Homework 5

1)

a) Vertex connectivity for the complete graph k(G) is n-1
Edge connectivity for the complete graph k'(G) is n-1

b) Vertex connectivity for the cycle graph k(G) is 2
Edge connectivity for the cycle graph k'(G) is 2

c) Vertex connectivity for the complete bipartite graph(Kn,m) k(G) is =minimum(n,m)
Edge connectivity for the complete bipartite graph(Kn,m) k'(G) is =minimum-degree(G) or minimum(n,m)

d) Vertex connectivity for the cube graph (Q_r) k(G) is r
Edge connectivity for the cube graph (Q_r) k'(G) is r

2)

(6.8)

2

6.8) known relationships are:

$$\kappa(G) \le \kappa'(G) \le \delta(G) \text{ \& } \quad — 1$$

$$\delta(G) \le d_G(u) \;\forall\, u \in V(G) \quad — 2$$

Hand-shaking lemma says,

$$\sum_{v \in V} deg(v) = 2|E|$$ where, $v \in V$, all vertices in graph 'G' and $|V| = n$

$\delta(G)$ minimum-degree of any node in G is $\left\lfloor \frac{2|E|}{n} \right\rfloor$

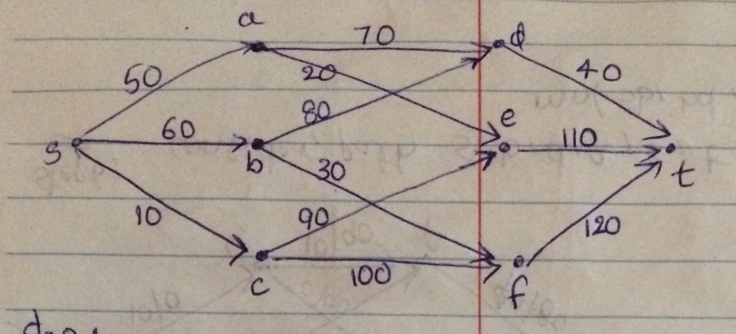
Substitute $\delta(G)$ as $\frac{2E}{n}$ in relation 1

$$\kappa(G) \le \kappa'(G) \le \left\lfloor \frac{2|E|}{n} \right\rfloor$$

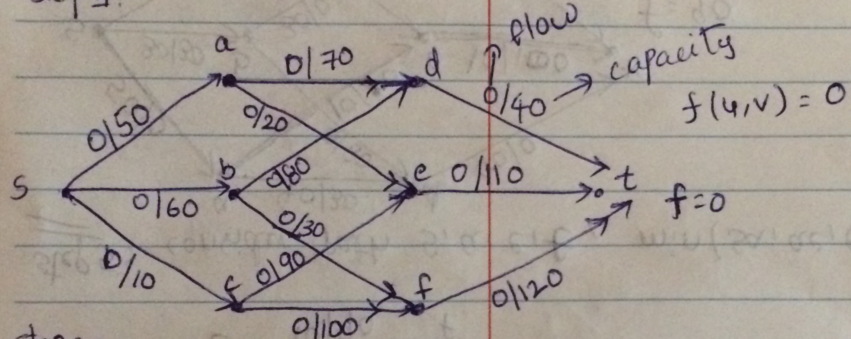
$m = |E|$

$$\Rightarrow \kappa(G) \le \kappa'(G) \le \left\lfloor \frac{2m}{n} \right\rfloor$$

6.33)



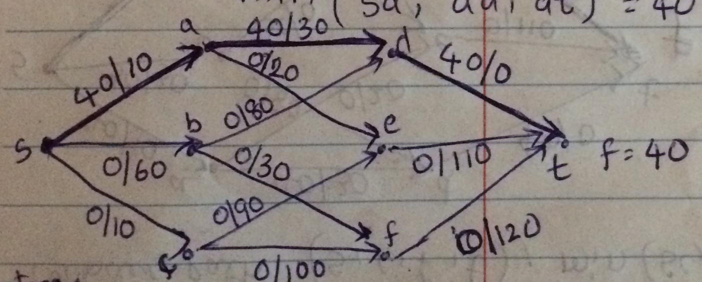
step 1:



step 2:

consider, path from s to t is, {s, a, d, t}

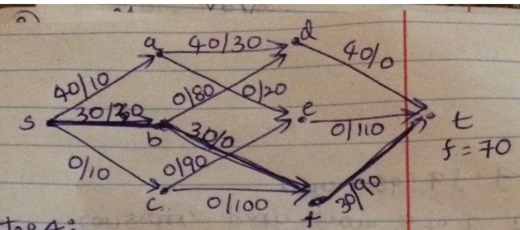
min (sa, ad, dt) = 40



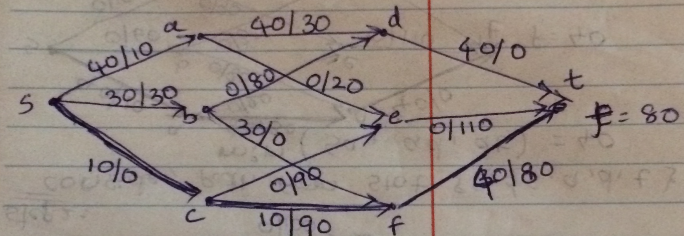
step3:

consider, path from 's' to 't' is {s, b, f, t}

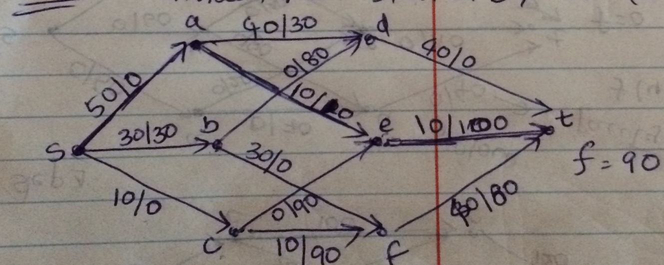
min (sb, bf, ft) = 30



step 4: consider, path (s, c, f, t), $\min(sc, cf, ft) = 10$

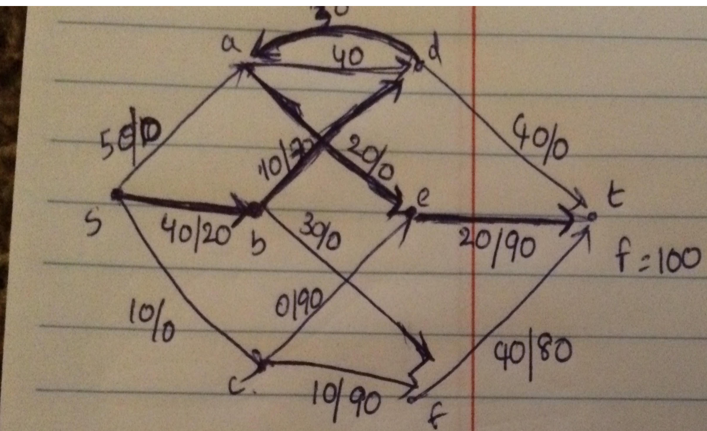


step 5: consider, path s, a, e, t, $\min(sa, ae, et) = 10$



step 6: consider, path s, b, d, a, e, t

$\min(sb, bd, da, ae, et) = 10$



we can't consider any more path from s to t

$\therefore$ Max flow = 100

→ min_cut = {sc, bf, ae, dt} which has capacity = 100
(s-t)

⇒ maxflow obtained from ford-fulkerson and minimum s-t cut is equal, hence verified.

3)

a) We can model this problem into undirected graph with n nodes and m edges, G(V,E) where |V|=n and |E|=m. In the graph, nodes represent the corners and edges represents streets.

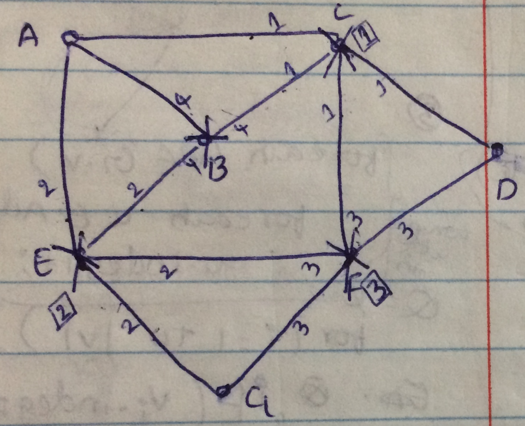
b) Vertex cover for the graph can be used to solve the problem. A vertex cover number for the graph gives the minimum number of cameras required for covering all streets. Vertex cover on a graph is A set of nodes L in a graph G = (V,E) is a vertex cover if every edges in E(G) has an endpoint in L.
The vertex cover number *beta (G)* of a graph G is the number of nodes in one of G's minimum vertex covers.

c) Minimum number of cameras need in the following graph is, minimum vertex cover on the graph. Minimum vertex cover is beta (G)= {E, B, C, F}.
That is, |beta (G)|=4.

Number of cameras needed is to cover all edges is 4.

3)

(c)

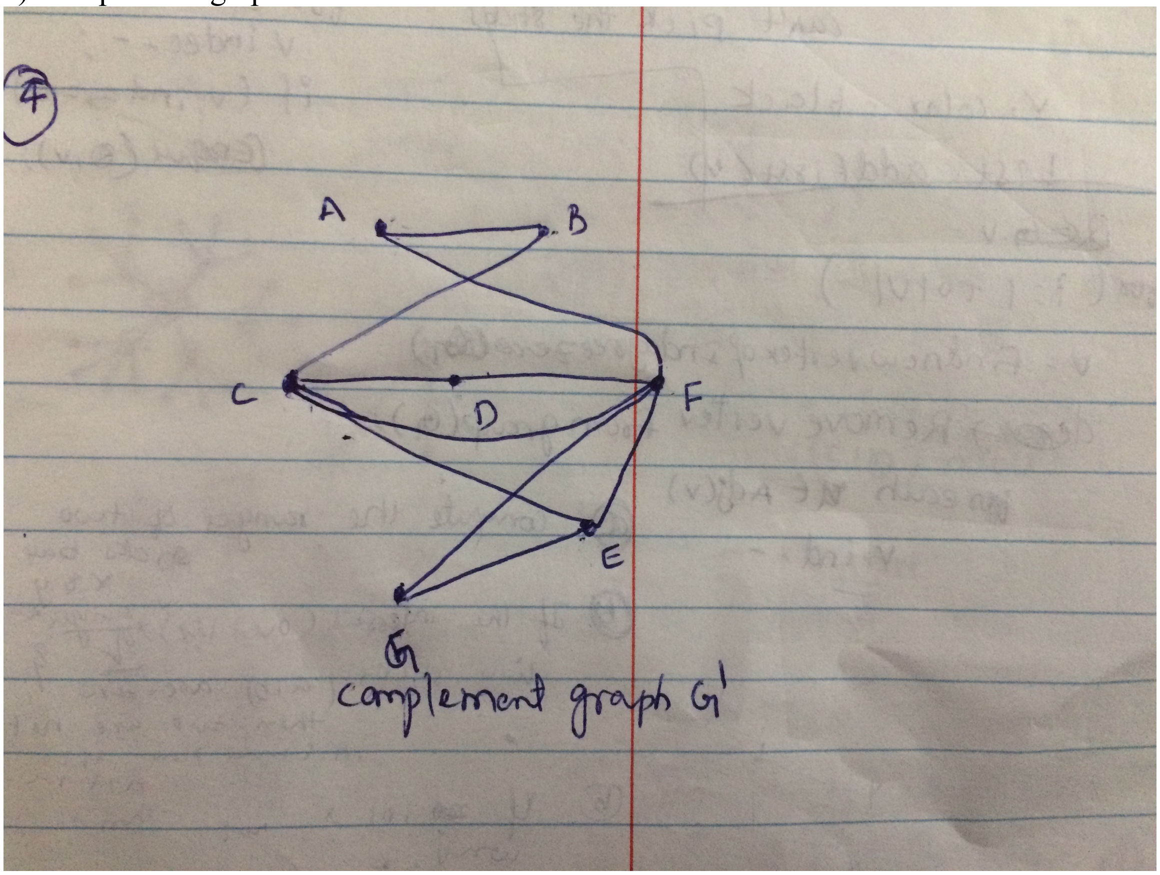


min vertex cover is = { E, B, C, F }

B(G) = 4

4)

a) Complement graph is:



b&c)

(b) 4 (A F), (B C), D, (G, E)

P_1	P_2
A	F
B	C
D	
G	E

Above matching doesn't give correspond to 2p feasible scheduling.

(c) 4 AB, CD, FG, E

P_1	P_2
A	B
C	D
F	G
E	

Above matching gives feasible 2p scheduling

d) No of edges in matching are 3. Where, matching in the graph is M={AC, DG, BG} and feasible schedule S={AB, CD, FG, E} is 4. No of nodes (n) in the graph are 7.

The relationship, S=n-|M|

=7-3➔4

(4) d

$S = n - |M|$

$n \to 7$

$|M| \to 3$; where, $M = \{AB, CD, FG\}$

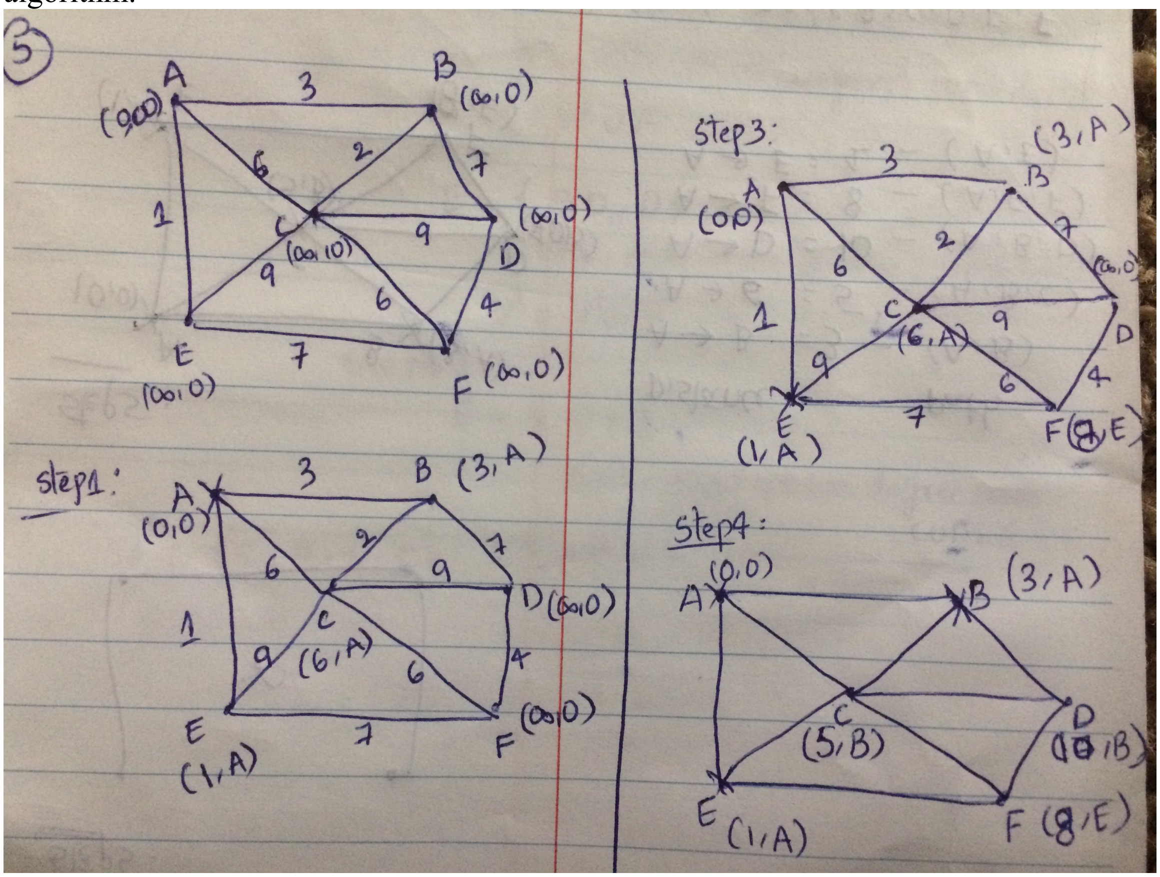
$S = n - |M|$ Feasible schedule: $\{AB, CD, FG, E\}$

$= 4$ $|S| = 4$

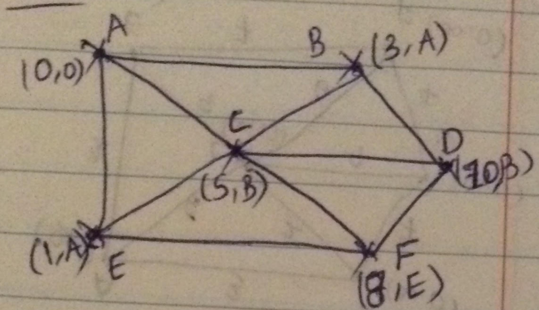
$\therefore$ $S = n - |M|$ is verified

5) The Prims algorithm searches for the minimum spanning tree, only cares about the minimum of the total edges cover all the vertices. so it looks like: v.key = w(u,v) The Dijkstra's, which searches for the minimum path length, so it cares about the edge accumulation. So it looks like :v.key = w(u,v) + u.key.

In the Prims algorithm cost to reach each node is calculated from minimum weighted edge from its neighbor node. This does not include path cost from the source node. If the minimum path cost from source is calculated on each edge it will be turned in Dijkstra's algorithm.



Steps:



Distance		Path
A → B	= 3	(A,B)
A → C	= 5	(A,B,C)
A → D	= 10	(A,B,D)
A → F	= 8	(A,E,F)
A → E	= 1	(A,E)

Similarly, distance from source node B; C, D, E, F

node F:
F → A = 8
F → B = 8
F → C = 6
F → D = 4
F → E = 7

node B: Path
B → A = 3 → {BA}
B → C = 2 → {BC}
B → D = 7 → {BD}
B → E = 4 → {BA, AE}
B → F = 8 → {BC, CF}

node D:
D → A = 10 = (D,B,A
D → B = 7 = (D,B)
D → C = 9 = (D,C)
D → E = 11 = (D,F,E)
D → F = 4 = (D,E)

from node C:
C → A = 5 → (C,B,A)
C → B = 2 → (C,B)
C → D = 9 → (C,D)
C → F = 6 → (C,F)
C → E = 6 → (C,B,A,E)

node E:
E → A = 1 = (E,A)
E → B = 4 = (E,A,B)
E → C = 6 = (E,A,B,C)
E → D = 11 = (E,F,D)
E → F = 7 = (E,F)

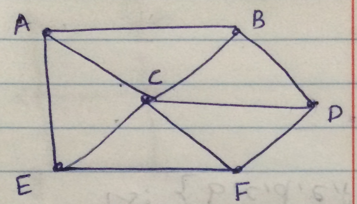
∴ Since, graph is undirected from node F to all node can be derived.

6) (a)
This heuristic always finds a vertex cover. Heuristic, in each iteration it adds highest degree node to vertex cover set, which covers all edges associated with it. Since, it will add vertices to vertex cover set until all edges are covered. Once all the edges are covered, it will return vertex cover set. So, all edges will be covered that defines feasible vertex cover set. But this heuristic doesn't always give minimum vertex cover.

(b) For below graph algorithm computes efficient vertex cover.

(6)

(b)



$V' = \emptyset$

degrees

C - 5

A - 3

B - 3

D - 3

E - 3

F - 3

$E = \{AB, BC, AC, CD, BD, DF, AE, EF, CE\}$

Step 1: maximum degree node is C

$V' = V' \cup C$

Step2:

$E = \{AB, BD, DF, EF, AE\}$

Next, maximum degree, consider A

$V' = V' \cup A$

Step3:

$E = \{BD, DF, EF\}$

Next, maximum degree, consider D

$V' = V' \cup D$

Step4:

$E = \{EF\}$

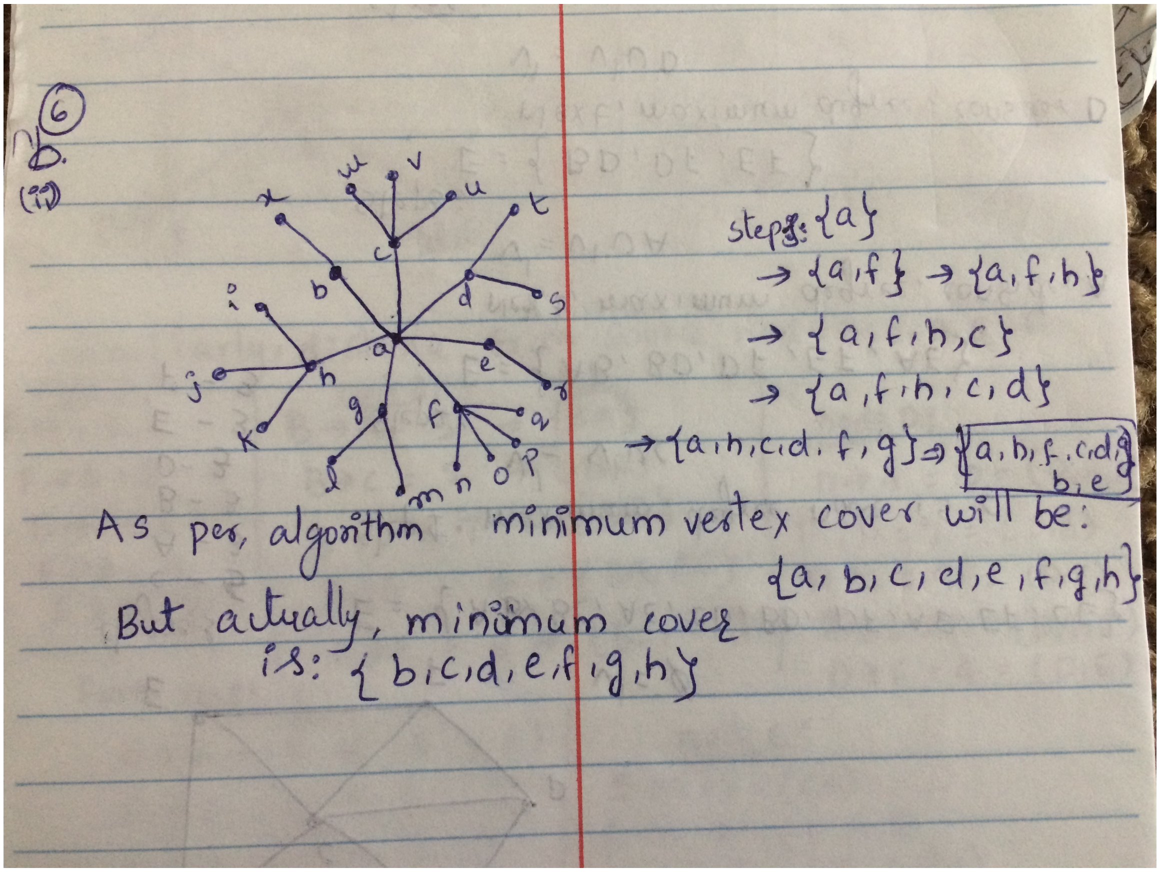
Net, maximum degree node, consider E

$V' = V' \cup E$

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

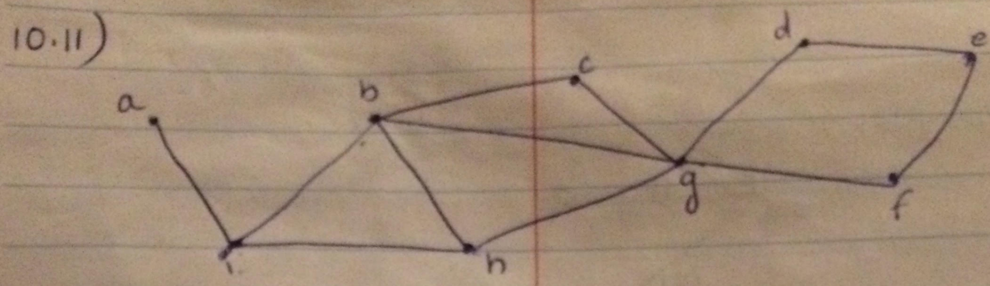
$\beta(G) = 4$

b) For below graph algorithm does not computes minimum vertex cover.



7)
(10.11 & 10.36)

10.11)



Dominance set for the above graph is:

$\gamma(G) = \{i, b, e\} = |\gamma(G)| = 3$

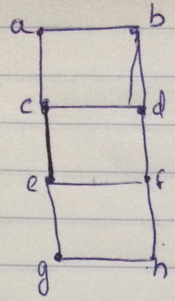
10.36) consider M' is the Maximum matching in the graph & M is maximal matching of the graph.
Assume, M' has size k. for every edge 'e' one of the edge end points must be matched in M & M'. If not we can add the edge to Matching. Each edge in the (M) maximal matching can be adjacent to at most two edges in maximum matching (M'). Since, M is maximal matching. And, every edge in M is adjacent to or in in M'(maximum), by maximality of M'.
consider, In worst case, If every edge of M (maximal) is adjacent to two edge in M'(maximum). Then M' will be of size 2M.

Hence:

$M' \leq 2M$

8)a)

8
a)



(i) consider, a Matching(M) = {ec}

(ii) but there exist an augmenting path (P): ac, ce, eg

∴ New matching(M) = {ac, eg}

(iii) again, ∃ an augmenting path (P): ba, ac, ce, eg, gh

∴ New matching (M) = {ab, ce, gh}

(iv) And, again ∃ an augmenting path (P) = {db, ba, ac, ce, eg, gh, hf}

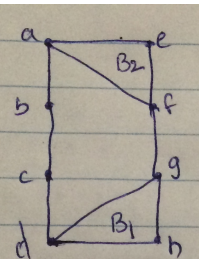
∴ New matching (M) = {db, ac, eg, fh}

⇒ Maximum matching is |M| = {ac, bd, eg, fh}

|M| = 4

8)(b) Below graph has odd cycles in the graph, Consider each cycle as single node and redraw the graph then find matching set, add matching from cycle to set.

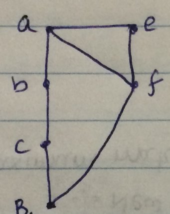
8(b).



Graph contains an odd cycle of length 3. draw an odd cycle Block as single node B1.

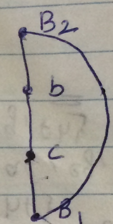
that is,

①



②

Similarly, with B2



Now apply, matching algorithm:

cosider Matching $M = \{bc\}$

$\exists$ an augumenting path $(P) = \{B_2b, bc, cB_1\}$

New Matching $(M) = \{B_2b, cB_1\}$

there is no augumenting path now,

cycle 3 has one matching.

$\therefore$ Matching $(M) = \{ab, ef, cd, gh\}$