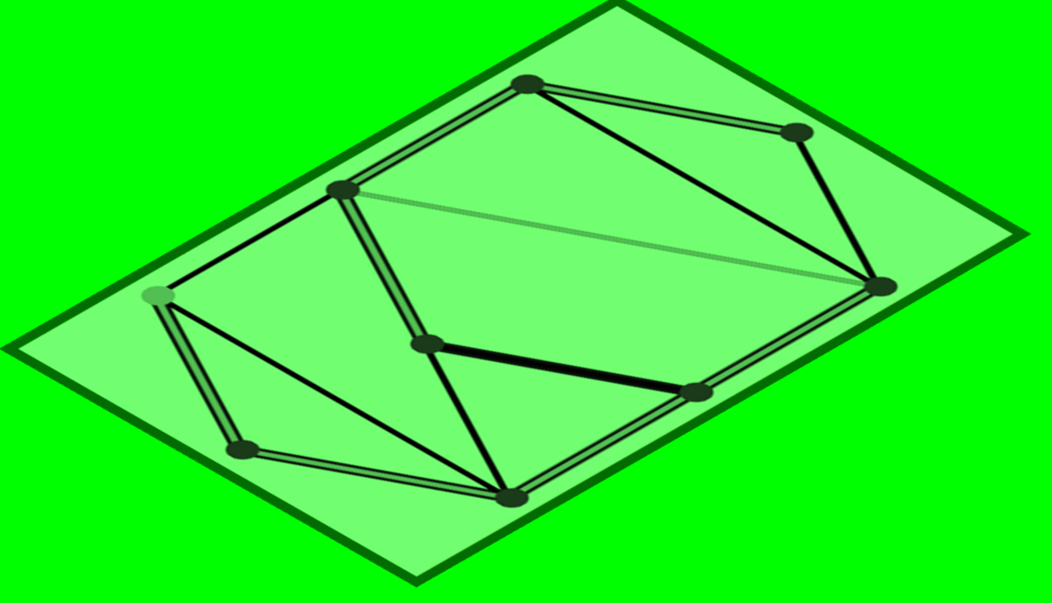
Arab Open University
Faculty of Computer Studies

MT131 - Discrete Mathematics

10. Graphs



Graphs and Graph Models

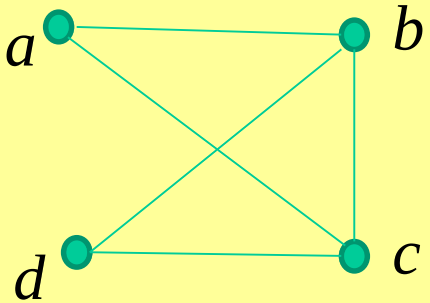
A **graph** $G = (V, E)$ consists of V, a nonempty set of vertices (or nodes) and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints.

Remark: The set of vertices V of a graph G may be infinite. A graph with an infinite vertex set is called an infinite graph, and in comparison, a graph with a finite vertex set is called a finite graph. In this course we will usually consider only finite graphs.

Summary

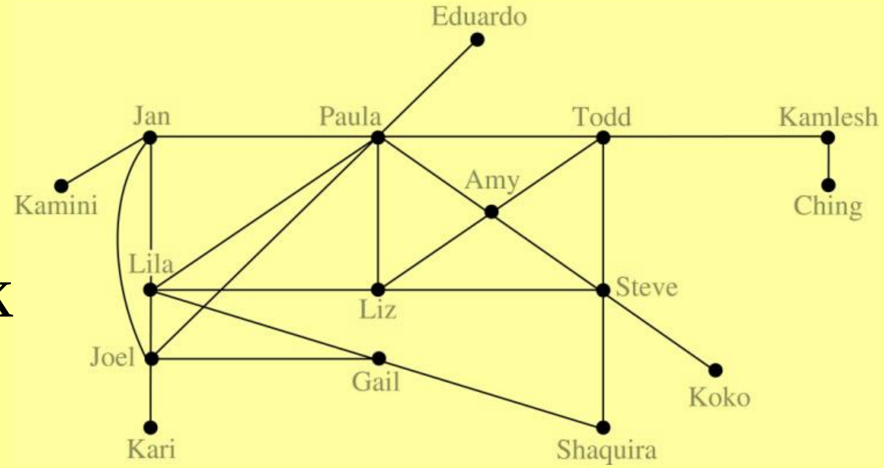
- It is a pair $G = (V, E)$, where
  - ○ $V = V(G)$ = Set of vertices (or nodes)
  - ○ $E = E(G)$ = Set of edges

- Example: This is a graph with four vertices and five edges.
  - ○ $V = \{a, b, c, d\}$
  - ○ $E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}\}$



Graphs and Graph Models

– Computer network
– Social networks
– Communications networks
– Information networks
– Software design
– Transportation network
– Biological networks



A friendship graph where two people are connected if they are Facebook friends

Example

This computer network can be modeled using a graph in which the vertices of the graph represent the data centers and the edges represent communication links.

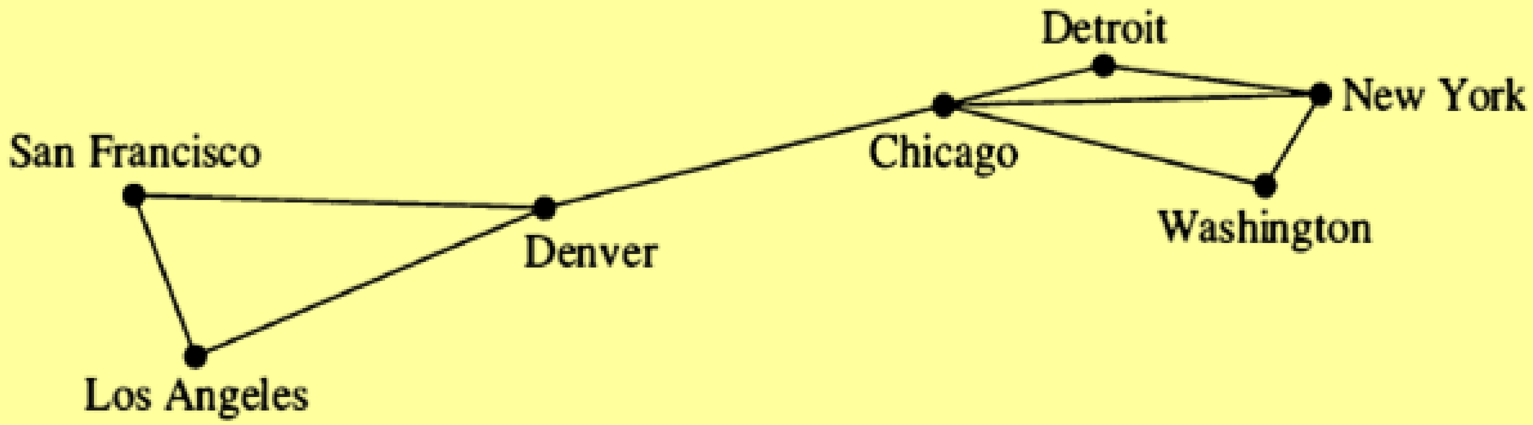


FIGURE 1 A Computer Network.

Simple Graphs

A graph in which each edge connects two different vertices and where no more than one edge connect the same pair of vertices is called a **simple graph**.

e.g. The graph in figure 1 is an example of a simple graph.

Multigraphs

A computer network may contain multiple links between data centers, as shown in Figure 2. To model such networks we need graphs that have more than one edge connecting the same pair of vertices. Graphs that may have multiple edges connecting the same vertices are called **multigraphs**.

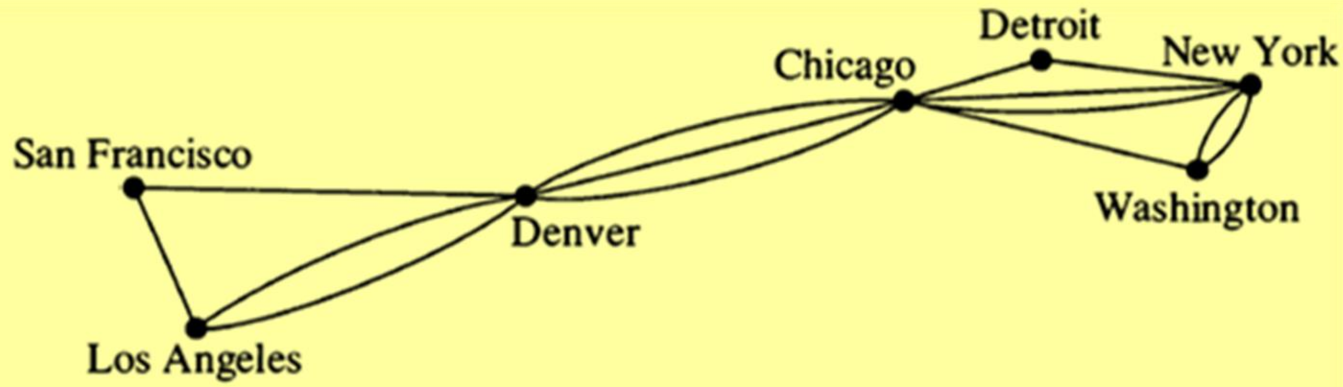


FIGURE 2 A Computer Network with Multiple Links between Data Centers.

Pseudographs

Sometimes a communication link connects a data center with itself. To model this network we need to include edges that connect a vertex to itself. Such edges are called **loops**. Graphs that may include loops, and possibly multiple edges are sometimes called **pseudographs**.

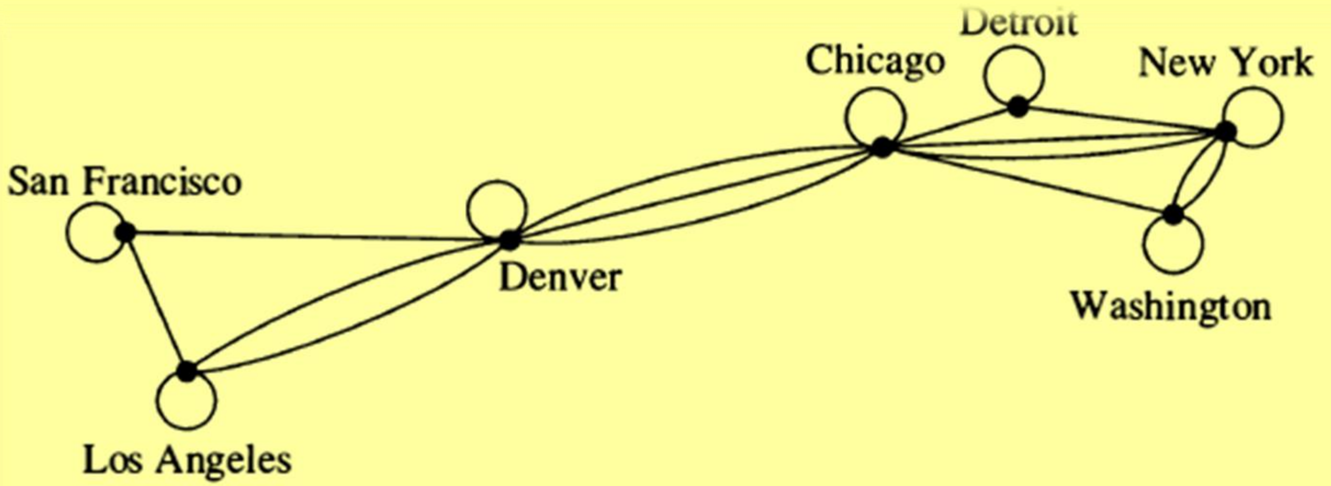


FIGURE 3 A Computer Network with Diagnostic Links.

Directed Graphs

Definition 2: A **directed graph** (or digraph) (V, E) consists of a nonempty set of vertices V and a set of directed edges (or arcs) E. Each directed edge is associated with an ordered pair of vertices.

The directed edge associated with the ordered pair (u, v) is said to start at u and end at v. When a directed graph has no loops and has no multiple directed edges, it is called a **simple directed graph**.

Simple Directed Graph

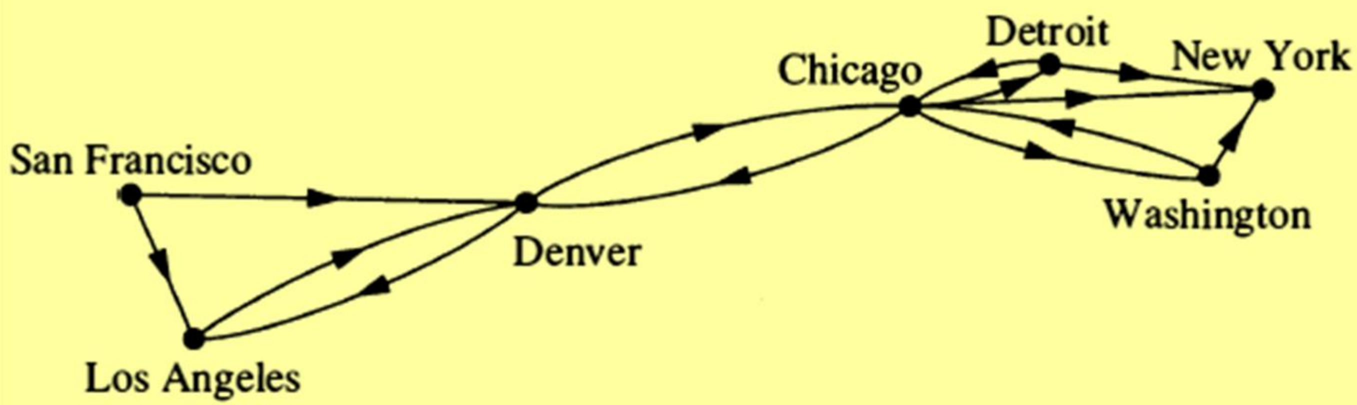


FIGURE 4 A Communications Network with One-Way Communications Links.

Directed Multigraph

Directed graphs that may have multiple directed edges from a vertex to a second (possibly the same) vertex are to used model such networks. We call such graphs **directed multigraphs**.

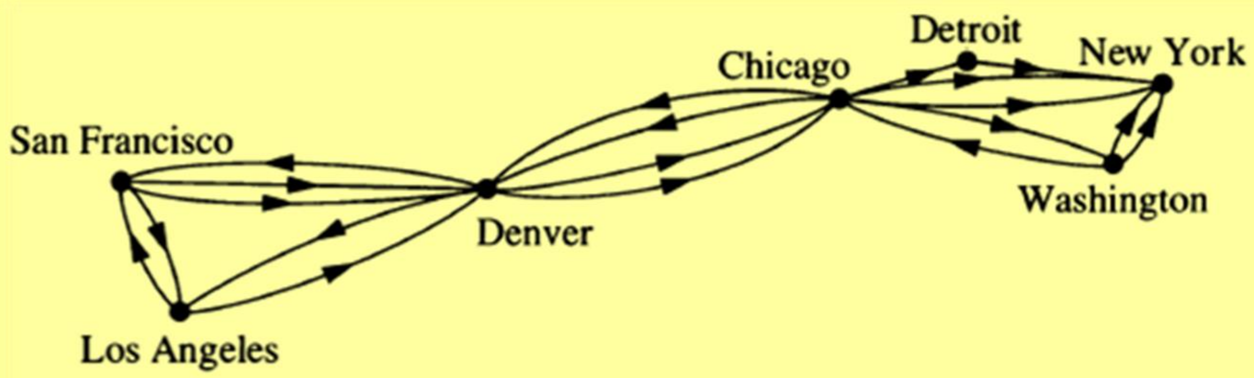


FIGURE 5 A Computer Network with Multiple One-Way Links.

Graph Terminology: Summary

To understand the structure of a graph and to build a graph model, we ask these questions:

- Are the edges of the graph undirected or directed (or both)?
- If the edges are undirected, are multiple edges present that connect the same pair of vertices? If the edges are directed, are multiple directed edges present?
- Are loops present?

TABLE 1 Graph Terminology.

Type	*Edges*	*Multiple Edges Allowed?*	*Loops Allowed?*
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

Graph Terminology

- **Adjacent**, **connects**, **endpoints**, **degree**, **initial**, **terminal**, **in-degree**, **out-degree**, **complete**, **complete bipartite**, **cycles**, **wheels**, ***n*-cubes**.

Undirected Graph: Adjacency

Let G be an undirected graph with edge set E. Let $e \in E$ where $e = \{u, v\}$. We say that:

- The vertices u and v are **adjacent** / neighbors / connected.
- The edge e is **incident with** vertices u and v.
- The edge e **connects** u and v.
- The vertices u and v are **endpoints** of the edge e.

Undirected Graph: Degree of a Vertex

- Let G be an undirected graph, $v \in V$ a vertex.
- The **degree** of v, $\deg(v)$, is its number of incident edges. (Except that any self-loops are counted twice.)
- A vertex with degree 0 is **isolated.**
- A vertex of degree 1 is **pendant.**

Handshaking Theorem

- Let $G = (V, E)$ be an undirected (simple, multi- or pseudo-) graph with e edges. Then:

$$\sum_{v \in V} \deg(v) = 2|E|$$

Example

deg(a) = 6

deg(b) = 4

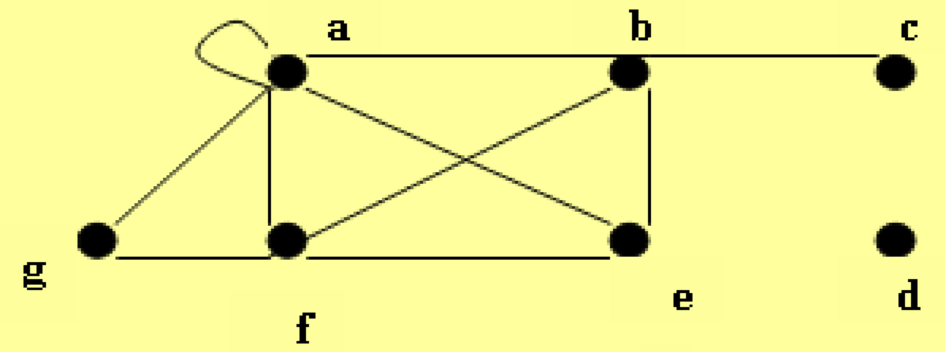
deg(c) = 1 pendant

deg(d) = 0 isolated

deg(e) = 3

deg(f) = 4

deg(g) = 2



$$\sum \deg(v) = 20 = 2\,(10) = 2 \sum \text{edges}$$

Handshaking Theorem

Example 1: How many edges are there in a graph with ten vertices each of degree 6?

Solution: Sum of the degrees of the vertices $6 \times 10 = 60$. Therefore: $60 = 2|E|$, $|E| = 30$.

Example 2: If a graph has 5 vertices, can each vertex have degree 3?

Solution: This is not possible by the handshaking theorem, because the sum of the degrees of the vertices $3 \times 5 = 15$ is odd.

Directed Graph: Adjacency

- Let G be a directed (possibly multi-) graph, and let e be an edge of G that is (u ,v). Then we say:

$$u \longrightarrow v$$

  - u is **adjacent to** v, v is **adjacent from** u
  - e **comes from** u, e **goes to** v.
  - e **connects** u **to** v, e **goes from** u **to** v
  - The **initial vertex** of e is u
  - The **terminal vertex** of e is v

Directed Graph: Degree of a vertex

- Let G be a directed graph and v a vertex of G.
- The **in-degree** of v, $\deg^{-}(v)$, is the number of edges going to v (v is terminal).
- The **out-degree** of v, $\deg^{+}(v)$, is the number of edges coming from v (v is initial).
- The **degree** of v, $\deg(v) = \deg^{-}(v) + \deg^{+}(v)$, is the sum of v's in-degree and out-degree.
- A loop at a vertex contributes 1 to both in-degree and out-degree of this vertex.

Directed Handshaking Theorem

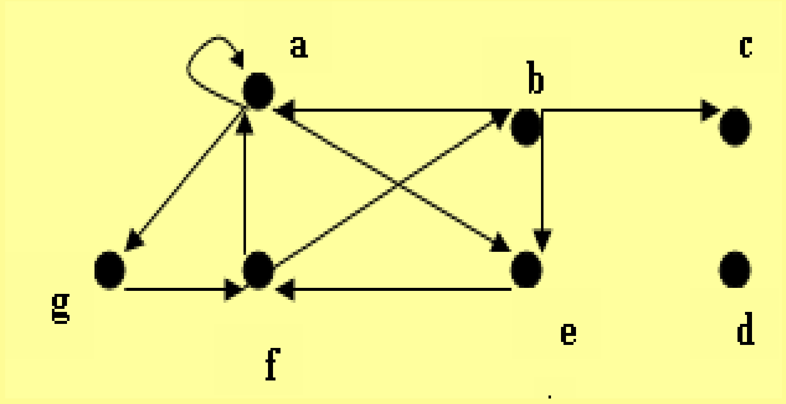
- Let G be a directed (possibly multi-) graph with vertex set V and edge set E. Then:

$$\sum_{v \in V} \deg^{-}(v) = \sum_{v \in V} \deg^{+}(v) = \frac{1}{2} \sum_{v \in V} \deg(v) = |E|$$

- Note that the degree of a node is unchanged by whether we consider its edges to be directed or undirected.

Example

$\deg^+(a) = 3$ $\deg^-(a) = 3$
$\deg^+(b) = 3$ $\deg^-(b) = 1$
$\deg^+(c) = 0$ $\deg^-(c) = 1$
$\deg^+(d) = 0$ $\deg^-(d) = 0$
$\deg^+(e) = 1$ $\deg^-(e) = 2$
$\deg^+(f) = 2$ $\deg^-(f) = 2$
$\deg^+(g) = 1$ $\deg^-(g) = 1$



$$\sum \deg^+(v) = \sum \deg^-(v) = 1/2 \sum \deg(v) = \sum \text{edges} = 10$$

Special Graph Structures

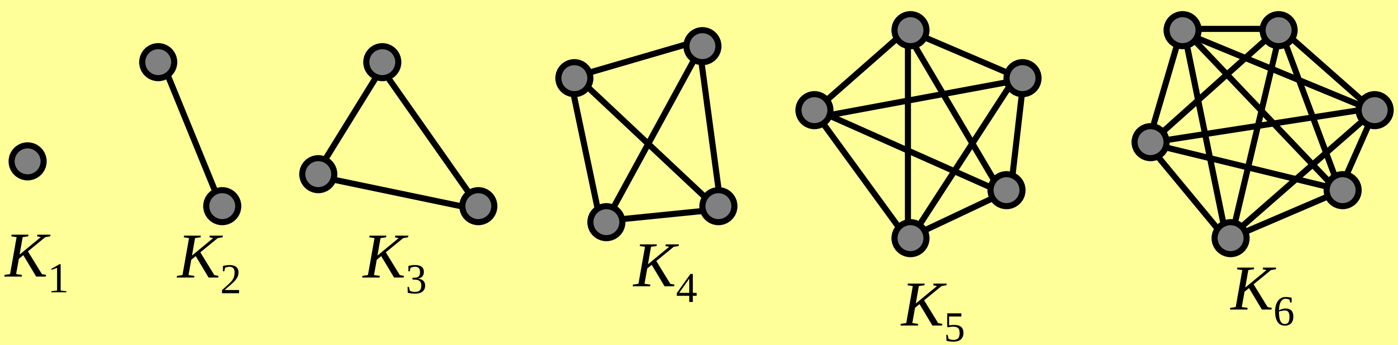
Special cases of undirected graph structures:

- Complete graphs: K_n
- Complete bipartite graphs: $K_{m,n}$

Complete Graphs

- For any $n \in \mathbf{N}$, a **complete graph** on n vertices, K_n, is a simple graph with n nodes in which every node is adjacent to every other node:

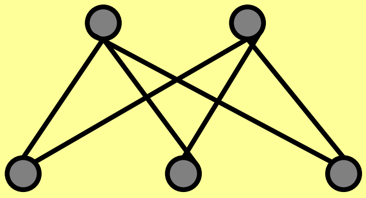
$$\forall u, v \in V: u \neq v \leftrightarrow \{u, v\} \in E.$$



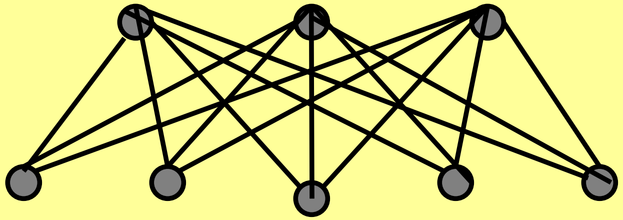
Note that K_n has n vertices and $\dfrac{n(n-1)}{2}$ edges.

Complete Bipartite Graphs

- The **complete bipartite** graph $K_{m,n}$ is the graph that has its vertex set partitioned into two subsets of m and n vertices such that there is an edge between vertices if and only if one vertex in the first set and the other vertex in the second set.



$K_{2,3}$



$K_{3,5}$

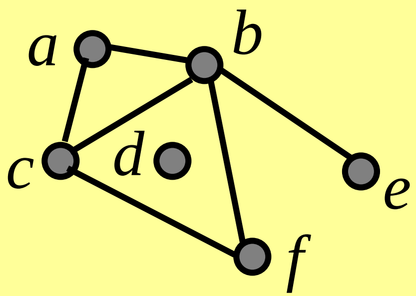
Note that $K_{m,n}$ has $m+n$ vertices and mn edges.

Graph Representations

- Graph representations:
  - Adjacency Lists.
  - Adjacency Matrices.
  - Incidence Matrices.

Undirected: Adjacency Lists

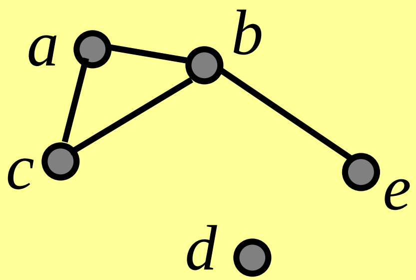
- A table with 1 row per vertex, listing its adjacent vertices.



Vertex	Adjacent Vertices
a	b, c
b	a, c, e, f
c	a, b, f
d	-
e	b
f	c, b

Undirected: Adjacency Matrices

- Matrix $A = [a_{ij}]$, where a_{ij} is 1 if $\{v_i, v_j\}$ is an edge of G, 0 otherwise.

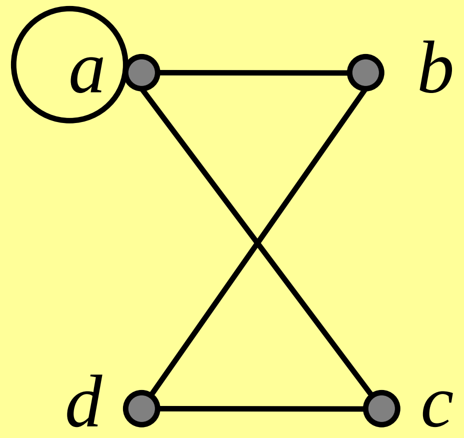


$$\begin{array}{c c} & \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Example 1

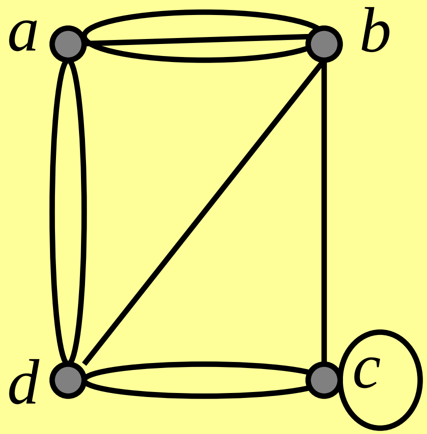
Draw a graph using the following adjacency matrix A with respect to the vertices: a, b, c, d :

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$



Example 2

Find an adjacency matrix that represents the graph:



$$\begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix} \end{array}$$

Example 3

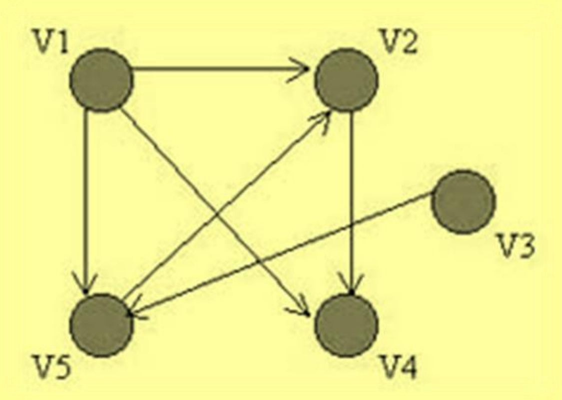
Let A be the adjacency matrix of a graph G with vertices a, b, c, d . Find the degrees of the vertices of G.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Answer: The degrees of the vertices a, b, c and d of G are 3, 2, 2 and 1, respectively.

Directed Adjacency Lists

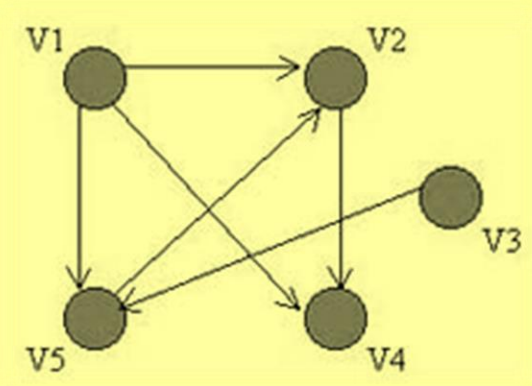
- 1 row per node, listing the terminal nodes of each edge incident from that node.



Initial vertex	Terminal vertices
V1	V2, V4, V5
V2	V4
V3	V5
V4	
V5	V2

Directed Adjacency Matrices

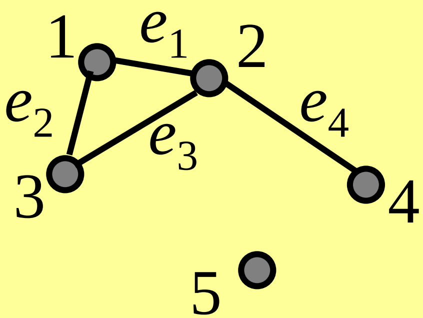
- Matrix $A = [a_{ij}]$, where a_{ij} is 1 if $\{v_i, v_j\}$ is an edge of G, 0 otherwise.



$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

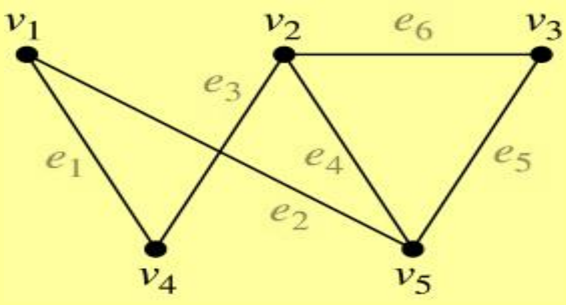
Incidence Matrices

- It is a matrix $A = [a_{ij}]$, where a_{ij} is 1 if the edge e_j is incident with the vertex v_i , 0 otherwise.

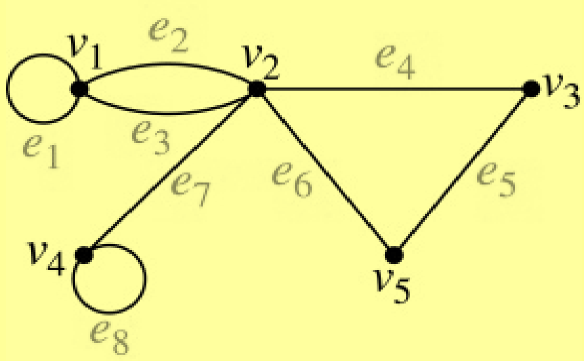


$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccc} e_1 & e_2 & e_3 & e_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Incidence Matrices



$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Example

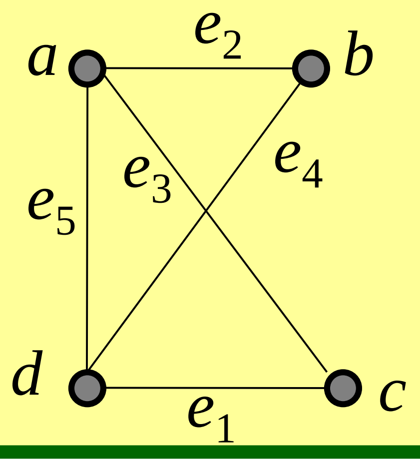
Let A be the incident matrix of a graph G. Draw G.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Let $G = \{V, E\}$,

where V= $\{a, b, c, d\}$ and

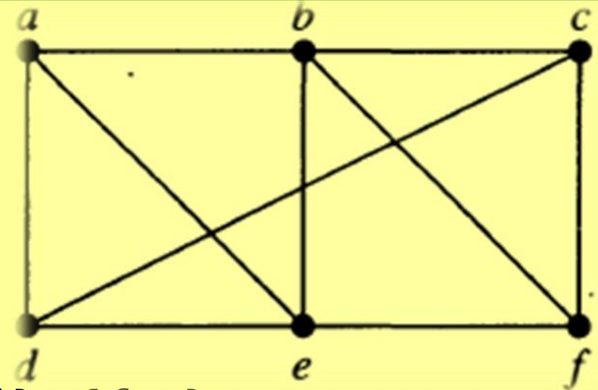
$E = \{e_1, e_2, e_3, e_4, e_5\}$.



Connectivity

- In an undirected graph, a **path of length *n*** from u to v is a sequence of adjacent edges going from vertex u to vertex v.
- A path is a **circuit** if $u = v$.
- A path **traverses** the vertices along it.
- A path is **simple** if it contains no edge more than once.

Connectivity

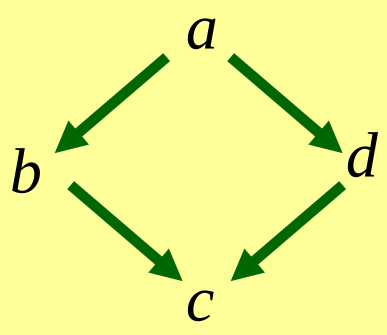


- **Simple path:** *a, d, c, f, e*
  The edges are {*a*, *d* }, {*d*, *c*}, {*c*, *f* }, {*f*, e}
  Length = 4
- **Circuit:** *b, c, f, e, b*
  The edges are {*b*, *c*},{*c*, *f* }, {*f*, *e*},{*e*, *b*}
  Length = 4
- **Not a path:** *d, e, c, a* because {*e*, *c*} is not an edge
- **Not a simple path:** *a, b, e, d, a, b* because {*a*, *b*} appears twice, Length = 5

Paths in Directed Graphs

- Same as in undirected graphs, but the path must go in the direction of the arrows.

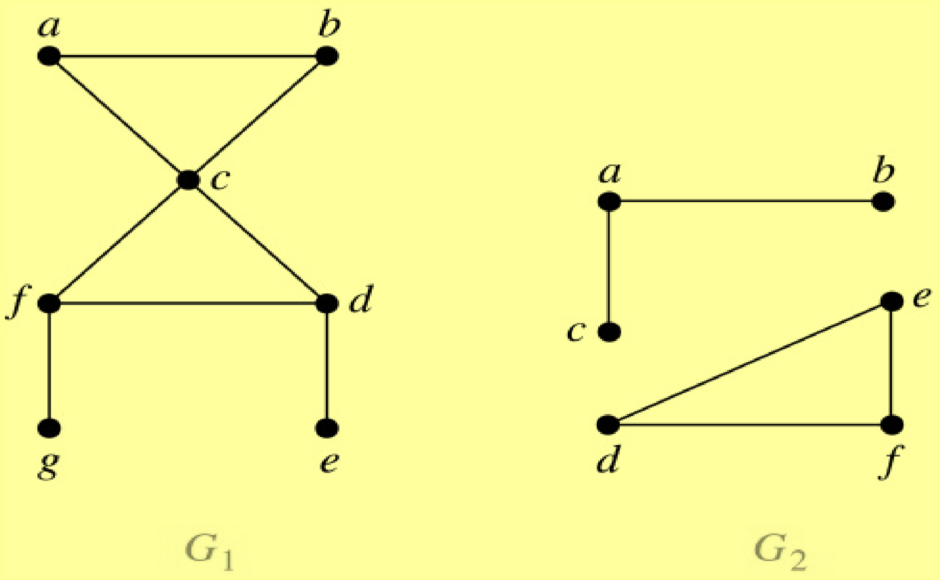
e.g. Simple path a, b, c



Undirected Graphs: Connectedness

- An undirected graph is called **connected** if there is a path between every pair of distinct vertices of the graph.

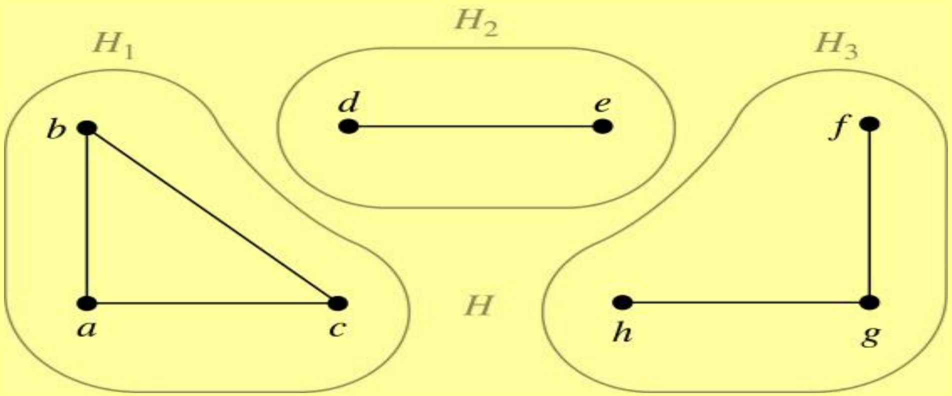
e.g.



G_1 is connected and G_2 is not connected (No path between a and d)

Connected Components

- A graph that is not connected is the union of two or more connected subgraphs with no vertices in common. These disjoint connected subgraphs are called the **connected components** of the graph.



Graph *H* has three connected components

Directed Graphs: Connectedness

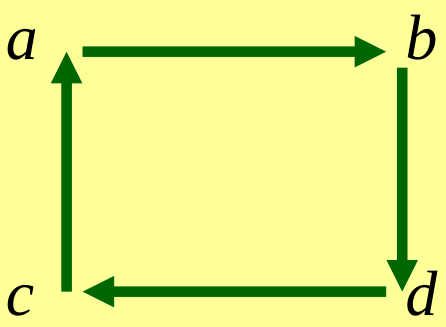
- <u>Definition 1:</u>

  A directed graph is **strongly connected** if there is a path from *a* to *b* and from *b* to *a* whenever *a* and *b* are vertices in the graph.

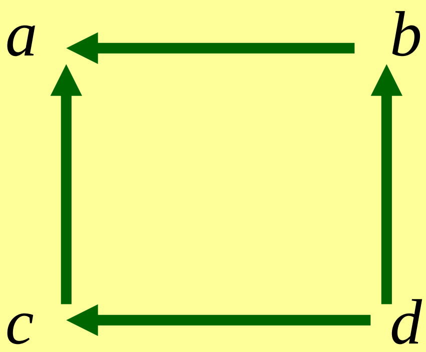
- <u>Definition 2:</u>

  A directed graph is **weakly connected** if there is a path between every two vertices when the direction of edges are disregarded.

Example



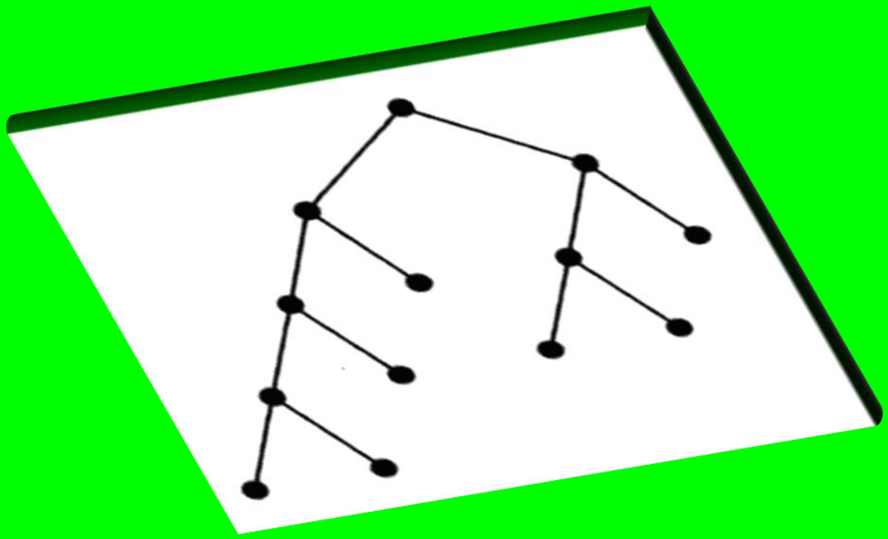
Strongly connected



weakly connected

(No directed path from *a* to *b*)

11. Trees

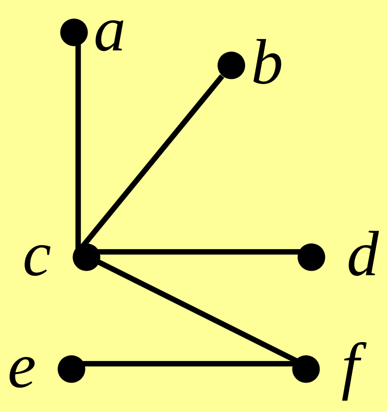


Trees

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

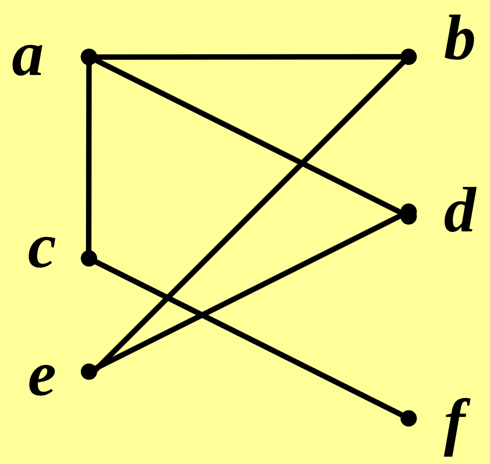
- **Properties:**
  - There is a unique simple path between any 2 of its vertices.
  - No loops.
  - No multiple edges.

Example 1



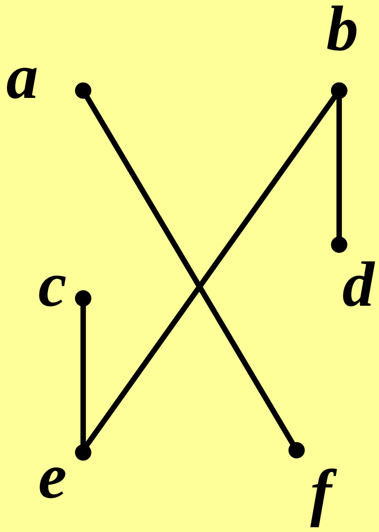
This graph is a <u>Tree</u> because it is a connected graph with no simple circuits

Example 2



It is <u>not a tree</u> " because there is a cycle *a, b, e, d, a*"

Example 3



It is <u>not a tree</u> "because it's not connected". In this case it's called a forest in which each connected component is a tree.

Component 1: ***a, f***

Component 2: *c, e, b, d*

Forest

- An undirected graph without simple circuits is called a **forest**.

– You can think of it as a set of trees having disjoint sets of nodes.

Forest

This is one graph with three connected components.

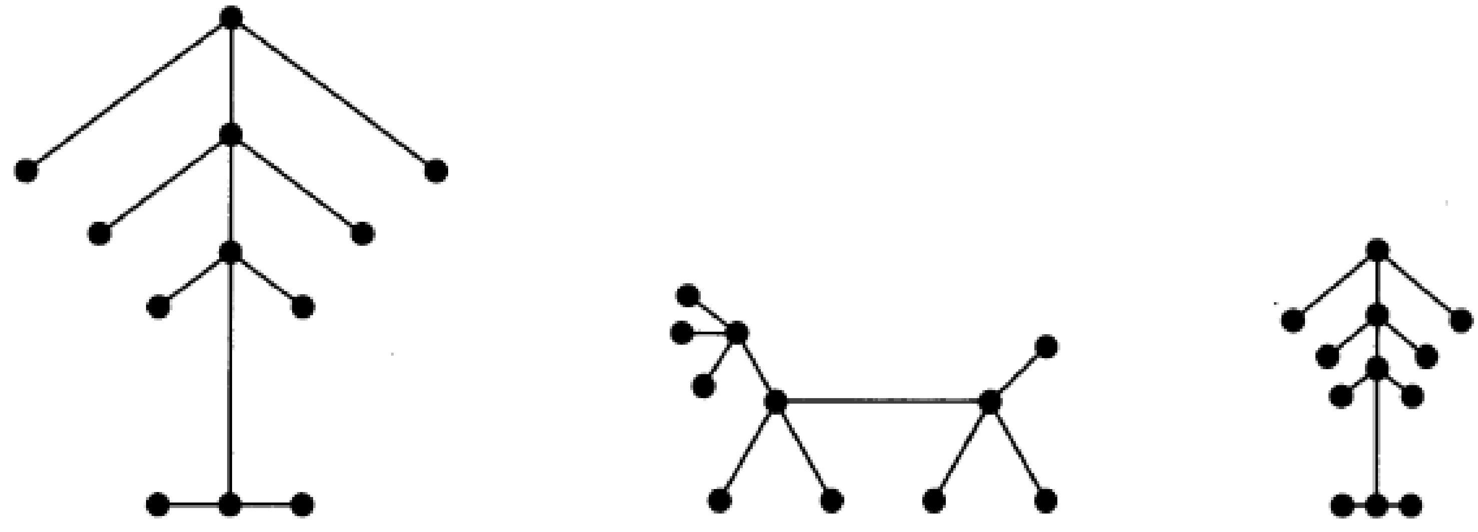


FIGURE 3 Example of a Forest.

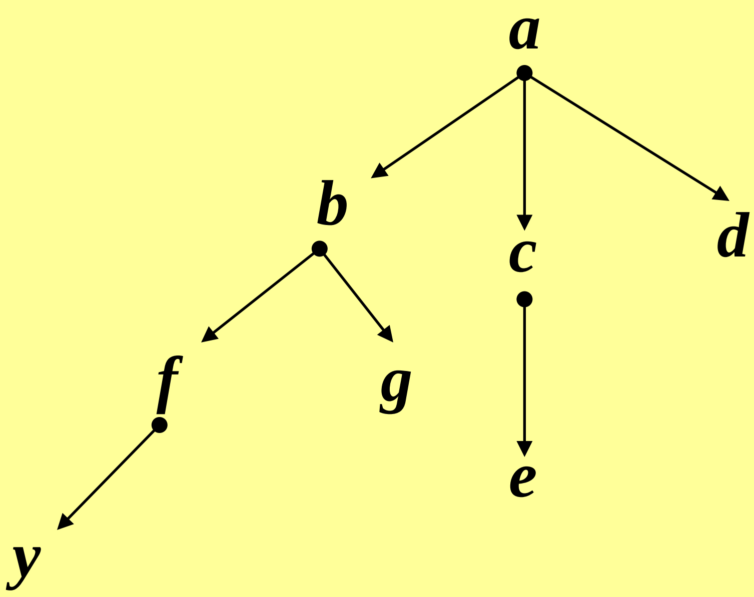
Rooted (Directed) Trees

- A **rooted tree** is a tree in which one node has been designated the **root** and every edge is directed away from the root.
- You should know the following terms about rooted trees:

  Root, Parent, Child, Siblings, Ancestors,

  Descendents, Leaf, Internal node, Subtree.

Definitions



- **Root**: Vertex with in-degree 0

[Node *a* is the root]

Definitions

- **Parent**: Vertex u is a parent, such that there is directed edge from u to v.

  [*b* is parent of *g* and *f*]

- **Child:** If u is parent of v, then v is child of u.

  [*g* and *f* are children of *b*]

- **Siblings:** Vertices with the same parents.

  [*f* and *g*]

- **Ancestors:** Vertices in path from the root to vertex v, excluding v itself, including the root.

  [Ancestors of *g* : *b, a*]

Definitions

- **Descendants**: All vertices that have *v* as ancestors.

  [**Descendants of *b* : *f, g, y***]

- **Leaf:** Vertex with no children.

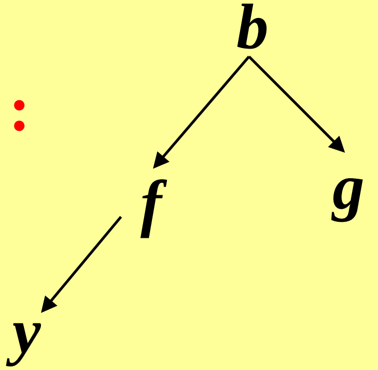
  [*y, g, e, d*]

- **Internal vertices:** Vertices that have children.

  [*a, b, c, f*]

- **Subtree:** Subgraphs consisting of *v* and its descendants and their incident edges.

  Subtree rooted at *b* :



Definitions

- **Level (of *v*)** is length of unique path from root to *v*.

  [level of root = 0, level of *b* = 1, level of *g* = 2]

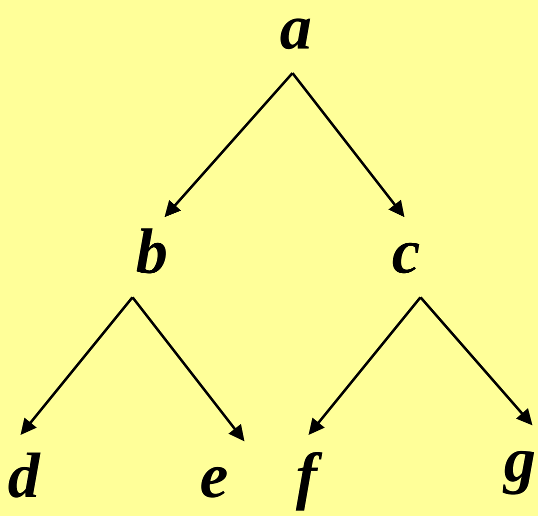
- **Height** is maximum of vertices levels.

  [Height = 3]

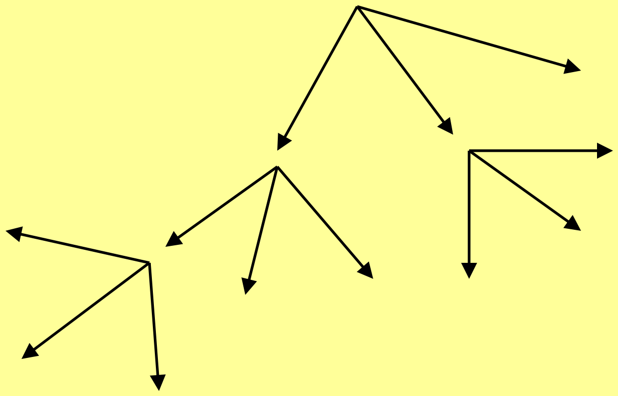
m-ary Trees

- A rooted tree is called ***m*-ary** if every internal vertex has no more than *m* children.
- It is called **full *m*-array** if every internal vertex has **exactly** *m* children.
- A 2-ary tree is called a **binary tree**.

Example



Full binary tree

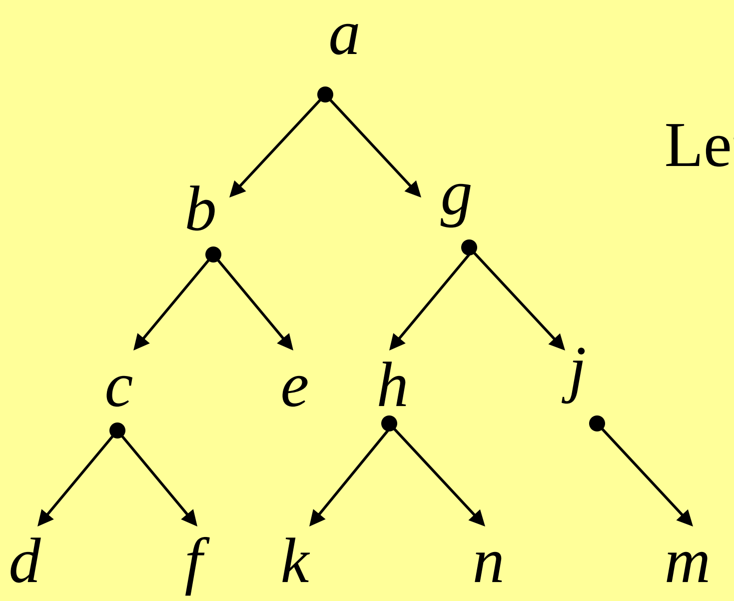


Full 3-ary tree

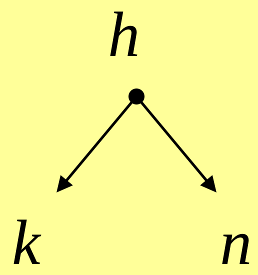
Ordered Rooted Tree

- A rooted tree where the children of each internal node are ordered.
- In ordered binary trees, we can define:
  - **left child, right child**
  - **left subtree, right subtree**
- For m-ary trees with $m > 2$, we can use terms like “leftmost”, “rightmost,” etc.

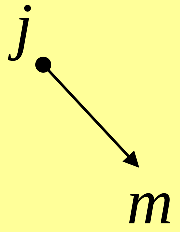
Examples



Left subtree of *g*



Right subtree of *g*



Left child of c is d , Right child of c is f

Trees as Models

Saturated Hydrocarbons and Trees.

Graphs can be used to represent molecules, where atoms are represented by vertices and bonds between them by edges.

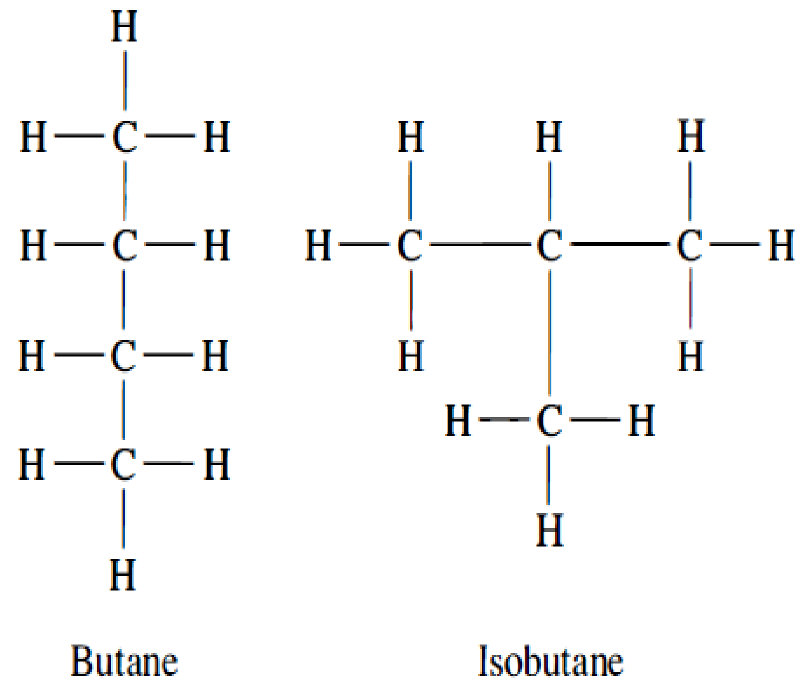


FIGURE 9 The Two Isomers of Butane.

Trees as Models

Representing Organizations The structure of a large organization can be modeled using a rooted tree. Each vertex in this tree represents a position in the organization. An edge from one vertex to another indicates that the person represented by the initial vertex is the (direct) boss of the person represented by the terminal vertex.

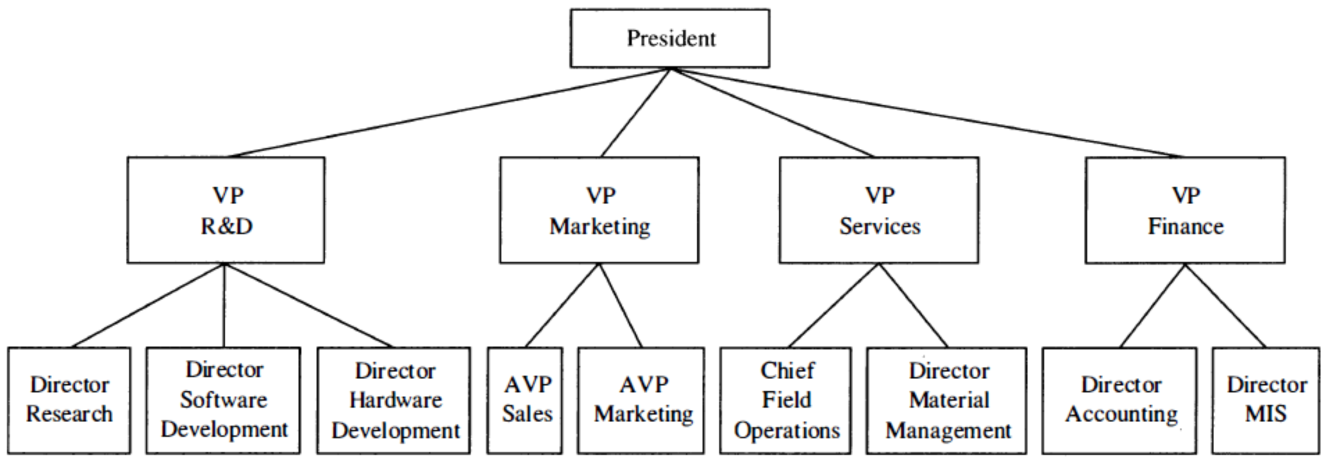


FIGURE 10 **An Organizational Tree for a Computer Company.**

Trees as Models

Computer File Systems Files in computer memory can be organized into directories. A directory can contain both files and subdirectories. The root directory contains the entire file system. Thus, a file system may be represented by a rooted tree, where the root represents the root directory, internal vertices represent subdirectories, and leaves represent ordinary files or empty directories.

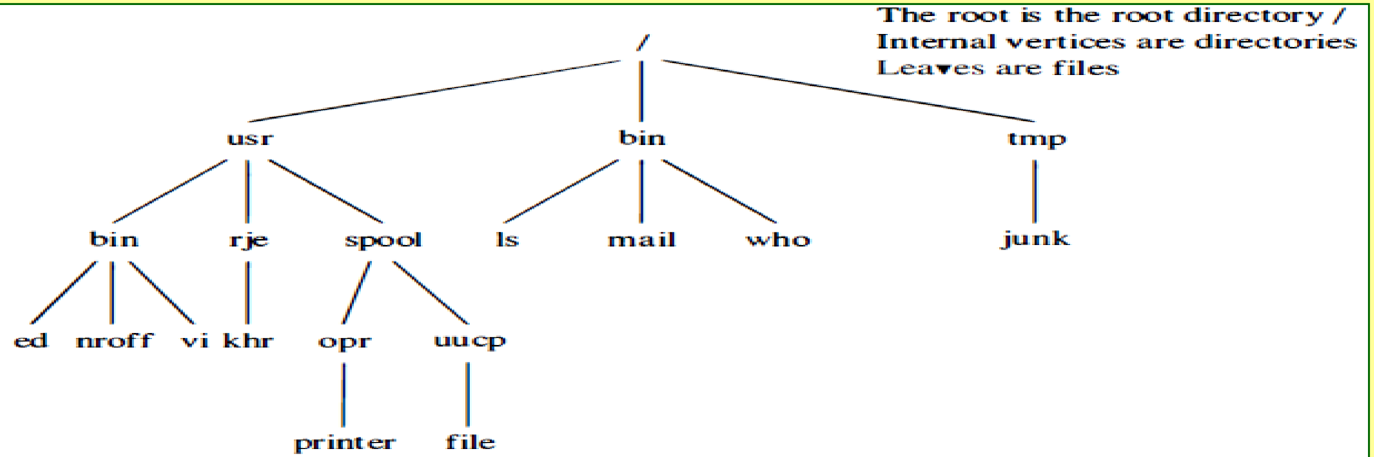
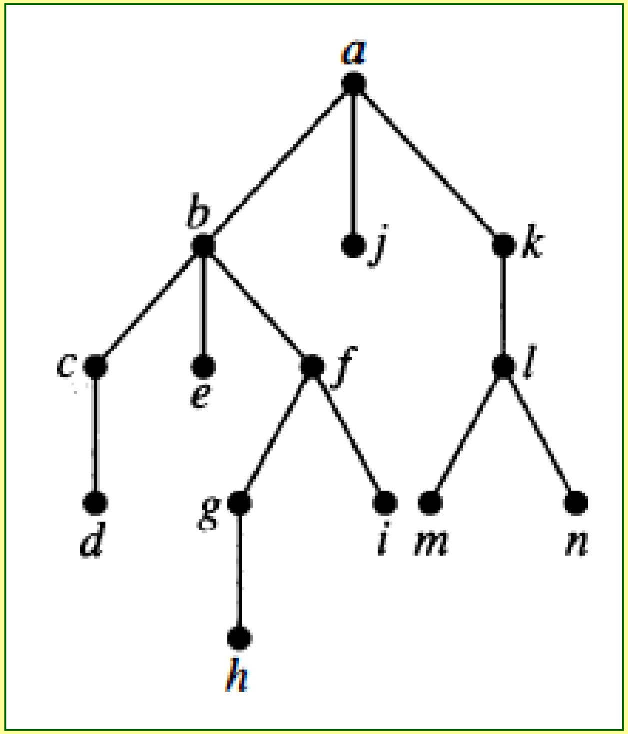


FIGURE 11 **A Computer File System.**

Properties of Trees

1- A tree with *n* vertices has **n - 1 edges.**

e.g. The tree in the figure has 14 vertices and 13 edges

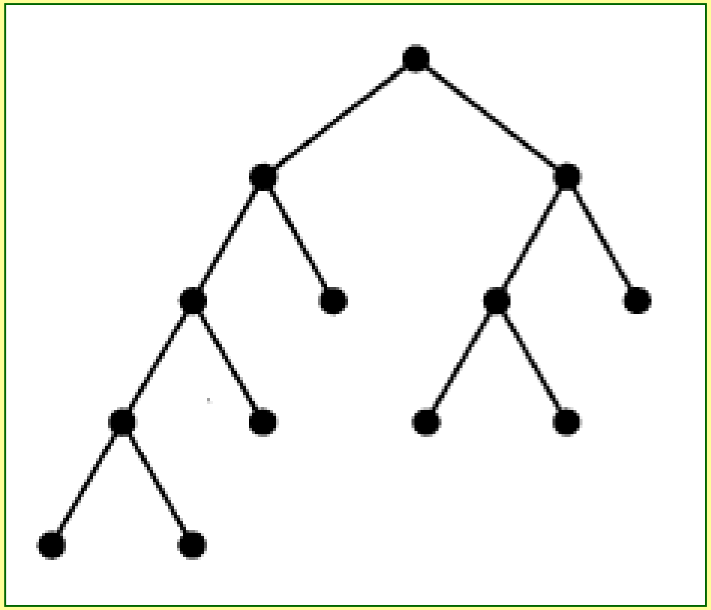


Properties of Trees

2- A full m-ary tree with I internal vertices and L leaves contains:

$$\boldsymbol{n = m \times I + 1} \textbf{ vertices}$$

$$\boldsymbol{n = I + L} \textbf{ vertices}$$



e.g. The full binary tree in the figure has:

Internal vertices $I = 6$

Leaves $L = 7$

Vertices $13 = (2)(6) + 1$

Summary

For a full *m*-ary tree:

(i) Given n vertices, if $m = 2$ and $n = 13$, then

$$I = (n - 1) / m = 6 \text{ internal vertices and}$$

$$L = n - I = 7 \text{ leaves.}$$

(ii) Given I internal vertices, if $m = 2$ and $I = 6$, then

$$n = m \times I + 1 = 13 \text{ vertices and}$$

$$L = n - I = 7 \text{ leaves.}$$

(iii) Given L leaves, if $m = 2$ and $L = 6$, then

$$n = (m \times L - 1) / (m - 1) = 13 \text{ vertices and}$$

$$I = n - L = 6 \text{ internal vertices.}$$

Properties of Trees

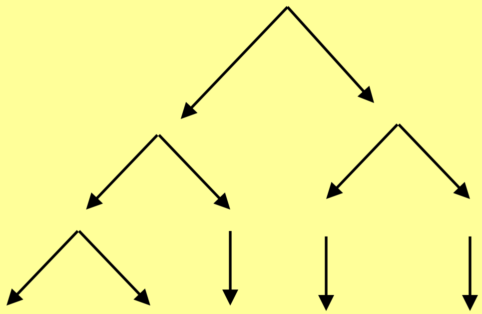
3- The **level** of a vertex in a rooted tree is the length of the path from the root to the vertex (level of the root is 0)

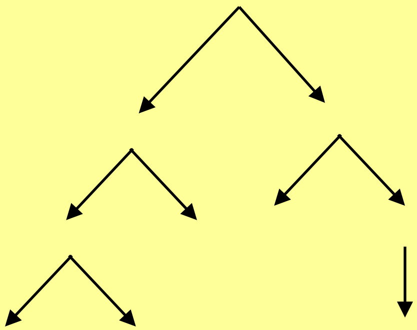
4- The **height** of the rooted tree is the maximum of the levels of vertices (length of the longest path from the root to any vertex)

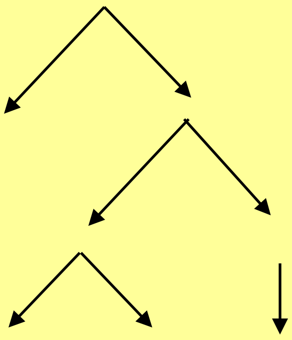
Balanced Trees

- **Balanced Tree**

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







Balanced **Balanced** **Not Balanced**

Binary Search Trees

A **binary search tree** is a binary tree in which each child of a vertex is designated as a right or left child, no vertex has more than one right child or left child, and each vertex is labeled with a key, which is one of the items. Furthermore, vertices are assigned keys so that the key of a vertex is both larger than the keys of all vertices in its left subtree and smaller than the keys of all vertices in its right subtree.

Example 2

Set up a binary tree for the following list of numbers, in an increasing order: 45, 13, 55, 50, 20, 60, 10, 15, 40, 32.

- What is the height of the shortest binary search tree that can hold all 10 numbers?
- Is it a full binary tree?
- Is it a balanced tree?

45, 13, 55, 50, 20, 60, 10, 15, 40, 32

- The height of the binary search tree is 4
- It is not a full binary tree
- It is not a balance tree