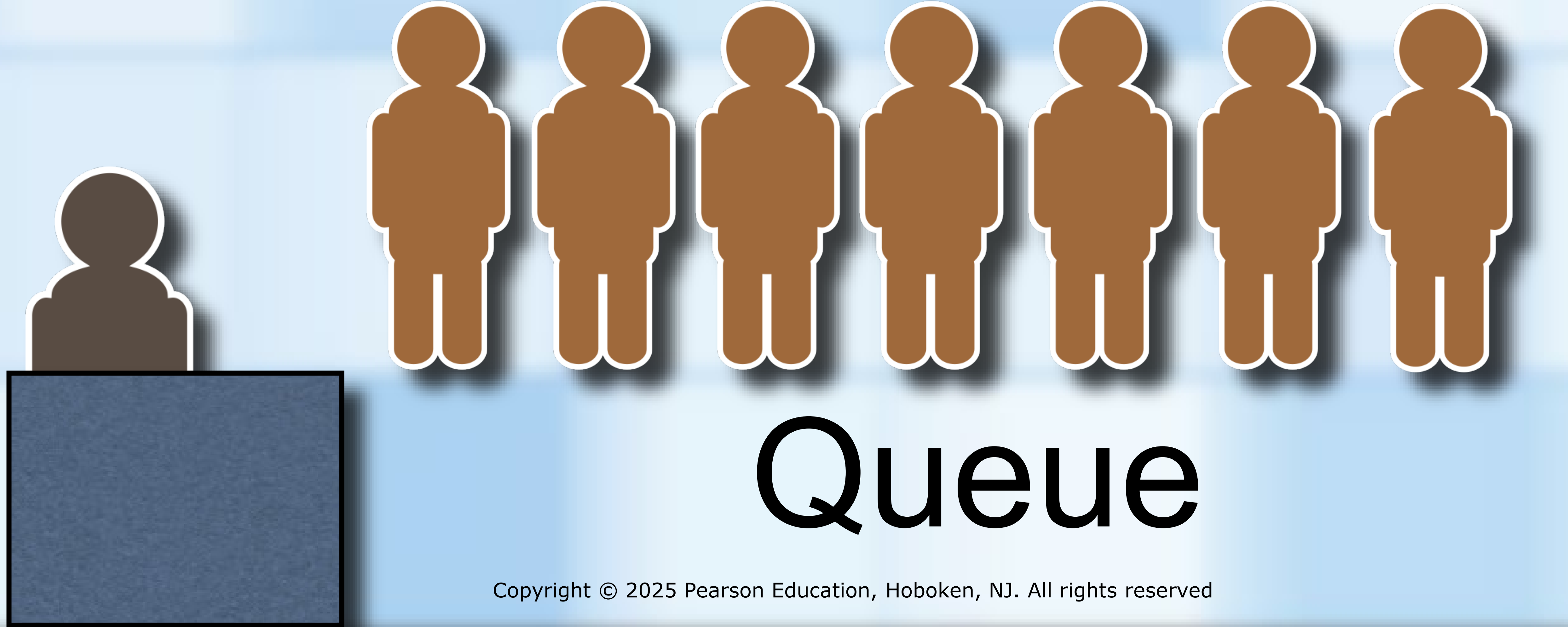
The ADT Tree

ORGANIZING DATA BY LINEAR POSITION

Top

Stack



Front

Back

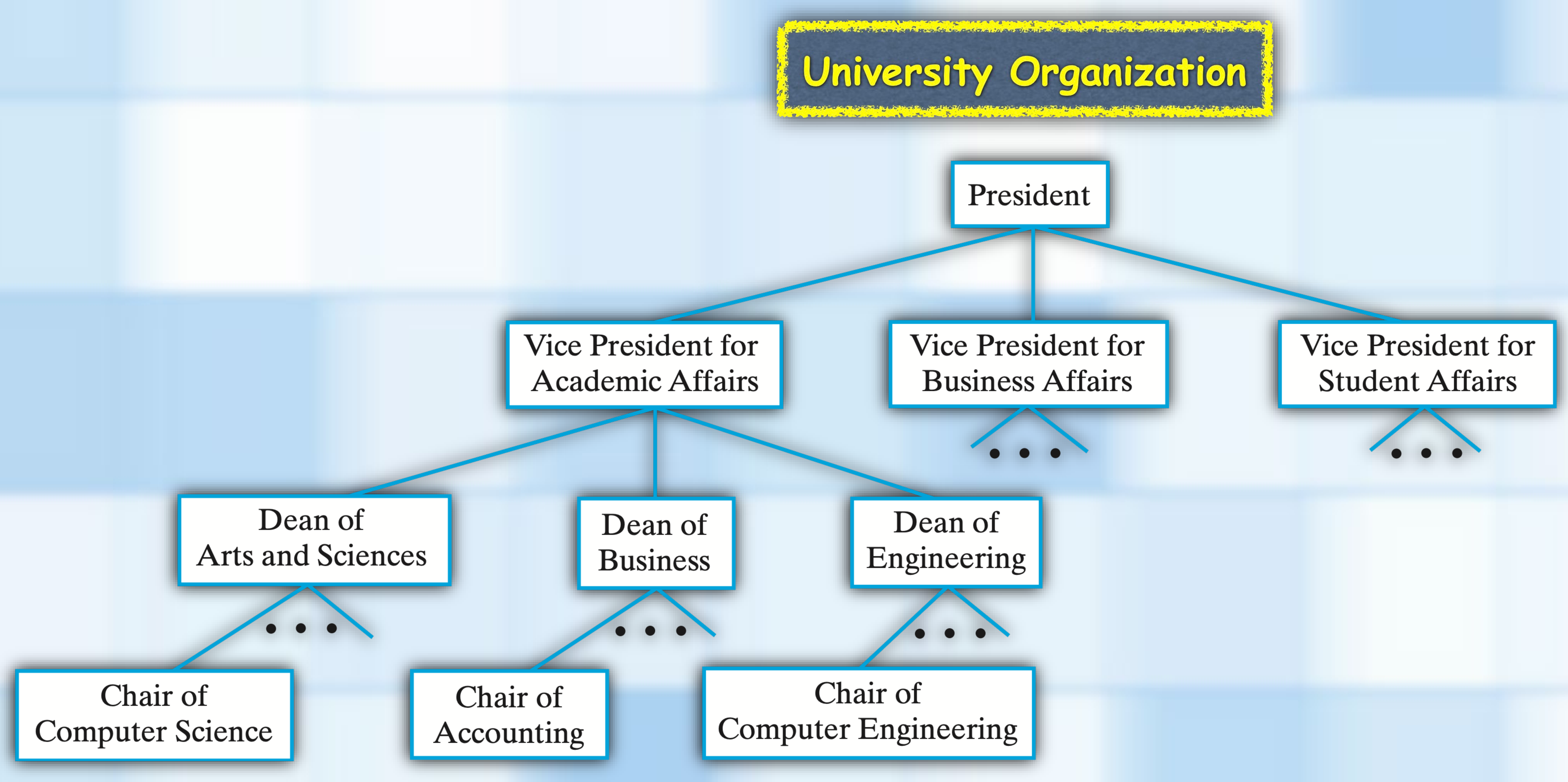
Queue

Position

List

	Grocery List
1	Bread
2	Oranges
3	Tortillas
4	Beans
5	Apples
6	Lettuce
7	Milk

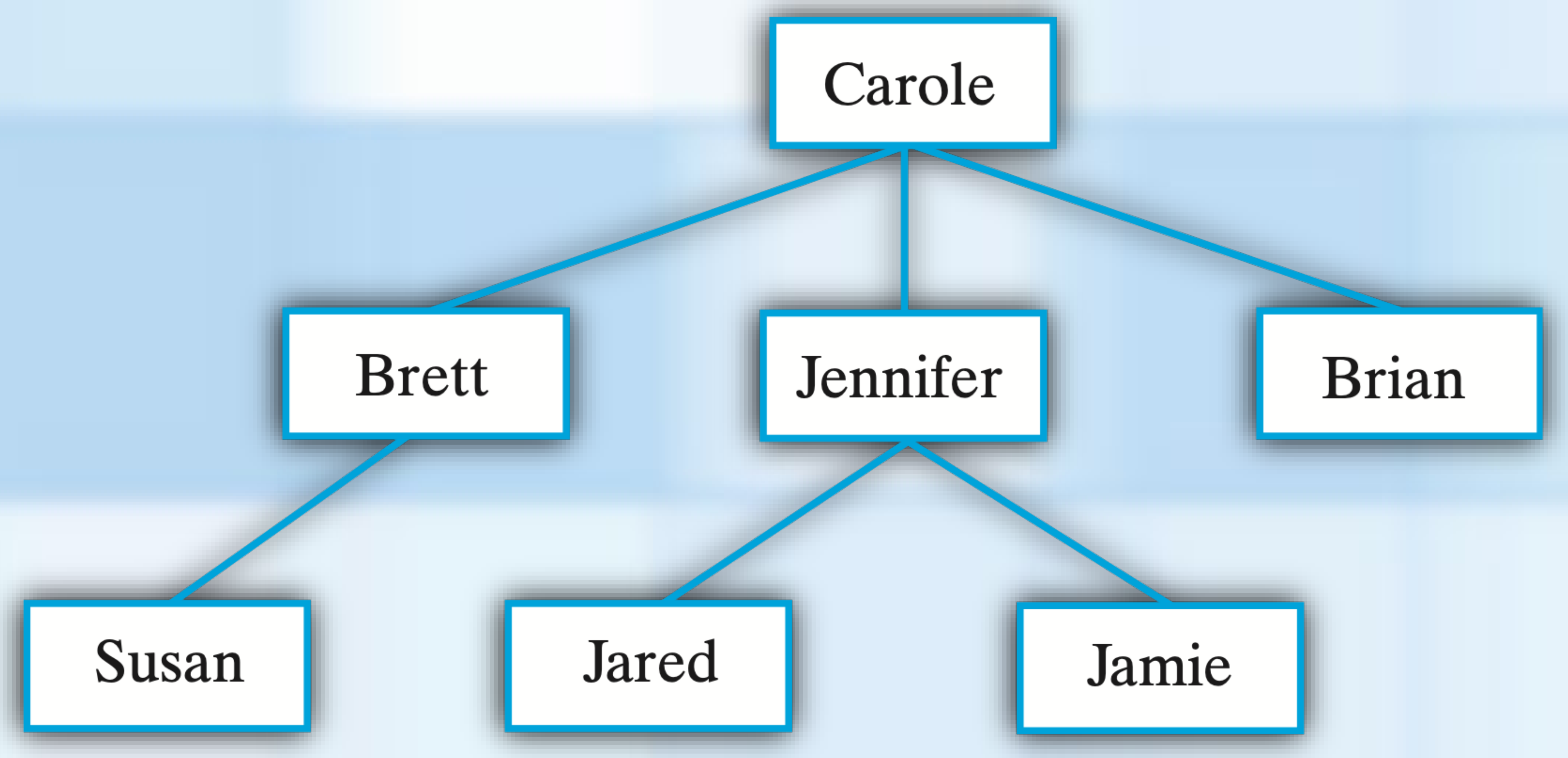
Hierarchical Data Organizations



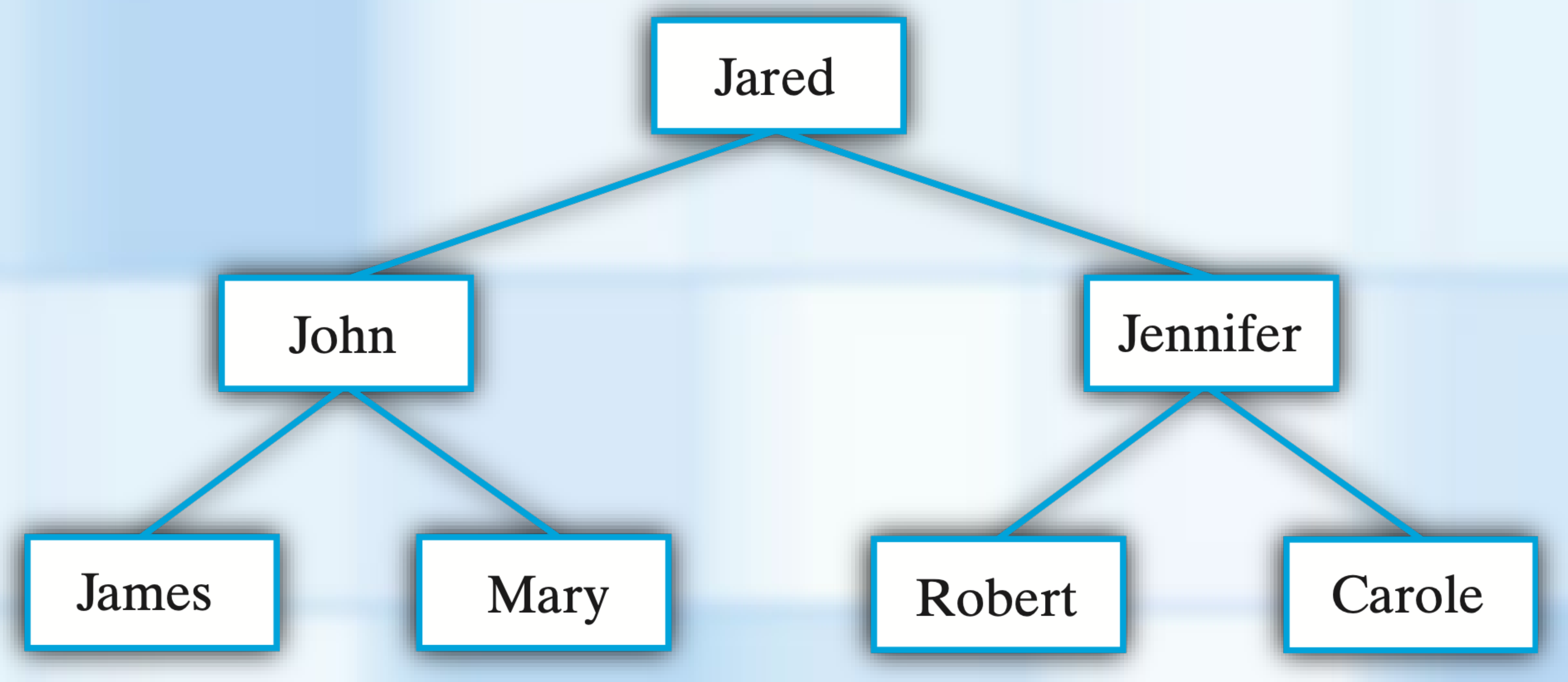
University Organization
President
Vice President for Academic Affairs
Vice President for Business Affairs
Vice President for Student Affairs
Dean of Arts and Sciences
Dean of Business
Dean of Engineering
Chair of Computer Science
Chair of Accounting
Chair of Computer Engineering
. . .

Hierarchical Data Organizations

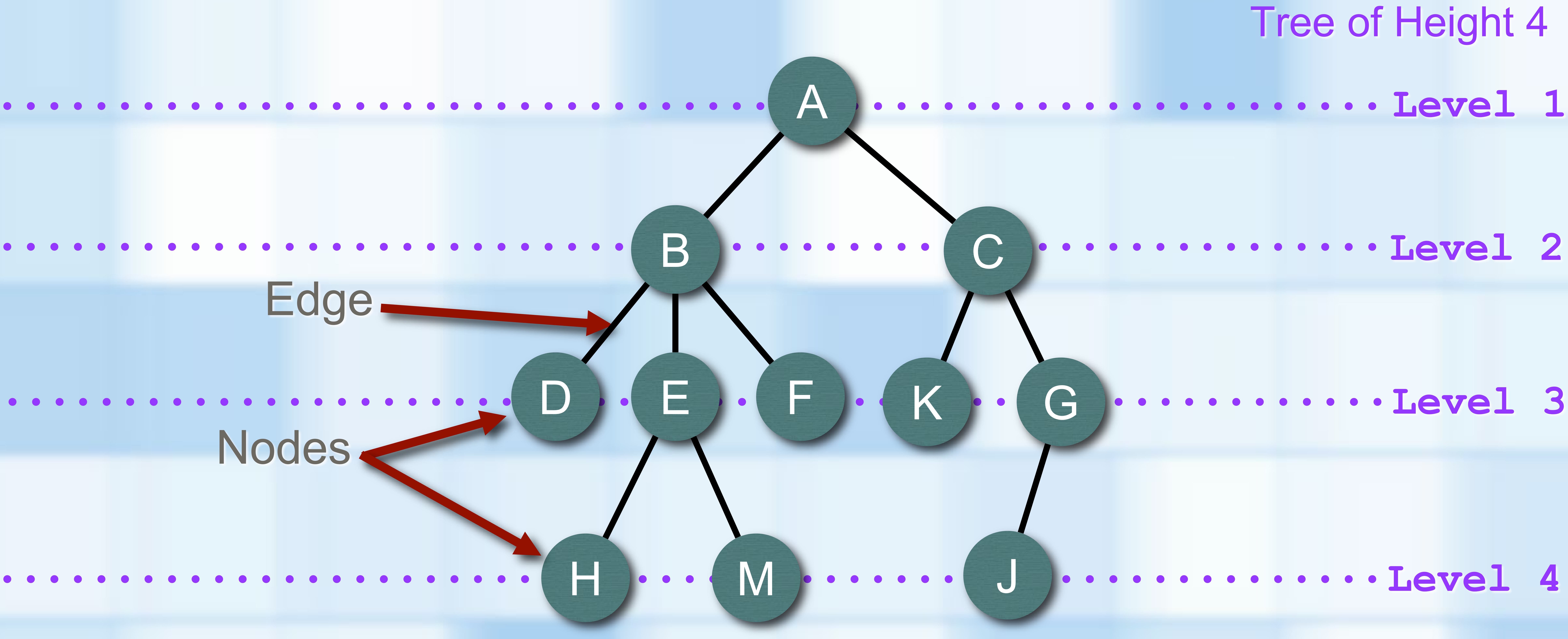
Carole's children and grandchildren



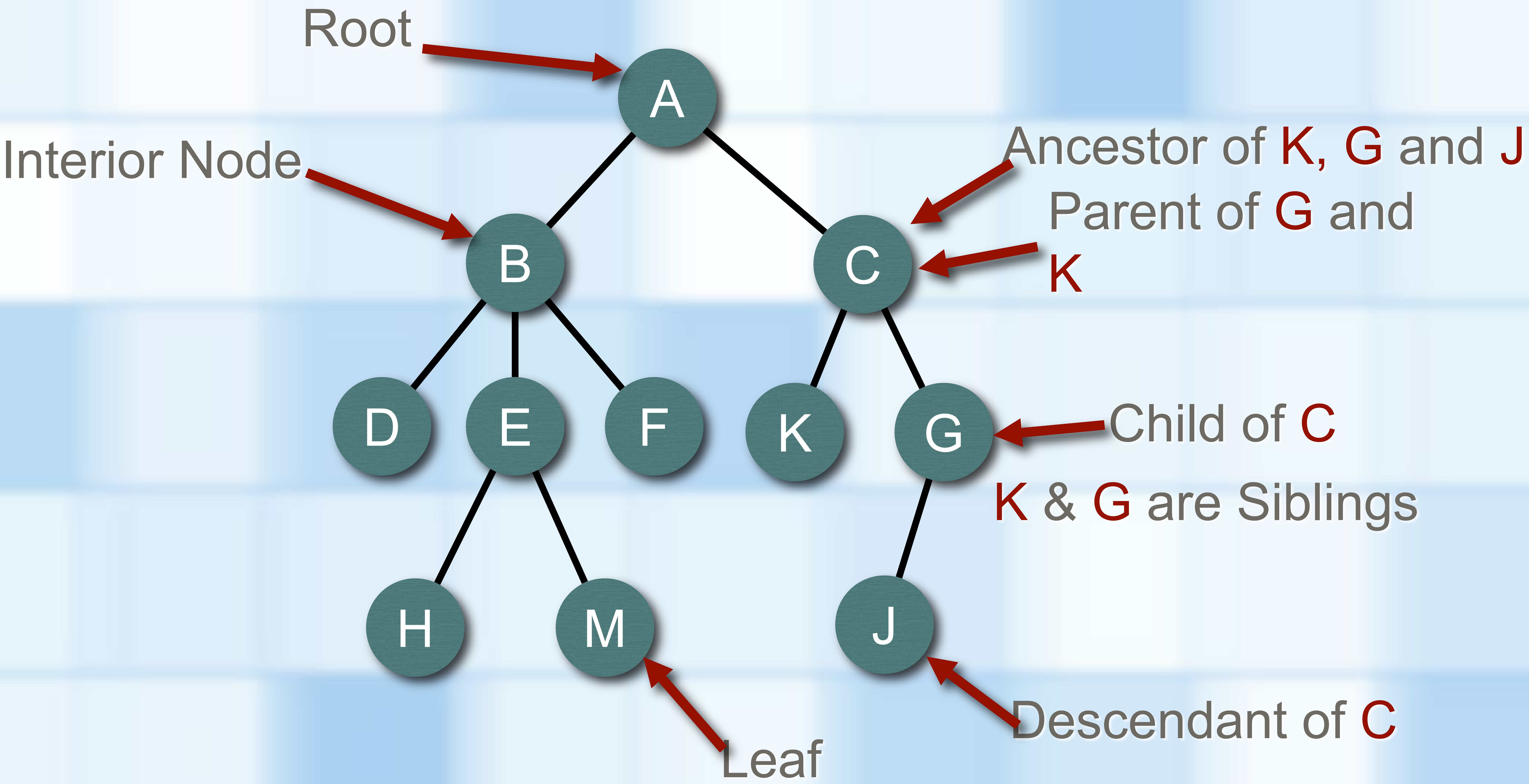
Jared's parents and grandparents



TREES


Tree of Height 4
Level 1
Level 2
Level 3
Level 4
Edge
Nodes
A
B
C
D
E
F
K
G
H
M
J

TREES



Root

Interior Node

Ancestor of K, G and J

Parent of G and K

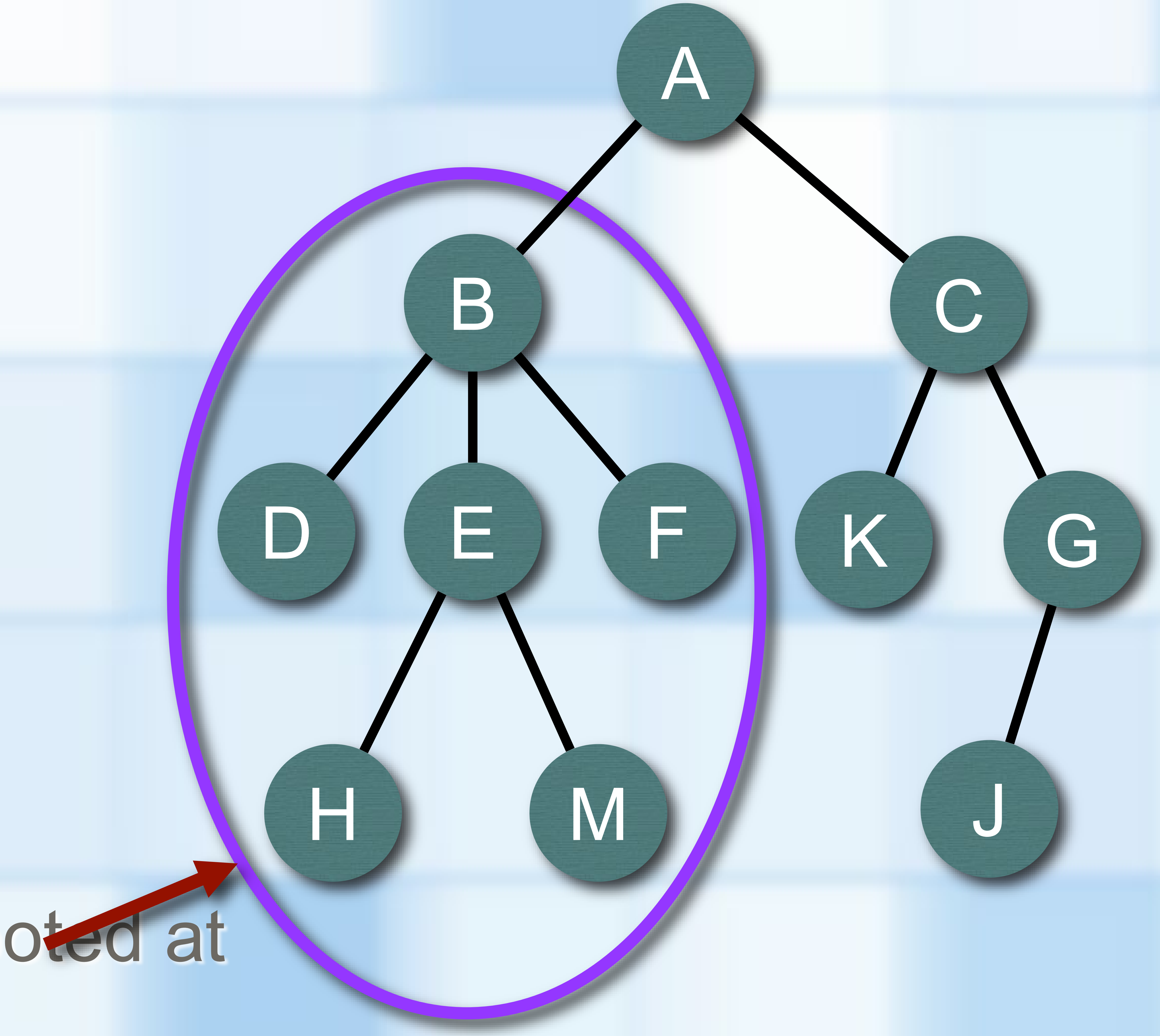
Child of C

K & G are Siblings

Leaf

Descendant of C

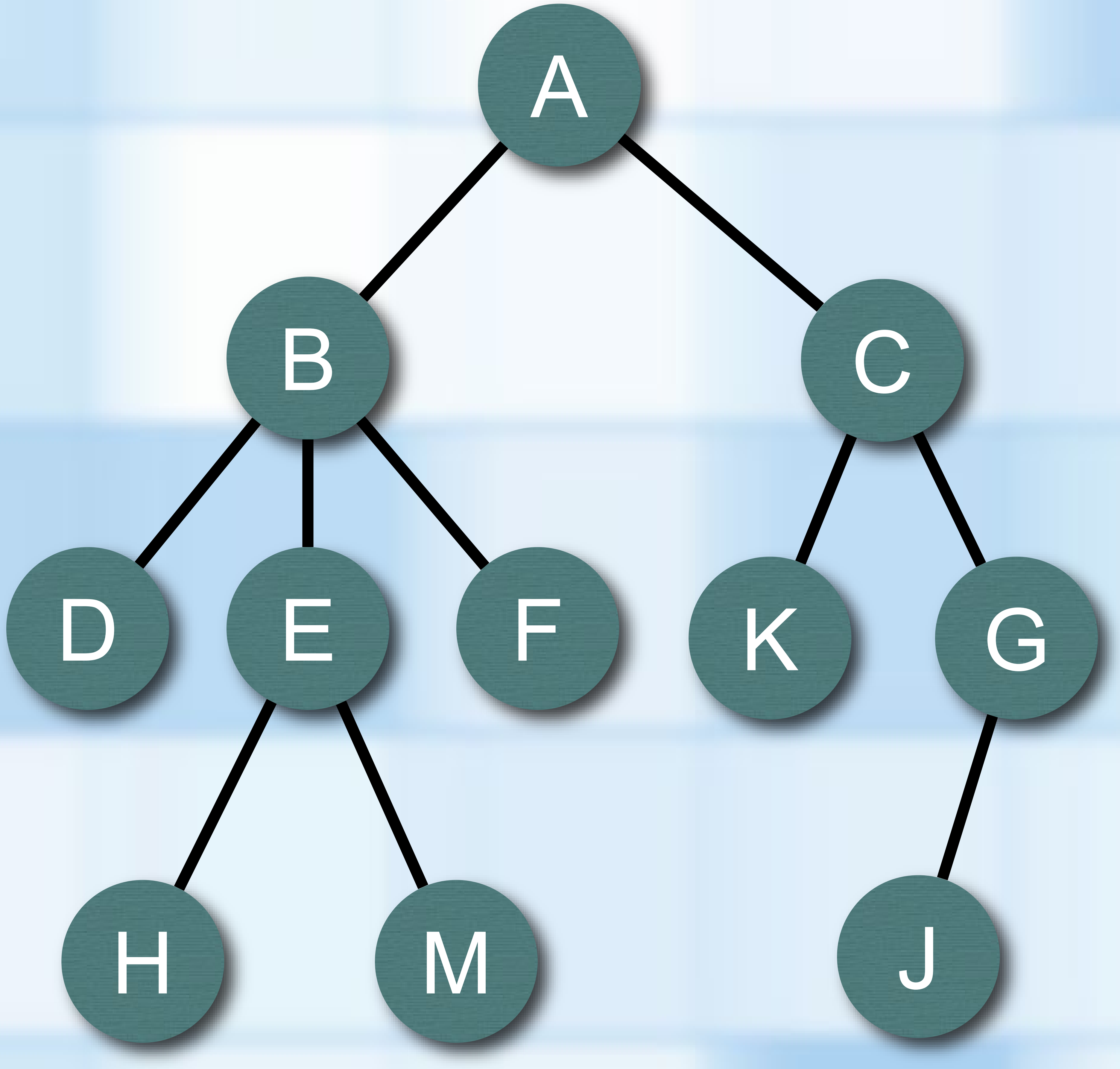
Trees



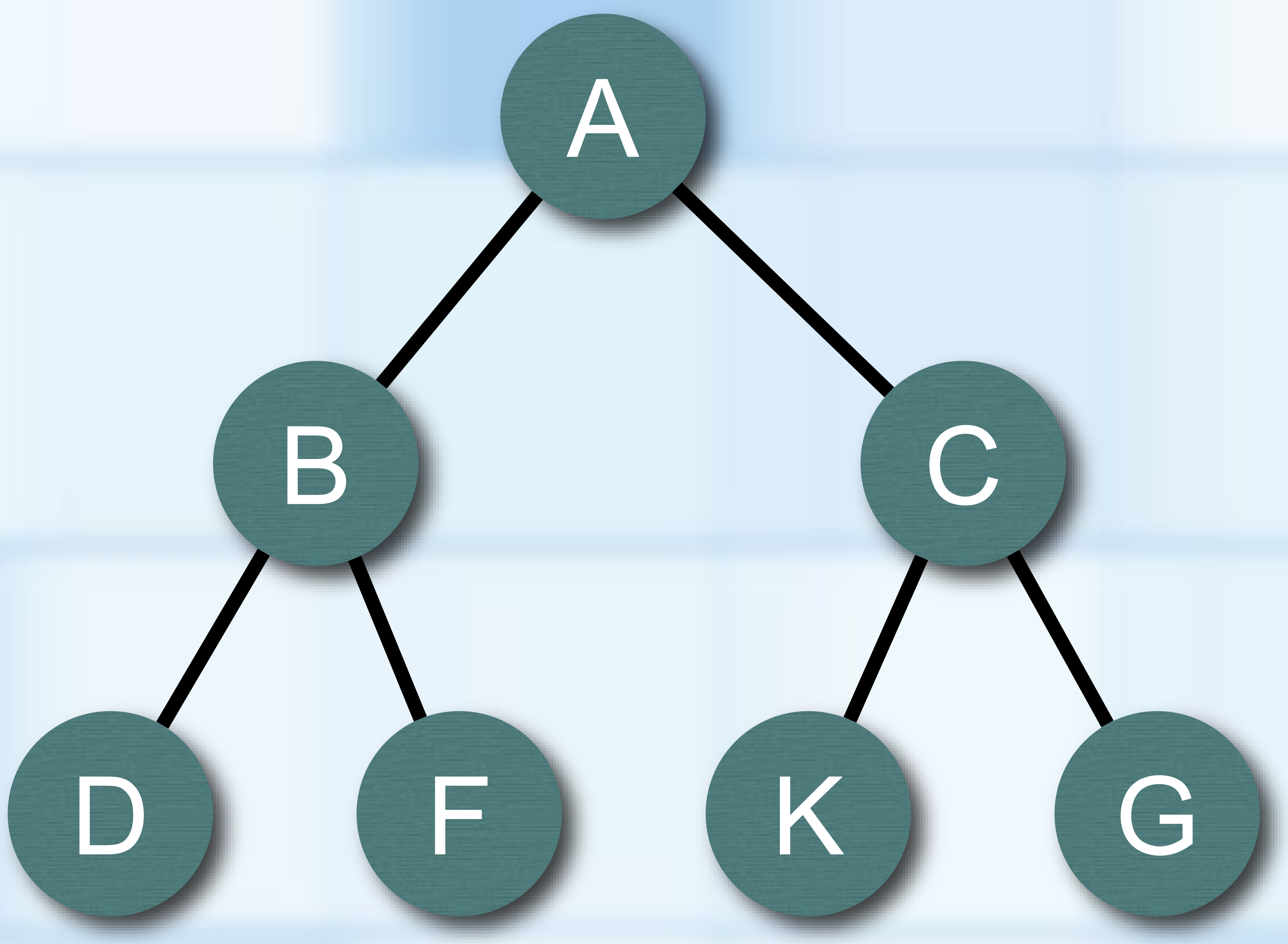
Subtree Rooted at
B

TREES

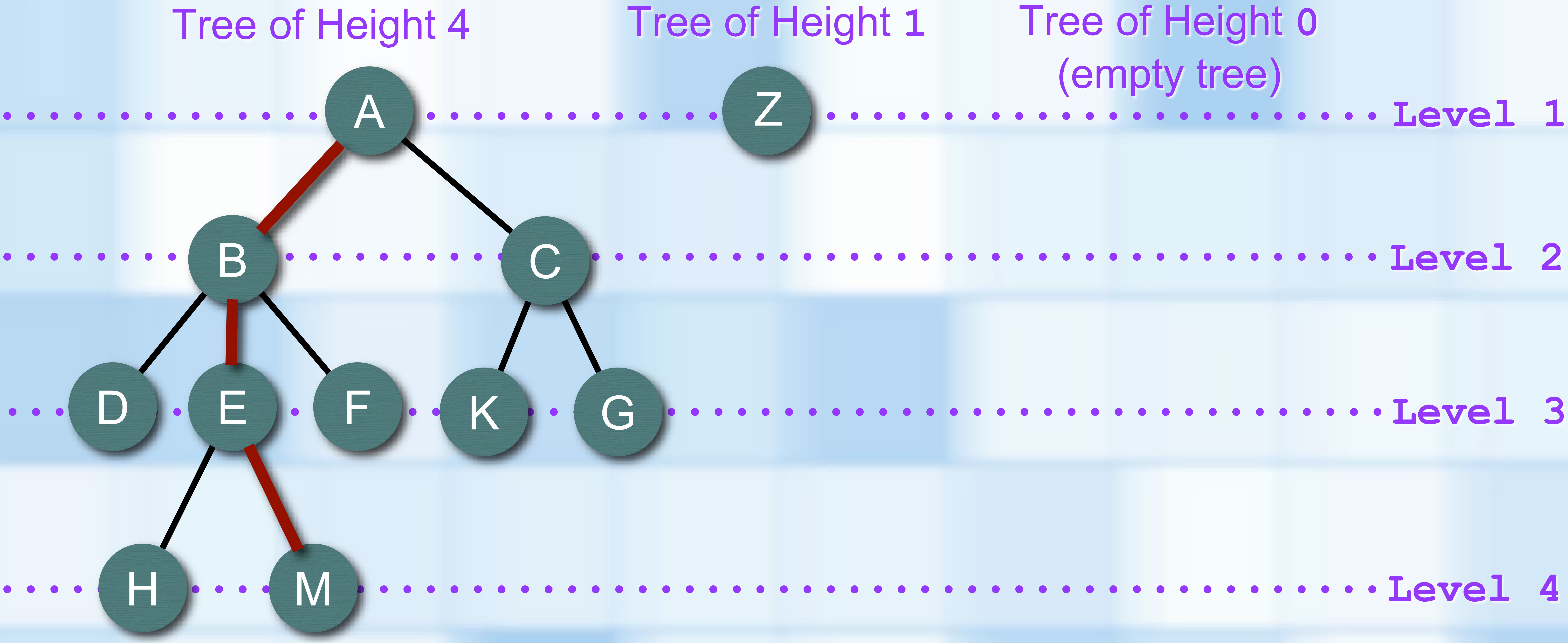
General Tree



Binary Tree



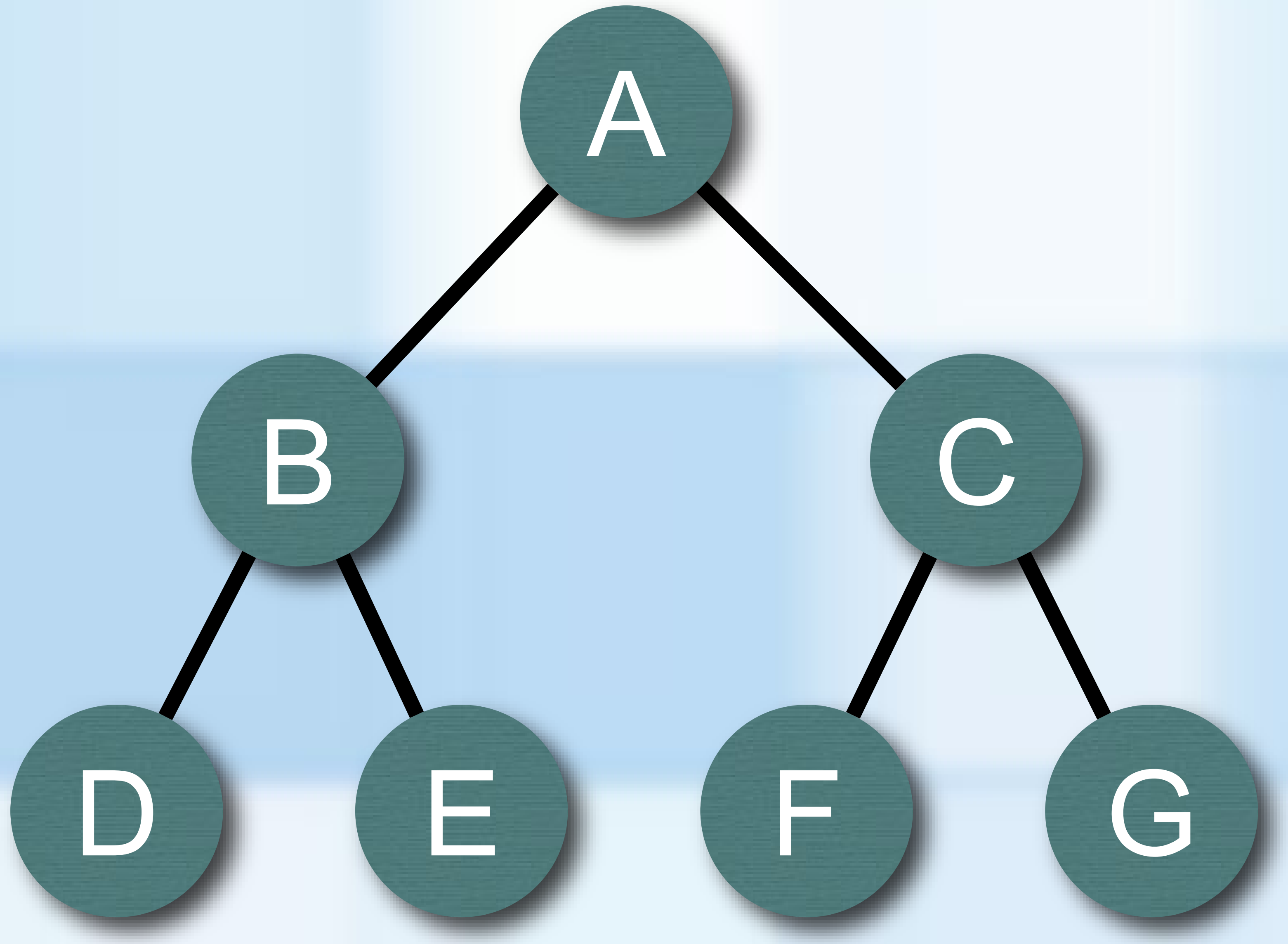
TREES

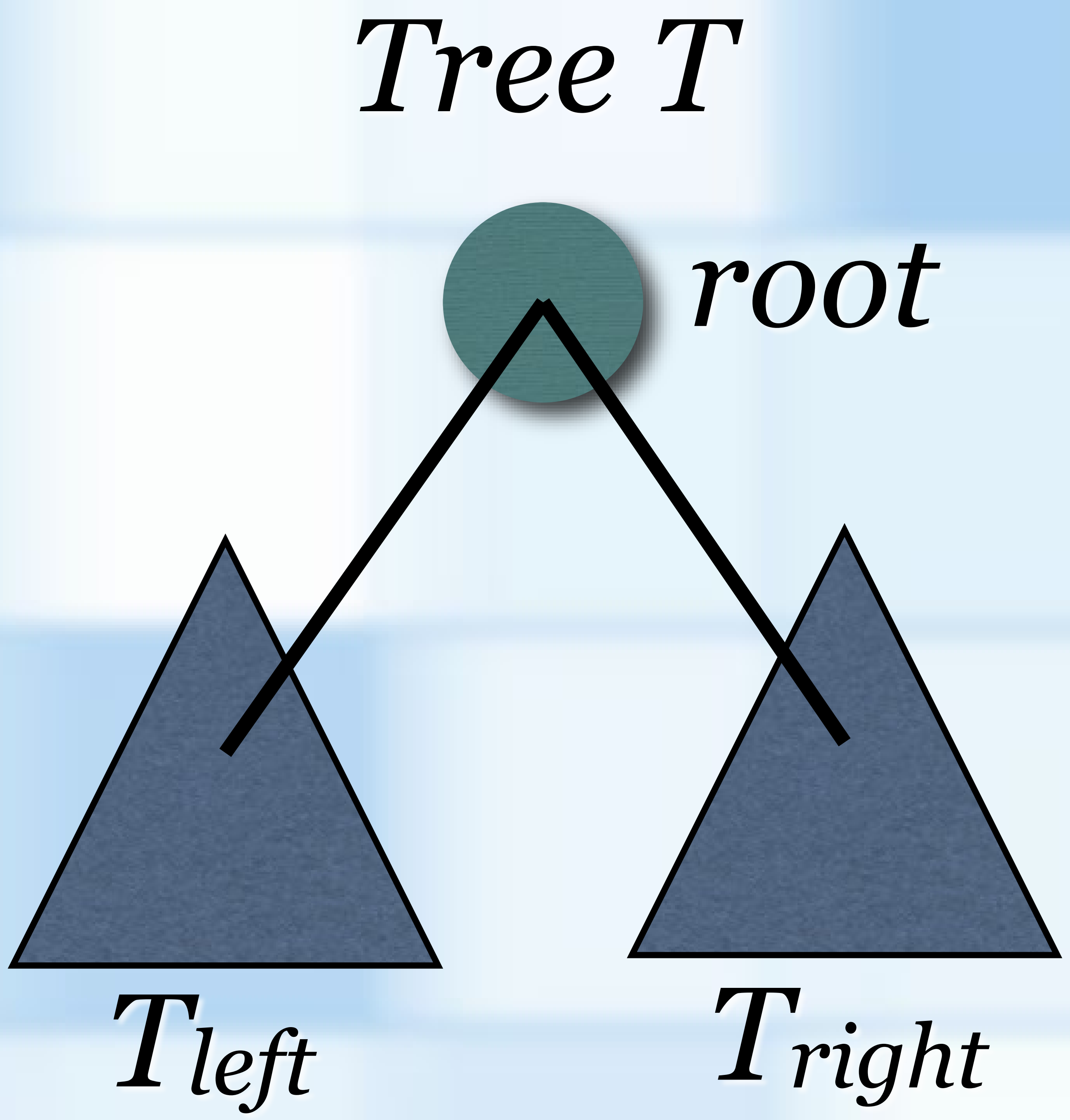


Height of tree *T* = 1 + height of the tallest subtree of *T*

Binary Trees

Each node has at most two children.

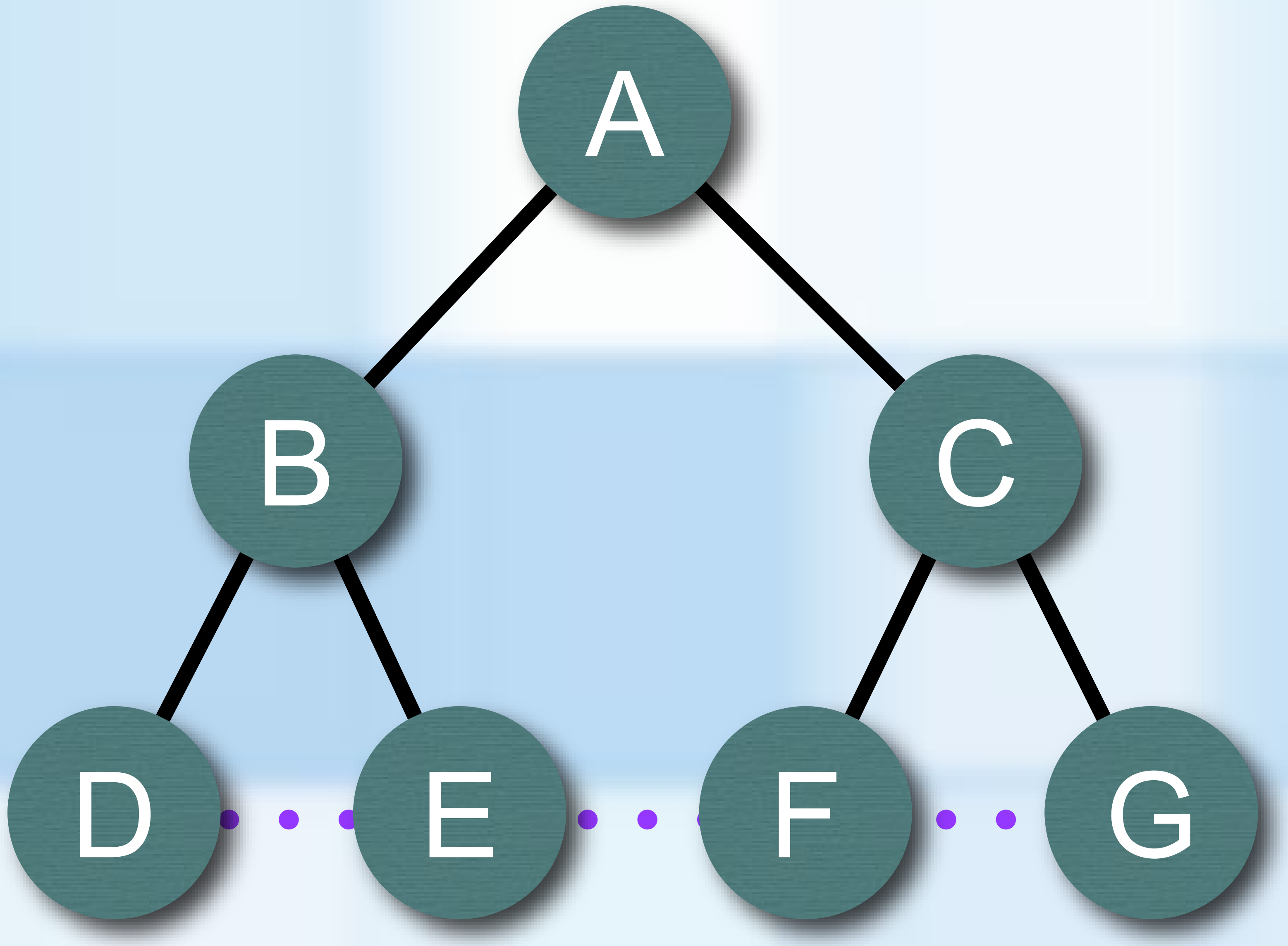




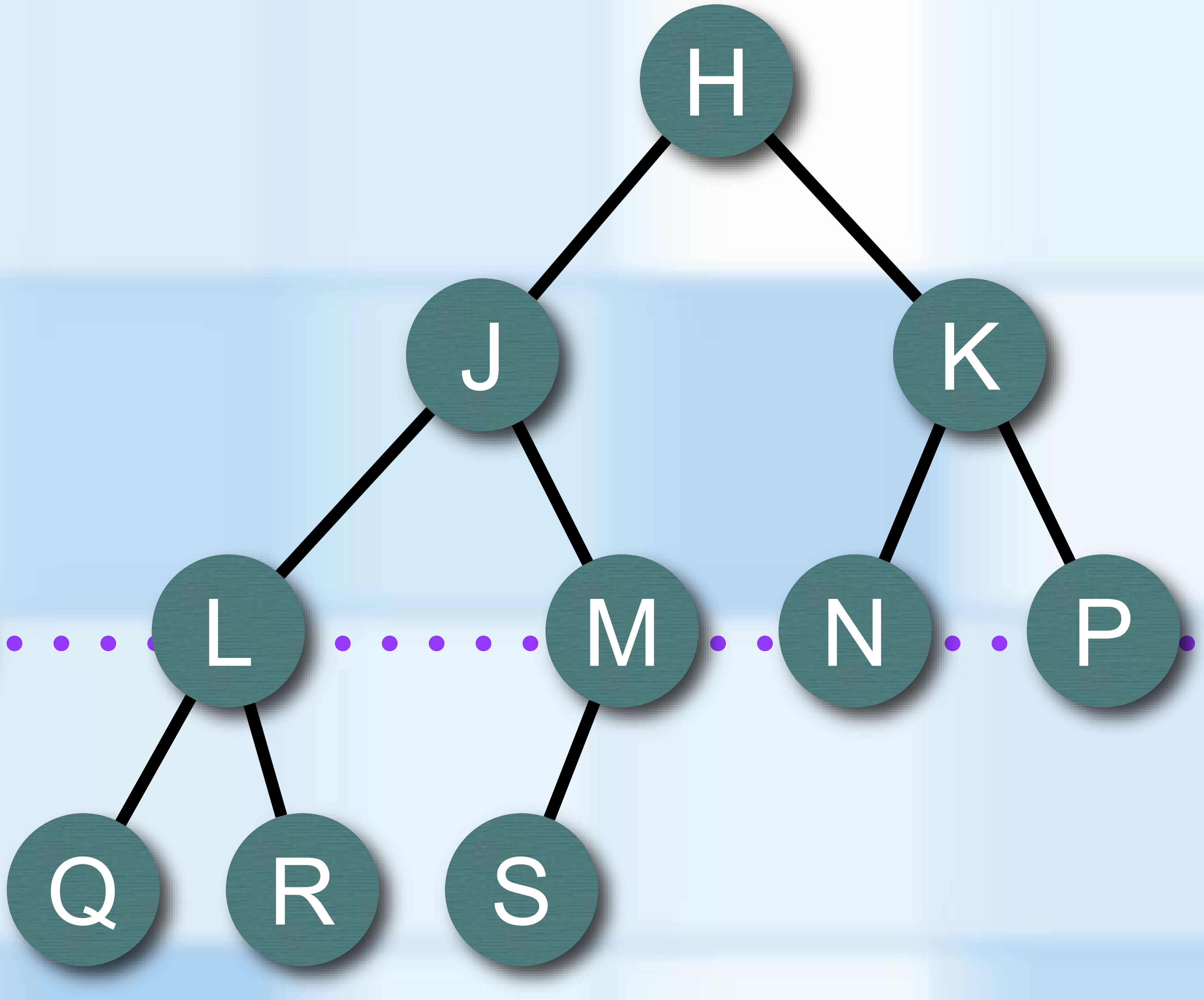
where T_{left} and T_{right} are binary trees.

BINARY TREES

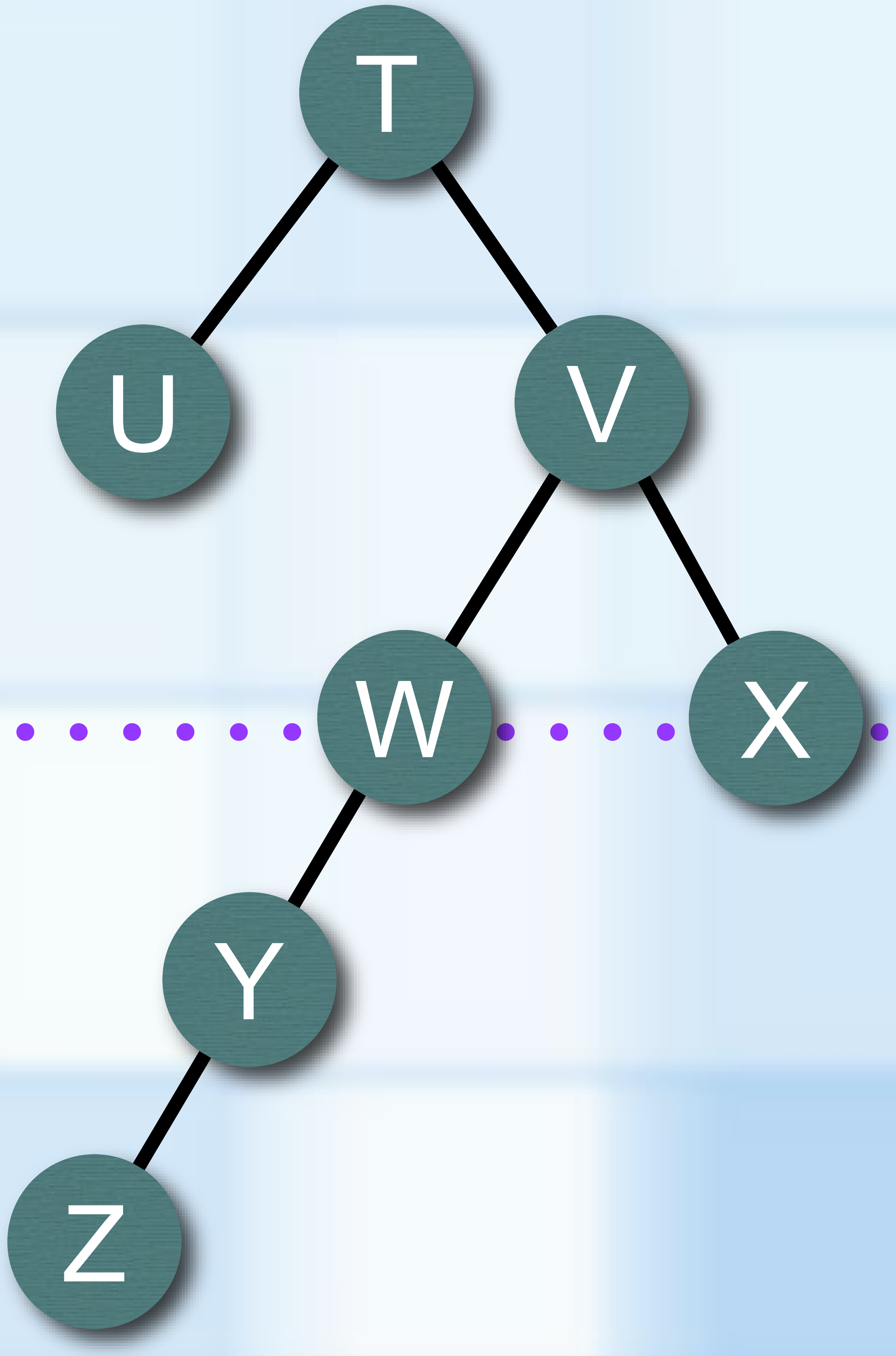
Full Binary Tree



Complete Binary Tree

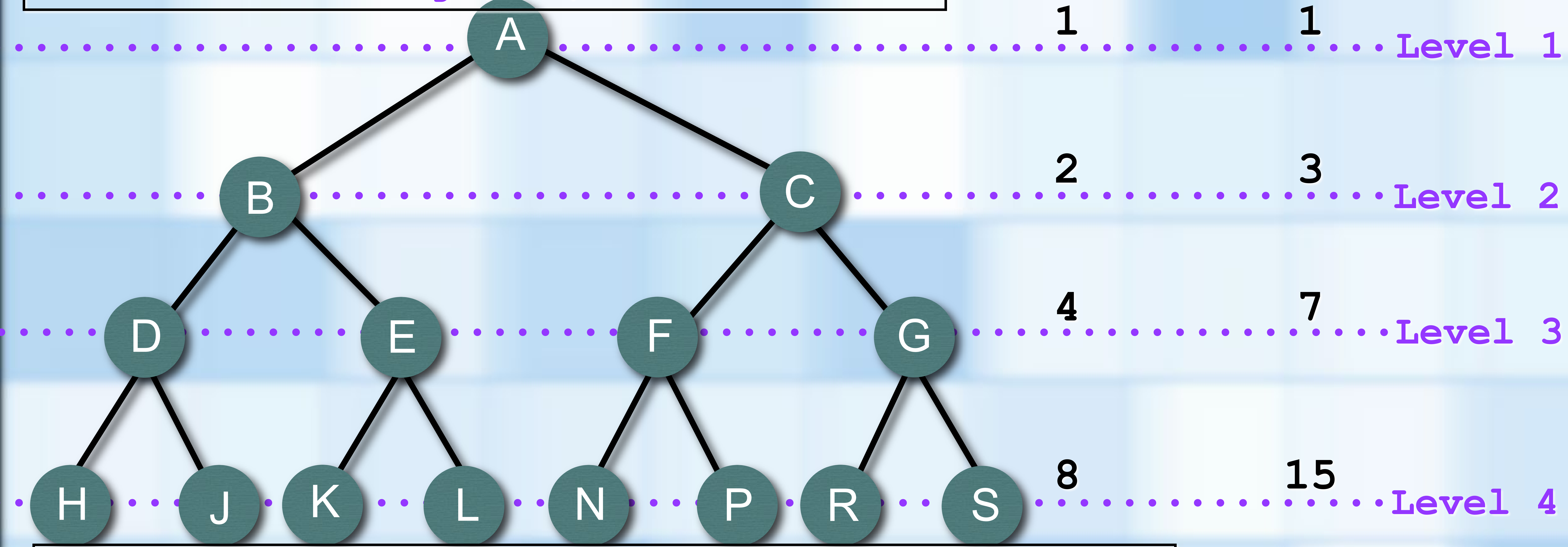


Binary tree that is not full and not complete



BINARY TREES

The minimum height of a complete binary tree is $\log_2(n+1)$



Number of nodes at level	Total nodes in full tree	
1	1	Level 1
2	3	Level 2
4	7	Level 3
8	15	Level 4

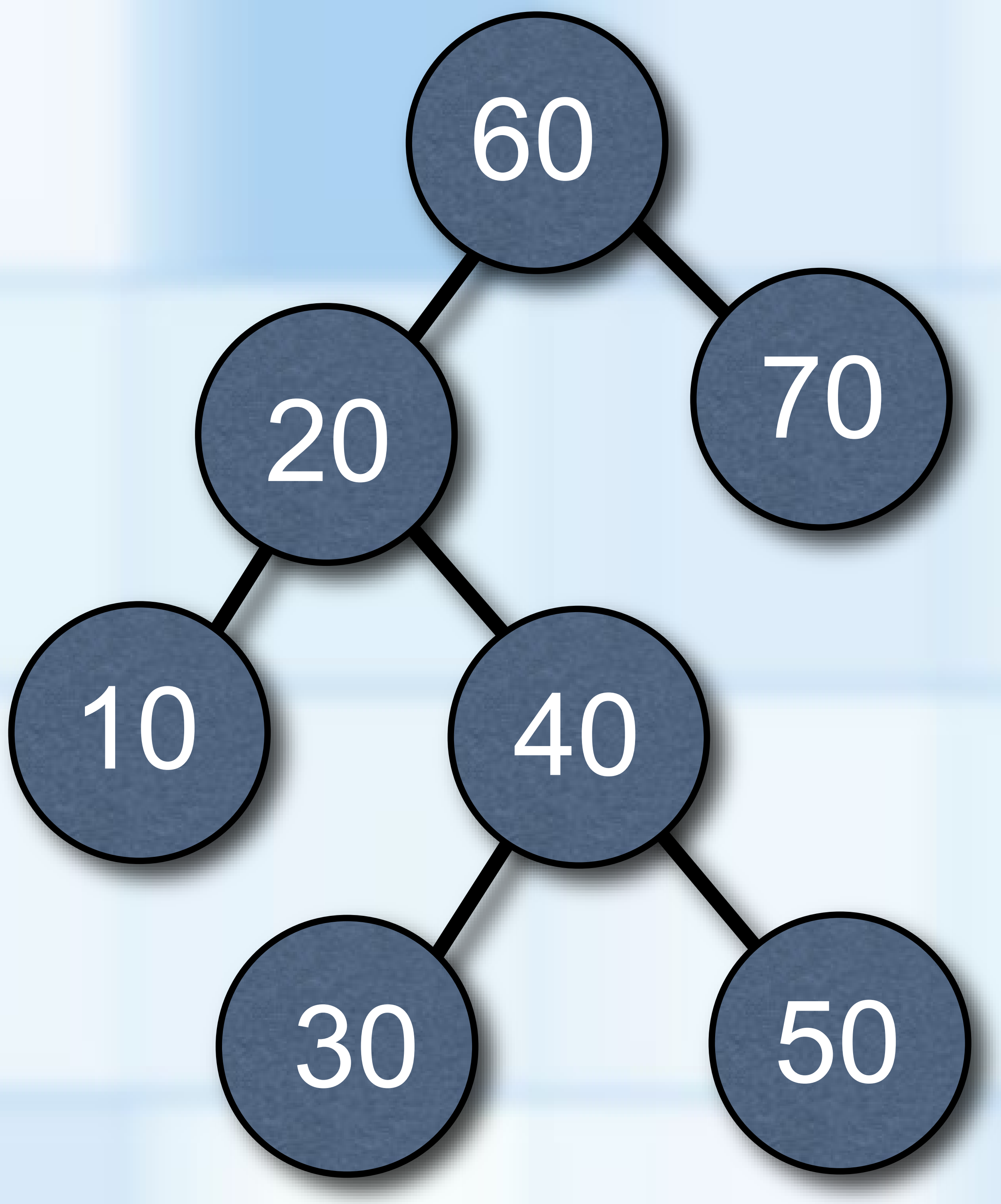
For full binary tree of height h, the number of nodes n is 2^h-1

Pearson

TREE TRAVERSALS

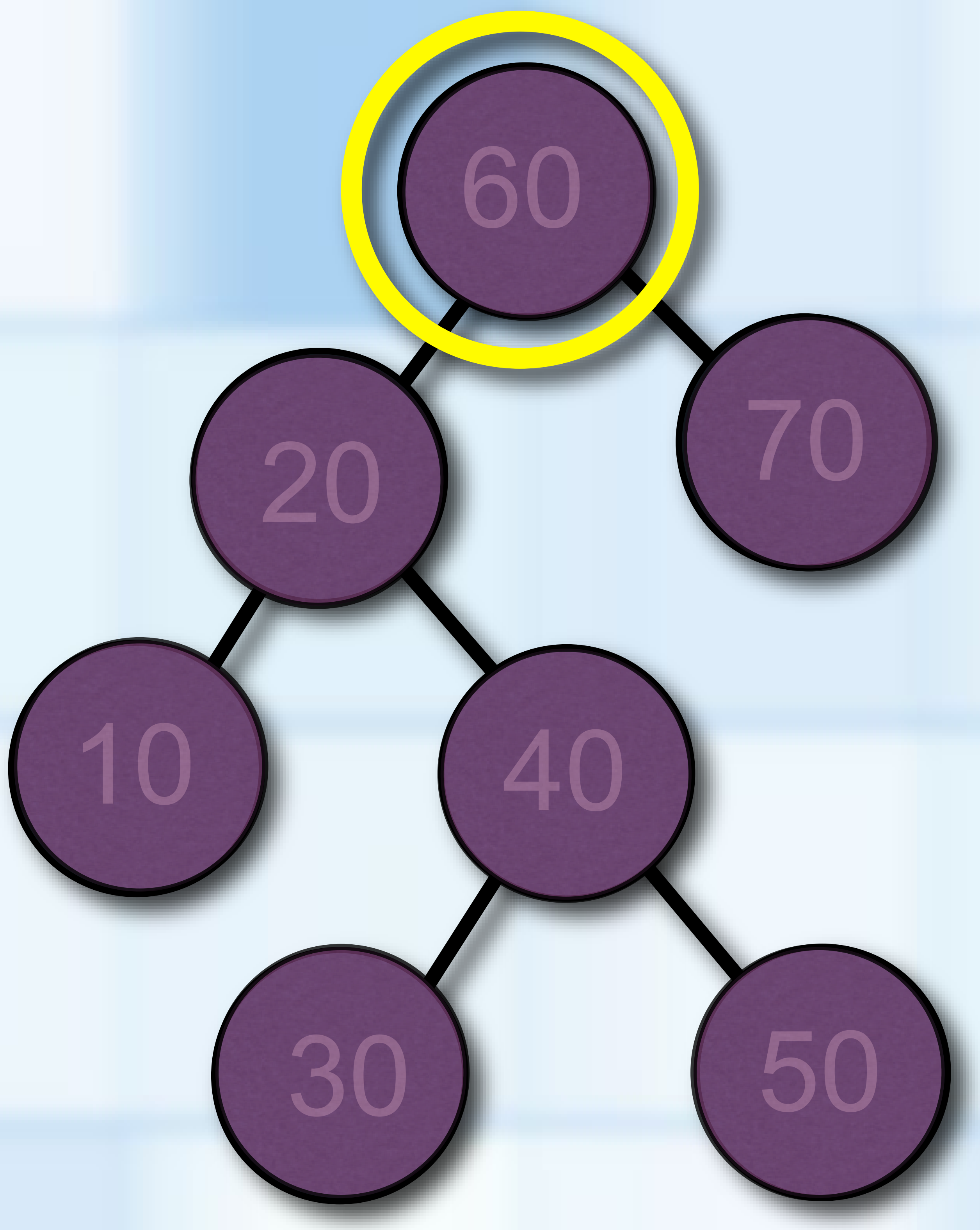
BINARY TREE TRAVERSALS

- **A traversal visits each node in a tree**
- **Process each node during a visit**
  - For example, display the data in the node
- **Traversals are generally recursive algorithms**
- Consider the call **`traverse(treeRoot)`**
  - Visit the root
  - Traverse nodes in the root's left subtree
  - Traverse nodes in the root's right subtree



Pre-Order Traversal

- **Visit root before visiting it's subtrees**
    - before the recursive calls
- **if (Tree is not empty)**
    - **process** (visit) root
    - **traverse**(Left subtree of Tree's root)
    - **traverse**(Right subtree of Tree's root)



Preorder Traversal

60 20 10 40 30 50 70

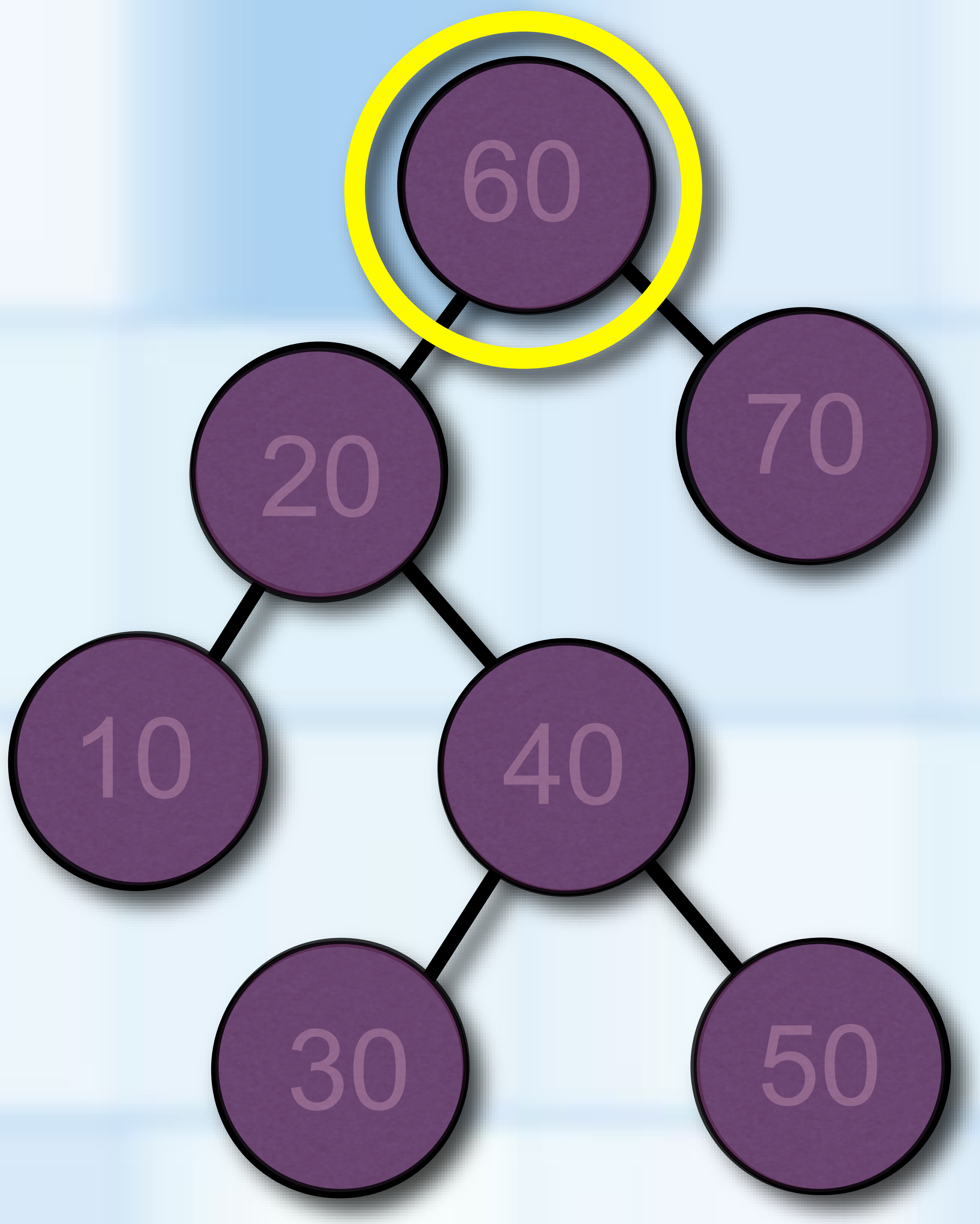
Check your work:
Root should be first

Depth-First Traversal

Pearson

In-Order Traversal

- **Visit root after visiting it's left subtree**
  - between the recursive calls
- **if (Tree is not empty)**
  - **traverse**(Left subtree of Tree's root)
  - **process** (visit) root
  - **traverse**(Right subtree of Tree's root)

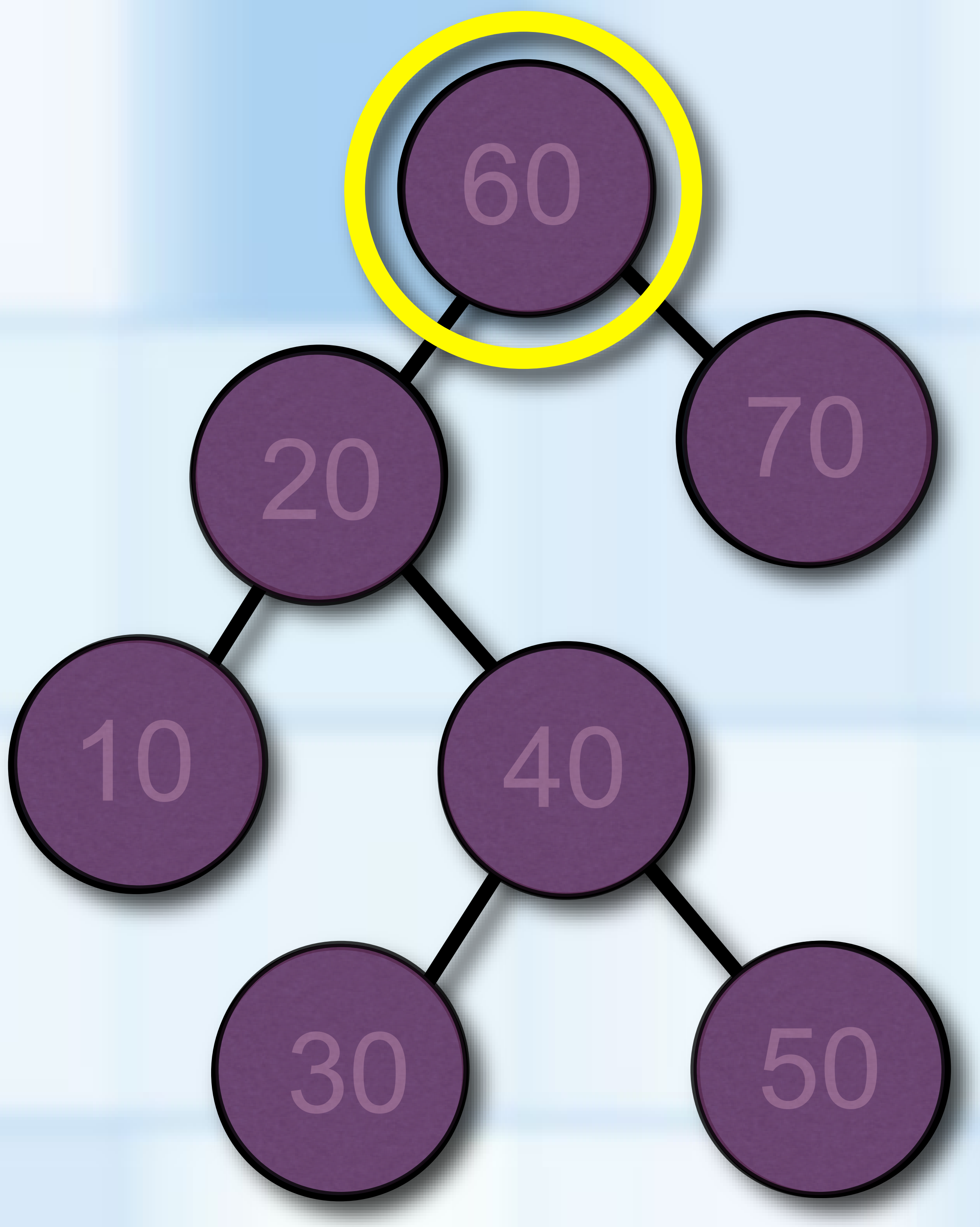


Inorder Traversal

10 20 30 40 50 60 70

Post-Order Traversal

- **Visit root after visiting it's subtrees**
  - after the recursive calls
- **if (Tree is not empty)**
  - **traverse**(Left subtree of Tree's root)
  - **traverse**(Right subtree of Tree's root)
  - **process** (visit) root



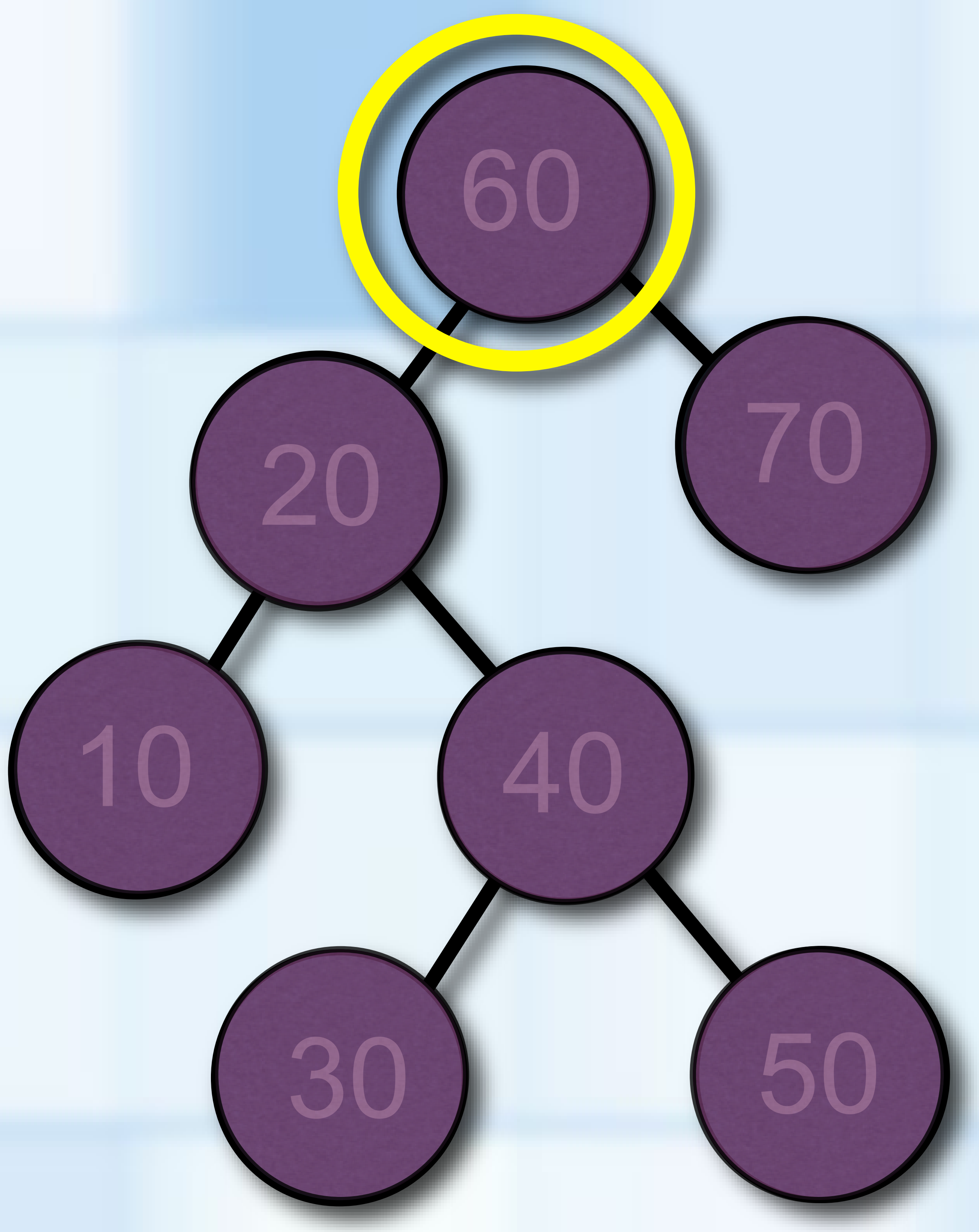
Postorder Traversal

10 30 50 40 20 70 60

Check your work:
Root should be last

Level-Order Traversal

- **Visit Nodes by Level**
  - *Visit each node, top down, left to right.*
- **if (Tree is not empty)**
  - Visit root
  - Visit Level 2, left to right
  - Visit Level n, left to right



Level Order Traversal

60 20 70 10 40 30 50

Check your work:
Root should be first

Breadth-First Traversal