DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room903, Nanshan iPark A7 Building
Email: wangqi@sustc.edu.cn

Bipartite Graphs and Matchings

- **Definition** 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.

  An equivalent definition of a bipartite graph is a graph where it is possible to color the vertices red or blue so that no two adjacent vertices are of the same color.

Bipartite Graphs and Matchings

- **Definition** 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.

  An equivalent definition of a bipartite graph is a graph where it is possible to color the vertices red or blue so that no two adjacent vertices are of the same color.

  Matching the elements of one set to elements in another. A *matching* is a subset of E s.t. no two edges are incident with the same vertex.

Bipartite Graphs and Matchings

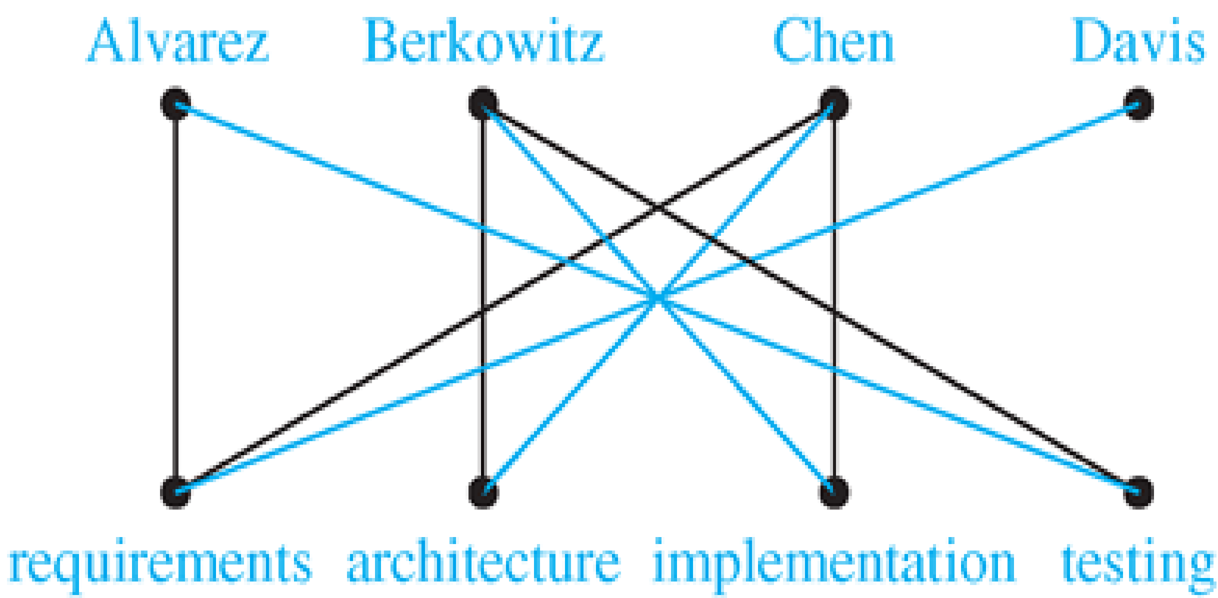
- *Matching* the elements of one set to elements in another. A *matching* is a subset of E s.t. no two edges are incident with the same vertex.

Bipartite Graphs and Matchings

- *Matching* the elements of one set to elements in another. A *matching* is a subset of E s.t. no two edges are incident with the same vertex.
  Job assignments: vertices represent the jobs and the employees, edges link employees with those jobs they have been trained to do. A common goal is to match jobs to employees so that the most jobs are done.

Bipartite Graphs and Matchings

- *Matching* the elements of one set to elements in another. A *matching* is a subset of E s.t. no two edges are incident with the same vertex.
  Job assignments: vertices represent the jobs and the employees, edges link employees with those jobs they have been trained to do. A common goal is to match jobs to employees so that the most jobs are done.



Bipartite Graphs and Matchings

- A *maximum matching* is a matching with the largest number of edges.

Bipartite Graphs and Matchings

- A *maximum matching* is a matching with the largest number of edges.
  A matching M in a bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) is a *complete matching from V_1 to V_2* if every vertex in V_1 is the endpoint of an edge in the matching, or equivalently, if $|M| = |V_1|$.

Bipartite Graphs and Matchings

- A *maximum matching* is a matching with the largest number of edges.

  A matching M in a bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) is a *complete matching from V_1 to V_2* if every vertex in V_1 is the endpoint of an edge in the matching, or equivalently, if $|M| = |V_1|$.

  Theorem (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "only if" $\rightarrow$

Suppose that there is a matching M from V_1 to V_2. Consider an arbitrary subset $A \subseteq V_1$.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "only if" $\rightarrow$

Suppose that there is a matching M from V_1 to V_2. Consider an arbitrary subset $A \subseteq V_1$.

Then, for every vertex $v \in A$, there is an edge in M connecting v to a vertex in V_2. Thus, there are at least as many vertices in V_2 that are neighbors of vertices in V_1 as there are vertices in V_1.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "only if" $\rightarrow$

Suppose that there is a matching M from V_1 to V_2. Consider an arbitrary subset $A \subseteq V_1$.

Then, for every vertex $v \in A$, there is an edge in M connecting v to a vertex in V_2. Thus, there are at least as many vertices in V_2 that are neighbors of vertices in V_1 as there are vertices in V_1.

Hence, $|N(A)| \geq |A|$.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Use **strong induction** to prove it.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Use **strong induction** to prove it.

Basic step: $|V_1| = 1$

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Use **strong induction** to prove it.

Basic step: $|V_1| = 1$

Inductive hypothesis: Let k be a positive integer. If $G = (V, E)$ is a bipartite graph with bipartition (V_1, V_2), and $|V_1| = j \leq k$, then there is a complete mathching M from V_1 to V_2 whenever the condition that $|N(A)| \geq |A|$ for all $A \subseteq V_1$ is met.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Use **strong induction** to prove it.

Basic step: $|V_1| = 1$

Inductive hypothesis: Let k be a positive integer. If $G = (V, E)$ is a bipartite graph with bipartition (V_1, V_2), and $|V_1| = j \leq k$, then there is a complete mathching M from V_1 to V_2 whenever the condition that $|N(A)| \geq |A|$ for all $A \subseteq V_1$ is met.

Inductive step: suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Inductive step: suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Inductive step: suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$.

Case (i): For all integers j with $1 \leq j \leq k$, the vertices in every set of j elements from W_1 are adjacent to at least $j + 1$ elements of W_2

Case (ii): For some integer j with $1 \leq j \leq k$, there is a subset W_1' of j vertices such that there are exactly j neighbors of these vertices in W_2

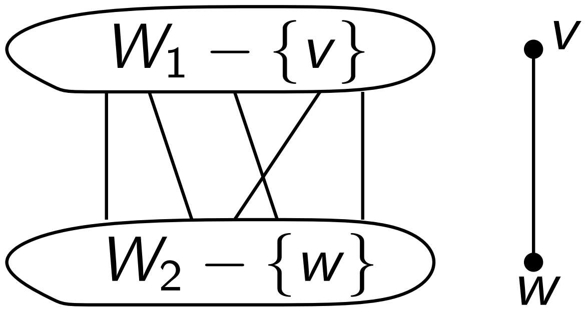
Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Inductive step: suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$.

Case (i):



Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Inductive step: suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$.

Case (ii): For some integer j with $1 \leq j \leq k$, there is a subset W_1' of j vertices such that there are exactly j neighbors of these vertices in W_2

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Inductive step: suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$.

Case (ii): For some integer j with $1 \leq j \leq k$, there is a subset W_1' of j vertices such that there are exactly j neighbors of these vertices in W_2

Let W_2' be the set of these neighbors. Then by i.h., there is a complete matching from W_1' to W_2'. Now consider the graph $K = (W_1 - W_1', W_2 - W_2')$. We will show that the condition $|N(A)| \geq |A|$ is met for all subsets A of $W_1 - W_1'$.

Proof of Hall's Theorem

- **Theorem** (Hall's Marriage Theorem) The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof. "if" $\leftarrow$

Inductive step: suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$.

Case (ii): For some integer j with $1 \leq j \leq k$, there is a subset W_1' of j vertices such that there are exactly j neighbors of these vertices in W_2

Let W_2' be the set of these neighbors. Then by i.h., there is a complete matching from W_1' to W_2'. Now consider the graph $K = (W_1 - W_1', W_2 - W_2')$. We will show that the condition $|N(A)| \geq |A|$ is met for all subsets A of $W_1 - W_1'$.

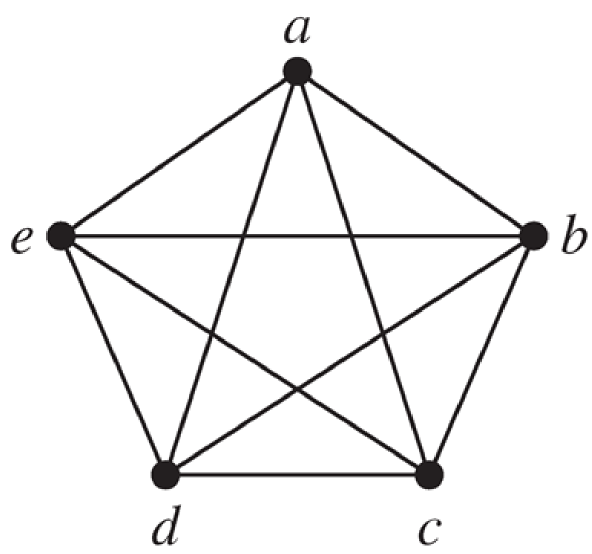
If not, there is a subset B of t vertices with $1 \leq t \leq k + 1 - j$ s.t. $|N(B)| < t$.

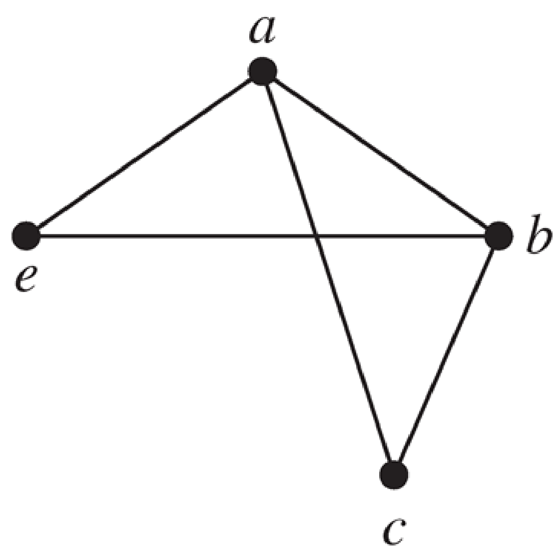
Subgraphs

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

Subgraphs

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



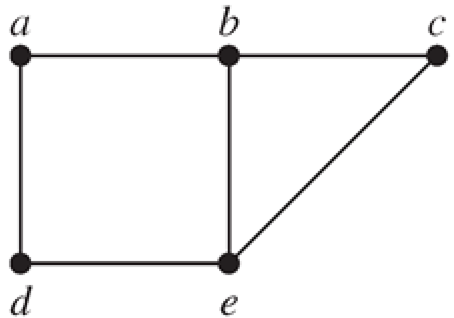


Union of Graphs

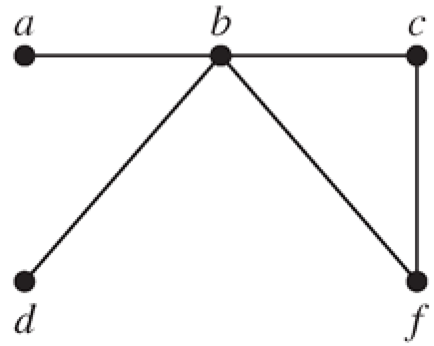
- **Definition** The *union of two simple graphs* $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$, denoted by $G_1 \cup G_2$.

Union of Graphs

- **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 $E_1 \cup E_2$, denoted by $G_1 \cup G_2$.



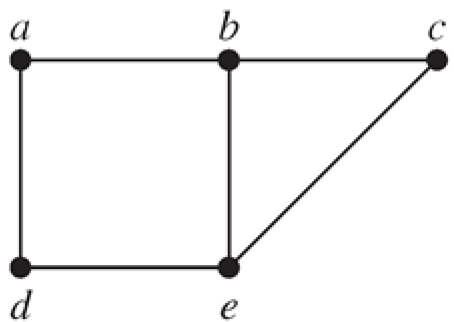
G_1



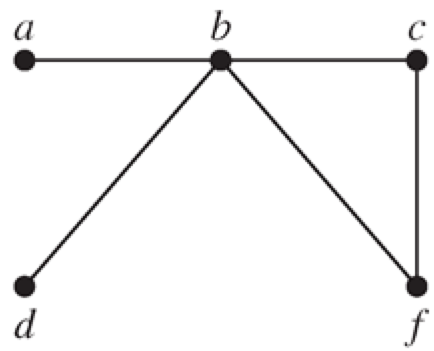
G_2

Union of Graphs

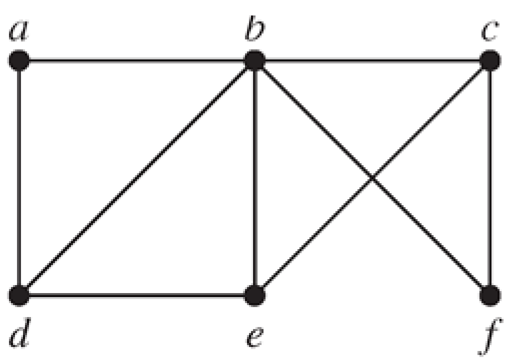
- **Definition** The *union of two simple graphs* $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$, denoted by $G_1 \cup G_2$.



G_1



G_2



$G_1 \cup G_2$

Representation of Graphs

- To represent a graph, we may use *adjacency lists*, *adjacency matrices*, and *incidence matrices*.

Representation of Graphs

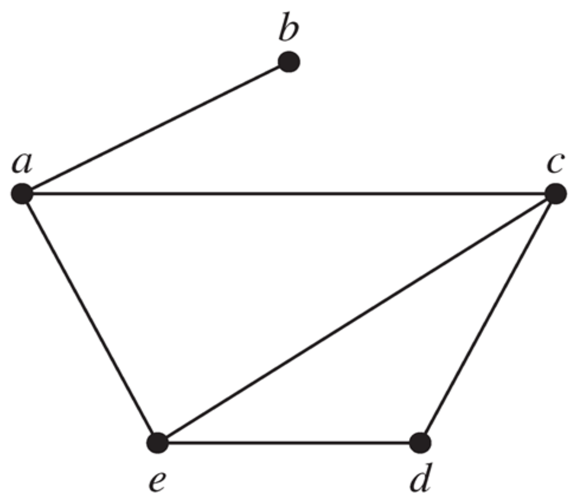
- To represent a graph, we may use *adjacency lists*, *adjacency matrices*, and *incidence matrices*.

  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.

Representation of Graphs

- To represent a graph, we may use *adjacency lists*, *adjacency matrices*, and *incidence matrices*.

  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.



Representation of Graphs

- To represent a graph, we may use *adjacency lists*, *adjacency matrices*, and *incidence matrices*.

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.

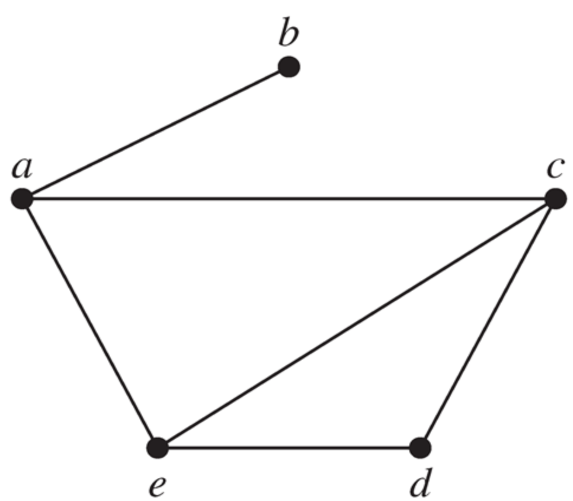
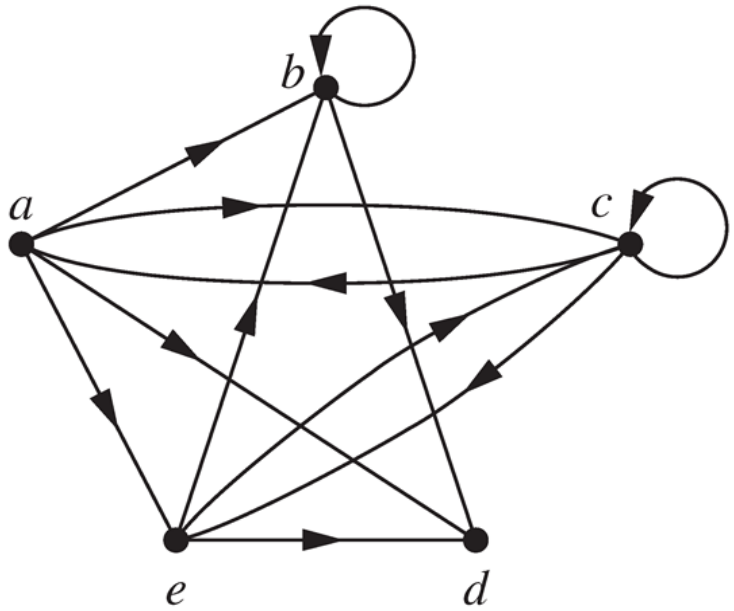


TABLE 1 An Adjacency List for a Simple Graph.

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

Representation of Graphs

- **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.



Representation of Graphs

- **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.

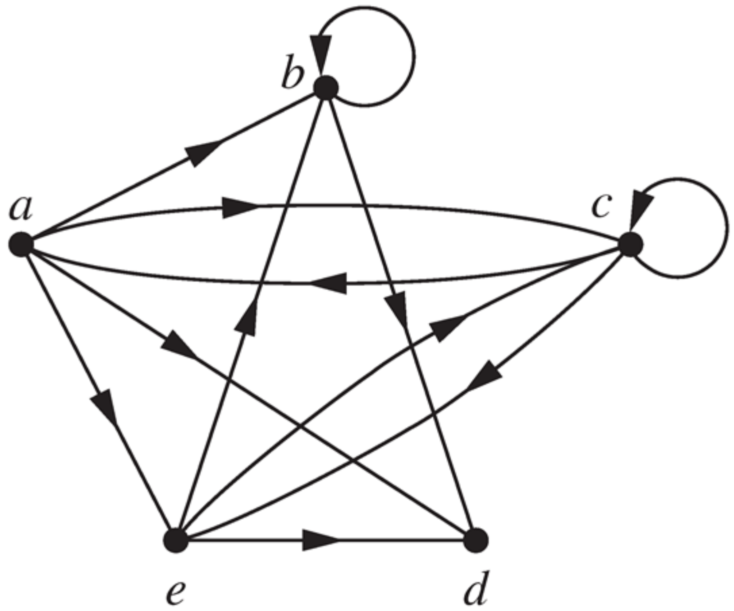


TABLE 2 An Adjacency List for a Directed Graph.

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

Adjacency Matrices

- **Definition** Suppose that $G = (V, E)$ is a simple graph with $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, is the $n \times n$ zero-one matrix with 1 as its (i, j)-th entry when v_i and v_j are adjacent, and 0 as its (i, j)-th entry when they are not adjacent.

Adjacency Matrices

- **Definition** Suppose that $G = (V, E)$ is a simple graph with $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, is the $n \times n$ zero-one matrix with 1 as its (i, j)-th entry when v_i and v_j are adjacent, and 0 as its (i, j)-th entry when they are not adjacent.

$$\mathbf{A}_G = [a_{ij}]_{n\times n}, \text{ where}$$

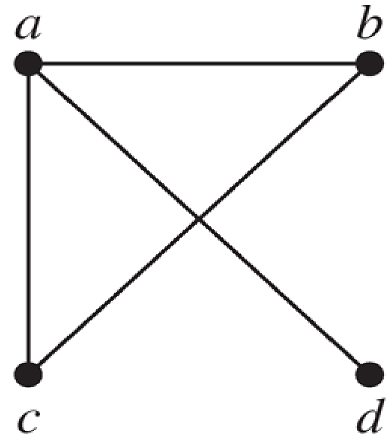
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Adjacency Matrices

- **Definition** Suppose that $G = (V, E)$ is a simple graph with $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, is the $n \times n$ zero-one matrix with 1 as its (i, j)-th entry when v_i and v_j are adjacent, and 0 as its (i, j)-th entry when they are not adjacent.

$$\mathbf{A}_G = [a_{ij}]_{n\times n}, \text{ where}$$

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

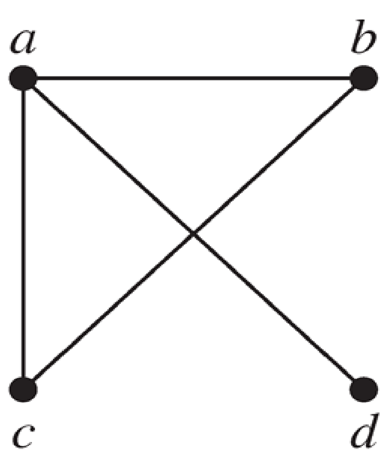


Adjacency Matrices

- **Definition** Suppose that $G = (V, E)$ is a simple graph with $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, is the $n \times n$ zero-one matrix with 1 as its (i, j)-th entry when v_i and v_j are adjacent, and 0 as its (i, j)-th entry when they are not adjacent.

$$\mathbf{A}_G = [a_{ij}]_{n\times n}, \text{ where}$$

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



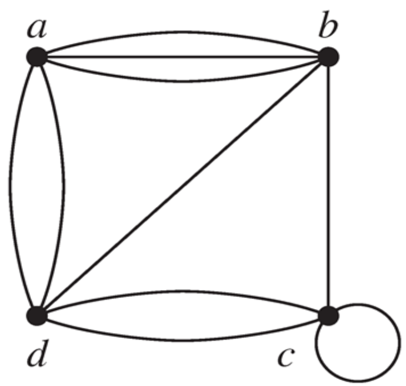
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Adjacency Matrices

- Adjacency matrices can also be used to represent graphs with loops and multiple edges. The matrix is no longer a zero-one matrix.

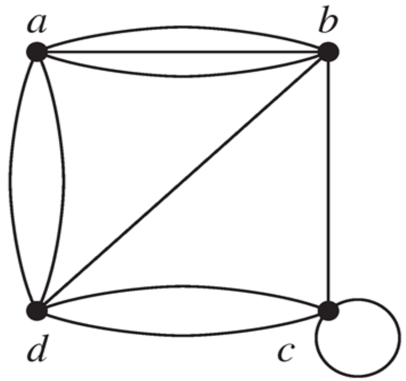
Adjacency Matrices

- Adjacency matrices can also be used to represent graphs with loops and multiple edges. The matrix is no longer a zero-one matrix.



Adjacency Matrices

- Adjacency matrices can also be used to represent graphs with loops and multiple edges. The matrix is no longer a zero-one matrix.



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

Incidence Matrices

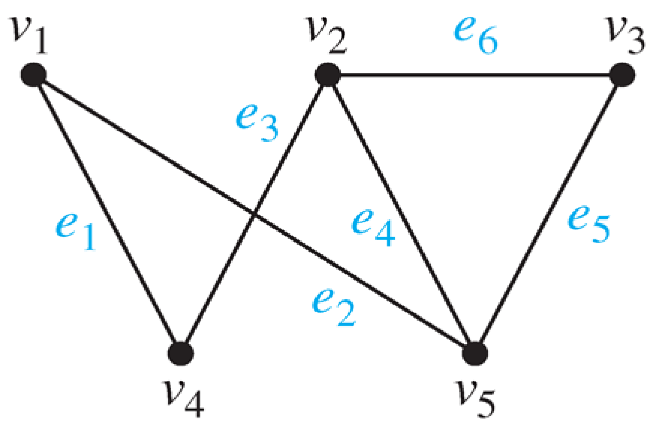
- **Definition** Let $G = (V, E)$ be an undirected graph with vertices $v_1, v_2, \ldots, v_n$ and edges $e_1, e_2, \ldots, e_m$. The *incidence matrix* with respect to the ordering of V and E is the $n \times m$ matrix $\mathbf{M} = [m_{ij}]_{n\times m}$, where

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

Incidence Matrices

- **Definition** Let $G = (V, E)$ be an undirected graph with vertices $v_1, v_2, \ldots, v_n$ 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 $\mathbf{M} = [m_{ij}]_{n\times m}$, where

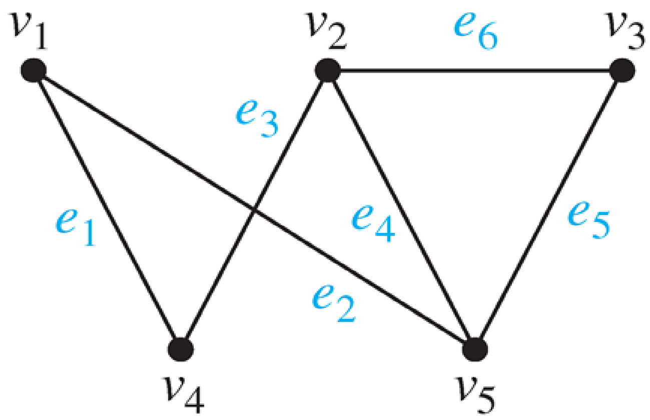
$$m_{ij} = \begin{cases} 1 & \text{if edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$



Incidence Matrices

- **Definition** Let $G = (V, E)$ be an undirected graph with vertices $v_1, v_2, \ldots, v_n$ 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 $\mathbf{M} = [m_{ij}]_{n\times m}$, where

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

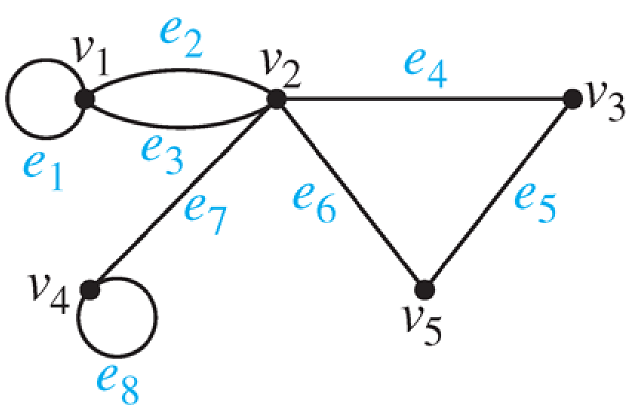


$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Incidence Matrices

- **Definition** Let $G = (V, E)$ be an undirected graph with vertices $v_1, v_2, \ldots, v_n$ 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 $\mathbf{M} = [m_{ij}]_{n \times m}$, where

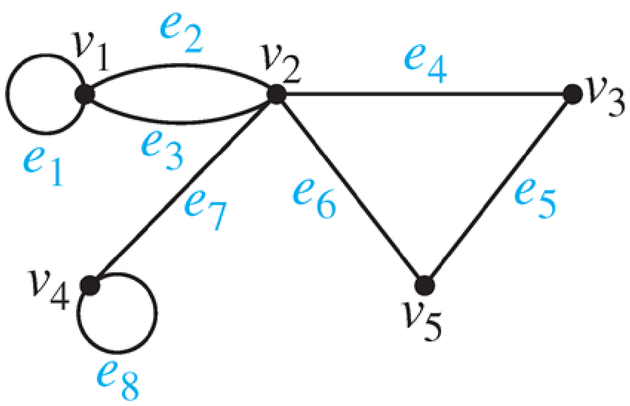
$$m_{ij} = \begin{cases} 1 & \text{if edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$



Incidence Matrices

- **Definition** Let $G = (V, E)$ be an undirected graph with vertices $v_1, v_2, \ldots, v_n$ 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 $\mathbf{M} = [m_{ij}]_{n\times m}$, where

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



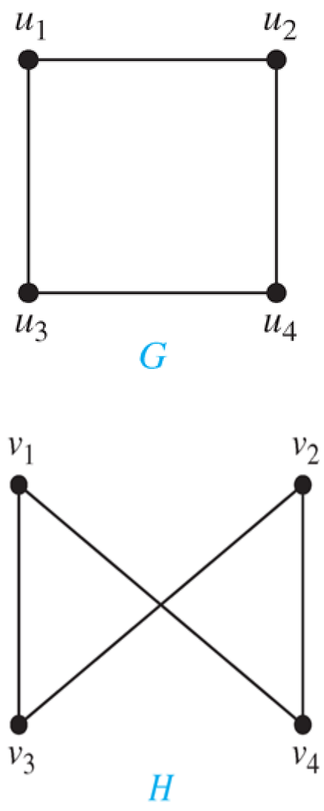
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Isomorphism of Graphs

- **Definition** The simple 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 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 is called an *isomorphism*.

Isomorphism of Graphs

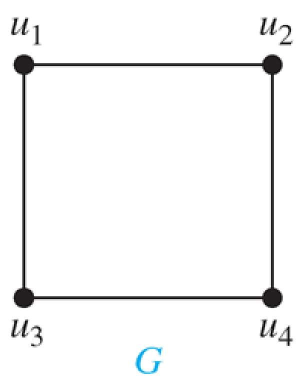
- **Definition** The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a one-to-one and onto function from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2, for all a and b in V_1. Such a function is called an *isomorphism*.

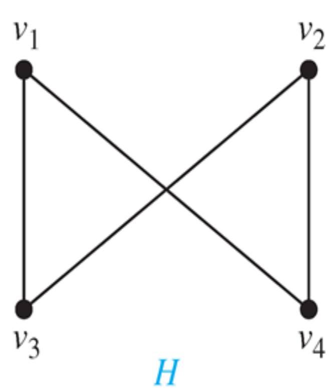


Are the two graphs isomorphic?

Isomorphism of Graphs

- **Definition** The simple 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 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 is called an *isomorphism*.





Are the two graphs isomorphic?

Define a one-to-one correspondence:
$f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, and $f(u_4) = v_2$

Isomorphism of Graphs

- It is usually difficult to determine whether two simple graphs are isomorphic using brute force since there are $n!$ possible one-to-one correspondences.

Isomorphism of Graphs

- It is usually difficult to determine whether two simple graphs are isomorphic using brute force since there are $n!$ possible one-to-one correspondences.

- Sometimes it is not difficult to show that two graphs are not isomorphic. We can achieve this by checking some *graph invariants*.

Isomorphism of Graphs

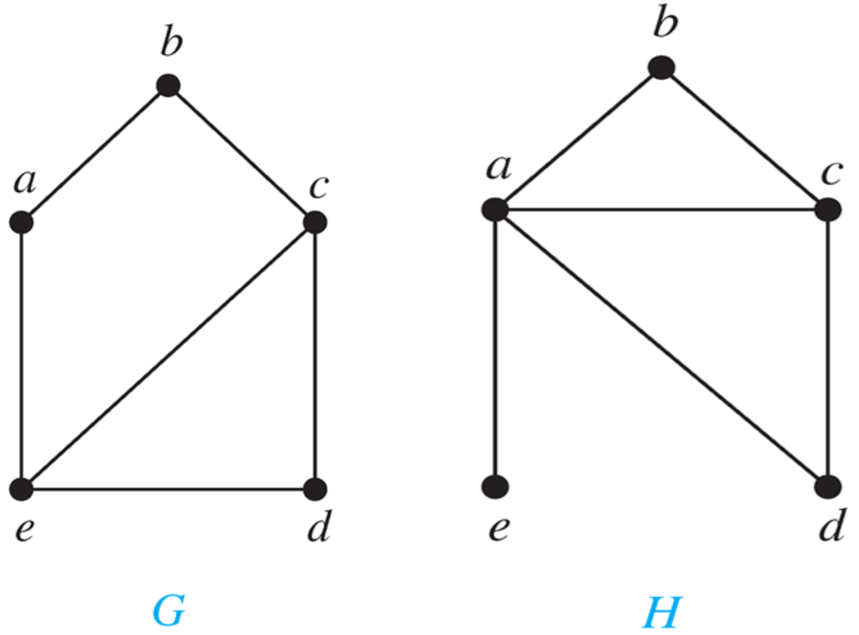
- It is usually difficult to determine whether two simple graphs are isomorphic using brute force since there are $n!$ possible one-to-one correspondences.

- Sometimes it is not difficult to show that two graphs are not isomorphic. We can achieve this by checking some *graph invariants*.

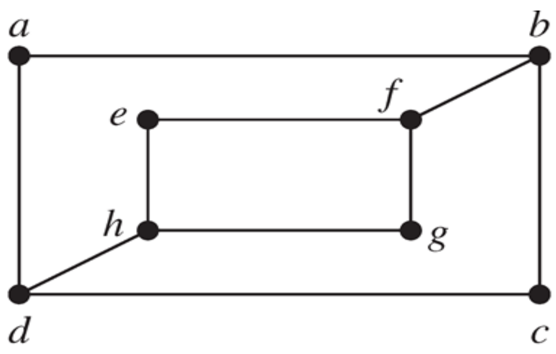
- Useful graph invariants include the number of vertices, number of edges, degree sequence, etc.

Isomorphism of Graphs

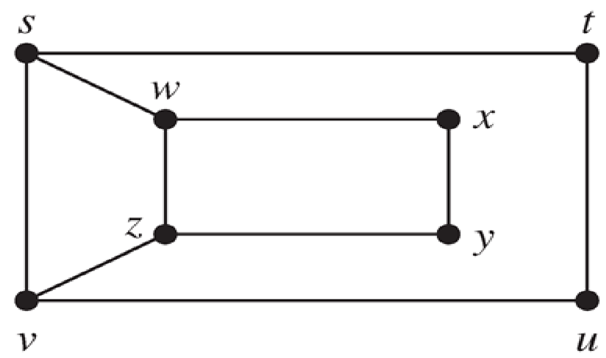
- **Example** Determine whether these two graphs are isomorphic.



- **Example** Determine whether these two graphs are isomorphic.

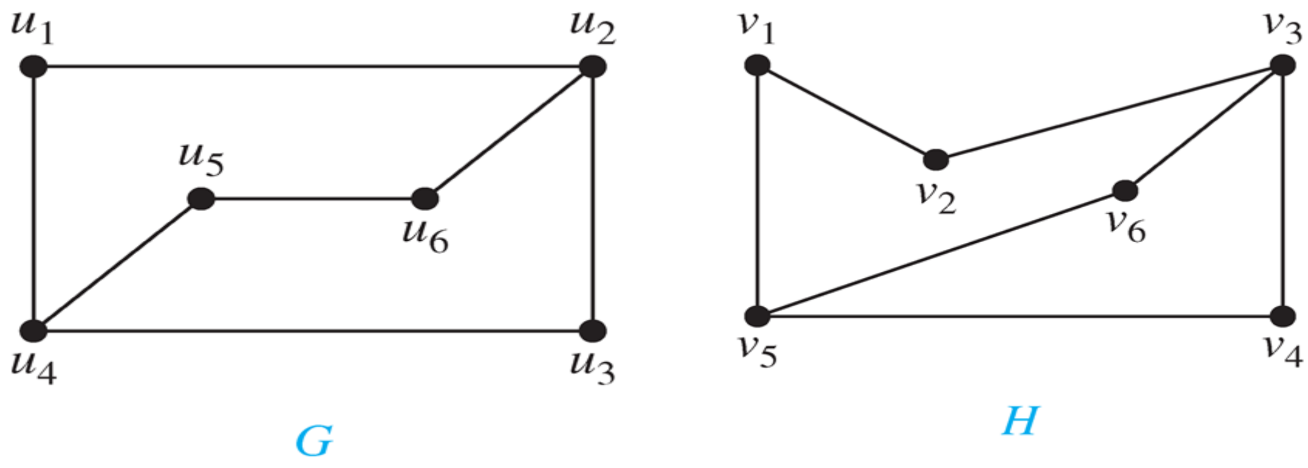


G



H

- **Example** Determine whether these two graphs are isomorphic.



Path

- **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 $e_1, e_2, \ldots, e_n$ 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 the endpoints x_{i-1} and x_i for $i = 1, \ldots, n$. The path is a *circuit* if it begins and ends at the same vertex, i.e., if $u = v$ and has length greater than zero. A path or circuit is *simple* if it does not contain the same edge (vertex) more than once.

Path

- **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 $e_1, e_2, \ldots, e_n$ 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 the endpoints x_{i-1} and x_i for $i = 1, \ldots, n$. The path is a *circuit* if it begins and ends at the same vertex, i.e., if $u = v$ and has length greater than zero. A path or circuit is *simple* if it does not contain the same edge (vertex) more than once.

  ◇ it starts and ends with a vertex
  ◇ each edge joins the vertex before it in the sequence to the vertex after it in the sequence
  ◇ no edge (vertex) appears more than once in the sequence

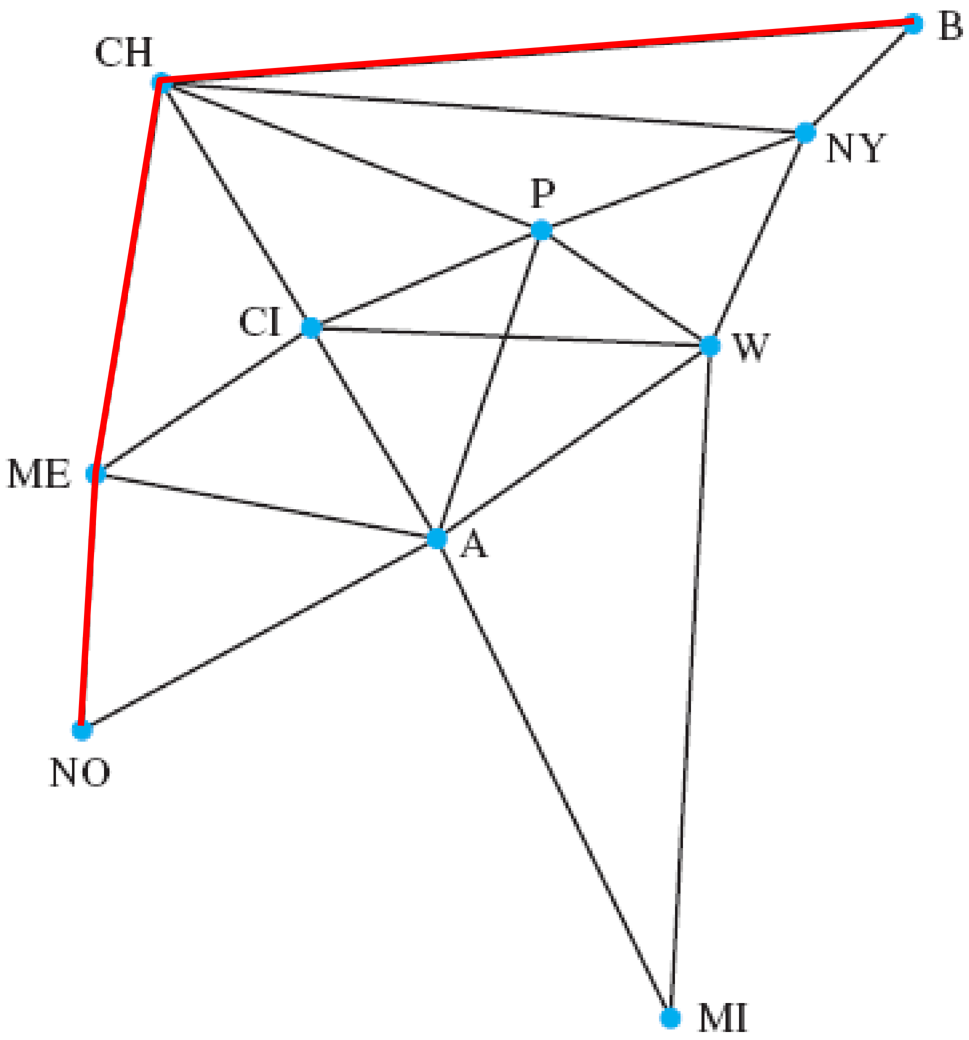
Path

- **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 $e_1, e_2, \ldots, e_n$ 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 the endpoints x_{i-1} and x_i for $i = 1, \ldots, n$. The path is a *circuit* if it begins and ends at the same vertex, i.e., if $u = v$ and has length greater than zero. A path or circuit is *simple* if it does not contain the same edge (vertex) more than once.

  ◇ it starts and ends with a vertex
  ◇ each edge joins the vertex before it in the sequence to the vertex after it in the sequence
  ◇ no edge (vertex) appears more than once in the sequence

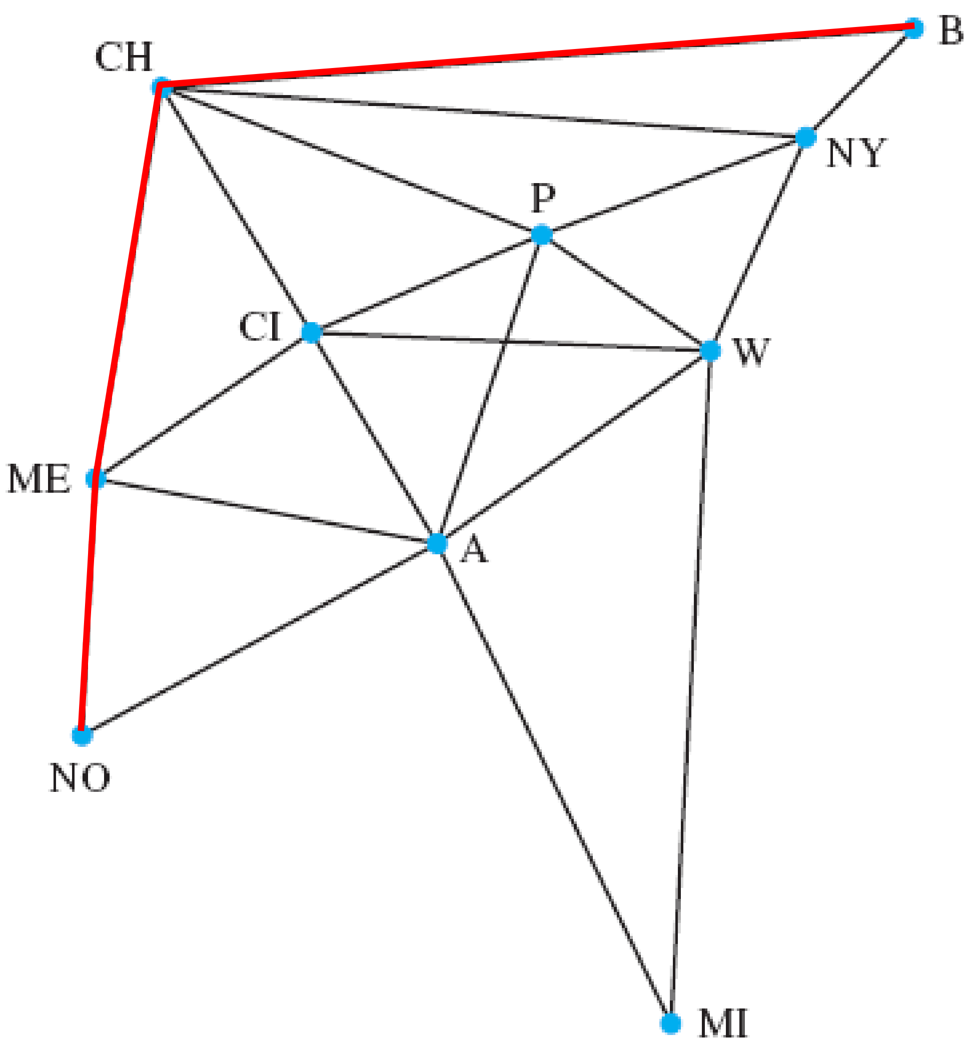
Length of a path $=$ $\#$ of edges on path

Path



Path

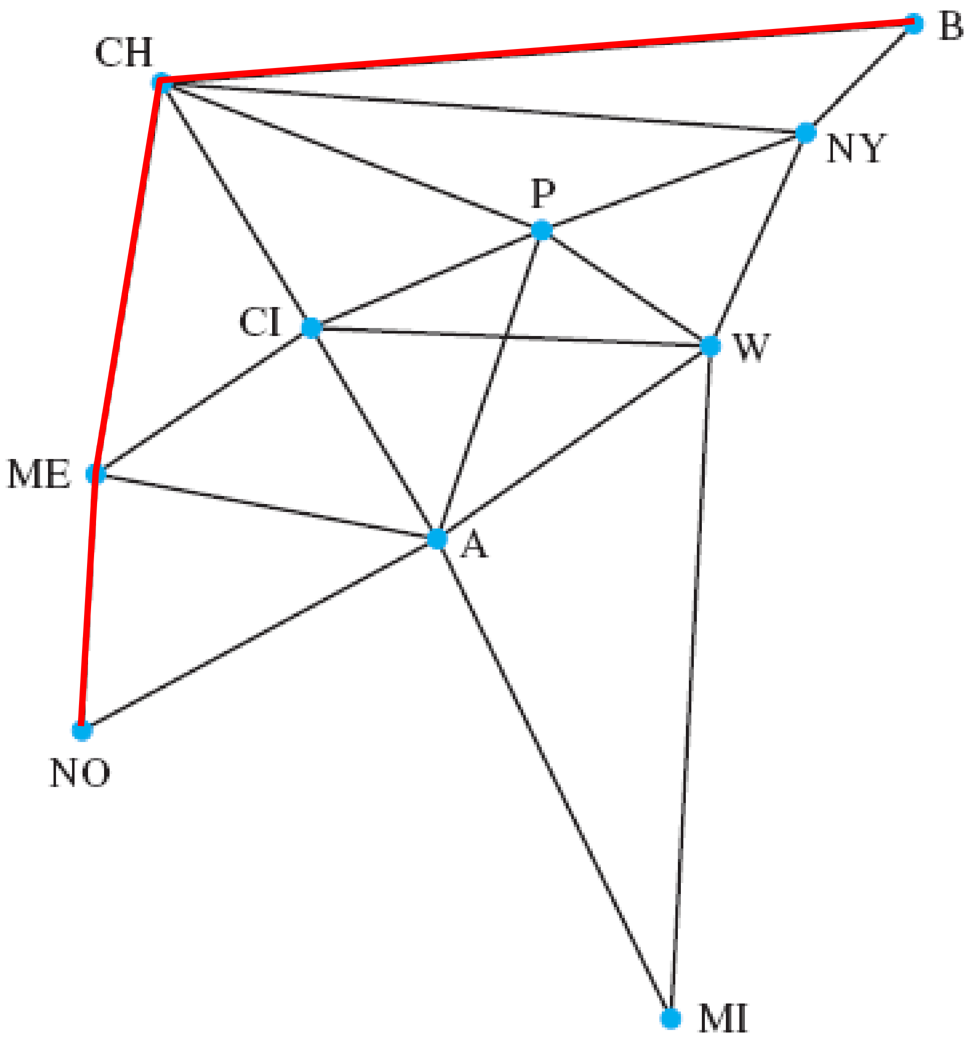
Path from Boston to New Orleans is B, CH, ME, NO



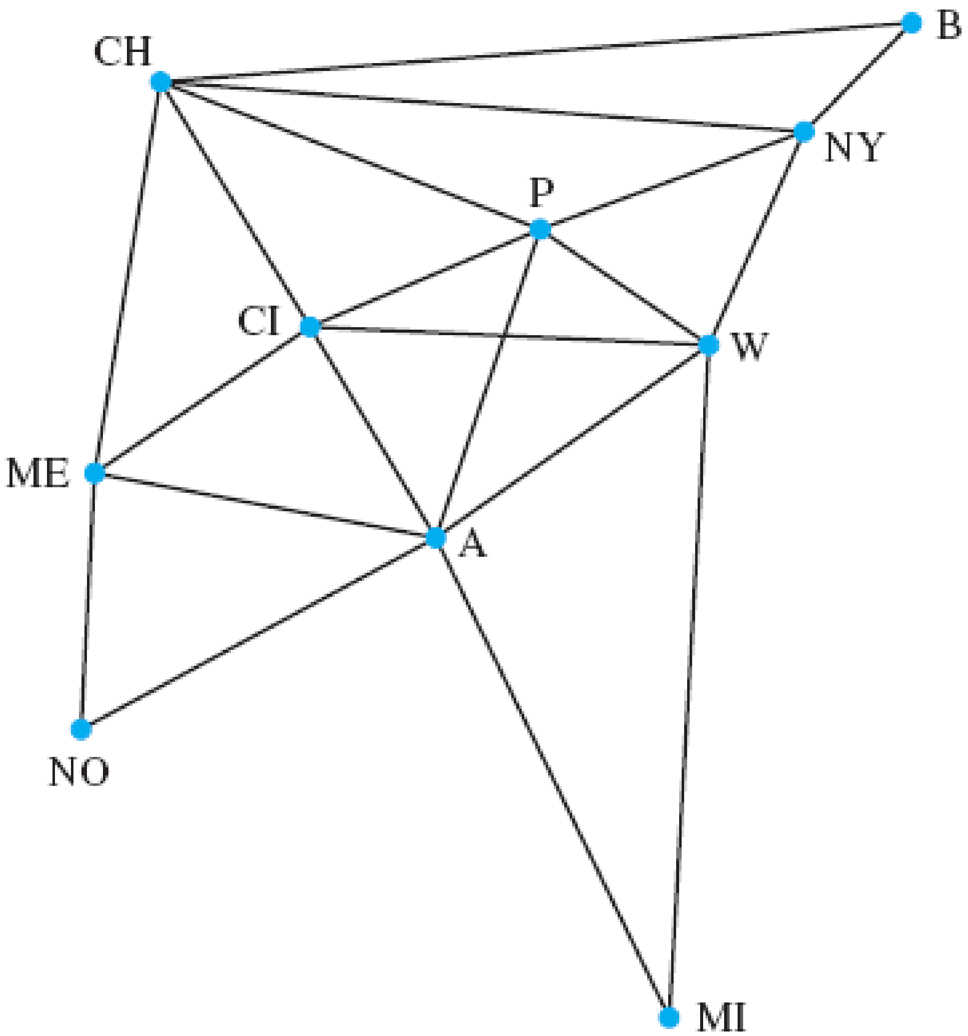
Path

Path from Boston to New Orleans is B, CH, ME, NO

This path has length 3.



Connectivity



Company decides to lease only minimum number of communication lines it needs to be able to send a message from any city to any other city by using any number of intermediate cities.

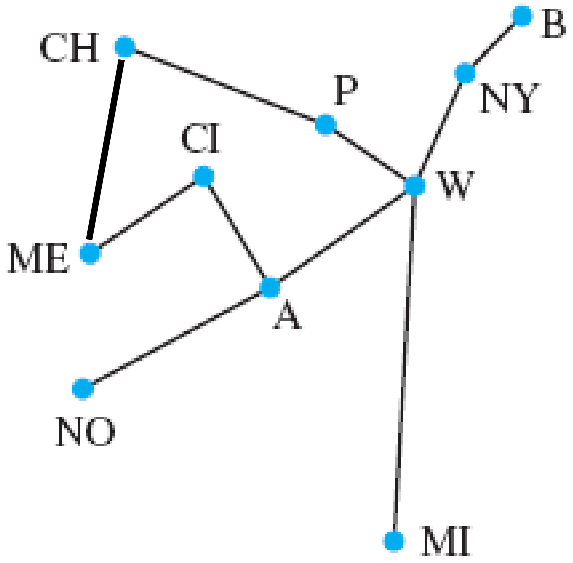
What is the minimum number of lines it needs to lease?

Connectivity

- Choosing 10 edges?

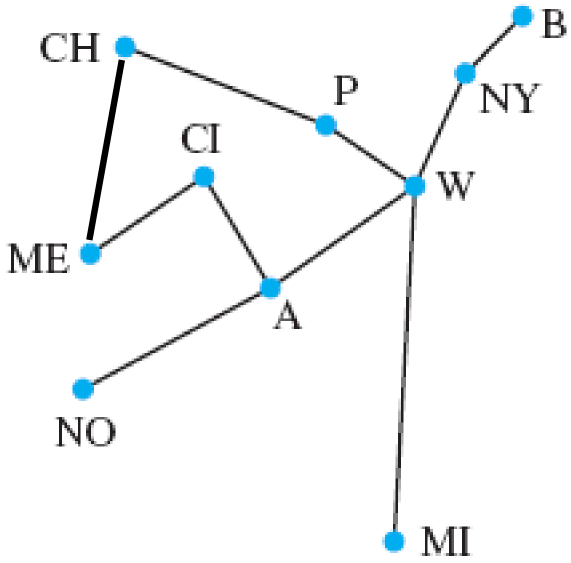
Connectivity

- Choosing 10 edges?



Connectivity

- Choosing 10 edges?

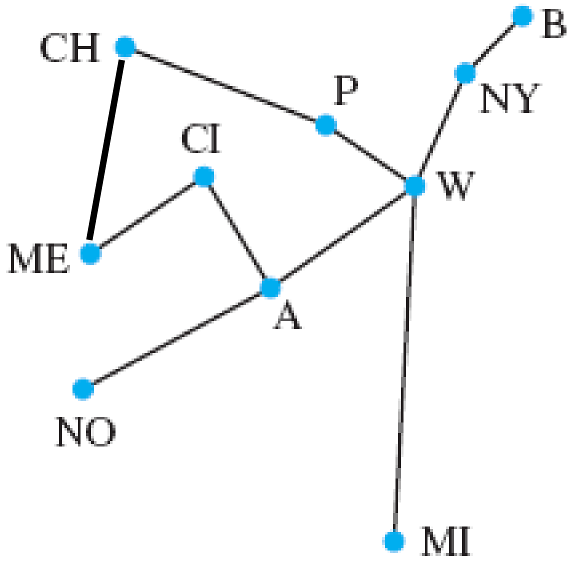


Too many.

Could throw away edge CI, A, and still have a solution.

Connectivity

- Choosing 10 edges?



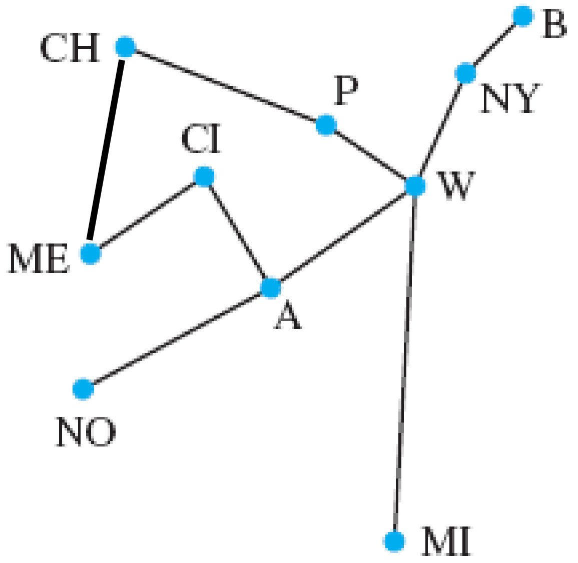
Choosing 8 edges?

Too many.

Could throw away edge CI, A, and still have a solution.

Connectivity

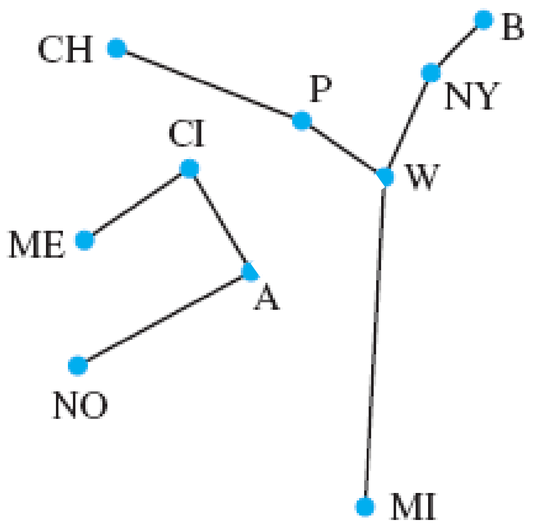
- Choosing 10 edges?



Too many.

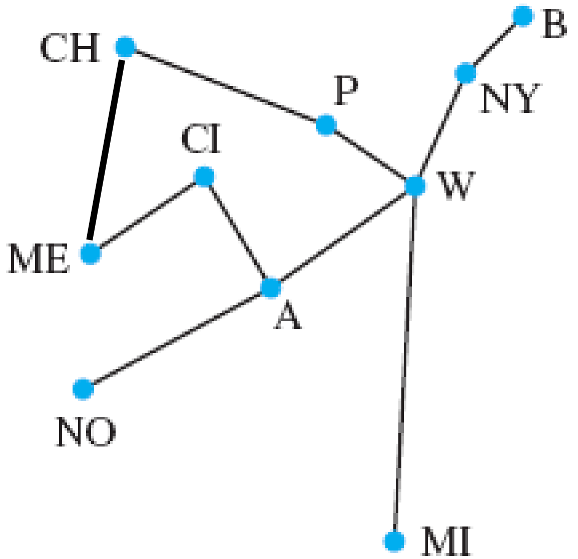
Could throw away edge CI, A, and still have a solution.

Choosing 8 edges?



Connectivity

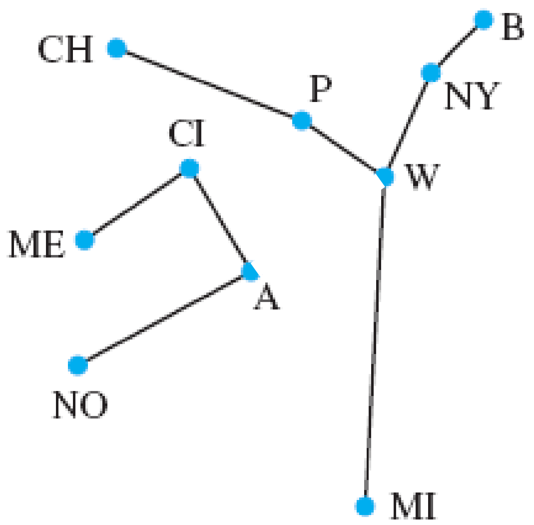
- Choosing 10 edges?



Too many.

Could throw away edge CI, A, and still have a solution.

Choosing 8 edges?



Not enough.

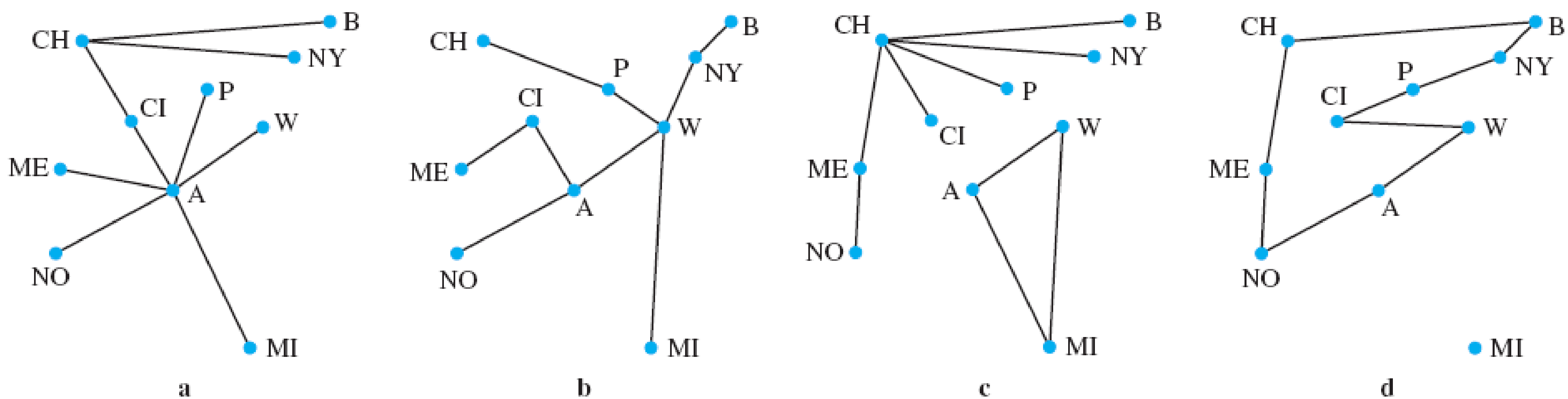
There is no path from, e.g., NO to B.

Connectivity

- Choosing 9 edges:

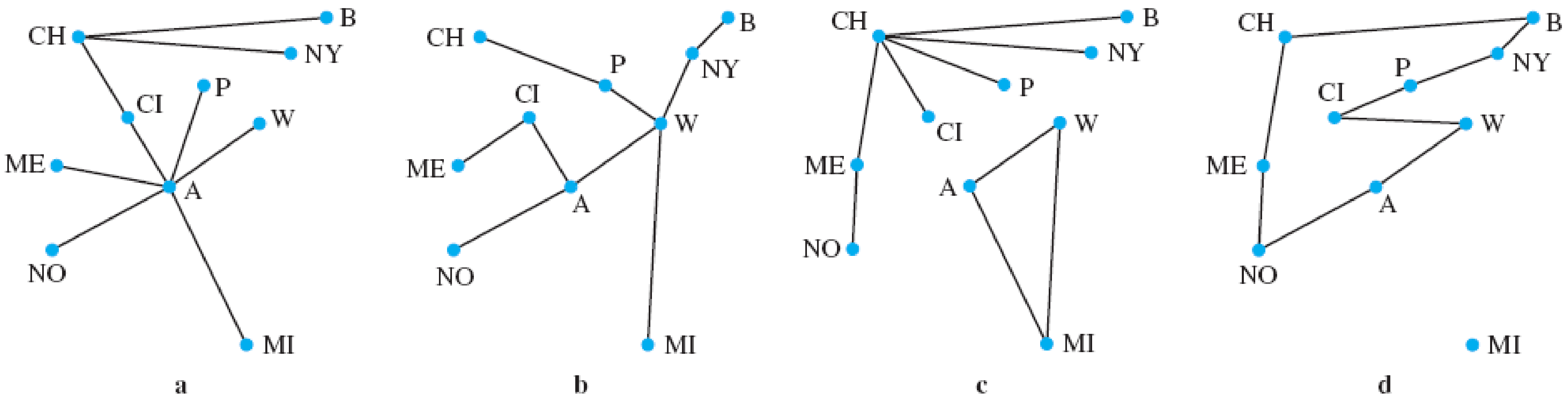
Connectivity

- Choosing 9 edges:



Connectivity

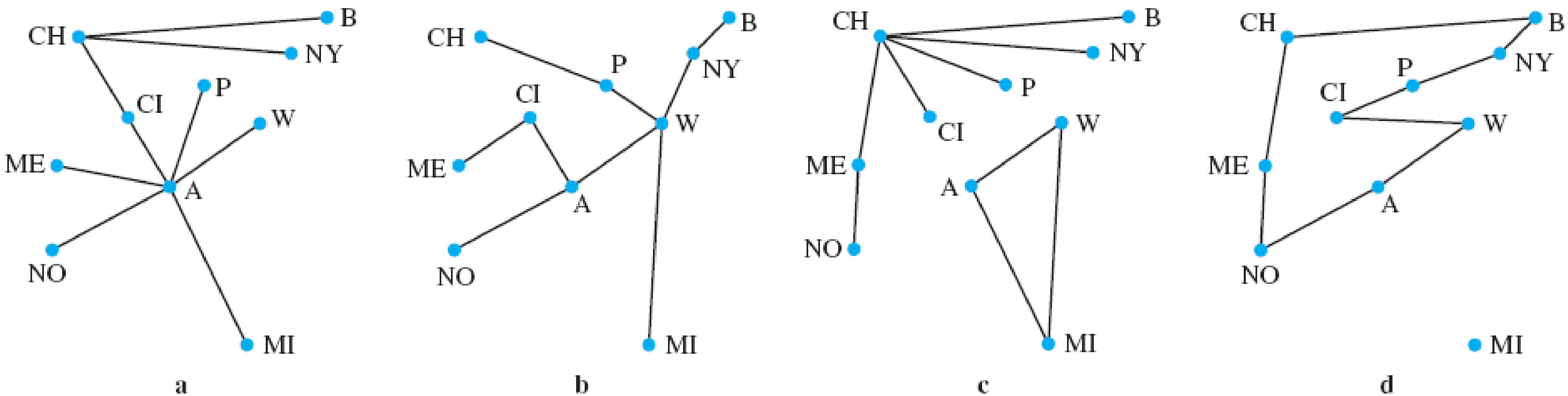
- Choosing 9 edges:



Two vertices are *connected* if there is a path between them.

Connectivity

- Choosing 9 edges:

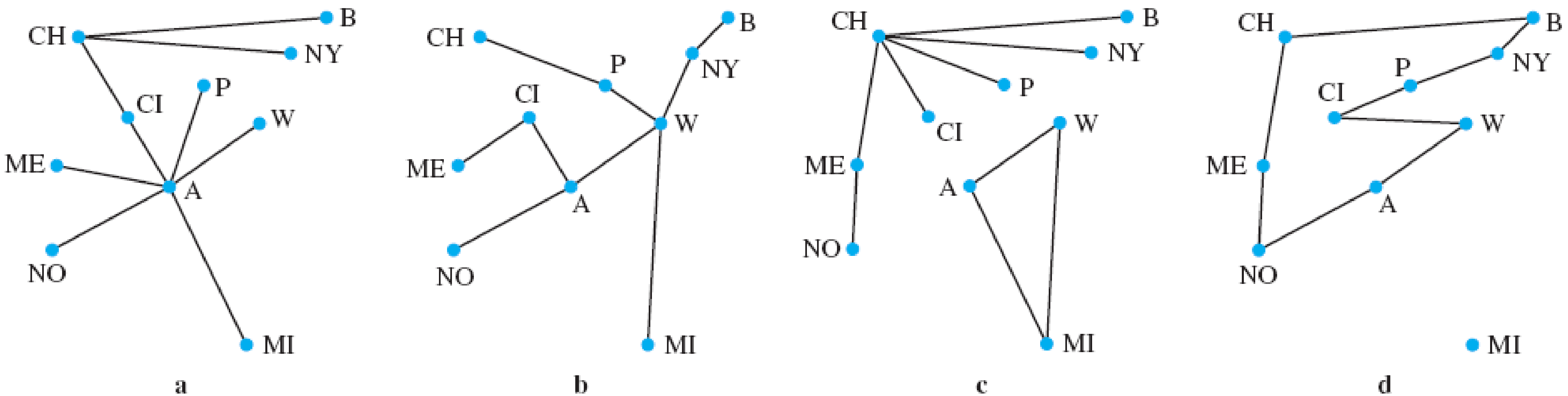


Two vertices are *connected* if there is a path between them.

Example: W, B are connected in (b), but are disconnected in (c).

Connectivity

- Choosing 9 edges:



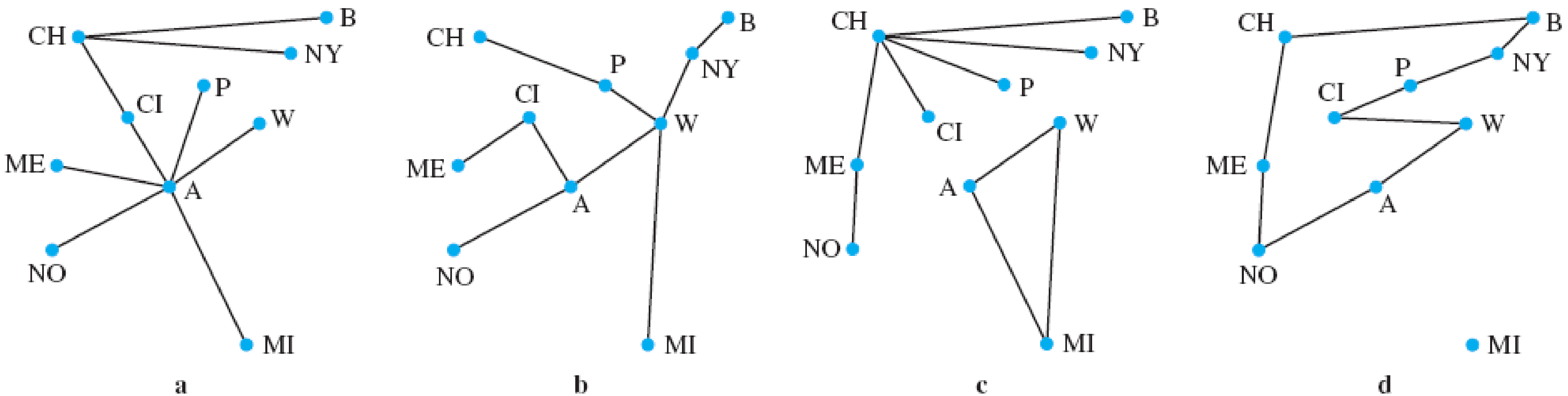
Two vertices are *connected* if there is a path between them.

Example: W, B are connected in (b), but are disconnected in (c).

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

Connectivity

- Choosing 9 edges:



Two vertices are *connected* if there is a path between them.

Example: W, B are connected in (b), but are disconnected in (c).

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

Example: (a) and (b) are connected, (c) and (d) are disconnected.

Path vs Walk

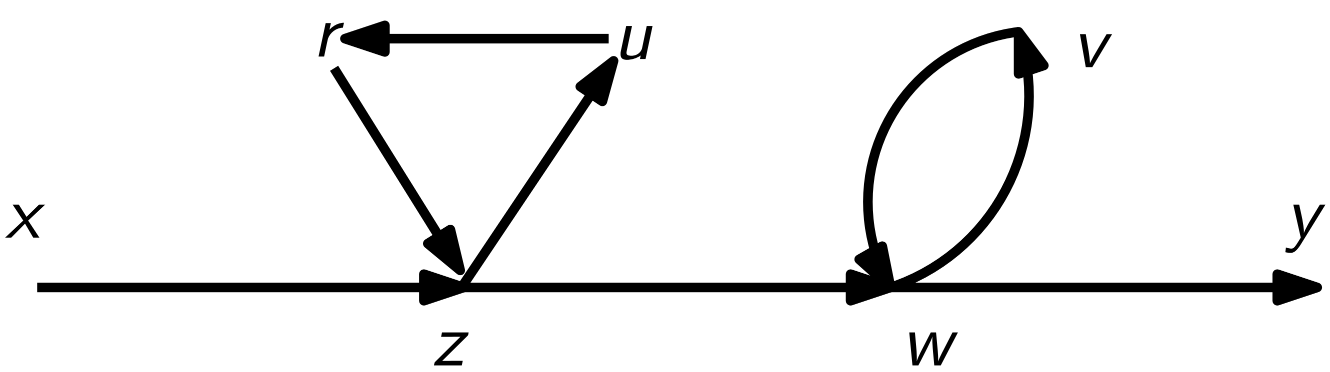
- **Lemma** If there is a path between two distinct vertices x and y of a graph G, then there is a simple path between x and y in G.

Path vs Walk

- **Lemma** If there is a path between two distinct vertices x and y of a graph G, then there is a simple path between x and y in G.
  Proof Just delete cycles (loops).

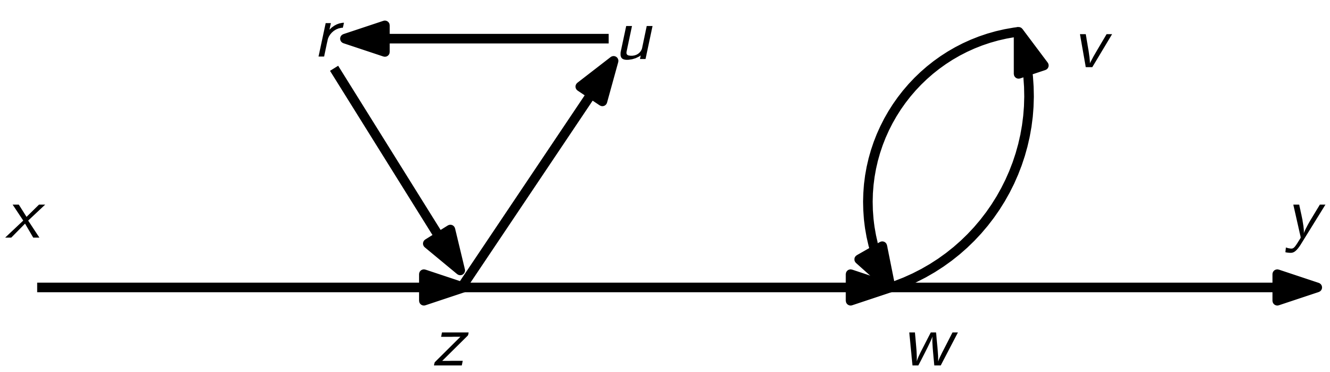
Path vs Walk

- **Lemma** If there is a path between two distinct vertices x and y of a graph G, then there is a simple path between x and y in G.
  Proof Just delete cycles (loops).



Path vs Walk

- **Lemma** If there is a path between two distinct vertices x and y of a graph G, then there is a simple path between x and y in G.
  Proof Just delete cycles (loops).

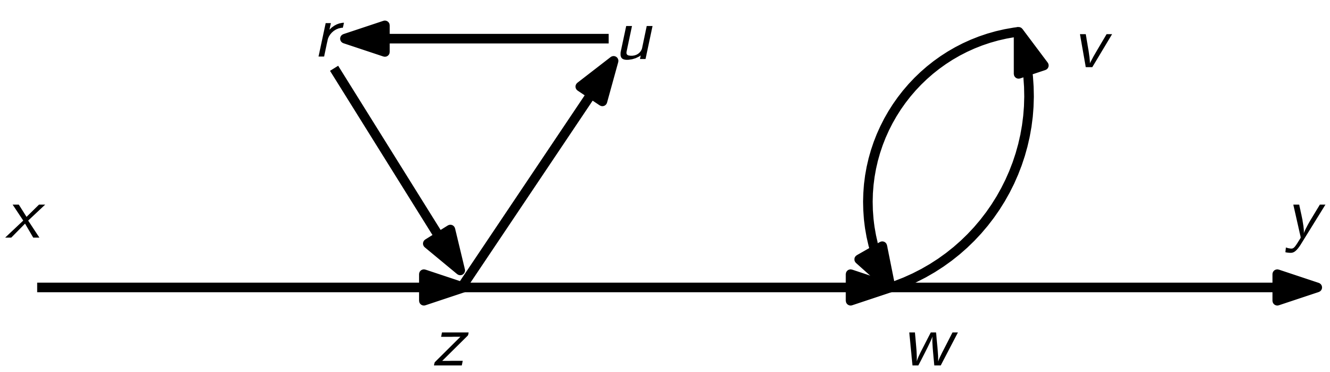


Path from x to y

$x, z, u, r, z, w, v, w, y$.

Path vs Walk

- **Lemma** If there is a path between two distinct vertices x and y of a graph G, then there is a simple path between x and y in G.
  Proof Just delete cycles (loops).



Path from x to y

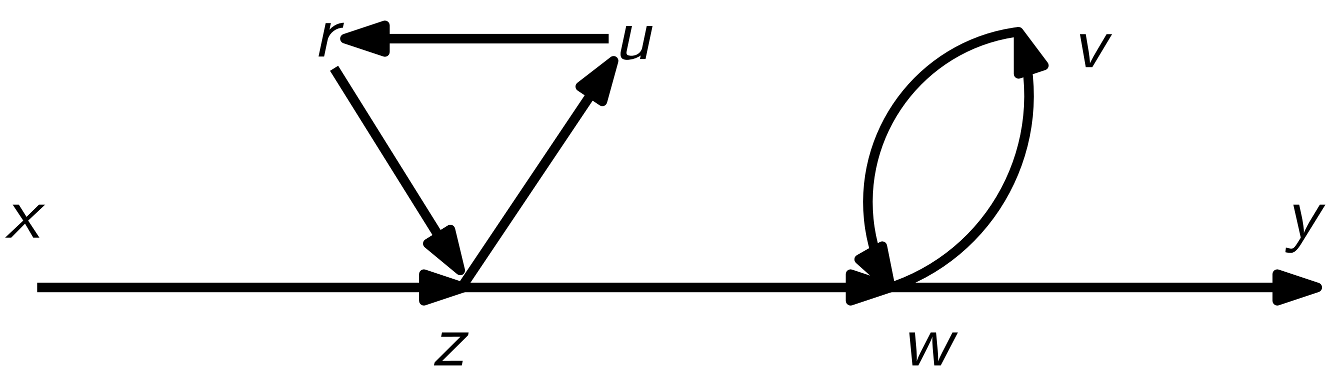
$x, z, u, r, z, w, v, w, y$.

Path from x to y

x, z, w, y.

Path vs Walk

- **Lemma** If there is a path between two distinct vertices x and y of a graph G, then there is a simple path between x and y in G.
  Proof Just delete cycles (loops).



Path from x to y
$x, z, u, r, z, w, v, w, y$.

→

Path from x to y
x, z, w, y.

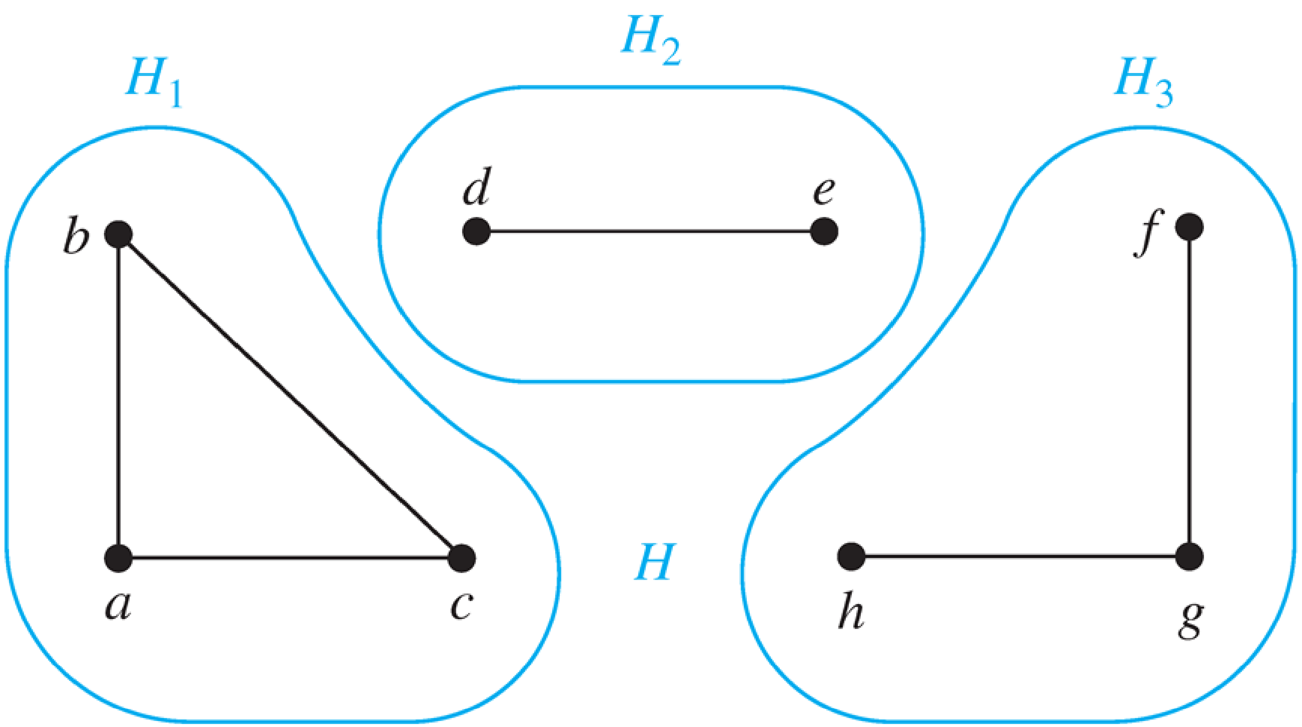
Theorem There is a simple path between every pair of distinct vertices of a connected undirected graph.

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.

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.



Connectedness in Directed Graphs

- **Definition** A directed graph is *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.

Connectedness in Directed Graphs

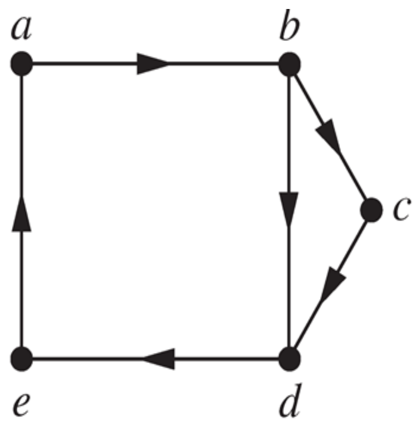
- **Definition** A directed graph is *strongly connected* if there is a path from a to b and a path from b to a whenever a and b are vertices in the graph.

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

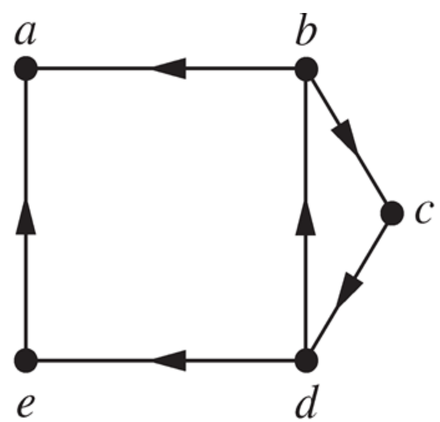
Connectedness in Directed Graphs

- **Definition** A directed graph is *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.

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



G



H

Cut Vertices and Cut Edges

- Sometimes the removal from a graph of a vertex and all incident edges disconnect the graph. Such vertices are called *cut vertices*. Similarly we may define *cut edges*.

Cut Vertices and Cut Edges

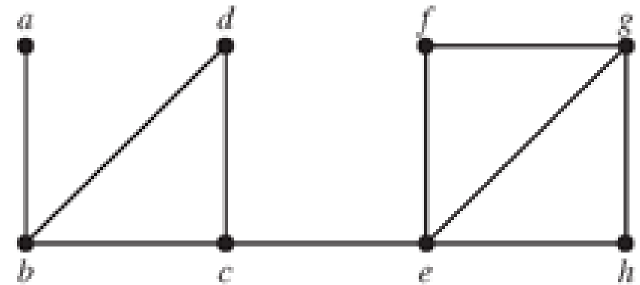
- Sometimes the removal from a graph of a vertex and all incident edges disconnect the graph. Such vertices are called *cut vertices*. Similarly we may define *cut edges*.

  A set of edges E' is called an *edge cut* of G if the subgraph $G - E'$ is disconnected. The *edge connectivity* $\lambda(G)$ is the minimum number of edges in an edge cut of G.

Cut Vertices and Cut Edges

- Sometimes the removal from a graph of a vertex and all incident edges disconnect the graph. Such vertices are called *cut vertices*. Similarly we may define *cut edges*.

  A set of edges E' is called an *edge cut* of G if the subgraph $G - E'$ is disconnected. The *edge connectivity* $\lambda(G)$ is the minimum number of edges in an edge cut of G.



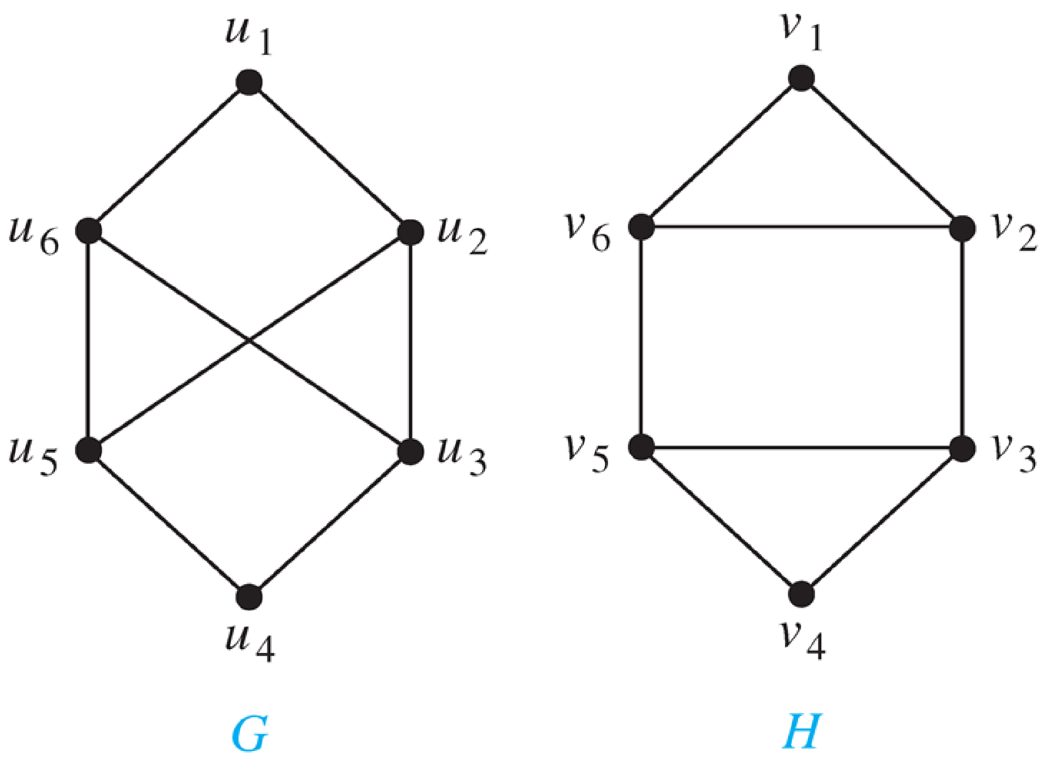
G_1

Paths and Isomorphism

- The existence of a simple circuit of length k is isomorphic invariant. In addition, paths can be used to construct mappings that may be isomorphisms.

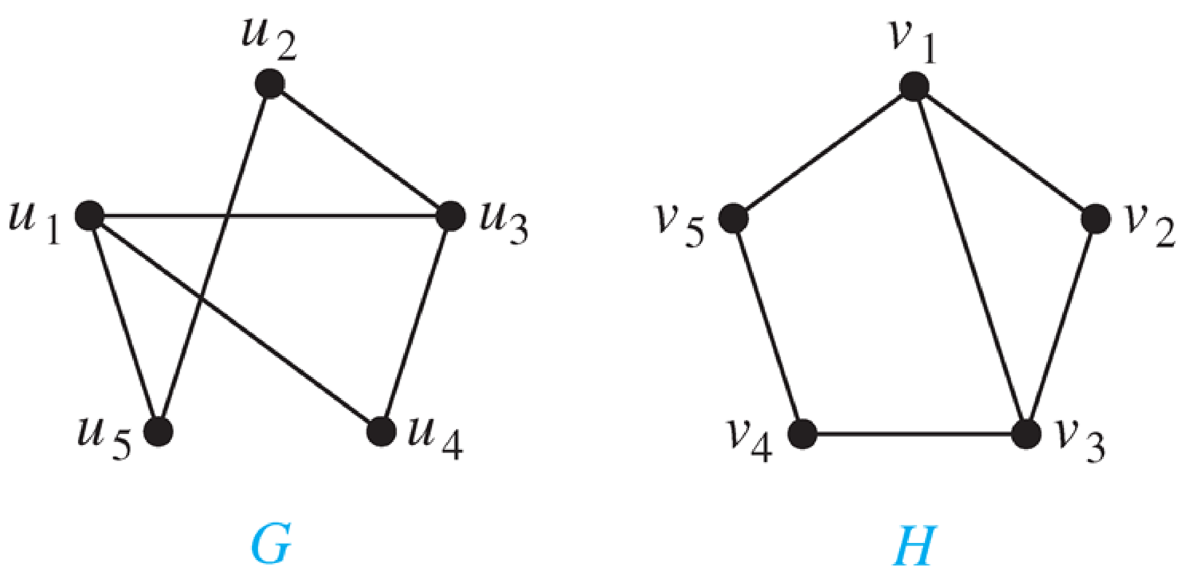
Paths and Isomorphism

- The existence of a simple circuit of length k is isomorphic invariant. In addition, paths can be used to construct mappings that may be isomorphisms.



Paths and Isomorphism

- The existence of a simple circuit of length k is isomorphic invariant. In addition, paths can be used to construct mappings that may be isomorphisms.



Counting Paths between Vertices

- **Theorem** Let G be a graph with adjacency matrix $\mathbf{A}$ with respect to the ordering $v_1, v_2, \ldots, v_n$ of vertices. The number of different paths of length r from v_i to v_j, where $r > 0$ is positive, equals the (i, j)-th entry of $\mathbf{A}^r$.

Counting Paths between Vertices

- **Theorem** Let G be a graph with adjacency matrix $\mathbf{A}$ with respect to the ordering $v_1, v_2, \ldots, v_n$ of vertices. The number of different paths of length r from v_i to v_j, where $r > 0$ is positive, equals the (i,j)-th entry of $\mathbf{A}^r$.

Proof (by **induction**)

Counting Paths between Vertices

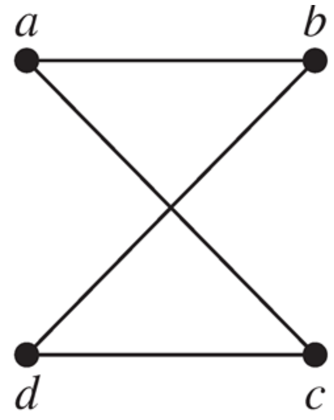
- **Theorem** Let G be a graph with adjacency matrix $\mathbf{A}$ with respect to the ordering $v_1, v_2, \ldots, v_n$ of vertices. The number of different paths of length r from v_i to v_j, where $r > 0$ is positive, equals the (i, j)-th entry of $\mathbf{A}^r$.

Proof (by **induction**)

$\mathbf{A}^{r+1} = \mathbf{A}^r\mathbf{A}$, the (i, j)-th entry of $\mathbf{A}^{r+1}$ equals $b_{i1}a_{1j} + b_{i2}a_{2j} + \cdots + b_{in}a_{nj}$, where b_{ik} is the (i, k)-th entry of $\mathbf{A}^r$.

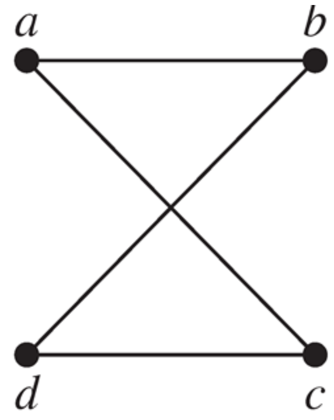
Counting Paths between Vertices

- **Example** How many paths of length 4 are there from a to d in the graph G?



Counting Paths between Vertices

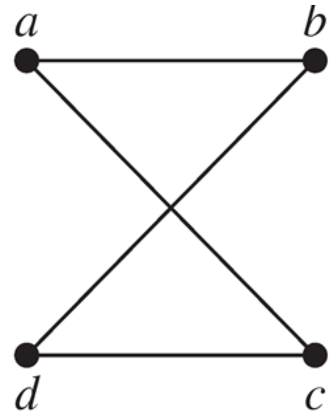
- **Example** How many paths of length 4 are there from a to d in the graph G?



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

Counting Paths between Vertices

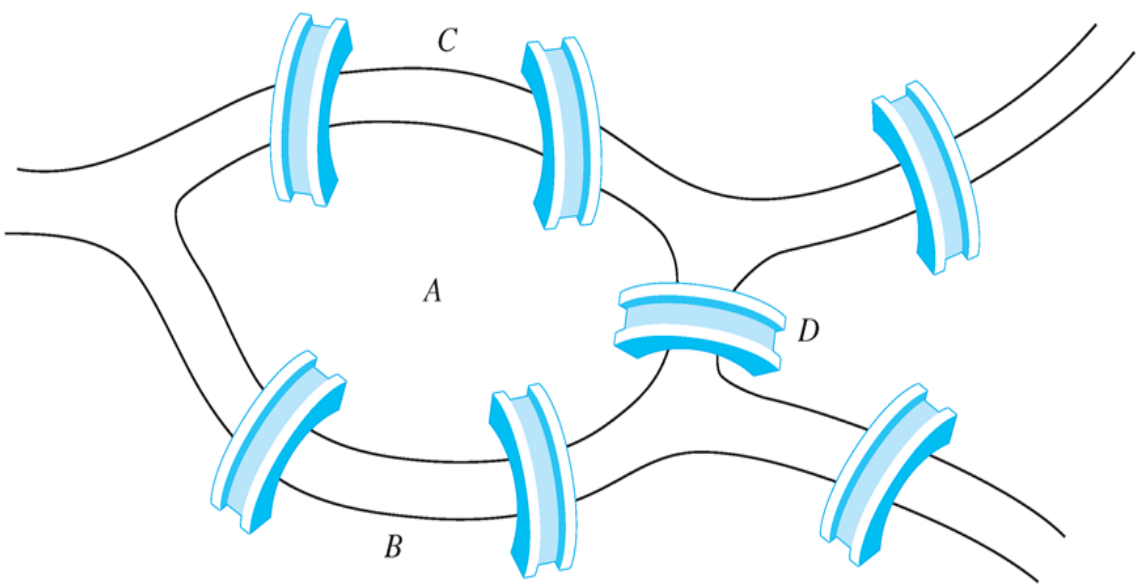
- **Example** How many paths of length 4 are there from a to d in the graph G?



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

- **Königsberg seven-bridge problem**

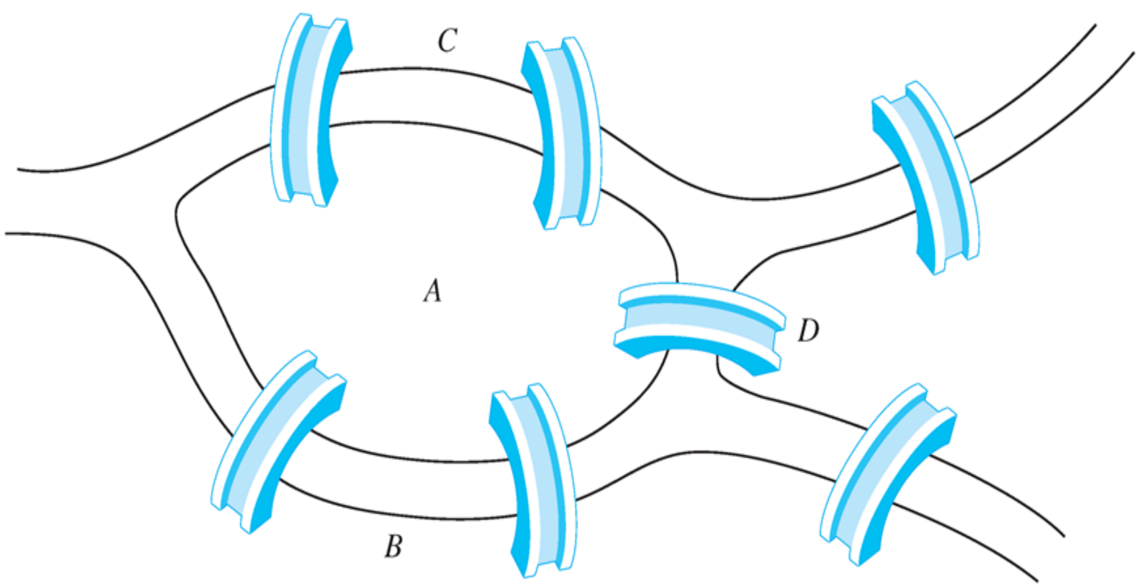
  People wondered whether it was possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point.

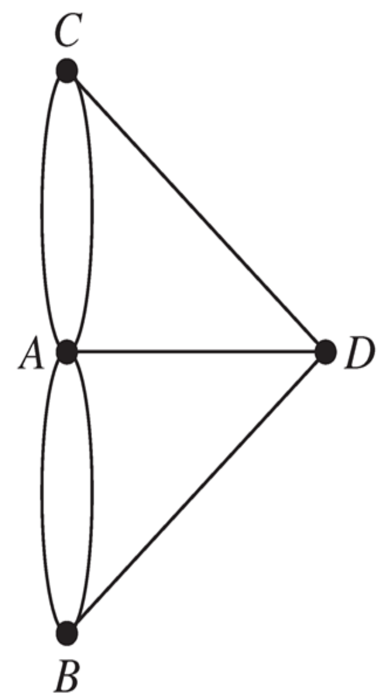


Euler Paths

Königsberg seven-bridge problem

People wondered whether it was possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point.





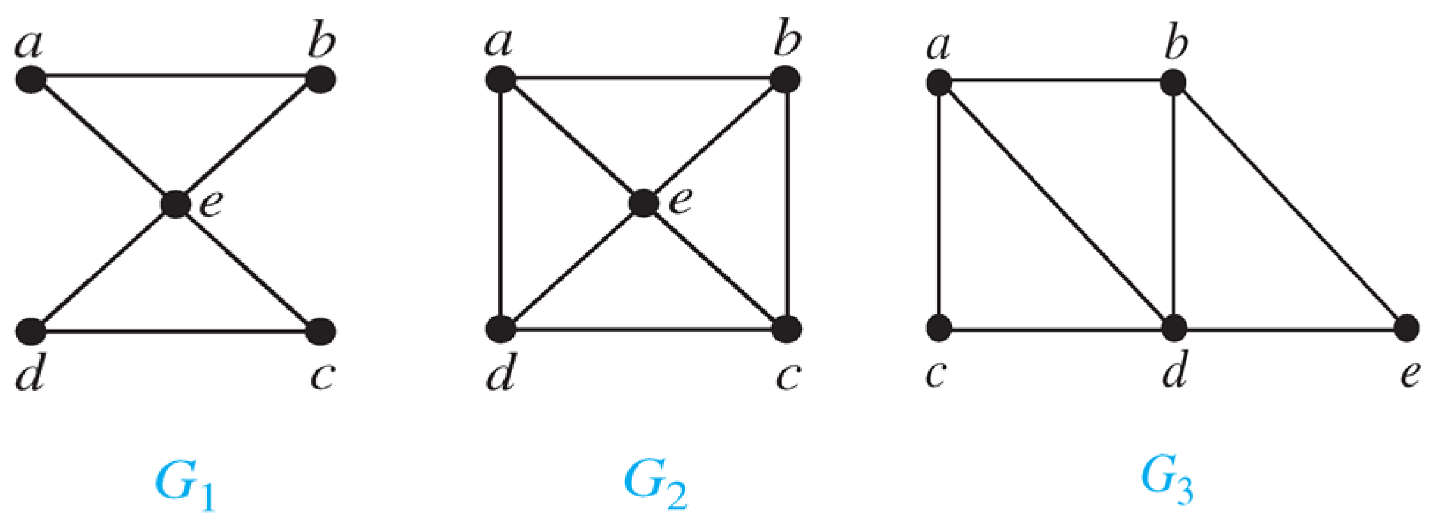
Euler Paths and Circuits

- **Definition** An *Euler circuit* in a graph G is a simple circuit containing every edge of G. An *Euler path* in G is a simple path containing every edge of G.

Euler Paths and Circuits

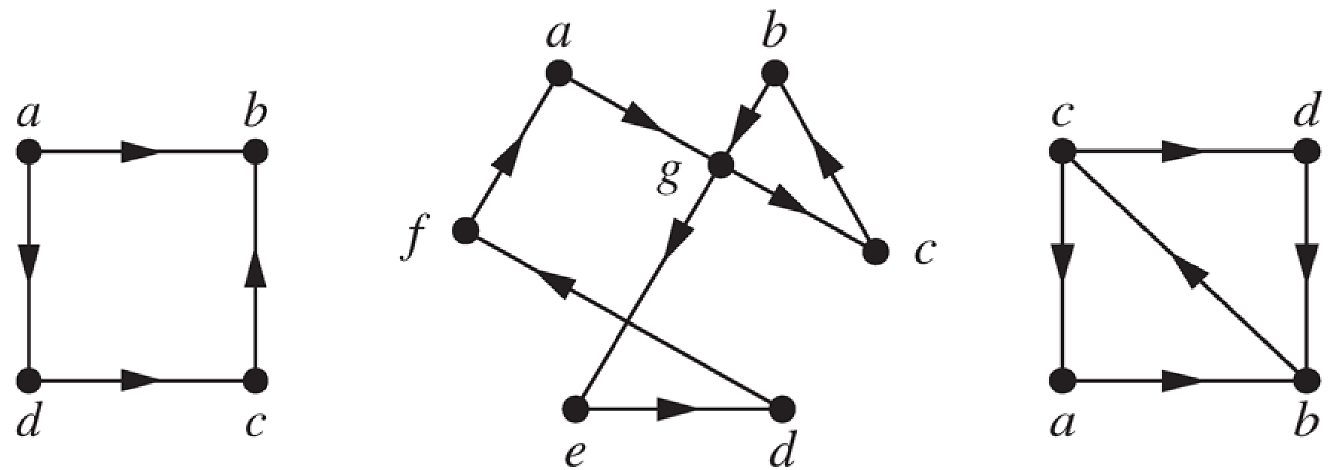
- **Definition** An *Euler circuit* in a graph G is a simple circuit containing every edge of G. An *Euler path* in G is a simple path containing every edge of G.

Example Which of the undirected graphs have an Euler circuit? Of those that do not, which have an Euler path?



- **Definition** An *Euler circuit* in a graph G is a simple circuit containing every edge of G. An *Euler path* in G is a simple path containing every edge of G.

Example Which of the undirected graphs have an Euler circuit? Of those that do not, which have an Euler path?



H_1 H_2 H_3

Necessary Conditions for Euler Circuits and Paths

- Euler Circuit $\Rightarrow$ The degree of every vertex must be even

Necessary Conditions for Euler Circuits and Paths

- Euler Circuit $\Rightarrow$ The degree of every vertex must be even
  - Each time the circuit passes through a vertex, it contributes two to the vertex's degree.

Necessary Conditions for Euler Circuits and Paths

- Euler Circuit $\Rightarrow$ The degree of every vertex must be even
  - Each time the circuit passes through a vertex, it contributes two to the vertex's degree.
  - The circuit starts with a vertex a and ends at a, then contributes two to $\deg(a)$.

Necessary Conditions for Euler Circuits and Paths

- Euler Circuit $\Rightarrow$ The degree of every vertex must be even
  - Each time the circuit passes through a vertex, it contributes two to the vertex's degree.
  - The circuit starts with a vertex a and ends at a, then contributes two to $\deg(a)$.

Euler Path $\Rightarrow$ The graph has exactly two vertices of odd degree

Necessary Conditions for Euler Circuits and Paths

- Euler Circuit $\Rightarrow$ The degree of every vertex must be even
  - ◇ Each time the circuit passes through a vertex, it contributes two to the vertex's degree.
  - ◇ The circuit starts with a vertex a and ends at a, then contributes two to $\deg(a)$.

Euler Path $\Rightarrow$ The graph has exactly two vertices of odd degree

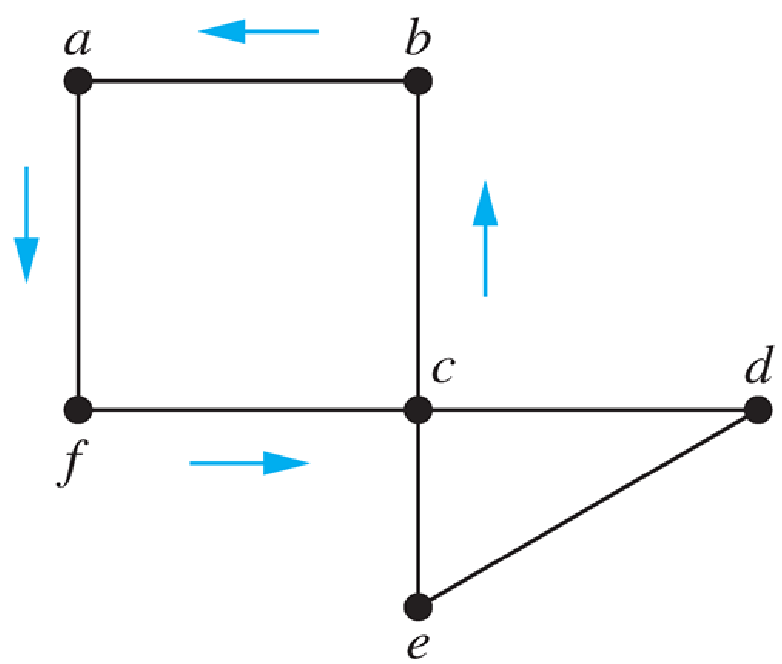
- ◇ The initial vertex and the final vertex of an Euler path have odd degree.

Sufficient Conditons for Euler Circuits and Paths

- Suppose that G is a connected multigraph with ≥ 2 vertices, all of even degree.

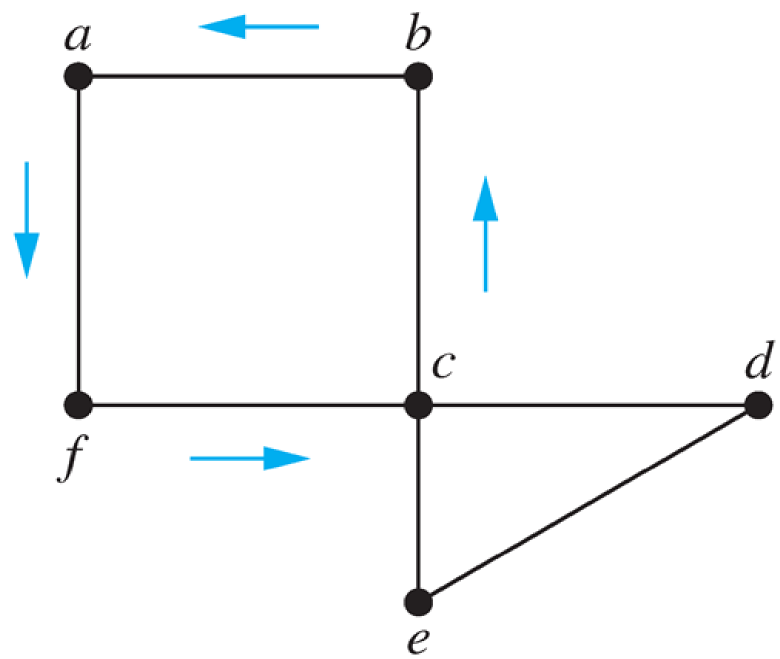
Sufficient Conditons for Euler Circuits and Paths

- Suppose that G is a connected multigraph with ≥ 2 vertices, all of even degree.

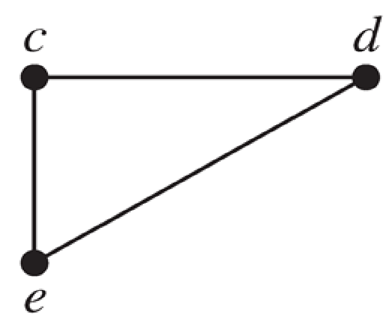


G

- Suppose that G is a connected multigraph with ≥ 2 vertices, all of even degree.



G



H

Algorithm for Constructing an Euler Circuits

procedure *Euler*(G: connected multigraph with all vertices of even degree)
circuit := a circuit in G beginning at an arbitrarily chosen vertex with edges successively added to form a path that returns to this vertex.
H := G with the edges of this circuit removed
while H has edges
  subciruit := a circuit in H beginning at a vertex in H that also is an endpoint of an edge in circuit.
  H := H with edges of *subciruit* and all isolated vertices removed
  circuit := *circuit* with *subcircuit* inserted at the appropriate vertex.
return *circuit*{*circuit* is an Euler circuit}

Algorithm for Constructing an Euler Circuits

- **procedure** *Euler*(G: connected multigraph with all vertices of even degree)
  circuit := a circuit in G beginning at an arbitrarily chosen vertex with edges
  successively added to form a path that returns to this vertex.
  $\boxed{H := G \text{ with the edges of this circuit removed}}$
  while H has edges
  subciruit := a circuit in H beginning at a vertex in H that also is
  an endpoint of an edge in circuit.
  $H := H$ with edges of *subciruit* and all isolated vertices removed
  circuit := *circuit* with *subcircuit* inserted at the appropriate vertex.
  return *circuit*{*circuit* is an Euler circuit}

Algorithm for Constructing an Euler Circuits

procedure *Euler*(G: connected multigraph with all vertices of even degree)
circuit := a circuit in G beginning at an arbitrarily chosen vertex with edges
successively added to form a path that returns to this vertex.
H := G with the edges of this circuit removed
while H has edges
subciruit := a circuit in H beginning at a vertex in H that also is
an endpoint of an edge in circuit.
H := H with edges of *subciruit* and all isolated vertices removed
circuit := *circuit* with *subcircuit* inserted at the appropriate vertex.
return *circuit*{*circuit* is an Euler circuit}

Necessary and Sufficient Conditions

- **Theorem** A connected multigraph with at least two vertices has an *Euler circuit* if and only if each of its vertices has **even** degree.

Necessary and Sufficient Conditions

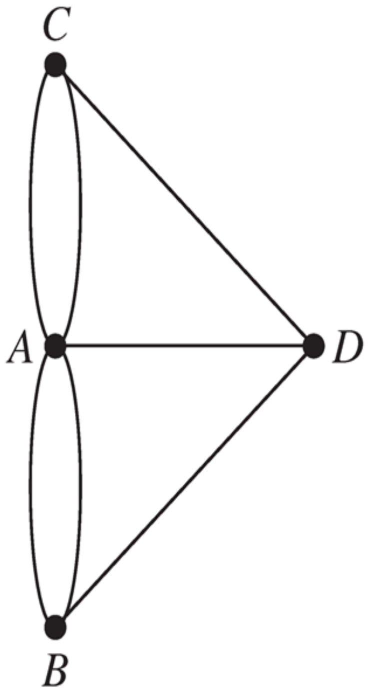
- **Theorem** A connected multigraph with at least two vertices has an *Euler circuit* if and only if each of its vertices has **even** degree.

  Theorem A connected multigraph has an *Euler path* but not an *Euler circuit* if and only if it has exactly two vertices of odd degree.

Necessary and Sufficient Conditions

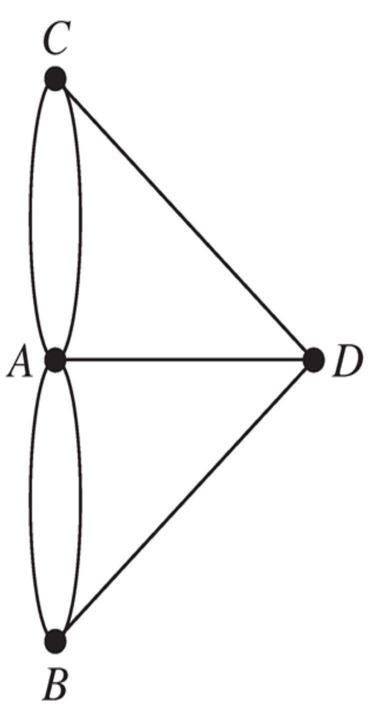
- **Theorem** A connected multigraph with at least two vertices has an *Euler circuit* if and only if each of its vertices has **even** degree.

  Theorem A connected multigraph has an *Euler path* but not an *Euler circuit* if and only if it has exactly two vertices of odd degree.



- **Theorem** A connected multigraph with at least two vertices has an *Euler circuit* if and only if each of its vertices has **even** degree.

  Theorem A connected multigraph has an *Euler path* but not an *Euler circuit* if and only if it has exactly two vertices of odd degree.



No Euler circuit

Euler Circuits and Paths

- **Example**

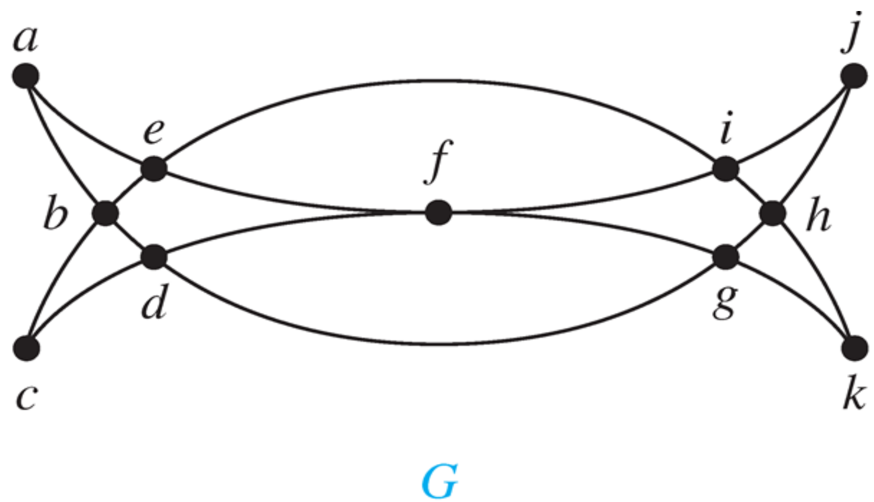
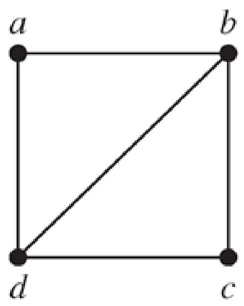
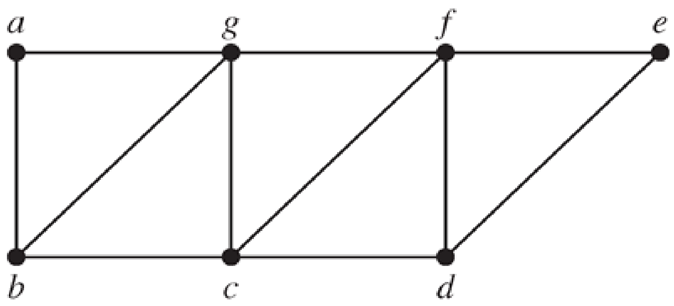


FIGURE 6 **Mohammed's Scimitars.**

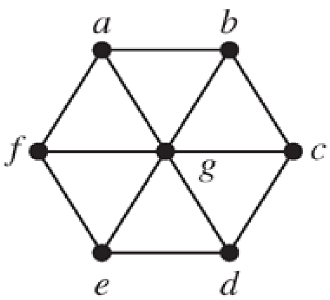
- **Example**



G_1



G_2



G_3

Applications of Euler Paths and Circuits

- Finding a path or circuit that traverses each
  - street in a neighborhood
  - road in a transportation network
  - link in a communication network
  - ...

Applications of Euler Paths and Circuits

- Finding a path or circuit that traverses each
  - street in a neighborhood
  - road in a transportation network
  - link in a communication network
  - ...

Chinese Postman Problem Meigu Guan [60']

Applications of Euler Paths and Circuits

- Finding a path or circuit that traverses each
  - ◇ street in a neighborhood
  - ◇ road in a transportation network
  - ◇ link in a communication network
  - ◇ ...

Chinese Postman Problem Meigu Guan [60']

Given a graph $G = (V, E)$, for every $e \in E$, there is a nonnegative weight $w(e)$. Find a circuit W such that

$$\sum_{e \in W} w(e) = \min$$

Applications of Euler Paths and Circuits

- Finding a path or circuit that traverses each
  - street in a neighborhood
  - road in a transportation network
  - link in a communication network
  - ...

Chinese Postman Problem Meigu Guan [60']

Given a graph $G = (V, E)$, for every $e \in E$, there is a nonnegative weight $w(e)$. Find a circuit W such that

$$\sum_{e \in W} w(e) = \min$$

k-Postman Chinese Postman Problem (k-PCPP)

Applications of Euler Paths and Circuits

- Finding a path or circuit that traverses each
  - street in a neighborhood
  - road in a transportation network
  - link in a communication network
  - ...

Chinese Postman Problem Meigu Guan [60']

Given a graph $G = (V, E)$, for every $e \in E$, there is a nonnegative weight $w(e)$. Find a circuit W such that

$$\sum_{e \in W} w(e) = \min$$

k-Postman Chinese Postman Problem (k-PCPP)

$\in$ NPC

Next Lecture

- graph theory III ...