Second Year of Computer Engineering (2019 Course)

(210241): Discrete Mathematics

Teaching Scheme	Credit Scheme	Examination Scheme and Marks
Lecture: 03 Hours/Week	03	Mid_Semester (TH): 30 Marks End_Semester (TH): 70 Marks

Marks weightage per unit for examination						
Unit Number	**I**	**II**	**III**	**IV**	**V**	**VI**
Mid_Semester	15	15	-	-	-	-
End_Semester	-	-	18	17	18	17

Prerequisites: Basic Mathematics

Course Objectives

To introduce several Discrete Mathematical Structures found to be serving as tools even today in the development of theoretical computer science.

1. To introduce students to understand, explain, and apply the foundational mathematical concepts at the core of computer science.

2. To understand use of set, function and relation models to understand practical examples, and interpret the associated operations and terminologies in context.

3. To acquire knowledge of logic and proof techniques to expand mathematical maturity.

4. To learn the fundamental counting principle, permutations, and combinations.

5. To study how to model problem using graph and tree.

6. To learn how abstract algebra is used in coding theory.

Course Outcomes

On completion of the course, learner will be able to –

CO1: Formulate problems precisely, solve the problems, apply formal proof techniques, and explain the reasoning clearly.

CO2: Apply appropriate mathematical concepts and skills to solve problems in both familiar and unfamiliar situations including those in real-life contexts.

CO3: Design and analyze real world engineering problems by applying set theory, propositional logic and to construct proofs using mathematical induction.

CO4: Specify, manipulate and apply equivalence relations; construct and use functions and apply these concepts to solve new problems.

CO5: Calculate numbers of possible outcomes using permutations and combinations; to model and analyze computational processes using combinatorics.

CO6: Model and solve computing problem using tree and graph and solve problems using appropriate algorithms.

CO7: Analyze the properties of binary operations, apply abstract algebra in coding theory and evaluate the algebraic structures.

Learning Resources

❖ **Text Books:**

1. C. L. Liu, "Elements of Discrete Mathematics"‖, TMH, ISBN 10:0-07-066913-9.2.
2. N. Biggs, "Discrete Mathematics", 3rd Ed, Oxford University Press, ISBN 0 –19-850717–8.

❖ **Reference Books:**

1. Kenneth H. Rosen, "Discrete Mathematics and its Applications"‖, Tata McGraw-Hill, ISBN 978-0-07-288008-3
2. Bernard Kolman, Robert C. Busby and Sharon Ross, "Discrete Mathematical Structures"‖, Prentice-Hall of India /Pearson, ISBN: 0132078457, 9780132078450.
3. Narsingh Deo, "Graph with application to Engineering and Computer Science", Prentice Hall of India, 1990, 0 –87692 –145 –4.
4. Eric Gossett, "Discrete Mathematical Structures with Proofs", Wiley India Ltd, ISBN:978-81-265-2758-8.
5. Sriram P.and Steven S., "Computational Discrete Mathematics", Cambridge University Press, ISBN 13: 978-0-521-73311-3

Unit IV

Graph Theory

Duration: (07 Hours)

Mapping of Course Outcomes: CO1,CO2,CO6

Unit-IV: Contents

- Graph Terminology and Special Types of Graphs,
- Representing Graphs and Graph Isomorphism, Connectivity,
- Euler and Hamilton Paths,
- The handshaking lemma,
- Single source shortest path-Dijkstra's Algorithm,
- Planar Graphs, Graph Colouring.

- **Exemplar/ Case Studies:** Three utility problem, Web Graph, Google map

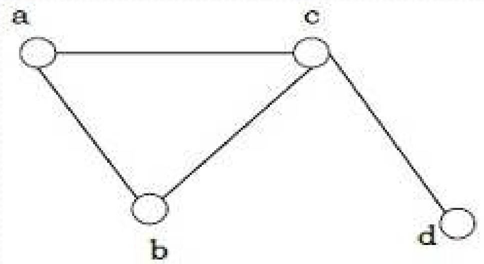
Introduction

- Graphs are discrete structures consisting of vertices and edges that connect these vertices.
- There are different kinds of graphs, depending on whether edges have directions, whether multiple edges can connect the same pair of vertices, and whether loops are allowed.

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

Introduction

- **Definition:** A Graph G = (V, E) consists of V, a nonempty set of vertices (or nodes) and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints.
- **Example 1:** Let us consider, a Graph is G = (V, E) where V = {a, b, c, d} and E ={{a,b},{a,c},{b,c},{c,d}}

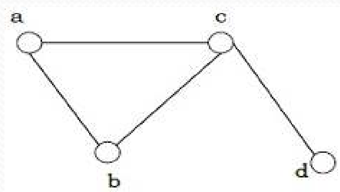


Introduction

- **Degree of a Vertex** - The degree of a vertex V of a graph G (denoted by deg (V)) is the number of edges incident with the vertex V.
- **Even and Odd Vertex** - If the degree of a vertex is even, the vertex is called an even vertex and if the degree of a vertex is odd, the vertex is called an odd vertex.
- **The Handshaking Lemma-** In a graph, the sum of all the degrees of vertices is equal to twice the number of edges.

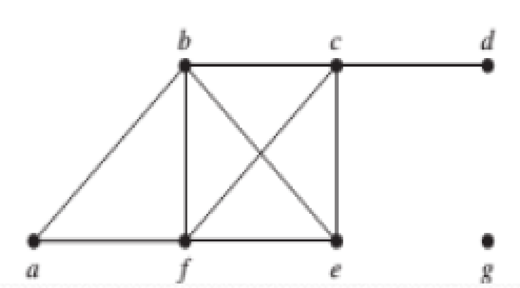
$$2m = \sum_{v \in V} \deg(v).$$

Vertex	Degree	Even / Odd
a	2	even
b	2	even
c	3	odd
d	1	odd



Graph Terminology

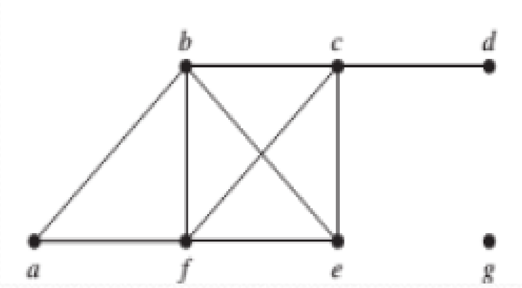
- **Adjacent Vertex :** Two vertices u and v in an undirected graph G are called adjacent (or neighbors) in G if u and v are endpoints of an edge e of G. Such an edge e is called incident with the vertices u and v and e is said to connect u and v.
- **Neighborhood :** The set of all neighbors of a vertex 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.



Graph Terminology

- **Isolated vertex :** A vertex of degree zero is called **isolated**. It follows that an isolated vertex is not adjacent to any vertex. In fig vertex g in graph G is isolated vertex.

- **Pendent vertex :** A vertex is **pendant** if and only if it has degree one. Consequently, a pendant vertex is adjacent to exactly one other vertex. In fig vertex d in graph G is pendant vertex.



Graph Terminology

- **Degree of a Vertex :** 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).

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

Graph Terminology

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

- **Out-degree :** In a graph with directed edges the **out-degree of v**, denoted by $deg^+(v)$, is the number of edges with v as their initial vertex.

- **Imp Note:** that a loop at a vertex contributes 1 to both the in-degree and the out-degree of this vertex.).

Theorem

❖ **Theorem 1: The Handshaking Theorem::**

➢ Let G = (V , E) be an undirected graph with m edges. Then

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

➢ (Note that this applies even if multiple edges and loops are present.)

❖ **Theorem 2: An undirected graph has an even number of vertices of odd degree.**

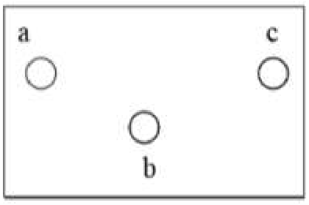
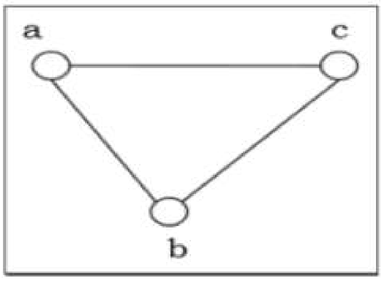
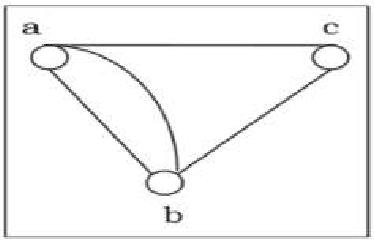
➢ Proof: Let V1 and V2 be the set of vertices of even degree and the set of vertices of odd degree, respectively, in an undirected graph G = (V,E) with m edges. Then

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

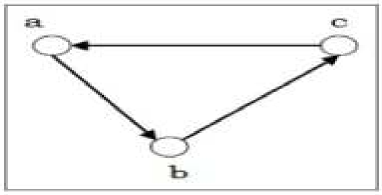
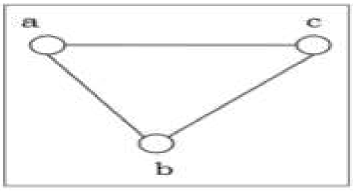
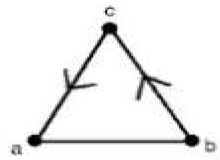
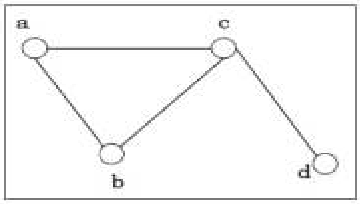
❖ **Theorem 3: Let G = (V, E) be a graph with directed edges. Then**

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

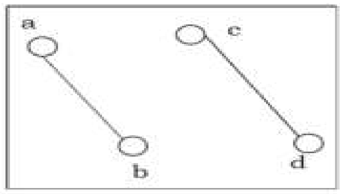
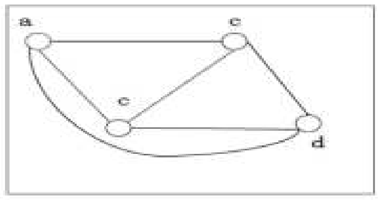
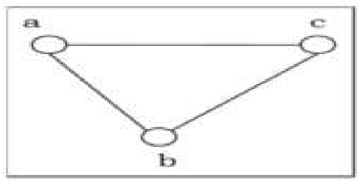
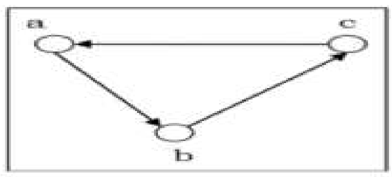
Type of Graph

Null Graph	A null graph has no edges. The null graph of n vertices is denoted by Nn	
Simple Graph	A graph is called simple graph/strict graph if the graph is undirected and does not contain any loops or multiple edges. In a simple graph each edge connects two different vertices and no two edges connect the same pair of vertices.	
Multi-Graph	If in a graph multiple edges between the same set of vertices are allowed, it is called Multigraph. When m different edges connect the vertices u and v, we say that {u,v} is an edge of multiplicity m.	

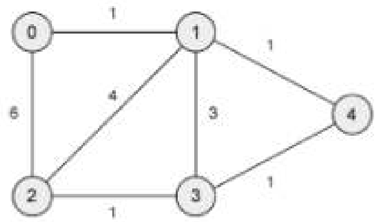
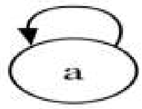
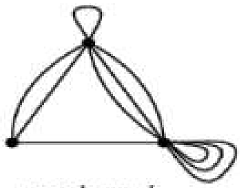
Type of Graph

Directed Graph	A graph G = (V, E) is called a directed graph if the edge set is made of ordered vertex pair.	
Undirected Graph	A graph G = (V, E) is called a undirected if the edge set is made of unordered vertex pair.	
Mixed Graph	A graph with both directed and undirected edges is called a mixed graph.	
Connected Graph	A graph is connected if any two vertices of the graph are connected by a path	

Type of Graph

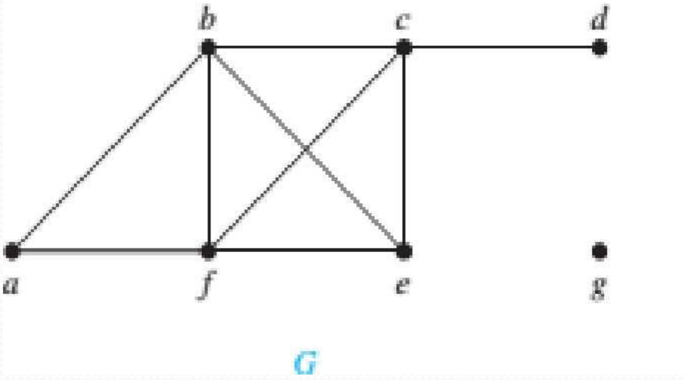
Disconnected Graph	A graph is disconnected if at least two vertices of the graph are not connected by a path. If a graph G is unconnected, then every maximal connected subgraph of G is called a connected component of the graph G.	
Regular Graph	A graph is regular if all the vertices of the graph have the same degree. In a regular graph G of degree r, the degree of each vertex of G is r.	
Complete Graph	A graph is called complete graph if every two vertices pair are joined by exactly one edge. The complete graph with n vertices is denoted by Kn	
Cycle Graph	If a graph consists of a single cycle, it is called cycle graph. The cycle graph with n vertices is denoted by Cn	

Type of Graph

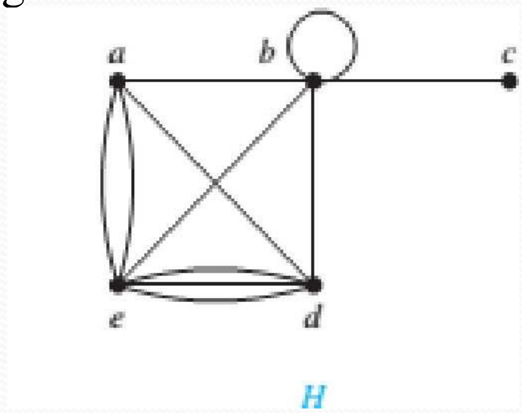
Infinite graph	A graph with an infinite vertex set or an infinite number of edges is called an infinite graph	
Finite graph	A graph with a finite vertex set and a finite edge set is called a finite graph	
Weighted Graph	A graph having a weight, or number, associated with each edge.	
Loop	An edge that connects a vertex to itself is called a loop.	
Pseudograph	A pseudograph may include loops, as well as multiple edges connecting the same pair of vertices.	

Example

❖ **Example 1:** What are the degrees and what are the neighborhoods of the vertices in the graphs G and H displayed in figure below.



G

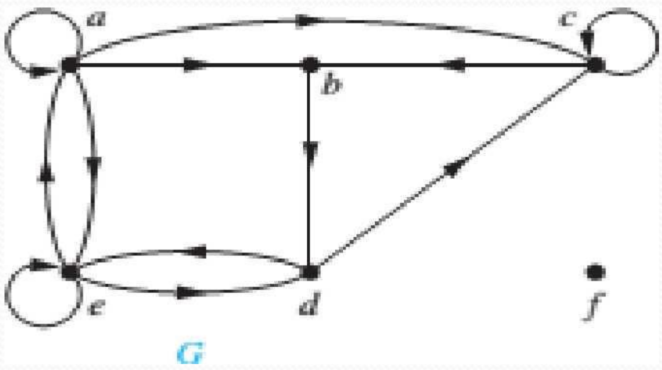


H

In Graph G, deg(a) = 2, deg(b) = deg(c) = deg(f) = 4, deg(d) = 1, deg(e) = 3, and deg(g) = 0.	In Graph H, deg(a) = 4, deg(b) = deg(e) = 6, deg(c) = 1, and deg(d) = 5.
The neighborhoods of these vertices are N(a)={b, f}, N(b)={a, c, e, f}, N(c)={b, d, e, f}, N(d)={c}, N(e)={b, c, f}, N(f)={a, b, c, e}, and N(g)=∅.	The neighborhoods of these vertices are N(a)={b, d, e}, N(b)={a, b, c, d, e}, N(c)={b}, N(d)={a, b, e}, and N(e)={a, b, d}.

Example

❖ **Example 2:** Find the in-degree and out-degree of each vertex in the graph G with directed edges shown in figure below.



Solution:

The in-degrees in G are :::

$deg^-(a) = 2$, $deg^-(b) = 2$,

$deg^-(c) = 3$, $deg^-(d) = 2$,

$deg^-(e) = 3$, $deg^-(f) = 0$.

The out-degrees in G are :::

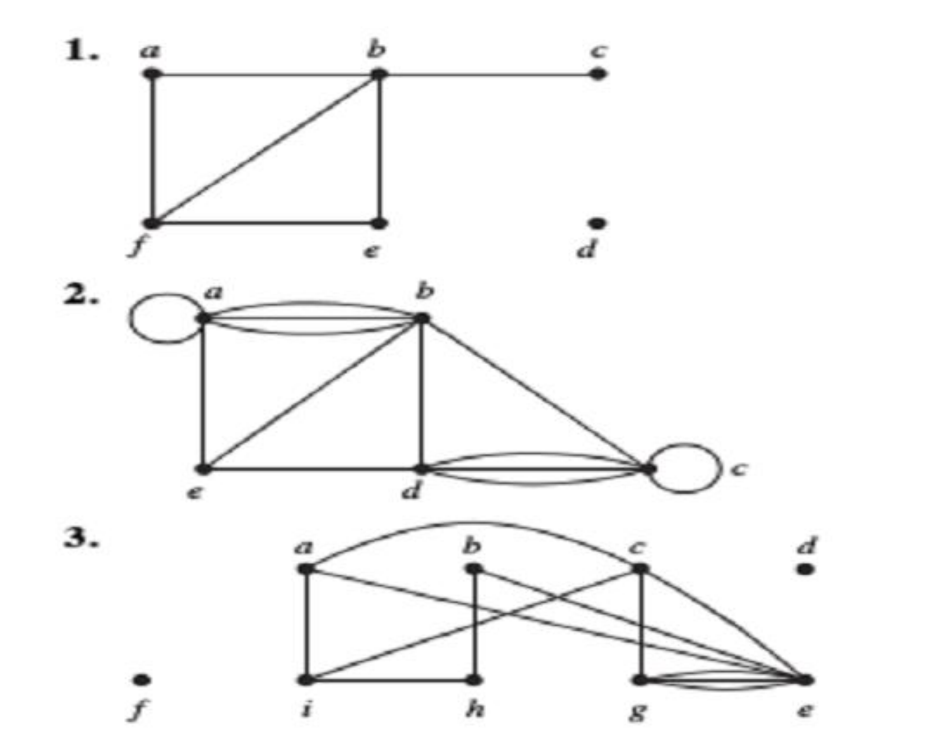
$deg^+(a) = 4$, $deg^+(b) = 1$,

$deg^+(c) = 2$, $deg^+(d) = 2$,

$deg^+(e) = 3$, $deg^+(f) = 0$.

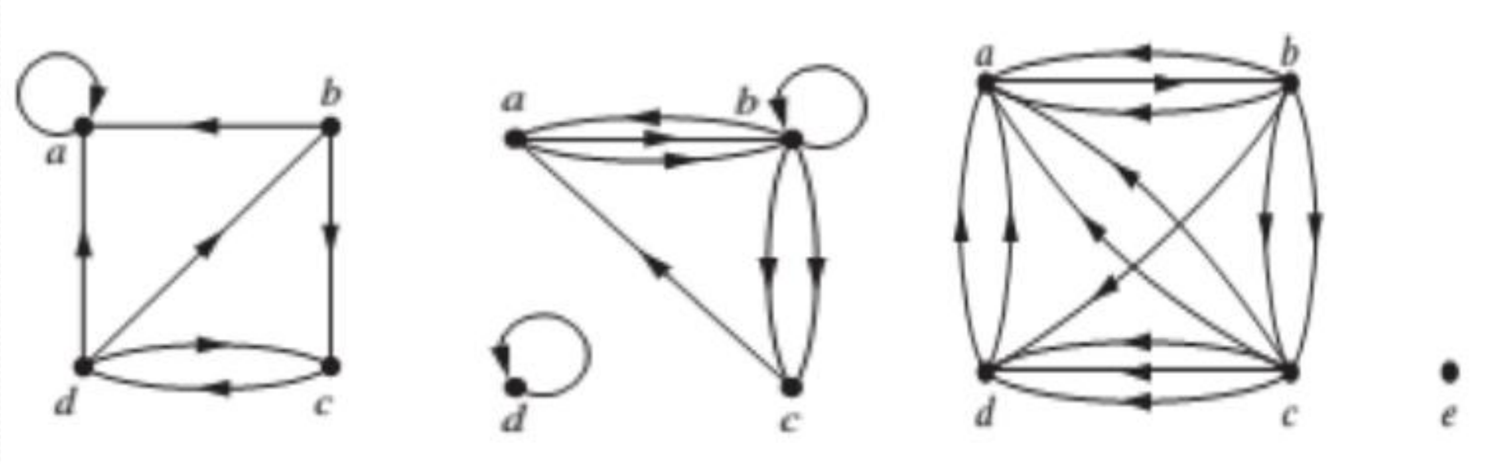
Example

❖ **Example 3:** Find the number of vertices, the number of edges, and the degree of each vertex in the given undirected graph. Identify all isolated and pendant vertices.



Example

❖ **Example 4:** Determine the number of vertices and edges and find the in-degree and out-degree of each vertex for the given directed multi-graph. Also determine sum of the in-degrees of the vertices and the sum of the out-degrees of the vertices directly.



Example

- **Example 5:** How many edges are there in a graph with 10 vertices each of degree six?

  Solution: Because the sum of the degrees of the vertices is **6 * 10 = 60**, it follows that **2m = 60** where m is the number of edges. Therefore, **m=30.**

- **Example 6:** How many vertices does a regular graph of degree four with 10 edges have?

  Solution: If a graph is regular of degree 4 and has n vertices, then by the handshaking theorem it has **4n/2 = 2n edges**. Since we are told that there are 10 edges, we just need to solve **2n = 10**. Thus the graph has 5 vertices. The complete graph K5 is one such graph.

Representation of Graphs

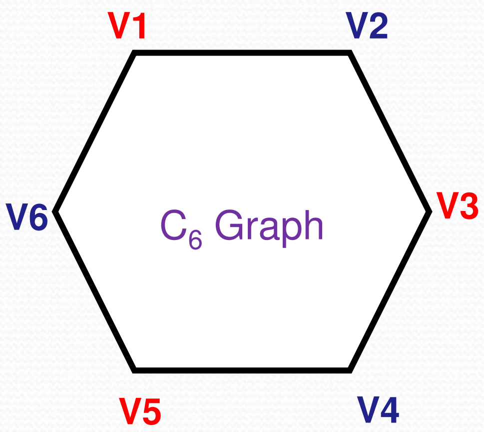
Representation of Graphs

Bipartite Graph

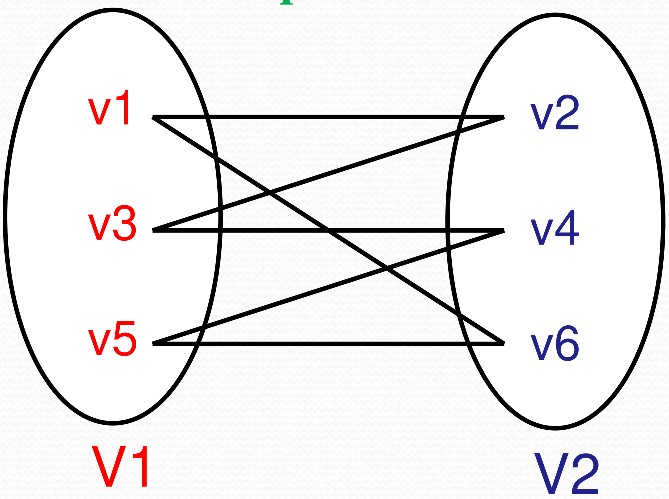
- A simple graph G is called **bipartite** if its vertex set V can be partitioned into two disjoint sets V1 and V2 such that every edge in the graph connects a vertex in V1 and a vertex in V2.
- (so that no edge in G connects either two vertices in V1 or two vertices in V2).
- When this condition holds, we call the pair (V1,V2) a bipartition of the vertex set V of G.

Bipartite Graph

- **C6 is bipartite**, as shown in below, because its vertex set can be partitioned into the two sets
- V1 = {v1, v3, v5} and V2 = {v2, v4, v6}, and every edge of C6 connects a vertex in V1 and a vertex in V2 .

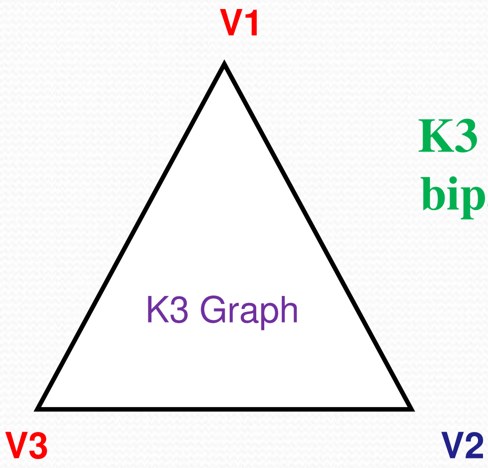


C_6 bipartite

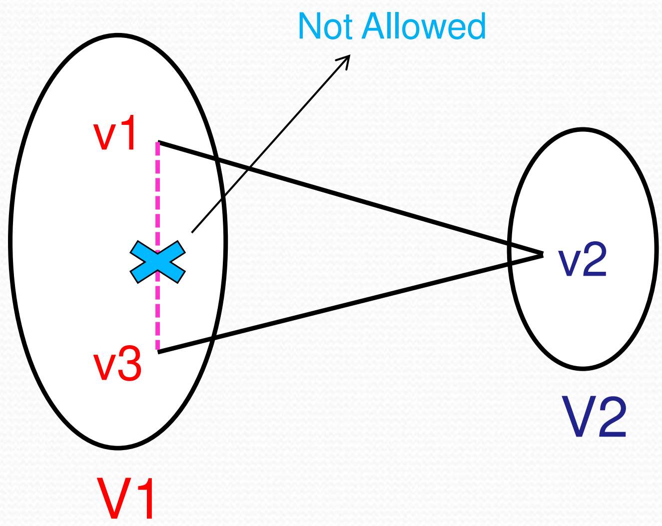


Bipartite Graph

- **K3 is not bipartite**. To verify this, note that if we divide the vertex set of K3 into two disjoint sets, one of the two sets must contain two vertices. If the graph were bipartite, these two vertices could not be connected by an edge, but in K3 each vertex is connected to every other vertex by an edge.

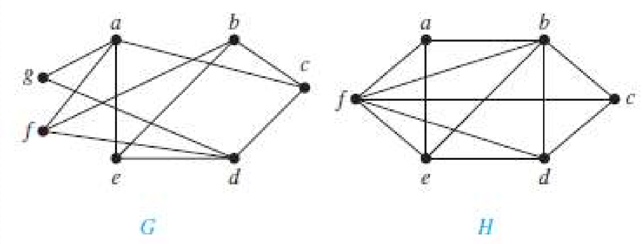


K3 is not bipartite



Bipartite Graph

❖ **Example 15:** Are the graphs G and H displayed in figure bipartite?



Solution: Graph **G is bipartite** because its vertex set is the union of two disjoint sets, {a, b, d} and {c, e, f, g}, and each edge connects a vertex in one of these subsets to a vertex in the other subset.

(Note that for G to be bipartite it is not necessary that every vertex in {a, b, d} be adjacent to every vertex in {c, e, f, g}. For instance, b and g are not adjacent.)

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

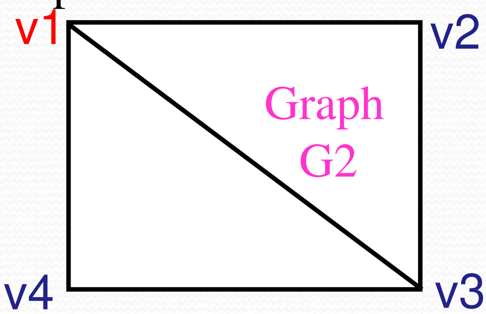
Bipartite Graph

- **THEOREM:** A simple graph is bipartite if and only if it is possible to assign one of two different colors to each vertex of the graph so that no two adjacent vertices are assigned the same color.
- **Step:**
  1. Assign RED color to the source vertex (putting into set V1).
  2. Color all the neighbors with BLUE color (putting into set V2).
  3. Color all neighbor's neighbor with RED color (putting into set V1).
  4. This way, assign color to all vertices such that it satisfies all the constraints of m ways of coloring the problem where m=2.
  5. While assigning colors, if we find a neighbor which is colored with same color as current vertex, then the graph cannot be colored with 2 vertices (or graph is not Bipartite).

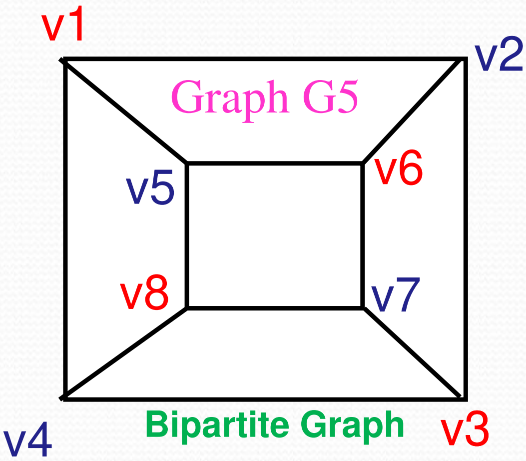
Example on Bipartite Graph

❖ **Example 16:** Determine whether the graph is bipartite:

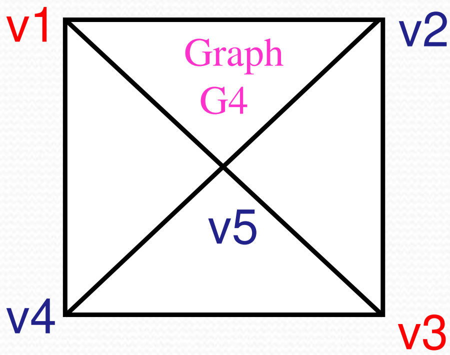
Bipartite Graph



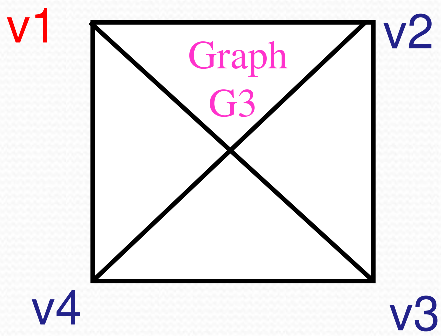
Not Bipartite Graph



Bipartite Graph



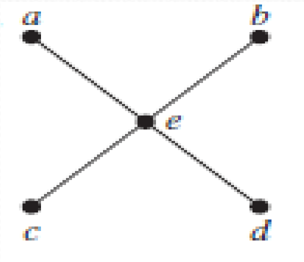
Not Bipartite Graph



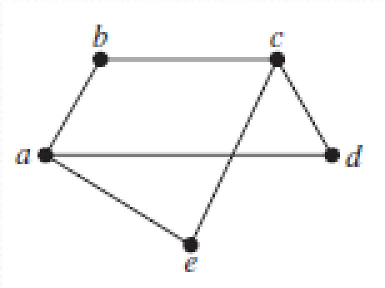
Not Bipartite Graph

Example on Bipartite Graph

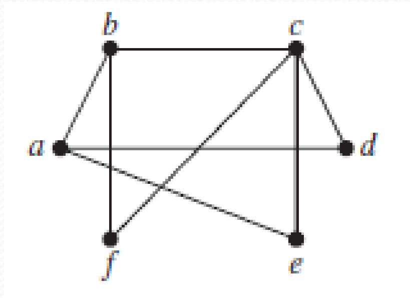
❖ **Example 17:** Determine whether the graph is bipartite:



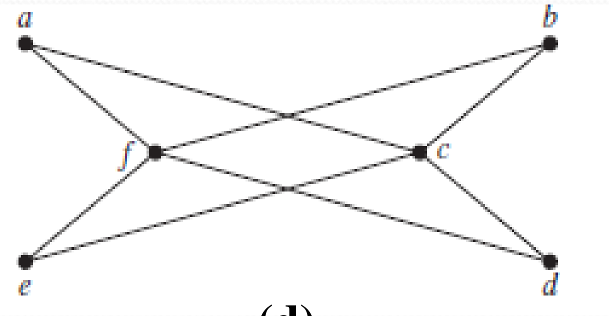
(a)



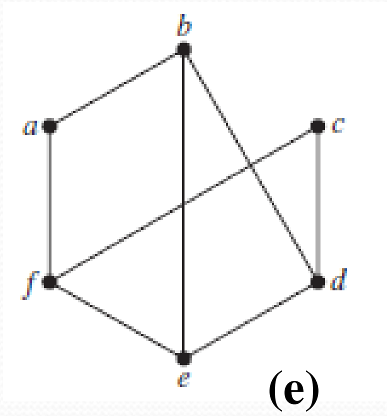
(b)



(c)



(d)

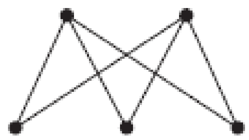


(e)

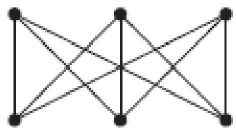
Solution: Bipartite Graph:: a, b, d

Complete Bipartite Graphs

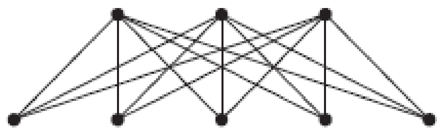
- A **Complete Bipartite Graph $K_{m,n}$** is a graph that has its vertex set partitioned into two subsets of m and n vertices, respectively with an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.
- i.e. each vertex of Vi is joined to every vertex of Vj by an unique edge.
- The complete bipartite graphs $K_{2,3}$, $K_{3,3}$, $K_{3,5}$, and $K_{2,6}$ are displayed below:



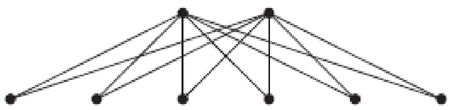
$K_{2,3}$



$K_{3,3}$



$K_{3,5}$



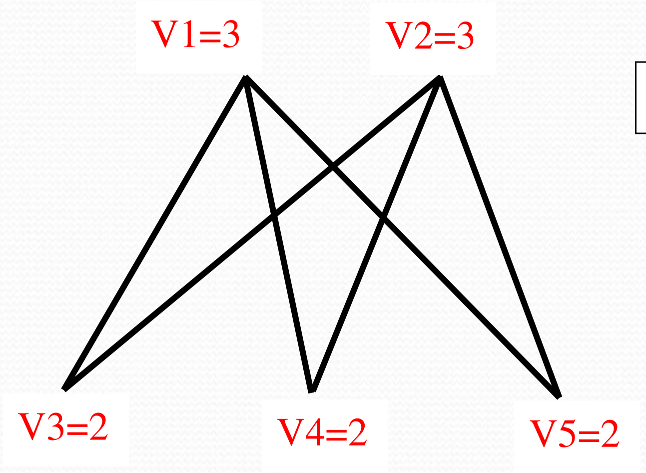
$K_{2,6}$

Complete Bipartite Graphs

V={V1,V2}

V1-Subset Deg = 3

V2-Subset Deg = 2



$K_{2,3}$

Example on Complete Bipartite Graphs

❖ **Example 18:** For which values of n are these graphs bipartite?

a) K_n b) C_n c) W_n d) Q_n

Solution:

- a) K_n is bipartite if and only if $n = 2$
- b) C_n is bipartite if and only if n is even.
- c) W_n is not bipartite for any n.
- d) Q_n is bipartite for all $n \geq 2$.

Isomorphism of Graphs

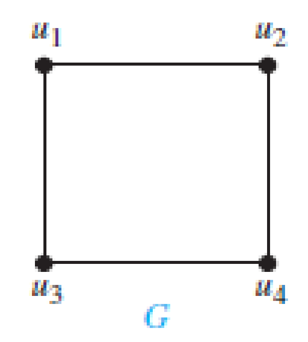
- We often need to know whether it is possible to draw two graphs in the same way. That is, do the graphs have the same structure when we ignore the identities of their vertices?
- The simple graphs **G1 = (V1,E1)** and **G2 = (V2,E2)** are isomorphic if there exists a one to- one and onto function f 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 in G2, 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, when two simple graphs are isomorphic, there is a one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship.

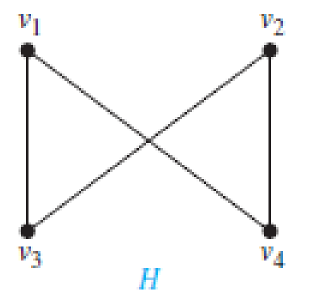
Example: Isomorphism of Graphs

❖ **Example 19:** Show that the graphs G = (V ,E) and H = (W, F), displayed in Figure below , are isomorphic.

Solution: The function f with f (u1) = v1, f (u2) = v4, f (u3) = v3, and f (u4) = v2 is a one to-one correspondence between V and W.

Adjacent vertices in G are u1 and u2, u1 and u3, u2 and u4, and u3 and u4, and each of the pairs f (u1) = v1 and f (u2) = v4, f (u1) = v1 and f (u3) = v3, f (u2) = v4 and f (u4) = v2, and f (u3) = v3 and f (u4) = v2 consists of two adjacent vertices in H.





Isomorphism of Graphs

- It is often difficult to determine whether two simple graphs are isomorphic.
- There are ***n*!** possible one-to-one correspondences between the vertex sets of two simple graphs with *n* vertices.
- Testing each such correspondence to see whether it **preserves adjacency** and non adjacency is impractical if *n* is at all large.
- Sometimes it is not hard to show that two graphs are not isomorphic. In particular, we can show that two graphs are not isomorphic if we can find a property only one of the two graphs has, but that is preserved by isomorphism.
- A property preserved by isomorphism of graphs is called a **graph invariant**.

Isomorphism of Graphs

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

- Isomorphic simple graphs also must have the **same number of edges**, because the one-to-one correspondence between vertices establishes a one-to-one correspondence between edges.

- The degrees of the vertices in isomorphic simple graphs **must be the same**.

- That is, a vertex v of degree d in G must correspond to a vertex $f(v)$ of degree d in H, because a vertex w in G is adjacent to v if and only if $f(v)$ and $f(w)$ are adjacent in H.

Isomorphism of Graphs

- Note: If two graphs are isomorphic, they must have:
  - Must have same number of vertices
  - Must have same number of edges
  - Must have equal number of vertices with same degree.
  - Must have equal number of loops
  - Must have equal number of pendent
  - G1 and G2 must have equal number of pendent edges.
  - If u and v are adjacent in G1 then the corresponding vertices in G2 are also adjacent.
- In general, it is easier to prove two graphs are not isomorphic by proving that one of the above properties fails.

Example: Isomorphism of Graphs

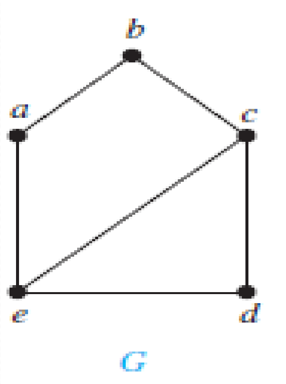
❖ **Example 20:** Show that the graphs G = (V, E) and H = (W, F), displayed in figure below, are isomorphic or not.

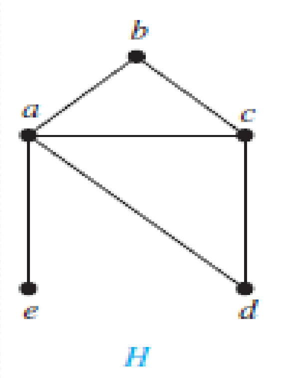
Solution:

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

➢ However, H has a vertex of degree one, namely, e, whereas G has no vertices of degree one.

➢ It follows that G and H are not isomorphic

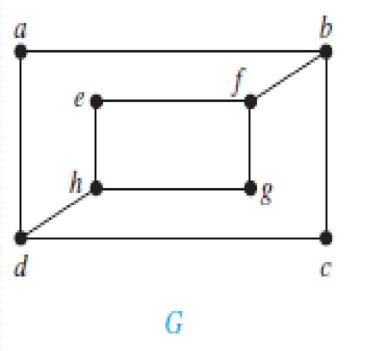




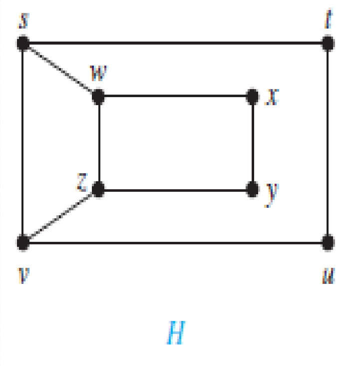
Example: Isomorphism of Graphs

❖ **Example 21:** Determine whether the following graphs are isomorphic.

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

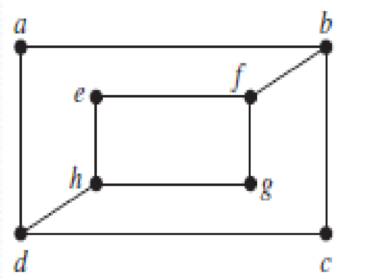


➢ However, G and H are not isomorphic. To see this, note that because deg(a) = 2 in G, a must correspond to either t , u, x, or y in H, because these are the vertices of degree two in H. However, each of these four vertices in H is adjacent to another vertex of degree two in H, which is not true for a in G.



Example: Isomorphism of Graphs

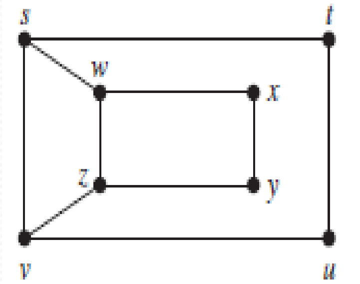
- **Example 21: Solution: Continued**
- However, G and H are not isomorphic. To see this, note that because deg(a) = 2 in G, a must correspond to either t , u, x, or y in H, because these are the vertices of degree two in H. However, each of these four vertices in H is adjacent to another vertex of degree two in H, which is not true for a in G.



G

- In G vertex a ➔ should map in H vertex t, u, x, y
- Adjacent to vertex a in G ➔ b=3 and d=3
- In Graph H

t ➔ s=3 and u=2	u ➔ t=2 and v=3
x ➔ w=3 and y=2	y ➔ x=2 and z=3



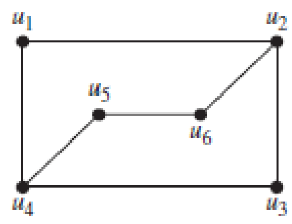
H

Example: Isomorphism of Graphs

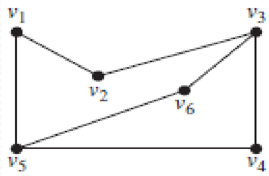
❖ **Example 22:** Determine whether the following graphs are isomorphic.

Solution:

- Both G and H have six vertices and seven edges.
- Both have four vertices of degree two and two vertices of degree three.
- It is also easy to see that the subgraphs of G and H consisting of all vertices of degree two and the edges connecting them are isomorphic.



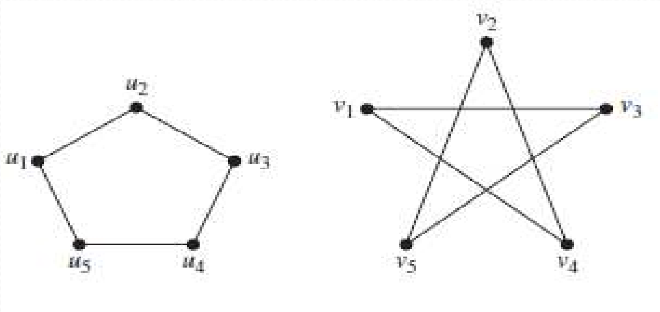
G



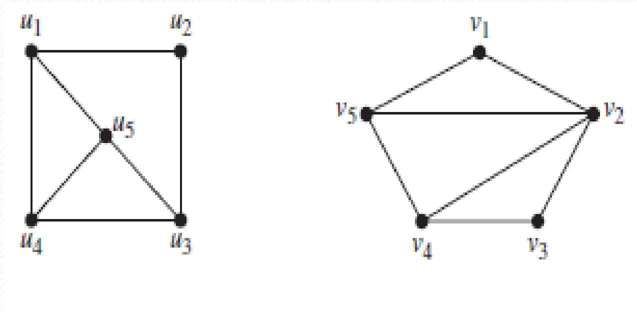
H

Example: Isomorphism of Graphs

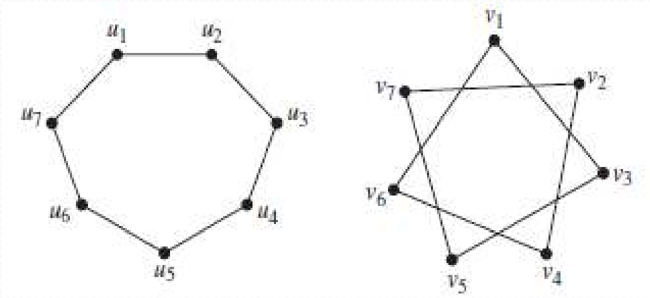
❖ **Example 23:** Determine whether the following graphs are isomorphic or not.



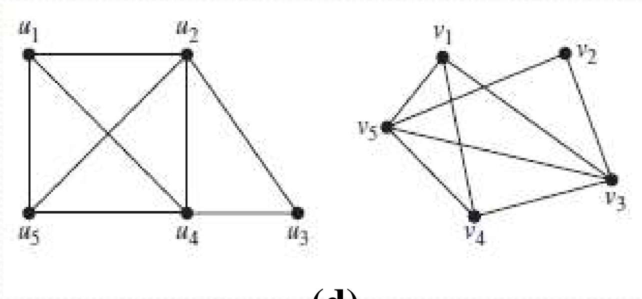
(a)



(b)



(c)



(d)

Example: Isomorphism of Graphs

❖ **Example 24:** Are the simple graphs with the following adjacency matrices isomorphic?

Solution:

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

a. Both graphs consist of 2 sides of a triangle; they are clearly isomorphic.

b. The graphs are not isomorphic, since the first has 4 edges and the second has 5 edges,

c. The graphs are not isomorphic, since the first has 4 edges and the second has 3 edges,

Connectivity-Paths

- 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.

- The **path is a circuit** if it begins and ends at the same vertex, that is, if u = v, and has length greater than zero.

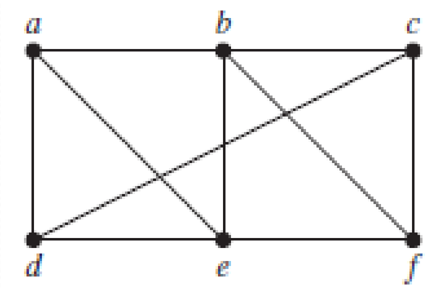
- A **path or circuit** is simple if it does not contain the same edge more than once.

Connectivity-Paths

- A **circuit** in a graph is also called as cycle in a graph.
- A **walk** is an alternating sequence of vertices and edges of a graph.
- A **path is a walk** that does not include any vertex twice, except that its first vertex might be the same as its last.
- A **trail is a walk** that does not pass over the same edge twice.
- A trail might visit the **same vertex twice**, but only if it comes and goes from a different edge each time.

Connectivity-Paths

❖ **Example 25 :** In the simple graph shown in figure below:



➢ In graph a, d, c, f , e is a simple path of length 4, because {a, d}, {d, c}, {c, f }, and {f, e} are all edges.

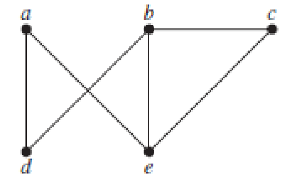
➢ However, d, e, c, a is not a path, because {e, c} is not an edge.

➢ Note that b, c, f , e, b is a circuit of length 4 because {b, c}, {c, f }, {f, e}, and {e, b} are edges, and this path begins and ends at b.

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

Connectivity-Paths

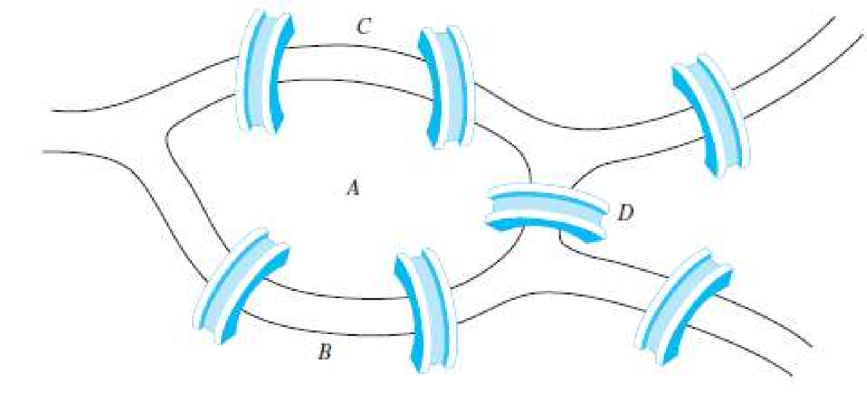
❖ **Example 26 :** Does each of these lists of vertices form a path in the following graph? Which paths are simple? Which are circuits? What are the lengths of those that are paths?

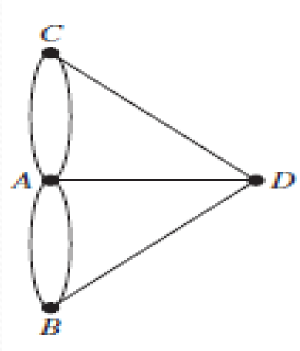


Solution:

- a) a, e, b, c, b **:** This is a path of length 4, but it is not simple, since edge {b, c} is used twice. It is not a circuit, since it ends at a different vertex from the one at which it began.
- b) a, e, a, d, b, c, a : This is not a path, since there is no edge from c to a.
- c) e, b, a, d, b, e : This is not a path, since there is no edge from b to a.
- d) c, b, d, a, e, c : This is a path of length 5 (it has 5 edges in it). It is simple, since no edge is repeated. It is a circuit since it ends at the same vertex at which it began.

Euler Paths & Euler Circuits





- 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.

Euler Paths & Euler Circuits

Euler Paths	Euler Circuits
❖ An **Euler path** is a path that uses every edge of a graph exactly once. ❖ An Euler path starts and ends at **different vertices**. ❖ If a graph G has an Euler path, then it must have exactly **two odd vertices**. OR ❖ If the number of odd vertices in G is anything other than 2, then G cannot have an **Euler path**.	❖ An **Euler circuit** is a circuit that uses every edge of a graph exactly once. ❖ An Euler circuit starts and ends at the **same vertex**. ❖ If a graph G has an Euler circuit, then all of its vertices must be **even vertices.** OR ❖ If the number of odd vertices in G is anything other than 0, then G cannot have an Euler circuit.

In **Euler paths and Euler circuits,** the game is to find paths or circuits that include **every edge** of the graph once (and only once).

Euler Paths & Euler Circuits

❖ In **Euler paths and Euler circuits,** the game is to find paths or circuits that include **every edge** of the graph once (and only once).

#Odd Vertices	Euler Path?	Euler Circuit?
0	No	Yes*
2	Yes*	No
4, 6, 8,……	No	No
1, 3, 5,……	No such graph exist	

(* Provided the graph is connected)

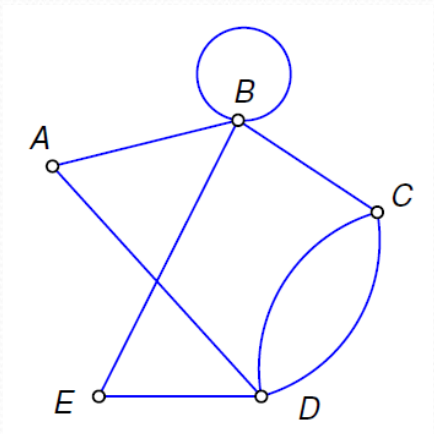
Euler Paths & Euler Circuits

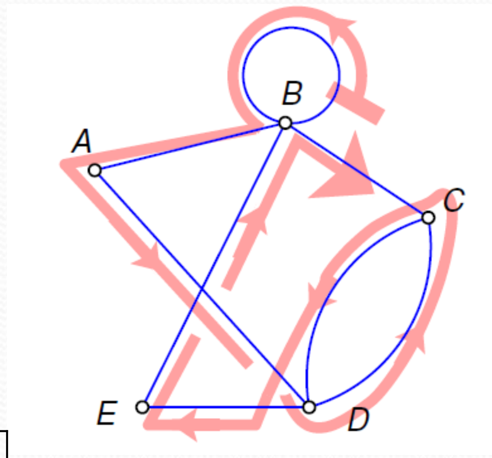
- **Bridges -** Removing a single edge from a connected graph can make it disconnected. Such an edge is called a **bridge**.

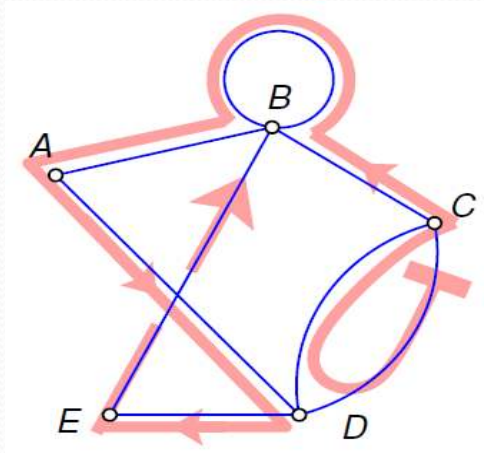
- Loops cannot be bridges, because removing a loop from a graph cannot make it disconnected.

- If two or more edges share both endpoints, then removing any one of them cannot make the graph disconnected. Therefore, none of those edges is a bridge.

Euler Paths & Euler Circuits





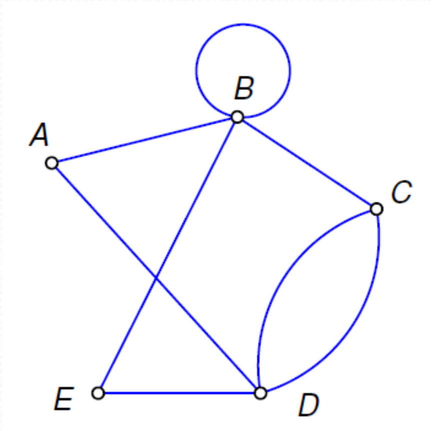


Vertex	Degree	Odd/Even
A	2	Even
B	5	Odd
C	3	Odd
D	4	Even
E	2	Even

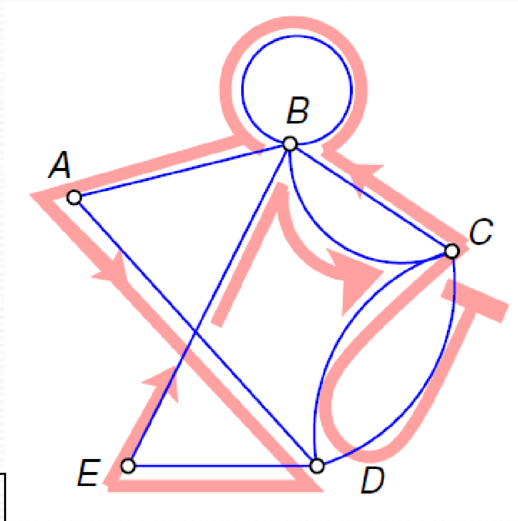
Euler Path:
BBADCDEBC

Euler Path:
CDCBBADEB

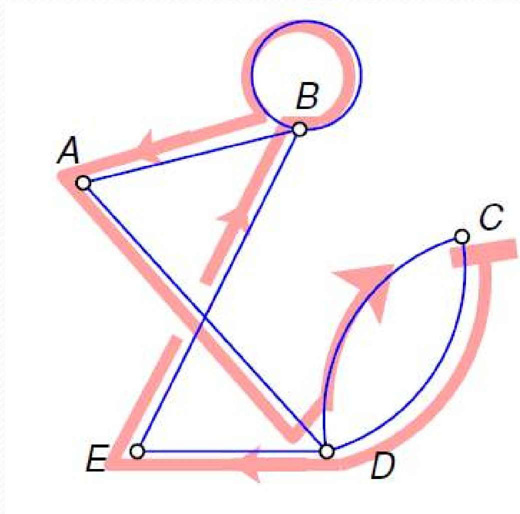
Euler Paths & Euler Circuits



Vertex	Degree	Odd/Even
A	2	Even
B	5	Odd
C	3	Odd
D	4	Even
E	2	Even



Euler Circuit : CDCBBADEBC



Euler Circuit : CDEBBADC

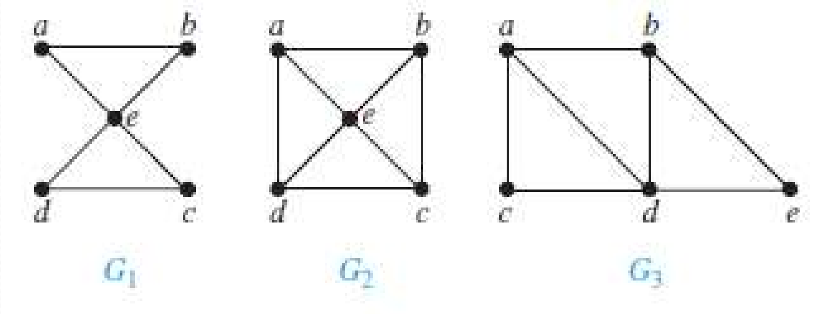
Euler Paths & Euler Circuits

❖Euler path & Euler circuit for directed graphs::

- For a directed graph to have Euler path, on each node, number of incoming edges should be equal to number of outgoing nodes except start node where out degree is one more than in degree and end node where incoming is one more than outgoing.
- To have Euler circuit, all nodes should have in degree equal to out degree.
- We have to keep in mind that for both directed and undirected graphs, above conditions hold when all nodes with non-zero degree are part of strongly connected component of graph.

Euler Paths & Euler Circuits

❖ **Example 27 :** Which of the undirected graphs in Figure have an Euler circuit? Of those that do not, which have an Euler path?



Solution:

➢ The graph G1 has an Euler circuit, for example, a, e, c, d, e, b, a.

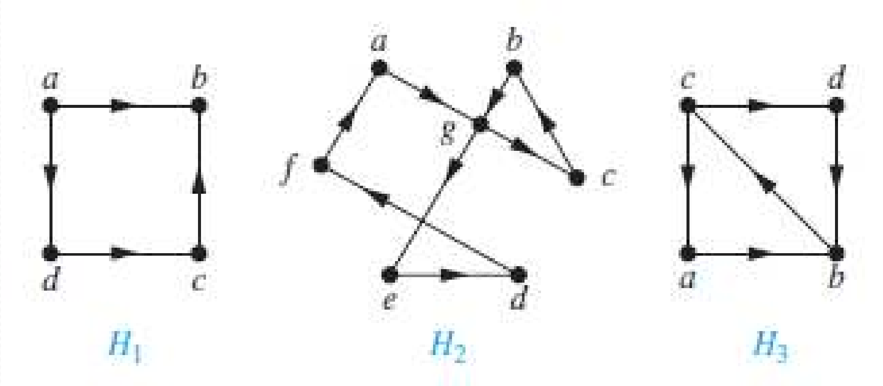
➢ Neither of the graphs G2 or G3 has an Euler circuit.

➢ G2 does not have an Euler path.

➢ G3 has an Euler path, namely, a, c, d, e, b, d, a, b.

Euler Paths & Euler Circuits

❖ **Example 28 :** Which of the directed graphs in Figure have an Euler circuit? Of those that do not, which have an Euler path?



Solution:

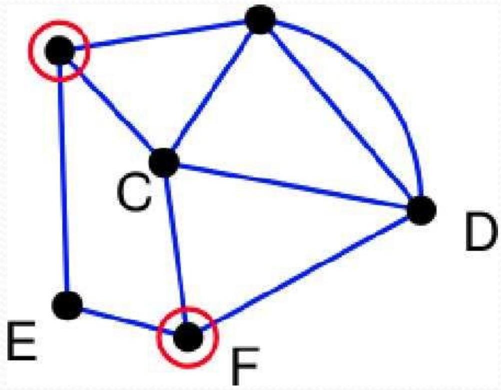
➢ The graph H2 has an Euler circuit, for example a, g, c, b, g, e, d, f, a.

➢ Neither H1 nor H3 has an Euler circuit.

➢ H3 has an Euler path, namely, c, a, b, c, d, b, but H1 does not.

Euler Paths & Euler Circuits

❖ **Example 29:** Finding Euler Circuits and Paths



Solution: Euler Path: FEACBDCFDBA

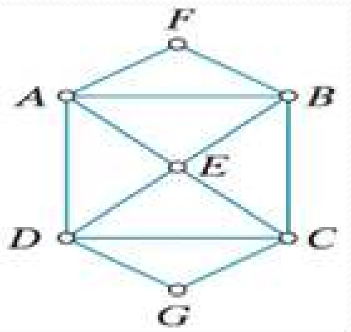
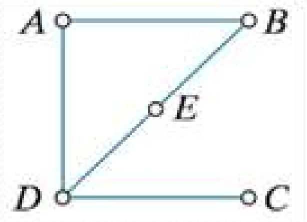
Hamilton Paths and Circuits

- A **Hamilton path** in a graph is a path that includes **each vertex** of the graph once and only once.

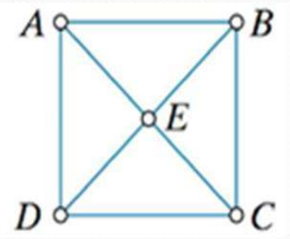
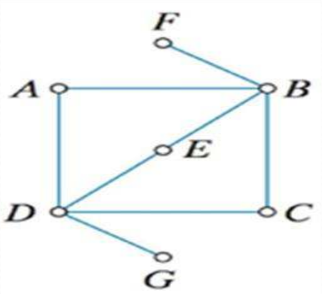
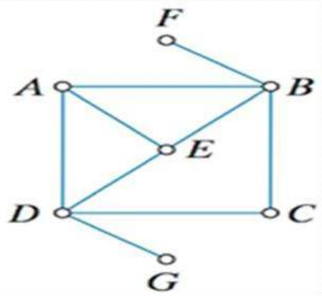
- A **Hamilton circuit** is a circuit that includes **each vertex** of the graph once and only once.

- In **Hamilton paths and Hamilton circuits,** the game is to find paths and circuits that include **every vertex** of the graph once and only once.

Hamilton versus Euler

	▪ If a graph has a Hamilton circuit, then it automatically has a Hamilton path-(the Hamilton circuit can always be truncated into a Hamilton path by dropping the last vertex of the circuit.) ▪ Contrast this with the mutually exclusive relationship between Euler circuits and paths: If a graph has an Euler circuit it cannot have an Euler path and vice versa. ▪ Hamilton circuit is A, F, B, C, G, D, E, A & ▪ Hamilton path is A, F, B, C, G, D, E
	▪ Has no Euler circuits but does have Euler paths as C, D, E, B, A, D. ▪ Has no Hamilton circuits (sooner or later you have to go to C, and then you are stuck) but does have Hamilton paths as A, B, E, D, C ▪ This shows that a graph can have a Hamilton path but no Hamilton circuit!.

Hamilton versus Euler

	▪ Has neither Euler circuits nor paths (it has four odd vertices) ▪ Has Hamilton circuits as A,B,C,D,E,A - there are plenty more) and consequently has Hamilton paths as A, B, C, D, E.
	▪ Has no Euler circuits but has Euler paths (F and G are the two odd vertices) and ▪ Has neither Hamilton circuits nor Hamilton paths.
	▪ Has neither Euler circuits nor Euler paths (too many odd vertices) and ▪ Has neither Hamilton circuits nor Hamilton paths

Shortest Path Problem

- In **graph theory**, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.
- The problem of finding the shortest path between two intersections on a road map (the graph's vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of its road segment) may be modeled by a special case of the shortest path problem in graphs.
- There are several different algorithms that find a shortest path between two vertices in a weighted graph.
- **Dijkstra's Algorithm::** is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

Dijkstra's Algorithm- Step:

- Dijkshtra's algorithm to find the shortest path from vertex a to z of a graph G. Let G(V,E) be a simple graph and **a, z ∈ V**.
- Suppose L(x) is the label of the vertex which represents the length of the shortest path from vertex a. Wij=Weight of an edge eij=(vi,vj).

- **Consider following Steps:**

- **Step 1:** Let P be the set of those vertices which have permanent labels and T be set of all vertices of G.

$$\text{Set } L(a) = 0, \quad L(x) = \infty \quad \forall\ x \in T \quad \text{and } x \neq a$$

$$P = \emptyset \text{ and } T = v.$$

Dijkstra's Algorithm- Step:

- **Step 2:** Select the vertex v in T which has smallest label. This label is called the permanent label of v. Also set P as P U {v} and T - {v}
  - ✓ If $v = z$ then L(z) is the length of the shortest path from the vertex a to z and stop the procedure.
- **Step 3:** If $v \neq z$, then revise the labels of the vertices of T. i.e. The vertices which do not have permanent labels.
  - ✓ The new label of x in T is given by

$$L(x) = \min\{\text{old } L(x), L(v) + w(v,x)\}$$

  - ✓ Where w(v,x) is the weight of the edge joining v and x. If there is no edge joining v and x then take $w(v,x)= \infty$.
- **Step 4:** Repeat the step 2 and 3 until z gets permanent label.

Dijkstra's Algorithm- Step:

❖ **Example 30:** Use Dijkstra's algorithm to find a shortest path between a and z.

Solution: Possible Path

a-b-d-z = 2+5+2=9

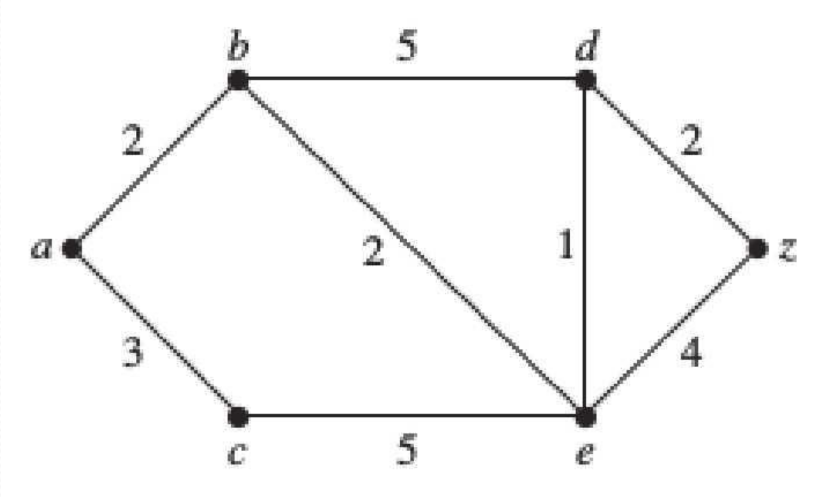
a-c-e-z = 3+5+4=12

a-b-e-z = 8

a-b-e-d-z=7

a-c-e-d-z =11

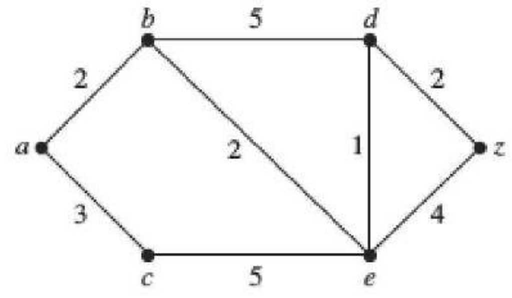
a-c-e-b-d-z = 17



Dijkstra's Algorithm- Step:

❖ **Example 30:** Use Dijkstra's algorithm to find a shortest path between a and z.

Solution: Step 1: Let P be the set of vertices which have permanent labels and T be set of all vertices of G.



$P = \emptyset$ and $T = \{a, b, c, d, e, z\}$.

Set $L(a) = 0$, $L(x) = \infty$

Step 2: v = a the permanent label of a = 0. $P = \{a\}$ and $T=\{b, c, d, e, z\}$

$L(b) = \min\{\text{old } L(b), L(a) + w(a,b)\} = \min\{\infty, 0 + 2\} = 2$

$L(c) = \min\{\text{old } L(c), L(a) + w(a,c)\} = \min\{\infty, 0 + 3\} = 3$

$L(d) = \min\{\text{old } L(d), L(a) + w(a,d)\} = \min\{\infty, 0 + \infty\} = \infty$

$L(e) = \min\{\text{old } L(e), L(a) + w(a,e)\} = \min\{\infty, 0 + \infty\} = \infty$

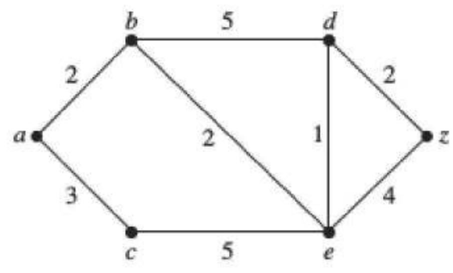
$L(z) = \min\{\text{old } L(z), L(a) + w(a,z)\} = \min\{\infty, 0 + \infty\} = \infty$

Therefore L(b) = 2 is minimum label.

Dijkstra's Algorithm- Step:

Step 3: v = b the permanent label of b = 2.   P = {a, b} and T={c, d, e, z}

L(c) = min{old L(c), L(b) + w(b,c)} = min{3, 2 + ∞} = 3



L(d) = min{old L(d), L(b) + w(b,d)} = min{∞, 2 + 5} = 7

L(e) = min{old L(e), L(b) + w(b,e)} = min{∞, 2 + 2} = 4

L(z) = min{old L(z), L(b) + w(b,z)} = min{∞, 2 + ∞} = ∞

Therefore L(c) = 3 is minimum label.

Step 4: v = c the permanent label of c = 3.   P = {a, b, c} and T={d, e, z}

L(d) = min{old L(d), L(c) + w(c,d)} = min{7, 3 + ∞} = 7

L(e) = min{old L(e), L(c) + w(c,e)} = min{4, 3 + 5} = 4

L(z) = min{old L(z), L(c) + w(c,z)} = min{∞, 3 + ∞} = ∞

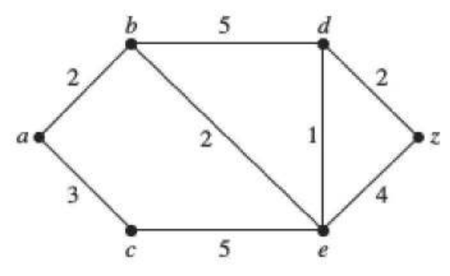
No labels are changed. Then e is put into P

Therefore L(e) = 4 is minimum label.

Dijkstra's Algorithm- Step:

Step 5: v = e the permanent label of e = 4.   P = {a, b, c, e} and T={d, z}

L(d) = min{old L(d), L(e) + w(e,d)} = min{7, 4 + 1} = 5



L(z) = min{old L(z), L(e) + w(e,z)} = min{∞, 4 + 4} = 8

Therefore L(d) = 5 is minimum label.

Step 6: v = d the permanent label of d = 5.   P = {a, b, c, e, d} and T={z}

L(z) = min{old L(z), L(d) + w(d,z)} = min{8, 5 + 2} = 7

Therefore L(z) = 7 is minimum label.

Step 7: v = z the permanent label of z is 7.

Therefore a shortest path is a, b, e, d, z, with length 7.

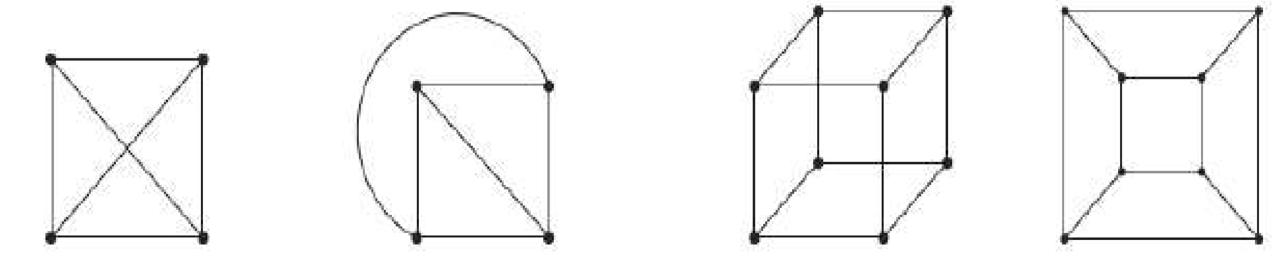
Planner Graph

- ❖ A graph is called **planar** if it can be drawn in the plane without any edges crossing (where a crossing of edges is the intersection of the lines or arcs representing them at a point other than their common endpoint). Such a drawing is called a planar representation of the graph.
- ❖ A graph may be planar even if it is usually drawn with crossings, because it may be possible to draw it in a different way without crossings.
- ❖ We can show that a graph is planar by displaying a planar representation. It is harder to show that a graph is nonplanar.

Planner Graph

❖ **Example 31:** Is K4 and Q3 (shown in Figure a and c) planar?

Solution: K4 and Q3 are planar because it can be drawn without crossings, as shown in figure b and d.



a) The Graph K_4 b) K_4 Drawn with No Crossings c) The Graph Q_3 d) A Planar Representation of Q_3