b) Prim's Algorithm-

i) Remove all self loop if exist in the graph G.

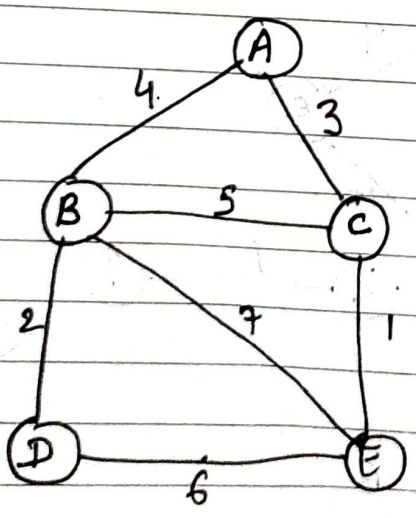
ii) If there exist paralled edges btw two vertices then delete them except that having min weight.

iii) Then find out the adjacency matrix for G.
- The weight of the edge joining v_i to v_i is 0.
- If there is no edge connecting v_i to v_j then the weight of the edge is ∞.
- If there is edge connecting v_i to v_j then the weight of the edge is the actual weight.

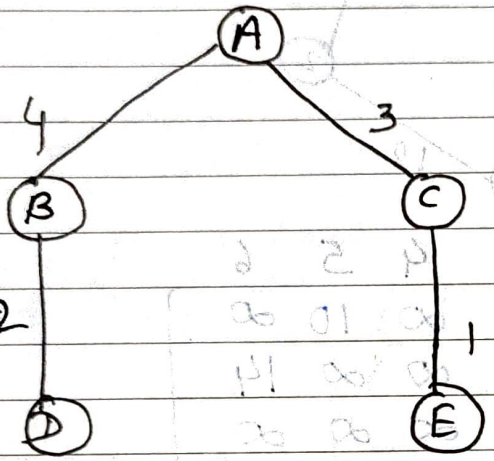
iv) Choose any arbitrary node as root node.

v) Check outgoing edges and select the one with less cost. Now treat this selected edge as one node and again check for all edges going out from it and so on.

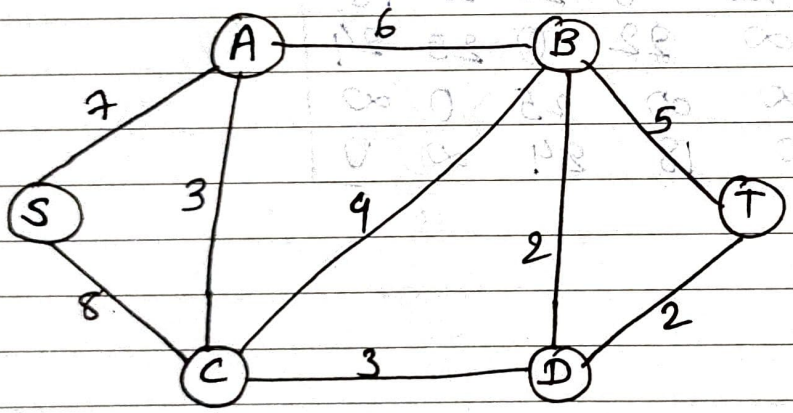
Eg



	A	B	C	D	E
A	0	4	3	∞	∞
B	4	0	5	2	7
C	3	5	0	∞	1
D	∞	2	∞	0	6
E	∞	7	1	6	0



Eg.



	S	A	B	T	D	C
S	0	7	∞	∞	∞	8
A	7	0	6	∞	∞	3
B	∞	6	0	5	2	4
T	∞	∞	5	0	2	∞
D	∞	∞	2	2	0	3
C	8	3	4	∞	3	0

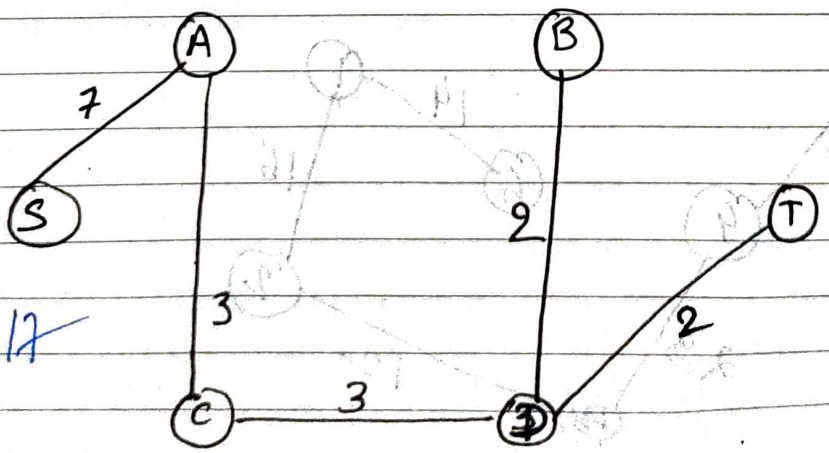
S-7-A

S-7-A-3-C

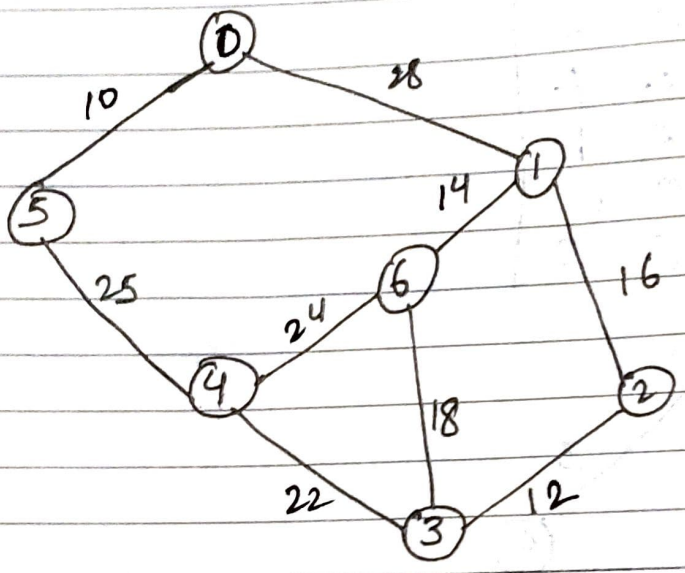
C-3-D

D-2-B

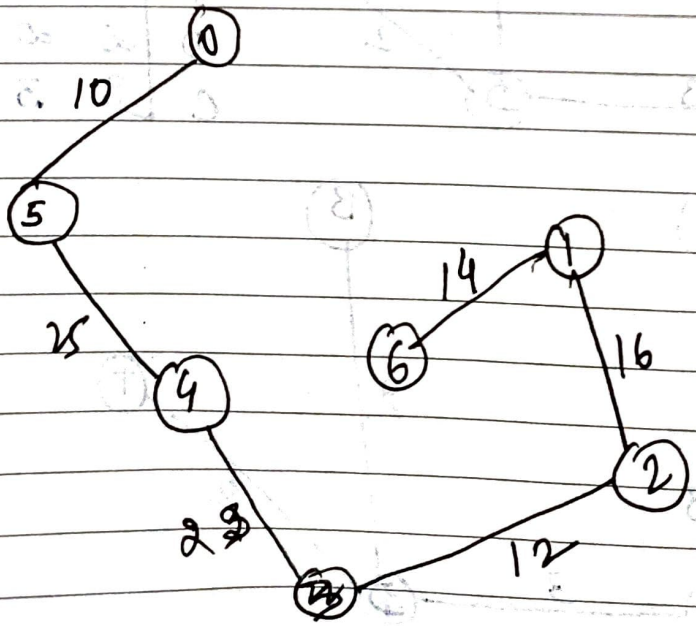
17



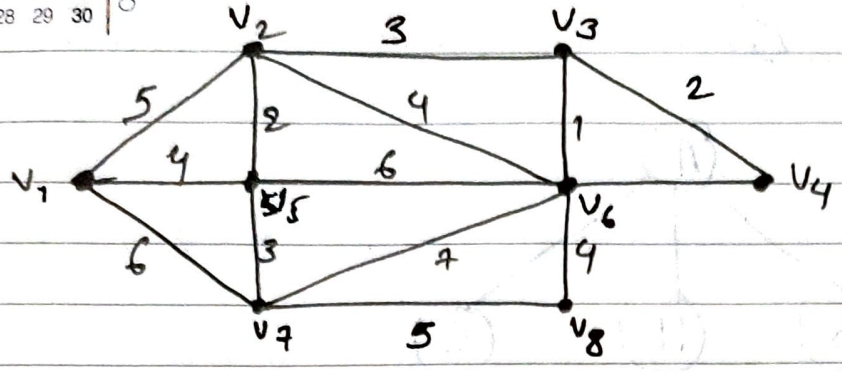
Q.



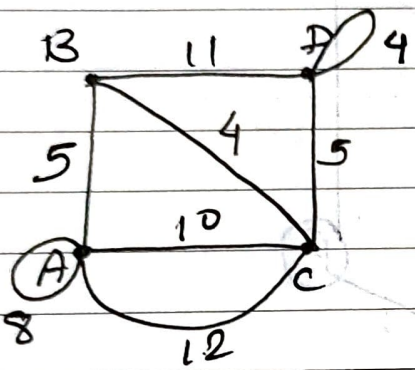
	0	1	2	3	4	5	6
0	0	28	∞	∞	∞	10	∞
1	28	0	16	∞	∞	∞	14
2	∞	16	0	12	∞	∞	∞
3	∞	∞	12	0	22	∞	18
4	∞	∞	∞	22	0	25	24
5	10	∞	∞	∞	25	0	∞
6	0	14	∞	18	24	∞	0

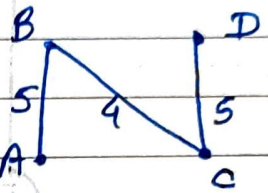


Q.

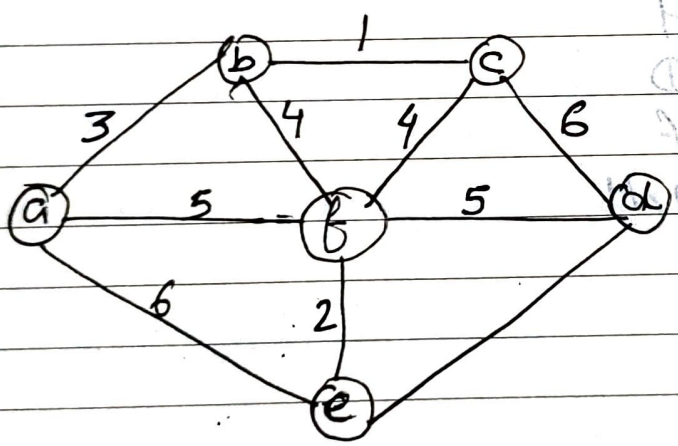


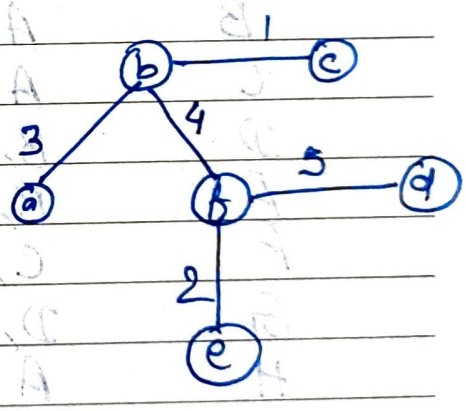
Q.





Q..





Q. Same as Krushkal's 3rd Eg.