CS-2001
Data Structures
Spring 2022
Introduction to Tree

Mr. Muhammad Yousaf
National University of Computer and Emerging Sciences,
Faisalabad, Pakistan.

Previously - Linear Data Structures

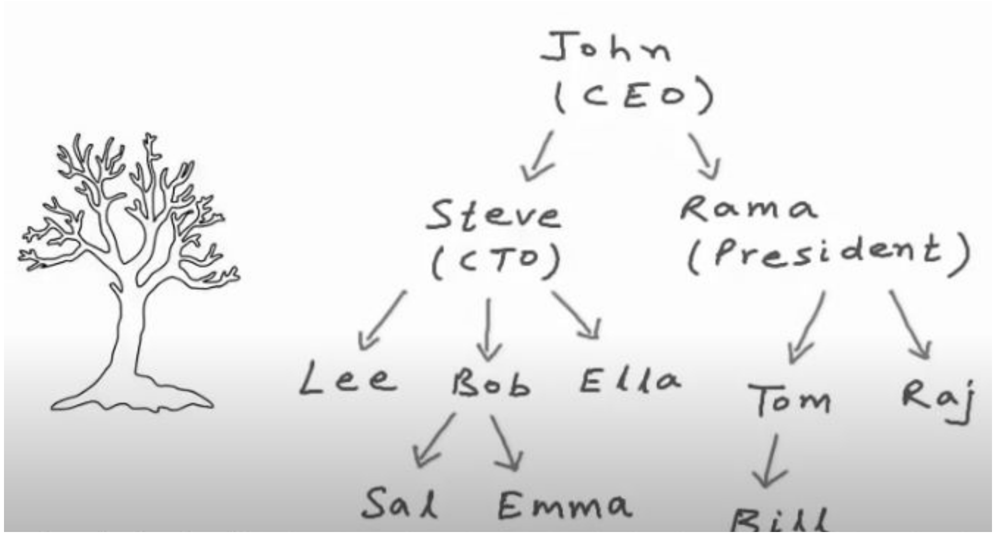
- Since now, we've talked about only Linear data structures i.e.
  - Arrays, linked lists, stacks and queues etc.
  - items were linked with next and previous pointers etc.

- How should I decide which data structures to use?
  - Depends what needs to be stored?
  - Minimize the cost of operations i.e. binary search.
  - Memory usage?
  - Ease of implementation?

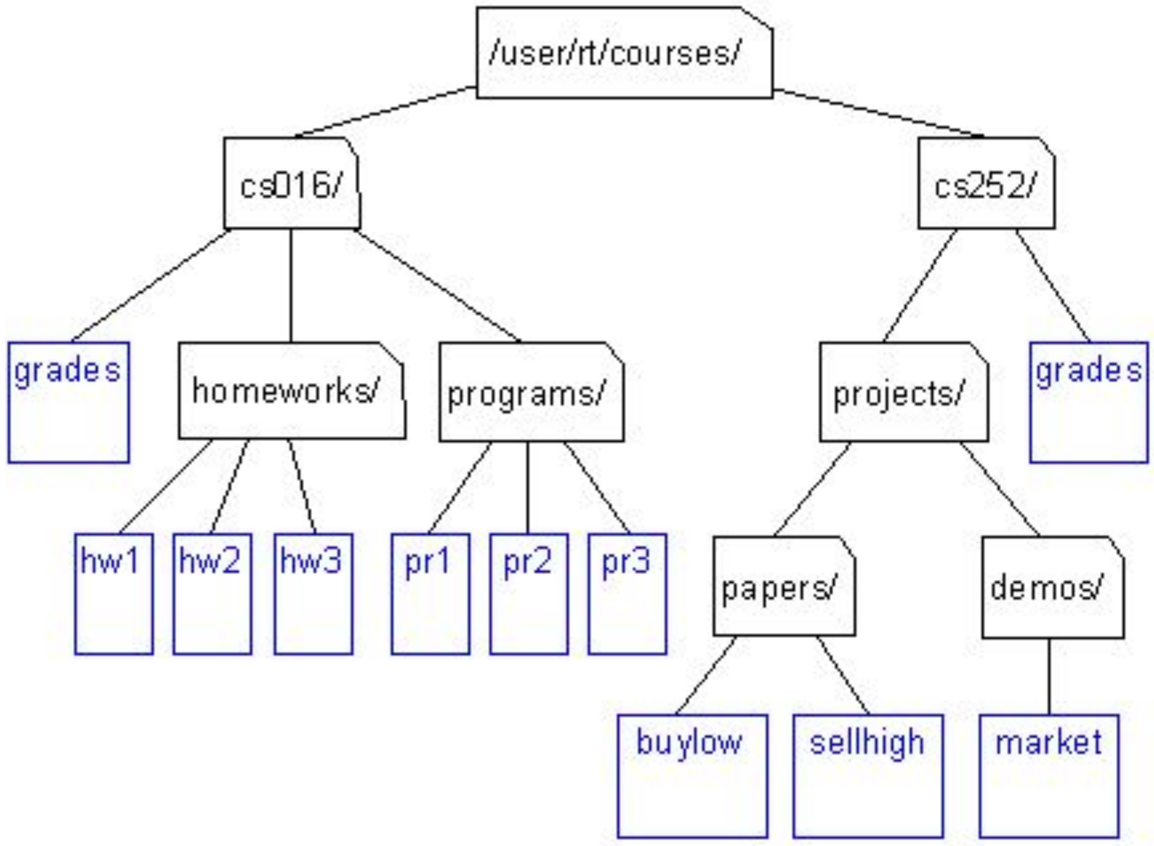
Trees - Non-Linear Data Structures

- Hierarchical data structure
- Finite collection of Items, stored in hierarchal fashion
- A connected acyclic graph
-
- Examples:
  - ToC in a book has a tree structure
  - A family tree
  - Organizational-positions tree

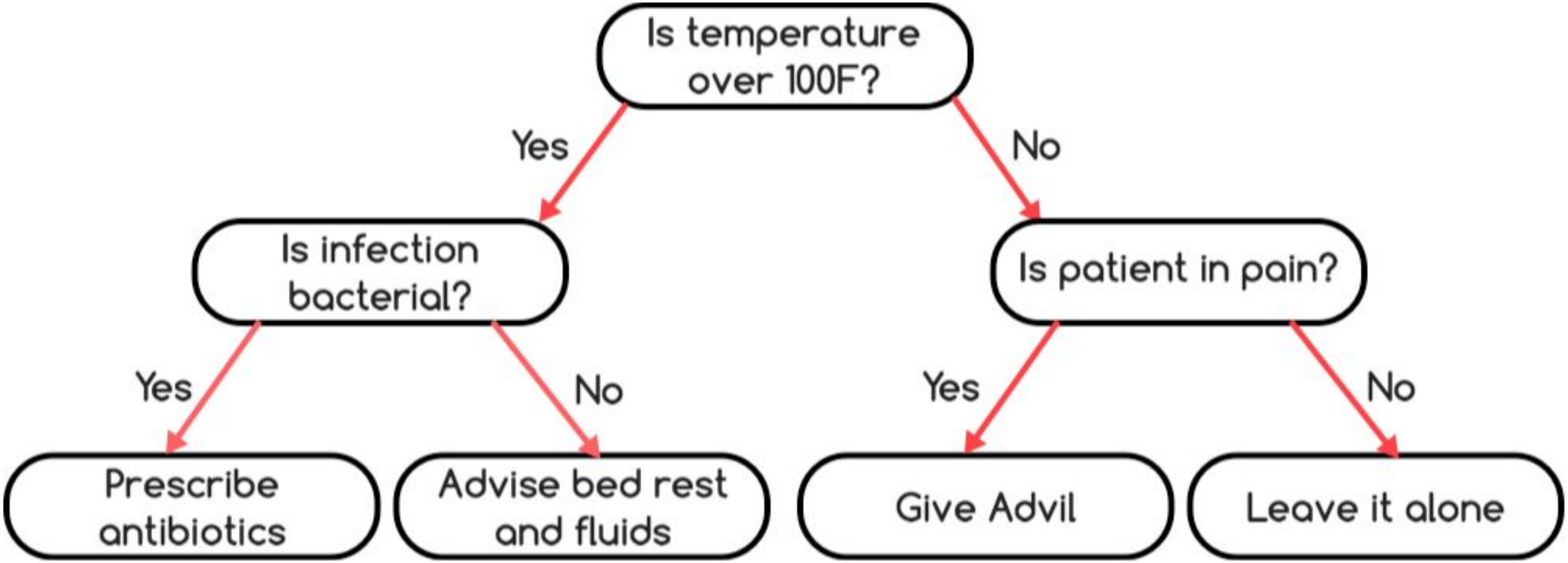
  - Others?



Unix / Windows file structure tree

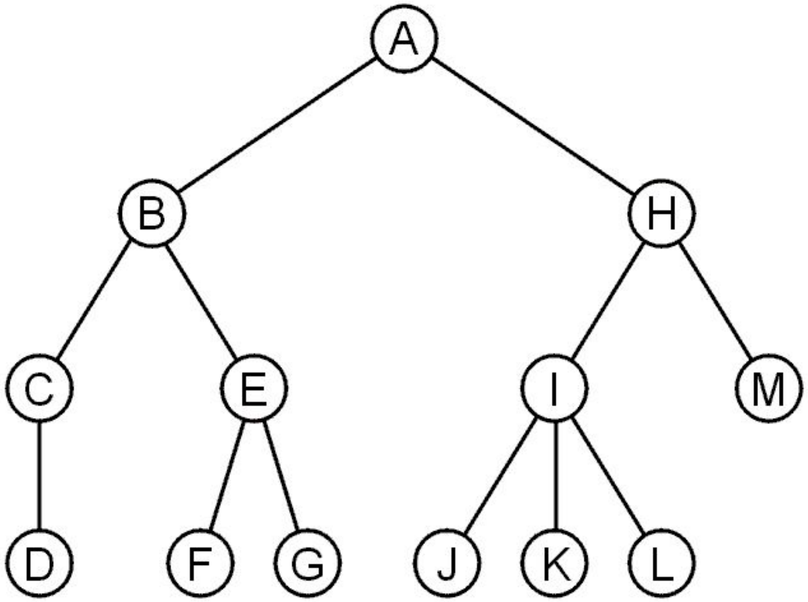


Decision Tree in in Healthcare



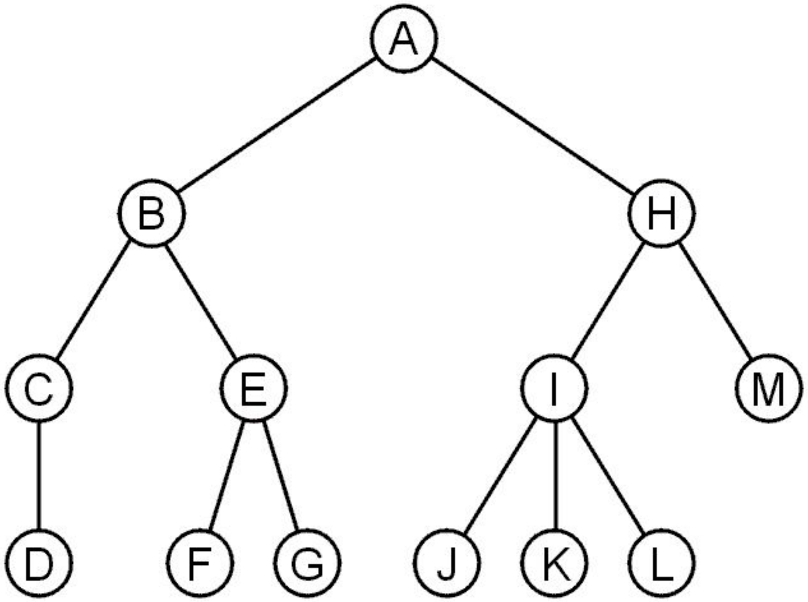
Trees - Terminologies

- Root
- Parent
- Child
- Siblings
- Ancestor (above)
- Descendents (below)
- Internal/External nodes OR
- Leave/non-leaf nodes
- Degree of a node
- Level/Depth of a node
- Height of a node



Trees

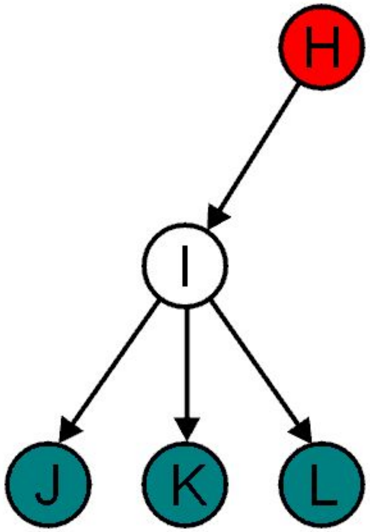
- A rooted tree data structure stores information in nodes
- Similar to lists:
  - There is a first node, or root
  - Each node has variable number of references or links to successors
  - Each node, other than the root, has exactly one node pointing to it



Terminology: Parent Child Relations

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

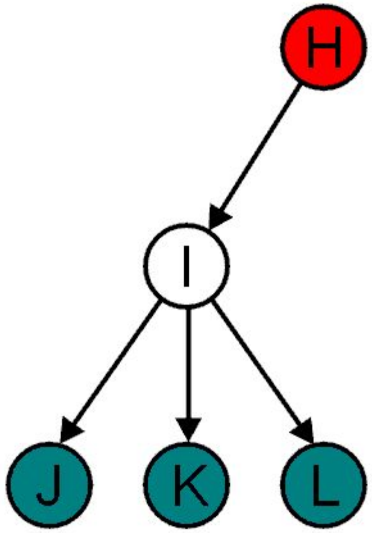
- For all nodes other than the root node, there is one parent node
  - H is the parent I



Terminology: Degree

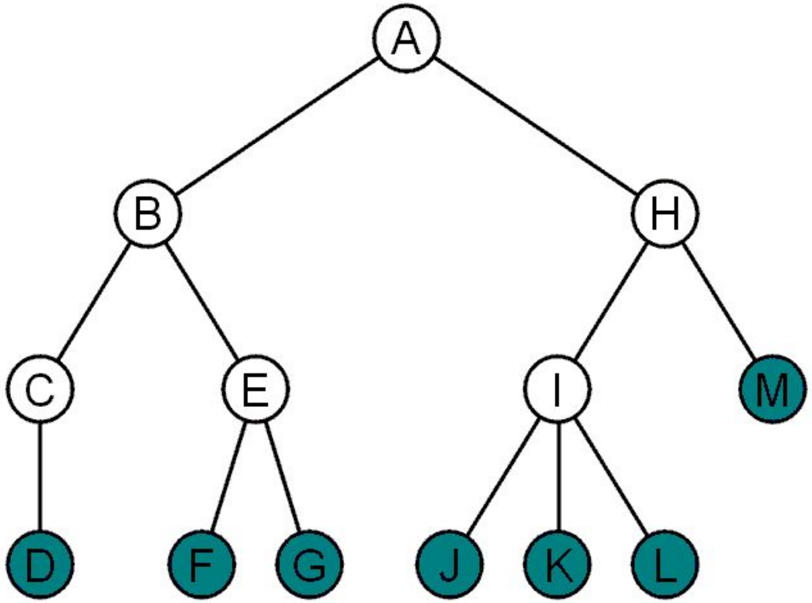
- The **degree** of a node is defined as the number of its children
  - deg(I) = 3

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



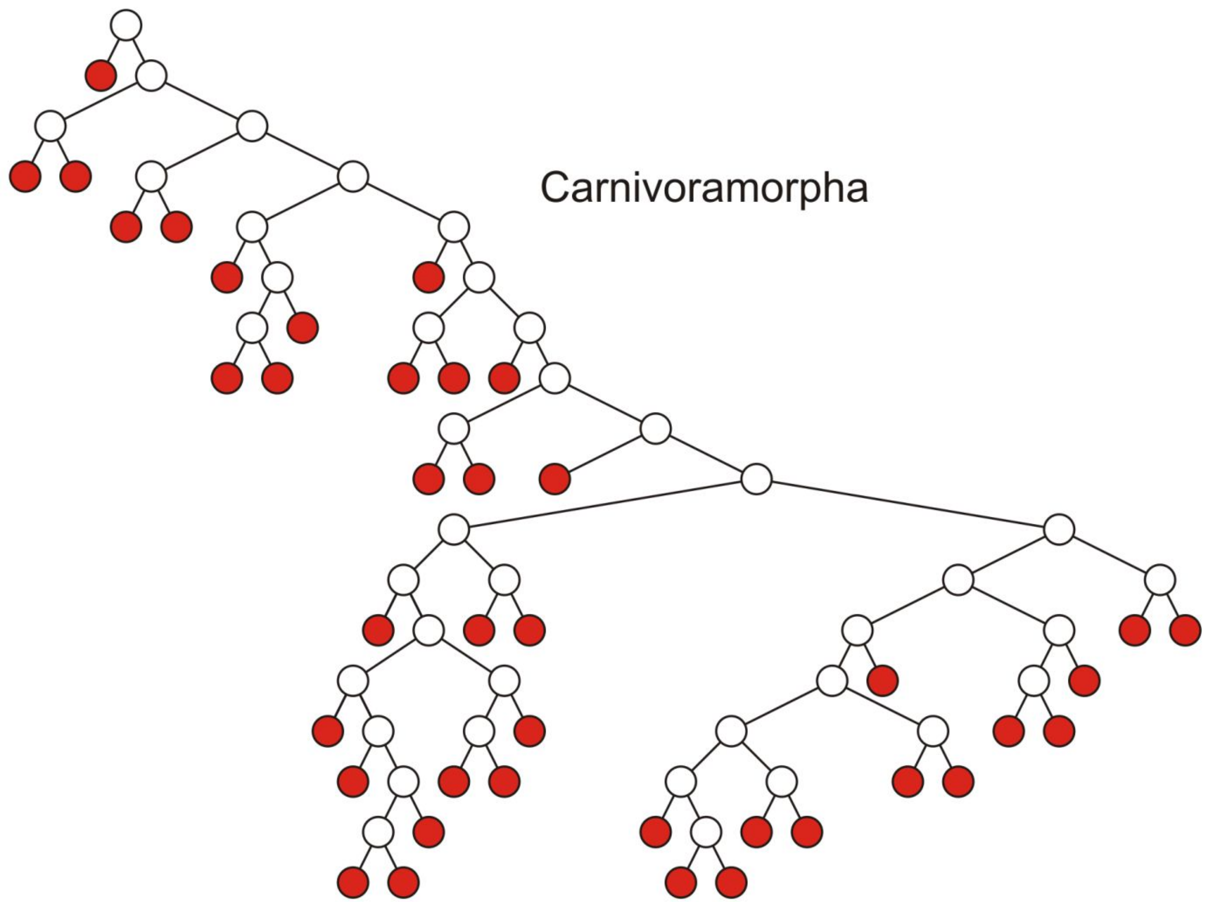
Terminology: Leaf And Internal Nodes

- 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

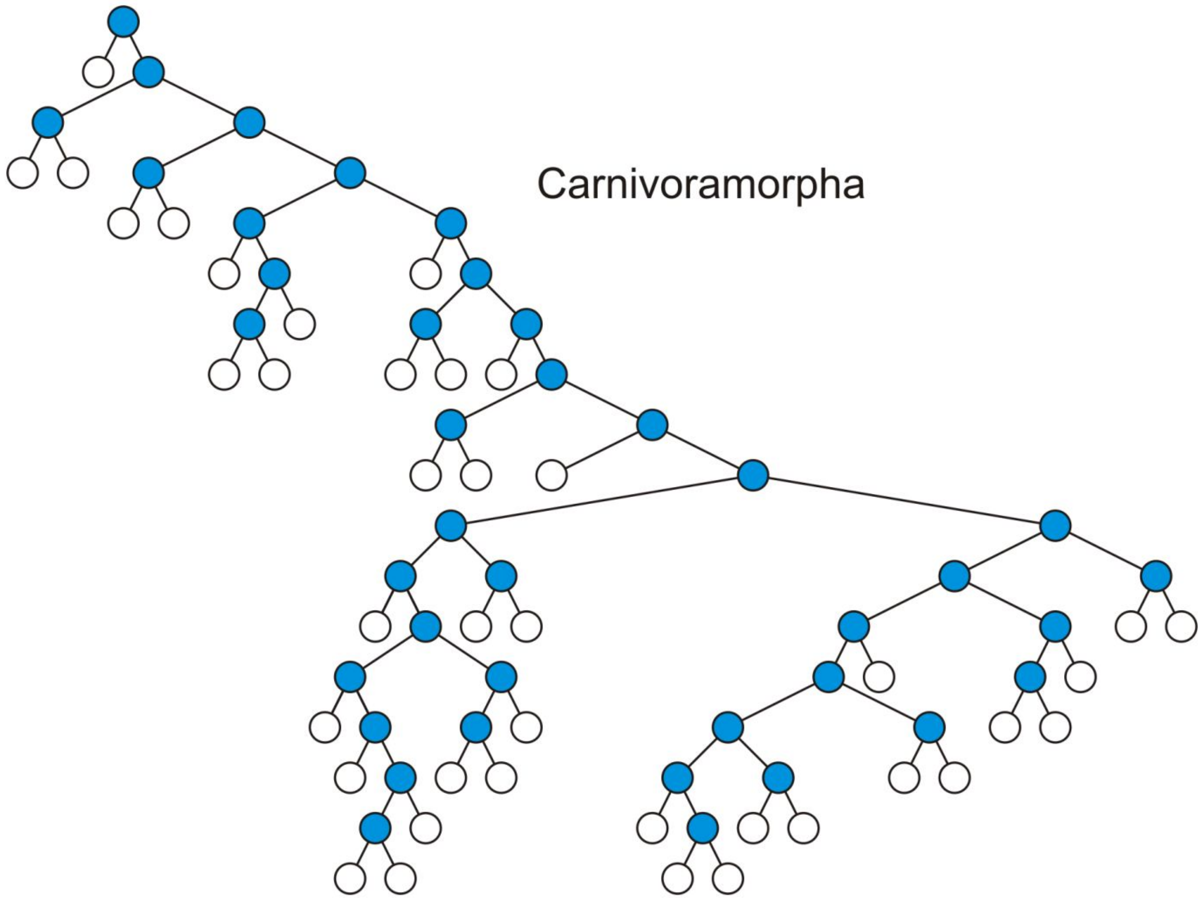
- Leaf nodes



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

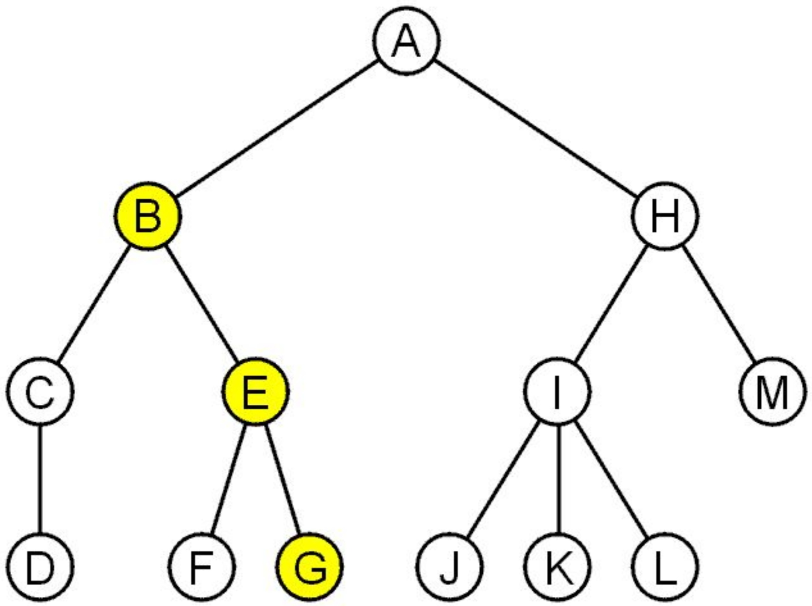


Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Terminology: Path

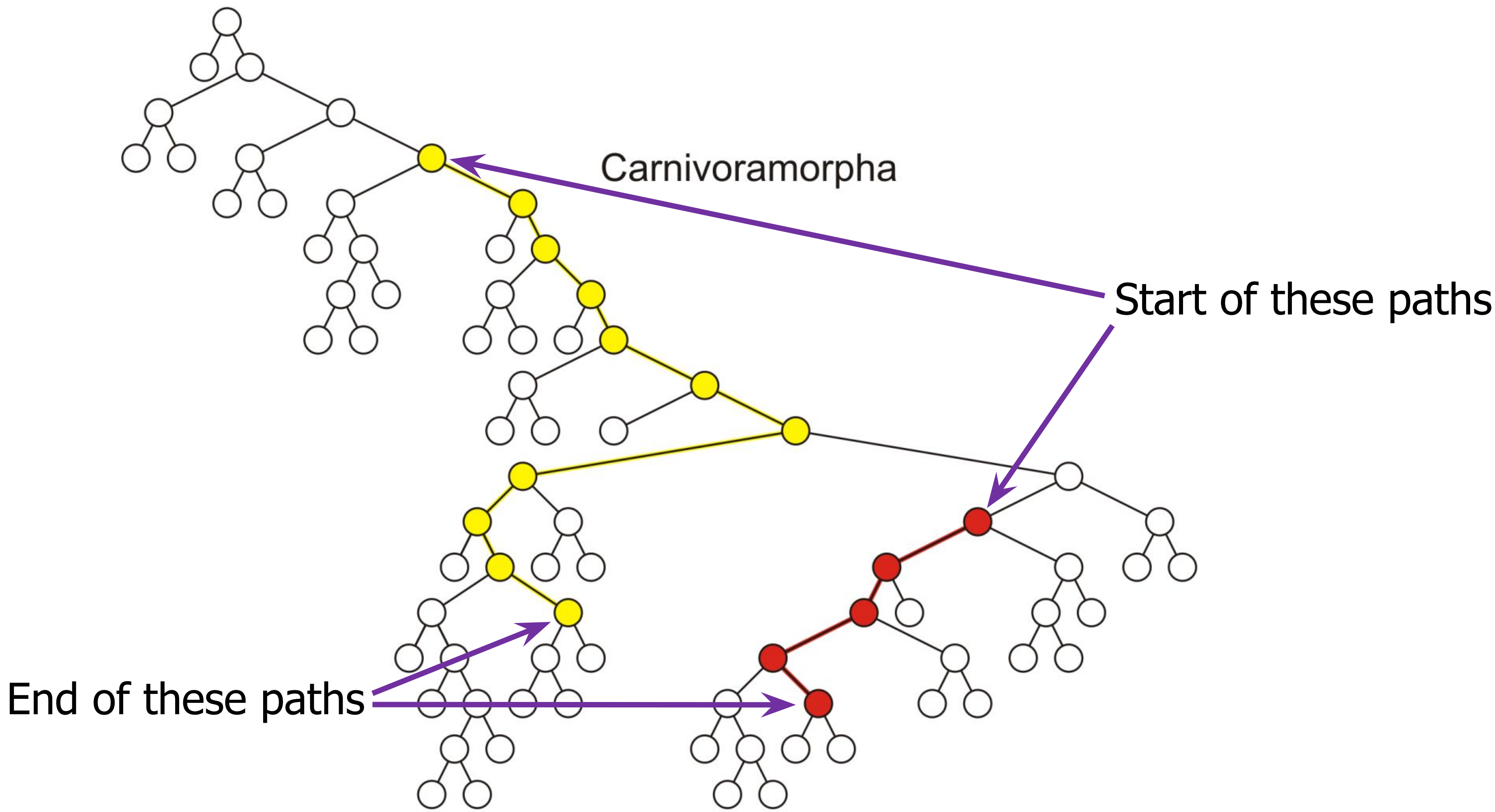
- A path is a sequence of nodes $(a_0, a_1, \ldots, a_n)$
  - Where a_{k+1} is a child of a_k

- The length of this path is ***n***
  - For example, the path (B, E, G) has length 2



Terminology: Path Example

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

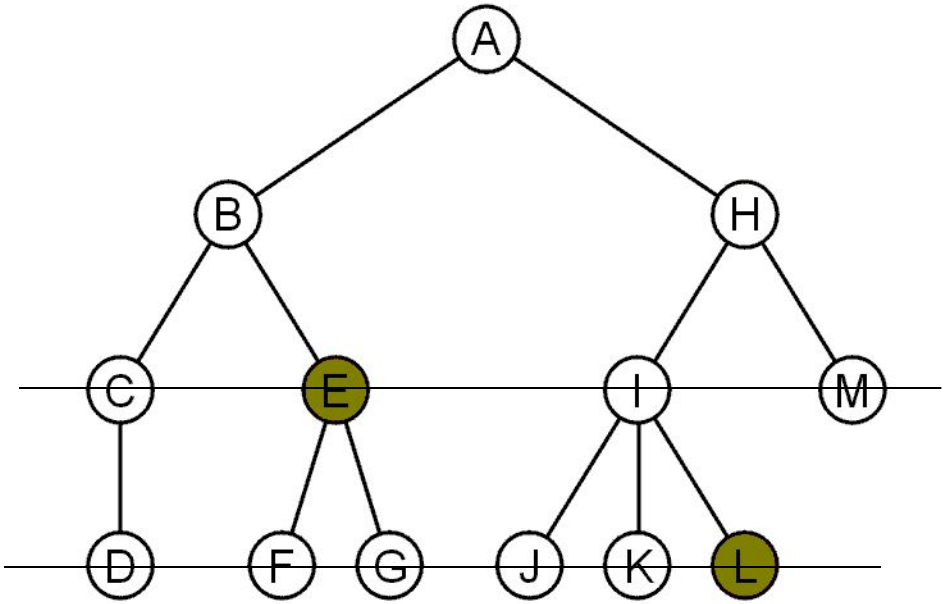


Wesley-Hunt, G. D.; Flynn, J. J. “Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of ‘Miacoidea’

Terminology: Depth (or Level) of a node

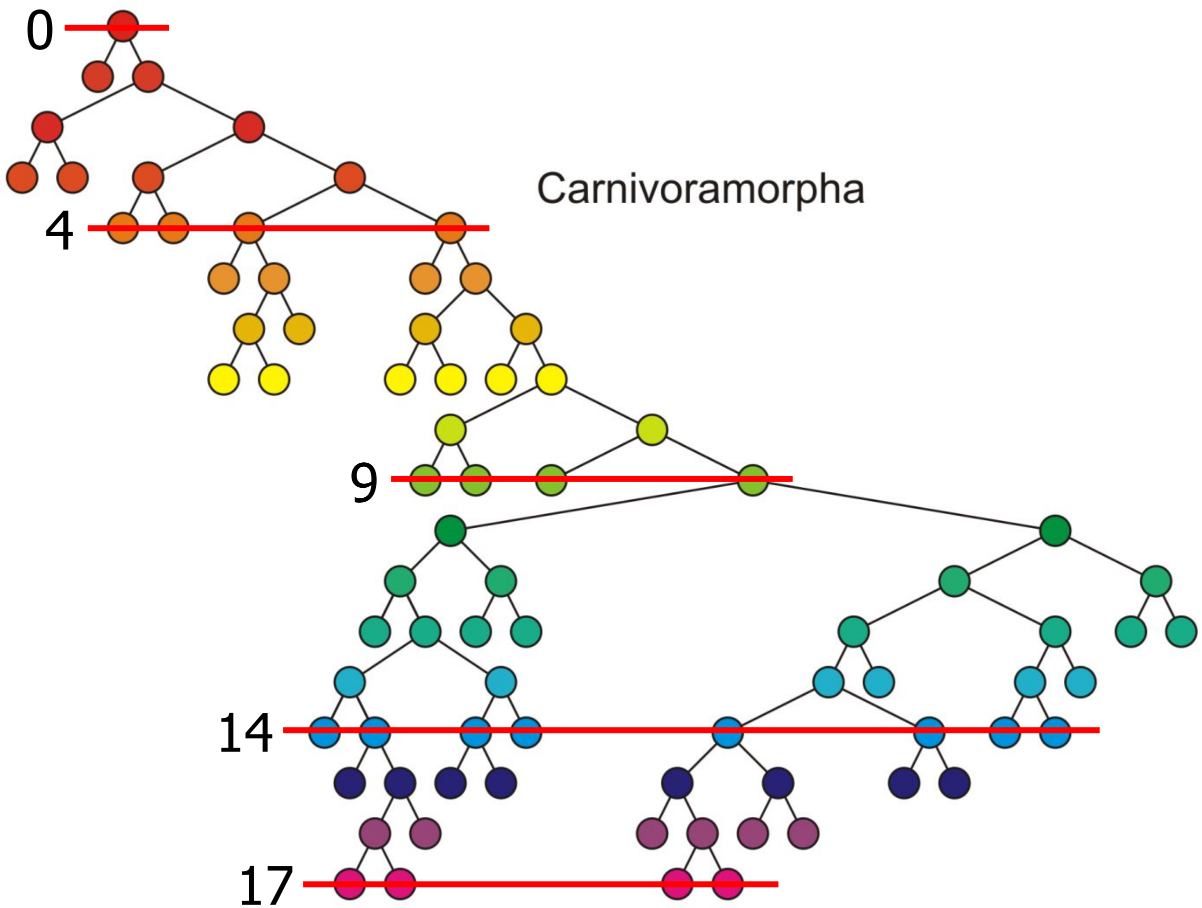
- For each node in a tree, there exists a unique path from the root node to that node

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

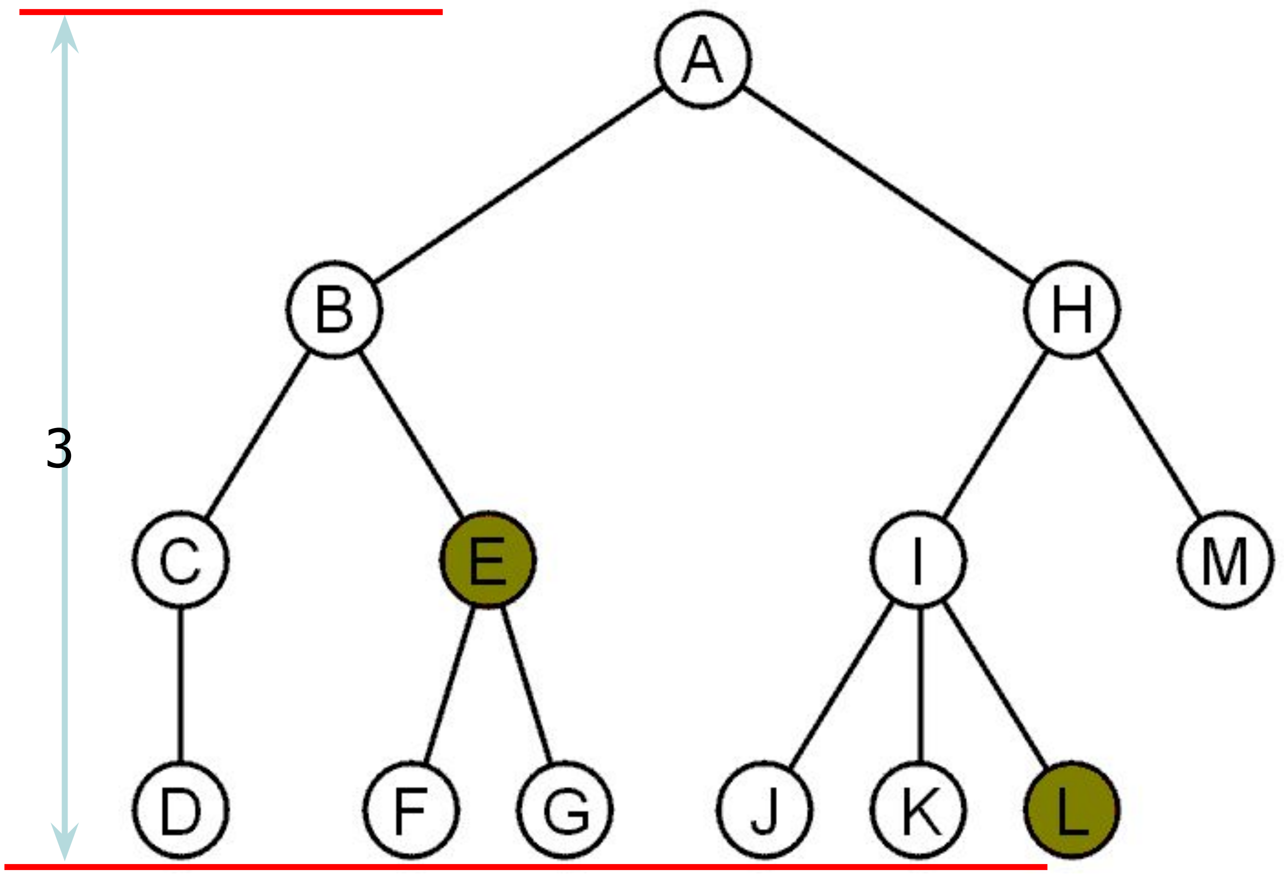
- The **height of a tree** is defined as the maximum depth or level of any node within the tree

- 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 3

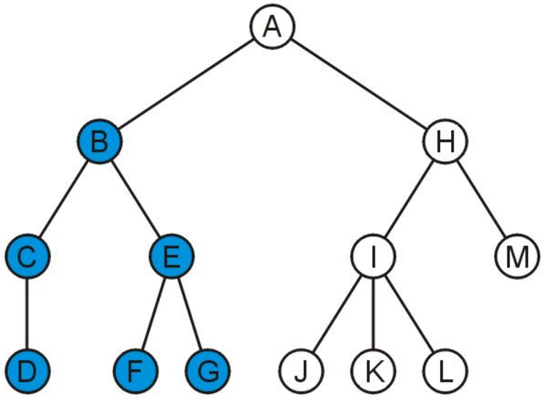


Terminology: Ancestors And Descendants

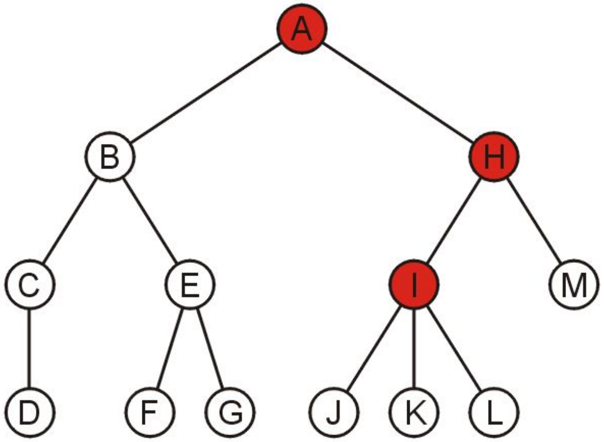
- If a path exists from node *a* to node *b*
  - *a* is an ancestor of b
  - *b* is a descendent of a
- Thus, a node is both an ancestor and a descendant of itself
  - We can add the adjective **strict** to exclude equality
  - *a* is a **strict descendent** of *b* if *a* is a descendant of *b* but a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

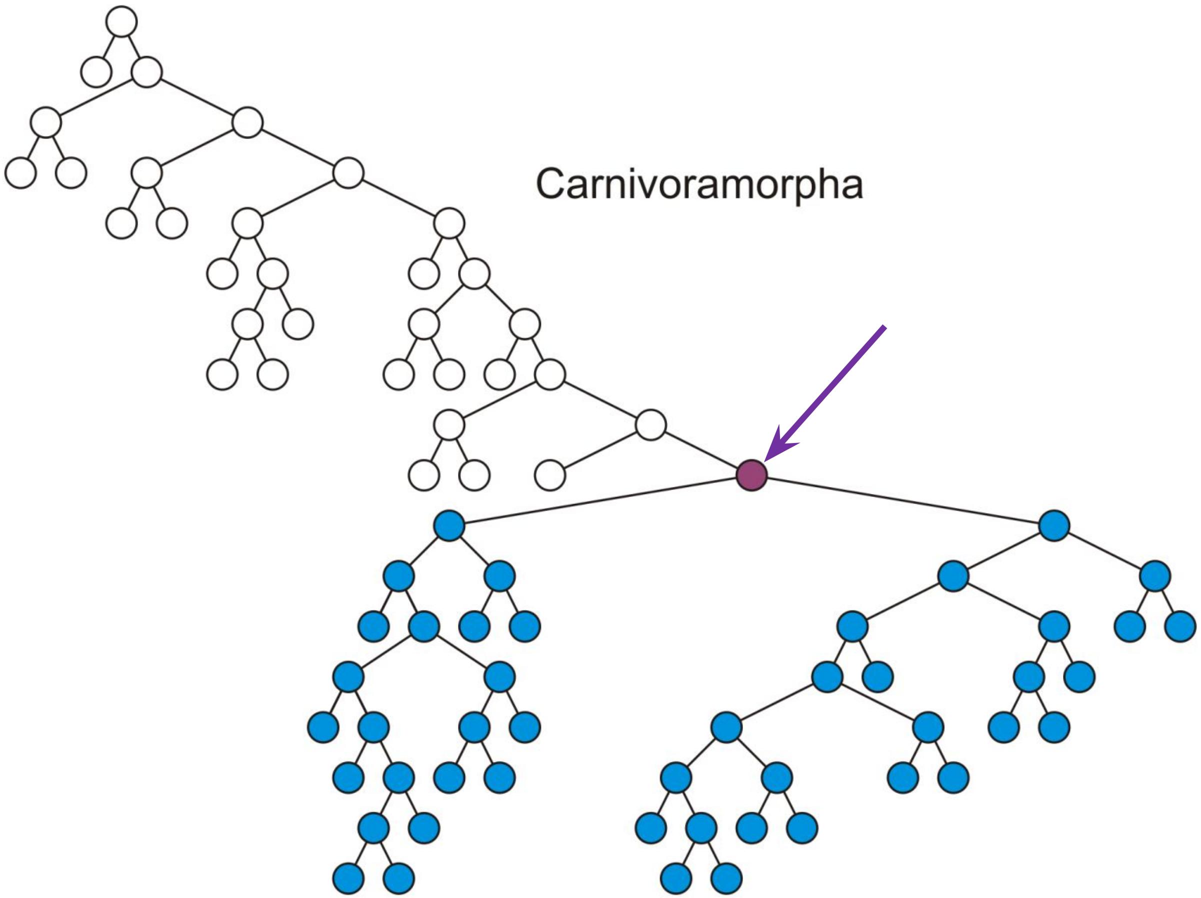


- The ancestors of node I are H and A



Terminology: Descendants Example

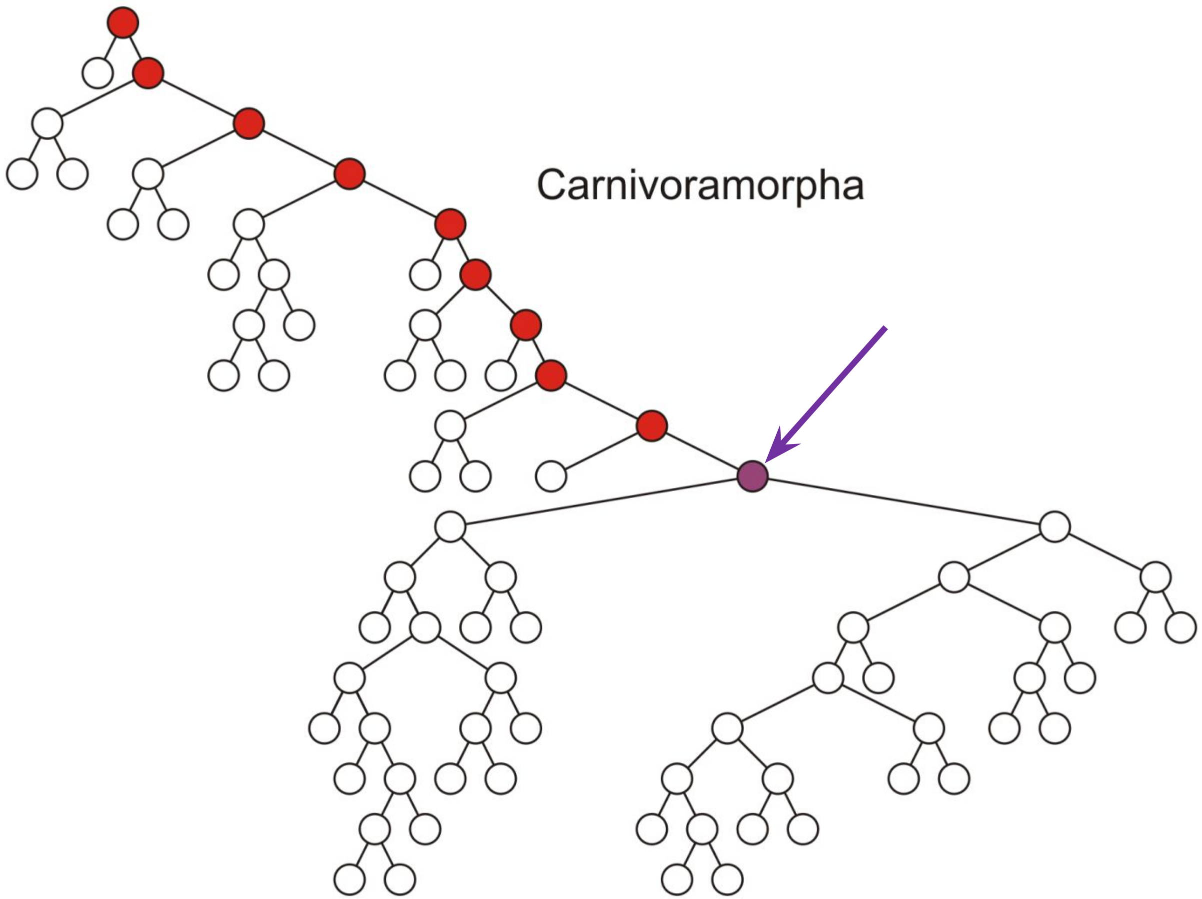
- All descendants (including itself) of the indicated node



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

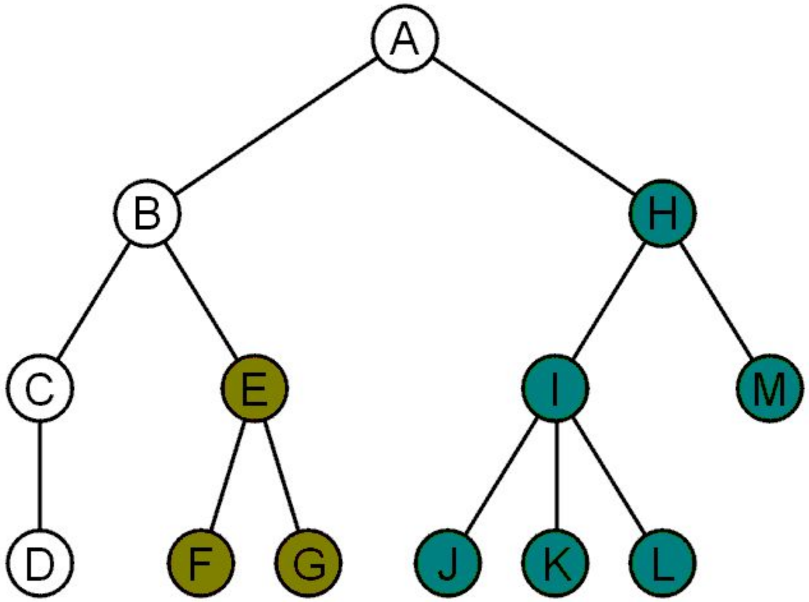
- All ancestors (including itself) of the indicated node



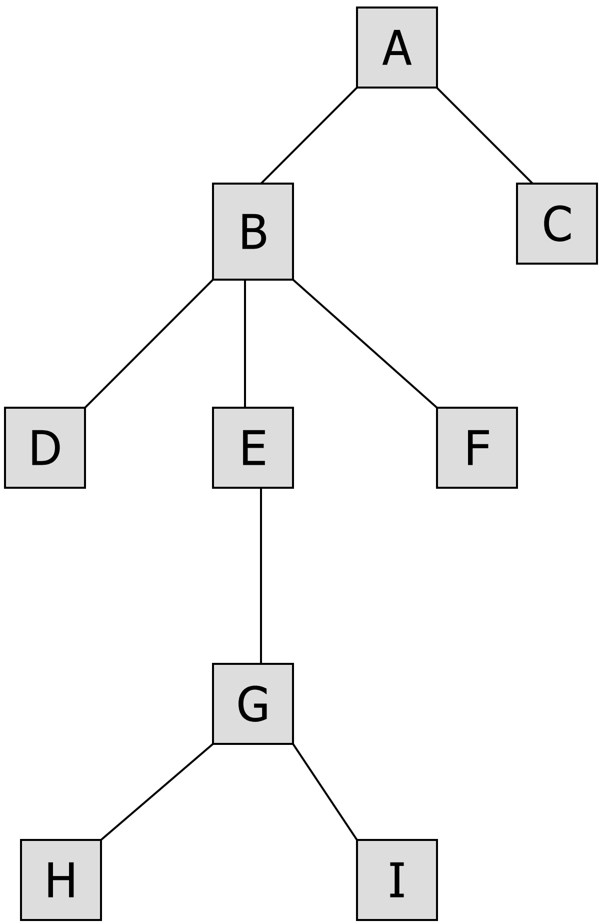
Wesley-Hunt, G. D.; Flynn, J. J. "Phylogeny of the Carnivora: basal relationships among the Carnivoramorphans, and assessment of the position of 'Miacoidea'

Terminology: Sub-Tree

- 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 (here n>1)
  - 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>
   <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

heading

```
<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>
```

body of page

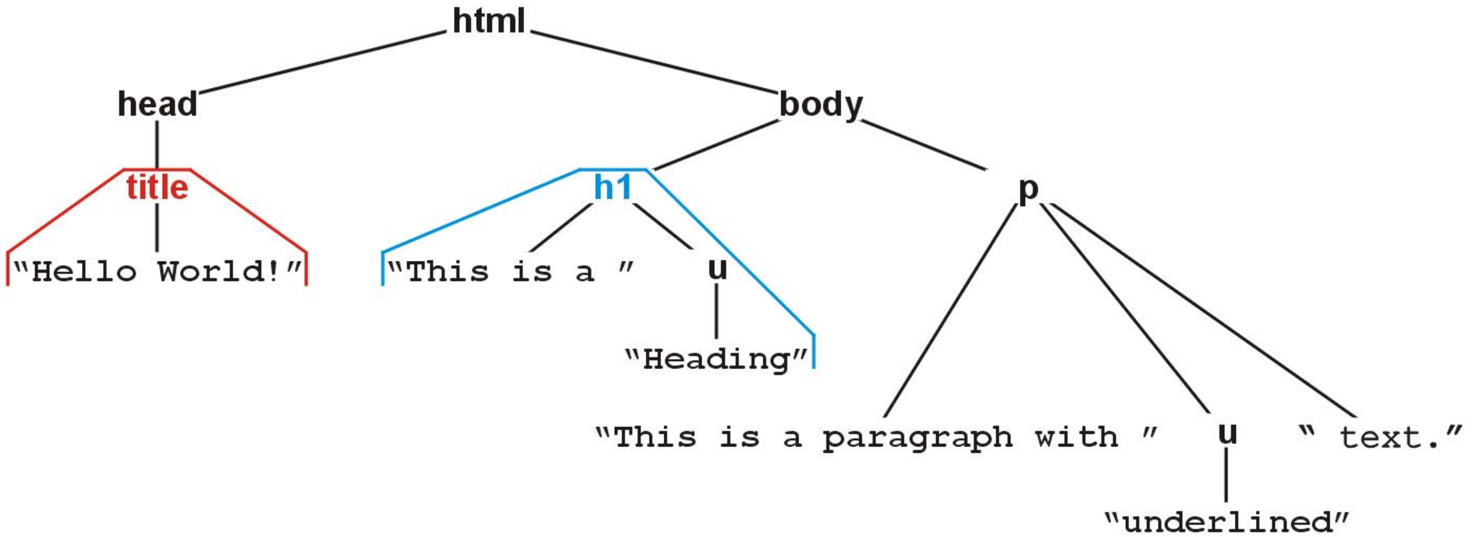
underlining

paragraph

Example: HTML (3)

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

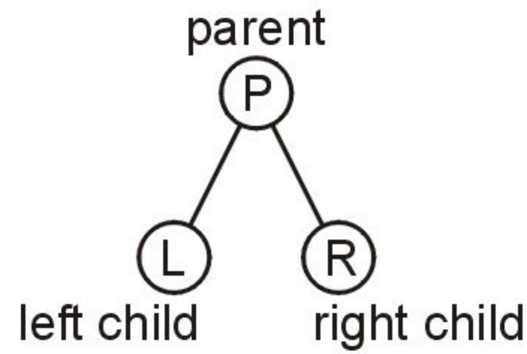
```
<html>
   <head>
      <title>Hello World!</title>
   </head>
   <body>
      <h1>This is a <u>Heading</u></h1>

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



Binary Tree

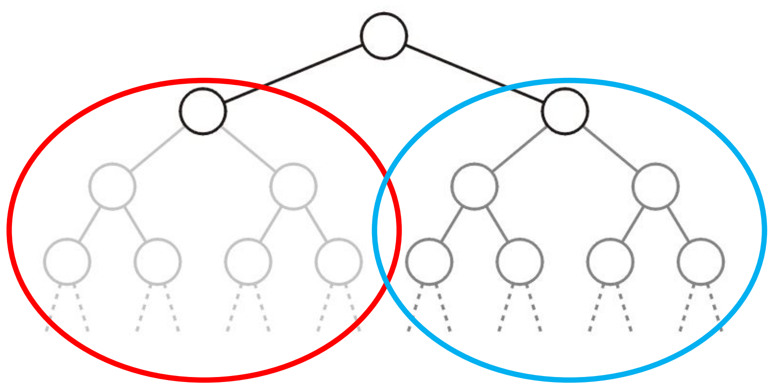
Binary Tree

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



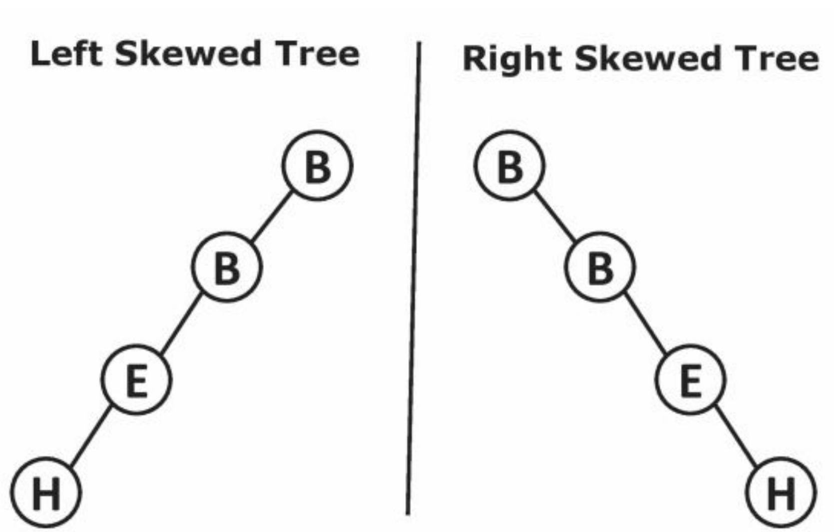
- Deg(tree) = 2
- Children = {0,1,2}

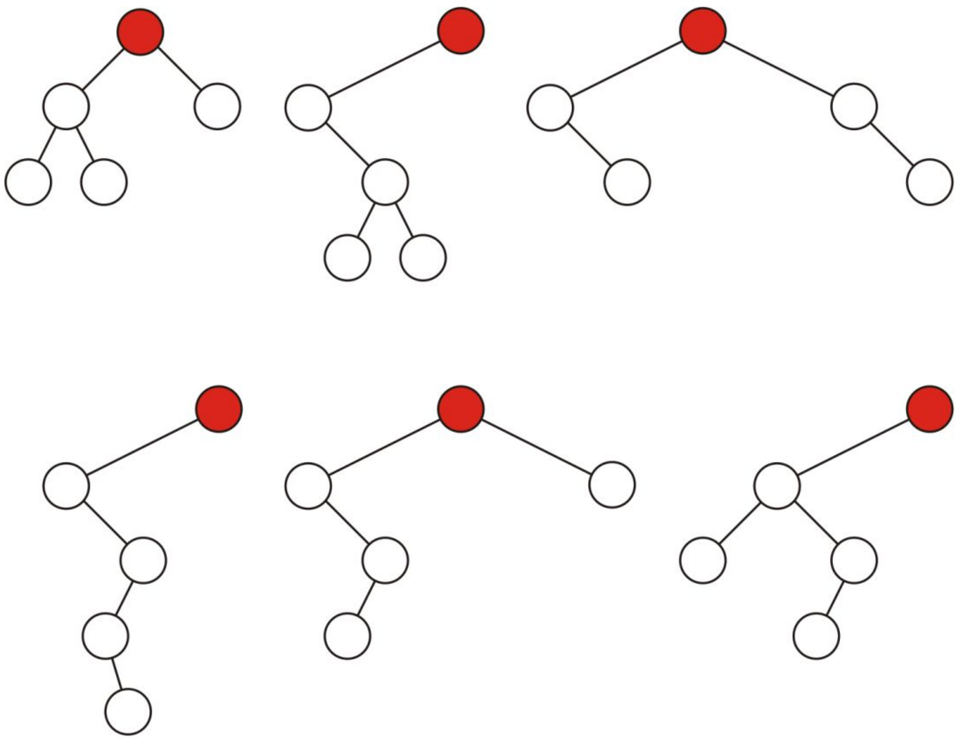
- Likewise, the two sub-trees are referred as
  - Left sub-tree
  - Right sub-tree



Binary Tree: Example

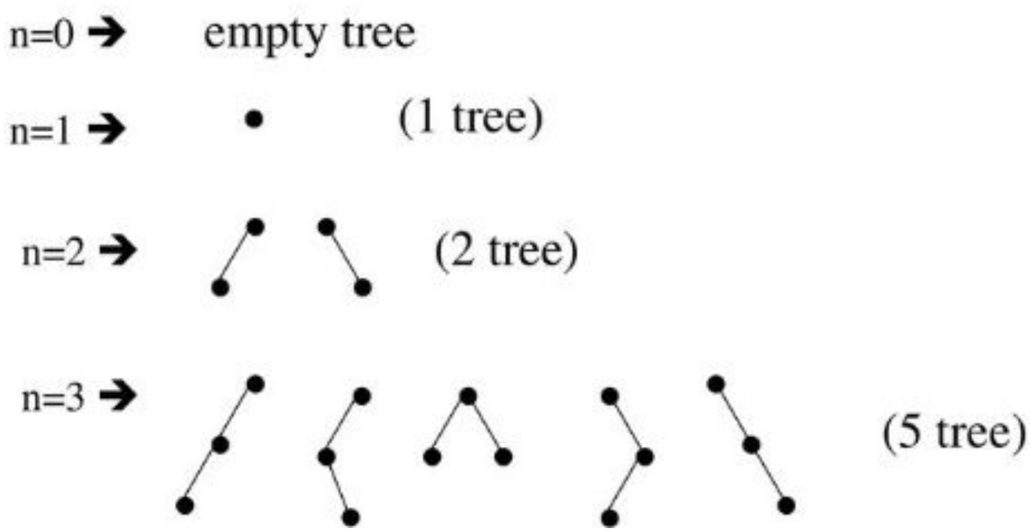
- Some variations on binary trees with five nodes





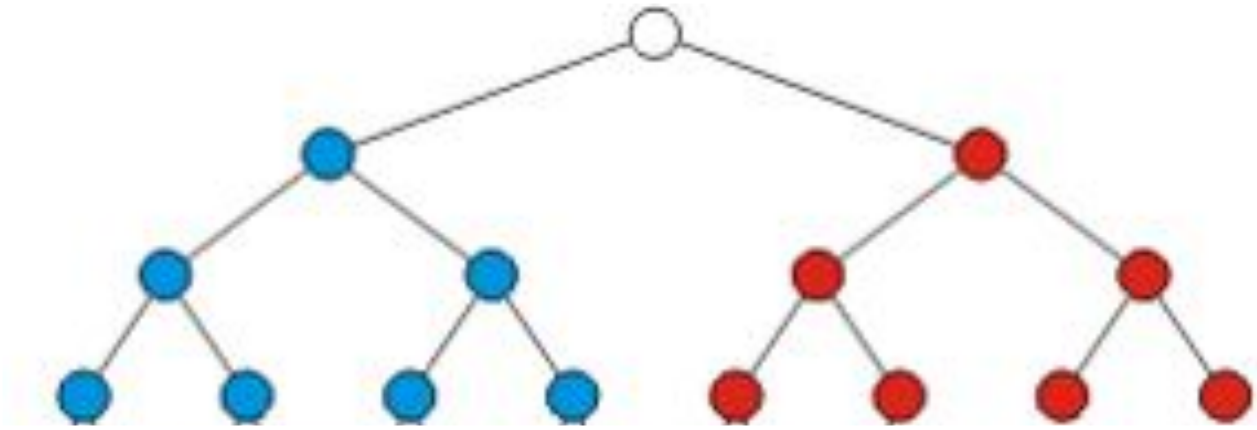
No. of Possible Binary Trees?

- How many binary trees can be formed for a given set of nodes n?
- T(3) = 5
- T(4) = 14
- T(5) = ?
- T(n) = $\dfrac{2n!}{(n+1)! * n!}$



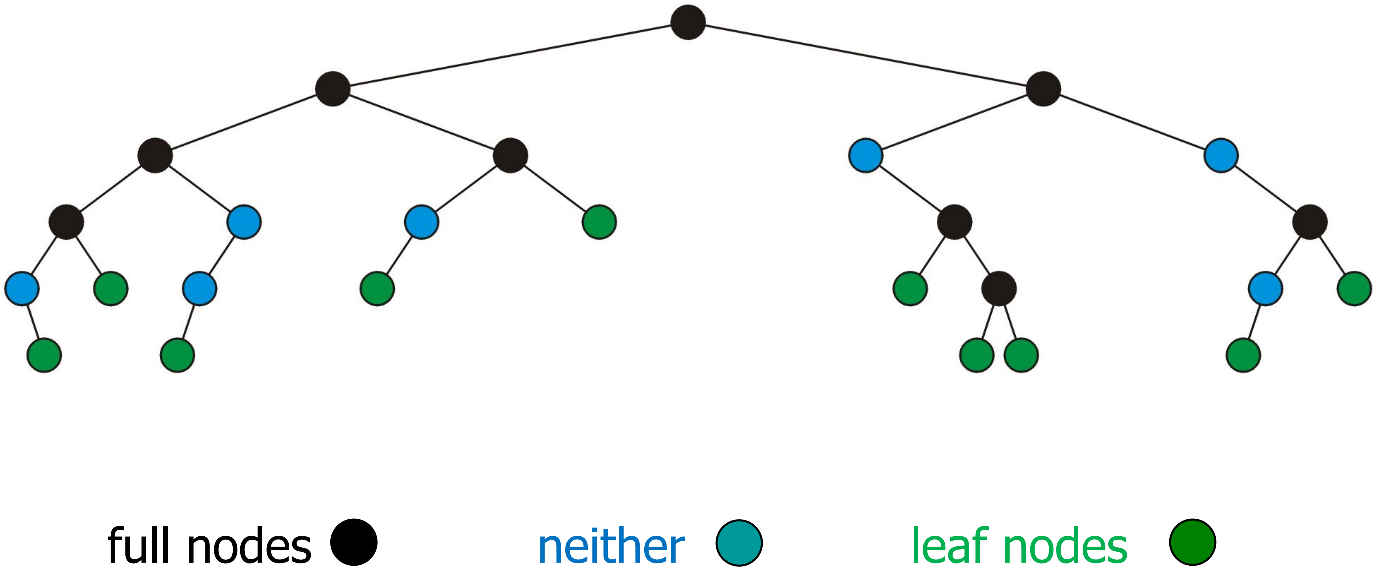
Sum of Internal and External nodes

- Sum of all the internal and external nodes is always equal to total no. of nodes in a binary tree
  - deg(0) = 8
  - deg(1) = 0
  - deg(2) = 7
  - total nodes = 8 + 0 + 7 = 15



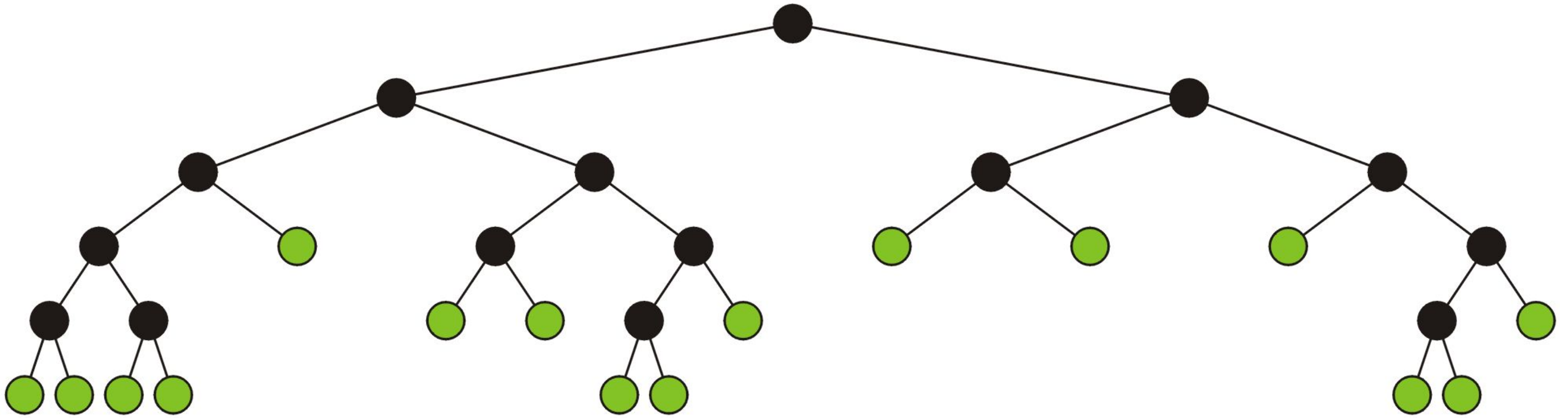
Binary Tree: Full **Node**

- A full node is a node where both the left and right sub-trees are non-empty trees
- (OR) if it has exactly two child nodes



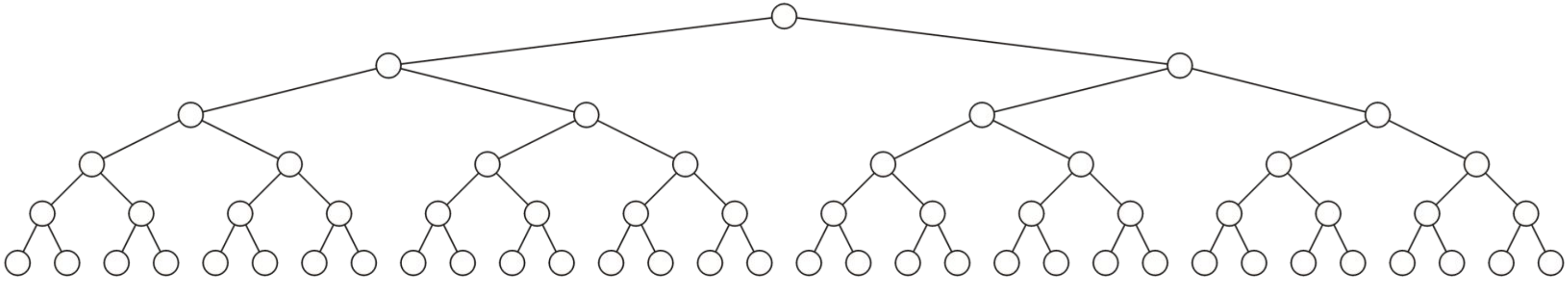
Full/Proper/Strict Binary Tree

- A full binary tree is where each node has { 0 or 2 } childrens
  - A full node, or
  - A leaf node
- Full binary tree is also called **proper binary tree**, **strictly binary tree** or **2-tree**



Complete/Perfect Binary Tree (CBT)

- A perfect binary tree of height h is a binary tree where:
  - All leaf nodes have the same depth or level *L*
  - All other nodes are full-nodes
- Each level must be completely filled i.e. 2^h nodes



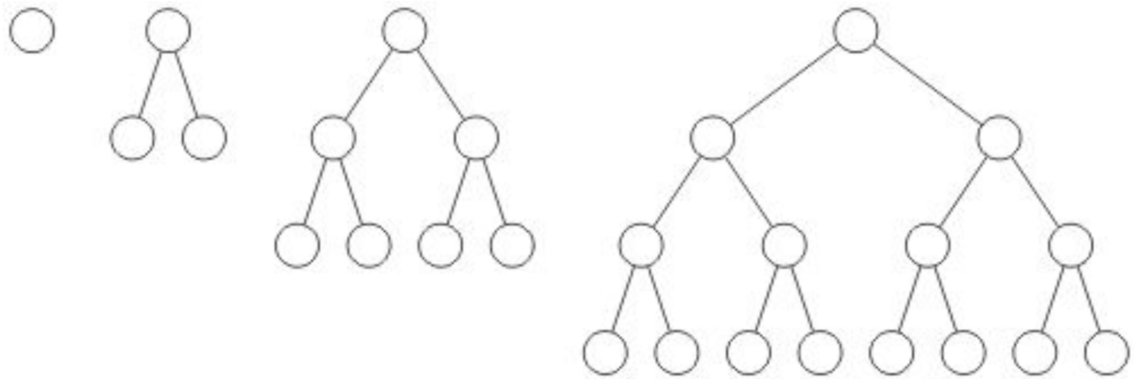
- Is it a Full binary tree as well?

Perfect Binary Tree: Recursive Definition

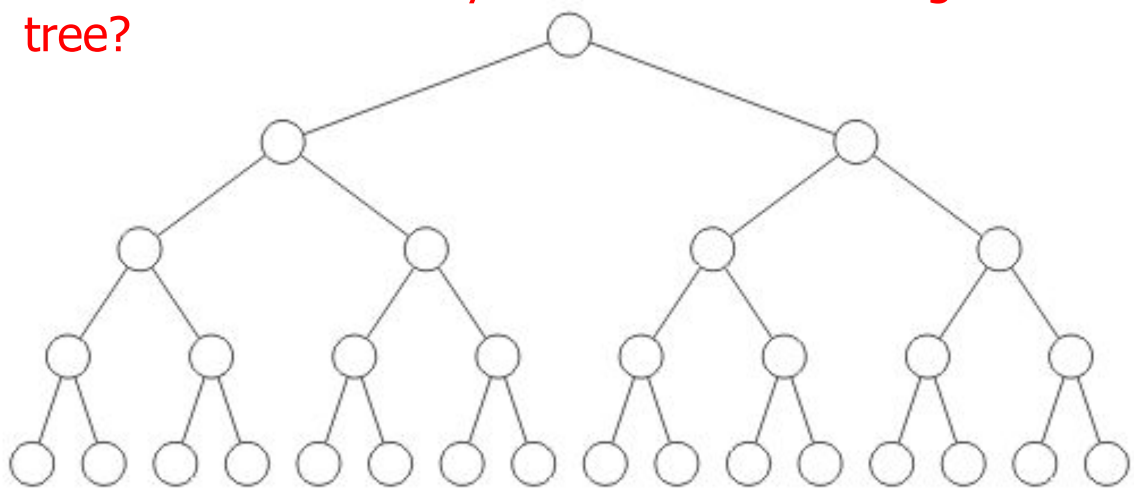
- A binary tree of height $h = 0$ is perfect
- A binary tree with height $h > 0$ is perfect
  - If both sub-trees are prefect binary trees of height $h - 1$

Perfect Binary Tree: Example

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

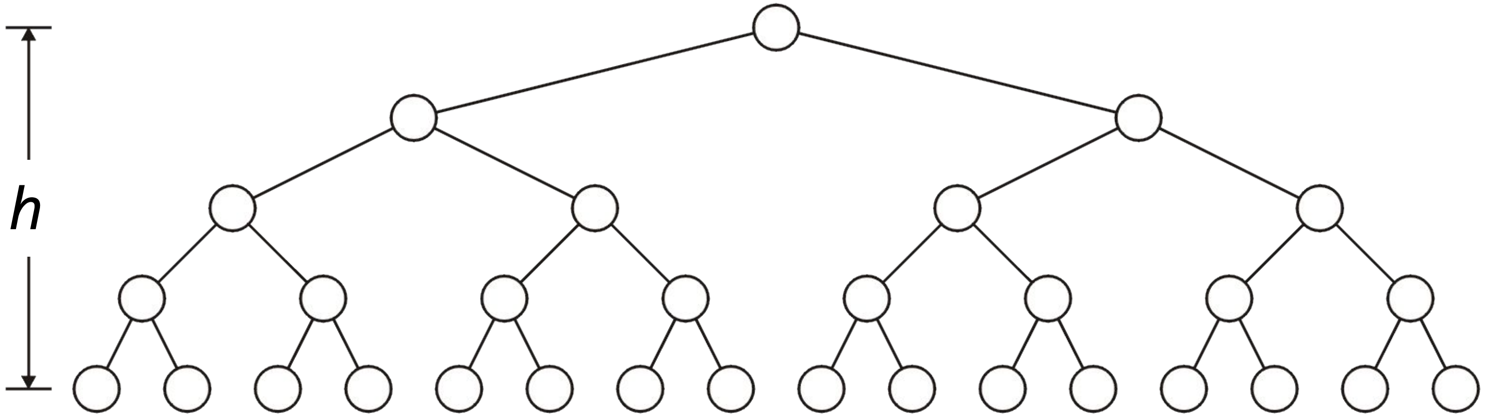


No of leaf nodes? Any relation with the height of the tree?



Binary Tree: Properties (1)

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

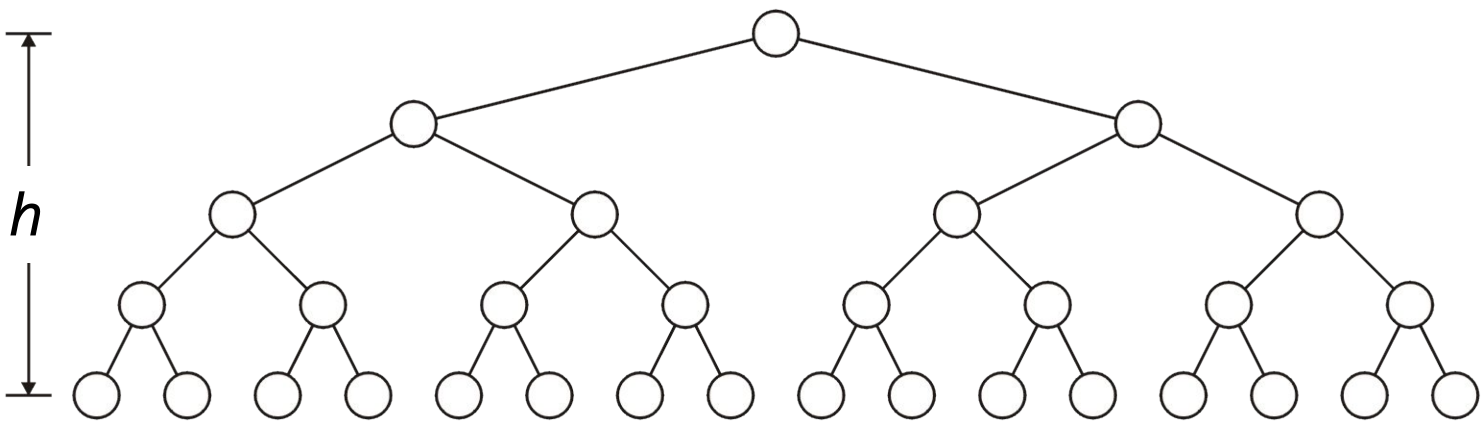


Binary Tree: Properties (2)

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

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

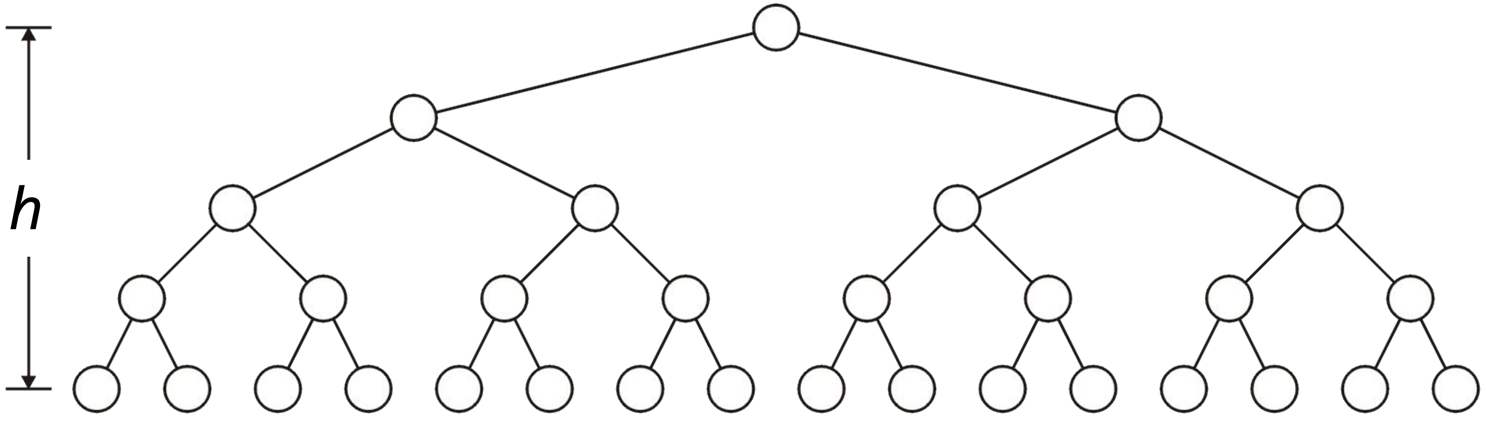
$$n = 2^0 + 2^1 + 2^2 + \ldots + 2^h = \sum_{j=0}^{h} 2^j = 2^{h+1} - 1$$



Binary Tree: Properties (3)

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

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

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

- A perfect 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 perfect binary tree with n nodes has height $\log_2(n + 1) - 1$

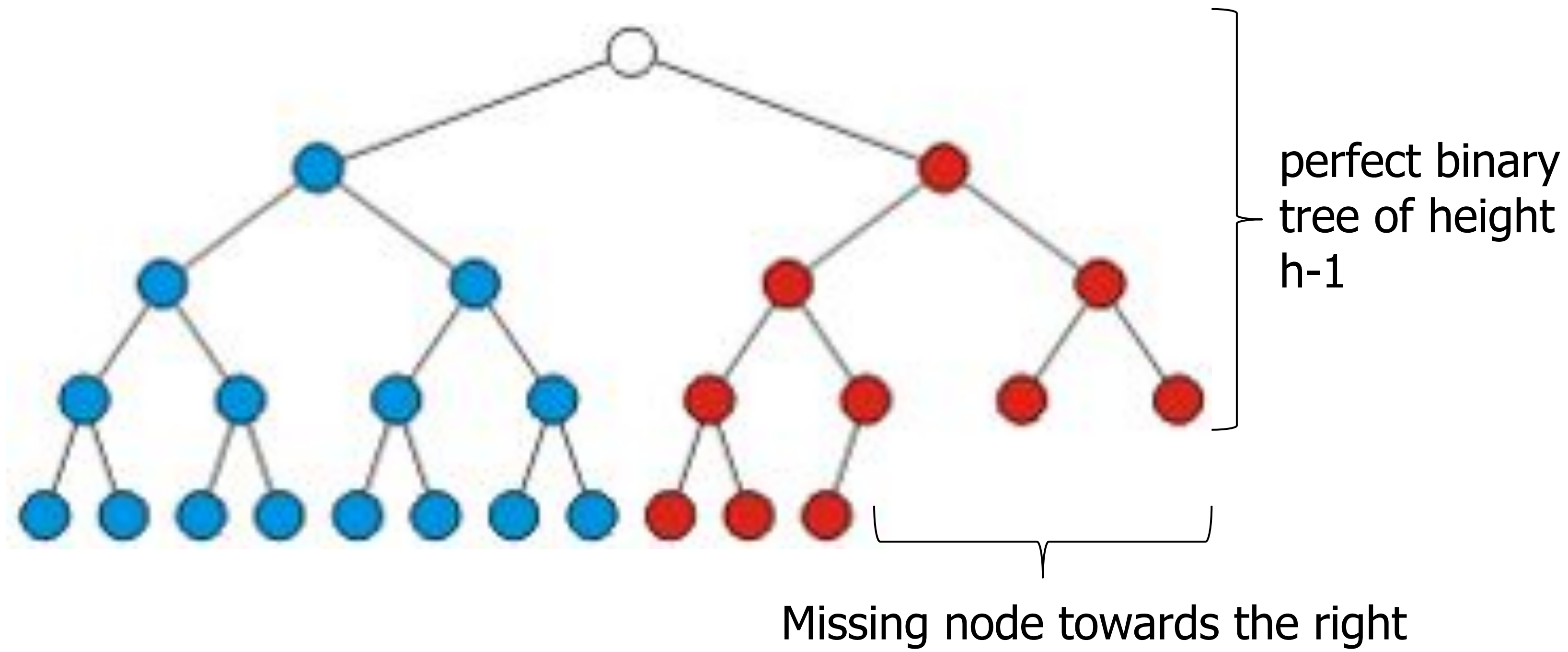
- **Number n of nodes in a binary tree of height h is at least $h+1$ and at most $2^{h+1} - 1$**

Height vs No. of Nodes?

- If you know 'height' then what will be the no. of nodes?
  - minimum no. of nodes = h+1
  - maximum no. of nodes = $2^{h+1} - 1$
- If you know the 'no. of nodes' then what will be the height?
  - minimum height = $\log_2(n + 1) - 1$
  - maximum height = n-1

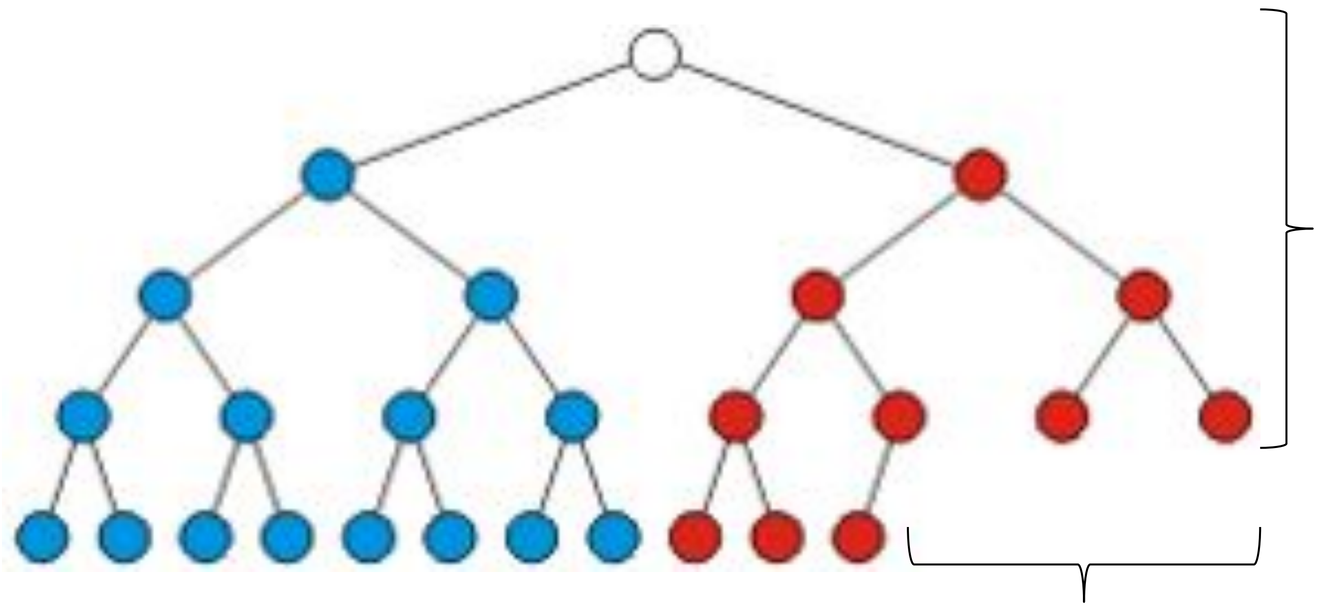
Almost (or Nearly) Complete Binary Tree

- 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

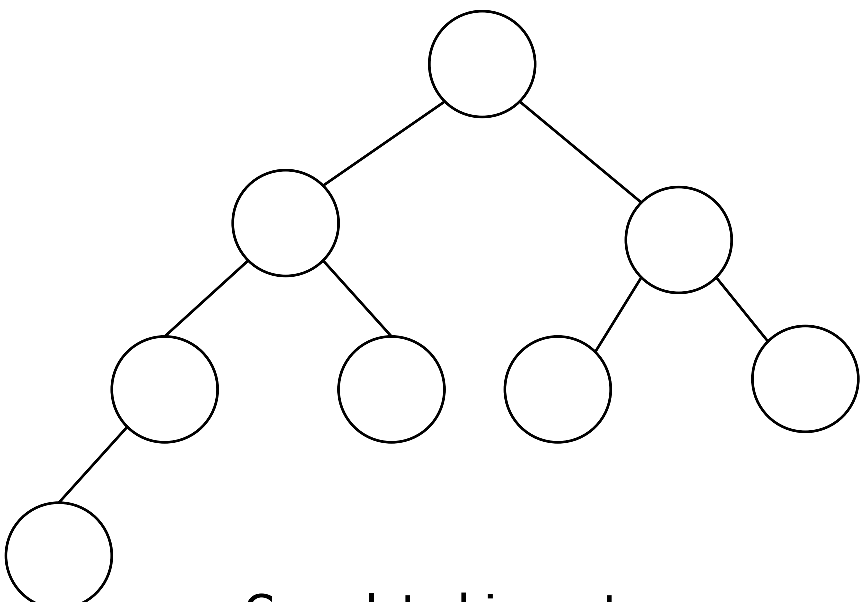
- Almost complete binary tree of height h is a binary tree in which
  1. There are 2^d nodes at depth d for $d = 1,2,...,h-1$
     - Each leaf in the tree is either at level h or at level $h-1$
  2. **The nodes at depth h are as far left as possible (Formal ?)**



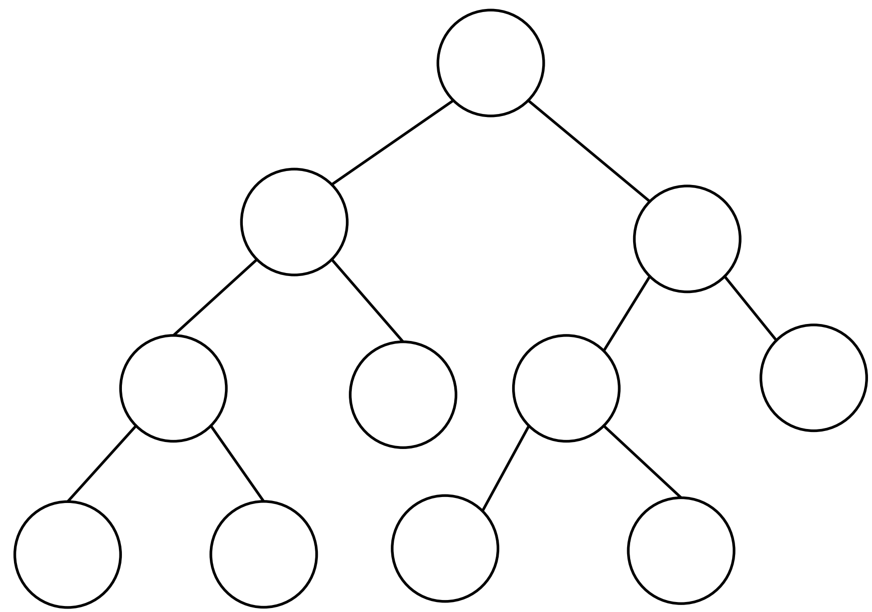
Almost (or Nearly) Complete Binary Tree

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

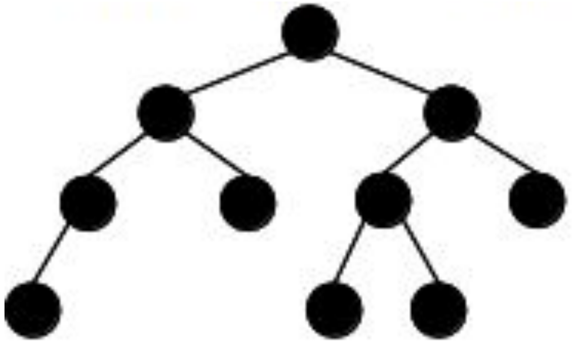


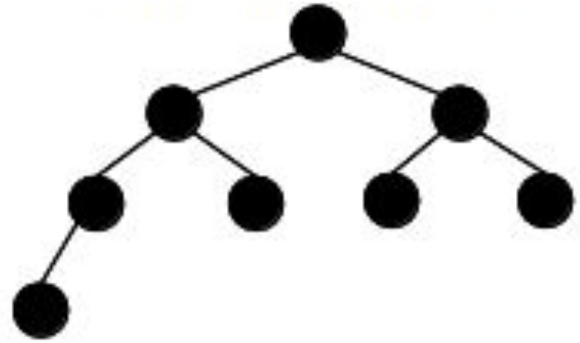
Complete binary tree

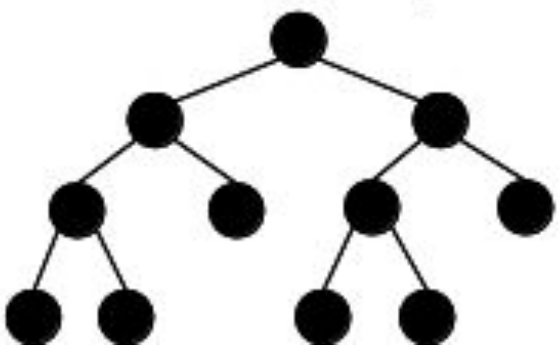


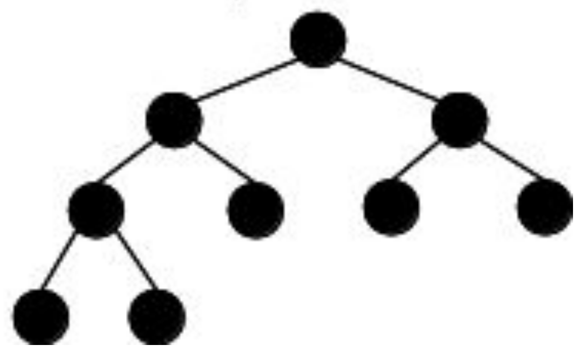
Not a Complete binary tree
(condition 2 violated)

Full vs. Almost Complete Binary Tree









Almost Complete Binary Tree: Properties

- Total number of nodes `n` are between
  - perfect binary tree of height `h-1 + 1(1 in the next level)`, i.e., $2^h - 1 + 1 = 2^h$ nodes
  - perfect binary tree of height `h`, i.e., $2^{h+1} - 1$ nodes
- Height `h` is the largest integer less than or equal to $\mathit{Log}_2(n)$

(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

Any Question So Far?