Why did the programmer quit their job?

A never got arrays!

CMPT 225

Lecture 12 – Introduction to Trees

Learning Outcomes

- At the end of the next few lectures, a student will be able to:
  - Define the following data collections:
    - Binary search tree
    - Balanced binary search tree (AVL)
    - Binary heap

    as well as demonstrate and hand-trace their operations
  - Implement the operations of binary search tree and binary heap
  - Implement and analyze sorting algorithms: tree sort and heap sort
  - Write recursive solutions to non-trivial problems, such as binary search tree traversals

Last Lecture

- We saw how to …
  - Describe how quick sort works
  - Analyze the time efficiency of quick sort
  - Improve quick sort's time efficiency
  - Analyze its space efficiency
  - Improve quick sort's space efficiency
  - Describe how merge sort works
  - Analyze the time efficiency of merge sort
  - Analyze its space efficiency

Today's menu

- Let's introduce another way of organizing our data
  - Another **category of data organization**
    - Hierarchical -> **tree**
- Defining some tree-related terms and concepts

Way back in lecture 3, we introduced …

Categories of data organizations

- **Linear**
  - Data organization in which each element has a unique predecessor (except for the first element, which has none) and a unique successor (except for the last element, which has none)
- **Non-Linear**
  - Data organization in which there is no first element, no last element and for each element, there is no concept of a predecessor and a successor

6

Categories of data organizations – cont'd

- **Hierarchical**
  - Data organization in which each element has only one predecessor -> its parent (except for the first element, which has none) and up to many successors (except for the last element(s), which has none)
- **Graph**
  - Data organization in which each element can have many predecessors and many successors

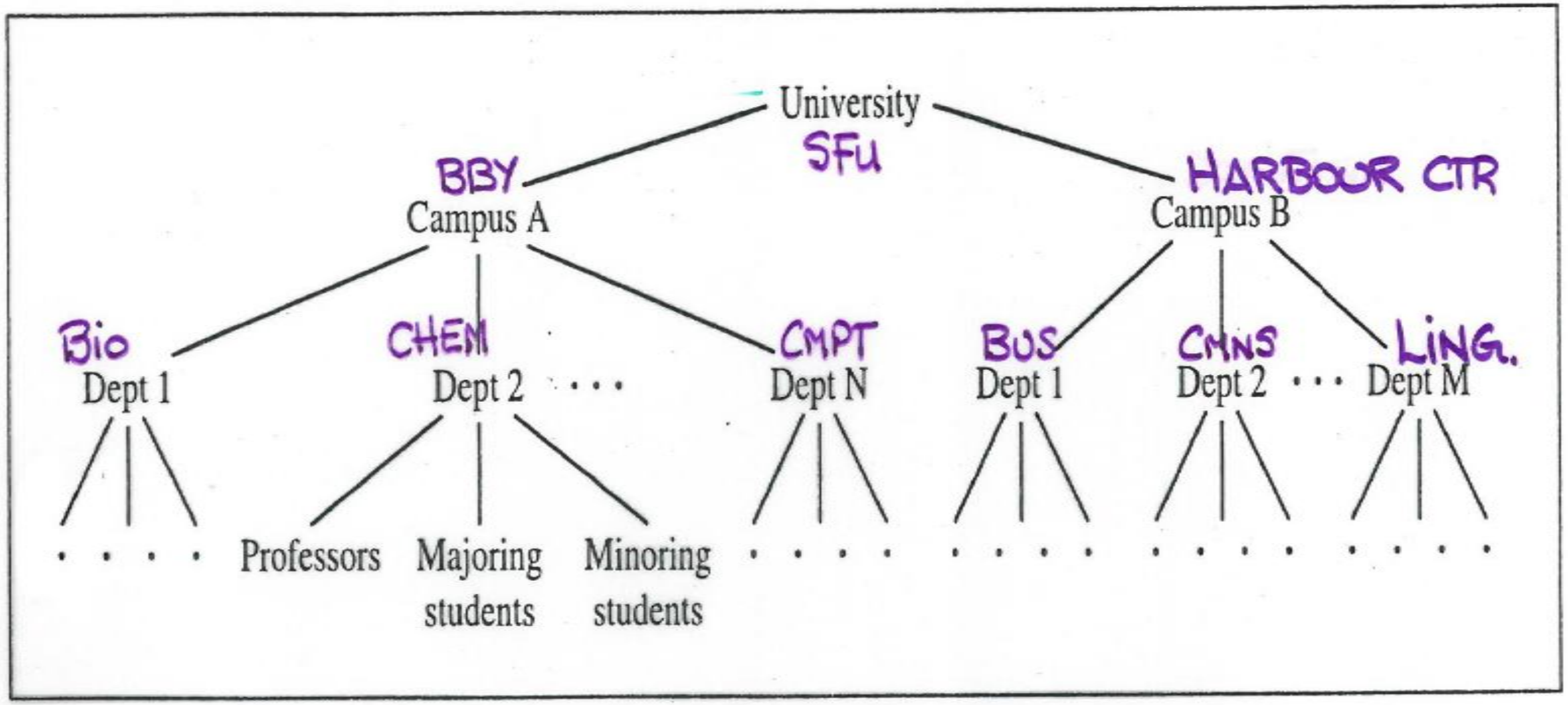
7

in the real world!

Example of data represented as **hierarchical** data organizations -> **tree**

Textbook:
We use **Trees** to represent relationships.

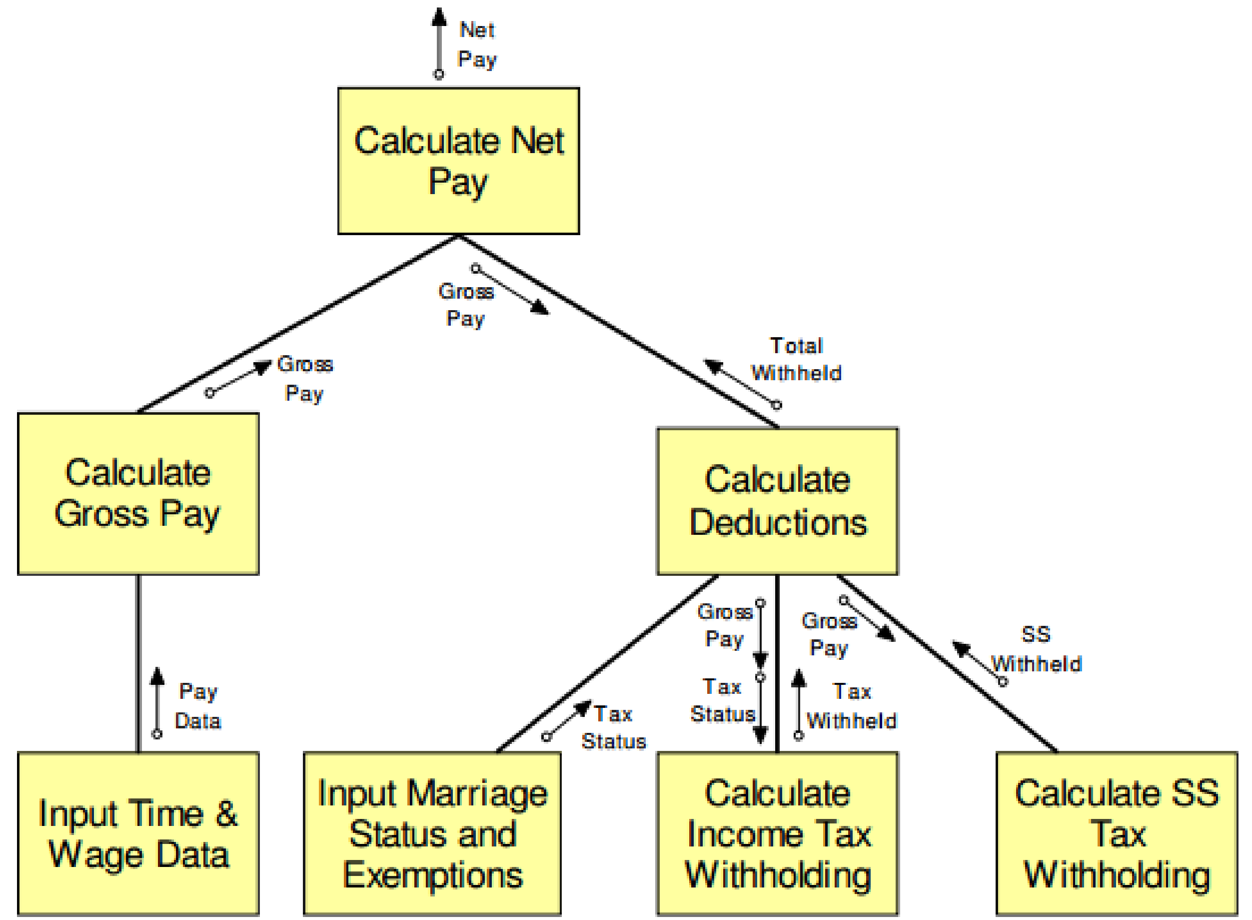
FIGURE 6.2 Hierarchical structure of a university shown as a tree.



in the software world!

Example

This structure chart shows the hierarchical relationship between the methods within a computer program. Here the line symbol represents a method call and the data arrows represent arguments and return values. For instance, it shows that the program has a method to *Calculate Deductions* that receives the *Gross Pay* as an argument and returns the *Total Withheld*.



Source: http://faculty.salina.k-state.edu/tmertz/Java/210proceduralprogramdesign/02proceduraldesignmethodology.pdf

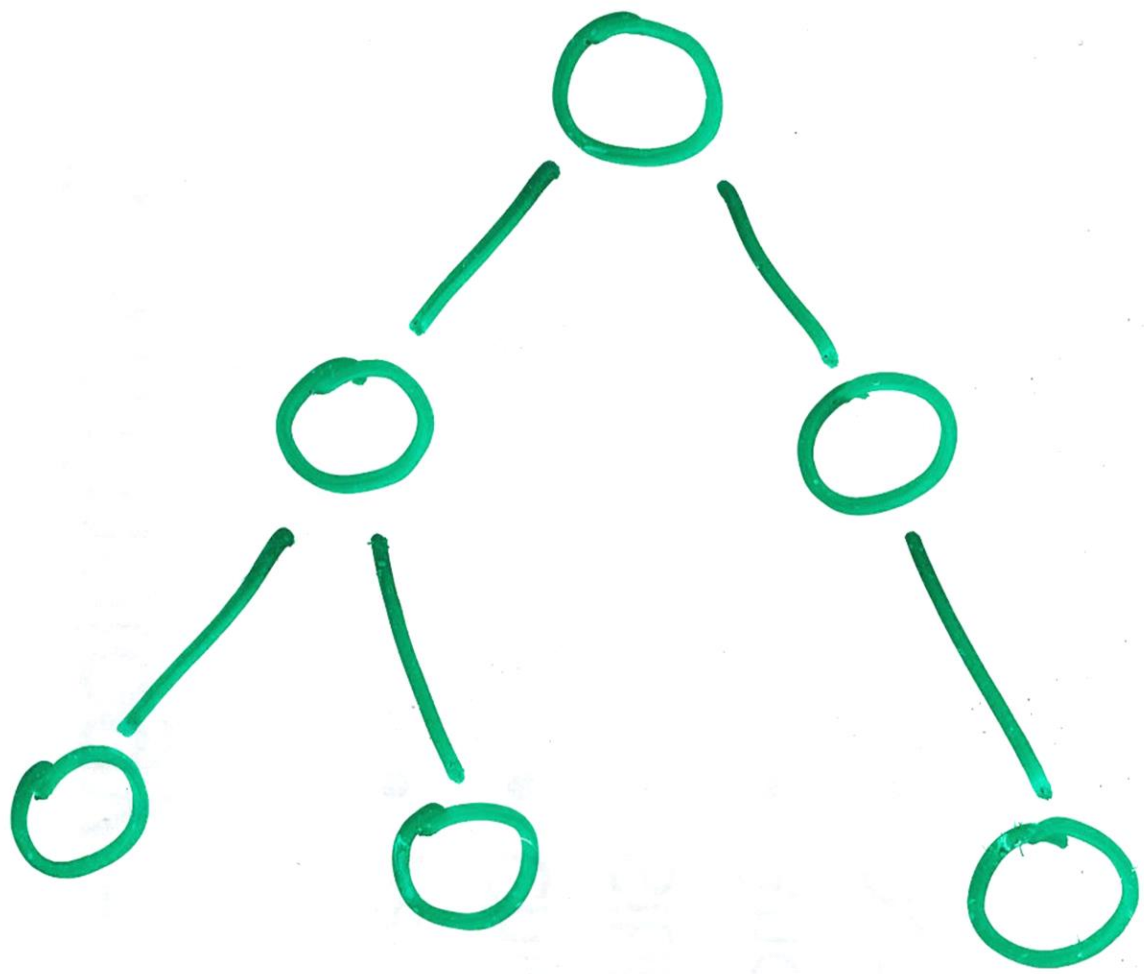
What is a (rooted) **Tree**?

- Definition:
  - A set of zero or more **nodes** partitioned into a **root node** and **subtrees** (of the root)
  - **Root node** is the access point into tree
  - **Subtree** is a node together with its descendants
  - A **tree** must be connected and have no cycle (**acyclical**)

- NOTE: ***node*** != object of class Node (with which we use to build linked lists)

Tree terminology

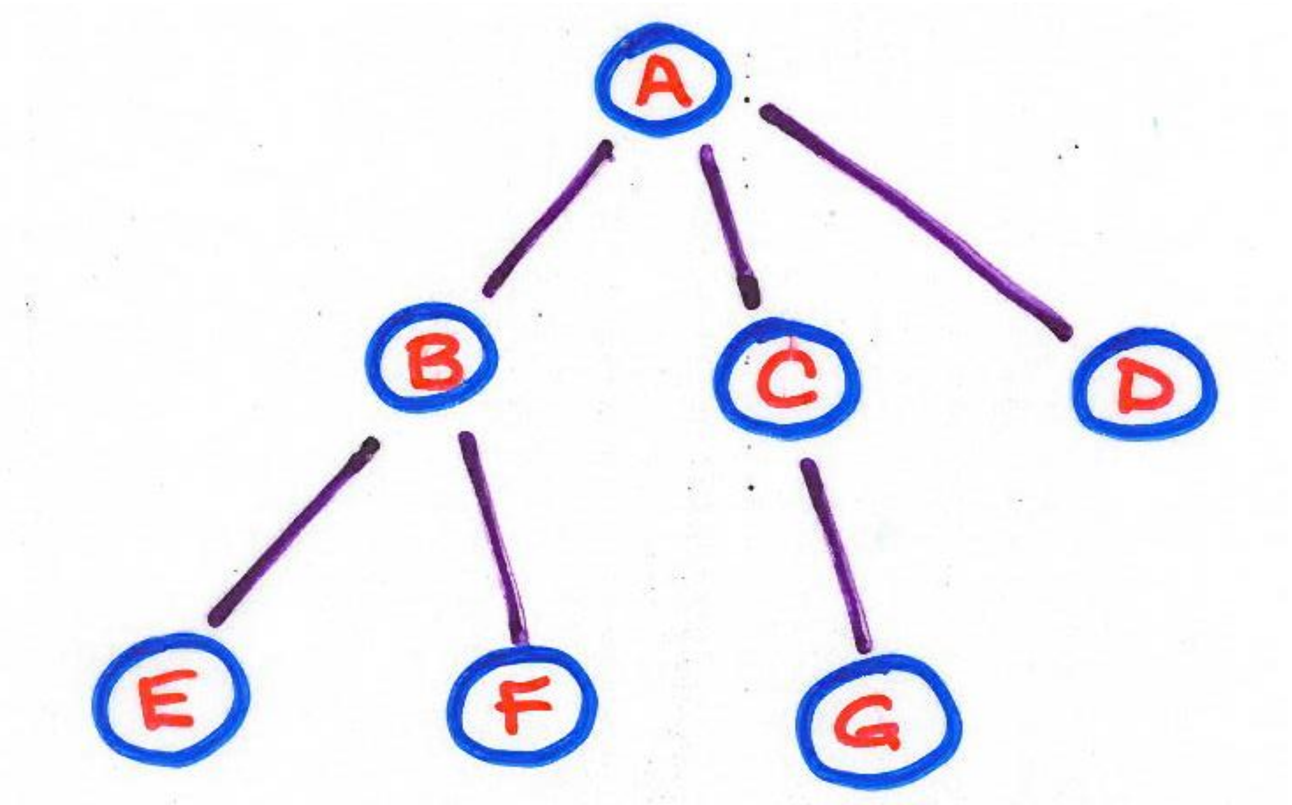
- Root
- Node (**v**ertex)
- Edge
- Leaf
- Parent
- Child
- Sibling
- Ancestor
- Descendant
- Adjacent nodes
- Subtree



Definition of tree terminology

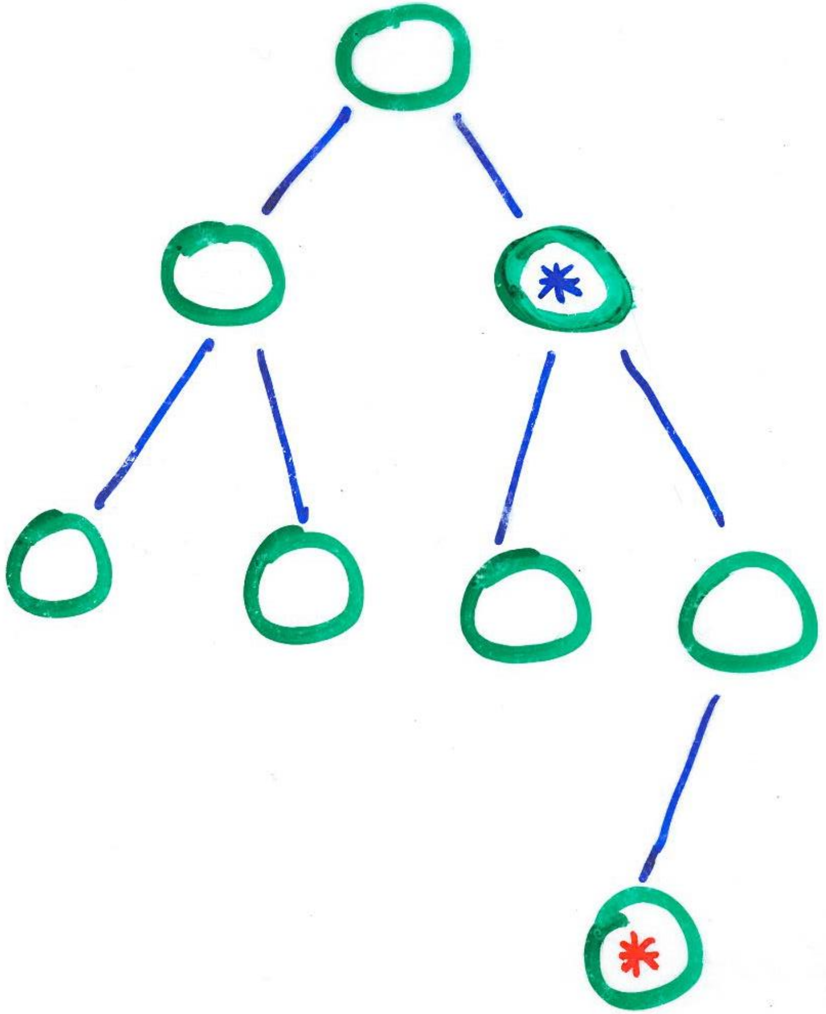
- **Parent:** The predecessor of a node
- **Child**: Any successor of a node
- **Siblings**: Any pair of nodes that have the same parent
- **Ancestor**: The predecessor of a node together with all the ancestors of the predecessor of a node. The root node has no ancestors
- **Descendant**: The children of a node together with all the descendants of the children of a node. A leaf node has no descendants
  - If there is a path from node **a** to node **b**, and **a** is "above" **b**, then **a** is called an ***ancestor*** of **b** and **b** is called a ***descendant*** of **a**

Recursive nature of tree and subtrees



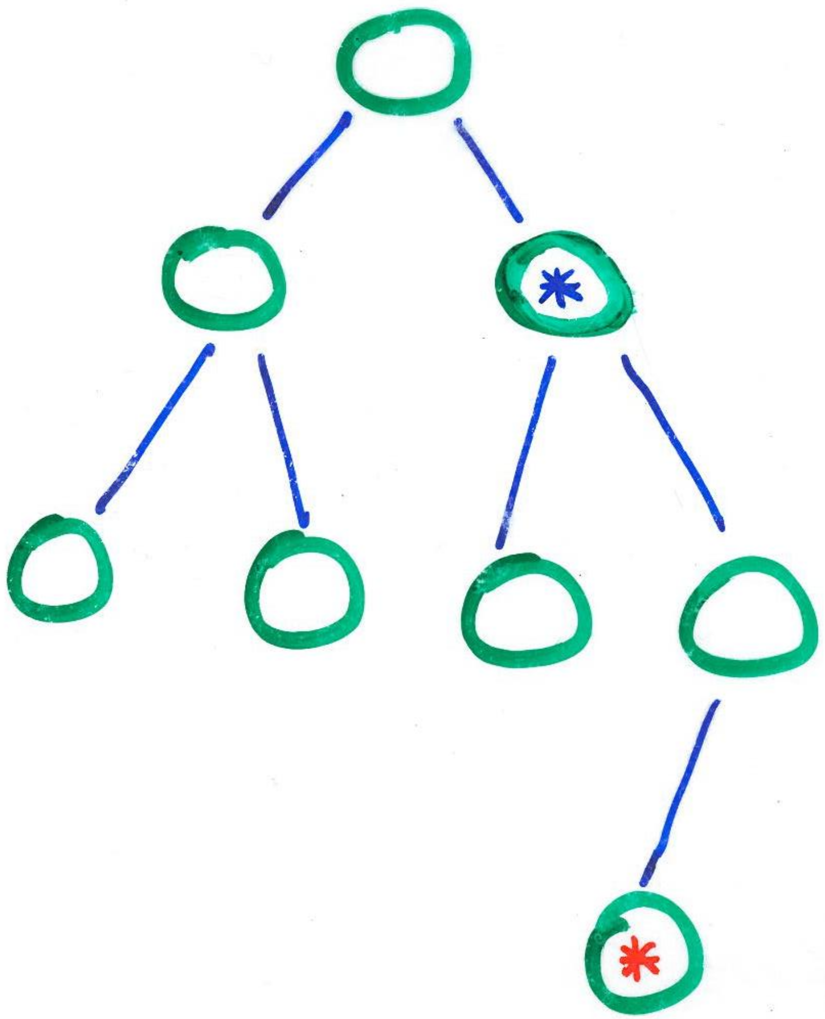
Path and Path Length

- A **path** is a sequence of nodes $\mathbf{v}_1 \ldots \mathbf{v}_\mathbf{m}$ where $\mathbf{v}_i$ is a parent of $\mathbf{v}_{\mathbf{i}+1}$ $(1 \leq \mathbf{i} \leq \mathbf{m}-1)$
- The **path length** is the number of nodes in a path
  - From node $\mathbf{v}_1$ to another node $\mathbf{v}_k$: path length is $\mathbf{k}$
- **Property of tree**: unique **path** from root node to any other node



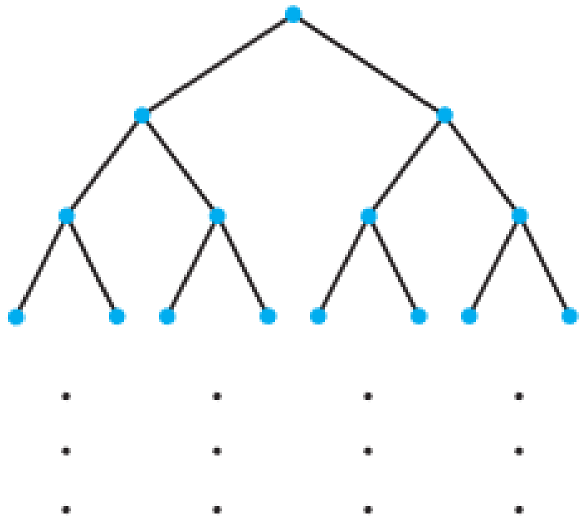
Height

- Textbook: The **height** of a tree **T** is the number of nodes on the longest path from the root to a leaf
- The **height** of a node **v** is the length of longest path from node **v** to a leaf
- The **height** of a tree **T** is the height of **T**'s root



From our Textbook:

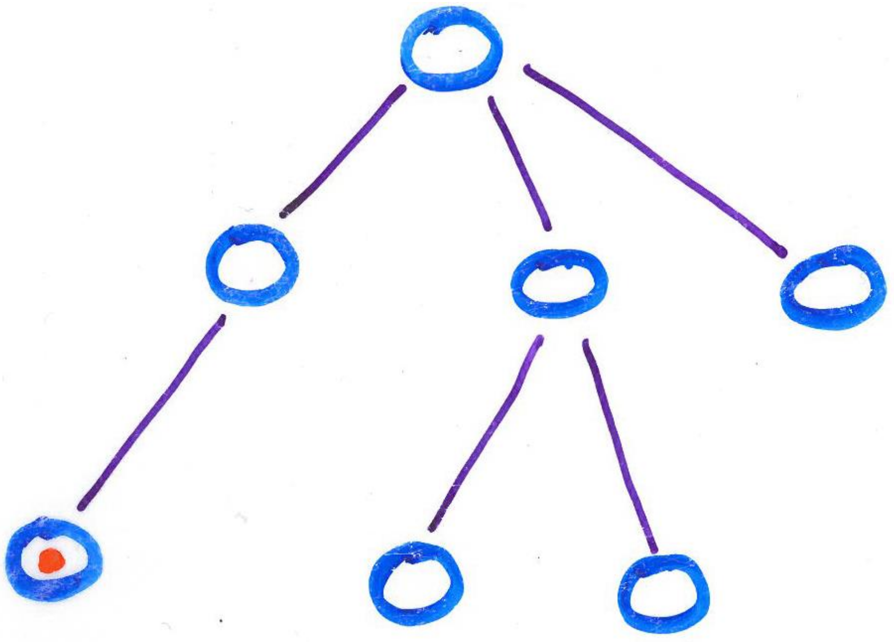
FIGURE 15-9 Counting the nodes in a full binary tree of height h



Level	Number of nodes at this level	Total number of nodes at this level and all previous levels
1	$1 = 2^0$	$1 = 2^1 - 1$
2	$2 = 2^1$	$3 = 2^2 - 1$
3	$4 = 2^2$	$7 = 2^3 - 1$
4	$8 = 2^3$	$15 = 2^4 - 1$
⋮	⋮	⋮
h	2^{h-1}	$2^h - 1$

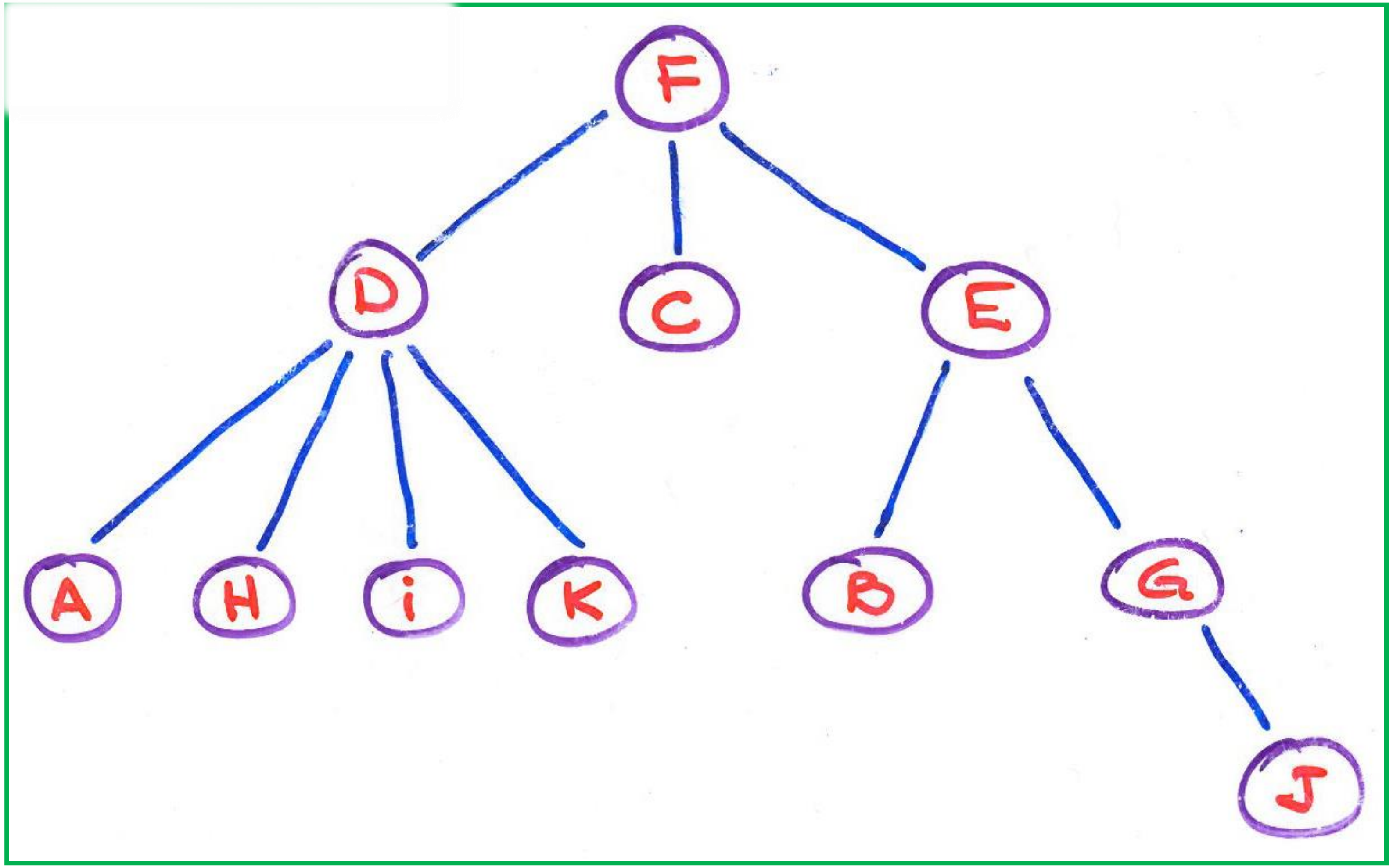
Degree of a node

- Number of edges that touch a node **v**
- **Internal node** has a degree > 1
- **External node** has a degree == 1 or == 0



Activity

- Parent of C?
- Parent of H?
- Children of E?
- Children of C?
- Root?
- Path length F to J?
- Ancestors of G?
- Descendants of E?
- Siblings of D?
- Adjacent node of E?
- Height of tree?
- Number of levels in tree?



Tree Classification

- We classify trees by the maximum number of **children** (or subtrees) a node of the tree can have:
  - **N-ary tree**

<u>Textbook</u>: An ***n*-ary tree** is a set T of nodes that is either empty or partitioned into disjoint subsets:

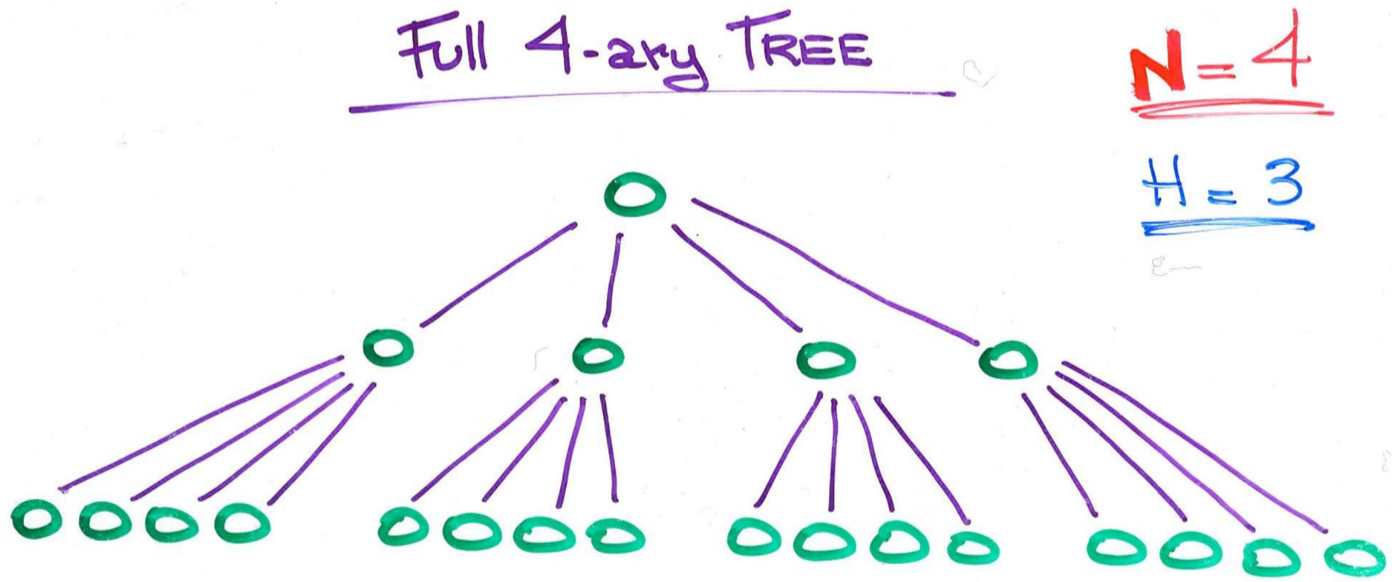
- A single node r, the root
- n possibly empty sets that are n-ary subtrees of r

Each node can have no more than n children.

- Examples:

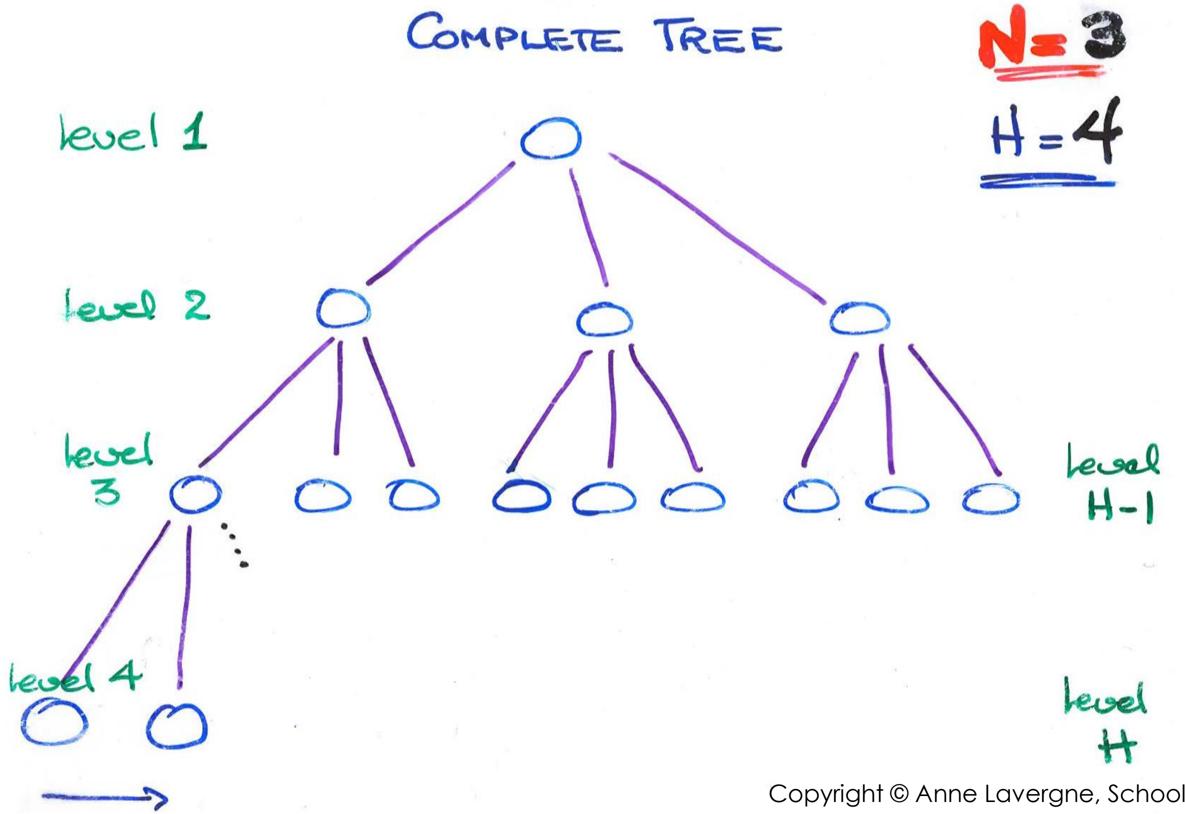
Full tree of height **H**

- **Full tree** of height **H** is a tree in which all nodes at level < **H** have max number of children
- Nodes @ level **H** have 0 children (they are leaves)



Complete tree of height **H**

- **Complete tree** of height **H** is a tree that is **full** all the way down to level **H-1** with level **H** filled in from left to right without any gap



Examples of **complete trees**

Balanced tree

- A **N-ary tree** is height balanced, or simply **balanced**, if the height of any node's right subtree differs from the height of the node's left subtree by at most 1.

√ Learning Check

- We can now …
  - Define some tree-related terms and concepts

Next Lecture

- Describe **binary tree** and its properties
- Describe **binary search tree** and its properties
- Given a **binary search tree**, perform some operations such as:
  - Insert an element (a node containing an element)
  - Retrieve (get) an element
  - Remove an element