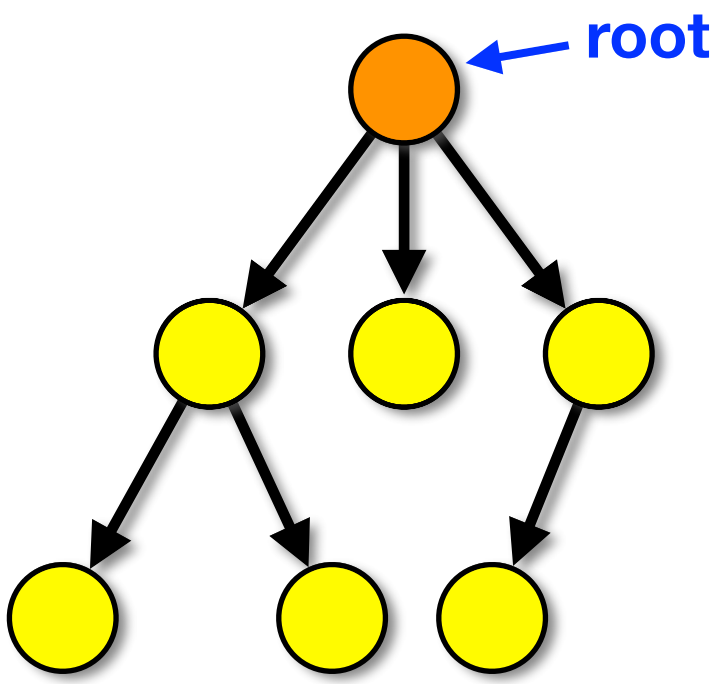
CS350: Data Structures

Trees / Binary Trees

James Moscola
Department of Engineering & Computer Science
York College of Pennsylvania

Trees

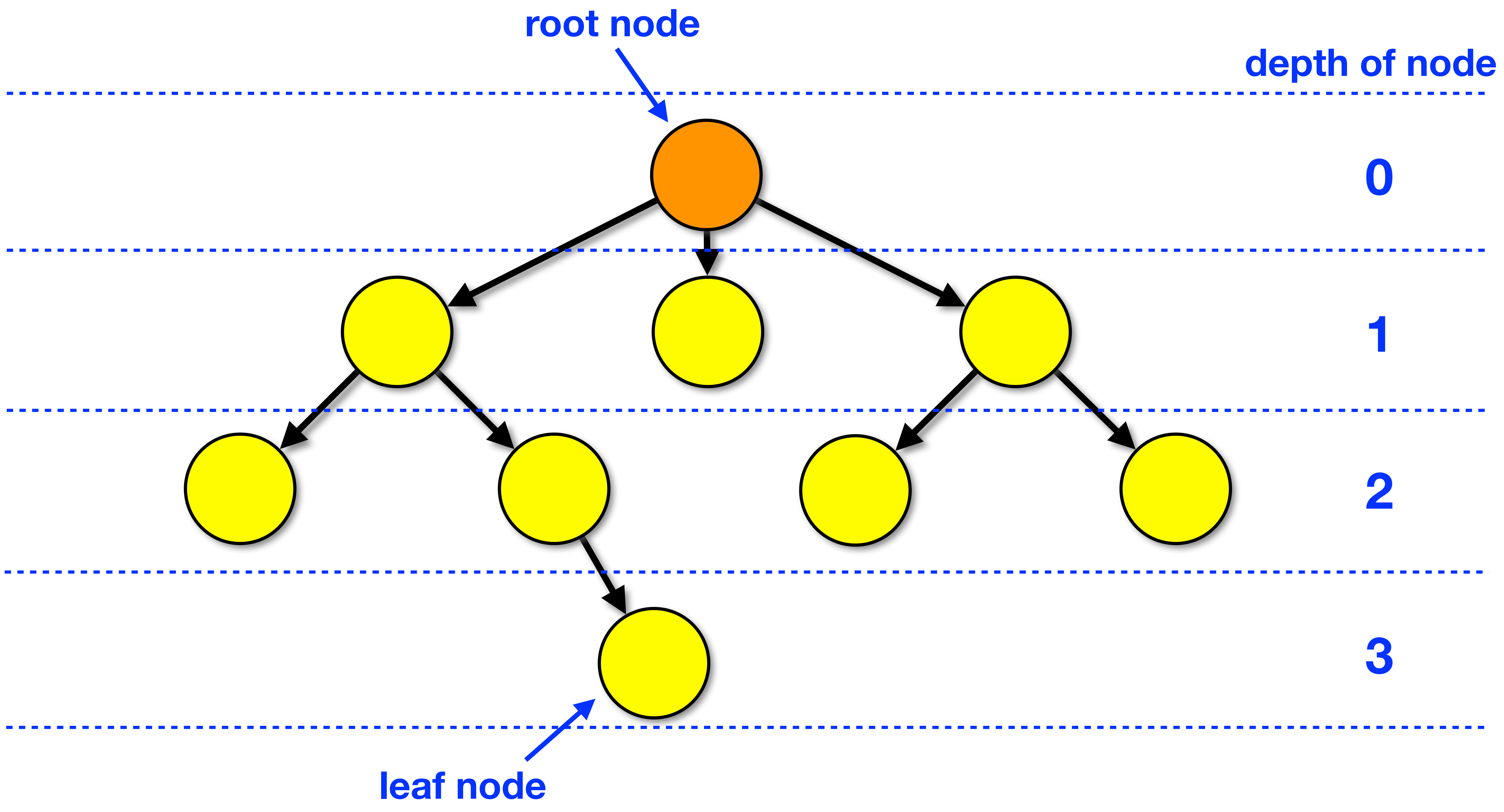
- **A tree consists of a set of nodes and a set of directed edges that connect pairs of nodes**
- **A tree has the following properties:**
  - A single node that is distinguished as the root
  - Every node *c*, except the root, has a single incoming edge from one other node *p*
    - *p* is the parent of *c*
    - *c* is the child of *p*
  - A single path exists from the root to each node in the tree (no cycles in the tree)
  - A tree node has zero or more subtrees
    - Subtrees may be empty



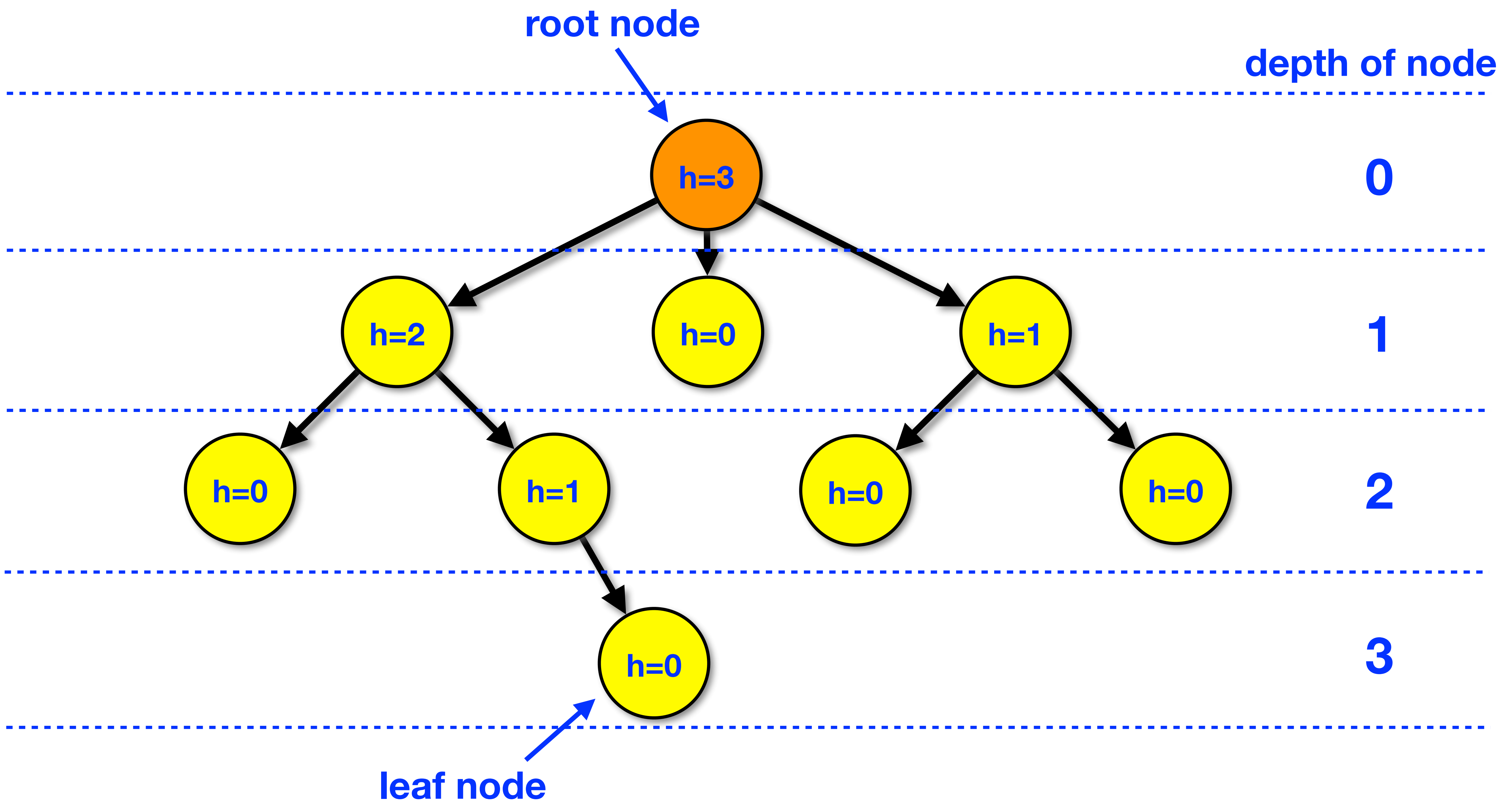
Tree Properties

- **A tree with *N* nodes must have *N-1* edges**
  - Every node, except the root node, has one incoming edge
- **The depth of a node in a tree is the length of the path from the root to the node**
  - The depth of the root is always 0
- **The height of a node in a tree is length of the path from the node to the deepest leaf**
  - The height of the tree is the height of the root node
- **A node that has no children is called a leaf node**
- **A ‘arity’ of a node is the maximum number of children the node can have**

Tree Properties

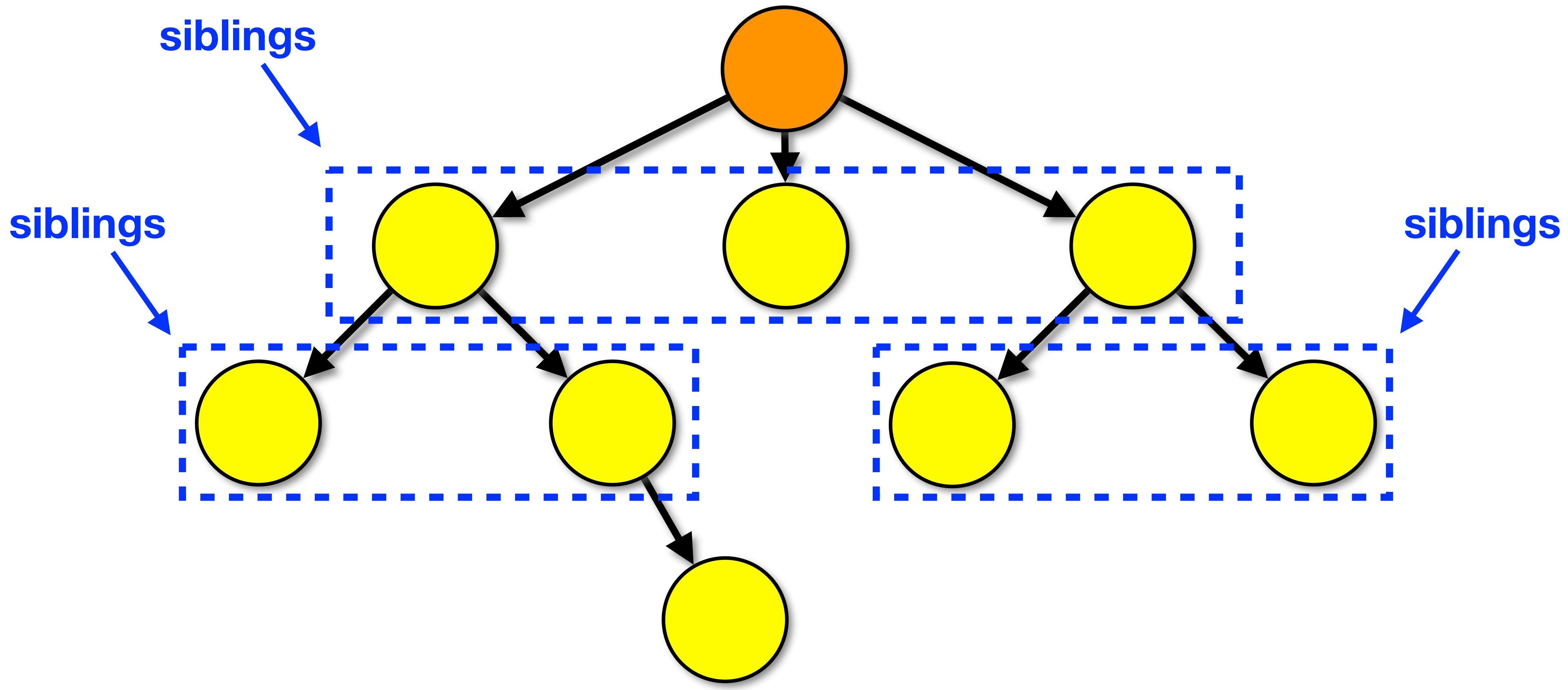


Tree Properties (Each Node Showing Its Height)



Tree Properties (Siblings)

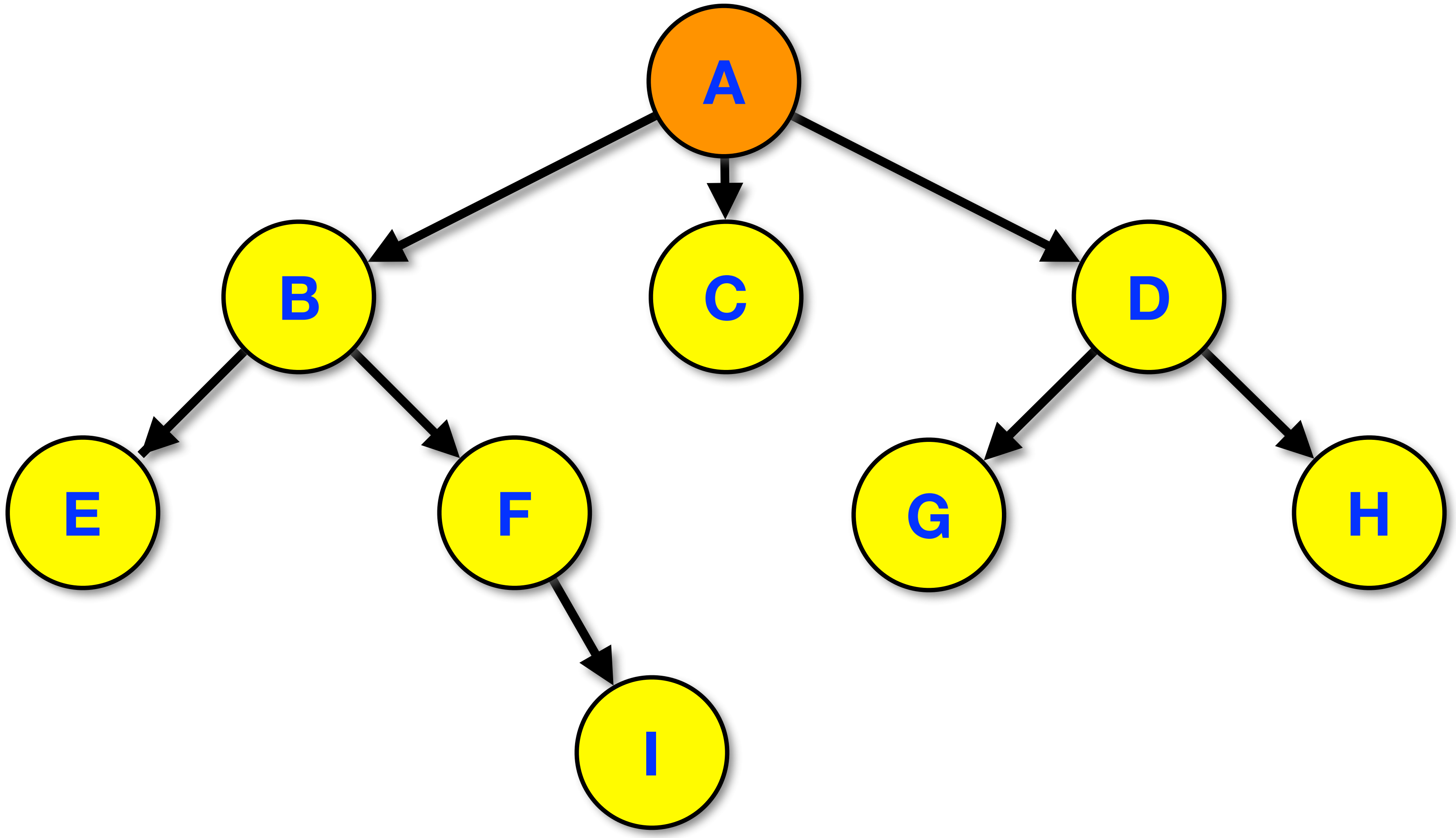
All children of a single node are siblings



Tree Properties (Ancestors)

The node E has two ancestors: B, A
The node C has one ancestor: A
The node I has three ancestors: F, B, A
The root node is an ancestor to all other nodes

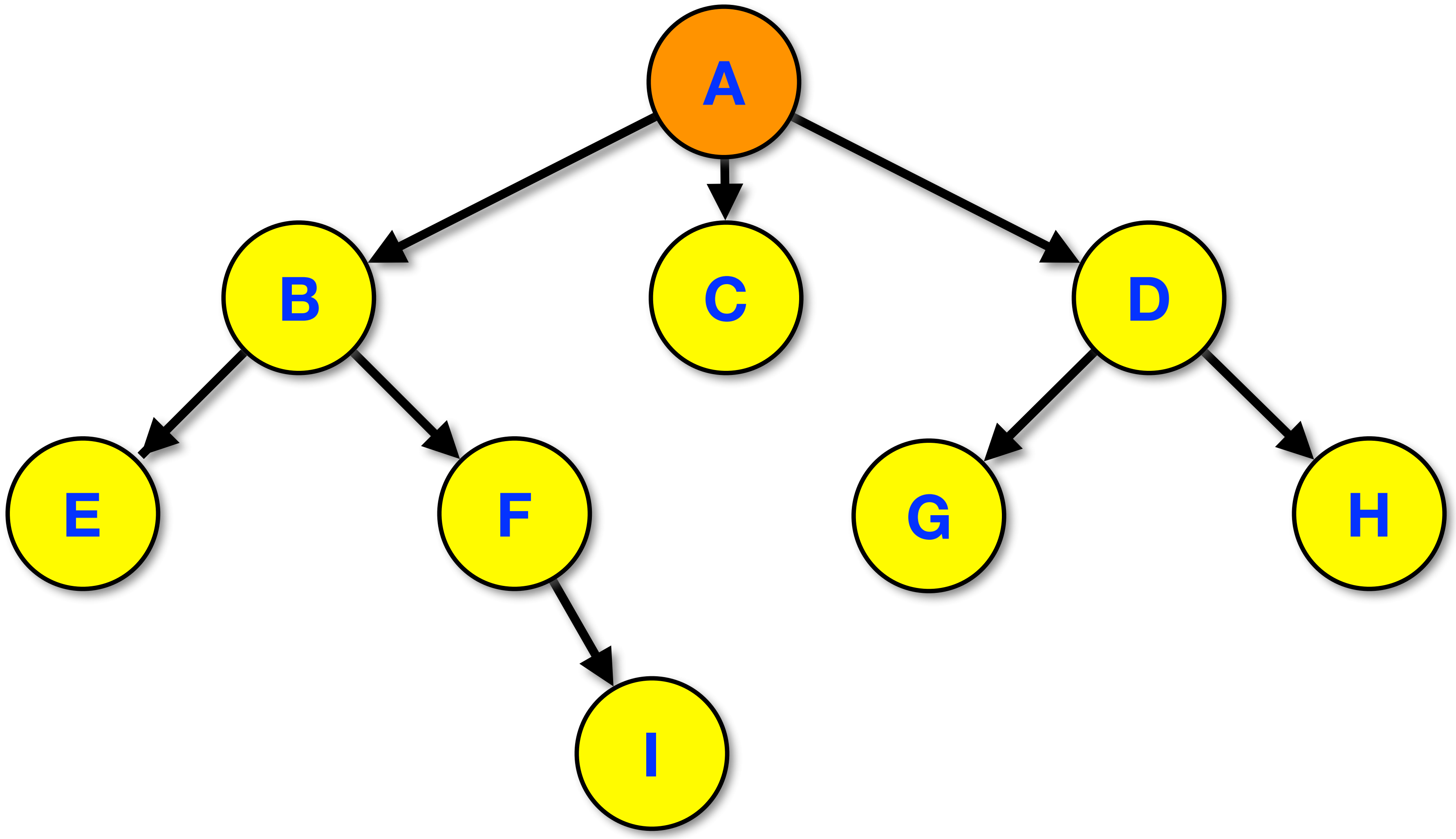
p is an ancestor of q iff there exists a path from p to q



Tree Properties (Descendants)

The node D has two **descendants**: G, H
The node F has one **descendant**: I
The node C has zero **descendants**
All nodes are **descendants of the root node**

p is an descendant of q iff there exists a path from q to p



Tree Implementation

- **A tree structure can be implemented in multiple different ways:**
  - (1) Each parent node stores references to all of its children
    - good approach when arity is small (e.g. binary trees)
    - Not a good choice when arity is high
    - Not a good choice when the number of children is unknown beforehand
  - (2) Each parent node stores references to first child, each child stores references to next sibling
    - A good approach when large fanout (large number of children per node)
    - Parent can have many children without wasting space in cases where there are only a few children

A Parent/Sibling Tree