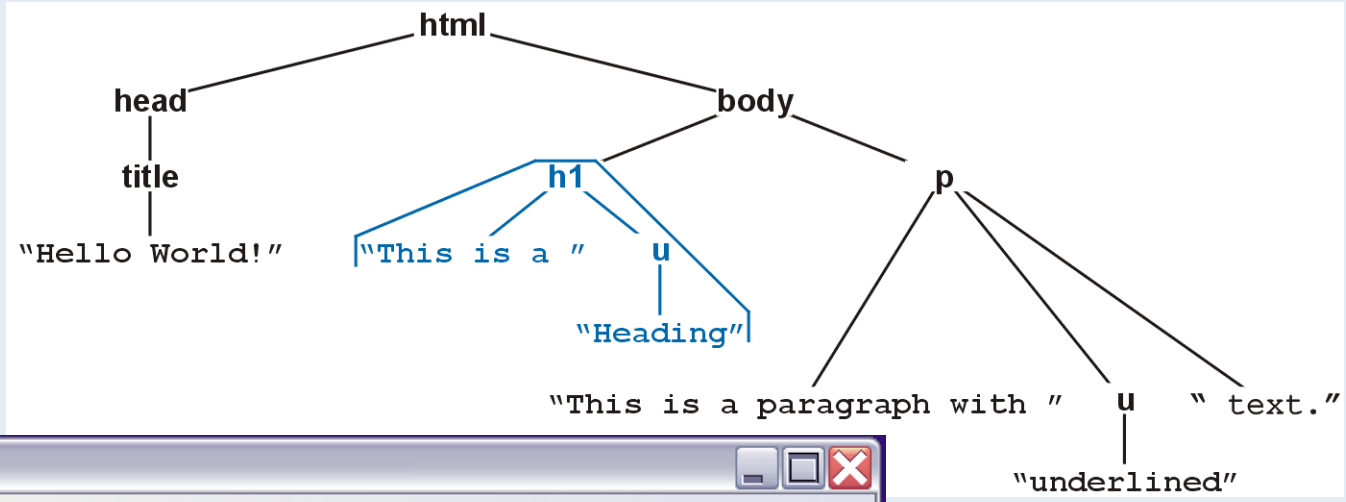
```
<style type="text/css">
    h1 { color:blue; }
</style>
```

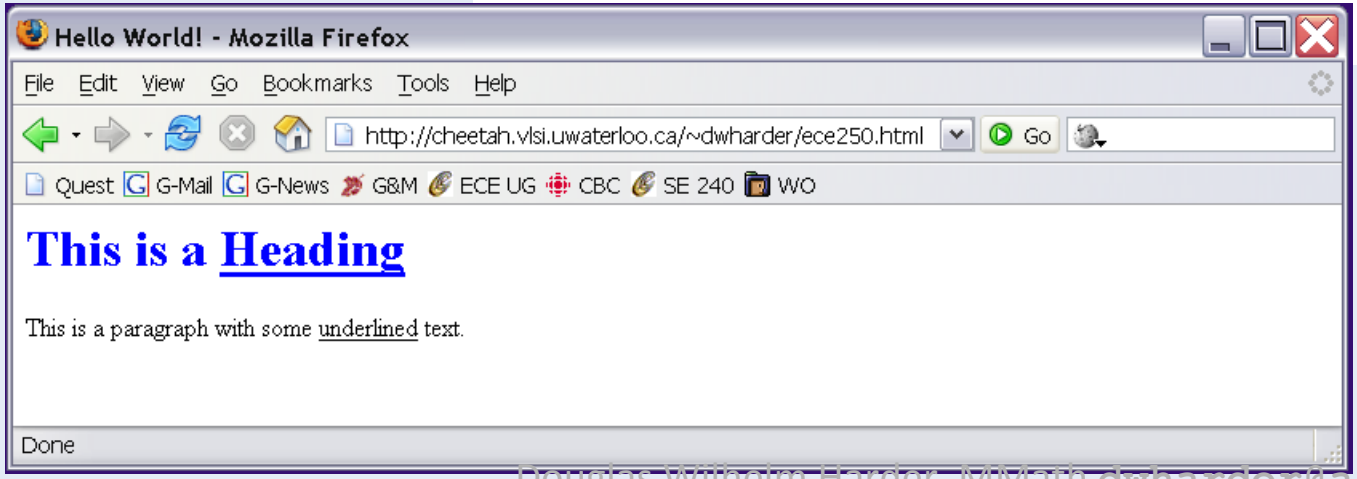
indicates all text/decorations descendant from an `h1` header should be blue

Example: XHTML and CSS

For example, this style renders as follows:

```
<style type="text/css">
   h1 { color:blue; }
</style>
```

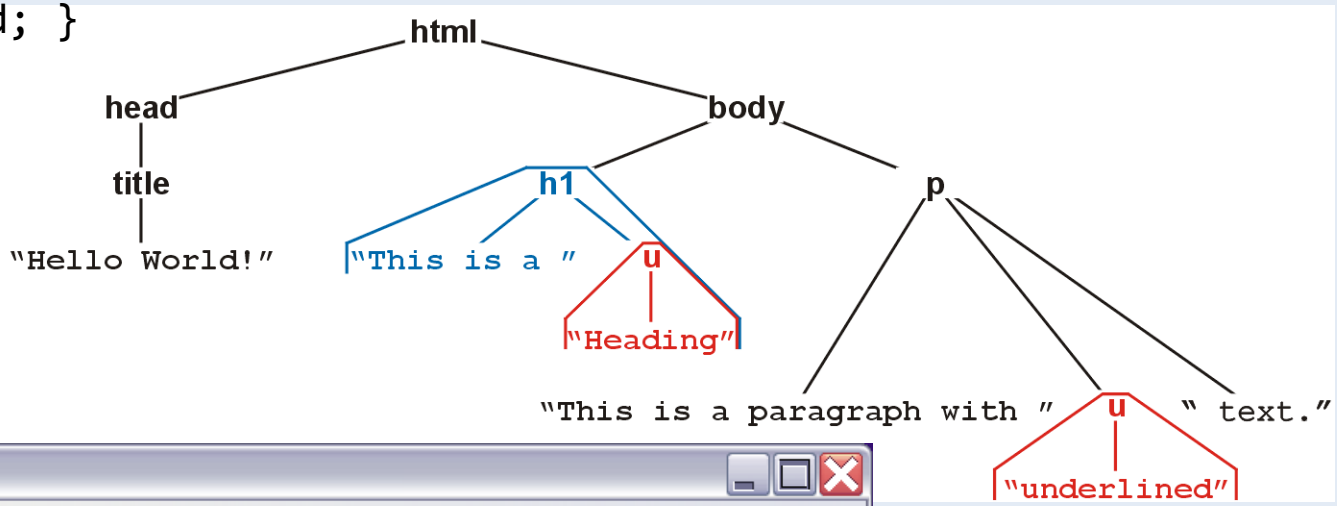


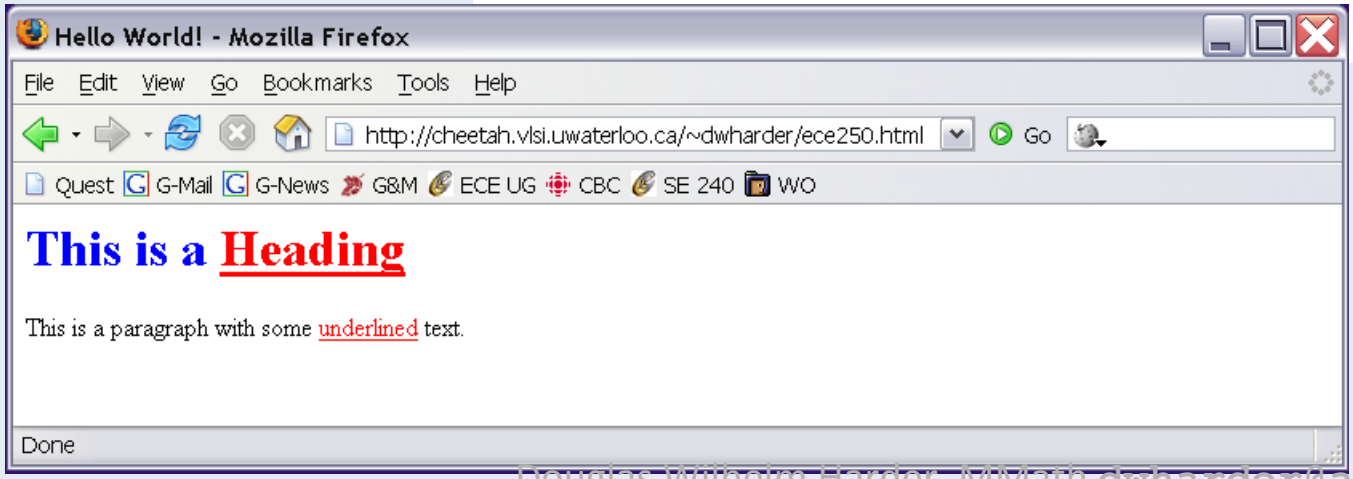


Example: XHTML and CSS

For example, this style renders as follows:

```
<style type="text/css">
   h1 { color:blue; }
   u { color:red; }
</style>
```





Example: XHTML and CSS

Suppose you don't want underlined items in headers (h1) to be red

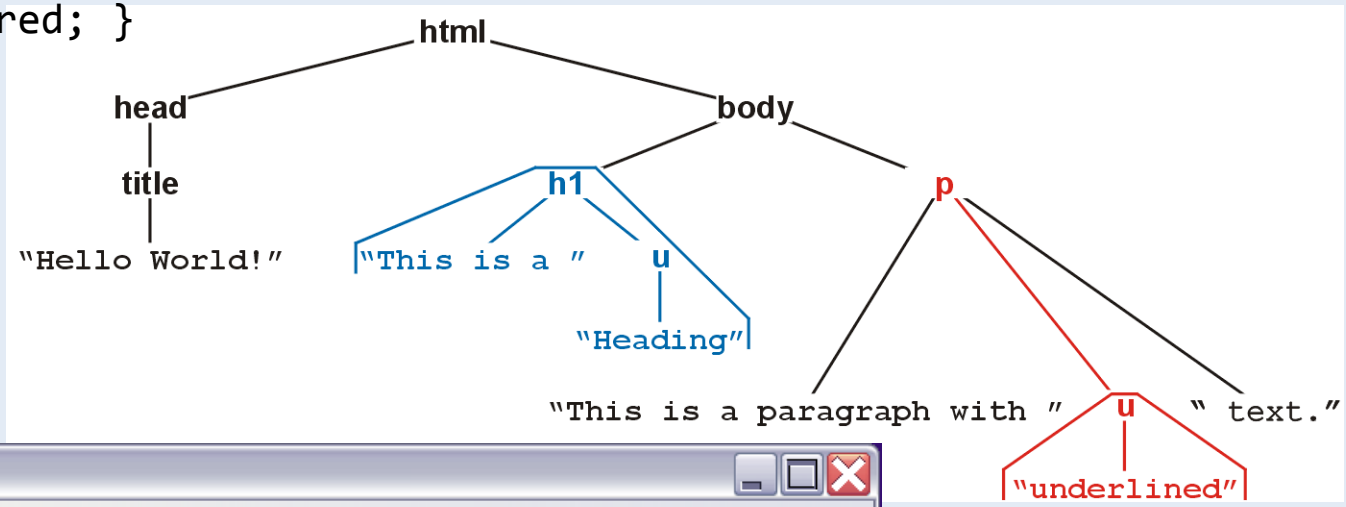
- More specifically, suppose you want any underlined text within paragraphs to be red

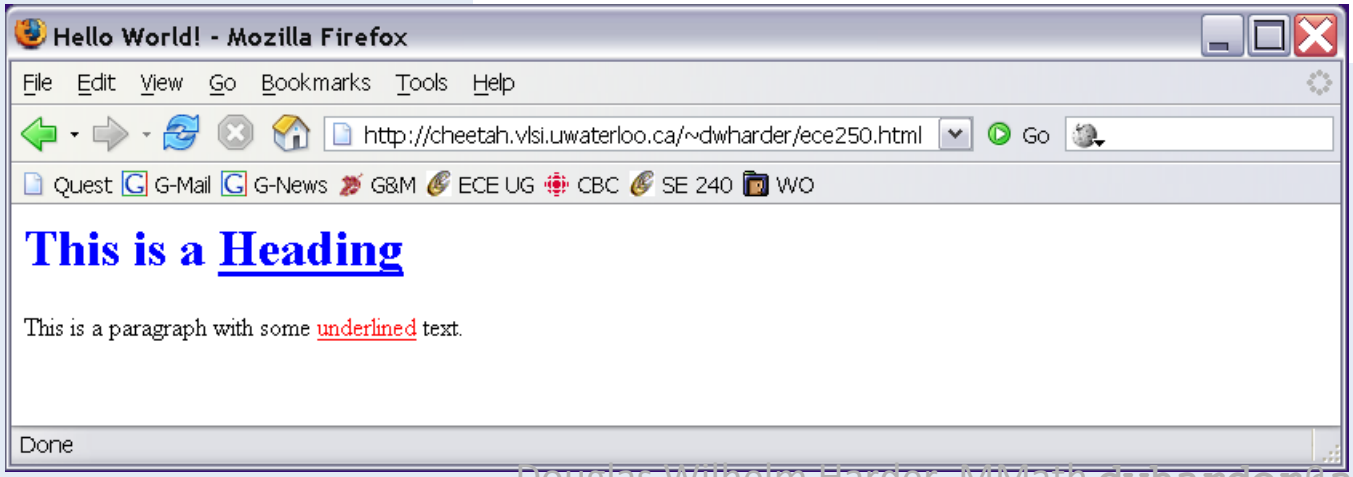
That is, you only want text marked as `<u>text</u>` to be underlined if it is a descendant of a `<p>` tag

Example: XHTML and CSS

For example, this style renders as follows:

```
<style type="text/css">
   h1 { color:blue; }
   p u { color:red; }
</style>
```





Example: XHTML and CSS

You can read the second style

```
<style type="text/css">
   h1 { color:blue; }
   p u { color:red; }
</style>
```

as saying “text/decorations descendant from the underlining tag (<u>) which itself is a descendant of a paragraph tag should be coloured red”

Example: XML

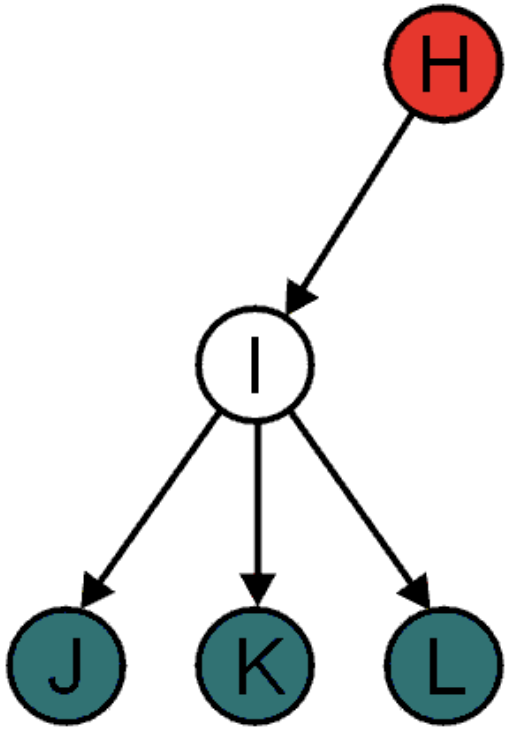
In general, any XML can be represented as a tree

- All XML tools make use of this feature
- Parsers convert XML into an internal tree structure
- XML transformation languages manipulate the tree structure
  - *E.g.*, XMLT

QUESTIONS TO PONDER

Questions

1. Given the following diagram, identify
   a) Any child nodes
   b) Any sibling nodes
   c) Any root nodes
   d) Any parent nodes
   e) Any leaf nodes
   f) The degree of node I



Questions

1. What are nodes called that have a degree of zero?
2. A tree with n nodes has how many edges?
3. Is it possible to have 2 paths to get to a node in a tree?
4. When recursively traversing a tree, does it usually end up traversing *depth-first* or *breadth-first?*

Questions

- List the depths in the diagram.
- What do the numbers in the nodes represent?