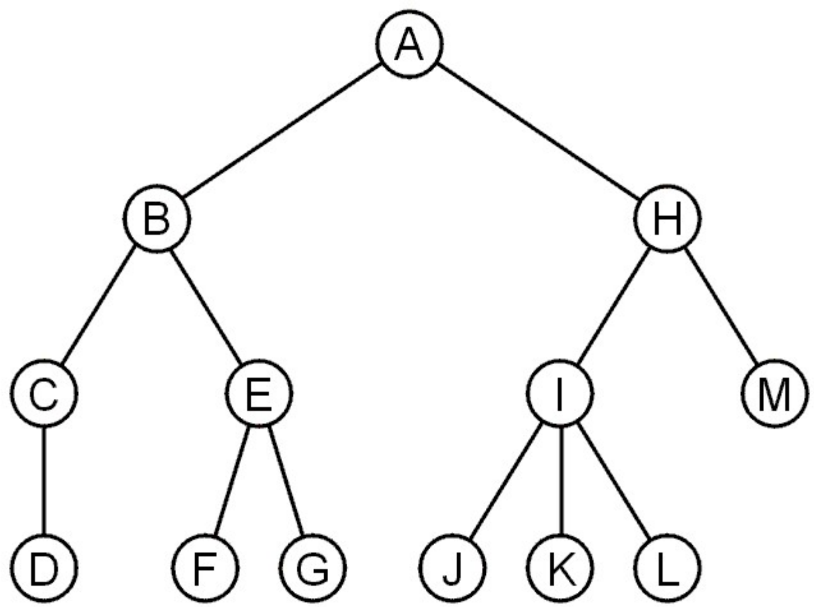
Data Structures

15. Tree Data Structure

Trees

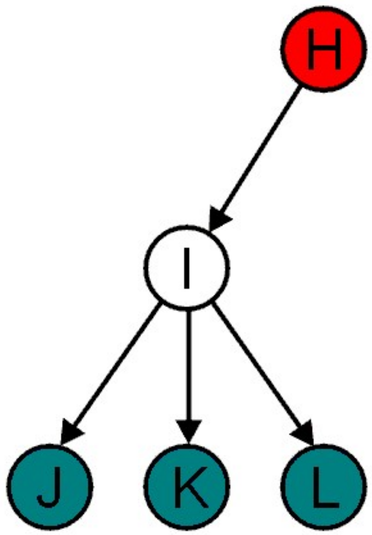
- A rooted tree data structure stores information in nodes
- Similar to linked lists:
  - There is a first node, or root
  - Each node has variable number of references to successors
  - Each node, other than the root, has exactly one node pointing to it



Terminology: Parent Child Relations

- All nodes have zero or more child nodes or children
  - I has three children: J, K and L

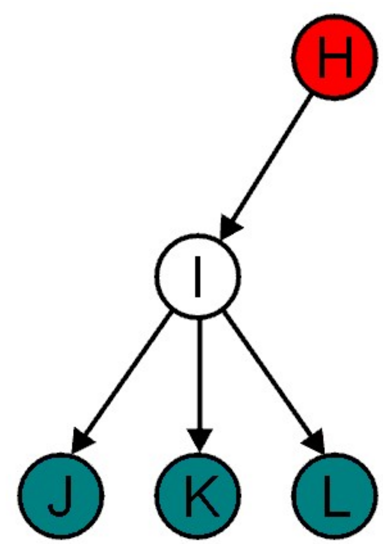
- For all nodes other than the root node, there is one parent node
  - H is the parent I



Terminology: Degree

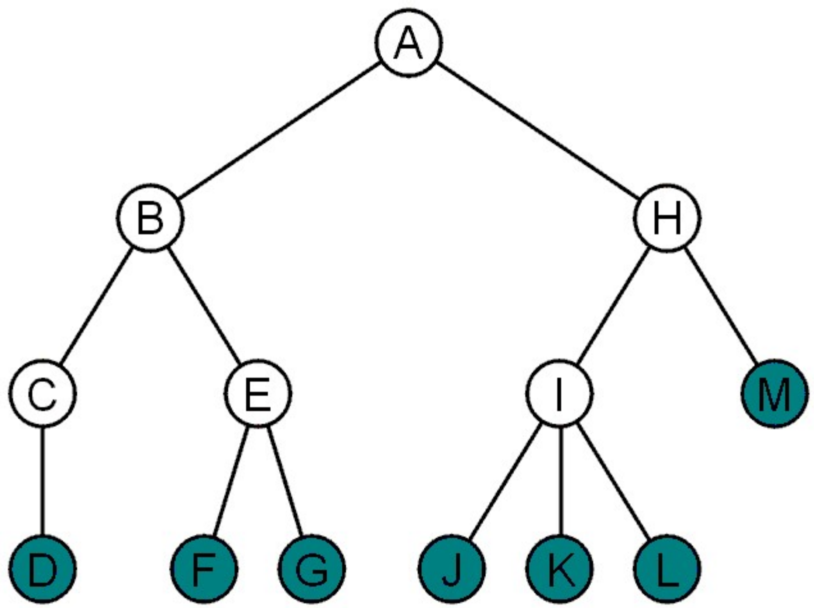
- The degree of a node is defined as the number of its children
  - `deg(I) = 3`

- Nodes with the same parent are siblings
  - J, K, and L are siblings



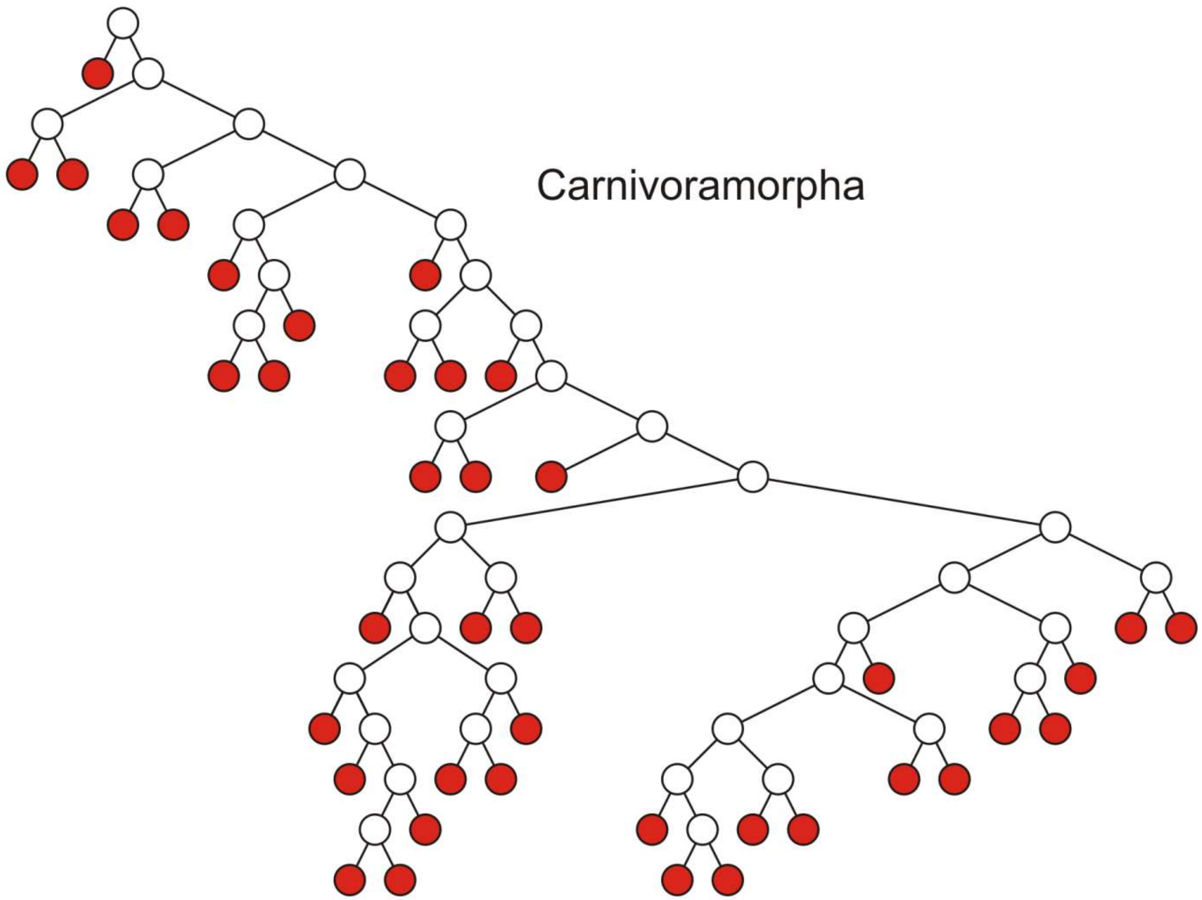
Terminology: Leaf And Internal Nodes

- 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 Examples

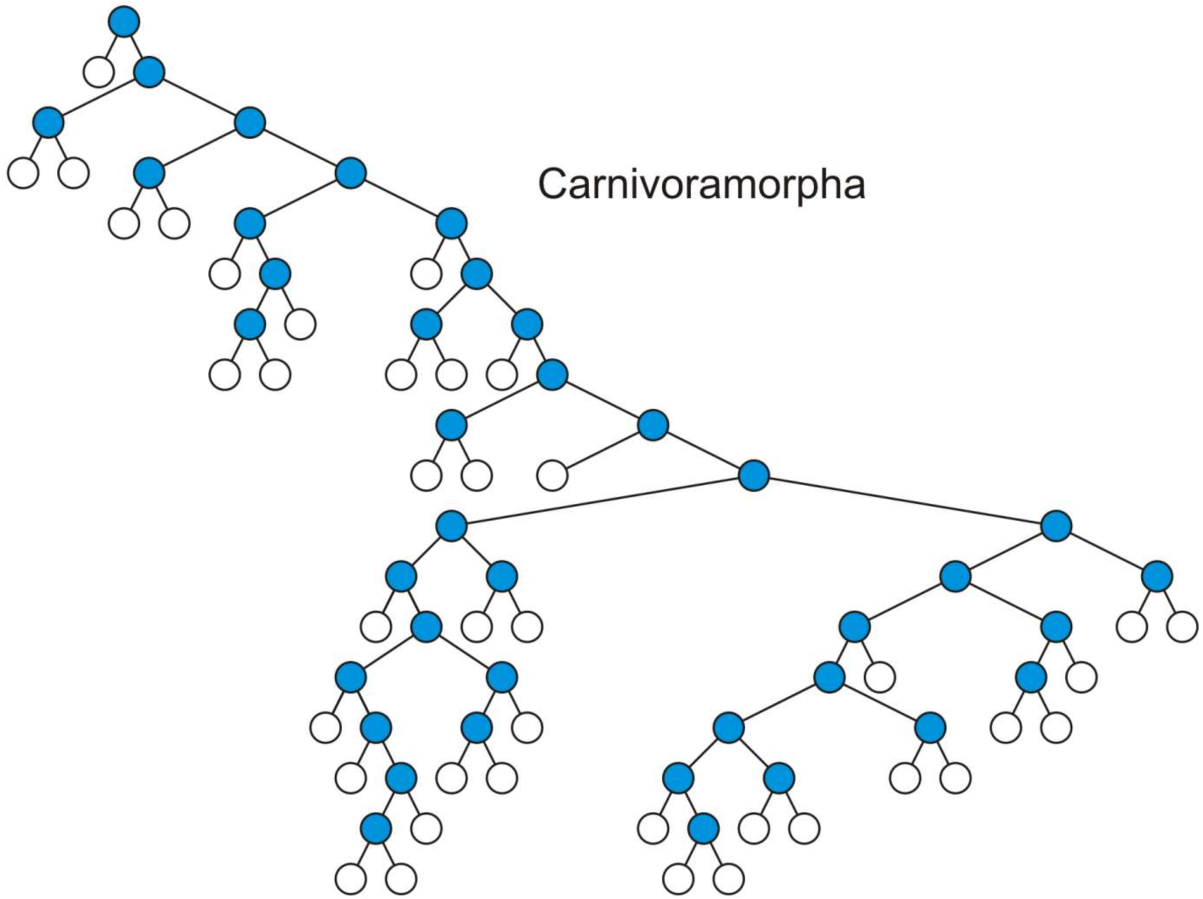
- Leaf nodes



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 Example

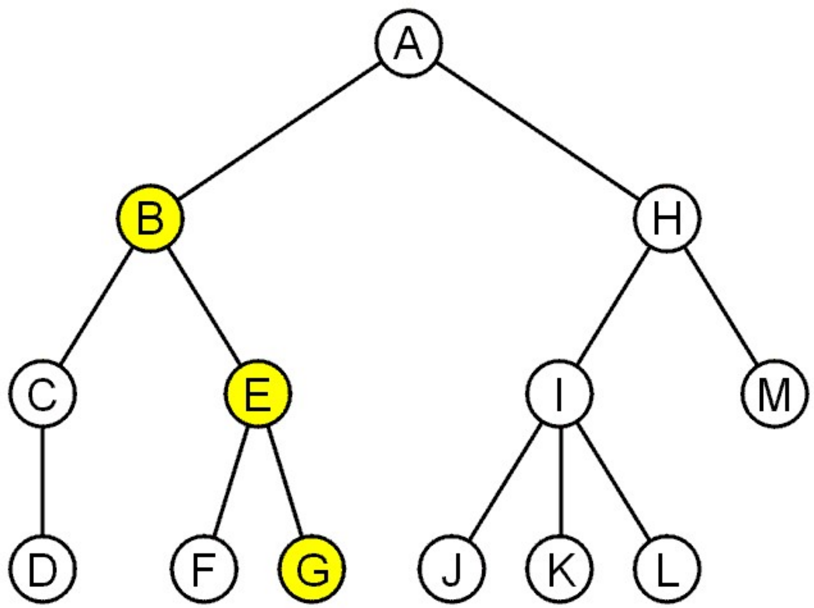
- Internal nodes



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

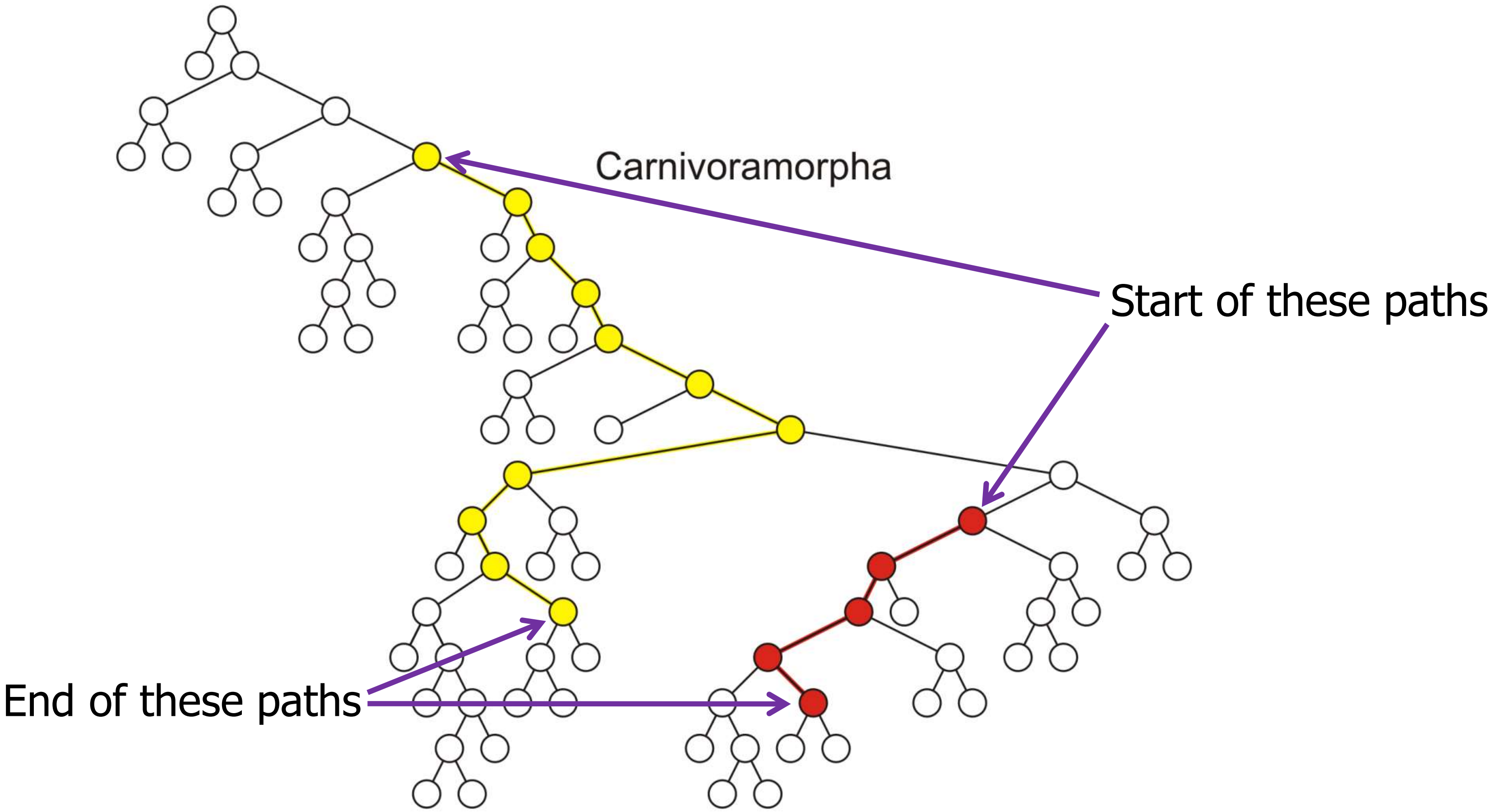
Terminology: Path

- 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: $n = |\text{nodes in the path}| - 1$
  - For example, the path (B, E, G) has length 2



Terminology: Path Example

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

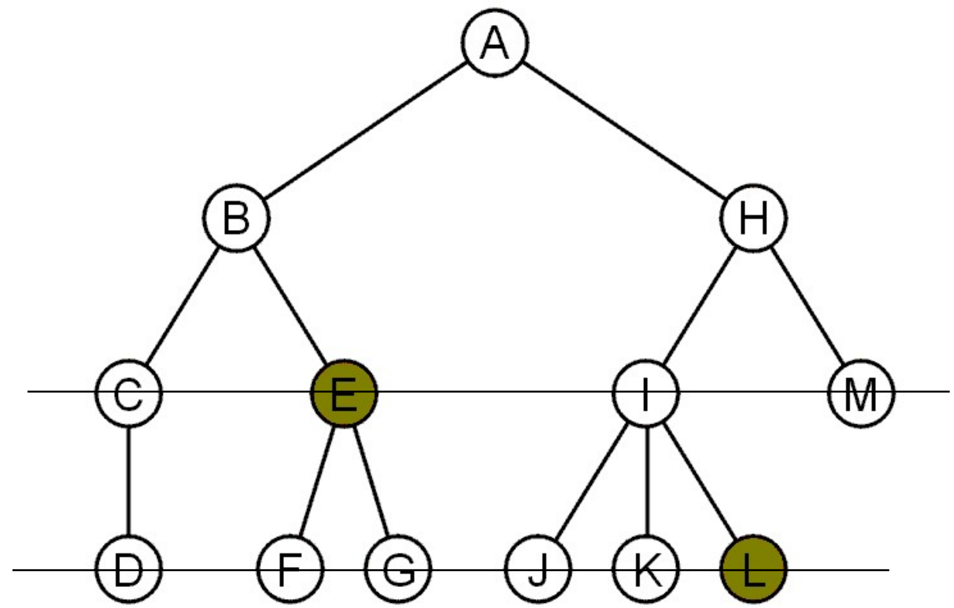


Wesley-Hunt, G. D.; Flynn, J. J. "Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of 'Miacoidea'

Terminology: Depth (or Level)

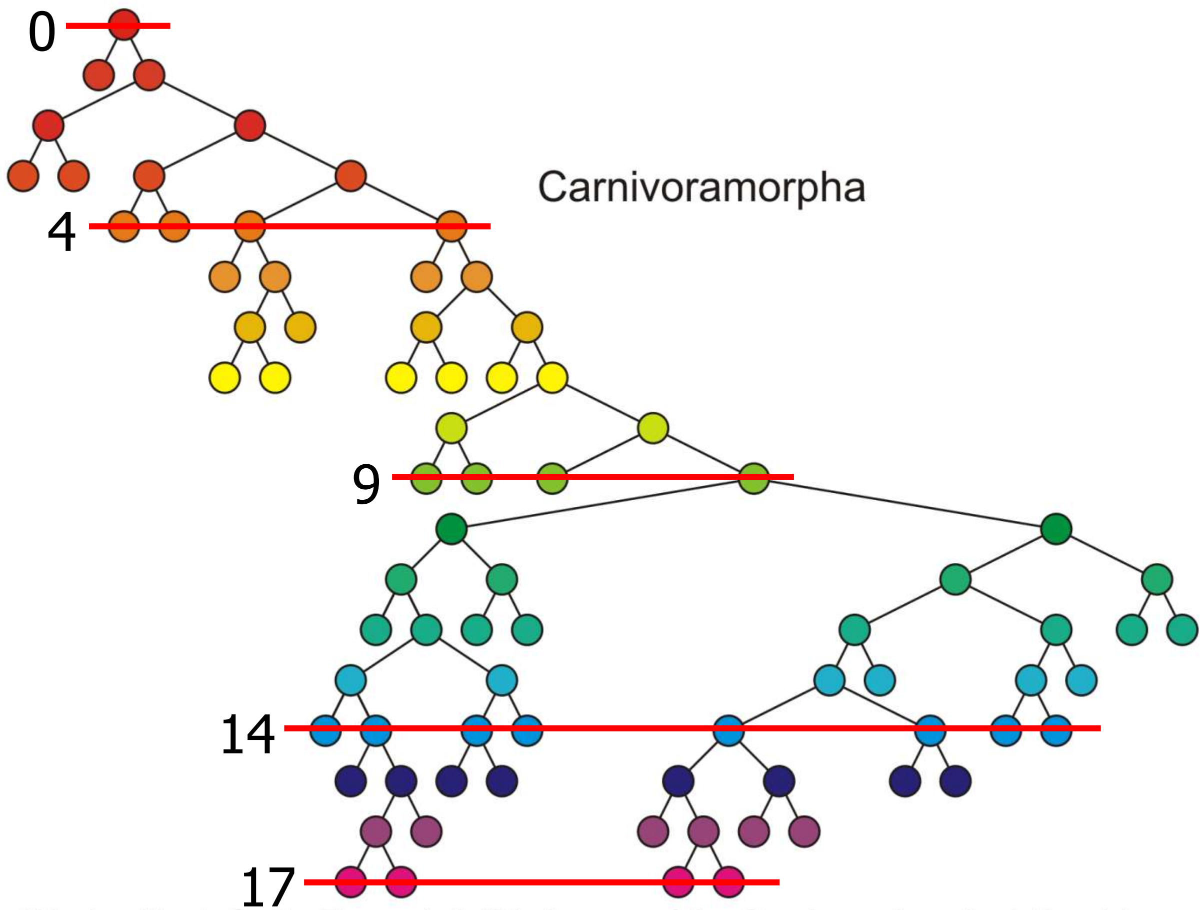
- For each node in a tree, there exists a unique path from the root node to that node

- The length of this path is the depth of the node, e.g.,
  - E has depth 2
  - L has depth 3



Terminology: Depth Example

- Nodes of depth up to 17



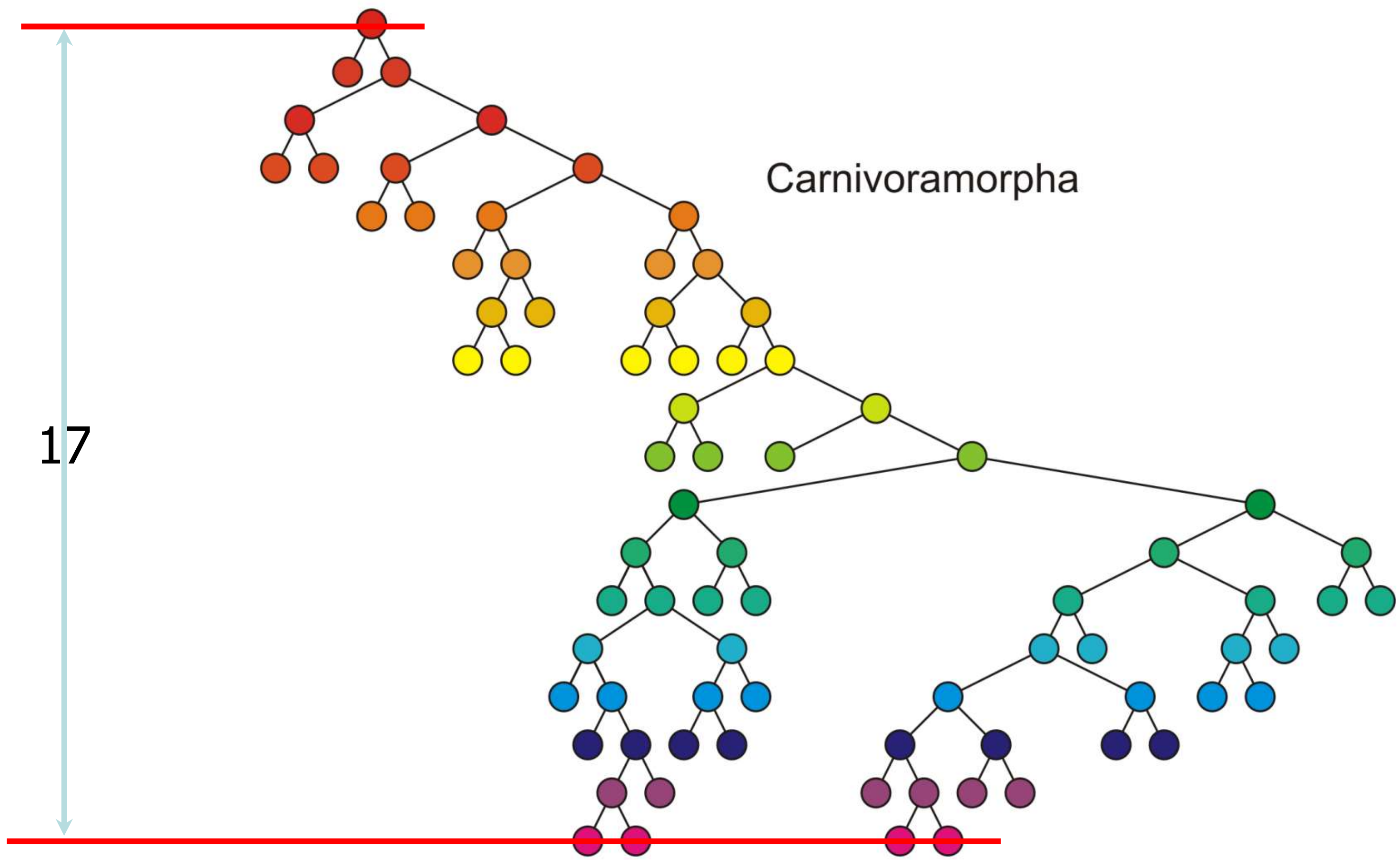
Wesley-Hunt, G. D.; Flynn, J. J. "Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of 'Miacoidea'

Terminology: Height

- 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

Terminology: Height Example

- Height of this tree is 17



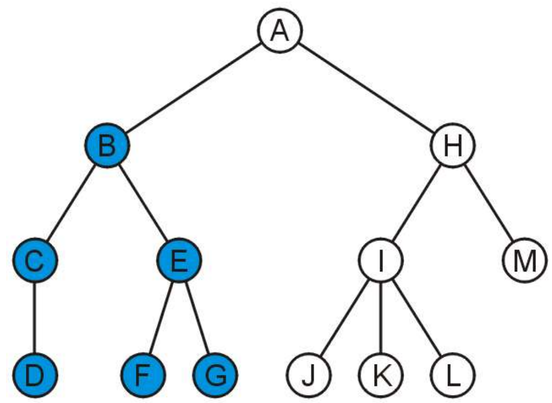
Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Terminology: Ancestors And Descendants

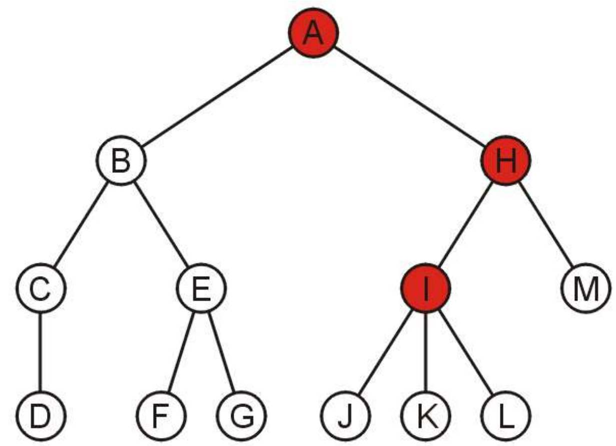
- If a path exists from node a to node b
  - a is an ancestor of b
  - b is a descendent of a
- Thus, a node is both an ancestor and a descendant of itself
  - We can add the adjective strict to exclude equality
  - a is a strict descendent of b if a is a descendant of b but a ≠ b
- The root node is an ancestor of all nodes

Terminology: Ancestors And Descendants Example

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

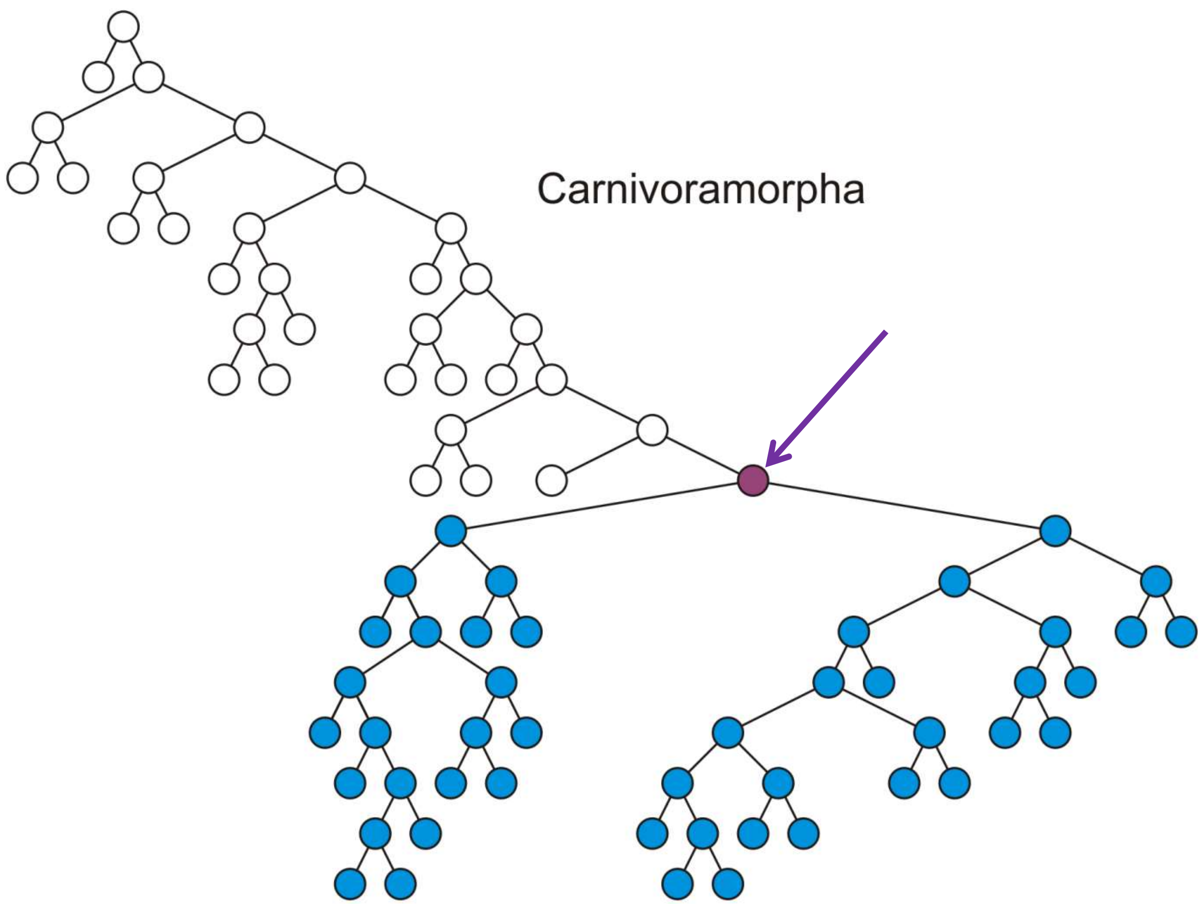


- The ancestors of node I are H and A



Terminology: Descendants Example

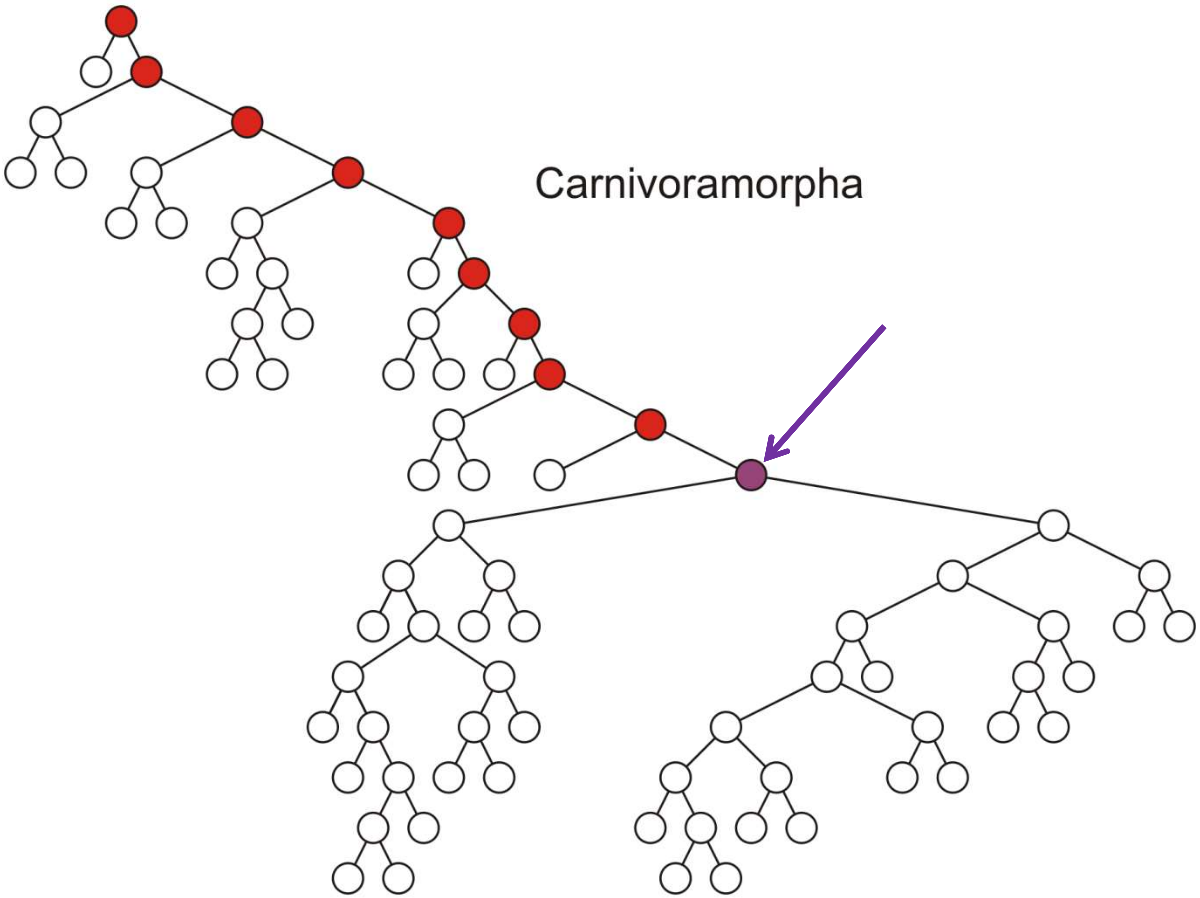
- All descendants (including itself) of the indicated node



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

Terminology: Ancestors Example

- All ancestors (including itself) of the indicated node



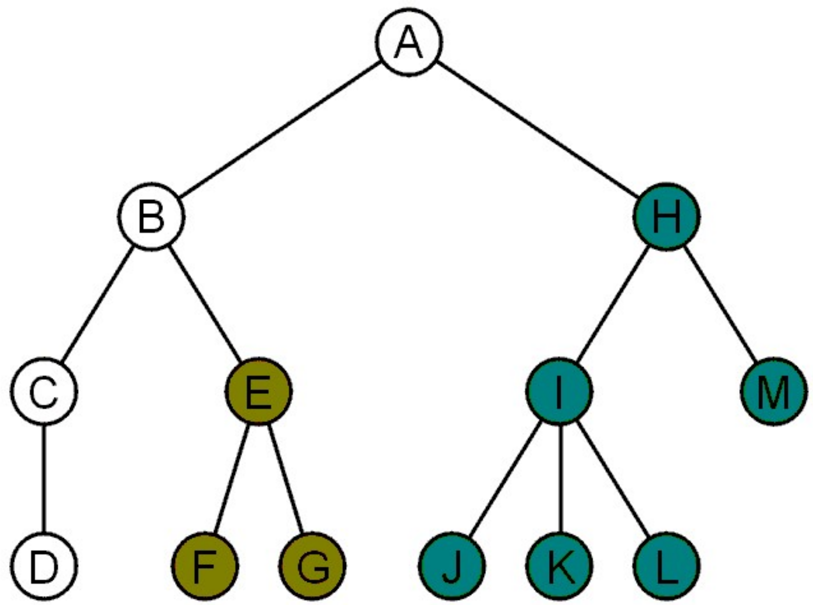
Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Terminology: Subtree

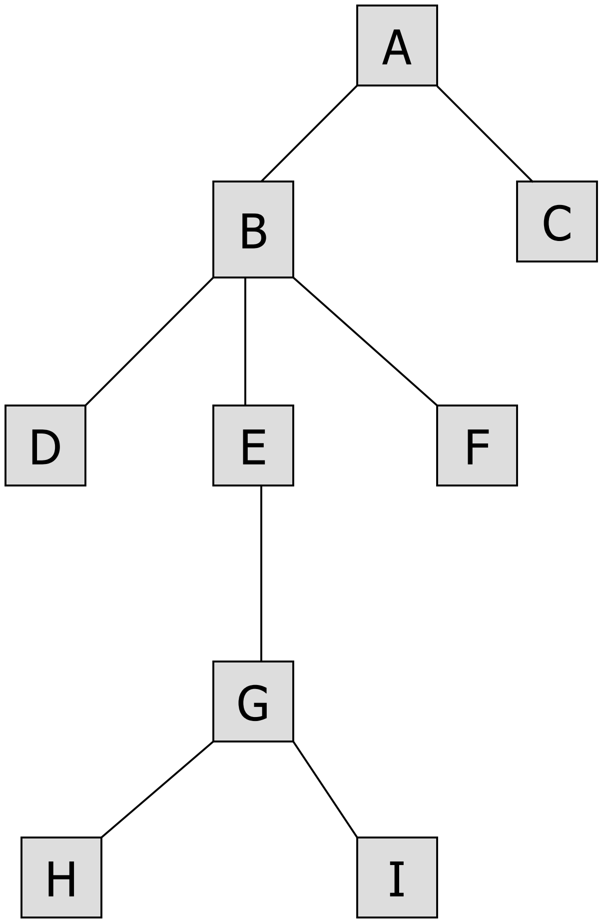
- Another approach to a tree is to define the tree recursively
  - A degree-0 node is a tree

- A node with degree n is a tree if it has n children
  - 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



Tree Properties



Property	Value
Number of nodes	
Height	
Root Node	
Leaves	
Ancestors of H	
Descendants of B	
Siblings of E	
Left subtree	

Example: HTML (1)

- HTML document has a tree structure

```html
<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: HTML (2)

- HTML document has a tree structure

title

```
<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>
```

heading

body of page

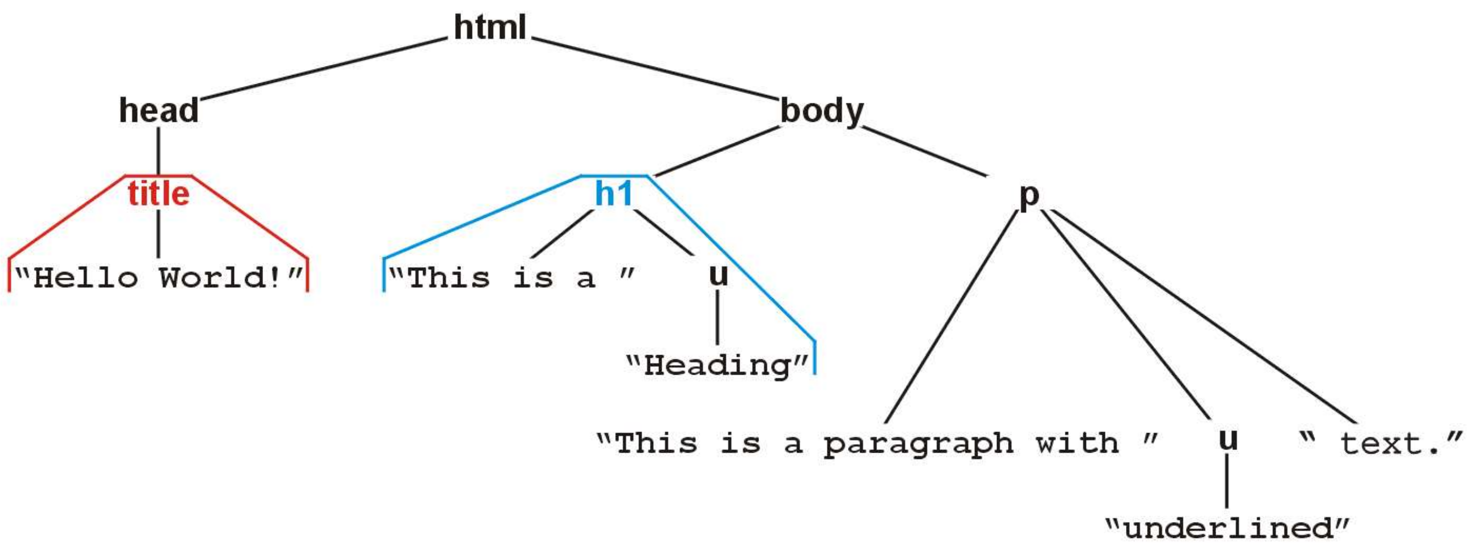
paragraph

underlining

Example: HTML (3)

- The nested tags define a tree rooted at the HTML tag

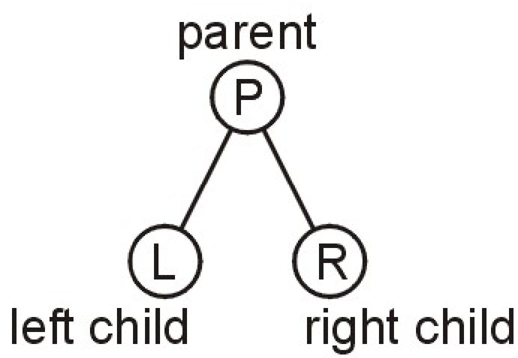
```
<html>
   <head>
      <title>Hello World!</title>
   </head>
   <body>
      <h1>This is a <u>Heading</u></h1>

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



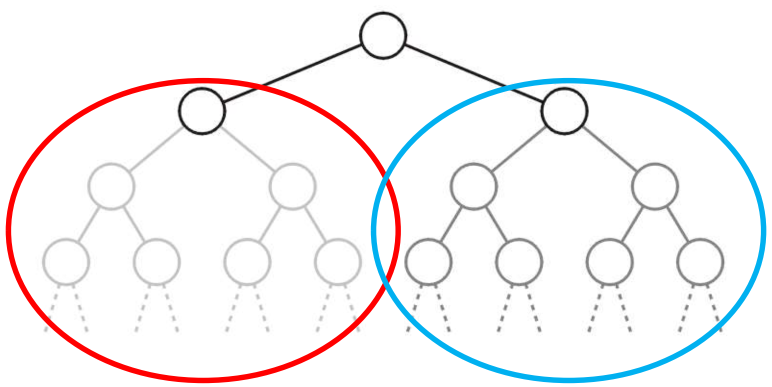
Binary Tree

Binary Tree

- In a binary tree each node has at most two children
  - Allows to label the children as left and right

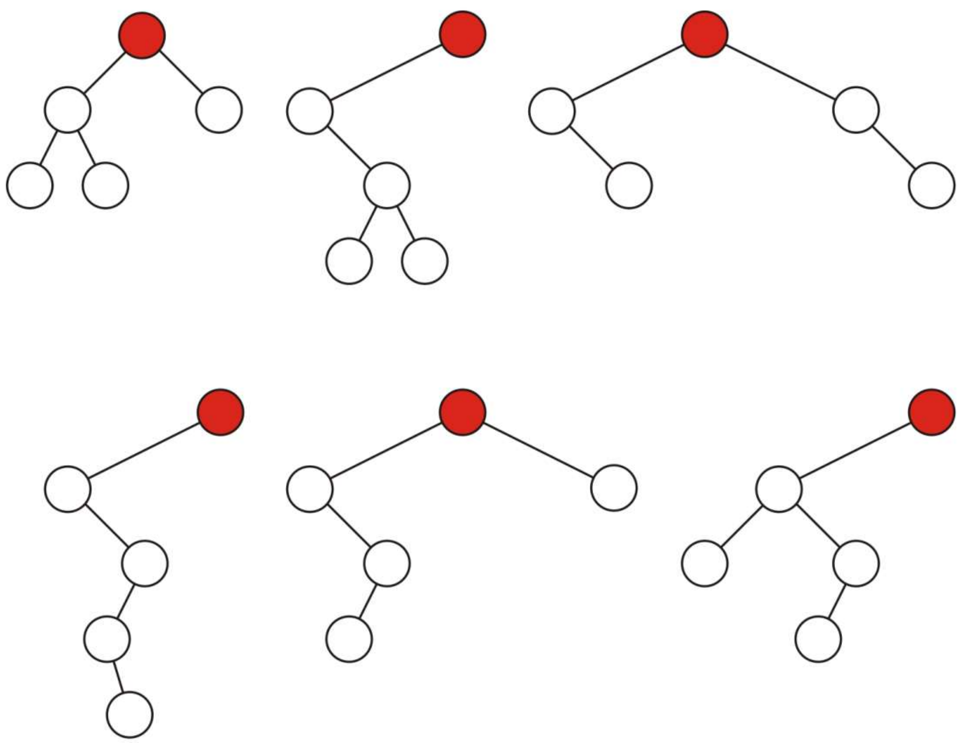


- Likewise, the two sub-trees are referred as
  - Left-hand subtree
  - Right-hand subtree



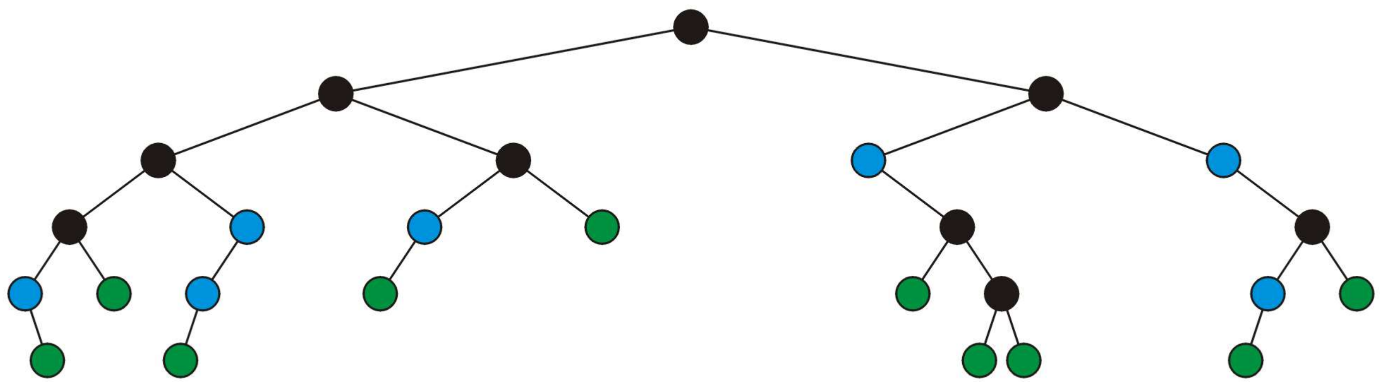
Binary Tree: Example

- Some variations on binary trees with five nodes



Binary Tree: Full Node

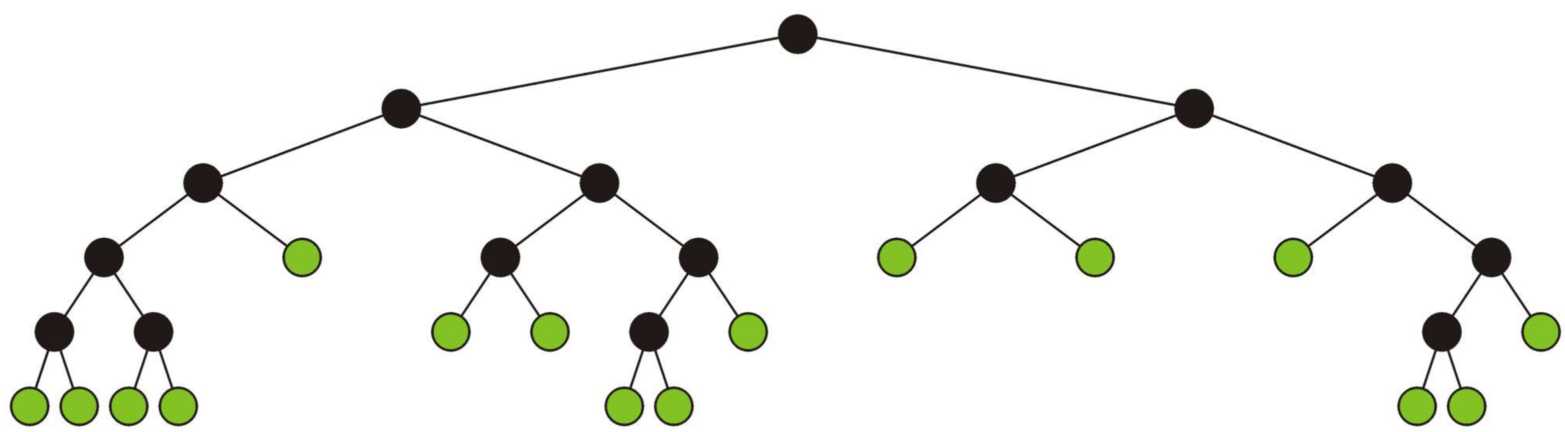
- A full node is a node where both the left and right sub-trees are non-empty trees



full nodes ● neither ● leaf nodes ●

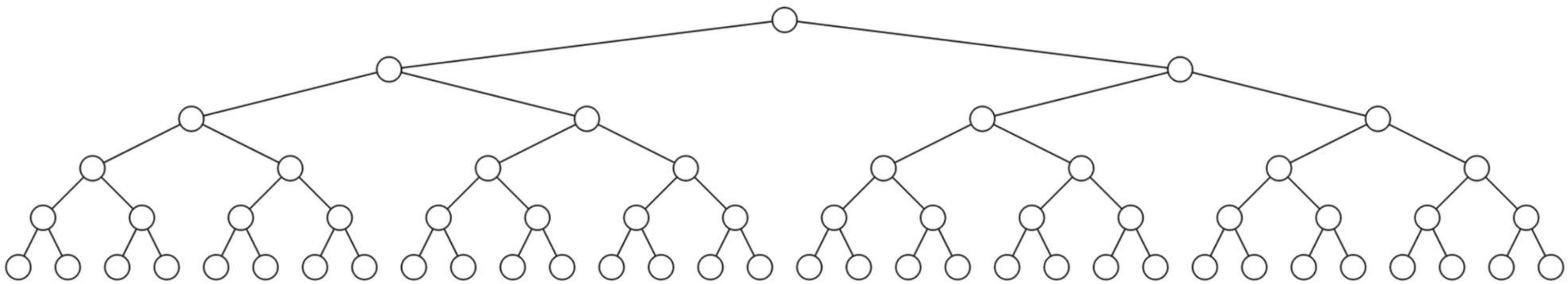
Full Binary Tree

- A full binary tree is where each node is:
  - A full node, or
  - A leaf node
- Full binary tree is also called proper binary tree, strictly binary tree or 2-tree



Complete (Or Perfect) Binary Tree

- A complete binary tree of height h is a binary tree where
  - All leaf nodes have the same depth h
  - All other nodes are full

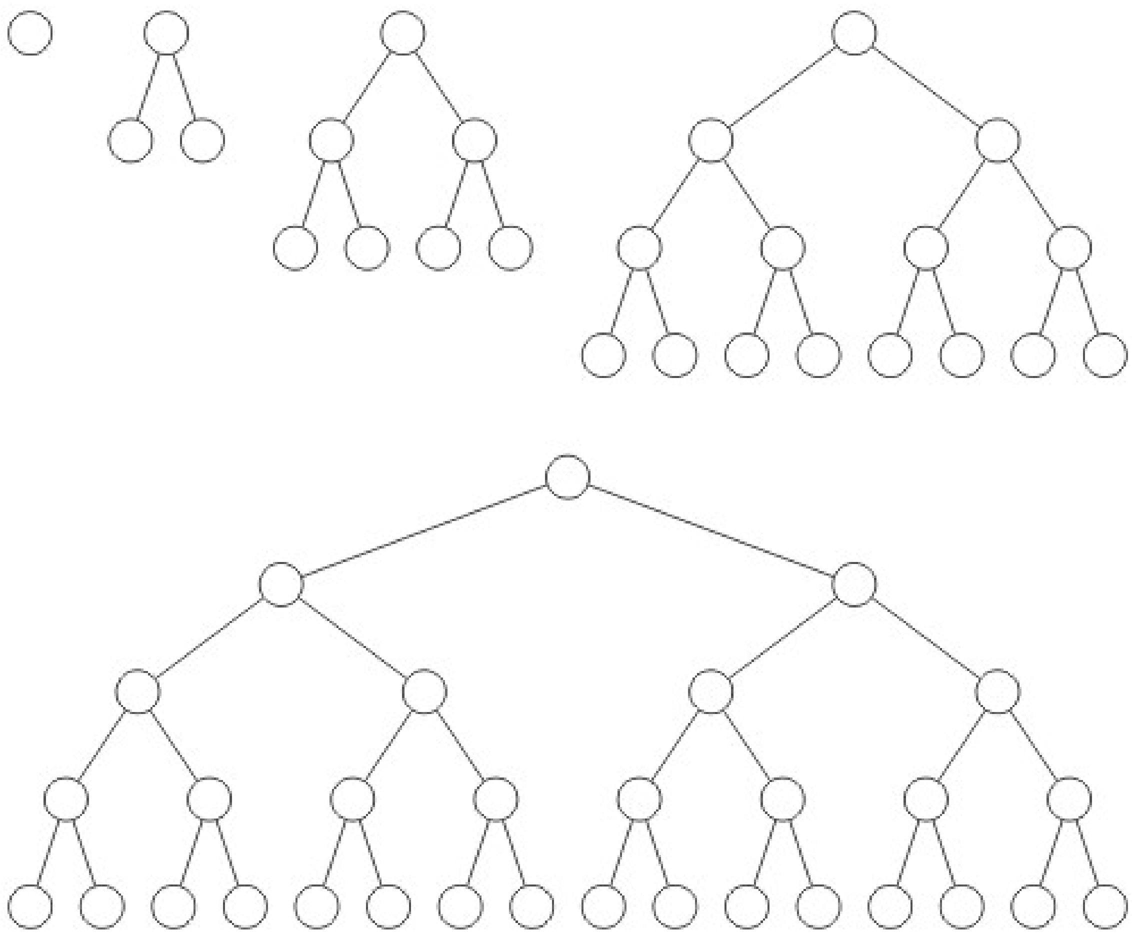


Complete Binary Tree: Recursive Definition

- A binary tree of height h = 0 is perfect
- A binary tree with height h > 0 is perfect
  - If both sub-trees are prefect binary trees of height h – 1

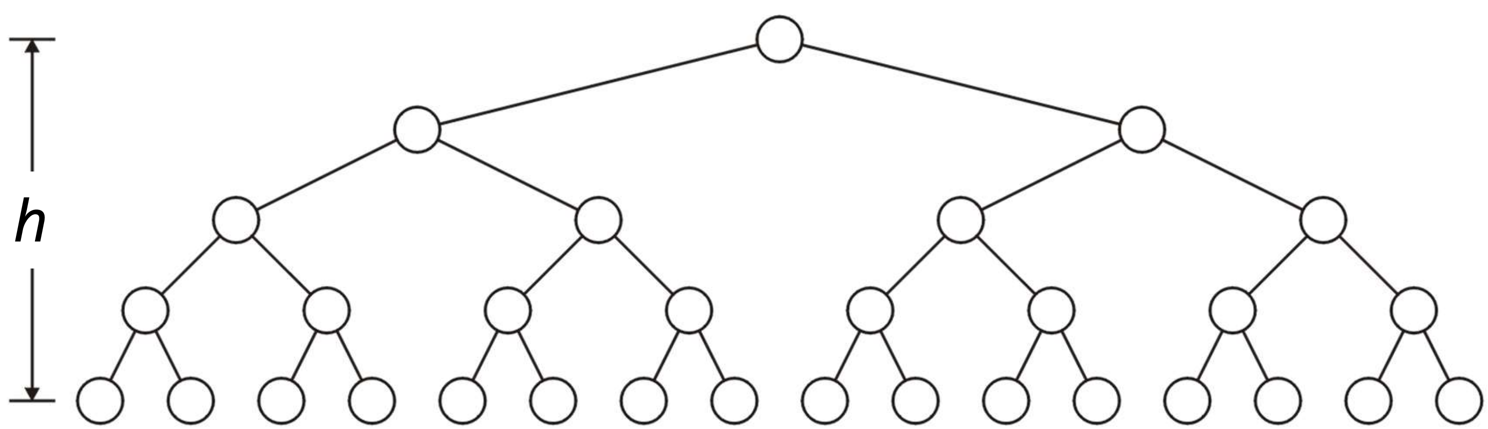
Complete Binary Tree: Example

- Complete binary trees of height h = 0, 1, 2, 3 and 4



Binary Tree: Properties (1)

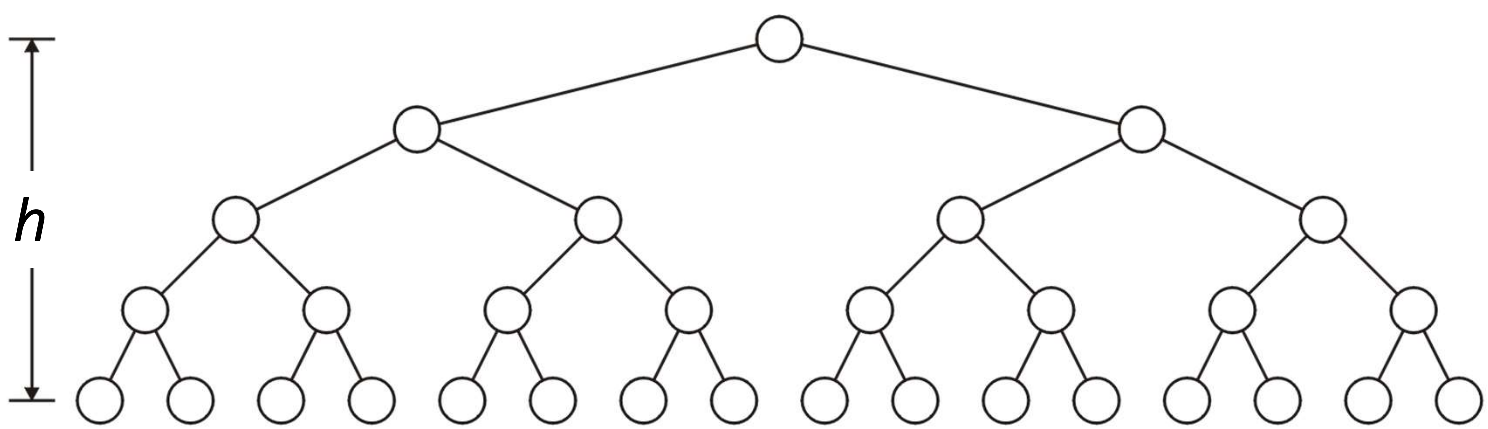
- A complete binary tree with height h has 2^h leaf nodes



Binary Tree: Properties (2)

- A complete binary tree with height h has 2^h leaf nodes
- A complete binary tree of height h has $2^{h+1} - 1$ nodes

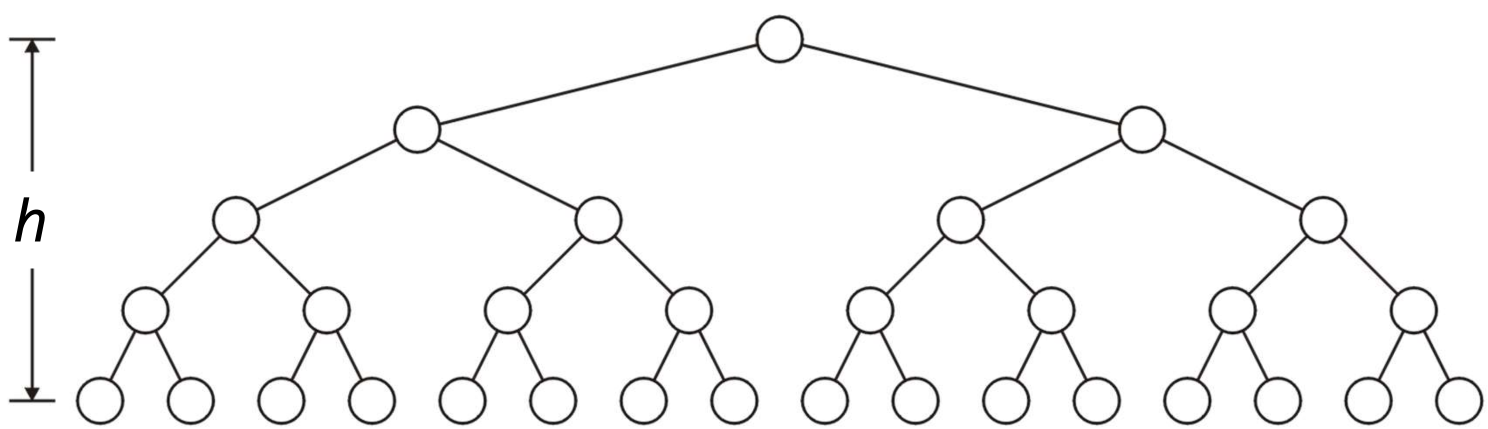
$$n = 2^0 + 2^1 + 2^2 + \ldots + 2^h = \sum_{j=0}^{h} 2^j = 2^{h+1} - 1$$



Binary Tree: Properties (3)

- A complete binary tree with height h has 2^h leaf nodes

- A complete binary tree of height h has $2^{h+1} - 1$ nodes
  - Number of leaf nodes: $L = 2^h$
  - Number of internal nodes: $2^h - 1$
  - Total number of nodes: $2L - 1 = 2^{h+1} - 1$



Binary Tree: Properties (4)

- A complete binary tree with height h has 2^h leaf nodes
- A complete binary tree of height h has $2^{h+1} - 1$ nodes
  - Number of leaf nodes: $L = 2^h$
  - Number of internal nodes: $2^h - 1$
  - Total number of nodes: $2L-1 = 2^{h+1} - 1$
- A complete binary tree with n nodes has height $\log_2(n + 1) - 1$

$$n = 2^{h+1} - 1$$

$$2^{h+1} = n + 1$$

$$h + 1 = \log_2(n + 1)$$

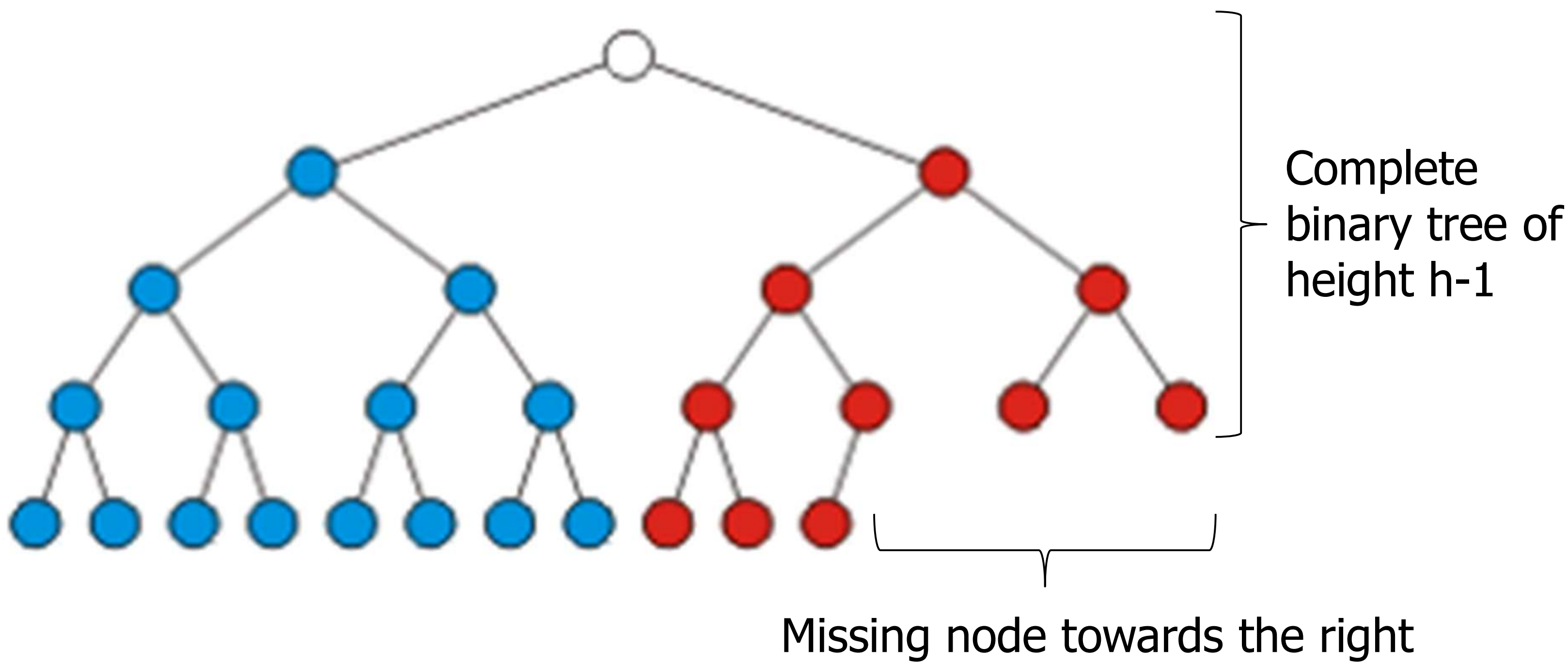
$$\Rightarrow h = \log_2(n + 1) - 1$$

Binary Tree: Properties (4)

- A complete binary tree with height h has 2^h leaf nodes
- A complete binary tree of height h has $2^{h+1} - 1$ nodes
  - Number of leaf nodes: $L = 2^h$
  - Number of internal nodes: $2^h - 1$
  - Total number of nodes: $2L-1 = 2^{h+1} - 1$
- A complete binary tree with n nodes has height $\log_2(n + 1) - 1$
- Number n of nodes in a binary tree of height h is at least $h+1$ and at most $2^{h+1} - 1$

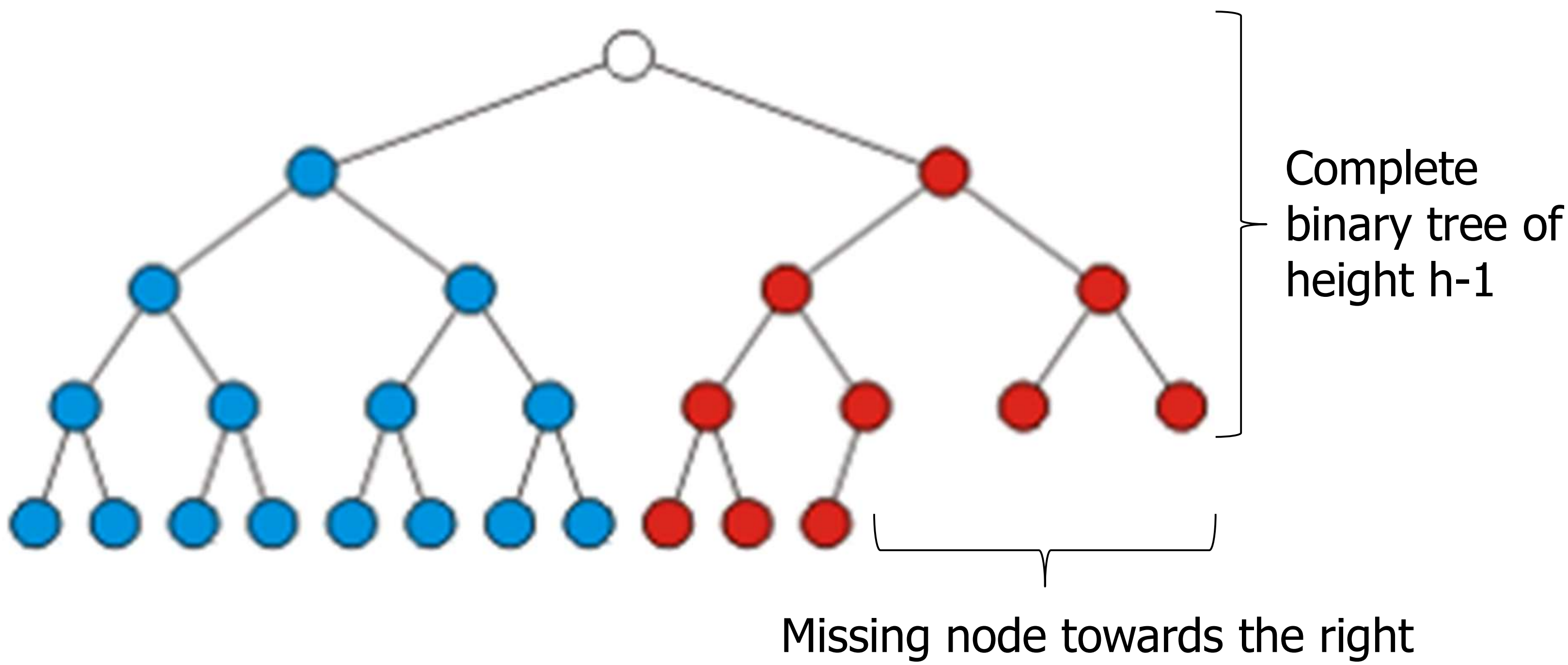
Almost (or Nearly) Complete Binary Tree

- Almost complete binary tree of height h is a binary tree in which
  1. There are 2^d nodes at depth d for d = 1,2,...,h−1
     - Each leaf in the tree is either at level h or at level h− 1
  2. The nodes at depth h are as far left as possible



Almost (or Nearly) Complete Binary Tree

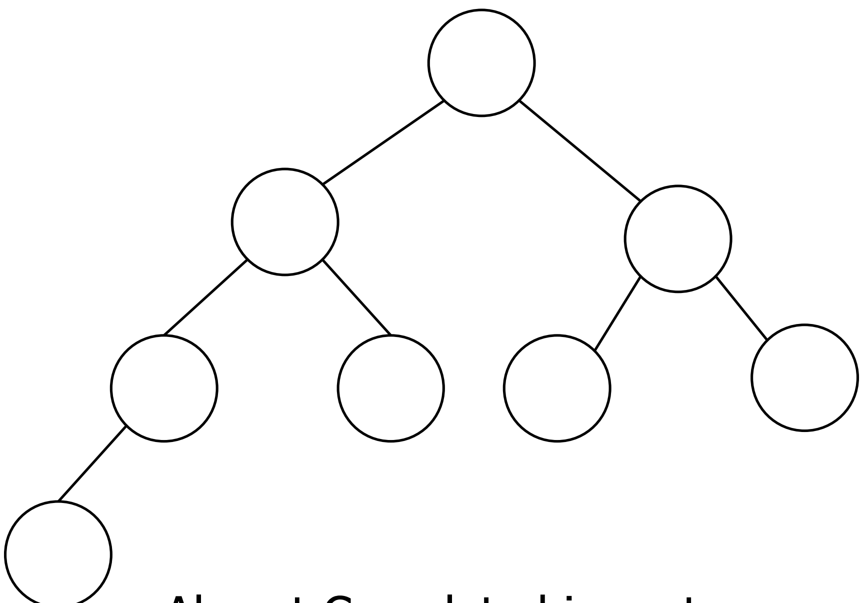
- Almost complete binary tree of height h is a binary tree in which
  1. There are 2^d nodes at depth `d` for `d = 1,2,...,h-1`
     - ➢ Each leaf in the tree is either at level `h` or at level `h- 1`
  2. **The nodes at depth h are as far left as possible (Formal ?)**



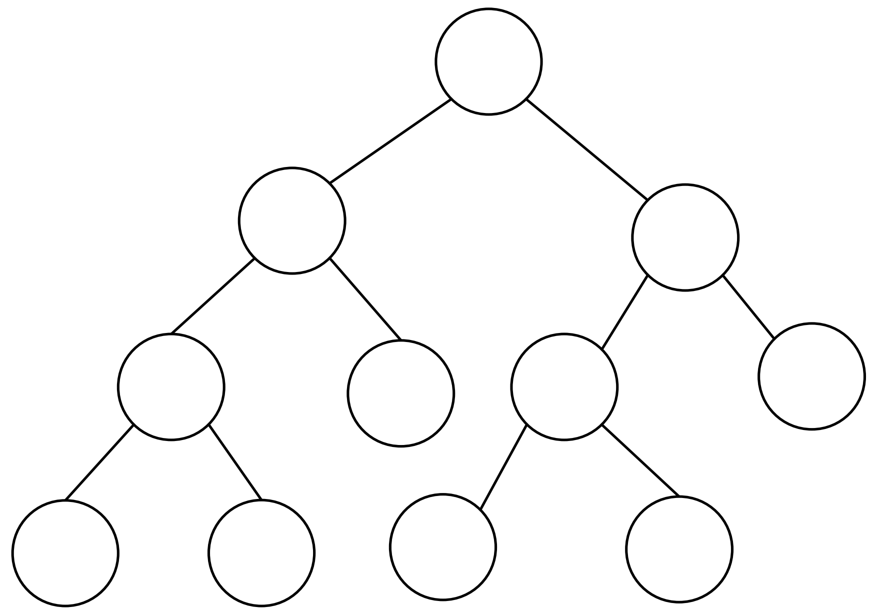
Almost (or Nearly) Complete Binary Tree

Condition 2: The nodes at depth h are as far left as possible

- If a node p at depth h−1 has a left child
  - Every node at depth h−1 to the left of p has 2 children
- If a node at depth h−1 has a right child
  - It also has a left child

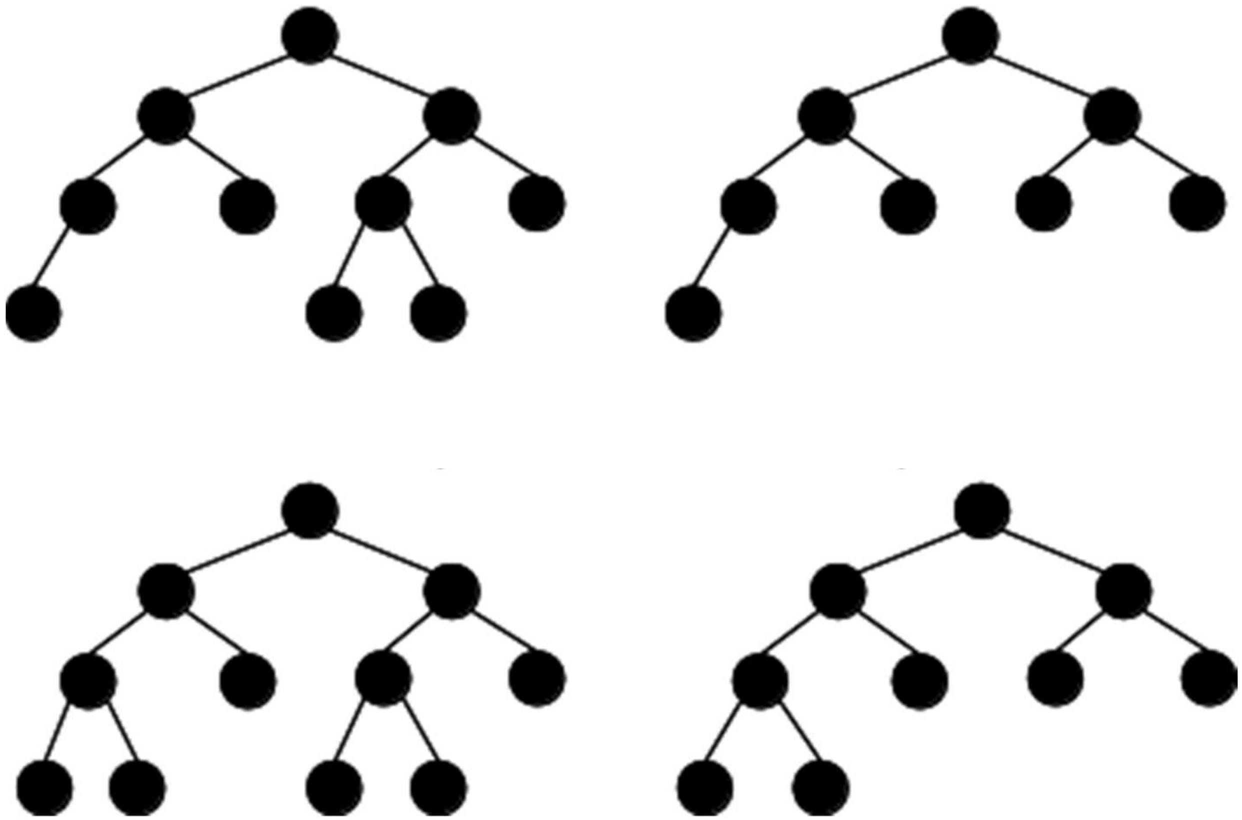


Almost Complete binary tree



Not Almost Complete binary tree
(condition 2 violated)

Full vs. Almost Complete Binary Tree



Almost Complete Binary Tree: Properties

- Total number of nodes `n` are between
  - Complete binary tree of height `h-1`, i.e., 2^h nodes
  - Complete binary tree of height `h`, i.e., $2^{h+1} - 1$ nodes

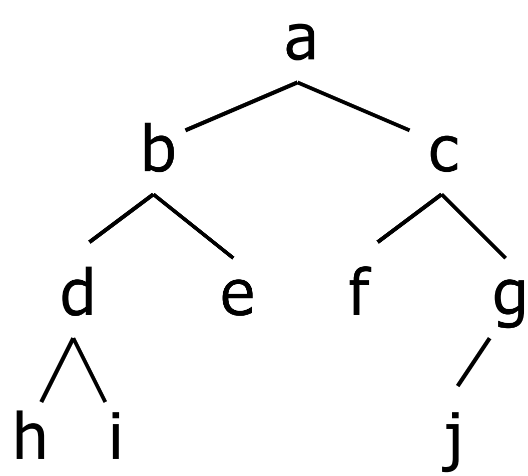
- Height `h` is the largest integer less than or equal to $\log_2(n)$

(Completely) Balanced Binary Tree

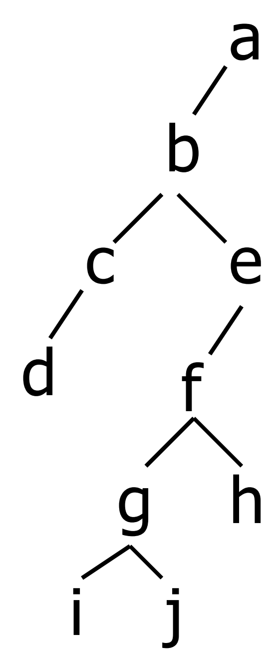
- Balanced binary tree
  - For each node, the difference in height of the right and left sub-trees is no more than one

- Completely balance binary tree
  - Left and right sub-trees of every node have the same height

Balanced Binary Tree: Example



A balanced binary tree



An unbalanced binary tree

Any Question So Far?