Graphs

Chapter 10

Chapter Summary

- Graphs and Graph Models
- Graph Terminology and Special Types of Graphs
- Representing Graphs and Graph Isomorphism
- Connectivity
- Euler and Hamiltonian Graphs
- Shortest-Path Problems (*not currently included in overheads*)
- Planar Graphs (*not currently included in overheads*)
- Graph Coloring (*not currently included in overheads*)

Graphs and Graph Models

Section 10.1

Section Summary

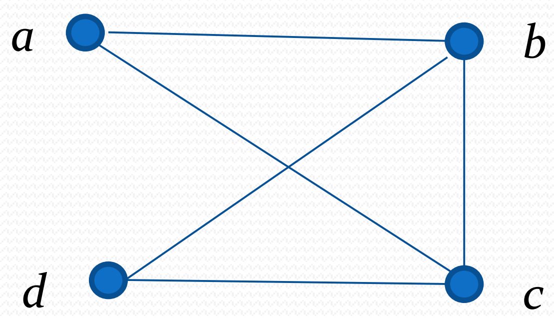
- Introduction to Graphs
- Graph Taxonomy
- Graph Models

Graphs

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

Example:

This is a graph with four vertices and five edges.



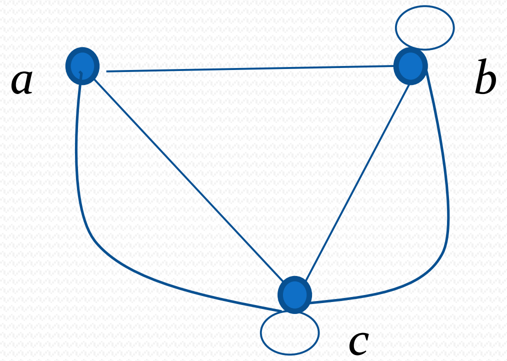
Remarks:

- The graphs we study here are unrelated to graphs of functions studied in Chapter 2.
- We have a lot of freedom when we draw a picture of a graph. All that matters is the connections made by the edges, not the particular geometry depicted. For example, the lengths of edges, whether edges cross, how vertices are depicted, and so on, do not matter
- A graph with an infinite vertex set is called an *infinite graph*. A graph with a finite vertex set is called a *finite graph*. We (following the text) restrict our attention to finite graphs.

Some Terminology

- In a *simple graph* each edge connects two different vertices and no two edges connect the same pair of vertices.
- *Multigraphs* may have multiple edges connecting the same two vertices. When m different edges connect the vertices u and v, we say that $\{u,v\}$ is an edge of *multiplicity* m.
- An edge that connects a vertex to itself is called a *loop*.
- A *pseudograph* may include loops, as well as multiple edges connecting the same pair of vertices.

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



Remark: There is no standard terminology for graph theory. So, it is crucial that you understand the terminology being used whenever you read material about graphs.

Directed Graphs

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

Remark:

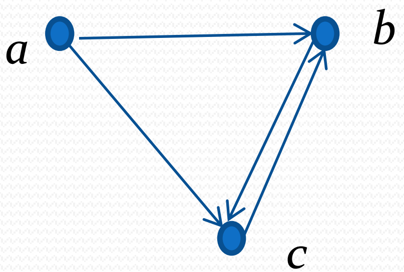
- Graphs where the end points of an edge are not ordered are said to be *undirected graphs*.

Some Terminology (*continued*)

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

Example:

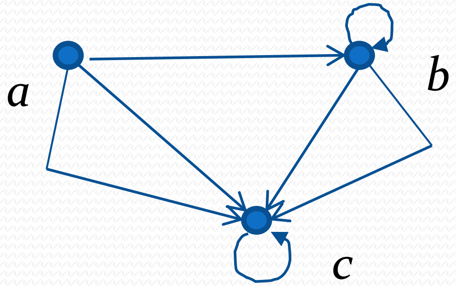
This is a directed graph with three vertices and four edges.



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

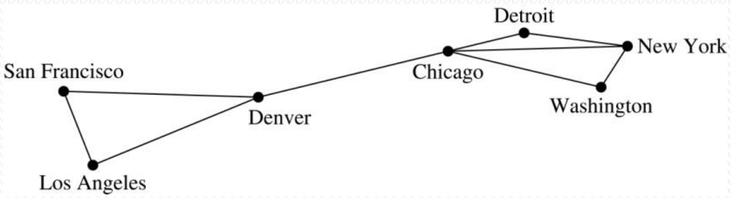
Example:

In this directed multigraph the multiplicity of (a,b) is 1 and the multiplicity of (b,c) is 2.



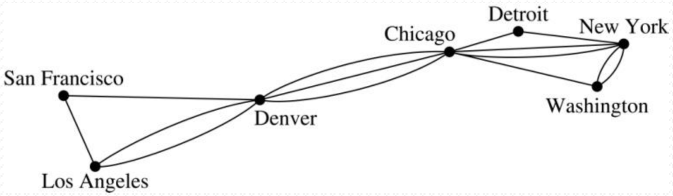
Graph Models: Computer Networks

- When we build a graph model, we use the appropriate type of graph to capture the important features of the application.
- We illustrate this process using graph models of different types of computer networks. In all these graph models, the vertices represent data centers and the edges represent communication links.
- To model a computer network where we are only concerned whether two data centers are connected by a communications link, we use a simple graph. This is the appropriate type of graph when we only care whether two data centers are directly linked (and not how many links there may be) and all communications links work in both directions.

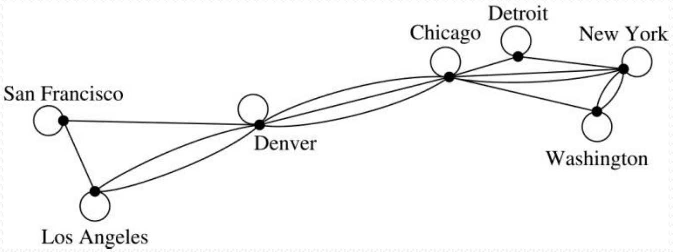


Graph Models: Computer Networks (*continued*)

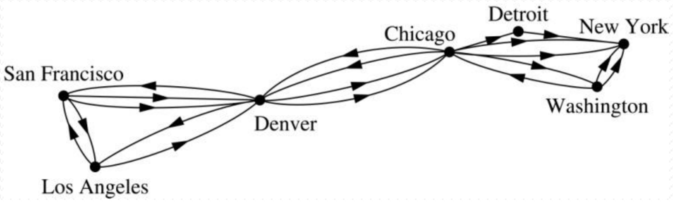
- To model a computer network where we care about the number of links between data centers, we use a multigraph.



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



- To model a network with multiple one-way links, we use a directed multigraph. Note that we could use a directed graph without multiple edges if we only care whether there is at least one link from a data center to another data center.



Graph Terminology: Summary

- To understand the structure of a graph and to build a graph model, we ask these questions:
  - Are the edges of the graph undirected or directed (or both)?
  - If the edges are undirected, are multiple edges present that connect the same pair of vertices? If the edges are directed, are multiple directed edges present?
  - Are loops present?

TABLE 1 **Graph Terminology.**

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

Other Applications of Graphs

- We will illustrate how graph theory can be used in models of:
  - Social networks
  - Communications networks
  - Information networks
  - Software design
  - Transportation networks
  - Biological networks
- It's a challenge to find a subject to which graph theory has not yet been applied. Can you find an area without applications of graph theory?

Graph Terminology and Special Types of Graphs

Section 10.2

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

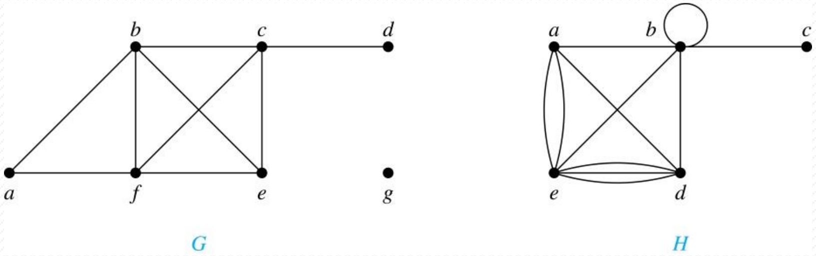
Definition 1. Two vertices u, v in an undirected graph G are called *adjacent* (or *neighbors*) in G if there is an edge e between u and v. Such an edge e is called *incident with* the vertices u and v and e is said to *connect* u and v.

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?



Solution:

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

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

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 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

even

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

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.

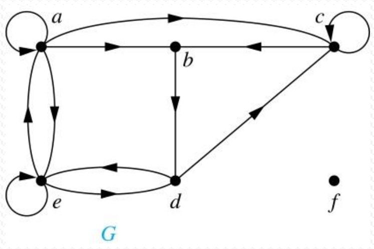
Definition: An *directed graph* $G = (V, E)$ consists of V, a nonempty set of *vertices* (or *nodes*), and E, a set of *directed edges* or *arcs*. Each edge is an ordered pair of vertices. The directed edge (u,v) is said to start at u and end at v.

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



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

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

Directed Graphs (*continued*)

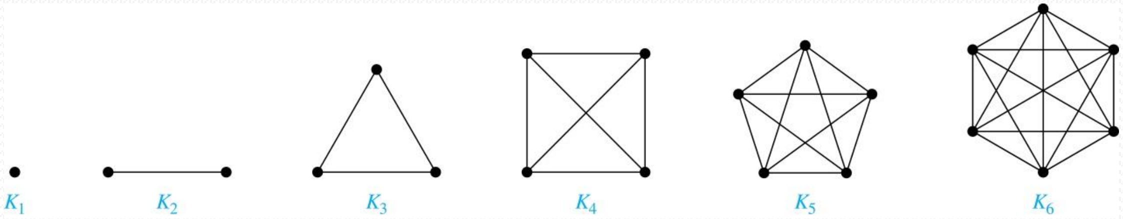
Theorem 3: Let $G = (V, E)$ be a graph with directed edges. Then:

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

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. ◄

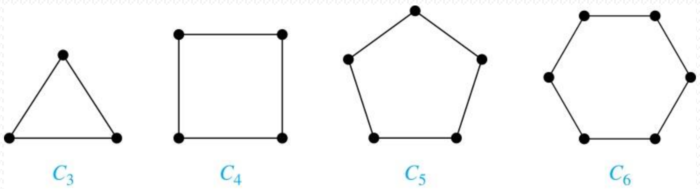
Special Types of Simple Graphs: Complete Graphs

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.

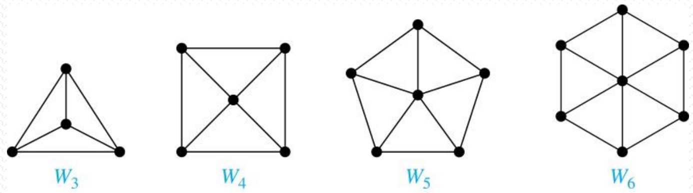


Special Types of Simple Graphs: Cycles and Wheels

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

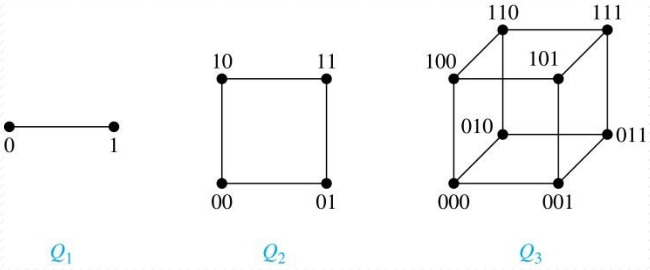


A *wheel* W_n is obtained by adding an additional vertex to a cycle C_n for $n \geq 3$ and connecting this new vertex to each of the n vertices in C_n by new edges.



Special Types of Simple Graphs: *n*-Cubes

An *n-dimensional hypercube*, or *n-cube*, $\mathbf{Q}_n$, 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.

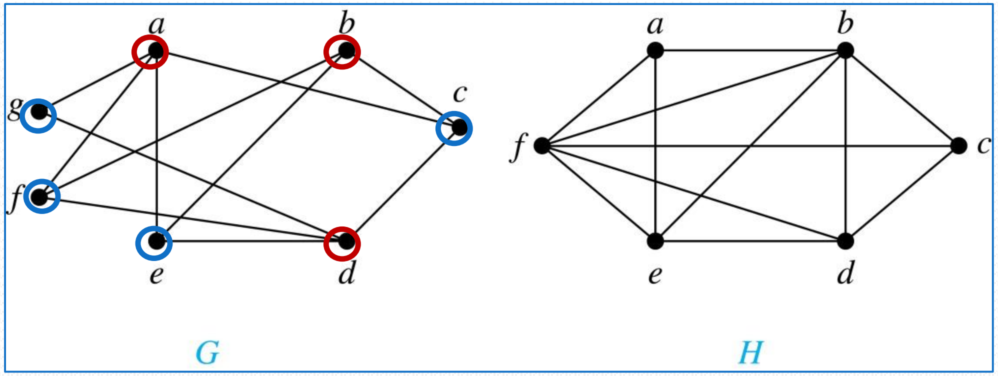


Bipartite Graphs

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

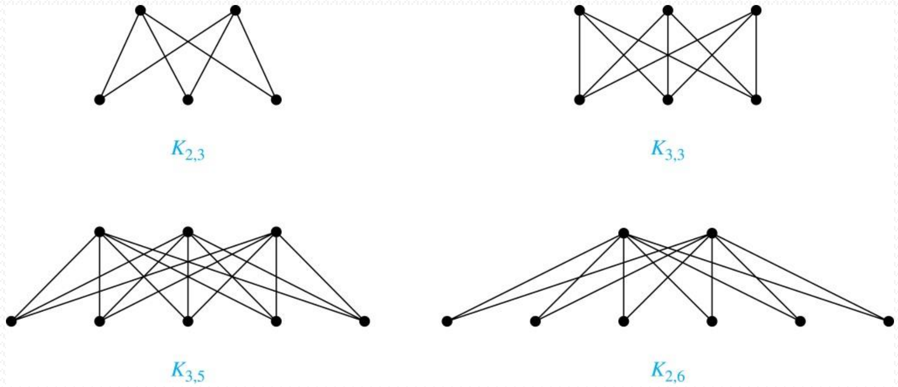


H is not bipartite since if we color a red, then the adjacent vertices f and b must both be blue.

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.

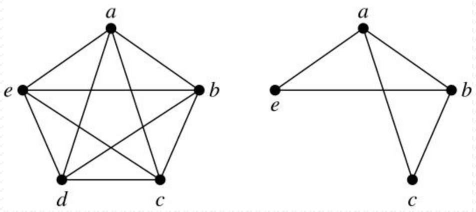
Example: We display four complete bipartite graphs here.



New Graphs from Old

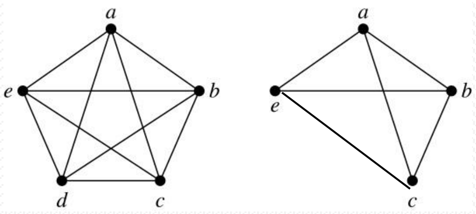
Definition: A *subgraph of a graph* $G = (V,E)$ is a graph (W,F), where $W \subset V$ and $F \subset E$. A subgraph H of G is a proper subgraph of G if $H \neq G$.

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.

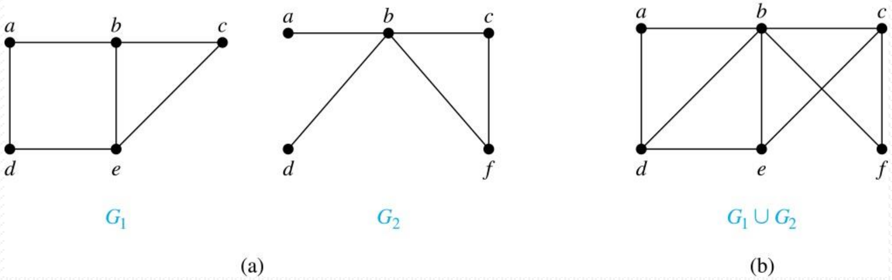
Example: Here we show K_5 and the subgraph induced by $W = \{a,b,c,e\}$.



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:



Representing Graphs and Graph Isomorphism

Section 10.3

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:

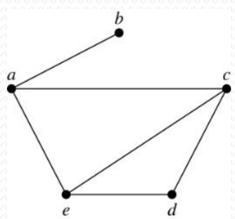


TABLE 1 An Adjacency List for a Simple Graph.

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

Example:

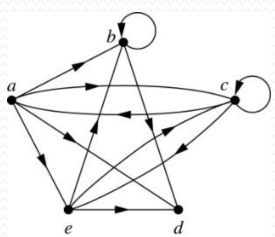


TABLE 2 An Adjacency List for a Directed Graph.

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

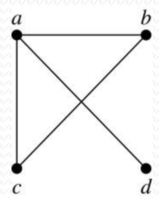
Definition: Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, with respect to the listing of vertices, is the $n \times n$ zero-one matrix with 1 as its (i, j)th entry when v_i and v_j 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

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

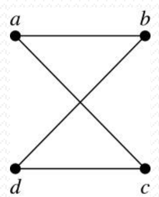
Adjacency Matrices (*continued*)

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.

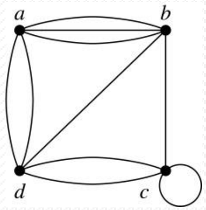
When a graph is sparse, that is, it has few edges relatively to the total number of possible edges, it is much more efficient to represent the graph using an adjacency list than an adjacency matrix. But for a dense graph, which includes a high percentage of possible edges, an adjacency matrix is preferable.

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*.



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

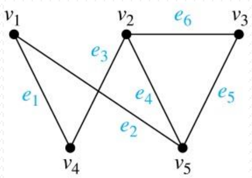
Representation of Graphs: Incidence Matrices

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

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

Incidence Matrices (*continued*)

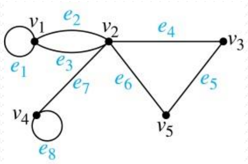
Example: Simple Graph and Incidence Matrix



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

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



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

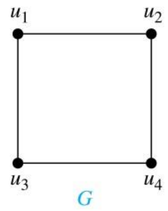
The rows going from top to bottom represent v_1 through v_5 and the columns going from left to right represent e_1 through e_8.

Isomorphism of Graphs

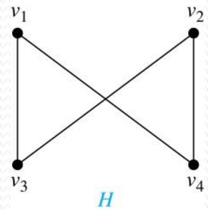
Definition: The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a one-to-one and onto function f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2 , for all a and b in V_1 . Such a function f is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

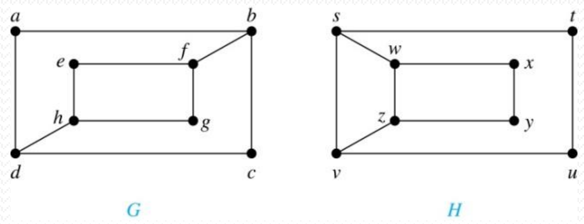


Solution: The function f with $f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, and $f(u_4) = v_2$ is a one-to-one correspondence between V and W. Note that adjacent vertices in G are u_1 and u_2, u_1 and u_3, u_2 and u_4, and u_3 and u_4. Each of the pairs $f(u_1) = v_1$ and $f(u_2) = v_4$, $f(u_1) = v_1$ 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.*)

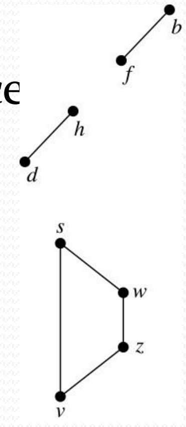
Example: Determine whether these two graphs are isomorphic.



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

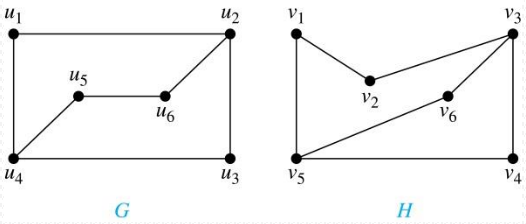
However, *G* and *H* are not isomorphic. Note that since $deg(a) = 2$ in *G*, *a* must correspond to *t*, *u*, *x*, or *y* in H, because these are the vertices of degree 2. But each of these vertices is adjacent to another vertex of degree two in *H*, which is not true for *a* in *G*.

Alternatively, note that the subgraphs of *G* and *H* made up of vertice 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.

Connectivity

Section 10.4

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.

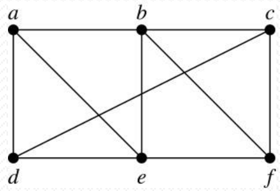
Paths

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*)



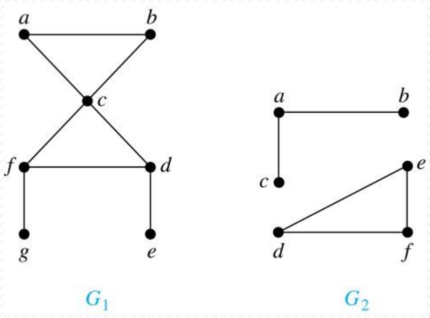
Example: In the simple graph here:

- *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.

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.

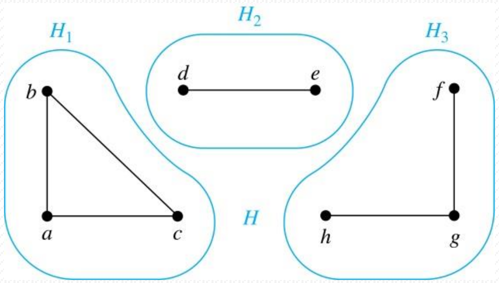
Example: G_1 is connected because there is a path between any pair of its vertices, as can be easily seen. However G_2 is not connected because there is no path between vertices a and f, for example.



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.

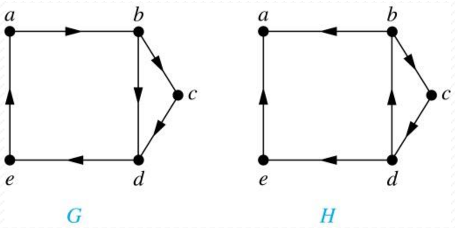


Connectedness in Directed Graphs

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*.

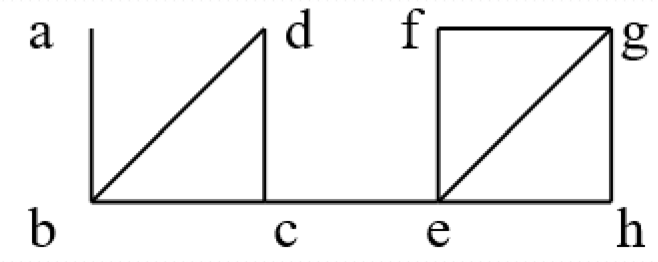
Example (*continued*): The graph *H* has three strongly connected components, consisting of the vertex *a*; the vertex *e;* and the subgraph consisting of the vertices *b*, *c*, *d* and edges (*b*,*c*), (*c*,*d*), and (*d*,*b*).

Cut vertices and Bridges

- Let G be a connected graph. A vertex v in G is called a cut vertex if G-v is connected. Sometimes the removal of a vertex and all edges incident with it produces a subgraph with more connected components than in the original graph. Such vertices are called cut vertices (or articulation points). The removal of a cut vertex from a connected graph produces a subgraph that is not connected.

Cut vertices and Bridges (*continued*)

- An edge e of G is called a bridge if G –e is disconnected. An edge whose removal produces a graph with more connected components than in the original graph is called a cut edge or bridge.

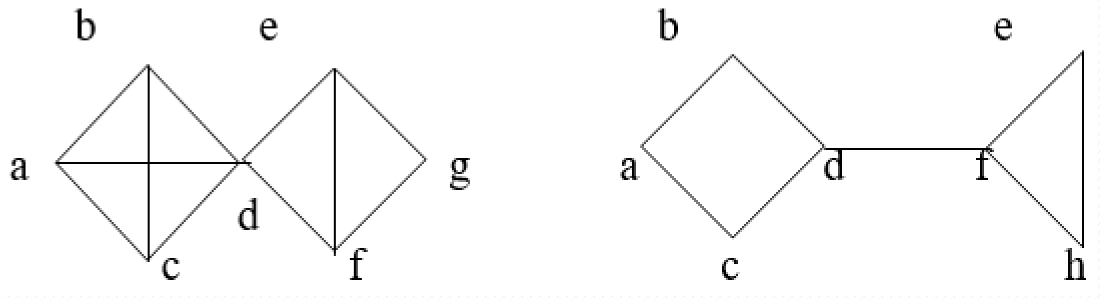


Cut vertices- b, c, e
Cut edges- {a, b}, {c, e}

Distance and Diameter

- Let G be a connected graph. The distance between vertices u and v in G, written d (u, v) is the length of the shortest path between u and v.
- The diameter of G, written diam (G), is the maximum distance between any two points in G.

Distance and Diameter (*continued*)



G1	G2
d (a, f)= 2 diam (G1)=3	d (a , f)=3 diam (G2)=4

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**.

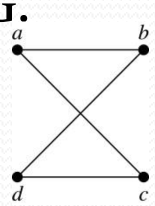
Inductive Step: For the inductive hypothesis, we assume that that the (i,j)th entry of $\mathbf{A}^r$ is the number of different paths of length r from v_i to v_j.

- 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

Euler and Hamiltonian Graphs

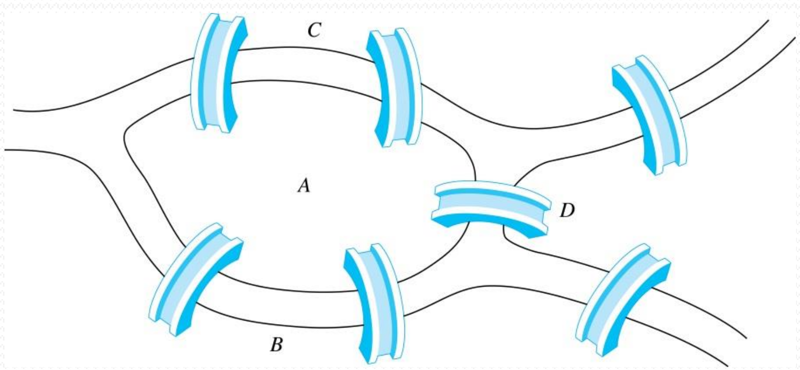
Section 10.5

Section Summary

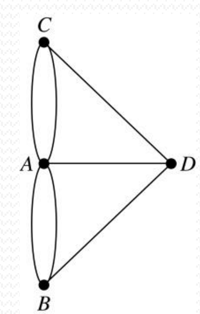
- Euler Paths and Circuits
- Hamilton Paths and Circuits
- Applications of Hamilton Circuits

Euler Paths and Circuits

- 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

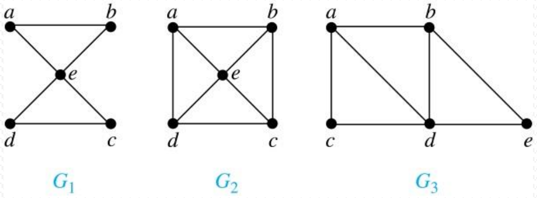


Multigraph Model of the Bridges of Königsberg

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.

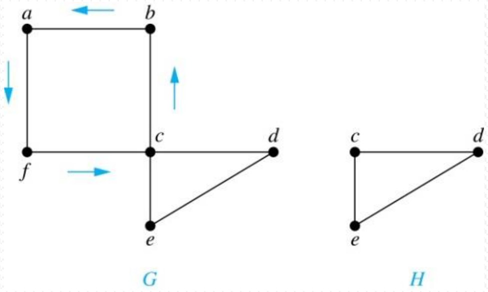
Necessary Conditions for Euler Circuits and Paths

- 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.

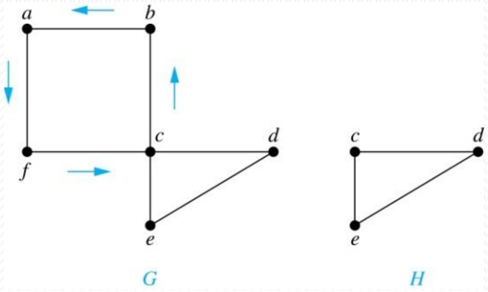
We illustrate this idea in the graph G here. We begin at a and choose the edges $\{a, f\}$, $\{f, c\}$, $\{c, b\}$, and $\{b, a\}$ in succession.



- 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

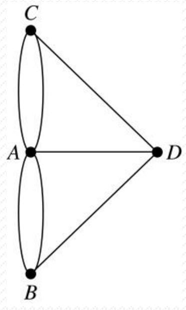
In our proof we developed this algorithms for constructing a Euler circuit in a graph with no vertices of odd degree.

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*)

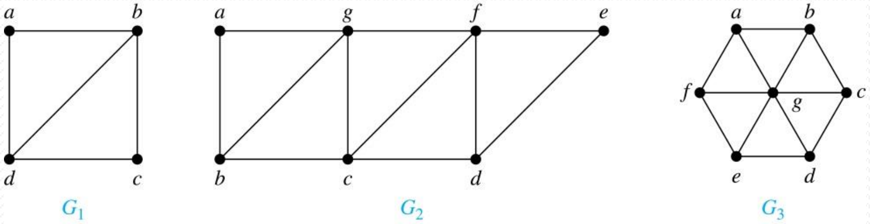
Theorem: A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has an even degree and it has an Euler path if and only if it has exactly two vertices of odd degree.

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.

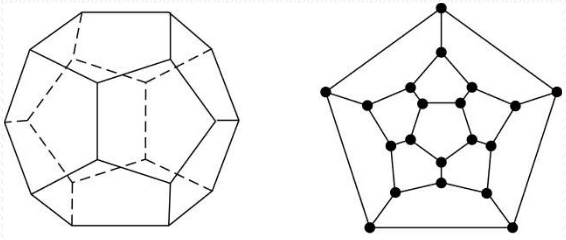
G_2 has exactly two vertices of odd degree (b and d). Hence it has an Euler path, e.g., $b, a, g, f, e, d, c, g, b, c, f, d$.

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

William Rowan Hamilton (1805- 1865)

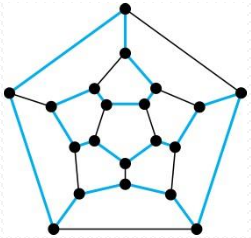
Hamilton Paths and Circuits

- 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.



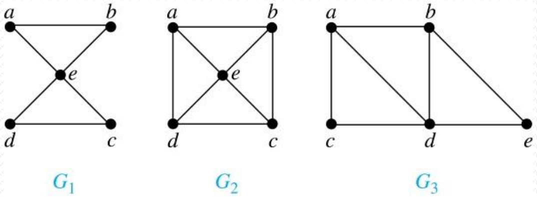
Hamilton Paths and Circuits

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 $x_0, x_1, \ldots, x_{n-1}, x_n$ 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 $x_0, x_1, \ldots, x_{n-1}, x_n, x_0$ (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?

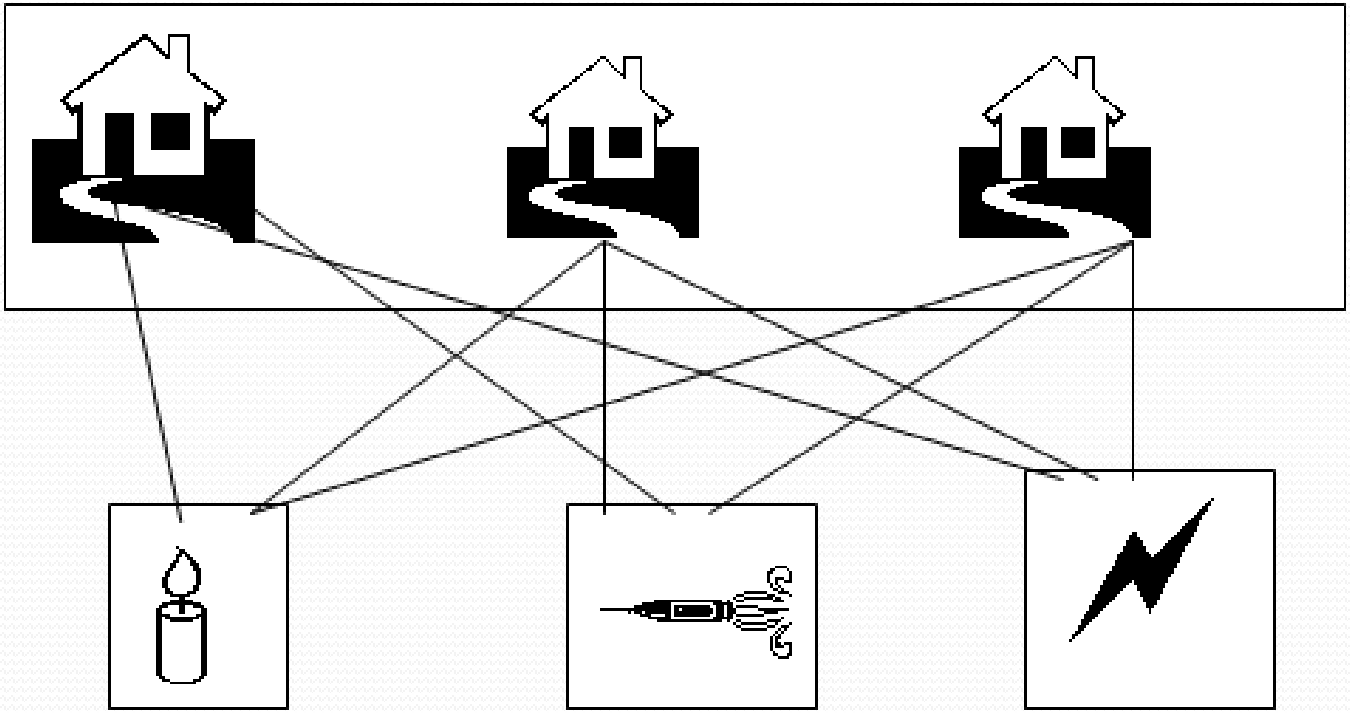
Shortest Path Problems

- Many problems can be modeled using graphs with weights assigns to their edges. As an illustration, consider how an airline system can be modeled. We set up the basic graph model by representing cities by vertices and flights by edges. Problems involving distances can be modeled by assigning distances between cities to the edges.
- Weighted graph: Graphs that have a number assigned to each edge are called weighted graph.
- Length: To be more specific, let the length of a path in a weighted graph be the sum of the weights of the edges of this path.

- Algorithm : **Dijkstra's algorithm,**
-
- **Procedure** Dijkstra(G: weighted connected simple graph, with all weights positive)
- {G has vertices a= v_o, v_1, ..., v_n = z and weights w(v_i, v_J)
- where w(v_i, v_J) =∝ if {v_i, v_J } is not an edge inG}
- **for** i := 1**to** n
- L(v_i) := ∝
- L(a):= 0
- S:= {the table are now initialized so that the label of a is zero and all other labels are ∝,and sis empty set }
- **while** z ∉S
- **begin**
- u:=a vertex not in S with L(u)minimal
- S:=S∪{u}
- **for** all vertices v not in S
- **if** L(u)+ w(u, v) <L(v) **then** L(v): = L(u) + w(u, v)
- {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
- **end** {L(z) =length of shortest path from a to z}

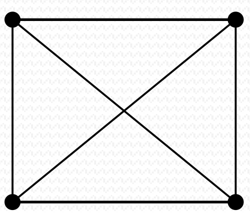
Planner Graph

- Consider the problem of joining three houses to each of three separate utilities, as shown in the following figure. Is it possible to join these houses and utilities so that none of the connection cross? This problem can be modeled using the complete bipartite graph K3, 3. The original question can be rephrased as: Can K3, 3 be drawn in the plane so that no two of its edges cross?

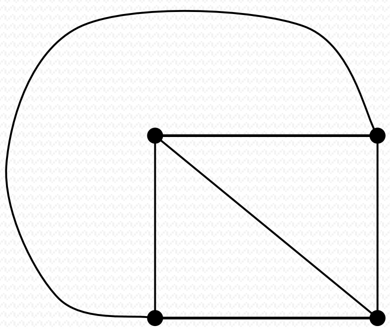


- Definition: A graph is called planner if it can be drawn in the plane without any edges crossing (where a crossing of edges is the intersection of the lines or arcs representing them at a point other then their common endpoint). Such a drawing is called a planner representing of the graph.

- Example: Is K4 (shown in the following figure with edges crossing) planner?
- Solution: K4 is planner because it can be drawn without crossing, as shown in the following figure.

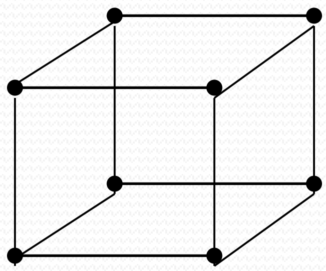


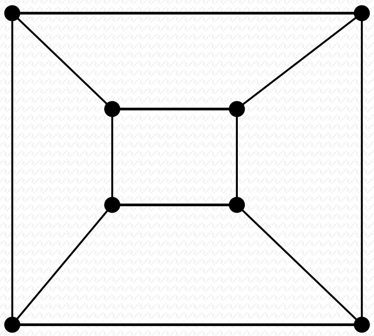
The Graph K_4



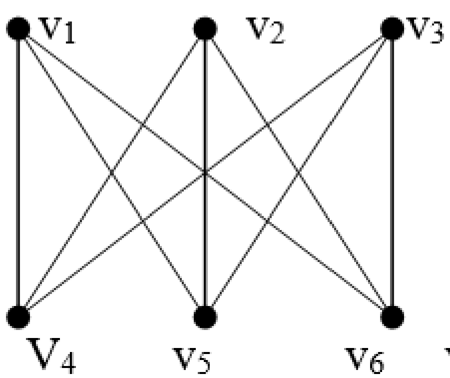
K_4 drawn with no crossing

- Example: Is Q3 shown in the following figure planner?
- Solution: Q3 is planner, because it can be drawn without any edges crossing, as shown in the following figure.

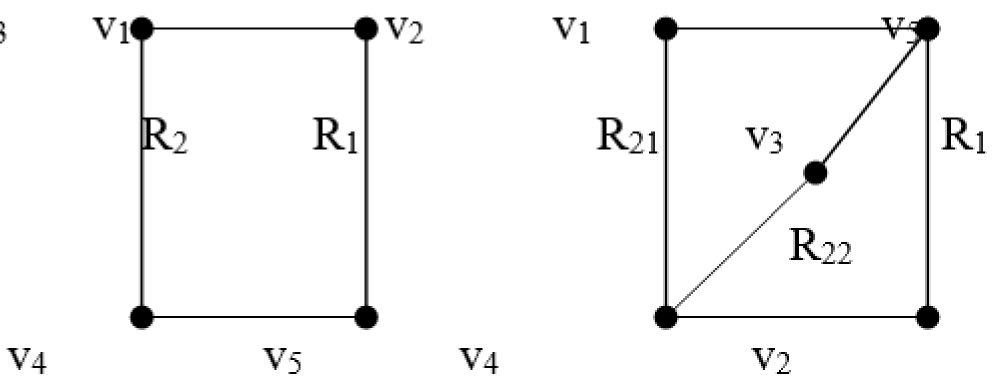




- **Example:** Is $k_{3,3}$, shown in the following figure, planner?
-
- **Solution:** Any attempt to drawn $k_{3,3}$, in the plane with no edges crossing is denoted. We now show why in any planner representation of $k_{3,3}$, the vertices $v_1$and v_2 must be connected to both v_4 and v_5. These four edges from a closed curve that splits the plane into two reigons,R_1 and R_2, as shown in figure 7(a). The vertex v_3 is in either R_1 or R_2. When v_3 is in R_2, the inside of the closed curve, the edges between v_3 and v_4 and between v_3 and $v_5$seperate R_2 into two sub regions, R_{21} and R_{22}, as shown in figure 7(b). Next note that there is no way to place the final vertex v_6 without forcing a crossing. For if v_6 is in R_1, then the edge between v_6 and v_3 cannot be drawn without a crossing. If v_6 is in R_{21}, then the edge between v_1 and v_6 cannot be drawn without a crossing. If v_6 is in R_{22}, then the edge between v_1 and v_6 cannot be drawn without a crossing.
- A similar argument can be used when $v_3$is in R_1. The completion of this argument is left for the reader. It follows that $K_{3,3}$ is not planner



The Graph $K_{3,3}$



Showing $K_{3,3}$ is non planner.

- **Euler’s Formula**
- Let G be a connected simple plannar graph with e edges and v vertices. Let r be the number of regions in a plannar representation of G. Than
- r = e-v +2
- **Example:** suppose that a connected plannar simple graph has 20 vertices, each of degree 3. Into how many regions does a representation of this plannar graph split the plane?
- **Solution:** 2e=3*20
- e=30
- r=30-20+2
- r=12
- **Corollary-1:** If g is a connected plannar graph with e edges and v vertices where v>=3 then e<=3v-6.
- **Example:** show that k5 is nonplannar using corollary 1?
- **Solution:**
- 10<=3.5-6
- 10<=9 so k5 is non plannar.
- **Corollary-2:** If g is a connected plannar graph with e edges and v vertices where v>=3 and no circuis of length 3, then e<=2v-4.
- **Example:** show that k3,3 is nonplannar using corollary 2?
- **Solution:**
- 9<=2.6-4
- 10<=8 so k3,3 is non plannar.

- **Region**
- A plannar representation of a graph splits the plane into regions, including an unbounded region.
- **Degree of region**
- The number of the edges on the boundary of the region denoted by deg(r). When an edge occurs twice on the boundary, it contributes 2 to the degree.
- In the above figure,
- Deg (r1)=3
- Deg (r2)=3
- Deg (r3)=3
- Deg (r4)=3
- Deg (r5)=3
- Deg (r6)=7

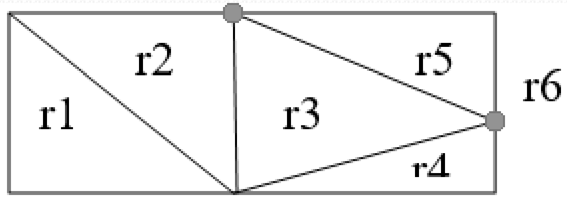


Fig: 6 regions

Graph Coloring

- **Definition:** A *coloring* of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color
- **Definition:** The *chromatic number* of a graph is the least number of colors needed for coloring of this graph.
- **The Four Color Theorem:** The chromatic number of a planner graph is no greater than four.
- ***The four color theorem*** was originally posed as a conjecture in the 1850s. It was finally proved by the American mathematicians Kenneth Appel and Wolfgang Haken in 1976. Prior to 1976, many incorrect proofs were published, often with hard-to-find errors. In addition, many futile attempts were made to construct counterexamples by drawing maps that require more than four colors.
- **Example:** What is the chromatic number of K_n?
- **Solution:** A coloring of K_n can be constructed using n colors by assigning a different color to each vertex. Is there a coloring using fewer colors? The answer is no. No two vertices can be assigned the same color, since every two vertices of this graph are adjacent. Hence, the chromatic number of K_n=n (recall that K_n is not planner when $n \geq 5$, so this does not contradict the four color theorem.) A coloring of k_5 using five colors is shown in figure.