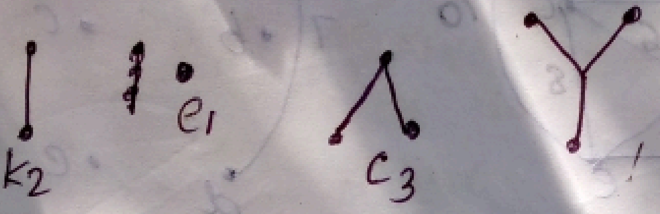
Module : 4 Trees

A graph with no cycle is called acyclic
A tree is a connected acyclic graph ie
A tree is a connected graph without
any cycle. Tree must have atleast one
vertex. There has to be a simple graph
having neither a self loop or ||el edges.

eg:

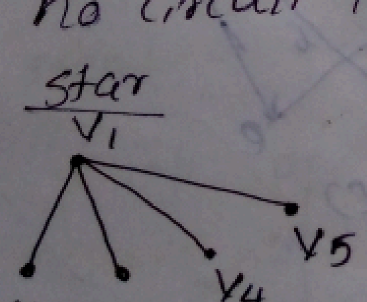


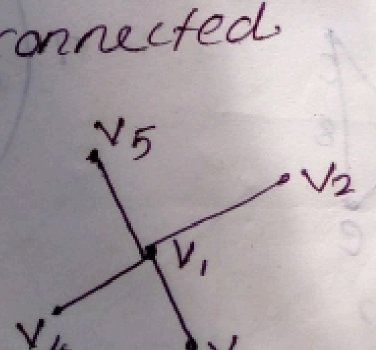
pendant vertex.
A vertex having one degree is called a
pendant vertex.

properties of Trees

- There is one and only one path b/w every pair vertex in a tree.
- A tree with n vertices has $n-1$ edges
- A graph is a tree if it's minimally connected
- A graph G with n vertices $n-1$ edges and no circuit is connected

Star





A star is a tree consisting one vertex
and is adjacent to all other vertex.

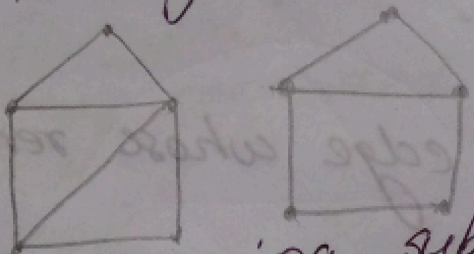
Note

A graph with no cycle has no odd cycle then it's bipartile

∴ Tree is a bipartile graph

every tree has a bipartile graph

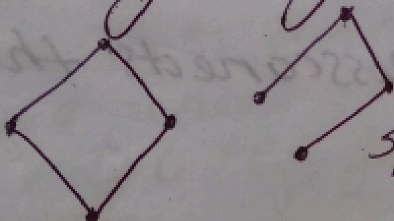
spanning tree



→ spanning subgraph

A spanning subgraph of G is a subgraph with vertex set V(G). A spanning tree is a spanning subgraph that is a tree.

Eg:



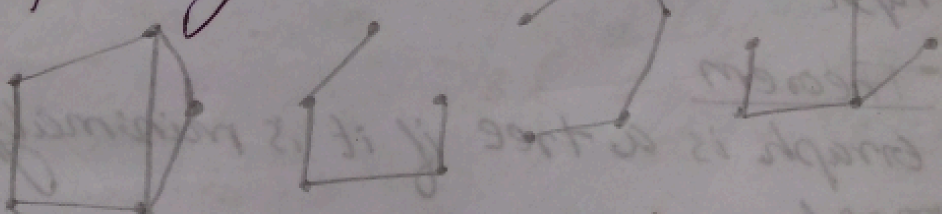
spanning tree

Note

A spanning subgraph of G need not be connected.

Eg: If $n(G) > 1$ Let $V(G) = \{v_1, v_2, \ldots, v_n\}$ $E(G) = \phi$ is a spanning subgraph, but not connected.

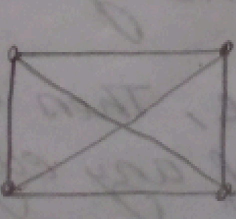
Eg: spanning tree.



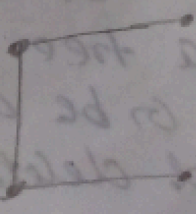
problem

Draw the spanning tree of

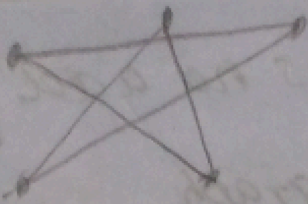
a)



an:



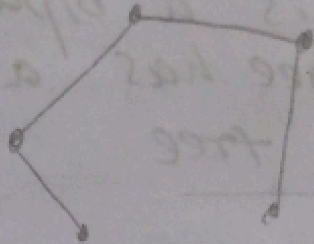
b)



an:

c)

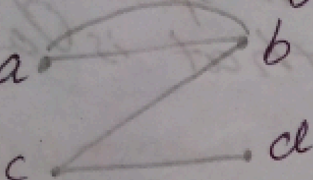
an:



cut edge

A cut edge is a single edge whose removal dissconnects a graph

Eg:

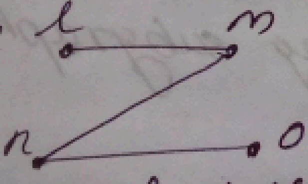


removal of edge bc dissconects the graph into two components.

cut vertex.

A cut vertex is a single vertex whose removal dissconnects a graph.

Eg:



removal of vertex n, results a disconnected graph

Theorem

A Graph is a tree if it is minimaly connected

proof

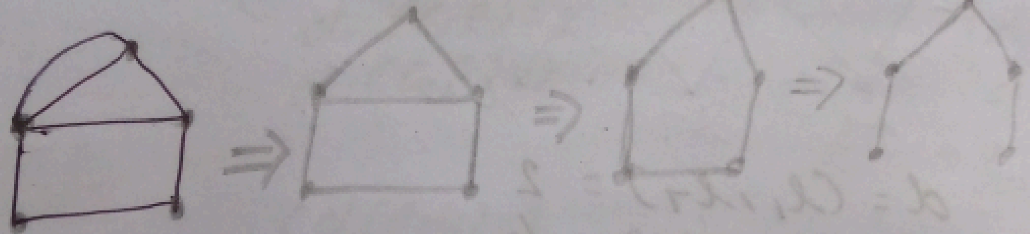
Let a graph G be minimaly connected which means G has no cycle.

∴ G is a tree.

conversly let G be a tree, Then G contains no cycle and deletion of any edge from

G is disconnects the graph, hence G is minimally connected. Hence the theorem is proved.

Eg:



Theorem

Every connected graph contains a spanning tree.

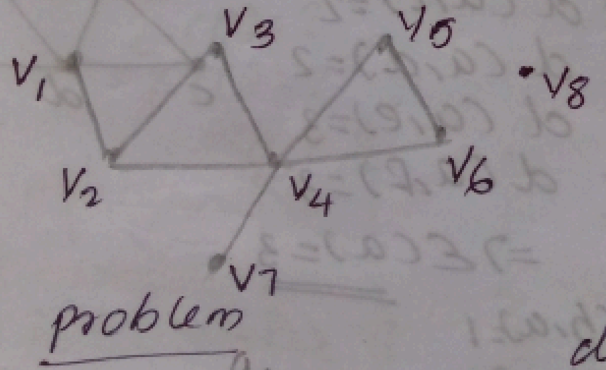
proof

consider the connected graph G with n vertices and m edges. If $m = n-1$, then G is a tree.

since G is connected $m \geq n-1$, i.e $m \geq n$, where there is a circuit in G. we remove an edge 'e' from that circuit ∴ G-e is connected. we repeat until there are n-1 edges then we are left with a tree.

Distance in Tree and Graph

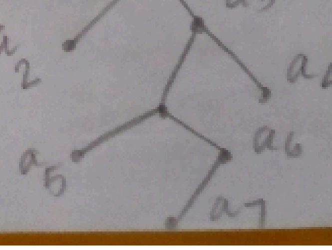
If G has a u-v, then the distance from u to v written as $d_G(u,v)$ or $d(u,v)$ is the least length a u-v path. If G has no such path, then $d(u,v) = \infty$



$d(v_1, v_5) = 3$
$d(v_2, v_7) = 2$
$d(v_1, v_8) = \infty$

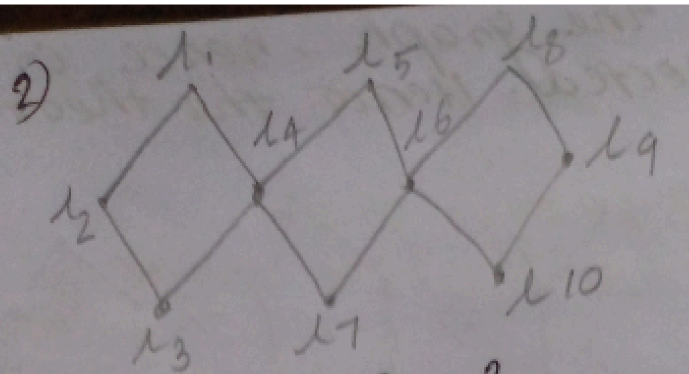
problem

1)



$d(a_1, a_4) = 2$
$d(a_1, a_7) = 4$
$d(a_3, a_6) = 2$

2)

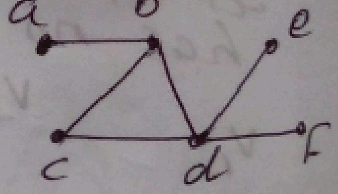


$d = (l_1, l_7) = 2$
$d = (l_3, l_8) = 4$
$d = (l_2, l_3) = 1$
$d = (l_4, l_9) = 4$

Graph eccentricity

The eccentricity $\varepsilon(v)$ with vertex v, in a connected graph G is the maximum distance b/w v and any other vertex $u(G)$.

→ The maximum eccentricity ~~$\varepsilon(v)$ with vertex v, is a connected graph G is the maximum dist~~ of a graph is called graph diameter and the ~~maxim~~ minimum graph eccentricity is called graph radius.



Eg: $\varepsilon(a)$ max: $\begin{cases} d(a,b)=1 \\ d(a,c)=2 \\ d(a,d)=2 \\ d(a,e)=3 \\ d(a,f)=3 \end{cases}$

$\Rightarrow \varepsilon(a) = 3$

$\varepsilon(b) = \max: \begin{cases} d(b,a)=1 \\ d(b,c)=1 \\ d(b,d)=2 \\ d(b,f)=2 \end{cases}$

$= 2$

$\epsilon(c) = \max \begin{cases} d(c,a) = 2 \\ d(c,b) = 1 \\ d(c,d) = 1 \\ d(c,e) = 2 \\ d(c,f) = 2 \end{cases}$

$= \underline{\underline{2}}$

$\epsilon(d) = \max \begin{cases} d(d,a) = 2 \\ d(d,b) = 1 \\ d(d,c) = 1 \\ d(d,e) = 1 \\ d(d,f) = 1 \end{cases}$

$= \underline{\underline{2}}$

$\epsilon(e) = \max \begin{cases} d(e,a) = 3 \\ d(e,b) = 2 \\ d(e,c) = 2 \\ d(e,d) = 1 \\ d(e,f) = 2 \end{cases}$

$= \underline{\underline{3}}$

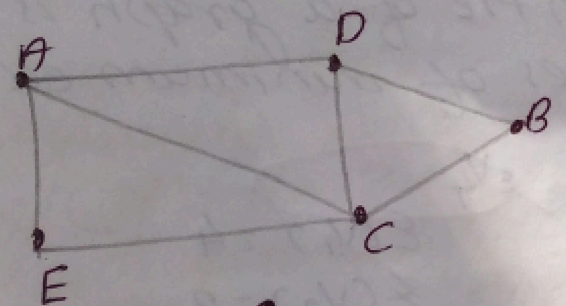
$\epsilon(f) = \max \begin{cases} d(f,a) = 3 \\ d(f,b) = 2 \\ d(f,c) = 2 \\ d(f,e) = 2 \\ d(f,d) = 1 \end{cases}$

$= \underline{\underline{3}}$

Diameter = maxi eccentricity = 3

Radius = mini eccentricity = $\underline{\underline{2}}$

Q)

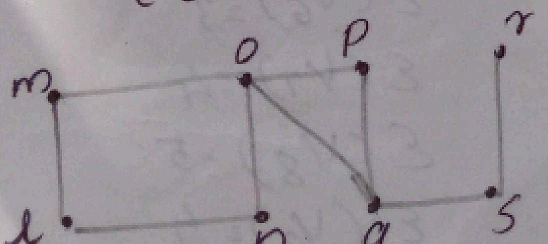


~~$\epsilon(A)$~~ =

an: $\epsilon(A) = 2$
$\epsilon(B) = 2$
$\epsilon(C) = 1$
$\epsilon(D) = 2$
$\epsilon(E) = 2$

radius = 1
Diameter = $\underline{\underline{2}}$

Q)

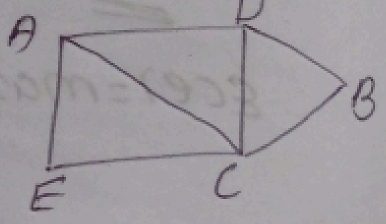


an: $\epsilon(m) = 4$
$\epsilon(l) = 5$
$\epsilon(n) = 4$
$\epsilon(o) = 3$
$\epsilon(p) = 3$
$\epsilon(q) = 3$
$\epsilon(r) = 5$
$\epsilon(s) = 4$

radius = 3
Diameter = 5.

central point: If the eccentricity of a graph is equal to its radius, then it is called central point.

ie $\mathcal{E}(V) = R(V)$ the V is central point of G.



$\mathcal{E}(A) = 2$

$\mathcal{E}(B) = 2$ $R(G) = 1$

$\mathcal{E}(C) = 1$ Here $E(C) = R(G)$

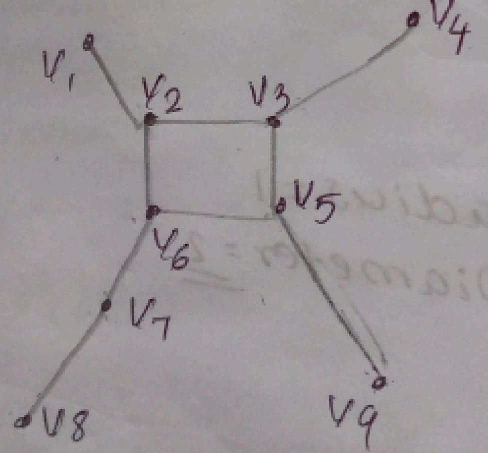
$\mathcal{E}(D) = 2$ $\therefore$ C is the central point.

$\mathcal{E}(E) = 2$

centre of a graph

set of central point is called centre of a graph / It is the set of vertices with eccentricity equal to the graph's radius. The centre of a graph is the set of all vertices of minimum eccentricity.

Eg:



$\mathcal{E}(V_1) = 4$

$\mathcal{E}(V_2) = 3$

$\mathcal{E}(V_3) = 4$

$\mathcal{E}(V_4) = 5$

$\mathcal{E}(V_5) = 3$

$\mathcal{E}(V_6) = 3$

$\mathcal{E}(V_7) = 4$

$\mathcal{E}(V_8) = 5$

$\mathcal{E}(V_9) = 4$

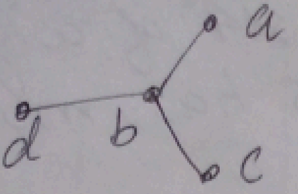
$R(G) = 3$

$D(G) = 5$

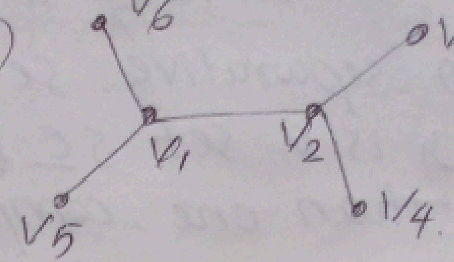
Here $E(V_5) = \mathcal{E}(V_6) = E(V_2) = R(G)$.

$\therefore \{V_5, V_2, V_6\}$ is the centre of the graph problem

1)



2)



$\varepsilon(a) = 2$
$\varepsilon(b) = 1$
$\varepsilon(c) = 2$
$\varepsilon(d) = 2$
$R(G) = 1$
$R(G) = \varepsilon(b)$
$\therefore \{b\}$ is the centre of graph.

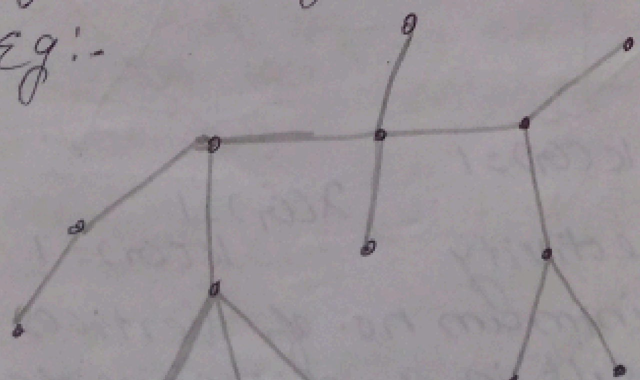
$\varepsilon(V_1) = 2$
$\varepsilon(V_2) = 2$
$\varepsilon(V_3) = 3$
$\varepsilon(V_4) = 3$
$\varepsilon(V_5) = 3$
$\varepsilon(V_6) = 3$
$R(G) = 2$ $\quad R(G) = \varepsilon(V_1) = \varepsilon(V_2)$.

$\therefore \{V_1, V_2\}$ is the centre of graph

spanning forest

A spanning forest is a forest that spans all of the vertices meaning only that each vertex of the graph is a vertex in the forest. Even a connected graph may have a disconnected spanning forest, such as the forest in which each vertex form a single vertex tree.

Eg:-



connectivity and separativity

A separative set / vertex set of a graph G is a set $S \subseteq V(G)$: $G-S$ has more than one component.

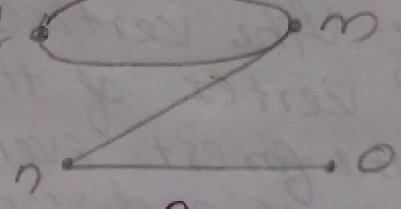
The connectivity of G written $k(G)$ is the minisize of a vertex set S: $G-S$ is disconnected or has only one vertex.

Edge connectivity

A disconnecting set of edges is a set $F \subseteq E(G)$: $G-F$ has more than one component. the edge connectivity of G is written $\lambda(G)$ is the minisize of a disconnecting set. A graph has λ edges connected every disconnecting set has atleast λ edges.

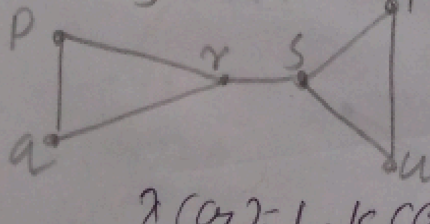
Eg: G_1

G_2.



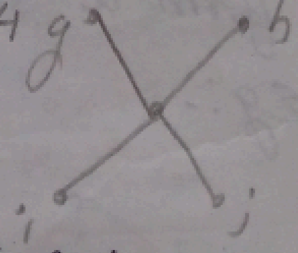
$\lambda(G)=1$
$k(G)=1$

G_3



$\lambda(G)=1, k(G)=1$

G_4



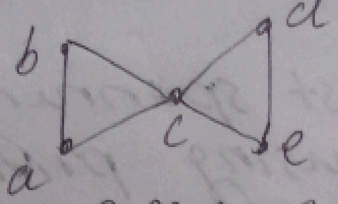
$\lambda(G)=1$
$k(G)=1$

vertex connectivity

It is the minimum no. of vertices whose removal result in a disconnected graph it is denoted by $k(G)$.

Problem

find $\lambda(G)$ and $k(G)$ of.



$\lambda(G) = 2$

$k(G) = 1.$

PRIM'S ALGORITHM

I) choose a vertex V_1 of R. Let $V = \{V_1\}$ and $E = \{\}$.

II) choose a nearest neighbour V_i of V that is adjacent to $V_i, V_j \in V$ and for which the edges (V_i, V_j) does not form a cycle with member of E. Add V_i to V and add (V_i, V_j) to E.

III) repeat step 2 until $|E| = n-1$. Then V contains all n vertices of R and E contains the edges of a minimal spanning tree for R.

(let R be a symmetric, connected relation with n vertices).

KRUSKAL'S Algorithm

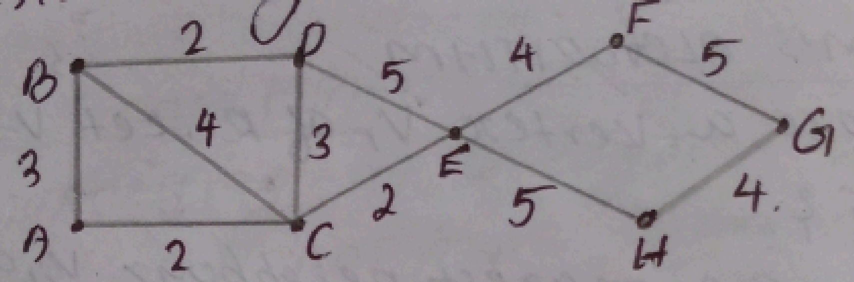
Let R be a symmetric, connected relation with n vertices and let $S = \{e_1, e_2, \ldots e_k\}$ be the set of weighted edges of R.

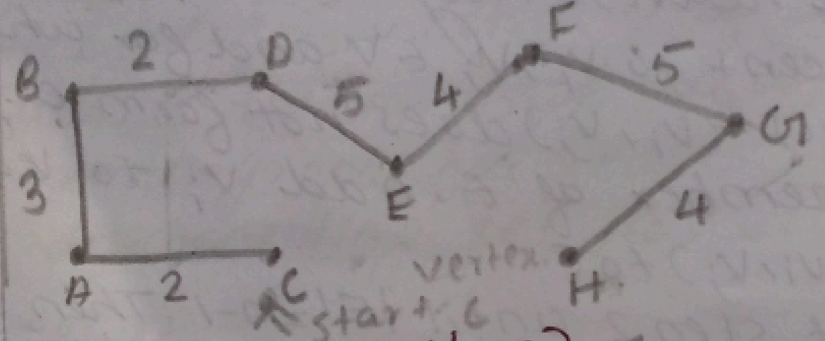
step 1: choose an edge e_1 in S of least weight. Let $E = \{e_1\}$. Replace S with $S - \{e_1\}$.

step 2: select an edge e_i in S of least weight that will not make a cycle with members of E. Replace E with

E ∪ {e_i} and S with S − {e_i}.
step 3: Repeat step 2 until $|E| = n-1$.

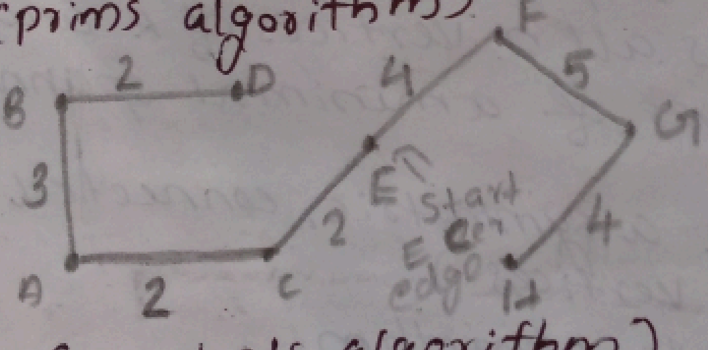
1 Q) find the minimum cost spanning tree of the given graph using prims and kruskals algorithm.





V = 8
E = 8 − 1 = 7

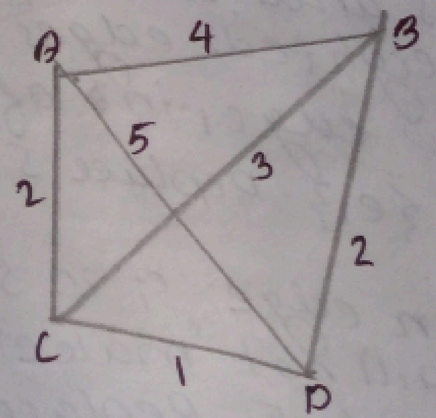
(prims algorithm)



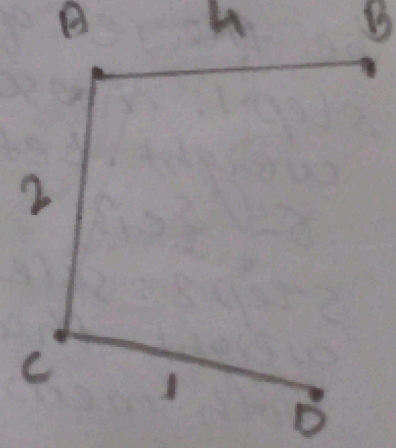
V = 8
E = 8 − 1 = 7

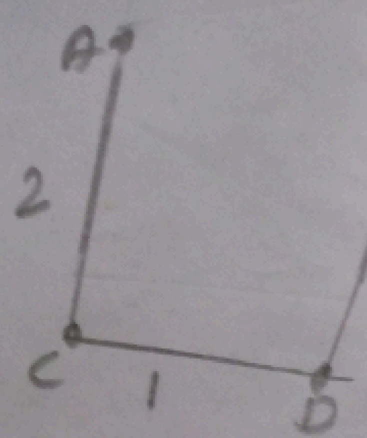
(kruskals algorithm)

2Q)



prims algorithm





kruskals

5) Let G be the graph shown in figure A 60. Begin at K.

i) find the minimum cost spanning tree using prim's method

ii) using kruskals method.

6) Let G be the graph shown in figure A Begin at M.

i) find the minimum cost spanning tree using prims method

ii) using kruskals method,

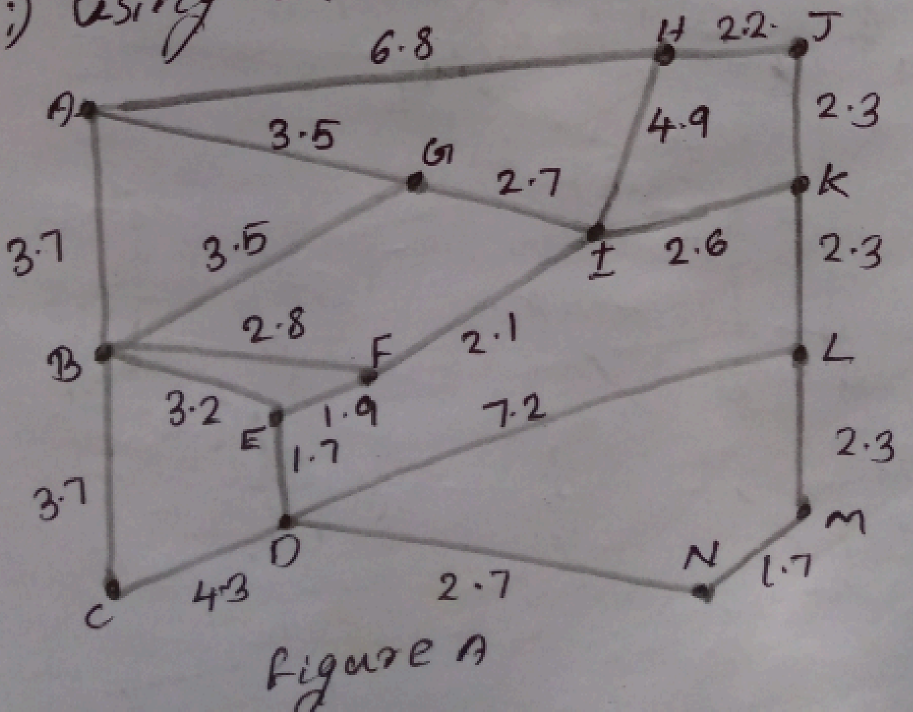
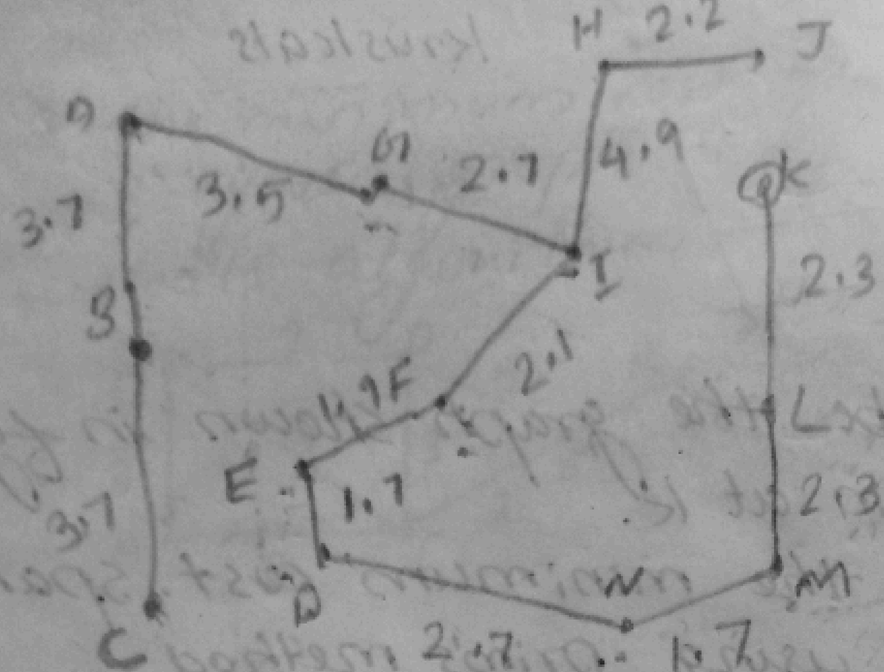
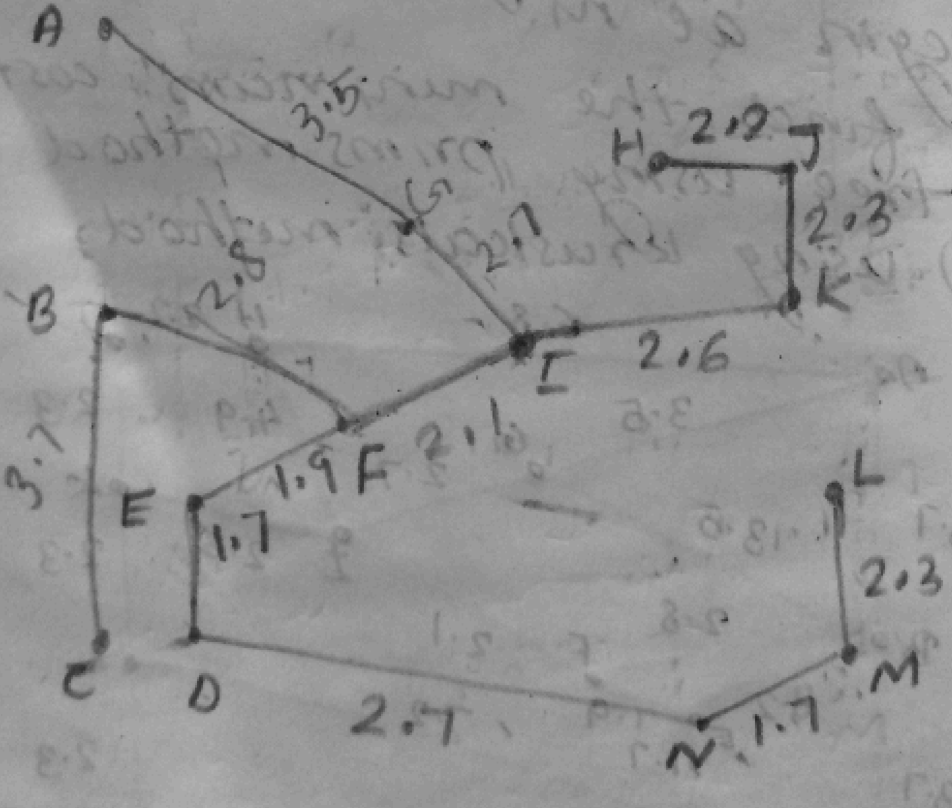


figure A

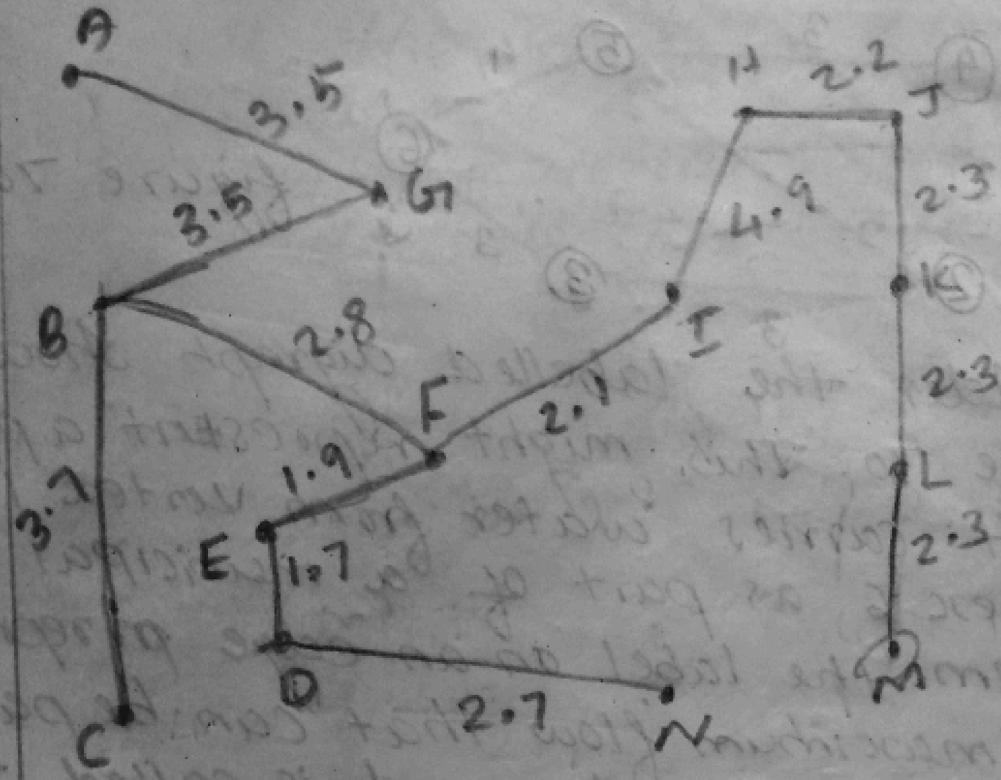
5) i)


H 2.2 J
A
G
2.7
4.9
3.7
3.5
K
I
2.3
B
2.1
1.9 F
L
E
1.7
2.3
3.7
D
N
M
C
2.7
1.7

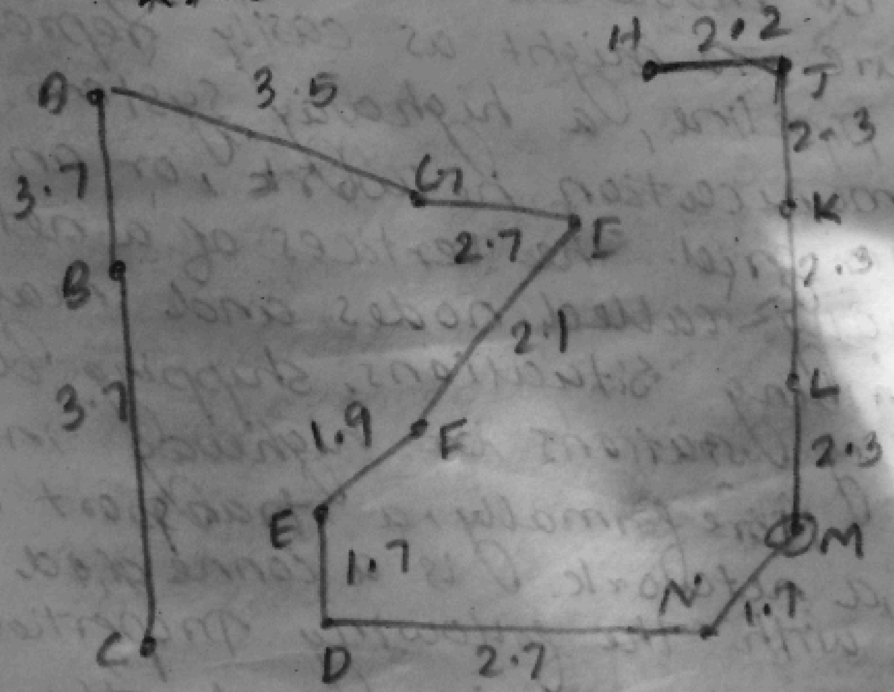
ii)


A
3.5
H
2.2
J
G
2.3
2.7
2.8
B
K
I
2.6
3.7
2.1
1.9 F
L
E
1.7
2.3
C
D
M
2.7
N
1.7

5) i)



ii) kruskals



Transport Networks

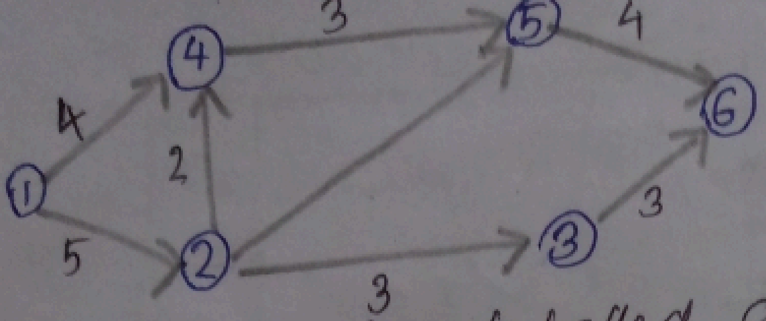


figure 70.

consider the labelled digraph shown in figure 70. This might represent a pipeline that carries water from vertex 1 to vertex 6 as part of a municipal water system. The label on an edge represents the maximum flow that can be passed through that edge and is called the capacity of the edge. Many situations can be modelled in this way. For instance figure 70 might as easily represent an oil pipeline, a highway system, a communication network, or an electric power grid. The vertices of a network are usually called nodes and may denote pumbing situations, shipping depots, relay stations or highway interchange

=> More formally, a transport network, or a network. is a connected digraph N with the following properties:

a) There is a unique node, the source, that has in-degree 0. we generally label the source node 1.

b) There is a unique node, the sink, that has out degree 0. If N has n nodes. we generally label the sink as node n.

c) ~~The~~ The graph N is labelled. The label, C_{ij} on edge (i,j) is a nonnegative number called the capacity of the edge. For simplicity we also assume that all edges carry material in one direction only that is, if (i,j) is in N, the (j,i) is not.

<u>Flows</u>

Mathematically, a **flow** in a network N is a function that assigns to each edge (i,j) of N a nonnegative number F_{ij} that does not exceed C_{ij}. Intuitively, F_{ij} represents the amount of material passing through edge (i,j) we also require that for each nod other the source and sink, the sum of the F_{ik} on edges entering node k must be equal to the sum of the F_{kj} on edges leaving node k. This means that material cannot accumulate, be created, dissipte or be lost at any node other than the source or the sink. This is called conservation of flow.

⇒ A ~~cons~~ consequence of this requirement is that the sum of the flows leaving the source must equal the sum of the flows entering the sink. This sum is called the value of the flow, written value (F). We can represent a flow F by labelling each edge (i,j) with the pair (C_{ij}, F_{ij}). A flow F in the network 70 shown in figure represented by ~~by~~ figure 71.

Example 1

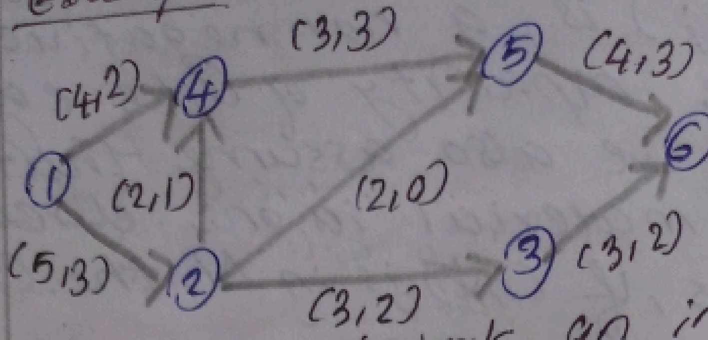


Figure 71

For any network an important problem is to determine the maximum value of a flow through the network and to describe a flow that has the maximum value. For obvious reasons this is commonly reffered to as the maximum flow problem.

Example 2

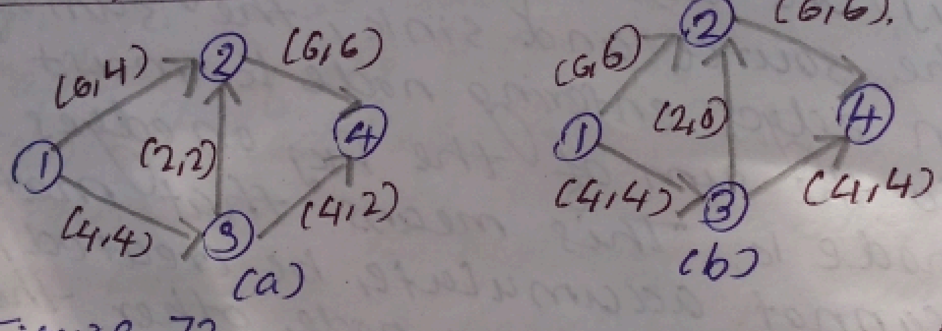


Figure 72

Figure 72 (a) shows a flow that has value 8. Three of the five edges are carrying their maximum capacity. This seems to be a good flow function but figure 72(b) shows a flow with value 10 for the same network.

Max-flow min-cut theorem

The max-flow min-cut theorem states that in a flow network, the maximum amount of flow passing from the source to sink is equal to the total weight of the edges in a minimum cut, ie, the smallest total weight of the edges which is removed would disconnect the source from the sink.

Labelling algorithm → is a greedy approach for calculating the max possible flow in a network or a graph.

The ford-fulkerson algorithm (labelling) is used to detect maximum flow from start vertex to sink vertex in a given graph. In this graph, every edges has the capacity. Two vertices are provided named source and sink. The source vertex has all outward edge, no inward edge, and the sink will have all inward edge no outward edge.

Augmenting path (Terminologies used).
It is the path available in a flow network.

Residual graph
It represents the flow network that has additional possible flow).

Residual capacity Flow/capacity.
It is the capacity of the edge after substracting the flow from the maximum capacity.

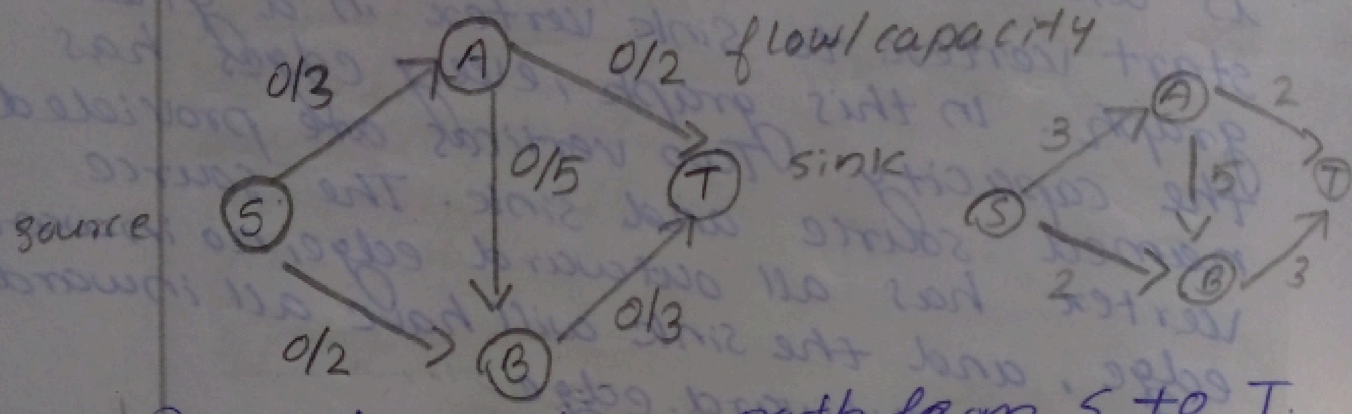
Algorithm follows:
1. Initialize the flow in all the edges to 0.
2. while there is an augmenting path between the source and the sink add this path to the flow

3. update the residual graph.

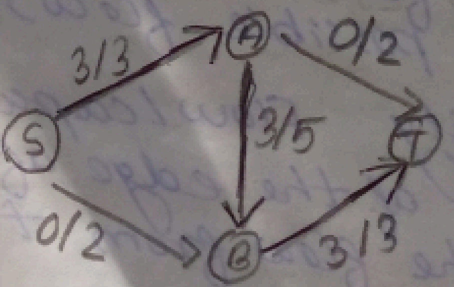
Note: we can also consider reverse path if required because if we donot consider them, we may never find a maximum flow

Example

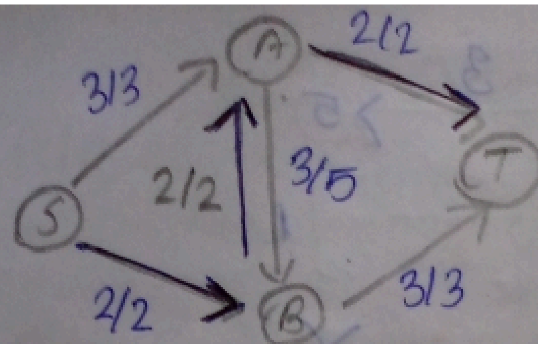
1) initial flow is set to 0



2) select an arbitary path from S to T. in this step we have selected a path S-A-B-T. The minimum capacity among the 3 edge is ~~time~~ 3 (S-A), update the flow capacity for each step (Residual capacity).



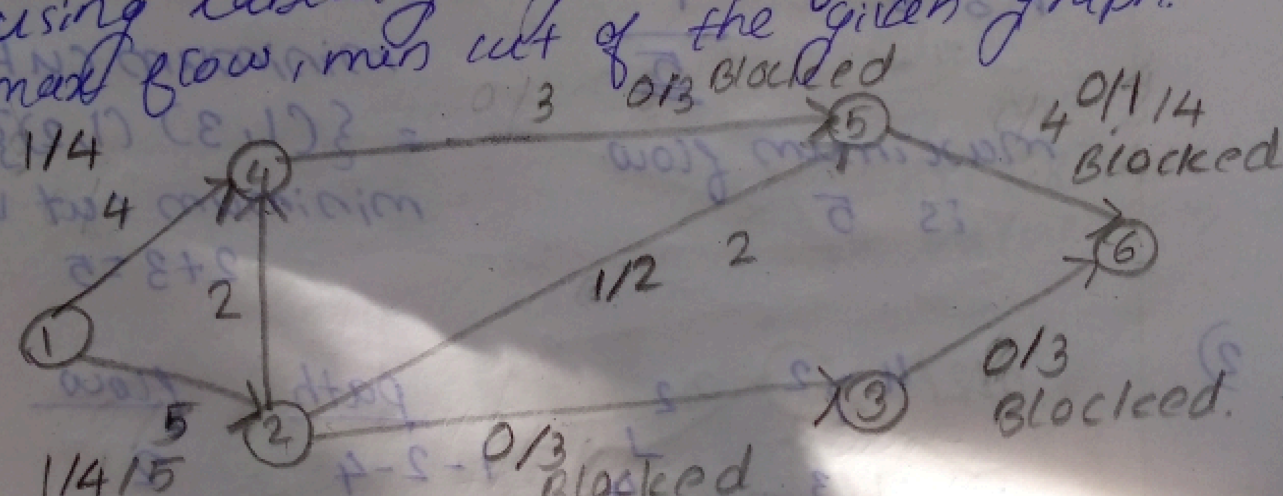
3) Let us consider reverse path B-A as well. selecting path S-B-A-T. The minimum capacity among the three edge is 2. update the residual capacity for each step.



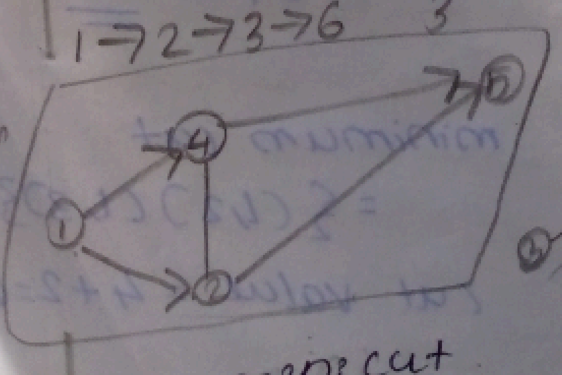
4) Adding all the flows = 3+2=5, which is the maximum possible flow on the flow network.

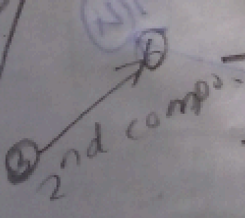
note: if the capacity for any edge is full, then that path cannot be used.

Q) using labelling algorithm find the max flow, min cut of the given graph.



path	flow
1→4→5→6	3
1→2→5→6	1
1→2→3→6	3





maximum flow

3+
1
3

7

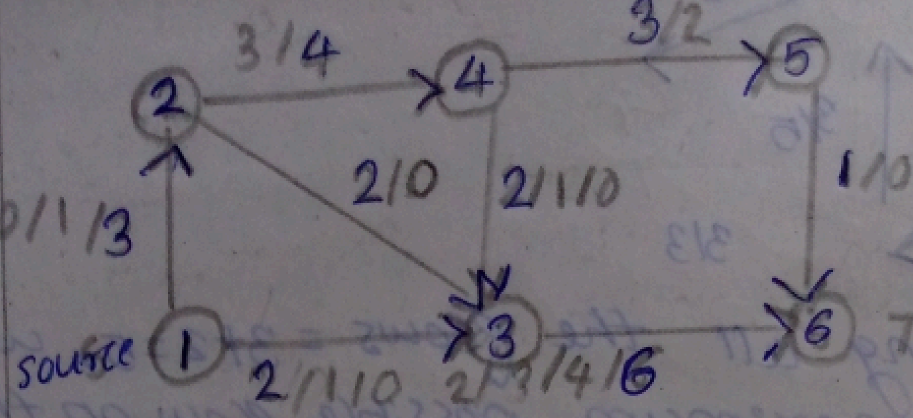
mini cut

{(2,3), (5,6)}

minimum cut value

3+4 = 7

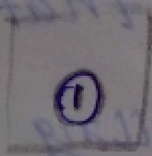
1)



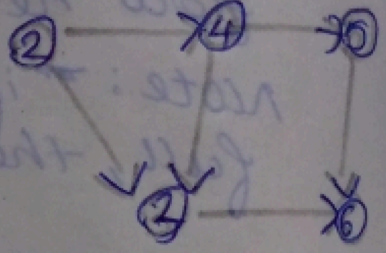
path	flow
1-2-3-6	2
1-2-4-3-6	1
1-3-4-5-6	1
1-3-6	1
	5

maximum flow is 5

1st component

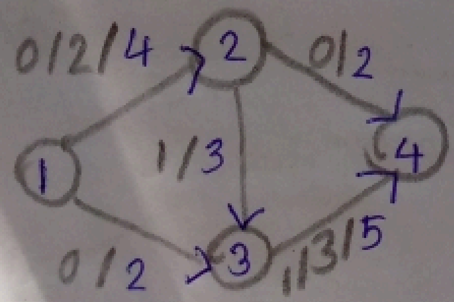


2nd.



minimum cut
$= \{(1,3)\ (1,2)\}$.
minimum cut value
$2+3=5$

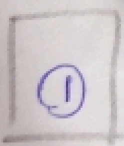
2)

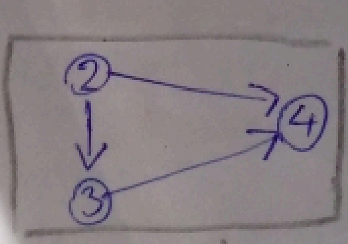


path	flow
1-2-4	2
1-2-3-4	2
1-3-4	2

maximum flow
$=2+2+2=6$

1st component

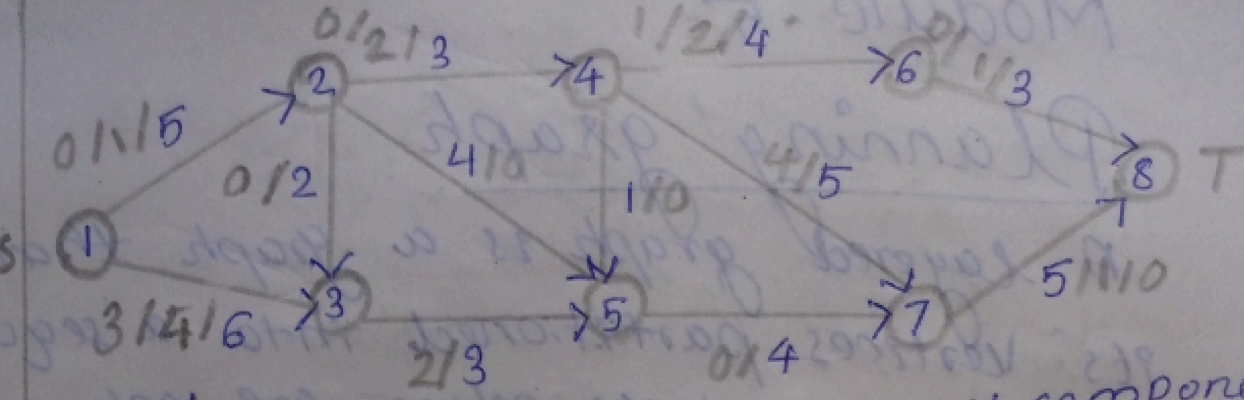




second component

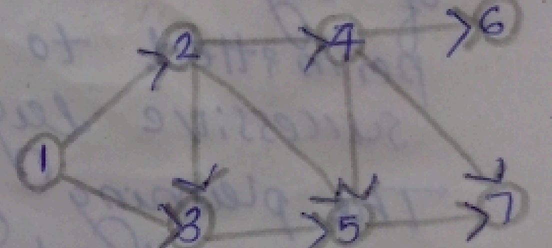
minimum cut
$= \{(1,2)\ (1,3)\}$
cut value $= 4+2=6$

3)



path	flow
1-2-5-7-8	4
1-2-4-7-8	1
1-3-2-4-6-8	2
1-3-5-4-6-8	1

1st componet



2nd component.

$\boxed{8}$

maximum flow
$4+1+2+1=\underline{8}$

minimum cut
$=\{(6,8)\ (7,8)\}$
minimum cut value $= 3+5=8$
minimumcut