MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

DISCRETE MATHEMATICS AND GRAPH THEORY

Unit 14
Graphs and Their Properties

Table of Contents

1. INTRODUCTION

Graphs may be represented in a variety of helpful ways. As we shall see throughout this chapter, having the option to choose a graph's most practical representation is useful when dealing with it. This section will demonstrate numerous alternative methods to depict graphs. There are situations when two graphs have the exact same shape. In this part, we'll look at an essential graph theory problem: determining if two graphs are isomorphic.

Paths that are created by moving along the edges of graphs may be used to describe a variety of issues. A graph model, for instance, may be used to investigate the issue of determining if a message can be transmitted between two computers utilizing intermediary connections. Models involving pathways in graphs may be used to effectively design routes for mail delivery, trash collection, computer network diagnostics, and other tasks.

1.1. Objectives:

At studying this unit, you should be able to:

- ❖ *Understand concept of Isomorphism and Subgraphs.*
- ❖ *Apply concept of walk, path, Circuits.*
- ❖ *Understand connectedness in graph.*

2. ISOMORPHISM OF GRAPHS

2.1. Isomorphism

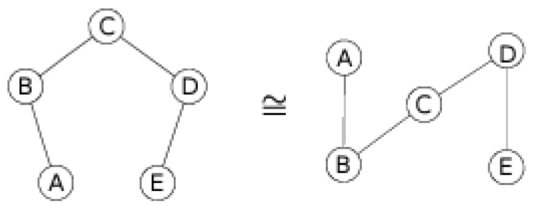
The simple graphs G1 = (V1,E1) and G2 = (V2,E2) are isomorphic if there exists a one to- one and onto function f from V1 to V2 with the property that a and b are adjacent in G1 if and only if f (a) and f (b) are adjacent inG2, for all a and b in V1. Such a function f is called an isomorphism.∗ Two simple graphs that are not isomorphic are called nonisomorphic.

In other words, two graphs G and G′ are said to be isomorphic (to each other) if there is one-to-one correspondence between their vertices and between their edges such that adjacency of vertices is preserved.

Such graph will have same structure; they differ only in the way their vertices and edges are labelled or only in the way they represented geometrically.

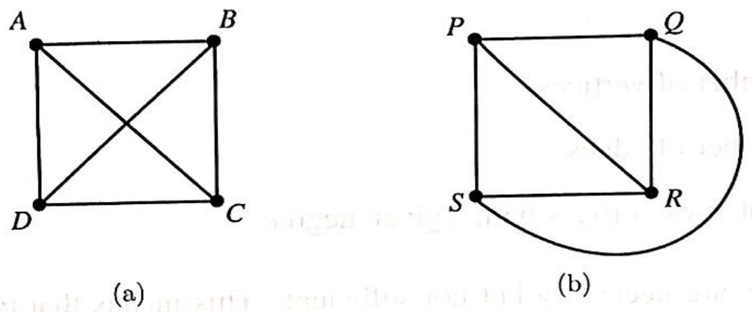
When G and G′ are isomorphic, we write G ≅ G′.

Example: 1-1 matching between the vertices so that if pairs of vertices are connected by an edge in one graph, then corresponding vertices will be connected in the other.



The above graphs are isomorphic each other.

Example: **Prove that the graphs shown below are isomorphic.**



Solution

In graphs shown above Fig Consider the following

one-to-one correspondence between the vertices of these two graphs

$A \leftrightarrow P, B \leftrightarrow Q, C \leftrightarrow R, D \leftrightarrow S$

Under this correspondence, the edge in two graphs correspond with each other, as indicated below:

$$\{A, B\} \leftrightarrow \{P, Q\},\ \{A, C\} \leftrightarrow \{P, R\}, \{A, D\} \leftrightarrow \{P, S\}$$

$$\{B, C\} \leftrightarrow \{Q, R\},\ \{B, D\} \leftrightarrow \{Q, S\},\ \{C, D\} \leftrightarrow \{R, S\}$$

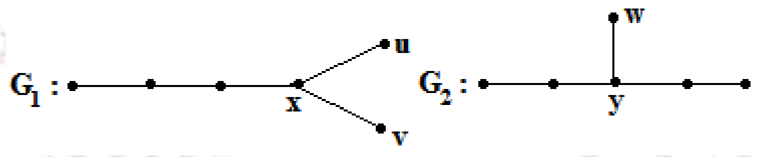
Therefore two graphs are isomorphic.

Note: From the definition of isomorphism of graphs, it follows that if two graphs are isomorphic then they must have:

1. The same number of vertices
2. The same number of edges
3. An equal number of vertices with a given degree

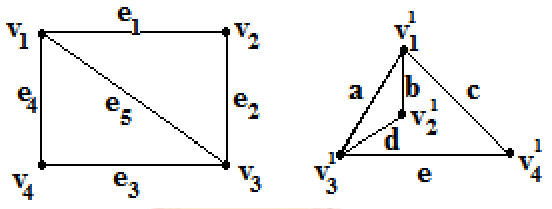
These conditions are necessary but not sufficient.

Example: Consider the graphs



The graphs in this Figure satisfy all three conditions. Yet they are not isomorphic can be shown as follows. The graph G_1 has a vertex x of degree 3 which is adjacent to two pedent vertices u and v. But in G_2, the vertex y of degree 3 is a adjacent to only one pendent vertex w. Hence adjacency is not preserved. Therefore $G_1 \cong G_2$.

Example: **Prove that the graphs shown below are isomorphic**



Solution:

Vertices correspondence	Edge correspondence
$v_1 \rightarrow v_1'$	$(v_1, v_2) = e_1 \rightarrow b$
$v_2 \rightarrow v_2'$	$(v_2, v_3) = e_2 \rightarrow d$
$v_3 \rightarrow v_3'$	$(v_4, v_3) = e_3 \rightarrow e$
$v_4 \rightarrow v_4'$	$(v_1, v_4) = e_4 \rightarrow c$
	$(v_1, v_3) = e_5 \rightarrow a$

Therefore two graphs are isomorphic.

3. SUBGRAPHS

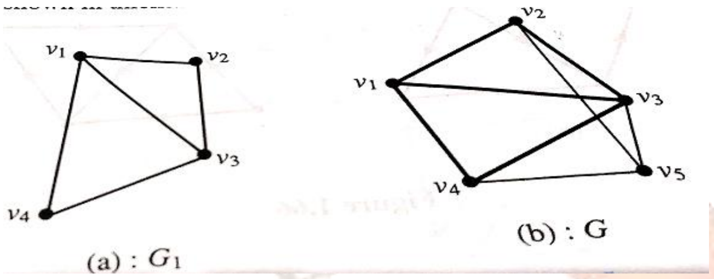
3.1. Subgraphs

A ordered pair H = (V', X') is said to be subgraph of a graph G=(V,X) if all the vertices and the edges of H are in G and each edge of H are in G and each edge of H has the same end vertices as in G i.e., $V' \subseteq V$ and $X' \subseteq X$ Clearly, any graph G is a subgraph of itself.

Note: Given two graphs G and G_1, we say that G_1 is a **subgraph** of G if the following condition holds.

1. All the vertices and all the edges of G_1 are in G.
2. Each edge of G_1 has the same end vertices in G as in G_1.

Example:



All vertices and all edges of the graph G_1 are in the graph G.

Every edge in G_1 has the same end vertices in G as in G_1.

Therefore, G_1 is a subgraph of G.

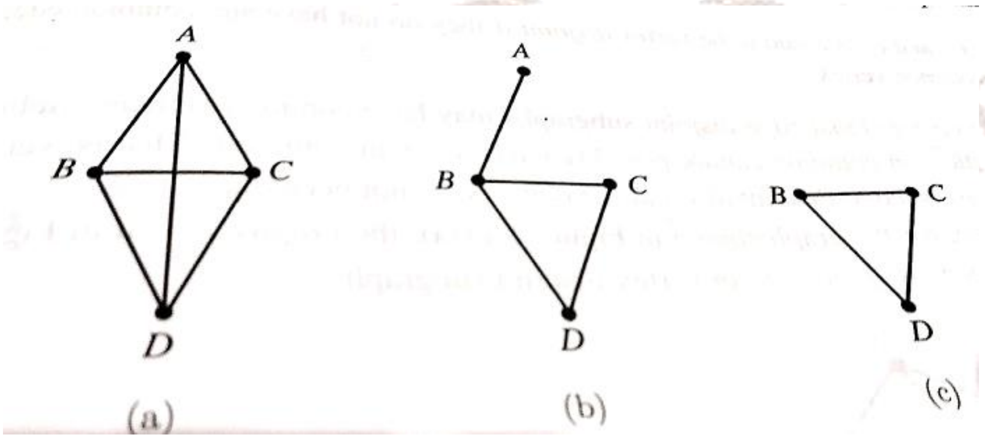
3.2. Spanning subgraphs

Given a graph $G = (V, E)$, if there is a subgraph $G_1 = (V_1, E_1)$ of G such that $V_1 = V$, then G_1 is called a spanning subgraph of G.

In other words, a subgraph G_1 of a graph G is a spanning subgraph of G whenever G_1 contains all vertices of G. Thus, a graph and all its spanning subgraphs have the same vertex set.

Every graph is its own spanning subgraph.

Example:

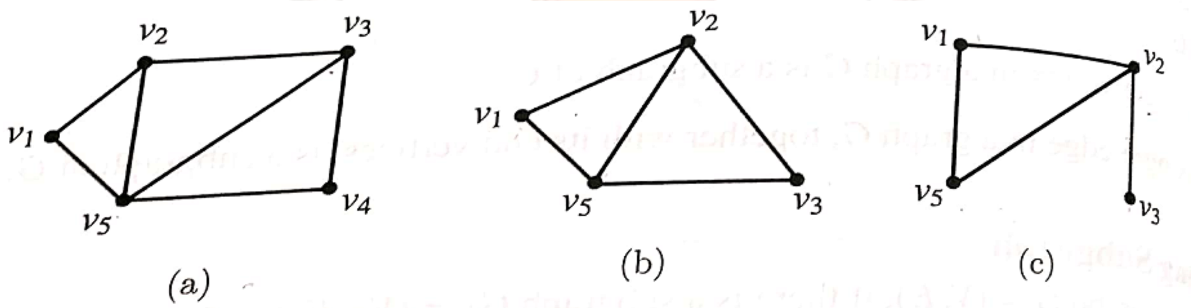


In the Fig (b) Is spanning subgraph of (a) and (c) Is not spanning subgraph of (a)

3.3. Induced subgraphs

Given a graph G = (V, E), suppose there is a subgraph $G_1 = (V_1, E_1)$ of G such that every edge of {A, B} of G, where A, B $\in V_1$ is an edge of G_1 also. Then G_1 is called an induced subgraph of G (induced by V_1) and is denoted by $< A >$.

Example:



In the Fig (b) Is induced subgraph of (a) and (c) Is not induced subgraph of (a)

4. DIGRAPH OR DIRECTED GRAPH

A digraph, or directed graph, is a fundamental concept in graph theory and discrete mathematics. It consists of a set of vertices (also called nodes) and a set of edges, where each edge has a direction associated with it, indicating a one-way relationship between two vertices. The key components and principles of a digraph are as follows:

4.1. Definition of a Digraph

- Vertices (Nodes): The individual objects or entities in a digraph. The set of vertices in a digraph is often denoted as V.
- Edges (Arcs): Ordered pairs of vertices that represent the connections or relationships between vertices. An edge in a digraph is represented as an ordered pair (u,v) where u and v are vertices, indicating an edge from u to v. The set of edges is often denoted as E.

4.2. Properties of a Digraph

- Direction: Each edge in a digraph has a direction, shown by an arrow pointing from the start vertex to the end vertex. This direction is crucial because it differentiates a digraph from an undirected graph, where edges have no orientation.
- Self-loop: A self-loop is an edge that connects a vertex to itself. In notation, it would be shown as (v,v) where the start and end vertices are the same.
- Multiplicity: Digraphs can have multiple edges (also called parallel edges) going from one vertex to another. However, each edge is considered distinct based on its start and end vertices or additional attributes like weight or label.
- Weight: Edges in a digraph can be assigned weights, making it a weighted digraph. Weights often represent costs, capacities, or lengths associated with traversing from one vertex to another.

4.3. Fundamental Concepts in Digraphs

- In-degree and Out-degree: The in-degree of a vertex is the number of edges coming into the vertex, and the out-degree is the number of edges going out from the vertex.
- Path: A sequence of edges that connects a sequence of vertices, following the direction of the edges. A path's length might be counted by the number of edges it traverses or the sum of the weights of those edges in a weighted digraph.
- Cycle: A path that starts and ends at the same vertex, without repeating any vertices or edges (except the start/end vertex).
- Connectivity: A digraph is strongly connected if there is a directed path from any vertex to every other vertex. It is weakly connected if replacing all of its directed edges with undirected edges results in a connected undirected graph.
- Subgraph: A subgraph of a digraph is another digraph formed from a subset of the vertices and edges of the original digraph, maintaining the original edge directions.

<u>Representation of relations:</u>

Matrix of a relation (Zero-one matrix):

Let R be a binary relation defined on a set A then a matrix $[m_{ij}]$ with

$$m_{ij} = \begin{cases} 1 \; if \; (a_i, b_j) \in R \;\; or \; a_i R b_j \\ 0 \; if \; (a_i, b_j) \notin R \;\; or \; a_i R b_j \end{cases}$$

The matrix $[m_{ij}]$ is called matrix of a relation. $[m_{ij}]$ is also called zero-one matrix.

Example

Let $A = \{1,2,3,4\}$ and $R = \{(1,1)(2,3),(3,4),(4,1)\}$ then $M_R = M(R) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

<u>Digraph or Directed graph of a relation:</u>

Let A relation R on set A $[R \subseteq A \times A]$ can be represented pictorially as follows

Step 1: Draw small circles representing the element of A. This circle is called **vertex or node.**

Step 2: Draw an arrow from vertex a_1 to vertex a_2 iff a_1 is related to a_2 . This arrow is called **edge.**

Note: If a_1 is related to itself then, we write the arrow which starts from vertex a_1 and ends at vertex a_1 and it is called **loop**

The resulting pictorial representation is known as directed graph of a relation.

<u>Note:</u>

In a digraph the number of edges leaving a vertex is known as **out degree** and number of edges which enters a vertex is known as **in degree** of a vertex.

Example:

Let $A = \{1,2,3,4,6\}$ Let R be a relation defined by xRy iff x is a multiple of y

a) *Write R as ordered pair.*

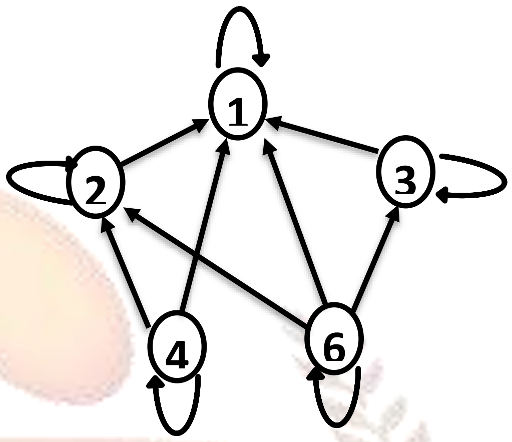
b) *Write $M(R)$.*

c) *Draw directed graph of relation.*

d) *Find indegree and out degree*.

Solution

(a) $R =$



$\{(1,1), (2,1), (3,1), (4,1), (6,1), (2,2), (4,2), (6,2), (3,3), (6,3), (4,4), (6,6)\}$.

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

Vertex	In degree	Out degree
1	5	1
2	3	2
3	2	2
4	1	3
6	1	4

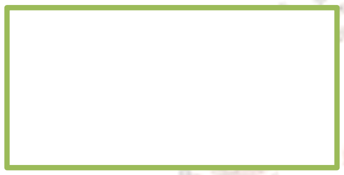
4.4. Orientation of Digraph

The orientation of a digraph refers to the assignment of directions to the edges of an undirected graph. Given an undirected graph, orienting its edges means assigning directions to each edge, transforming it into a directed graph (digraph). There can be multiple orientations for a given undirected graph. The process of orientation can be done in different ways based on the requirements of the specific problem or application. In some cases, certain orientations might introduce cycles or create a strongly connected graph, while other orientations might result in directed acyclic graphs (DAGs).

Note: There are 2|E| distinct orientations of G.

Example: For the following graph how many different orientations can be formed? Also write two different orientations.

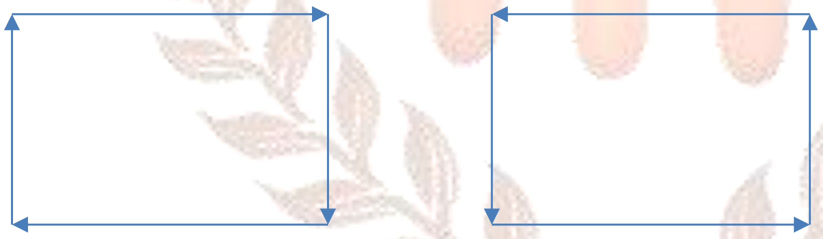
i)



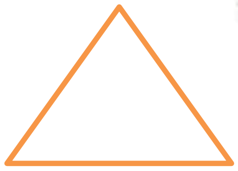
Solution:

The given graph has four edges i.e., |E|=4

Thus, the graph has 2|E|=24 distinct orientations.



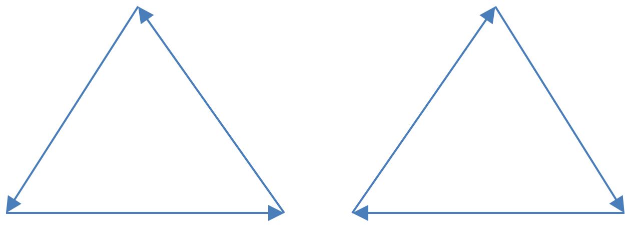
ii)



Solution:

The given graph has four edges i.e., |E|=3

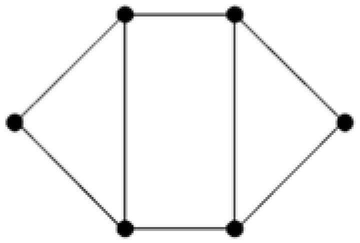
Thus, the graph has 2|E|=23 distinct orientations.



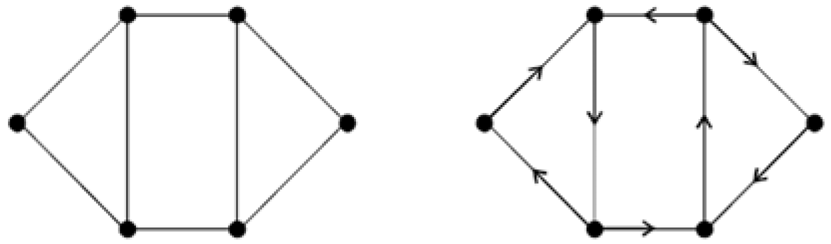
Example:

Convert the following underlying graph into Orientation of Graph

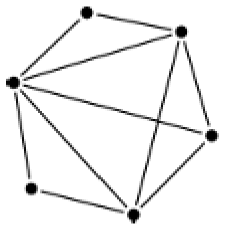
iii)



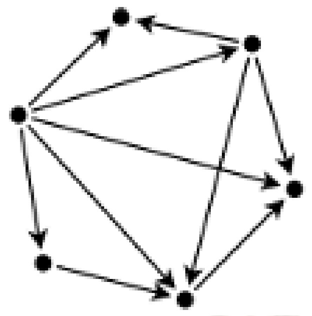
Solution

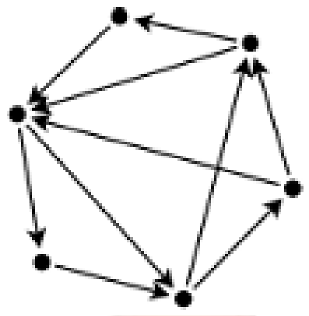


iv)



Solution





4.5. Tournament Graph

It’s a directed graph in which a single directed edge connects each pair of distinct vertices in one of the two possible directions.

In other words, a tournament graph is a complete graph where each edge is directed either from one vertex to the other or vice versa.

We often use tournament graphs to model situations where pairs of competitors face off against each other in a series of one-on-one matches, such as in a round-robin tournament.

Additionally, the direction of the edges in the graph represents the outcome of each match, with an edge pointing from the winner to the loser.

Note:

(a) A tournament graph is a directed graph with the property that no edge connects a vertex to itself, and between any two vertices there is at most one edge.

(b) A complete (or round-robin) tournament graph is a tournament graph with the property that between any two distinct vertices there is exactly one edge.

(c) A single-elimination tournament graph is a tournament graph with the properties that:

(i) one vertex (the champion) has no edge terminating at it and at least one edge initiating from it;

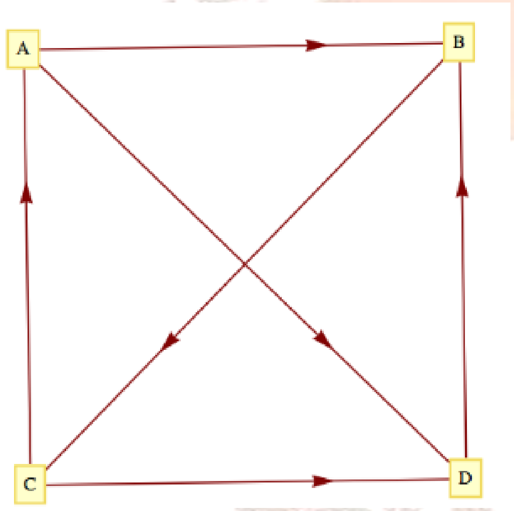
(ii) every other vertex is the terminal vertex of exactly one edge; and

(iii) there is a path from the champion vertex to every other vertex.

Note: If the edges of a complete graph are each given an orientation, the resulting directed graph is called a tournament.(Clearly, a tournament is an orientation of Kn)

Example:

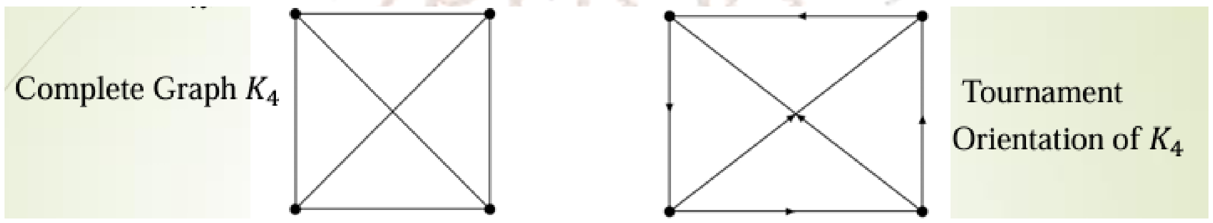
Suppose that four teams compete in a round-robin sporting event; that is, each team meets every other team once, and each game is played until a winner is determined. If the teams are named A, B, C, and D, then the graph can be written as follows



Example:

write the Tournament Orientation of $K4$

Solution:

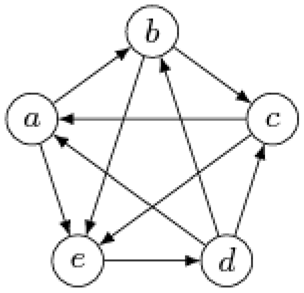


Example:

Convert the complete graph K5 into tournament graph and represent the graph in matrix form.

Solution

The directed K5 graph is the tournament graph which can be written as follows



T

The matrix representation of the above graph can be written as follows

$$\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0 & 1 & 0 & 0 & 1 \\ b & 0 & 0 & 1 & 0 & 1 \\ c & 1 & 0 & 0 & 0 & 1 \\ d & 1 & 1 & 1 & 0 & 0 \\ e & 0 & 0 & 0 & 1 & 0 \end{array}$$

$M(T)$

Example:

Consider a job interview process involving four candidates: Alice, Bob, Carol, and Dave. Describe how a tournament graph can be utilized to represent the outcomes of the interviews. Provide an example of the directed edges in the tournament graph based on hypothetical interview results. Additionally, explain the significance of the tournament graph in modeling competitive interactions in this context.

Solution

In the job interview process with candidates Alice, Bob, Carol, and Dave, a tournament graph can be employed to represent the outcomes of the interviews. Each directed edge in the graph signifies the result of an interview, where the tail of the arrow indicates the interviewer, and the head represents the interviewee.

Suppose the interview outcomes are as follows:

Alice interviewed by Bob (Alice -> Bob)

Alice interviewed by Carol (Alice -> Carol)

Alice interviewed by Dave (Alice -> Dave)

Bob interviewed by Carol (Bob -> Carol)

Bob interviewed by Dave (Bob -> Dave)

Carol interviewed by Dave (Carol -> Dave)

Now, let's construct the tournament graph:

$$\begin{tikzcd}
\text{Alice} \arrow[r] \arrow[d] \arrow[dr] & \text{Bob} \arrow[dl] \arrow[d] \\
\text{Carol} \arrow[r] & \text{Dave}
\end{tikzcd}$$

The directed edges capture the competitive interactions in the interview process. For instance, if Alice outperforms Bob in their interview, there is an edge from Alice to Bob (Alice -> Bob). The complete tournament graph forms a directed acyclic graph (DAG), ensuring that each candidate interviews every other candidate exactly once.

The significance of the tournament graph in this context lies in its ability to model and visualize the competitive dynamics among the candidates during the interview process. It provides a clear representation of who interviewed whom and the outcomes of these interactions, offering a structured and comprehensive view of the competitive relationships among the candidates.

5. WALKS AND THEIR CLASSIFICATION

5.1. Walk

Let G be a graph having at least one edge. In G, consider a finite, alternating sequence of the vertices and edges. A graph which begins and ends with vertices and which is such that each edge in the sequence is incident in the vertices preceding and following it in the sequence. Such a sequence is called a **walk** in G.

Note:

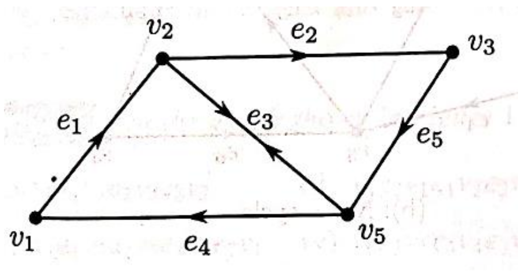
- In a walk, a vertex or an edge (or both) can appear more than once.
- The number of edges present in a walk is called its **length.**
- The vertex with which a walk begins is called the initial vertex (or the origin) of the walk
- and the vertex with which a walk ends is called the final vertex (or the terminus) of the walk.
- The initial vertex and the final vertex of a walk are together called its terminal vertices.
- The terminal vertices of a walk need not be distinct. Nonterminal vertices of a walk are called its internal vertices.
- A walk having u as the initial vertex and ν as the final vertex is called a walk from u to ν, or briefly a u- ν walk.

5.2. Open and Closed Walk

A walk that begins and ends at the same vertex is called a **closed walk**.

In other words, a closed walk is a walk in which the terminal vertices are coincident.

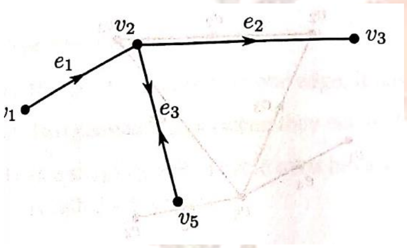
Example: $v_1 e_1 v_2 e_3 v_5 e_3 v_2 e_2 v_3 e_5 v_5 e_4 v_1$



A walk which is not closed is called an **open walk**.

In other words, an open walk is a walk that begins and ends at two different vertices.

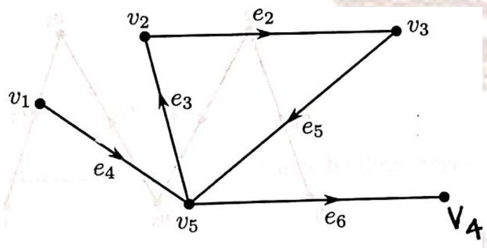
Example: v1e1v2e3v5v2e2v3



5.3. Trail and circuit

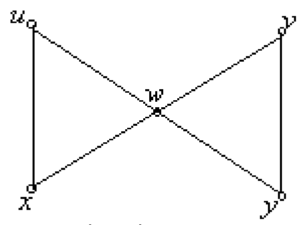
If in an open walk no edge appears more than once, then the walk is called a **trail**.

Example: v1v5v2v3v5v4



A closed walk in which no edge appears more than once is called a **circuit.**

Example:



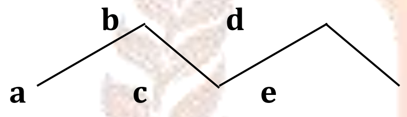
circuit

u, x, w, v, y, w, u

5.4. Path and cycle

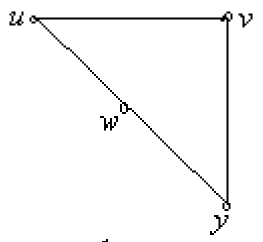
A trail in which no vertex appears more than once is called a **path**.

Example: abcde



A circuit in which the terminal vertex does not appear as an internal vertex and no internal vertex is repeated is called a **cycle**.

Example:



cycle uwyvu

Note:

1. A walk can be open or closed. In a walk (closed or open), a vertex and/or an edge can appear more than once.
2. A trail is an open walk in which a vertex can appear more than once but an edge cannot appear more than once.
3. A circuit is closed walk in which a vertex can appear more than once but an edge cannot appear more than once.

4. A path is an open walk in which neither a vertex nor an edge can appear more than once. Every path is a trail, but a trail need not be a path.
5. A cycle is closed walk in which neither a vertex nor an edge can appear more than once. Every cycle is circuit; but, a circuit need not be a cycle.

6. CONNECTIVITY

6.1. Connected and disconnected Graph

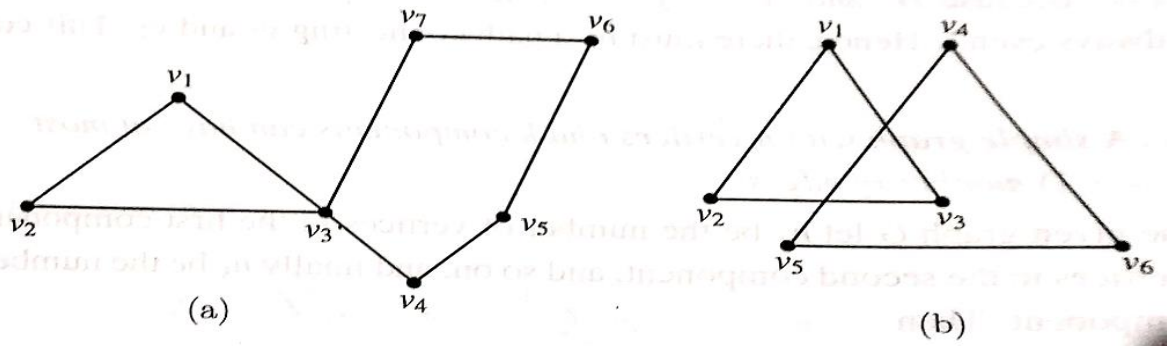
Consider a Graph G of order greater than or equal to two. Two vertices in G are said to be **connected** if there is at least one **path** from one vertex to the other.

We say that a graph G is a **connected graph** if every pair if every pair of distinct vertices in G are connected. Otherwise, G is called a **disconnected graph**.

In other words, a graph G is said to be (i) connected if there is a at least one path between every two distinct vertices in G, and (ii) disconnected if G has at least one pair of distinct vertices between which there is no path.

A graph G is connected if we can reach any vertex of G from any other vertex of G by travelling along the edges, and disconnected otherwise.

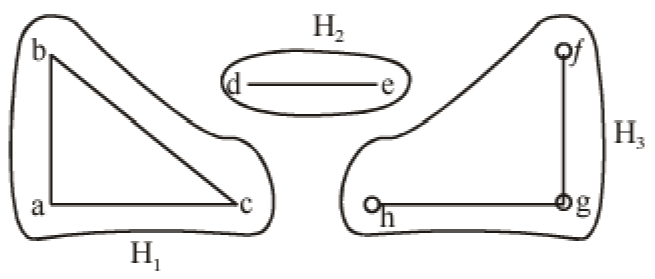
Example: Graph shown in Fig (a) is connected whereas the graph shown in fig 9.1 is disconnected.



6.2. Connected components

A graph that is not connects is the union of two or more connected subgraphs, each pair of which has no vertex in common. These disjoints connected subgraphs are called the connected components of the graph.

Example: Here H1 , H2 , H3 are three subgraphs which form the connected components of H. (Which is union of H1 , H2 , H3).



6.3. Strongly connected

A directed graph is called 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.

A directed graph can be strongly connected if there is a sequence of directed edges from any vertex in the graph to any other vertex.

A directed graph can fail to be strongly connected but still be in one piece.

6.4. Weakly connected

A directed graph is weakly connected if there is a path between every two vertices in the under lying undirected graph.

Hence a directed graph is weakly connected iff there is always a path between two vertices when the direction of the edges are disregarded.

Thus any strongly connected directed graph is also weakly connected.

NOTE:

1. It is obvious that in a graph G all walks and, therefore, all trails, all circuits, all paths and all cycles (when they exist) are connected subgraphs of G.

2. It is evident that every (nontrivial) graph G consists of one or more connected graphs. Each connected graph is a subgraph of G and is called a component of G.
3. Obviously, a connected graph has only one component and a disconnected graph has two or more components. The number of components of a graph G is denoted by k(G).

6.5. Euler circuits and Euler trails

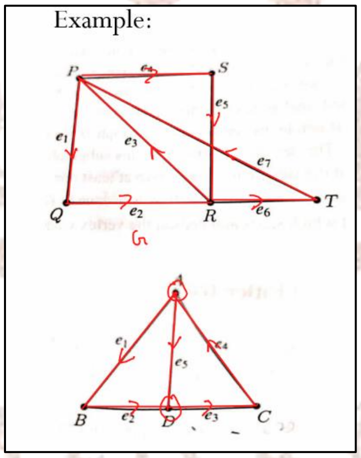
Consider a connected graph G. if there is circuit in G that contains all the edges of G, then that circuit is called an **Euler circuit** (or Euclerian line, or Euler tour) in G.

If there is a trail in G that contains all the edges of G, then that trail is called an **Euler trail in G.**

A connected graph that contains an Euler circuit is called an **Euler Graph (or Euclearian graph).**

A connected graph that contains an Euler trail is called an **semi-Euler Graph (or Euclearian graph).**

Example:



Closed walk $Pe_1Qe_2Re_3Pe_4Se_5Te_7P$. It is a Euler circuit. Therefore the graph is Euler graph

Open walk $Ae_1Be_2De_3Ce_4Ae_5D$ It is an Euler trail, it is an semi Euler graph.

6.6. Hamilton cycles and Hamilton paths

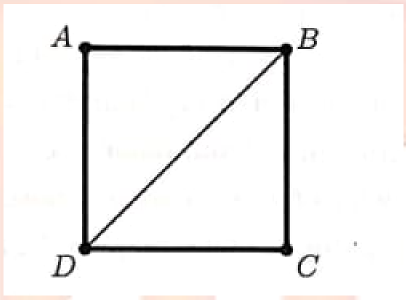
Let G be a connected graph. If there is a cycle in G that contains all the vertices of G, then that cycle is called a **Hamilton cycle** in G.

A Hamilton cycle (when it exists) in a graph of n vertices consists of exactly n edges. Because, a cycle with n vertices has n edges.

By definition, a Hamilton cycle (when it exists) in a graph G must include all vertices is G. This does not mean that it should include all edges of G.

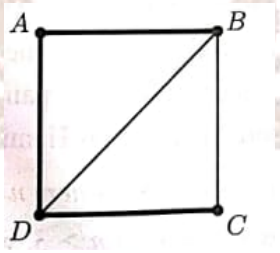
A graph that contains a Hamilton cycle is called **Hamilton graph** (or Hamiltonian graph).

For example, in the graph shown in figure, the cycle shows in thick lines is a Hamilton cycle. (Observe that this cycle does not include the edge BD). The graph is therefore a Hamilton graph.



A path (if any) in a connected graph which includes every vertex (but not necessarily every edge) of the graph is called a **Hamilton Path** (Hamiltonian path) in the graph.

For example, in the graph shown in figure , the path shown in thick lines is a Hamilton path.



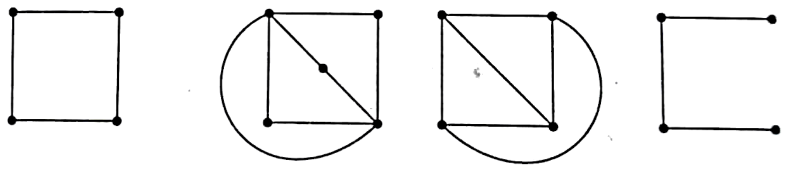
Example:

Write the graph for the following

a) A graph which has both an Euler circuit and a Hamilton cycle
b) A graph which has an Euler circuit but no Hamilton cycle.
c) A graph which has a Hamilton cycle but no Euler circuit.
d) A graph which has neither a Hamilton cycle nor an Euler circuit.

Solution

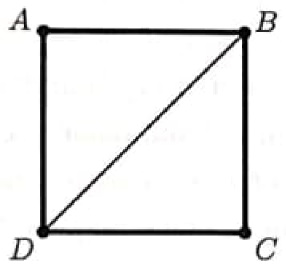
The graphs (a) - (d) shown below are the required graphs in the desired order.



Planar graphs

A graph which can be represented by at least one plane drawing (drawing done on a plane surface) in which the edges meet only at the vertices is called a planar graph.

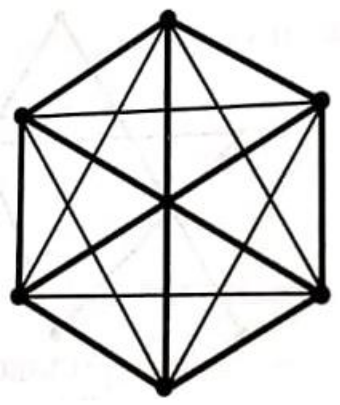
Example:



On the other hand, a graph which cannot be represented by a plane drawing in which the edges meet only at the vertices is called a non-planar graph.

In other words, a non-planar graph whose every possible plane drawing contains at least two edges which intersect each other at points other than vertices.

Example:



7. ALGORITHMS FOR MINIMAL SPANNING TREE

There are several methods of constructing minimal spanning trees. The first of these is **Kruskal** and second is **Prim** algorithm.

7.1. Kruskal's algorithm

The working rule for the Kruskal's method (usually called **Kruskal's algorithm**) may be stated as follows

Step1: Given a connected, weighted graph G with n vertices, list the edges of G in the order of nondecreasing weights.

Step2: Starting with a smallest weighted edge, proceed sequentially by selecting one edge at a time such that no cycle is formed.

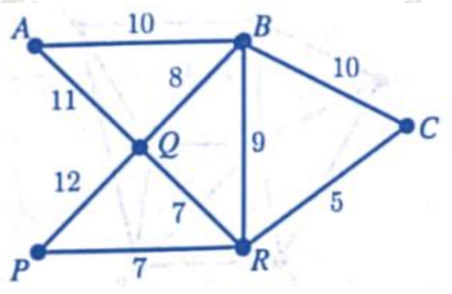
Step3: Stop the process of step2 when (n-1) edges are selected. These (n-1) edges constitute a minimal spanning tree of G.

Remarks

- If two or more edges have the same weight, there will be more than one listing of edges in non-decreasing order of weights. Different listing may yield different minimal spanning trees. As such, **Kruskal algorithm does not determine a unique minimal spanning tree.**
- The process in step2 is called **Greedy Process.**

Example:

Using Kruskal's Algorithm, find a minimal spanning tree for the given weighted graph



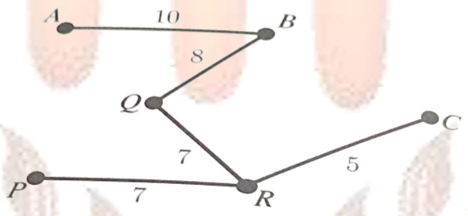
Solution

We observe that the given graph has 6 vertices; hence a spanning tree there of will have 5 edges (branches).

Let us put the graph in a non decreasing order of their weights and successively select 5 edges such a way that no cycle is created. This scheme is summarized in the following table

Edge	**CR**	**PR**	**QR**	**BQ**	**BR**	**AB**	**BC**	**AQ**	**PQ**
Weight	5	7	7	8	9	10	10	11	12
Select?	YES	YES	YES	YES	NO	YES			

Thus, a minimal spanning tree of the given graph contains the five edges CR, PR, QR, BQ, AB. This tree is shown in the figure. The weight of the tree is 37 units.



7.2. Prim's Algorithm

The working rule for the Prim's Method(Usually called Prim's algorithm) may be stated as follows

Step1: Given a connected, weighted graph G with n vertices, Assign **n** names (say **v1,v2,...,vn or A, B, C, and so on**) to these vertices, and prepare a **n x n** table in which the weights of all edges are shown. The entries in the table will be symmetric with respect to the diagonal and no entries appear on the diagonal, indicate the weights of the **non existing edges as ∞.**

Step2: Start from vertex **v1** (or A, as the case may be) and connect it to its nearest neighbour (i.e., to the vertex has the smallest entry) in the v1 row, say **vk.** Now, consider the edge **{v1, vk}** and connect it to its closest neighbour (i.e., to a vertex, other than v1 and vk, that has the smallest entry among all entries in v1 and vk rows). Let this be **vm**.

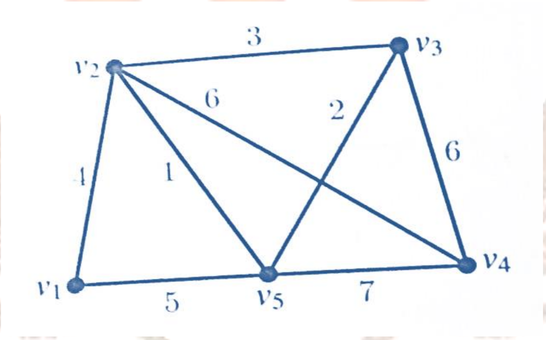
Step3: Start from the vertex **vm** and repeat the process of step2. stop the process when all the **n** vertices have been connected by **n-1** edges. These n-1 edges constitute a minimal spanning tree.

Remarks

- In the process of connected an edge to its nearest neighbour as explained above, care has to be taken that cycle are not created by the connections.
- Like in the Kruskal's method, a minimal spanning tree determined by the **prims method is not unique**.

Example:

Using Prim algorithm find a minimal spanning tree for the given weighted graph.



Solution

We observe that the graph has 5 vertices. Therefore the minimal spanning tree there of will have 4 edges. Let us tabulate the weights of the edges between every pair of vertices as shown below

	v1	v2	v3	v4	v5
v1	-	4	∞	∞	5
v2	4	-	3	6	1

v3	∞	**3**	**-**	**6**	**2**
v4	∞	**6**	**6**	**-**	**7**
v5	**5**	**1**	**2**	**7**	**-**

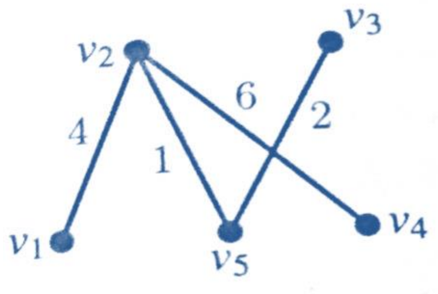
Now let us start with the first row (v1 row) and pick the smallest entry therein. This is 4 which corresponds to the edge {v1, v2}. By examining all the entries in v1- and v2-rows, we find that the vertex other than v1 and v2 which corresponds to smallest entry is v5 (smallest entry being 1). Thus v5 is closest to the edge {v1, v2}. Let us connect v5 to the {v1, v2} at v2 (because {v5, v2} has smaller weight than {v5, v1}).

Let us now examine the v5-row and note that small entry is 1 which corresponds to the edge {v5, v2}. By examining all entries in v2- and v5 rows, we find that the vertex other than v2 and v5 which corresponds to the smallest entry v3. thus v3 is closest to the edge {v2, v5}.

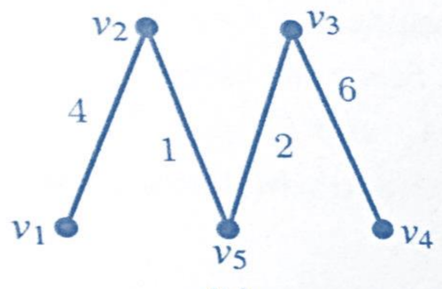
Let us connect v3 to the edge {v2, v5} at v5.

Thus the edges {v1, v2}, {v2, v5}, {v5, v3} belongs to a minimal spanning tree. The vertices left over at this stage is v4 which is joined to v2, v3, and v5 in the given graph. Among the edges that contain v4, the edges {v2, v4} and {v3, v4} have equal minimal weights. Therefore we can include either of those edges in the minimal spanning tree.

Accordingly, the degree {v1, v2}, {v2, v5}, {v5, v3} together with the edge {v2, v4} or the edge {v3, v4} constitute a minimal spanning tree. Thus, for the given graph, there are two minimal spanning trees as shown in the figure (a), (b). The weight of each of these trees is 13 units.



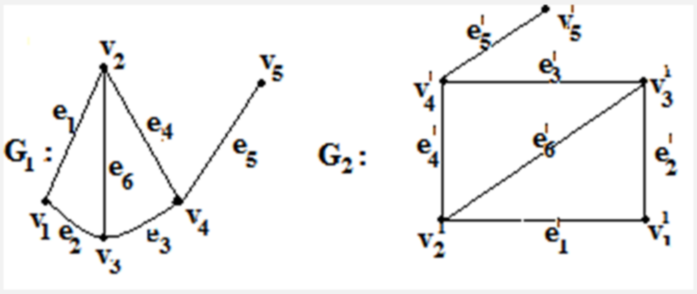
(a)



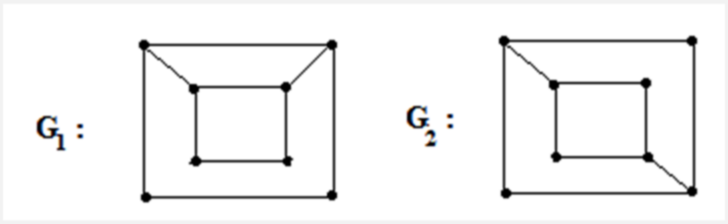
(b)

8. SELF-ASSESSMENT QUESTIONS

1. Verify that the two graphs shown below are isomorphic



2. Determine whether following graphs are isomorphic or not.



3. In a directed graph which of the following are paths

(i) a, b, e, d;

(ii) a, e, c, d, b;

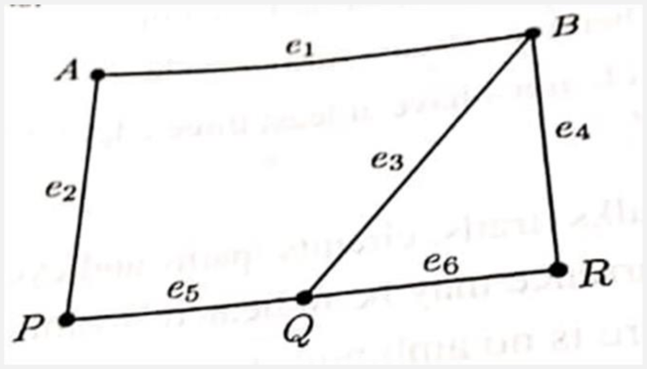
(iii) b, a, c, b, a, a, b;

(iv) d, c;

(v) e, b, a;

(vi) e, b, a, b, a, b, e ?. find their lengths. which of them are circuits.

4. Consider the graph shown below. Find all paths from vertex A to vertex R. also, indicate their lengths.



9. SUMMARY

In this Chapter we introduced some fundamentals and Introduction to the Isomorphism, later on we discussed about Subgraphs,

Many problems can be modelled with paths formed by traveling along the edges of graphs. For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model using the concepts of Walks, Paths, Circuits, Connectedness, Components.

10. ANSWERS TO SELF ASSESSMENT QUESTIONS

1. The correspondence between the graphs is as follows:

   The vertices (v_1, v_2, v_3, v_4, v_5) in G_1 correspond to $v_1^1, v_2^1, v_3^1, v_4^1, v_5^1$, respectively in G_2. The edges ($e_1, e_2, e_3, e_4, e_5, e_6$) in G_1 correspond to $e_1^1, e_2^1, e_3^1, e_4^1, e_5^1$, respectively in G_2. Here the incidence property is preserved. Therefore the graphs G_1 and G_2 are isomorphic to each other.

2. Here both the graphs G_1 and G_2 contain 8 vertices and 10 edges. The numbers of vertices of degree 2 in both the graphs are four. Also the number of vertices degrees 3 in both the graphs is four.

   For adjacency, consider the vertex of degree 3 in G_1. It is adjacent to two vertices of degree 3 and one vertex of degree 2. But in G_2 there does not exist any vertex of degree 3, which is adjacent to two vertices of degree 3 and one vertex of degree2. i.e., adjacency is not preserved. Hence, given graphs are not isomorphic.

3. (i) Here each of (a, b), (b, e) and (e, d) is an edge so a, b, e, d is a path length three.

   (ii) a, e, c, d, b is not a path because (c, d) is not an edge.

   (iii) b, a, c, b, a, a, b is path of length six because (b, a), (a, c), (e, b), (b, a), (a, a) and (a, b) are all edges.

   (iv) (d, c) is an edge and so d, c is a path of length one .

   (v) e, b, a is also a path and of length two because (c, b) and (b, a) are edges.

   (vi) The two paths b, a, c, b, a, a, b and e, b, a, b, a, b, e are circuits since they begin and end at the same vertex. However, paths a, b, e, d; c, b, a and d, e are not circuits.

4. Ae2Pe5Qe3Be4R is the path of length 4

   Ae1Be3Qe6R and Ae2Pe5Qe6R are the Paths of length 3

   Ae1Be4R is Path of length 2

11. REFERENCES

- Ralph P. Grimaldi (2019) Discrete and Combinatorial Mathematics: An Applied Introduction (5ed.) Pearson Publication.
- Kenneth H. Rosen (2012) Discrete Mathematics and Its Applications (7 th Edition) McGraw-Hill Publication.
- Dr. DSC (2016) Discrete Mathematical Structures (5th Edition), PRISM publication.
- https://www.geeksforgeeks.org/elements-of-poset/