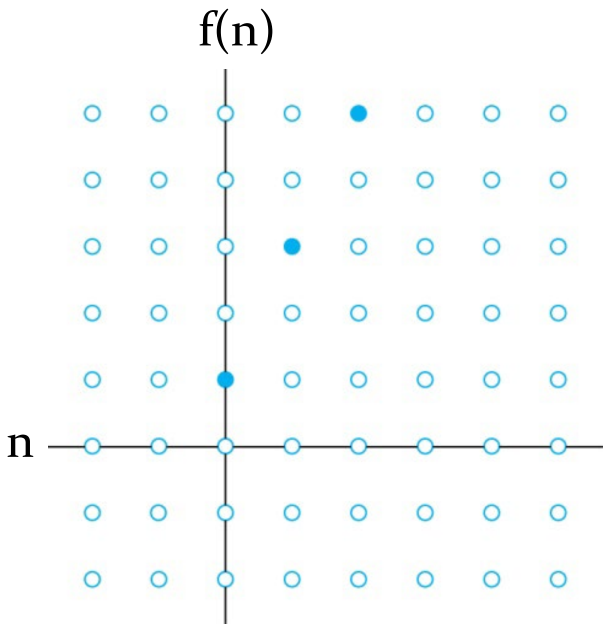
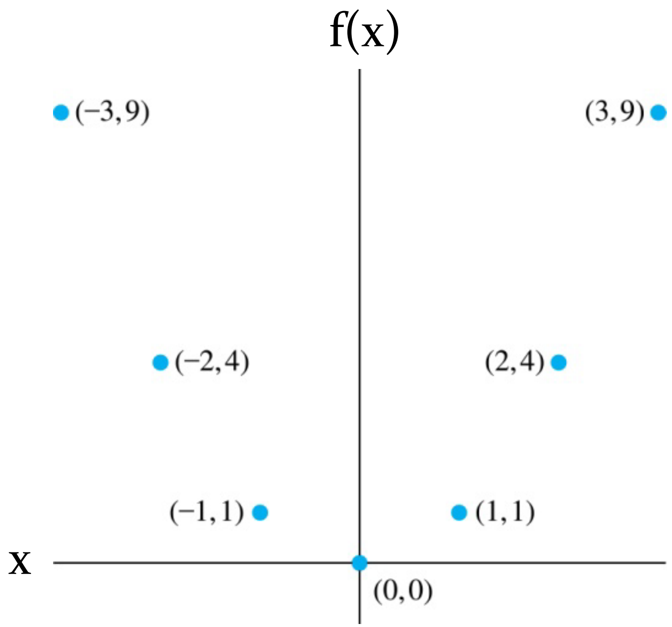
Graphs and Graph Models

Graphs of Functions

- Let f be a function from the set A to the set B.
  - The ***graph*** of the function f is the set of ordered pairs $\{(a,b) \mid a \in A \text{ and } f(a) = b\}$.



Graph of $f(n) = 2n + 1$ from Z to Z

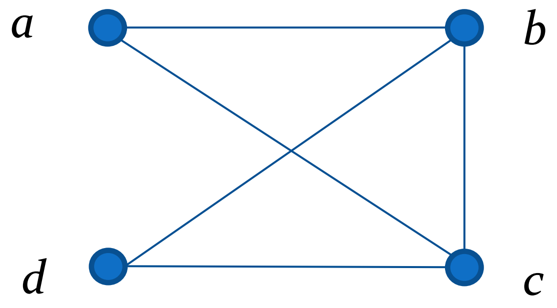


Graph of $f(x) = x^2$ from Z to Z

Graphs

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

Example: This is a graph with four vertices and five edges.



Remarks on Graphs

- We have a lot of freedom when we draw a picture of a graph.
  - All that matters is the connections made by the edges, not the particular geometry depicted.
    - For example, the lengths of edges, whether edges cross, how vertices are depicted, and so on, do not matter

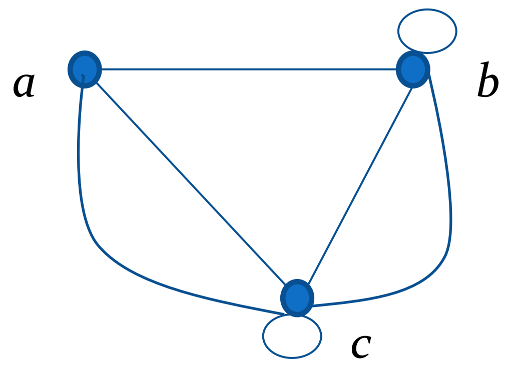
- A graph with an infinite vertex set is called an ***infinite graph***.
  - A graph with a finite vertex set is called a ***finite graph***.

- There is no standard terminology for graph theory.
  - So, it is crucial that you understand the terminology being used whenever you read material about graphs.
  - The book has its own terminology, but you should not be limited to it.

Some Terminology

- ***Simple graph***
  - Each edge connects two different vertices
  - No two edges connect the same pair of vertices.
- ***Multigraphs***
  - May have **multiple edges** connecting the same two vertices.
  - When m different edges connect the vertices u and v, we say that $\{u,v\}$ is an edge of ***multiplicity*** m.
- ***Pseudograph***
  - May include **loops**, an edge that connects a vertex to itself
  - May also have **multiple edges** connecting the same pair of vertices.

Example: This **pseudograph** has both multiple edges and a loop.



Directed Graphs

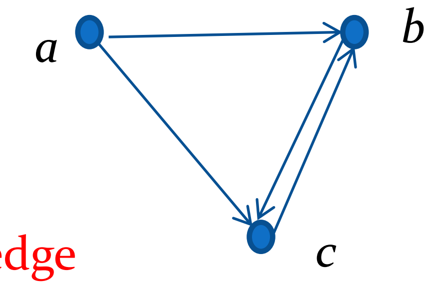
- **Definition:** A ***directed graph*** (or ***digraph***) $G = (V, E)$ consists of
  - a nonempty set V of ***vertices*** (or ***nodes***) and
  - a set E of ***directed edges*** (or ***arcs***).

- Each 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.

- Graphs where the end points of an edge are not ordered are said to be ***undirected graphs***.

Directed Graph Terminology

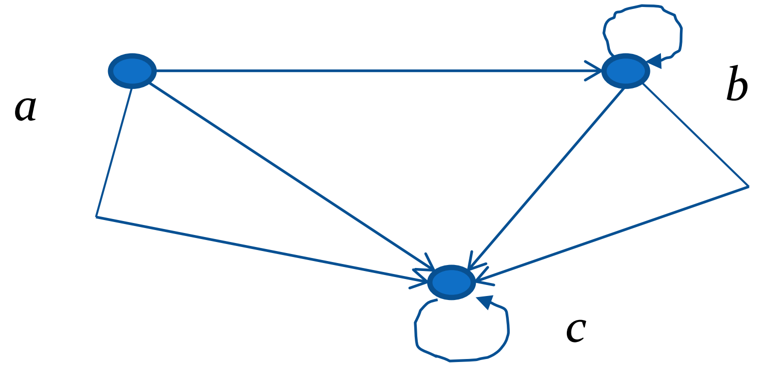
- A ***simple directed graph*** has no loops and no multiple edges.



- **Example**:
  - This is a directed graph with three vertices and four edges.
  - Edges ordered differently, but with same vertices, are not considered the same edge

- A ***directed multigraph*** may have multiple directed edges.
  - May also have loops.
  - When there are m directed edges from the vertex u to the vertex v,
    - we say that (u,v) is an edge of ***multiplicity*** m.

- **Example**:
  - In this directed multigraph
    - **multiplicity of (a,b)** is 1
    - **multiplicity of** (b,c) is 2.

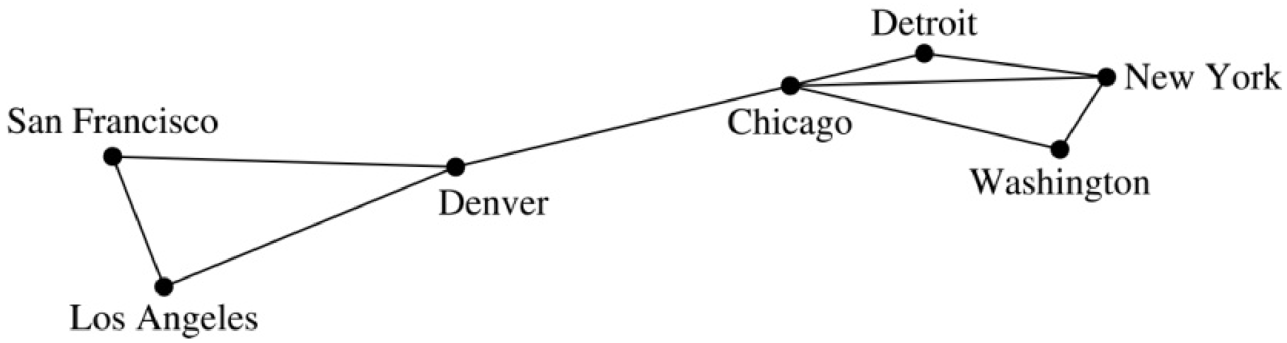


Graph Models: Computer Networks

- When we build a **graph model**, we use the appropriate type of graph to capture the important features of the application.

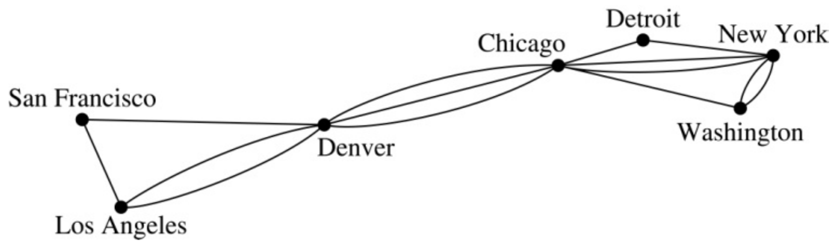
- **Example**: graph models of different types of **computer networks**.
  - The vertices represent data centers
  - the edges represent communication links

- To model a computer network where we are only concerned whether two data centers are connected by a communications link, we use a **simple graph**.
  - Only care whether two data centers are directly linked
  - Don't care how many links there may be
  - All communications links work in both directions.

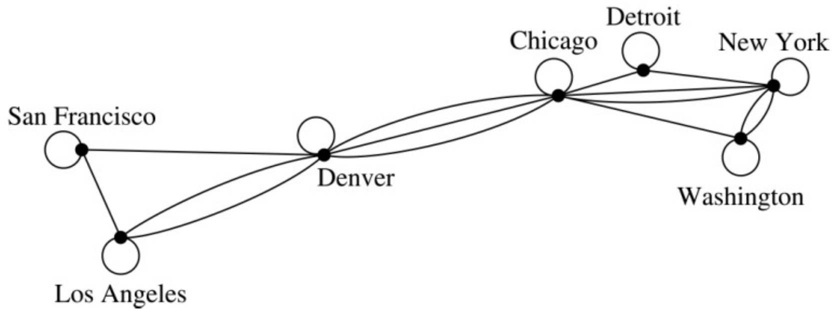


Graph Models: Computer Networks

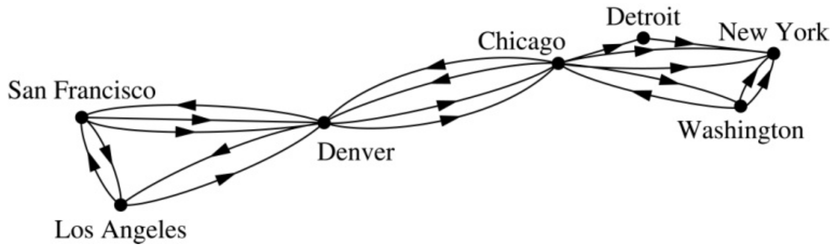
- To model a computer network where we care about the number of links between data centers, we use a **multigraph**.



- To model a computer network with diagnostic links at data centers, we use a **pseudograph**, as loops are needed.



- To model a network with multiple one-way links, we use a **directed multigraph**.



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

Software Design Applications

- ***Precedence graph***
  - Represents which statements must have already been executed before we execute each statement.
  - **Vertices** represent statements in a computer program
  - There is a **directed edge** from a vertex to a second vertex if the second vertex cannot be executed before the first

Example: This precedence graph shows which statements must already have been executed before we can execute each of the six statements in the program.

S_1 $a := 0$

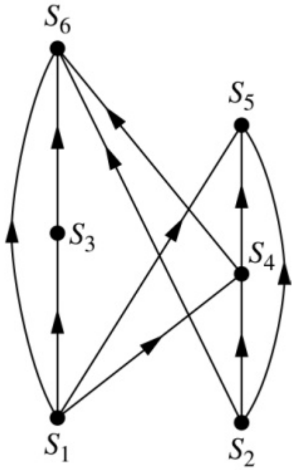
S_2 $b := 1$

S_3 $c := a + 1$

S_4 $d := b + a$

S_5 $e := d + 1$

S_6 $e := c + d$



Graph Terminology and Special Types of Graphs

Basic Terminology

- **Definition 1**. Two vertices *u*, *v* in an undirected graph *G* are called ***adjacent*** (or ***neighbors***) in *G* if there is an edge *e* between *u* and *v*.
  - Such an edge *e* is called ***incident*** *with* the vertices *u* and *v* and *e* is said to ***connect*** *u* and *v*.

- **Definition 2**. The set of all neighbors of a vertex *v*, denoted by ***N*(*v*)**, is called the ***neighborhood*** of *v*.
  - If v has a loop, *v* will be included in the neighborhood of *v*
  - If *A* is a subset of *V*, we denote by *N*(*A*) the set of all vertices that are adjacent to at least one vertex in *A*.
  - So, $N(A) = \bigcup_{v \in A} N(v)$.

- **Definition 3**. The ***degree*** *of a vertex in a undirected graph* is the number of edges incident with it,
  - Except that a loop at a vertex contributes two to the degree of that vertex.
  - The degree of the vertex *v* is denoted by **deg(*v*)**.

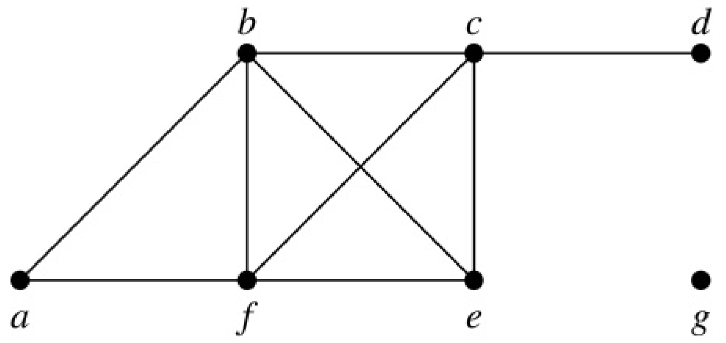
Degree Terminology

- A vertex of degree zero is called an **isolated** vertex

- A vertex is **pendant** if and only if it has degree one.
  - A pendant vertex is adjacent to exactly one other vertex

Degrees and Neighborhoods of Vertices

Example: What are the **degrees** and **neighborhoods** of the vertices in the graphs G and H?



G

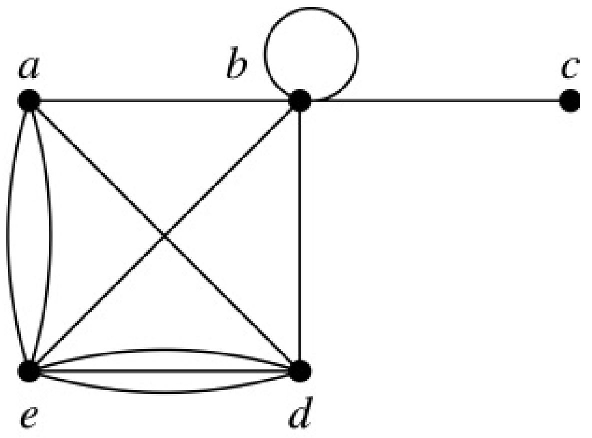
Solution:

G: $\deg(a) = 2$, $\deg(b) = \deg(c) = \deg(f) = 4$, $\deg(d) = 1$, $\deg(e) = 3$, $\deg(g) = 0$.

$N(a) = \{b, f\}$, $N(b) = \{a, c, e, f\}$, $N(c) = \{b, d, e, f\}$, $N(d) = \{c\}$, $N(e) = \{b, c, f\}$, $N(f) = \{a, b, c, e\}$, $N(g) = \varnothing$.

Degrees and Neighborhoods of Vertices

Example: What are the **degrees** and **neighborhoods** of the vertices in the graphs G and H?



H

Solution:

H: deg(a) = 4, deg(b) = deg(e) = 6, deg(c) = 1, deg(d) = 5.

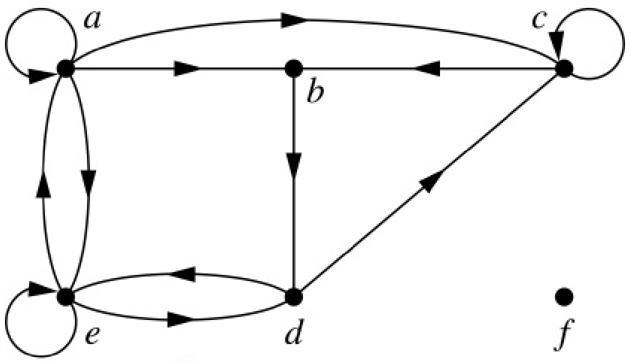
$N(a) = \{b, d, e\}$, $N(b) = \{a, b, c, d, e\}$, $N(c) = \{b\}$,
$N(d) = \{a, b, e\}$, $N(e) = \{a, b, d\}$.

Vertices of Directed Graphs

- **Definition**: Let (u,v) be a directed edge in the directed graph G.
    - Then u is the ***initial vertex*** of this edge and is ***adjacent to*** v and v is the ***terminal*** (or ***end***) ***vertex*** of this edge and is ***adjacent from*** u.
    - The initial and terminal vertices of a loop are the same.

Directed Graphs

- **Definition:** The ***in-degree*** *of a vertex v*, denoted ***deg⁻(v)***, is the number of edges which terminate at v.
  - The ***out-degree*** *of v*, denoted ***deg⁺(v)***, is the number of edges with v as their initial vertex.
  - Note that a loop at a vertex contributes 1 to both the in-degree and the out-degree of the vertex.
- **Example:** In the graph G we have



G

$\deg^-(a) = 2$, $\deg^-(b) = 2$, $\deg^-(c) = 3$,
$\deg^-(d) = 2$, $\deg^-(e) = 3$, $\deg^-(f) = 0$.

$\deg^+(a) = 4$, $\deg^+(b) = 1$, $\deg^+(c) = 2$,
$\deg^+(d) = 2$, $\deg^+(e) = 3$, $\deg^+(f) = 0$.

Directed Graphs

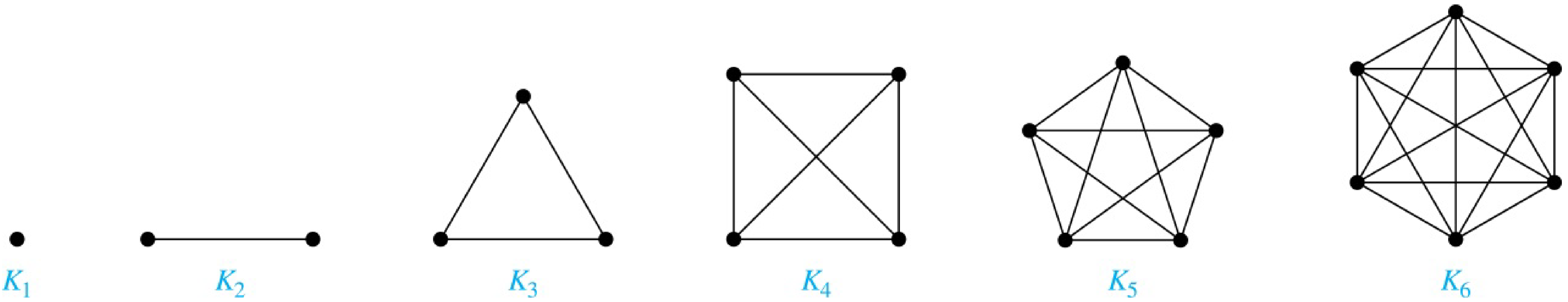
- **Theorem 3**: Let $G = (V, E)$ be a graph with directed edges.
  - Then:

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

- ***Proof***: The first sum counts the number of outgoing edges over all vertices and the second sum counts the number of incoming edges over all vertices.
  - It follows that both sums equal the number of edges in the graph. ◀

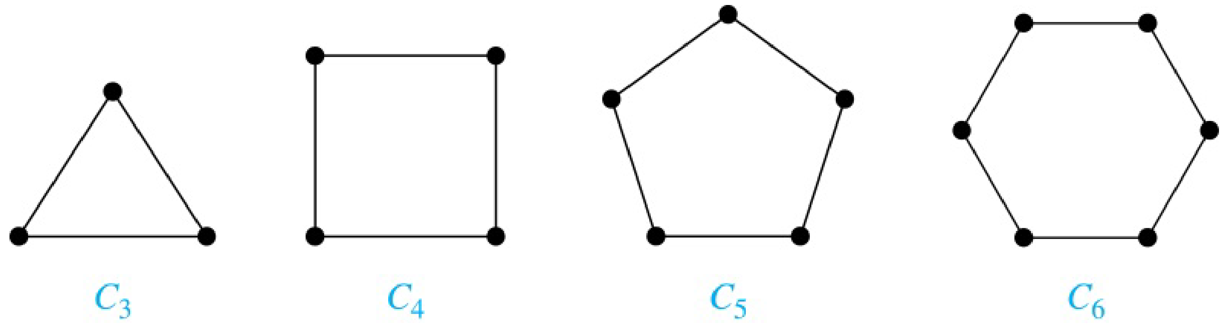
Special Types: Complete Graphs

A ***complete graph*** *on n vertices*, denoted by K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.

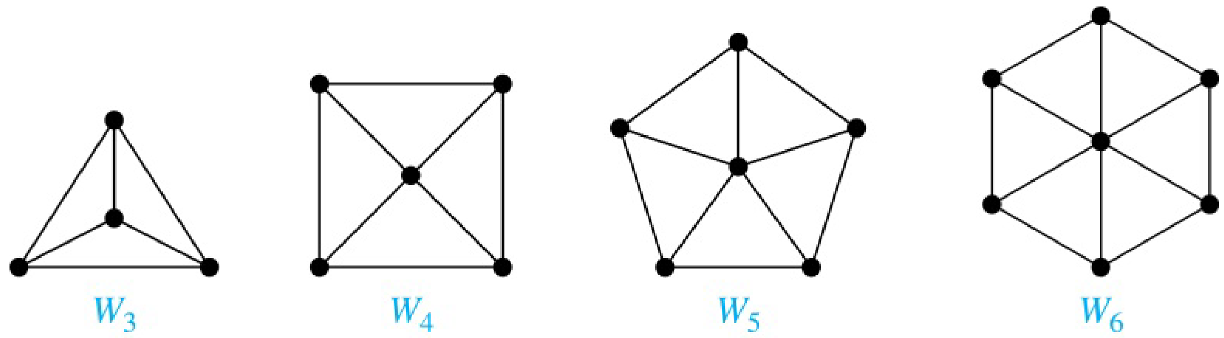


Special Types: Cycles and Wheels

- A ***cycle*** C_n for $n \geq 3$ consists of n vertices $v_1, v_2, \cdots, v_n$, and edges $\{v_1, v_2\}$, $\{v_2, v_3\}, \cdots, \{v_{n-1}, v_n\}$, $\{v_n, v_1\}$.

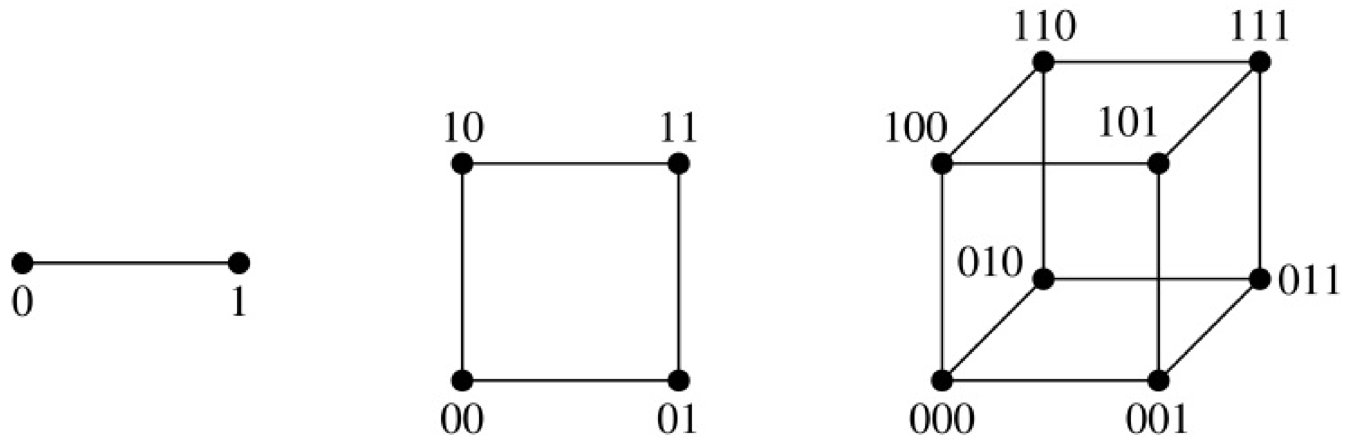


- A ***wheel*** W_n is obtained by adding an additional vertex to a cycle C_n for $n \geq 3$ and connecting this new vertex to each of the n vertices in C_n by new edges.



Special Types: *n*-Cubes

- An ***n-dimensional hypercube***, or ***n-cube***, $\boldsymbol{Q_n}$, is a graph with 2^n vertices representing all bit strings of length n, where there is an edge between two vertices that differ in exactly one bit position.

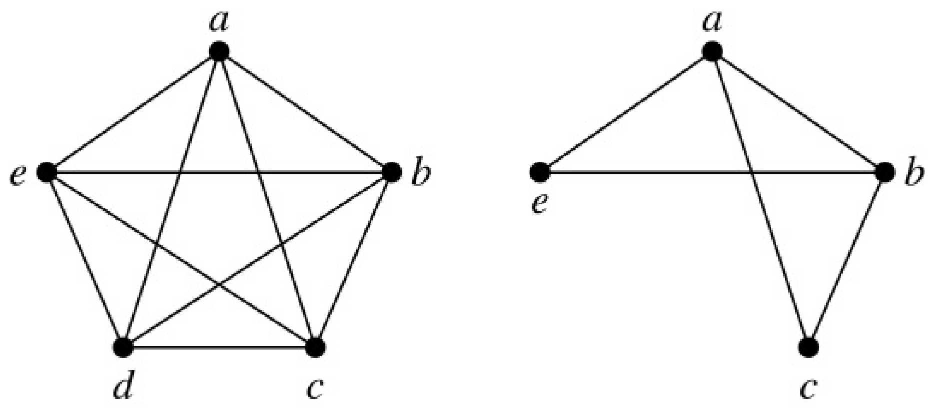


Q_1 Q_2 Q_3

Subgraphs

- **Definition:** A ***subgraph*** *of a graph* $G = (V,E)$ is a graph $H = (W,F)$
  - $W \subset V$ and $F \subset E$.
  - A subgraph H of G is a **proper subgraph** of G if $H \neq G$.

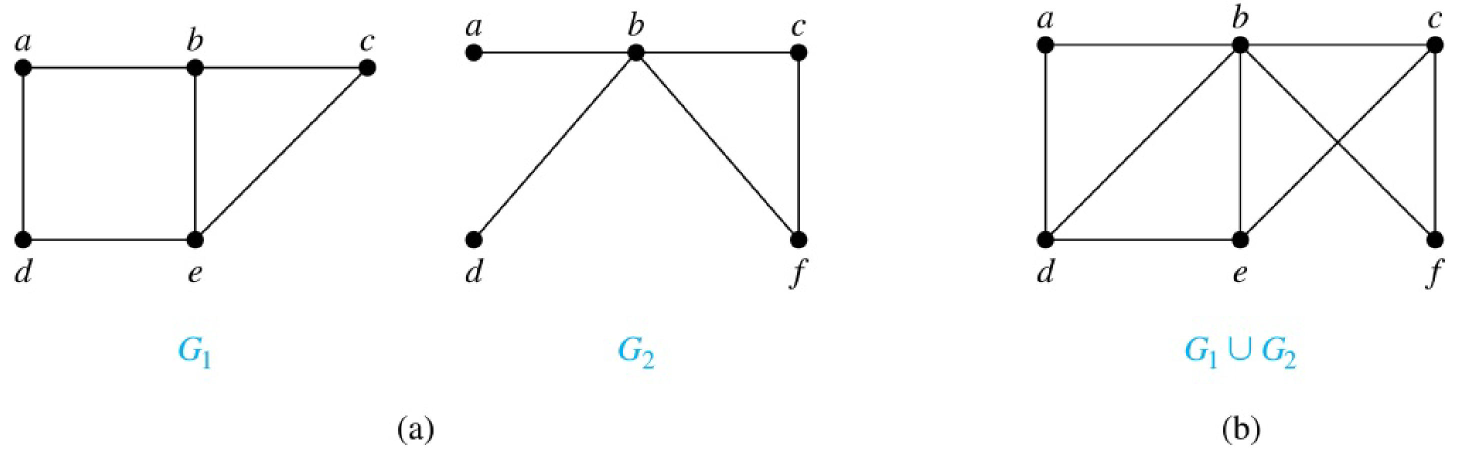
- **Example**: Here we show a graph and one of its subgraphs.



New Graphs from Old

- **Definition**: The ***union*** of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$.
  - The union of G_1 and G_2 is denoted by $G_1 \cup G_2$.

- **Example**:



Representing Graphs

Representing Graphs: Adjacency Lists

- **Definition**: An ***adjacency list*** can be used to represent a graph with no multiple edges by specifying the vertices that are adjacent to each vertex of the graph.

Example:

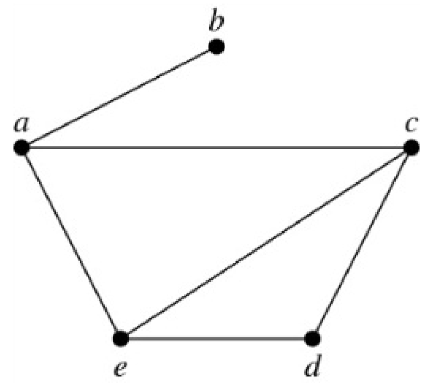


TABLE 1 An Adjacency List for a Simple Graph.

Vertex	*Adjacent Vertices*
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

Example:

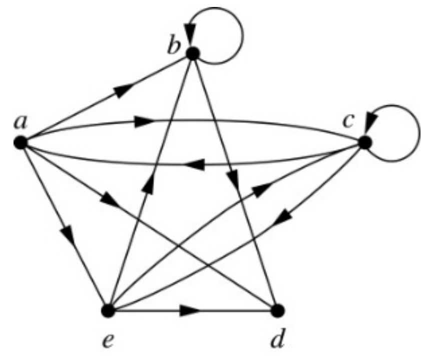


TABLE 2 An Adjacency List for a Directed Graph.

Initial Vertex	*Terminal Vertices*
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

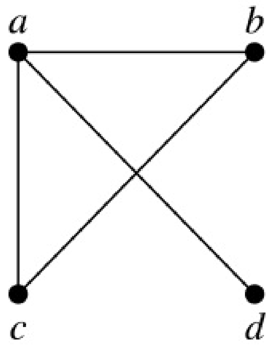
Representation of Graphs: Adjacency Matrices

- **Definition**: Suppose that $G = (V, E)$ is a simple graph where $|V| = n$.
  - Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$.

- The ***adjacency matrix*** $\mathbf{A}_G$ of G, with respect to the listing of vertices, is the $n \times n$ zero-one matrix
  - 1 as its (i, j)th entry when v_i and v_j are adjacent
  - 0 as its (i, j)th entry when they are not adjacent.
  - $$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

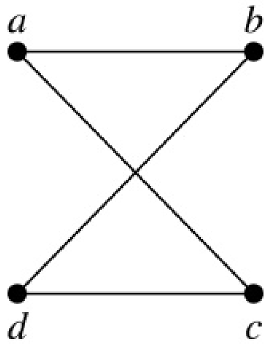
Adjacency Matrices

Example:



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

The ordering of vertices is a, b, c, d.



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

The ordering of vertices is a, b, c, d.

- When a graph is **sparse**
  - it is much more efficient to represent the graph using an adjacency list than an adjacency matrix.
- But for a **dense** graph, an adjacency matrix is preferable.

Note: The adjacency matrix of a **simple graph** is symmetric, i.e., $a_{ij} = a_{ji}$
Also, since there are no loops, each diagonal entry a_{ij} for $i = 1, 2, 3, \ldots, n$, is 0.

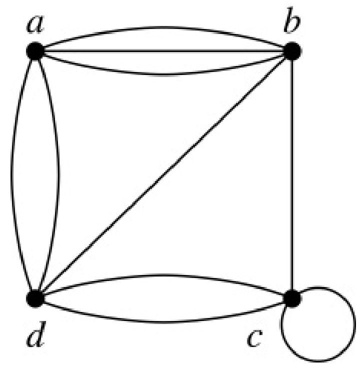
Adjacency Matrices

- Adjacency matrices can also be used to represent graphs with **loops** and **multiple edges**.

- A **loop** at the vertex v_i is represented by a 1 at the (i,i)th position of the matrix.

- When **multiple edges** connect the same pair of vertices v_i and v_j, (or if multiple loops are present at the same vertex), the (i,j)th entry equals the number of edges connecting the pair of vertices.

- **Example**: We give the adjacency matrix of the **pseudograph** shown here using the ordering of vertices a, b, c, d.



$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

Adjacency Matrices

- Adjacency matrices can also be used to represent **directed graphs**.
  - The matrix for a directed graph $G = (V, E)$ has a
  - 1 in its (i, j)th position if there is an edge from v_i to v_j
  - In other words, if the graphs adjacency matrix is $\mathbf{A}_G = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

  - The adjacency matrix for a directed graph does not have to be symmetric, because there may not be an edge from v_i to v_j, when there is an edge from v_j to v_i.
  - To represent **directed multigraphs**, the value of a_{ij} is the number of edges connecting v_i to v_j.

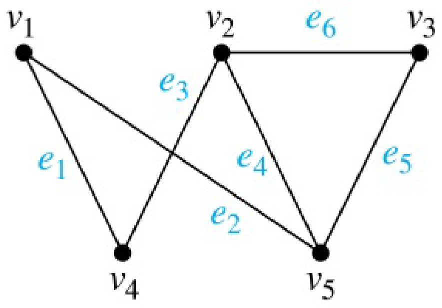
Representation of Graphs: Incidence Matrices

- **Definition**: Let $G = (V, E)$ be an undirected graph with vertices where $v_1, v_2, \ldots v_n$ and edges $e_1, e_2, \ldots e_m$.
  - The **incidence matrix** with respect to the ordering of V and E is the $n \times m$ matrix $\mathbf{M} = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

Incidence Matrices

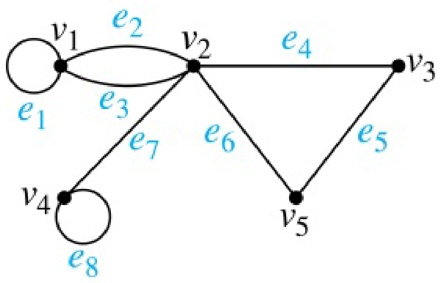
Example: **Simple Graph** and **Incidence Matrix**



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

*The **rows** going from represent v_1 through v_5 and the **columns** going from represent e_1 through e_6.*

Example: **Pseudograph** and **Incidence Matrix**



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

*The **rows** going from represent v_1 through v_5 and the **columns** going from represent e_1 through e_8.*

Loops only count once

Incidence Matrices of Directed Graphs

- Two methods:
  1. Convert directed graph to undirected graph then find incidence matrix
  2. Use -1 to specify that an edge is directed away from the vertex

- $m_{ij} = \begin{cases} 1 & if\ the\ edge\ e_j\ enters\ vertex\ v_i \\ 0 & if\ there\ is\ no\ edge\ e_j\ incident\ with\ vertex\ v_i \\ -1 & if\ the\ edge\ e_j\ leaves\ vertex\ v_i \end{cases}$

Connectivity

Paths

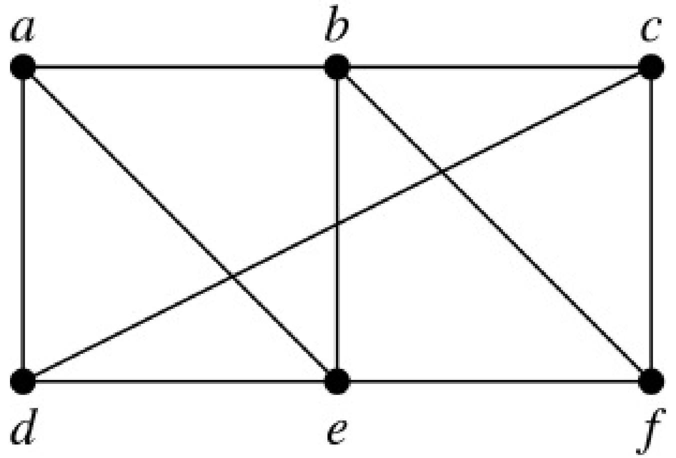
- **Informal Definition:** A ***path*** is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph.
  - As the path travels along its edges, it visits the vertices along this path.
- **Applications**: Numerous problems can be modeled with paths formed by traveling along edges of graphs such as:
  - determining whether a message can be sent between two computers.
  - efficiently planning routes for mail delivery.

Paths

- **Definition:** Let n be a nonnegative integer and G an undirected graph.
  - A ***path of length n*** from u to v in G is a sequence of n edges $e_1, \ldots, e_n$ of G
    - There exists a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices such that e_i has, for $i = 1, \ldots, n$, the endpoints x_{i-1} and x_i.
  - Denote this path by its vertex sequence $x_0, x_1, \ldots, x_n$
    - Listing the vertices uniquely determines the path.
  - The path is a ***circuit*** if it begins and ends at the same vertex ($u = v$) and has length greater than zero.
  - The path or circuit is said to ***pass through*** the vertices $x_1, x_2, \ldots, x_{n-1}$ and ***traverse*** the edges $e_1, \ldots, e_n$.
  - A path or circuit is ***simple*** if it does not contain the same edge more than once.
  - This terminology is readily extended to directed graphs.

Paths

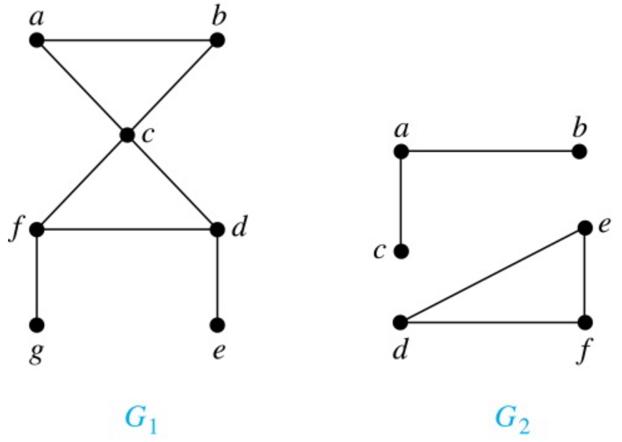
- **Example**: In the simple graph here:
  - a, d, c, f, e is a **simple path** of length 4.
  - d, e, c, a is **not a path** because e is not connected to c.
  - b, c, f, e, b is a **circuit** of length 4.
  - a, b, e, d, a, b is a path of length 5, but it is **not a simple path**.



Connectedness in Undirected Graphs

- **Definition**: An undirected graph is called ***connected*** if there is a path between every pair of vertices.
  - An undirected graph that is not *connected* is called ***disconnected***.
  - We say that we ***disconnect*** a graph when we remove vertices or edges, or both, to produce a disconnected subgraph.

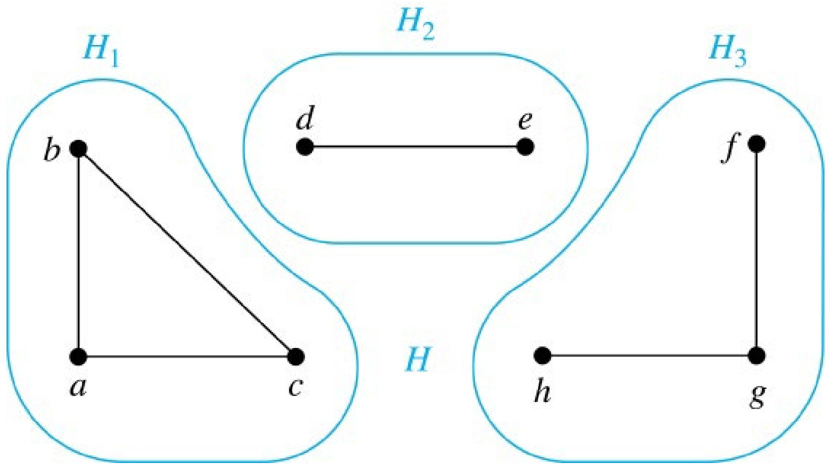
- **Example**: G_1 is **connected** because there is a path between any pair of its vertices, as can be easily seen.
  - However G_2 is **not connected** because there is no path between vertices a and f, for example.



Connected Components

- **Definition**: A ***connected component*** of a graph G is a connected subgraph of G that is not a proper subgraph of another connected subgraph of G.
  - A graph G that is **not connected** has two or more connected components
    - that are disjoint and have G as their union.

- **Example**: The graph H is the **union** of three **disjoint subgraphs** H_1, H_2, and H_3.
  - These three subgraphs are the **connected components** of H.



Connectedness in Directed Graphs

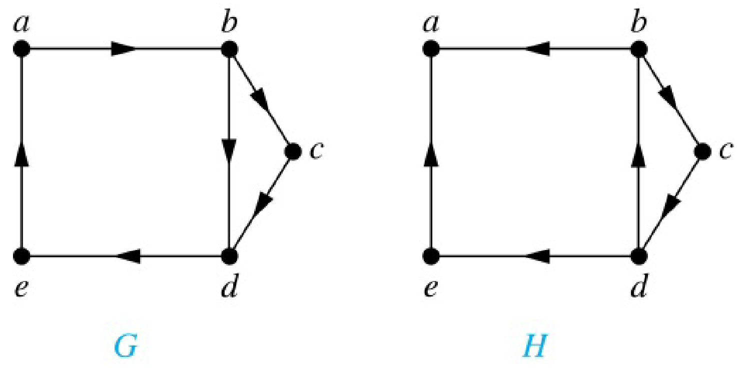
- **Definition**: A directed graph is ***strongly connected*** if there is a path from every pair of vertices a to b and a path from b to a.

- **Definition**: A directed graph is ***weakly connected*** if there is a path between every two vertices in the underlying undirected graph,
  - Undirected graph is obtained by ignoring the directions of the edges of the directed graph.

- Every strongly connected directed graph is also a weakly connected.

Connectedness in Directed Graphs

- **Example**: G is **strongly connected** because there is a path between any two vertices in the directed graph.
  - Hence, G is also weakly connected.
  - The graph H is <u>**not**</u> **strongly connected**, since there is no directed path from a to b, but it is weakly connected.



Counting Paths between Vertices

- We can use the **adjacency matrix** of a graph to find the number of paths between two vertices in the graph.

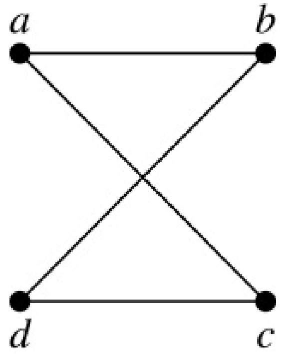
- **Theorem**: Let G be a graph with adjacency matrix **A** with respect to the ordering $v_1, \dots, v_n$ of vertices

  - The **number of different paths of length *r*** from v_i to v_j, where $r > 0$ is a positive integer, equals the (i,j)th entry of $\mathbf{A}^r$.

  - Directed or undirected edges, multiple edges and loops allowed.

Counting Paths between Vertices

- **Example**: How many paths of length four are there from a to d in the graph G.



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$ *adjacency matrix of G*

- **Solution**: The adjacency matrix of G is given above.
  - The ordering of the vertices is a, b, c, d
  - Hence the number of paths of length four from a to d is the (1, 4)th entry of $\mathbf{A}^4$.
  - The eight paths are as:

a, b, a, b, d a, b, a, c, d
a, b, d, b, d a, b, d, c, d
a, c, a, b, d a, c, a, c, d
a, c, d, b, d a, c, d, c, d

$$\mathbf{A}^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$