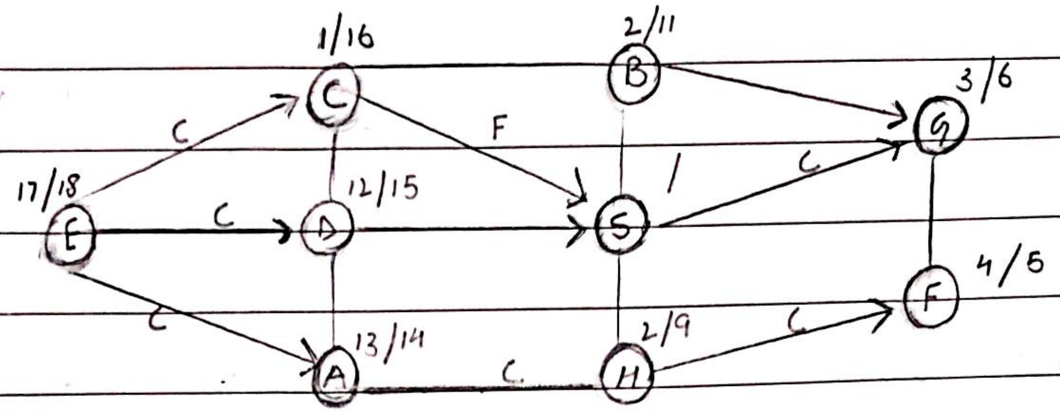
SEC-7 SHUBHAM SM1841

Date: / / /

1720032335

Q1. (a)



Start and finish time of vertices are
A(13,14), B(2,11), C(1,16), D(12,15), E(17,18),
F(4,5), G(3,6), H(8,9), S(7,10)

Three Edges:

CB, BG, GF, BS, SH, CD, DA

~~Bad~~ Back Edge

None

Forward Edge

CS

Cross Edges

EC, ED, EA, DS, AH, HF, SG

(b) E → C → D → A → B → S → H

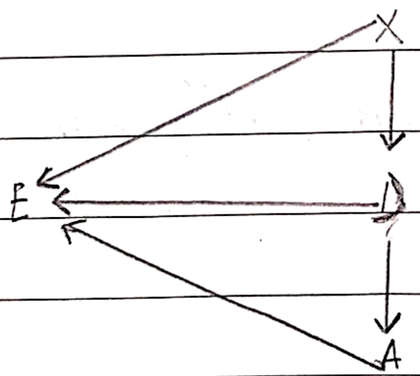
↓

F ← G

(c) Strongly connected components for graph: A, B, C, D, E, F, G, H, S which is total of 9

Now, reverse direction of edges D to A and E to D then strongly connected components are:- which is 6

C - D - E - A   B, F, G, H, S



Date: / / /

Q2. This problem can be converted to equivalent graph.

Consider person as vertex of graph. If two people ~~shake~~ shake hands then connect those equivalent vertices with edge. Here edge represents the ~~shake~~ shake hand between those two people. If two vertices are not connected to each other than 2 people did not shake hand. No. of people any person has shaken hands with is degree of equivalent vertex.

In other words, we have to prove that there are 2 vertices whose degree is same.

Remaining vertex are $\{v_1, v_2, v_3 --- v_{n-1}\}$

To preserve uniqueness, remaining people must shake hands with somebody. Remaining vertices must have one degree.

The degrees ~~can't~~ be more than 1 as no one will ~~that~~ shake hand with $(n-1)$ people

So, $(n-2)$ unique degrees to assign $(n-1)$ vertices.

Suppose that ~~we~~, we assign vertices $\{v_1, v_2, v_3 \ldots v_{n-2}\}$ unique degrees

Vertex v_{n-1} whose degree has not been assigned. So, we do any option other than to assign degree that has been assigned already.

So, it can be concluded that the degree of 2 people is same.

So, there exist people who have shaken same ~~not~~ number of hands.

Hence Proved.

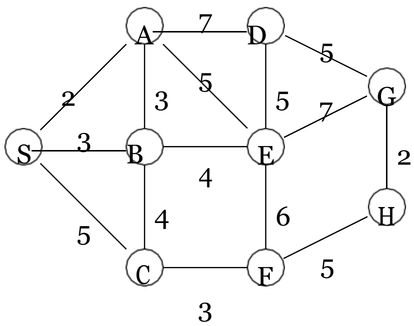
Problem 3. (20 pts) Suppose in the process of computing the MST of a graph with 17 vertices via Kruskal's algorithm we have obtained connected components having sizes 2, 4, 5, 6. Using weighted-union, give all possible scenarios as well the minimum and maximum number of possible operations needed to get an MST. (An operation is changing a link. For example, if the component with 2 elements gets added to the component with 4 elements, we change two links for the smaller set plus linking the head of shorter to the tail of the longer, for a total of 3 operations.)

1) **2 to 4 can be combined with 3 operations. We then can combine 5 to 6 with 6 operations. Then combine 6 to 11 with 7 operations. Total 16 operations.**
2) **2 to 4 can be combined with 3 operations. We then can combine 6 to 6 with 7 operations. Then combine 5 to 12 with 6 operations. Total 16 operations.**
3) **We can combine 2 to 5 with 3 operations. Then we can combine 4 to 7 with 5 operations. Then we can combine 6 to 11 with 7 operations. Total 15 operations.**
4) **We can combine 2 to 5 with 3 operations. Then we can combine 6 to 7 with 7 operations. Then we can combine 4 to 13 with 5 operations. Total 15 operations.**
5) **We can combine 2 to 6 with 3 operations. Then we can combine 4 to 8 with 5 operations. Then we can combine 5 to 12 with 6 operations. Total 14 operations.**
6) **We can combine 2 to 6 with 3 operations. Then we can combine 5 to 8 with 6 operations. Then we can combine 4 to 13 with 5 operations. Total 14 operations.**
7) **4 to 5 can be combined with 5 operations. We then can combine 2 to 9 with 3 operations. Then combine 6 to 11 with 7 operations. Total 15 operations.**
8) **We can combine 4 to 5 with 5 operations. Then we can combine 6 to 9 with 7 operations. Then we can combine 2 to 15 with 3 operations. Total 15 operations.**
9) **We can combine 4 to 6 with 5 operations. Then we can combine 5 to 10 with 6 operations. Then we can combine 2 to 15 with 3 operations. Total 14 operations.**
10) **We can combine 4 to 6 with 5 operations. Then we can combine 2 to 10 with 3 operations. Then we can combine 5 to 12 with 6 operations. Total 14 operations.**
11) **We can combine 5 to 6 with 6 operations. Then we can combine 2 to 11 with 3 operations. Then we can combine 4 to 13 with 5 operations. Total 14 operations.**
12) **We can combine 5 to 6 with 6 operations. Then we can combine 4 to 11 with 5 operations. Then we can combine 2 to 15 with 3 operations. Total 14 operations.**

Total number of cases is 12. Maximum number of operations is 16. Minimum number of operations is 14.

Problem 4. (50 pts) Show all your work on graphs. The edge weights are given.

(a) (15 pts) Compute MST using Kruskal's algorithm. Show the order of the selected edges as well as the sets formed. If there is a tie use alphabetical ordering.



We start by sorting the edges in Non-Decreasing order of weights.

Here is the sorted list: AS, GH, AB, BS, CF, BC, BE, AE, CS, DE, DG, FH, EF, AD, EG.

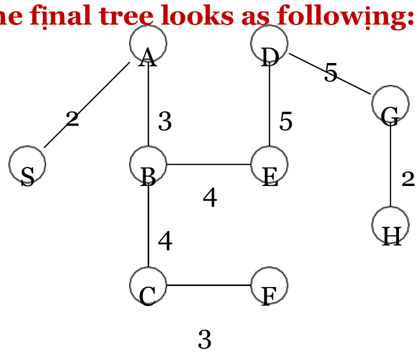
1) Start by picking the edge AS. Chosen edges will be AS. Linked list formed is A->S
2) Pick the edge GH. Chosen edges are AS and GH. Linked Lists formed are A->S and G->H
3) We pick the edge AB. Chosen edges are AS, GH, and AB. Linked Lists formed are A->S->B and G->H
4) Now we pick edge BS. Since BS are in the same list, we don't change anything.
5) We now pick edge CF. Chosen edges are AS, GH, AB, and CF. Linked Lists formed are A->S->B, G->H and C->F
6) We now pick edge BC. We join the two list with B and C. Chosen edges are AS, GH, AB, CF, and BC Linked Lists formed are

   A->S->B->C->F and G->H
7) We now pick edge BE. Chosen edges are AS, GH, AB, CF, BC, and BE. Linked List formed are A->S->B->C->F->E and G->H
8) We pick the edge AE. AE are in the same list, so we don't change anything.
9) We pick the edge CS. CS are in the same list, so we don't change anything.
10) We pick the edge DE. Chosen edges are AS, GH, AB, CF, BC, BE and DE. The Linked Lists formed are

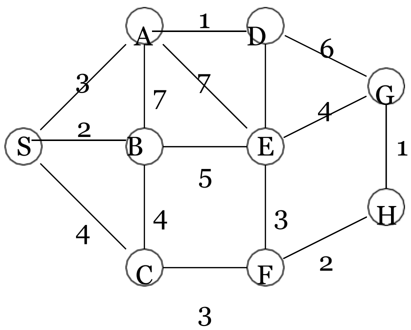
    A->S->B->C->F->E->D and G->H
11) We pick the edge DG. We join the two lists we have. Chosen edges are AS, GH, AB, CF, BC, BE, DE and DG. The Linked List formed is

    A->S->B->C->F->E->D->G->H

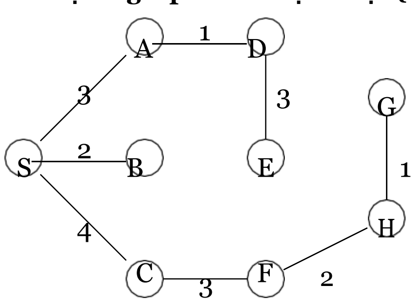
    The final tree looks as following:



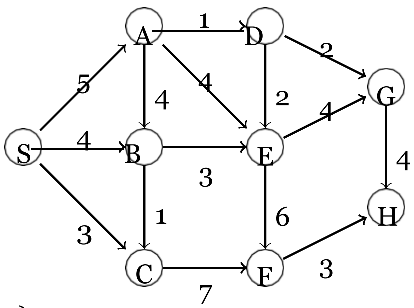
(b) (15 pts) Compute MST using Prim's algorithm. Start at *S* and give the order of the selected edges.



1) **We start by choosing S.**
   Result List{S}
2) **Pick the edge with minimum cost from S. Add (S, B) edge. Result List {S, B}**
3) **Pick the edge with minimum cost from S or from B that is not in the list yet. Add (S,A) to the list. Result List {S, B, A}**
4) **Pick the edge with minimum cost from S or B or A, that is not in the list. We can either add (S,C) or (A,D). We follow alphabetical order and add (S,C). Result List {S, B, A, C}**
5) **Pick the edge with minimum cost from B or A or C (S does not have any neighbor that has not been visited). Add (A,D) to the list. Result List {S, B, A, C, D}**
6) **Pick the edge with minimum cost from B or A or C or D. Follow the alphabetical order and add (D,E) to the list.**
   Result List is {S, B, A, C, D, E}
7) **Pick the edge with minimum cost from D or C or E. Add (C,F) to the list. Result List {S, B, A, C, D, E, F}**
8) **Pick the edge with minimum cost from D or E or F. Add (F,H) to the list. Result List {S, B, A, C, D, E,F, H}**
9) **Pick the edge with minimum cost from H or E or D. Add (H,G) to the list. Result List {S,B,A,C,D,E,F,H,G}**
   The final graph looks like this {SB, SA, SC, AD, DE, CF, FH, HG}



(c) (20 pts) Using Dijkstra's algorithm find the shortest path from S to all other vertices. Show the complete update history of each vertex v, starting with infinity as the value of $d(v)$, $v \mathrel{!}= S$, and finishing with the length of the shortest path.



1)

S	0
A	Null, inf
B	Null, inf
C	Null, inf
D	Null, inf
E	Null, inf
F	Null, inf
G	Null, inf
H	Null, inf

2) Consider S. Done list {S}

S	0
A	S, 5
B	S, 4
C	S, 3
D	Null, inf
E	Null, inf
F	Null, inf
G	Null, inf
H	Null, inf

3) Consider C. Done list {S, C}. d(C) = 3

S	0
A	S, 5
B	S, 4
C	S, 3
D	Null, inf
E	Null, inf
F	C, 10
G	Null, inf
H	Null, inf

4) Consider B. Done list {S, C, B}. d(B) = 4

S	0
A	S, 5
B	S, 4
C	S, 3
D	Null, inf
E	B, 7
F	C, 10
G	Null, inf
H	Null, inf

5) Consider A. Done list {S, C, B, A}. d(A) = 5

S	0
A	S, 5
B	S, 4
C	S, 3
D	A, 6
E	B, 7
F	C, 10
G	Null, inf
H	Null, inf

6) Consider D. Done list {S, C, B, A, D}. d(D) = 6

S	0
A	S, 5
B	S, 4
C	S, 3
D	A, 6
E	B, 7
F	C, 10
G	D, 8
H	Null, inf

7) Consider E. Done list {S, C, B, A, D, E}. d(E) = 7

S	0
A	S, 5
B	S, 4
C	S, 3
D	A, 6
E	B, 7
F	C, 10
G	D, 8
H	Null, inf

8) Consider G. Done list {S, C, B, A, D, E, G}. d(G) = 8

S	0
A	S, 5
B	S, 4
C	S, 3
D	A, 6
E	B, 7
F	C, 10
G	D, 8
H	G, 12

9) Consider F. Done list {S, C, A, D, E, G, F}. d(F) = 10

S	0
A	S, 5
B	S, 4
C	S, 3
D	A, 6
E	B, 7
F	C, 10
G	D, 8
H	G, 12

10) Consider H. Done list {S, C, A, D, E, G, F, H}. d(H) = 12

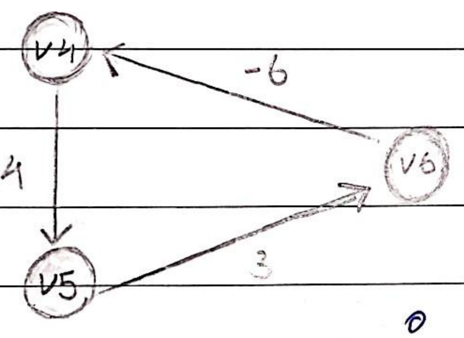
S	0
A	S, 5
B	S, 4
C	S, 3
D	A, 6
E	B, 7
F	C, 10
G	D, 8
H	G, 12

D.

Q5

There is no solution for above graph by Bellman Ford.

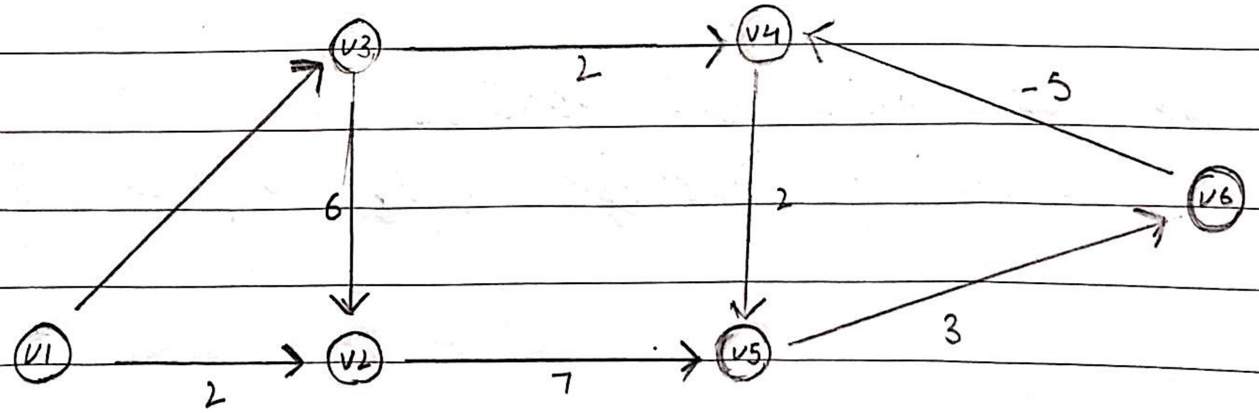
This is due to the fact it has negative cycles.



$2+3-6=-1$

$\therefore$ We don't get sol.

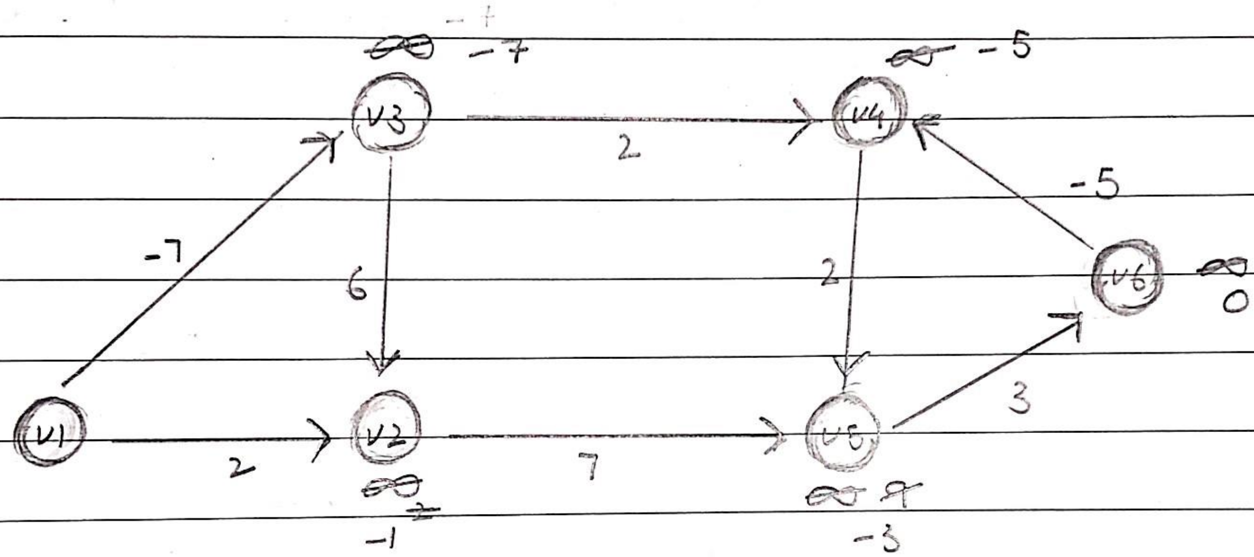
Changing weight edge (v_6, v_4) to -5



+TRỊ KAS

Date: / / /

Now due to this we can get a solution as there is no negative cycle



By using Bell man Ford Alg we are getting -ve values to final path.

So, we can't get solution by changing edge (v_6, v_4) to -5

Date: / / /

Q6. (a) Prims Algo →

Create a MST set ~~add~~ which is initially empty. Start at any vertex at random and add ~~to~~ MST. Find all edges connected to MST and univisited vertices and ~~add~~ add minimum of them to MST. Repeat that until vertices are covered in MST.

Complexity = O(V log V) or O(n log n) if we use fibonacci heap and adjecent list.

(b) Let nod that ain't in MST be v & E1 be edges adjecent to v. (E1 be max of n) T be the MST of n node Apply prims alg to whole graph and return netie MST.

Date: / / /

Complexity = $O(V \log V)$ or $O(n \log n)$ using fibonacci heap & adjacency list.

ⓒ Let Y & Z be unique MST of G. which is assumed to differs by 1 ~~degree~~ edge.
~~Let~~ Adding ~~Z~~ results in cycle X. So e1 & e2 are added to get minimum span tree but Z is in MST already.

So we can have only one MST for distinct weight edges.

MAT RIKRS

Problem 7. (20 pts) True-False.

(a) In an undirected graph the sum of the degrees of vertices is always twice the number of edges. (Note: The degree of a vertex is the number of edges that are incident to that vertex.)

True

(b) We can use BFS to decide if a graph is bipartite. (Note: A graph is bipartite if its vertices can be divided into two disjoint sets so that all edges go from a vertex in one set to a vertex in the other set.)

True

(c) Given a complete graph on n nodes, the number of cycles with at most 4 vertices is $O(2^n)$.

False

(d) We can use any minimum spanning tree algorithm to compute the maximum spanning tree, i.e. a spanning tree where the sum of the weights of its edges is maximum.

True