National University
of computer and emerging sciences

DISCRETE STRUCTURES

COURSE INSTRUCTOR: MUHAMMAD SAIF UL ISLAM

Course Outline

- **Logic and Proofs** (Chapter 1)
- **Sets and Functions** (Chapter 2)
- **Relations** (Chapter 9)
- **Number Theory** (Chapter 4)
- Combinatorics and Recurrence
- **Graphs** (Chapter 10)
- **Trees** (Chapter 11)
- Discrete Probability

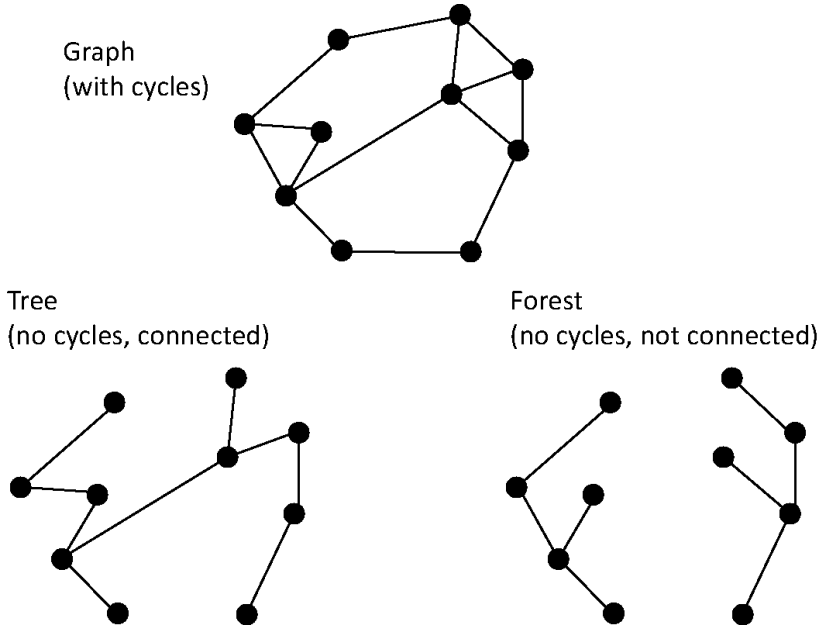
Lecture Outline

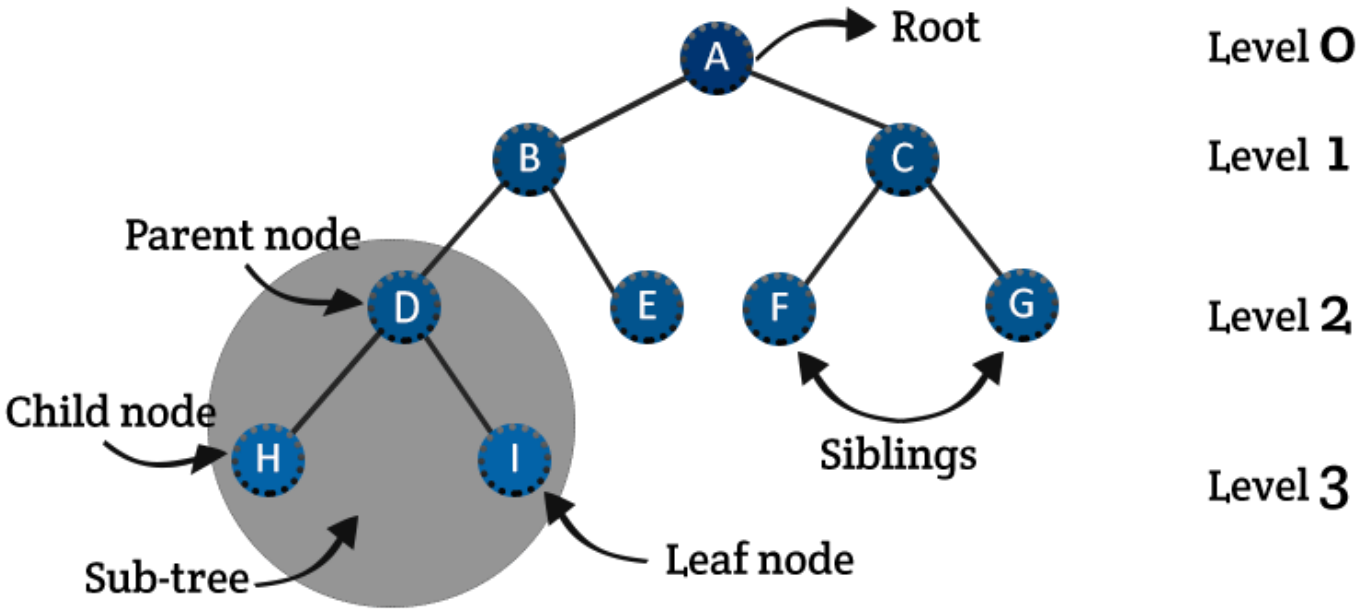
- Introduction to Trees
- Rooted Trees
- Properties of Trees
- Applications
- Tree Traversal
- Spanning Trees
- Minimum Spanning Tree Algorithms

Trees

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

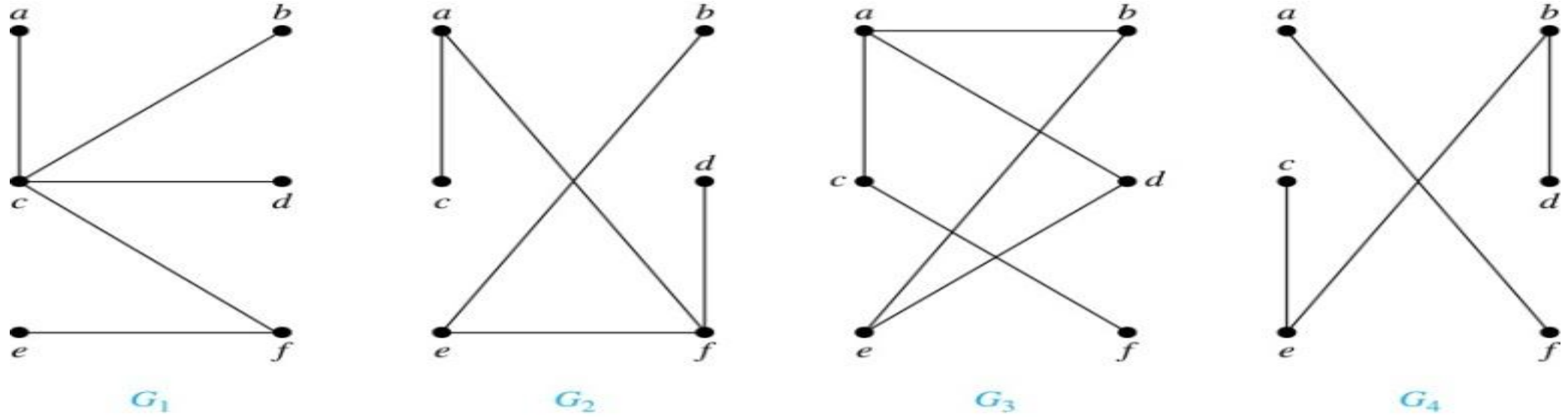
Definition: An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices. A tree cannot contain multiple edges or loops.





Trees

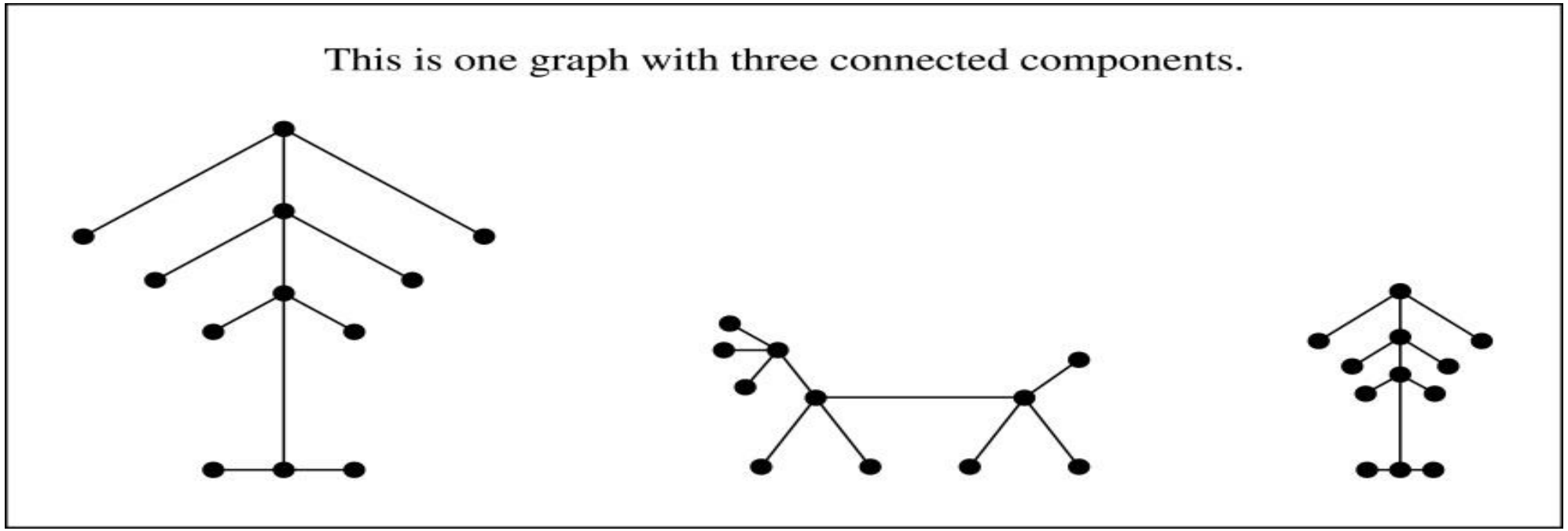
Example: Which of these graphs are trees?



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.

FOREST

Definition: A *forest* is a graph that has no simple circuit, but is not connected. Each of the connected components in a forest is a tree.

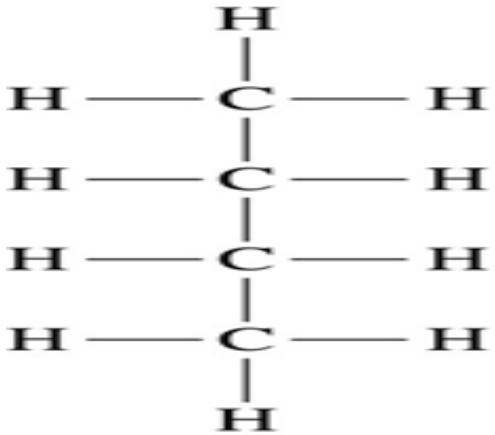


Trees as Models

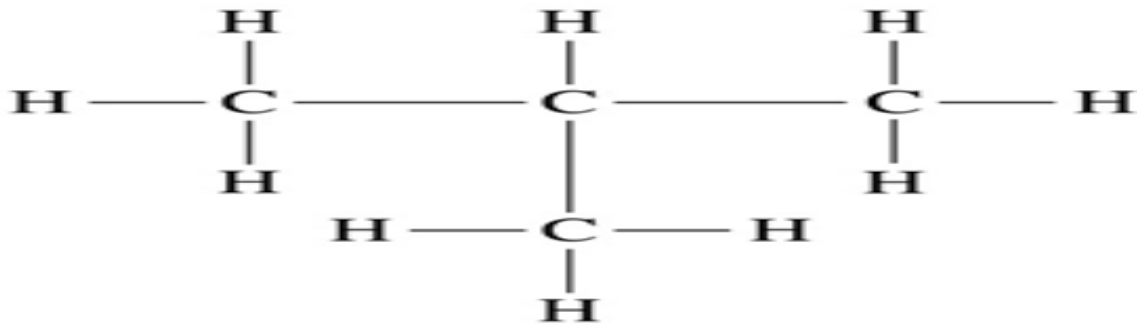
Arthur Cayley
(1821-1895)

Trees are used as models in computer science, chemistry, geology, botany, psychology, and many other areas.

Trees were introduced by the mathematician Cayley in 1857 in his work counting the number of isomers of saturated hydrocarbons. The two isomers of butane are:



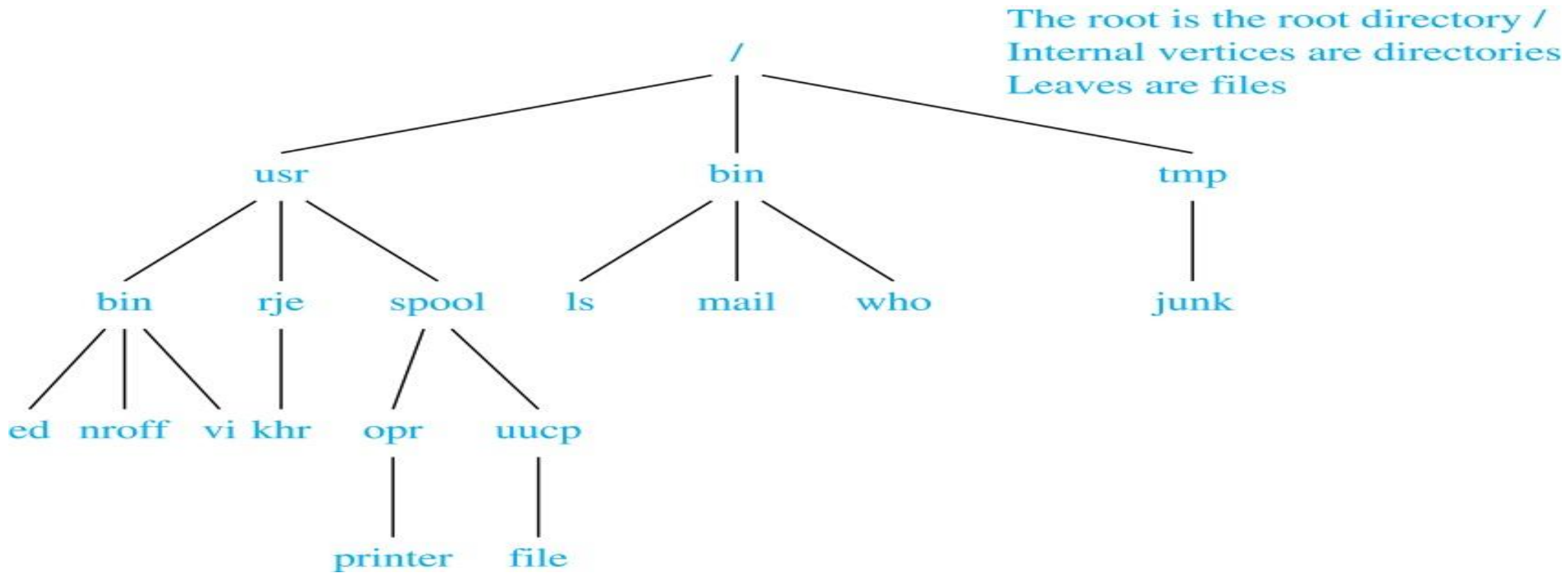
Butane



Isobutane

Trees as Models

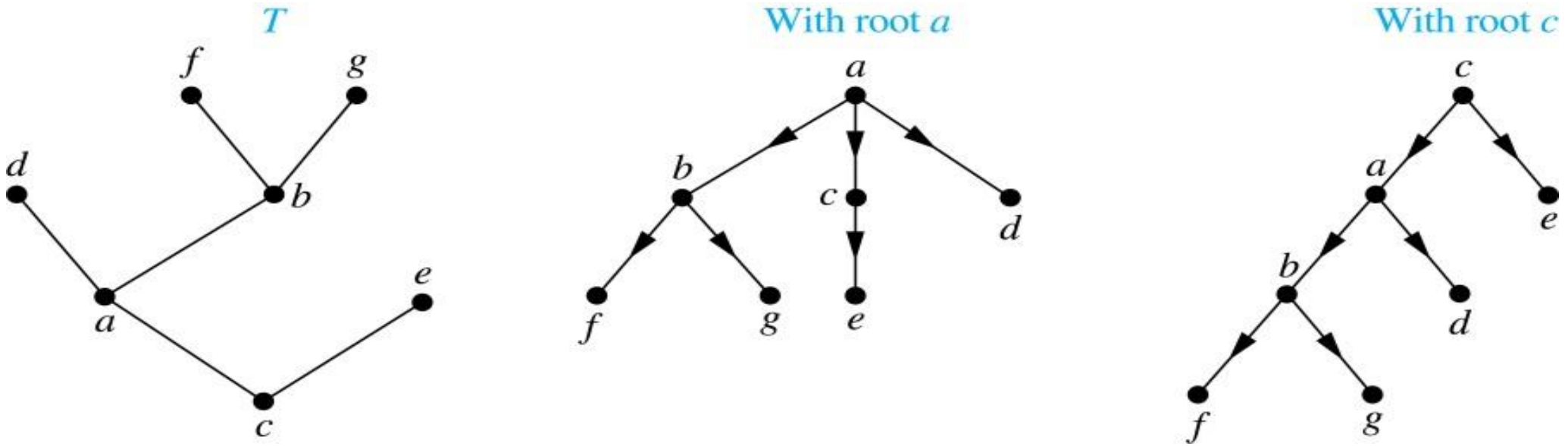
The organization of a computer file system into directories, subdirectories, and files is naturally represented as a tree.



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*.

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.

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

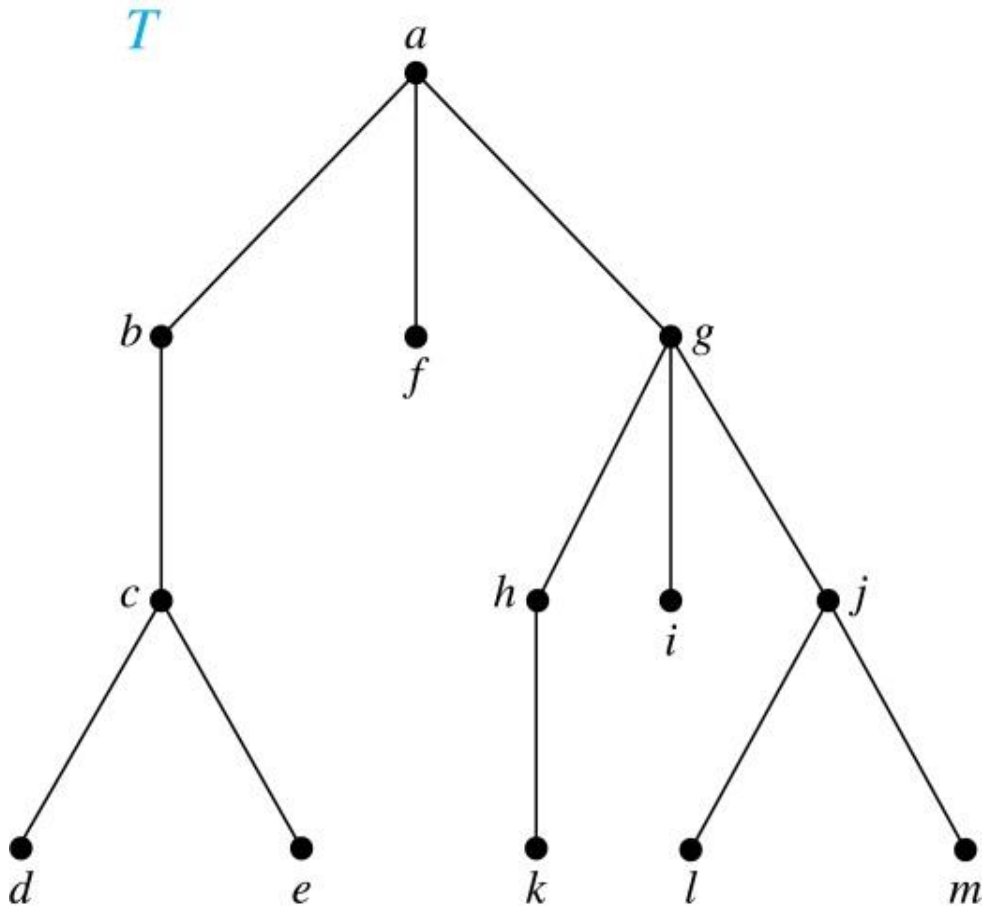
If *a* is a vertex in a tree, the *subtree* with *a* as its root is the subgraph of the tree consisting of *a* and its descendants and all edges incident to these descendants.

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.

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*.



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*.

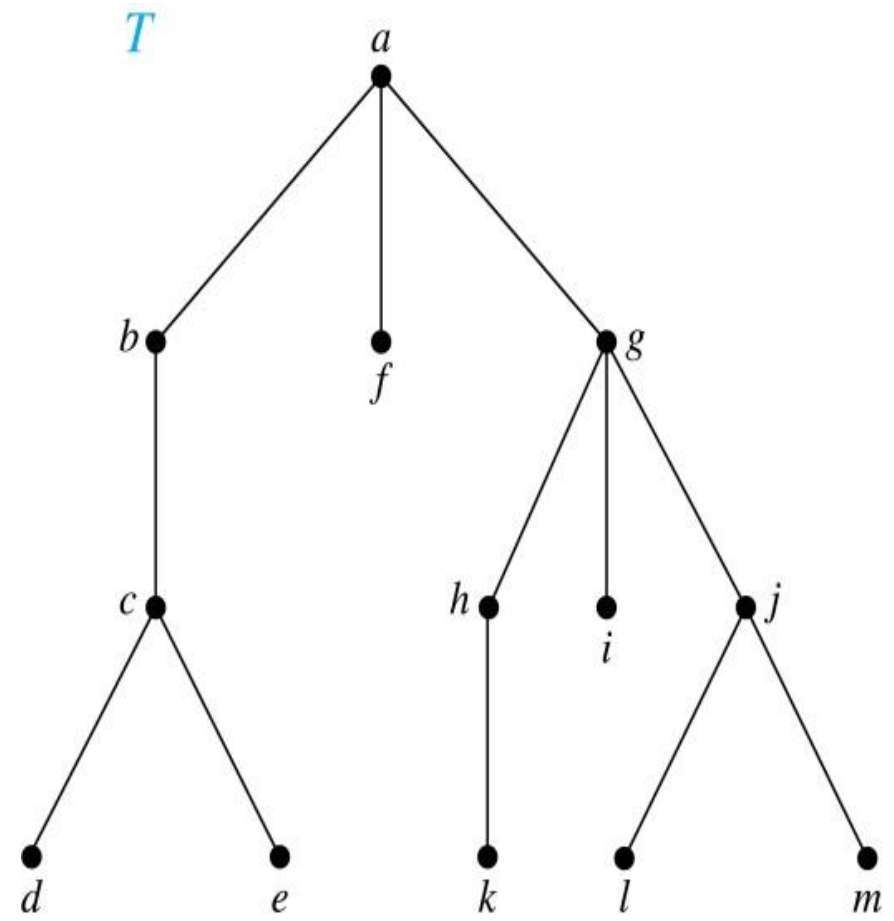
Terminology for Rooted Trees

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

(i) Find all internal vertices and all leaves.

Solution:

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

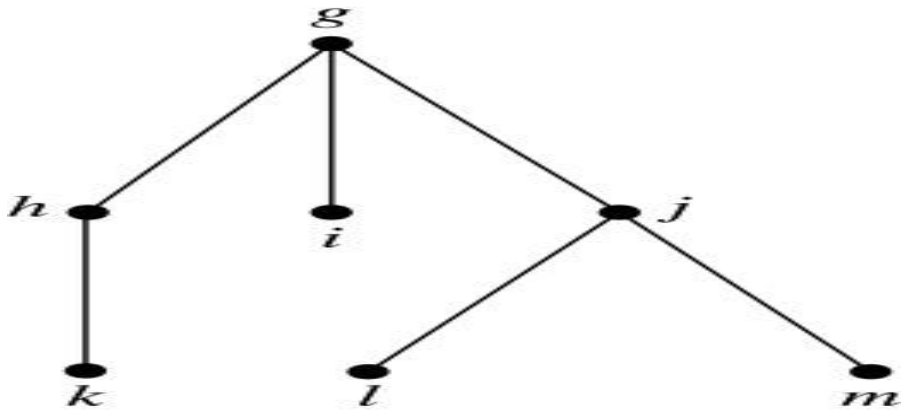


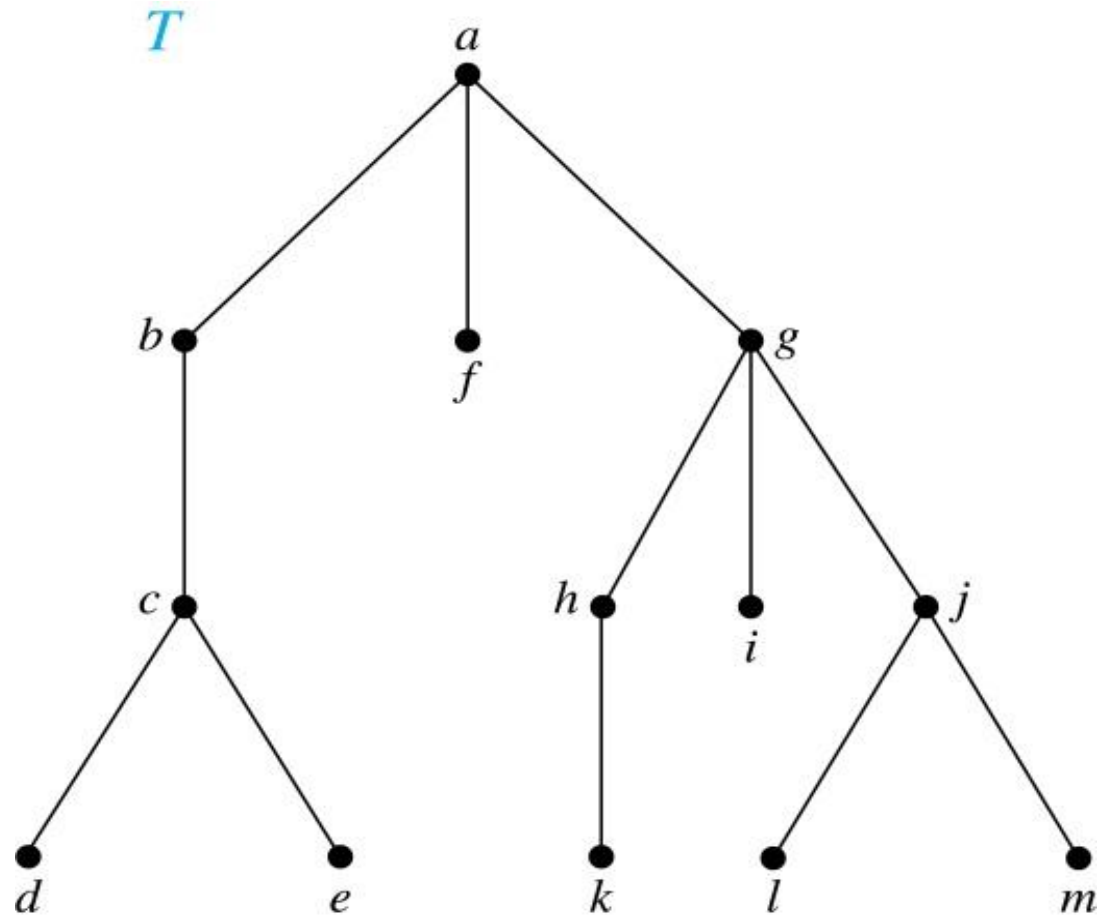
Terminology for Rooted Trees

(i) What is the subtree rooted at *G*?

Solution:

(i) We display the subtree rooted at *g*.

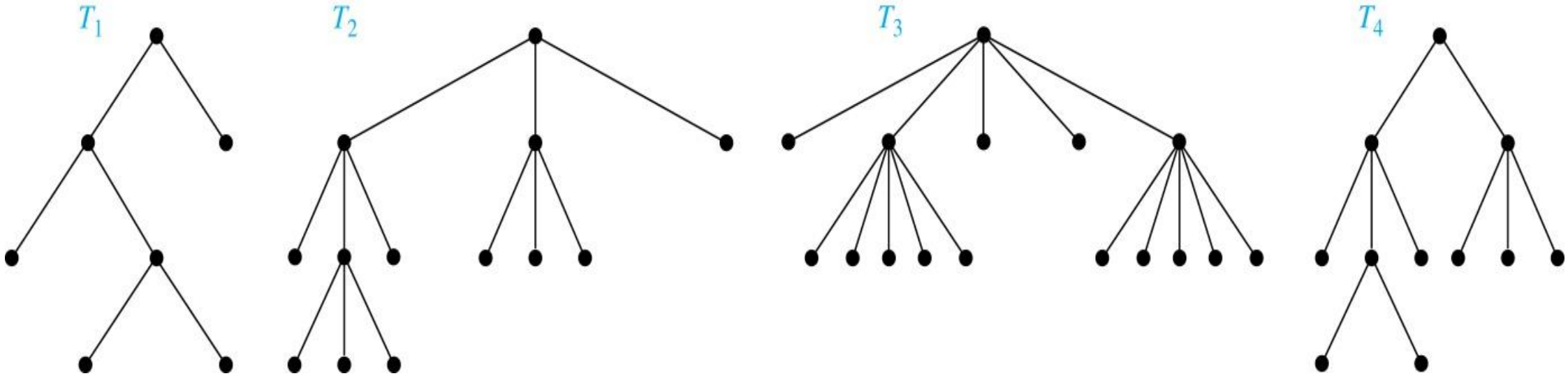




m-ary Rooted Trees

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.

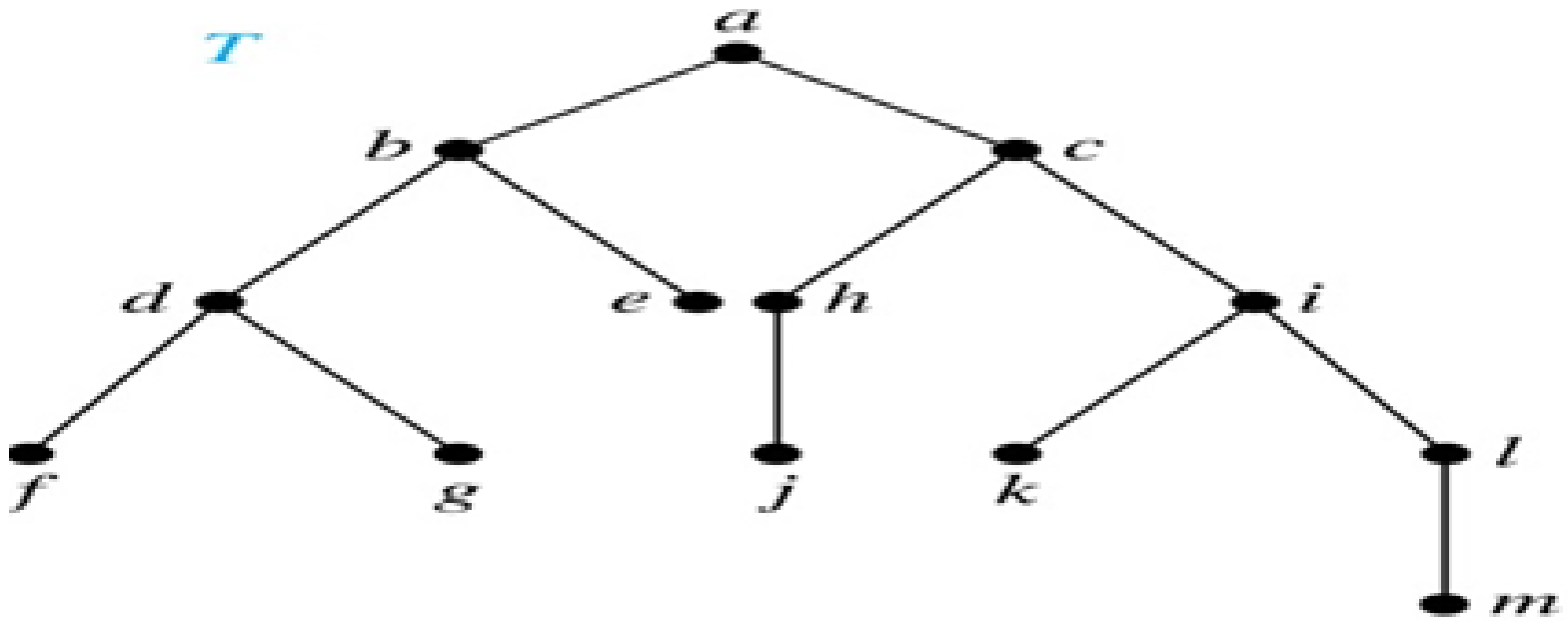
Example: Are the following rooted trees full *m*-ary trees for some positive integer *m*?



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.



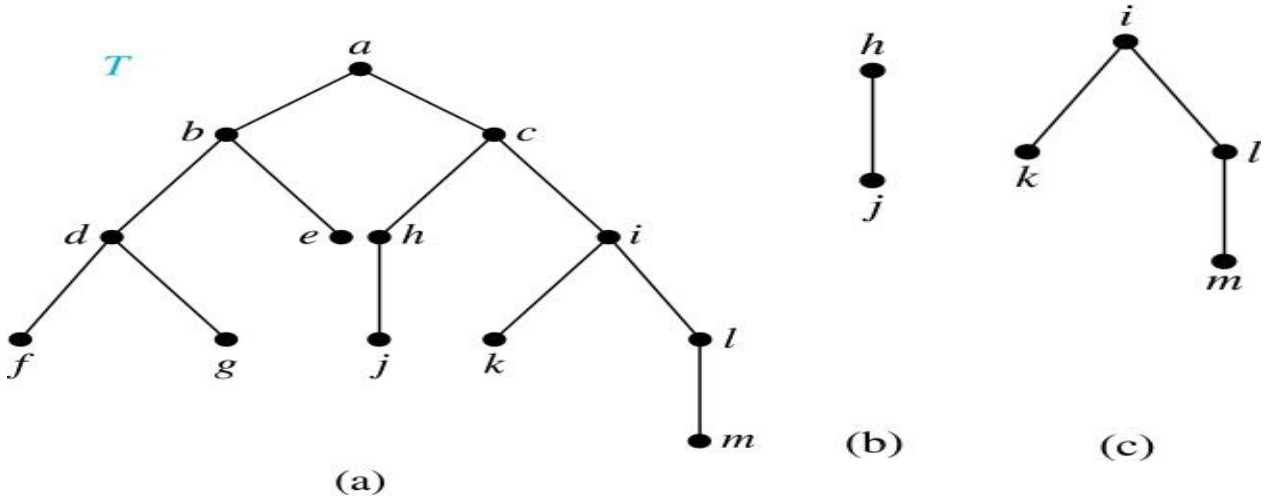
Binary Trees

Definition: A *binary tree* is an ordered rooted where 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.

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

- A tree with n vertices has $n - 1$ edges
- A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices
- 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 and height of trees

When working with trees, we often want to have rooted trees where the subtrees at each vertex contain paths of approximately the same length.

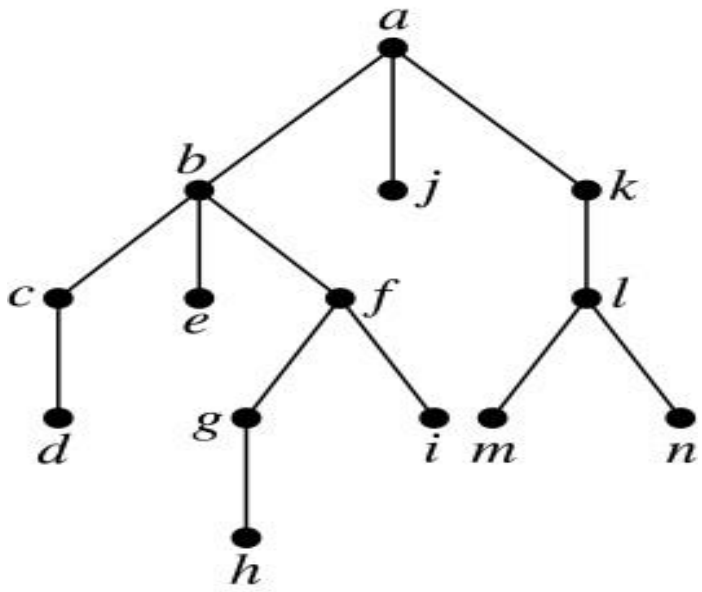
To make this idea precise we need some definitions:

- 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

Example:

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

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

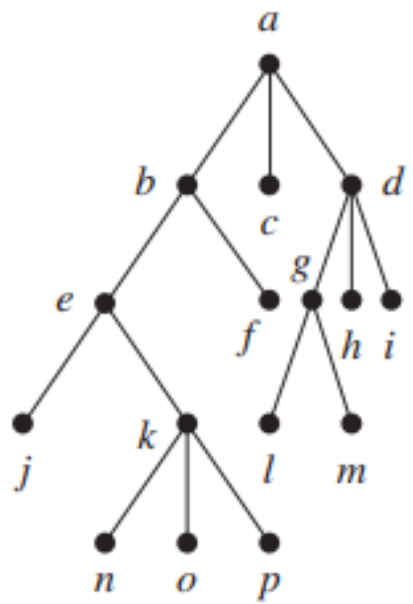


Tree Traversal

Preorder traversal: Visit root, visit subtrees left to righ

Inorder traversal: Visit leftmost subtree, visit root, visit other subtrees left to right

Postorder traversal: Visit subtrees left to right; visit root



Preorder

a b e j k n o p f c d g l m h i

Inorder

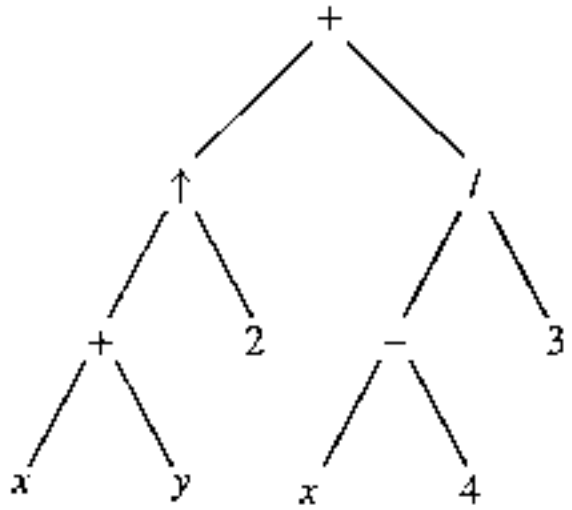
j e n k o p b f a c l g m d h i

Postorder

j n o p k e f b c l m g h i d a

Tree Traversal: Example

A binary tree representing ((*x* + *y*) ↑ 2) + ((*x* – 4)/3).

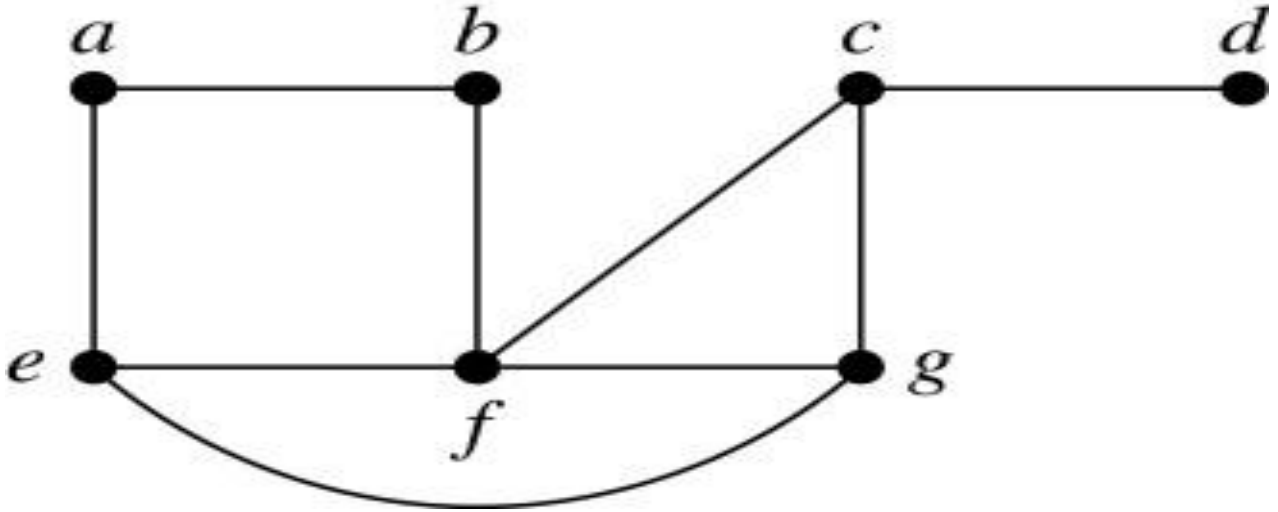


Prefix expression (preorder): + ↑ + *x y* 2 / – *x* 4 3
Postfix expression (postorder): *x y* + 2 ↑ *x* 4 – 3 / +
Infix expression (inorder): **?**

Spanning Trees

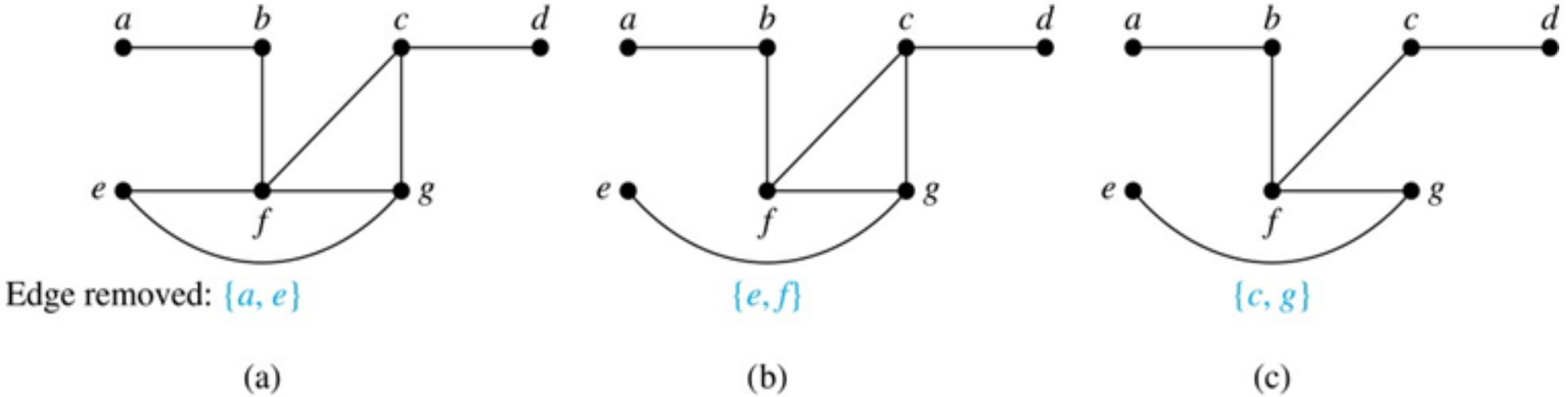
Definition: Let *G* be a simple graph. A spanning tree of *G* is a subgraph of *G* that is a tree containing every vertex of *G*.

Example: Find the spanning tree of the simple graph:



Spanning Trees

Solution: The graph is connected, but is not a tree because it contains simple circuits. Remove the edge $\{a, e\}$. Now one simple circuit is gone, but the remaining subgraph still has a simple circuit. Remove the edge $\{e, f\}$ and then the edge $\{c, g\}$ to produce a simple graph with no simple circuits. It is a spanning tree, because it contains every vertex of the original graph.



Spanning Trees

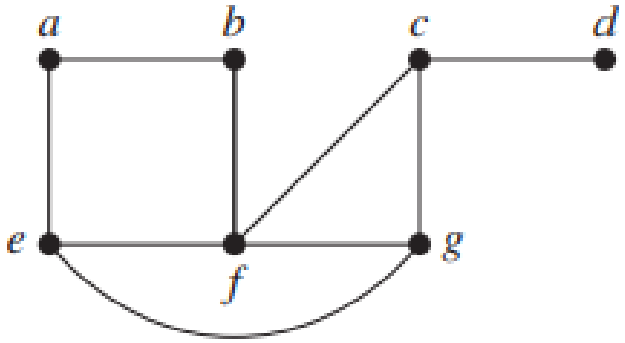
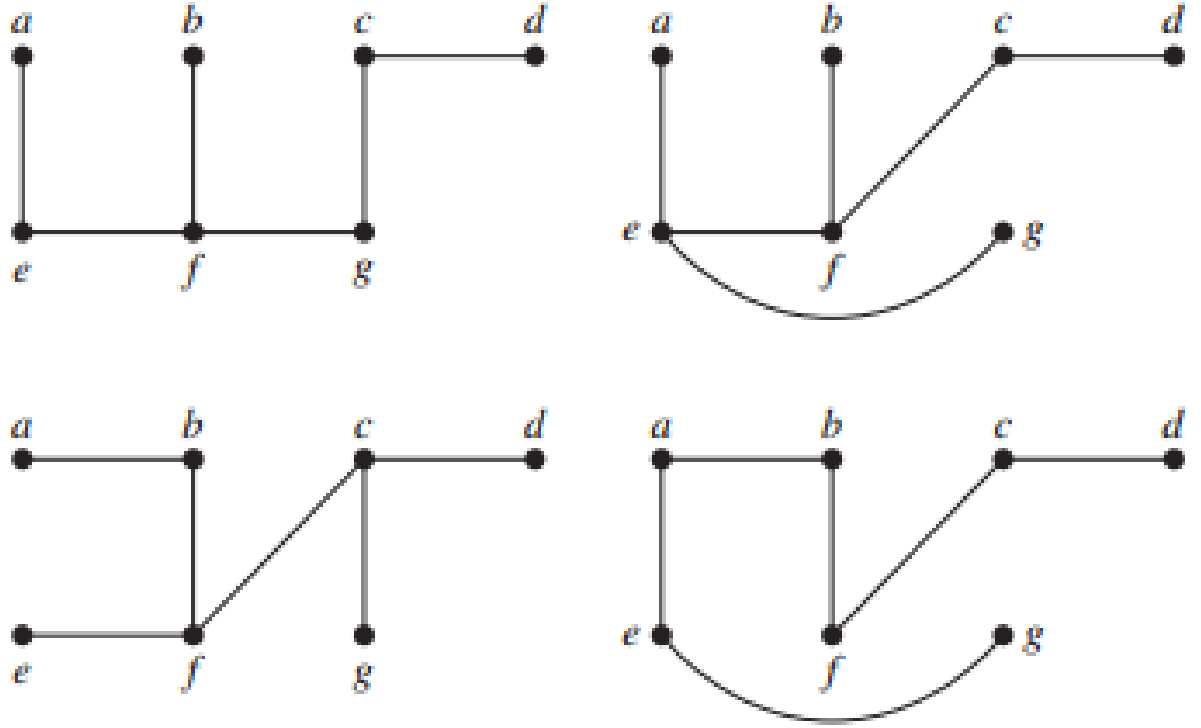
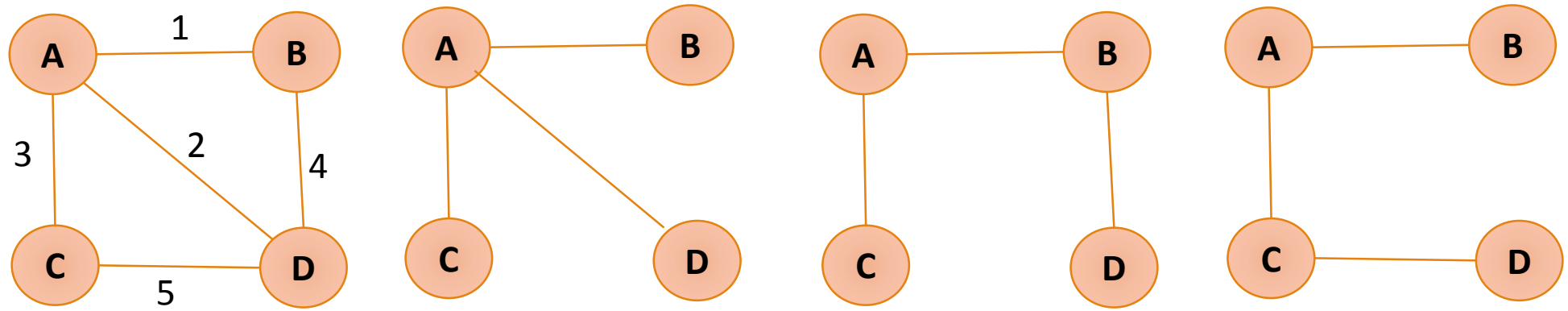


FIGURE 2 **The simple graph *G*.**



Minimum spanning tree

A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.



Example: A company plans to build a communications network connecting its five computer centers. Any pair of these centers can be linked with a leased telephone line. Which links should be made to ensure that there is a path between any two computer centers so that the total cost of the network is minimized?

Minimum spanning tree

Solution: We can model this problem using the weighted graph shown in Figure 1, where vertices represent computer centers, edges represent possible leased lines, and the weights on edges are the monthly lease rates of the lines represented by the edges. We can solve this problem by finding a spanning tree so that the sum of the weights of the edges of the tree is minimized. Such a spanning tree is called a **minimum spanning tree**.

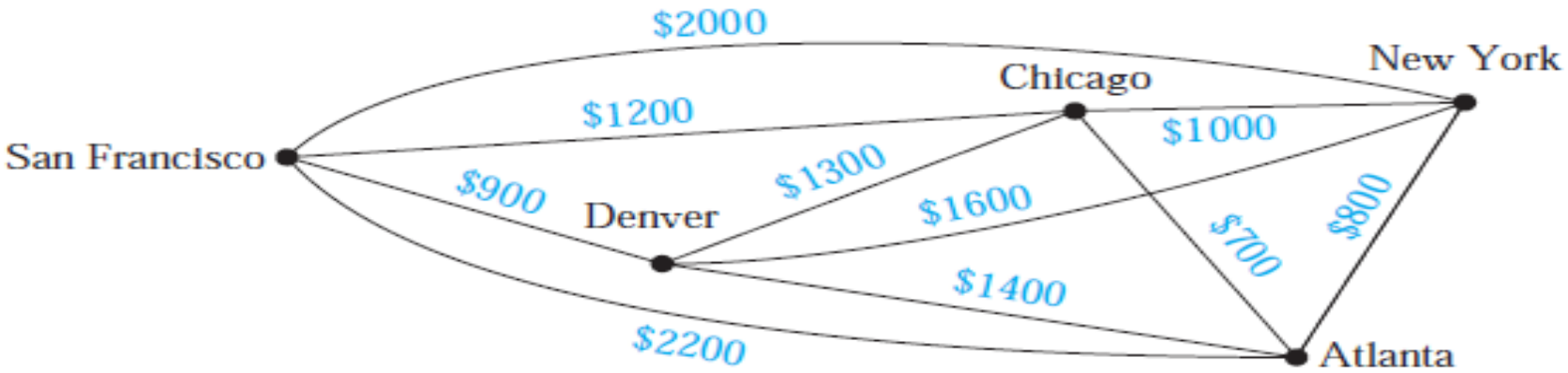


FIGURE 1 A Weighted Graph Showing Monthly Lease Costs for Lines in a Computer Network.

PRIM'S ALGORITHM

ALGORITHM 1 Prim's Algorithm.

procedure *Prim*(G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge
for $i := 1$ **to** $n - 2$
    e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
    $T := T$ with e added
return T {T is a minimum spanning tree of G}

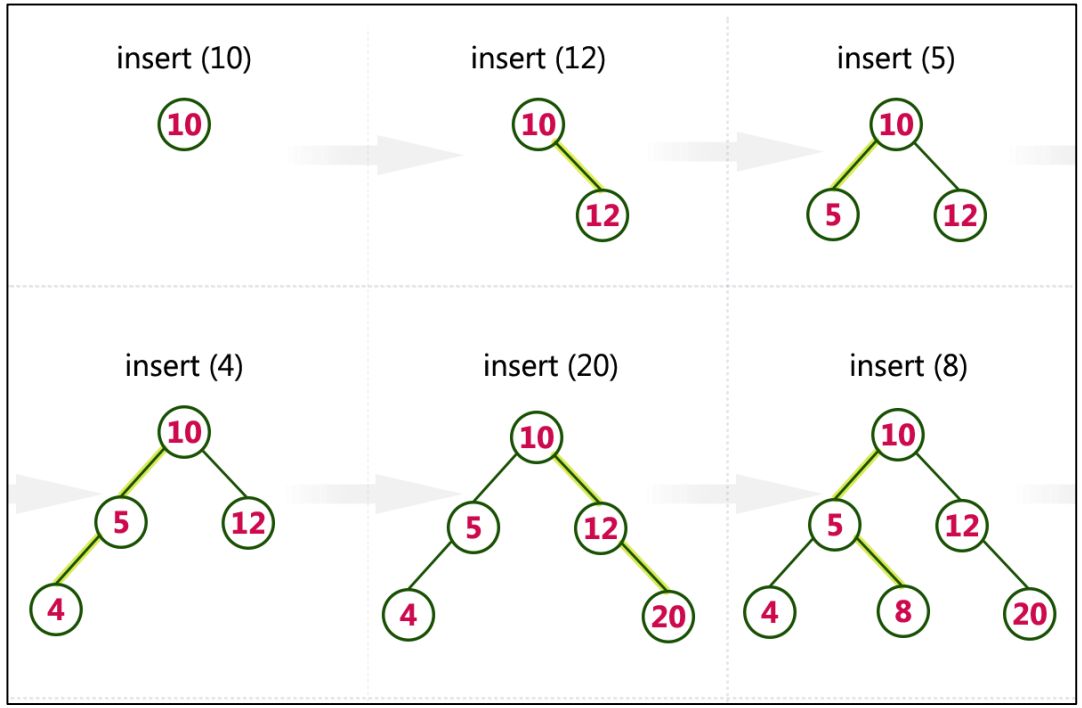
KRUSKAL'S ALGORITHM

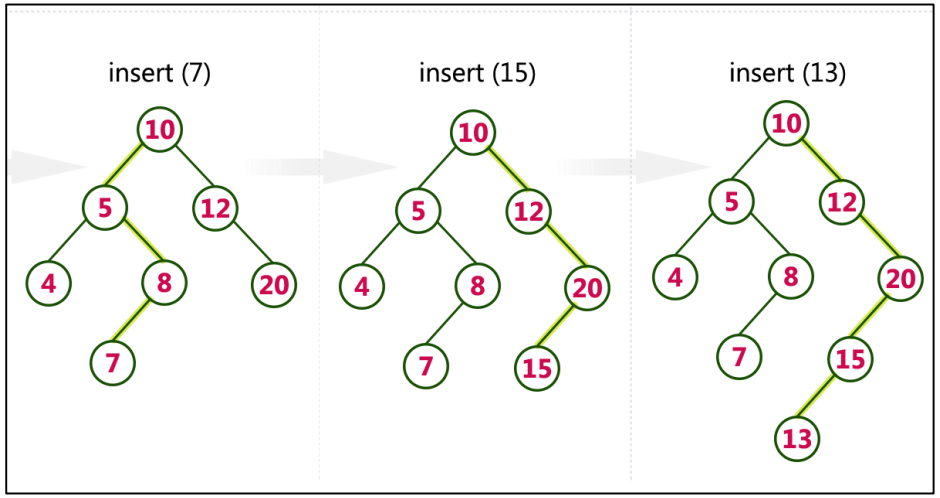
ALGORITHM 2 Kruskal's Algorithm.

procedure *Kruskal*(G: weighted connected undirected graph with n vertices)
T := empty graph
for $i := 1$ **to** $n - 1$
    e := any edge in G with smallest weight that does not form a simple circuit when added to T
    $T := T$ with e added
return T {T is a minimum spanning tree of G}

Application: **Binary Search Tree**

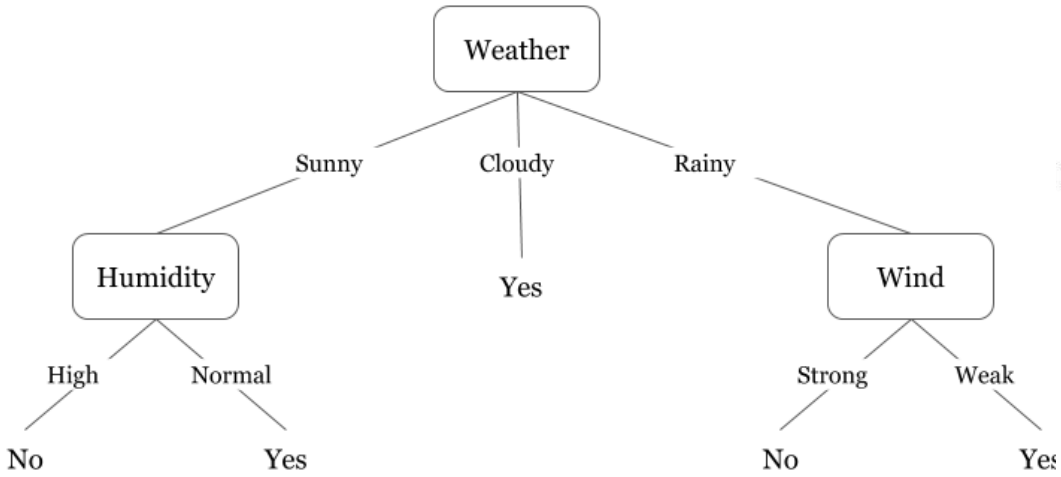
A binary tree in which the vertices are labeled with items so that a label of a vertex is greater than the labels of all vertices in the left subtree of this vertex and is less than the labels of all vertices in the right subtree of this vertex.

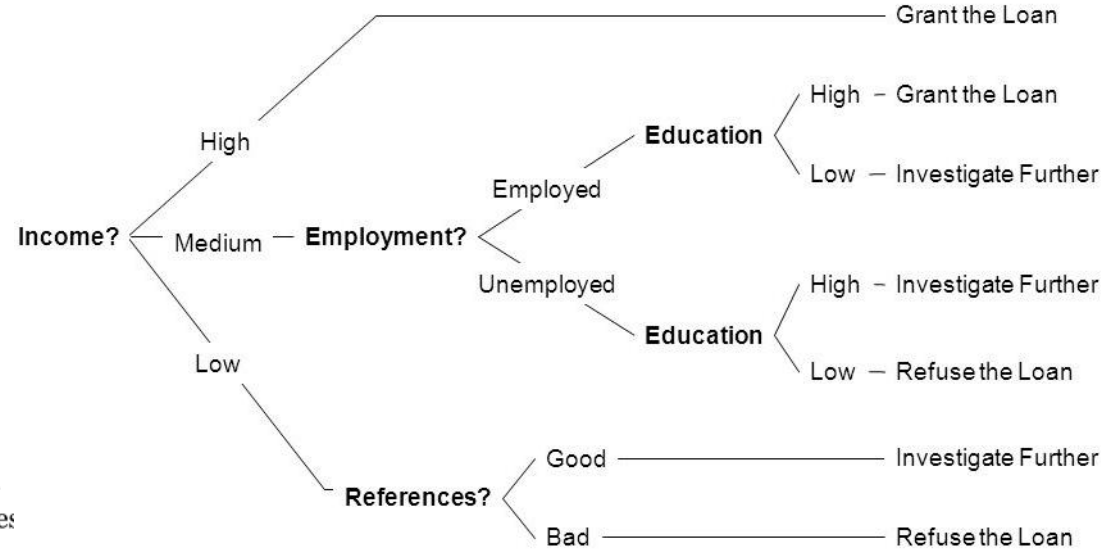




Application: **Decision Trees**

A rooted tree where each vertex represents a possible outcome of a decision and the leaves represent the possible solutions of a problem.





Application: **Gaming**