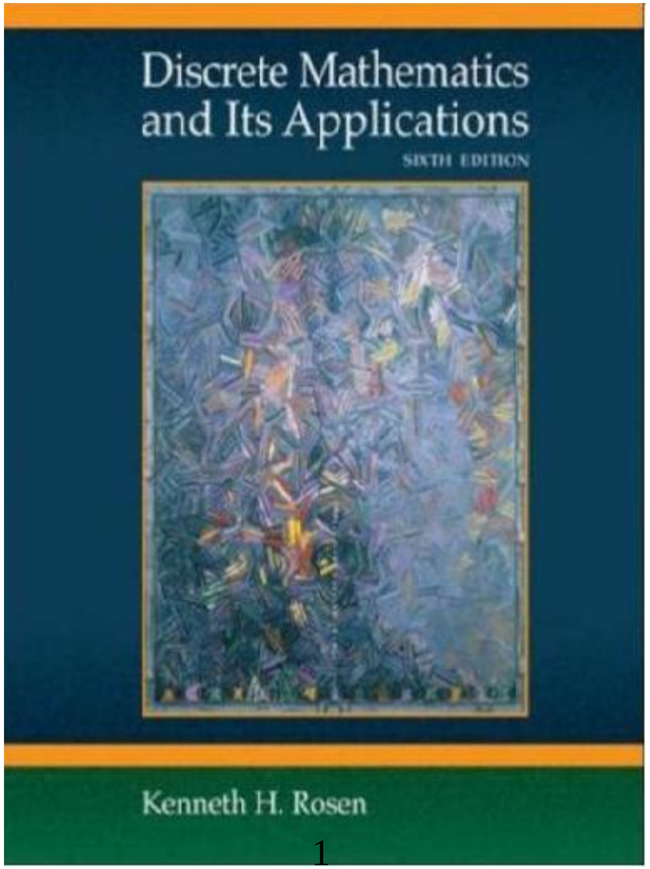
Jiangxi University of Science and Technology

Discrete Mathematics and Its Applications



Lecture019: Chapter 10:Graphs

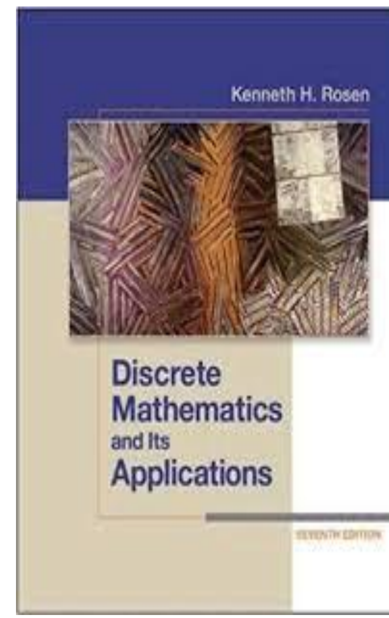
Section 10.2:

Graph Terminology and Special Types of Graphs

Acknowledgement

Most of these slides are adapted from ones created by Professor Bart Selman at Cornell University ,

and Dr Johnnie Baker and **Discrete Mathematics and Its Applications** **(Seventh Edition)** **Kenneth H. Rosen**



Section Summary

- Basic Terminology
- Some Special Types of Graphs
- Bipartite Graphs
- Bipartite Graphs and Matchings (*not currently included in overheads*)
- Some Applications of Special Types of Graphs (*not currently included in overheads*)
- New Graphs from Old

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.

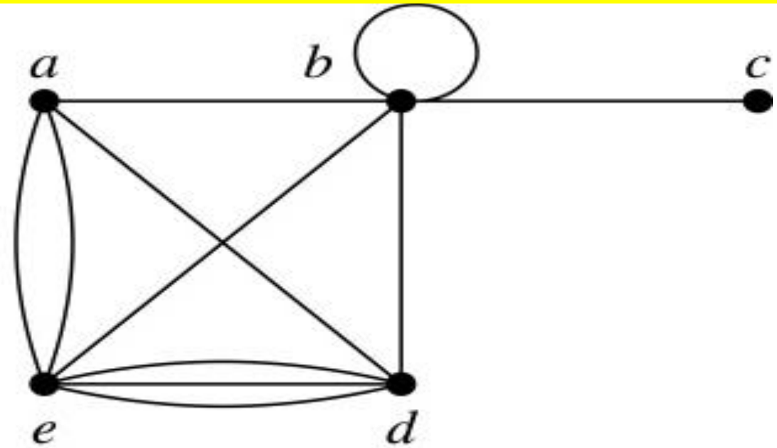
Definition 2. The set of all neighbors of a vertex v of $G = (V, E)$, denoted by $N(v)$, is called the *neighborhood* of v. If A is a subset of V, we denote by $N(A)$ the set of all vertices in G that are adjacent to at least one vertex in A. So, $N(A) = \bigcup_{v\in A} N(v)$.

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).

Degrees and Neighborhoods of Vertices

Example:

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



H

H: deg(a) = 4, deg(b) = deg(e) = 6, deg(c) = 1, deg(d) = 5.
$N(a) = \{b, d, e\}$, $N(b) = \{a, b, c, d, e\}$, $N(c) = \{b\}$,
$N(d) = \{a, b, e\}$, $N(e) = \{a, b, d\}$.

Degrees and Neighborhoods of Vertices

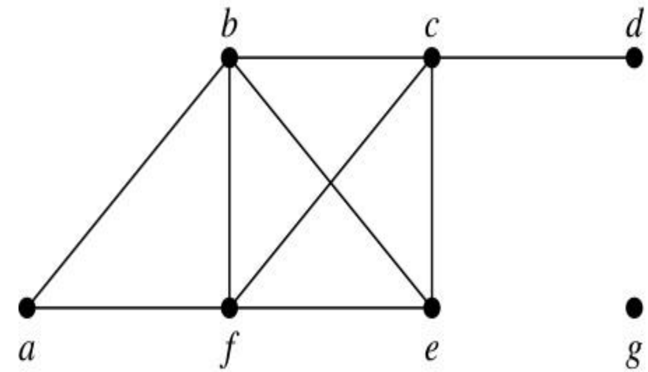
Example:

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

Solution:

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

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



G

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.

◀

Think about the graph where vertices represent the people at a party and an edge connects two people who have shaken hands.

Handshaking Theorem

We now give two examples illustrating the usefulness of the handshaking theorem.

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

Solution:
Because the sum of the degrees of the vertices is $6 \cdot 10 = 60$, the handshaking theorem tells us that $2m = 60$. So the number of edges $m = 30$

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

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

Degree of Vertices (*continued*)

Theorem 2: An undirected graph has an even number of vertices of odd degree.

Proof: Let V_1 be the vertices of even degree and V_2 be the vertices of odd degree in an undirected graph $G = (V, E)$ with m edges. Then

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

must be even since $\deg(v)$ is even for each $v \in V_1$

This sum must be even because $2m$ is even and the sum of the degrees of the vertices of even degrees is also even. Because this is the sum of the degrees of all vertices of odd degree in the graph, there must be an even number of such vertices.

Directed Graphs

Recall the definition of a directed graph.

Definition: An *directed graph G* = (*V*, *E*) consists of *V*, a nonempty set of *vertices* (or *nodes*), and *E*, a set of *directed edges* or *arcs*. Each edge is an ordered pair of vertices. The directed edge (*u*,*v*) is said to start at *u* and end at *v*.

Definition: Let (*u*,*v*) be an edge in *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.

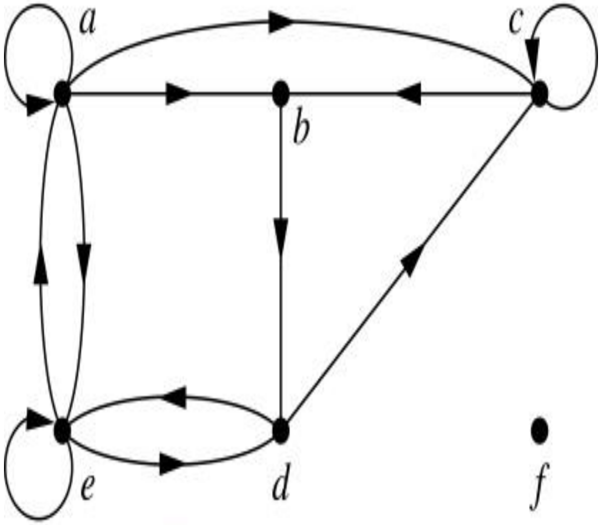
Directed Graphs (*continued*)

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

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

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



G

Directed Graphs (*continued*)

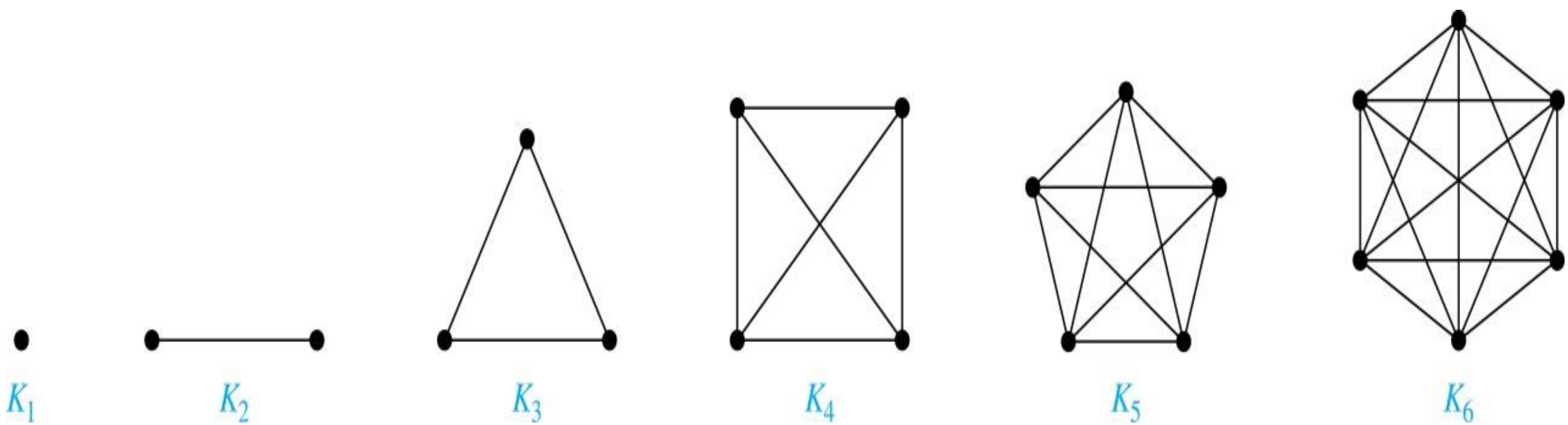
Theorem 3: Let G = (V, E) be a graph with directed edges. Then:

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

Proof: The first sum counts the number of outgoing edges over all vertices 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. ◀

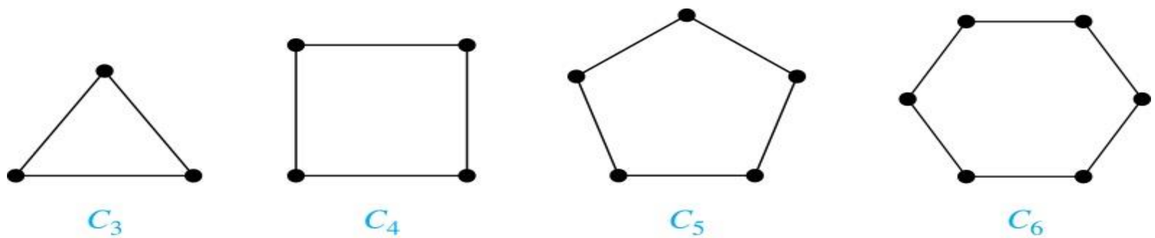
Special Types of Simple Graphs: Complete Graphs

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.

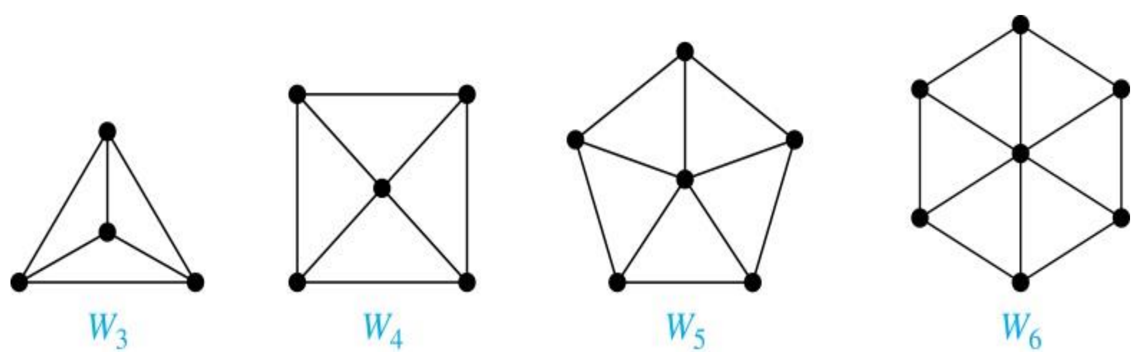


Special Types of Simple Graphs: Cycles and Wheels

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

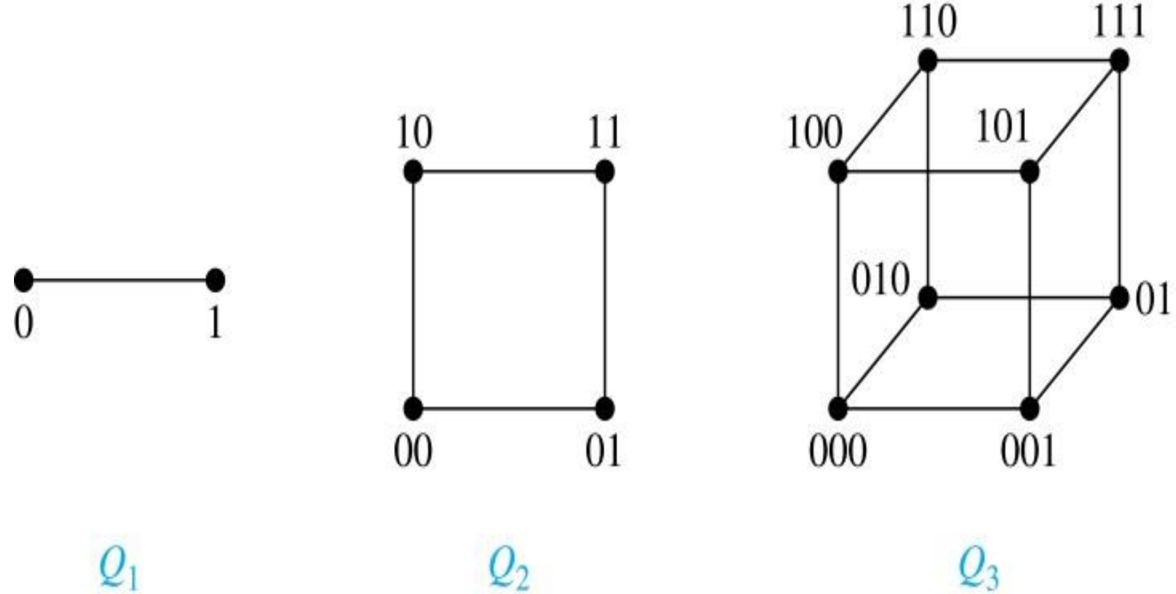


A *wheel* W_n is obtained by adding an additional vertex to a cycle C_n for $n \geq 3$ and connecting this new vertex to each of the n vertices in C_n by new edges.



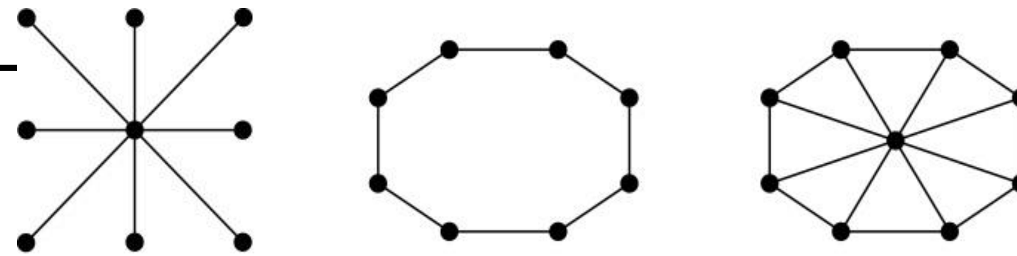
Special Types of Simple Graphs: *n*-Cubes

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.



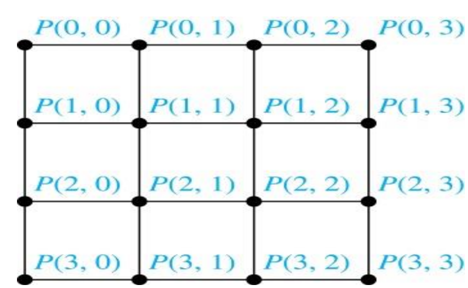
Special Types of Graphs and Computer Network Architecture

Various special graphs play an important role in the design of computer networks.



Some local area networks use a *star topology*, which is a complete bipartite graph $K_{1,n}$,as shown in (a). All devices are connected to a central control device.
Other local networks are based on a *ring topology*, where each device is connected to exactly two others using C_n ,as illustrated in (b). Messages may be sent around the ring.
Others, as illustrated in (c), use a W_n – based topology, combining the features of a star topology and a ring topology.
Various special graphs also play a role in parallel processing where processors need to be interconnected as one processor may need the output generated by another.

The *n-dimensional hypercube*, or *n-cube*, $\boldsymbol{Q_n}$, is a common way to connect processors in parallel, e.g., Intel Hypercube.
Another common method is the *mesh* network, illustrated here for 16 processors.

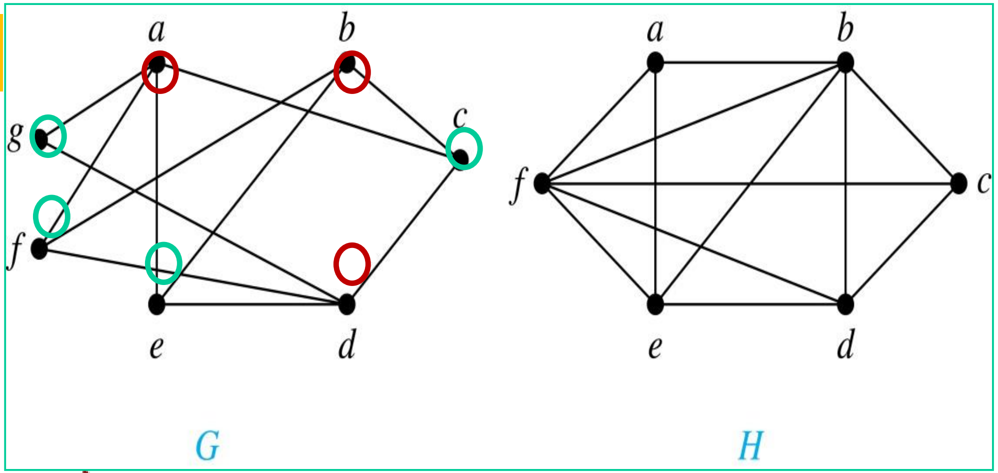


Bipartite Graphs

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. In other words, there are no edges which connect two vertices in V_1 or in V_2.

It is not hard to show that 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 the same color.

G is bipartite

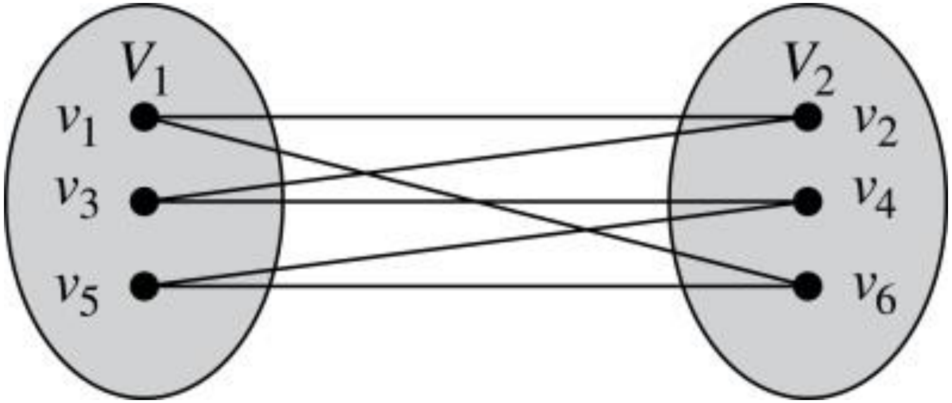


H is not bipartite since if we color a red, then the adjacent vertices f and b must both be blue.

Bipartite Graphs (*continued*)

Example: Show that C_6 is bipartite.

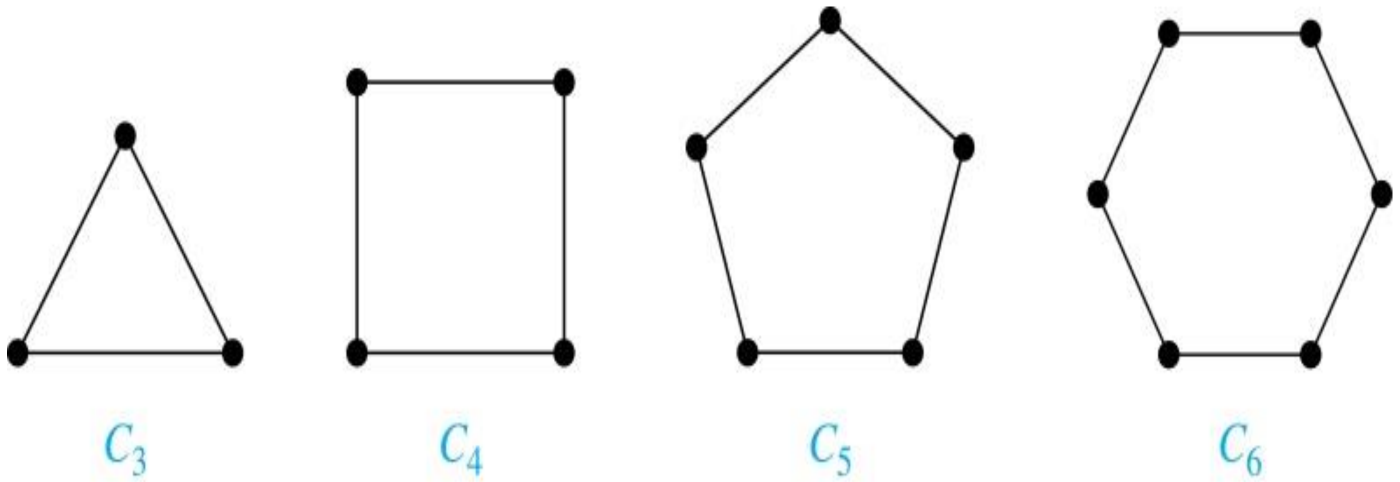
Solution: We can partition the vertex set into $V_1 = \{v_1, v_3, v_5\}$ and $V_2 = \{v_2, v_4, v_6\}$ so that every edge of C_6 connects a vertex in V_1 and V_2 .



Bipartite Graphs (*continued*)

Example: Show that C_3 is not bipartite.

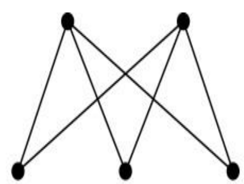
Solution: If we divide the vertex set of C_3 into two nonempty sets, one of the two must contain two vertices. But in C_3 every vertex is connected to every other vertex. Therefore, the two vertices in the same partition are connected. Hence, C_3 is not bipartite.



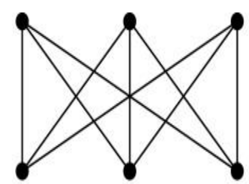
Complete Bipartite Graphs

Definition: A *complete bipartite graph* $K_{m,n}$ is 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.

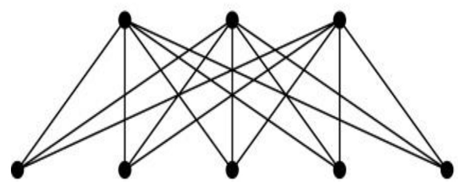
Example:
We display four complete bipartite graphs here.



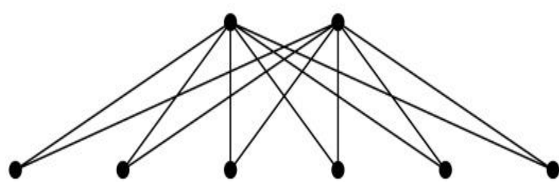
$K_{2,3}$



$K_{3,3}$



$K_{3,5}$

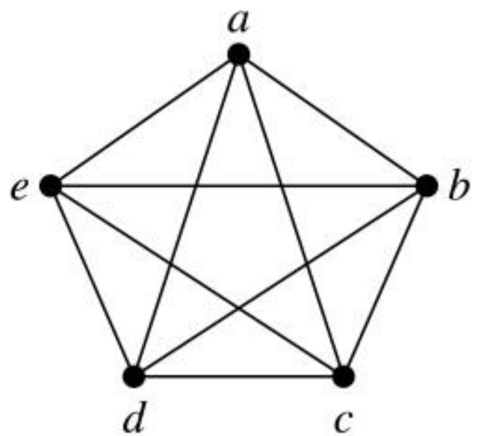


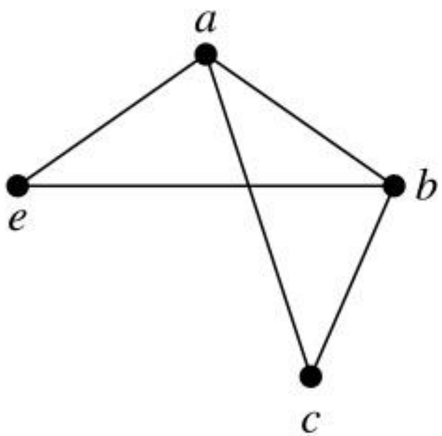
$K_{2,6}$

New Graphs from Old

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

Example: Here we show K_5 and one of its subgraphs.

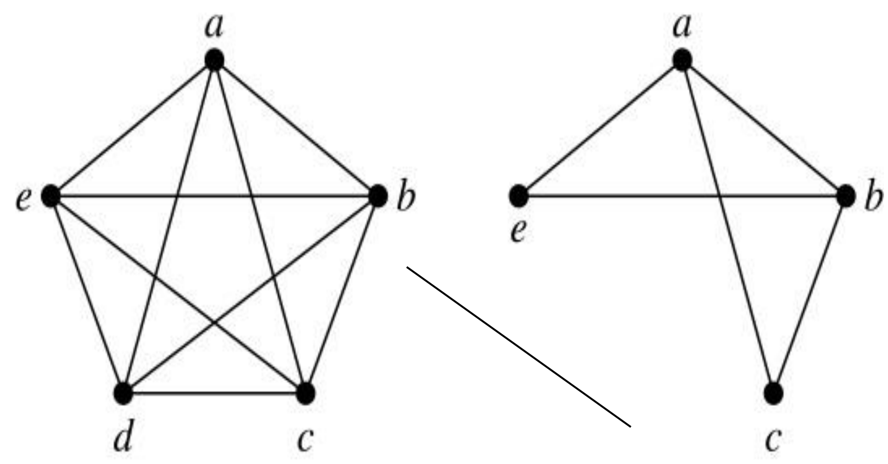




Definition: Let $G = (V, E)$ be a simple graph. The *subgraph induced* by a subset W of the vertex set V is the graph (W,F), where the edge set F contains an edge in E if and only if both endpoints are in W.

Example:

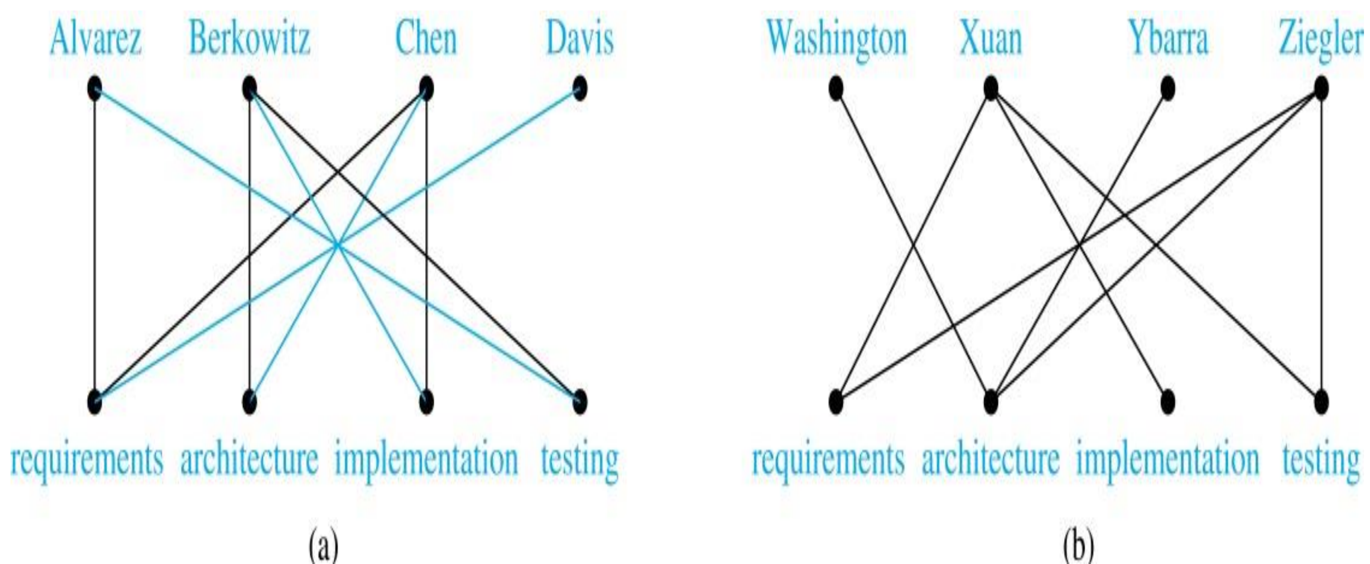
Here we show K_5 and the subgraph induced by $W = \{a,b,c,e\}$.



Bipartite Graphs and Matchings

- Bipartite graphs are used to model applications that involve matching the elements of one set to elements in another, for example:

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.



江西理工大学
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

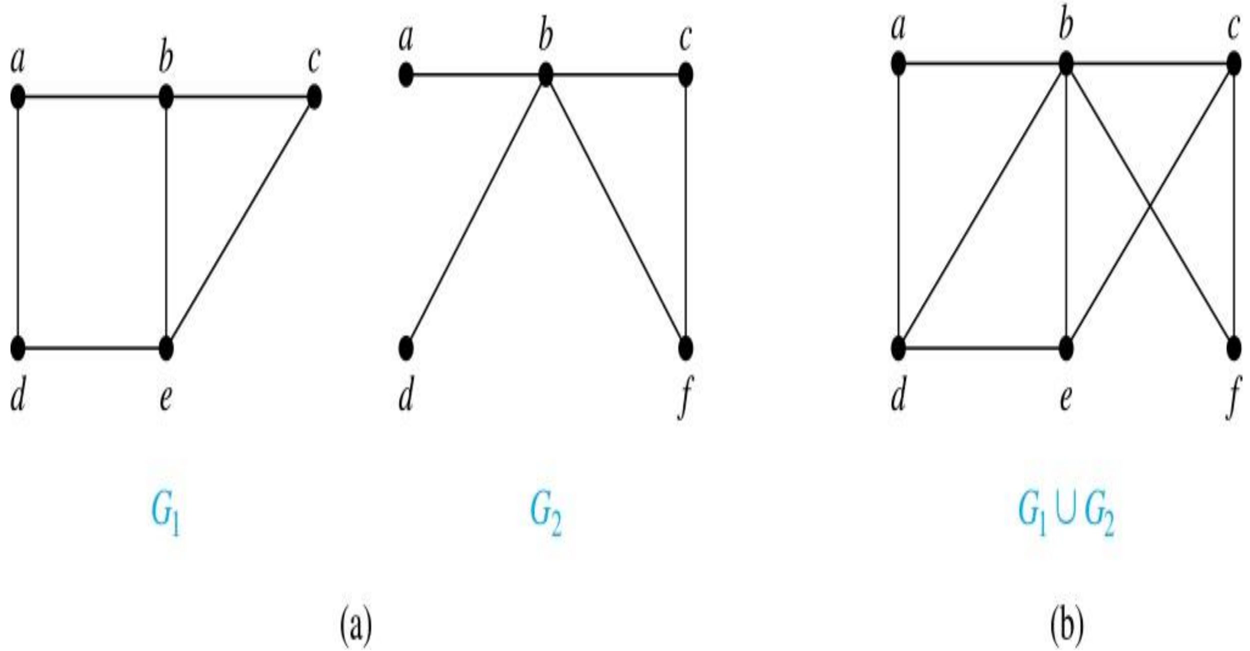
- *Marriage* - vertices represent the men and the women and edges link a a man and a woman if they are an acceptable spouse. We may wish to find the largest number of possible marriages.

See the text for more about matchings in bipartite graphs.

New Graphs from Old (*continued*)

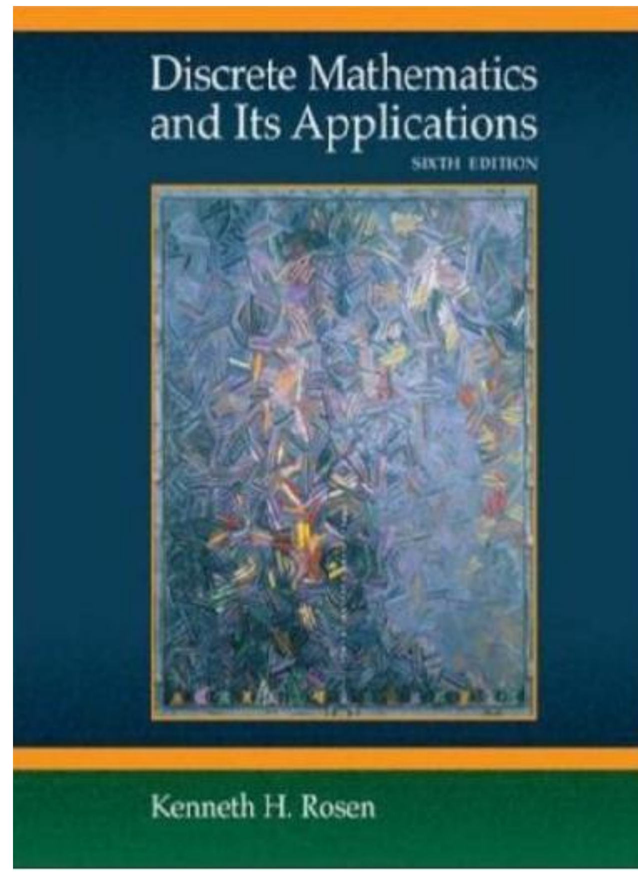
Definition: The *union* of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$. The union of G_1 and G_2 is denoted by $G_1 \cup G_2$.

Example:



Jiangxi University of Science and Technology

Section 10.3
Representing Graphs and Graph Isomorphism



江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Section Summary

- Adjacency Lists
- Adjacency Matrices
- Incidence Matrices
- Isomorphism of 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:

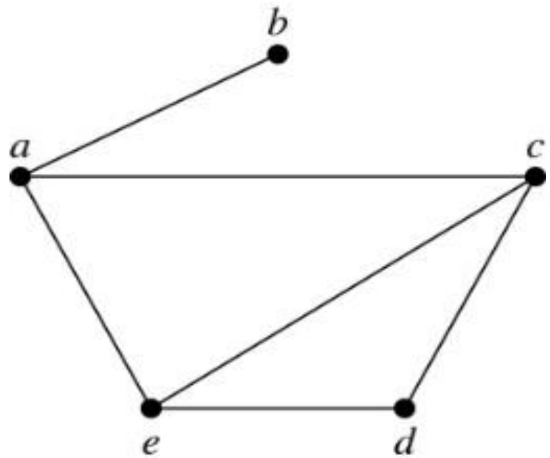


TABLE 1 An Adjacency List for a Simple Graph.

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

Example:

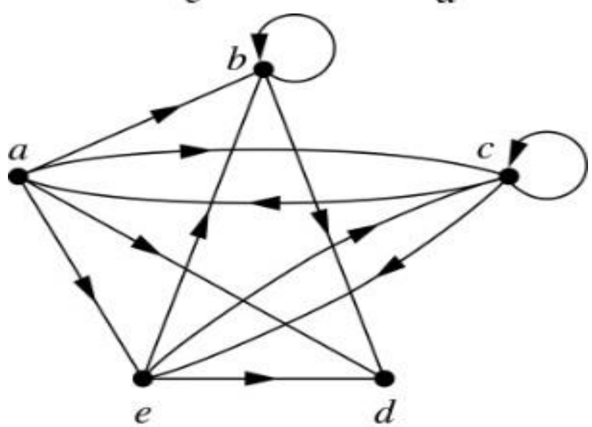


TABLE 2 An Adjacency List for a Directed Graph.

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

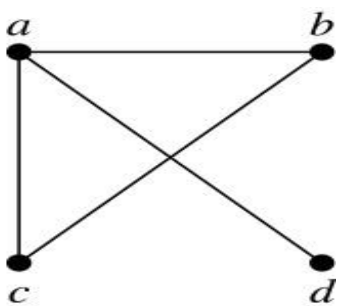
Definition: Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, with respect to the listing of vertices, is the $n \times n$ zero-one matrix with 1 as its (i, j)th entry when v_i and v_j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

– In other words, if the graphs adjacency matrix is $\mathbf{A}_G = [a_{ij}]$, then

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

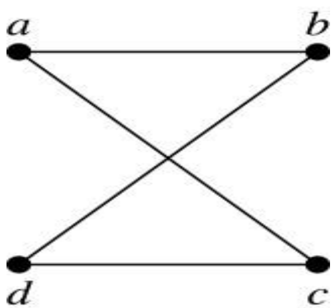
Adjacency Matrices (*continued*)

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.

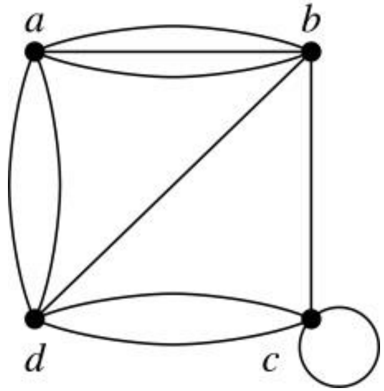
When a graph is sparse, that is, it has few edges relatively to the total number of possible edges, it is much more efficient to represent the graph using an adjacency list than an adjacency matrix. But for a dense graph, which includes a high percentage of possible edges, an adjacency matrix is preferable.

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 (*continued*)

- 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, j)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 (*continued*)

- 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, where $v_1, v_2, \ldots v_n$ is a list of the vertices.

- **In other words, if the graphs adjacency matrix is $A_G = [a_{ij}]$, 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.

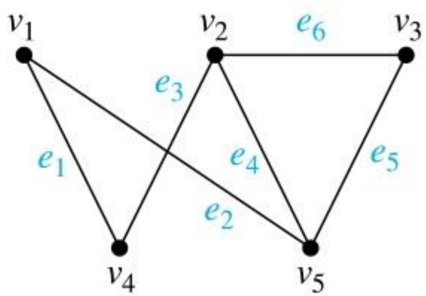
Representation of Graphs: Incidence Matrices

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 $M = [m_{ij}]$, where

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

Incidence Matrices (*continued*)

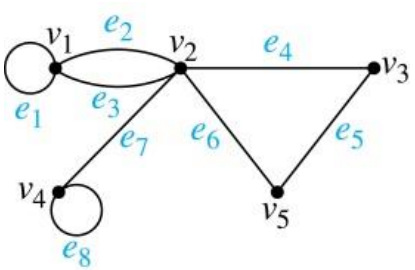
Example: Simple Graph and Incidence Matrix



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

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

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 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*. Two simple graphs that are not isomorphic are called *nonisomorphic*.

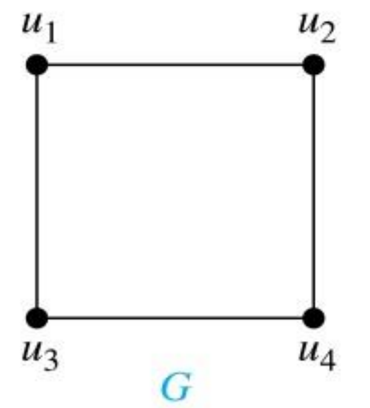
Isomorphism of Graphs (*cont.*)

Example:

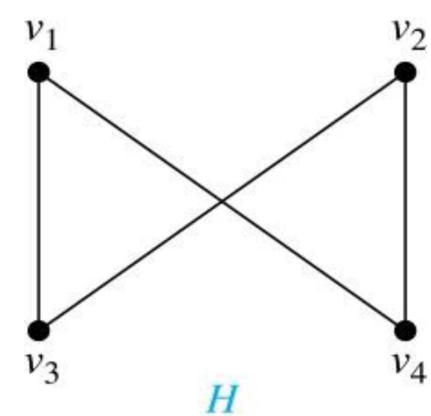
Show that the graphs $G = (V, E)$ and $H = (W, F)$ are isomorphic.

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.



Note that adjacent vertices in G are u_1 and u_2, u_1 and u_3, u_2 and u_4, and u_3 and u_4.
Each of the pairs $f(u_1) = v_1$ and $f(u_2) = v_4$, $f(u_1) = v_1$ and $f(u_3) = v_3$, $f(u_2) = v_4$ and $f(u_4) = v_2$, and $f(u_3) = v_3$ and $f(u_4) = v_2$ consists of two adjacent vertices in H.



Isomorphism of Graphs (*cont.*)

It is difficult to determine whether two simple graphs are isomorphic using brute force because there are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices.

The best algorithms for determining weather two graphs are isomorphic have exponential worst case complexity in terms of the number of vertices of the graphs.

Sometimes it is not hard to show that two graphs are not isomorphic. We can do so by finding a property, preserved by isomorphism, that only one of the two graphs has. Such a property is called *graph invariant*.

There are many different useful graph invariants that can be used to distinguish nonisomorphic graphs, such as the number of vertices, number of edges, and degree sequence (list of the degrees of the vertices in nonincreasing order).

We will encounter others in later sections of this chapter.

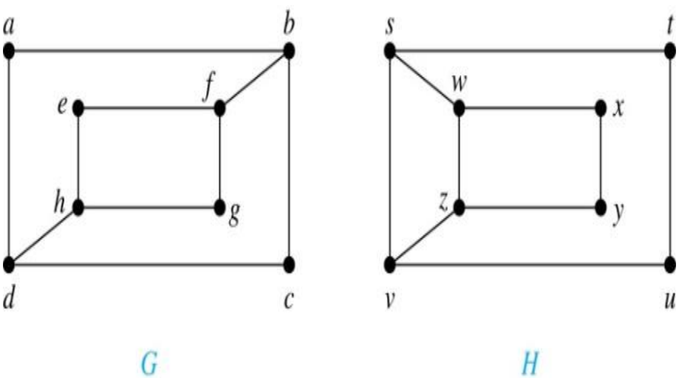
Isomorphism of Graphs (*cont.*)

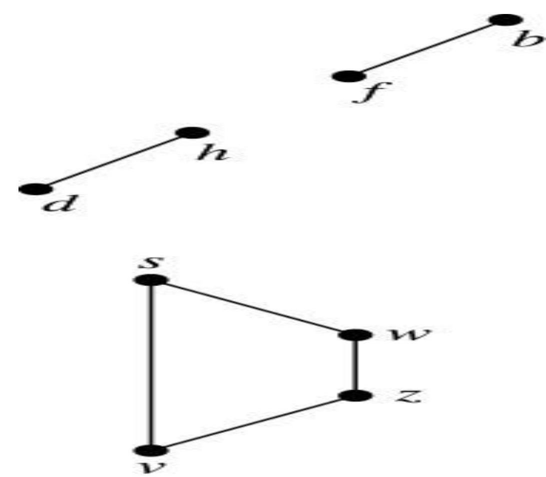
Example: Determine whether these two graphs are isomorphic.

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. Note that since $deg(a) = 2$ in G, a must correspond to t, u, x, or y in H, because these are the vertices of degree 2. But each of these vertices is adjacent to another vertex of degree two in H, which is not true for a in G.

Alternatively, note that the subgraphs of G and H made up of vertices of degree three and the edges connecting them must be isomorphic. But the subgraphs, as shown at the right, are not isomorphic.





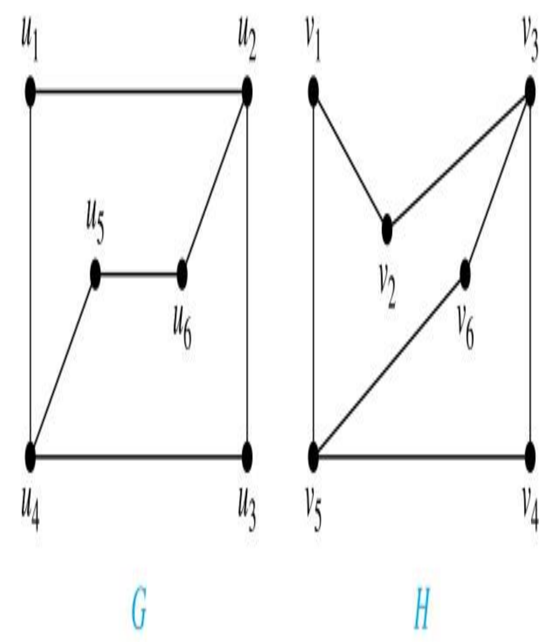
Isomorphism of Graphs (*cont.*)

Example: Determine whether these two graphs are isomorphic.

Solution: Both graphs have six vertices and seven edges.

They also both have four vertices of degree two and two of degree three.
The subgraphs of *G* and *H* consisting of all the vertices of degree two and the edges connecting them are isomorphic. So, it is reasonable to try to find an isomorphism *f*.
We define an injection *f* from the vertices of *G* to the vertices of *H* that preserves the degree of vertices. We will determine whether it is an isomorphism.

The function *f* with $f(u_1) = v_6$, $f(u_2) = v_3$, $f(u_3) = v_4$, and $f(u_4) = v_5$, $f(u_5) = v_1$, and $f(u_6) = v_2$ is a one-to-one correspondence between *G* and *H*. Showing that this correspondence preserves edges is straightforward, so we will omit the details here. Because *f* is an isomorphism, it follows that *G* and *H* are isomorphic graphs.
See the text for an illustration of how adjacency matrices can be used for this verification.



Algorithms for Graph Isomorphism

The best algorithms known for determining whether two graphs are isomorphic have exponential worst-case time complexity (in the number of vertices of the graphs).

However, there are algorithms with linear average-case time complexity.

- You can use a public domain program called NAUTY to determine in less than a second whether two graphs with as many as 100 vertices are isomoprhic.
- Graph isomorphism is a problem of special interest because it is one of a few NP problems not known to be either tractable or NP-complete (see Section 3.3).

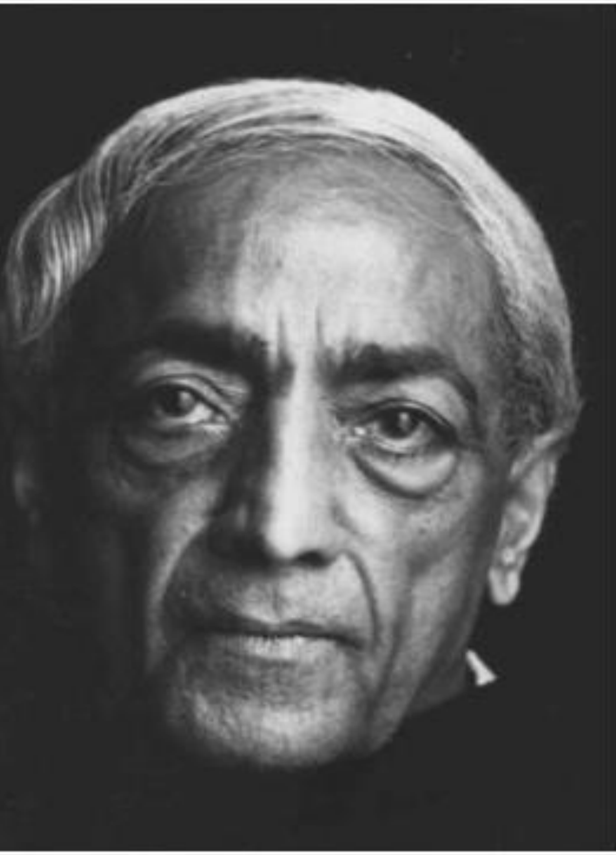
Applications of Graph Isomorphism

The question whether graphs are isomorphic plays an important role in applications of graph theory. For example,

- chemists use molecular graphs to model chemical compounds. Vertices represent atoms and edges represent chemical bonds. When a new compound is synthesized, a database of molecular graphs is checked to determine whether the graph representing the new compound is isomorphic to the graph of a compound that this already known.

Applications of Graph Isomorphism

- Electronic circuits are modeled as graphs in which the vertices represent components and the edges represent connections between them. Graph isomorphism is the basis for
    - the verification that a particular layout of a circuit corresponds to the design's original schematics.
    - determining whether a chip from one vendor includes the intellectual property of another vendor.



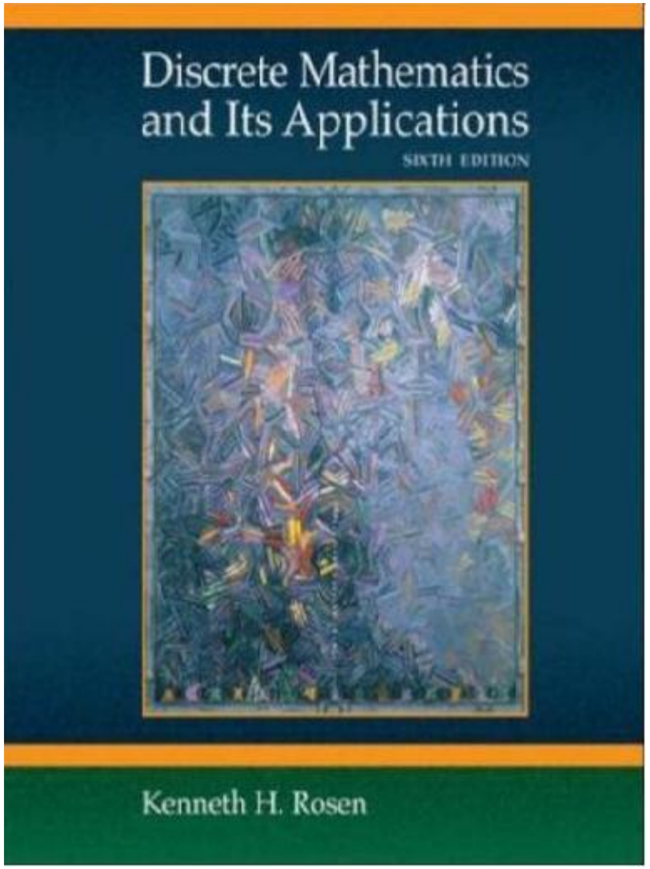
There is no end to education. It is not that you read a book, pass an examination, and finish with education. The whole of life, from the moment you are born to the moment you die, is a process of learning.

— *Jiddu Krishnamurti* —

Jiangxi University of Science and Technology

Section 10.4

Connectivity



Section Summary

- Paths
- Connectedness in Undirected Graphs
- Vertex Connectivity and Edge Connectivity (*not currently included in overheads*)
- Connectedness in Directed Graphs
- Paths and Isomorphism (*not currently included in overheads*)
- Counting Paths between Vertices

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, that is, the endpoints of these.

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 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.

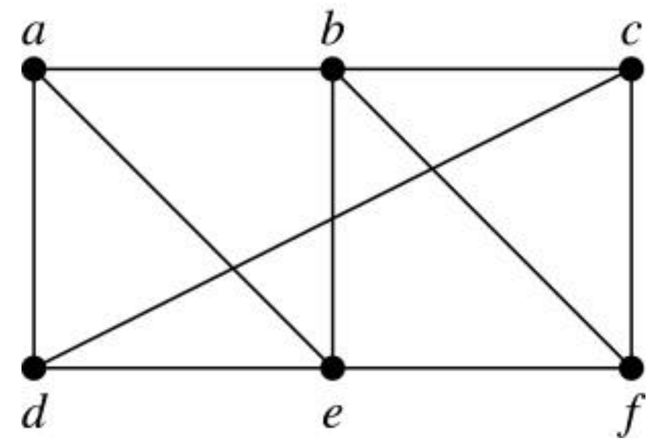
- When the graph is simple, we denote this path by its vertex sequence $x_0, x_1, \ldots, x_n$(since listing the vertices uniquely determines the path).
- The path is a *circuit* if it begins and ends at the same vertex ($u = v$) and has length greater than zero.
- The path or circuit is said to *pass 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. (*see text*)

Paths (*continued*)

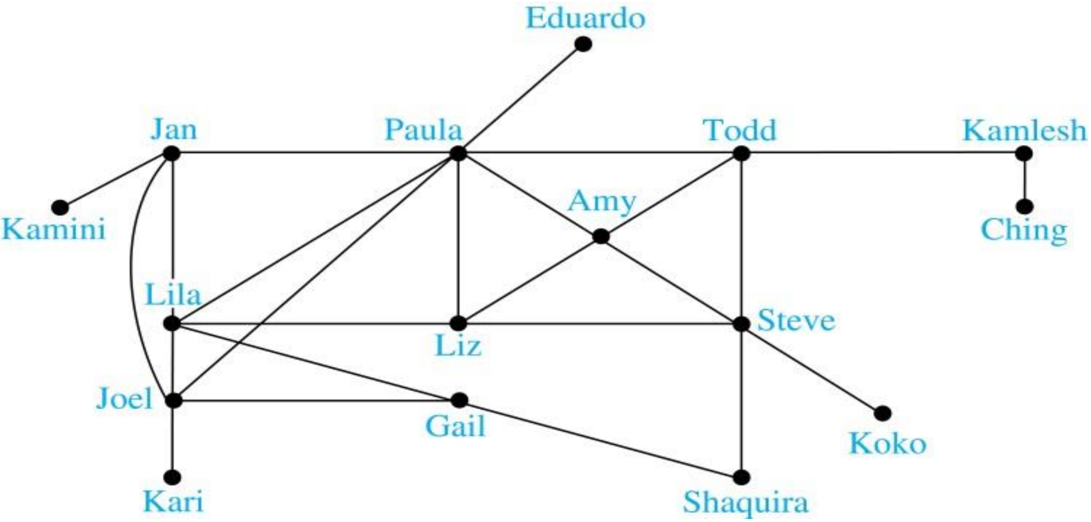
Example: In the simple graph here:

- *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.



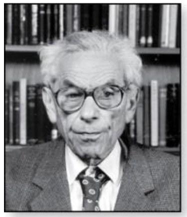
Degrees of Separation

Example: ***Paths in Acquaintanceship Graphs***. In an acquaintanceship graph there is a path between two people if there is a chain of people linking these people, where two people adjacent in the chain know one another. In this graph there is a chain of six people linking Kamini and Ching.



Some have speculated that almost every pair of people in the world are linked by a small chain of no more than six, or maybe even, five people. The play *Six Degrees of Separation* by John Guare is based on this notion.

Erdős numbers



Paul Erdős

Example: ***Erdős numbers***.
In a collaboration graph, two people *a* and *b* are connected by a path when there is a sequence of people starting with *a* and ending with *b* such that the endpoints of each edge in the path are people who have collaborated.

- In the academic collaboration graph of people who have written papers in mathematics, the *Erdős number* of a person *m* is the length of the shortest path between *m* and the prolific mathematician Paul Erdős.
  - To learn more about Erdős numbers, visit http://www.ams.org/mathscinet/collaborationDistance.html

TABLE 1 The Number of Mathematicians with a Given Erdős Number (as of early 2006).

Erdős Number	*Number of People*
0	1
1	504
2	6,593
3	33,605
4	83,642
5	87,760
6	40,014
7	11,591
8	3,146
9	819
10	244
11	68
12	23
13	5

Bacon Numbrers

- In the Hollywood graph, two actors *a* and *b* are linked when there is a chain of actors linking *a* and *b*, where every two actors adjacent in the chain have acted in the same movie.
- The *Bacon number* of an actor *c* is defined to be the length of the shortest path connecting *c* and the well-known actor Kevin Bacon. (Note that we can define a similar number by replacing Kevin Bacon by a different actor.)
  – The *oracle of Bacon* web site http://oracleofbacon.org/how.php provides a tool for finding Bacon numbers.

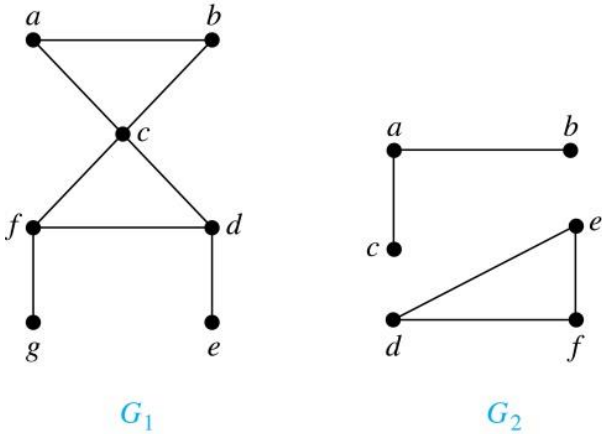
TABLE 2 The Number of Actors with a Given Bacon Number (as of early 2011).

Bacon Number	*Number of People*
0	1
1	2,367
2	242,407
3	785,389
4	200,602
5	14,048
6	1,277
7	114
8	16

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.

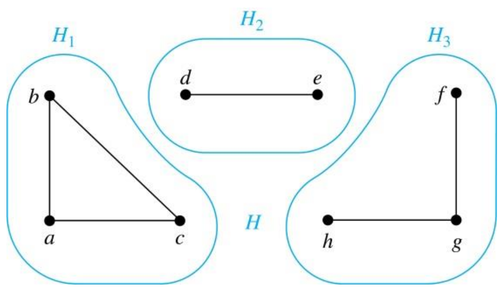
Example: G_1 is connected because there is a path between any pair of its vertices, as can be easily seen. However G_2 is not connected because there is no path between vertices a and f, for example.



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, none of which are proper subgraphs of a larger connected subgraph of G.These three subgraphs are the connected components of H.

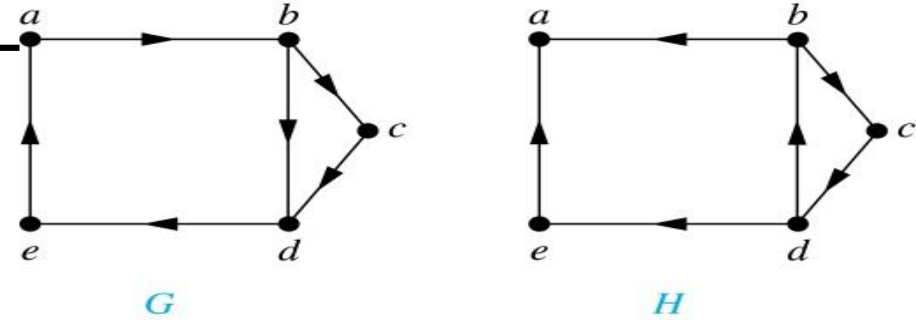


Connectedness in Directed Graphs

Definition: A directed graph is *strongly connected* if there is a path from *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 of the directed graph.

Connectedness in Directed Graphs (*continued*)



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 not strongly connected, since there is no directed path from *a* to *b*, but it is weakly connected.

Definition: The subgraphs of a directed graph *G* that are strongly connected but not contained in larger strongly connected subgraphs, that is, the maximal strongly connected subgraphs, are called the *strongly connected components* or *strong components* of *G*.

Example (*continued*): The graph *H* has three strongly connected components, consisting of the vertex *a*; the vertex *e;* and the subgraph consisting of the vertices *b*, *c*, *d* and edges (*b*,*c*), (*c*,*d*), and (*d*,*b*).

The Connected Components of the Web Graph

- Recall that at any particular instant the web graph provides a snapshot of the web, where vertices represent web pages and edges represent links. According to a 1999 study, the Web graph at that time had over 200 million vertices and over 1.5 billion edges. (The numbers today are several orders of magnitude larger.)
- The underlying undirected graph of this Web graph has a connected component that includes approximately 90% of the vertices.
- There is a *giant strongly connected component (GSCC)* consisting of more than 53 million vertices. A Web page in this component can be reached by following links starting in any other page of the component. There are three other categories of pages with each having about 44 million vertices:
  - pages that can be reached from a page in the GSCC, but do not link back.
  - pages that link back to the GSCC, but can not be reached by following links from pages in the GSCC.
  - pages that cannot reach pages in the GSCC and can not be reached from pages in the GSCC

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 $v_1, \ldots, v_n$ of vertices (with directed or undirected edges, multiple edges and loops allowed). 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 $\mathbf{A}^r$.

Proof by mathematical induction:

Basis Step: By definition of the adjacency matrix, the number of paths from v_i to v_j of length 1 is the (i,j)th entry of **A**.

Inductive Step: For the inductive hypothesis, we assume that that the (i,j)th entry of $\mathbf{A}^r$ is the number of different paths of length r from v_i to v_j.

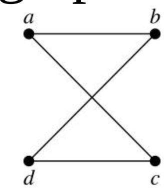
– Because $\mathbf{A}^{r+1} = \mathbf{A}^r\,\mathbf{A}$, the (i,j)th entry of $\mathbf{A}^{r+1}$ equals $b_{i1}a_{1j} + b_{i2}a_{2j} + \cdots + b_{in}a_{nj}$, where b_{ik} is the (i,k)*th* entry of $\mathbf{A}^r$. By the inductive hypothesis, b_{ik} is the number of paths of length r from v_i to v_k.

– A path of length $r + 1$ from v_i to v_j is made up of a path of length r from v_i to some v_k, and an edge from v_k to v_j. By the product rule for counting, the number of such paths is the product of the number of paths of length r from v_i to v_k (i.e., b_{ik}) and the number of edges from from v_k to v_j (i.e, a_{kj}). The sum over all possible intermediate vertices v_k *is* $b_{i1}a_{1j} + b_{i2}a_{2j} + \cdots + b_{in}a_{nj}$.

Counting Paths between Vertices (*continued*)

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

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 (ordering the vertices as a, b, c, d) is given above. 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:

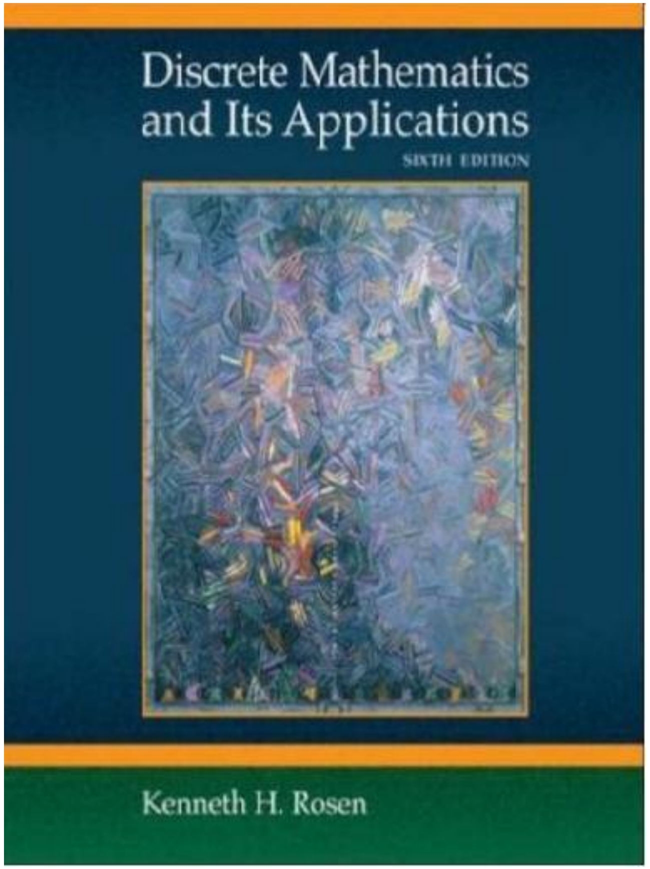
$$\mathbf{A}^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

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

Jiangxi University of Science and Technology

Euler and Hamiltonian Graphs

Section 10.5



江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

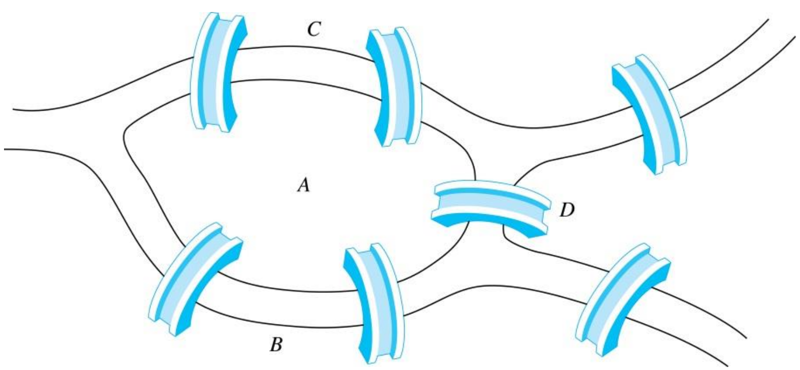
Section Summary

- Euler Paths and Circuits
- Hamilton Paths and Circuits
- Applications of Hamilton Circuits

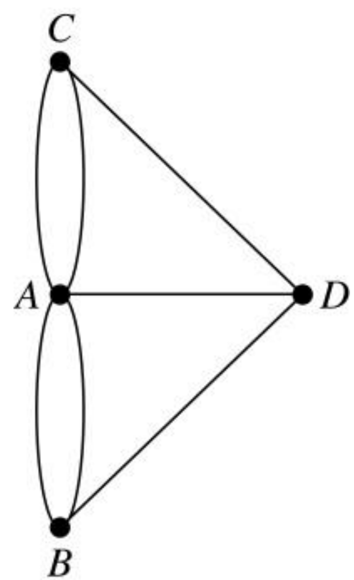
Euler Paths and Circuits

Leonard Euler (1707-1783)

- The town of Königsberg, Prussia (now Kalingrad, Russia) was divided into four sections by the branches of the Pregel river. In the 18th century seven bridges connected these regions.
- People wondered whether whether it was possible to follow a path that crosses each bridge exactly once and returns to the starting point.
- The Swiss mathematician Leonard Euler proved that no such path exists. This result is often considered to be the first theorem ever proved in graph theory.



The 7 Bridges of Königsberg



Multigraph Model of the Bridges of Königsberg

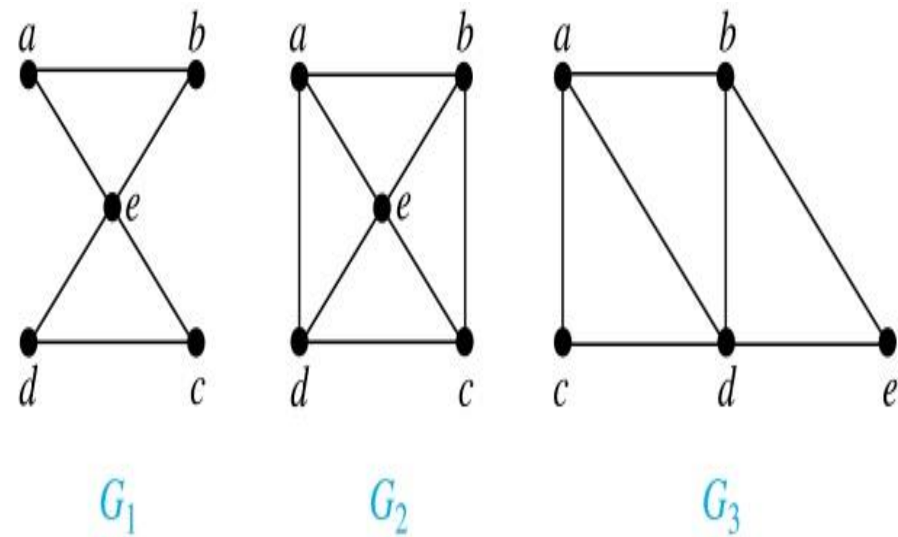
江西理工大学
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Euler Paths and Circuits (*continued*)

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 G_1, G_2, and G_3 has a Euler circuit? Of those that do not, which has an Euler path?

Solution:
The graph G_1 has an Euler circuit (e.g., *a*, *e*, *c*, *d*, *e*, *b*, *a*). But, as can easily be verified by inspection, neither G_2 nor G_3 has an Euler circuit. Note that G_3 has an Euler path (e.g., *a*, *c*, *d*, *e*, *b*, *d*, *a*, *b*), but there is no Euler path in G_2, which can be verified by inspection.



Necessary Conditions for Euler Circuits and Paths

An Euler circuit begins with a vertex a and continues with an edge incident with a, say $\{a, b\}$. The edge $\{a, b\}$ contributes one to $\deg(a)$.

Each time the circuit passes through a vertex it contributes two to the vertex's degree.

Finally, the circuit terminates where it started, contributing one to $\deg(a)$. Therefore $\deg(a)$ must be even.

We conclude that the degree of every other vertex must also be even.

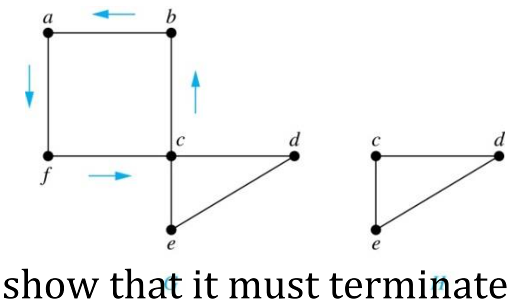
By the same reasoning, we see that the initial vertex and the final vertex of an Euler path have odd degree, while every other vertex has even degree. So, a graph with an Euler path has exactly two vertices of odd degree.

In the next slide we will show that these necessary conditions are also sufficient conditions.

Sufficient Conditions for Euler Circuits and Paths

Suppose that G is a connected multigraph with ≥ 2 vertices, all of even degree. Let $x_0 = a$ be a vertex of even degree. Choose an edge $\{x_0, x_1\}$ incident with a and proceed to build a simple path $\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{n-1}, x_n\}$ by adding edges one by one until another edge can not be added.

We illustrate this idea in the graph G here. We begin at a and choose the edges $\{a, f\}$, $\{f, c\}$, $\{c, b\}$, and $\{b, a\}$ in succession.

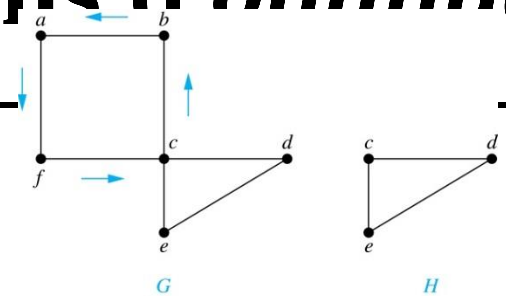


•The path begins at a with an edge of the form $\{a, x\}$; we show that it must terminate at a with an edge of the form $\{y, a\}$. Since each vertex has an even degree, there must be an even number of edges incident with this vertex. Hence, every time we enter a vertex other than a, we can leave it. Therefore, the path can only end at a.

•If all of the edges have been used, an Euler circuit has been constructed. Otherwise, consider the subgraph H obtained from G by deleting the edges already used.

In the example H consists of the vertices c, d, e.

Sufficient Conditions for Euler Circuits and Paths (*continued*)



•Because G is connected, H must have at least one vertex in common with the circuit that has been deleted.

In the example, the vertex is *c*.

• Every vertex in H must have even degree because all the vertices in *G* have even degree and for each vertex, pairs of edges incident with this vertex have been deleted. Beginning with the shared vertex construct a path ending in the same vertex (as was done before). Then splice this new circuit into the original circuit.

In the example, we end up with the circuit *a, f, c, d, e, c, b, a*.

•Continue this process until all edges have been used. This produces an Euler circuit. Since every edge is included and no edge is included more than once.

•Similar reasoning can be used to show that a graph with exactly two vertices of odd degree must have an Euler path connecting these two vertices of odd degree

Algorithm for Constructing an Euler Circuits

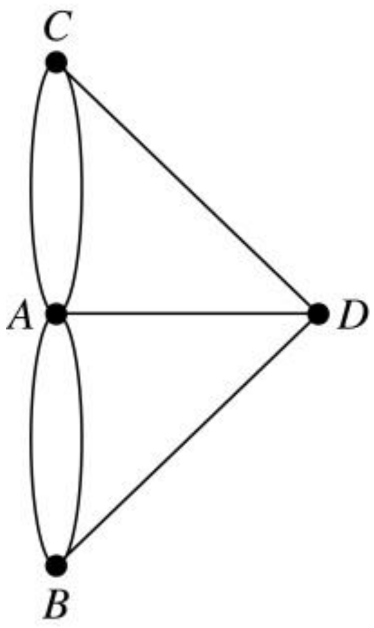
In our proof we developed this algorithms for constructing a Euler circuit in a graph with no vertices of odd degree.

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 for Euler Circuits and Paths (*continued*)

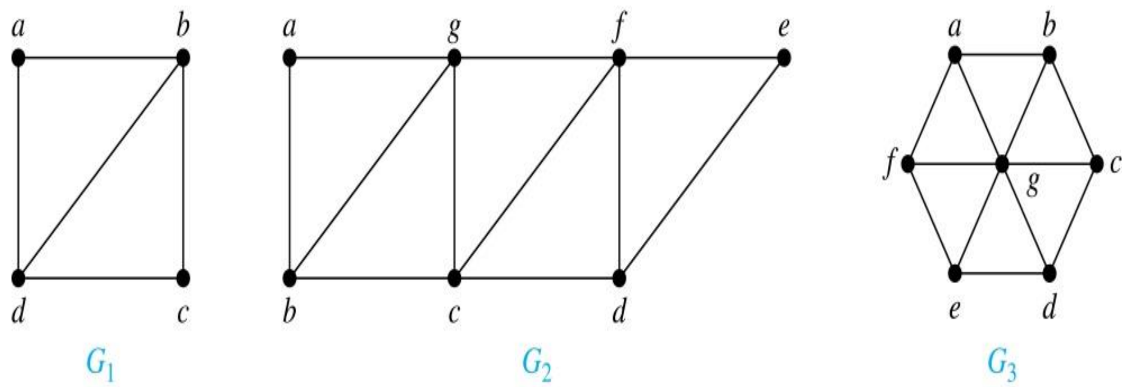
Theorem: A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has an even degree and it has an Euler path if and only if it has exactly two vertices of odd degree.

Example: Two of the vertices in the multigraph model of the Königsberg bridge problem have odd degree. Hence, there is no Euler circuit in this multigraph and it is impossible to start at a given point, cross each bridge exactly once, and return to the starting point.



Euler Circuits and Paths

Example:



G_1 contains exactly two vertices of odd degree (b and d). Hence it has an Euler path, e.g., d, a, b, c, d, b.

G_2 has exactly two vertices of odd degree (b and d). Hence it has an Euler path, e.g., $b, a, g, f, e, d, c, g, b, c, f, d$.

G_3 has six vertices of odd degree. Hence, it does not have an Euler path.

Applications of Euler Paths and Circuits

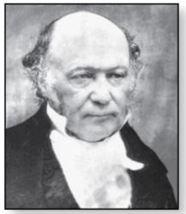
Euler paths and circuits can be used to solve many practical problems such as finding a path or circuit that traverses each

- street in a neighborhood,
- road in a transportation network,
- connection in a utility grid,
- link in a communications network.

Other applications are found in the

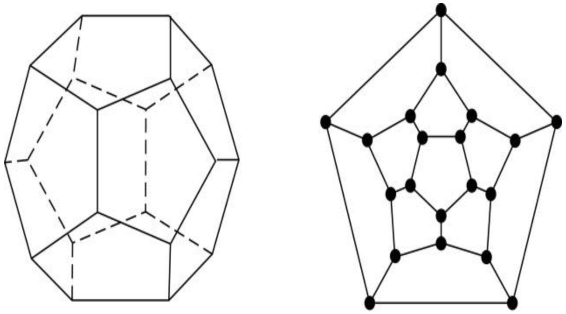
- layout of circuits,
- network multicasting,
- molecular biology, where Euler paths are used in the sequencing of DNA.

Hamilton Paths and Circuits



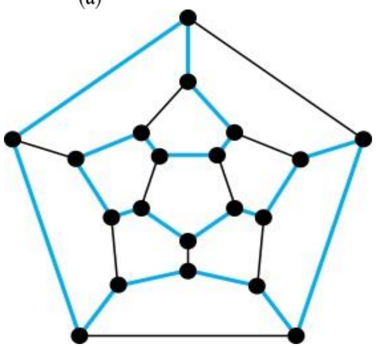
William Rowan Hamilton (1805- 1865)

- Euler paths and circuits contained every edge only once. Now we look at paths and circuits that contain every vertex exactly once.
- William Hamilton invented the *Icosian puzzle* in 1857. It consisted of a wooden dodecahedron (with 12 regular pentagons as faces), illustrated in (a), with a peg at each vertex, labeled with the names of different cities. String was used to used to plot a circuit visiting 20 cities exactly once
- The graph form of the puzzle is given in (b).



(a) (b)

- The solution (a Hamilton circuit) is given here.



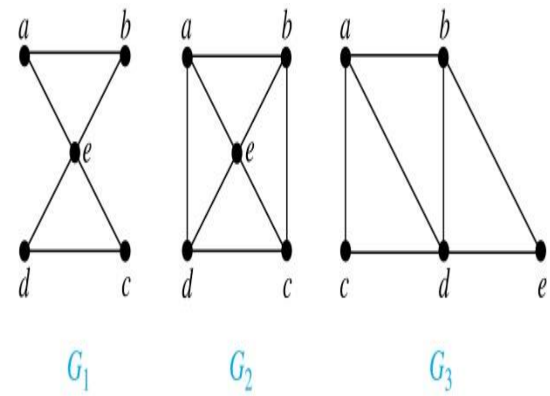
Hamilton Paths and Circuits

Definition: A simple path in a graph G that passes through every vertex exactly once is called a *Hamilton path*, and a simple circuit in a graph G that passes through every vertex exactly once is called a *Hamilton circuit*.

That is, a simple path $x_0, x_1, ..., x_{n-1}, x_n$ in the graph $G = (V, E)$ is called a Hamilton path if $V = \{x_0, x_1, \ldots, x_{n-1}, x_n\}$ and $x_i \neq x_j$ for $0 \leq i < j \leq n$, and the simple circuit $x_0, x_1, ..., x_{n-1}, x_n, x_0$ (with $n > 0$) is a Hamilton circuit if $x_0, x_1, \ldots, x_{n-1}, x_n$ is a Hamilton path.

Hamilton Paths and Circuits (*continued*)

Example: Which of these simple graphs has a Hamilton circuit or, if not, a Hamilton path?



Solution: G_1 has a Hamilton circuit: a, b, c, d, e, a.
G_2 does not have a Hamilton circuit (Why?), but does have a Hamilton path : a, b, c, d.
G_3 does not have a Hamilton circuit, or a Hamilton path. Why?

Necessary Conditions for Hamilton Circuits

- Unlike for an Euler circuit, no simple necessary and sufficient conditions are known for the existence of a Hamiton circuit.
- However, there are some useful necessary conditions. We describe two of these now.

Dirac's Theorem: If G is a simple graph with $n \geq 3$ vertices such that the degree of every vertex in G is $\geq n/2$, then G has a Hamilton circuit.

Ore's Theorem: If G is a simple graph with $n \geq 3$ vertices such that $\deg(u) + \deg(v) \geq n$ for every pair of nonadjacent vertices, then G has a Hamilton circuit.

Gabriel Andrew Dirac (1925-1984)

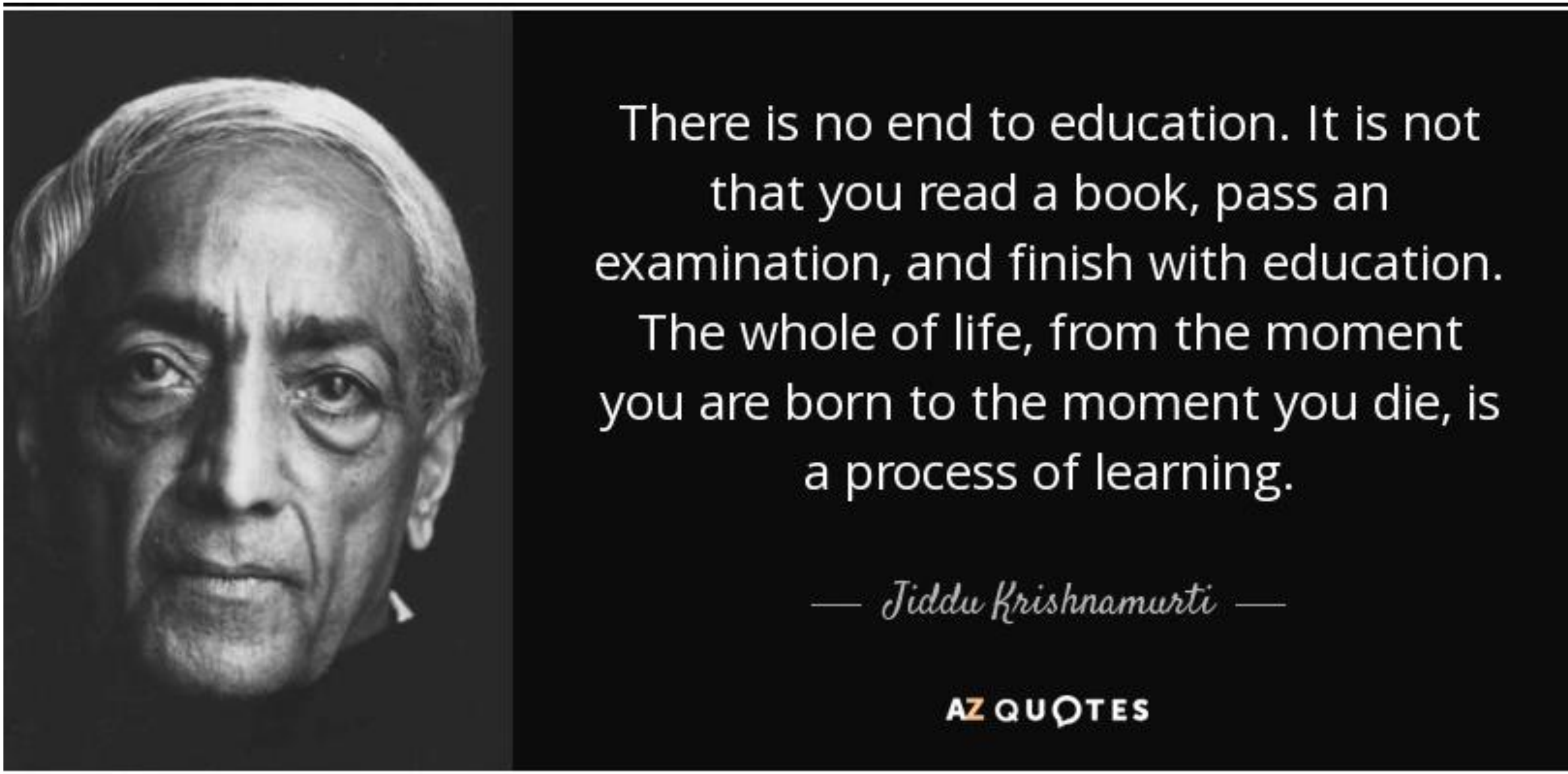
Øysten Ore (1899-1968)

Applications of Hamilton Paths and Circuits

Applications that ask for a path or a circuit that visits each intersection of a city, each place pipelines intersect in a utility grid, or each node in a communications network exactly once, can be solved by finding a Hamilton path in the appropriate graph.

The famous *traveling salesperson problem* (*TSP*) asks for the shortest route a traveling salesperson should take to visit a set of cities. This problem reduces to finding a Hamilton circuit such that the total sum of the weights of its edges is as small as possible.

A family of binary codes, known as *Gray codes*, which minimize the effect of transmission errors, correspond to Hamilton circuits in the n-cube Q_n. (*See the text for details*.)


There is no end to education. It is not that you read a book, pass an examination, and finish with education. The whole of life, from the moment you are born to the moment you die, is a process of learning.
— Jiddu Krishnamurti —
AZ QUOTES