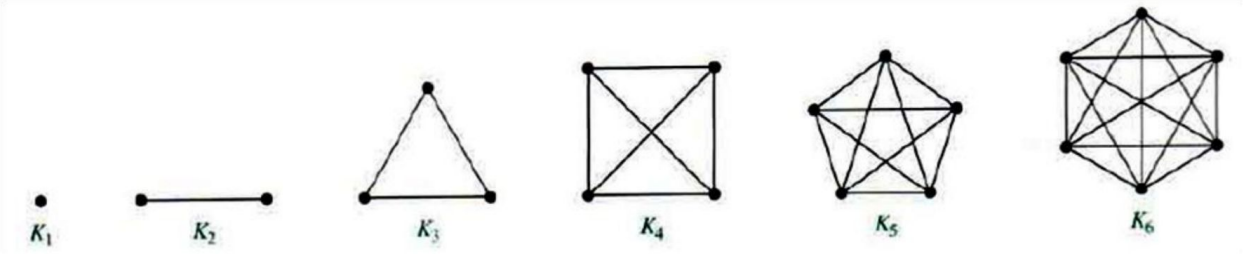
3.2

Special Types of Graphs and Representation of graphs

Some special simple Graphs

The following graphs are often used as examples and arise in many applications.

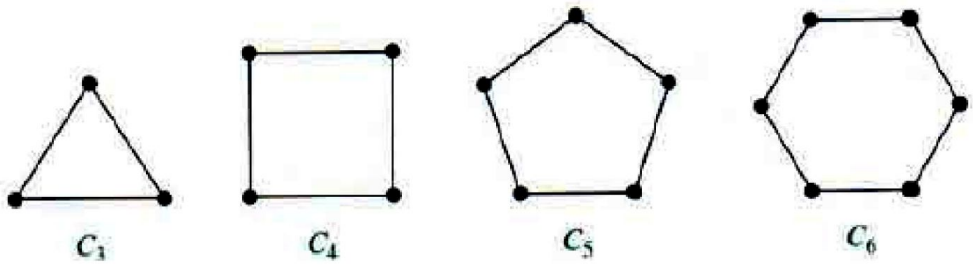
Complete Graphs: The complete graph on n vertices, denoted by K_n is the simple graph that contains exactly one edge between each pair of distinct vertices. The following are the complete graphs K_n , for $n = 1,2,3,4,5,6$:



In K_n ,the number of vertices is n, the number of edges is ${}^nC_2$ and the degree of each vertex is $n - 1$.

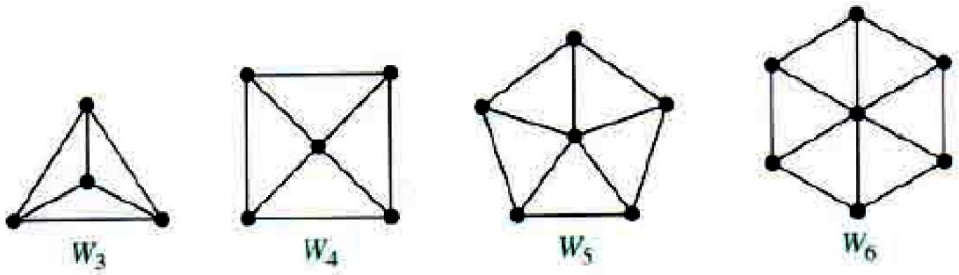
Cycles: The **cycle C_n** , $n \geq 3$, consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, …$\{v_{n-1}, v_n\}$ and $\{v_n, v_1\}$.

The following are the cycles C_3 , C_4 , C_5 and C_6.



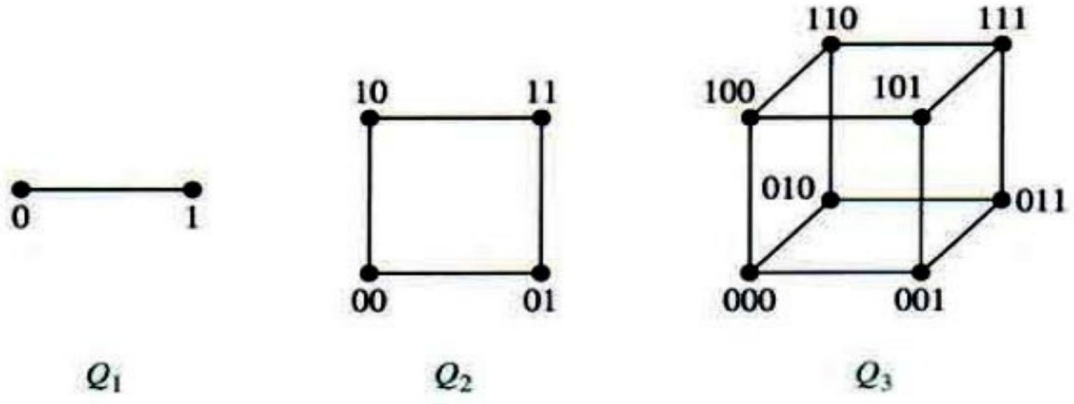
In a cycle graph C_n , $n \geq 3$; the number of vertices is n, the number of edges is n and the degree of each vertex is 2.

Wheels: We obtain the **wheel** when we add an additional vertex to the 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 shown below.



In a wheel graph W_n , $n \geq 3$; the number of vertices is $n + 1$, the number of edges is $2n$ and the degree of each vertex, except the additional vertex is 3 and the degree of the additional vertex is n.

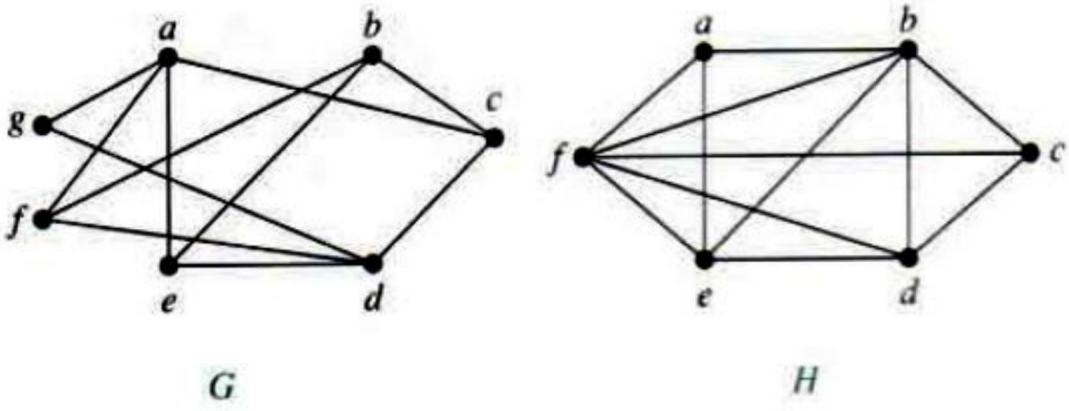
***n*-cubes:** The ***n*-cube** or the ***n*-dimensional hypercube**, denoted by Q_n; is the graph that has vertices representing the 2^n bit strings of length n. Two vertices are *adjacent* iff the strings that they represent differ in exactly one bit position. The graph Q_1, Q_2 and Q_3 are shown below:



Note: The $(n + 1) -$ cube Q_{n+1} can be constructed from the n-cube Q_n by making two copies of Q_n , prefacing the labels on the vertices with a 0 in one copy of Q_n and with a 1 in the other copy of Q_n and connecting two vertices that have labels differing only in the first bit by edges.

Bipartite graphs: A simple graph $G = (V, E)$ 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). If this condition holds, then we call the pair (V_1, V_2) a **bipartition** of the vertex set V of G.

Example 1: Are the graphs G and H, shown below are bipartite?



Solution:

(i) Graph G is bipartite because its vertex set $V = \{a, b, c, d, e, f, g\}$ is the union of two disjoint sets $V_1 = \{a, b, d\}$ and $V_2 = \{c, e, f, g\}$ and each edge of G connects a vertex of V_1 to a vertex of V_2.
Note: A graph G to be bipartite it is not necessary that every vertex of V_1 be adjacent to every vertex of V_2. In this case b, g are not adjacent.

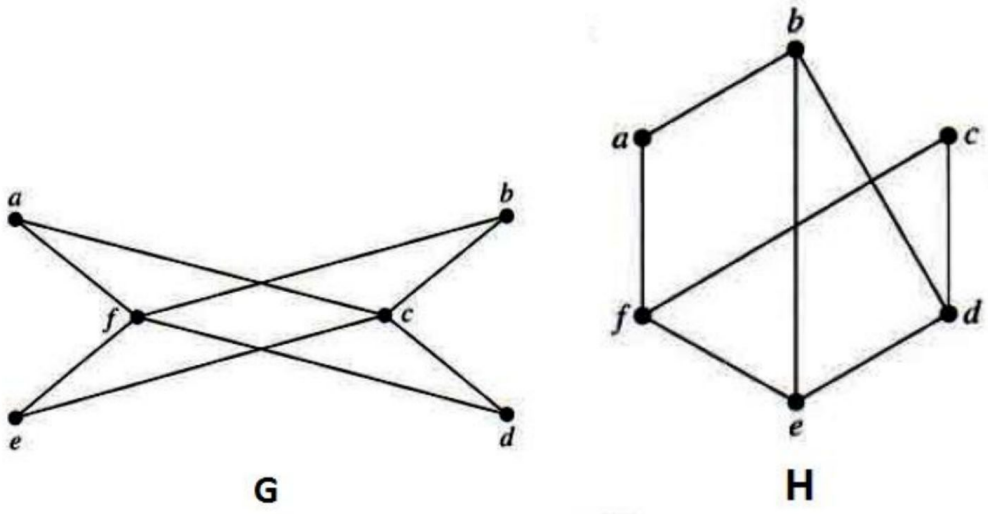
(ii) The graph H is not bipartite because its vertex set $V = \{a, b, c, d, e, f\}$ cannot be partitioned into two subsets so that edges do not connect two vertices of the same set.

The following theorem provides a useful criterion for determining whether a given simple graph is bipartite.

Theorem 1: A simple graph is bipartite iff it is possible to assign one of two different colours to each vertex of the graph so that no two adjacent vertices are assigned the same colour.

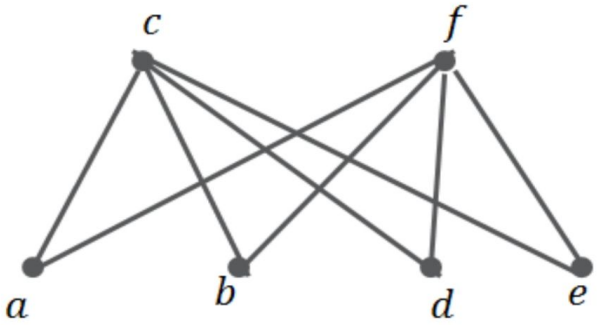
Note: This theorem is an example of a result in graph colourings. Graph colourings are an important part of graph theory with important applications.

Example 2: Determine whether the following graphs are bipartite using Theorem 1.



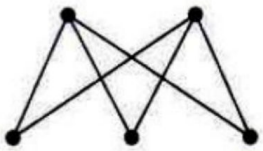
Solution:

(i) We first consider the graph G. We take two colours, **red**, **blue** and assign one of the two colours to each vertex in G. We begin and arbitrarily assign red to a. Then we must assign blue to c and f because each of these vertices is adjacent to a. We assign red to all vertices adjacent to c and f. We have now assigned colours to all vertices of G, with c and f blue and a, b, d and e are red. Note that every edge connected a red vertex and a blue vertex. Therefore, by Theorem 1, G is bipartite and it can be redrawn as

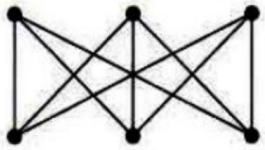


(ii) We now consider the graph H. We take two colours, red, blue and assign one of the two colours to each vertex in H. We begin and arbitrarily assign red to a. Then we must assign blue to b and f because each of these vertices is adjacent to a. Now d, e are adjacent to b, we assign red to d, e. Now note that d, e are adjacent and assigned the same colour red. Therefore, H is not bipartite by Theorem 1.

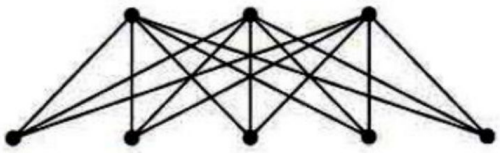
Complete bipartite graphs: The complete bipartite graph, denoted by $\boldsymbol{K_{m,n}}$, is the graph that has its vertex set partitioned into two subsets V_1 and V_2 of m and n vertices respectively. There is an edge between two vertices iff one vertex is in V_1 and the other vertex is in V_2. ($i.e.$, each vertex in V_1 is adjacent to every vertex in V_2. Therefore, there are $m + n$ vertices and mn edges in $K_{m,n}$ and $\deg(v) = n$ when $v \in V_1$, $\deg(v) = m$ when $v \in V_2$).The following are the complete bipartite graphs $K_{2,3}$, $K_{3,3}$, $K_{3,5}$ and $K_{2,6}$.



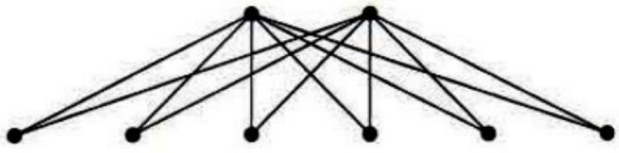
$K_{2,3}$



$K_{3,3}$



$K_{3,5}$

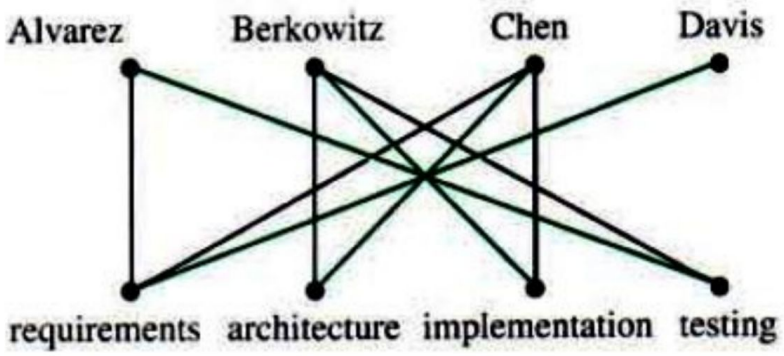


$K_{2,6}$

Some applications of special types of graphs

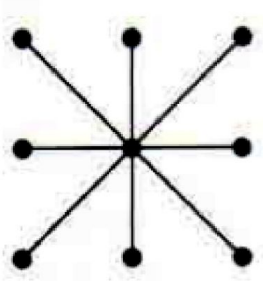
Job assignments: Suppose that there are m employees in a group and j different jobs that need to be done where $m \leq j$. Each employee is trained to do one or more of these j jobs. We represent each employee by a vertex and each job by a vertex. For each employee, we include an edge from the vertex representing that employee to the vertices representing 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 vertices representing employees and the set of vertices representing the jobs, and each edge connects a vertex representing an employee to a vertex representing a job. Thus it represents a bipartite graph.



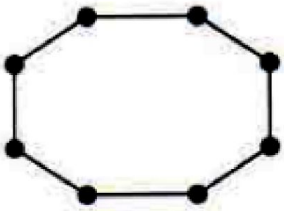
Local Area Networks (LANs)

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 below:



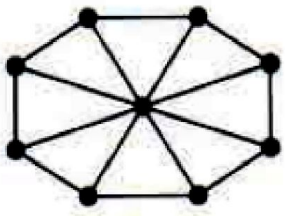
Star topology for LANs

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 below: Messages are sent from device to device around the cycle until the intended recipient of a message is reached.



Ring topology for LANs

Some LANs use hybrid of these two topologies. Messages may be sent around the ring, or through a central device. This redundancy makes the network more reliable. LANs with this redundancy can be modeled using wheels W_n as shown below:



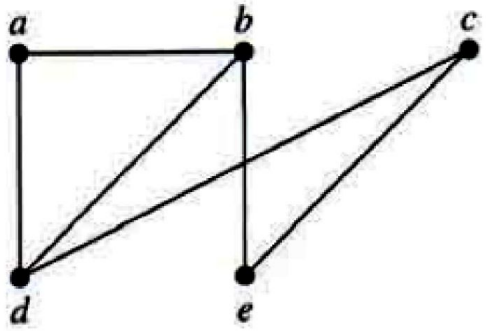
Hybrid topology for LANs

Complete graphs K_n and hypercubes Q_n have applications in interconnection networks for parallel computation.

Representation of Graphs- Adjacency Lists

One way to represent a graph without multiple edges is to list all the edges of the graph. Another way to represent a graph with no multiple edges is the use of **adjacency lists**, which specify the vertices that are adjacent to each vertex of the graph.

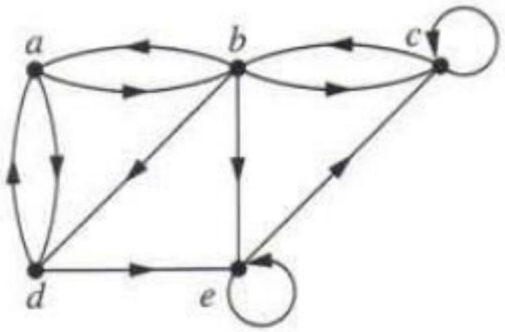
Example 3: Use an adjacency list to represent the following graph:



Solution: We list those vertices adjacent to each of the vertices of the graph.

Adjacency list of the given digraph	
Vertex	Adjacent vertices
a	b, d
b	a, d, e
c	d, e
d	a, b, c
e	b, c

Example 4: Use an adjacency list to represent the given digraph.



Solution: We represent the digraph by listing all the vertices that are the terminal nodes of edges starting at each vertex of the graph.

Adjacency list of the given digraph	
Initial vertex	Terminal vertices
a	b, d
b	a, c, d, e
c	b, c
d	a, e
e	c, e

Representation of Graphs- Adjacency Matrices

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

Two types of matrices are commonly used to represent the graphs. One is based on the adjacency of vertices and the other is based on the incidence of vertices and edges.

Adjacency matrix of a simple graph: Let $G = (V, E)$ be a simple undirected graph, where $|V| = n$. Suppose that the vertices of V are ordered and listed as $v_1, v_2, \dots, v_n$. The adjacency matrix of G w.r.t this ordering is the $n \times n$ zero-one matrix, denoted by $\boldsymbol{A}$ or $\boldsymbol{A_G}$ where $A = (a_{ij})_{n \times n}$ defined by

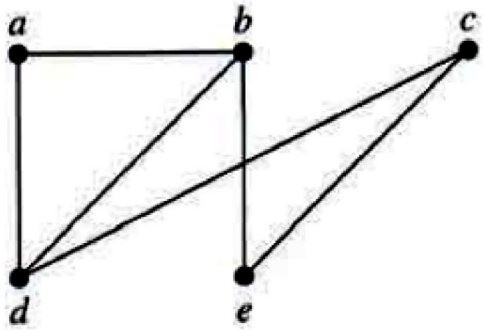
$$a_{ij} = \begin{cases} 1 & if\ \{v_i, v_j\}\ is\ an\ edge\ of\ G \\ 0 & otherwise \end{cases}$$

Note:

(i) The adjacency matrix of a simple undirected graph is a Boolean matrix and it is symmetric.

(ii) The adjacency matrix of a graph is bases on the ordering of the vertices. Therefore, there are $n!$ different adjacency matrices for a graph with n vertices, because there are $n!$ different orderings of n vertices.

Example 5: Represent the following graph with an adjacency matrix.



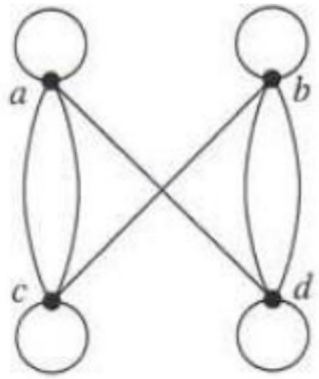
Solution: We order the vertices as a, b, c, d, e. The adjacency matrix A of the graph w.r.t. this ordering is the following Boolean matrix of order 5.

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

Adjacency matrix of multigraphs: 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 are present, the adjacency matrix is no longer a zero-one matrix, because the $(i, j)^{th}$ entry of the adjacency matrix is the number of edges that are associated with the edge $\{v_i, v_j\}$.

All undirected graphs have symmetric adjacency matrices.

Example 6: Represent the following graph using an adjacency matrix.



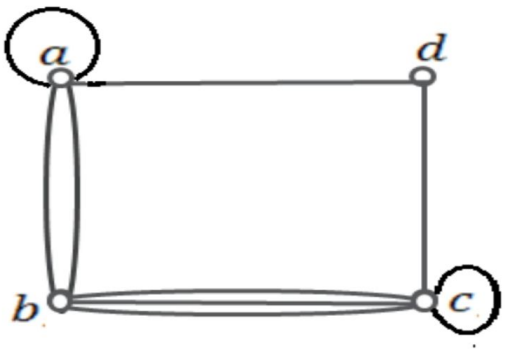
Solution: The adjacency matrix using the ordering of vertices a, b, c, d is

$$A = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$$

Example 7: Draw an undirected graph represented by the following adjacency matrix.

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

Solution: We draw the graph w.r.t the ordering of the vertices a, b, c, d.



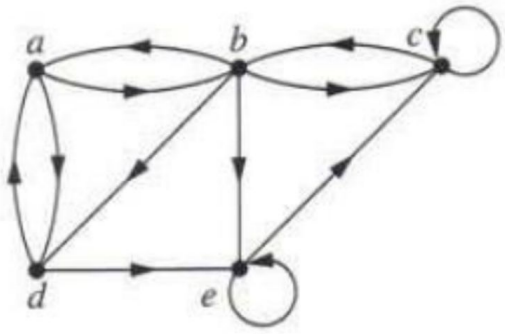
Adjacency matrix of simple digraphs:

Let $G = (V, E)$ be a simple digraph with $|V| = n$. Suppose that the vertices of V are ordered and listed as $v_1, v_2, \dots, v_n$. The $n \times n$ matrix $A = (a_{ij})_{n\times n}$ is the adjacency matrix of the simple digraph G if

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

Note: The adjacency matrix of a simple directed graph is a Boolean matrix and it does not have to be symmetric, because there may not be an edge (v_j, v_i) when there is an edge (v_i, v_j).

Example 8: Represent the following digraph with an adjacency matrix.

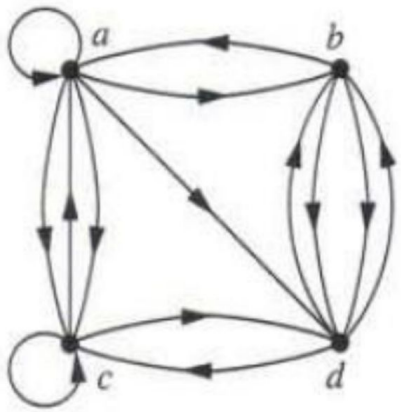


Solution: The adjacency matrix using the ordering of vertices a, b, c, d, e is

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

Adjacency matrices can also be used to represent directed multigraphs. In the adjacency matrix $A = (a_{ij})$ for a directed multigraph, a_{ij} is the number of edges that are associated to the edge $(v_i\ , v_j)$. Such matrices are not zero-one matrices and not symmetric in general.

Example 9: Represent the following directed multigraph using an adjacency matrix:



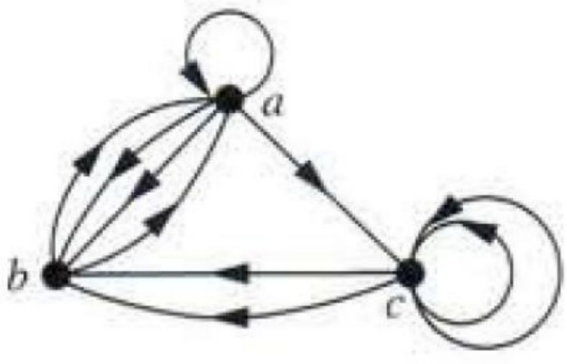
Solution: The adjacency matrix using the ordering of vertices a, b, c, d is

$$A = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 1 & 0 & 0 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

Example 10: Draw the digraph represented by the following adjacency matrix

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 0 \\ 0 & 2 & 2 \end{bmatrix}$$

Solution: We draw a digraph with the given adjacency matrix w.r.t the ordering of vertices a, b, c.



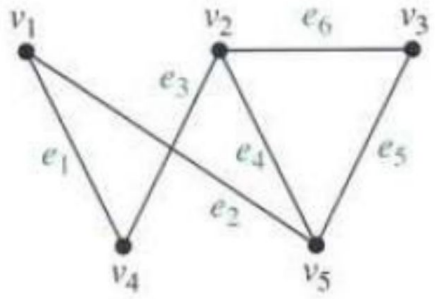
Representation of graphs-Incidence Matrices

Another common way to represent graphs is to use **incidence matrices**.

Let $G = (V, E)$ be an undirected graph. Let $v_1, v_2, \ldots, v_n$ be vertices and $e_1, e_2, \ldots, e_m$ be its edges. Then the incidence matrix w.r.t this ordering of vertices of V and edges of E is the $n \times m$ matrix $\boldsymbol{M} = \left(\boldsymbol{m_{ij}}\right)_{\boldsymbol{n \times m}}$, where

$$\boldsymbol{m_{ij}} = \begin{cases} \mathbf{1} & \boldsymbol{when\ edge\ e_j\ incident\ with\ vertex\ v_i} \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Example 11: Represent the following simple graph with an incidence matrix.

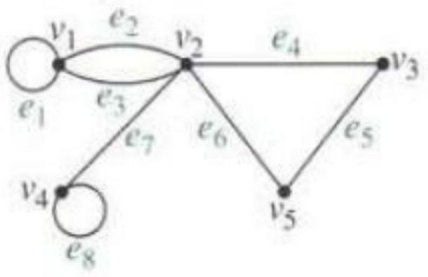


Solution: The incidence matrix using the ordering of vertices v_1, v_2, v_3, v_4, v_5 and edges $e_1, e_2, \ldots, e_6$ is

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

It is easy to write column wise. For example e_1, incident with v_1 and v_4.

Example 12: Represent the following multigraph with an incidence matrix.



Solution: The incidence matrix using the ordering of vertices v_1, v_2, v_3, v_4, v_5 and edges $e_1, e_2, \ldots, e_8$ is

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

Trade-offs between adjacency lists and adjacency matrices:

When a simple graph (with n vertices) contains relatively few edges, it is usually preferable to use adjacency lists rather than adjacency matrix to represent the graph. If each vertex has degree not exceeding c, where c is much less than n, then there are cn terms in the adjacency list of the graph. On the other hand, the adjacency matrix for the graph has n^2 entries (where cn is much less than n^2)

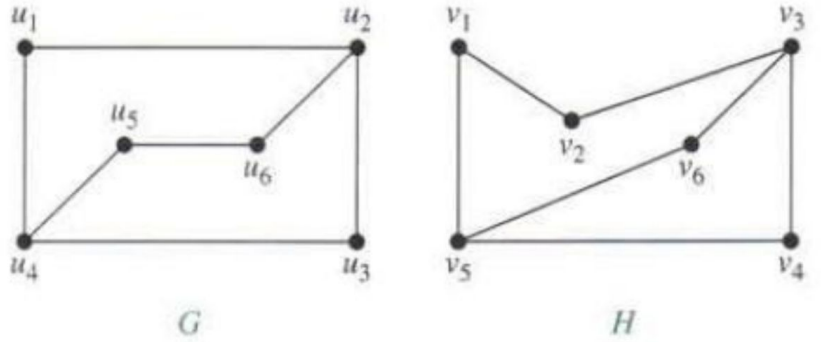
If a simple graph contains more than half of all possible edges then using an adjacency matrix to represent the graph is usually preferable over using the adjacency lists. This may be seen by comparing their complexities.

Isomorphic graphs and adjacency matrices:

Let $G = (V, E)$ and $G' = (V', E')$. To show that $f: V \longrightarrow V'$ is an isomorphism, we need to show that f preserves the presence and absence of edges. One helpful way to do this is to show that the adjacency matrix of G is the same as the

adjacency matrix of H with rows and columns labeled by the images under f of the corresponding vertices of G.

Example 13: Determine whether the following G and H are isomorphic.



Solution: Both G and H are undirected graphs and both have six vertices and seven edges. Further, both have the degree sequence $3, 3, 2, 2, 2, 2$.

Notice that in G, $deg(u_1) = 2, u_1$ is not adjacent to any other vertex of degree two and u_1 is adjacent to two vertices u_2, u_4 of degree 3.

In H the vertices with the above characters are v_4 and v_6. We arbitrarily set $f(u_1) = v_4$. Since u_2 is adjacent to u_1, the possible images of u_2 are v_3 and v_5. We arbitrarily set $f(u_2) = v_3$. Considering adjacency and degree we continue and set $f(u_3) = v_6, f(u_4) = v_5, f(u_5) = v_1, f(u_6) = v_1$.

Thus, f is a bijection from the vertex set of G to the vertex set of H. To verify whether f preserves the adjacency, we write down the adjacency matrices of G and H.

The adjacency matrix A_G of G w.r.t the ordering of the vertices $u_1, u_2, \dots, u_6$ and A_H of H w.r.t the ordering of the vertices $v_4, v_3, v_6, v_5, v_1, v_2$ are

$$A_G = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} = A_H$$

This shows that f preserves the adjacency. Thus, f is an isomorphism and $G \cong H$.

Note that $f(u_1) = v_6, f(u_2) = v_3, f(u_3) = v_4, f(u_4) = v_5, f(u_5) = v_1, f(u_6) = v_2$ is also an isomorphism. Try others!

Example 14: Are the simple graphs with the following adjacency matrices isomorphic?

$a)$ $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

$b)$ $\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$

$c)$ $\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$

Solution: Let G, H be simple graphs and let A_G, A_H be their adjacency matrices, where $A_G = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, A_H = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

Notice that both G, H have the 3 vertices and 4 edges. Now,

$$A_H = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \xleftrightarrow{R_1 \leftrightarrow R_3} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \xleftrightarrow{C_1 \leftrightarrow C_3} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} = A_G$$

(Note that for each exchange of rows, we have to immediately perform the corresponding exchange of columns: exchange of row and column operations are only to be performed)

Aliter:

a) Let the vertices of H be p, q, r. Let A_H be the adjacency matrix of H w.r.t the ordering of vertices p, q, r. Now, the adjacency matrix B of H w.r.t the ordering of the vertices r, q, p.

$$B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} = A_G$$

Thus, $G \cong H$.

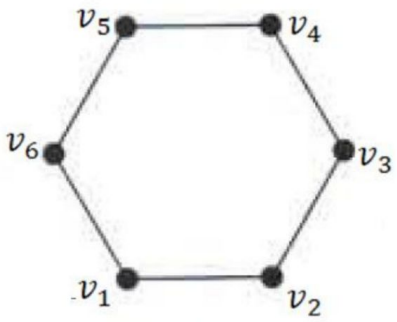
b) Notice that the first graph has 8 edges and the second graph has 10 edges. Therefore, the graphs are not isomorphic.

c) Notice that the first graph has 8 edges and the second graph has 6 edges. Therefore, the graphs are not isomorphic.

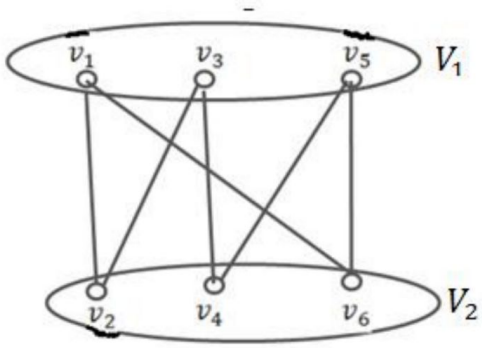
P1:

Is the Cycle C_6 bipartite?

Solution:



C_6



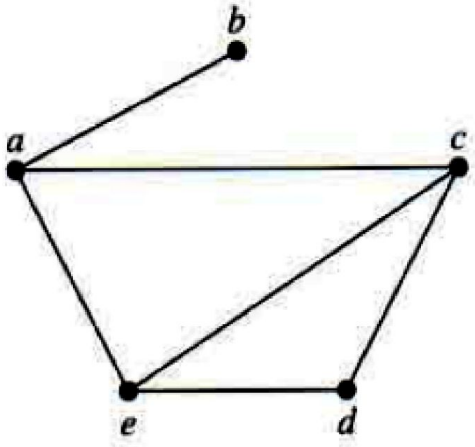
G

The vertex set $V = \{v_1, v_2, \dots, v_6\}$ can be partitioned into the sets $V_1 = \{v_1, v_3, v_5\}$ and $V_2 = \{v_2, v_4, v_6\}$.Note that edges of C_6 connects every vertex of V_1 and a vertex V_2 and no two vertices of V_1 and V_2 are adjacent (see the diagram of G). Therefore, C_6 is bipartite.

Is C_7 bipartite?

P2:

Use an adjacency list to represent the following graph:



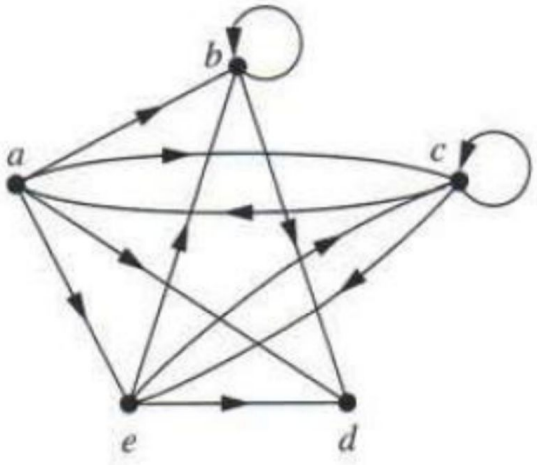
Solution:

We list those vertices adjacent to each of the vertices of the graph.

Adjacency list of the given digraph	
Vertex	Adjacent vertices
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

P3:

Use an adjacency list to represent the given digraph



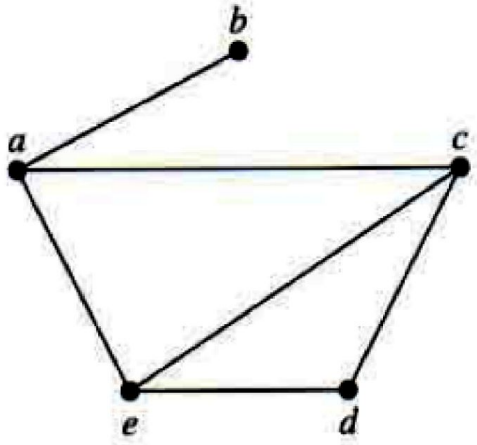
Solution:

We represent the digraph by listing all the vertices that are the terminal nodes of edges starting at each vertex of the graph.

Adjacency list of the given graph	
Initial vertex	Terminal vertices
a	b, c, d, e
b	b, d
c	a, c, e
d	$-$
e	b, c, d

P4:

Represent the following graph with an adjacency matrix



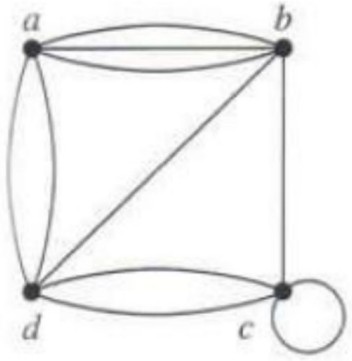
Solution:

We order the vertices as a, b, c, d, e. The adjacency matrix A of the graph w.r.t. this ordering is the following Boolean matrix of order 5.

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

P5:

Represent the following graph using an adjacency matrix.



Solution:

The adjacency matrix using the ordering of vertices a, b, c, d is

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

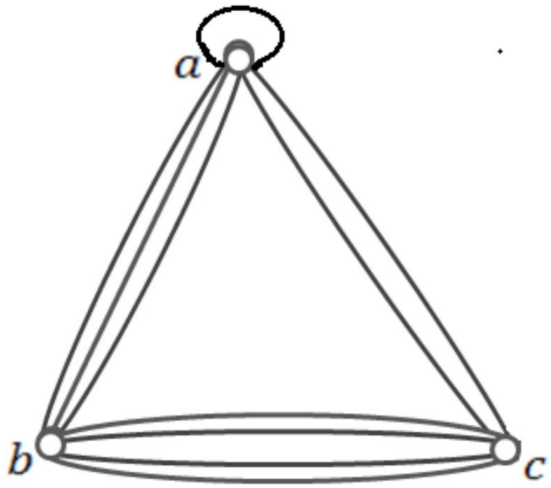
P6:

Draw an undirected graph represented by the following adjacency matrix.

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 0 & 4 \\ 2 & 4 & 0 \end{bmatrix}$$

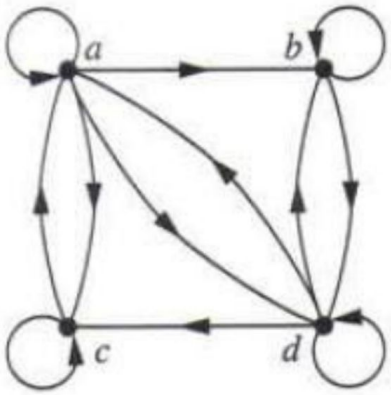
Solution:

We draw the graph w.r.t the ordering of the vertices a, b, c.



P7:

Represent the following digraph with an adjacency matrix.



Solution:

The adjacency matrix using the ordering of vertices a, b, c, d is

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

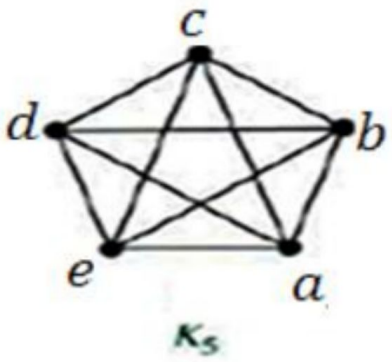
P8:

Represent each of these graphs with an adjacency matrix.

a) K_5 b) $K_{2,3}$ c) C_5 d) W_5 e) Q_3

Solution:

a) K_5

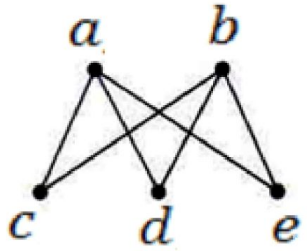


K_5

The ordering of the vertices is a, b, c, d, e. Then the adjacency matrix of K_5 is

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

b) $K_{2,3}$

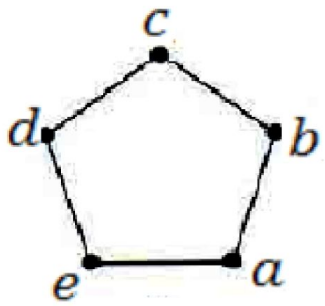


$K_{2,3}$

The vertices are ordered as a, b, c, d, e. Then the adjacency matrix w.r.t this ordering is

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

c) C_5

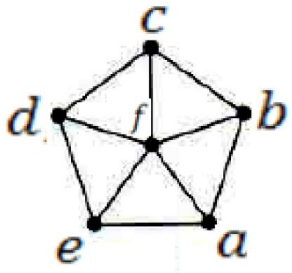


C_5

The vertices are ordered as a, b, c, d, e. The adjacency matrix w.r.t this ordering is

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

d) W_5



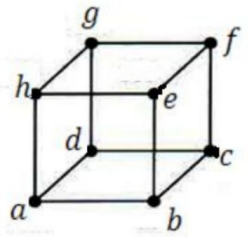
W_5

The vertices are ordered as a, b, c, d, e, f. The adjacency matrix w.r.t this ordering is

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

e) Q_3

The vertices are ordered as a, b, c, d, e, f, g, h. The adjacency matrix w.r.t this ordering is



Q_3

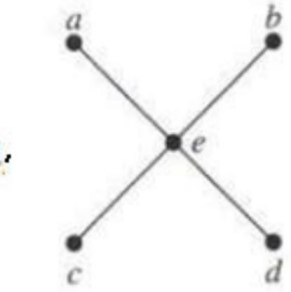
$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

3.2

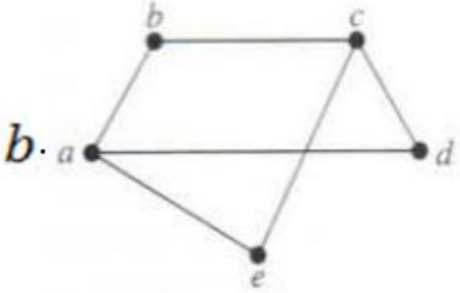
Exercise:

1. Determine whether the following graphs are bipartite by using Theorem 1.

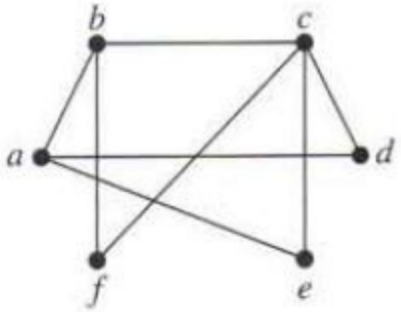
a.



b.



c.

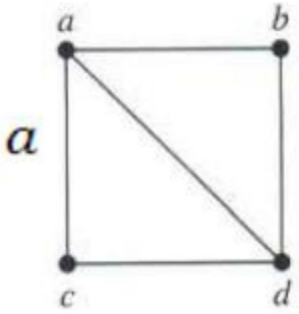


2. Draw the following graphs

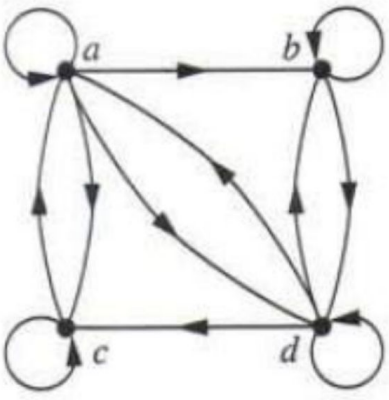
   a. K_7 b. $K_{1,8}$ c. $K_{4,4}$ d.C_7 e.W_7 f.Q_4

3. Use an adjacency list to represent the given graph

a



b



4. Represent the above graph 3 (a) & (b) with their adjacency matrices.
5. Represent the given graph using an adjacency matrix.