III sem A

DMS notes

CSE

Graphs

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.

Types of Graphs

1. A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called a **simple graph.**
2. Graphs that may have **multiple edges** connecting the same vertices are called **multigraphs.**
3. Graphs that may include loops, and possibly multiple edges connecting the same pair of vertices or a vertex to itself, are sometimes called **pseudographs.**
4. When a directed graph has no loops and has no multiple directed edges, it is called a **simple directed graph.**
5. 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.**
6. A graph with both directed and undirected edges is called a **mixed graph.**

TABLE 1 Graph Terminology.

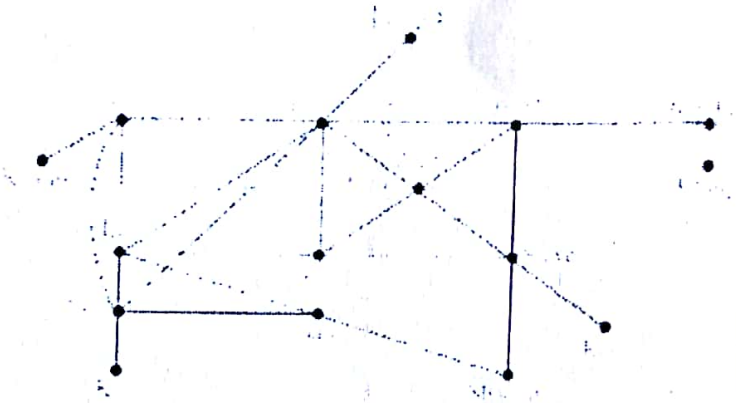
Type	*Edges*	*Multiple Edges Allowed?*	*Loops Allowed?*
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

Graph Models : Graphs are used in a wide variety of models.

1. **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.

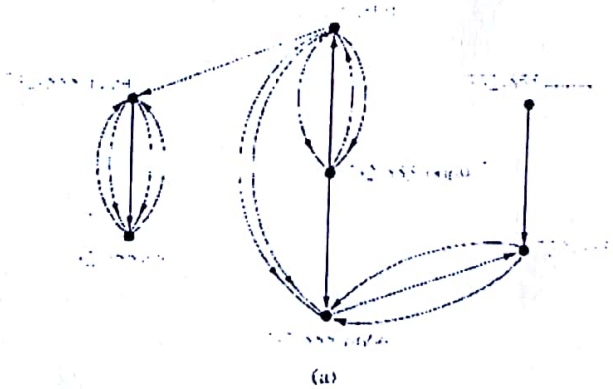
   Example : Friendship Graphs We can use a simple graph to represent whether two people know each other, that is, whether they are acquainted, or whether they

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 isused 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.)



2. 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.

Example :Call 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.



(a)

FIGURE 8 **A Call Graph.**

3. SOFTWARE DESIGN APPLICATIONS Graph models are useful tools in the design of software.

Example : Module Dependency Graphs One of the most important tasks in designing software is how to structure a program into different parts, or modules. Understanding how the different modules of a program interact is essential not only for program design, but also for testing and maintenance of the resulting software. A module dependency graph provides a useful tool for understanding how different modules of a program interact. In a program dependency graph, each module is represented by a vertex. There is a directed edge from a module to a second module if the second module depends on the first.

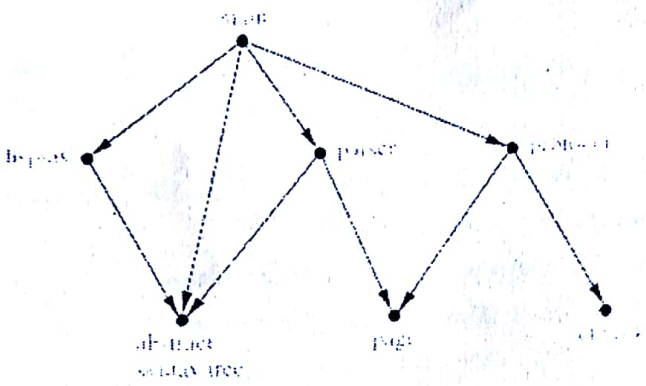


FIGURE 9 A Module Dependency Graph.

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

Example : Niche Overlap Graphs in Ecology 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.

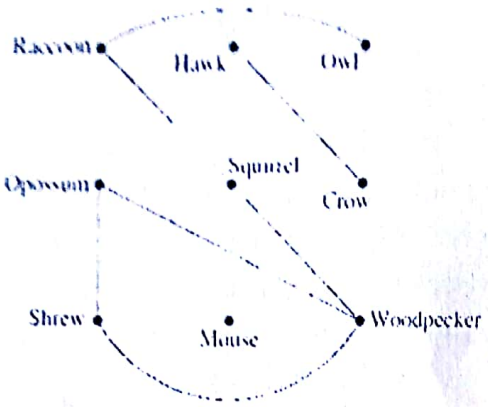


FIGURE 11 A Niche Overlap Graph.

5. TOURNAMENTS We now give some examples that show how graphs can also be used to model different kinds of tournaments.

Example : Round-Robin Tournaments A tournament where each team plays every other team exactly once and no ties are allowed is called a round-robin tournament. Such tournaments can be modeled using directed graphs where each team is represented by a vertex. Note that (a, b) is an edge if team a beats team b. This graph is a simple directed graph, containing no loops or multiple directed edges (because no two teams play each other more than once).

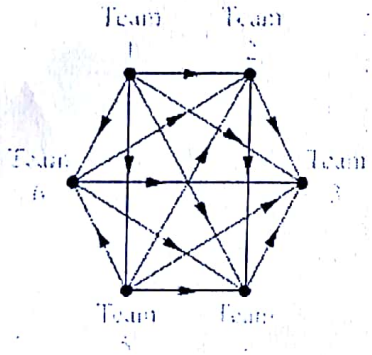


FIGURE 13 A Graph Model of a Round-Robin Tournament.

Basic Terminology

1. 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.

(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.

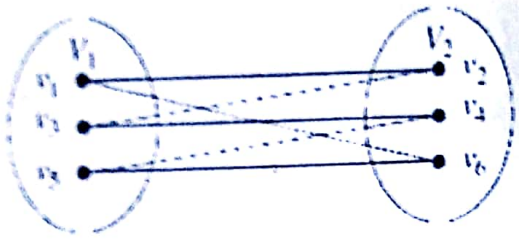
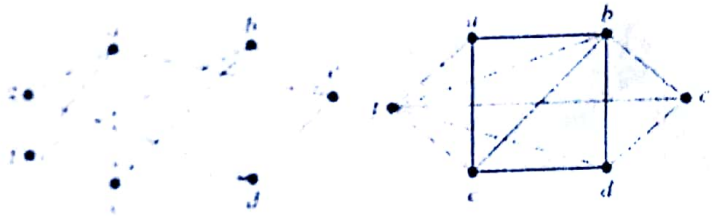


FIGURE 7 Showing That C_6 Is Bipartite.



Note : 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.

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.

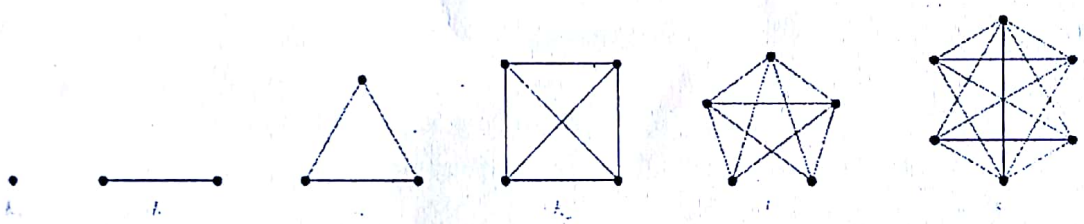
2. The set of all neighbors of a vertex v of $G = (V, E)$, denoted by $N(v)$, is called the *neighborhood* of v. If A is a subset of V, we denote by $N(A)$ the set of all vertices in G that are adjacent to at least one vertex in A. So, $N(A) = \bigcup_{v \in A} N(v)$.

3. 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).

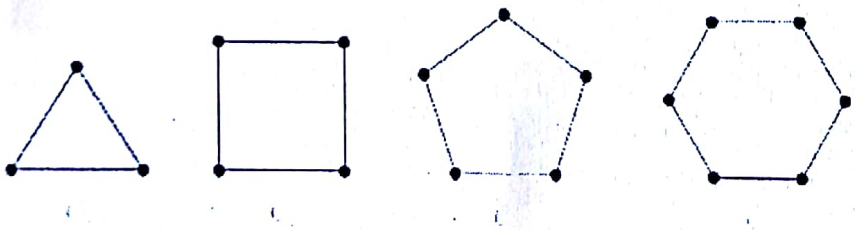
4. In a graph with directed edges the in-degree of a vertex v, is the number of edges with v as their terminal vertex. The out-degree of v is the number of edges with v as their initial vertex.

Some Special Simple Graphs

1. 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.

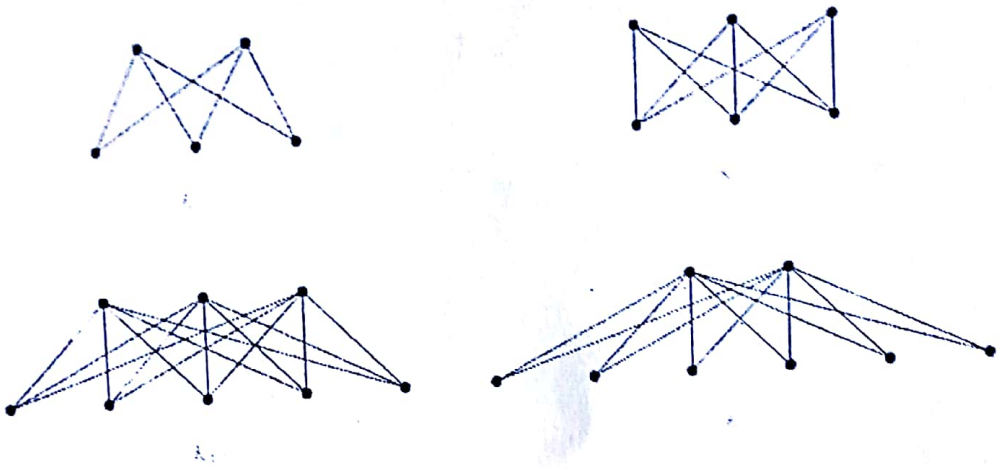


2. A cycle Cn, $n \geq 3$, consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}$, $\{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}$, and $\{v_n, v_1\}$.



Bipartite Graphs

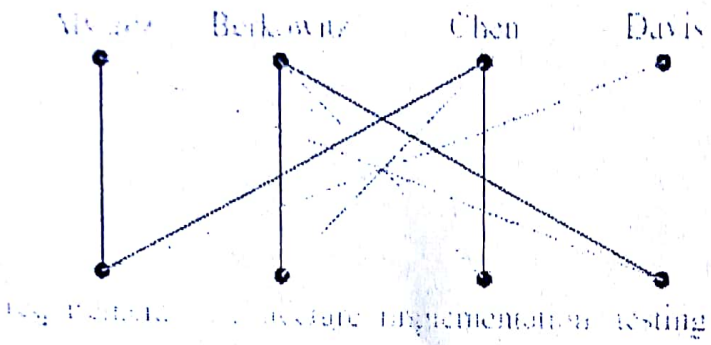
A simple graph G is called *bipartite* if its vertex set V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2



Application of Bipartite graph

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.

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 10(a).



(a)

Matching

A **matching** M in a simple graph $G = (V, E)$ is a subset of the set E of edges of the graph such that no two edges are incident with the same vertex. A vertex that is the endpoint of an edge of a matching M is said to be **matched** in M; otherwise it is said to be **unmatched**.

A **maximum matching** is a matching with the largest number of edges. We say that a matching M in a bipartite graph $G = (V, E)$ with bipartition $(V1, V2)$ is a **complete matching from** $V1$ **to** $V2$ if every vertex in $V1$ is the endpoint of an edge in the matching.

HALL'S THEOREM

The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of $V1$.

Subgraph

Subgraph of a graph G = (V ,E) is a graph H = (W, F), where W ⊆ V and F ⊆ E. A subgraph H of G is a proper subgraph of G if H ≠ G.

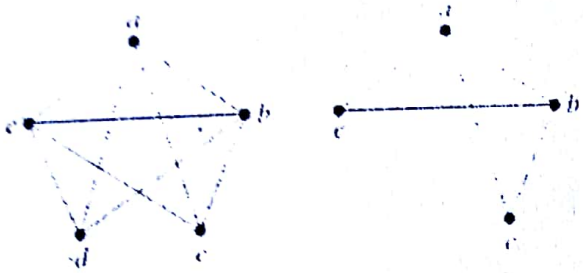


FIGURE 15 A Subgraph of K_5.

Induced subgraph

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 of this edge are in W.

The graph G shown in Figure 15 is a subgraph of K_5. If we add the edge connecting c and e to G, we obtain the subgraph induced by W = {a, b, c, e}.

Representing Graphs

1. **Adjacency lists** specifies the vertices that are adjacent to each vertex of the graph.

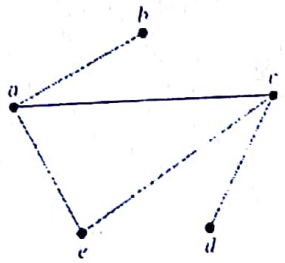


FIGURE 1 A Simple Graph.

TABLE 1 An Adjacency List for a Simple Graph.

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

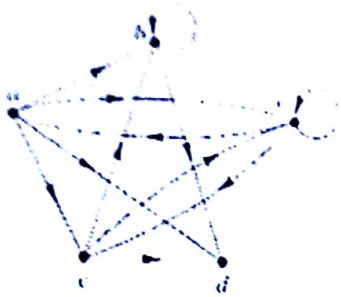


FIGURE 2 A Directed Graph.

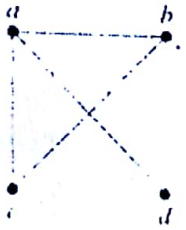
TABLE 2 An Adjacency List for a Directed Graph.

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

2. Adjacency Matrices

The **adjacency matrix** A (or AG) of G, with respect to this listing of the vertices, is the n x 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 its adjacency matrix is $A = [a_{ij}]$, then

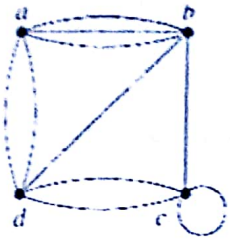
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$



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

FIGURE 3 Simple Graph.

Adjacency matrices can also be used to represent undirected graphs with loops and with multiple edges. A loop at the vertex v_i is represented by a 1 at the $(i, i)^{th}$ position of the adjacency matrix. When multiple edges connecting the same pair of vertices v_i and v_j, or multiple loops at the same vertex, are present, the adjacency matrix is no longer a zero–one matrix, because the (i, j)th entry of this matrix equals the number of edges that are associated to $\{vi, vj\}$. *All undirected graphs, including multigraphs and pseudographs, have symmetric adjacency matrices.*



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

FIGURE 5
A Pseudograph.

The matrix for a directed graph $G = (V, E)$ has a 1 in its $(i, j)^{\text{th}}$ position if there is an edge from v_i to v_j. In other words, if $A = [a_{ij}]$ is the adjacency matrix for the directed graph with, then

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

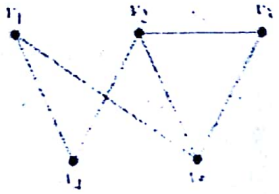
The adjacency matrix for a directed graph does not have to be symmetric, because there may not be an edge from v_j to v_i when there is an edge from v_i to v_j.

Adjacency matrices can also be used to represent directed multigraphs. Again, such matrices are not zero–one matrices when there are multiple edges in the same direction connecting two vertices. In the adjacency matrix for a directed multigraph, a_{ij} equals the number of edges that are associated to (v_i, v_j).

3. 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. Then the incidence matrix with respect to this ordering of V and E is the $n \times m$ matrix $M = [m_{ij}]$, where

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



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

Incidence matrices can also be used to represent multiple edges and loops. Multiple edges are represented in the incidence matrix using columns with identical entries, because these edges are incident with the same pair of vertices. Loops are

represented using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with this loop.

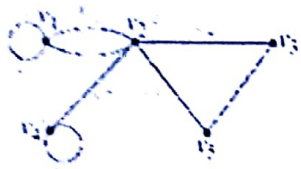


FIGURE 7
A Pseudograph.

Solution: The incidence matrix for this graph is

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}.$$

Connectivity

1. **Paths :** Informally, 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 edges.

   Formally, 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 the end points x_{i-1} and x_i, for $i = 1, \ldots, n$.

   A path of length greater than zero that begins and ends at the same vertex is called a *circuit* or *cycle*. A path or circuit is called *simple* if it does not contain the same edge more than once.

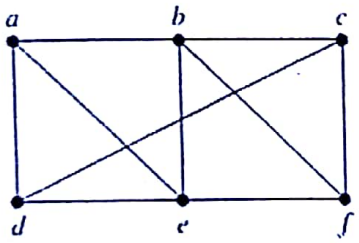


FIGURE 1 A Simple Graph.

In the simple graph shown in Figure 1, a, d, c, f, e is a simple path of length 4.

However, d, e, c, a is not a path, because $\{e, c\}$ is not an edge.

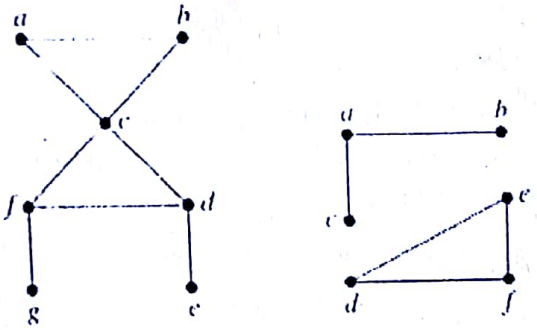
b, c, f, e, b is a circuit of length 4

The path a, b, e, d, a, b, which is of length 5, is not simple because it contains the edge $\{a, b\}$ twice.

2. **Connectedness in Undirected Graphs**

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

connected and disconnected graphs



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

3. **CONNECTED COMPONENTS**

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.

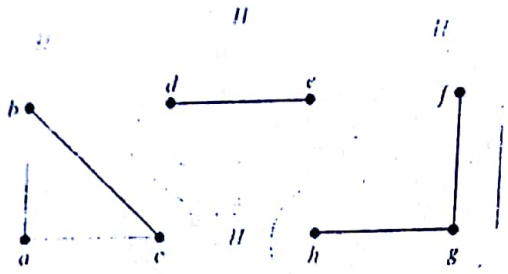


FIGURE 3 The Graph H and Its Connected Components H_1, H_2, and H_3.

4. VERTEX CONNECTIVITY

We define the **vertex connectivity** of a non complete graph G, denoted by $\kappa(G)$, as the minimum number of vertices to be removed to make the graph disconnected.

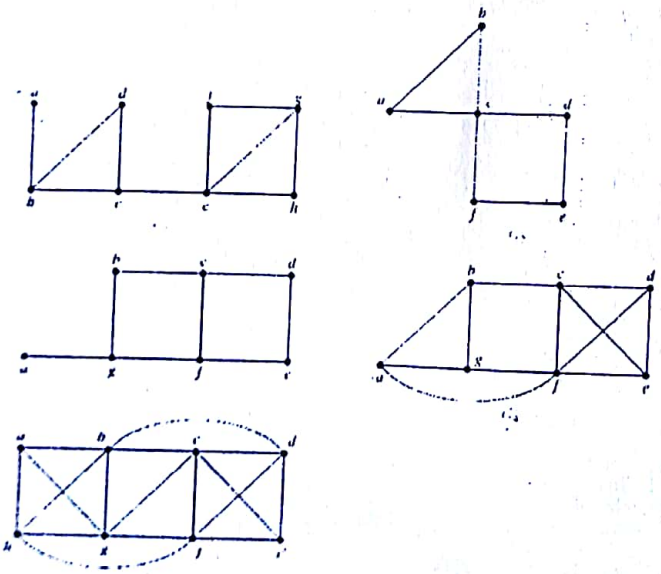
When G is a complete graph, it has no vertex connectivity , because removing any subset of its vertices and all incident edges still leaves a complete graph. Consequently, we cannot define $\kappa(G)$ as the minimum number of vertices in a vertex cut when G s complete. Instead, we set $\kappa(K_n) = n - 1$, the number of vertices needed to be removed to produce a graph with a single vertex.

$\kappa(G) = 0$ if and only if G is disconnected or $G = K_1$,

$\kappa(G) = n - 1$ if and only if G is complete

Thus, $0 \leq \kappa(G) \leq n - 1$

Find the vertex connectivity for each of the graphs in the following Figure



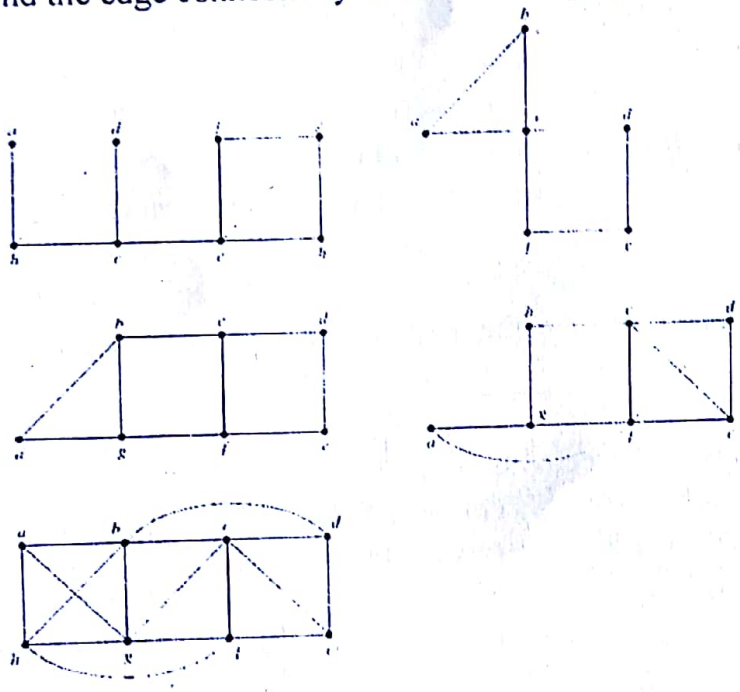
$\kappa(G_1) = 1, \kappa(G_2) = 1, \kappa(G_3) = 2, \kappa(G_4) = 2, \kappa(G_5) = 3.$

5. EDGE CONNECTIVITY

The **edge connectivity** of a graph G, denoted by $\lambda(G)$, is the minimum number of edges whose removal makes the graph disconnected.

$\lambda(G) = 0$ if G is not connected and if G is a graph consisting of a single vertex. $\lambda(G) = n - 1$ if $G = K_n$. Thus, $0 \leq \lambda(G) \leq n - 1$.

Find the edge connectivity of each of the graphs



$\lambda(G_1) = 1, \lambda(G_2) = 2, \lambda(G_3) = 2, \lambda(G_4) = 3, \lambda(G_5) = 3.$

When G = (V ,E) is a non-complete connected graph with at least three vertices. the minimum degree of a vertex of G is an upper bound for both the vertex connectivity of G and the edge connectivity of G. That is, $\kappa(G) \leq \min_{v \in V} \deg(v)$ and $\lambda(G) \leq \min_{v \in V} \deg(v)$. Also $\kappa(G) \leq \lambda(G)$ when G is a connected non complete graph. Thus, $\kappa(G) \leq \lambda(G) \leq \min_{v \in V} \deg(v)$.

APPLICATIONS OF VERTEX AND EDGE CONNECTIVITY

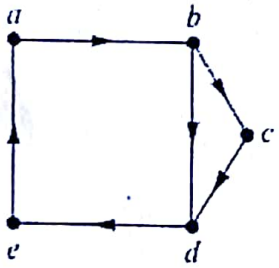
We can model a data network using vertices to represent routers and edges to represent links between them. The vertex connectivity of the resulting graph equals the minimum number of routers that disconnect the network when

they are out of service. If fewer routers are down, data transmission between every pair of routers is still possible. The edge connectivity represents the minimum number of fiber optic links that can be down to disconnect the network. If fewer links are down, it will still be possible for data to be transmitted between every pair of routers.

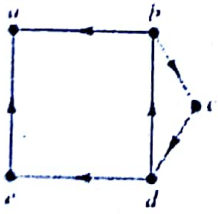
We can model a highway network, using vertices to represent highway intersections and edges to represent sections of roads running between intersections. The vertex connectivity of the resulting graph represents the minimum number of intersections that can be closed at a particular time that makes it impossible to travel between every two intersections. If fewer intersections are closed, travel between every pair of intersections is still possible. The edge connectivity represents the minimum number of roads that can be closed to disconnect the highway network. If fewer highways are closed, it will still be possible to travel between any two intersections. Clearly, it would be useful for the highway department to take this information into account when planning road repairs.

6. Connectedness in Directed Graphs

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



The subgraphs of a directed graph G that are strongly connected but not contained in larger strongly connected subgraphs are called the **strongly connected components**



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, and d and edges (b, c), (c, d), and (d, b).

Euler Paths and Circuits

The Seven Bridges of Königsberg

The town of Königsberg, was divided into four sections by the branches of the Pregel River. These four sections included the two regions on the banks of the Pregel, Kneiphof Island, and the region between the two branches of the Pregel. In the eighteenth century seven bridges connected these regions.

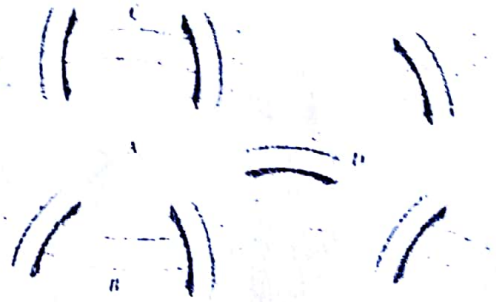


FIGURE 1 The Seven Bridges of Königsberg.

The townspeople took long walks through town on Sundays. They wondered whether it was possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point. The Swiss mathematician Leonhard Euler solved this problem. His solution, published in 1736, may be the first use of graph theory. Euler studied this problem using the multigraph obtained when the four regions are represented by vertices and the bridges by edges. This multigraph is shown in Figure 2.

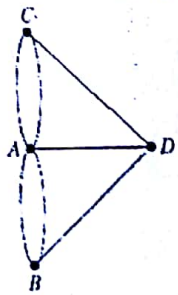


FIGURE 2 Multigraph Model of the Town of Königsberg.

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.

NECESSARY AND SUFFICIENT CONDITIONS FOR EULER CIRCUITS AND PATHS

1. A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree.
2. A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.

APPLICATIONS OF EULER PATHS AND CIRCUITS

Euler paths and circuits can be used to solve many practical problems. For example, many applications ask for a path or circuit that traverses each street in a neighborhood, each road in a transportation network, each connection in a utility grid, or each link in a communications network exactly once. Finding an Euler path or circuit in the appropriate graph model can solve such problems. For example, if a postman can find an Euler path in the graph that represents the streets the postman needs to cover, this path produces a route that traverses each street of the route exactly once. If no Euler path exists some streets will have to be traversed more than once.

Hamilton Paths and Circuits

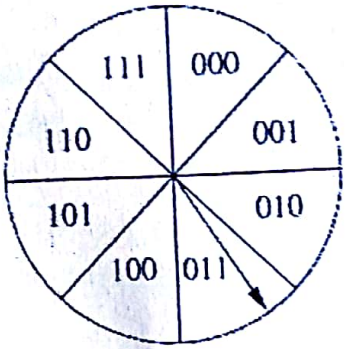
A simple path in a graph G that passes through every vertex exactly once is called a *Hamilton path*, and a simple circuit in a graph G that passes through every vertex exactly once is called a *Hamilton circuit*.

Surprisingly, there are no known simple necessary and sufficient criteria for the existence of Hamilton circuits. Certain properties can be used to show that a graph has no Hamilton circuit. For instance, a graph with a vertex of degree one cannot have a Hamilton circuit, Moreover, if a vertex in the graph has degree two, then both edges that are incident with this vertex must be part of any Hamilton circuit.

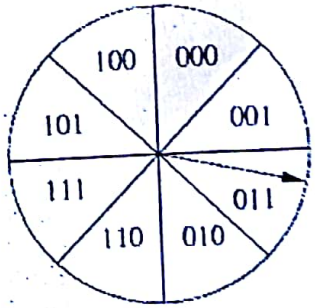
Applications of Hamilton Circuits

1. The famous **traveling salesperson problem** or **TSP** asks for the shortest route a traveling salesperson should take to visit a set of cities. This problem reduces to finding a Hamilton circuit in a complete graph such that the total weight of its edges is as small as possible.
2. **Gray Codes**
   The position of a rotating pointer can be represented in digital form. One way to do this is to split the circle into 2^n arcs of equal length and to assign a bit string of length

n to each arc. Two ways to do this using bit strings of length three are shown in the Figure below .



When the pointer is near the boundary of two arcs, a mistake may be made in reading its position. This may result in a major error in the bit string read. For instance, in the coding scheme in the above Figure, if a small error is made in determining the position of the pointer, the bit string 100 is read instead of 011. All three bits are incorrect! To minimize the effect of an error in determining the position of the pointer, the assignment of the bit strings to the 2^n arcs should be made so that only one bit is different in the bit strings represented by adjacent arcs. This is exactly the situation in the coding scheme in Figure below. An error in determining the position of the pointer gives the bit string 010 instead of 011. Only one bit is wrong.



Planar Graphs

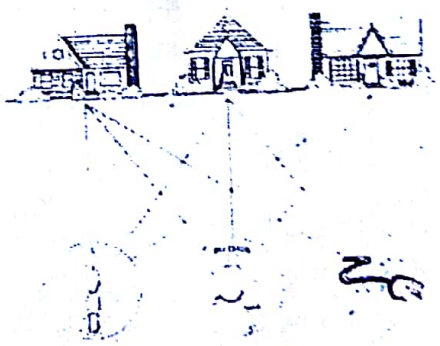


FIGURE 1 Three Houses and Three Utilities.

Consider the problem of joining three houses to each of three separate utilities, as shown in Figure 1. Is it possible to join these houses and utilities so that none of the connections cross? This problem can be modeled using the complete bipartite graph $K_{3,3}$. The original question can be rephrased as: Can $K_{3,3}$ be drawn in the plane so that no two of its edges cross?

A graph is called *planar* if it can be drawn in the plane without any edges crossing. Such a drawing is called a *planar representation* of the graph.

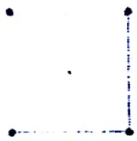


FIGURE 2 The Graph K_4.

FIGURE 3 K_4 Drawn with No Crossings.

FIGURE 4 The Graph Q_3.

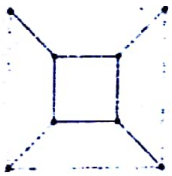


FIGURE 5 A Planar Representation of Q_3.

$K_{3,3}$, is non- planar.

APPLICATIONS OF PLANAR GRAPHS

Planarity of graphs plays an important role in the design of electronic circuits. We can model a circuit with a graph by representing components of the circuit by vertices and connections between them by edges. We can print a circuit on a single board with no connections crossing if the graph representing the circuit is planar. When this graph is not planar, we must turn to more expensive options. For example, we can partition the vertices in the graph representing the circuit into planar subgraphs. We then construct the circuit using multiple layers. We can construct the circuit using insulated wires whenever connections cross. In this case, drawing the graph with the fewest possible crossings is important.

The planarity of graphs is also useful in the design of road networks. Suppose we want to connect a group of cities by roads. We can model a road network connecting these cities using a simple graph with vertices representing the cities and edges representing the highways connecting them. We can built this road network without using underpasses or overpasses if the resulting graph is planar.

Euler's Formula

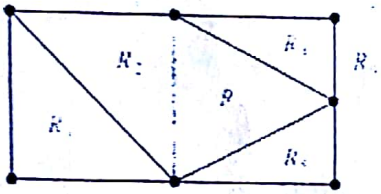
Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then $r = e - v + 2$.

a) Suppose that a connected planar simple graph has 20 vertices, each of degree 3. Into how many regions does a representation of this planar graph split the plane?

This graph has 20 vertices, each of degree 3, so $v = 20$. Because the sum of the degrees of the vertices, $3v = 3 \cdot 20 = 60$, is equal to twice the number of edges, $2e$, we have $2e = 60$, or $e = 30$. Consequently, from Euler's formula, the number of regions is

$$r = e - v + 2 = 30 - 20 + 2 = 12$$

b)



r = 11 -7+2 = 6

Note :

1) If G is a connected planar simple graph with e edges and v vertices, where $v \geq 3$, then $e \leq 3v - 6$.
2) If G is a connected planar simple graph, then G has a vertex of degree not exceeding five.
3) If a connected planar simple graph has e edges and v vertices with $v \geq 3$ and no circuits of length three, then $e \leq 2v - 4$.

Show that K_5 is non-planar

sol: The graph K_5 has five vertices and 10 edges. However, the inequality $e \leq 3v - 6$ is not satisfied for this graph because $e = 10$ and $3v - 6 = 9$. Therefore, K_5 is not planar.

Show that $K3,3$ is nonplanar

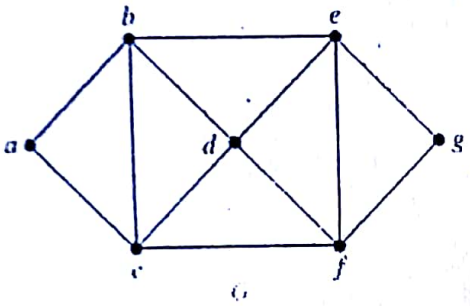
sol : Because $K_{3,3}$ has no circuits of length three $K_{3,3}$ has six vertices and nine edges. Because $e = 9$ and $2v - 4 = 8$, Corollary 3 shows that $K_{3,3}$ is nonplanar.

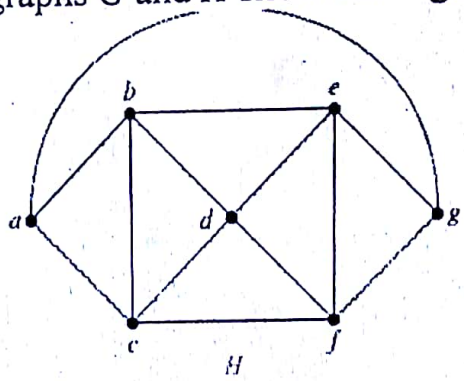
Graph Coloring

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.

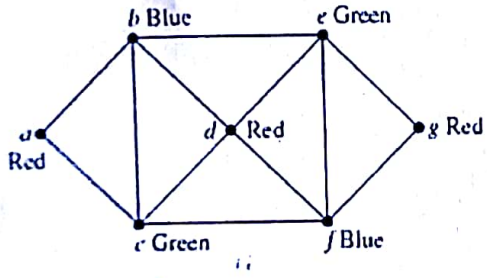
The *chromatic number* of a graph is the least number of colors needed for a coloring of this graph. The chromatic number of a graph G is denoted by $\chi(G)$. (Here χ is the Greek letter *chi*.)

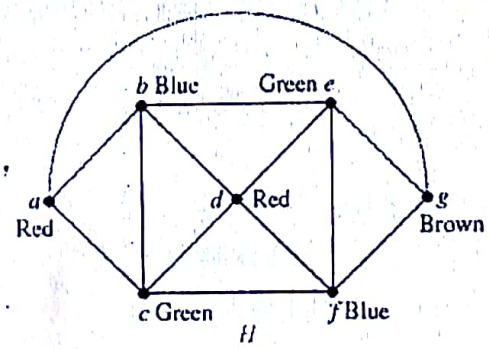
What are the chromatic numbers of the graphs G and H shown in Figure below?





sol.





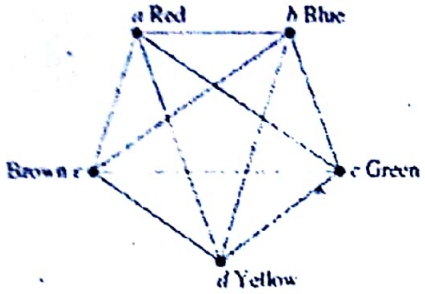


FIGURE 5 A Coloring of K_5.

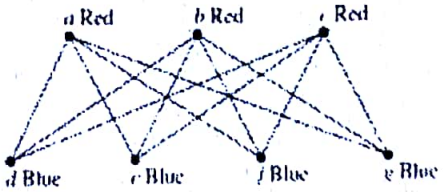


FIGURE 6 A Coloring of $K_{3,4}$.

Note:

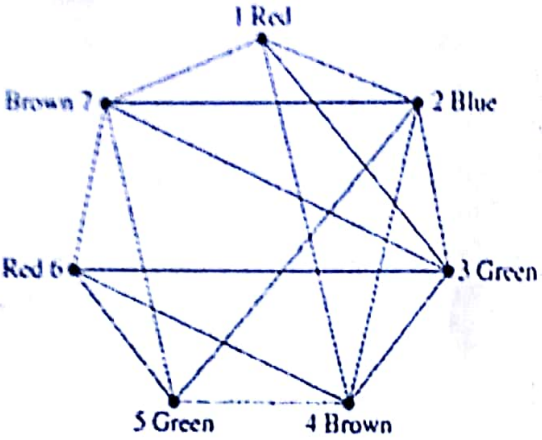
1. $\chi(K_n) = n$.
2. $\chi(K_{m,n}) = 2$.
3. $\chi(Cn) = 2$ if n is an even positive integer with $n \geq 4$ and $\chi(Cn) = 3$ if n is an odd positive integer with $n \geq 3$.
4. The chromatic number of a planar graph is no greater than four.

Applications of Graph Colorings

Graph coloring has a variety of applications to problems involving scheduling and assignments.

Scheduling Final Exams How can the final exams at a university be scheduled so that no student has two exams at the same time?

For instance, suppose there are seven finals to be scheduled. Suppose the courses are numbered 1 through 7. Suppose that the following pairs of courses have common students: 1 and 2, 1 and 3, 1 and 4, 1 and 7, 2 and 3, 2 and 4, 2 and 5, 2 and 7, 3 and 4, 3 and 6, 3 and 7, 4 and 5, 4 and 6, 5 and 6, 5 and 7, and 6 and 7. In Figure 8 the graph associated with this set of classes is shown. A scheduling consists of a coloring of this graph.



Time Period	Courses
I	1, 6
II	2
III	3, 5
IV	4, 7

FIGURE 9 Using a Coloring to Schedule Final Exams.