Q1)

A	B	C	D	E	F	G	H
INF	INF	INF	INF	INF	INF	0	INF

The set *sptSet* is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with minimum distance value. The vertex G is picked, include it in *sptSet*. So *knownSet* becomes {G}. After including 0 to *sptSet*, update distance values of its adjacent vertices. Adjacent vertices of G are F, H and C. The distance values of F, C and H are updated as 8, 2 and 2. Following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown.

A	B	C	D	E	F	G	H
INF	INF	2	INF	INF	8	0	2

Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex C and H has the minimum value, so pick among one of them. The vertex H is picked and added to sptSet. So knownSet now becomes {G, H}. Update the distance values of adjacent vertices of H. The distance value of vertex 2 becomes 12.

A	B	C	D	E	F	G	H
6	INF	2	INF	INF	8	0	2

Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex C has the minimum value. The vertex C is picked and added to sptSet. So knownSet now becomes {G, H, C}. Update the distance values of adjacent vertices of C. The distance value of vertex B becomes 6.

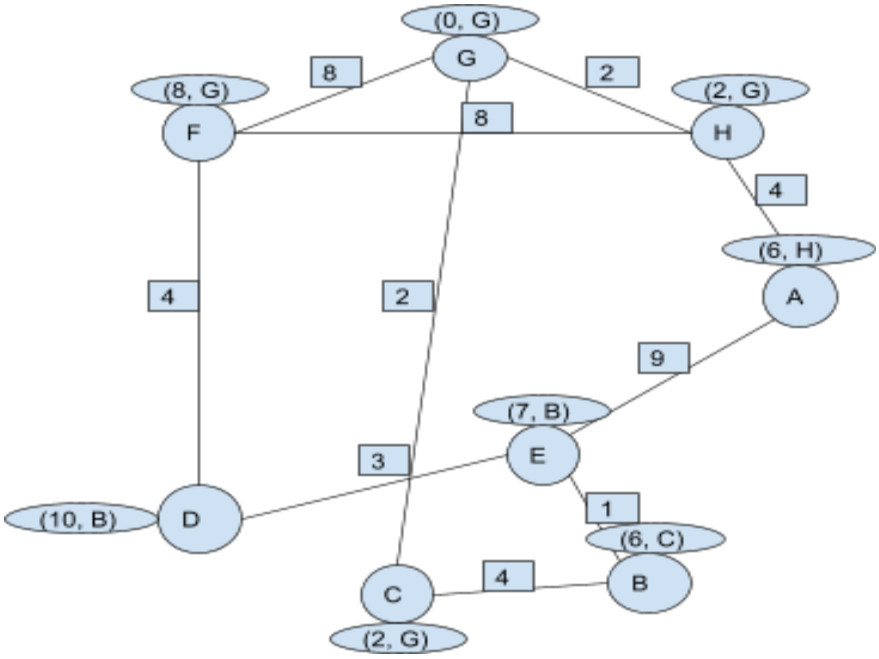
A	B	C	D	E	F	G	H
6	6	2	INF	INF	8	0	2

Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex A and B has the minimum value, so pick among one of them. The vertex B is picked and added to sptSet. So knownSet now becomes {G, H, C, B}. Update the distance values of adjacent vertices of B. The distance value of vertex E becomes 7.

A	B	C	D	E	F	G	H
6	6	2	INF	7	8	0	2

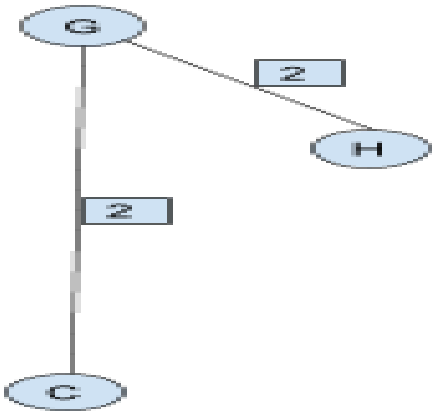
Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex E has the minimum value. The vertex E is picked and added to sptSet. So knownSet now becomes {G, H, C, B, E}. Update the distance values of adjacent vertices of E. The distance value of vertex D becomes 10.

A	B	C	D	E	F	G	H
6	6	2	10	7	8	0	2

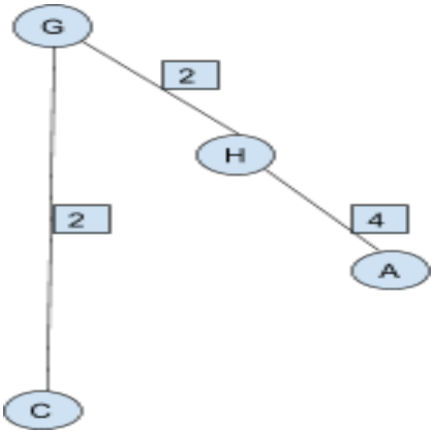


Q2)

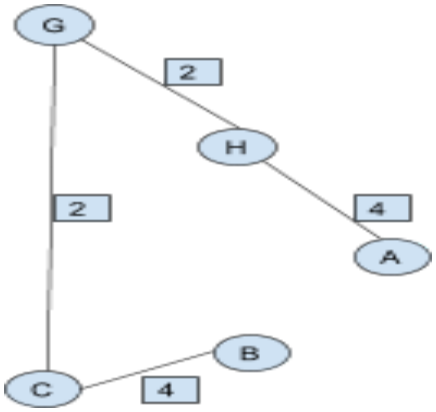
1)



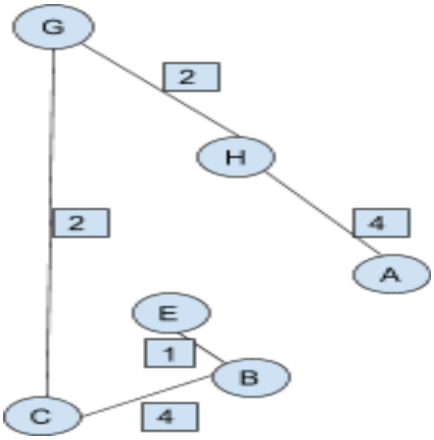
2)



3)

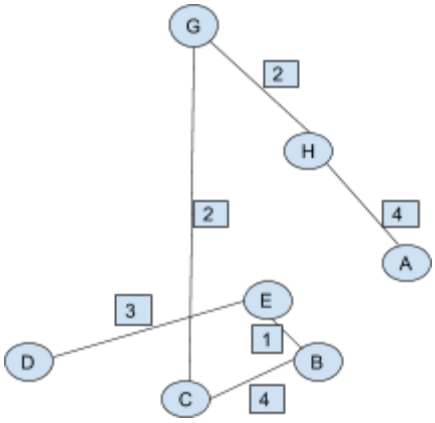


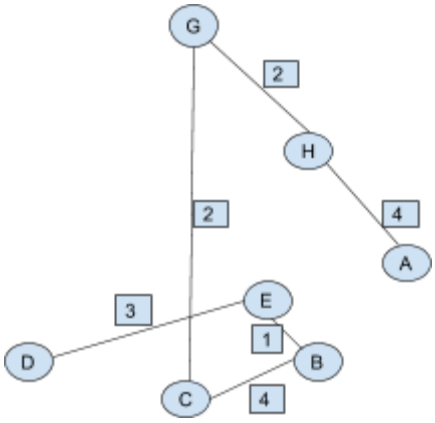
4)



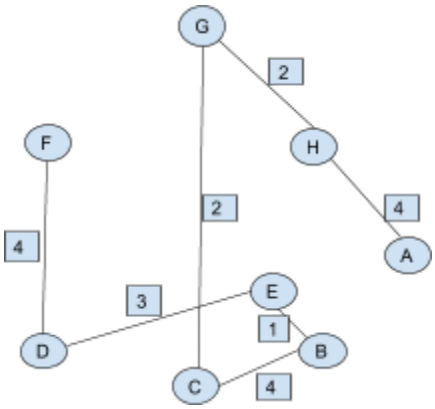
5)

6)





7)



Cost = Eg, h + Eh, a + Ee, b + Ee, d + Eb, c + Ec, g + Ed, f

= 1 + 3 + 4 + 4 + 2 + 4 + 2 = 20

Q3)

Below are the steps for finding MST using Kruskal's algorithm:

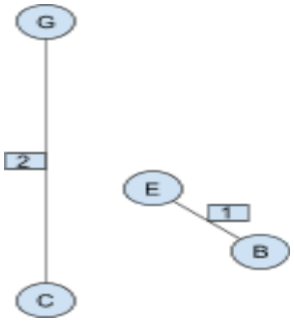
1. *Sort all the edges in non-decreasing order of their weight.*

2. *Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.*

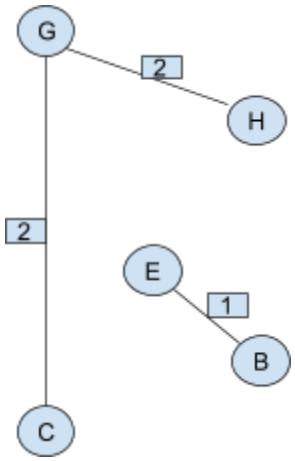
3. *Repeat step#2 until there are (V-1) edges in the spanning tree.*

1)

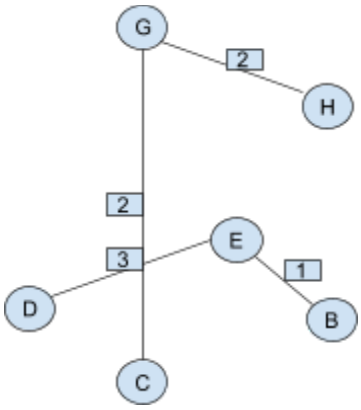
2)



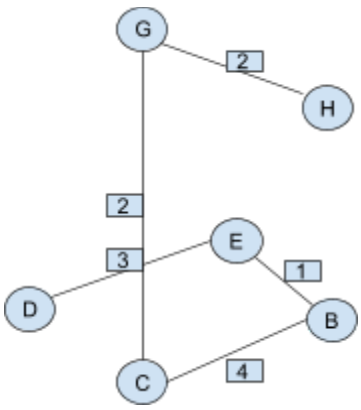
3)



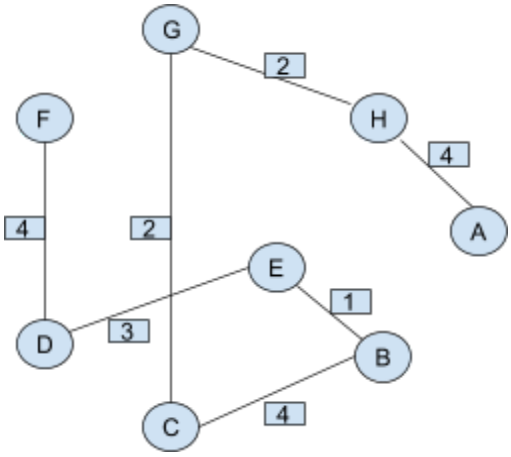
4)



5)



6)



Thrown edges due to cycle = Ef,g , Ef,h , Ea, e

Trace and cost

1. Ve
2. Ee, b(1)
3. Ec, g(2)
4. Eg, h(2)
5. Ee, d(3)
6. Ec, b(4)
7. Ed, f(4)
8. Eh, a(4)

Q4)

Trace steps:

q = queued set
S = visited vertices set

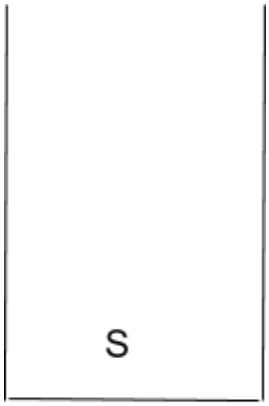
1. Enqueue S. q = {S}. S = {}
2. Enqueue A, D and B. Dequeue S. q = {A, D, B}. S = {S}
3. Enqueue C. Dequeue A. q = {D, B, C}. S = {S, A}
4. Enqueue E, T and F. Dequeue D. q = {B, C, E, T, F}. S = {S, A, D}
5. Dequeue B. q = {C, E, T, F}. S = {S, A, D, B}
6. Dequeue C. q = {E, T, F}. S ={S, A, D, B, C}
7. Enqueue G. Dequeue E. q = {T, F, G}. S = {S, A, D, B, C, E}
8. Dequeue T. q = {F, G}. S = {S, A, D, B, C, E, T]
9. Dequeue F. q = {G}. S = {S, A, D, B, C, E, T, F}
10. Dequeue G. q = {}. S = {S, A, D, B, C, E, T, F, G}

Output: S, A, D, B, C, E, T, F, G

Q5)

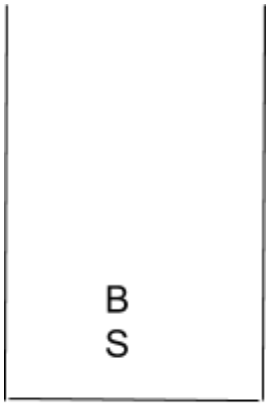
a)

1)



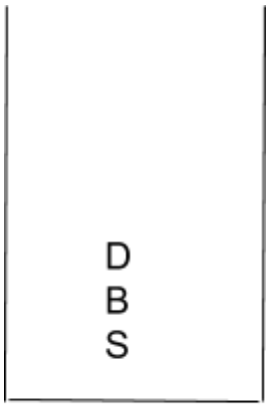
Stack

2)



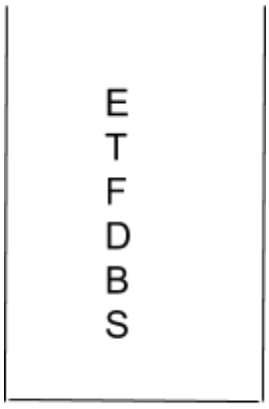
Stack

3)



Stack

4)

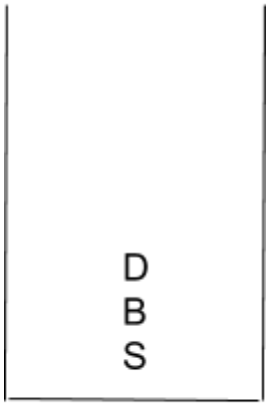


Stack

5)

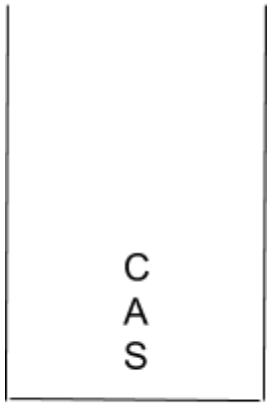
Stack

6)



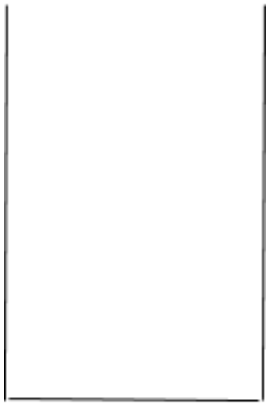
Stack

7)



Stack

8)



Stack

b) Post order: S(9), A(8), C(7), B(6), D(5), E(4), G(3), T(2), F(1)
c) Pre order: S(1), B(2), D(3), F(4), T(5), E(6), G(7), A(8), C(9)
d)

Tree arcs: S->B, B->D, D->F, D->T, D->E, E->G, S->A, A->C
Cross arcs: S->D
Backward arcs: B->S, G->E
Forward arcs: C->D, E->T, G->T

Q6)

Topological sorting: take in degree 0 vertices.

S = In degree 0-vertices set

Step 1: Take A and G. S = {A, G}. Select one of them in S, take A. Print A.
Step 2: Take B. S = {B, G}. Select one of them in S, take B. Print B.
Step 3: Take C. S = {G, C}. Select G. Print G.
Step 4: Take D. S = {C, D}. Select D. Print D.
Step 5: Select C. Print C.
Step 6: Take E. S = {E}. Print E.
Step 7: Take F. S = {F}. Print F.

Output: A B G D C E F