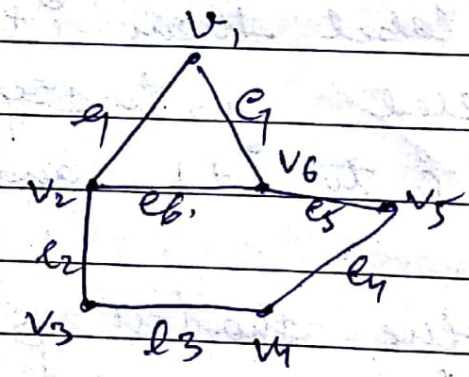
3/10/18

Matrix Representations of Graph: $G=(V,E)$, $|V|=n$

- Adjacency matrix → $n \times n$ matrix $A=[a_{ij}]$
  where $a_{ij} = 1$, if v_i & v_j are adjacent
  $= 0$, otherwise

- Incidence matrix → $n \times m$ matrix $B=[b_{ij}]$
  where $b_{ij} = 1$, if e_i is incident on vertex v_j.
  0, otherwise.

Ex:



Adj. matrix, $A_{6\times6}$ =

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	1
v_2	1	0	1	0	0	1
v_3	0	1	0	1	0	0
v_4	0	0	1	0	1	0
v_5	0	0	0	1	0	1
v_6	1	1	0	0	1	0

Date

Observations :-

1. A is symmetric (∵ undirected graph)
2. All the entries along the principal diagonal of A all 0's iff the graph has no self-loop. If the graph has self loop at the vertex v_i then $a_{ii} = 1$.
3. If the graph is simple, the degree of vertex equals the no. of 1's in the corresponding row or column of A.
4. If we have ll edges b/w v_i & v_j then a_{ij} = no. of edges b/w them.

Ex. $|V| = 6$, $|E| = 7$

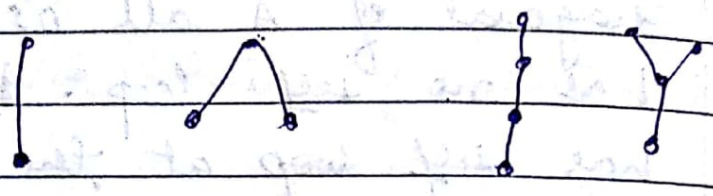
Inc matrix, $B_{6\times7}$ or (7×6) =

	v_1	v_2	v_3	v_4	v_5	v_6
e_1	1	1	0	0	0	0
e_2	0	1	1	0	0	0
e_3	0	0	1	1	0	0
e_4	0	0	0	1	1	0
e_5	0	0	0	0	1	1
e_6	0	1	0	0	0	1
e_7	1	0	0	0	0	1

Unit - II →

Tree: It is a connected graph without cycles.

Ex:



n=1 n=2 n=3. n=4

Theorem: There is only one path between every pair of vertices in a tree.

Proof:

Suppose T is a tree and let v_i and $v_j \in V(T)$

Let ∃ more than 1 path b/w v_i & v_j

Union of these 2 paths forms a cycle in T which is a contradiction ∴ T is a tree i.e. connected graph with no cycles.

Hence, <theorem>.

Theorem: If in a graph G, there is only one path b/w every pair of vertices, G is a tree.

Proof: Prove 2 things ⟺ connected graph, no cycles

DIY

Page No.
Date

Theorem: A tree with n vertices has $(n-1)$ edges.

Proof:

For $n=1$ → result is true.

Suppose the result is true $(n-1)$ vertices

Now, consider a tree with n vertices.

∴ no. of edges $= (n-2) + 1$.

↓ only one edge will be added, else cycle will be formed.

$= n-1$.

Hence proved.

Theorem: A graph is a tree iff it is minimally connected.

Proof: Let $G = (V, E)$ be a tree.

∴ ∃ only one path b/w every pair of vertices ⇒ removal of any edge disconnects the graph.

converse: Let G be minimally connected & $|V(G)| = n$.

⇒ G is connected.

Suppose ∃ a cycle in G b/w v_i & v_j.

⇒ ∃ more than one path b/w v_i & v_j

Date

That means removal of all paths except one will still let v_i & v_j to be connected.
⇒ contradiction to the fact that G is minimally connected.
⇒ no cycle
⇒ G is tree.

* Rooted Tree - It is a tree in which one particular vertex is distinguished from the others and is called the root of the tree.

* level of a vertex in a rooted tree :- is the no. of edges along the unique path. b/w it and the root.

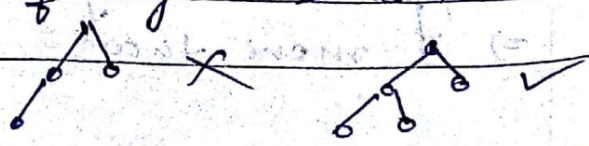
⇒ level of ~~root~~ vertex = 0.

* Height of rooted tree = max. level to any vertex of tree.

* child, parent, leaf nodes.

8/10/18

Binary Tree: It is a tree in which there is exactly one vertex of degree 2 and all other vertices of degree 1 or 3. The vertex of degree 2 is the root vertex.



Result 1: The no. of vertices in a binary tree is always odd.

Proof: Suppose the no. of vertices = n.

1 vertex of 2 degree or even degree

$n-1$ vertices of odd degree (1 or 3)

But no. of odd degree vertices must be even.

$\therefore n-1$ = even

$\Rightarrow n$ = odd.

Result 2: find the no. of pendant vertices in a binary tree.

→ Let p = no. of pendant vertices in G.

n = no. of vertices in G

$n-(p+1)$ = n " " n of deg. 3.

$$p + 3(n-p-1) + 2 = \underbrace{2(n-1)}_{\text{no. of edges in tree with } n \text{ vertices.}}$$

$$n - 2p + 1 = 0$$

$$\therefore \boxed{p = \frac{n+1}{2}}$$

Result 3: Prove ~~by~~ ~~XXXX~~ that every non-trivial tree T has atleast two vertices of deg. 1.

Proof: n = no. of vertices

m = no. of vertices of deg. 1. $(\rightarrow v_1, v_2 \cdots v_m)$

$d(v_j) \geq 2$.

Page No.
Date

$$\sum_{i=1}^{m} d(v_i) + \sum_{j=m+1}^{n} d(v_j) \geq 2(n-1)$$

$$m + \sum d(v_j) = 2(n-1)$$

$$m + 2(n-m) \leq 2n-2$$

$$\Rightarrow m \geq 2.$$

Hence proved.

Theorem: Prove by PMI that the max. no. of vertices on level n of a binary tree is 2^n, $n \geq 0$.

Proof: For $n=0$:

There is only 1 vertex $= 2^0 = 1$

Hence true.

Suppose the result is true for $n=k$.

$\Rightarrow$ max. no. of vertices at k level $= 2^k$.

For $n=k+1$:

max. no. of vert. $= \underbrace{2}_{\text{2 children of every vertex.}} \cdot 2^k$

$= 2^{k+1}$

Hence proved.

Page No.
Date

Theorem: The max. no. of vertices in a binary tree of height h is $2^{h+1} - 1$, $h \geq 0$.

Proof:

$$2^0 + 2^1 + 2^2 + \dots + 2^h$$

$$= 1 + 2^1 + 2^2 + \dots + 2^h$$

$$= \frac{1(2^{h+1} - 1)}{2 - 1}$$

$$= 2^{h+1} - 1$$

* **Spanning Tree:**

G - connected graph

T - Tree

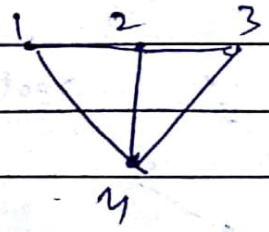
T is a spanning tree of G if $T \subseteq_{gn} G$ and contains all the vertices of G.

$|V(G)| = n$, $|E(G)| = m$

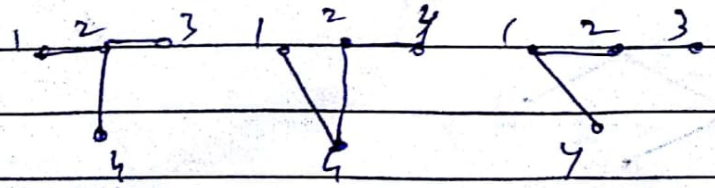
we'll remove $m - (n-1)$ edges to get a spanning tree.

Ex: $|V| = 4 = n$

$|E| = 5 = m$



$5 - (4-1) = 5 - 3 = 2 \Rightarrow$ remove 2 edges

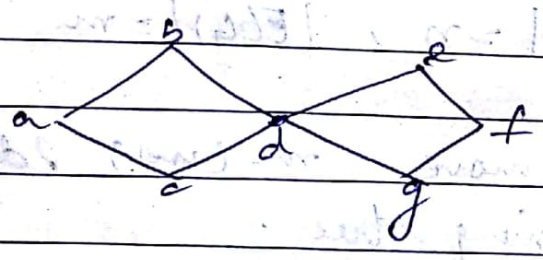


Date

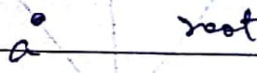
* BFS algorithm for constructing spanning tree of a connected graph.

Arbitrarily choose a vertex and designate it as the root. Then add all edges incident to this vertex, i.e. the addition of the edges don't produce any cycle. The new vertices added at this stage become the vertices at level 1 in the spanning tree, arbitrarily order them. Next for each vertex at level 1, visited in order and each edge incident to this vertex to the tree as long as it doesn't produce a cycle. Continue the same until all the vertices in the tree have been added.

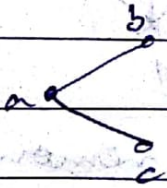
ex:



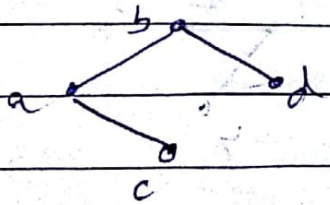
I:



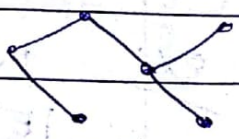
II:



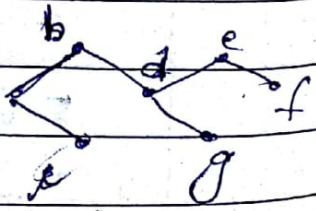
III:



IV:



V:



branches → edges of a tree.
Chords → " " G which are not in T.

Page No.
Date

*. Minimum Spanning Tree: ← Kruskal's / Prim's

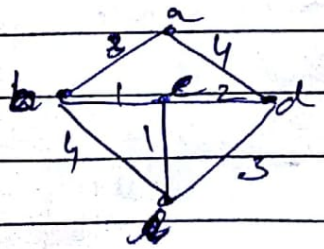
G - Connected graph (weighted)
T - spanning-tree of G.

Fundamental ckt:- ckt. formed on adding a chord to a spanning tree.

1. Kruskal's Algorithm:

Step 1: list of all edges of G in an increasing order of their weights.
Step-2: Select an edge of min. weight (if more than one edge of min. wt, choose any of them).
Step-3: At each stage, select an edge of min. wt. from all the remaining edges of G if it doesn't form a cycle with the previously selected edges in T. Include the edge in T.
Step 4: Repeat step 3 until (n-1) edges have been selected when |V(G)| = n.

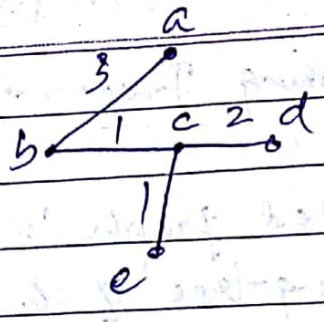
Ex:



(bc) (ce) (cd) (ab)
1 1 2 3

(de) (be)
3 4

Scanned by CamScanner



MST.

15/10/18 * **Prims Algorithm for finding the minimal spanning tree.**

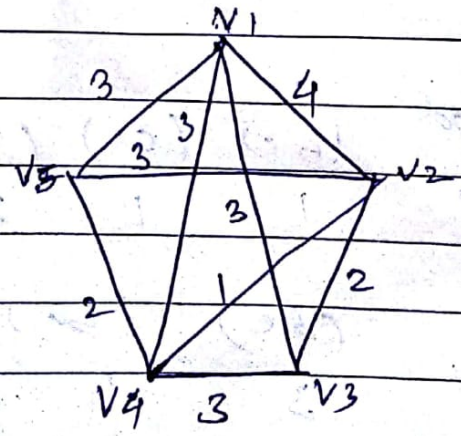
Let G be any connected graph.

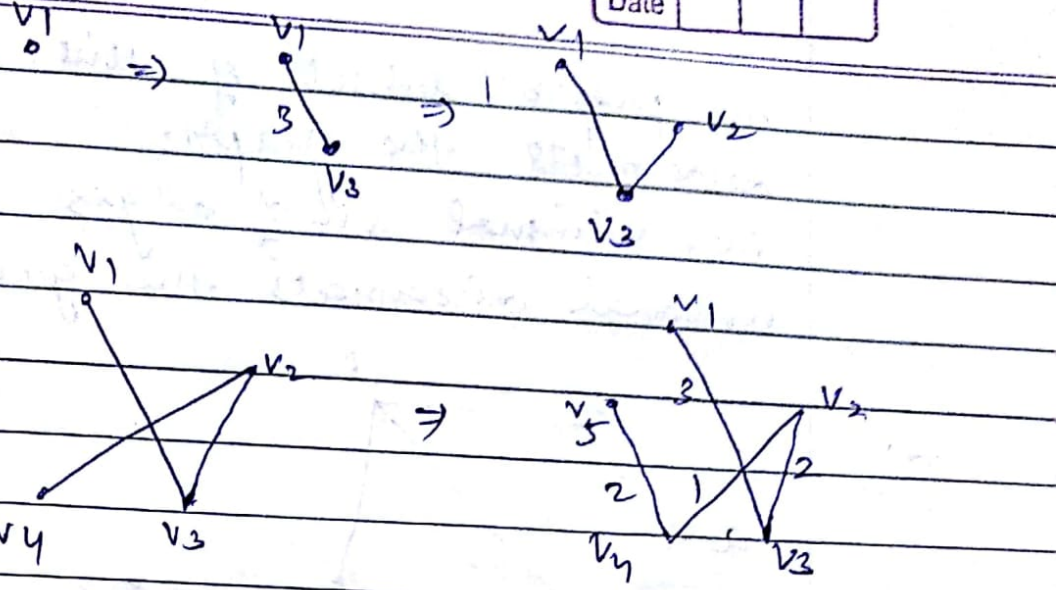
Step 1: Select any vertex in G. Among all the edges incident with the selected vertex, chose an edge of min. weight, include it in T.

Step 2: At each stage, choose an edge of smallest weight joining a vertex already included in T and a vertex not yet included, if it doesn't form a cycle with the edges in T. Include it in T.

Step 3: Repeat until all the vertices of G are included.

Ex:





wt. of MST = 3 + 1 + 2 + 2 = 8

---

* **CUT VERTEX**

G – any connected graph
$v \in V(G)$

v is called a cut vertex of G if removal of v disconnects the graph G.

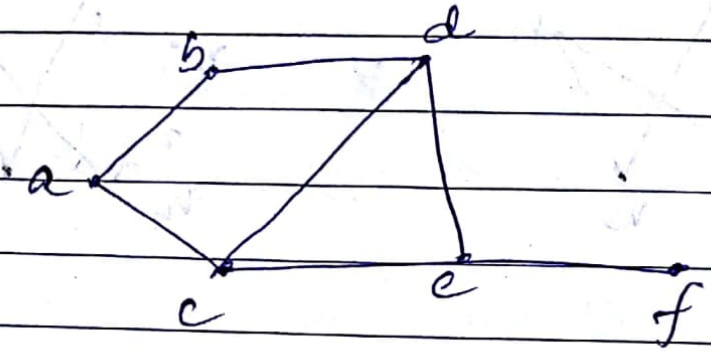
* **CUT EDGE** (or Bridge).
An edge $e \in E(G)$ is a cut edge of G if removal of 'e' disconnects the graph.

* **CUT SET** : A set of edges of G is a cut set if removal of these edges disconnects the graph s.t.

Date

no proper subset of this set disconnects the graph.
i.e. minimal set of edges of G whose removal disconnects the graph.

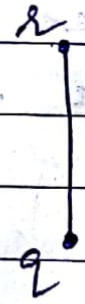
ex:



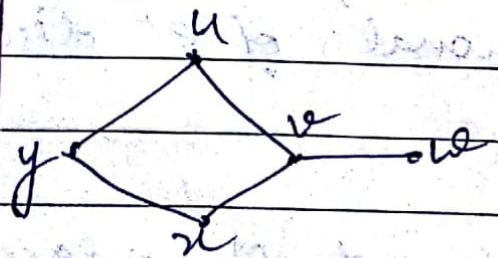
cut edge - {(c,f)}
cut sets - {(e,f)}, {(a,b), (b,d)},
{(b,d), (c,d), (c,e)}

no proper subset of any of these disconnects the graph.

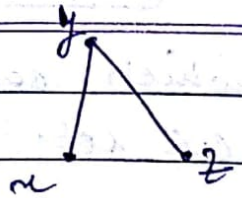
ex:



⇒ No cut vertex
cut-edge : (r,q)



⇒ cut vertex - v
cut edge - (v,w)



⇒ y - cut vertex

cut-edge :- (x, y) or (y, z)

↓

They'll also be cut sets.

* $\lambda(G)$ Edge connectivity - Min. no. of edges required to ~~con~~ disconnect the graph.

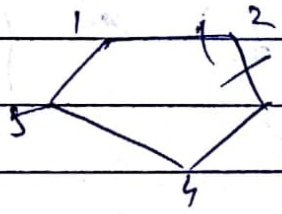
$\kappa(G)$ Vertex connectivity - Min. no. of vertices reqd. to disconnect the graph.

(K_n) $\lambda(K_n) = n-1$

(K_4) $\lambda(K_4) = 3.$

(C_n) $\lambda(C_n) = 2$

cycle of n vertices →



(C_5) $\lambda(C_5) = 2$

Theorem: Let G be a connected graph. S is a cut set of G iff it contains at least one branch of every spanning tree.

Proof: Suppose S is a cut set of G. T is a spanning tree of G s.t. S does not contain any edge of T.

Date

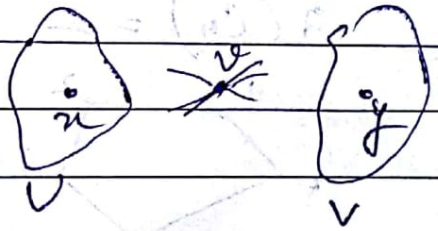
G is still connected which contradicts the fact that S is cut set.

Converse: Suppose S contains at least one edge of every spanning tree
(G-S) is disconnected.
e → any edge from any of the STs.
∴ (G-S+e) will be connected.

⇒ S is cut set.

Theorem: A vertex v of a connected graph G is a cut-vertex of G iff ∃ two vertices x & y in G s.t. every path b/w x & y passes through v.

Proof: Suppose v is a cut vertex of G



Converse, ____

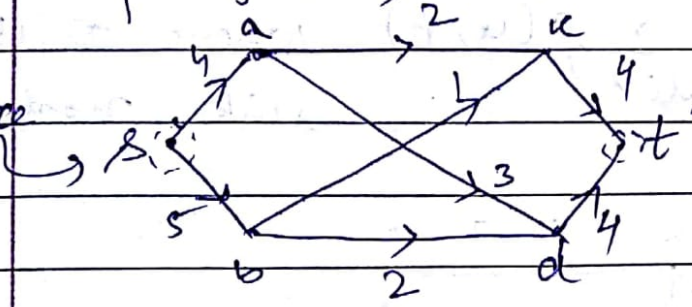
19/10/18.

Defn: Network: It is a simple connected weighted directed graph N(V, E) s.t.
1. ∃ a unique vertex s ∈ V having in degree 0. This vertex is called

source node.

2. ∃ a unique vertex $t \in V$ having out-degree 0, called as sink.

3. Every directed edge $e = (v, w) \in E$ has been assigned a non-negative no. called capacity e ; $c(e)$ or $c(v, w)$



Defn: A flow in a network is a fn. F that assigns each arc $e = (v, w) \in E$, a non-negative real no. s.t.

1. $f(e) \leq c(e) \quad \forall \; e \in E$ (capacity constraint)

2. For any intermediate vertex x equal to the total flow out of x i.e.

$$\sum_{w \in V} f(w, x) = \sum_{v \in V} f(x, v)$$

(Flow conservation)

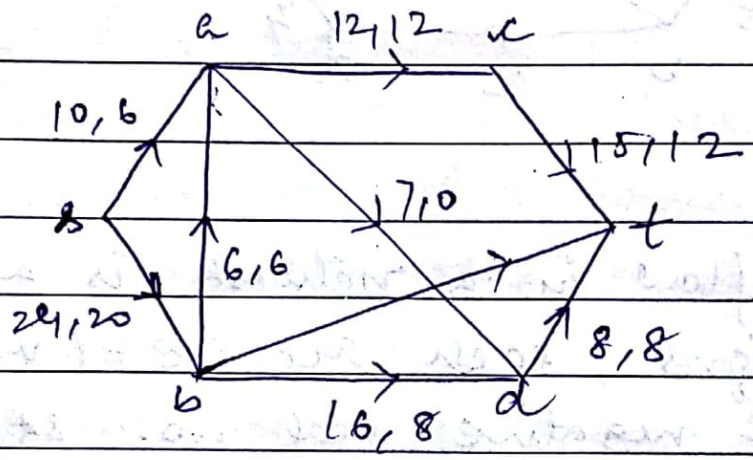
The flow along an edge $e = (v, w)$ is said to be saturated if $f(e) = c(e)$. And if $f(e) < c(e)$, then:

$$s(e) = c(e) - f(e) \quad \text{(residual capacity)}$$

The value of the flow is the net amount of flow per unit time reaching the sink node

$val(f) = \sum f(s, v)$, where s is the source node

$= \sum_{u \in V} f(u, t)$ where t is the sink node.



flow into a $= 12 = f(s,a) + f(b,a)$

flow out of a $= \sum_{u \in V} f(a,u) = f(a,c) + f(a,d)$

$= 0 + 12 = 12$

$val(f) \Rightarrow f(s,a) + f(s,b) = 6 + 20 = 26$

$\Rightarrow f(c,t) + f(b,t) + f(d,t)$

$= 12 + 6 + 8$

$= 26$

Date

* Flow that achieves the largest value is max. flow.

* $N(V, E)$

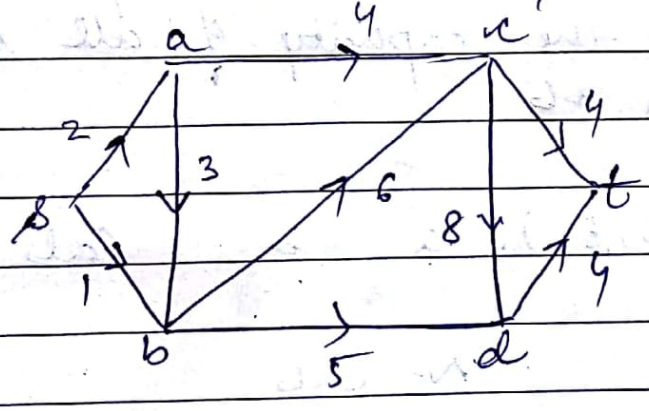
c - set of arcs of n or $C \subseteq E$ s.t. cut or simply a cut for the network N iff

check. P & $\bar{P}$ → vertex set of component having sink node.

↓

vertices having source node

Then a cut is defined by $(P, \bar{P})$.



$(P, \bar{P})$	P	$\bar{P}$	C
$\{(s,a),(s,b)\}$	$\{s\}$	$\{a,b,c,d,t\}$	3
$\{sb, ab, ad\}$	$\{s,a\}$	$\{b,c,d,t\}$	8
$\{sa, bd, bc\}$	$\{s,b\}$	$\{a,c,d,t\}$	13
	$\{s,c\}$	$\{a,b,d,t\}$	

* n intermediate nodes, so 2^n cut sets for the network.

The capacity of a cut denoted by $C(P, \bar{P})$ and given by:

$$C(P, \bar{P}) = \sum_{\substack{v \in P \\ w \in \bar{P}}} c(v, w)$$

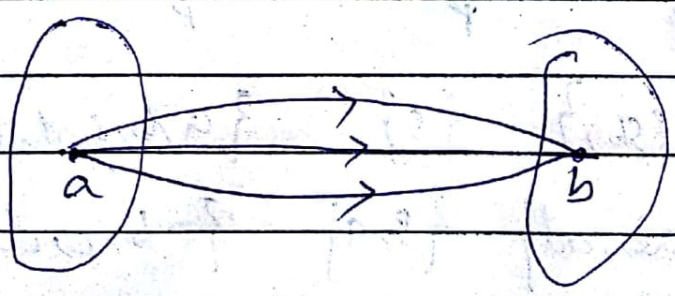
* Minimum cut:

Thm: Min-cut max. flow theorem -

The max flow possible b/w vertices a & b in a network is equal to the min. of the capacity of all cut sets wrt a & b

$N(V, E)$. a b Cab

N - Cab



31/10/18.

Date

Ford - Fulkerson Algorithm for max. flow:

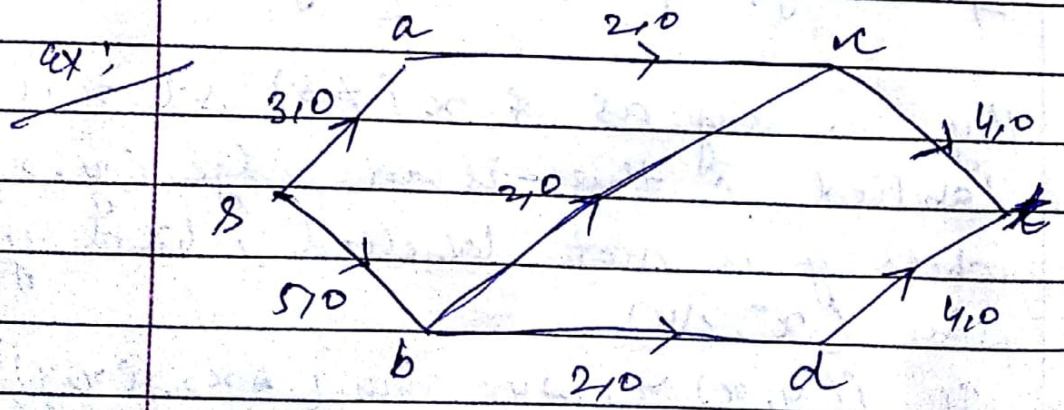
1. Given a network N, define an initial flow f by $f(e) = 0 \ \forall \ e \in E$
2. Label the source s as $(-, \infty)$
3. (It means the source can supply an ∞ amount of material to the other vertices.
3. For each vertex x that is adjacent from s is labelled with $(s^+, \Delta x)$
   (i). if $c(s,x) > f(s,x)$, $\Delta x = c(s,x) - f(s,x)$
   (ii). if $c(s,x) = f(s,x)$, x is not labelled
4. As long as $\exists\, x\ (\neq s)$ in V s.t. x is labelled and there is an edge (x,y) where y is not labelled, label y with $(x^+, \Delta y)$
   (i). If $c(x,y) > f(x,y)$, $\Delta y = \min\{\Delta x, c(x,y) - f(x,y)\}$
   (ii). If $c(x,y) = f(x,y)$, then y is not labelled.
5. Ily, as long as $\exists\, x\ (\neq s)$ s.t. x is labelled & there is an edge (y,x), where y is not labelled, label y with $(x^-, \Delta y)$
   (i). If $f(y,x) > 0$, $\Delta y = \min\{\Delta x, f(x,y)\}$
   (ii). If $f(y,x) = 0$, then y is not labelled.

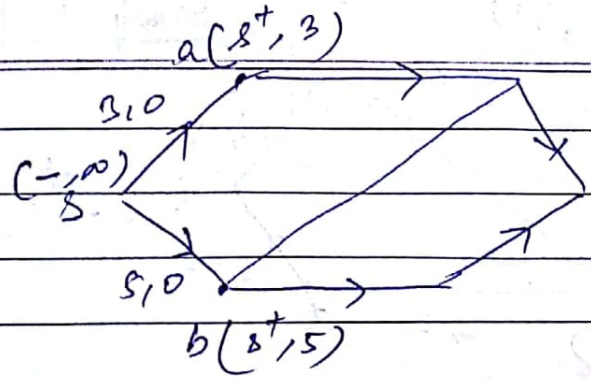
(If a vertex can be labelled in more than one way, an arbitrary choice can be made).

case 1: If sink t is labelled as $(x^+, \Delta t)$, then the flow in the edge (x,t) can be increased from $f(x,t)$ to $f(x,t) + \Delta t$. vertex x must be labelled either $(v^+, \Delta x)$, we may regard v as the source for increasing the flow in the edge (v,x) from $f(v,x)$ to $f(v,x) + \Delta t$. If x is labelled as $(v^-, \Delta x)$ the flow in the edge (v,x) changes from $f(v,x)$ to $f(v,x) - \Delta t$ so that the increment in Δt from x to t is
the process is continued back to the source s.

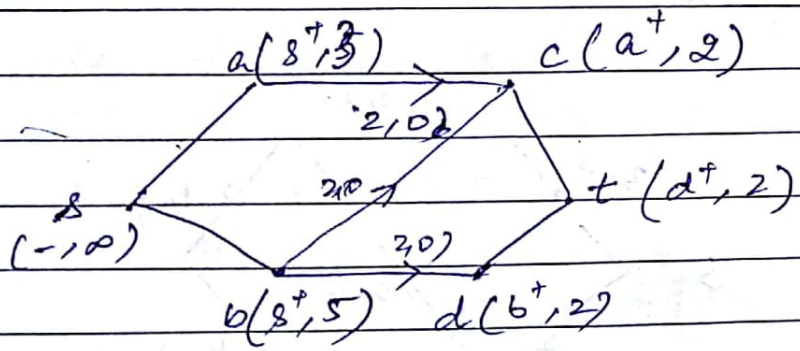
case 2: If sink t is not labelled, then the max. flow is attained.

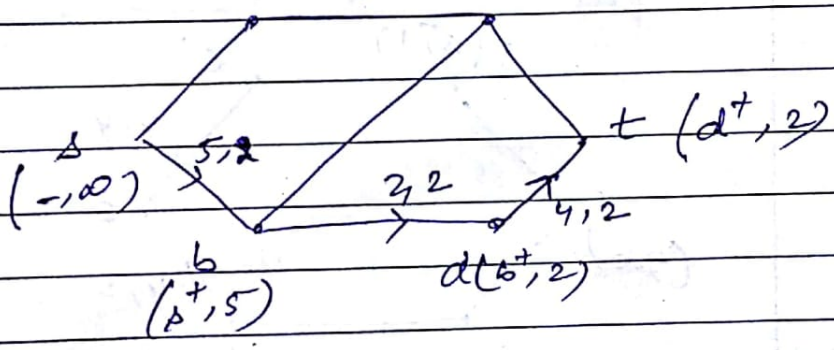
Ex:

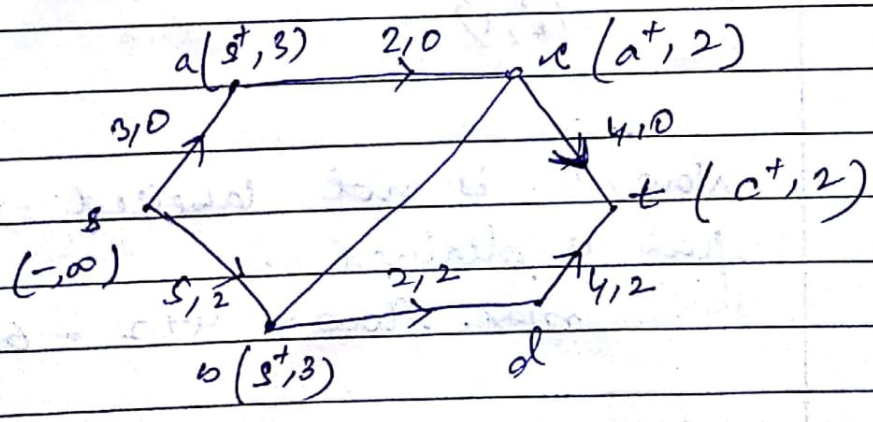


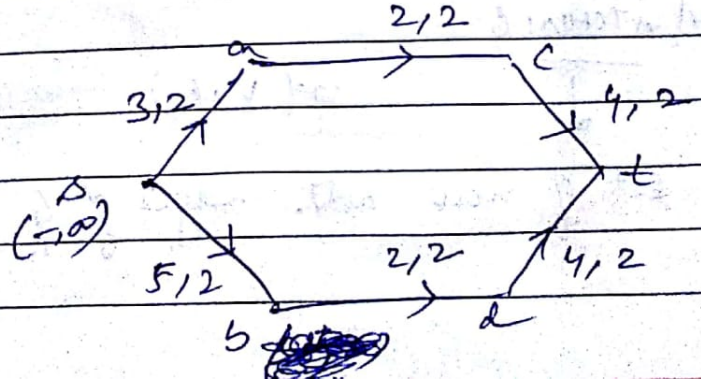


arbitrary choice (either a or b)

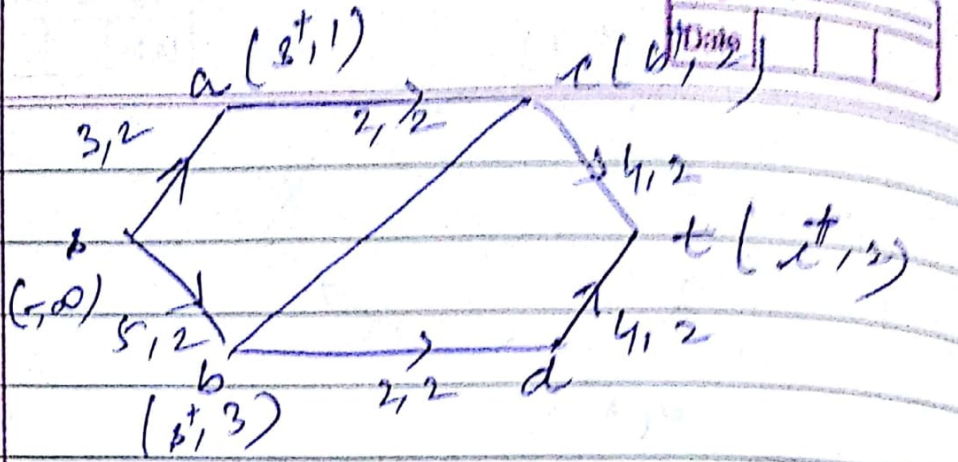


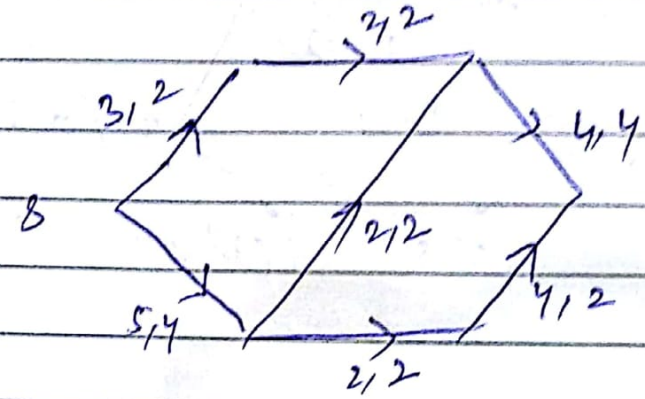


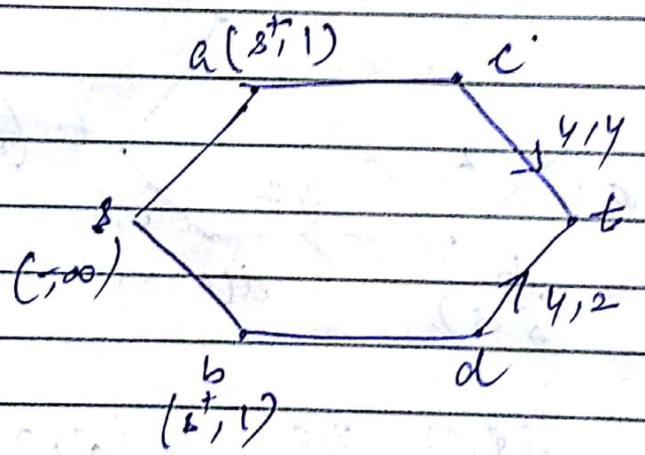




t is still labelled







∴

Now, t is not labelled ⇒ max. flow is attained.

max. flow = 4+2 = 6

*. MATCHING.

↓

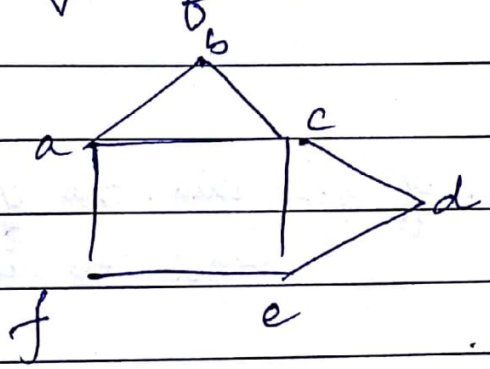
$G = (V, E)$

set of non adj. edges of G.

Page No.
Date

Saturated vertex :- vertex s.t. there is an edge in matching set incident on it; is called S.V.

A matching M is perfect iff every vertex of G is saturated i.e. every vertex is incident with precisely one edge of the matching.



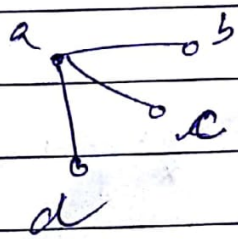
$M_1 = \{ab, ce\}$

$M_2 = \{ab, cd, ef\}$

↳ perfect matching.

$M_3 = \{af, ed\}$

$M_4 = \{ac, ef\}$



No perfect matching.

(Kn) for n = even → perfect matching exists.

(Kmn) $m \neq n \Rightarrow$ perfect matching

Date

no. of perfect matching in $K_{n,n} = \underline{\underline{n!}}$

* Graph colouring: It is an assignment of colors to the vertices of G s.t. no two adjacent vertex receives the same color.

n-colorable.

chromatic no. of G = min. no. of colors reqd. to color G ($\chi(G)$)

$\chi(G) \leq n$.

(K_n) $\chi(K_n) = n$.

For any bipartite graph, $\chi = 2$ i.e.

Any bipartite graph is 2-colorable.