COMP251: DATA STRUCTURES & ALGORITHMS

* Some slides from “Algorithms and Data Structures” by Douglas Wilhelm Harder

Introduction to Trees

Outline

In this topic, we will cover:

- Definition of a tree data structure and its components
- Concepts of:
  - Root, internal, and leaf nodes
  - Parents, children, and siblings
  - Paths, path length, height, and depth
  - Ancestors and descendants
  - Ordered and unordered trees
  - Subtrees
- Examples
  - XHTML and CSS

Linear Structures

- We use linear structures in many applications.
- Linux:
  - processes stored in *Linked List*
  - FIFO scheduler schedules jobs using *queue*
  - function calls push memory onto *stack*

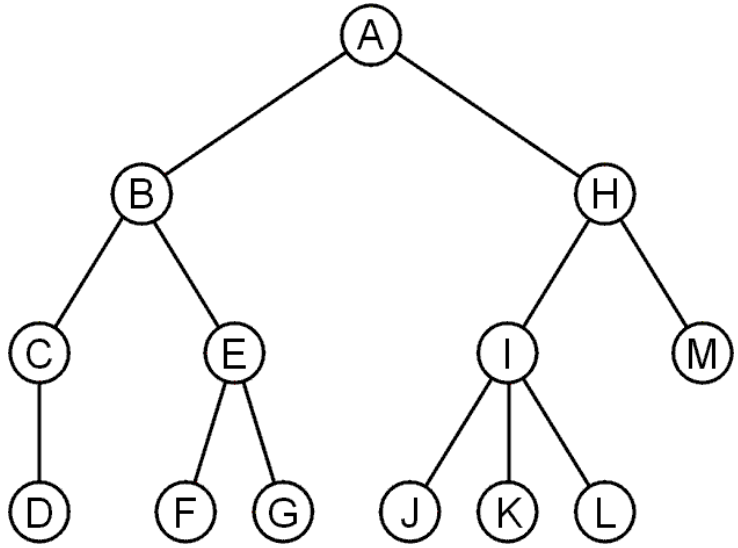
Drawbacks of Lists

- So far, the ADT's we examined have been linear
- $O(n)$ for simple operations
- Can we do better?
  - Recall binary search: O(log(n)) for find :-)
  - But list must be sorted. O(n log(n)) to sort :-(

Trees

A rooted tree data structure stores information in *nodes*

- Extension of linked lists:
  - Each node connects to multiple nodes
- Definition:
  - 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



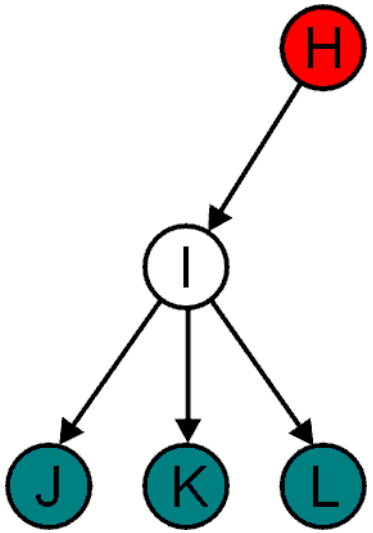
Terminology

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

– I has three children: J, K and L

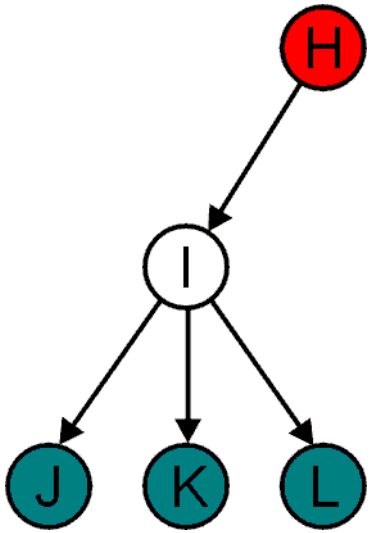
For all nodes other than the root node, there is one ***parent*** node

– H is the parent of I



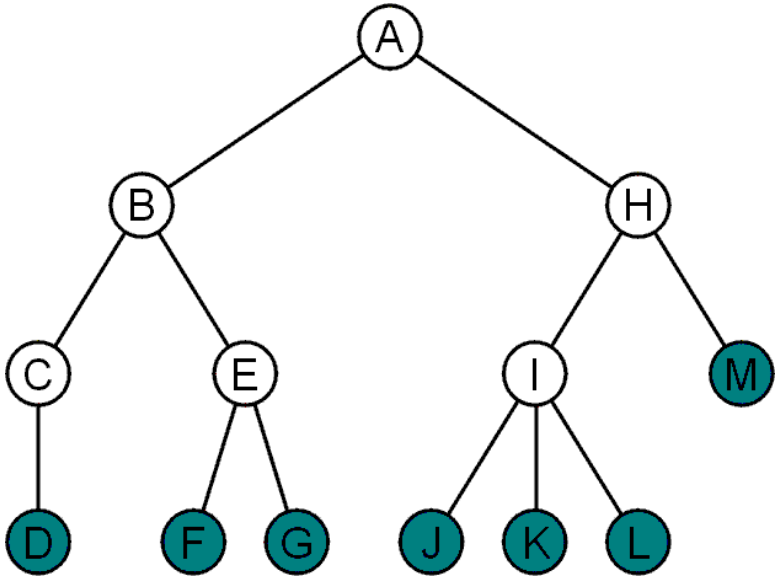
Terminology

- The ***degree*** of a node is defined as the number of its *children*: $\deg(\mathrm{I}) = 3$
- Nodes with the same parent are ***siblings***
  - J, K, and L are siblings



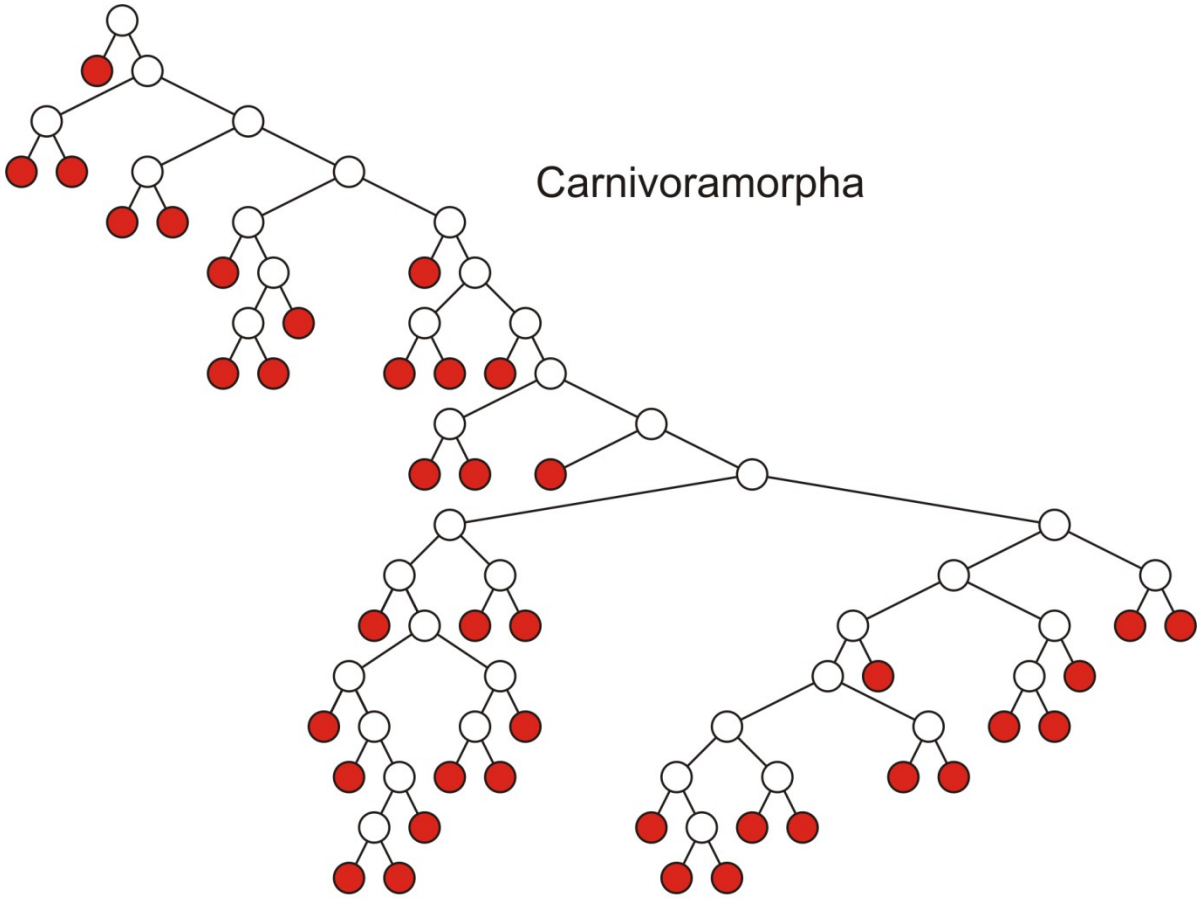
Terminology

- 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



Terminology

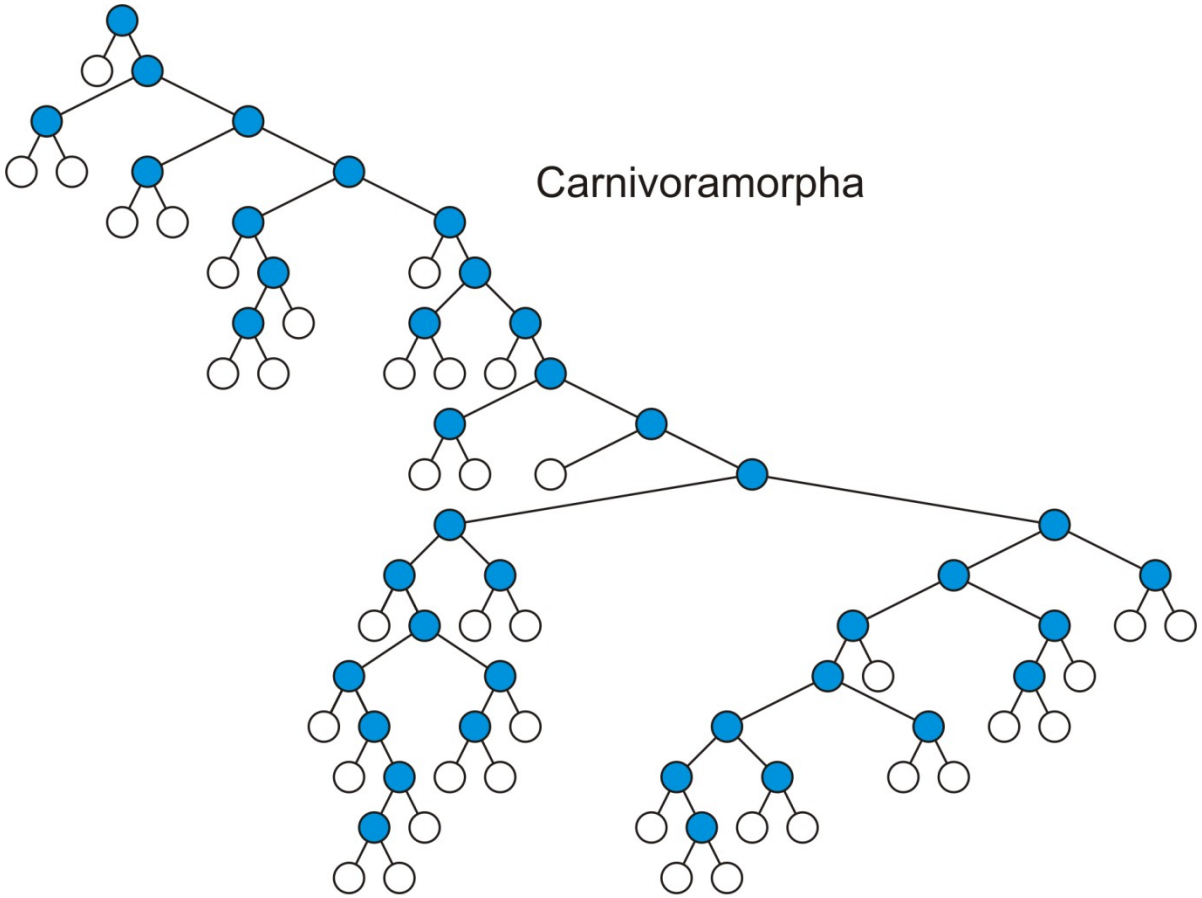
Leaf nodes:



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

Terminology

Internal nodes:

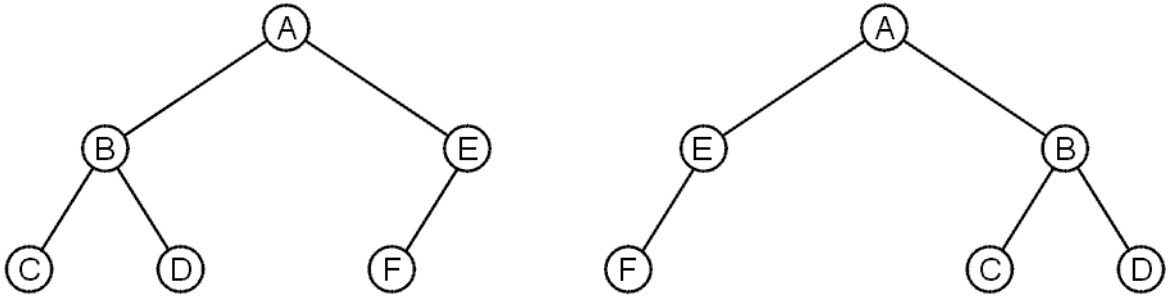


Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Terminology

These trees are equal if the order of the children is ignored

– ***unordered trees***



They are different if order is relevant (***ordered trees***)

– We will usually examine ordered trees (linear orders)

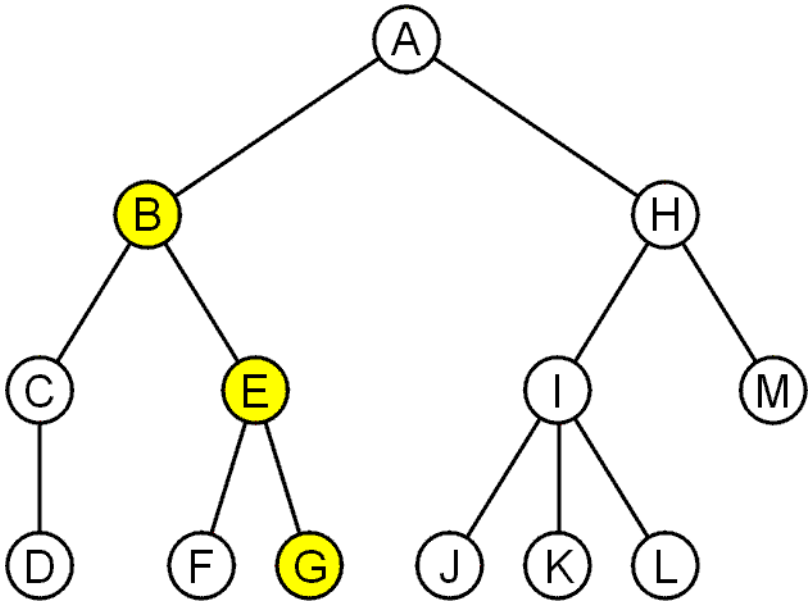
Terminology

- A path is a sequence of nodes
  $(a_0, a_1, ..., a_n)$

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

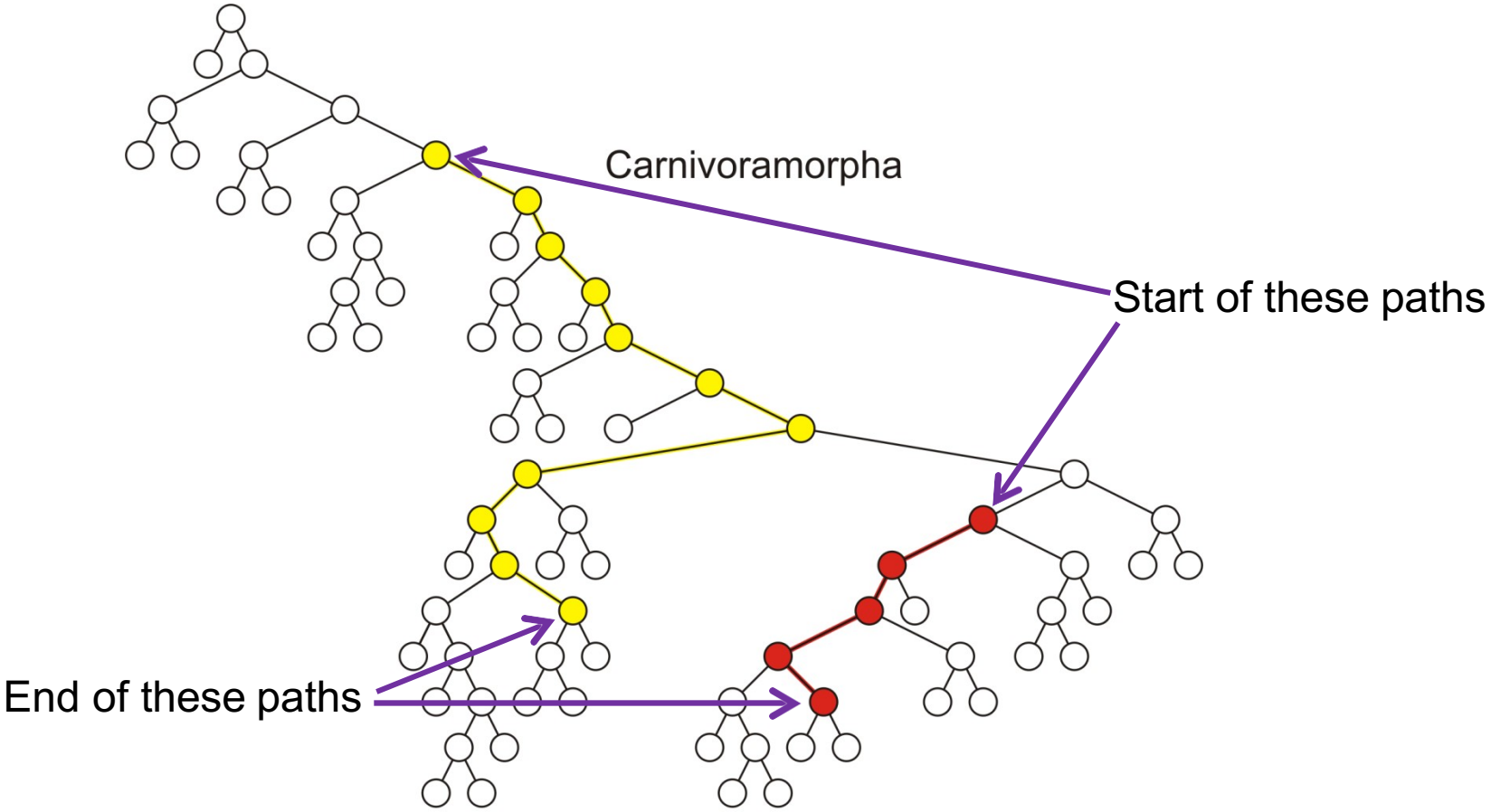
- The length of this path is n

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



Terminology

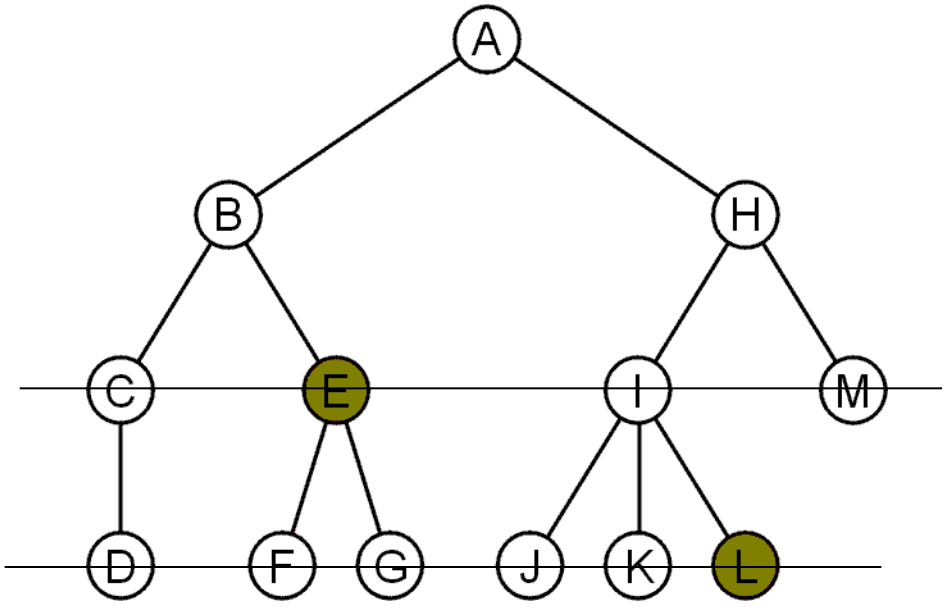
Paths of length 10 (11 nodes) and 4 (5 nodes)



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

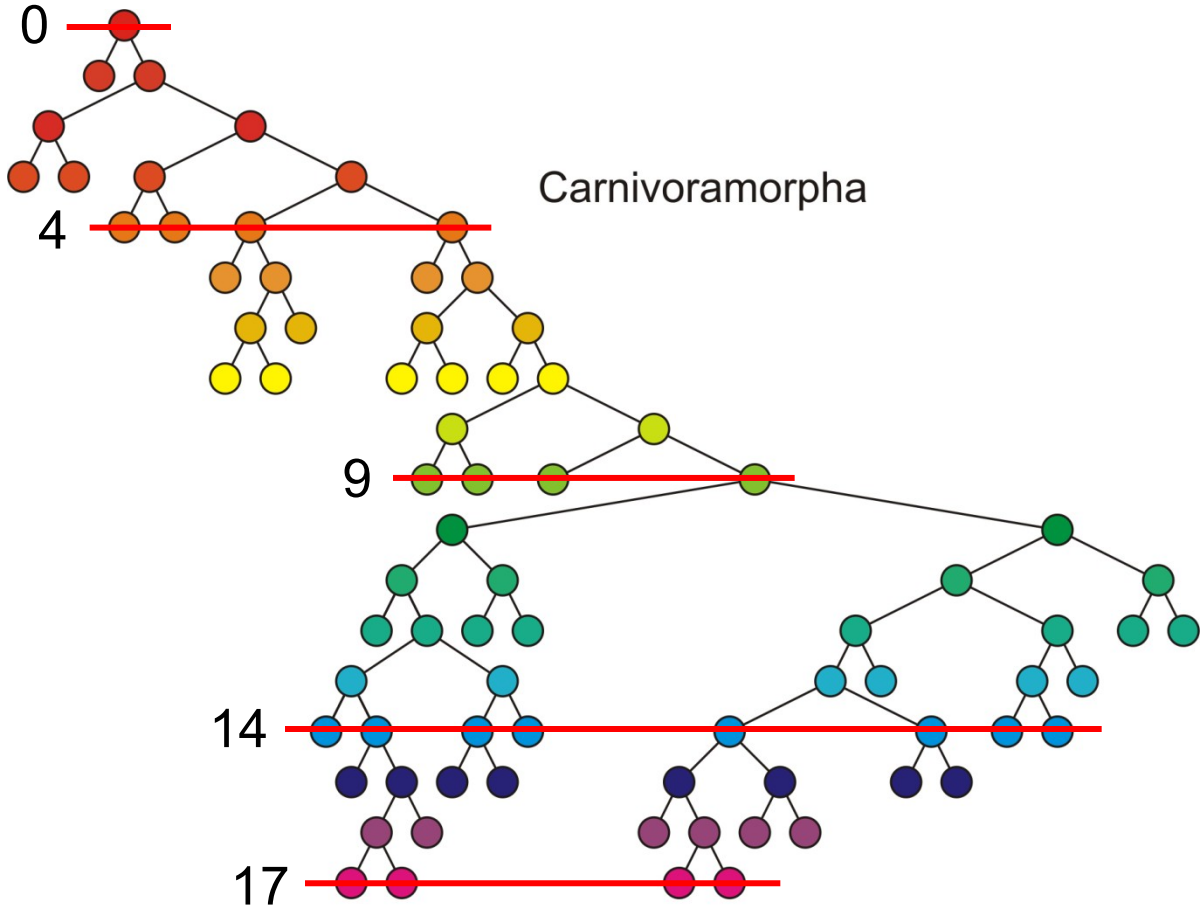
Terminology

- 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
  - –A has depth 0



Terminology

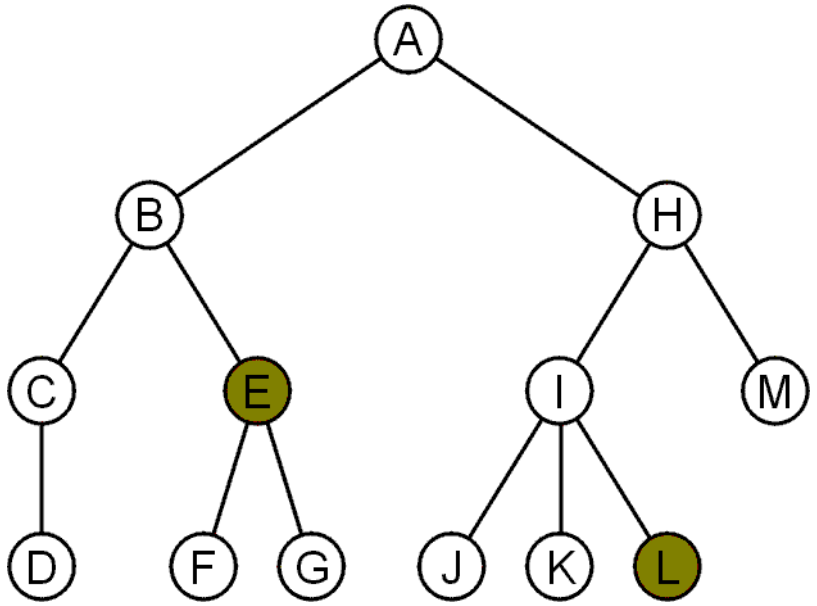
Nodes of depth up to 17



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

Terminology

- The ***height*** of a node v is the length of longest path from v to a leaf
- The height of a leaf node is 0
  - –E has height 1
  - –L has height 0
  - –A has height 3

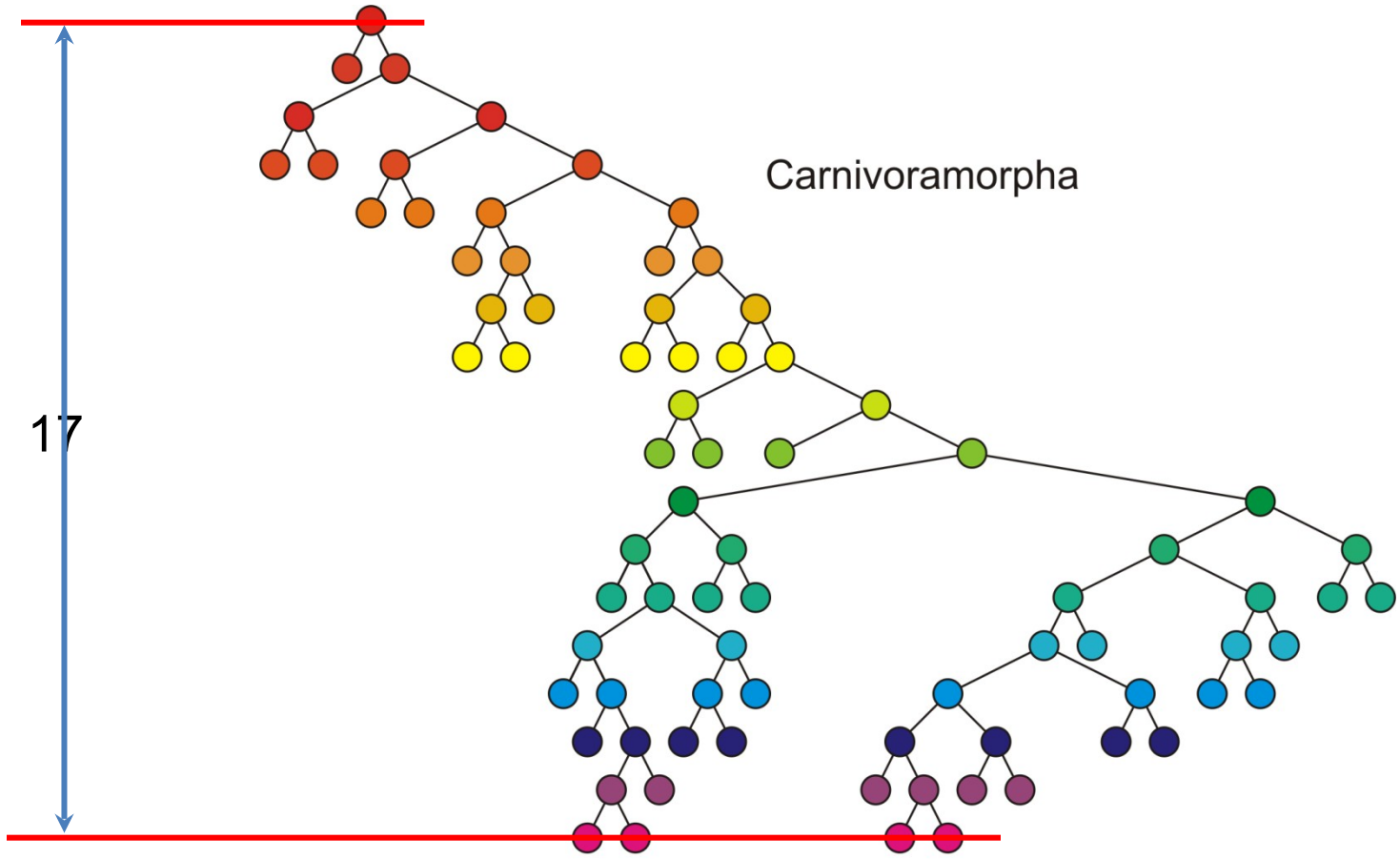


Terminology

- The ***height*** of a tree is defined as the height of the root, which is the length of the longest path from the root to 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 (there is no node) to be -1

Terminology

The height of this tree is 17



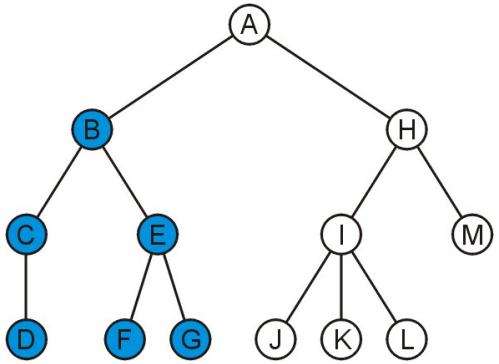
Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Terminology

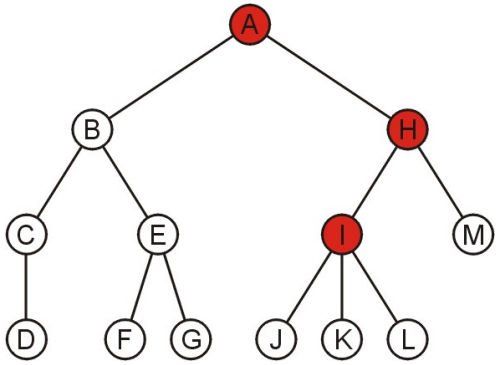
- If in a tree there is a path from node *a* to node *b*:
  - *a* is an ***ancestor*** of *b*
  - *b* is a ***descendent*** of *a*
- Thus, a node is both an ancestor and a descendant of itself
  - We can add the adjective strict to exclude equality: *a* is a strict descendent of *b* if *a* is a descendant of *b* but $a \neq b$
- The root node is an ancestor of all nodes

Terminology

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

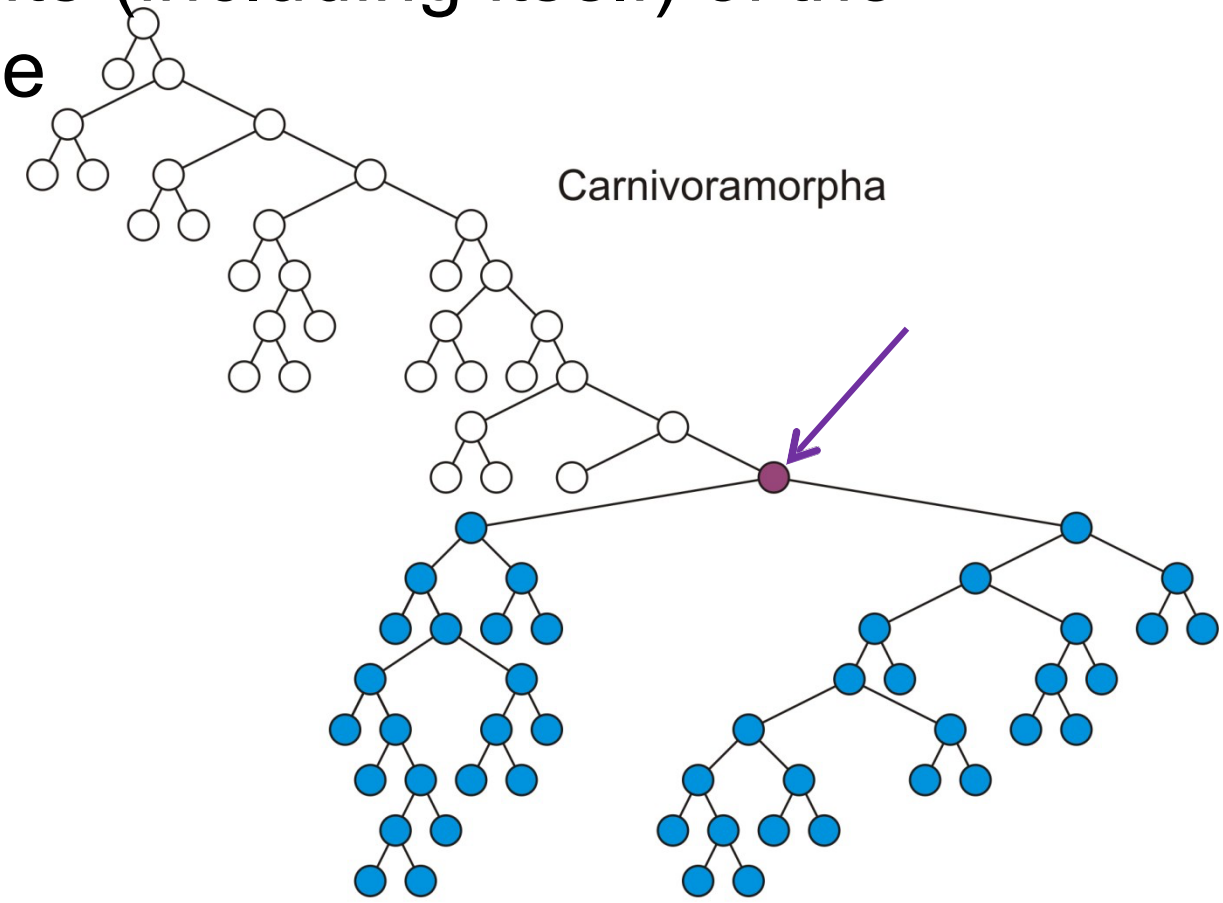


The ancestors of node I are I, H, and A:



Terminology

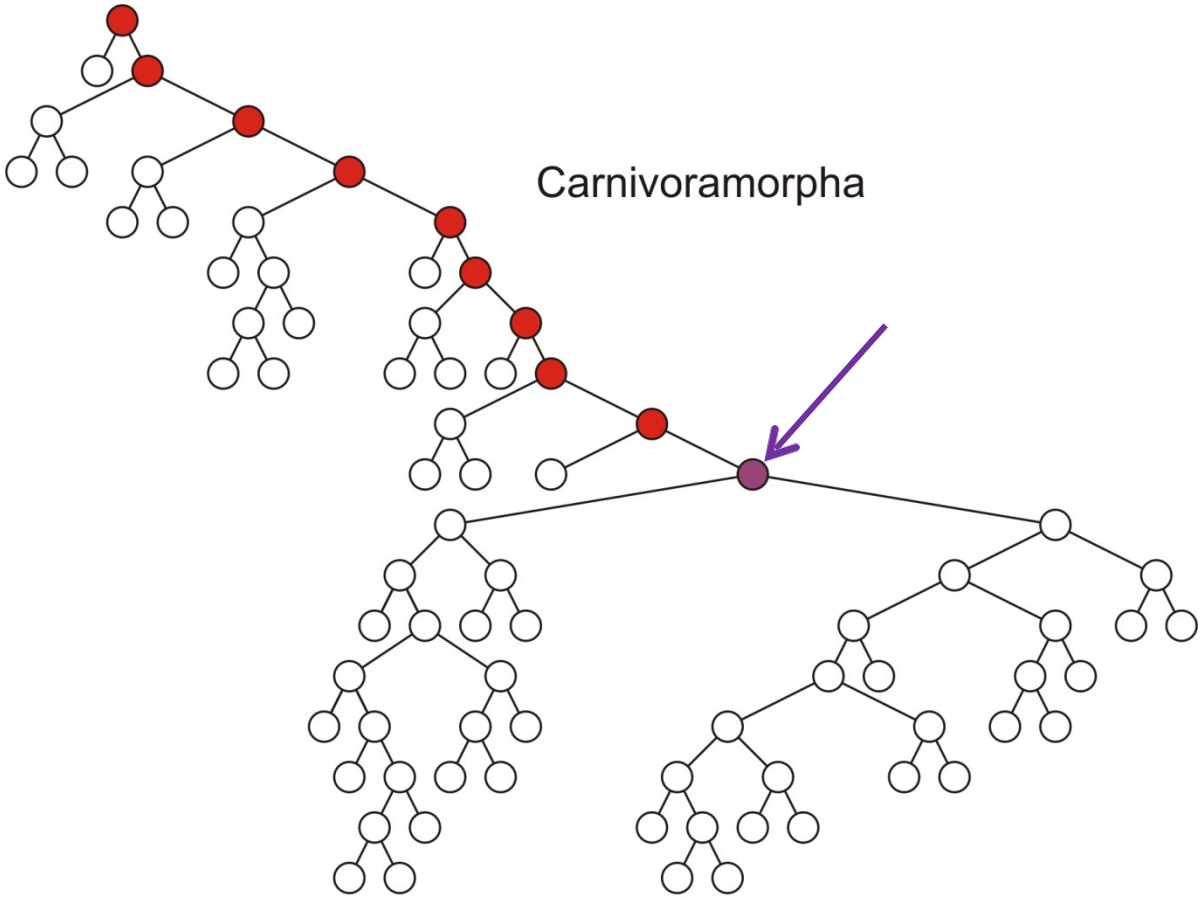
All descendants (including itself) of the indicated node



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

Terminology

All ancestors (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’

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

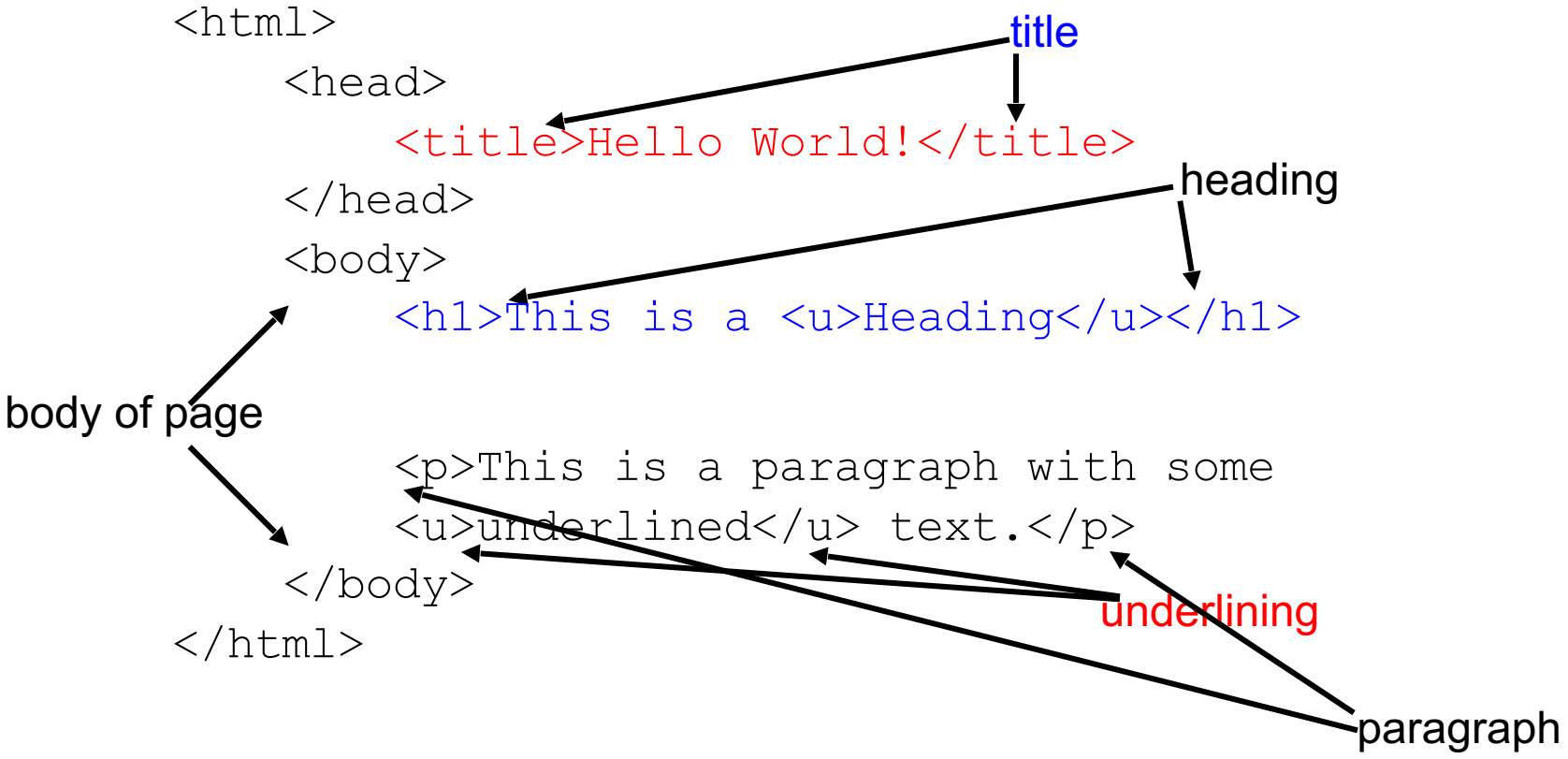
Example: XHTML and CSS

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

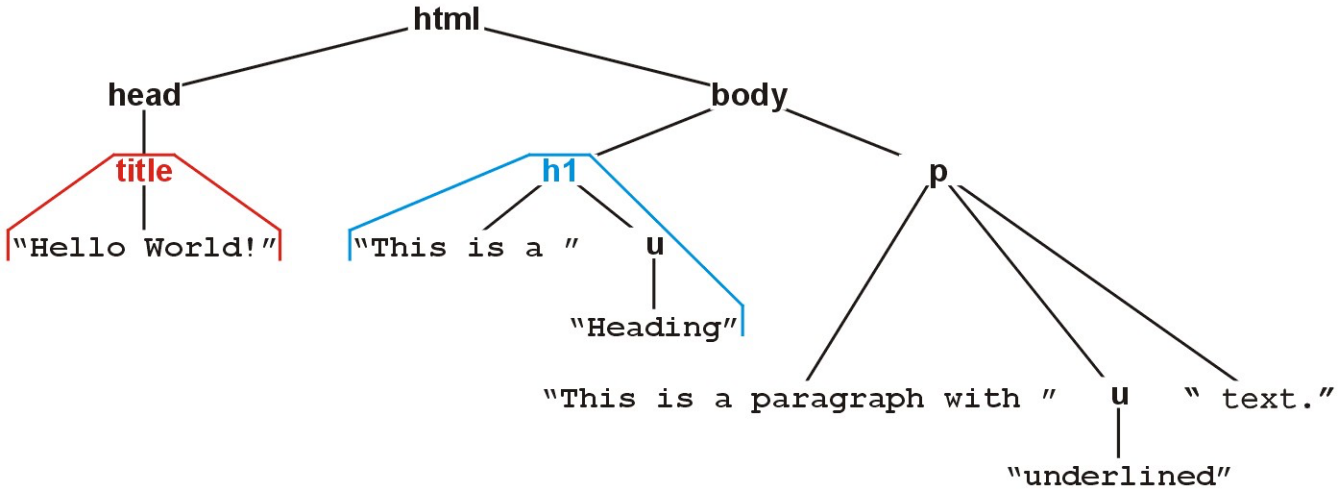
Consider the following XHTML document



Example: XHTML and CSS

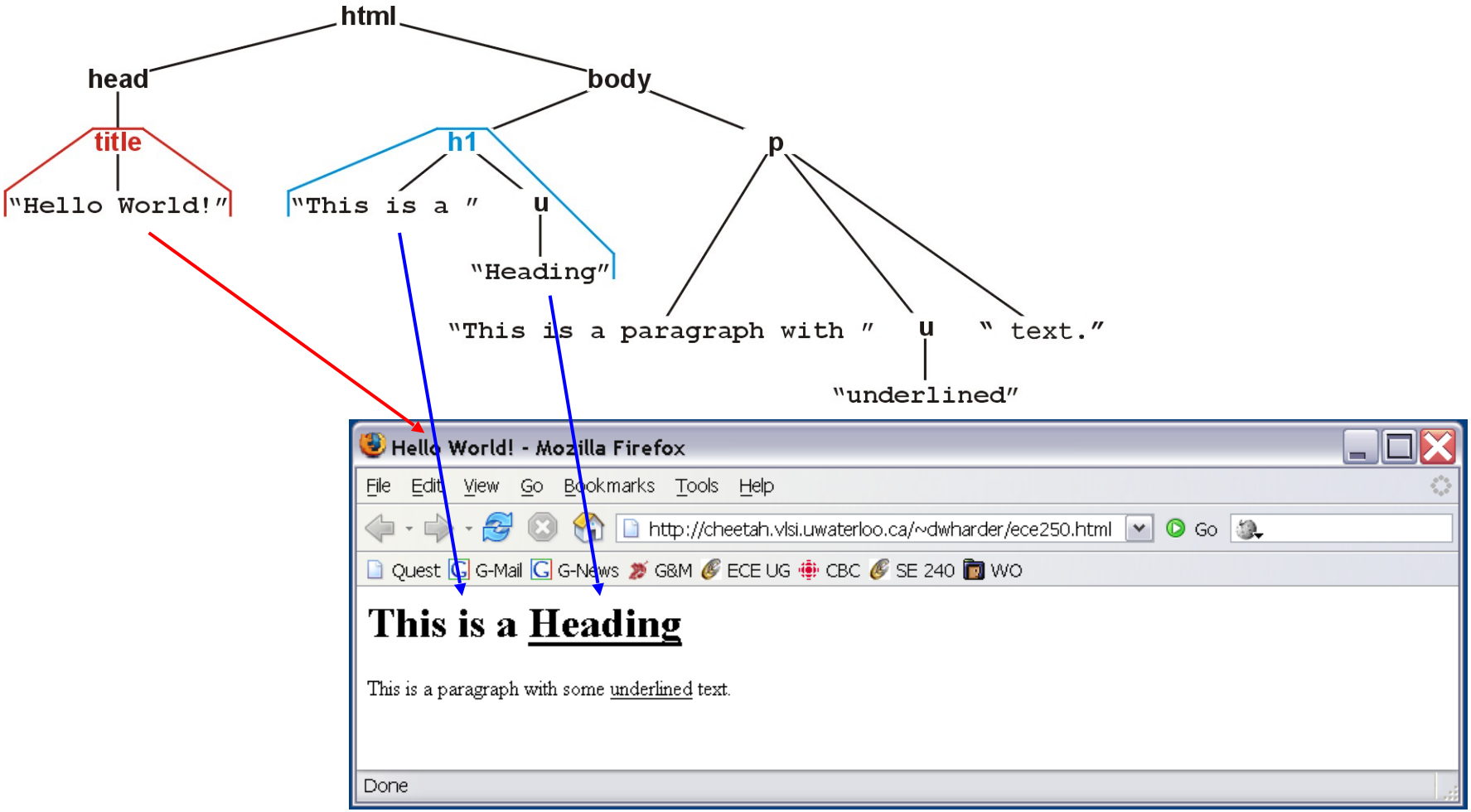
The nested tags define a tree rooted at the HTML tag

```
<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

- Cascading Style Sheets (CSS) make use of this tree structure to describe how HTML should be displayed
  - –For example:

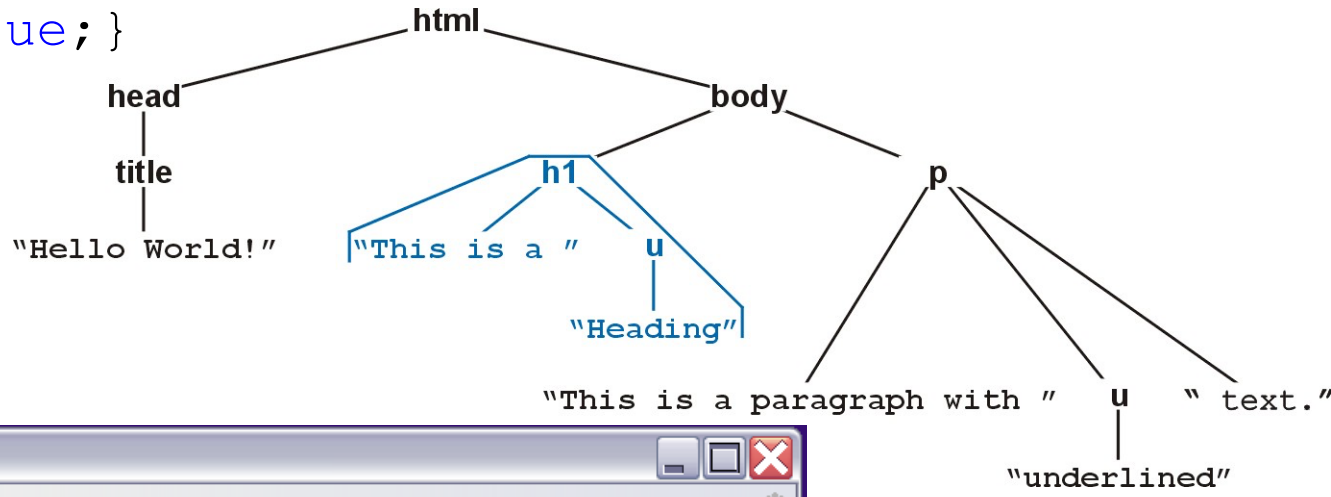
```
<style type="text/css">

   h1 { color:blue; }

</style>
```

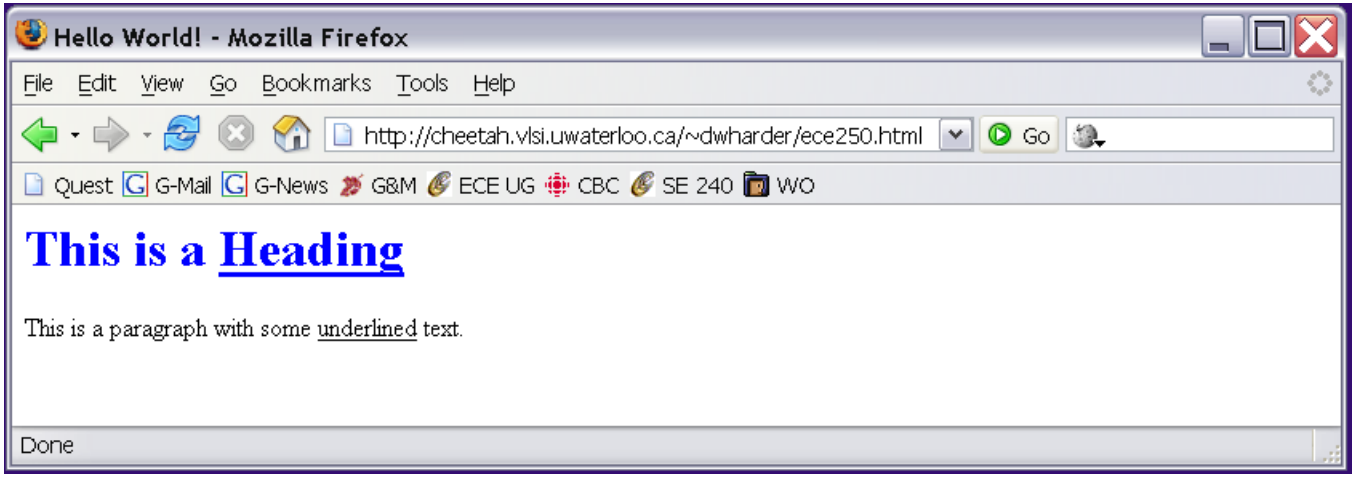
indicates all text/decorations descendant from an `h1` header should be blue

Example: XHTML and CSS

For example, this style renders as follows:

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

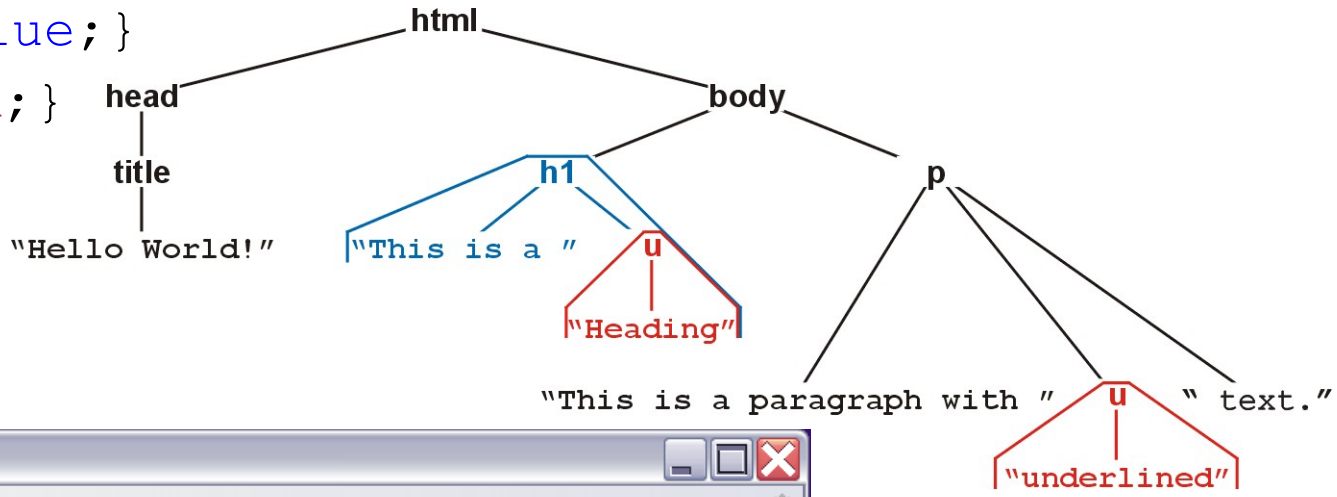


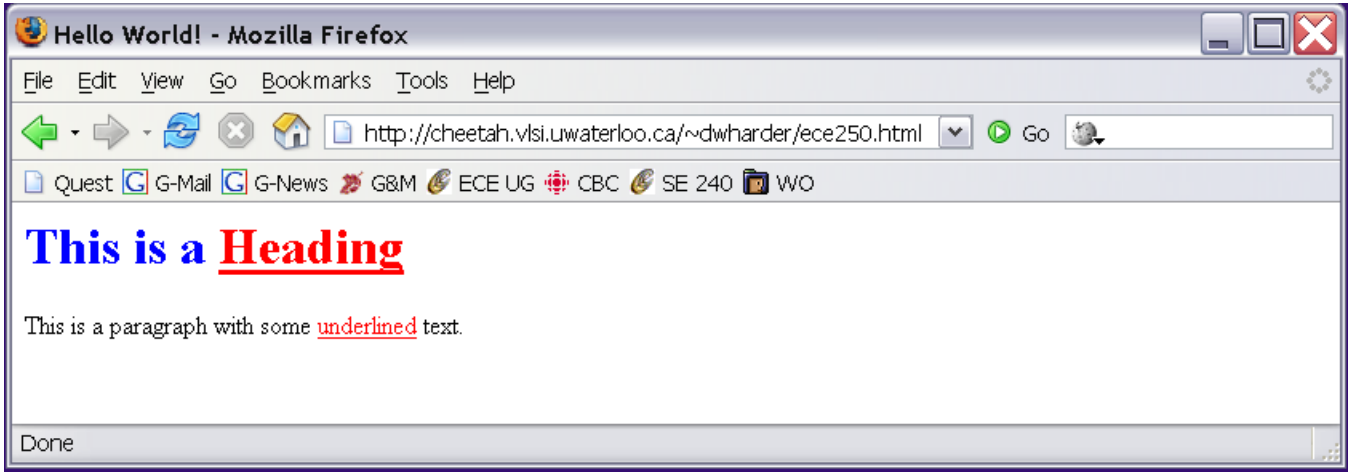


Example: XHTML and CSS

For example, this style renders as follows:

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





Example: XHTML and CSS

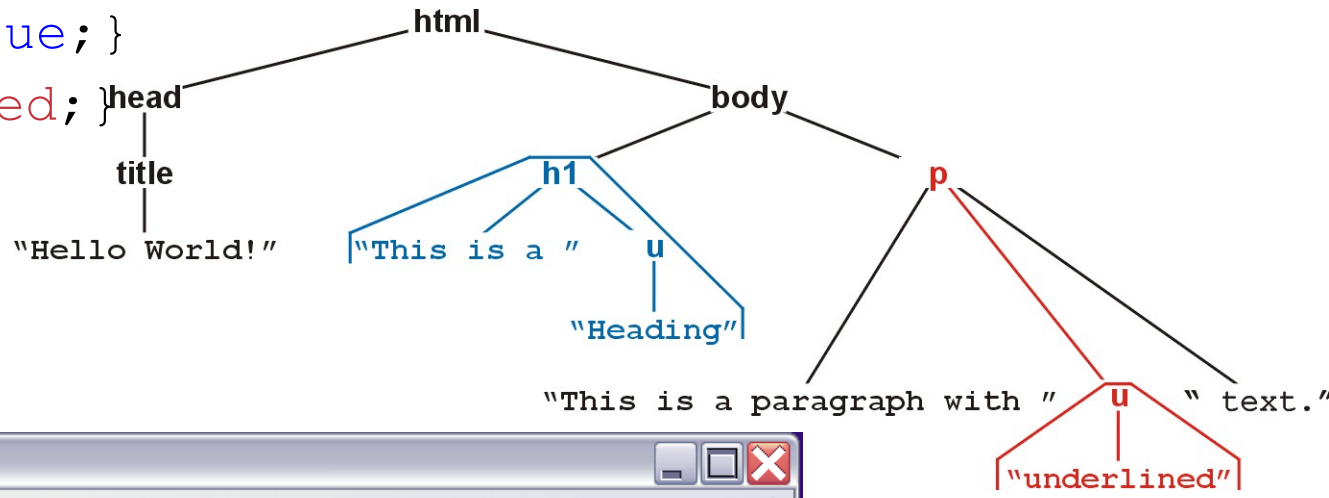
- Suppose you don't want underlined items in headers (`h1`) to be red
  - More specifically, suppose you want any underlined text within paragraphs to be red

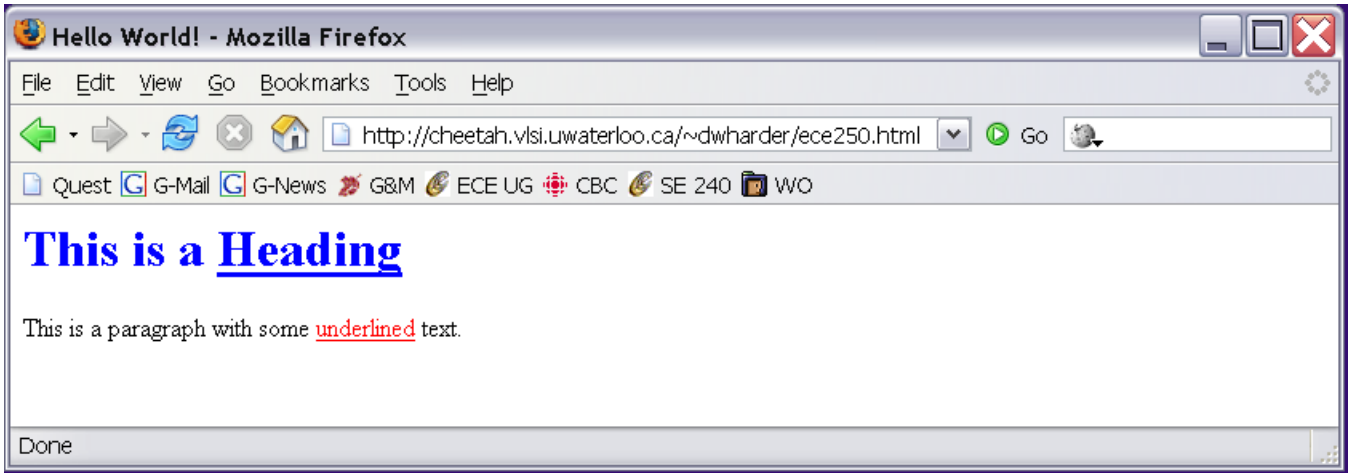
- That is, you only want text marked as `<u>text</u>` to be red if it is a descendant of a `<p>` tag

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”

References

[1] Donald E. Knuth, *The Art of Computer Programming, Volume 1: Fundamental Algorithms*, 3rd Ed., Addison Wesley, 1997, §2.2.1, p. 238.