Graph Theory

Problem can be solved by graph, very easily.

* **Graph ⇒**
A Graph $G=(V, E)$ is the collection of sets of vertices, $V=\{v_1, v_2, v_3, \dots\}$ & set of edges $E=\{e_1, e_2, \dots\}$

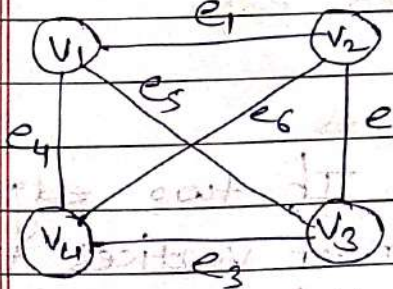
(OR)

A Graph is a structure defined by $G=(V, E)$ where V is the set of vertices $V=\{v_1, v_2, v_3, \dots\}$ & E is the set of edges $E=\{e_1, e_2, e_3, \dots\}$

* **Types of Graph ⇒** There are mainly two types of graph :- (1) Directed graph (2) Undirected graph

* **Undirected Graph ⇒** Unordered Pair ⇒ { }

Undirected graph G consists of set of vertices & set of edges $G=(V, E)$.
Undirected graph contains set of edges by ~~unordered~~ unordered pair of vertices.



All edges are undirected in this graph

$G=(V, E)$ Unordered Pair of vertices

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

$E=\{e_1, e_2, e_3, e_4, e_5, e_6\}$

$e_1=\{v_1, v_2\} / \{v_2, v_1\}$ Similarly

$e_2=\{v_2, v_3\} / \{v_3, v_2\}$

$e_3=\{v_3, v_4\} / \{v_4, v_3\}$, $e_4=\{v_1, v_4\} / \{v_4, v_1\}$

$e_5=\{v_1, v_3\} / \{v_3, v_1\}$, $e_6=\{v_2, v_4\} / \{v_4, v_2\}$

Note :- order of vertices set (Pair) are not fixed for edge.

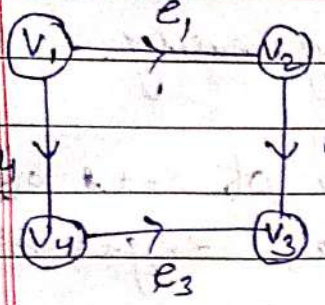
⇒ **Directed Graph ⇒** ordered Pair ⇒ ( )

Directed graph consists set of vertices & set of edges $G=(V, E)$

In directed graph set of edges contains the ordered pair of vertices means order is important.

All edges are directed in this graph.

$G=(V, E)$



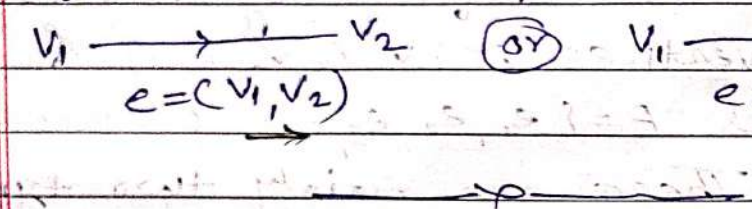
$V = \{V_1, V_2, V_3, \dots\}$ & $E = \{e_1, e_2, e_3, \dots\}$

$e = (V_1, V_2)$ $\quad e_2 = (V_2, V_3)$

$e_3 = (V_4, V_3)$ $\quad e_4 = (V_1, V_4)$

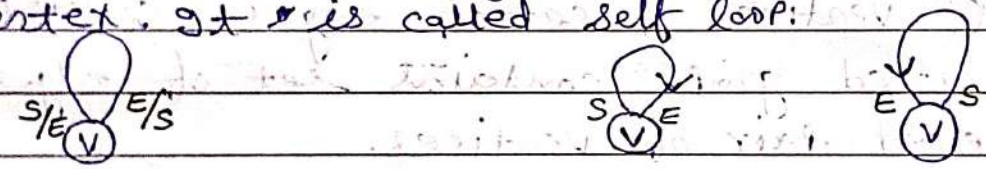
Here $(V_1, V_2) \neq (V_2, V_1)$ in ordered pair

If



* Special types of graph/cases/edges/vertex ⇒

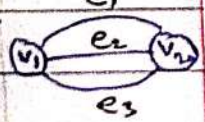
⇒ Self loop ⇒ If an edge start & end on the same vertex; it is called self loop:

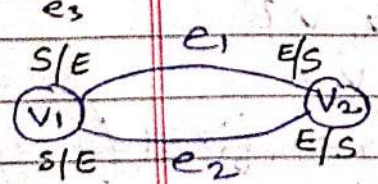


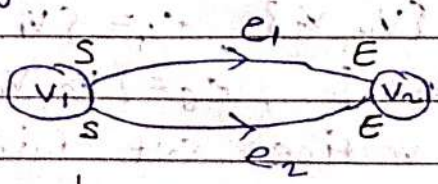
Undirected edge $\quad$ Directed edge

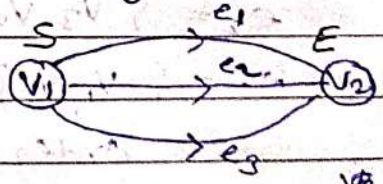
⇒ Parallel edges (Multiple edges) ⇒

If two edges start & end with the same pair of vertices then these two edges are called parallel edges.









$e_1 = \{V_1, V_2\} / \{V_2, V_1\}$

$e_2 = \{V_1, V_2\} / \{V_2, V_1\}$

$e_1 = (V_1, V_2)$

$e_2 = (V_1, V_2)$

$e_1 = (V_1, V_2)$

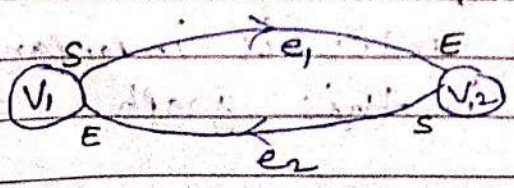
$e_2 = (V_1, V_2)$

$e_3 = (V_1, V_2)$

$e_1 \| e_2$

↑

Parallel



$e_1 \nparallel e_2$

not Parallel

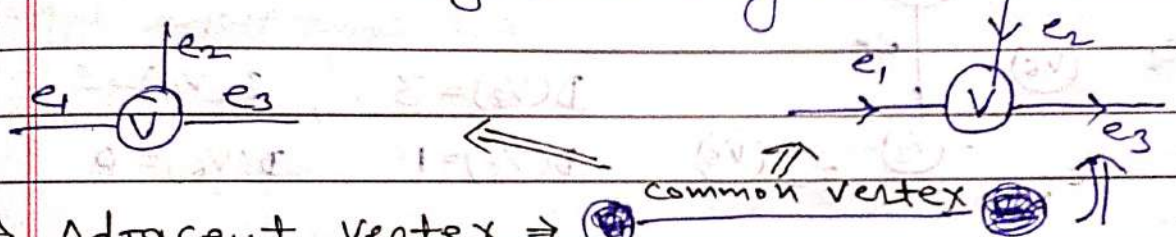
⇒ Order of graph = $|V(G)|$ = No. of vertices in graph

⇒ Size of graph = $|E(G)|$ = No. of edges in a graph

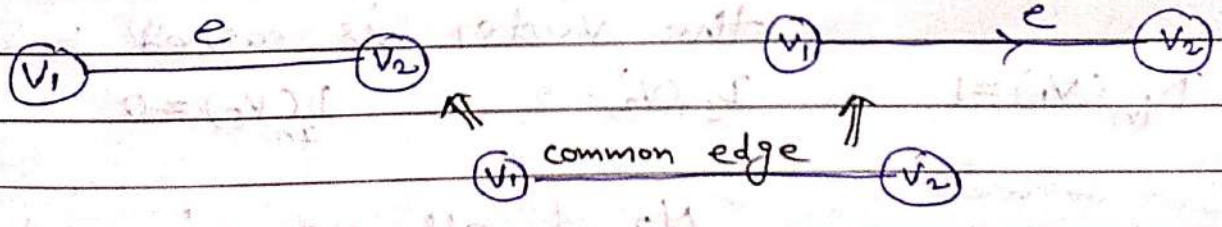
⇒ Incident / Incidence (or) Adjacent edge & vertex ⇒

⇒ Two or more edges are incident if they share a common vertex.

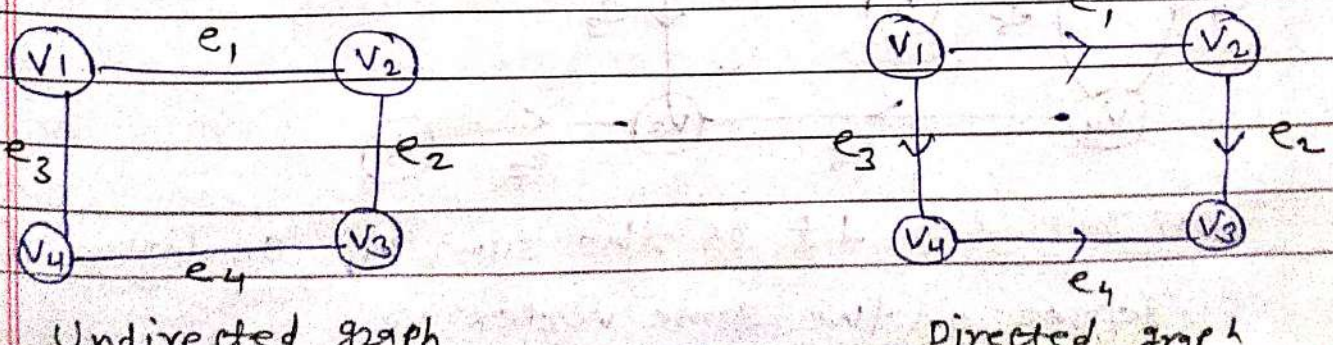
⇒ Adjacent edge ⇒ If two or more edges are incident on the same vertex they are called adjacent edges.



⇒ Adjacent Vertex ⇒ If two vertex are joined by the same edge they are called adjacent vertex.

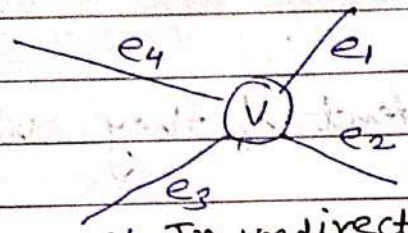


⇒ Reachability ⇒ In a diagram:- V_1 & V_2 or V_1 & V_4 are adjacent vertex but V_3 are reachable vertex for V_1 vertex because not direct path between V_1 & V_3 but we can reach on V_3 through Path



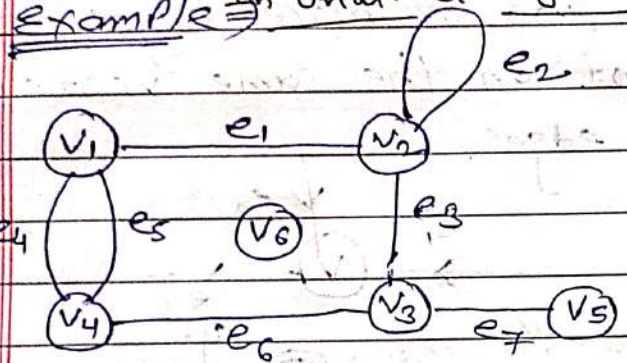
⇒ **Degree of vertex ⇒**

The number of edges are incident (adjacent) on a vertex V is called degree of vertex. It is denoted by D(V)



Degree of vertex V (or)
D(V) = 4 = no. of edges are connected to vertex V

Example ⇒ In undirected graph



Here D(V1) = 3
D(V2) = 2+1+1 = 4 {self loop count twice degree}
D(V3) = 3 , D(V4) = 3
D(V5) = 1 D(V6) = 0

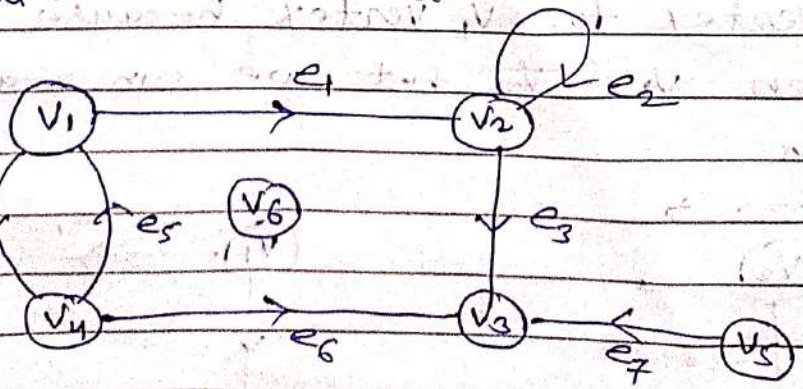
In directed graph ⇒ Degrees are two types In degree (D_{in}) & out degree (D_{out})

In degree (D_{in}) ⇒ No. of all incomming edges on the vertex is called In degree

$D_{in}(V_1) = 1$ $D_{in}(V_2) = 2$ $D_{in}(V_5) = 0$

Out degree (D_{out}) ⇒ No. of all out going edges from the vertex is called out degree

$D_{out}(V_1) = 2$ $D_{out}(V_2) = 2$ $D_{out}(V_5) = 1$



Total Degree ⇒ It is the sum of in degree & out degree on the same vertex

$$D_{Total}(V) = D_{In}(V) + D_{out}(V)$$

$D_T(V_1) = D_{In}(V_1) + D_{out}(V_1) = 1+2=3$

$D_T(V_2) = D_{In}(V_2) + D_{out}(V_2) = 2+2=4$

$D_T(V_5) = D_{In}(V_5) + D_{out}(V_5) = 0+1=1$

Similarly find out total degree of every vertex in a graph

⇒ No. of Indegree = No. of out degree in a graph

$D_{In}(V) = D_{out}(V)$ (or) = Total no. of edges in a directed graph

⇒ जीस vertex के पास

$D_{In}(V) = 0$ is called Source vertex

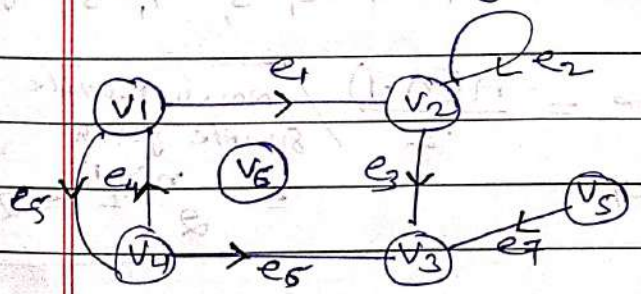
$D_{out}(V) = 0$ is called Sink vertex

⇒ **Isolated vertex** ⇒

A vertex having no incident edge (means no edges connected) is called isolated vertex

(OR)

A vertex having zero degree is called isolated vertex



Here vertex V_6 is a isolated vertex because $D(V_6)=0$

⇒ **Pendent vertex** ⇒ A vertex having Exactly one degree is called a Pendent vertex or End vertex.

In above digram (graph) vertex V_5 is a Pendent vertex because Degree of V_5 is $D(V_5)=1$

⇒ **Dead vertex** ⇒

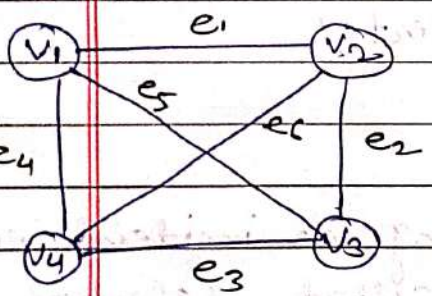
A vertex, we can not reach on this vertex (or) No reachability path available for this vertex is called dead vertex.

In previous given graph, vertex V_5 is a dead vertex { This type of vertex may be available in only directed graph not in undirected graph }

---

* **Simple Graph** ⇒

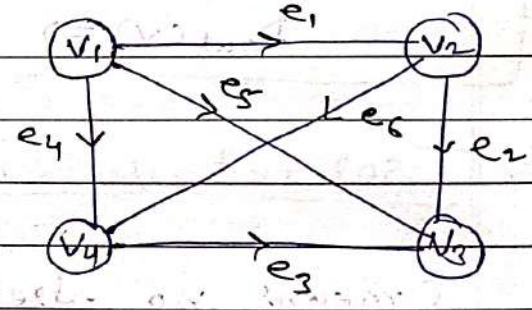
A Graph that has neither self loop nor parallel edges are called simple graph



$G = (V, E)$

$V = \{V_1, V_2, V_3, V_4\}$

$E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$



$G = (V, E)$

$V = \{V_1, V_2, V_3, V_4\}$

$E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$

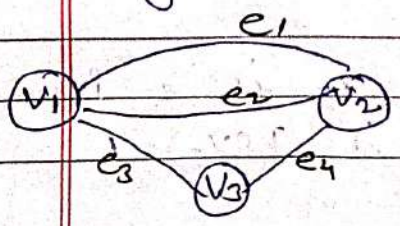
No. of simple graph possible $= \frac{n(n-1)}{2}$

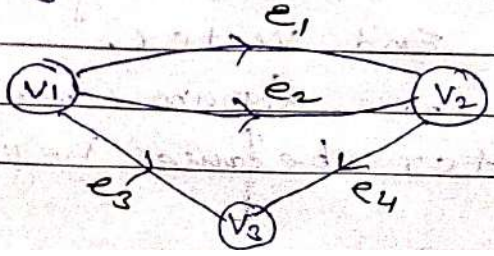
Here n is no. of vertex

---

* **Multi graph** ⇒

A Graph that have only paralled edges is called multi graph





GIRIRAJ

classmate Date Page

$G = (V, E)$

$V = \{v_1, v_2, v_3\}$

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

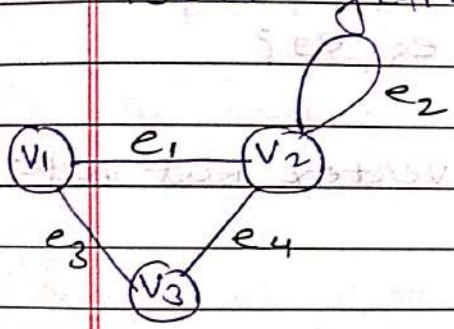
$G = (V, E)$

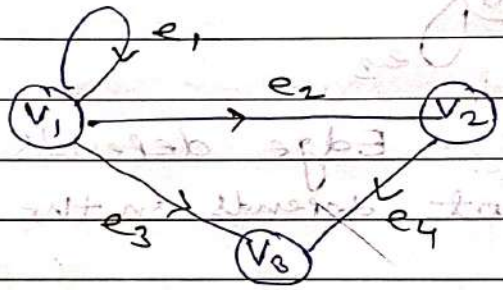
$V = \{v_1, v_2, v_3\}$

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

Pseudo graph (सूडो ग्राफ) ⇒

A Graph that have only self loop but not parallel edges is called Pseudo graph





$G = (V, E)$

$V = \{v_1, v_2, v_3\}$

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

$G = (V, E)$

$V = \{v_1, v_2, v_3\}$

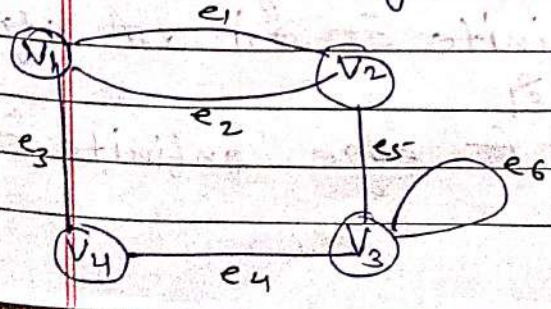
$E = \{e_1, e_2, e_3, e_4\}$

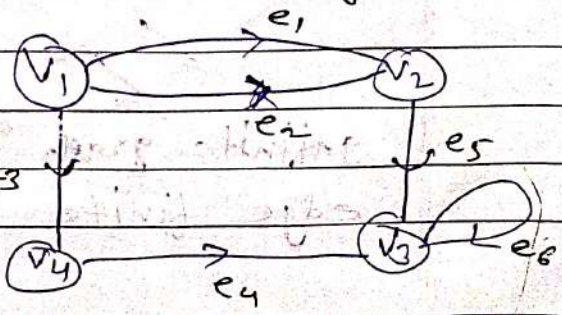
Summery table

Graph name	self loop edge	Parallel edge
Simple graph	X	X
General graph / Graph	✓	✓
Multi graph	X	✓
Pseudo graph	✓	X

General graph ⇒

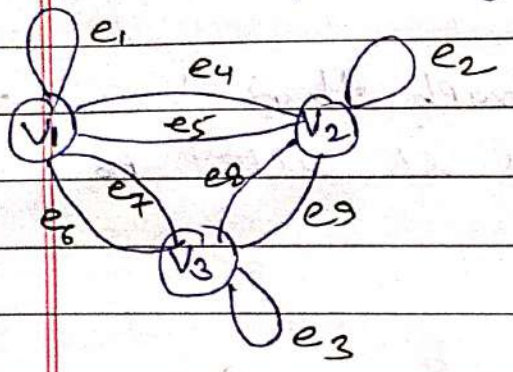
A Graph that have self loop & parallel edges both is called General graph





* Finite Graph ⇒

A Graph with a finite number of vertices as well as finite number of edges are called finite graph.

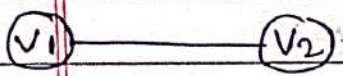


$G = (V, E)$

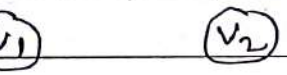
$V = \{ V_1, V_2, V_3 \}$

$E = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9 \}$

Note:- Edge depends on the vertex But vertex not depends on the edge.



Possible

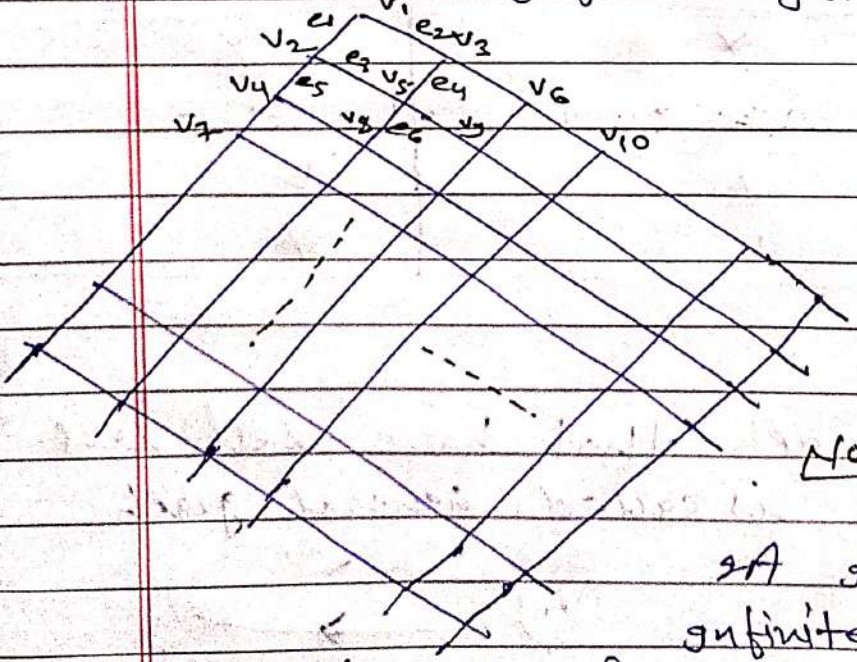


possible

not possible

* Infinite graph ⇒

A Graph with a infinite number of vertices as well as infinite number of edges are called infinite graph.



$G = (V, E)$

$V = \{ V_1, V_2, V_3, V_4 \dots \}$

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

Note:- infinite edges ⊛ होगी तो definitly no. of vertex ∞ infinite ~~है होगा~~ तभी edges infinite है

infinite graph में जरूरी नहीं कि edge infinite हो edge finite हो सकती है

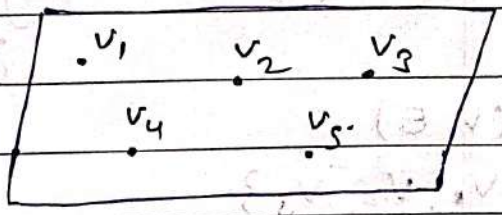
* **Null graph ⇒**

A Graph where vertex set is non-empty but edge set is empty is called Null graph

$G = (V, E)$

$V = \{v_1, v_2, v_3, v_4, v_5\}$

$E = \{\} = \phi$



Note:- Every vertex in a null graph is an isolated vertex.

In Null graph, vertex set must not be empty ~~but~~ otherwise graph is not possible.

* **Trivial (~~में~~ ~~त्रिविवल ग्राफ~~) ⇒**

It is also called smallest graph. A Graph where vertex set contains only one vertex & edge set is empty is called Trivial graph.

(v) or v

$G = (V, E)$

$V = \{v\}$ & $E = \{\} = \phi$

* **Complete graph / Universal graph / Full graph ⇒**

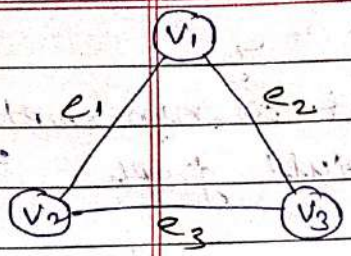
A Graph G is said to be complete graph if every vertex in graph G is connected to other vertex in graph.

(OR)

A Graph is said to be complete where there is an edge between every pair of vertex. It is denoted by K_n (Here n is no. of vertex in a graph) If no. of vertex is n then degree of every vertex is $(n-1)$

K_3

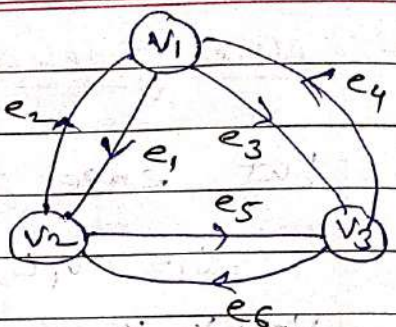
Undirected graph



In both graphs without self loop & without Parallel edges (in simple graph domain)

K_3

Directed graph



$G = (V, E)$

$V = \{V_1, V_2, V_3\}$

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

$G = (V, E)$

$V = \{V_1, V_2, V_3\}$

$E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$

No. of edges ⇒ Total number of edges in a complete graph [where n is the no. of vertex in graph]

In Undirected graph ⇒ No. of edges $= \frac{n(n-1)}{2}$

Example :- $n = 3$ No. of edges $= \frac{3(3-1)}{2} = \frac{3 \times 2}{2} = 3$

In directed graph ⇒ No. of edges $= n(n-1)$

Example :- $n = 3$ No. of edges $= 3(3-1) = 3 \times 2 = 6$

Note :- In undirected graph

No. of vertex = n & every vertex (n-1) vertex से connected होता है but each vertex counted twice edge so No. of edge $= \frac{n(n-1)}{2}$

* **Regular graph** ⇒

A Graph G is called a regular graph if every vertex having same degree.

(or)

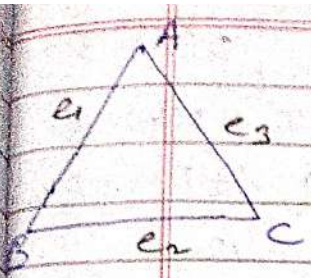
If degree of every vertex is K then graph is called K-regular graph.

Note :- If K = 2 then 2 regular graph called or Auto regular graph.

No. of vertices in a regular graph $= (n * K)/2$

Here n = no. of vertex

K = degree of vertex



$G = (V, E)$

$V = \{A, B, C\}$

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

Degree $D(A) = 2$

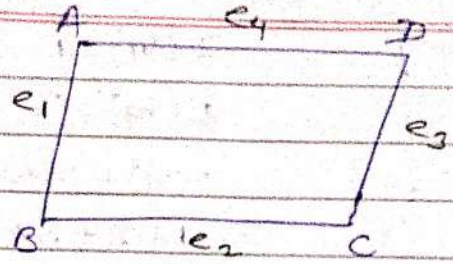
$D(B) = 2$

$D(C) = 2$

$\therefore$

$D(A) = D(B) = D(C)$

Having same degree



$G = (V, E)$

$V = \{A, B, C, D\}$

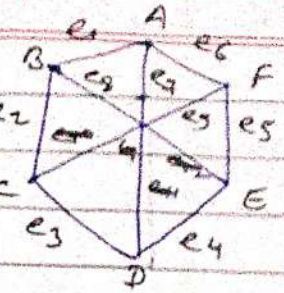
$E = \{e_1, e_2, e_3, e_4\}$

$D(A) = 2, \quad D(B) = 2$

$D(C) = 2 \quad D(D) = 2$

$D(A) = D(B) = D(C) = D(D)$

Having same degree



$G = (V, E)$

$V = \{A, B, C, D, E, F\}$

$E = \{e_1, e_2, e_3 \cdots e_9\}$

$D(A) = 3, \quad D(B) = 3$

$D(C) = 3, \quad D(D) = 3$

$D(E) = 3 \quad D(F) = 3$

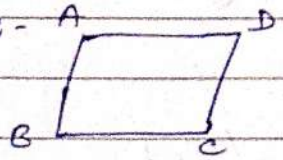
$D(A) = D(B) = D(C) = D(D)$

$D(E) = D(F)$

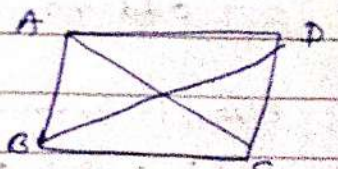
Having same degree

Note:- Regular & complete graph में every vertex की degree same होती है but in complete graph में every vertex connect to each vertex (each other)

Example:-



Regular graph



complete graph

---

* **Sum of Degree theorem [Hand-Shaking theorem] ⇒**

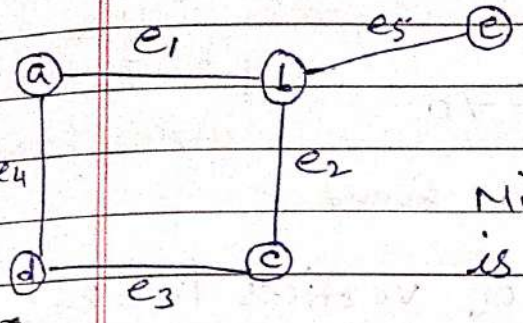
① $\sum_{i=1}^{n} D(V_i) = 2|E|$ Here D for degree & $|E|$ for no. of edges

② $\sum_{i=1}^{n} D(V_i) = \sum_{\text{Even}} D(V_j) + \sum_{\text{odd}} D(V_k)$

Note:- odd degree vertex in a graph always is even in counting.

* Minimum degree of a graph ⇒

It is denoted by $\delta(G)$. It is the minimum degree of it's vertices



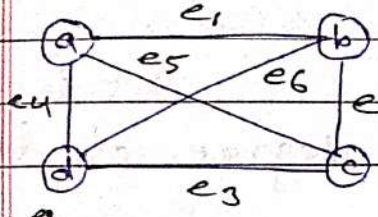
$D(a) = 2$, $D(b) = 3$, $D(c) = 2$, $D(d) = 2$

$D(e) = 1$

Minimum smallest degree of a graph G is 1 $\{D(e) = 1\}$.

$$\boxed{\delta(G) = 1}$$

K_4 - Complete graph ⇒



$D(a) = 3$, $D(b) = 3$, $D(c) = 3$, $D(d) = 3$

$\delta(G) = 3$

Degree sum of a graph is

$$\sum_{i=1}^{n} D(V_i) = 2|E| \implies |V| = 2|E|$$

$|V|$ ↑ Total no. of vertex; $|E|$ ↑ Total no. of edges

$$\boxed{\delta(G) \cdot |V| \leq 2|E|}$$

Example ⇒ $\delta(G) = 3$, $|V| = 4$, $|E| = 6$

$\delta(G) \cdot |V| \leq 2|E| \implies 3 \times 4 \leq 2 \times 6 \implies 12 \leq 12$

Example ⇒ $\delta(G) = 1$, $|V| = 5$, $|E| = 5$

$\delta(G) \cdot |V| \leq 2|E| \implies 1 \times 5 \leq 2 \times 5 \implies 5 \leq 10$

Note :- If $\delta(G) \cdot |V| = 2|E|$ means all degree of all the vertices are same in given graph.

If $\delta(G) \cdot |V| < 2|E|$ means At least (कम से कम) one vertex having minimum degree in given graph.

$$\implies \delta(G) \cdot |V| \leq 2|E| \leq \Delta(G) \cdot |V|$$

$\delta(G)$ Average degree $= \dfrac{2|E|}{|V|}$

Example

$\dfrac{2 \times 7}{6} = 2.33$

Minimum degree = 2 & Maximum degree = 3

$$\delta(G)\cdot|V| \le 2|E| \le \Delta(G)\cdot|V|$$

$\delta(G)$: Minimum degree; $2|E|$: Two times No. of edges; $\Delta(G)$: Maximum degree

Q.1) Maximum No. of vertices in a simple graph with 35 edges & degree of each vertex is at least three

Sol^{n} Given $|E|=35$ $\quad \delta(G)=3$

we know that $\delta(G)\cdot|V| \le 2|E|$

$3\times|V| \le 2\times 35 \Rightarrow 3|V| \le 70$

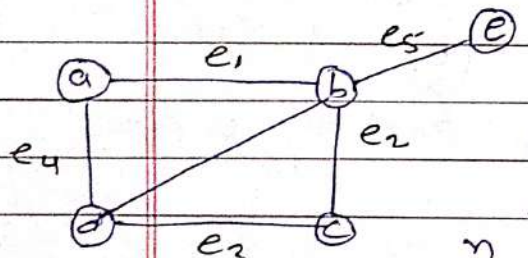
$\lceil 23.33 \rceil = 24$ upper bound

$|V| = \frac{70}{3} = \lfloor 23.33 \rfloor$ ← lower bound

$\therefore |V| = 23$ $\quad \therefore$ No. of vertices $|V|=23$

---

* **Maximum degree of a Graph $\Delta(G)$ ⇒** It is denoted by $\Delta(G)$. It is the maximum degree of it's vertices



$D(a)=2$ $\quad D(b)=4$ $\quad D(c)=2$

$D(d)=3$ $\quad D(e)=1$

Maximum degree of graph $\Delta(G)=4$

$$\sum_{i=1}^{n} D(V_i) = 2\cdot|E| \Rightarrow |V| = 2|E|$$

$$\boxed{\Delta(G)\cdot|V| \ge 2|E|}$$

Note ⇒ If $\Delta(G)\cdot|V| = 2|E|$ means degree of every Vertex are same

$\Delta(G)\cdot|V| > 2|E|$ means at least (कम से कम) one vertex having maximum degree.

Q.1) Minimum no. of vertices possible in a simple graph with 41 edges & degree of each vertex is at most 5

Sol^{n} Given $\Delta(G)=5$ $\quad |E|=41$

$2|E| \le \Delta(G)\cdot|V| \Rightarrow 2\times 41 \le 5|V| \Rightarrow 82 \le 5|V|$

$|V| \ge \frac{82}{5} = \lceil 16.4 \rceil = 17$

$\lceil 16.4 \rceil$ = upper bound = 17

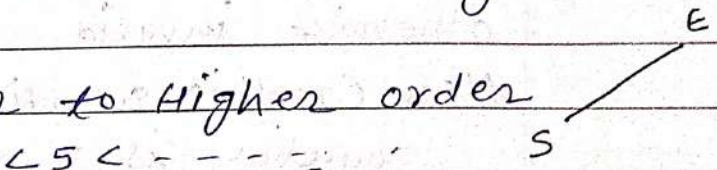
No. of vertex $|V|=17$

* Degree sequence of graph ⇒

The arrangement of degree of all vertices in a graph G either in non-increasing order or non-decreasing order.

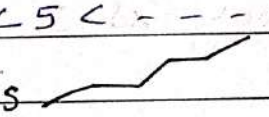
(Ascending) Increasing order ⇒ Lower to Higher order

Example ⇒ $1 < 2 < 3 < 4 < 5 < \cdots$



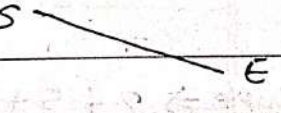
Non-decreasing order ⇒

Example ⇒ $1 < 2 < 3 \leq 3 < 4 < 5 \leq 5 < 6 < \cdots$



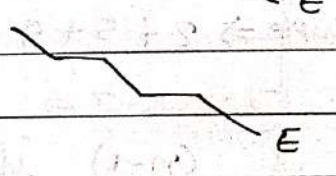
(Descending) (De-creasing order) ⇒ Higher to lower order

Example ⇒ $6 > 5 > 4 > 3 > 2 > 1$



NON-increasing order ⇒

Example ⇒ $6 > 5 \geq 5 > 4 > 3 \geq 3 > 2 > 1$

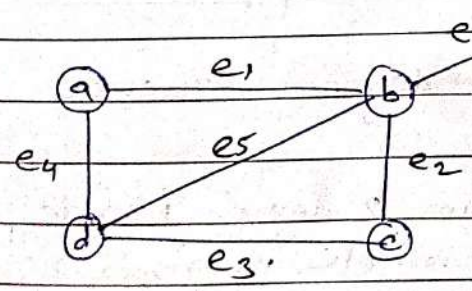


Note ⇒

Generally High Value (Non-increasing order) to Lower value (Non-decreasing order) Arrangement

Example ⇒ Degree Sequence = {4, 3, 2, 2, 1} It is non-increasing order

For degree sequence more than one simple graph possible means multiple simple graph.



D(a) = 2, D(b) = 4, D(c) = 2

D(d) = 3 D(e) = 1

∴ degree sequence = {4, 3, 2, 2, 1}

Q.5) {2, 3, 3, 3, 3} degree sequence valid / invalid ?

Soln → ~~Maximum~~ No. of vertex = 5

Maximum degree possible = n-1 = 5-1 = 4

No. of odd degree vertex = 3, 3, 3, 3 = even

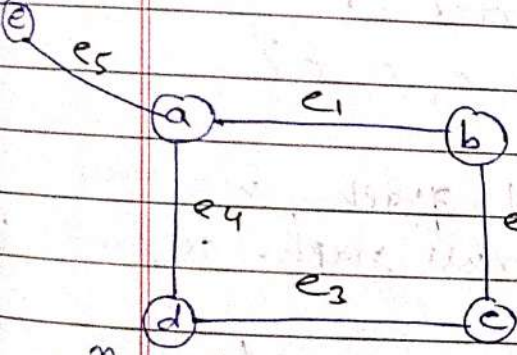
Sum of all degree is = 2+3+3+3+3 = 14 = even

इस questions में maximum all tricks apply हो रही हैं फिर भी 100% confidance के साथ यह नहीं कह सकते हैं की given degree sequence valid / invalid

इस type की problem को solve (100%) करने के लिये इस theorem (Hevel-Hakimi) का use किया जाता है

---

* Hevel-Hakimi Theorem ⇒



D(a) = 3, D(b) = 2, D(c) = 2

D(d) = 2 D(e) = 1

Example ⇒

Degree sequence = {4, 3, 2, 2, 1}

Soln Steps ⇒

(1) सबसे पहले दिये गये degree sequence को Non-increasing (Decreasing / Descending) order में arrange करेंगे

{4, 3, 2, 2, 1}

(2) Higher degree वाले vertex को remove कर देंगे

{~~4~~, 3, ~~2~~, 2, 1} ⇒ इसके बाद all vertex की degree को one से subtract (minus) करेंगे ⇒ {2, 1, 1, 0}

Note ⇒ जितनी degree को remove करेंगे उसके बाद की आने वाली उतनी degree को 1 से minus करते हैं

(3) फिर Again Higher degree को remove करेंगे {~~2~~, 1, 1, 0}

फिर one से Minus करेंगे

Q.1) Degree Sequence {7,6,5,4,4,2,1,3} is valid or not

Sol^n

Arrange in non-increasing order ⇒ {7,6,5,4,4,3,2,1}

Remove Highest degree vertex (node) means = 7

{(7) 6,5,4,4,3,2,1} subtract 1 after removing 7 count

Highest degree = {5,4,3,3,2,1,0}

Again remove Highest degree vertex (node) = 5

{(5) 4,3,3,2,1,0} —Subtract 1→ {3,2,2,1,0,0}

Again remove Highest degree vertex (node) = 3

{(3) 2,2,1,0,0} —Subtract 1→ {1,1,0,0,0}

Again remove Highest degree vertex = 1

{(1) 1,0,0,0} —Subtract 1→ {0,0,0,0}

It is Possible because Null graph, we know that All vertex degree zero in null graph. so given degree sequence are valid

Q.2) Degree sequence {6,6,6,6,3,3,2,2} valid / invalid

Sol^n

{6,6,6,6,3,3,2,2} Arrange in non-increasing order

{(6) 6,6,6,3,3,2,2} —Minus 1→ {5,5,5,2,2,1,2}

Re-arrange ⇒ {5,5,5,2,2,2,1}

{(5) 5,5,2,2,2,1} —Subtract 1→ {4,4,1,1,1,1}

{(4) 4,1,1,1,1} —Subtract 1→ {3,0,0,0,1}

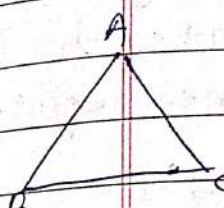
Re-arrange ⇒ {3,1,0,0,0}

{(3) 1,0,0,0} —Minus 1→ {0,-1,-1,0}

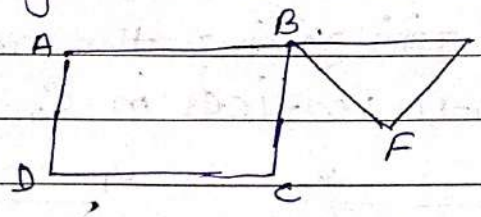
Minus degree is not possible in graph so given degree sequence is invalid.

* Even graph ⇒

If a graph, All vertices having even degree is called even graph

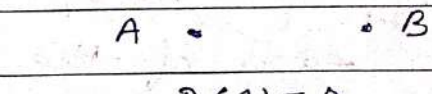


D(A) = 2
D(B) = 2
D(C) = 2



D(A) = 2, D(B) = 4
D(C) = 2, D(D) = 2
D(E) = 2, D(F) = 2

connected graph



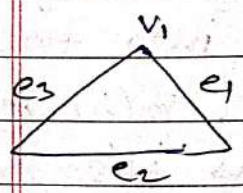
D(A) = 0, D(B) = 0

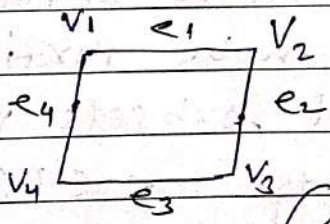
Dis-connected graph

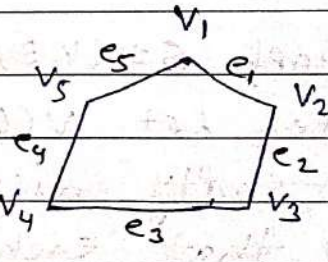
* Cycle graph ⇒

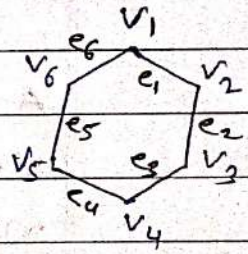
A Graph consists of n vertices ($n \geq 3$) $v_1, v_2, v_3, \ldots v_n$ and edges $(v_1, v_2), (v_2, v_3), (v_3, v_4), \ldots$ (v_n, v_1) is called a cycle graph. It is denoted by C_n

Example ⇒









Note ⇒ single vertex (v) e is self loop, not a cycle graph

Two vertex (V1) e1, e2 (V2) Multigraph because Parallel edges.

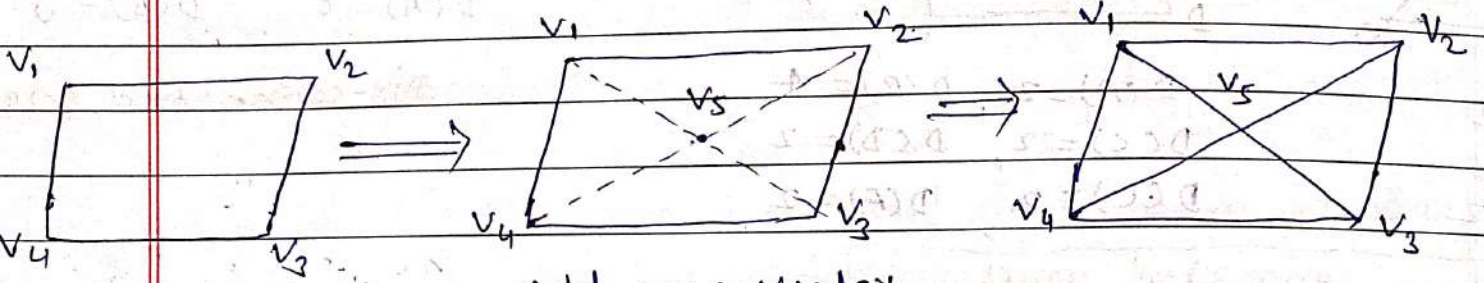
∴ Therefore minimum (at least) three vertex required for cycle graph.

Note :- जब starting vertex से continue traversing करते हुए उसी vertex पर reach करते है तब self loop & Parallel edges allow नही होगी

Note :- No. of vertex = No. of edges

* Wheel graph ⇒

A wheel graph W_n is obtained by introducing an additional vertex to the cycle graph C_{n-1} {$\because n \geq 3$} & joining the new vertex to each of the $(n-1)$ vertices in C_{n-1} by new edges.



cycle graph — Add new vertex V_5 in cycle graph & Join with edges from remaining vertices — Wheel graph W_5

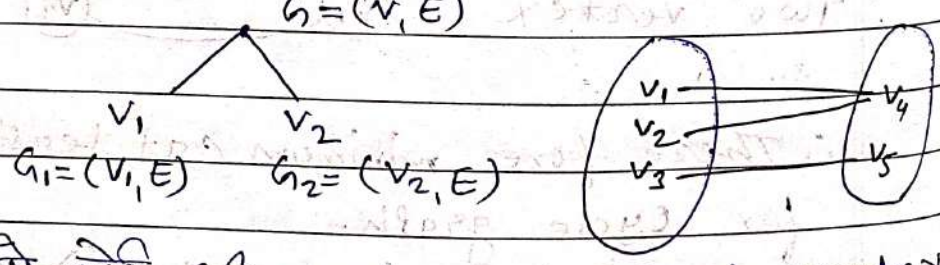
बाई पार्टाइट ग्राफ

* Bi-Partite graph ⇒

m = First partition se
n = second partition s

It is denoted by $K_{m,n}$

A graph $G = (V, E)$ is called Bi-Partite graph if it's vertex set $V(G)$ can be Partitioned into two non-empty disjoint subsets $V_1(G)$ & $V_2(G)$ in such way that each edge $e \in E(G)$ has it's one end point in $V_1(G)$ & other end point in $V_2(G)$. The Partition $V = V_1 \cup V_2$ called ~~Bi-Partition~~ Bi-Partition of graph G.



Bi-Partite graph में कोई भी vertex adjustent vertex नहीं होगा. In same group (Partition) में

Bi-Partite graph ~~can~~ represented by $K_{m,n}$

In Bi-Partite graph, if $K_{1,n}$ then these Bi-Partite graph is called ~~star~~ graph

(2212)

Example-I ⇒ $G = (V, E)$

$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ & $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$

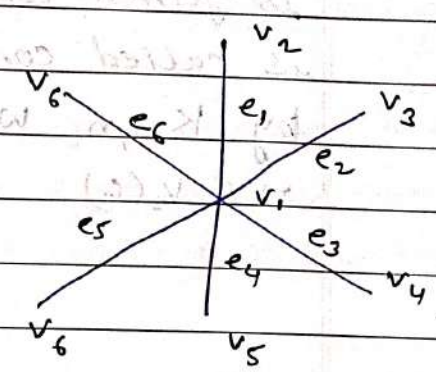
If Bi-Partitions of partite graph

$V_1' = \{v_1\} \Longrightarrow V_1'(G)$

$V \rightarrow V_1' = \{v_1\}$

$V_2' = \{v_2, v_3, v_4, v_5, v_6, v_7\}$

$V_2' = V_2'(G)$

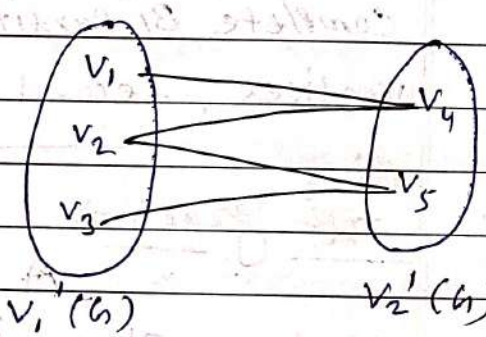


Stere graph ⇒ $K_{1,6}$

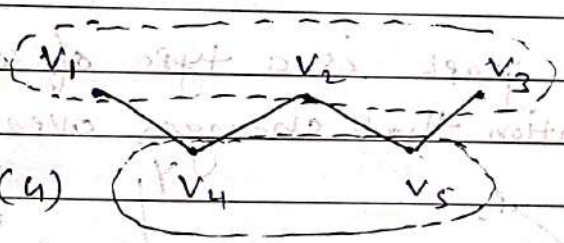
Example-II ⇒ $V = \{v_1, v_2, v_3, v_4, v_5\}$

$V \rightarrow V_1'(G) = V_1' = \{v_1, v_2, v_3\}$

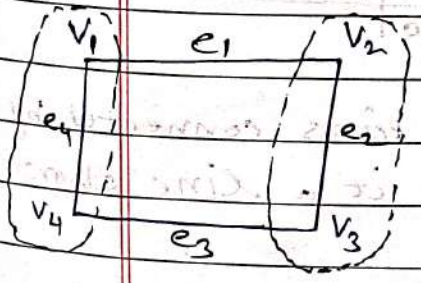
$V_2'(G) = V_2' = \{v_4, v_5\}$



$K_{3,2}$



Example-III ⇒

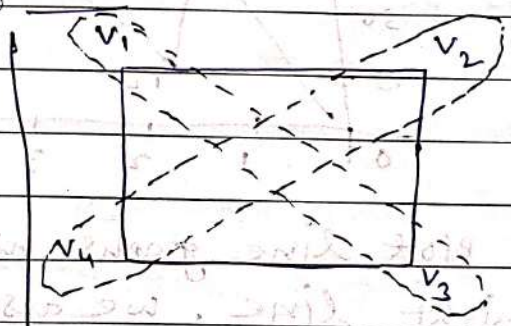


$V = \{v_1, v_2, v_3, v_4\}$

$V_1' = \{v_1, v_4\}$

$V_2' = \{v_2, v_3\}$

It is not partition possible because v_1 & v_4 edge (e_4) are connect but in the same group means it's adjacent vertex & edge similarly v_2 & v_3 (e_2 edge) edges & vertex. We know that adjacent vertex & edge not possible in bi-partite graph



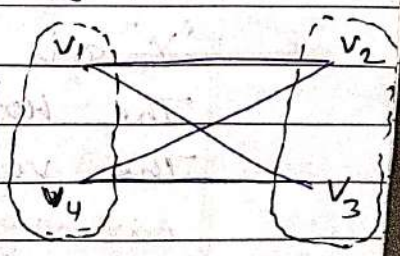
$V = \{v_1, v_2, v_3, v_4\}$

$V_1 = V_1'(G)$

$= \{v_1, v_3\}$

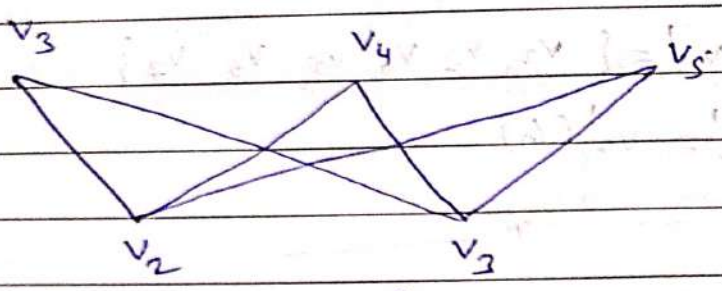
$V_2'(G) = V_2' = \{v_2, v_4\}$

It is Partition possible so it is Partitite/Partition graph



These is not adjancent vertex & edge in same group so it's partition graph.

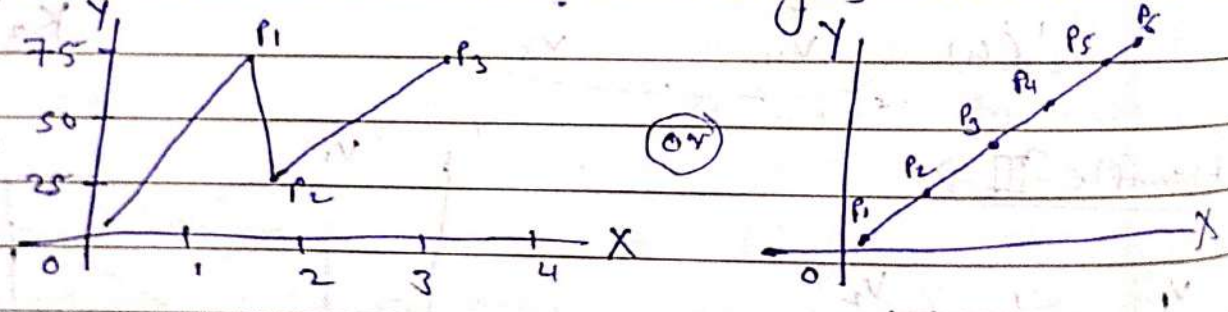
* **Complete Bi-Partite graph ⇒** An bi-partite graph. If each vertex of $V_1(G)$ is joined with each vertex $V_2(G)$ then the graph G is called complete Bi-Partited graph & it is denoted by $K_{m,n}$ where m is no. of vertices in $V_1(G)$ similarly $n \in V_2(G)$



Complete Bi-Partited graph में First Partition के m vertices, Second Partition के n vertices से connect होते हैं।

* **Line graph ⇒**

A Line graph is a type of chart used to show information that changes over time.



We plot line graphs using several points connected by straight line. we also called to it a line chart.

The line graph compromises of two axes known as "X" axis & "Y" axis

The Horizontal axis is known as the X-axis.

The vertical axis is known as the Y-axis.

Advantages (Benefits) →

(1) Easy Plotting

(2) Easy reading

Line graph Definition ⇒ A line graph displays data that changes continuously over periods of time.

* **Chordal graph** ⇒ (or definition)

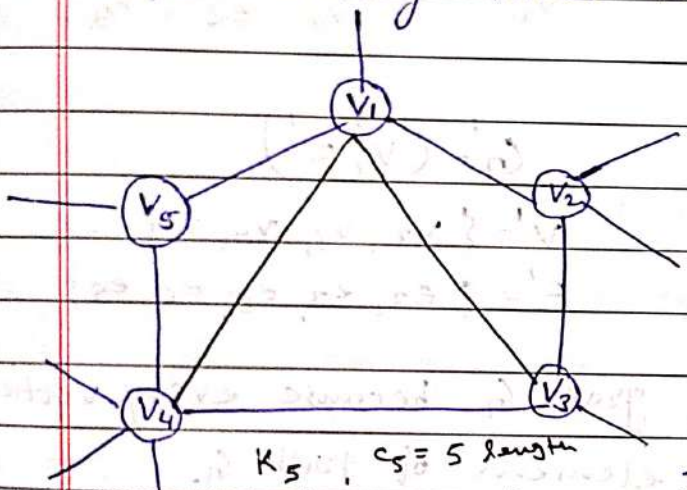
Ex:- every complete graph & every tree is chordal graph

A chordal graph is a simple graph in which every graph cycle of length four & greater has a chord.

(or)

A chordal graph is one in which all cycles of four or more vertices have a chord, which is an edge that is not part of the cycle but connects two vertices of the cycle.

Equivalently, every induced cycle in the graph should have exactly three vertices.



K_5 , C_5 = 5 length

chord C_5 = {V_1V_3, V_1V_4}

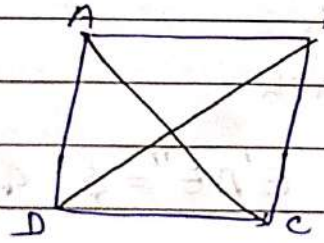
C_4 = [V_1V_4], C_4 = [V_1V_3]

A cycle (block) with two chords (black line). As for this part, The graph is chordal.

However, removing one black edge would result in a non-chordal graph.

Indeed, the other black line edge with three blue edges would form a cycle of length four with no chords.

Example ⇒

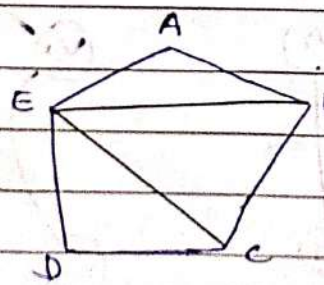


K_4 (chordal graph with 4 vertices)

cycle $C_4 = A \xrightarrow{1} B \xrightarrow{2} C \xrightarrow{3} D \xrightarrow{4} A$ = 4 length

chord of C_4 = {BD, AC}

Example ⇒



K_5 (chordal graph with 5 vertices)

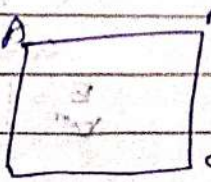
cycle $C_5 = A \xrightarrow{1} B \xrightarrow{2} C \xrightarrow{3} D \xrightarrow{4} E \xrightarrow{5} A$ = 5 length

chords C_5 = ABCDEA = {EB, EC}

chords C_4 = ABCEA = {EB}

chords C_4 = BCDEB = {EC}

Example ⇒

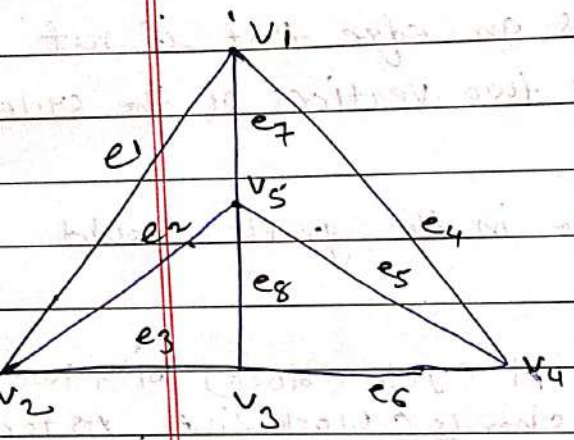


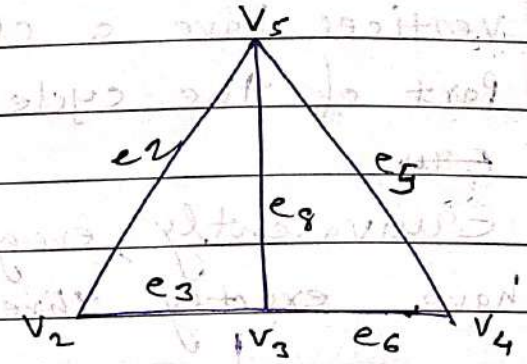
K_4 but not chordal graph because No chordal cycle is C_4 = 4 length but not chordal graph

Note ⇒ Every induced subgraph of a chordal graph is chordal.

* Sub graph ⇒

A graph G' is said to be a subgraph of a graph G if all the vertices & edges of graph G' are in graph G, such that each edge of G' has the same end vertices as in graph G.





$G = (V, E)$

$V = \{v_1, v_2, v_3, v_4, v_5\}$

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$

$G' = (V', E')$

$V' = \{v_2, v_3, v_4, v_5\}$

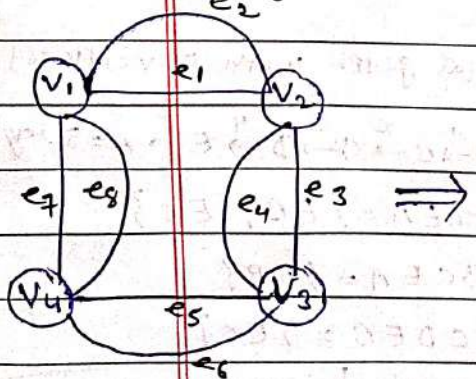
$E' = \{e_2, e_3, e_5, e_6, e_8\}$

So graph G' is a subgraph of graph G because every vertices & edges of sub graph G' is element of graph G.

* Types of sub graph ⇒ There are mainly two types

- Edge-disjoint sub graph
- Vertex-disjoint sub graph

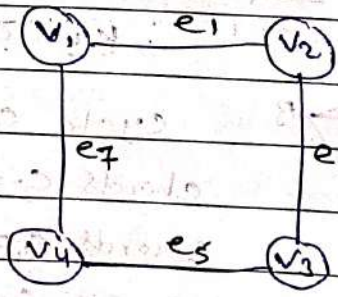
⇒ Edge-disjoint sub graph ⇒ $E' \cap E'' = \phi$ & $V' \cap V'' \neq \phi$



$G = (V, E)$

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

$E = \{e_1, e_2, e_3, \ldots e_8\}$

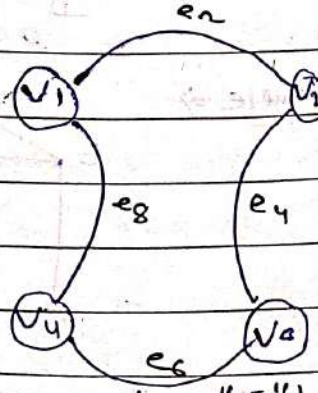


$G' = (V', E')$

$V' = \{v_1, v_2, v_3, v_4\}$

$E' = \{e_1, e_3, e_5, e_7\}$

$\because E' \cap E'' = \phi$ but $V' \cap V'' \neq \phi$



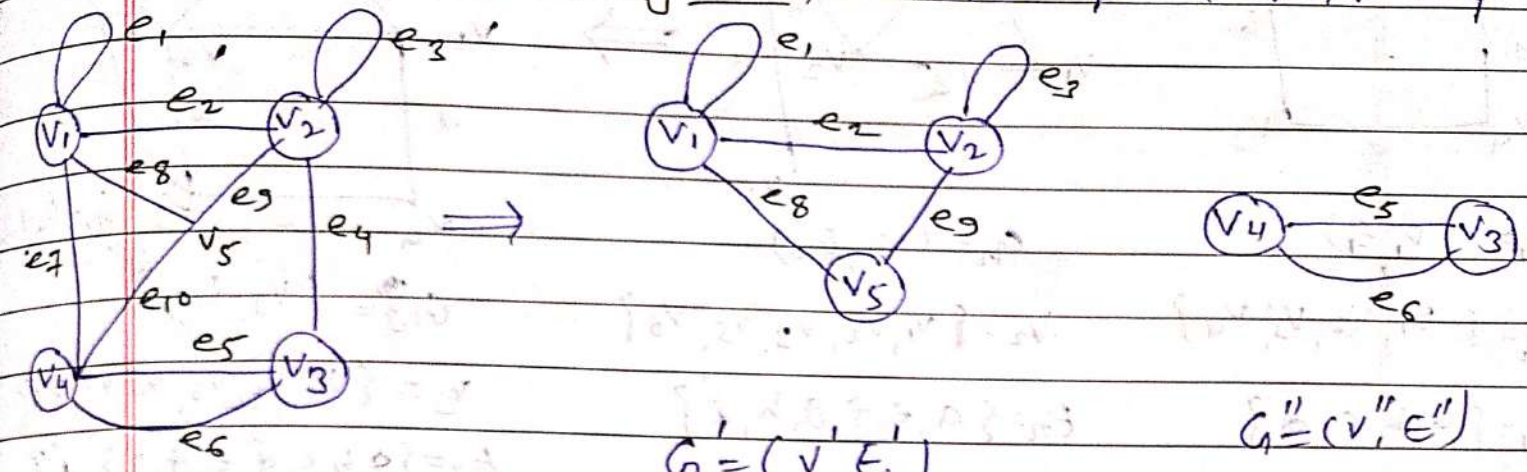
$G'' = (V'', E'')$

$E'' = \{e_2, e_4, e_6, e_8\}$

$V'' = \{v_1, v_2, v_3, v_4\}$

So given sub graph G' & G'' are edge-disjoint sub graph

⇒ Vertex-disjoint sub graph ⇒ $E' \cap E'' = \phi$ & $V' \cap V'' = \phi$



$G = (V, E)$

$V = \{v_1, v_2, v_3, v_4, v_5\}$

$E = \{e_1, e_2, e_3 \cdots e_9\}$

$G' = (V', E')$

$V' = \{v_1, v_2, v_5\}$

$E' = \{e_1, e_2, e_3, e_8, e_9\}$

$G'' = (V'', E'')$

$V'' = \{v_3, v_4\}$

$E'' = \{e_5, e_6\}$

$\because E' \cap E'' = \{e_1, e_2, e_3, e_8, e_9\} \cap \{e_5, e_6\} = \phi$

&

$V' \cap V'' = \{v_1, v_2, v_5\} \cap \{v_3, v_4\} = \phi$

So given subgraph are vertex-dis-joint sub graph.

---

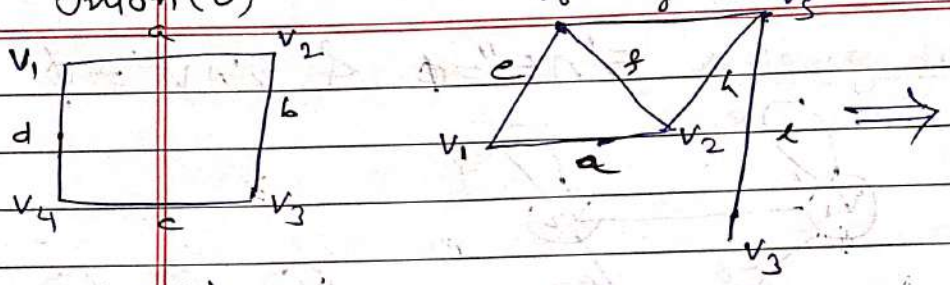
* Operations on graph ⇒ There are following types of operations

- Union (∪) & Intersection (∩)
- Ring sum of two graph (⊕)
- De-compositions of graph
- Deletion
- fusion

⇒ Union (∪) & Intersection (∩) ⇒ Let $G_1 = (V_1, E_1)$ & $G_2 = (V_2, E_2)$ then $G_3 = G_1 \cup G_2$ such that $V = V_1 \cup V_2$ & $E = E_1 \cup E_2$

Similarly $G_3 = G_1 \cap G_2$ $\because V_3 = V_1 \cap V_2$ & $E_3 = E_1 \cap E_2$

Union (∪)



$G_1 = (V_1, E_1)$ $G_2 = (V_2, E_2)$ $G_3 = G_1 \cup G_2$

$V_1 = \{v_1, v_2, v_3, v_4\}$ $V_2 = \{v_1, v_2, v_3, v_5, v_6\}$ $G_3 = (V_3, E_3)$

$E_1 = \{a, b, c, d\}$ $E_2 = \{a, e, f, g, h, i\}$ $V_3 = \{v_1, v_2, v_3, v_4, v_5, v_6\}$

$E_3 = \{a, b, c, d, e, f, g, h, i\}$

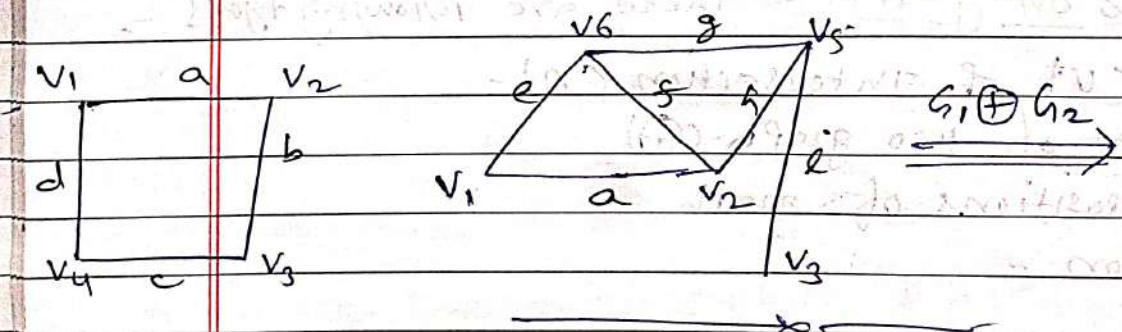
Intersection (∩) ⇒ Use above graph G_1 & G_2 then $G_3 = G_1 \cap G_2$

$V_3 = V_1 \cap V_2$ & $E_3 = E_1 \cap E_2$

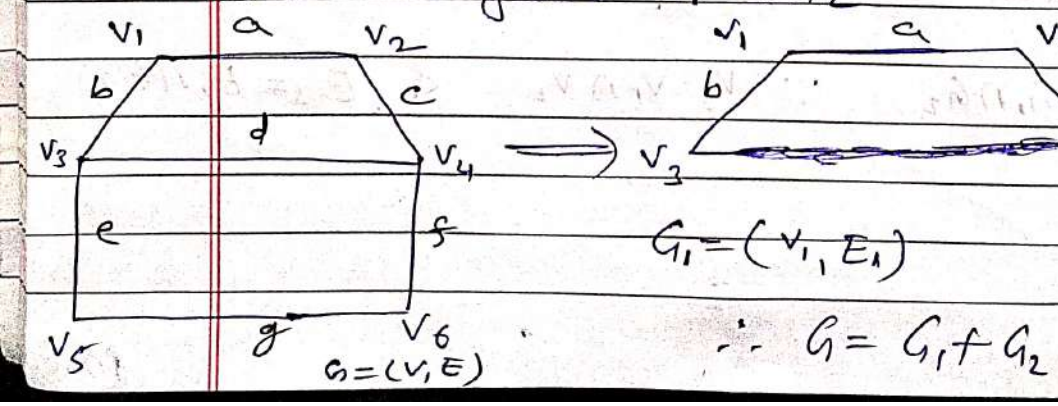
so $G_3 = G_1 \cap G_2 \Rightarrow$

⇒ Ring sum of two graph [⊕] ⇒ The Ring sum of two graphs G_1 & G_2 are denoted by $G_1 \oplus G_2$

$G_1 \oplus G_2 = (G_1 \cup G_2) - (G_1 \cap G_2)$

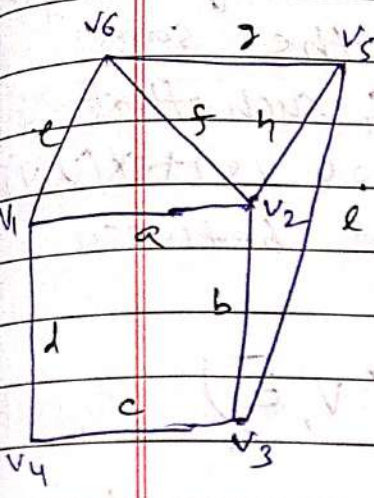


⇒ De-compositions of graph ⇒ $G_3 = G_1 + G_2$ Graph G_3 are decomposite into two graphs G_1 & G_2

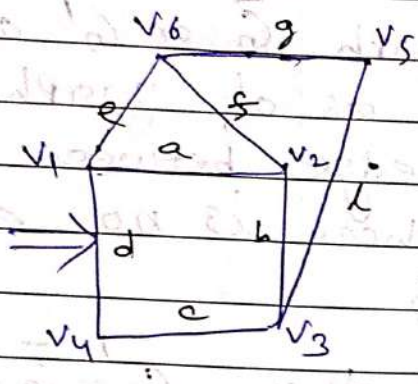


$\therefore G = G_1 + G_2$

⇒ Deletion ⇒



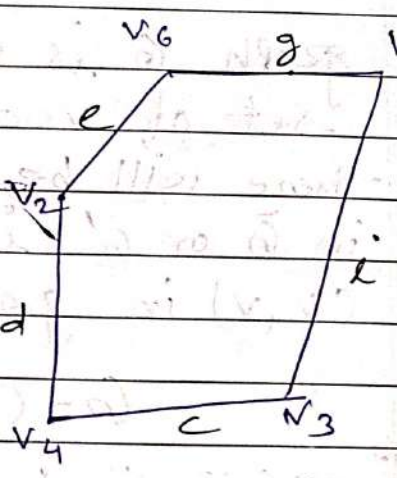
$G = (V, E)$



$G' = (G - h)$

जब edge delete की जाती है तो only वही edge delete होती है vertex & बाकी edges पर कोई effect नहीं पड़ता है

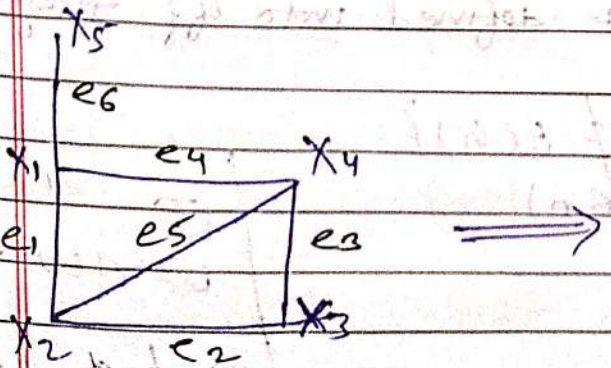
&

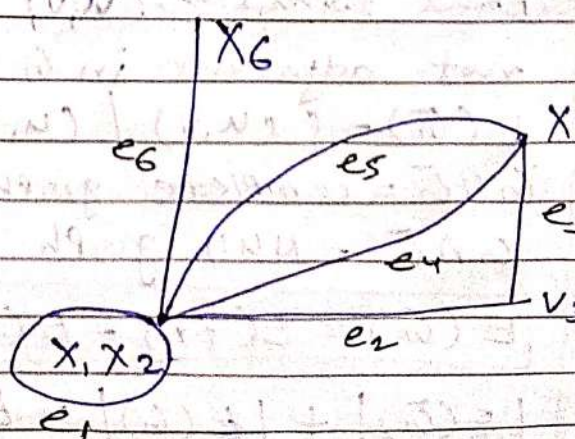


$G'' = (G - V_2)$

जब vertex delete किया जाता है तो vertex से connected all edge भी delete होती है

⇒ **fusion** ⇒ A pair of vertices X_1, X_2 in graph G are called fused if the two vertices are replaced by a single new vertex such that every edge that was incident on either X or Y or both is incident on the new vertex. Thus fusion of two vertices does not alter the number of edges but it reduces the number of vertices by one.





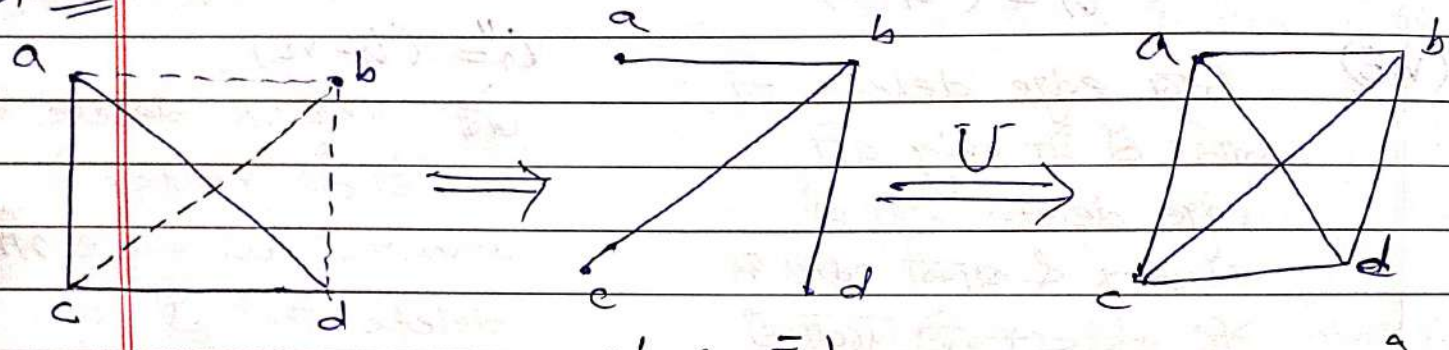
fusion of vertices are X_1 & X_2.

* Complement of graph ($\bar{G}$ or G') ⇒

Complement of a simple graph G is a graph $\bar{G}$ or G' on the same set of vertices as of graph G such that there will be an edge between two vertex (u,v) in $\bar{G}$ or G' iff there is no edge between (u,v) in graph G

$G = (V, E)$ then $G' = \bar{G} = (V, \bar{E})$

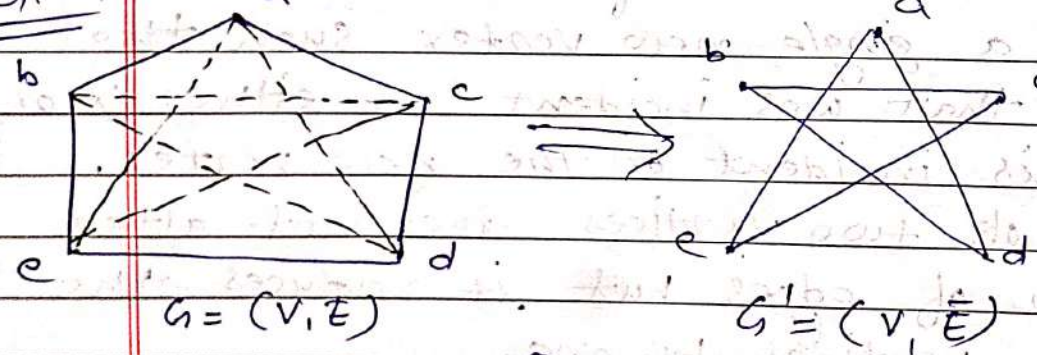
Example



$G = (V, E)$ $G' = (V, \bar{E})$

complement graph के case में complement graph का result draw करते है)

Example



$G = (V, E)$ $G' = (V, \bar{E})$

Important Points ⇒

① (u,v) are defined into iff they are not adjacent in G

② $E(\bar{G}) = \{(u,v) \mid (u,v) \notin E(G)\}$

③ $G \cup \bar{G}$ = complete graph (K_n)

④ $G \cap \bar{G}$ = NULL graph

⑤ $\boxed{E(\bar{G}) = E(K_n) - E(G)}$

⑥ $|E(\bar{G})| + |E(G)| = E(K_n) = \frac{n(n-1)}{2}$

⑦ $U = A \cup \bar{A}$ = complete graph

⑧ $\phi = A \cap \bar{A}$ NULL graph

Practice Questions based on complete graph ⇒

Q.1) Consider a simple graph G

(i)

$|E(G)| = 30$

$|E(\bar{G})| = 36$

$|V(G)| = ?$

$|E(G)| + |E(\bar{G})| = |E(K_n)|$

$|E(G)| + |E(\bar{G})| = \frac{n(n-1)}{2}$

$30 + 36 = \frac{n(n-1)}{2}$

$66 = \frac{n(n-1)}{2}$

$132 = n^2 - n$ $\therefore n^2 - n - 132 = 0$

$n = 12$ ✓ & $n = -1$ ✗

(ii) $|V(G)| = 8$

$|E(G)| = 12$

$|E(\bar{G})| = ?$

$|E(G)| + |E(\bar{G})| = |E(K_n)|$

$|E(G)| + |E(\bar{G})| = \frac{n(n-1)}{2}$

$12 + |E(\bar{G})| = \frac{8(8-1)}{2}$

$12 + |E(\bar{G})| = 28$

$|E(\bar{G})| = 28 - 12$

$\therefore |E(\bar{G})| = 16$

(iii) $|E(G)| = 56$

$|E(\bar{G})| = 80$

$|V(G)| = ?$

$|E(G)| + |E(\bar{G})| = |E(K_n)|$

$|E(G)| + |E(\bar{G})| = \frac{n(n-1)}{2}$

$56 + 80 = \frac{n(n-1)}{2}$

$n(n-1) = 2 \times 136$ $\therefore n^2 - n = 272$

$n^2 - n - 272 = 0$

$n = 17$ ✓ & $n = -16$ ✗

connected graph में all vertices की degree even / odd होती है।

Connected & Dis-connected Graph ⇒

A Graph is said to be connected when there is a path b/w every pair of vertices.

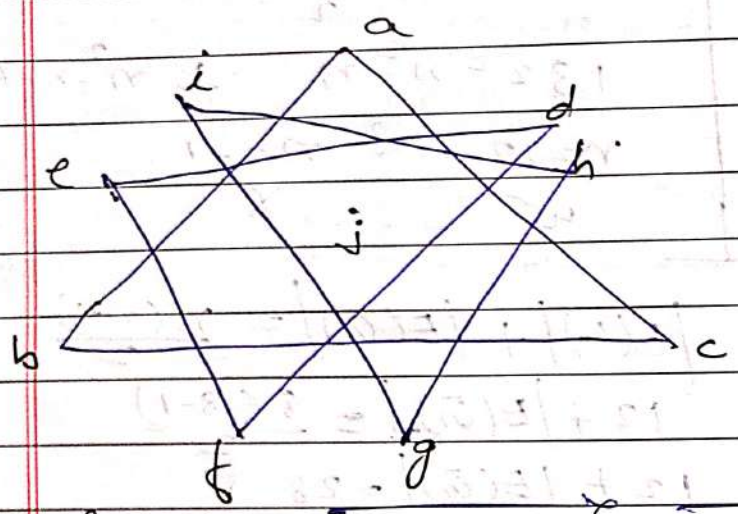
(or)

A Graph is called connected if there is a path from any vertex u to v or vice versa.

⇒ A Graph is called dis-connected if there is not path between any two vertices.

* **Connected component ⇒**

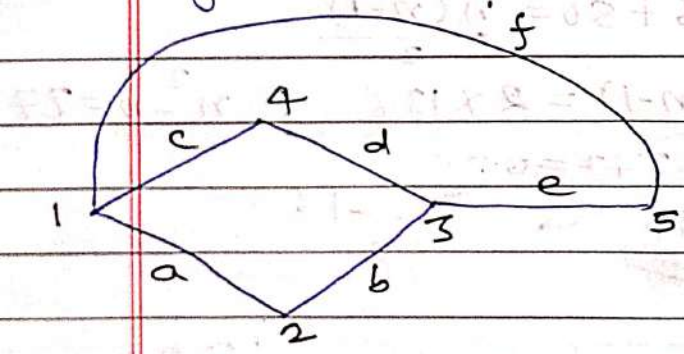
A sub graph of graph G is called the connected component of G if it is not contained in any bigger sub graph of graph G which is connected.

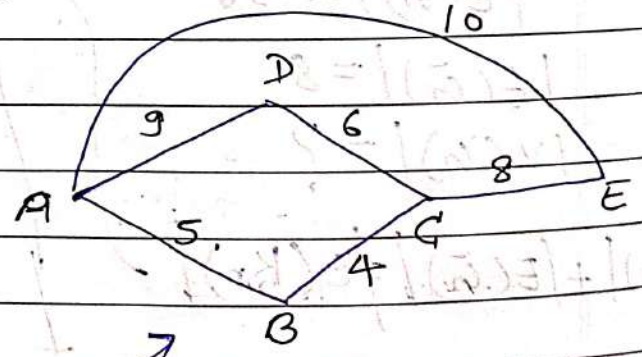


Connected components of this graph are
{a, b, c}, {d, e, f}, {g, h, i}

* **Labelled Graph & weighted graph ⇒**

If the vertices & edges of a graph G are ~~labelled~~ labelled with name or data then the graph is called labelled graph.

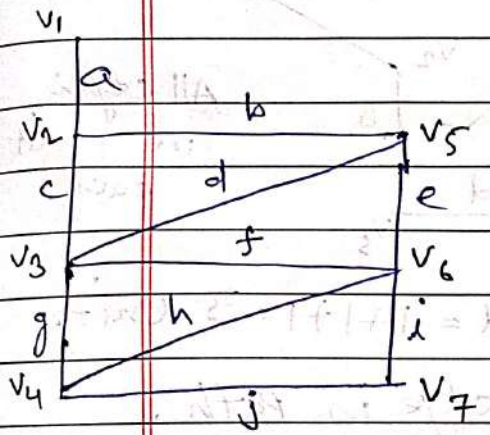


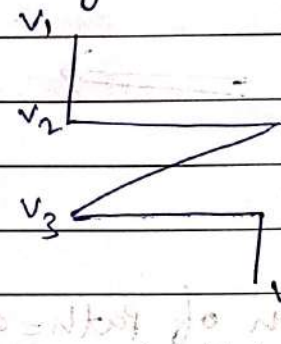


When, In graph G, some additional information is given by assigning positive number is called weight to the vertices to the vertices & edges then these type of graph is called weighted graph.

* ~~Path~~ Walk ⇒

Walk is a finite alternative sequence of vertices & edges, begining & ending with same or different vertices.
Such that, no edge can be appear more than once but vertex may be appear more than once.





walk = $v_1 a v_2 b v_5 d v_3 f v_6 i v_7$

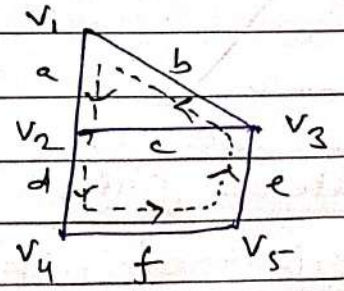
Walk may be two types → closed walk.
→ Open walk.

⇒ Open walk ⇒ When a walk starting & ending with different vertices are called open walk.

Example :- $v_1 a v_2 b v_5 d v_3 f v_6 i v_7$
(v_1 ↑ starting, v_7 ↑ Ending)
Starting vertex = v_1
Ending vertex = v_7

⇒ closed walk ⇒ When a walk starting & ending with same vertices are called closed walk.

walk = $v_1 a v_2 d v_4 f v_5 e v_3 b v_1$
(v_1 ↑ starting, v_1 ↑ Ending.)



Note :- vertex & edge may be repeated in walk.

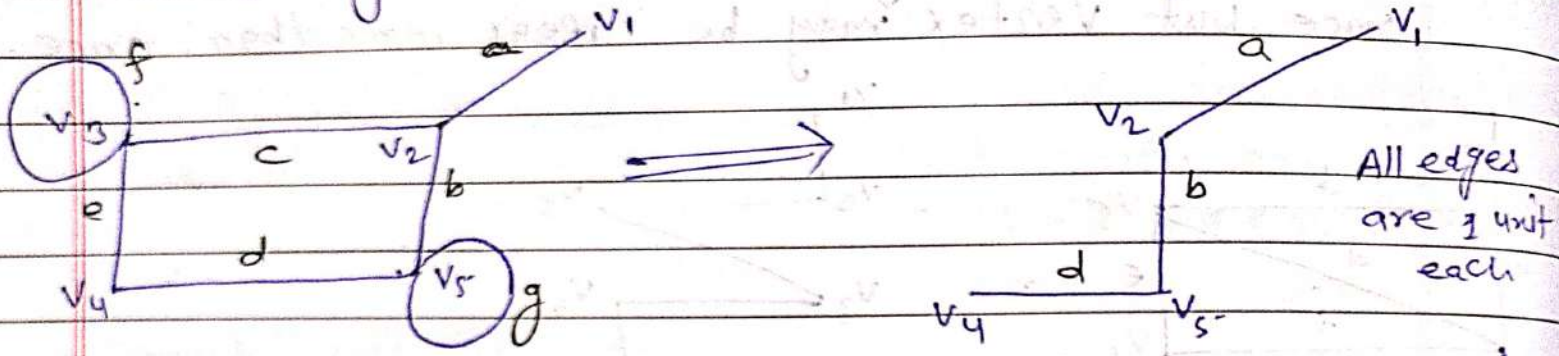
P.T.O.

* Path ⇒

A open walk in which no vertex appears more than once is called Path.

→ A self loop can not be a part of Path.

→ No. of edges in a Path is called length of Path.



Length of Path = a + b + d = 1 + 1 + 1 = 3 Unit.

Note:- we can not repeat vertex & edge in Path.

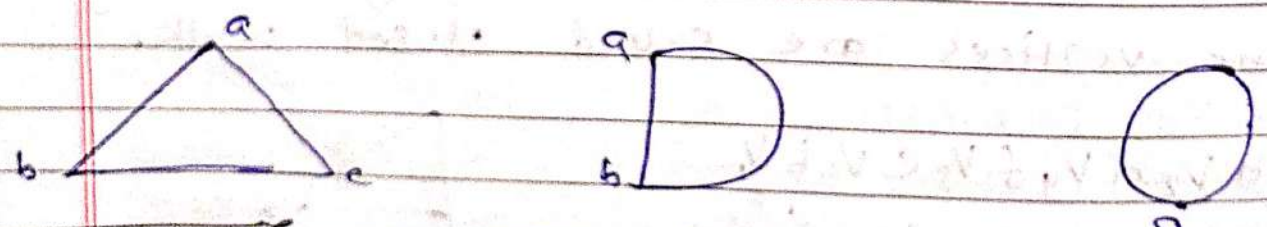
* Circuit ⇒

A closed walk, in which no vertex appears more than once is called circuit.

→ It is non-intersecting walk such that every vertex of degree is two.

→ A self loop is also a circuit. It means that circuit is also called a circular Path.

Example:-



Euler's graph

* Euler circuit/cycle ⇒

A closed walk when visit every edge of the graph exactly once is called euler circuit or euler cycle.

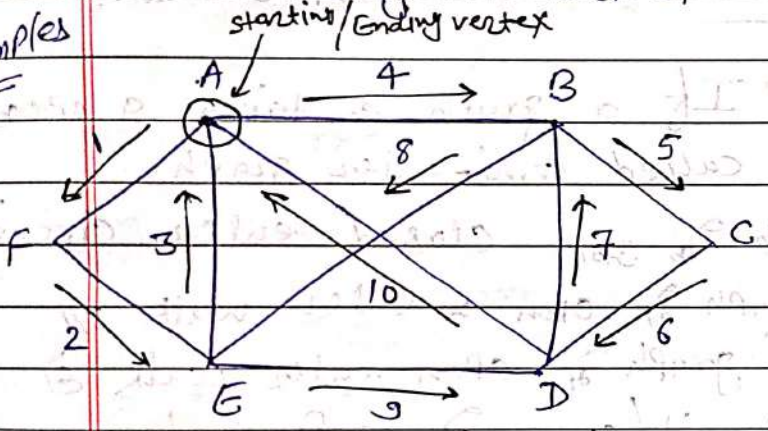
Note:- Vertex can be repeated but edges not repeated.

* Euler graph ⇒

A Graph G, which contains euler circuit or cycle is called euler graph.

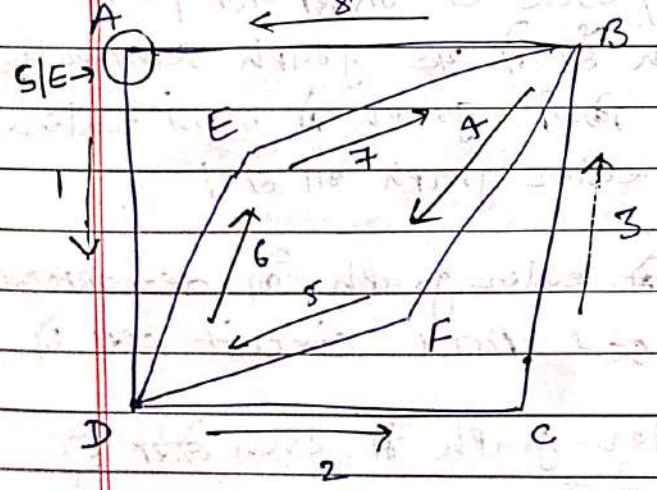
→ किसी graph में जीस vertex से Traversal करते हैं तो graph के all edges को only once traversal कर के उसी vertex पर आ जाते हैं तो उसे euler circuit/cycle कहते हैं एवं जीस graph में euler circuit present होता है उसे euler graph कहते हैं

Examples



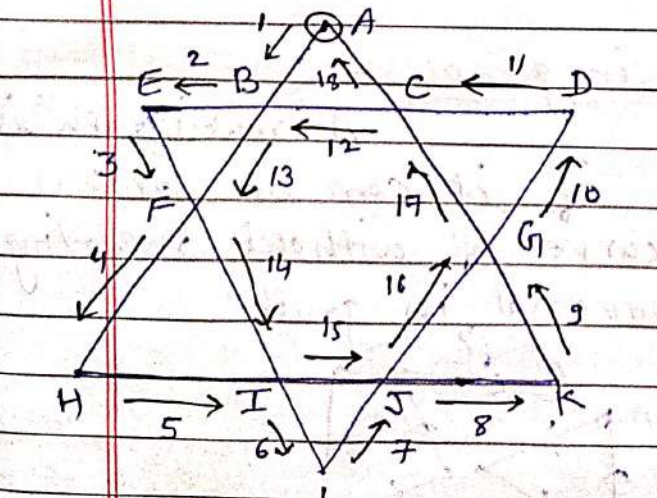
Degree of
D(A) = 4, D(B) = 4
D(C) = 2, D(D) = 4
D(E) = 4 D(F) = 2

All vertices even degree



~~Degree~~ Degree of
D(A) = 2, D(B) = 4
D(C) = 2 D(D) = 4

All vertices even degree

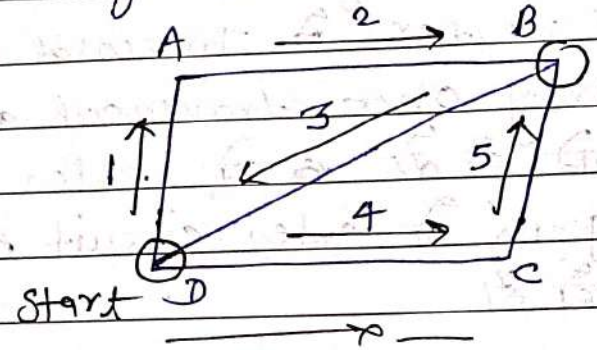


Degree of D(A) = 2, D(B) = 4, D(C) = 4
D(D) = 2, D(E) = 2, D(F) = 4
D(G) = 4, D(H) = 2 D(I) = 4
D(J) = 4 D(K) = 2 D(L) = 2
All vertices even degree

All above (three) graphs are euler graph because all graphs contains euler circuit or euler cycle in given graph.

Note:- More than one euler cycle/circuit, euler graph में हो सकते हैं

⇒ Open euler walk ⇒ A open walk which visite every edge of the graph exactly once, vertex may be repeated.



Start with vertex D & End with vertex B is called open euler walk.

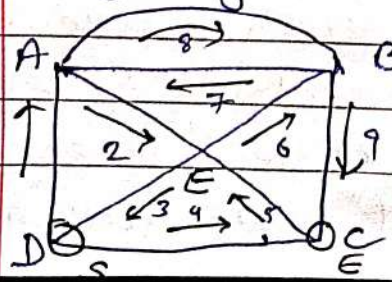
⇒ Semi-euler graph ⇒ If a graph contains a open euler walk is called semi-euler graph.

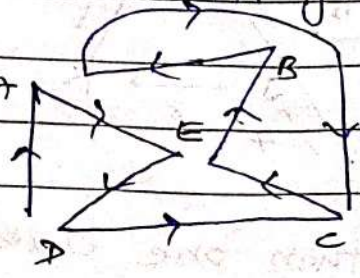
Note:- अगर किसी graph के अन्दर closed euler circuit/cycle हो तो इस graph में open euler walk भी होगा लेकिन अगर किसी graph मे open euler walk हो तो क्या closed euler circuit/cycle हो जरूरी नही।
अगर कोई graph euler graph है तो यह graph semi-euler graph भी होगा लेकीन अगर किसी graph मे semi euler graph हो तो जरूरी नही की euler graph भी हो।

Note:- ① connected graph मे euler graph को de-composed करने पर all decomposed part circuit होता है

② All vertices की degree euler graph मे even होती है

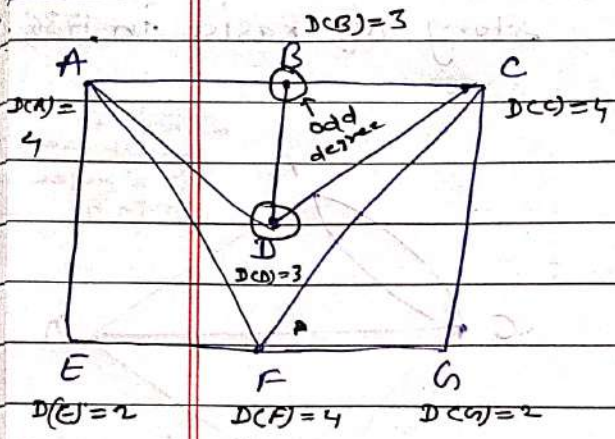
* Traversal/Traversable in graph ⇒ A multigraph is said to be traversable if it can be drawn without any break in curve & without repeating any edge is called traversal in graph



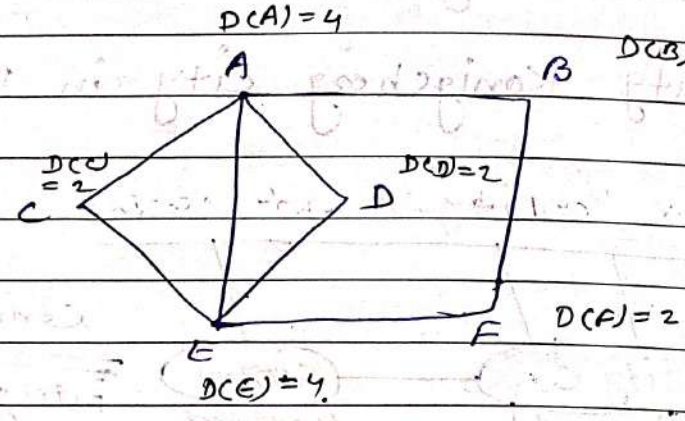


All edges travels without repeat of edge only vertex repeat.

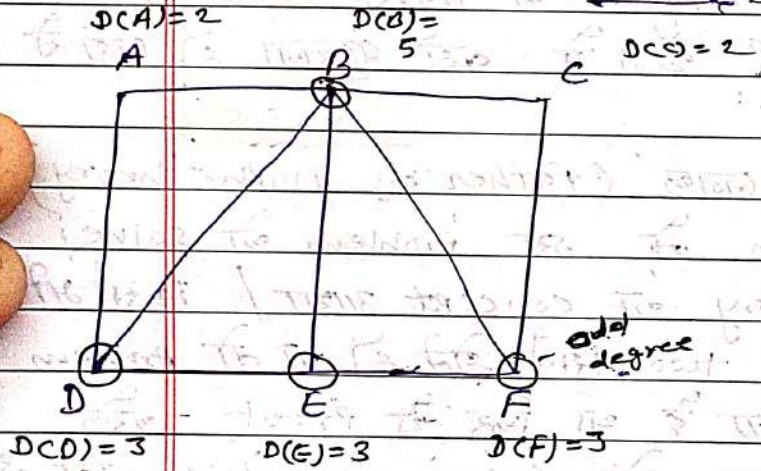
Q.1) Check which of following graph are euler or not?



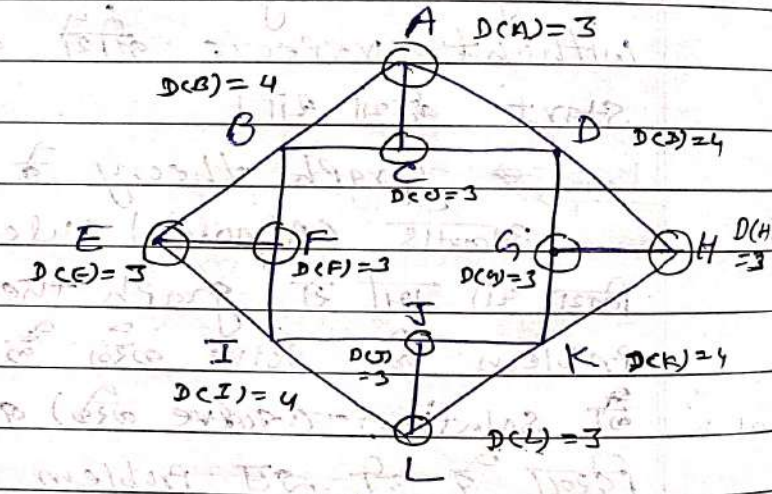
It's not euler graph because degree of every vertex is not even



It's euler graph because degree of every vertex is even



It is not euler graph



It's not euler graph

~~Planner graph के time पढ़े~~

Euler's formula ⇒

Euler's gives a formula which connects the number of vertices V, number of edges E & No. of regions R for any connected map (graph) then

$V-E+R=2$ for Planner graph

Q.1) If $V=6$, $E=9$ & $R=5$ then proof euler's formula.

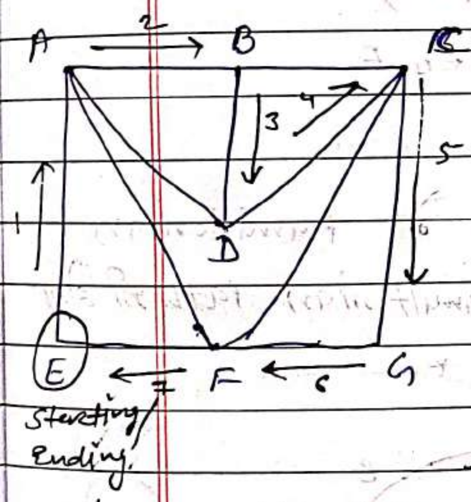
Sol) $V-E+R=2$ then $6-9+5=2 \Rightarrow 11-9=2$

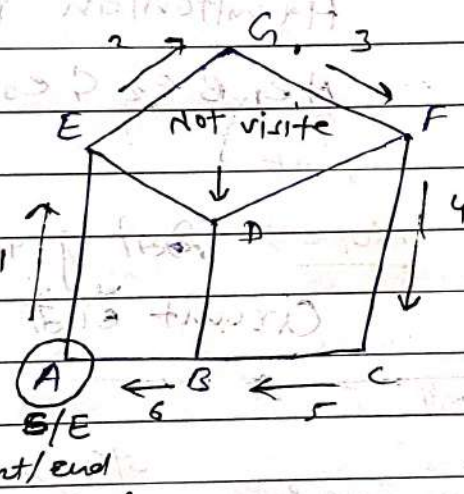
$2=2$ LHS = RHS so given graph (map) is Planner graph

* Hamiltonian Graph ⇒ इसमे All vertices को travel करते हो without repeat के only starting/ ending vertex को छोड़ के all edge travel करना जरूरी नही है।

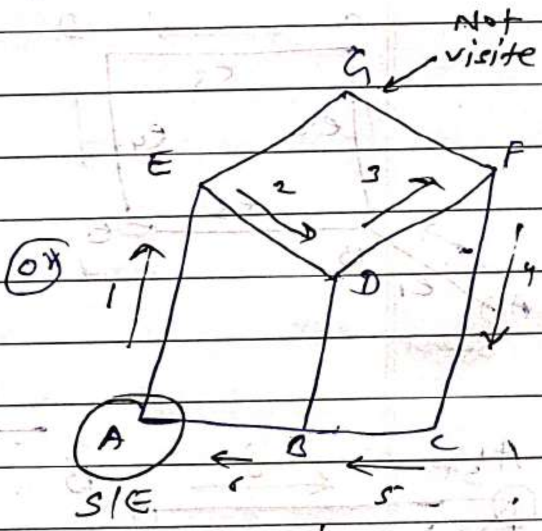
⇒ Hamiltonian circuit/cycle ⇒ In a connected graph is defined as :- a closed walk that traves (travels) every vertex of graph G exactly once, except of starting/ ending vertex.

⇒ Hamiltonian Graph ⇒ A Graph G is said to be Hamiltonian graph if it has a hamiltonian circuit/ cycle.





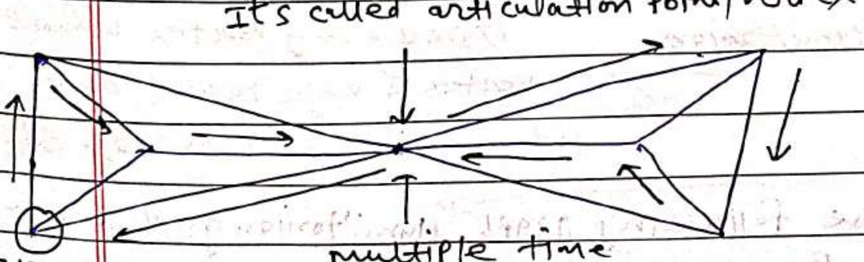
(OR)

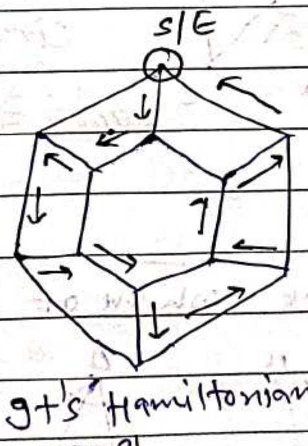


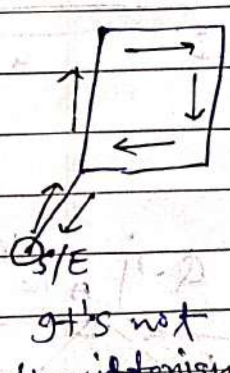
It's Hamiltonian Graph because All vertices are travers (travel)

It's not Hamiltonian Graph because some vertices are not Travers (visite) ~~the given graph is~~

It's called articulation point/vertex







It's not Hamiltonian graph Articulation Point/vertex are connect two different graph, अगर इस vertex को delete करने पर Two graph disconnected graph बन जाएंगे।

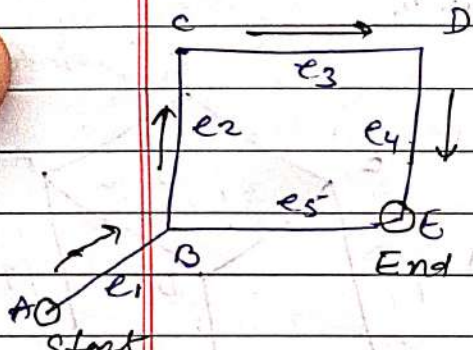
It's Hamiltonian graph

It's not Hamiltonian graph because edge repeat travers

⇒ Hamiltonian Path ⇒ किसी vertex से start करके पुरे graph के all vertex को visit (Travers) करके किसी और vertex पर end करे उसे Path को Hamiltonian Path कहते है क्योंकी Path मे start/ end vertex different (Not same) होते है क्योकी Path मे vertex repeat नही कर सकते है

Note:- अगर किसी graph मे Hamiltonian Path है तो जरुरी नही की वह Hamiltonian graph है।

⇒ Hamiltonian graph होने के लिये Hamiltonian circuit/ circle होना जरुरी है

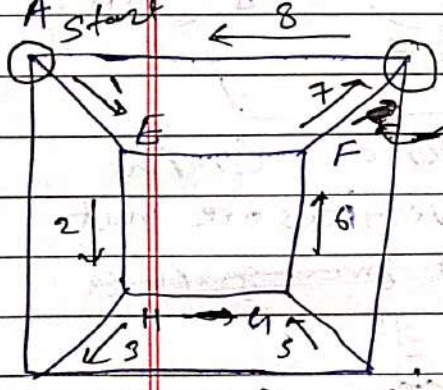


Hamiltonian Path ⇒

A e1 B e2 C e3 D e4 E

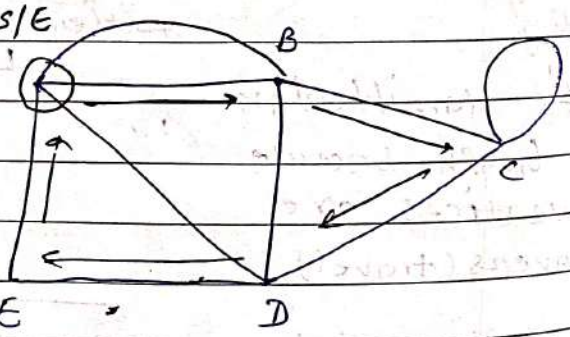
Start ↑ End ↑

Note ⇒ किसी graph मे Hamiltonian Circuit हो तो Hamiltonian Path भी होगा



Hamiltonian Path when remove these edge (8) otherwise Hamiltonian Circuit complete

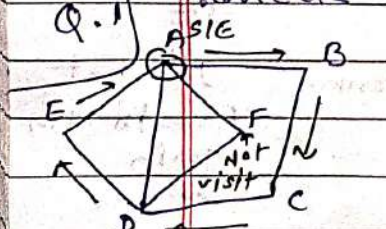
इसलिये ये Hamiltonian graph हो



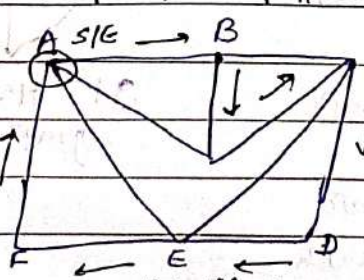
Travers every vertex without vertex & edge repeat But not compulsory every edge visit.

Check which of the following graph Hamiltonian graph or not?

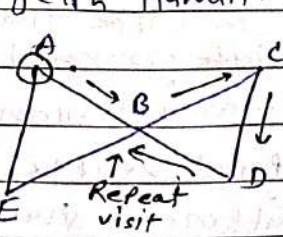
Q.1



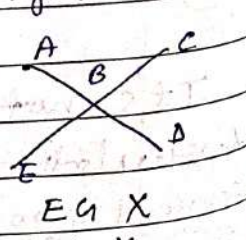
Euler graph (EG) ✓
Hamiltonian graph (HG) ✗



EG ✗ & HG ✓



EG ✓
HG ✗

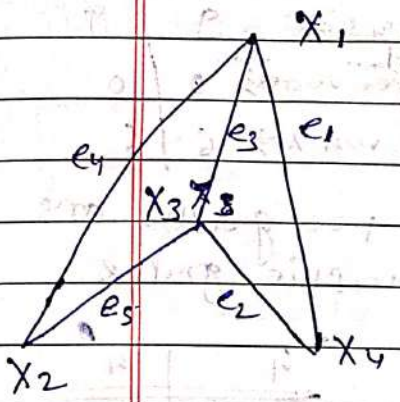


EG ✗
HG ✗

Isomorphic Graph ⇒ Two graphs G & G' are said to be isomorphic to each other if there is a one to one corresponding (correspondance) between their vertices & edges.

Conditions:-

① Same no. of vertices ② Same No. of edges
③ Equal no. of vertices with given degree.



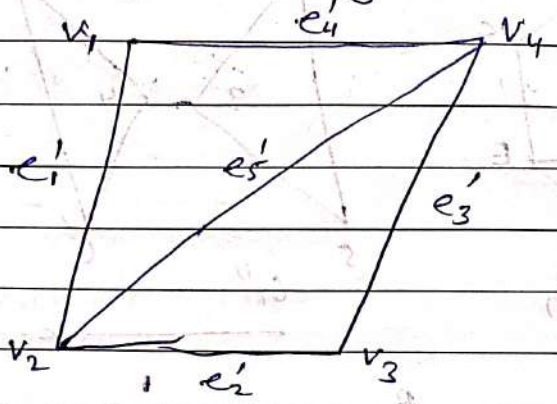
$G = (V, E)$

$V = \{X_1, X_2, X_3, X_4\}$

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

$D(X_1) = 3, \quad D(X_2) = 2$

$D(X_3) = 3 \quad D(X_4) = 2$



$G' = (V', E')$

$V' = \{V_1, V_2, V_3, V_4\}$

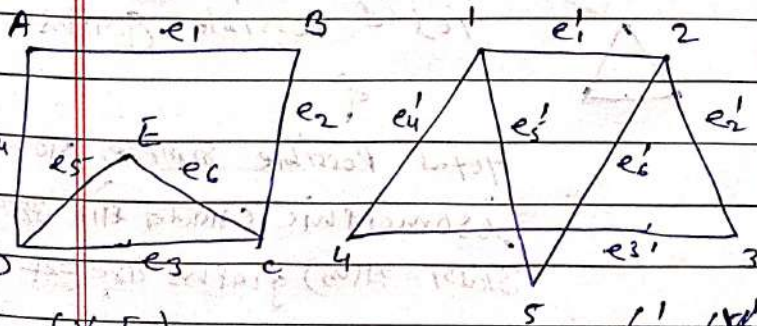
$E' = \{e_1', e_2', e_3', e_4', e_5'\}$

$D(V_1) = 2 \quad D(V_2) = 3$

$D(V_3) = 2 \quad D(V_4) = 3$

∵ No. of vertex & No. of edges & No. of degree vertices are equal so given two graphs are isomorphic graphs.

Q.1] Check which graphs are isomorphic ?



$G = (V, E)$

$V = \{A, B, C, D, E\}$

$E = \{e_1, e_2, e_3 \cdots e_6\}$

$D(A) = 2, \quad D(B) = 2 \quad D(C) = 3$

$D(D) = 3 \quad D(E) = 2$

$G' = (V', E')$

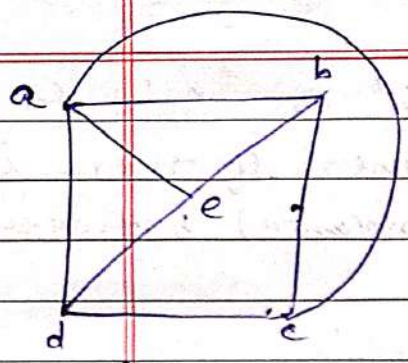
$V' = \{1, 2, 3, 4, 5\}$

$E' = \{e_1', e_2', e_3', \cdots e_6'\}$

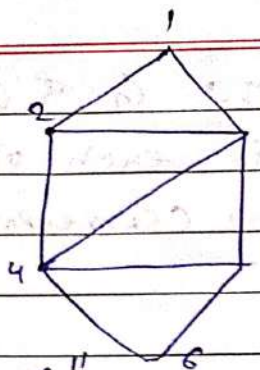
$D(1) = 3, \quad D(2) = 3, \quad D(3) = 2$

$D(4) = 2, \quad D(5) = 2$

In two graph G & G'
(i) No. of vertices equal
(ii) No. of edges are equal
(iii) Equal degree of vertices are same
Degree 2 = 3 & ~~deg~~
~~Degree~~ 3 = 2
so given two graph G & G' are isomorphics



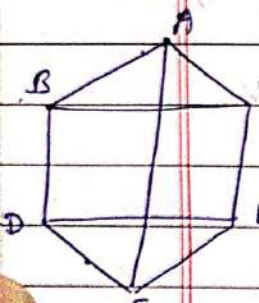
G



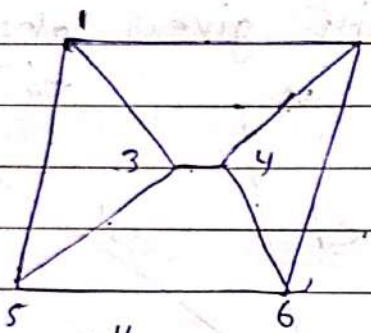
G''

	G	G''
No. of vertices ⇒	5	6
No. of edges ⇒	8	9

So given two graphs are not isomorphic



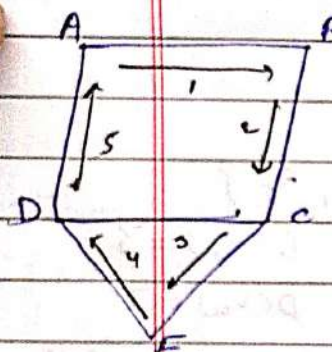
G



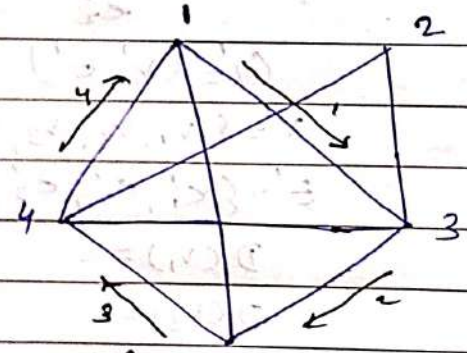
G''

	G	G''
No. of vertices ⇒	6	6
No. of edges ⇒	9	9
Two Degree vertex ⇒	0	0
Three Degree vertex ⇒	6	6

So given two graphs are isomorphic graphs.



G



G''

	G	G''
No. of vertex ⇒	5	5
No. of edges ⇒	6	8

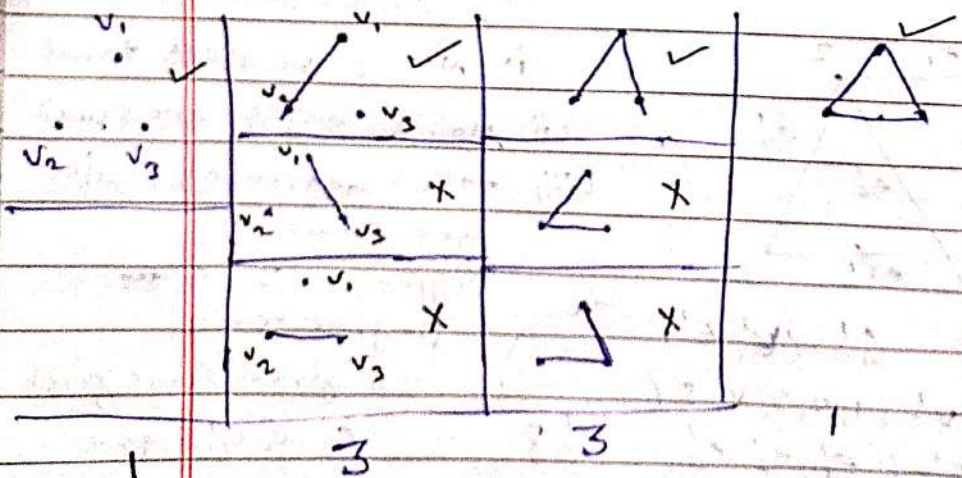
Not isomorphic graphs

Longest cycle length in

G ⇒ CL(G) = 5 G'' ⇒ CL(G'') = 4

Q) How many simple non-isomorphic graphs are possible

(i) with 3 vertices

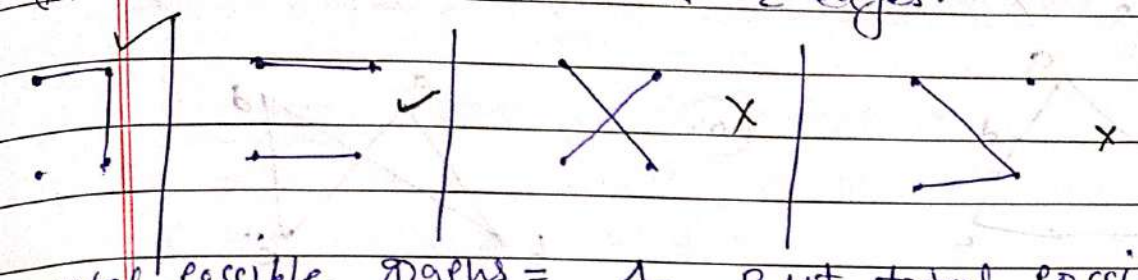


Total possible graphs = 8

Total possible simple non isomorphic (अलग अलग ग्राफ) graphs are = 4

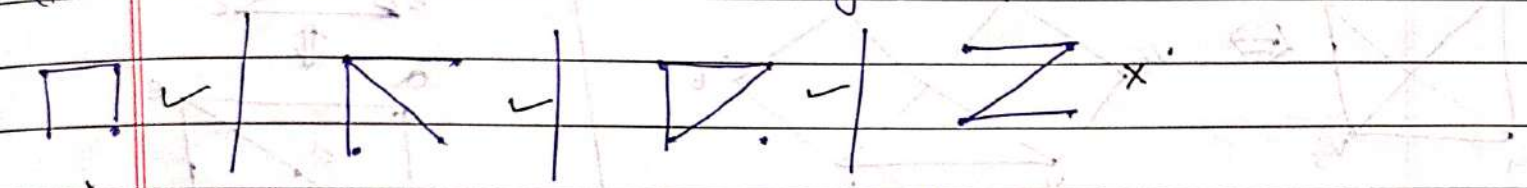
Non-isomorphic :- ये ऐसे graph नहीं होना चाहिये

(ii) with 4 vertices & 2 edges.



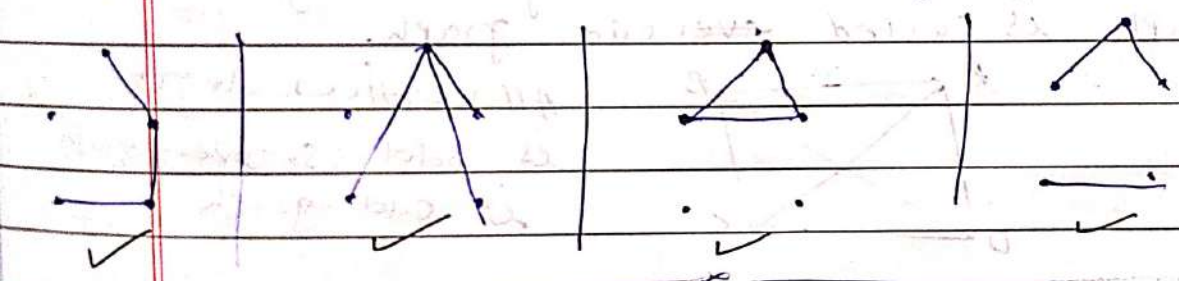
Total possible graphs = 4 But total possible simple non isomorphic graphs are = 2

(iii) with 4 vertices & 3 edges



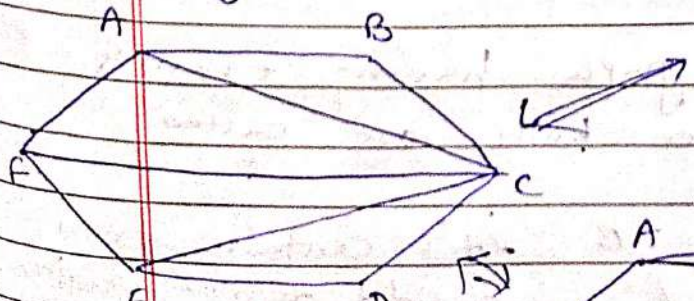
Total possible simple graph = 4 But
total possible simple non-isomorphic graphs = 3

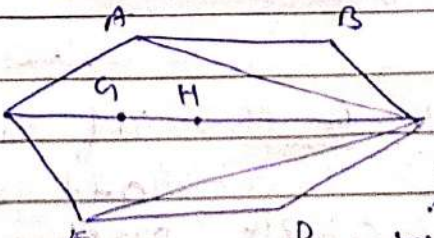
(iv) with 5 vertices & 3 edges



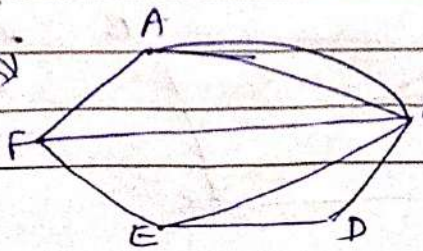
Total possible simple non isomorphic graphs = 4

* Homomorphic Graph ⇒ Two graphs are said to be ~~homomorphic homomorf~~ Homomorphic graph if one graph can be obtained from the other graph by creating of edges or merge of edges in series.





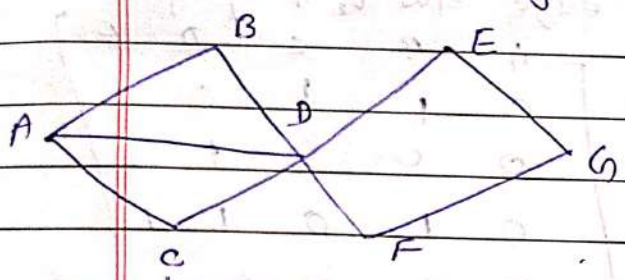
it's obtained by created of edge



it's obtained by merge of edges

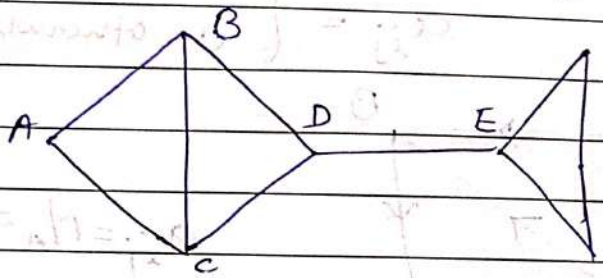
* **Distance & Diameter of graph** ⇒ Consider a connected graph D. The distance between vertices X & Y in graph G is written as d(X,Y), is the length of shortest path between X & Y.

The Diameter of graph G is written as Diam(G), is the maximum distance between any two vertex in a graph G.



Distance D(A,F) = 2
{AD → DF edges}

Diameter
Diam(G) = 3
{AG → CD → DF (or)
AB → BD → DF}



D(A,E) = 3 {AB → BD → DE (or)
AC → CD → DE}

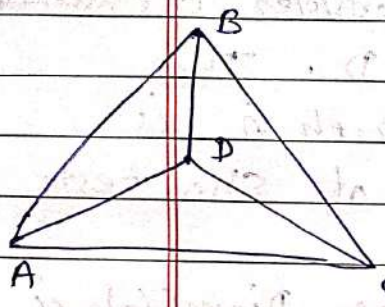
Diam(G) = 4 {AB → BC → CD → DE
(or)
AC → CB → BD → DE}

* **Graph representation in computer memory** ⇒ Graph representation in computer memory mainly two types (ways):-
  - Adjancy matrix representation
  - Linked list (Adjancy structure) representation

⇒ (1) **Adjancy Matrix representation** ⇒ ~~Undirected graph~~ ⇒ suppose graph G with m vertices & vertices have been ordered $V_1, V_2, V_3, \ldots, V_m$ then adjancy matrix of $A = [a_{ij}]$ of the graph G is the m x m matrix defined by.

$$a_{ij} = \begin{cases} 1 & \text{if } (V_i, V_j) \text{ is an edge means } V_i \text{ adjacent to } V_j \\ 0 & \text{otherwise } \{\text{No adjancent b/w } V_i \text{ & } V_j\} \end{cases}$$

In Un-directed graph ⇒



matrix of A

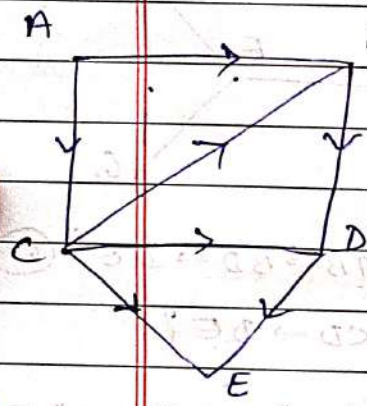
$a_{ij} = M_A =$

	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

4×4

In Directed graph ⇒

$a_{ij} = \begin{cases} 1 & \text{if } v_i, v_i \text{ is an edge Here } v_i \text{ initial \& } v_j \text{ final} \\ 0 & \text{otherwise No edge b/w } v_i \text{ \& } v_j \end{cases}$



$a_{ij} = M_A =$

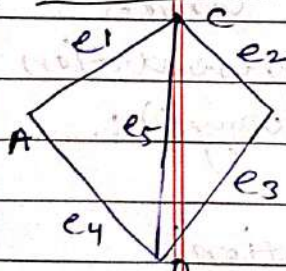
	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	1	0
C	0	1	0	1	1
D	0	0	0	0	1
E	0	0	0	0	0

5×5

⇒ Incidence Matrix representation ⇒

$a_{ij} = \begin{cases} 1 & \text{if vertex } v_i \text{ incident by edge } e_j \\ 0 & \text{otherwise} \end{cases}$

In un-directed graph



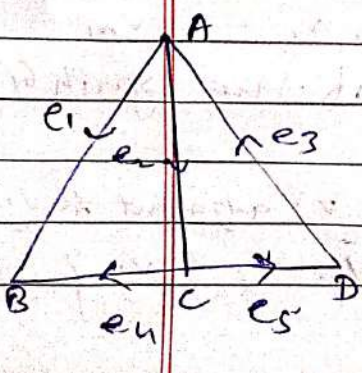
$a_{ij} = M_A =$

	e_1	e_2	e_3	e_4	e_5
A	1	0	0	1	0
B	0	1	1	0	0
C	1	1	0	0	1
D	0	0	1	1	1

4×5

In Directed graph ⇒

$a_{ij} = \begin{cases} 1 & \text{if } e_j \text{ is out going edges from } v_i \\ -1 & \text{if } e_j \text{ is incomming edge to } v_i \\ 0 & \text{otherwise \{ No incomming \& outgoing edge} \end{cases}$



$a_{ij} = M_A =$

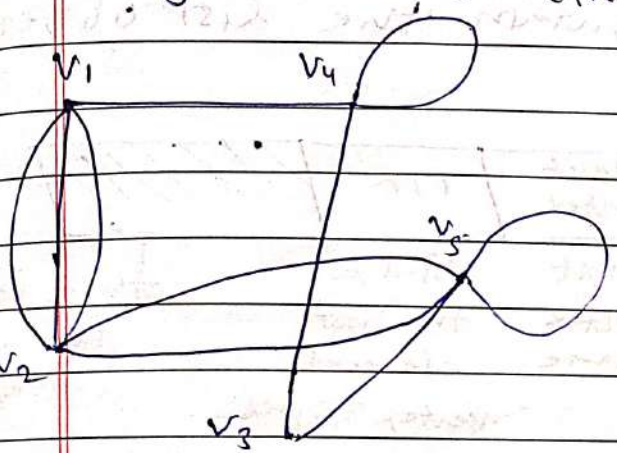
	e_1	e_2	e_3	e_4	e_5
A	1	1	-1	0	0
B	-1	0	0	-1	0
C	0	-1	1	1	0
D	0	0	1	0	-1

4×5

Representation in Multigraph/General Graph ⇒

Adjancy matrix representation

$a_{ij} = \begin{cases} N & \text{if there are one or more than one edges b/w } v_i \text{ \& } v_j \text{ where } N \text{ is No. of edges} \\ 0 & \text{otherwise} \end{cases}$



$a_{ij} = M_A =$

	v_1	v_2	v_3	v_4	v_5
v_1	0	3	0	1	0
v_2	3	0	0	0	2
v_3	0	0	0	1	1
v_4	1	0	1	(1)	0
v_5	0	2	1	0	(1)

self loop

⇒ **Linked list representation** ⇒ Major draw back in adjacent matrix representation

→ Difficult to insert & delete vertices in graph G, reason is size of matrix. If matrix size change then many changes in matrix

→ Space complexity $O(n \log n)$

→ Memory space will be wasted due to some zero entry (बीना कीसी काम की zero entry)

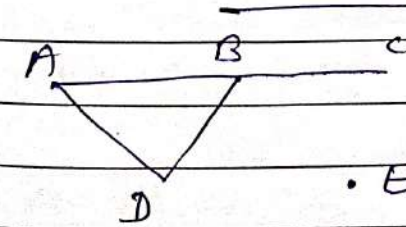
Soln ⇒ Here :-

ϕ denote the empty list

colone ⇒ : denotes seperates a vertex from it's list

semicolon ⇒ ; denotes seperates the different list.

Q.1)



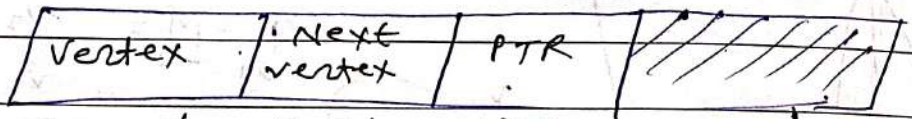
Steps (S.No.)	vertex	Adjancy list
1	A	B, D
2	B	A, C, D
3	C	B
4	D	A, B
5	E	ϕ

$G = [A: B, D ; B: A, C, D ; C: B ; D: A, B ; E: \phi]$

Linked list representation of a graph G in memory maintain only two files
- vertex file
- Edge file

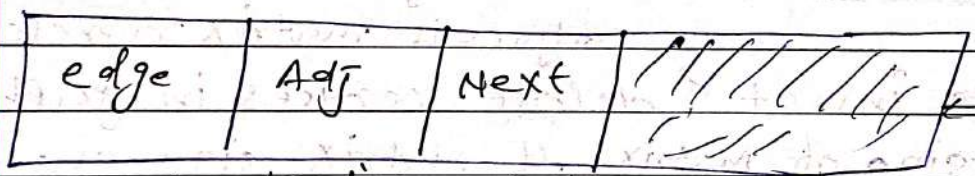
→ Vertex file ⇒ It will contain the list of vertices of the graph

File format:-



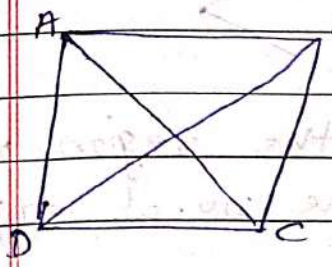
Name of vertex | Next vertex Name | Point to the first element vertex & edges | other information related vertex.

→ edge file ⇒ It will contain the list of edges

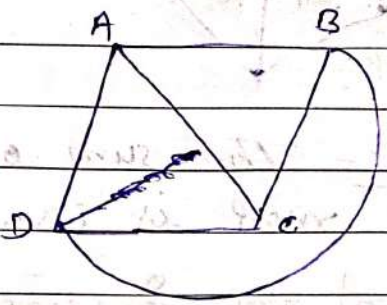


Name of edge | Location of vertex in the vertex file | Next point to the location of the next vertex in adjancy list

* **Planner Graph** ⇒ A Graph or multigraph which can be drawn in the plane so that it's edges do not cross (intersect) is said to be planner graph.



K_4 ⇒ Non-Planner graph



K_4 = Planner graph

If edges cross (intersect) to each other is called non-planner graph.

(or)

Planar graph ⇒ A Planar graph is a graph that can be drawn in the plane without any edge intersecting.

plane graph ⇒ A Plane graph is a planner graph that has been drawn in the plane without any edge intersecting (crossing)

A Plane graph divides the plane into region

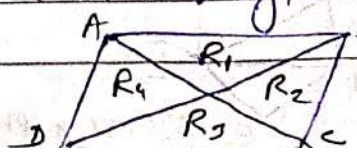
* **Maps & regions** (faces or) ⇒ A Particular Planar representation of a finite Planar multigraph is called a Map.

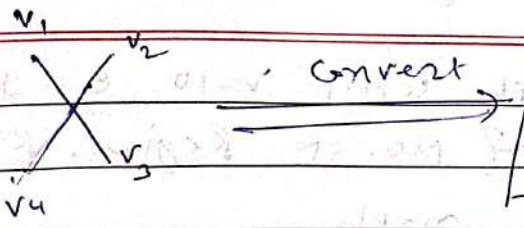
We can say that the Map is connected if the undering multigraph is connected.

⇒ A Map divides the plane into various (edges & vertices) regions called regions.

(or)

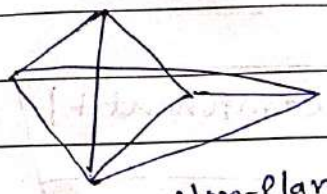
Every Plane graph has an unbounded regions called the exterior region.



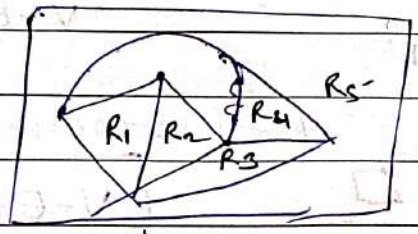
Q.1) 

Non-Planar graph

Planar graph

Q.2) 

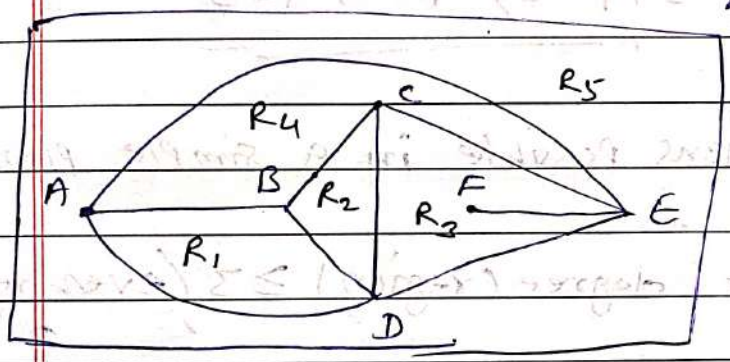
Non-Planar



Planar graph

R_1, R_2, R_3, R_4 are interior / bounded region & r_5 is exterior region

Degree of regions:- No. of edges it's bounded by region.



Degree of Region $(R_1) = 3$

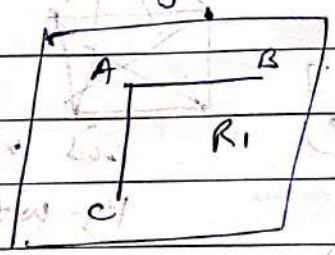
$DR(R_2) = 3$ $DR(R_3) = 5$

$DR(R_4) = 4$ $DR(R_5) = 3$

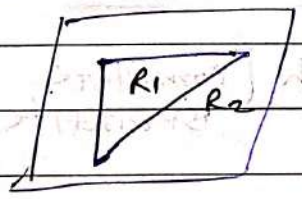
* **Simple Planar Graph ⇒** No self loop, No multiple edges means Parelled edgs.

Simple Planar graph contains with at least 2 edges & degree of region ≥ 3

Example ⇒



$D(R_1) = 4$



$D(R_1) = 3$

$D(R_2) = 3$

* **Euler's formula for Planar graph ⇒** for connected Graph G, E No. of edges, V no. of vertices & R No. of regions

$$V - E + R = 2 \implies V + R = 2 + E$$

Example On connected graph G with $V = 25$, $E = 60$ then $R = ?$

Soln $V + R = 2 + E \Rightarrow 25 + R = 2 + 60 \Rightarrow R = 62 - 25$

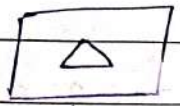
$R = 37$

$3R \leq 2E$

No. of region — No. of edges

Q.1) Let G be a planar graph with V=10, E=9 & 3 components then No. of Regions R=?

Soln

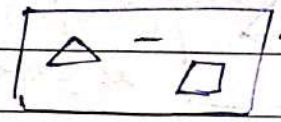


No. of component = 1

On connected graph

$V+E+R=2$

$1+1$



← Dis-connected graph.

No. of component = 3

$V-E+R$ = No. of component + 1

(or)

$V-E+R = n+1$ where n is no. of components

$10-9+R=3+1 \Rightarrow R=4-1=3$

Q.2) Maximum No. of regions possible in a simple planar graph with 10 edges is

Soln

Simple Planar graph ∴ degree (region) ≥ 3 (ever region)

& $e \geq 2$

Let No. of region r then No. of degree = $r \cdot 3$

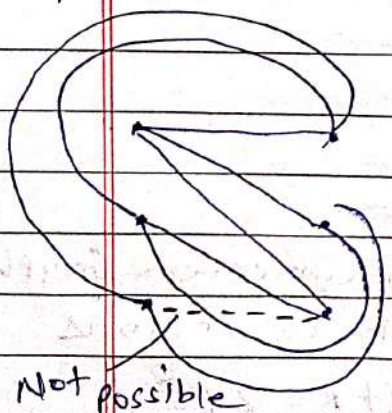
$r \cdot 3 \leq 2e$ then $3r \leq 2 \times 10$

$r \leq \frac{20}{3} = 6.66$ $\quad r \leq 6$

---

* Non-Planar Graph ⇒

$K_{3,3}$ - Bi-Partite graph [complete Bi-Partite graph]

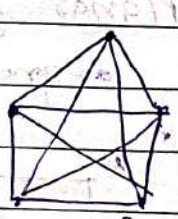


Not possible

So it's Non-Planar graph

$K_{3,2}$ Planar graph

K_4 complete graph is Planar graph

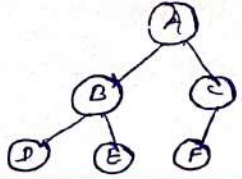


K_5 complete graph is non-Planar graph

K_5 with minimum vertices & 10 edges for minimum non-Planar graph

& $K_{3,3}$ vertices (6 5+3) vertices & 9 edge for minimum non Planar graph

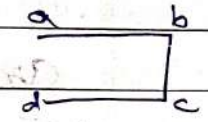
graph coloring



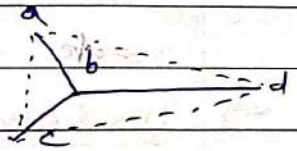
* **Tree ⇒** A Graph which has no cycle/circuit is called tree (OR)

A tree is an undirected graph that satisfies the following properties :-

(1) Tree is a connected & acyclic graph

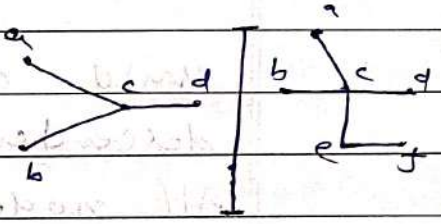


(2) Tree is a acyclic & a simple cycle is formed if any edge is added to tree then it's construct cycle



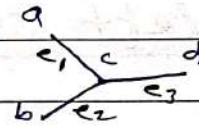
(3) Any two vertices in a tree can be connected by a simple unique path

Unique path b/w a to c, b to c, c to d.



(4) Tree is connected & has (n-1) edges

No. of edges in tree = (n-1) {n. no. of vertices}



$|V| = 4$

$|E| = 3$

if No. of edges $|e| < (n-1)$ it's disconnected graph

if No. of edges $|e| > (n-1)$ it's cycle

* **Basic terminology ⇒ related to tree ⇒**

⇒ **Root ⇒** A Node having only out degree is called root node. (OR)

⇒ **Left child ⇒** The nodes to the left side of the node/root node is called left child

⇒ **Right child ⇒** The node to the Right side of the node/root node is called Right child.

⇒ **Parent node ⇒** A node having left or Right or both child is called Parent of the nodes.

⇒ **Siblings ⇒** Two nodes having the same Parent are called siblings.

⇒ **Leaf node** ⇒ A node with no child (children) is called leaf node.

⇒ **Ancestor node** ⇒ If a node is the parent of another node then it is called ancestor of that node.
The root node is an ancestor of every other ~~node~~ node in the tree

⇒ **Descendent** ⇒ A node is called descendent of another node if it is the child of the node or child of some other descendent of that node.
All nodes in tree are descendent of root node.

⇒ **Left subtree** ⇒ The subtree whose root node is the left child of some node is called the left subtree of that node

⇒ **Right subtree** ⇒ The subtree whose root is the right child of some node is called right subtree of that node.

⇒ **Level of a node** ⇒ Level of a node is it's distance from the root node is called level of that node.
Level of root node is zero.
Maximum number of nodes at level N is $= 2^N$

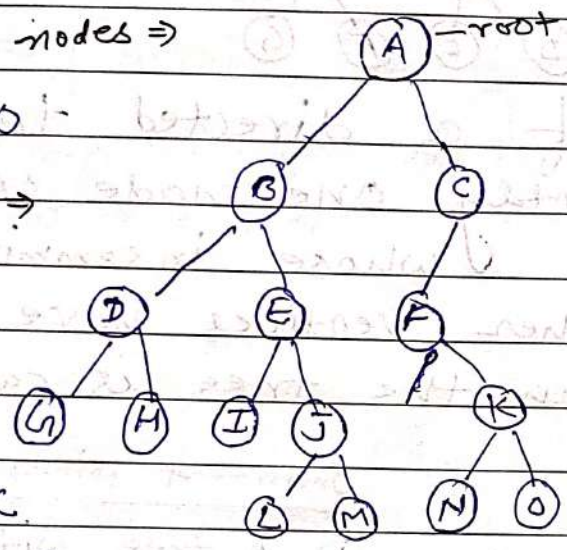
⇒ **~~Def~~ Depth or Hight of tree** ⇒
The ~~def~~ depth or Hight of a tree is defined as the maximum number of nodes in a branch of tree.

(or)

Depth of a tree is maximum level of tree.
Depth of root node is one.

⇒ Internal node ⇒ Nodes which has one or more than one child (children) are called internal (or) non-terminal node.

⇒ External node ⇒ Nodes which has no child (children) are called External or Terminal or leaf node.



External / leaf nodes ⇒ G, H, L, M, N, O

Internal nodes ⇒ B, C, D, E, F, I, J, K

Parent nodes
A for child B, C
B for child D, E

root node level = 0 Depth = 1

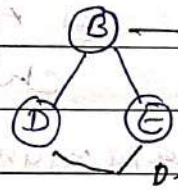
level = 1 Depth = 2

level = 2 depth = 3

level = 3 depth = 4

level = 4 depth = 5

Siblings ⇒ B & C, D & E, G & H, I & J, L & M, N & O

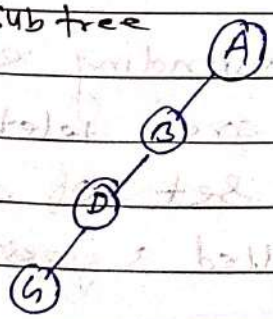


Right child ⇒ E
left child ⇒ D

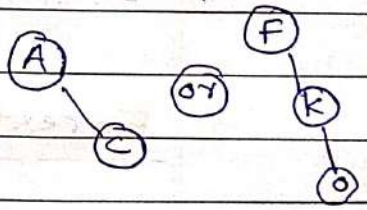
B — ~~Ascend~~ Ancestor

descendent

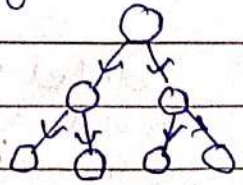
Left sub tree



Right sub tree ⇒

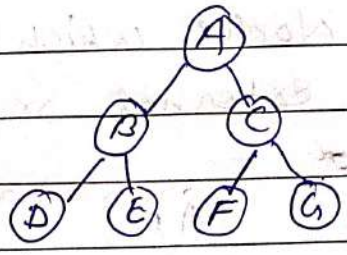


* Direct tree ⇒ A edge having direction is called directed tree

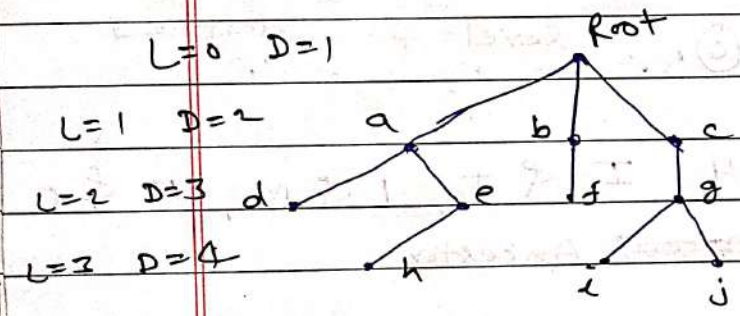


A node having only out degree is called root node. & a node having in & out degree is called internal node. A node having zero out degree is called external (leaf/terminal) node.

* **Ordered tree** ⇒ If on a tree, at each level an ordering is defined then such a tree is called an ordered tree



* **Rooted tree** ⇒ If a directed tree has exactly one node or vertex called root whose incomming degree is zero & all other vertices have incomming degree one then the tree is called rooted tree



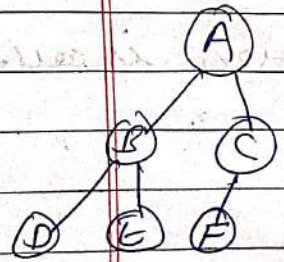
Important Points ⇒

① A tree with no node is a rooted tree [empty tree]

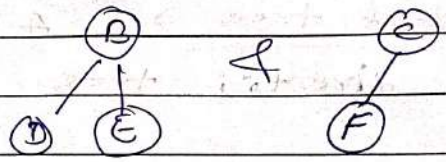
② A single node with no child is a rooted tree.

③ Depth of tree is maximum level of tree

* **Forest** ⇒ If the root & corresponding edges connecting the nodes are deleted from a tree, we obtain a set of disjoint trees. This disjoint tree is called a forest tree

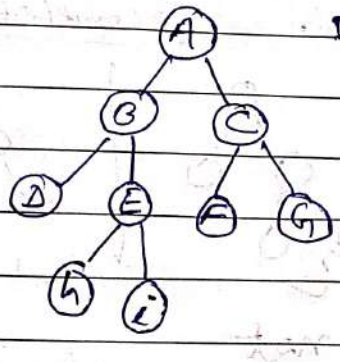


After delete root & it's connecting edges ⟹



These are forest tree

* **Binary Tree** ⇒ A binary tree is a special type of rooted tree, in which the root is having 2 degree & each remaining vertices degree is one or three



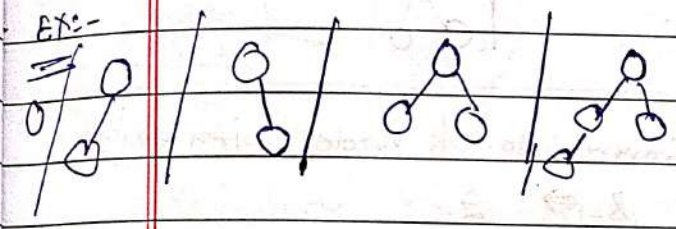
$D(A) = 2$

$D(B) = D(C) = D(E) = 3$

$D(D) = D(H) = D(I) = D(F) = D(G) = 1$

At most two children (child) at every node.

Ex:-

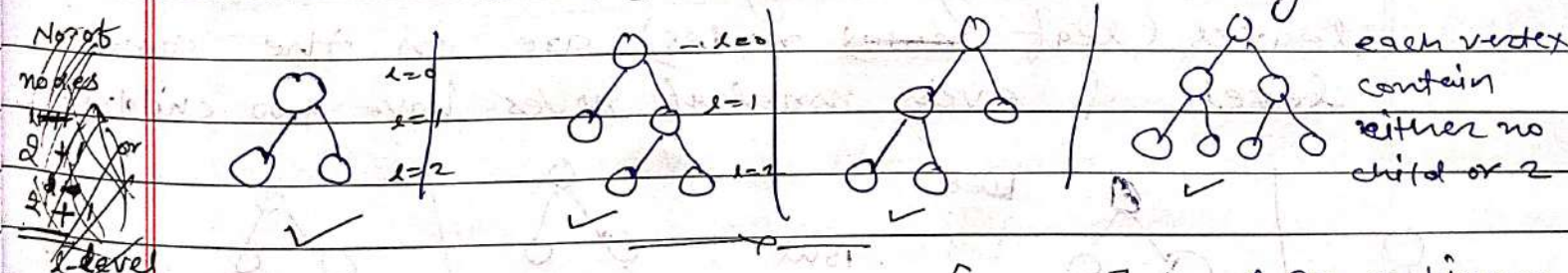


⇒ Maximum no. of nodes on level n in a binary tree is 2^n where $\{n \geq 0\}$

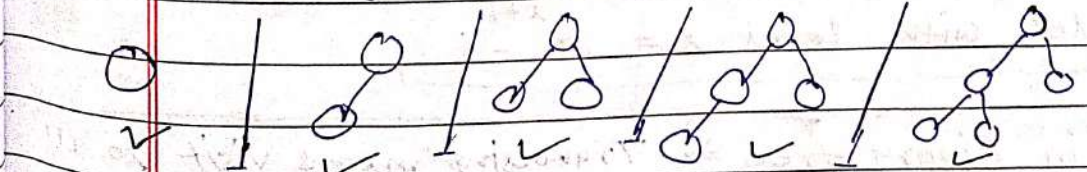
⇒ Maximum no. of nodes on depth or Hight (d/H) in a binary tree is $2^d - 1$ or $2^H - 1$ $\{$where $d/H \geq 1\}$

---

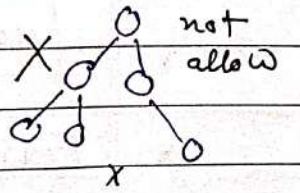
* **Full Binary tree (FBT)** ⇒ A binary tree which contains ~~0 children~~ either zero or two children (child) is called full Binary tree

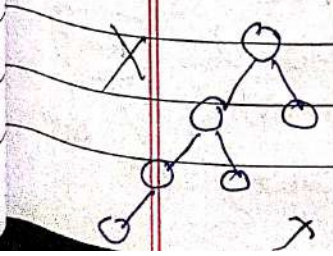


* **Almost complete binary tree [ACBT]** ⇒ In a binary tree always draw (make) left child before the right child.

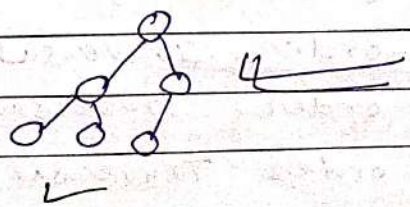


But

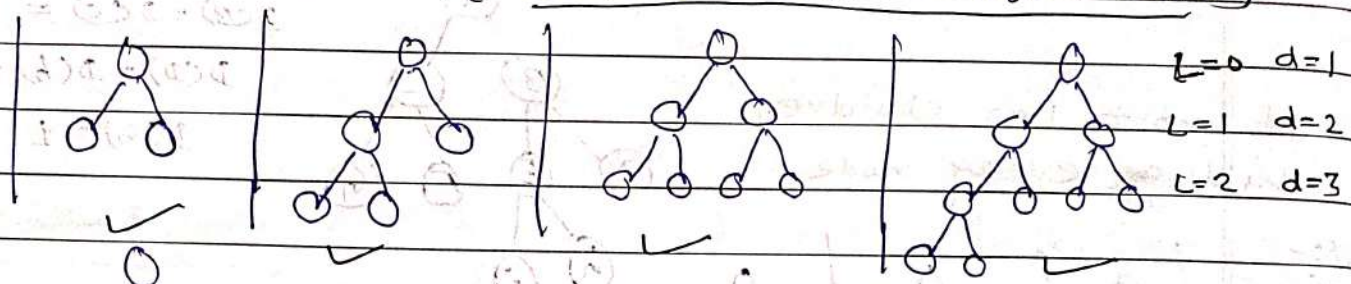


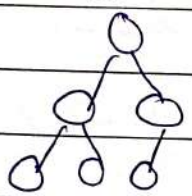


not allow



* **Complete Binary tree (CBT)** ⇒ Complete Binary tree is a binary tree if all it's level, except (शिवाय) possible last, have ~~the~~ the maximum no. of possible nodes. [complete left & Right child]



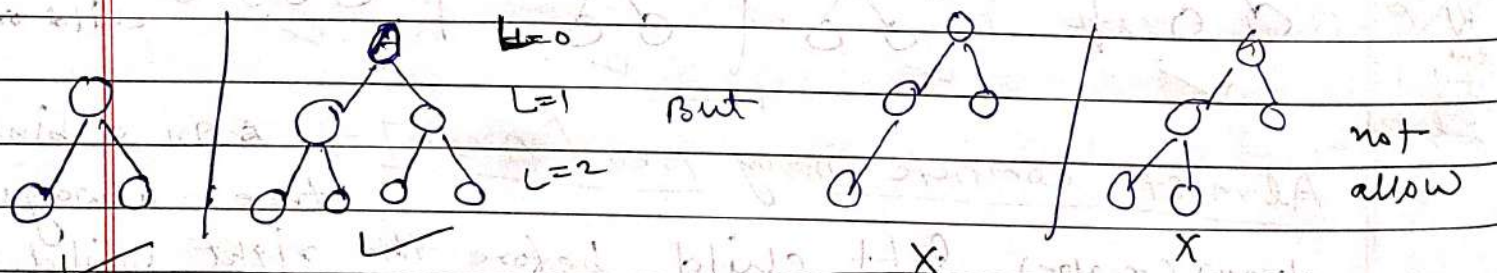
But  not allow

Maximum No. of nodes with level $l \Rightarrow 2^l$

Maximum No. of nodes in complete binary tree with depth d $= 2^d - 1$ or $2^H - 1$ H = ~~Height~~ Height

---

* **Strikly (स्ट्रिक्टली) Binary tree** ⇒ Strikly ~~Full~~ binary tree is a ~~strikly~~ binary tree in which all the leaves (leaf ~~nodes~~ nodes) are on the same level & every non-leaf nodes have two children

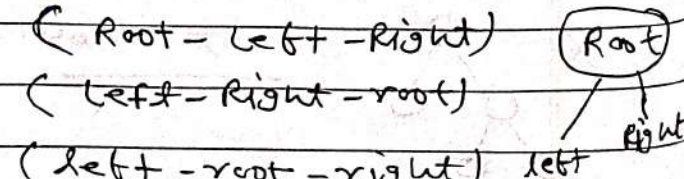


No. of nodes with level $l = 2^{l+1} - 1$

---

* **Traversing in Binary tree** ⇒ Traversing means visit to all nodes in tree

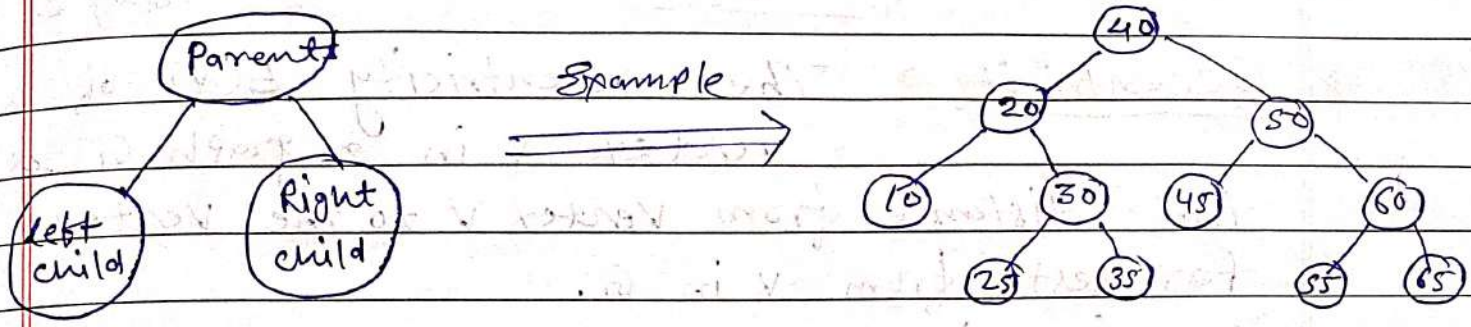
- → Pre-order Traversal (Root - Left - Right)
- → Post-order Traversal (Left - Right - root)
- → In-order Traversal (left - root - right)



* Binary search tree (BST) ⇒ Binary search trees has the following properties:-

(i) Left node contains smaller value of it's parent node.

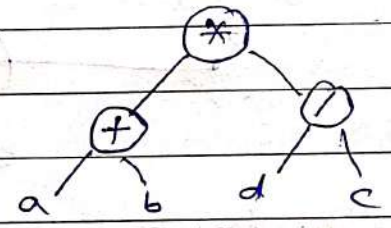
(ii) Right node contains large value of it's parent node.



* Binary Expression Tree ⇒ An algebraic expression represent in binary tree

<Left operand> (operator) <right operand>

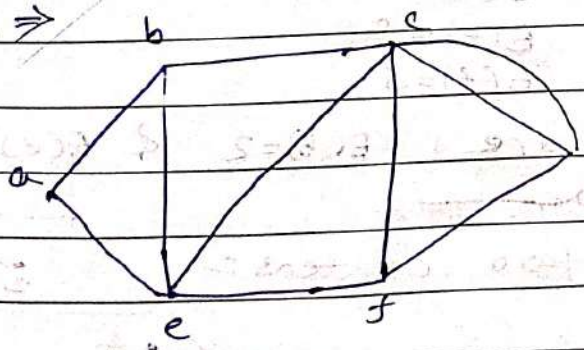
Example ⇒ (a+b) * (d/c)



* Distence & centers in tree ⇒

In connected graph G, the distence $d(v_i, v_j)$ between two of it's vertices v_i & v_j is the length of shortest path (number of edges in the shortest path) b/w them.

Example ⇒



D(a to c) or d(a, c)

⇒ a → b → c ⇒ length 2

⇒ a → e → c ⇒ length 2

⇒ a → e → b → c = length 3

⇒ a → e → f → c ⇒ 3 length

⇒ a → e → f → d → c = 4 length

So shortest path b/w a to c is 2 so distence of d(a,c) = 2

⇒ Metric property ⇒

(i) Non negativity :- $d(x,y) \geq 0$ & $d(x,y) = 0$ iff $x = y$

(ii) symmetric :- $d(x,y) = d(y,x)$

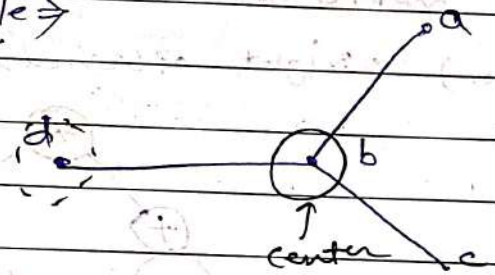
(iii) Triangle inequality :- $d(x,y) \leq d(x,z) + d(z,y)$ for any z

⇒ Eccentricity ⇒ The eccentricity $E(v)$ of a vertex v in a graph G is the distance from vertex v to the vertex farthest from v in G.

that is (i.e.) ⇒

$$E(v) = \max_{v_i \in G} d(v, v_i)$$

Example ⇒



$E(d) = 2$ { from d to a }

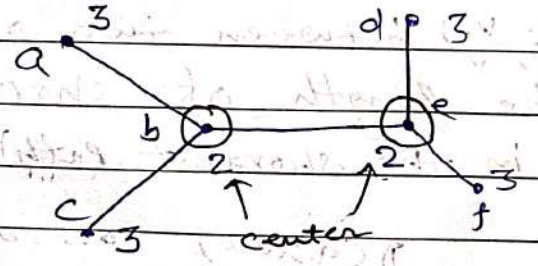
$E(b) = 1$ { from d to b }

$E(a) = 2$ { from a to d }

$E(c) = 2$ { from c to d }

⇒ Center ⇒ A vertex with minimum eccentricity value is called as center of graph G

In above example, minimum $E(b) = 1$



$E(a) = 3$ { from a to d or f }

$E(b) = 2$

$E(c) = 3$

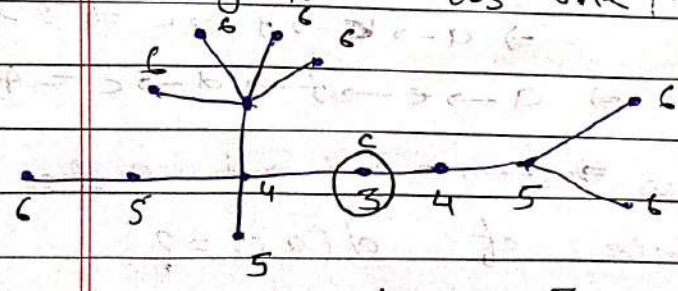
$E(d) = 3$ { from d to a or c }

$E(e) = 2$

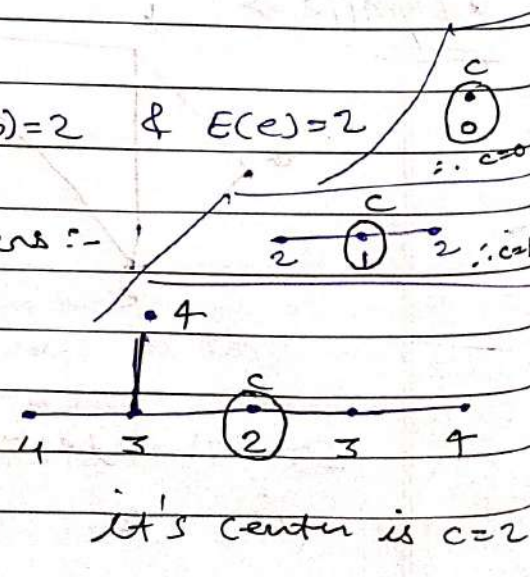
$E(f) = 3$

So two centers are $E(b) = 2$ & $E(e) = 2$

Every tree has one / two centers :-



∴ center is $c = 3$



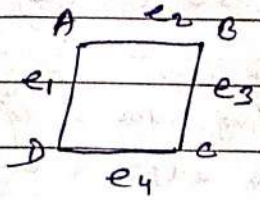
It's center is $c = 2$

* Spanning tree ⇒ Tree:- it is a graph having no cycle
↳ It is a connected subgraph of graph G(V, E).
In which all vertices are present.

→ If n is no. of vertice then (n-1) no. of edges.

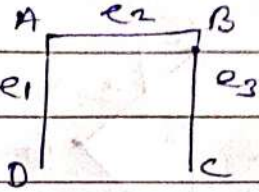
Graph G(V, E) ⟶ Subgraph G'(V', E')
All No. of vertices use but edges may be

Example

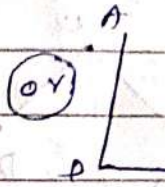


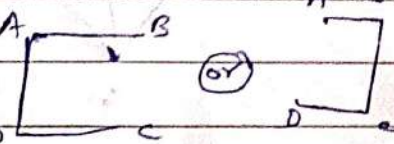
Graph G=(V, E)

Convert into spanning tree



(or)





No. of vertices = n = 4

No. of edges = n-1 = 4-1 = 3

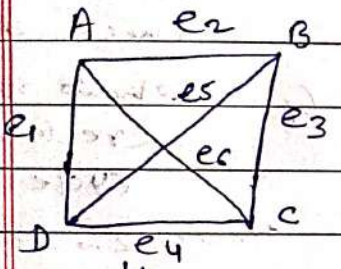
In complete graph:-

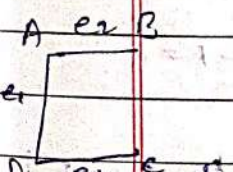
If no. of vertices is n then

No. of possible spanning tree = n^{n-2}

Ex:- n = 4 ∴ No. of possible spanning tree = $4^{4-2} = 4^2 = 16$

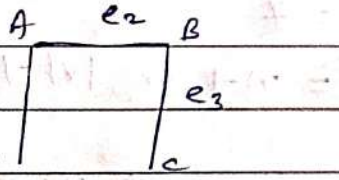
Q.





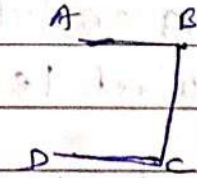
Used = {1, 2, 4}

Not used } = {3, 5, 6}



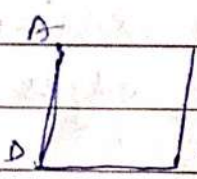
Used = {1, 2, 3}

NU = {4, 5, 6}



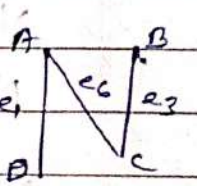
U = {2, 3, 4}

NU = {1, 5, 6}



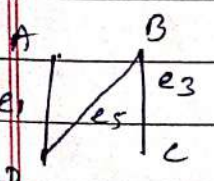
U = {2, 3, 4}

NU = {2, 5, 6}



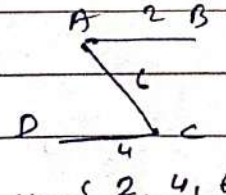
U = {1, 3, 6}

NU = {2, 4, 5}



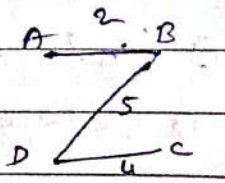
U = {1, 3, 5}

NU = {2, 4, 6}



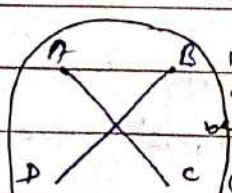
U = {2, 4, 6

NU = {1, 3, 5}



U = {2, 4, 5}

NU = {1, 3, 6}

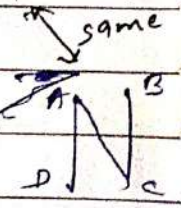


Not allow because edge not (n-1)

U = {5, 6}

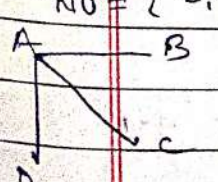
NU = {1, 2, 3, 4}

same



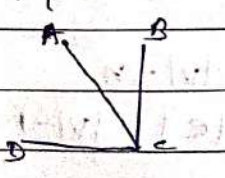
U = {1, 3, 6}

NU = {2, 4, 5}



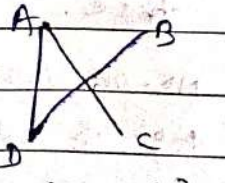
U = {1, 2, 6}

UN = {3, 4, 5}



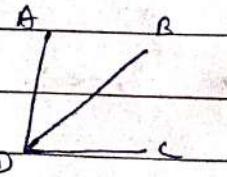
U = {3, 4, 6}

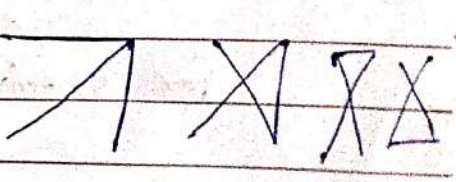
NU = {1, 2, 5}



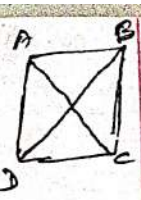
U = {1, 5, 6}

NU = {2, 3, 4}



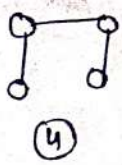


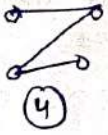
All possible spanning

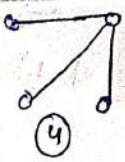


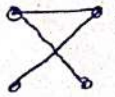
n = no. of vertices

n^{n-2} possible spanning tree









= 16 combinations

n = 4 ∴ $4^{4-2} = 4^2 = 16$

Date _____ Page _____ 84

Minimum cost spanning tree → Kruskal Algo. / Prim's algo.

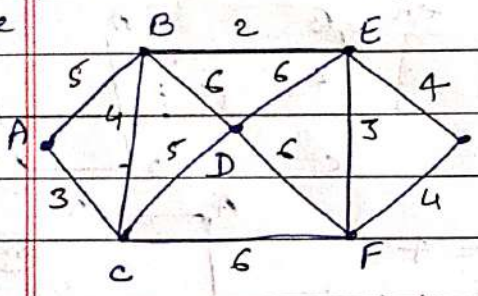
* Kruskal Algorithm

⇒ Construct minimum heap with e edges

⇒ Take one by one edge & add in spanning tree (cycle should not be created)

⇒ Best case (n-1) edge & worst case e edges

Example

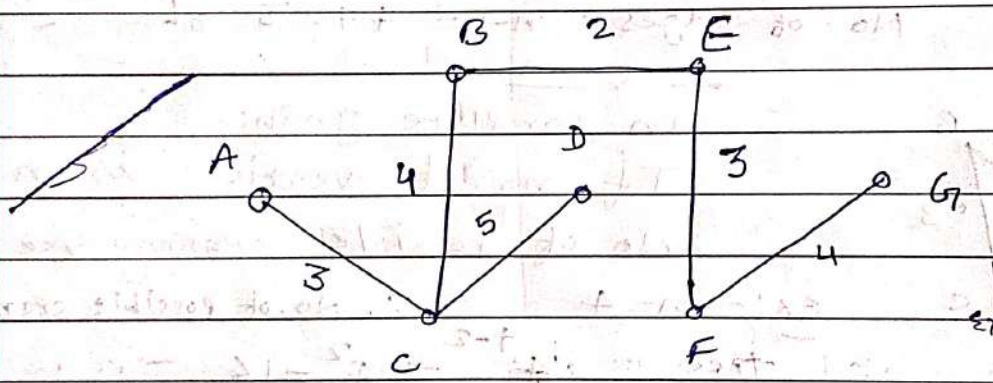


All cost edges arrange in increasing order & select from minimum edges & check cycle.

Start from select minimum weight, After continue & check should not be create loop cycle

Example 2 → 3 → 4 → 5

Sol^n =



No. of vertices n or |V| = 7

No. of edges required |e| = n-1 or |V|-1 = 7-1 = 6

Minimum cost (weight) spanning tree ⇒ 2+3+3+4+5 = 21

Time complexity O(t) = O(e) = 12

Best case ⇒ O(n log e)

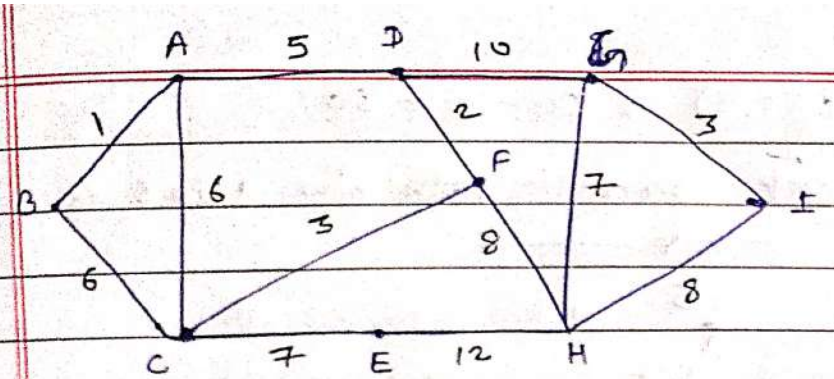
worst case = O(e log e)

---

* Prim's algorithm ⇒ To find minimum cost spanning tree.

for spanning tree: No. of vertices |V| = n

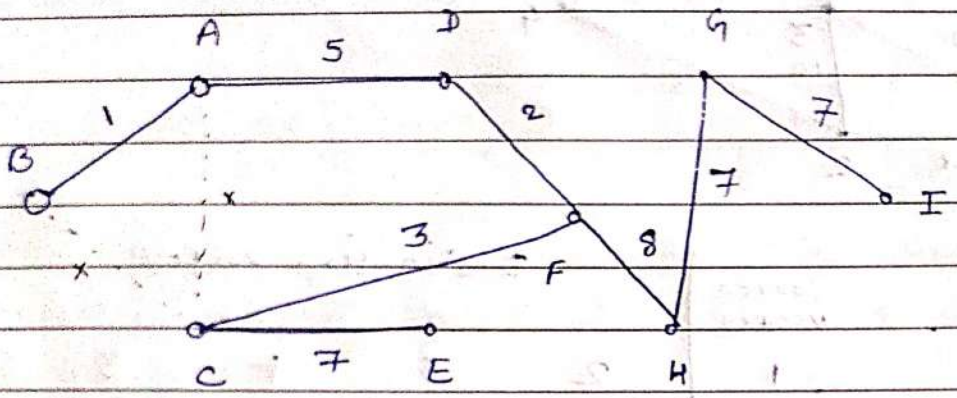
No. of edges |e| = |V|-1 or n-1



working steps ⇒

select any vertex

if select B node then check all connected edges with B node & select minimum cost/weight edge then reach A vertex again minimum edge with A & B



B → A → D → F → E → C → F → H → G → I

Minimum cost Spanning tree ⇒ BA + AD + DF + FC + CE + FH + HG + GI

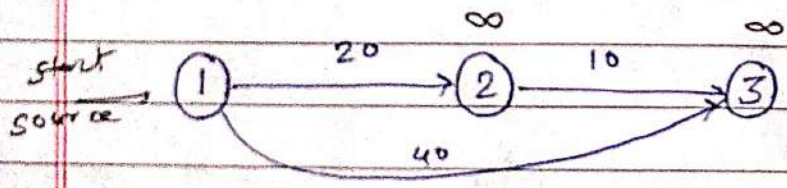
⇒ 1 + 5 + 2 + 3 + 7 + 8 + 7 + 7 = 40

Shortest Path

* Dijkstra's Algorithm (Single source shortest path) ⇒ it use greedy algorithm for (approach) optimal solution.

Relaxation ⇒ if $d(u) + c(u,v) < d(v)$ (c = cost)

$d(v) = d(u) + c(u,v)$



Explain

Start from vertex (1) it's source vertex

initially

(or) $d(u) = 0$ $c(u,v) = 20$ (or) $d(1) = 0$ & $c(1,2) = 20$

$d(\quad) = \infty$ $c(1,3) = 40$ $d(2) = \infty$

$\therefore$ $d(1) + c(1,2) < d(2)$ $\therefore$ $0 + 20 < \infty$

$20 < \infty$ it's true it's select 1 to 2 Path

$\therefore d(2) = 20$

$d(2) = 20$ $c(2,3) = 10$ $d(3) = \infty$

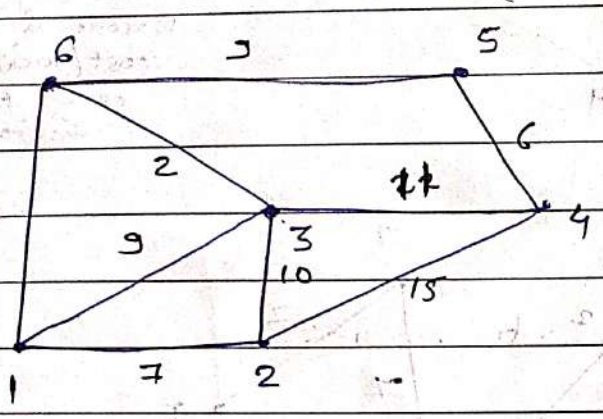
$\therefore$ $d(2) + c(2,3) < d(3)$ $30 < \infty$

$20 + 10 < \infty$ $\therefore$ select 2 to 3 Path.

$d(3) = d(2) + c(2,3) \Rightarrow 20 + 10 = 30$

∴ minimum cost shortest path b/w 1 to 3 is = 30

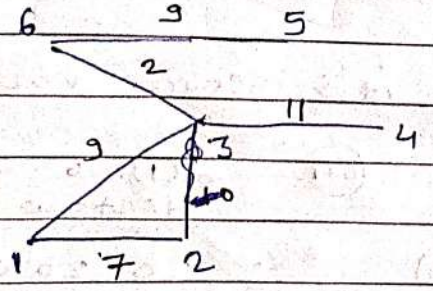
Q.1)



Source vertex	Destination vertex				
1	2	3	4	5	6
1	∞	∞	∞	∞	∞
0f cost ⇒ 1	(7) ←	9	∞	∞	14
1, 2	(7)	(9) ↑	22	∞	14
1, 2, 3	(7)	(9)	20	∞	(11) ↑
1, 2, 3, 6	(7)	(9)	(20) ↑	20	(11)
1, 2, 3, 6, 4	(7)	(9)	(20)	(20) ↑	(11)
1, 2, 3, 6, 4, 5	(7)	(9)	(20)	(20)	(11)

1 → 2 → ?

1 → 2 → 3 → ?



Minimum cost shortest (weight)

Path ⇒ 7 + 9 + 2 + 11 + 9

⇒ 38

Cut sets ⇒

Connectivity ⇒ A graph is not connected if it has two or more components.

(or)

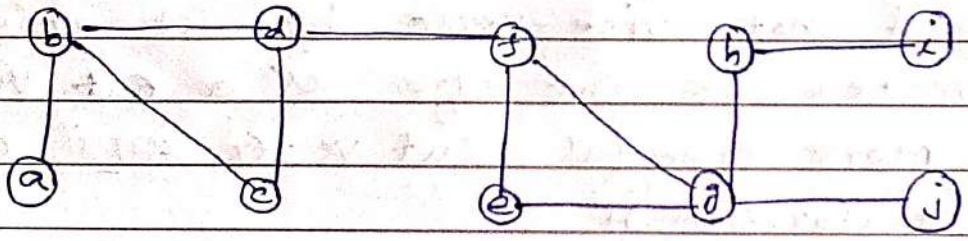
A Graph is said to be connected if there is a path b/w every pair of vertex. means traverse to every vertex is called connectivity.

* Cut vertex (Articulation point) ⇒ Let ~~Graph~~ G be a connected graph.

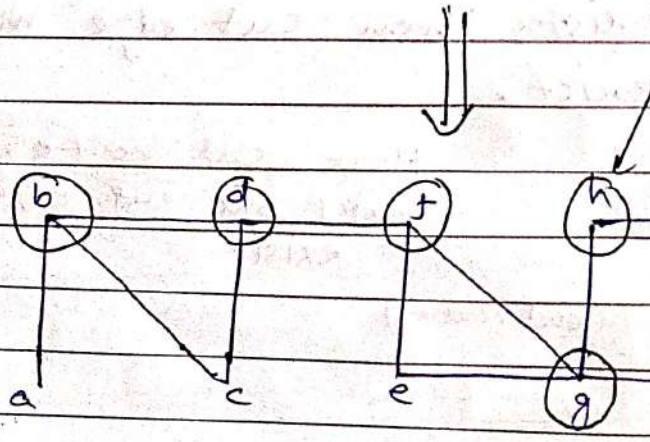
A vertex $V \in G$ is called cut vertex (Articulation point) of graph G if $(G-V)$ result in a dis-connected graph.

Note :- Removing a cut vertex may ~~render~~ render a graph dis-connected.

⇒ A connected graph G' may have at most $(n-2)$ cut vertices

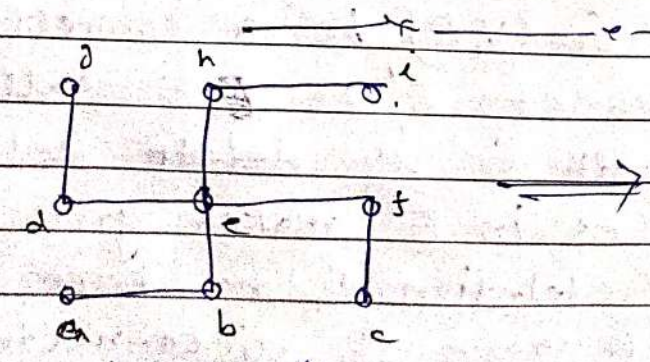


it's one component because it's connected.

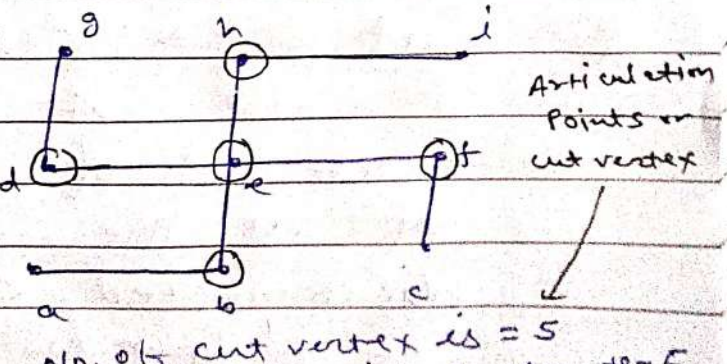


so five cut vertex

it's may be convert into 5 components



1 component



No. of cut vertex is = 5

No. of possible components = 5

Note:- If a graph not contain any cut vertex (articulation point) is called Bi-connected graph.

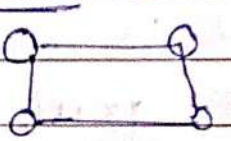
---

* Cut edge (Bridge) ⇒ If removing an edge in a graph then results into two or more graphs then these edge is called cut edge or Bridge. (or)

Let G be a connected graph & an edge $e \in G$ is called a cut edge of graph G if $(G-E)$ results in a dis-connected graph.

⇒ In a connected graph G if $E \in G$ is cut edge iff E is not part of any cycle in graph G

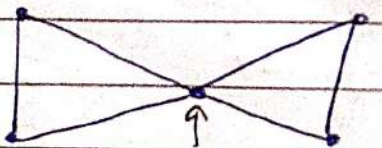
⇒



⇒ Whenever cut edge exist cut vertex also exists & not vice-verse because atleast one vertex of cut edge is a cut vertex

Note:- More powerful cut vertex होता है cut edge के comparision में.

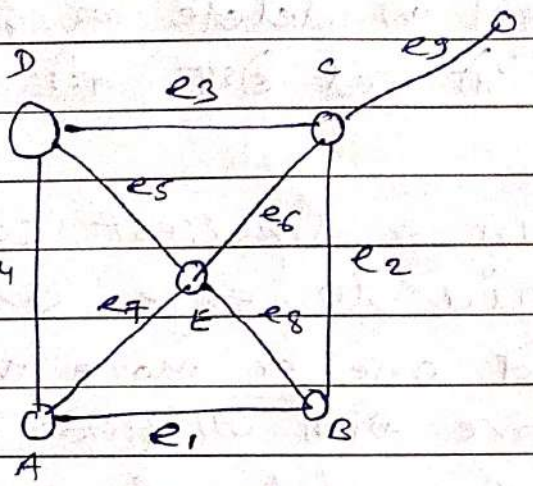
⇒ If cut vertex exists then cut edge may or may not be exist.



Here cut vertex is exist but cut edge is not exist.

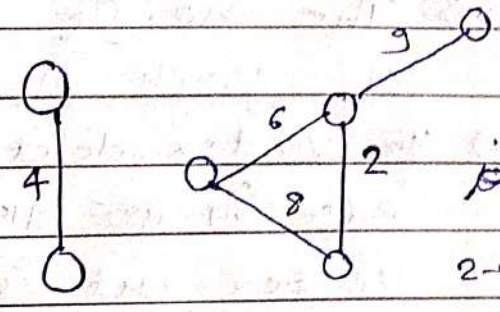
Cut set ⇒ Let $G=(V,E)$ be a connected graph. A subset E' of E called a cut set of graph G if deletion of all the edges in E' makes graph G disconnected & deletion of no proper subset of E' from graph G make G disconnected.

If deleting a certain number of edges from a graph makes it disconnected, then those deleted edges are called the cut set of the graph.



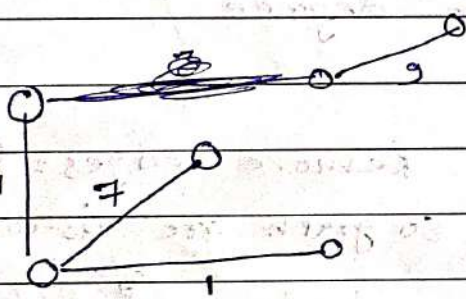
cut sets are :-

① $\{e_1, e_3, e_5, e_7\}$



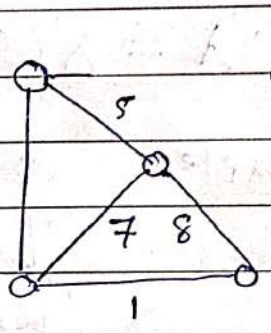
it's cut set 2-component

② $\{e_2, e_5, e_6, e_8\}$



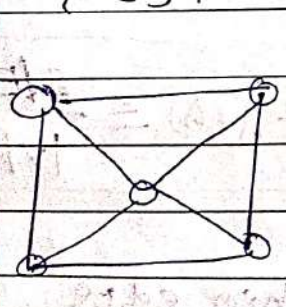
it's ~~not~~ cut set 2 component

③ $\{e_2, e_6, e_3, e_9\}$



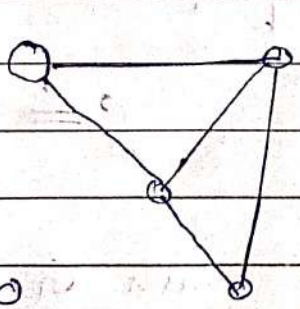
it's cut set ~~connected~~ graph in 3 component

④ $\{e_9\}$



it's cut set graph in two components

⑤ $\{e_1, e_4, e_7\}$



it's cut set graph in 3 components

~~Egge~~ Edge connectivity → In a connected graph G, The minimum no. of edges whose deletion makes graph G is disconnected is called Edge connectivity of graph G.

It is denoted by λ.

$\lambda \leq \frac{2|E|}{|V|}$

* **Vertex connectivity** ⇒ In a connected graph G, the minimum no. of vertices whose deletion makes a graph disconnected or reduces graph G into a trivial graph is called vertex connectivity of a connected graph G.
It is denoted by $K(G)$

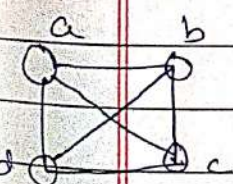
$$K(G) \leq \lambda \leq \delta \leq \frac{2|E|}{|V|}$$

⇒ If graph G has a cut vertex then $K(G) = 1$

Graph type	Edge connectivity (λ)	Vertex connectivity (K)
cycle graph - C_n	2	2
Wheel graph - W_n	3	3
Complete graph - K_n	$n-1$	$n-1$
Bi-partite graph - $K_{m,n}$	$\min\{m,n\}$	$\min\{m,n\}$
Start graph $K_{1,n-1}$	1	1

Explain :-

K_n

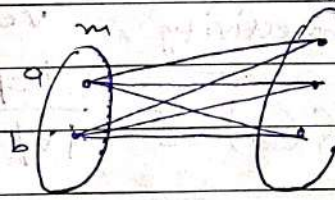


K_n

Vertex connectivity
$K(K_n) = 3$

Edge connectivity
$\lambda(G) = 3$

$K_{m,n} \Rightarrow K_{2,3}$

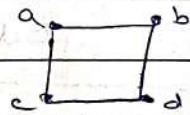


$K(G) = 2$

$\lambda(G) = 2$

$K \leq \lambda \leq 2$

$C_n \Rightarrow C_4$

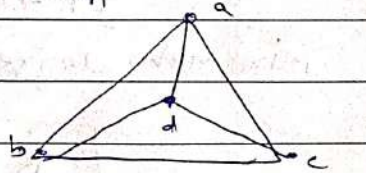


$K(G) = 2$

$\lambda(G) = 2$

$K \leq \lambda \leq 2$

$W_n \Rightarrow W_4$

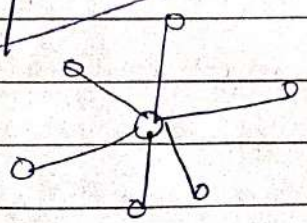


$K(G) = \frac{n+1}{3+1} = 4$

Edge $\lambda = 2(n-1)$
$= 2(4-1) = 2 \times 3 = 6$

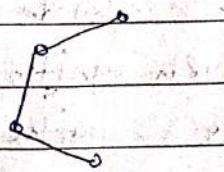
$\therefore K = 3$ & $\lambda = 3$

Star



$K(G) = 1$
$\lambda(G) = 1$

Tree with n vertices ($n \geq 2$)

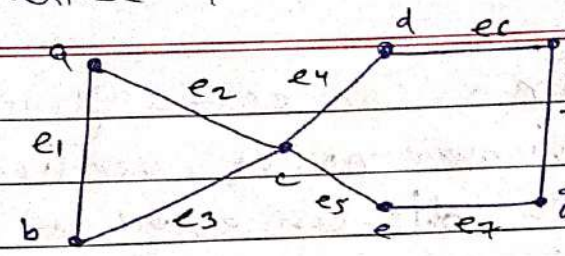


$K = 1$

$\lambda = 1$

Practice Questions

Q.1]



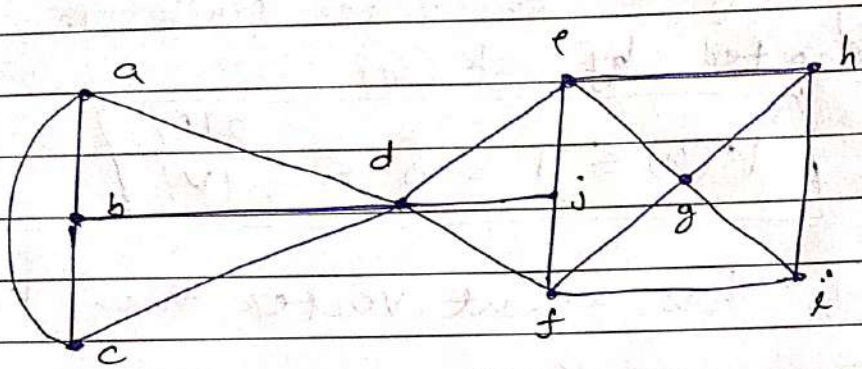
Vertex connectivity $K(G) = 1$ {c

Edge connectivity $\lambda(G) = 2$

{ e_2, e_3 (OR) e_4, e_5

any other combination may be

Q.2]



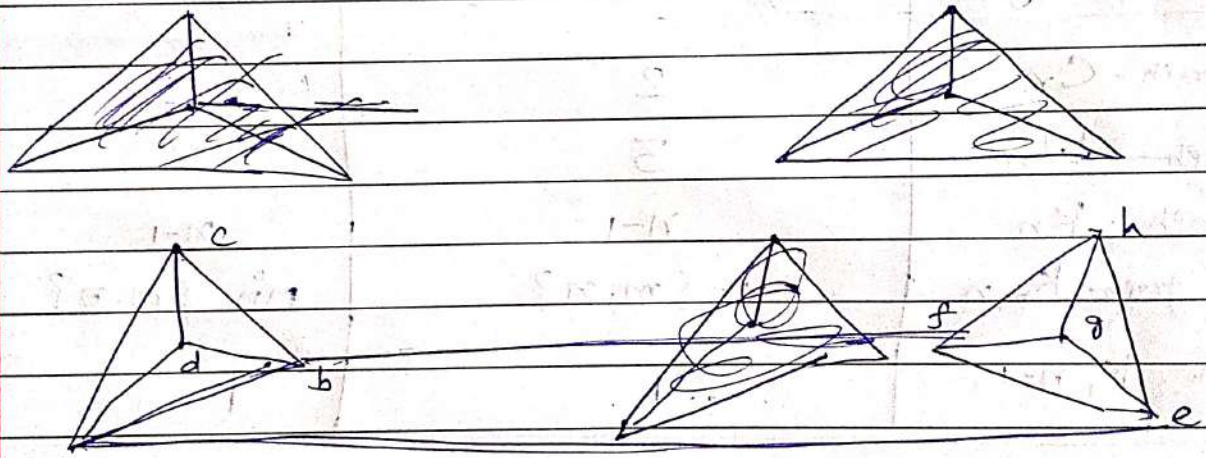
$\delta(G) = 3$

$1 \leq \lambda \leq 3$

$K(G) = 1$

$\lambda(G) = 3$

Q.3]



$\delta(G) = 3$ Minimum degree

$\lambda(G) = 2$ Edge connectivity

$K(G) = 2$ vertex connectivity

$$K(G) \leq \lambda(G) \leq \delta(G) \leq \frac{2|E|}{|V|}$$

Q.4] Which of the following graphs are necessarily connected ?

(a) A Graph with 6 vertices & 10 Edges

(b) A Graph with 7 V & 14 E

(c) A graph with 8 V & 22 E

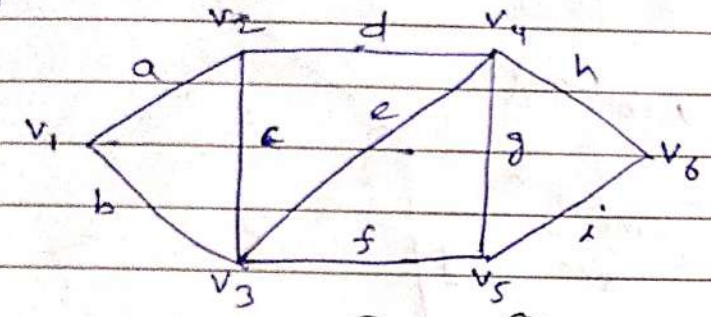
(d) A Graph with 9 V & 28 E

Basic cut sets (or)

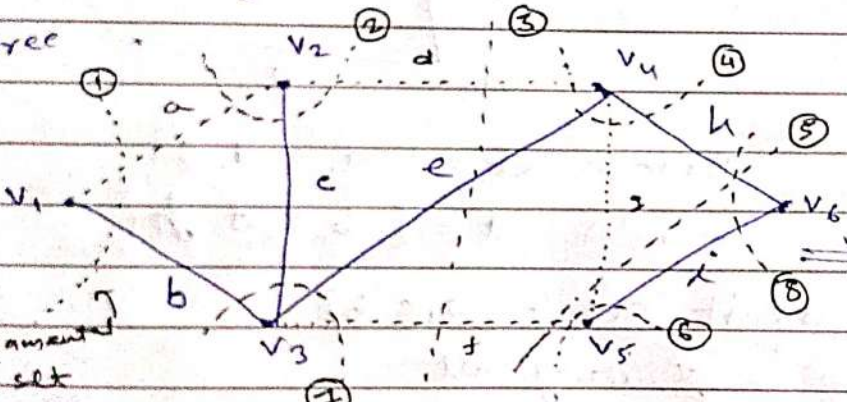
✗ fundamental cut sets ⇒

→ cut set Partitions all vertices into two disjoint sets.
→ cut set always contain only one Branch & rest of the edges are chords
→ such a cutset is called as fundamental cut sets or basic cut sets

Example



Arbitrary spanning tree



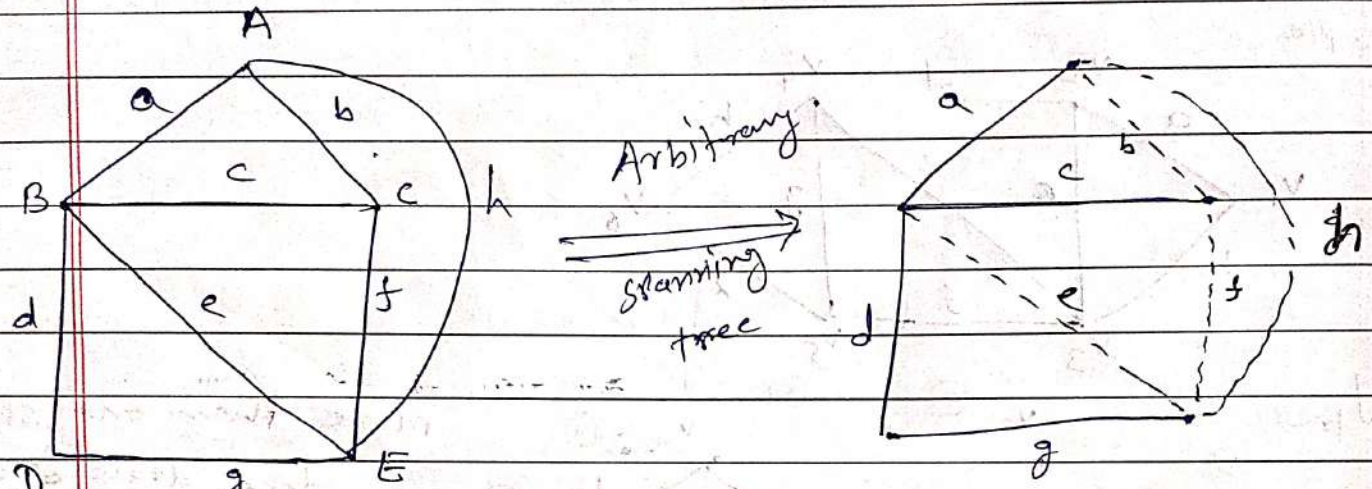
① fundamental cut set

more than one spanning tree draw ⇒ Branches set = { b, c, e, h, i }
chords set = { a, d, f, g }

① Fundamental cut set = { a, b } ⇒ { b-branch, a-chords set value }
② fundamental cut set = { a, c, d } ⇒ { c-branch, a & c chords }
③ fundamental cut sets = { d, e, f } ⇒ { e-branch & d, f chords
✗ ④ Not fundamental cut sets = { d, e, g, h } Here 2-branch e & h edges it's not allow only one branch allow.
⑤ fundamental cut sets = { f, g, h } ⇒ h-branch & f, g are chords
⑥ fundamental cut sets = { f, g, i } ⇒ i-branch & f, g-chords
✗ ⑦ Not fundament cut sets = { b, c, e, f } ⇒ more than one branch b, c, e & f-chords
✗ ⑧ Not fundamental cut sets = { h, i } ⇒ more than one branch h & i & no chords

∴ 5 fundamental cuts are ⇒ 1, 2, 3, 5, & 6

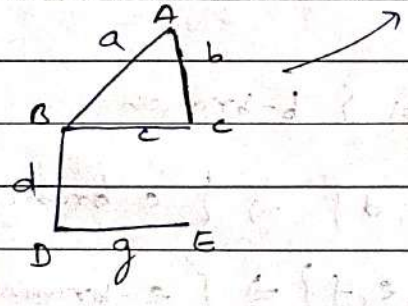
* **Fundamental Circuits** ⇒ Suppose G be connected graph & T be a spanning tree then a circuit formed by adding a chords to spanning tree T is called as a fundamental circuit.



Branch set = { a, c, d, g }

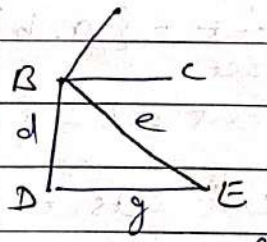
chords set = { b, e, f, h }

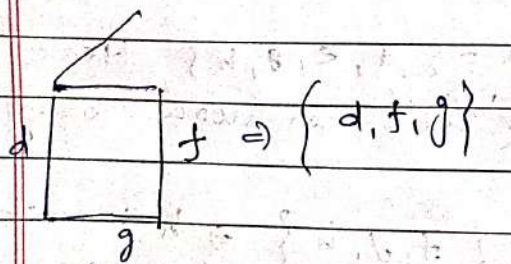
(1) fundamental circuit set = { a, c, b }



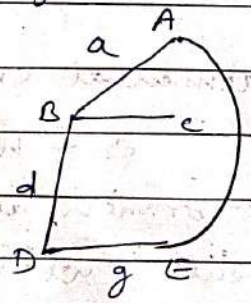
Similarly all

= { d, e, g }





⇒ { d, f, g }



⇒ { a, d, g, h }

How many fundamental circuits ⇒ If vertices = n edges ⇒ e & T = Spanning tree with (n-1) branches & exactly (e-n+1) chords

∴ $\boxed{e-n+1}$ fundamental circuits.

* Connectivity & Separability ⇒

Connectivity may be two types → Edge connectivity
→ vertex connectivity
Both already discuss in previous topics

⇒ Separability ⇒

A connected graph G is said to be separable if it's vertex connectivity is one (1). All other connected graphs are called non separable graph.

(OR)

Connected graph —vertex connectivity is one→ Dis-connected graph

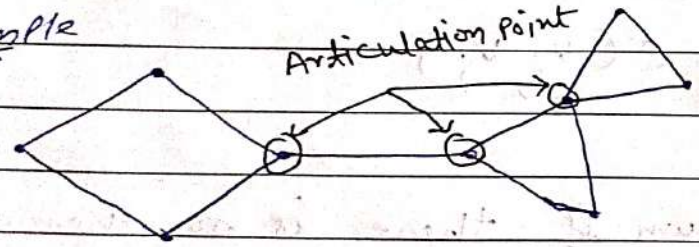
Graph G → subgraph G' → complement graph $\bar{G}'$

दोनों में only one vertex common होना चाहिये

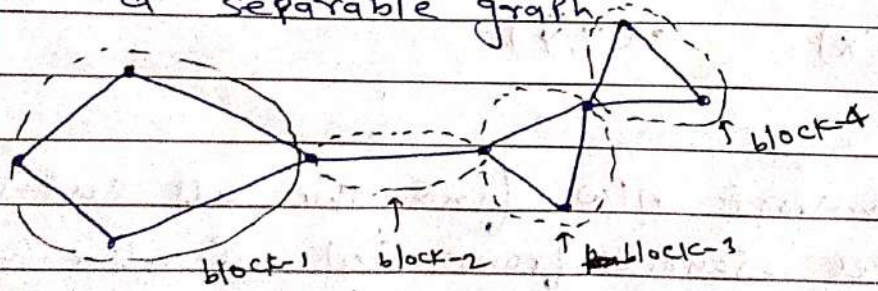
(OR)

A Graph G is said to be separable if it either dis-connected or can be disconnected by removing one vertex is called articulation point

Example



Block ⇒ Block is a largest non. Separable sub graph of a separable graph.

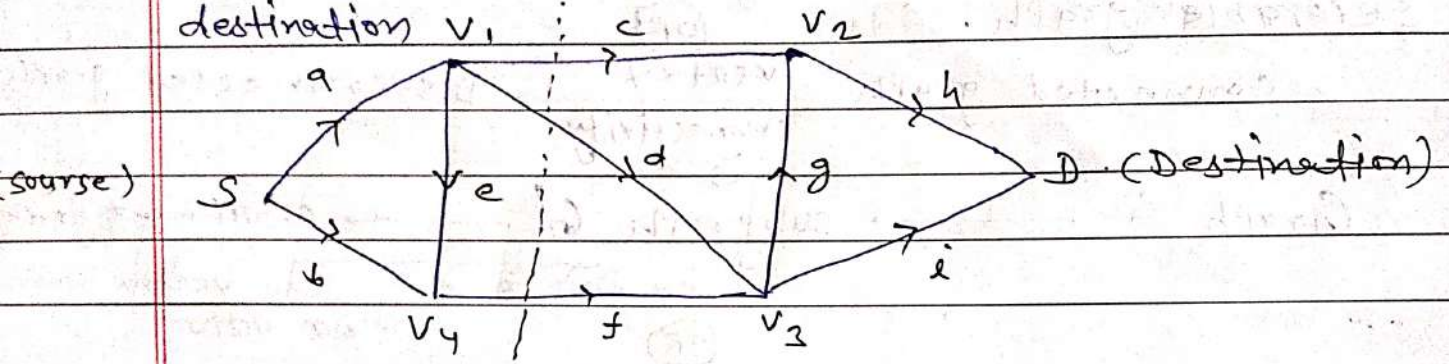


Separable graph with 3 cut vertices & 4 blocks.

* Cut set in a N/w flow & it's capacity ⇒

→ A cut set is a set of edges with separetes the source & sink which is obtain from n/w by according the direction of edges

→ It is denoted by $(X, \bar{X})$ Here X is contain the portion of sourse & $\bar{X}$ is contain the portion of destination



cut set = {c, d, f} remove edges c, d, f

Minimum-cut theorem ⇒

⇒ Capacity of the cut-set denoted by $c(X, \bar{X})$ (source ↓, destination ↓)

$$c(X, \bar{X}) = \sum_{\substack{v_i \in X \\ v_j \in \bar{X}}} c(v_i, v_j)$$

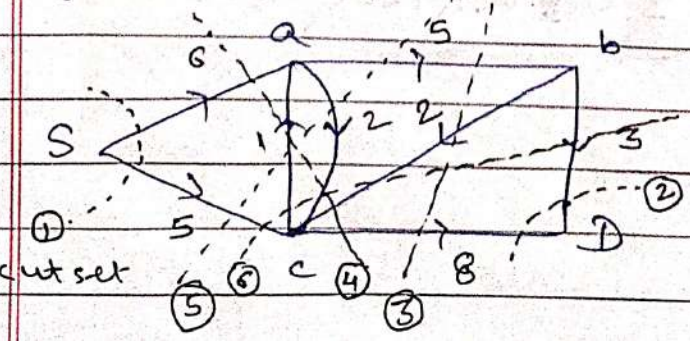
⇒ cut $(X, \bar{X})$ is minimum if there is no other cut set

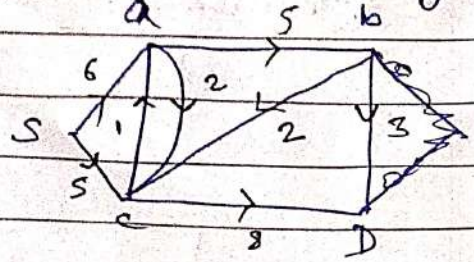
Example $(X, \bar{X})$ in the N/w such that

$c(X, \bar{X})$ $c(Y, \bar{Y})$

Example

Q.1) In given transport n/w, find the all cut-sets that separates source from sink & also we have find out the cut-set with minimum-capacity





cut set NO ↓ | Source ↓ | Destination ↓ | Capacity ↓

S.No.	X	$\bar{X}$	cut $(X, \bar{X})$
1	{S}	{a, b, c, D}	(11) } Minimum capacity
2	{S, a, b, c}	{D}	(11) } Minimum capacity
3	{S, a, c}	{b, D}	13
4	{S, c}	{a, b, D}	15
5	{S, a}	{b, c, D}	12
6	{S, a, b}	{c, D}	12

cut-set-(1)

$C(S, a\&c) = (S, a) + (S, c) = 6 + 5 = 11$

Move Destination to source because

cut set-(2)

$(b, D) + (c, D) = 3 + 8 = 11.$

cut-set-(3)

$(a, b) + (b, c) + (c, a) = 5 +$ not consider $+ 8 = 13$

cut-set-(4)

$= (S, a) + (c, a) + (a, c) + (b, c) + (c, D)$

$= 6 + 1 +$ not consider + Not consider $+ 8 = 15$

cut-set-(5)

$(S, c) + (c, a) + (a, c) + (a, b) = 5 +$ Not consider $+ 2 + 5 = 12$

cut-set-(6)

$(S, c) + (c, a) + (a, c) + (b, c) + (b, D)$

$= 5 +$ Not consider $+ 2 +$ ~~Not~~ $2 + 3 = 12$

~~There for~~ Minimum capacity ⇒

There are two cut-sets which having minimum capacity is 11

(1) {S} & {a, b, c, D} (2) {S, a, b, c} & {D}

* Maximum flow minimum-cut theorem ⇒ In a given transport n/w, the max-flow min-cut theorem states that the maximum flow through any n/w from a given source to a given sink is exactly equal to the minimum sum of cut.

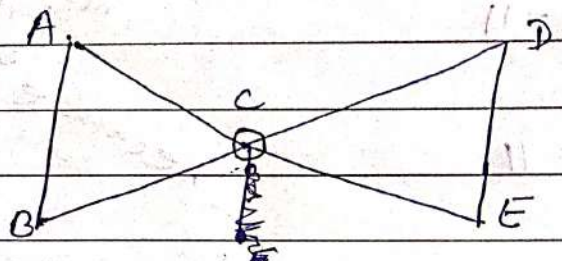
$$\text{Flow}_{max} = \min \text{capacity}(X, \bar{X})$$

* 1-Isomorphism & 2-Isomorphism ⇒

Operation-1 for (1-Isomorphic) ⇒ Articulation point

Split a cut vertex into two vertices to produce two disjoint sub graph

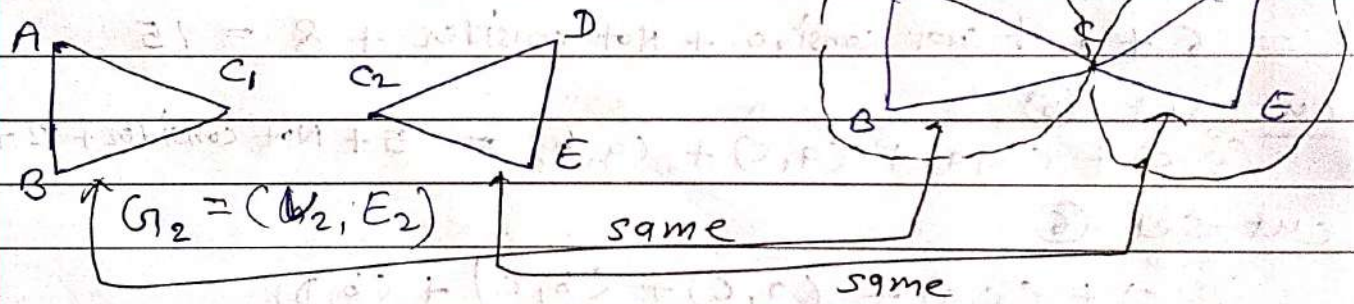
Note:- 1-Isomorphic graph वही graph होते हैं जो separable graph है मतलब ऐसे graph जिसकी vertex connectivity one हो।



It's separable graph because if remove C vertex then it makes dis-connected graph

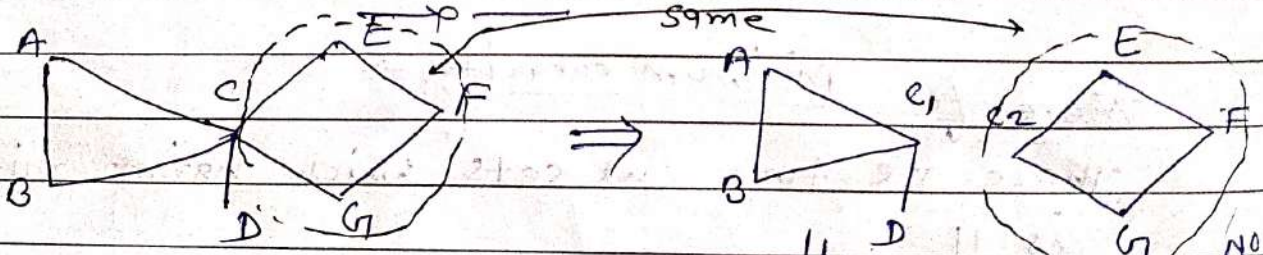
Divide cut-vertex

$G_1 = (V_1, E_1)$



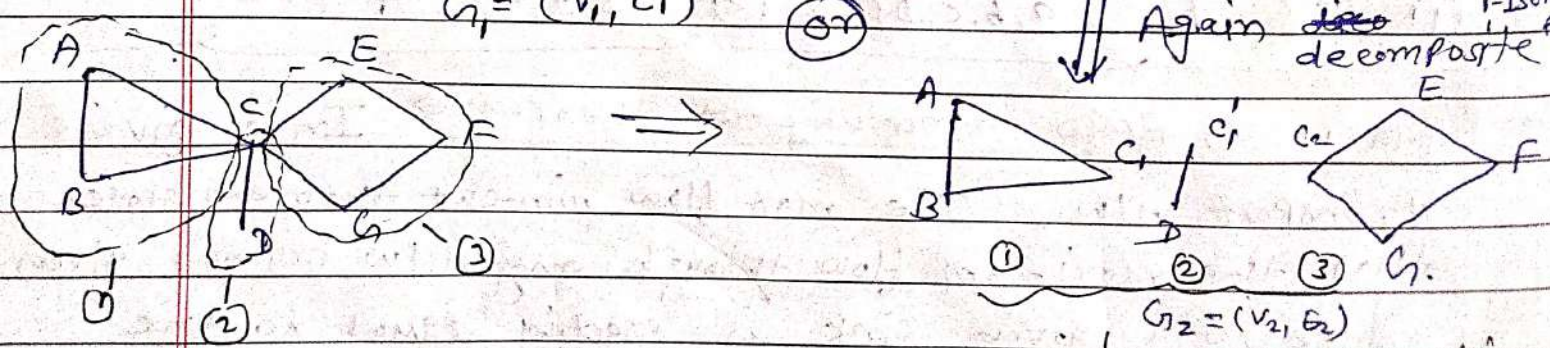
So it's called 1-Isomorphic graph

Example



$G_1 = (V_1, E_1)$ (or)

Again decompose

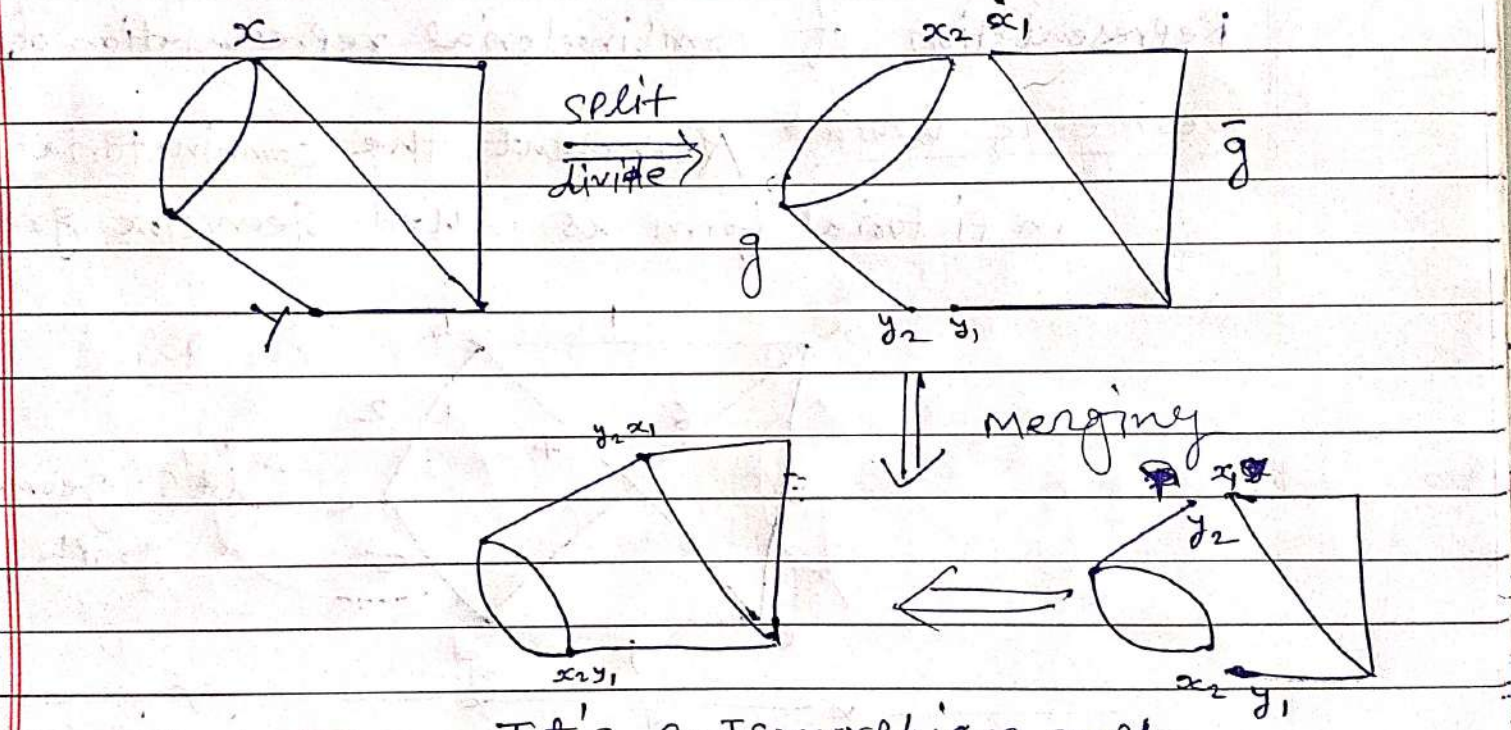


$G_2 = (V_2, E_2)$

It's 1-Isomorphic

2-Isomorphism ⇒ It's extend form of 1-Isomorphism

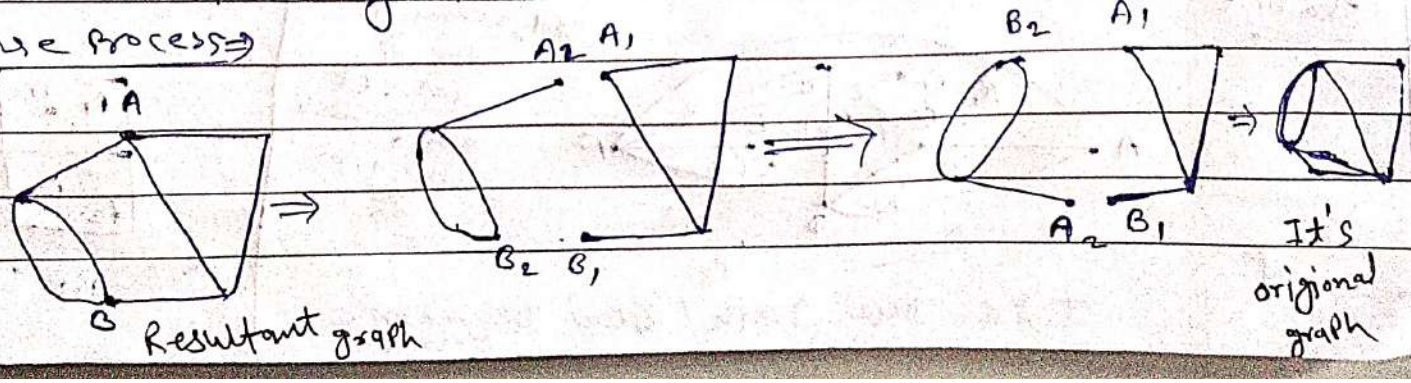
Operation-2 for 2-Isomorphic ⇒ split the vertex x into x_1 & x_2 & the vertex y into y_1 & y_2 such that graph G is split into g & $\bar{g}$ or (G & $\bar{G}$).
Let vertices x_1 & y_1 go with g & x_2 & y_2 with $\bar{g}$. Now rejoin the graph g & $\bar{g}$ by merging x_1 with y_2 & x_2 with y_1.



It's 2-Isomorphism graph

Note :- Isomorphic होने के लिये या तो operation-1 or operation-2 या दोनों operation को satisfy करे अगर यही process जो ऊपर कि हुई हो, को मैं n times apply करता हु तो जो final 2-Isomorphic graph हो वह हमे फिर से original graph मे convert हो जायेगा तो इसे हि 2-Isomorphic graph कहते हो

Reverse process ⇒



Combinatorial & Geometric graph (or)

Dual graph

* Combinational & Geometric Graph ⇒

An abstract graph $G = \{V, E, \phi\}$ Here V is set of vertices, E is set of edges & ϕ is relation b/w vertices & edges

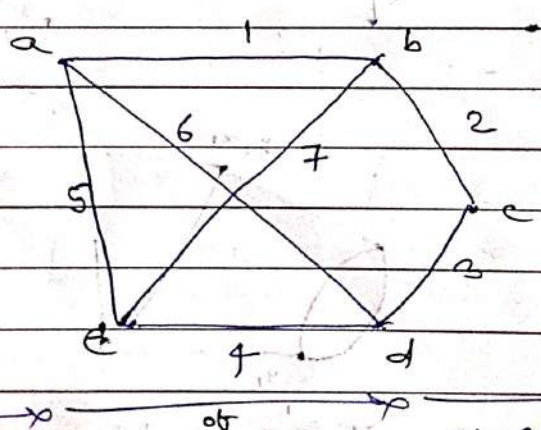
Combinatorial graph is a collection of $(G = \{V, E, \phi\})$

$V = \{a, b, c, d, e\}$ $\quad E = \{1, 2, 3, 4, 5, 6, 7\}$

$\phi = \{1 \to \{a,b\},\ 2 \to \{b,c\},\ 3 \to \{c,d\},\ 4 \to \{e,d\},\ 5 \to \{a,e\},\ 6 \to \{a,d\},\ 7 \to \{b,e\}\}$

Representation of combinatorial representation of any graph

Geometric Graph ⇒ Represent the combinatorial graph in pictorial form is called Geometric graph

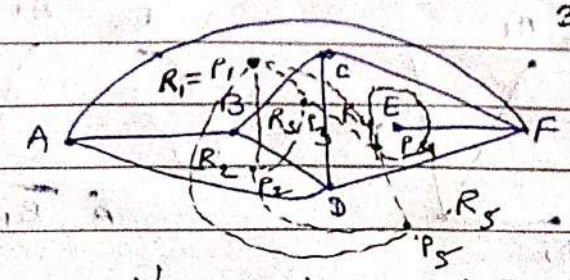


It's geometric graph

* Dual graph / Geometric Dual graph ⇒ ~~Steps:~~ Working steps :-

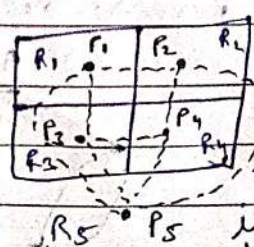
(1) Identified all regions in a planar graph (2) All regions replaced by points (3) Draw (joint) two adjacent edge (adjacent region / common edge region) points ~~to each other~~ Similar process for all points (4) If any region contains pendent vertex then draw a self loop for this vertex

Example



All common edge को intersect करते हुए desh line draw करें

It's Dual graph / Dual Geometric graph.



It's Dual graph