

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Discrete Structures 2

Trees

Computer Science and Artificial Intelligence Department
Umm Al-Qura University



Acknowledgment: Slides are mainly adopted from Rosen's book supplements.

AGENDA

جامعة أم القرى
UMM AL-QURA UNIVERSITY

- Introduction to Trees
- Application of Trees



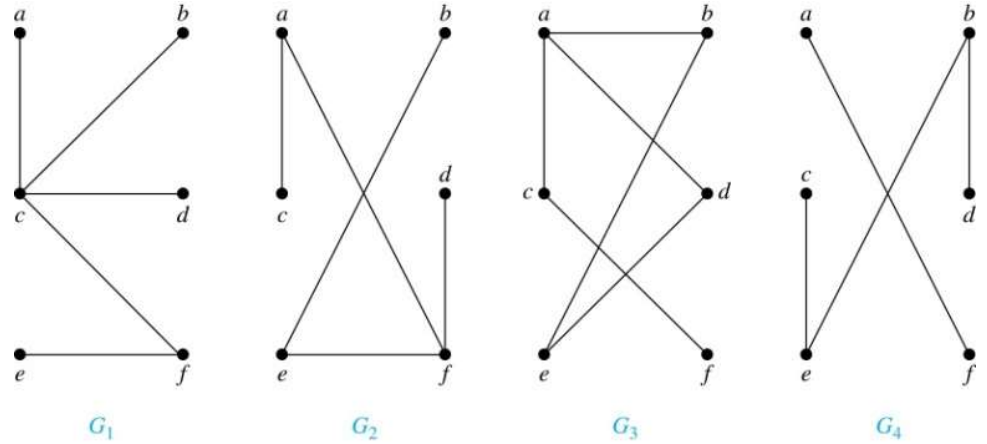


Introduction to Trees

Trees

Definition: A ***tree*** is a **connected undirected** graph with **no simple circuits**.

Example: Which of these graphs are trees?



Solution: G_1 and G_2 are trees - both are connected and have no simple circuits.
G_3 is not a tree because *e*, *b*, *a*, *d*, *e* is a simple circuit.
G_4 is not a tree because it is not connected.

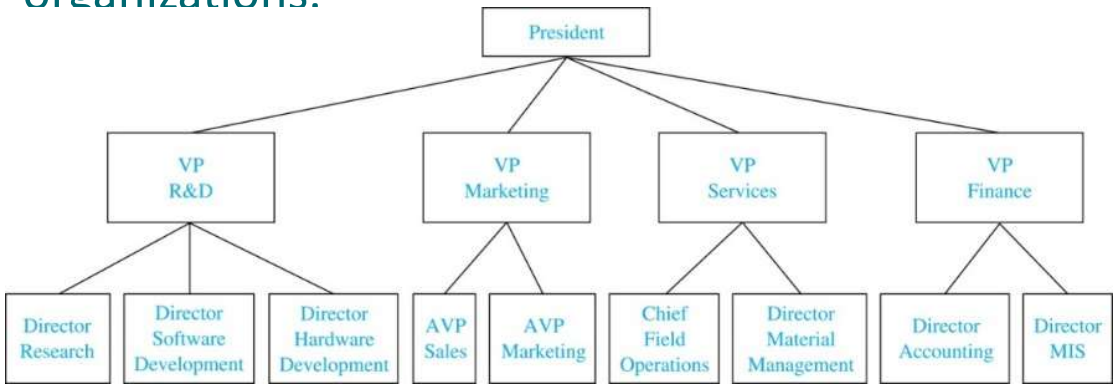
Trees

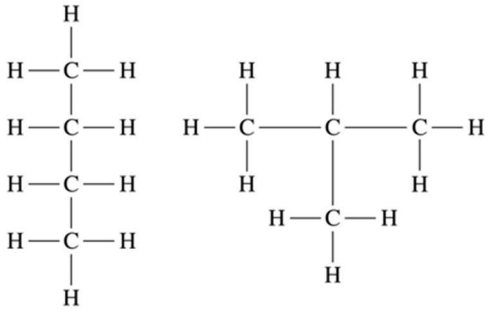
جامعة أم القرى
UMM AL-QURA UNIVERSITY

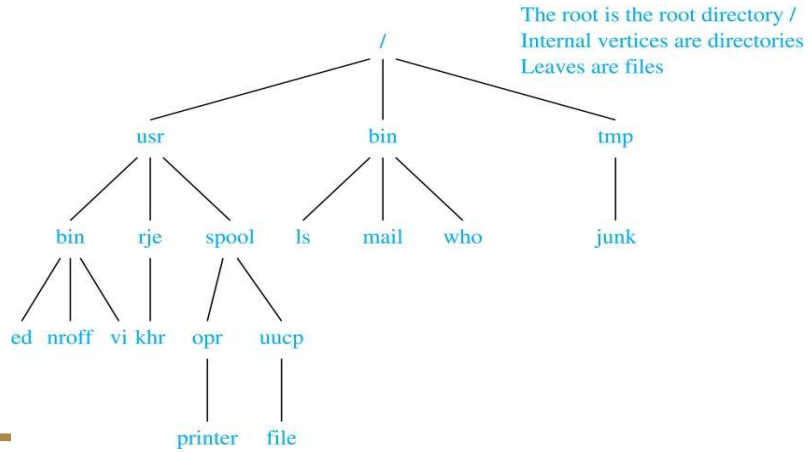
Theorem: An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Trees as Models

- Trees are used as models in computer science, chemistry, geology, botany, psychology, and many other areas.
- The organization of a computer file system into directories, subdirectories, and files is naturally represented as a tree.
- Trees are used to represent the structure of organizations.



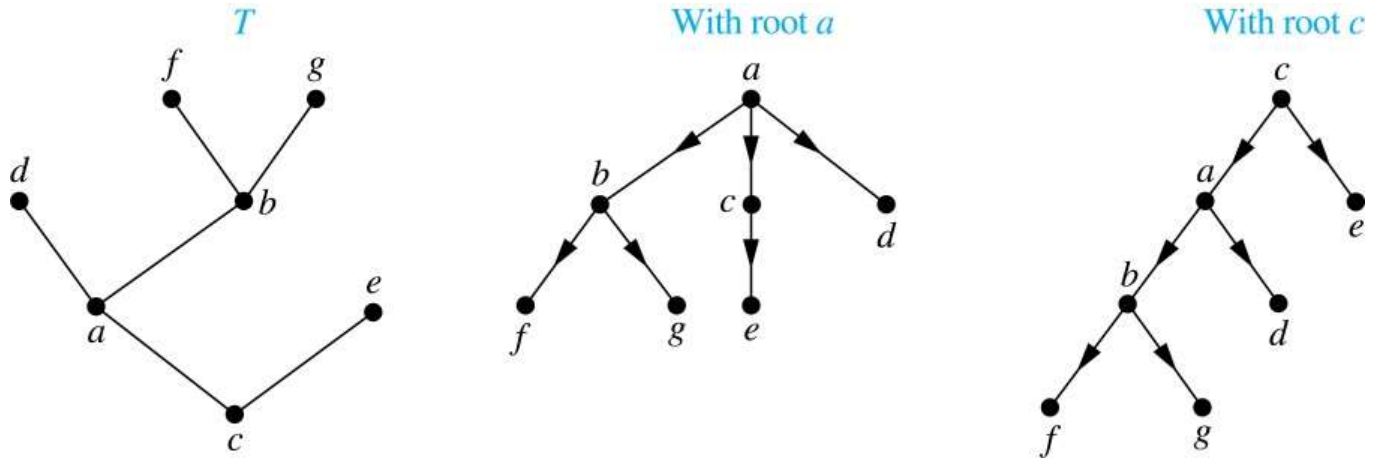




Rooted Trees

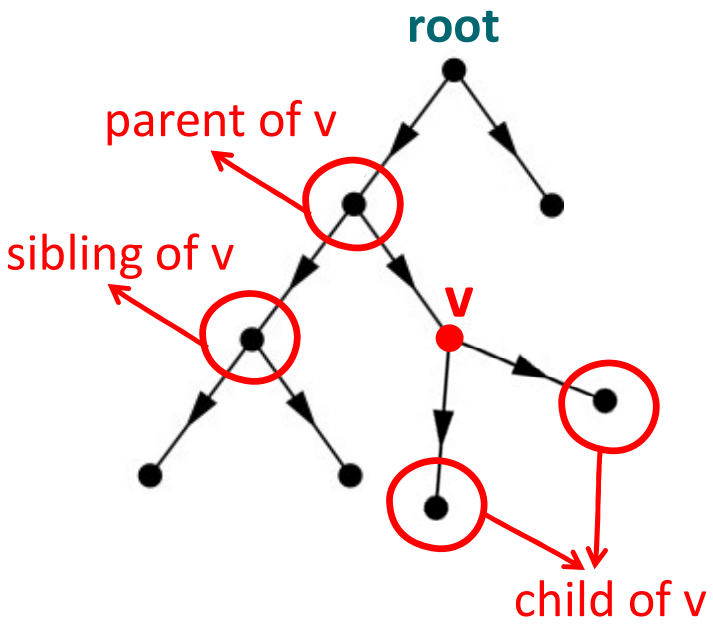
Definition: A ***rooted tree*** is a tree in which one vertex has been designated as the *root* and every edge is **directed away** from the root.

An **unrooted tree** is converted into different rooted trees when different vertices are chosen as the root.



Rooted Tree Terminology

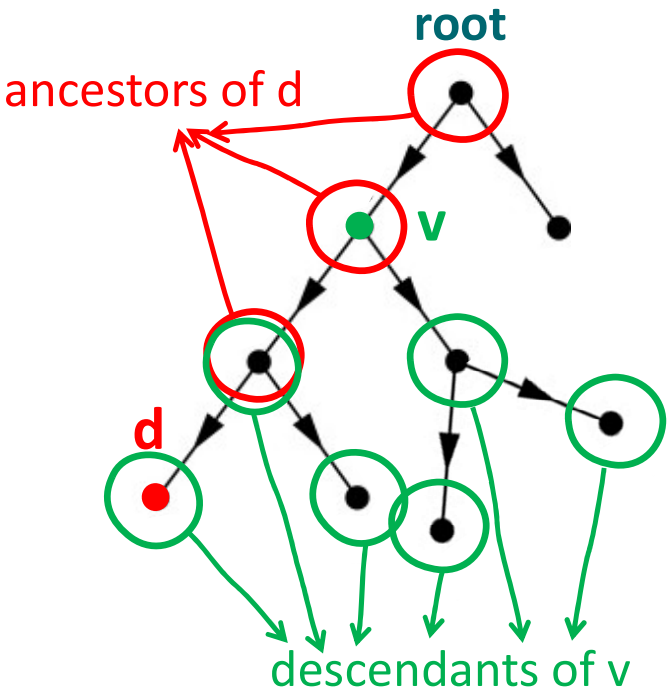
- If *v* is a vertex of a rooted tree other than the root,
- the *parent* of *v* is the unique vertex *u* such that there is a directed edge from *u* to *v*.
- When *u* is a parent of *v*, *v* is called a *child* of *u*.
- Vertices with the same parent are called *siblings*.



Rooted Tree Terminology

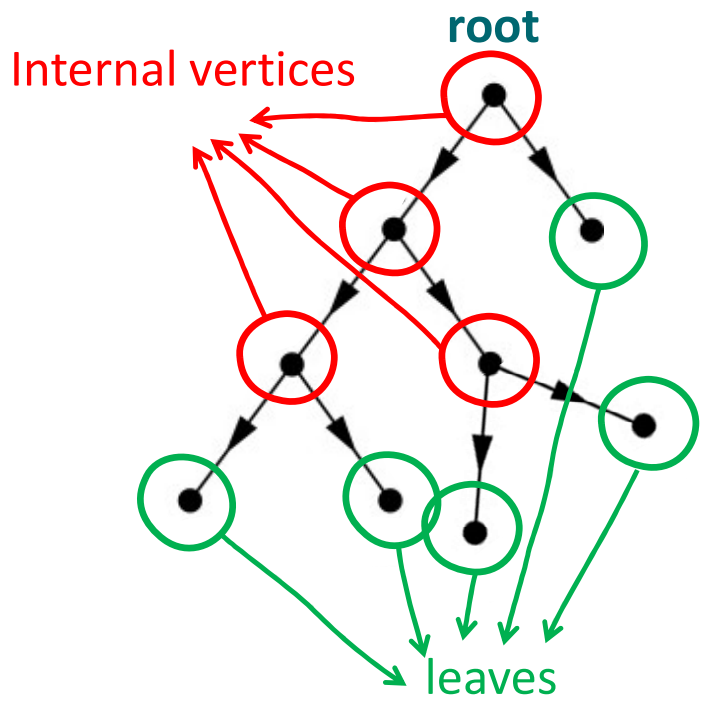
- The *ancestors* of a vertex are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.

- The *descendants* of a vertex *v* are those vertices that have *v* as an ancestor.



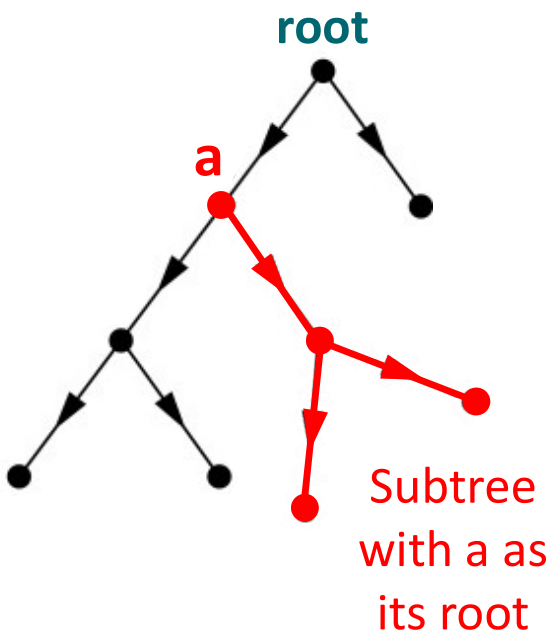
Rooted Tree Terminology

- A vertex of a rooted tree with no children is called a *leaf*.
- Vertices that have children are called *internal vertices*.



Rooted Tree Terminology

- If *a* is a vertex in a tree, the *subtree* with *a* as its root is the subgraph of the tree consisting of *a* and its descendants and all edges incident to these descendants.



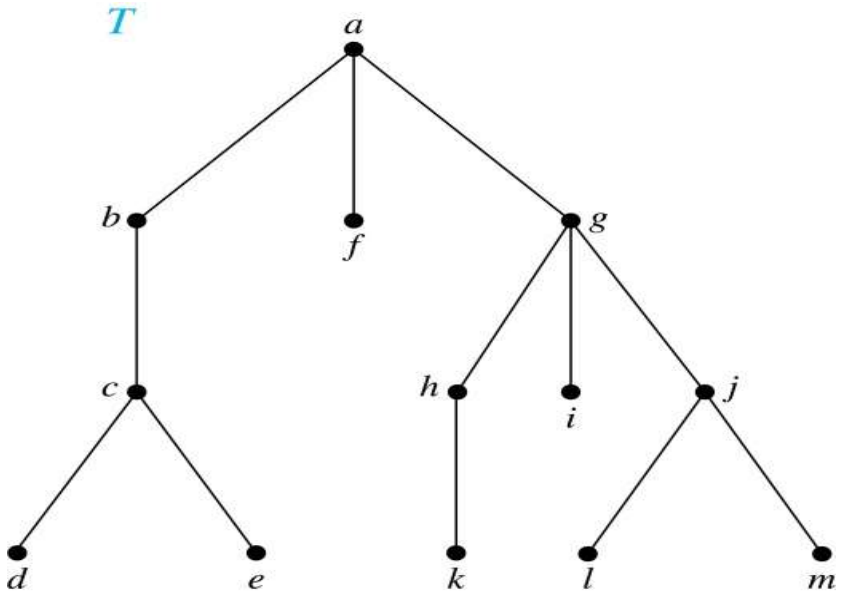
Terminology for Rooted Trees

Example: In the rooted tree ***T*** (with root ***a***):

(i) Find the parent of c, the children of g, the siblings of h, the ancestors of e, and the descendants of b.

(ii) Find all internal vertices and all leaves.

(iii) What is the subtree rooted at g?



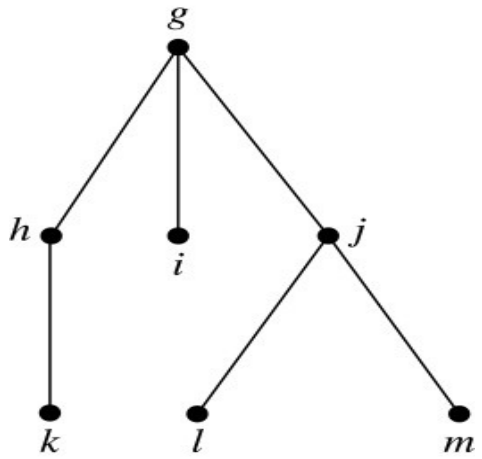
Terminology for Rooted Trees

Solution:

(i) The parent of *c* is *b*. The children of *g* are *h*, *i*, and *j*. The siblings of *h* are *i* and *j*. The ancestors of *e* are c, *b*, and *a*. The descendants of *b* are *c*, *d*, and *e*.

(ii) The internal vertices are a, b, c, g, h, and j. The leaves are d, e, f, i, k, l, and m.

(iii) We display the subtree rooted at g.

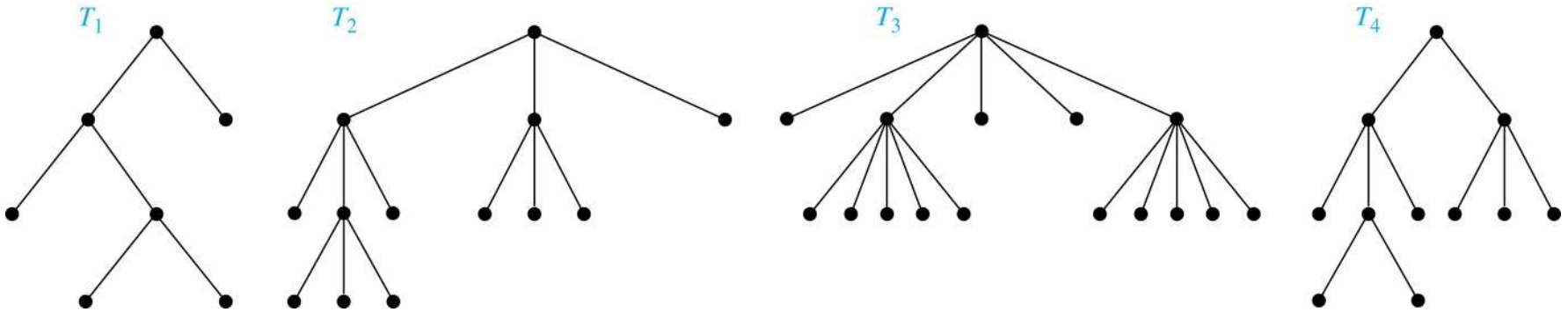


m-ary Rooted Trees

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Definition: A rooted tree is called an ***m-ary tree*** if every internal vertex has no more than *m* children. The tree is called a ***full m-ary tree*** if every internal vertex has exactly *m* children. An *m*-ary tree with $m = 2$ is called a ***binary* tree**.

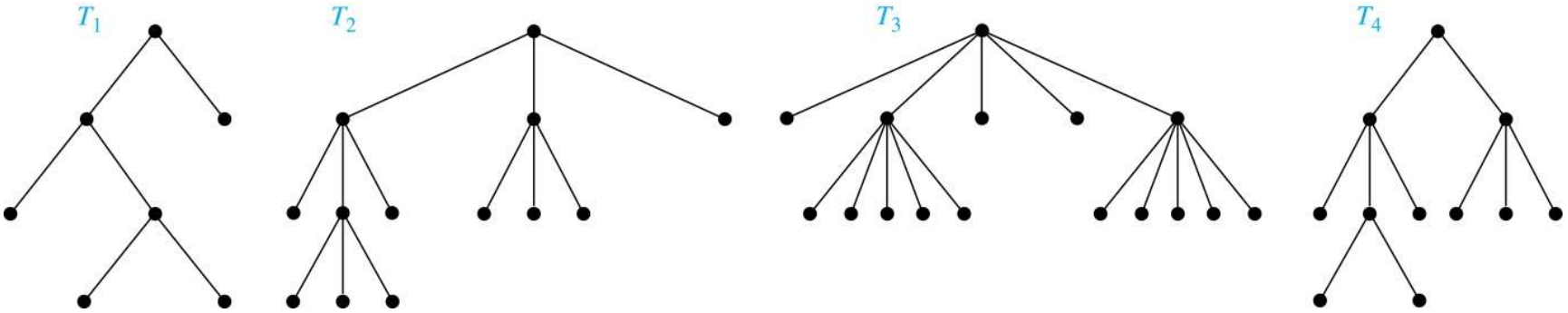
Example: Are the following rooted trees full *m*-ary trees for some positive integer *m*?



m-ary Rooted Trees

Solution:

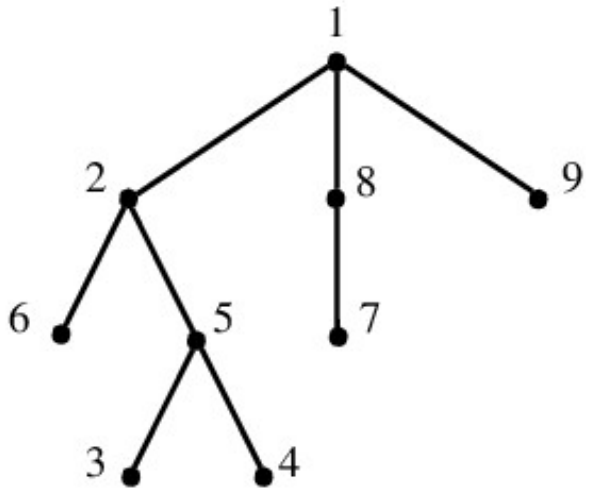
- T_1 is a full binary tree because each of its internal vertices has two children.
- T_2 is a full 3-ary tree because each of its internal vertices has three children.
- In T_3 each internal vertex has five children, so T_3 is a full 5-ary tree.
- T_4 is not a full *m*-ary tree for any m because some of its internal vertices have two children and others have three children.



Ordered Rooted Trees

Definition: An *ordered rooted tree* is a rooted tree where the children of each internal vertex are ordered.

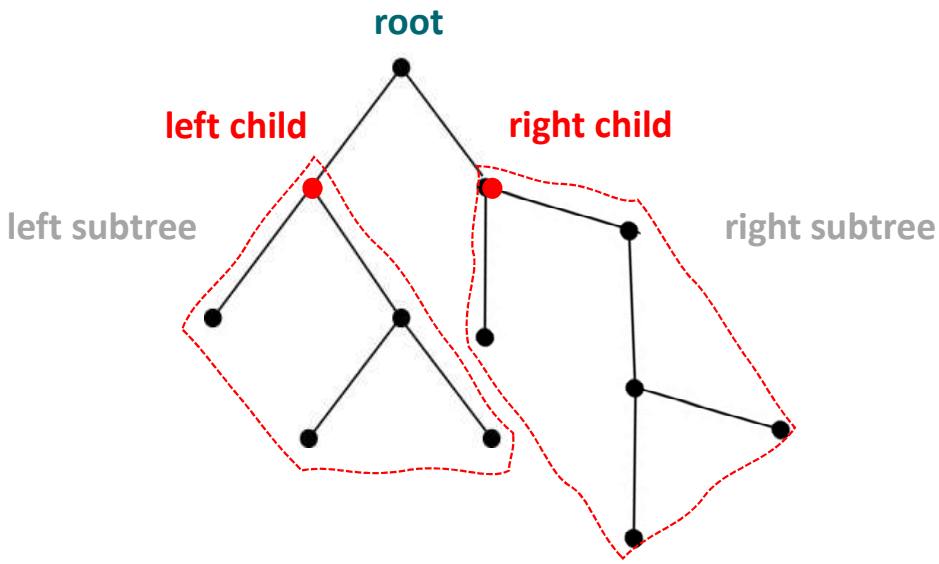
– We draw ordered rooted trees so that the children of each internal vertex are shown in order from left to right.



Ordered Rooted Trees

جامعة أم القرى
UMM AL-QURA UNIVERSITY

- **Definition**: A ***binary tree*** is an ordered rooted where each internal vertex has at most two children.
- If an internal vertex of a binary tree has two children, the first is called the *left child* and the second the *right child*.
- The tree rooted at the left child of a vertex is called the *left subtree* of this vertex, and the tree rooted at the right child of a vertex is called the *right subtree* of this vertex.

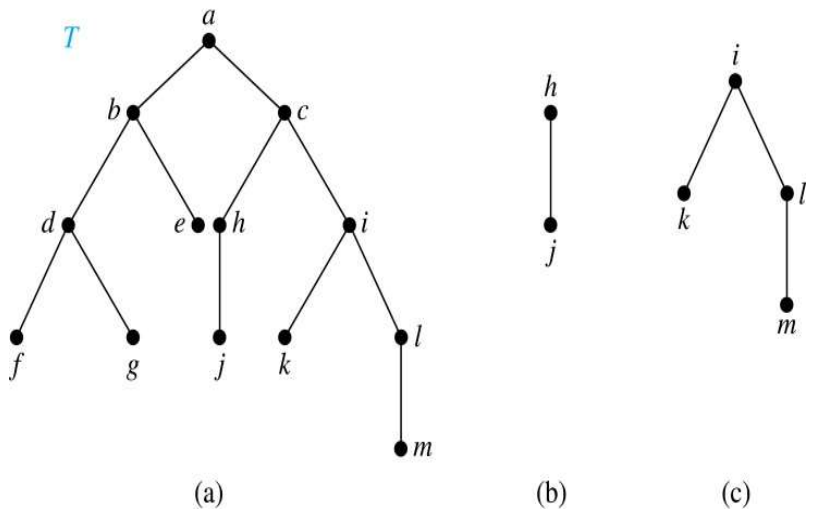


Ordered Rooted Trees

Example: Consider the binary tree T.

(*i*) What are the left and right children of d?

(*ii*) What are the left and right subtrees of c?



Solution:

(*i*) The left child of d is f and the right child is g.

(*ii*) The left and right subtrees of c are displayed in (b) and (c).

Properties of Trees

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Counting Edges in a Tree:

- **Theorem 2**: A tree with n vertices has $n - 1$ edges.

Counting Vertices in Full m-ary Trees:

- **Theorem 3**: A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.

Proof : Every vertex, except the root, is the child of an internal vertex. Because each of the i internal vertices has m children, there are mi vertices in the tree other than the root. Hence, the tree contains $n = mi + 1$ vertices.

Counting Vertices in Full *m*-ary Trees

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Theorem 4: A full *m*-ary tree with:

(*i*) *n* vertices has $i = (n-1)/m$ internal vertices and
$l = [(m-1)n+1]/m$ leaves,

(*ii*) *i* internal vertices has $n = mi + 1$ vertices and
$l = (m-1)i + 1$ leaves,

(*iii*) *l* leaves has $n = (ml-1)/(m-1)$ vertices and
$i = (l-1)/(m-1)$ internal vertices.

Level of Vertices & Height of Trees

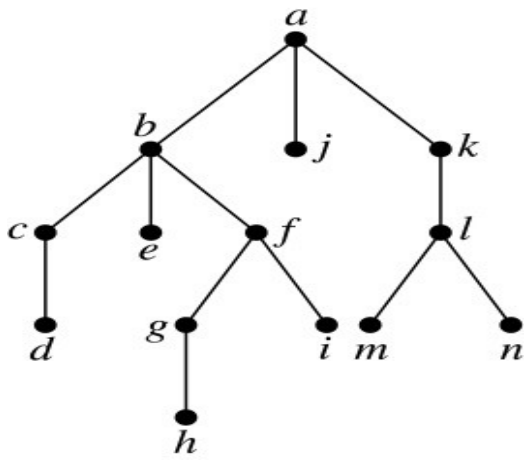
جامعة أم القرى
UMM AL-QURA UNIVERSITY

- **Definitions:**
- The ***level*** of a vertex *v* in a rooted tree is the length of the unique path from the root to this vertex.
- The ***height*** of a rooted tree is the maximum of the levels of the vertices.

Example:

(*i*) Find the level of each vertex in the tree to the right.

(*ii*) What is the height of the tree?



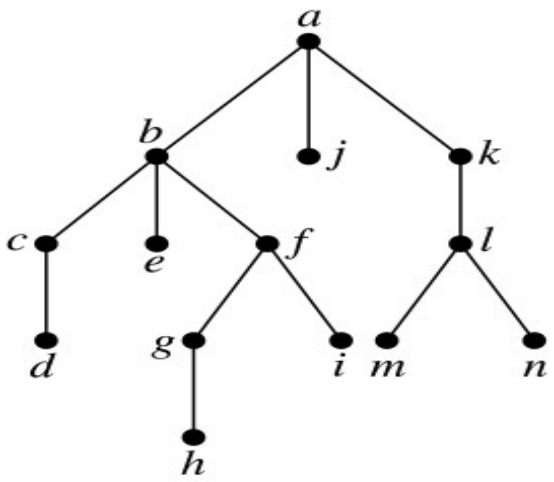
Level of Vertices & Height of Trees

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Solution:

(*i*) The root *a* is at **level** 0. Vertices *b*, *j*, and *k* are at **level 1** Vertices *c*, *e*, *f*, and *l* are at **level** 2. Vertices *d*, *g*, *i*, *m*, and *n* are at **level** 3. Vertex *h* is at **level** 4.

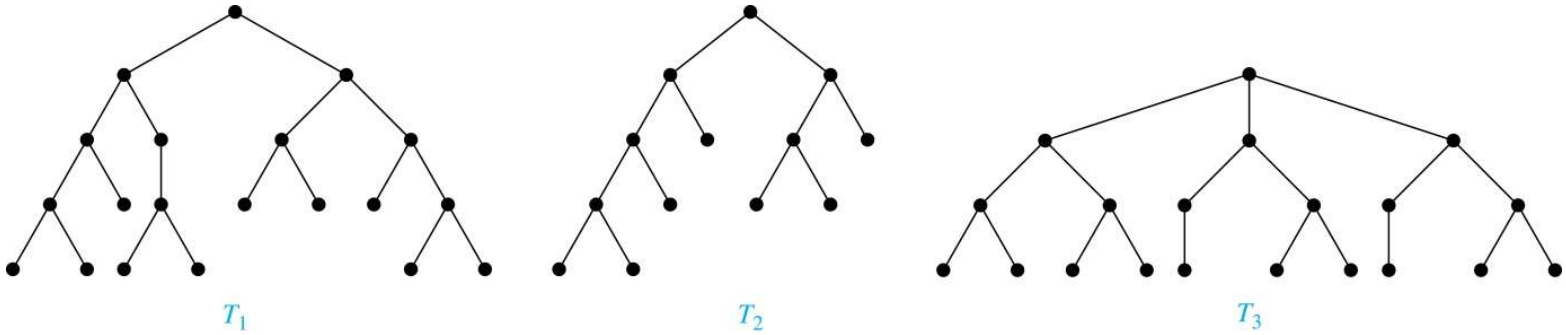
(*ii*) The height is 4, since 4 is the largest level of any vertex.



Balanced *m*-ary Trees

Definition: A rooted *m*-ary tree of height *h* is *balanced* if all leaves are at levels *h* or $h - 1$.

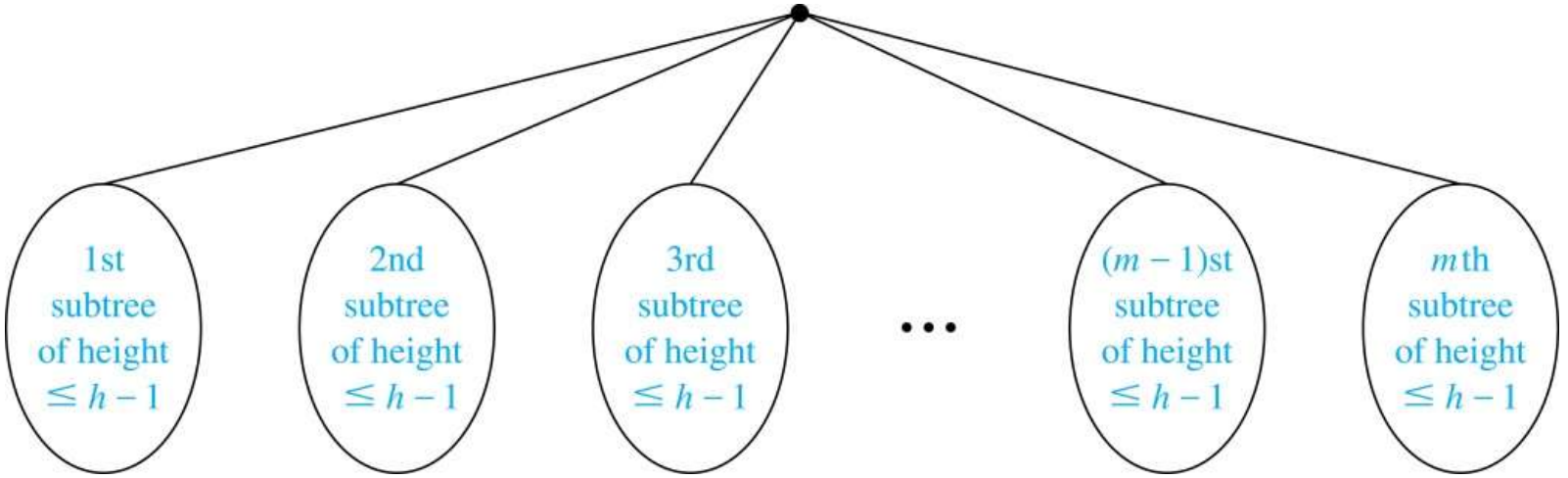
Example: Which of the rooted trees shown below is balanced?



Solution: T_1 and T_3 are balanced, but T_2 is not because it has leaves at levels 2, 3, and 4.

The Bound for the Number of Leaves in an m-ary Tree

Theorem 5: There are **at most** m^h leaves in an m-ary tree of height h.





جامعة أم القرى
UMM AL-QURA UNIVERSITY

Applications of Trees

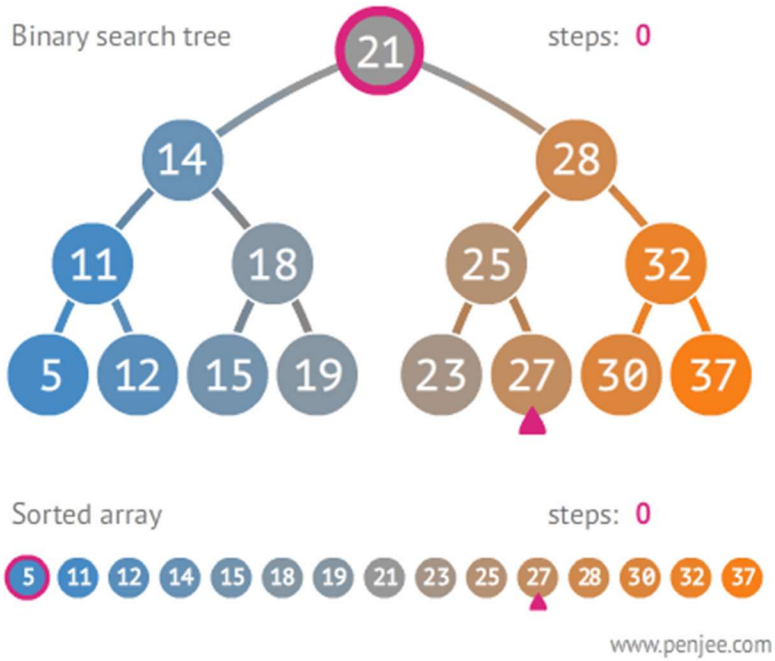
Applications of Trees



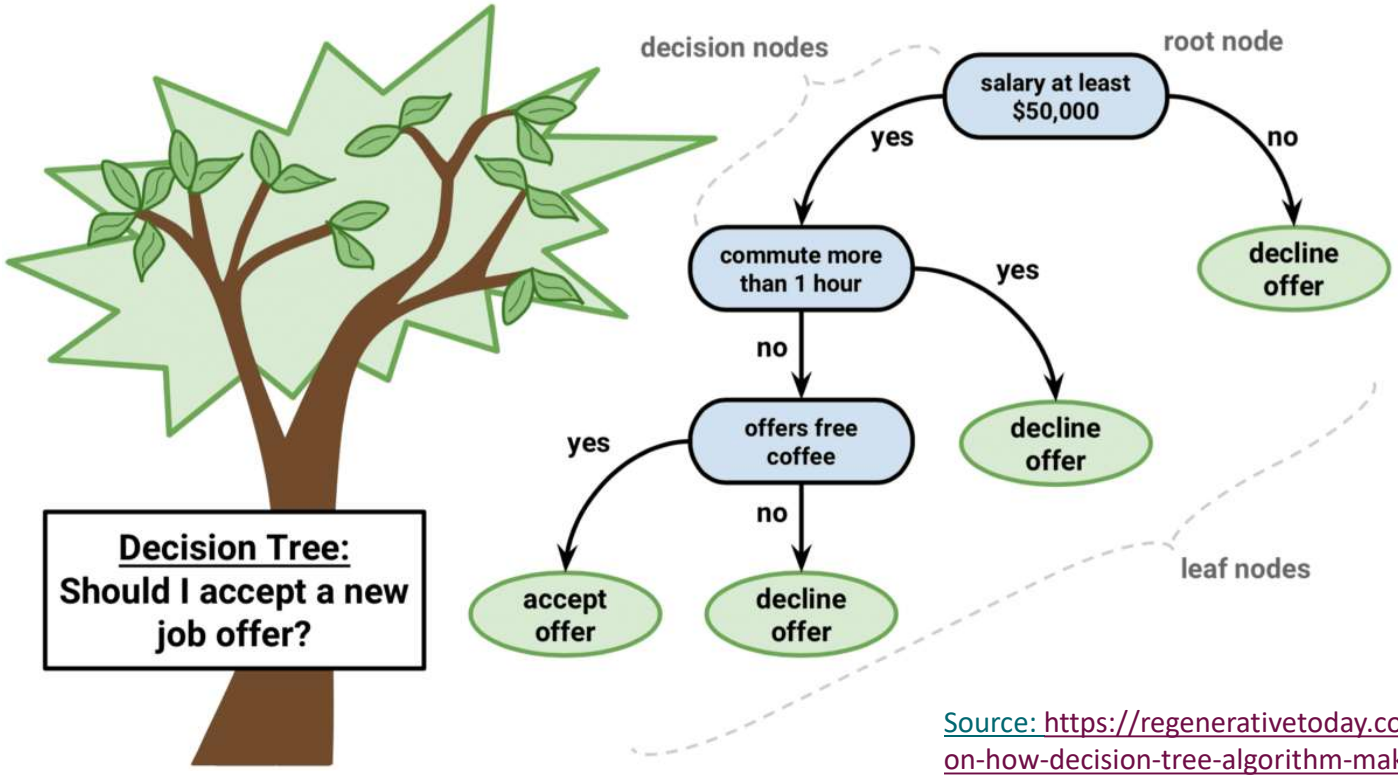
- Binary Search Trees
- Decision Trees
- Huffman Coding

Binary Search Trees

- Uses ordered binary trees.

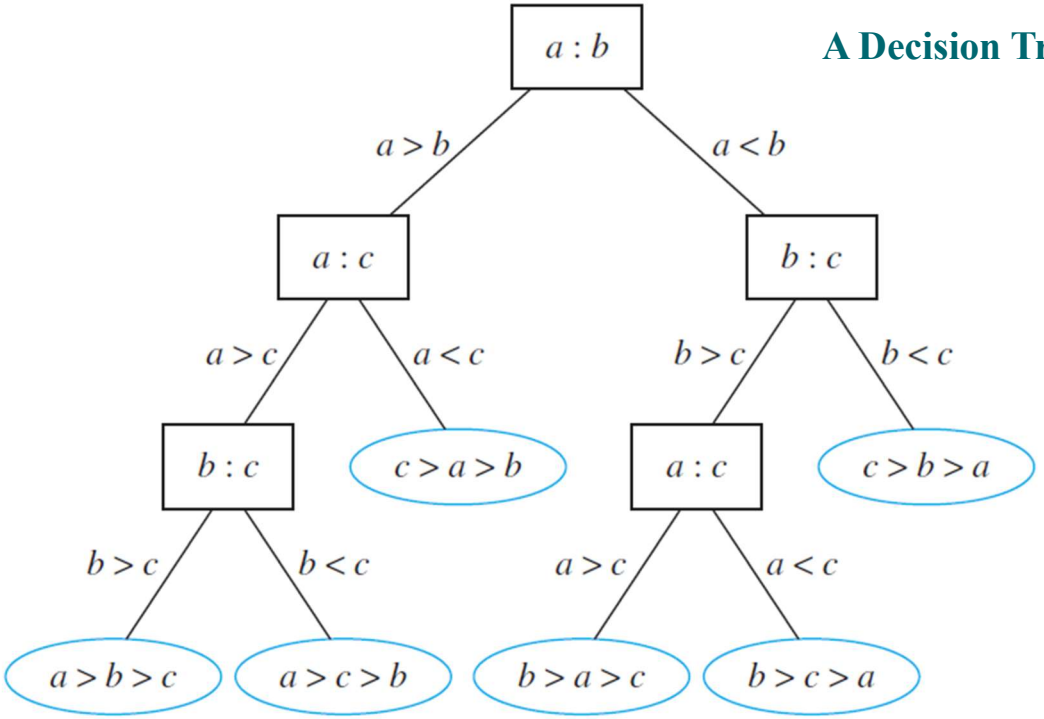


Decision Trees



Source: https://regenerativetoday.com/simple-explanation-on-how-decision-tree-algorithm-makes-decisions

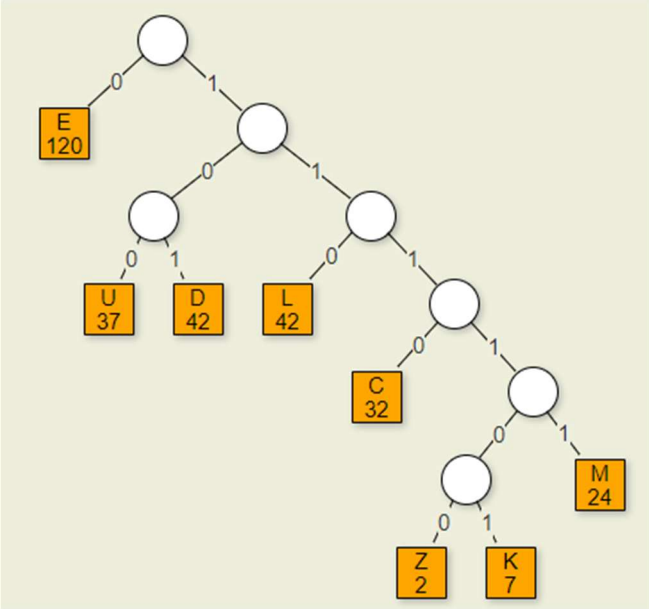
Decision Trees



A Decision Tree for Sorting Three Distinct Elements

Huffman Coding

- Huffman coding assigns codes to characters such that the length of the code depends on the relative frequency or weight of the corresponding character.
- It is a variable-length code.
- Using the given Huffman tree,
  - ‘L’ = 110 Encoding
  - ‘Z’ = 111100 Encoding
  - 101 = ‘D’ Decoding
  - 1110 = ‘C’ Decoding



https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/Huffman.html

Huffman Coding



- Building Huffman tree:
- Assume you have this frequency table

Letter	C	D	E	K	L	M	U	Z
Frequency	32	42	120	7	42	24	37	2

1. Assume each letter is a single-leaf tree
2. Sort trees ascendingly according to frequency of their roots.
3. Join the first two trees (with lowest weights) to form ordered binary tree. The root will have the sum of the weights from the root of the two trees.
4. Assign 0 to edge of left child and 1 to edge of the right child
5. Place the new tree with the list of trees.
6. Repeat steps 2-5 until single root is left.

Check: https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/Huffman.html

What is Next?

جامعة أم القرى
UMM AL-QURA UNIVERSITY

- **Read Sections 11.1 and 11.2** from Rosen's Book (8th Ed).

- **Solve** the **Tutorial Exercises** sheet on BlackBoard (practical section) before the practical session.