UNIT-IV

Graph : Basic terminology, directed and undirected graphs, path and connectivity, types of graphs- Null, Regular, Complementary, Complete, Weighted and Bipartite. Subgraphs, Operation on graphs- union, intersection, complement , product and composition. Representation of graphs in computer memory(matrix representation)- Adjacency matrix, Incidence matrix. Fusion of graphs. Isomorphic and Homeomorphic graphs, paths and cycles, Eulerian and Hamiltonian graphs, shortest path algorithm. Planar graphs, graph coloring. S Shortest path algorithms. Travelling salesman problem.

8

GRAPH

Graph

Graph G is a structure which have two elements :-

(a) Set of Vertices (V)

(b) Set of Edges (E)

So, structure (V,E) is a graph.

The elements of V are known as vertices, points or nodes. These are represented by '.' (dots) and use to locate cities, locations, places etc.

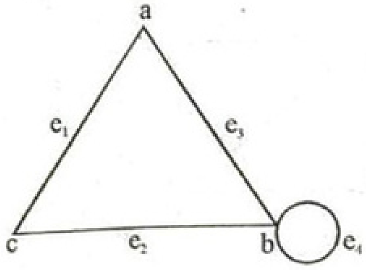
The elements of E are known as edges. These are represented by lines or curves and use to connect vertices. An edge is undered pair of vertices.

Eg. $V = \{a,b,c\}$

$E = \{e_1, e_2, e_3, e_4\}$

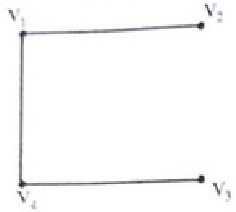
Where e1 (a,c), e_2 (b,c), e_3 (a,b), e_4 (b,b)

So the graph be:



Eg. $V = \{ v_1, v_2, v_3, v_4 \}$

$E = \{ (v_1, v_2), (v_1, v_4), (v_3, v_4) \}$



Applications of Graph

Graph are useful in network analysis like :

- Transportation network
- Water supply network
- Electricity supply network
- Security network

Degree of a Graph

Number of vertices in the graph is known as degree of graph. It is represented by |V|.

Size of a Graph

Number of edges in the graph is known as size of graph. It is represented by |E|.

|V| = 5

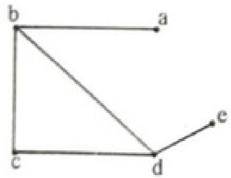
|E| = 6

Degree of a vertex

Total number of edges associated with a vertex is known as degree of vertex.

For a vertex v, it is represented by deg (v).

eg.



deg (a) = 1

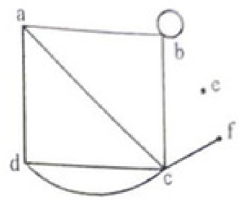
deg (b) = 3

deg (c) = 2

deg (d) = 3

deg (e) = 1

eg.



deg (a) = 3

deg (b) = 4

deg (c) = 5

deg (d) = 3

deg (e) = 0

deg (f) = 1

Types of Vertex :

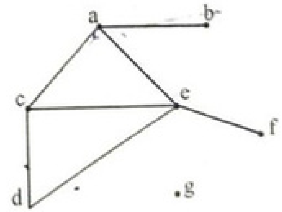
a. Pendent Vertex : A vertex which has degree 1 is known as pendent vertex.

b. Isolated Vertex : A vertex which have degree 0 is known as isolated vertex.

c. Even Vertex : A vertex which have degree even is known as even vertex.

d. Odd Vertex : A vertex which have degree odd is known as odd vertex.

eg.



Pendent Vertex : { b,f }

Isolated Vertex : { g }

Even Vertex : { d,e,g }

Odd Vertex : {a,b,c,f }

Note :

In any graph, the sum of all vertices degree is twice of total number of edges in the graph.

i.e. $\sum \deg(v) = 2|E|$

$$\Rightarrow |E| = \frac{\sum \deg(v)}{2}$$

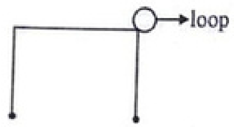
Note :

In any graph, number of vertices of odd degree is always even.

Special type of edges :

Loop : An edge whose start and end vertex is same i.e. an edge on a single vertex is known as loop.

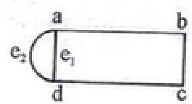
eg.



Parallel edge :

Two edge e_1 and e_2 are known as parallel if both edges are incident of same vertices i.e. both edges starting and ending points are same.

eg.



$e_1 \parallel e_2$

i.e. e_1 is parallel to e_2.

Different types of Graph :

- Undirected Graph (graph)
- Directed Graph (digraph)
- Finite Graph
- Infinite Graph
- Weighted Graph
- Simple Graph
- Multi Graph
- Pseudo Graph
- Null Graph
- Trivial Graph

- Regular Graph
- Complete Graph
- Cycle
- Wheel
- Bipartite Graph
- Complete Bipartite Graph
- Planer Graph
- Non Planer Graph
- Euler Graph
- Hamiltonian Graph
- Connected Graph
- Disconnected Graph
- Complementary Graph

Undirected Graph :

A graph in which edges have no directions, is known as undirected graph i.e. set of edges is unordered pair of vertices.

i.e. e (u,v) and e (v,u) are same edges.

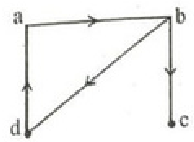
eg.

Directed Graph (digraph) :

A graph in which all edges have a specified direction is known as directed graph. It is also known as Digraph. In Digraph, set of edges is ordered pair of vertices.

i.e. e (u,v) and e (v,u) are different edges.

eg.



Degree in Digraph

In a digraph, a vertex have 2 types of degrees : Indegree and outdegree.

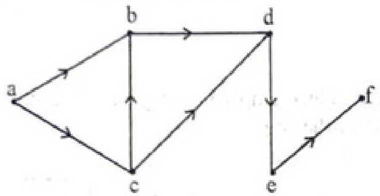
Indegree : Number of incoming edges to the vertex.

For any vertex v it is represented as $deg^-(v)$

Outdegree : Number of outgoing edges from the vertex.

For any vertex v it is represented as $deg^+(v)$

eg.



$deg^-(a) = 0$ $deg^+(a) = 2$

$deg^-(b) = 2$ $deg^+(b) = 1$

$deg^-(c) = 1$ $deg^+(c) = 2$

$deg^-(d) = 2$ $deg^+(d) = 1$

$deg^-(e) = 1$ $deg^+(e) = 1$

$deg^-(f) = 1$ $deg^+(f) = 0$

note : In a digraph, sum of all vertices indegree is equal to the sum of all vertices outdegree is equal to total number of edges in the digraph.

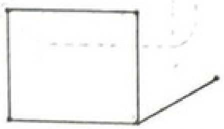
i.e. $$\sum deg^-(v) = \sum deg^+(v) = |E|$$

Finite Graph

A graph which degree is finite is known as finite graph.

i.e. A graph in which number of vertices are finite.

eg.



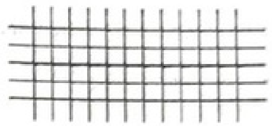
$|v| = 5$

Infinite Graph

A graph which degree is infinite is known as infinite graph.

i.e. A graph in which number of vertices are infinite.

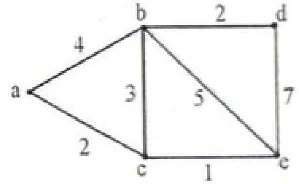
eg.



Weighted Graph

A graph is known as weighted graph if each graph edge has associated a number. The number is known as weight or cost of the edge.

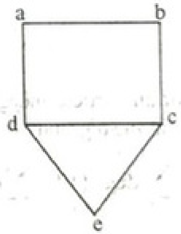
eg.



Simple Graph

A graph which not contains any loop or parallel edges is known as simple graph.

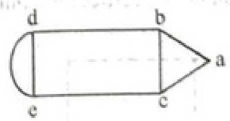
eg.



Multi Graph

A graph which contains parallel edges but not loops is known as multi graph.

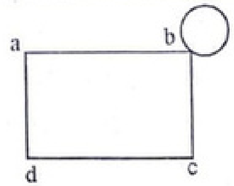
eg.



Pseudo Graph

A graph which contains loop is known as pseudo graph.

eg.

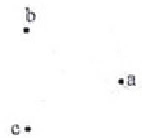


- **Null Graph**

  A graph which size is 0 is known as null graph.

  i.e. A graph which have only vertices, no edges is known as null graph.

eg.



Trivial Graph

A graph which has only one vertex and no edge is known as Trivial graph.

It is a minimum graph.

eg.

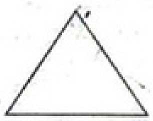
Note : Evrey trivial graph is also a null graph but not vice-versa.

Regular Graph

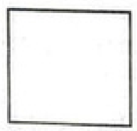
A graph is known as regular graph is each vertex degree is same.

If each vertex degree is r then it is known as r regular graph.

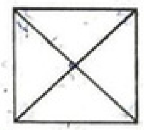
eg.



(2-regular graph)



(2-regular graph)



(3-regular graph)

Note : r regular graph with n vertex have total $(n.r)/2$ edges.

Eg. 3 regular graph with 4 vertex have $(4 \times 3)/2 = 6$ edges.

Complete Graph

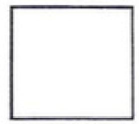
A graph is known as complete graph, if each vertex have edge with each other vertex.

A complete graph with n vertex is represented by K_n.

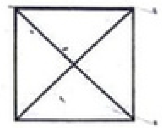
In complete graph K_n each vertex degree is $n - 1$. So K_n is also known as $(n - 1)$ regular graph.

Eg.

K_3



K_4



K_5

Note : A complete graph K_n have total $n(n-1)/2$ edges.

Eg. K_5 have $5(5-1)/2 = 10$ edges.

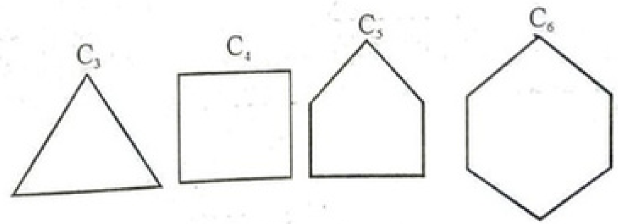
Note : Every complete graph is also a regular graph but not vice versa.

Cycle

Cycle is a special 2-regular graph which starting and ending vertex is same.

A cycle with n vertex is represented by C_n and here each vertex degree is 2.

Eg.



Note : Number of edges in C_n :

$$|E| = \frac{\sum \deg(v_i)}{2} = \frac{n \times 2}{2} = n$$

Wheel

Add a centre vertex in cycle C_n and connect this centre vertex to all other cyclic vertexes. It is represented by W_n which have n+1 vertexes.

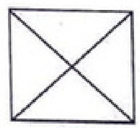
(n → Cyclic vertex and 1 center vertex)

Here, each cyclic vertex degree is 3 and centre vertex degree is n

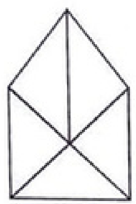
So, number of edges in W_n :

$$|E| = \frac{\sum \deg(v_i)}{2} = \frac{n.3 + 1.n}{2} = \frac{4n}{2} = 2n$$

Eg.



W_4



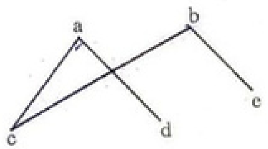
W_5

W_6

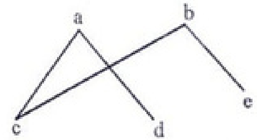
Bipartite Graph

A graph is known as bipartiate graph if we can partition (divide) the vertex set V into two disjoint subsets V_1 and V_2. Such that there be no edge between vertices of set V_1 and also there be no edge between vertices of set V_2. Edges may be possible between a vertex of set V_1 and another vertex of set V_2.

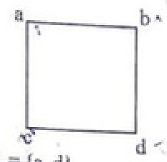
Eg.



$V_1 = \{a, b\}$
$V_2 = \{c, d, e\}$
Bipartite

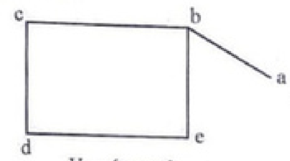


$V_1 = \{a, b\}$
$V_2 = \{c, d, e\}$
Bipartite



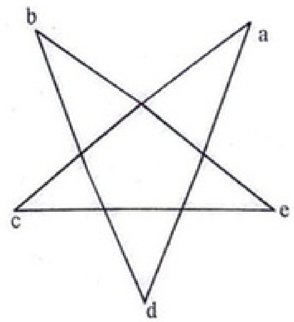
Bipartite

$V_1 = \{a, d\}$
$V_2 = \{c, b\}$

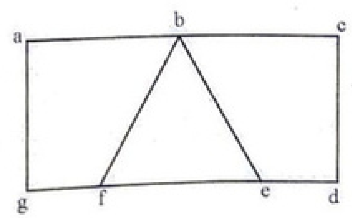


Bipartite

$V_1 = \{a, e, c\}$
$V_2 = \{b, d\}$



Not Bipartite

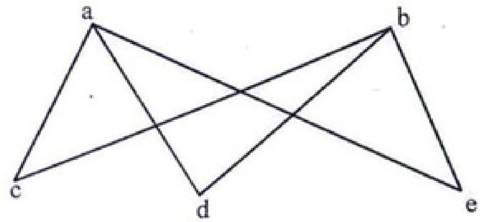


Not Bipartite

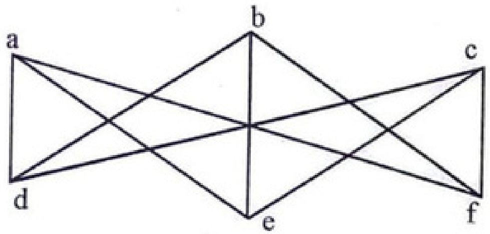
- **Complete Bipartite Graph**

A bipartite graph is known as complete bipartite graph, if every vertex of set V_1 has edge with every vertex of set V_2. It is represented by $k_{m,n}$ (Complete bipartite graph where set V_1 have m vertices and set V_2 have n vertices).

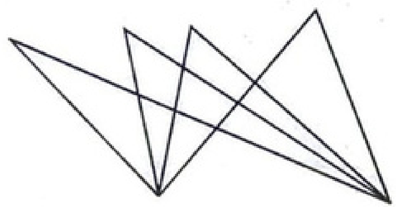
Eg. $K_{2,3}$



Eg. $K_{3,3}$



Eg. $K_{4,2}$:



Note : No. of edges in Km, n

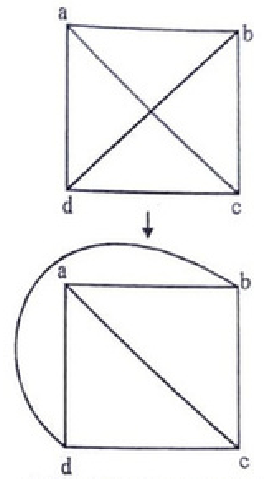
$= m.n$

Planer Graph

A graph is known as planer graph if the graph edges can be rearranged (if required) in such a manner that no edges intersect each other, then this graph is known as planer graph.

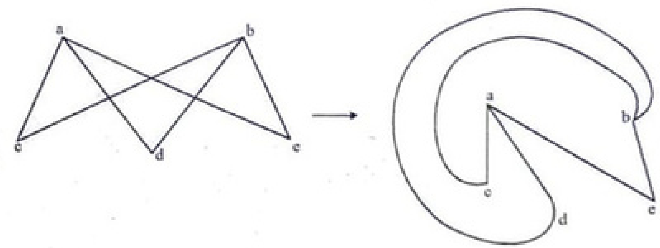
It this type of re-arrangement is impossible then the graph is known as non planer graph.

Eq.



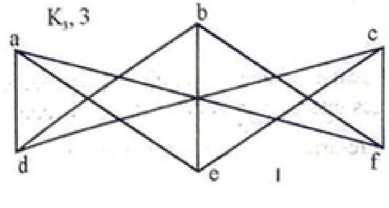
(planer) By rearrangement

Eq.

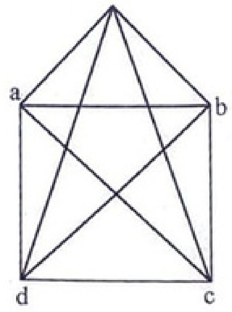


Planer graph

K_3, 3



Non Planer Graph



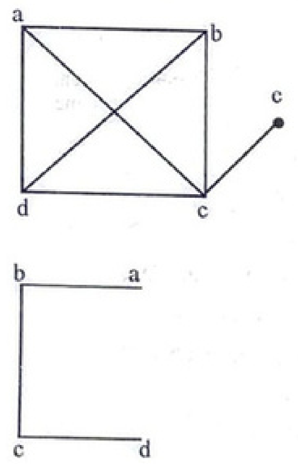
non planer graph

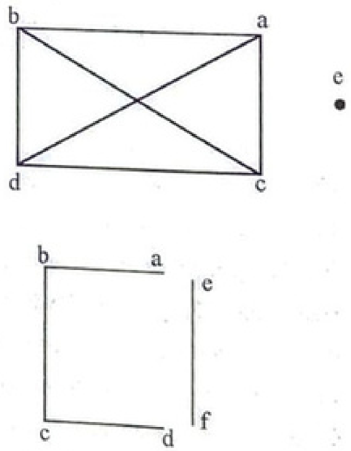
Connectdness in graph :

Undirected graph

- Connected Graph (If we can move from any vertex to any other vertex.)
- Disconnected Graph (If path is not available from all vertexes to all other vertexes)

Eq.

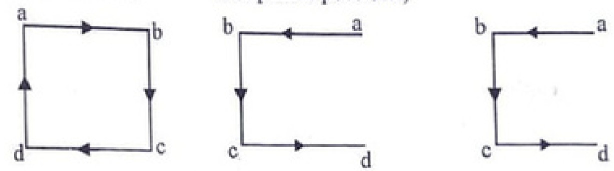




Directed Graph :

Directed Graph

- Strongly Connected Graph (If path is possible from any vertex to any other vertex according to direction.)
- Weakly Connected Graph (If according to direction path is not possible but if we ignore direction then path is possible.)
- Disconnected Graph (Path is impossible even after ignoring the direction.)



Walk mean

Alternate sequence of vertices and edges is known as walk.

Walk

- Open Walk: If starting and ending vertex is different.
- Close Walk: If starting and ending vertex is same.

Trail : An open walk with no edge repetition.

Path : An open walk with no vertex repetition.

Euler Trail : A trail which contains all edges

(i.e. An open walk with no edge repetition and have all edges.)

Hamiltonian Path : A path which contains all vertices.

(i.e. An open walk with no vertex repetition and have all vertices)

Circuit : A close walk with no edge repetition.

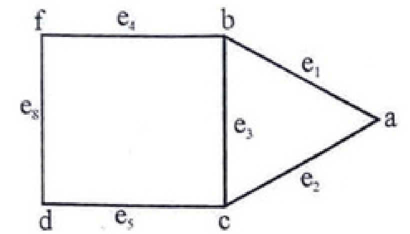
Cycle : A close walk with no vertex repetition.

Euler Circuit : A close walk with no edge repetition and have all edges.

Hamiltonian Cycle : A cycle which contains all vertices.

i.e. A close walk with no vertex repetition and have all vertices.

Eg.



Open Walk : a e1 b e3 c e5 d e6 f

Close Walk : a e2 c e5 d e6 f e4 b e1 a

Trail : a e1 b e3 c e5 d e6 f

Circuit : a e2 c e3 b e1 a

Path : a e1 b e3 c e5 d e6 f

Cycle : f e4 b e3 c e5 d e6 f

Euler Trail : a e1 b e4 f e6 d e5 c

Euler Crcuit : not available

Hamiltonian Path : a e1 b e4 f e6 d e5 c

Hamiltonian Cycle : a e1 b e4 f e6 d e5 c e2 a

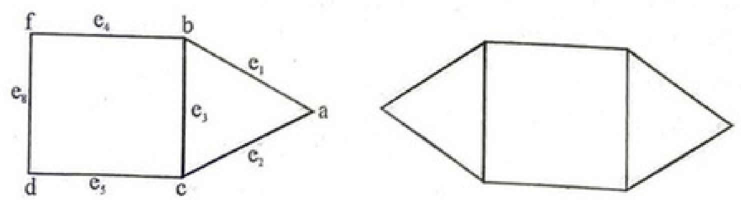
Euler Graph

A graph is known as Euler graph if it contain any euler circuit (i.e. close walk with no edge repetition and have all edges.)

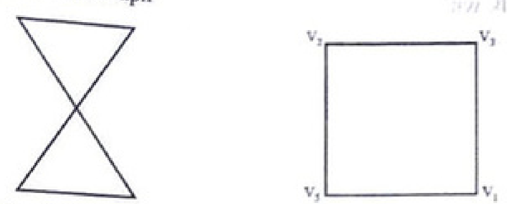
Hamiltonian Graph

A graph is known as Hamiltonian graph if it contain any Hamiltonian cycle (i.e. close walk with no vertex repetition and have all vertices.)

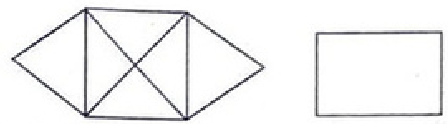
(a) Hamiltonian but not Euler Graph



(b) Euler but not Hamiltonian Graph



(c) Both Euler and Hamiltonian

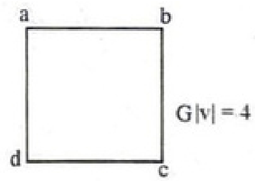


(d) Neither Euler nor Hamiltonian Graph

- **Complementary Graph** $\overline{G} = K_n - G$

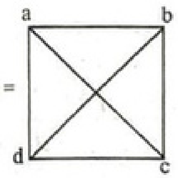
Complementary Graph $\overline{G}$ of graph G is a graph that contain all vertices of G but don't include any edges of G while contain the edge S which are not in G but exist in K_n

Eg. Find $\overline{G}$ of following :

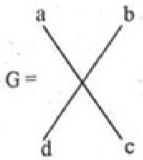


So

$K_4 =$



$\overline{G} = K_4 - G =$



Matrix representation of graph :

We can represent a graph in matrix form in two manners:

1. Adjacency Matrix :

V → V

C_{ij} — 1, if edge exist from Vi to Vj
— 0, if edge does not exist from Vi to Vj

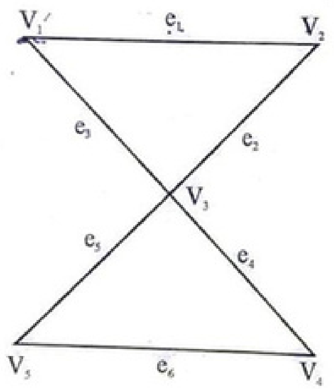
2. Incidency Matrix :

V → E

C_{ij} — 1, if edge e_j exist on V_i
— 0, if edge e_j not exist on V_i

(In digraph - 1 for incoming edge 1 for outgoing edge)

Eq.



Adjacency Matrix :

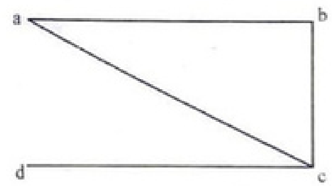
	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	0	0
v_2	1	0	1	0	0
v_3	1	1	0	1	1
v_4	0	0	1	0	1
v_5	0	0	1	1	0

Incidency Matrix :

	e_1	e_2	e_3	e_4	e_5	e_6
v_1	1	0	1	0	0	0
v_2	1	1	0	0	0	0
v_3	0	1	1	1	1	0
v_4	0	0	0	1	0	1
v_5	0	0	0	0	1	1

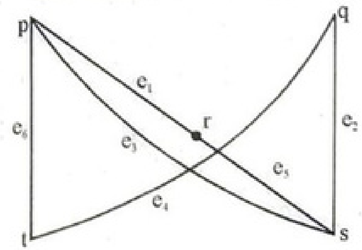
eg. Draw graph from following matrix :

	a	b	c	d
a	0	1	1	0
b	1	0	1	0
c	1	1	0	1
d	0	0	1	0

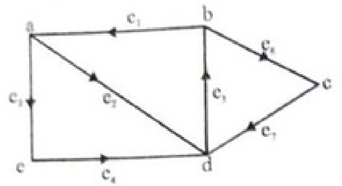


eg. Draw graph from following matrix :

	e_1	e_2	e_3	e_4	e_5	e_6
p	1	0	1	0	0	1
q	0	1	0	1	0	0
r	1	0	0	0	1	0
s	0	1	1	0	1	0
t	0	0	0	1	0	1



Eg. Find Adjancency and Incidency matrix of following digraph:

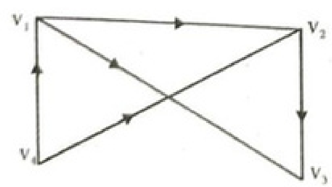


	a	b	c	d	e
a	0	0	0	1	1
b	1	0	1	0	0
c	0	0	0	1	0
d	0	1	0	0	0
e	0	0	0	1	0

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
a	−1	1	1	0	0	0	0
b	1	0	0	0	−1	1	0
c	0	0	0	0	0	−1	1
d	0	−1	0	−1	1	0	−1
e	0	0	−1	1	0	0	0

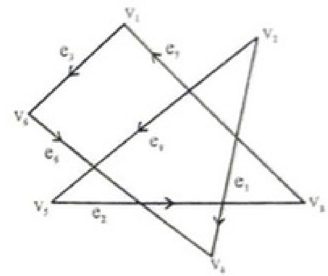
eg. Draw a digraph :

	v_1	v_2	v_3	v_4
v_1	0	1	1	0
v_2	0	0	1	0
v_3	0	0	0	0
v_4	1	1	0	0



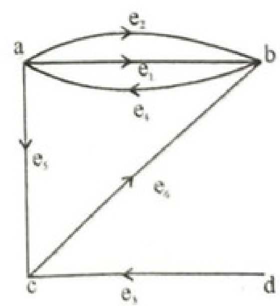
eg Draw a digraph :

	e_1	e_2	e_3	e_4	e_5	e_6
v_1	0	0	1	0	−1	0
v_2	1	0	0	1	0	0
v_3	0	−1	0	0	1	0
v_4	−1	0	0	0	0	−1
v_5	0	1	0	−1	0	0
v_6	0	0	−1	0	0	1



Draw a digraph :

	e_1	e_2	e_3	e_4	e_5	e_6
a	1	1	0	−1	1	0
b	−1	−1	0	1	0	−1
c	0	0	−1	0	−1	1
d	0	0	1	0	0	0



Isomorphism of Graph

Two graph G_1 and G_2 is known as isomorphic graph if they follow all these 3 properties :

1. $|V_1| = |V_2|$

i.e. Number of vertex in both graphs must be same.

2. $|E_1| = |E_2|$

i.e. Number of edges in both graphs must be same.

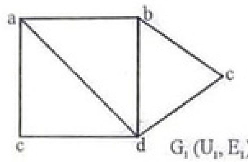
3. Degree sequence in both graphs must be same.

i.e. there exist a function F: $V_1 \rightarrow V_2$ that is one-one, onto

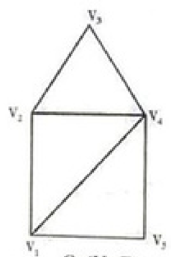
These kind of graph are known as isomorphic graphs represented by $G_1 \cong G_2$

G_1 is isomorphic with G_2 this process is known as isomorphism.

e.g.



$G_1 (U_1, E_1)$



$G_2 (V_2, E_2)$

$|V_1| = |V_2| = 5$

$|E_1| = |E_2| = 7$

$F(a) = v_1$

$F(b) = v_2$

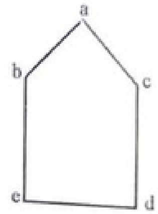
$F(c) = v_5$

$F(d) = v_4$

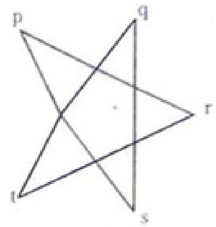
$F(e) = v_3$

so $G_1 \cong G_2$

Eg. Show that C_5 and $\overline{C_5}$ are isomorphic graph.



$G1 (V_1, E_1)$



$G_2 (V_2, E_2)$

$|V_1| = |V_2| = 5$

$|E_1| = |E_2| = 5$

$F(a) = p$

$F(b) = q$

$F(c) = r$

$F(d) = s$

$F(e) = t$

So $C_5 \cong \overline{C_5}$

Self Complimentary Graph

If G and $\overline{G}$ are isomorphic the G is known as self complementary graph.

eg. C_5

Graph Coloring and Chromatic numbers

Assign color to each vertex of the graph in such a way that no two adjacent vertex have the same coloured and required number of colors should be minimum.

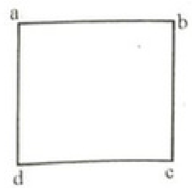
Required minimum number of color for this type of coloring is known as chromatic number.

It is represented by :

$\chi(G) = K$

χ-chi (chromatic number of graph G is k)

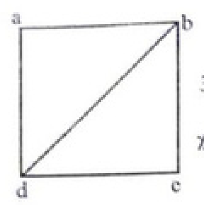
Eg.



2 colors

$\chi(G) = 2$

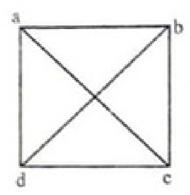
Eg.



3 colors

$\chi(G) = 3$

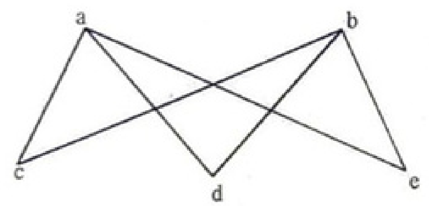
Eg.



4 colors

$\chi(G) = 4$

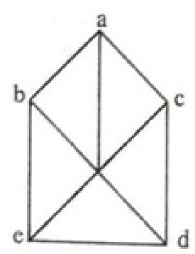
Eg.



2 colors

$\chi(G) = 2$

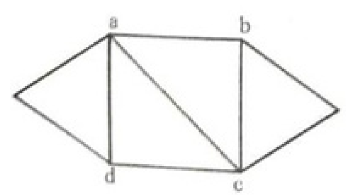
Eg.



4 colors

$\chi(G) = 4$

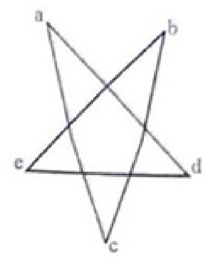
Eg.



3 colors

$\chi(G) = 3$

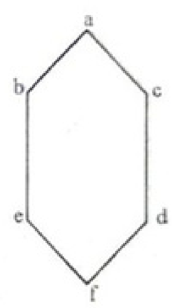
Eg.



3 colors

$\chi(G) = 3$

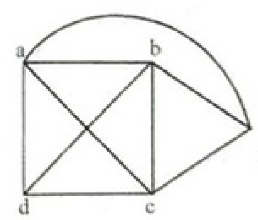
Eg.



2 colors

$\chi(G) = 2$

Eg.



3 colors

$\chi(G) = 3$

Chromatic number for special graphs

(a) K_n

$\chi(G) = n$

(b) C_n (n is even) :

$\chi(G) = 2$

(c) C_n (n is odd) :

$\chi(G) = 3$

(d) W_n (n is even) :

$\chi(G) = 3$

(e) W_n (n is odd) :

$\chi(G) = 4$

Single Source Shortest Path Problem :

Find the shortest path distance from the given source vertex to other vertex in the weighted graph, is known as single source shortest path problem.

To solve this problem, we use Djikstra alogorithm.

Steps to solve the shortest path problem by Dijkstra algo :

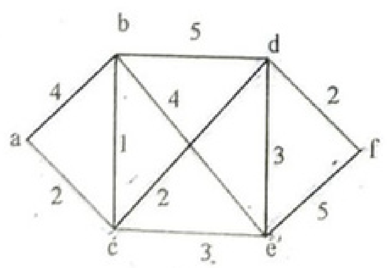
1. Assign 0 to source vertex and ∞ to other vertices initially.
2. Choose minimum assigned value vertex and select it. For this use □✓symbol.
3. Now, fix the selected vertex. For this use symbol □ and consider all vertex whose are adjacent to the selected vertex and assign updated value to these adjacent vertices by the following formula :

   $d(v) = \text{Min}\,[d(v),\ d(u) + w(u,v)]$

   Here u : selected vertex

   v : new vertex

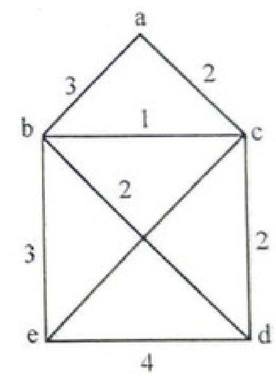
4. Repeat this process till not reach on destination. By using Dijkstra algo, we can find the shortest path distance from source to destination and the shortest path can be found by using BackTracking process.



a	b	c	d	e	f
[0]✓	∞	∞	∞	∞	∞
[0]	4	[2]✓	∞	∞	∞
[0]	[3]✓	[2]	4	5	∞
[0]	[3]	[2]	[4]✓	5	∞
[0]	[3]	[2]	[4]	[5]✓	∞
[0]	[3]	[2]	[4]	[5]	[6]✓

Shortest Path from a to f : a→c→ d→f

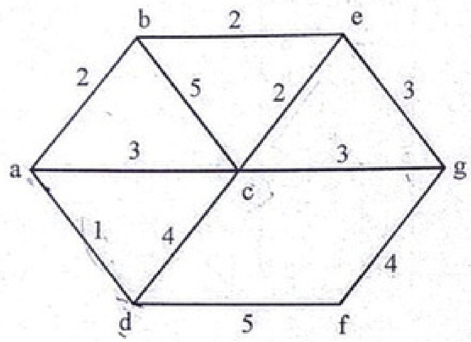
Eg. :



a	b	c	d	e
[0]✓	∞	∞	∞	∞
[0]	3	[2]✓	∞	∞
[0]	[3]✓	[2]	4	7
[0]	[3]	[2]	[4]✓	6
[0]	[3]	[2]	[4]	[6]✓

Shortest Path from a to e : a→b→e

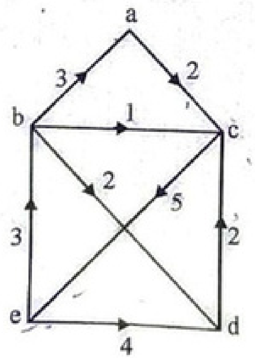
Eg.



a	*b*	*c*	*d*	*e*	*f*	*g*
[0] ✓	∞	∞	∞	∞	∞	∞
[0]	2	3	[1] ✓	∞	∞	∞
[0]	[2] ✓	[3]	[1]	∞	6	∞
[0]	[2]	[3] ✓	[1]	4	6	∞
[0]	[2]	[3]	[1]	[4] ✓	6	6
[0]	[2]	[3]	[1]	[4]	[6] ✓	6
[0]	[2]	[3]	[1]	[4]	[6]	6

Shortest Path a to g : a→c→g

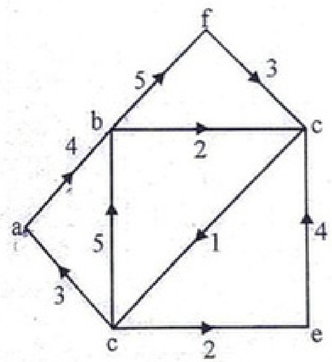
Eg. :



a	*b*	*c*	*d*	*e*
[0] ✓	∞	∞	∞	∞
[0]	∞	[2] ✓	∞	∞
[0]	∞	[2]	∞	[7] ✓
[0]	[1]	[2]	[11]	[7]
[0]	[1]	[2]	[11] ✓	[7]

Shortest Path from a to e : a→c→e

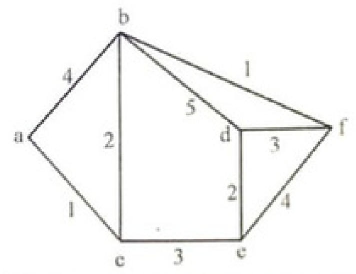
Eg.



a	*b*	*c*	*d*	*e*	*f*
∞	[0] ✓	∞	∞	∞	∞
∞	[0]	∞	[2] ✓	∞	5
∞	[0]	[3] ✓	[2]	∞	5
6	[0]	[3]	[2]	[5] ✓	5
6	[0]	[3]	[2]	[5]	[5] ✓
[6] ✓	[0]	[3]	[2]	[5]	[5]

Shortest Path from b to f : b→f

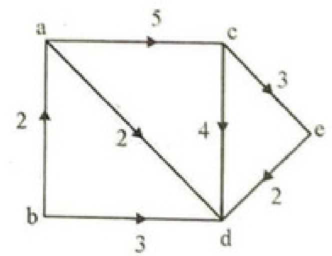
Eg.



a	*b*	*c*	*d*	*e*	*f*
[0]✓	∞	∞	∞	∞	∞
[0]	4	[1]✓	∞	∞	∞
[0]	[3]✓	[1]	∞	4	∞
[0]	[3]	[1]	8	[4]✓	4
[0]	[3]	[1]	6	[4]	[4]✓
[0]	[3]	[1]	[6]✓	[4]	[4]

Shortest Path from a to f : a→c→b→f

Eg .

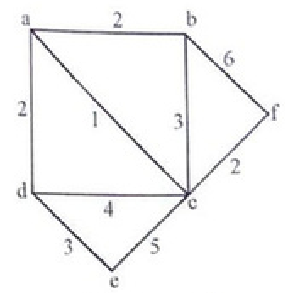


a	*b*	*c*	*d*	*e*
[0]✓	∞	∞	∞	∞
[0]	∞	5	[2]✓	∞
[0]	∞	[5]✓	[2]	∞
[0]	∞	[5]	[2]	∞

Shortest Path : (a to c) a→c

(a to d) a→d

Eg.



(Source e - a)

a	*b*	*c*	*d*	*e*	*f*
[0]✓	∞	∞	∞	∞	∞
[0]	[2]✓	1	2	∞	∞
[0]	[2]	[1]✓	2	∞	8
[0]	[2]	[1]	[2]✓	6	3
[0]	[2]	[1]	[2]	5	[3]✓
[0]	[2]	[1]	[2]	[5]✓	[3]

Shortest Path : (a to f) a→c→f

(Source e - b)

a	*b*	*c*	*d*	*e*	*f*
∞	[0]✓	∞	∞	∞	∞
[2]✓	[0]	3	∞	∞	6
[2]	[0]	[3]✓	4	∞	6
[2]	[0]	[3]	[4]✓	8	5
[2]	[0]	[3]	[4]	7	[5]✓
[2]	[0]	[3]	[4]	[7]✓	[5]

Shortest Path : (b to f): b→a→c→f

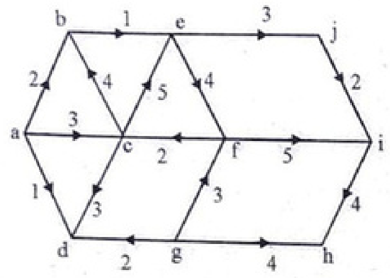
(b to c): b→a→c

(b to d): b→a→d

(b to e): b→a→d→e

(b to a): b→a

Eg.



a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
[0]✓	∞	∞	∞	∞	∞	∞	∞	∞	∞
[0]	2	3	[1]✓	∞	∞	∞	∞	∞	∞
[0]	[2]✓	3	[1]	∞	∞	∞	∞	∞	∞
[0]	[2]	3	[1]	[3]✓	∞	∞	∞	∞	∞
[0]	[2]	[3]✓	[1]	[3]	7	∞	∞	∞	6
[0]	[2]	[3]	[1]	[3]	7	∞	∞	∞	[6]✓
[0]	[2]	[3]	[1]	[3]	[7]✓	∞	∞	8	[6]
[0]	[2]	[3]	[1]	[3]	[7]	∞	∞	[8]✓	[6]
[0]	[2]	[3]	[1]	[3]	[7]	∞	[12]✓	[8]	[6]
[0]	[2]	[3]	[1]	[3]	[7]	∞	[12]	[8]	[6]

Shortest Path → (a to b): a→b

In one row one

a to b

a to c

a to d

a to d

a to e

a to f

(a to f) : a→b→e→f

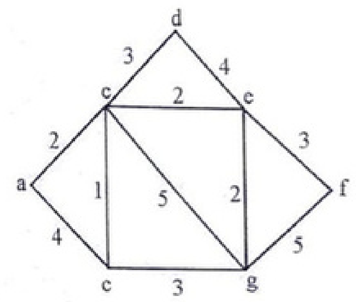
(a to c): a→c (a to h) : a→b→e→j→i→h

(a to d): a→d (a to i) : a→b→e→j→i

(a to e): a→b→e

(a to j) : a→b→e→j

Eg.



a	*b*	*c*	*d*	*e*	*f*	*g*
[0]✓	∞	∞	∞	∞	∞	∞
[0]	4	[2]✓	∞	∞	∞	∞
[0]	[3]✓	[2]	5	4	∞	7
[0]	[3]	[2]	5	[4]✓	∞	6
[0]	[3]	[2]	[5]✓	[4]	7	6
[0]	[3]	[2]	[5]	[4]	7	[6]✓
[0]	[3]	[2]	[5]	[4]	[7]✓	[6]

Shortest Path → (a to b): a→c→b

(a to c): a→c

(a to d): a→c→d

(a to e): a→c→e

(a to f) : a→c→e→f

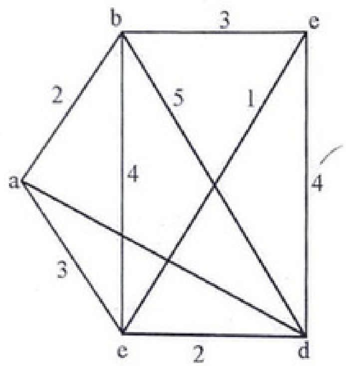
(a to g) : a→c→e→g

Travelling Salesman Problem : (TSP)

A salesperson have to visit the all given cities in such a way that no city will be revisit (i.e. repeat) and the cost or distance of this travelling must be minimum and he should return back to his hometown (Starting Place).

i.e. we have to find such a close path (route) which contain all vertices and no vertex should repeat. Here, cities are represented by vertices and connectivity between these cities are represented by weighted edge. Our target is to find a minimum cost Hamiltonian cycle.

Eg.



Hamiltonian cycles may be:

abcdea = 2+3+4+2+3 = 14

aecdba = 3+1+4+5+2 = 15

abecda = 2+4+1+4+7 = 18

Operations on Graphs

The basic operation that can be performed on two graphs are :

- Union
- Intersection
- Product
- Composition
- Fusion

Union of two graphs

Let two graphs

$G_1 (V_1, E_1)$

$G_2 (V_2, E_2)$

There union graph be G (V,E)

Where $V = V_1 \cup V_2$

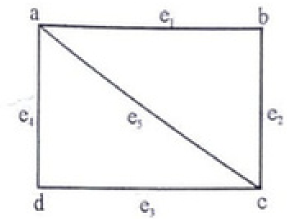
$E = E_1 \cup E_2$

This is represented as $G_1 \cup G_2$

e.g. $G_1 (V_1, E_1)$ where

$V_1 = \{ a,b,c,d \}$

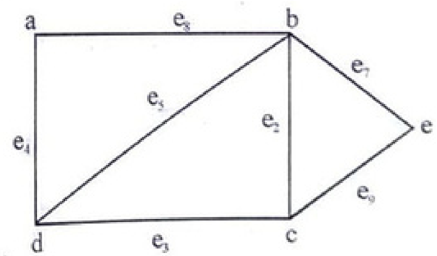
$E_1 = \{ e_1,e_2,e_3,e_4,e_5 \}$



$G_2 (V_2, E_2)$ where

$V_2 = \{ a,b,c,d,e \}$

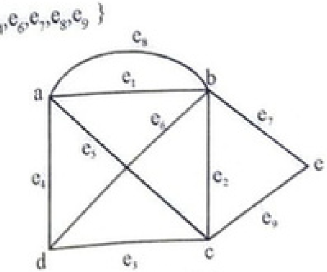
$E_2 = \{e_2, e_3,e_4,e_6,e_7,e_8,e_9 \}$



$G_1 \cup G_2 = (V, E)$

$V = V_1 \cup V_2 = \{ a,b,c,d,e \}$

$E = E_1 \cup E_2 = \{ e_1,e_2, e_3,e_4,e_6,e_7,e_8,e_9 \}$



$G = G_1 \cup G_2$

Intersection of two graphs

Let two graphs

$G_1(V_1, E_1)$

$G_2(V_2, E_2)$

Their intersection graph be G(V,E)

Where $V = V_1 \cap V_2$

$E = E_1 \cap E_2$

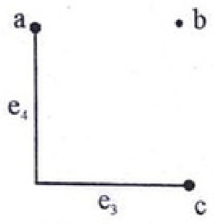
This is represented as $G_1 \cap G_2$

Eg. In the above example

$G_1 \cap G_2 = (V,E)$

$V = V_1 \cap V_2 = \{a,b,c,d\}$

$E = E_1 \cap E_2 = \{ e_3, e_4 \}$



$G = G_1 \cap G_2$

Product of two graphs :

Let two graphs

$G_1(V_1,E_1)$

$G_2(V_2,E_2)$

The product of these two graphs is sepresented by $G_1 \times G_2$.

Vertex set of $G_1 \times G_2$ is $V_1 \times V_2$.

Edge set of $G_1 \times G_2$ is defined as :

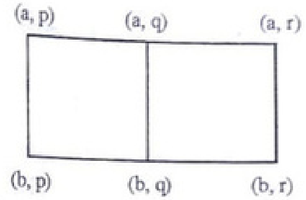
There be edge in between any two vertex (u_1, u_2) and (v_1, v_2) if

(a) $u_1 = v_1$ and $(u_2, v_2) \in E_2$ or

(b) $u_2 = v_2$ and $(u_1, v_1) \in E_1$

Eg.

$G_1 \times G_2$



Composition of two graphs :

Let two graphs

$G_1(V_1, E_1)$

$G_2(V_2, E_2)$

the compostion of the two graphs is represent by $G_1[G_2]$ or $G_2[G_1]$.

$G_1[G_2](V, E)$ where

$V = V_1 \times V_2$

and the edge set E of $G_1[G_2]$ is defined as :

there be edge in between any two vertex (u_1, u_2) and (v_1, v_2)

(a) $u_1 = v_1$ and $(u_2, v_2) \in E_2$ or

(b) $u_1 \neq v_1$ and $(u_1, v_1) \in E_1$

$G_2[G_1](V, E)$ where

$V = V_2 \times V_1$

and the edge set E of $G_2[G_1]$ is defined as :

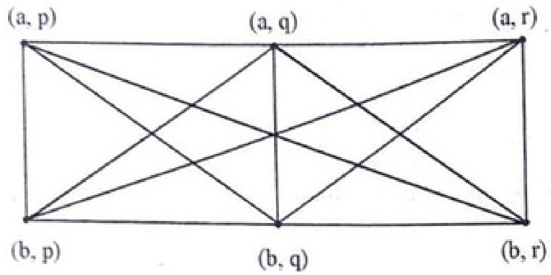
there be edge in between any two vertex (u_1, u_2) and (v_1, v_2) if :

(a) $u_2 \neq v_2$ and $(u_1, v_1) \in E_1$ or

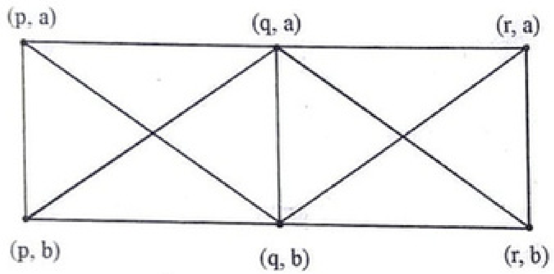
(b) $u_2 \neq v_2$ and $(u_2, v_2) \in E_2$

Eq.

$G_1 [G_2]$



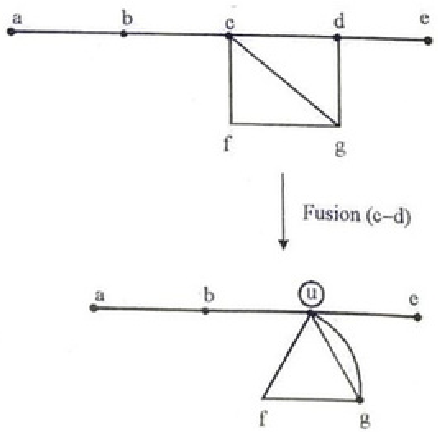
$G_2 [G_1]$



Fusion of graph :

Let G (V, E) be a graph and v_i and v_j be two distinct vertices of graph G. A new graph G' can be formed by fusing these two vertices and replacing then by a single new vertex u such that every edge that was incident with either v_i or v_j in G is now incident with u in G'

Eg.



Subgraph :

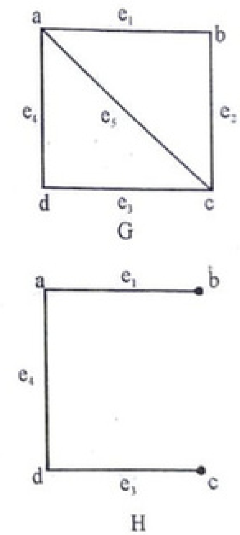
A graph $H(V_1, E_1)$ be subgraph of graph $G(V_2, E_2)$ if every vertex and edge of H is also exist in G

i.e. $V_1 \subseteq V_2$

$E_1 \subseteq E_2$

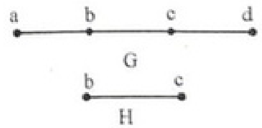
If H is subgraph of G then it is sepresented by $H \subseteq G$

Eg.



$H \subseteq G$

Eg.



$H \subseteq G$

Note :

- Each graph is subset of itself
- Subgraph of subgraph of G is also subgraph of G.

Homeomorphic graphs :

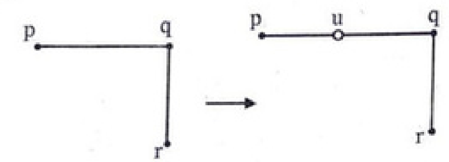
Two graphs G and H be homeomorphic graphs, if there is an isomorphism from a subdivision of G to a subdivision of H.

i.e. one graph can be obtained from the other graph by insertion of vertices of degree two or by the merger of edges in series.

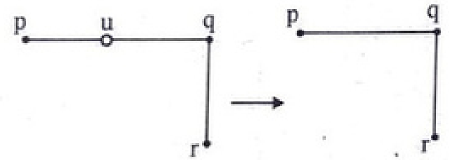
Insertion of vertex by done by subdividing an edge. i.e. insert a new vertex of degree two into the edge, there by creating edge, there by creating edges in series.

merger of edges by done by smoothing away a vertex i.e. replaces two edges that meet at a vertex of degree 2 by a single edge that joins their other and points.

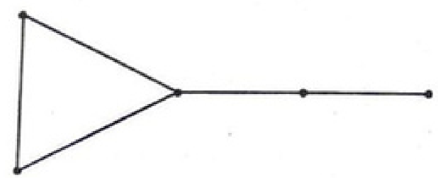
Eg. Subdividing in edge :



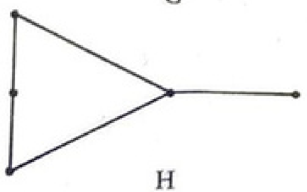
Eg. Merger of edge :



Eg. Homeomorphic graphs :



G



H

Eg.

G

H

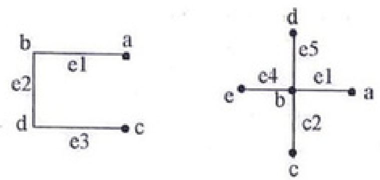
Questions

Very Short Answer Type :

1. Define
   (a) Null graph
   (b) Regular graph
   (c) Pendent vertex
   (d) Planer graph
2. Find number of edges in k_7,
3. Find number of vertices in w_5
4. Find number of edges in a graph which have 2 vertex of degree 4 and 4 vertex of degree 3.
5. Find chromatic number of w_6.
6. What do you mean by weight of an edge.
7. Draw a graph which is ruler but not hamiltonion.
8. What is self complementory graphs.
9. Define subgraph.
10. A graph size is 25 & degree is either 3 or 5. Find number of vertices of degree 3 and degree 5.

Short Answer Type

1. Explain TSP.
2. Show that the sum of all vertices degree is twice of total number of edges in the graph.
3. Show that number of vertices of odd degree vertex in any graph is always even.
4. Define Bipartite graph with example.
5. Draw the graph of 6 vertex which is both Euler & Homiltonial.
6. Show that graph k_6 is non planer.
7. Find complementary graph of k_3.
8. What do you mean by Isomarphic graphs.
9. Find union and intersection of following graphs :



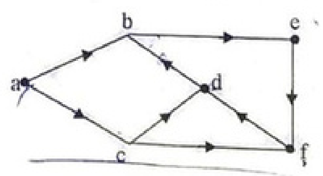
Long Answer Type :

1. What is graph? Write its applications and types.
2. Compute

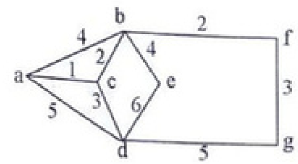
$$\sum_{i=1}^{n} E(k_i)$$

$E(k_i)$: number of edges in graph k_i.

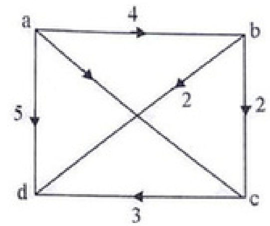
3. Show that a complete Bipartite graph with total n vertex have max $n^2/4$ edges.
4. Find adjancy and incidency matrix for following digraph :



5. Find shortest path from b to all other vertices in the given graph :



6. Find shortest path from a to d.



❑❑❑