CS1382 Discrete Computational Structures

Lecture 14: Graphs

Spring 2019

Richard Matovu

TEXAS TECH UNIVERSITY

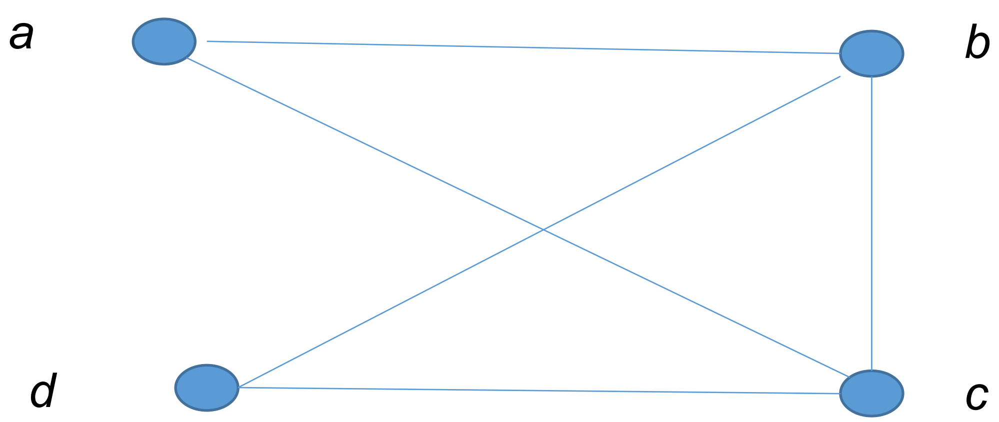
Graphs

- 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:



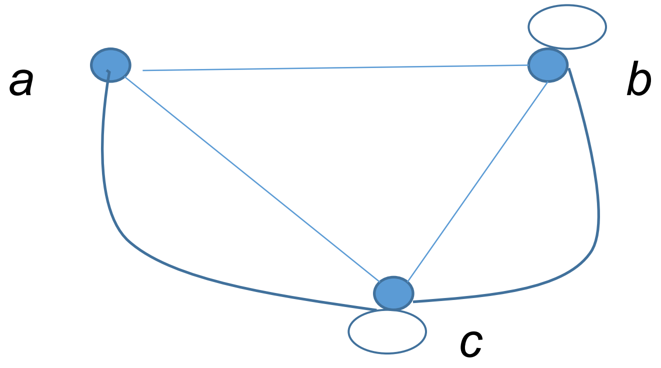
Directed and Undirected Graphs

- **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*.
- **Undirected Graph**
  - End points of an edge are not ordered

Graph Terminology

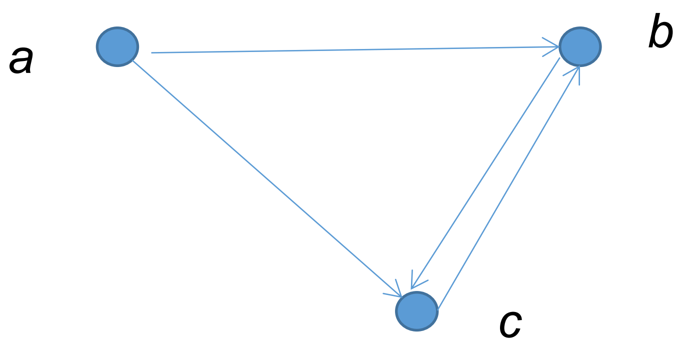
- **Simple Graph:** Each edge connects two different vertices and no two edges connect the same pair of vertices.
- **Multigraph:** May have multiple edges connecting the same two vertices.

  When *m* different edges connect the vertices *u* and *v*, we say that { *u, v* } is an edge of ***multiplicity m***.
- **Loop:** An edge that connects a vertex to itself
- **Pseudograph:** May include loops, as well as multiple edges connecting the same pair of vertices.

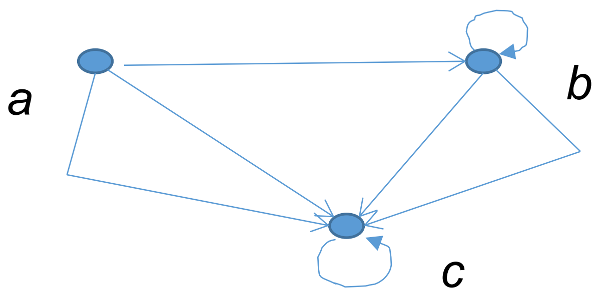


Graph Terminology (*Directed Graph*)

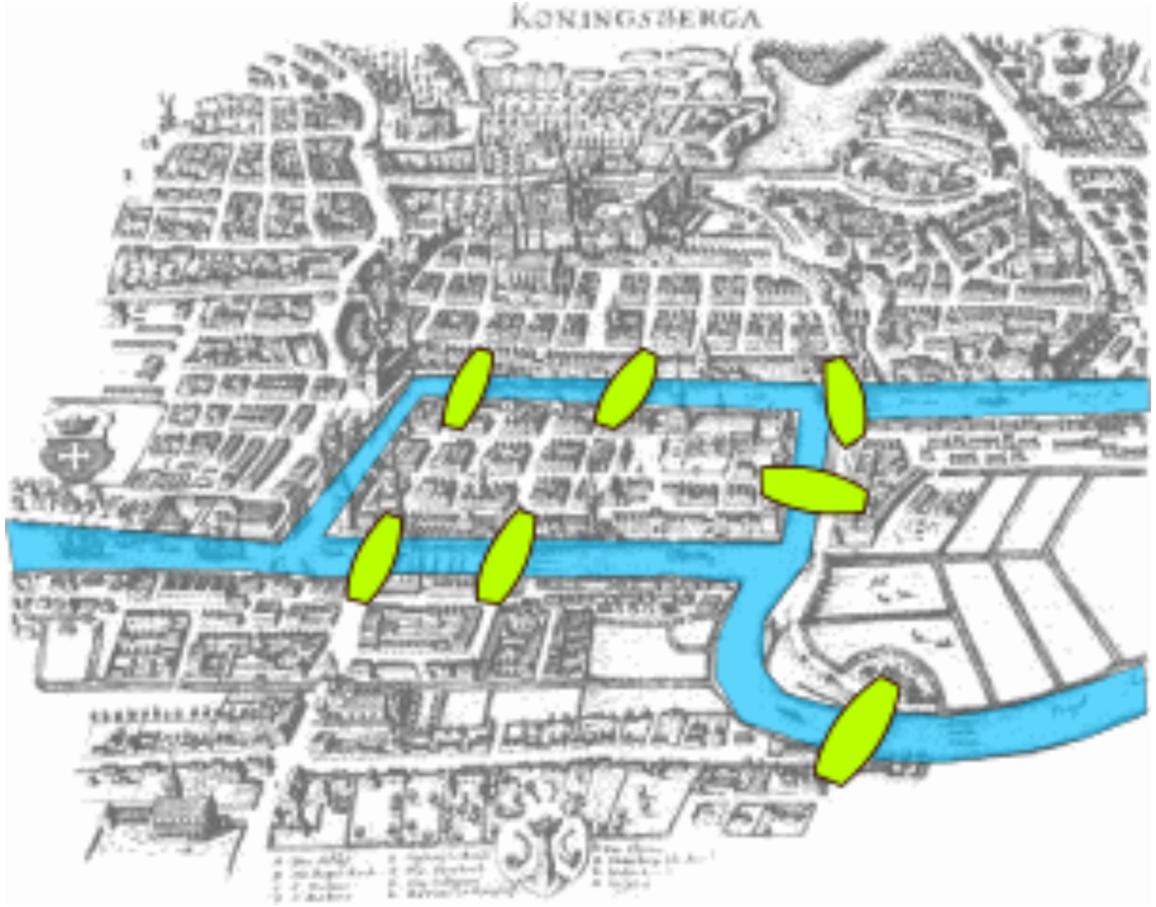
- **Simple directed graph** has no loops and no multiple edges.



- **Directed multigraph** may have multiple directed edges.

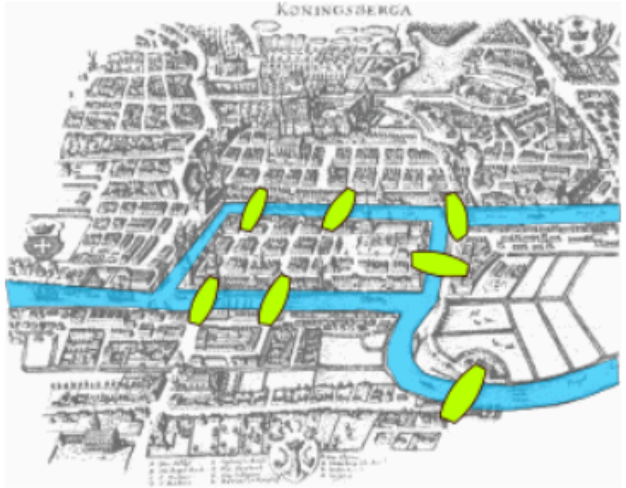


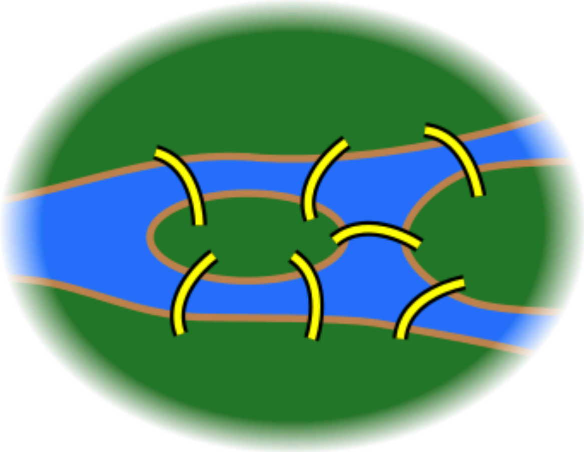
Examples: Seven Bridges of Königsberg

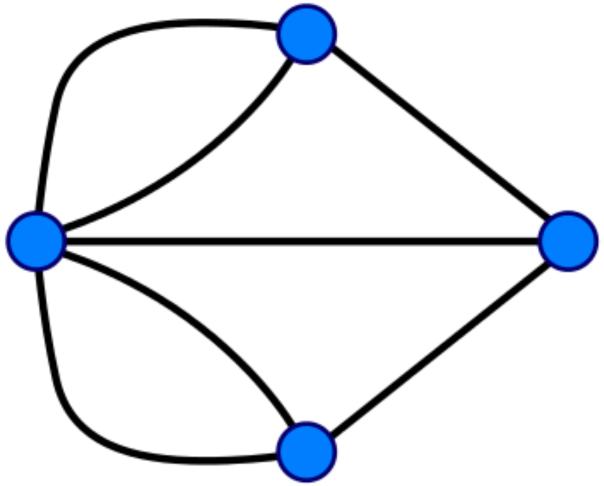


Is it possible to walk along a route that cross **each bridge exactly once?**

Seven Bridges of Königsberg







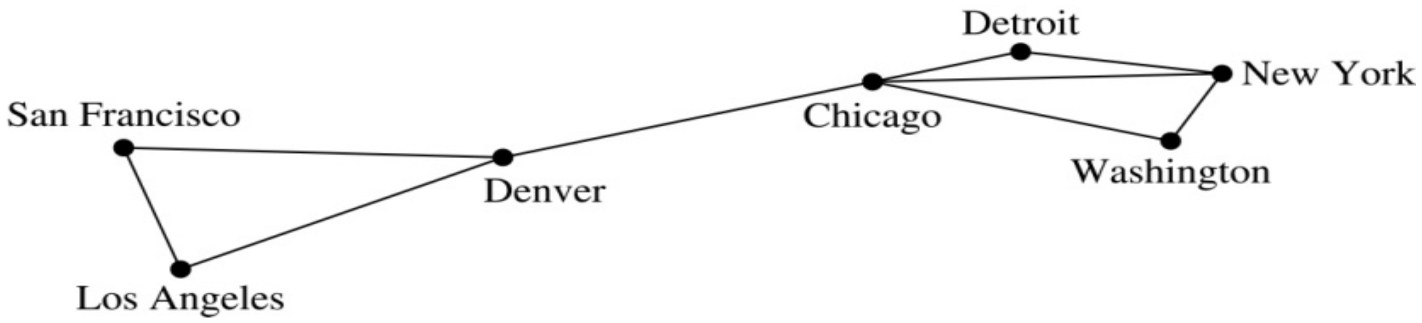
Forget unimportant details

Forget even more

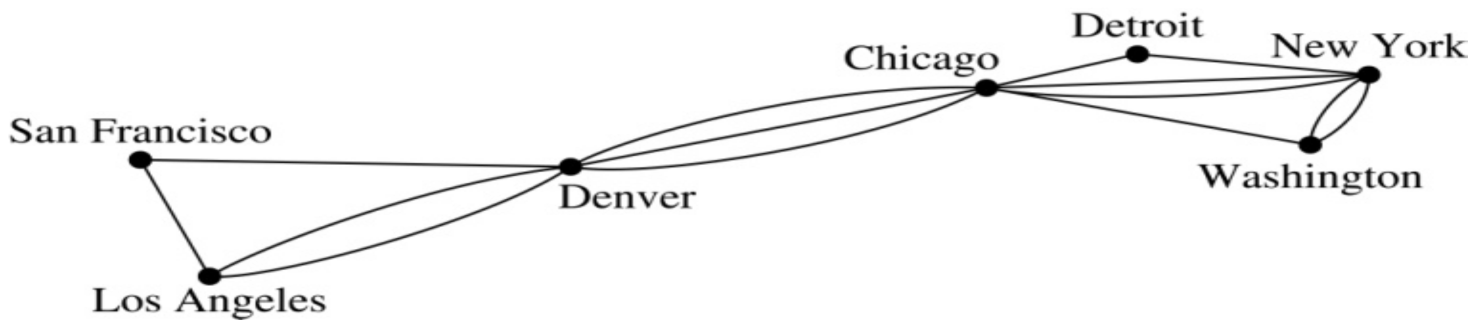
Is it possible to walk along a route that cross **each bridge exactly once?**

Graph Models: Computer Networks

- When building a graph model, we use the appropriate type of graph to capture the important features of the application
- Simple Graph

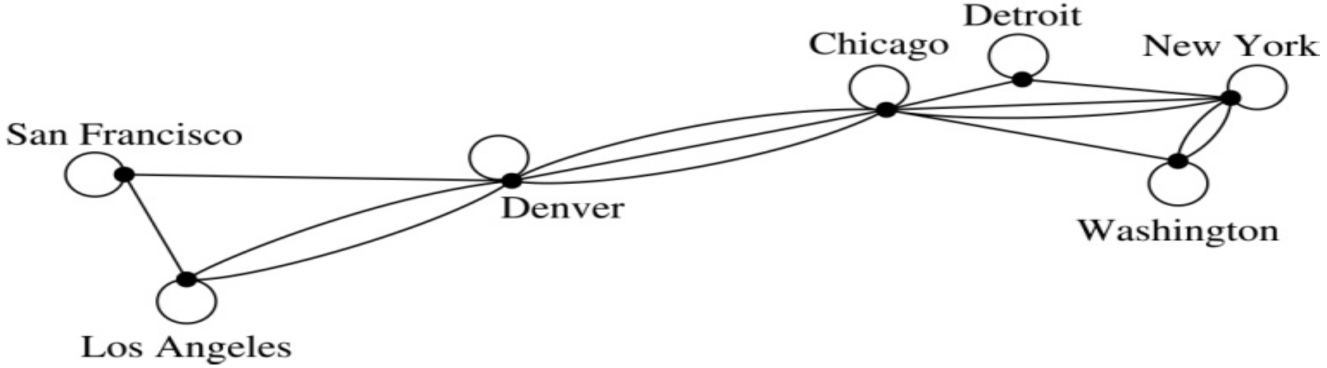


- Multigraph

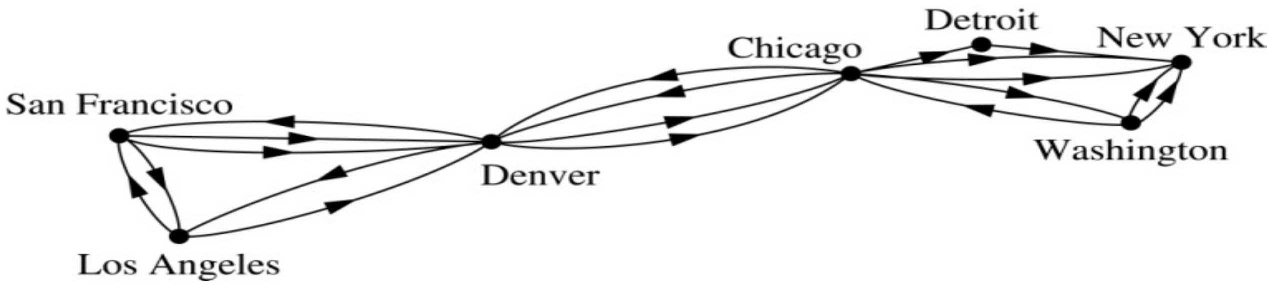


Graph Models: Computer Networks

- Pseudograph



- 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?

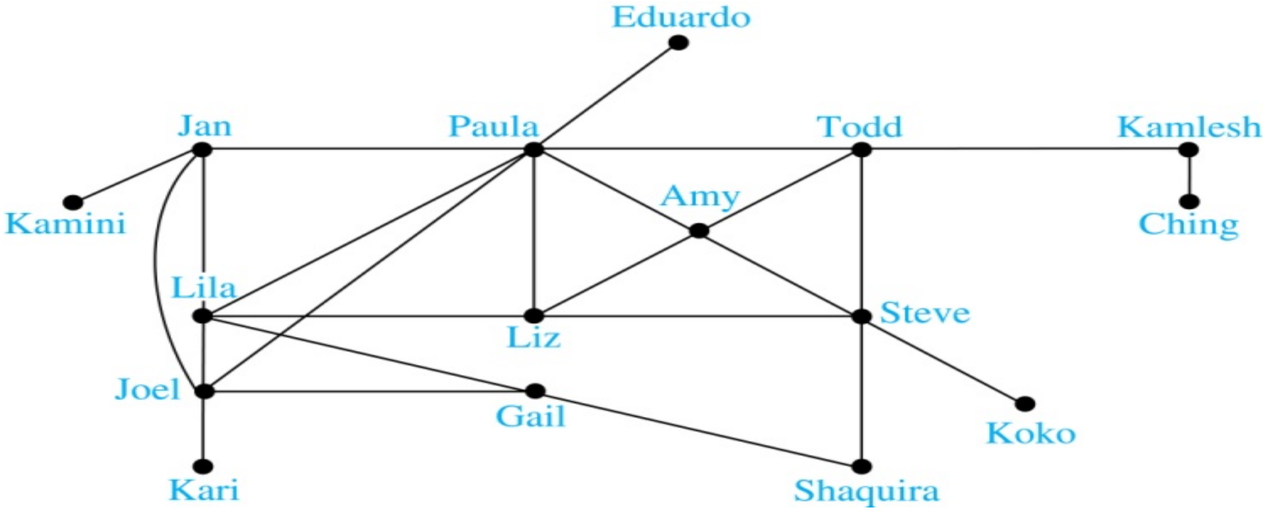
TABLE 1 Graph Terminology.

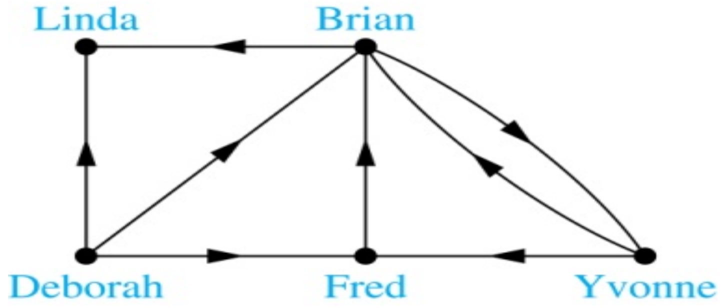
Type	*Edges*	*Multiple Edges Allowed?*	*Loops Allowed?*
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

Other Applications of Graphs

Graph theory models:

- Social networks
- Communications networks
- Information networks
- Software design
- Transportation networks
- Biological networks
- Power-grid networks





CS1382 Discrete Computational Structures

Graph Terminology and Special Types of Graphs

Spring 2019

Richard Matovu

Basic Terminology

- **Adjacent (Neighbors):** If there exists an edge between two vertices e.g., e = (u,v)

- **Incident:** If edge connects two vertices e.g., edge e is incident with vertices u and v

- **Neighborhood of v, *N*(*v*):** The set of all neighbors of a vertex *v* of *G* = (*V*, *E*)

  If *A* is a subset of *V*, we denote by *N*(*A*) the set of all vertices in *G* that are adjacent to at least one vertex in *A*.

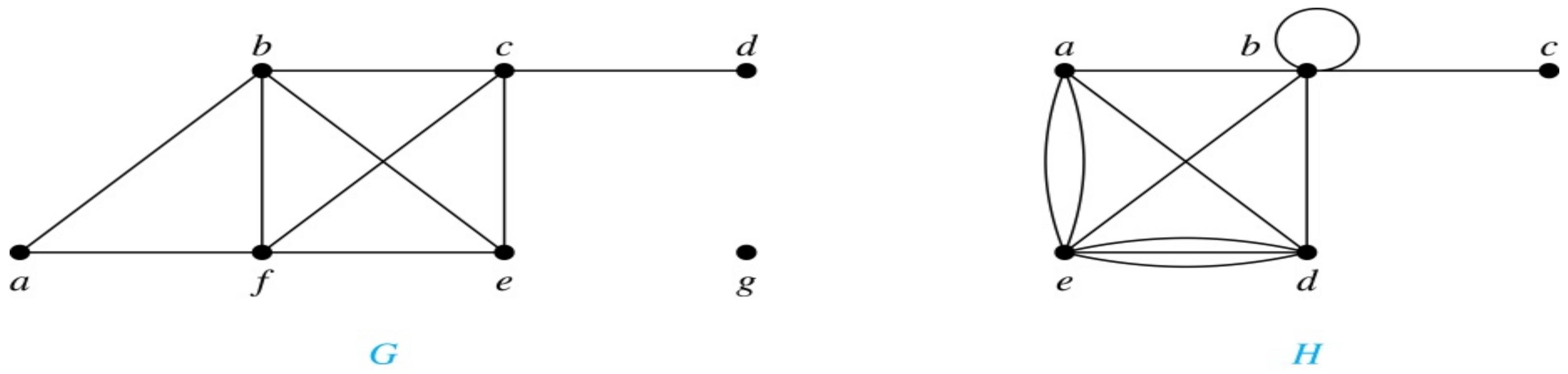
$$N(A) = \bigcup_{v \in A} N(v).$$

- **Degree of a vertex in a undirected graph, deg(*v*):** The number of edges incident with it

    - Except loop at a vertex contributes two to the degree of that vertex.

Degrees and Neighborhoods of Vertices

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



- **Solution**:

  - *G*: deg(a) = 2 , deg(g) = 0

    $N(a) = \{b, f\}$ $N(g) = \varnothing$

  - *H*: deg(a) = 4, deg(b) = 6

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

Degrees of Vertices

Theorem 1 (***Handshaking Theorem***): If $G = (V,E)$ is an undirected graph with m edges, then

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

Proof:

Each edge contributes twice to the degree count of all vertices. Hence, both the left-hand and right-hand sides of this equation equal twice the number of edges. ◄

Degree of Vertices

Theorem 2: An undirected graph $G = (V, E)$ with m edges, has an even number of vertices of odd degree.

Proof: Let V_1 be the vertices of even degree

V_2 be the vertices of odd degree in an undirected graph. Then

even

$$\underbrace{2m}_{\text{even}} = \sum_{v \in V} \deg(v) = \underbrace{\sum_{v \in V_1} \deg(v)}_{\text{must be even}} + \underbrace{\sum_{v \in V_2} \deg(v)}_{\text{must be even}}.$$

must be even

must be even

Handshaking Theorem - Usefulness

- How many edges are there in a graph with 10 vertices of degree six?
  - **Solution**:

    Sum of the degrees is $6 \cdot 10 = 60$, By the handshaking theorem, $2m = 60$.

    So the number of edges $m = 30$.

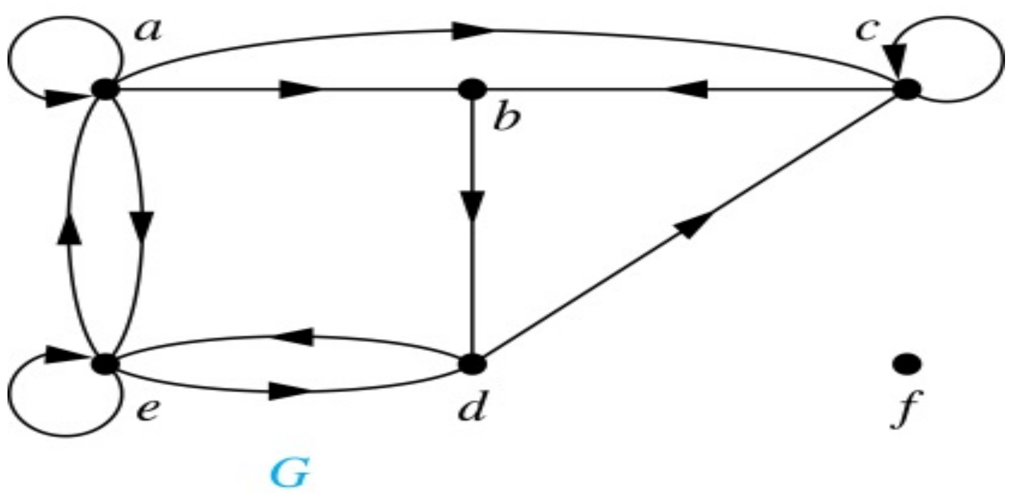
- If a graph has 5 vertices, can each vertex have degree 3?
  - **Solution**:

    This is not possible

    By the handshaking theorem, Sum of the degrees is $3 \cdot 5 = 15$ which is odd.

Directed Graphs

- **In-degree of a vertex v, $\deg^-(v)$:** The number of edges which terminate at *v*.
- **Out-degree of v, $\deg^+(v)$:** 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



$\deg^-(a) = 2$ $\deg^-(c) = 3$ $\deg^-(f) = 0$

$\deg^+(a) = 4$ $\deg^+(c) = 2$ $\deg^+(f) = 0$

Directed Graphs (*continued*)

Theorem 3:

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

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

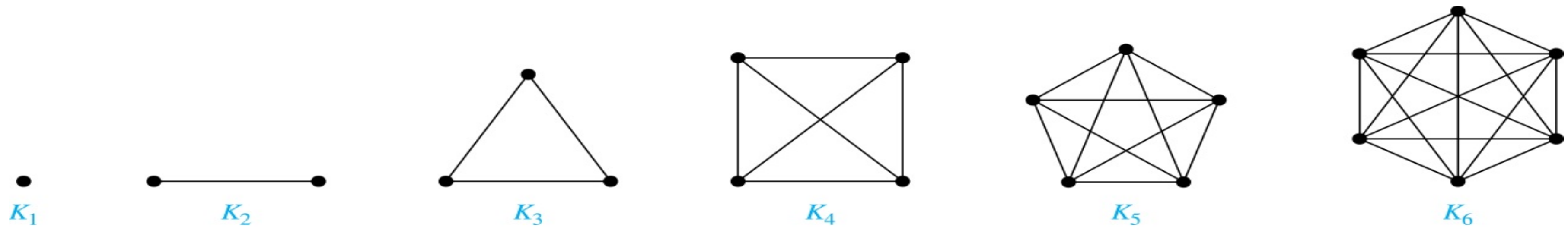
- ***Proof***:
  - The first sum counts the number of outgoing edges over all vertices
  - The second sum counts the number of incoming edges over all vertices.
  - It follows that both sums equal the number of edges in the graph.

◀

Complete Graphs

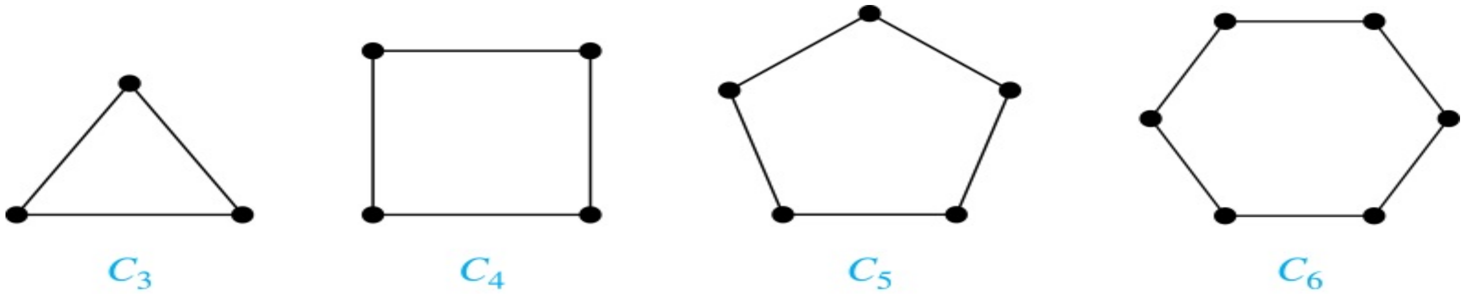
- **Complete Graph on n vertices, K_n**:

  Simple graph that contains exactly one edge between each pair of distinct vertices.

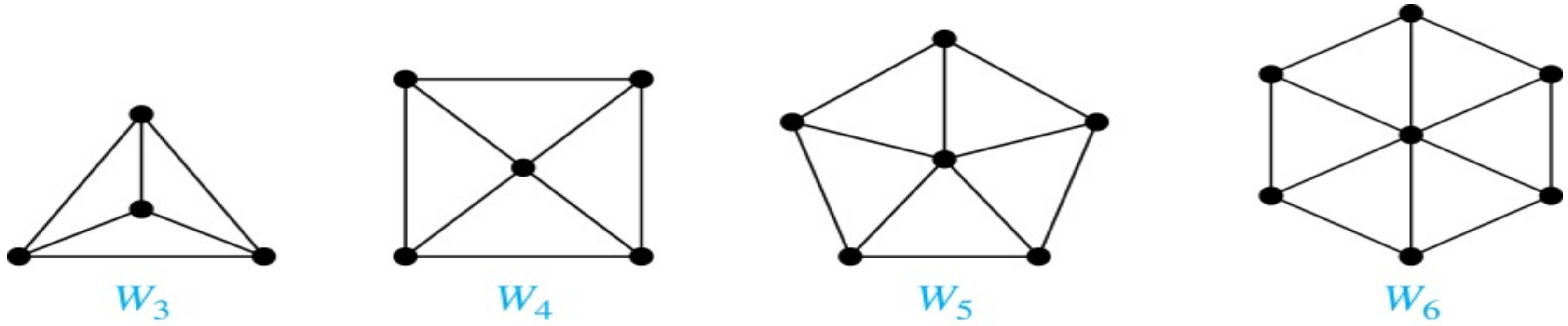


Cycles and Wheels

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

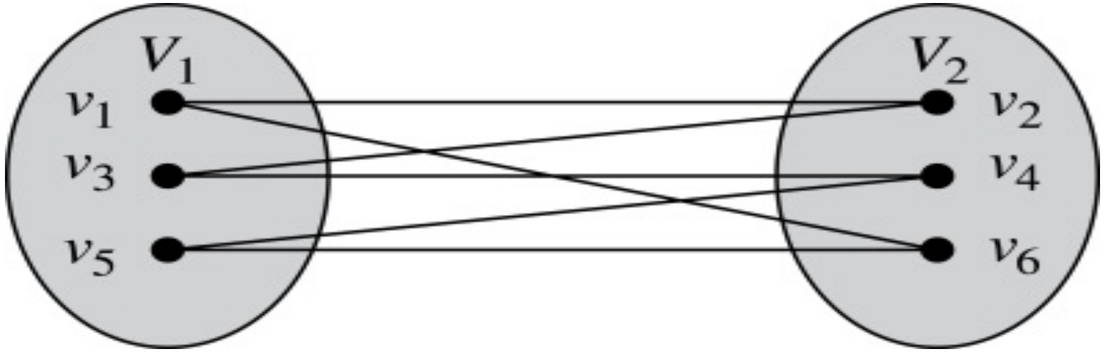


- A ***wheel*** W_n is obtained by adding an additional vertex to a cycle C_n for $n \geq 3$ and connecting this new vertex to each of the n vertices in C_n by new edges.

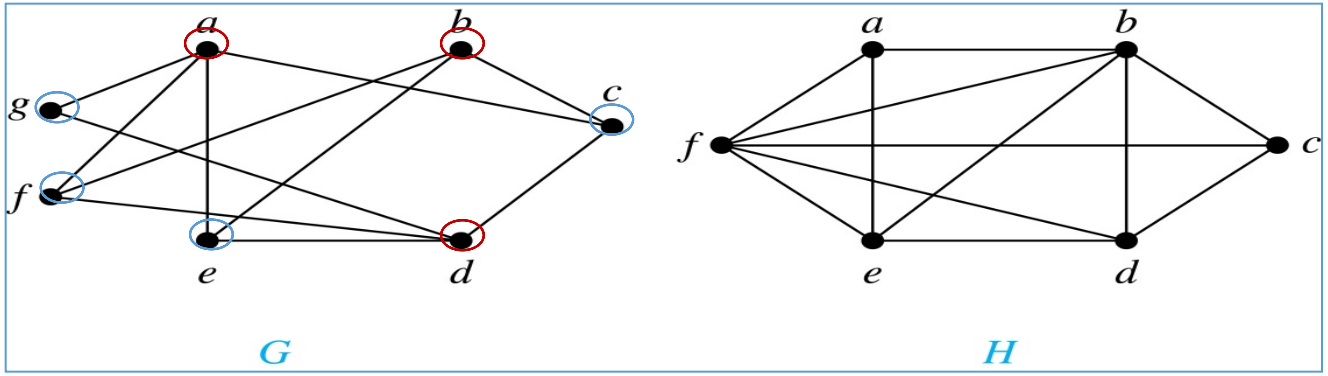


Bipartite Graphs

- A simple graph G is **bipartite** if V can be partitioned into two disjoint subsets V_1 and V_2 such that every edge connects a vertex in V_1 and a vertex in V_2.
- In bipartite graphs, there are no edges which connect two vertices in V_1 or in V_2.



G is bipartite

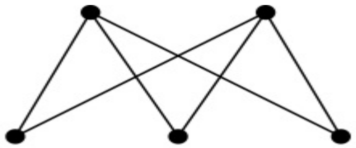


H is not bipartite since if we color *a* red, then the adjacent vertices *f* and *b* must both be blue.

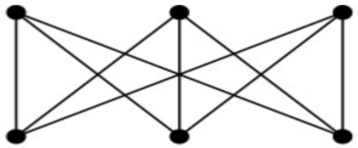
Complete Bipartite Graphs

Complete Bipartite graph $K_{m,n}$:

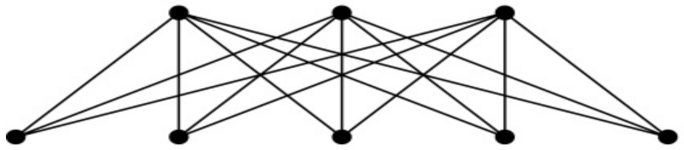
A graph that has its vertex set partitioned into two subsets V_1 of size *m* and V_2 of size *n* such that there is an edge from every vertex in V_1 to every vertex in V_2



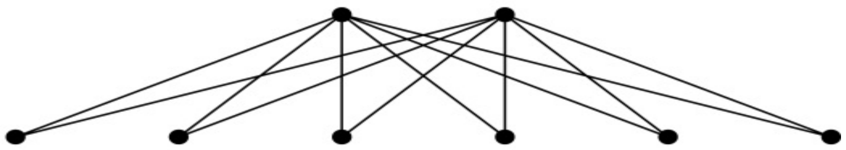
$K_{2,3}$



$K_{3,3}$



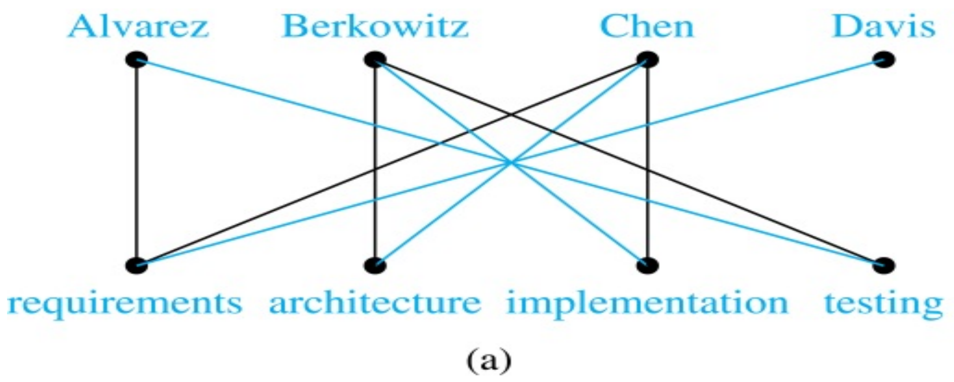
$K_{3,5}$



$K_{2,6}$

Bipartite Graphs and Matchings

- Can model applications that involve matching the elements of one set to elements in another

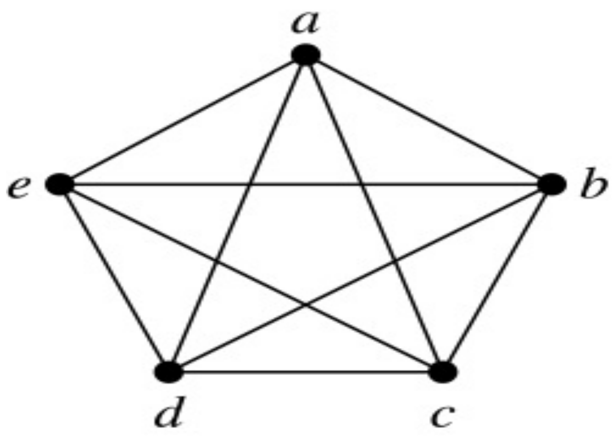


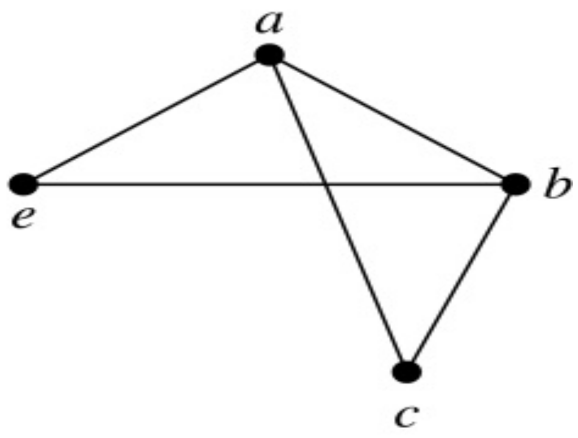
Creating new graphs from other graphs

- **Subgraph of a graph** $G = (V,E)$:

  A graph (W,F) where $W \subset V$ and $F \subset E$ *is*

- A subgraph H of G is a proper subgraph of G if $H \neq G$.

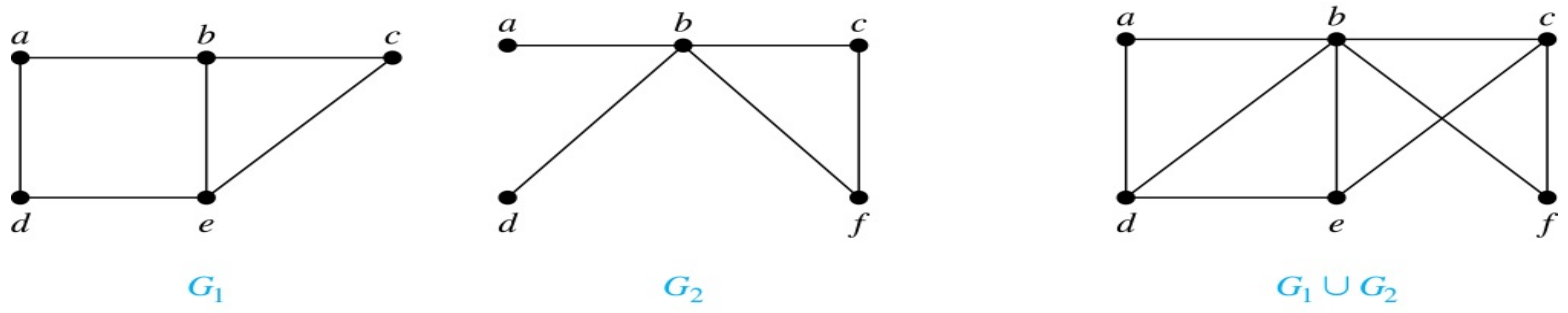




Creating new graphs from other graphs

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:



(a) (b)

CS1382 Discrete Computational Structures

Representing Graphs and Graph Isomorphism

Spring 2019

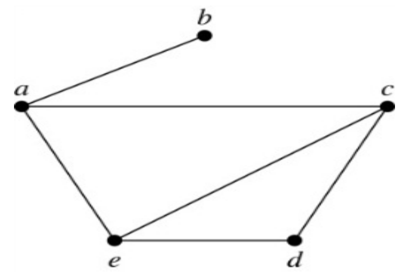
Richard Matovu

TEXAS TECH UNIVERSITY

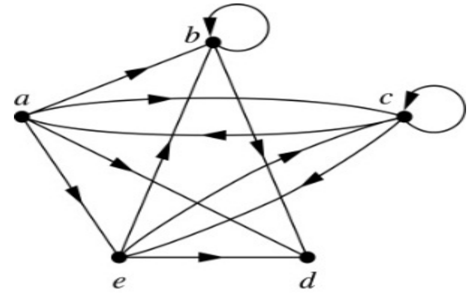
Representing Graphs: Adjacency Lists

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.

Examples:



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



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

Representing Graphs: Adjacency Matrices

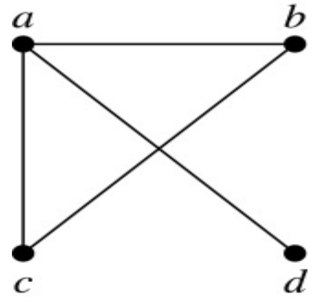
Suppose that $G = (V, E)$ is a simple graph where $|V| = n$.

The ***adjacency matrix,* $\mathbf{A}_G$** of G, with respect to the listing of vertices, is the.

$\mathbf{A}_G = [a_{ij}]$, then

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

Adjacency Matrices: Simple Graphs



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

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

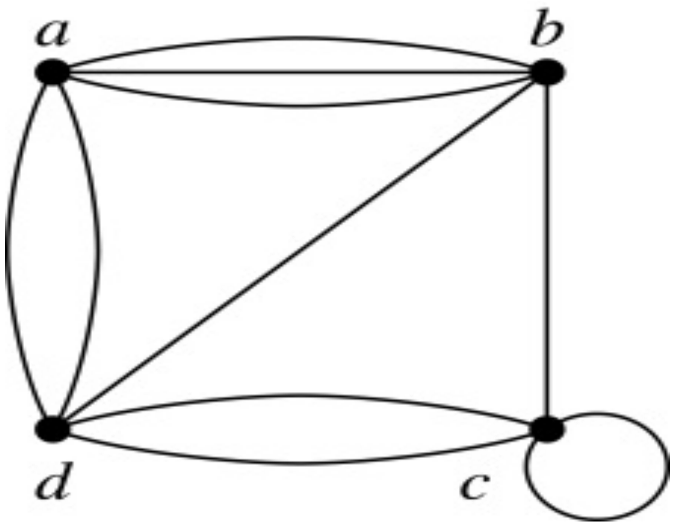
Note:

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

Draw the graph represented by the following:

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

Adjacency Matrices: Pseudograph



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

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

Adjacency Matrices: 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, where $v_1, v_2, \ldots v_n$ is a list of the vertices.

- If $\mathbf{A}_G = [a_{ij}]$, then

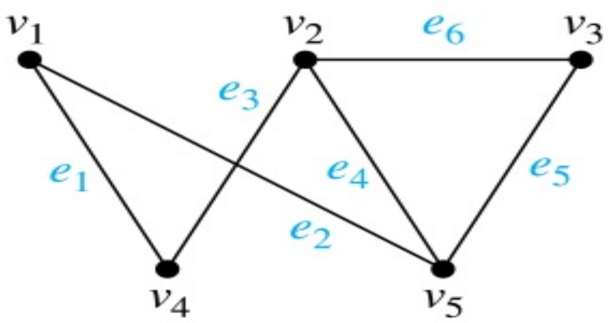
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Representation of Graphs: Incidence Matrices

Let $G = (V, E)$ be an undirected graph with vertices where $v_1, v_2, \ldots v_n$ and edges $e_1, e_2, \ldots e_m$.

If $n \times m$ matrix $\mathbf{M} = [m_{ij}]$ where

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

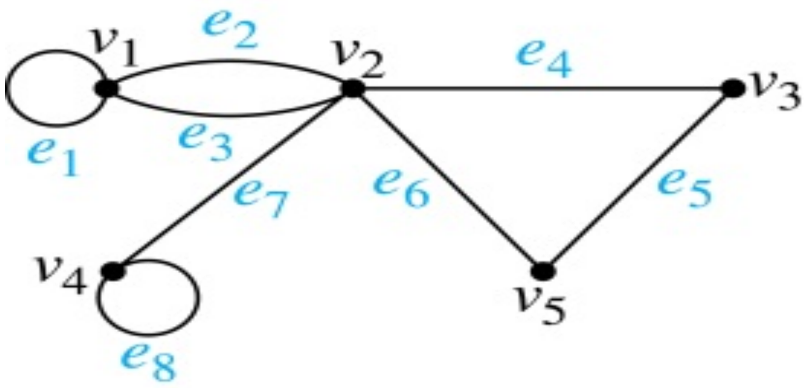


$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

The rows going from top to bottom represent v_1 through v_5 and the columns going from left to right represent e_1 through e_6.

Incidence Matrices

- What is the incidence matrix of the pseudograph below:



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

The rows going from top to bottom represent v_1 through v_5 and the columns going from left to right represent e_1 through e_8.

Isomorphism of Graphs

$G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are ***isomorphic***

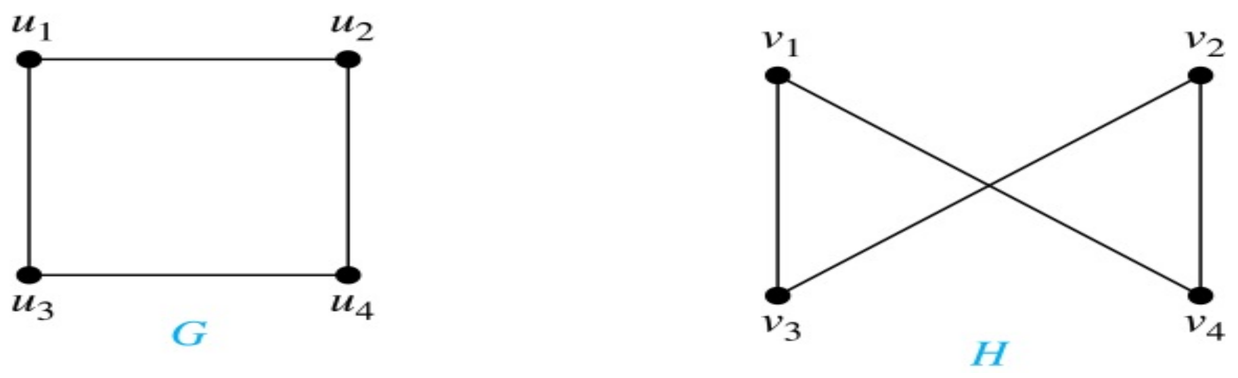
- If there is a one-to-one and onto function f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2 , for all a and b in V_1 .

  Such a function f is called an ***isomorphism***

- Otherwise, there are ***nonisomorphic***

Isomorphism of Graphs

- Show that the graphs G =(V, E) and H = (W, F) are isomorphic.



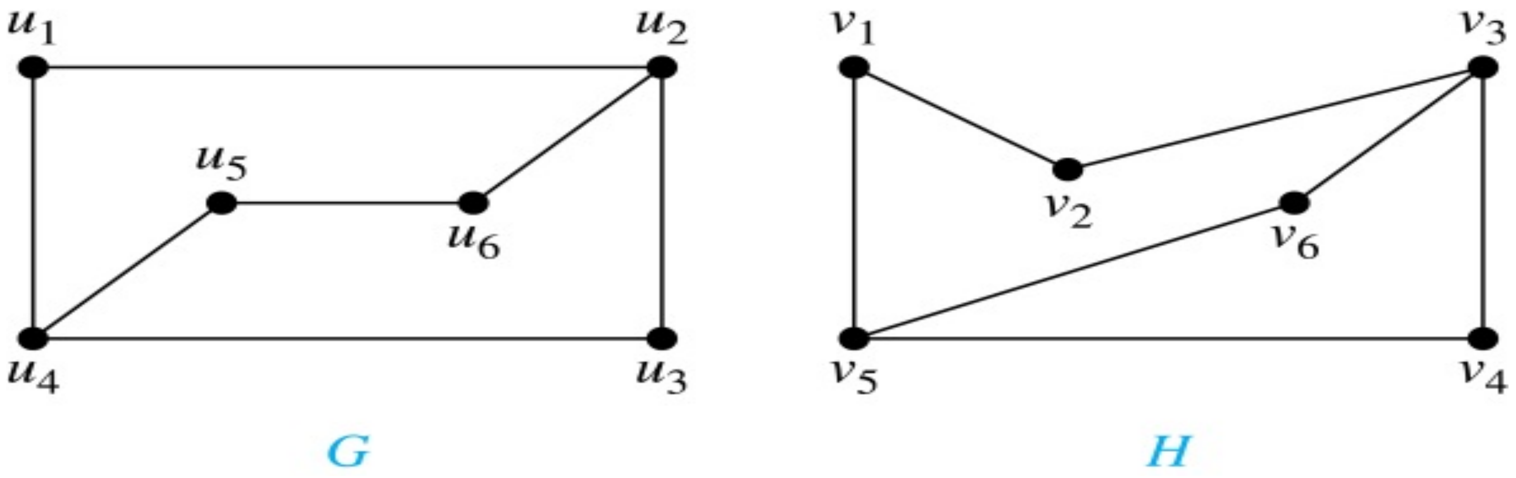
- **Solution**:

  The function f with $f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, and $f(u_4) = v_2$

  is a one-to-one correspondence between V and W.

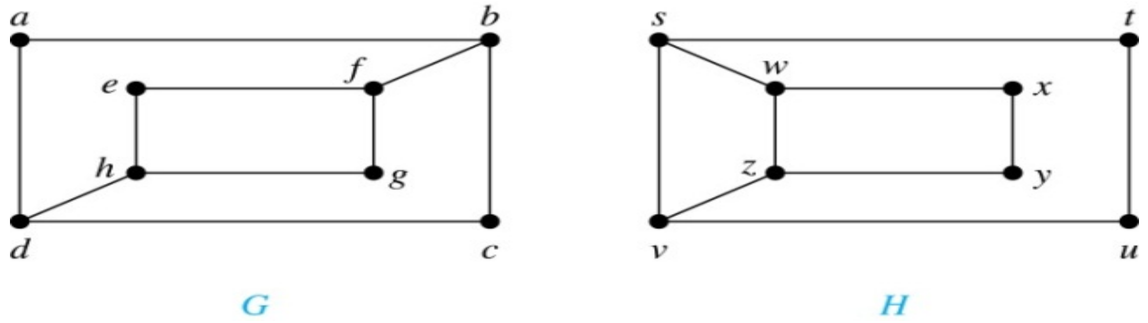
Isomorphism of Graphs

- Difficult to determine whether two simple graphs are isomorphic using brute force.
- Best algorithms have exponential worst case complexity in terms of the number of vertices of the graphs.
- Graph invariant



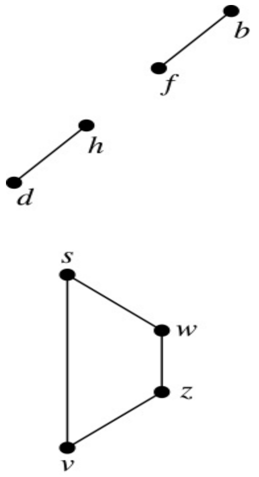
Isomorphism of Graphs

Example: Determine whether these two graphs are isomorphic.



Solution: Both graphs have eight vertices and ten edges.
They also both have four vertices of degree two and four of degree three.

However, *G* and *H* are not isomorphic.



CS1382 Discrete Computational Structures

Connectivity

Spring 2019

Richard Matovu

TEXAS TECH UNIVERSITY

Paths

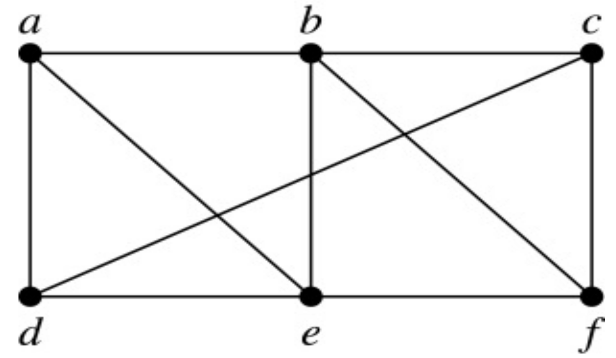
Let n be a nonnegative integer and G an undirected graph.

Path of length n from u to v in G is a sequence of n edges $e_1, \ldots, e_n$ of G for which 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.

- **Circuit:** Path that begins and ends at the same vertex ($u = v$) and has length greater than zero.
- **Simple Path / Circuit:** If it does not contain the same edge more than once.

Paths: Example

Example: In the simple graph here:



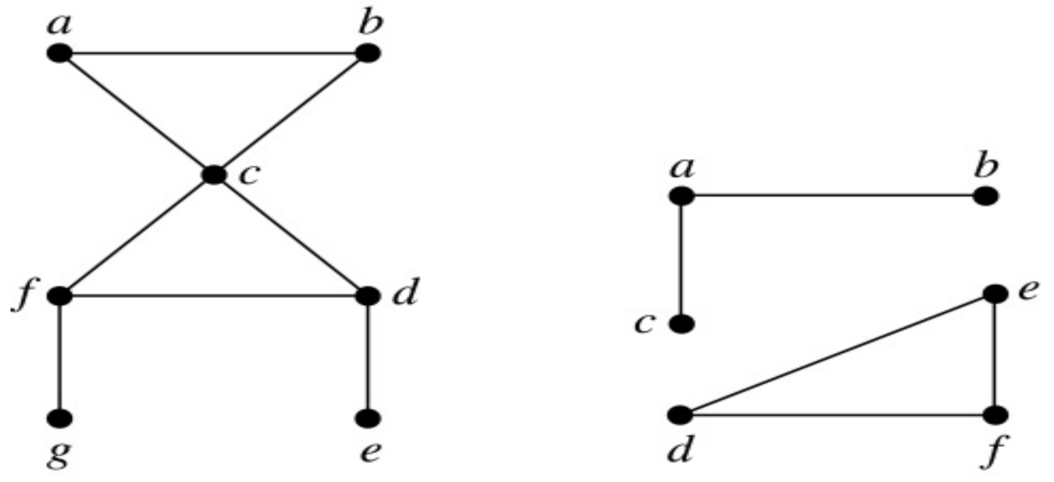
- *a*, *d*, *c*, *f*, *e* is a simple path of length 4.
- *d*, *e*, *c*, *a* is not a path because *e* is not connected to *c*.
- *b*, *c*, *f*, *e*, *b* is a circuit of length 4.
- *a*, *b*, *e*, *d*, *a*, *b* is a path of length 5, but it is not a simple path.

Connectedness in Undirected Graphs

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.

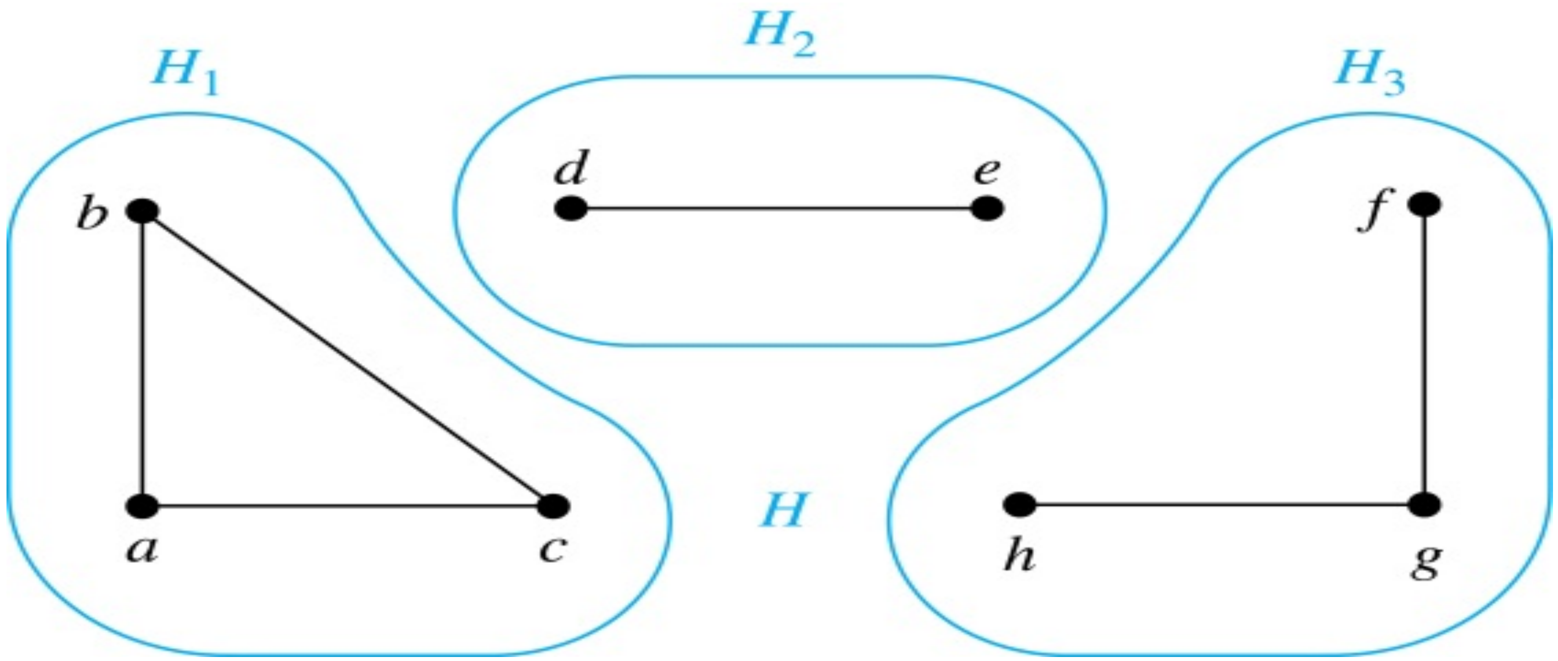


G_1 G_2

Connected Components

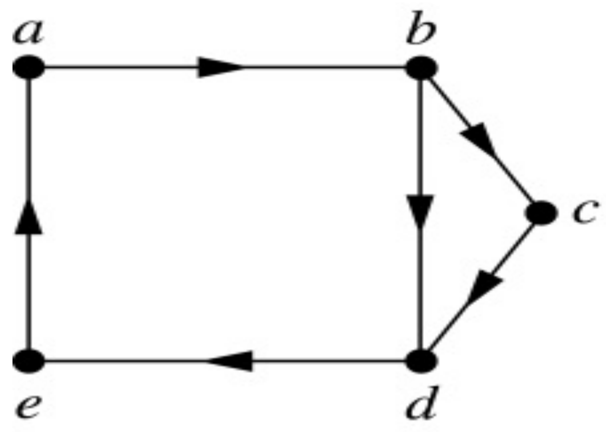
A connected subgraph of G that is not a proper subgraph of another connected subgraph of G.

A graph G that is not connected has two or more connected components that are disjoint and have G as their union.

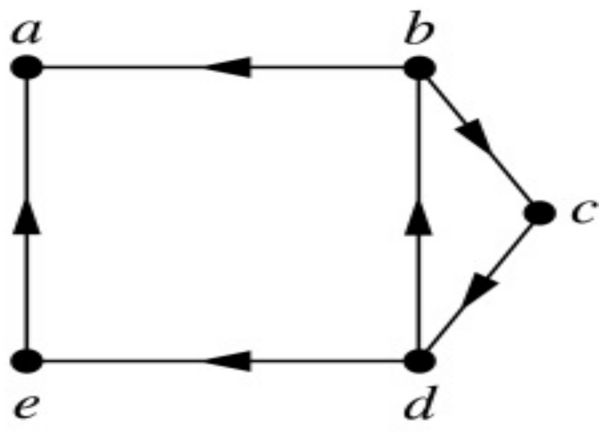


Connectedness in Directed Graphs

- **Strongly connected** if there is a path from *a* to *b* and a path from *b* to *a* whenever *a* and *b* are vertices in the graph.
- **Weakly connected** if there is a path between every two vertices in the underlying undirected graph
  - Ignoring the directions of the edges of the directed graph.

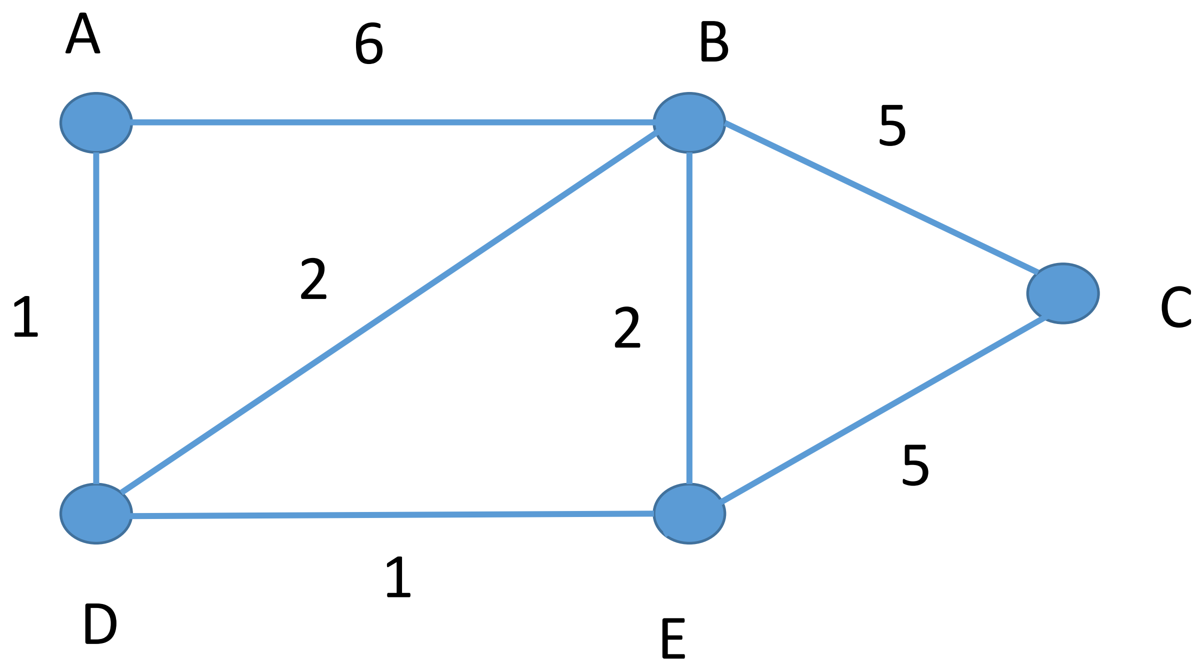


G



H

Dijkstra's Shortest Path Algorithm



Vertex	Shortest Distance from A	Previous Vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E, B, C]

Questions?

Thank You!