Discrete Mathematics, KOM1062
Lecture #7

Instructor: Dr. Yavuz Eren

Lecture Book: **"Discrete Mathematics, Seventh Edt., Kenneth H. Rosen, 2007, McGraw Books Discrete Mathematics and Applications, Susanna S. Epp, Brooks, 4th Edt., 2011".**

Spring 2024

Graphs

Graphs are discrete structures consisting of vertices and edges that connect these vertices.

There are different kinds of graphs, depending on whether edges have directions, whether multiple edges can connect the same pair of vertices, and whether loops are allowed.

Problems in almost every conceivable discipline can be solved using graph models.

We will give examples to illustrate how graphs are used as models in a variety of areas. For instance, we will show
-how graphs are used to represent **the competition of different species** in an ecological niche,
-how graphs are used to represent **who influences whom** in an organization, and
-how graphs are used to represent **the outcomes of round-robin tournaments**.

We will describe how graphs can be used to model
-acquaintanceships between people,
-collaboration between researchers,
-telephone calls between telephone numbers, and
-links between websites.

Graphs

We will show how graphs can be used to model roadmaps and the assignment of jobs to employees of an organization.

Using graph models,

-we can determine whether it is possible to walk down all the streets in a city without going down a street twice, and
-we can find the number of colors needed to color the regions of a map.

Graphs can be used to determine whether a circuit can be implemented on a planar circuit board.

We can distinguish between two chemical compounds with the same molecular formula but different structures using graphs.

We can determine whether two computers are connected by a communications link using graph models of computer networks.

3

Graphs

Graphs with weights assigned to their edges can be used to solve problems such as finding the shortest path between two cities in a transportation network.

This chapter will introduce the basic concepts of graph theory and present many different graph models.

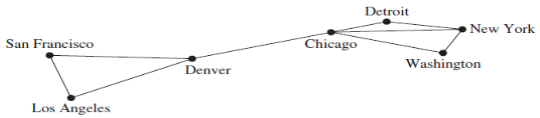
To solve the wide variety of problems that can be studied using graphs, we will introduce many different graph algorithms.

We will also study the complexity of these algorithms.

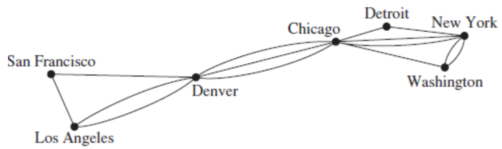
4

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

Now suppose that **a network is made up of data centers and communication links between computers**. We can represent the location of each data center by a point and each communications link by a line segment, as shown in the following Figure. This computer network can be modeled using a graph in which the vertices of the graph represent the data centers and the edges represent communication links. In general, we visualize



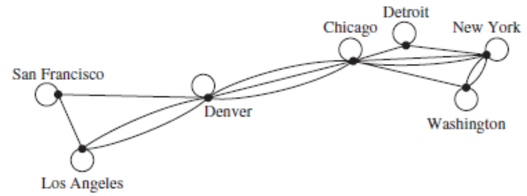
A computer network may contain multiple links between data centers, as shown in the following Figure. To model such networks we need graphs that have more than one edge connecting the same pair of vertices. Graphs that may have multiple edges connecting the same vertices are called multigraphs.



Sometimes a communications link connects a data center with itself, perhaps a feedback loop for diagnostic purposes. Such a network is illustrated in the following Figure.

To model this network we need to include edges that connect a vertex to itself. Such edges are called loops, and sometimes we may even have more than one loop at a vertex.

Graphs that may include loops, and possibly multiple edges connecting the same pair of vertices or a vertex to itself, are sometimes called pseudographs.

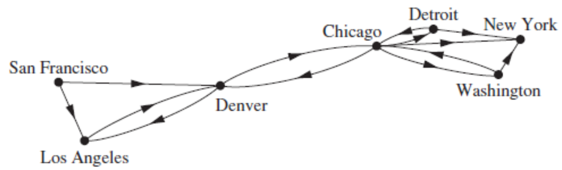


So far the graphs we have introduced are undirected graphs. Their edges are also said to be undirected.

However, to construct a graph model, we may find it necessary to assign directions to the edges of a graph.

For example, in a computer network, some links may operate in only one direction (such links are called single duplex lines).

This may be the case if there is a large amount of traffic sent to some data centers, with little or no traffic going in the opposite direction. Such a network is shown in the Figure (communication network with one-way communication links) above.

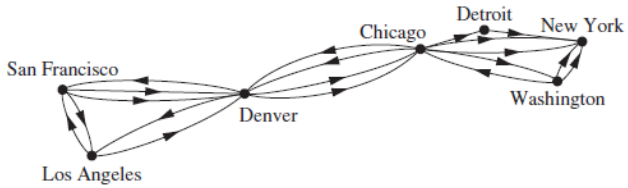


Definition: A *directed graph (or digraph) (V, E) consists of a nonempty set of vertices V and a set of directed edges (or arcs) E. Each directed edge is associated with an ordered pair of vertices.* The directed edge associated with the ordered pair (*u, v) is said to start at u and end at v.*

In some computer networks, multiple communication links between two data centers may be present, as illustrated in the Figure above.

Directed graphs that may have multiple directed edges from a vertex to a second (possibly the same) vertex are used to model such networks. We called such graphs directed multigraphs.

When there are *m directed edges, each associated to an* ordered pair of vertices *(u, v), we say that (u, v) is an edge of multiplicity m.*



9

For some models we may need a graph where some edges are undirected, while others are directed.

A graph with both directed and undirected edges is called a mixed graph.

For example, a mixed graph might be used to model a computer network containing links that operate in both directions and other links that operate only in one direction.

10

Graph Models

SOCIAL NETWORKS

Graphs are extensively used to model social structures based on different kinds of relationships between people or groups of people. These social structures, and the graphs that represent them, are known as social networks. In these graph models, **individuals or organizations are represented by vertices; relationships between individuals or organizations are represented by edges**.

Ex: (Acquaintanceship and Friendship Graphs). We can use a simple graph to represent whether two people know each other, that is, whether they are acquainted, or whether they are friends (either in the real world in the virtual world via a social networking site such as Facebook).

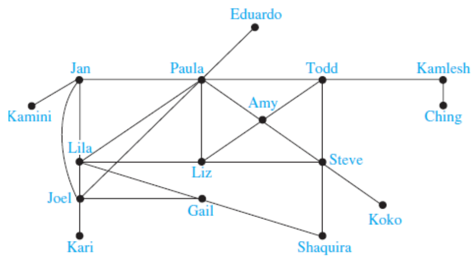
Each person in a particular group of people is represented by a vertex.

An undirected edge is used to connect two people when these people know each other, when we are concerned only with acquaintanceship, or whether they are friends.

No multiple edges and usually no loops are used. (If we want to include the notion of self-knowledge, we would include loops.)

A small acquaintanceship graph is shown in Figure below.

The acquaintanceship graph of all people in the world has more than six billion vertices and probably more than one trillion edges!



COMMUNICATION NETWORKS

We can model different communications networks using vertices to represent devices and edges to represent the particular type of communications links of interest.

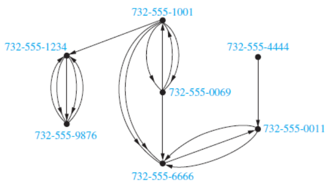
Ex: Call Graphs Graphs can be used to model telephone calls made in a network, such as a long distance telephone network.

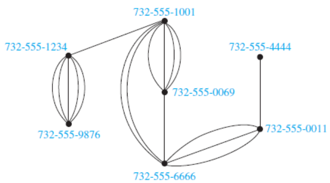
In particular, a directed multigraph can be used to model calls where each telephone number is represented by a vertex and each telephone call is represented by a directed edge.

The edge representing a call starts at the telephone number from which the call was made and ends at the telephone number to which the call was made.

We need directed edges because the direction in which the call is made matters.

We need multiple directed edges because we want to represent each call made from a particular telephone number to a second number.





TRANSPORTATION NETWORKS

We can use graphs to model many different types of transportation networks, including road, air, and rail networks, as well shipping networks.

Ex.: (Airline Routes)

We can model airline networks by representing each airport by a vertex.

In particular, we can model all the flights by a particular airline each day using a directed edge to represent each flight, going from the vertex representing the departure airport to the vertex representing the destination airport.

The resulting graph will generally be a directed multigraph, as there may be multiple flights from one airport to some other airport during the same day.

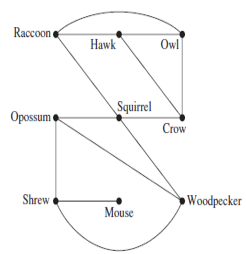
BIOLOGICAL NETWORKS
Many aspects of the biological sciences can be modeled using graphs.

Ex.: Graphs are used in many models involving the interaction of different species of animals.

For instance, the competition between species in an ecosystem can be modeled using a niche overlap graph.

Each species is represented by a vertex. An undirected edge connects two vertices if the two species represented by these vertices compete (that is, some of the food resources they use are the same).

A niche overlap graph is a simple graph because no loops or multiple edges are needed in this model.



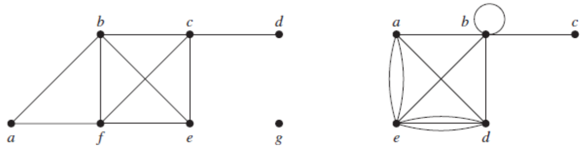
Terminology of Graphs

Definition: Two vertices *u and v in an undirected graph G are called adjacent (or neighbors) in G if u* and *v are endpoints of an edge e of G such an edge e is called incident with the vertices u* and *v and e is said to connect u and v.*

Definition: 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: The *degree of a vertex in an undirected graph is the number of edges incident with it, except* that a loop at a vertex contributes twice to the degree of that vertex. The degree of the vertex *v is denoted by deg(v).*

Ex.1:What are the degrees and what are the neighborhoods of the vertices in the graphs *G and H* displayed in Figure below, respectively?



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

The neighborhoods of these vertices are *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*}, and *N(g)* = ∅.

In *H*, deg*(a)* = 4, deg*(b)* = deg*(e)* = 6, deg*(c)* = 1, and deg*(d)* = 5. The neighborhoods of these vertices are *N(a)* = {*b, d, e*}, *N(b)* = {*a, b, c, d, e*}, *N(c)* = {*b*}, *N(d)* = {*a, b, e*}, and *N(e)* = {*a, b, d*}.

What do we get when we add the degrees of all the vertices of a graph *G = (V ,E)?*

Each edge contributes two to the sum of the degrees of the vertices because an edge is incident with exactly two (possibly equal) vertices.

This means that the sum of the degrees of the vertices is twice the number of edges.

We have the result in the following Theorem, which is sometimes called the handshaking theorem (and is also often known as the handshaking lemma), because of the analogy between an edge having two endpoints and a handshake involving two hands.

Theorem*: G = (V ,E) be an undirected graph with m* edges. Then

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

Ex.2:How many edges are there in a graph with 10 vertices each of degree six?

Solution: Because the sum of the degrees of the vertices is 6 * 10 = 60, it follows that

$$2m = 60$$

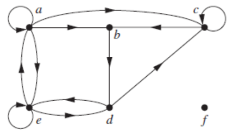
where *m is the number of edges. Therefore, m = 30.*

Theorem given above shows that the sum of the degrees of the vertices of an undirected graph is even. Therefore, it can be claimed that "An undirected graph has an even number of vertices of odd degree".

Definition: When *(u, v) is an edge of the graph G with directed edges, u is said to be adjacent to v and v* is said to be *adjacent from u. The vertex u is called the initial vertex of (u, v), and v is called* the *terminal or end vertex of (u, v). The initial vertex and terminal vertex of a loop are the* same.

Definition: In a graph with directed edges the *in-degree of a vertex v, denoted by* $\deg^-(v)$*, is the number* of edges with *v as their terminal vertex. The out-degree of v, denoted by* $\deg^+(v)$*, 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 this vertex.)

Ex.3:Find the in-degree and out-degree of each vertex in the graph G with directed edges shown in the Figure below.



Solution:

The in-degrees in *G* are
deg-(*a*) = 2,
deg-(*b*) = 2,
deg-(*c*) = 3,
deg-(*d*) = 2,
deg-(*e*) = 3, and
deg-(*f*) = 0.

The out-degrees are
deg+(*a*) = 4,
deg+(*b*) = 1,
deg+(*c*) = 2,
deg+(*d*) = 2,
deg+(*e*) = 3, and
deg+(*f*) = 0.

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

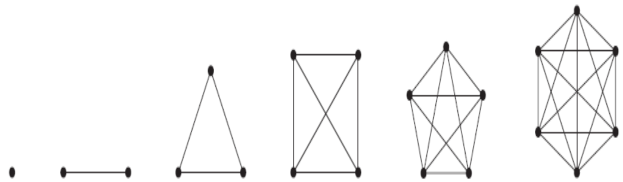
$$\sum_{v\in V} deg^-(v) = \sum_{v\in V} deg^+(v) = |E|.$$

Some Special Simple Graphs

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

The graphs K_n*, for n = 1, 2, 3, 4, 5, 6, are displayed in the Figures below.*

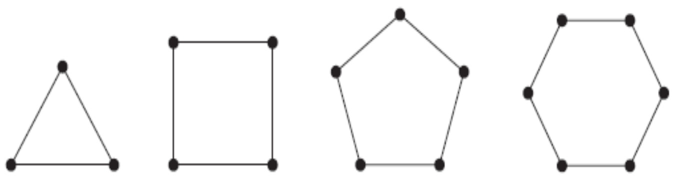
A simple graph for which there is at least one pair of distinct vertex not connected by an edge is called non-complete.



K_1 K_2 K_3 K_4 K_5 K_6

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

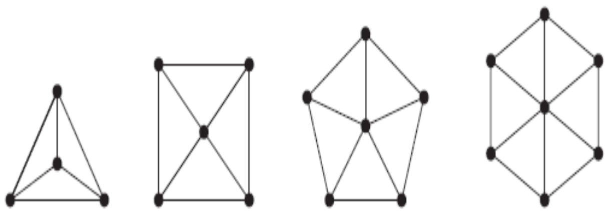
The cycles C_3, C_4, C_5, *and* C_6 *are displayed in the following* Figures.



C_3 C_4 C_5 C_6

Wheels: We obtain a wheel W_n *when we add an additional vertex to a cycle* C_n, *for* $n \geq 3$, and connect this new vertex to each of the *n vertices in* C_n, *by new edges.*

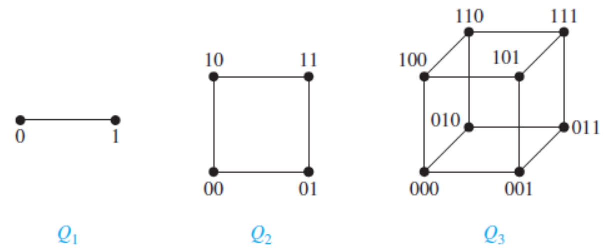
The wheels W_3, W_4, W_5, *and* W_6 *are displayed in Figure below.*



W_3 W_4 W_5 W_6

n-Cubes: An n-dimensional hypercube, or n-cube, denoted by Q_n, is a graph that has vertices representing the 2^n bit strings of length n.

Two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position. We display Q_1, Q_2, and Q_3 in the Figure below.



Bipartite Graphs

Sometimes a graph has the property that its vertex set can be divided into two disjoint subsets such that each edge connects a vertex in one of these subsets to a vertex in the other subset.

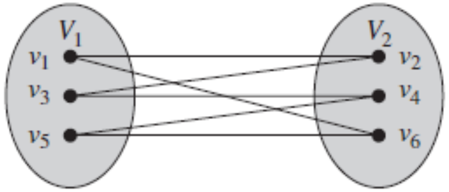
For example, consider the graph representing marriages between men and women in a village, where each person is represented by a vertex and a marriage is represented by an edge.

In this graph, each edge connects a vertex in the subset of vertices representing males and a vertex in the subset of vertices representing females.

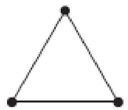
Definition: A simple graph *G is called bipartite if its vertex set V can* ***be partitioned into two disjoint sets*** V_1 *and* V_2 *such that every edge in the graph connects a vertex in* V_1 *and a vertex in* V_2 (so that no edge in *G connects either two vertices in* V_1 *or two vertices in* V_2*).*

When this condition holds, we call the pair *(V_1, V_2) a bipartition of the vertex set V of G.*

Ex.4: *C_6 is bipartite, as shown in the figure below, because its vertex set can be partitioned into the two sets $V_1 = \{v_1, v_3, v_5\}$ and $V_2 = \{v_2, v_4, v_6\}$, and every edge of C_6 connects a vertex in V_1 and a* vertex in *V_2.*

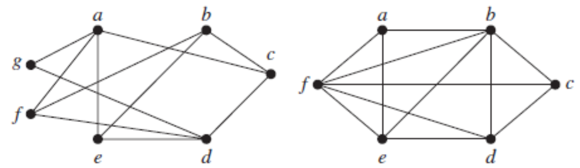


Ex.5: *K_3 is not bipartite. To verify this, note that if we divide the vertex set of K_3 into two disjoint* sets, one of the two sets must contain two vertices. If the graph were bipartite, these two vertices could not be connected by an edge, but in *K_3 each vertex is connected to every other vertex by* an edge.



K_3

Ex.6: Are the graphs *G and H displayed in the Figure below bipartite, respectively?*



Solution: Graph *G* is bipartite because its vertex set is the union of two disjoint sets, {*a, b, d* } and {*c, e, f, g*}, and each edge connects a vertex in one of these subsets to a vertex in the other subset. (Note that for *G* to be bipartite it is not necessary that every vertex in {*a, b, d*} be adjacent to every vertex in {*c, e, f, g*}. For instance, *b* and *g* are not adjacent.)

Graph *H* is not bipartite because its vertex set cannot be partitioned into two subsets so that edges do not connect two vertices from the same subset. (The reader should verify this by considering the vertices *a*, *b*, and *f* .)

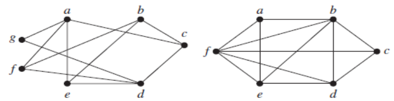
Following theorem provides a useful criterion for determining whether a graph is bipartite.

Theorem: A simple graph is bipartite if and only if it is possible to assign one of two different colors to each vertex of the graph so that no two adjacent vertices are assigned the same color.

Ex.7: Use the theorem to determine whether the graphs in Example 6 are bipartite.
Solution:
We first consider the graph *G*. We will try to assign one of two colors, say red and blue, to each vertex in *G* so that no edge in *G* connects a red vertex and a blue vertex.

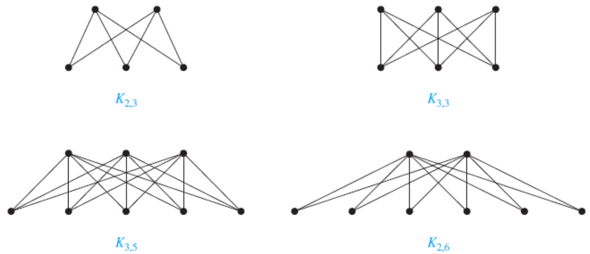
Without loss of generality we begin by arbitrarily assigning red to *a*. Then, we must assign blue to *c*, *e*, *f* , and *g*, because each of these vertices is adjacent to *a*.

To avoid having an edge with two blue endpoints, we must assign red to all the vertices adjacent to either *c*, *e*, *f* , or *g*. This means that we must assign red to both *b* and *d* (and means that *a* must be assigned red, which it already has been). We have now assigned colors to all vertices, with *a*, *b*, and *d* red and *c*, *e*, *f* , and *g* blue. Checking all edges, we see that every edge connects a red vertex and a blue vertex. Hence, by the theorem the graph *G* is bipartite.



Complete Bipartite Graphs: A complete bipartite graph $K_{m,n}$ *is a graph that has its vertex* set partitioned into two subsets of *m and n vertices, respectively with an edge between two* vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

The complete bipartite graphs $K_{2,3}$, $K_{3,3}$, $K_{3,5}$, *and* $K_{2,6}$ *are displayed in the figures below.*



Bipartite Graphs and Matchings

Bipartite graphs can be used to model many types of applications that involve matching the elements of one set to elements of another

(Job Assignments)
Suppose that there are *m employees in a group and n different jobs that* need to be done, where $m \geq n$.

Each employee is trained to do one or more of these n jobs. We would like to assign an employee to each job.

To help with this task, we can use a graph to model employee capabilities.

We represent each employee by a vertex and each job by a vertex.

For each employee, we include an edge from that employee to all jobs that the employee has been trained to do.

Note that the vertex set of this graph can be partitioned into two disjoint sets, the set of employees and the set of jobs, and each edge connects an employee to a job.

Consequently, this graph is bipartite, where the bipartition is *(E, J) where E is the set of employees and J is* the set of jobs. We now consider two different scenarios.

First, suppose that a group has four employees:

Alvarez, Berkowitz, Chen, and Davis; and suppose that four jobs need to be done to complete

Project 1: requirements, architecture, implementation, and testing.

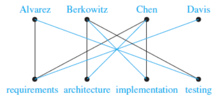
Suppose that Alvarez has been trained to do requirements and testing;

Berkowitz has been trained to do architecture, implementation, and testing;

Chen has been trained to do requirements, architecture, and implementation; and

Davis has only been trained to do requirements.

We model these employee capabilities using the bipartite graph in Figure below.

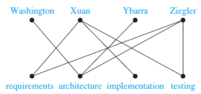


Second, suppose that a group has second group also has four employees:
Washington, Xuan, Ybarra, and Ziegler; and suppose that the same four jobs need to be done to complete Project 2 as are needed to complete Project 1. Suppose also that Washington has been trained to do architecture;

Xuan has been trained to do requirements, implementation, and testing;

Ybarra has been trained to do architecture; and

Ziegler has been trained to do requirements, architecture and testing. We model these employee capabilities using the bipartite graph in Figure below.



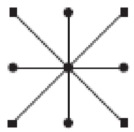
To complete Project 2, we must also assign an employee to each job so that every job has an employee assigned to it and no employee is assigned more than one job.
However, this is impossible because there are only two employees, Xuan and Ziegler, who have been trained for at least one of the three jobs of requirements, implementation, and testing.
Consequently, there is no way to assign three different employees to these three job so that each job is assigned an employee with the appropriate training.

Some Applications of Special Types of Graphs

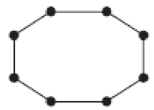
Local Area Networks: The various computers in a building, such as minicomputers and personal computers, as well as peripheral devices such as printers and plotters, can be connected using a *local area network.*

Some of these networks are based on a star topology, where all devices are connected to a central control device.

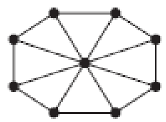
A local area network can be represented using a complete bipartite graph $K_{1,n}$, *as shown in figure below. Messages are sent from device to* device through the central control device.



Other local area networks are based on a *ring topology, where each device is connected to* exactly two others. Local area networks with a ring topology are modeled using *n-cycles,* C_n, as shown in figure below. Messages are sent from device to device around the cycle until the intended recipient of a message is reached.



Finally, some local area networks use a hybrid of these two topologies. Messages may be sent around the ring, or through a central device. This redundancy makes the network more reliable. Local area networks with this redundancy can be modeled using wheels W_n, *as shown* in Figure (c) below.



REMOVING OR ADDING EDGES OF A GRAPH

Given a graph *G = (V ,E) and an edge e ∈ E, we can produce a subgraph of G by removing the edge e. The resulting subgraph, denoted* by *G − e, has the same vertex set V as G. Its edge set is E − e. Hence,*

$$G - e = (V, E - \{e\}).$$

Similarly, if *E′* is a subset of *E, we can produce a subgraph of G by removing the edges in E′* from the graph. The resulting subgraph has the same vertex set *V as G. Its edge set is E − E′.*

We can also add an edge *e to a graph to produce a new larger graph when this edge connects* two vertices already in *G. We denote by G + e the new graph produced by adding a new edge e,* connecting two previously non-incident vertices, to the graph *G. Hence,*

$$G + e = (V, E \cup \{e\}).$$

The vertex set of *G + e is the same as the vertex set of G and the edge set is the union of the* edge set of *G and the set {e}.*

EDGE CONTRACTIONS

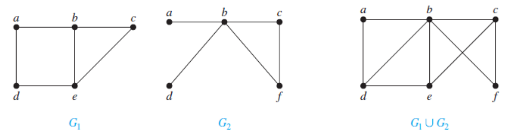
Sometimes when we remove an edge from a graph, we do not want to retain the endpoints of this edge as separate vertices in the resulting subgraph.

In such a case, we perform an edge contraction which removes an edge *e with endpoints u and v and* merges *u and w into a new single vertex w, and for each edge with u or v as an endpoint replaces* the edge with one with *w as endpoint in place of u or v and with the same second endpoint.*

Hence, the contraction of the edge *e with endpoints u and v in the graph G = (V ,E) produces a* new graph *G′ = (V′,E′) (which is not a subgraph of G), where V ′= V − {u, v} ∪ {w} and E′* contains the edges in *E which do not have either u or v as endpoints and an edge connecting w* to every neighbor of either *u or v in V .*

For example, the contraction of the edge connecting the vertices *e and c in the graph G_1 in figure below produces a new graph G'_1* with vertices *a, b, d,* and *w. As in G_1, there is an edge in G_1* connecting *a and b and an edge connecting a and d.*

There also is an edge in *G'_1* that connects *b and w that replaces the edges connecting b and c and* connecting *b and e in G_1 and an edge in G'_1* that connects *d and w replacing the edge connecting d and e in G_1.*



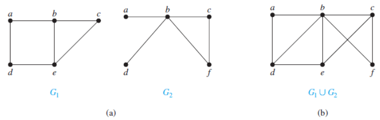
(a) (b)

REMOVING VERTICES FROM A GRAPH
When we remove a vertex *v and all edges* incident to it from *G = (V ,E), we produce a subgraph, denoted by G − v. Observe that G − v = (V − v, E'), where E* is the set of edges of *G not incident to v. Similarly, if V* is a subset of *V , then the graph G − V'* is the subgraph *(V − V', E'), where E'* is the set of edges of *G not incident to a vertex in V'*.

GRAPH UNIONS
Two or more graphs can be combined in various ways. The new graph that contains all the vertices and edges of these graphs is called the union of the graphs. We will give a more formal definition for the union of two simple graphs.

Definition: The *union of two simple graphs G1 = (V1, E1) and G2 = (V2, E2) is the simple graph with* vertex set *V1 ∪ V2 and edge set E1 ∪ E2. The union of G1 and G2 is denoted by G1 ∪ G2.*

Ex. : Find the union of the graphs *G1 and G2 shown in figure (b) above.*
Solution: The vertex set of the union *G1 ∪ G2 is the union of the two vertex sets, namely, {a, b, c, d, e, f }. The edge set of the union is the union of the two edge sets. The union is* displayed in Figure (b).



Representing Graphs and Graph Isomorphism

There are many useful ways to represent graphs. As we will see throughout this chapter, in working with a graph it is helpful to be able to choose its most convenient representation. In this lecture, we will show how to represent graphs in several different ways.

Sometimes, **two graphs have exactly the same form**, in the sense that there is a one-to-one correspondence between their vertex sets that preserves edges.

In such a case, ***we say that the two graphs are isomorphic***.

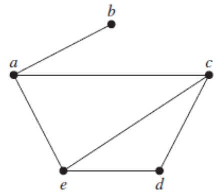
Determining whether two graphs are isomorphic is an important problem of graph theory

Representing Graphs

One way to represent a graph without multiple edges is to list all the edges of this graph.

Another way to represent a graph with no multiple edges is to use adjacency lists, which specify the vertices that are adjacent to each vertex of the graph.

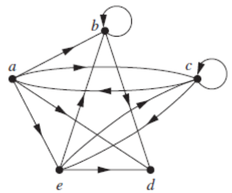
Ex.1: Use adjacency lists to describe the simple graph given in Figure below.



Solution:

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

Ex.2: Represent the directed graph shown in the figure below by listing all the vertices that are the terminal vertices of edges starting at each vertex of the graph.

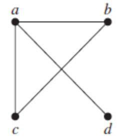


Solution:

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

Carrying out graph algorithms using the representation of graphs by lists of edges, or by adjacency lists, can be cumbersome if there are many edges in the graph. To simplify computation, graphs can be represented using matrices.

Ex.3: Use an adjacency matrix to represent the graph shown in Figure below.



Solution: We order the vertices as *a, b, c, d*. The matrix representing this graph is

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

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 an arbitrary listing of the vertices of the directed graph.

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

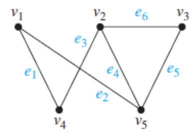
When a simple graph contains relatively few edges, that is, when it is sparse, it is usually preferable to use adjacency lists rather than an adjacency matrix to represent the graph.

Incidence Matrices

Another common way to represent graphs is to use incidence matrices. Let *G = (V ,E) be an* undirected graph. Suppose that *$v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges* of *G.(*M = *[m_{ij}])*

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

Ex.4: Represent the graph shown in Figure below with an incidence matrix.



Solution:

$$\begin{array}{c c} & \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

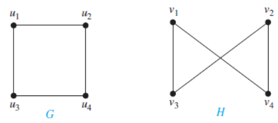
Isomorphism of Graphs

Definition: The simple graphs *$G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists 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 nonsomorphic.*

In other words, when two simple graphs are isomorphic, there is a one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship.

Isomorphism of simple graphs is an equivalence relation.

Ex.5: Show that the graphs *$G = (V, E)$ and $H = (W, F)$, displayed in figure below, 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.*

To see that this correspondence preserves adjacency, 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, and 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.

Note that: G and H are represent the corresponding domain and image set for the function theory, respectively.

Determining whether Two Simple Graphs are Isomorphic

It is often **difficult to determine whether two simple graphs are isomorphic**.

There are *$n!$ possible* one-to-one correspondences between the vertex sets of two simple graphs with *n vertices*

Sometimes it is not hard to show that two graphs are not isomorphic(Check for the properties preserved by isomorphism).

A property preserved by isomorphism of graphs is called a graph invariant.

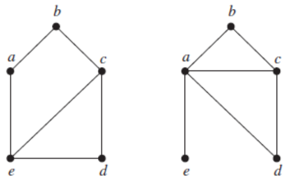
For instance, isomorphic simple graphs must have the same number of vertices, because there is a one-to-one correspondence between the sets of vertices of the graphs.

Isomorphic simple graphs also must have the same number of edges.

In addition, the degrees of the vertices in isomorphic simple graphs must be the same.

Determining whether Two Simple Graphs are Isomorphic

Ex.6: Show that the graphs displayed in figure below are not isomorphic.



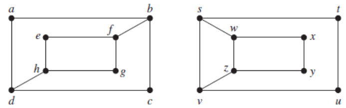
G H

Solution:

Both *G and H have five vertices and six edges.*

However, H has a vertex of degree one, namely, *e, whereas G has no vertices of degree one. It follows that G and H are not isomorphic.*

Ex.7: Determine whether the graphs shown in Figure below are isomorphic.



G H

Solution:

The graphs *G* and *H* both have eight vertices and 10 edges. They also both have four vertices of degree two and four of degree three. Because these invariants all agree, it is still conceivable that these graphs are isomorphic.

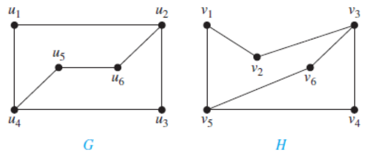
However, *G* and *H* are not isomorphic. To see this, note that because deg(*a*) = 2 in *G*, *a* must correspond to either *t* , *u*, *x*, or *y* in *H*, because these are the vertices of degree two in *H*.

However, each of these four vertices in *H* is adjacent to another vertex of degree two in *H*, which is not true for *a* in *G*.

Another way to see that *G* and *H* are not isomorphic is to note that the subgraphs of *G* and *H* made up of vertices of degree three and the edges connecting them must be isomorphic if these two graphs are isomorphic (the reader should verify this). However, these subgraphs, shown in figure, are not isomorphic.

To show that a function *f from the vertex set of a graph G to the vertex set of a graph H is an* isomorphism, we need to show that *f preserves the presence and absence of edges. One helpful* way to do this is to use adjacency matrices. In particular, to show that *f is an isomorphism, we* can show that the adjacency matrix of *G is the same as the adjacency matrix of H.*

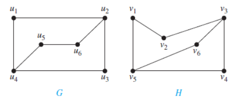
Ex.7: Determine whether the graphs *G and H displayed in Figure below are isomorphic.*



Solution:
Both *G* and *H* have six vertices and seven edges.
Both have four vertices of degree two and two vertices of degree three. It is also easy to see that the subgraphs of *G* and *H* consisting of all vertices of degree two and the edges connecting them are isomorphic (as the reader should verify). Because *G* and *H* agree with respect to these invariants, it is reasonable to try to find an isomorphism *f* .

We now will define a function *f* and then determine whether it is an isomorphism. Because deg(*u*1*)* = 2 and because *u*1 is not adjacent to any other vertex of degree two, the image of *u*1 must be either *v*4 or *v*6, the only vertices of degree two in *H* not adjacent to a vertex of degree two.

We arbitrarily set *f (u*1*)* = *v*6. [If we found that this choice did not lead to isomorphism, we would then try *f (u*1*)* = *v*4.] Because *u*2 is adjacent to *u*1, the possible images of *u*2 are *v*3 and *v*5. We arbitrarily set *f (u*2*)* = *v*3. Continuing in this way, using adjacency of vertices and degrees as a guide, we set *f (u*3*)* = *v*4, *f (u*4*)* = *v*5, *f (u*5*)* = *v*1, and *f (u*6*)* = *v*2.We now have a one-to-one correspondence between the vertex set of *G* and the vertex set of *H*, namely, *f (u*1*)* = *v*6, *f (u*2*)* = *v*3, *f (u*3*)* = *v*4, *f (u*4*)* = *v*5, *f (u*5*)* = *v*1, *f (u*6*)* = *v*2. To see whether *f* preserves edges, we examine the adjacency matrix of *G*,



Solution: (continued)

$$\mathbf{A}_G = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array},$$

and the adjacency matrix of *H* with the rows and columns labeled by the images of the corresponding vertices in *G*,

$$\mathbf{A}_H = \begin{array}{c} \\ v_6 \\ v_3 \\ v_4 \\ v_5 \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccccc} v_6 & v_3 & v_4 & v_5 & v_1 & v_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}.$$

Because $\mathbf{A}_G = \mathbf{A}_h$ it follows that *f* preserves edges. We conclude that *f* is an isomorphism, so *G* and *H* are isomorphic. Note that if *f* turned out not to be an isomorphism, we would *not* have established that *G* and *H* are not isomorphic, because another correspondence of the vertices in *G* and *H* may be an isomorphism.