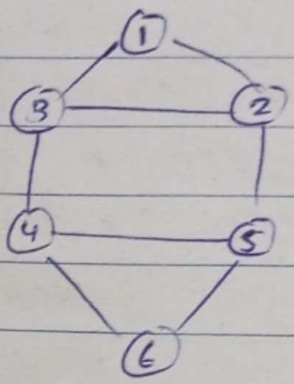
Date

Day

"DISTANCE, RADIUS, DIAMETER & ECCENTRICITY"

• **Distance :** The distance b/w two vertices in a graph is no. of edges in a shortest path.

$d(1,2)=1$	$d(2,3)=1$	$d(3,5)=2$
$d(1,3)=1$	$d(2,4)=2$	$d(3,6)=2$
$d(1,4)=2$	$d(2,5)=1$	$d(4,5)=1$
$d(1,5)=2$	$d(2,6)=2$	$d(4,6)=1$
$d(1,6)=3$	$d(3,4)=1$	$d(5,6)=1$



• **Eccentricity :** Ek vertex 'x' ka maximum distance of a vertex $\epsilon(x) = \max\limits_{y \in V(G)} d(x,y)$

$\therefore$ $\epsilon(1)=3$, $\epsilon(2)=2$, $\epsilon(3)=2$, $\epsilon(4)=2$, $\epsilon(5)=2$
$\epsilon(6)=3$

maximum distance b/w two vertices

• **Diameter of Graph :** Maximum of ecentricity of the vertices. $\text{diam}(G) = \max\limits_{y \in V(G)} d(x,y)$ or $\epsilon(y)$

$$\text{diam}(G) = 3$$

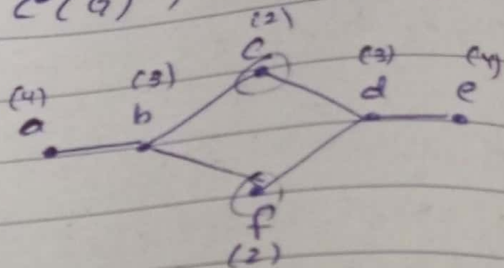
• **Radius of Graph :** Minimum of the all eccentricities

$$\text{rad}(G) = \min_{x \in V(G)} \epsilon(x)$$

$\text{rad}(G)=2$ Page No.

Date Day

→ A vertex w in G is called a central vertex if $e(w) = rad(G)$ and the center of G, denoted $C(G)$, is the set of all centeral vertices



$diam(G) = 4$

$Rad(G) = 2$

$C(G) = \{c, f\}$

↳ centeral vertices, $e(c) = e(f) = rad(G) = 2$

Theorem 2.26: If G is disconnected then $\overline{G}$ is connected & $diam(\overline{G}) \leq 2$

Theorem 2.27: For a simple graph G if $rad(G) \geq 3$ then $rad(\overline{G}) \leq 2$.

Theorem 2.28: For any simple graph, ~~and~~ $rad(G) \leq diam(G) \leq 2\,rad(G)$

Girth: Minimum length of a cycle in G, denoted $g(G)$

Circumference: Max length of a cycle.

↳ the circ—nce of a graph can be atmost n.

↳ if G doesnot have cycle (tree graph) then $g(G) = \infty$ and circum $= 0$

Date ______ Day ______

Theorem 2-31: If G is a graph with at least one cycle the $g(G) \leq 2\,diam(G) + 1$

Theorem 2-32: Let G be a graph with n vertices, radius at most k, and max degree at most d, with $d \geq 3$, then

$$n < \frac{d}{d-2}(d-1)^k$$

Day

"TREES"

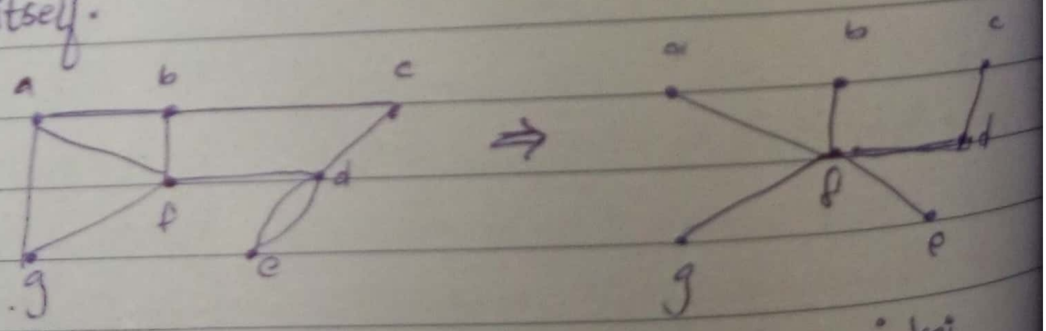
↳ A graph is called tree if it's connected and has no cycles

"A graph G is a tree iff every two vertices in G are joined by unique path."

→ acyclic → a graph which has no cycles/circuits
→ tree → a tree if it is both acyclic & connected.
→ forest → a graph which is acyclic ?
↳ a forest is simply a graph in which every component itself is a tree

→ Vertex of degree 1 is called leaf.

→ Spanning Tree:
- A spanning tree is a spanning subgraph that is also a tree.
- Tree has exactly one spanning tree; the full graph itself.



Spaning mein saare vertices include krna zaroori hai

Date

Day

Q: Under what conditions will a graph have a spanning tree? Clearly the more difficult criteria is the tree, not spanning, since every graph contains a spanning subgraph.

A: The existence of a spanning tree of a graph G is directly linked to the connectedness of G.

"Let G be a graph, then G is connected iff G contains a spanning tree."

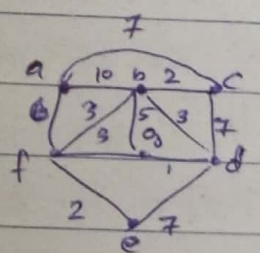
Minimum Spanning Tree:

The MST of $G = (V, E, \omega)$ is the Spanning Tree with the least total weight.

Kruskal Algo:

① Ek ek karke sabse minimum weight waali edge choose krte rhenge.

② Agr kisi edge se cycle/circuit ban rha ho to use choose nhi krenge.



Prims Algo:

① Kisi bhi vertex ko choose krlenge

For eg: a

② Ab 'a' se minimum edge choose krenge i.e af (6).

Date Day

(3) Now again choose minimum edge but this time we will consider the adjacent edges of both a & f.
We choose fe (2)
(4) Now considering min from the adjacent edges of a, f & e. We choose fg (3)
(5) We will not choose an edge which cause cycle.
(6) this goes on until we ~~find~~ ~~an~~ didn't find an edge

"PROPERTIES OF TREE"

"a tree with 'n' vertices has $n-1$ edges $\forall\, n \geq 1$"
Proof Using Mathemetical induction:

- At $n = 1$:
  At $n=1$ (a tree with no edges). So no. of edges is $1-1 = 0$. Correct.

- At $n = k$:
  For tree with $n=k$ has $k-1$ edges.

- At $n = k+1$
  If we add one more vertex to a tree with k edges, we must also add an edge to make it connected.
  Now the tree has $k+1$ vertices & no. of edges increase by one, so $(k+1) - 1 = k$

Date   Day

Therefore a tree with $k+1$ vertices has k edges.

"Every tree with at least 2 vertices has a leaf"

Proof: Because a tree is always connected and have not a cycle. So if there aren't any vertices of degree 1 (i.e leaf vertice) then there would always be a cycle. And tree cannot have a cycle.

"The total degree of a tree on n vertices is $2n-2$."

The sum of all degrees or total degree in a graph = 2 (no. of edges) = $2n$

In a tree no. of edges $= n-1$

So total degree $= 2(n-1) = 2n-2$.

→ Every tree is minimally connected, that is removal of any edge dissconnects the graph.

→ Every edge of a tree is bridge.

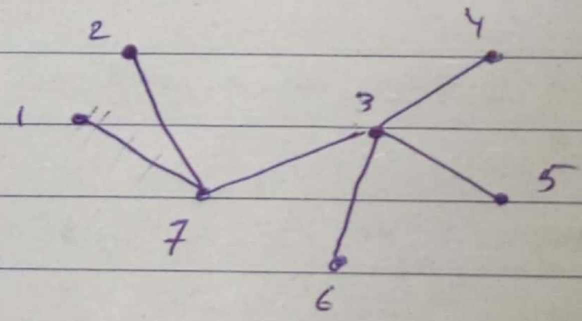
Enumerated Tree :

→ A Tree labelled with 1, 2 .. n is called enumerated tree.

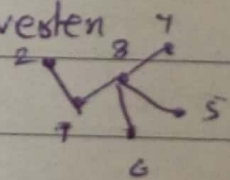
Prüfer Code / Sequence :

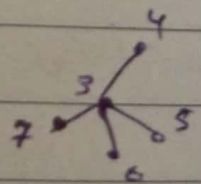
→ Sabse smallest label wala ~~vertex~~ leaf delete karenge or use sath jo adjacent vertex hoga we side mein likh denge.

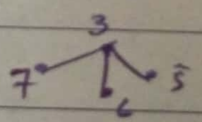
→ Phir 2nd smallest & so on....

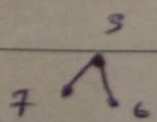


leaf	adjacent vertex
1	7
2	7
4	3
5	3
6	3
7	3

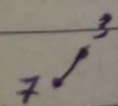








→ yaha par stop



Prüfer Code → (7, 7, 3, 3, 3)

From Prüfer code we can also draw a tree.

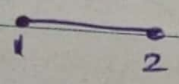
Prüfer Code → (1, 1, 1, 1, 6, 5)

No. of enteries in tree = No. of enteries in Prüfer Code + 2

= 6 + 2 = 8

→ We choose the 1st no. in the list & connect it with the smallest no. not included in list.

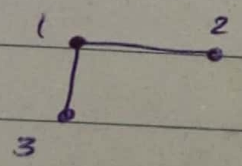
Choose 1 ; & connect with 2.



Now drop the number we choose i.e 1 and add the smallest no. that is 2.

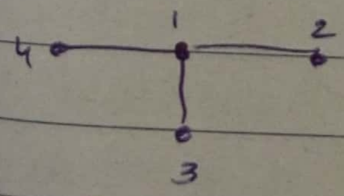
So sequence becoms (1, 1, 1, 6, 5, 2)

Choose 1 , smallest : 3



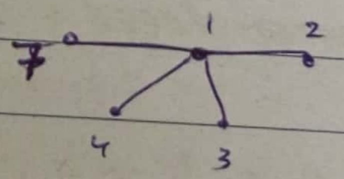
(1, 1, 6, 5, 2, 3)

Choose : 1 , smallest , 4



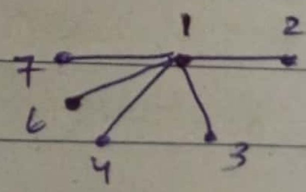
(1, 6, 5, 2, 3, 4)

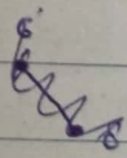
Choose : 1 , Smallest : 7



(6, 5, 2, 3, 4, 7)

Choose : 6 , smallest : 1

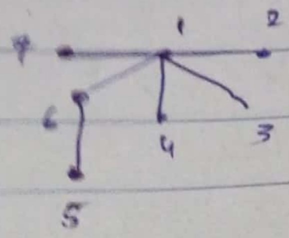




Date   Day

(5, 2, 3, 4, 7, 1)

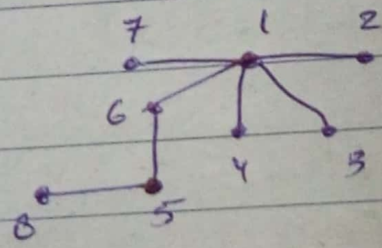
Choose : 5 , Smallest : 6

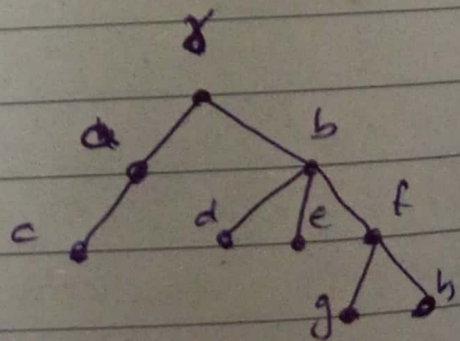


(2, 3, 4, 7, 1, 6) → Now we have iterated all the no.s that were originally in the code. Therefore we stop.

Now we check which 2 numbers are missing & then connect them.

In this case they are 5 & 8





▷ ROOTED TREES :

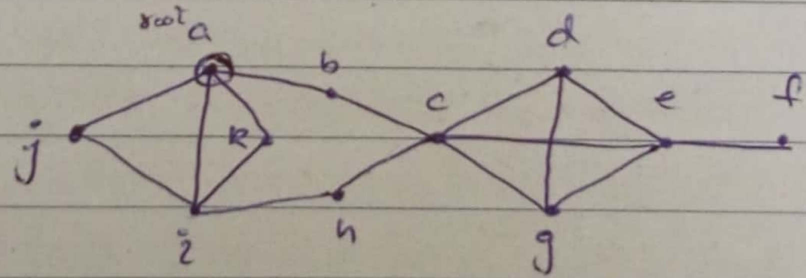
- Level of a, b → 1
- level of c, d, e, f → 2
- level of g, h → 3
- level of r → 0
- Height of tree = 3

Date   Day

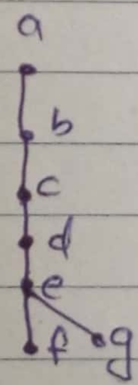
→ For b → f, g, h are $\boxed{\text{descendant}}$
→ For b → f is $\boxed{\text{child}}$
→ For g → b, r, f are $\boxed{\text{ancestor}}$
→ for g → f is $\boxed{\text{parent}}$
→ for g → h is $\boxed{\text{Sibling}}$

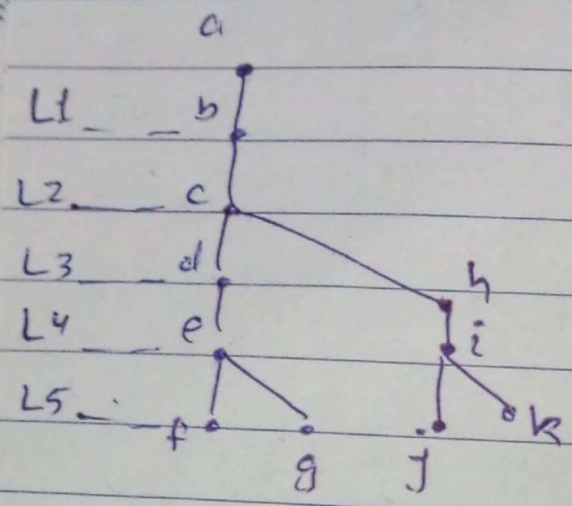
→ A tree having at most two children is called binary tree
→ If every parent has exactly two children we have full binary tree
→ If every vertex has at most k-children the k-nary tree

▷ Depth First Search Tree :



· We choose root vertex.
· Then choose ~~any~~ first neighbour i.e b.
· Now choose 1st neighbour of b i.e c
· Now choose d then e, then f
· Ab backtrack krke peeche ayenge or check krenge k koi unvisited vertex to nhi.
· 'e' par hum g tak bhi jaa sakte hai
· And so on.



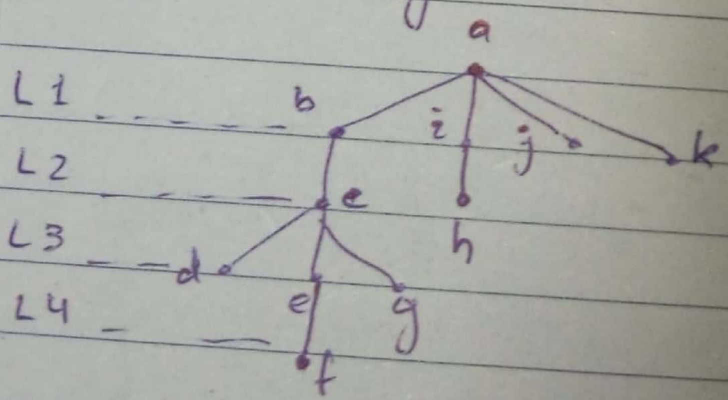


Height = 5

▷ BFS Tree :

For the same graph.

1. 'a' k saare neighbours likh denge.
2. Phir b, i, j, k k saare neighbours that are not already included



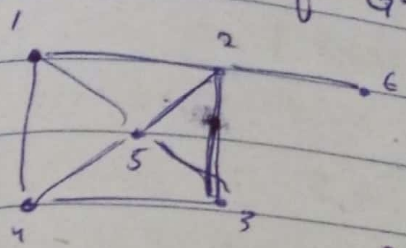
Height = 4

Date   Day

" Connectivity And Flow "

→ A cut-vertex of a graph G is a vertex 'v' whose removal disconnects the graph.

→ A set S of vertices within a graph G is a cut-set. if G-S is disconnected.



~~{2, 5}~~

cutset { {2}, {1, 5, 3} }

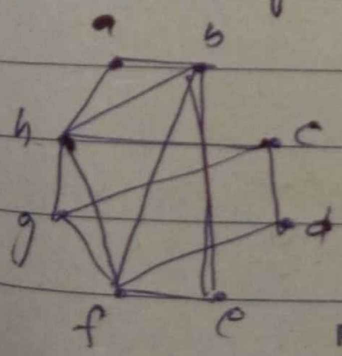
{~~4~~, 5, 2} → ye cutset nhi hosakta because agr hum 5 k bajae sirf 2 bhi hatadein to bhi dissconect hojayega graph.

→ K- connected, A graph is k-connected if the smallest cut-set is of size "k".

Jese uper wale graph mein kaafi saare cutset possible hai but sabse least wala is {2} so k = 1. Uper wala graph is 1-connected

Connectivity of G, K(G) = k, to be maximum 'k' such that G is k-connected.

If Graph is 3-connected then there cannot be a cut set of size 2 or less in graph.



K(G) = 2

cutset {b, h}

→ this graph is 1-connected as well as 2-connected

Page No. [ ]

Date   Day

→ A bridge in a graph $G = (V, E)$ is an edge E whose removal disconnects the graph.

→ An edge-cut is a set F where $F \subseteq E$ such that $G - F$ is disconnected.

→ We say G is k-edge-connected if the smallest edge-cut is of size at least k.
Denoted by $K'(G) = k$.
For previous graph $K'(G) = 2$.

▷ Whitney's Theorem :

minimum degree

$$K(G) \leq K'(G) \leq \delta(G)$$

→ Let P_1 & P_2 be two paths within the same graph G. We say these paths are :

↳ disjoint if they have no vertices or edges in common.

↳ internally disjoint if they only common vertices are starting & ending.

↳ edge-disjoint if the have no edges in common

Date   Day

→ Let x & y be two vertices in a graph G. A set S (of either vertices or edges) seperates x & y in different components of G-S. Then S is called Seperating set.

"A vertex v is a cut-vertex of a graph G iff there exists vertices x & y such that v is on every x-y path"

A vertex V is a cut vertex if removing it from the graph disconnects some part of the graph. This means that there exist vertices x & y such that every path b/w x & y must pass thru v. If v is removed, there will be no way to travel b/w x & y making v essential for mantaining the connection between them.

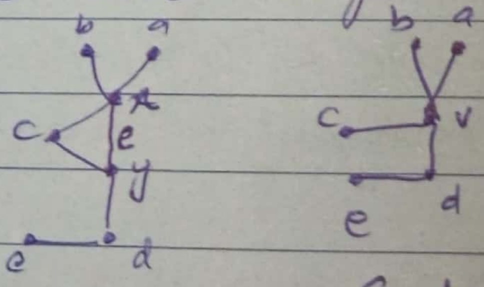
"An edge e is a bridge of G iff there exist vertices x & y such that e is on every x-y paths."

"Every non-trival connected graph contains at least two vertices that are not cut-vertices"

In any connected graph with more than one vertex, there are always at least two vertices that are not cut-vertices. This is bcz in the connected graph, the end vertices of the longest path (also called leaf nodes) can't be cut-vertices. Removing them won't disconnect the graph. Hence, a non trival connected graph connected graph always has at least two such vertices.

- **Contraction of Graph:**

Agr koi edge hai i.e $e = xy$ to us edge ko hata kr we make a new vertex for suppose v and jo jo vertices x ya y se connected hai unhe v se connect krdenge.



G_4 $\qquad$ G_4/e

Contracting of an edge creates a smaller graph

Date

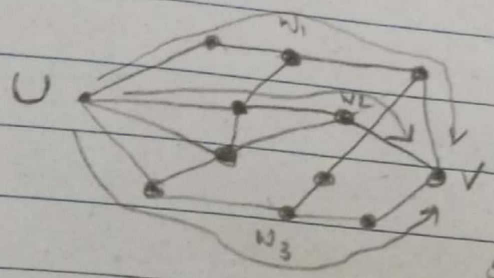
minimum no. of vertices needed to remove b/w U & V to divide them into diff components

Day

- **Menger's Theorem:**

Let U & V be non-adjacent vertices in a graph G. The minimum no. of vertices in a U-V seperating set equals the max no. of internally disjoint U-V paths.

esa path U-V k darmian jismein ~~U&V ke ilawa~~ koi common vertice na ho.



max 3 internally disjoint paths

~~So~~ min no. of seperating set vertices = 3

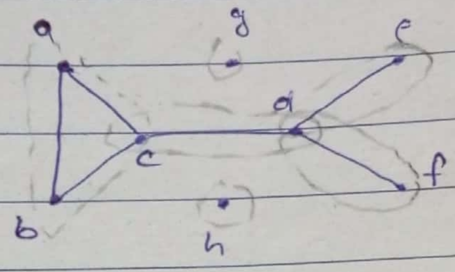
- **Theorem 4.17:**

"A grap G with a least 3-vertices is 2-connected iff G is connected & doesnot have any cut-vertex"

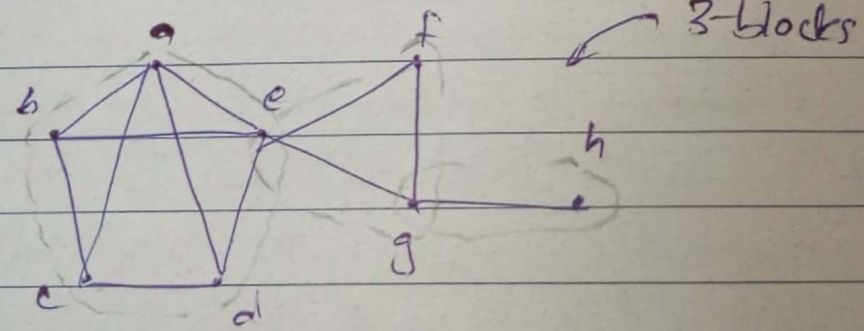
Proof: Assume G is 2-connected. Then any cut set of G must be of size at least 2. Therefore G must be connected & cannot have a cut-vertex, as in either of these situation we would have a cut set of less than 2.

Date   Day

▷ Block of a Graph: A block of a graph G is a maximal 2-connected sub-graph G, i.e a subgraph with as many edges as possible without a cut vertex.



G-blocks
↳ Every isolated vertex is a block → g, h
Every K_2 (means graph with 2 vertex) is a block → cd, de, df
a bca is also block as it doesnot have cut vertex.

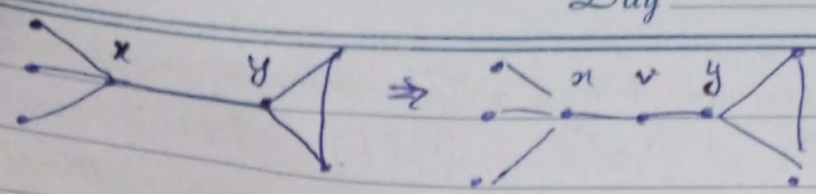


A vertex may be a cut vertex of G but cannot be a cut vertex of the blocks to which it belongs.

▷ Subdivision: Let e=xy be an edge in a graph G. The subdivision of e adds a vertex v so as to replace it with the path xvy. Denoted G'

Date

Day

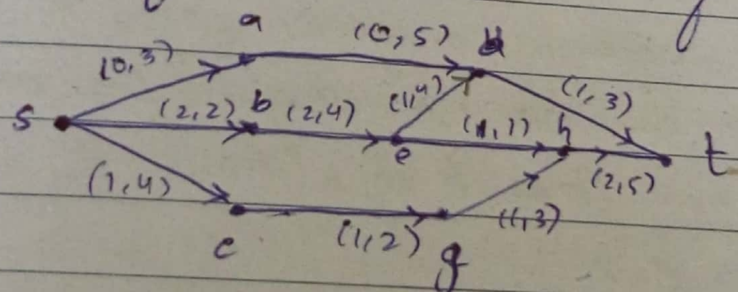


Theorem 4.21: G is connected iff G' is 2-connected

"Network Flow"

"A network is a diagraph where each arc e has an associated nonnegative integer c(e), called **capacity**. In addition, the network has a designated starting vertex s, called source, and ending vertex t, called sink."

A flow 'f' is a func that assigns a value f(e) to each arc of the network.



$f^-(v) \rightarrow$ total flow entering v
$f^+(v) \rightarrow$ " " exiting v

1) $f(e) \geq 0 \ \forall e$

2) $f(e) \leq c(e) \ \forall e$

3) $f^+(v) = f^-(v) \ \forall v$ other than s & t

4) $f^-(s) = f^+(t) = 0$

Date ____ Day ____

▷ Maximum Flow : The value of a flow is defined as $|f| = f^+(s) = f^-(t)$, that is amount exiting the src which must also equal the flow entering the sink.
A maximum flow is a feasible flow of largest value

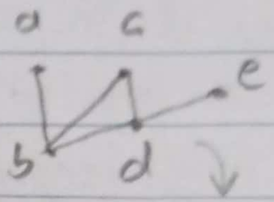
▷ Slack : The slack 'k' of an arc is the diff b/w it's capacity & flow ; that is, $k(e) = c(e) - f(e)$

▷ Augment Flow Algo:
From book example!

▷ Min-Cut Max Flow:
Ismein bs humein ek ese set of edges find krne hai jinko hatane se src or sink do components mein divide hojayein or jo un edges ki capacity ka sum ho wo Augment flow wale algo k max flow k equal ho.
(Example from Book)

"CHAPTER # 05"

"MATCHING AND FACTORS"



- **Independent Vertices & Independent Set:** {a, c, e}

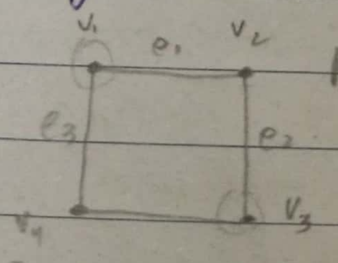
Two vertices are said to be independent if they are not adjacent in G.
A set "A" of vertices in G is called an independent set if every two vertices in A are independent in G.

→ A set of vertices that covers all edges

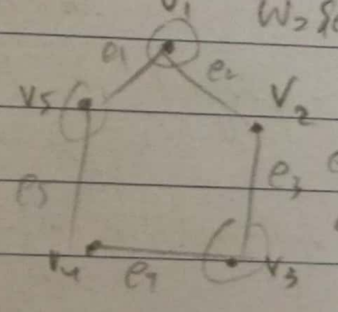
→ set of edges that cover all vertices

- **Vertex Cover & Edge Cover:**

A set Q of vertices in G is called a vertex cover if every edge in G is adjacent with a vertex in Q.
A set W of edges in G is called an edge cover of G if every vertex in G is incident with an edge in W.



$W = \{e_3, e_2\}$



$W = \{e_1, e_3, e_5\}$
↓ min edge cover

$Q = \{V_1, V_3\}$ or $\{V_1, V_2, V_3\}$
$\{V_2, V_4\}$

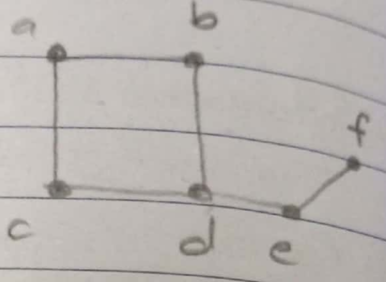
$Q = \{V_1, V_3, V_5\}$
↳ minimum vertex cover

↳ minimum vertex covering number is 2.

Page No.

Date Day

▷ Matching : A matching M is a subset of the edges of G so that no two edges share an endpoint. The size of a matching, denoted $|M|$, is the no. of edges in the matching.



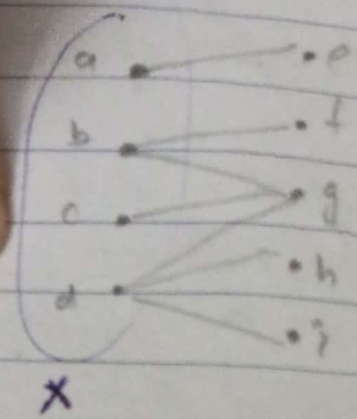
$M_1 = \{ac, bd, ef\}$ or
$M_2 = \{ab, cd, ef\}$ or
$M_3 = \{ac, de\}$

→ a vertex is saturated by a matching M if it is incident to an edge of the matching; otherwise, it is called unsaturated.

→ Given a matching M on graph G, we say M is
- maximal if M cannot be enlarged by adding an edge. Like M_1, M_2, M_3
- maximum if M is of the largest size amongst all possible matching. Like, M_1, M_2
- perfect if M saturates / includes every vertex of G. Like M_1 & M_2. Not M_3 as b, f not included.
- X-matching if it saturates every vertex from the collection of vertices X.

▷ Hall's Marriage Theorem:

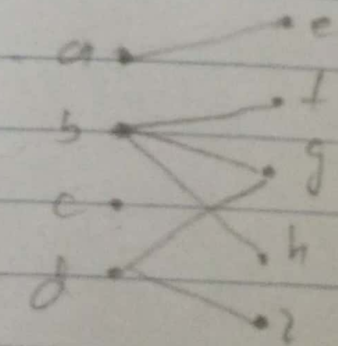
Given a bipartite graph $G = (X \cup Y, E)$, there exists an X-matching if and only if
$|S| \leq |N(S)|$ for any $S \subseteq X$



Let $S = \{a, b\}$ TRUE

$N(S) = \{e, f, g\}$

Contains all vertices which are adjacent to at least one of the vertices in S.



Let $S = \{c\} \rightarrow |S| = 1$

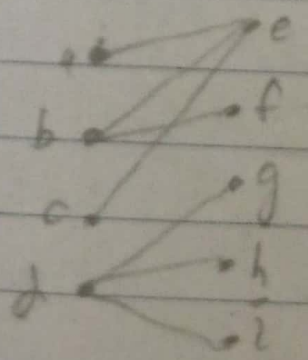
$N(S) = \{\ \} \rightarrow |N(S)| = 0$

False (No matching possible)

Let $S = \{a, c\} \rightarrow |S| = 2$

$N(S) = \{e\} \rightarrow |N(S)| = 1$

False (No matching possible)



Let $S = \{a, b, c\} \rightarrow |S| = 3$

$N(S) = \{e, f,\} \rightarrow |N(S)| = 2$

False

→ Every k-regular bipartite graph has a perfect matching for all $k > 0$.

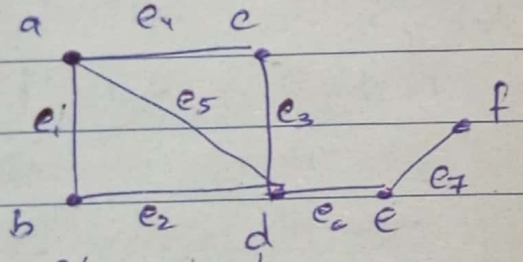
Date ________ Day ________

" Augmenting Paths & Vertex Cover "

→ Given a matching M of a graph G, a path is called.

M-alternating: if the edges in the path alternate b/w edges that are part of M & not part of M.

M-augmenting: an alternating path in which both endpoints of the path are unsaturated by M. and the starting & ending edges are not part of M.



$M_1 = \{e_1, e_5, e_7\}$ (a,b a,d d,e)

$\bar{M}_1 = \{e_2, e_4, e_5, e_6\}$

Alternating path: a b d c ⇒ a $\underset{M}{e_1}$ b $\underset{M'}{e_2}$ d $\underset{M}{e_3}$ c

↳ but not augmenting bcz a & c are saturated by M.

$M_2 = \{e_1, e_6\}$ (a,b d,e)

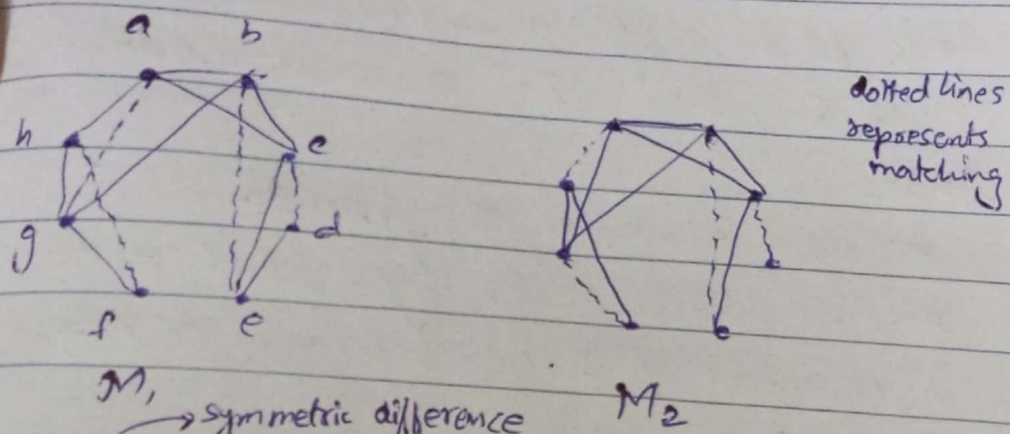
$\bar{M} = \{e_2, e_3, e_4, e_5, e_7\}$

Alternating path: f e d c ⇒ f $\underset{M'}{e_7}$ e $\underset{M}{e_6}$ d $\underset{M'}{e_3}$ c

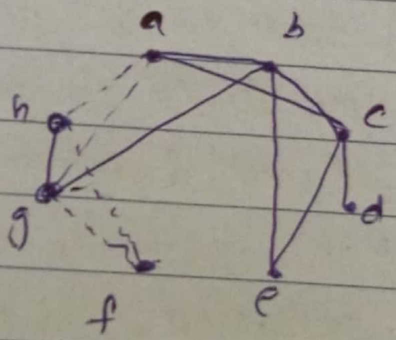
↳ also augmenting

Date   Day

Berge's Theorem: A matching M of a graph G is maximum iff G doesnot contain any M-augmenting path.
Like M_1 ~~was a maximum~~ didn't have the augmenting path and it was also the maximum



$M_1 \Delta M_2$ → symmetric difference



Let M_1 & M_2 be two matchings in a graph G. Then every component of $M_1 \Delta M_2$ is either a path or even cycle.

Date

Day

Augmenting Flow Algorithm :

① A graph will be given with some arbitary matching.

② Mark all the unsaturated vertex in X as U.

↳ if U is empty (means no unsaturated vertex in X) then M is a maximum matching

↳ else

i- Select a vertex 'x' from U.

ii- Now select vertex 'y' which is the neighbour of x i.e it belongs to N(x).

iii- if

'y' is also unsaturated then add the edge xy to M to obtain a larger matching M' & return to Step#2.

else if

'y' is saturated then find a maximal M-alternating path from x using xy as the 1st edge.

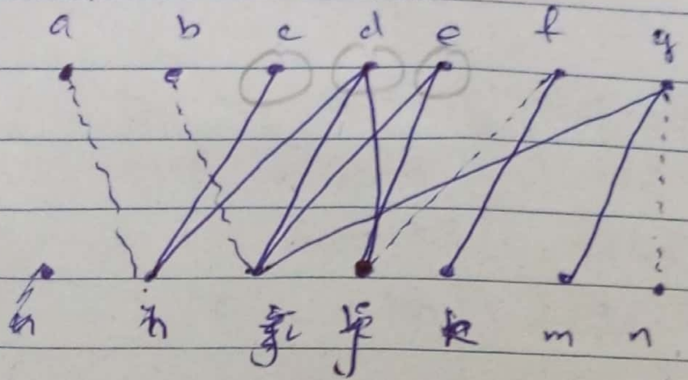
↳ if the path which we made is augmenting than add those edges of the path into the matching M that are not part of Matching M. And remove those edges of the path from M which are prev includ in M.

Page No.

Date   Day

↳ else if path is not M-augmenting then choose a new neighbour of x.

③ Stop repeating when all vertices from U have been considered.



$M = \{ah, bi, fj, gn\}$

$U = \{c, d, e\}$

→ Selecting c as 'x' & h as 'y' bcz then only neighbour of c is h.

→ h is staturated so we make an alternating path i.e cha. This path is not augmenting as a is ~~an~~ a saturated vertex. Since no neighbours left in c we choose d.

→ $N(d) = \{h, i, j\}$

→ Taking h as 'y'. 1st path → dha (not augmenting). 2nd path kd

→ Taking i as y. Path → dib (not augmenting)

→ Taking j as y. Path → djfk (augmenting as d & k both are unsaturated so now we

Page No.

Date ____ Day ____

→ So now we remove "fj" from M and add "dj" & "fk". M' = {ah, bi, dj, fk, gn}

→ Now again computing U. i.e U = {c, e}
→ Now again checking for c, however the path obtained is cha. So no change.

→ Now checking paths from e we get
Path 1 → e i b (not augmen—)
Path 2 → ejdha (not augm.)
Path 3 → ejdbib (not c—)

As U set completely considered so we get M' as our maximum matching.

Theorem 5.11 : For a bipartite Graph G, the size of maximum matching of G equals the size of minimum vertex cover for G.

▷ Finding Minimum Vertex Cover :
→ Apply the Augmenting Path Algo.
→ Define the vertex cover Q as the unmarked vertices from X and marked vertices from Y.
↳ Marked vertice wo hai jo U set mein defined thi or wo thi jo 'U' wali vertices k zariye track hoi thi like

Date   Day

jab humne d ko as x consider kro to ek path d ha bna, dib bna, or djfk to ye saari vertices marked hongi.

So the vertex cover we get will be

$$Q = \{\underbrace{g}_{\text{unmarked in } X}, \underbrace{h, i, j, k}_{\text{marked in } Y}\}$$

"MATCHING IN GENERAL GRAPHS"

Tutte's Theorem : A graph G has a perfect matching iff for every subset of vertices S the no. of odd components of G−S is at most |S|.

$$o(G-S) \leq |S|$$

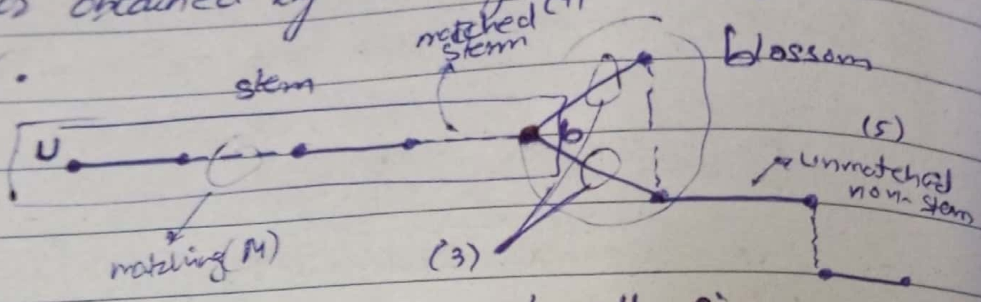
↳ Define o(G) to be the no. of odd components of G i.e the no. of components containing an odd no. of vertices.

➤ Edmond's Blossom Algorithims :

Given a graph G and matching M, a flower is a union of two M-alternating paths from an unsaturated vertex 'u' to another vertex 'v', where one path has odd length & other has even.

Date   Day

The stem of the flower is the maximal common initial path out of 'u', that ends at a vertex b, called . The is the odd cycle that is obtained by removing the stem from the flower.



↳ The stem must be of even length since the last edge must be from the matching.

↳2 The blossom is an odd cycle C_{2k+1}, where k is the no. of edges from M, & two edges from base must be unmatched edges.

↳3 Every vertex of blossom must be saturated

↳4 The only edges that come off the blossom and are also from M must be from the stem.

↳5 Any non-stem edge comming from blossom must be unmatched.

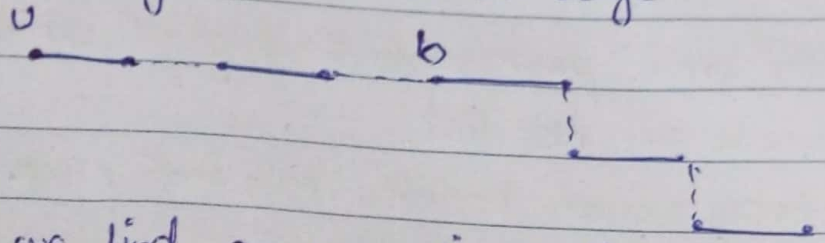
▷ Algorithm:

① the algo start same as the augmenting path

② We examine the alternating paths originating from an unsaturated vertex

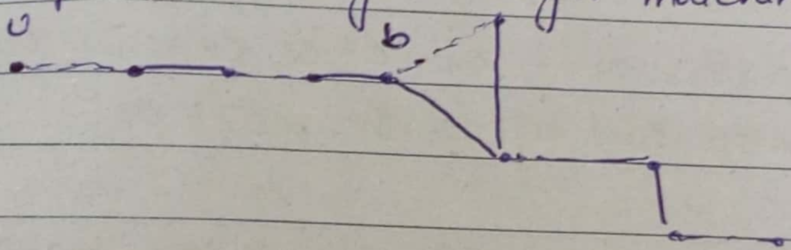
③ When blossom is encountered (means when if two diff paths to a vertex are found to end in different types of edges.

(4) We can the contract the blossom, much in the same way we contract an edge.



(5) If we find an augmenting path in the contracted graph, then we can find an augmenting path in the original graph by choosing the correct direction along blossom.

(6) ~~Just like~~ We than just swap edges along this path, creating a larger matching



"STABLE MATCHING"

A perfect matching is stable if no unmatched pair is **[unstable]**; that is if x & y ~~are~~ ~~a~~ matched nhi hai but dono kisi or ke zyada pasand krte hai apne ~~p~~ current partner k relative. Then x & y form an unstable pair.

Date

▷ Gale-Shapley Algorithm :

① Each man proposes to the highest ranking woman on his list.

② If every woman recieves only one proposal, this matching is stable.
↳ Otherwise move to Step (3).

③ If a woman recieves more than one proposal,

(a) She accepts if it is from the man she prefers above all other currently available men and rejects the rest ; or

(b) delays with with a may be to the highest ranked and rejects the rest.

④ Each man now proposes to the highest ranking unmatched woman on his list who has not rejected him.

⑤ Repeat Step (2) - (4) untill all people have been paired.

Date

Day

This algorithm favours the proposing group.

→ Unacceptable Partners :

① Saare man apni highest ranking women ko proposal bhejenge

② Agr har women ko ek hi proposal milega and that is the one they deem acceptable, they all accept & it is stable matching. Otherwise move to step 3

③ If the proposals are not all different, then each women :

(a) Rejects a proposal if it is from an unacceptable man

(b) accepts if the proposal is from the man she prefers above all other currently available men & rejects the rest; or

(c) delays with a maybe to the highest ranked proposal and rejects the rest.

④ Each man now proposes to the highest ranking unmatched woman on their list who has not rejected him.

Date Day

(5) Repeat Steps (2)-(4) untill all people have been paired or until no unmatched man has any acceptable partners remaing.

Anne : $t > r > s > w$

Brenda : $w > r > t$

Carol : $w > r > s > t$

Diana : $s > r > t$

Rob : $a > b > c > d$

Stan : $a > b$

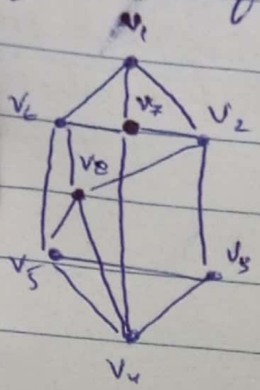
Ted : $c > d > a > b$

Will : $c > b > a$

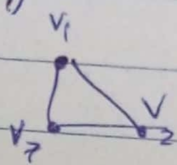
Date ____________ Day ____________

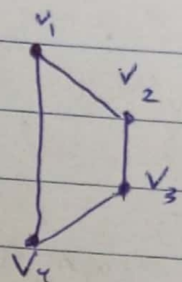
"FACTORS"

Let G be a graph with spanning subgraph H and let K be a positive integer. Then H is a "k-factor" of G if H is a K-regular.



for a 2-factor of this graph we only have to find a spanning Sub graph that is 2-regular (having all degrees 2)





If G is a 2k-regular graph, then G has a 2-factor.

→ k-factor mein bus hum spanning sub-graph bnana hoga jo k-regular ho.

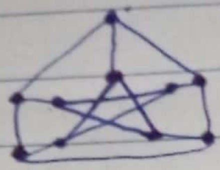
→ K-factorization mein we will partition our edge set into disjoint k factors.

↳ jab un tamam k-factors ka union krenge to original graph mil jayega.

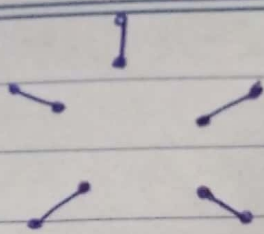
Page No. ____

Date

it's itself a 3-factor



G

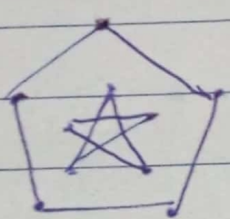


1-factor of G

(dissconected bhi hosakta hai)

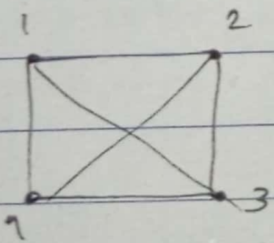
ns edges saari include ho or degree 1.

1-factor is a perfect matching of a graph.

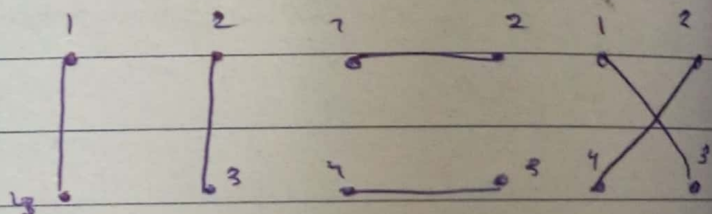


2-factor of G.

This graph G doesnot have k-factorization.



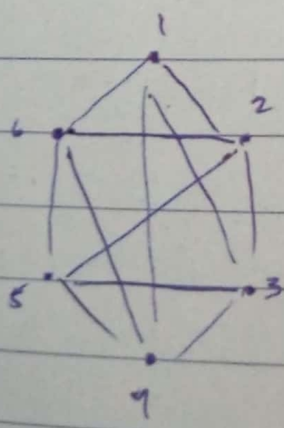
$\Rightarrow$



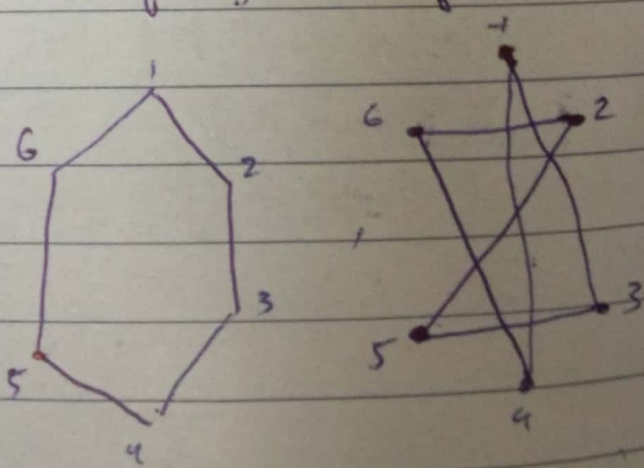
1-factors

1-factorization

in factorization no edges must be common



$\Rightarrow$



2-factorization

Date

Day

→ For a graph to have 1-factorization
↳ the graph must have an even no. of vertices & must be regular

→ Every k-regular bipartite graph has a 1-factor for all $k \geq 1$

→ A graph G has k-factorization iff G is $2k$-regular.

Page No.