Matrix Representation of Graph —

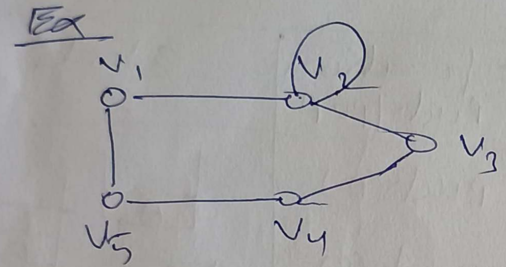
① Adjacency Matrix —

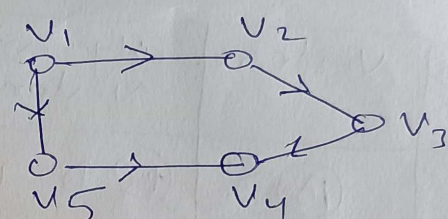
Let a_{ij} is number of edges (v_i, v_j) then

$A = [a_{ij}]_{m \times n}$ is called Adjacency matrix of G if

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

Ex





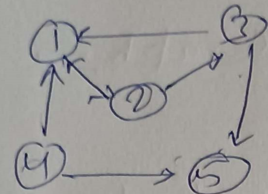
	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	0	1
v_2	1	ϕ	1	0	0
v_3	0	1	0	1	0
v_4	0	0	1	0	1
v_5	1	0	0	1	0

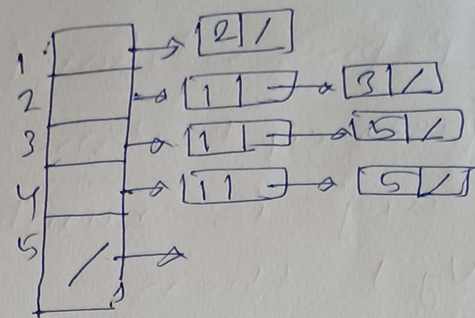
	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	0	1
v_2	0	0	1	0	0
v_3	0	0	0	1	0
v_4	0	0	0	0	0
v_5	0	0	0	1	0

② Adjacency List —

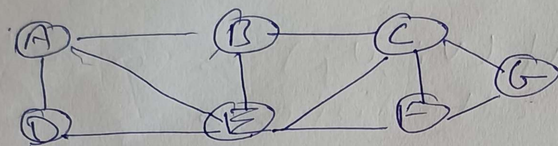
- it's an array of Lists
- each list showing the vertices a given vertex is adjacent to...

Ex



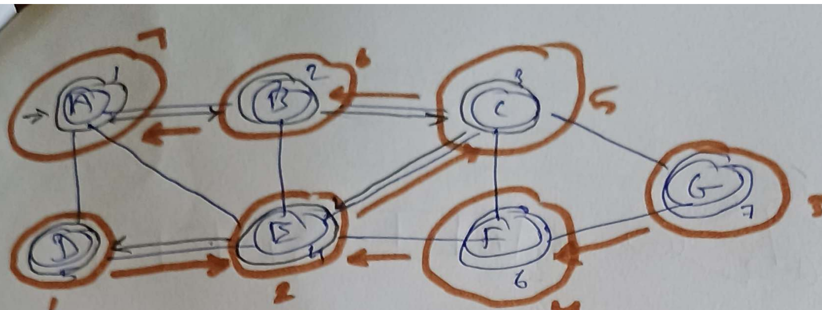


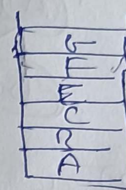
DFS



S.No	Vertex	adjacent Vertices	operation Push/POP	stack
1	A	B D E	push(A)	A
2	B	A E C	push(B)	B A
3	C	B E F G	push(C)	C B A
4	E	D A B C F	push(E)	E C B A
5	D	—	push(D)	D E C B A
6	D	use Backtrack at E	POP (D) make it final visit of D	E C B A
7	E	no new vertex from E Backtrack at C	POP(E) make it final	C B A
8	C	Backtrack B	POP(C) make it final	B A
9	B	Backtrack A	POP(B)	A
10	A		POP(A)	

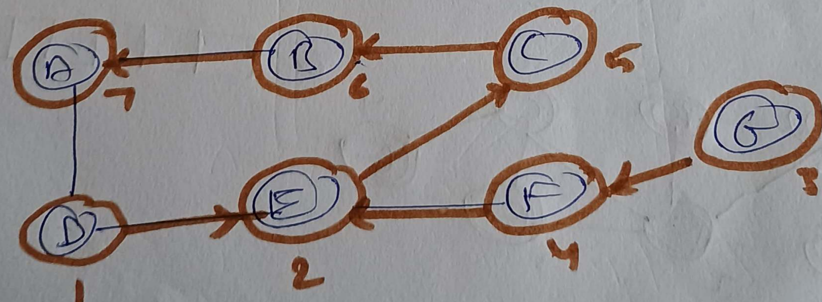
2





7	F	adjacent of E is F pair	Push(F)
8	G	adj of F is G	Push(G)
9	G	no new node at G so Backtrack	POP(G) mark it final
10			POP(F)
			POP(E)
11			POP(C)
12			POP(B)
13			POP(A)
14			

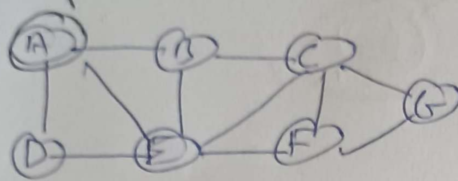
Final



B

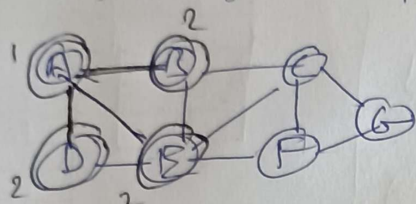
BFS

(1)



Q: | A | | | | | |

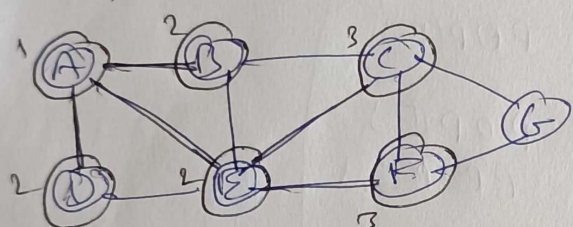
(2) adjecent of A is B, D, E insert it into Queue and visit all & delete A



Q: | | D | E | B | | | |

(3) adj of D is no; Delete D Q: | | | E | B | |

(4) adj of E is C, F; insert it into Queue & Delete E



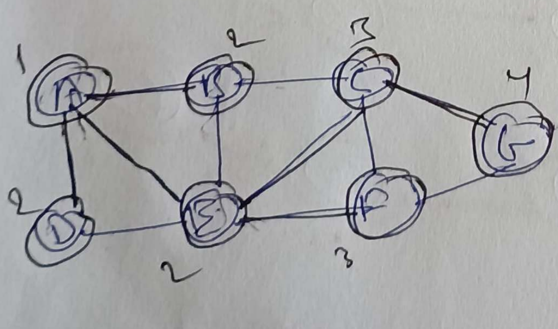
Q: | | | | B | C | F |

(5) adj of B are visited so Delete B

Q: | | | | | C | F |

(6) adj of C is G; insert G & Delete C

Q: | | | | | F | G |



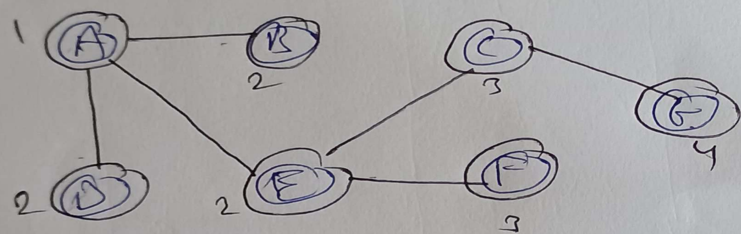
(7) adj of F is no so Delete F

Q: | . | . | . | . | . | . | G |

(8) adj of G is no; so Delete G

Q: | . | . | . | . | . | . | . |

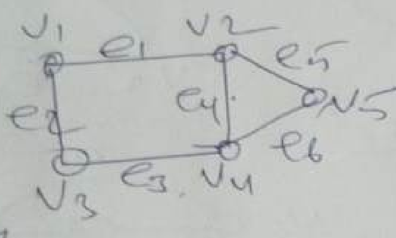
(9) Final Result of BFS is spanning Tree

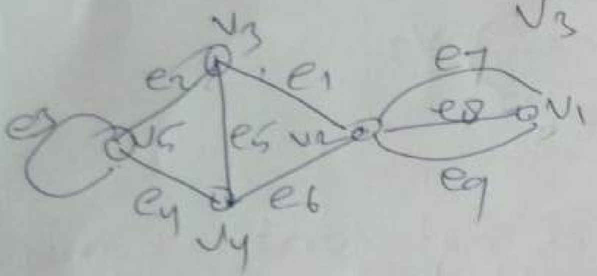


Graph

$G = (V, E)$

$V \rightarrow$ set of Vertices
$E \rightarrow$ " " Edge



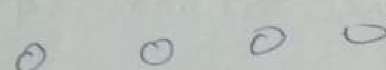


Basic terminology

① trivial Graph.
⇒ only one vertics.
no edge.

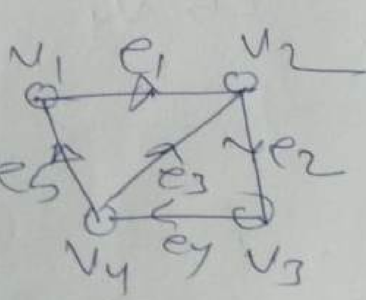
$\boxed{\begin{matrix} V = 1 \\ E = 0 \end{matrix}}$

Null Graph

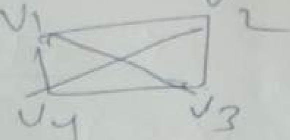


$\boxed{\begin{matrix} V = n \\ E = 0 \end{matrix}}$

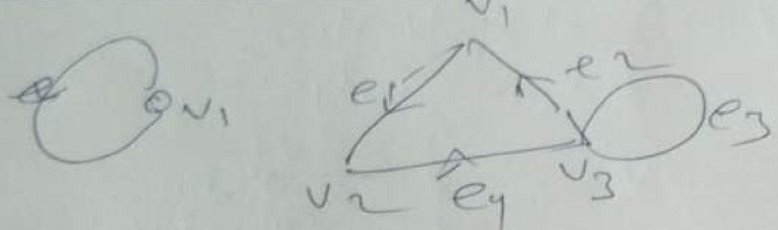
Directed Graph.



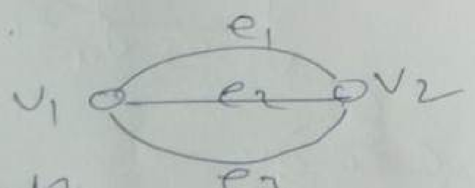
undirected graph —



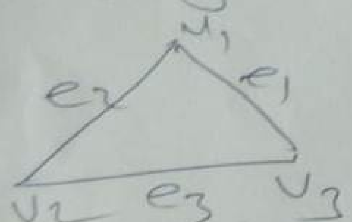
self loop in Graph —



Multiedge.



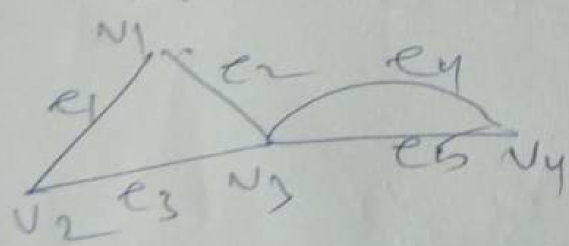
simple graph.



does not contain any self loop.

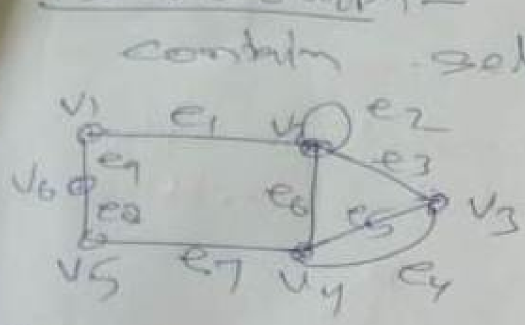
Multigraph —

No self loop
but have multiedge.



Pseudo Graph —

contain selfloop and multiedge.

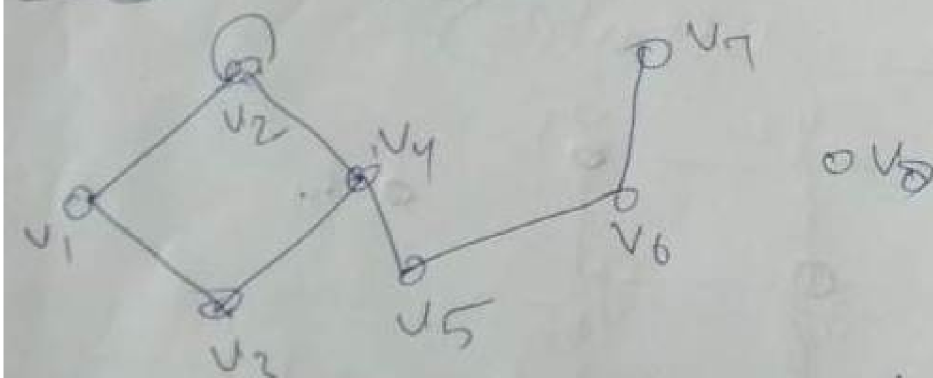


Incedence and Adjacency —

(a) e_1 is incedence
v_1, v_2 is Adjacent Vertex

(b) e_2 is incedence
$v_1\ v_3$ is adjacent

Degree of Vertex —



$d(V)$ = Total edges incedent on V.

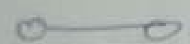
$d(V_1) = 2$ $d(V_2) = 2+2 = 4$

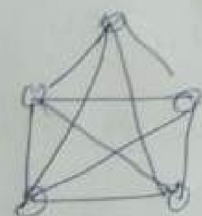
$d(V_4) = 3$ $d(V_8) = 0$

Types of Graph —

- complete Graph.

- A simple connected graph.
- each vertex connected with every other vertex.





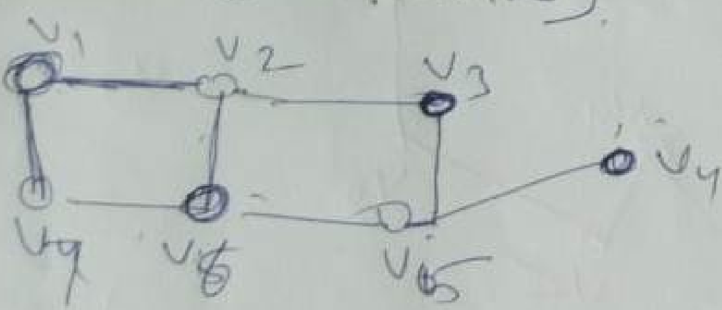
Regular Graph

if every vertex have same degree.

If degree of each vertex of Graph G is K then it is K-regular graph.

2-Regular Graph.

Bigraph = (Bipartite).



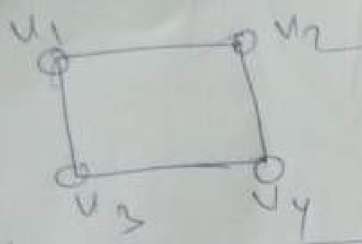
$X = \{ V_1, V_6, V_3, V_4 \}$

$Y = \{ V_2, V_7, V_5 \}$

Where X Y are two disjoint set.

one ~~ete~~ end of edge is in X and other in Y

Complete Bipartite Graph

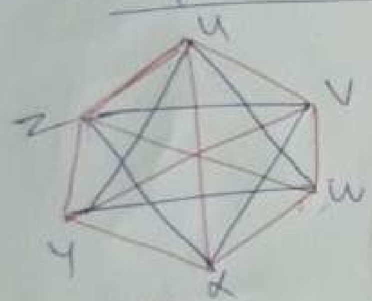


$X = \{ v_1, v_4 \}$

$Y = \{ v_2, v_3 \}$

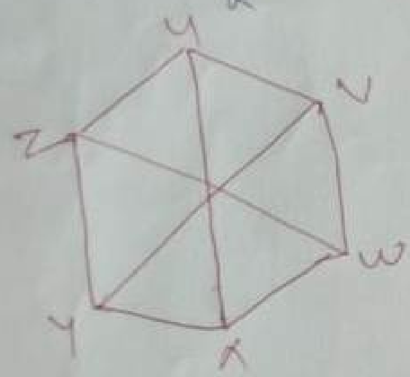
This is $\boxed{K_{2,2}}$ every vertex of X must be connected with every vertex of Y and vice-versa.

Complement of graph

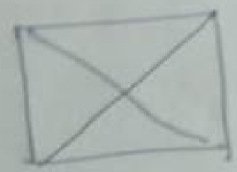


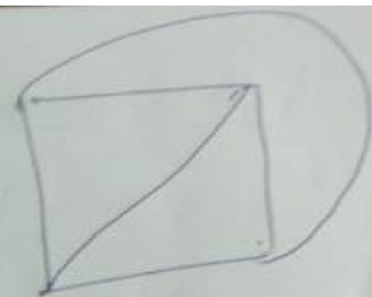
① connect the point by edge which is not already connected with edge

② remove connected edge, i.e. blue lines

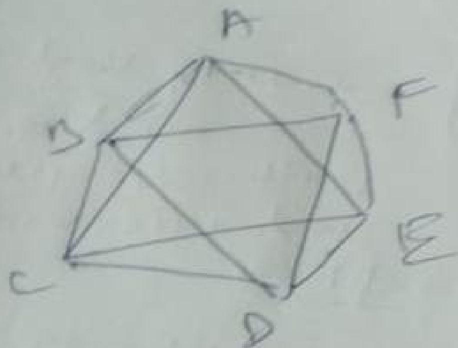


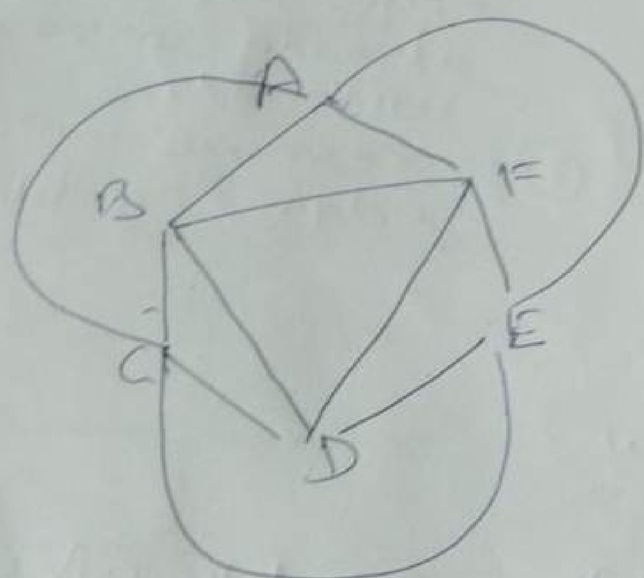
Planar Graph Graph which can be drawn in plane so that its edge do not cross is called planar Graph.





Ex



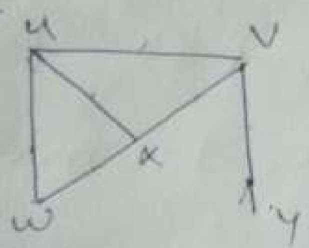


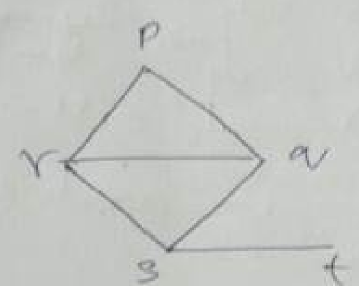
Isomorphism of Graph—

Let G_1 & G_2 are two Graph
it is Isomorphic if

(i) No. of vertices are same
(ii) No. of Edges " "
(iii) An equal no. of vertices with given degree
(iv) Vertex and Edge correspondence valid.

Ex





	G	G'	
① No of Vertex	5	5	same
② Edge	6	6	"
③ Degree	$u \Rightarrow 3$	$p \Rightarrow 2$	
	$v \Rightarrow 3$	$q \Rightarrow 3$	
	$w \Rightarrow 2$	$r \Rightarrow 3$	
	$x \Rightarrow 3$	$s \Rightarrow 3$	
	$y \Rightarrow 1$	$t \Rightarrow 1$	

(i) vertex correspondence

y — t ⇒ 1
w — p ⇒ 2
v — s ⇒ 3
u — a ⇒ 3
x — r ⇒ 3

(ii) Edge correspondence.

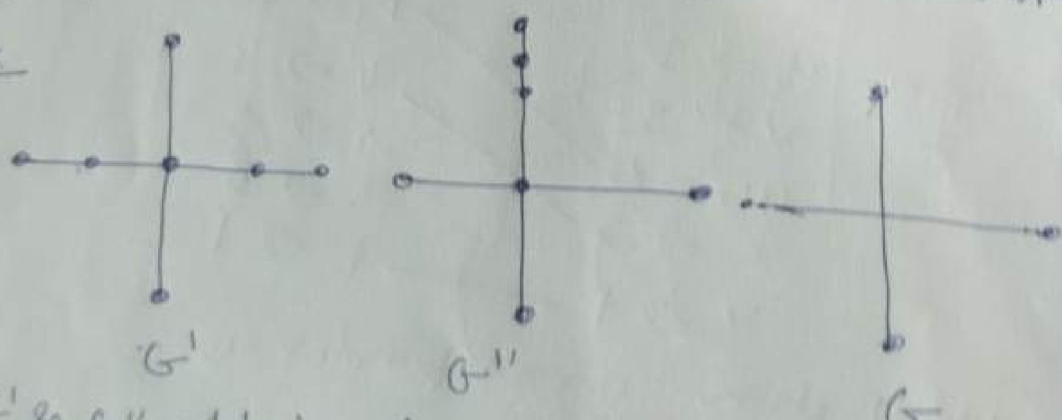
y — t, w — p	→	y — w / t — p ⇒ No edge
w — p, v — s	→	w — v / v — s ⇒ "
v — s, u — a	→	~~v — w~~ v — u / s — a ⇒ 1 "
u — a, x — r	→	u — x / a — r ⇒ 1 "

Hence G and G' are Isomorphic graph.

Homomorphic Graph—

G and G' are homomorphic if they can be obtained by the same graph.

Ex

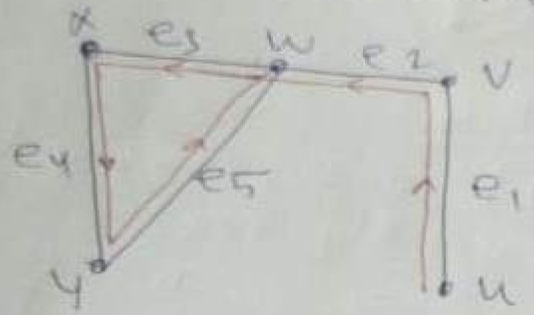


G'&G'' obtain from G, Hence it's homomorphic

Eulerian Path

A path, if it traverse each edge in the graph once and only once.

Ex

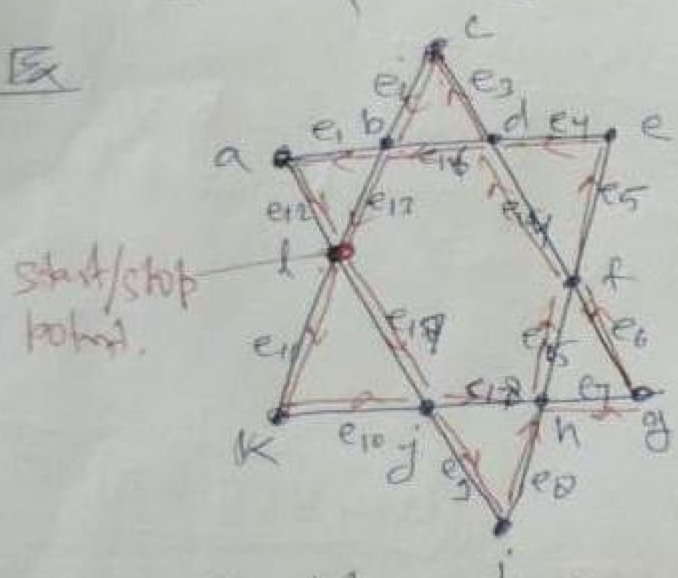


$\boxed{u e_1, v e_2, w e_3\, x e_4\, y e_5\, w}$

Eulerian circuit;

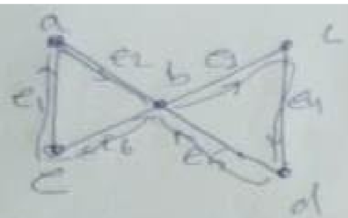
If it traverse each edge in the graph once & only once.

Ex



Eulerian Graph — A connected graph which contain an Eulerian circuit is called Eulerian Graph.

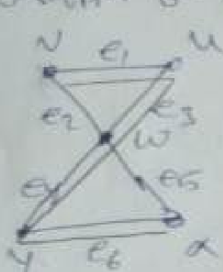
Ex

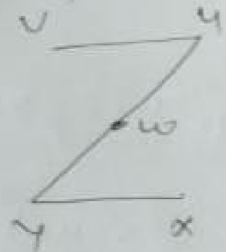


Hamiltonian Path

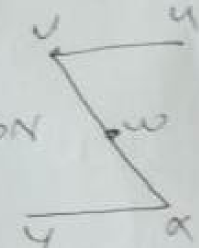
"A path which contain every vertex of a Graph G exactly once"

Ex





or

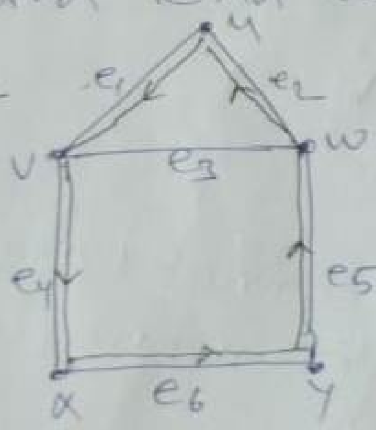


$\boxed{u e_1 v e_2, w e_5 x e_6 y}$

Hamiltonian circuit —

"A circuit, posses through each vertices in G exactly once except starting and end vertex."

Ex



$\boxed{u e_1 v e_4 x e_6 y e_5 w e_2 u}$

Note— Graph cover Hamiltonian circuit is Hamiltonian Graph.