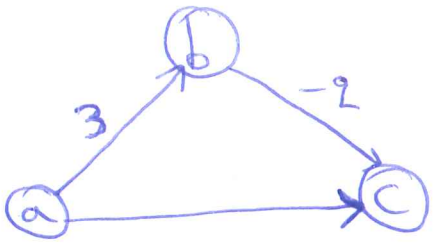
9.7) a) Let the graph be:



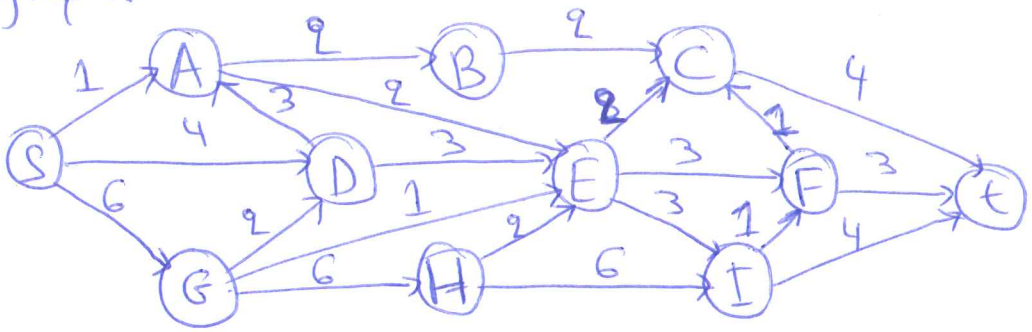
In this example, the Dijkstra algorithm gives a path from a to c with the cost 2 but the path from a to c via b is 1.

Thus, c is visited immediately after a in the beginning. So, Dijkstra will erringly find the shortest distance from a to c as 2 instead of 1. Thus, the actual shortest path from a to c is a->b->c which has the distance 1.

9.11) Network flow problems are defined as the problems where the flow across a network is balanced. It consists of a directed graph where each edge contains a capacity which represents the maximum flow that can pass through that node.

In network flow problems, there are two types of vertices, one is source s and the other is sink t.

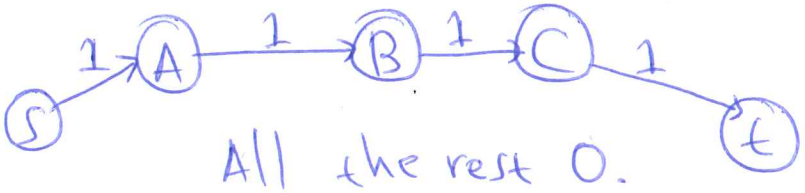
The maximum flow problem is defined as the problem to find out the maximum flow passing through the graph.



Gr

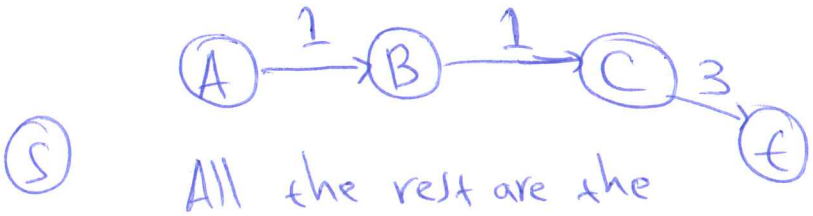
The maximum flow coming out of the s is 11. So, the maximum flow that can come to t is 11. We create two graphs: one is the residual graph Gr that tells for any edge, how much flow can be added. The other is the flow graph Gf that tells the amount of flow attained at any time. The initial flow has all the edges with zero weight.

There are many paths from s to t. Select any path such as s, A, B, C, t. Send one unit of flow through every edge on this path.



All the rest 0.

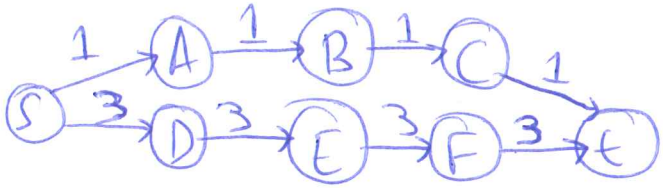
Gf



All the rest are the same as before *

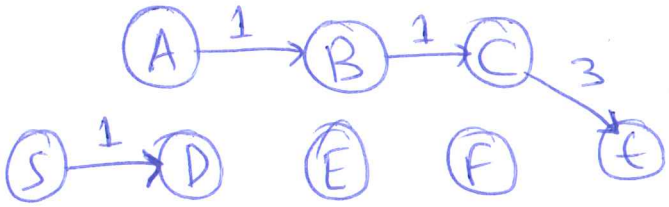
Gr

9.11 Continuation > Next, select the path S, D, E, F, t. Send 3 unit of flow through every edge on this path.



G_f

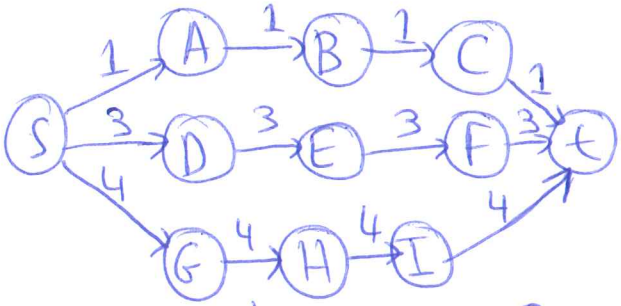
All the rest 0.



G_r

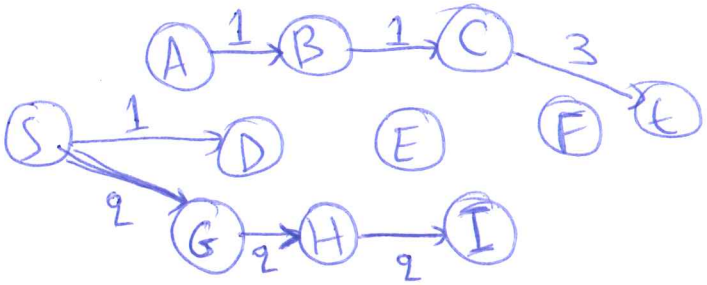
All the rest are the same as before *

Next, select the path S, G, H, I, t. Send 4 unit of flow through every edge on this path.



G_f

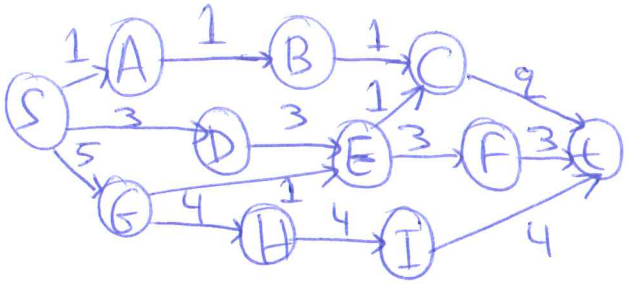
All the rest 0.



G_r

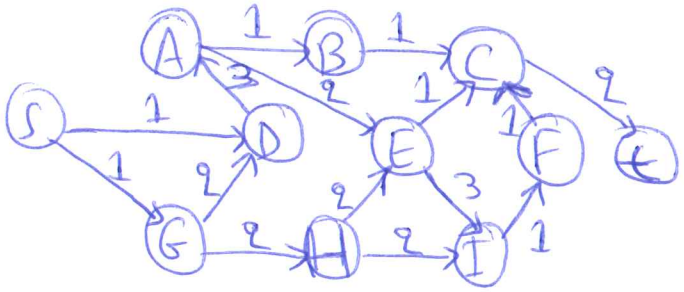
All the rest are the same as before *

Next, select the path s, G, E, C, t. Send 1 unit of flow through every edge on this path.



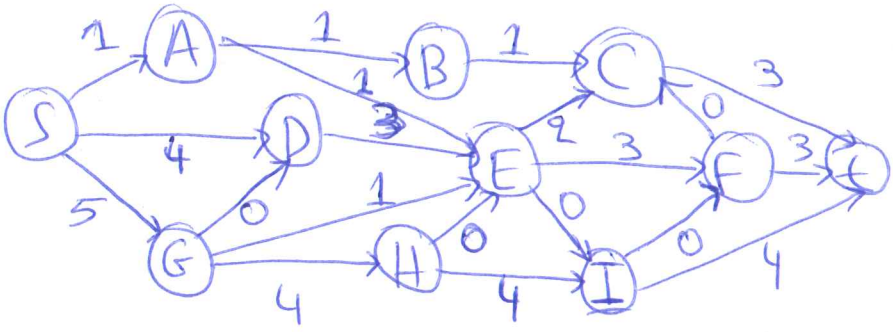
G_f

All the rest 0.

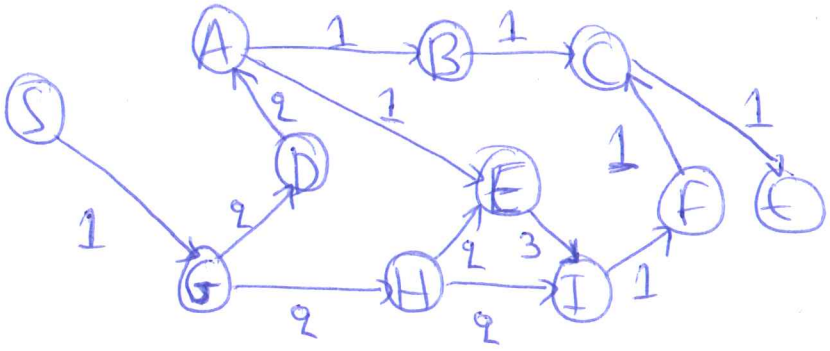


G_r

Next, select the path s, D, A, E, C, t. Send 1 unit of flow through every edge on this path.

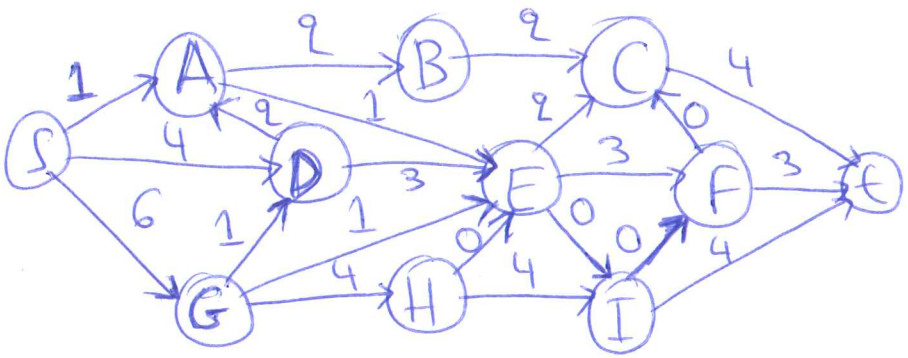


G_f

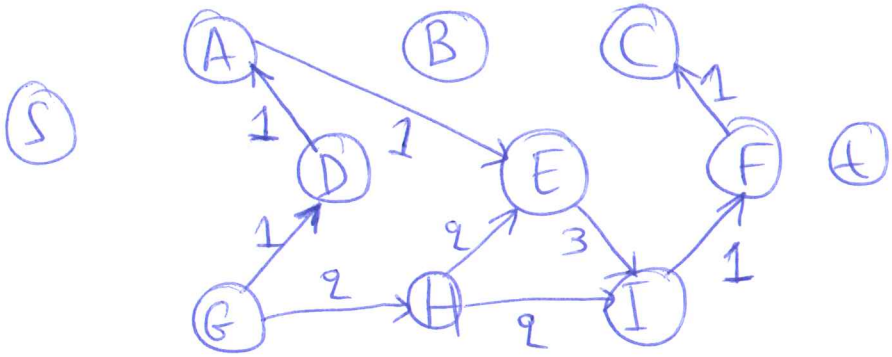


G_r

9.11) Continuation Next, select the path s, G, D, A, B, C, t. Send 1 unit of flow through every edge on this path.



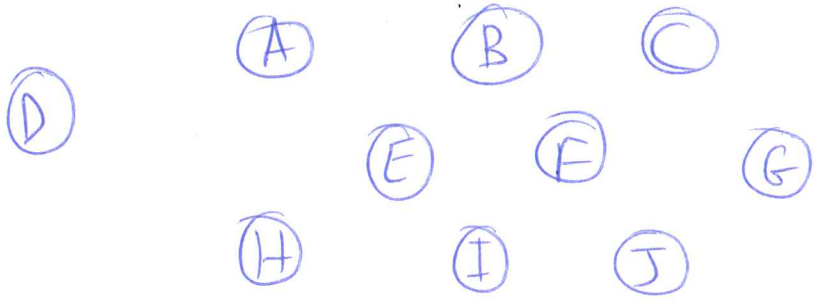
G_f



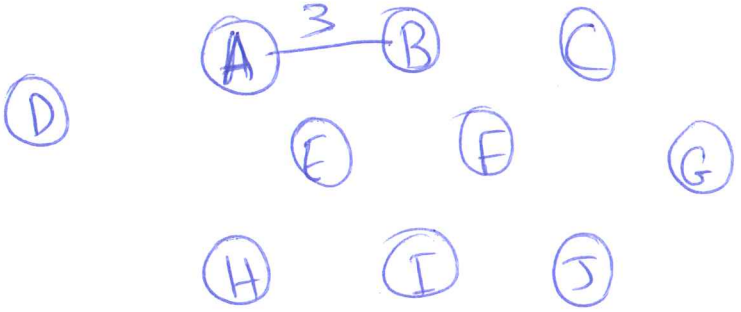
G_r

Maximum flow at t is shown in the flow graph as 11.

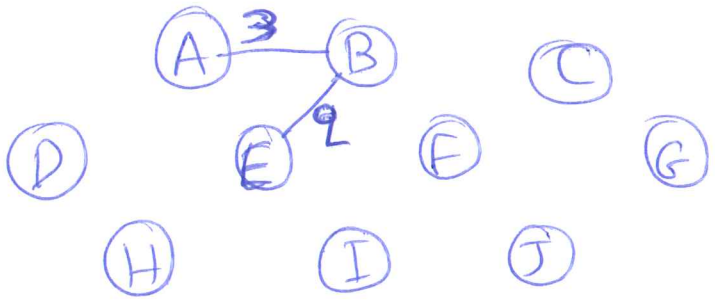
9.15) a) Prim's Algorithm :



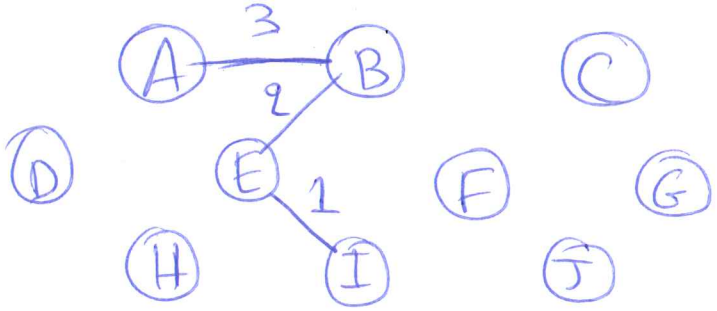
Start traversing from the vertex A which is considered as the source vertex. Find the edge from A with minimum weight. The smallest edge is from A to B with minimum weight cost. Select the vertex B and join the edge A to B.



Select the vertex pair B to E with weight 2. Because we consider vertex A and B, and find the minimum weight.

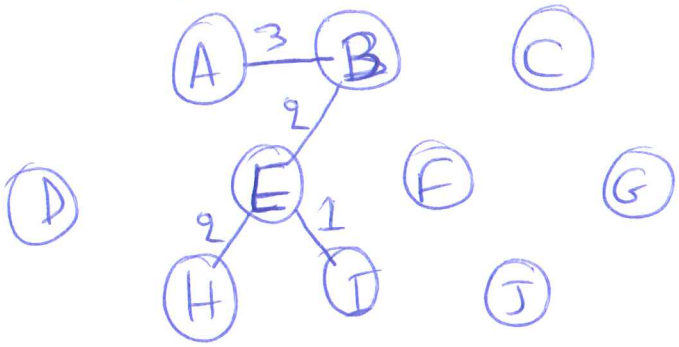


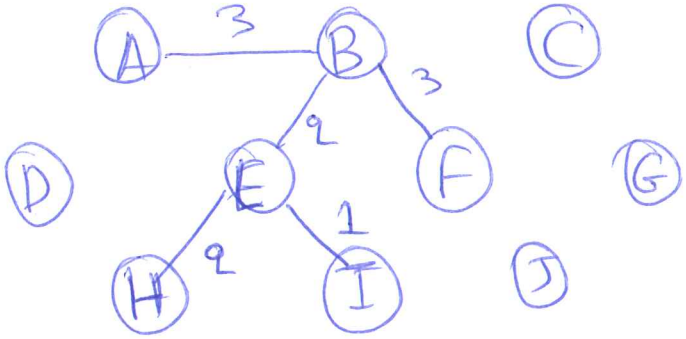
From A, B, and E, we select E and I with weight 1.

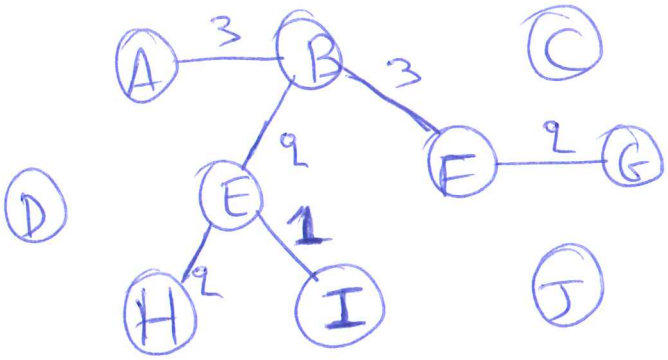


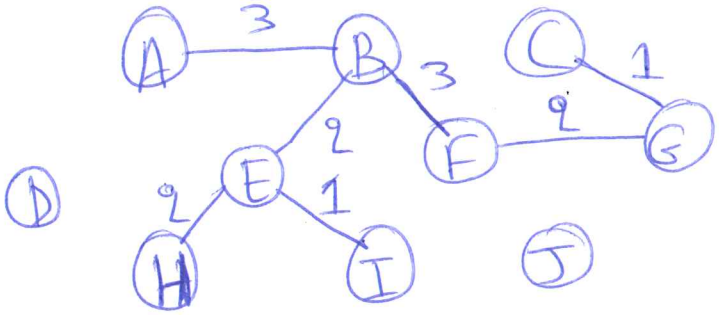
9.15) a) Continuation →

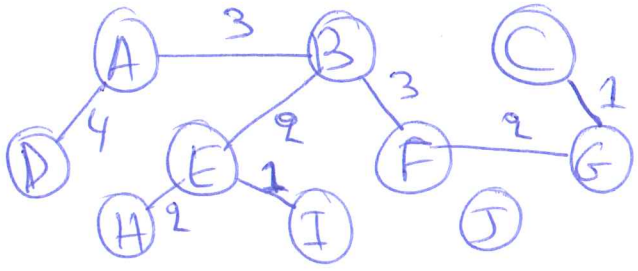
From A, B, E, and I, we select H and connect the edge E to I with weight 2.

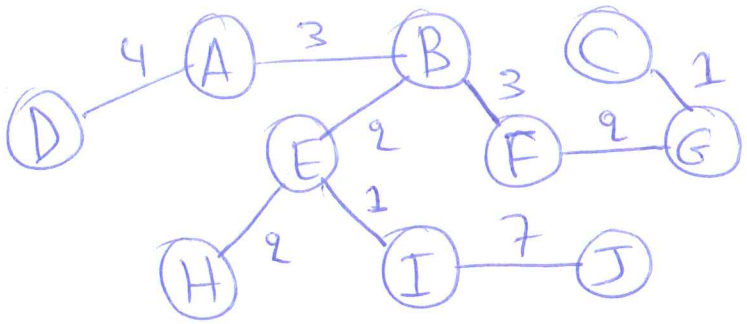






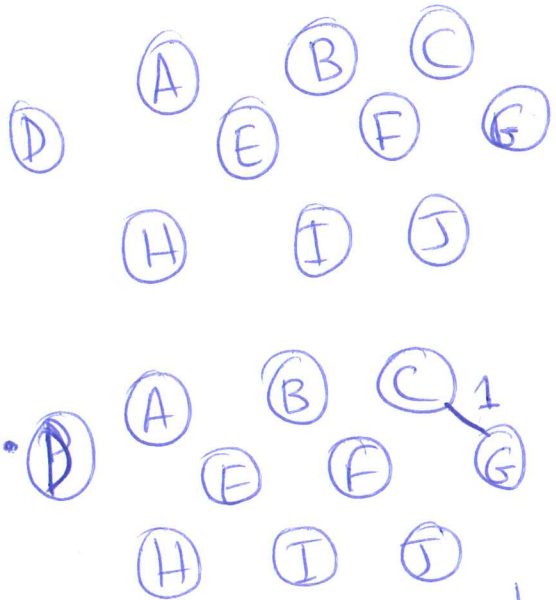




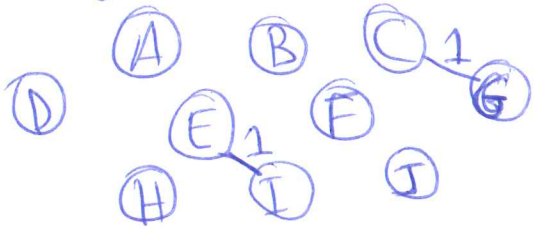


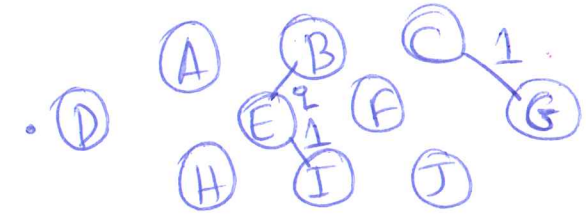
Kruskal's Algorithm:

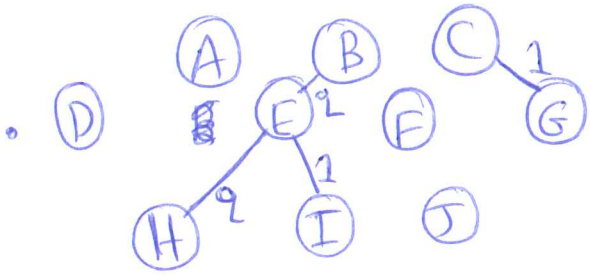
Edge	Weight	Action
(C, G)	1	Accepted (A)
(E, I)	1	A
(E, B)	2	A
(E, H)	2	A
(F, G)	2	A
(F, I)	3	Rejected (R)
(A, B)	3	A
(A, E)	4	R
(H, I)	4	R
(D, E)	5	R
(D, H)	6	R
(F, C)	6	R
(B, F)	3	A
(A, D)	4	A
(I, J)	7	A
(J, G)	8	R
(B, C)	10	R
(F, J)	11	R
(E, F)	11	R



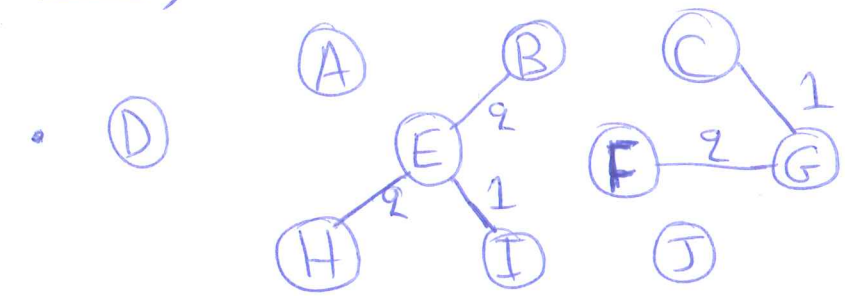
Select the edge with minimum weight.
C and G is chosen with weight 1.

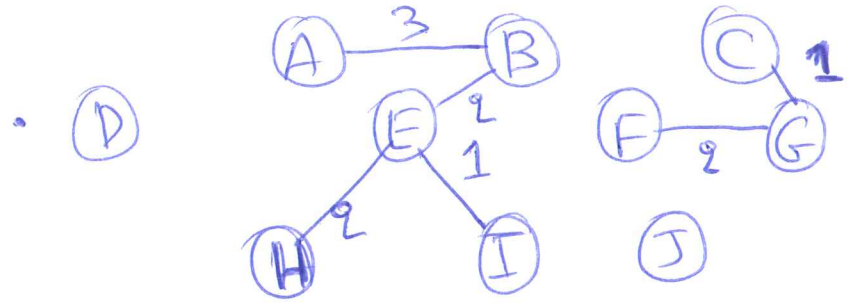


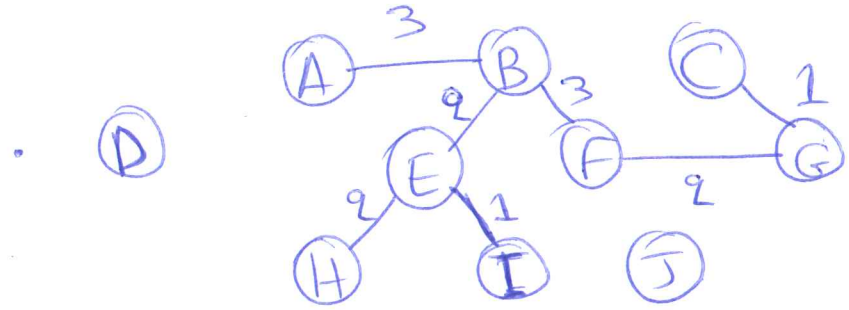


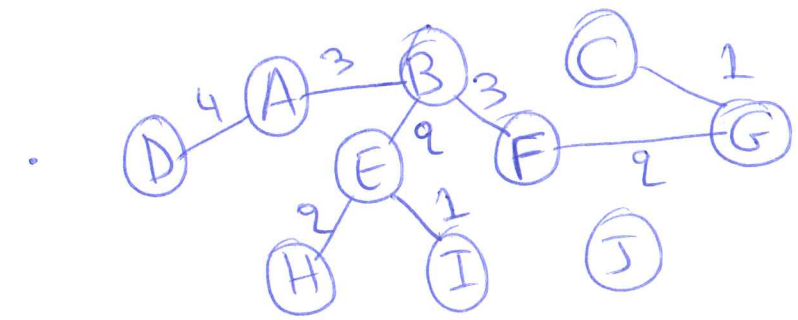


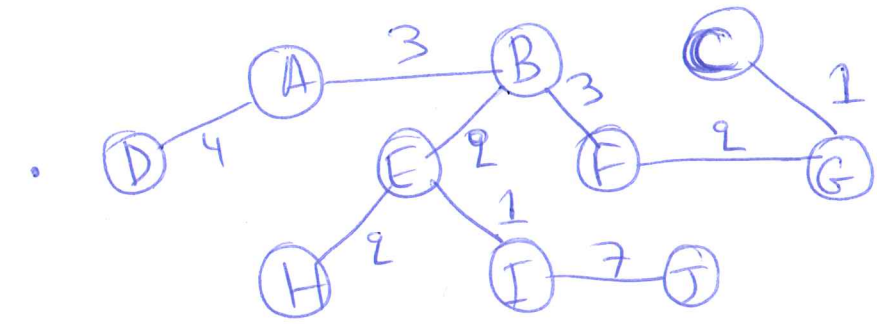
9.15) a) Continuation



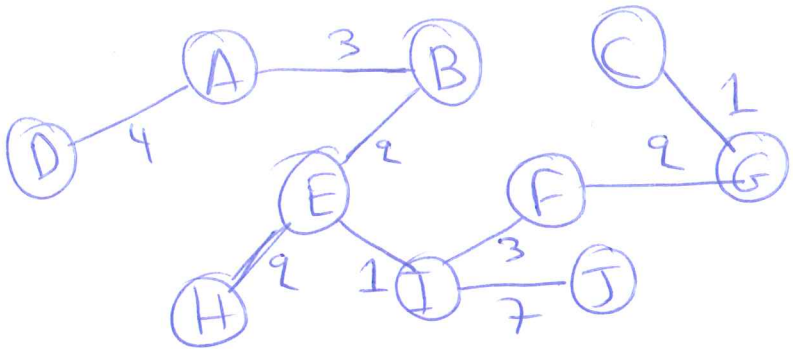








This is the minimum spanning tree.



Another variant of the minimum spanning tree.

9.20) Check Moodle.

9.23) a) Check Moodle.