Homework – 4

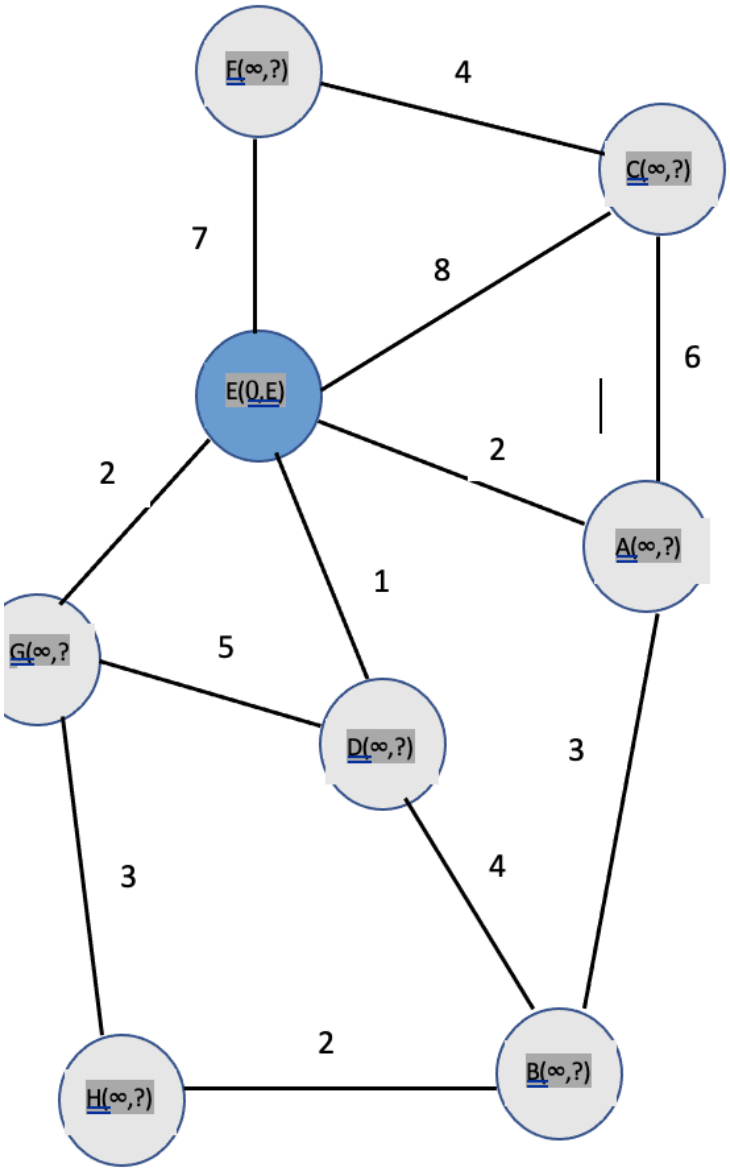
(Graph Problems)

Ayça Elif Aktaş

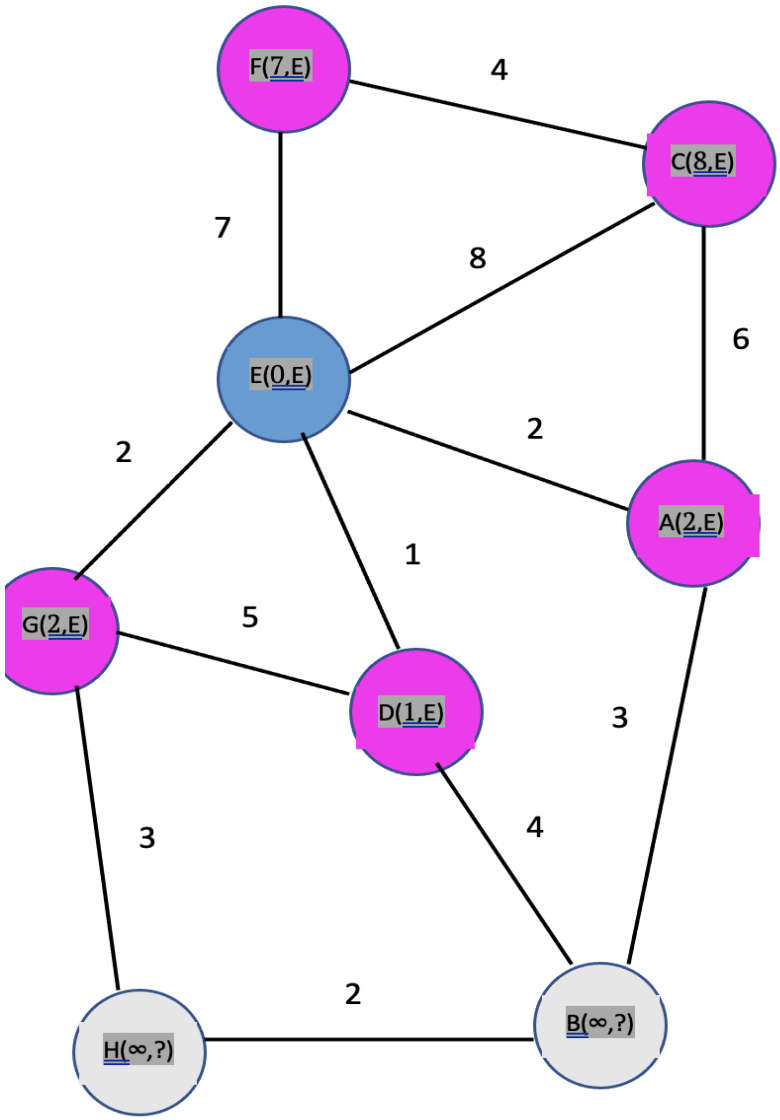
27802

Question 1:

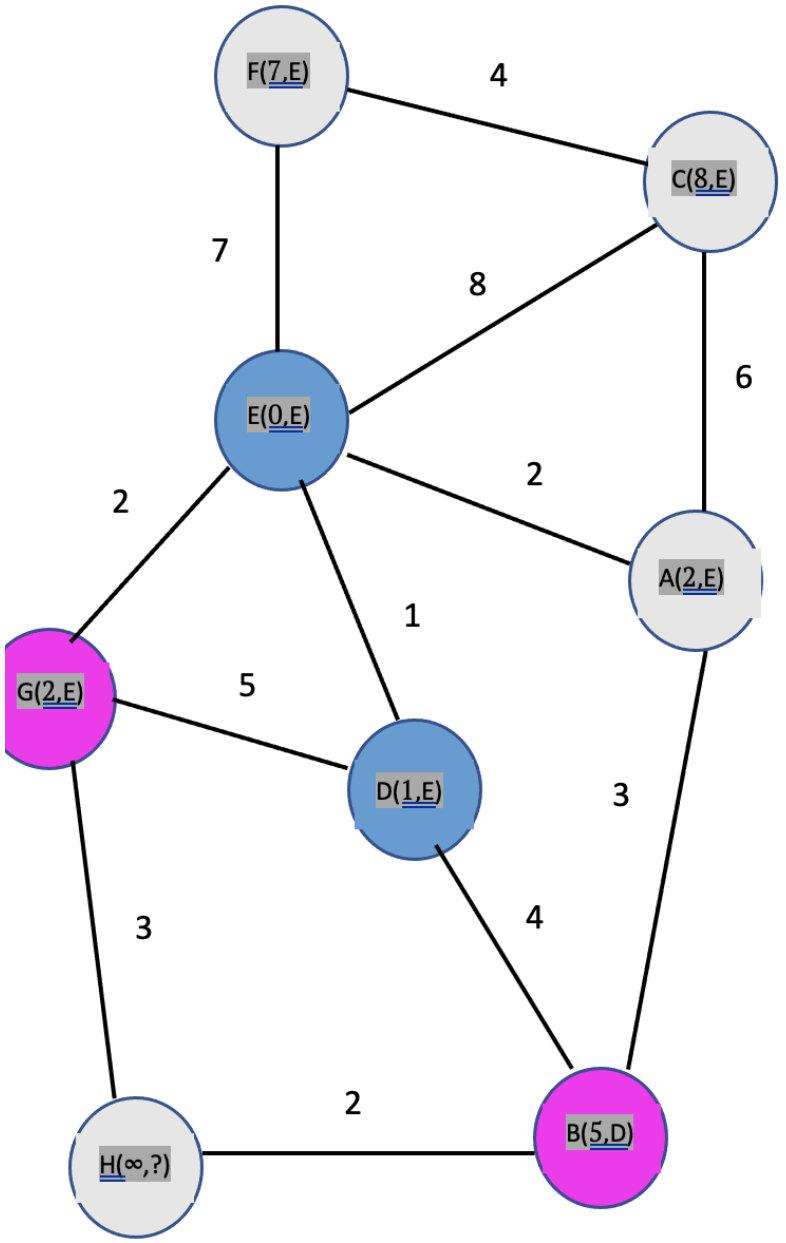
1. First, the starting vertex **(*E*)** is selected and painted blue.



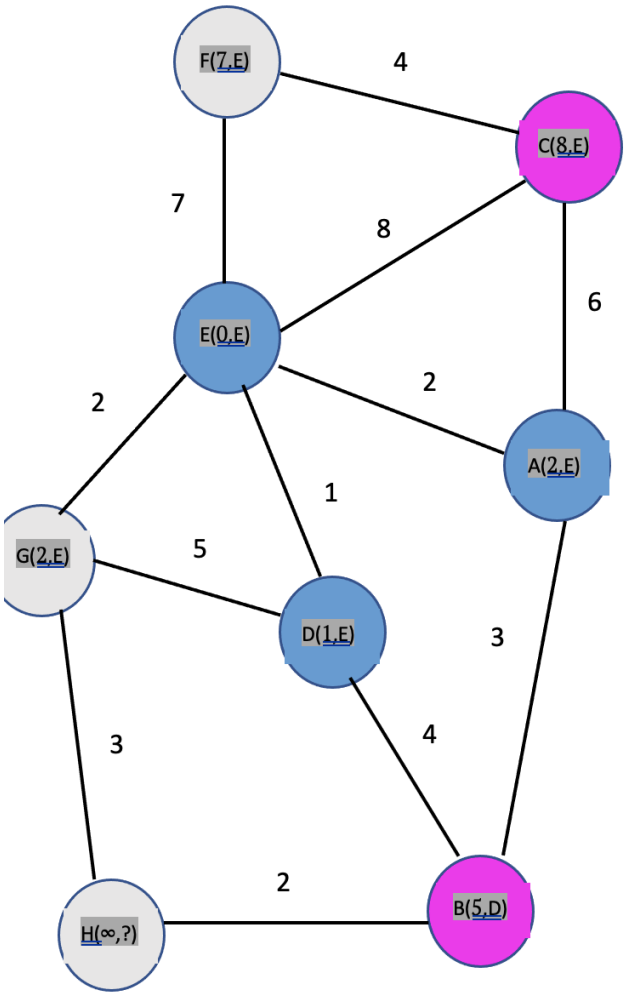
2. Then vertices adjacent to ***E*** are colored pink and their distance to *E* is calculated.



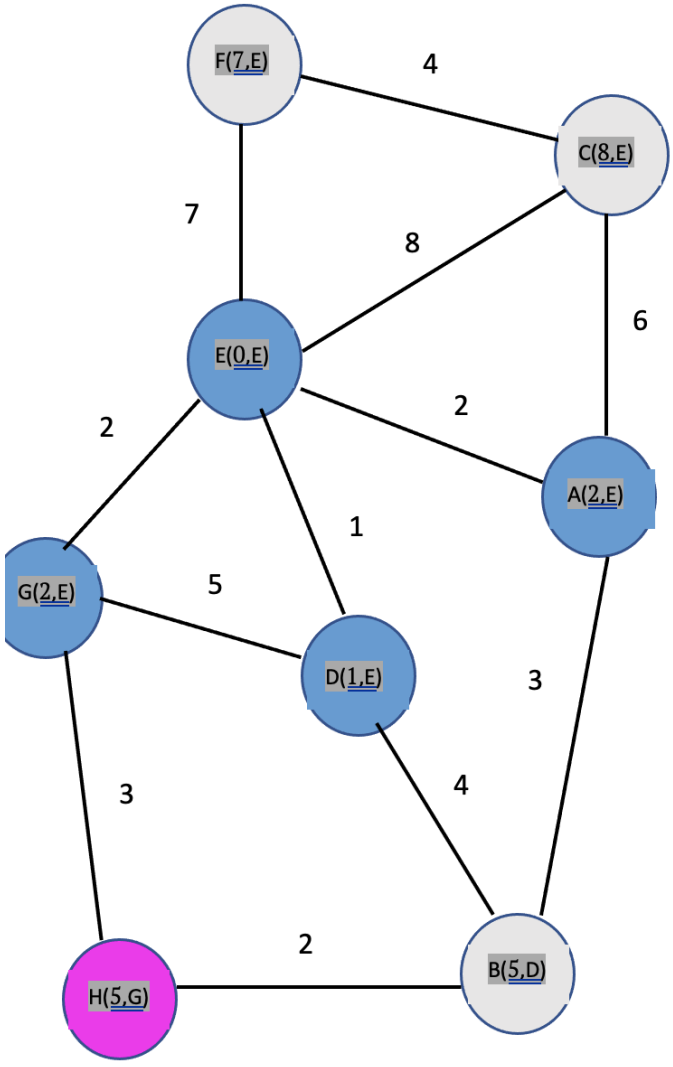
3. After the vertices adjacent to *E* are calculated, all the vertices are painted back to gray, the vertex with the smallest value among the **unknown vertices** whose length is calculated **(*D*)** is painted **blue(known)**, and the vertices adjacent to this vertex **(*D*)** are painted pink and their lengths are calculated**(G,B).** At this stage there has been no update to the length value of vertex G because it's distance to Vertex E would be larger if path goes over from Vertex D. And the length value of vertex B is calculated.



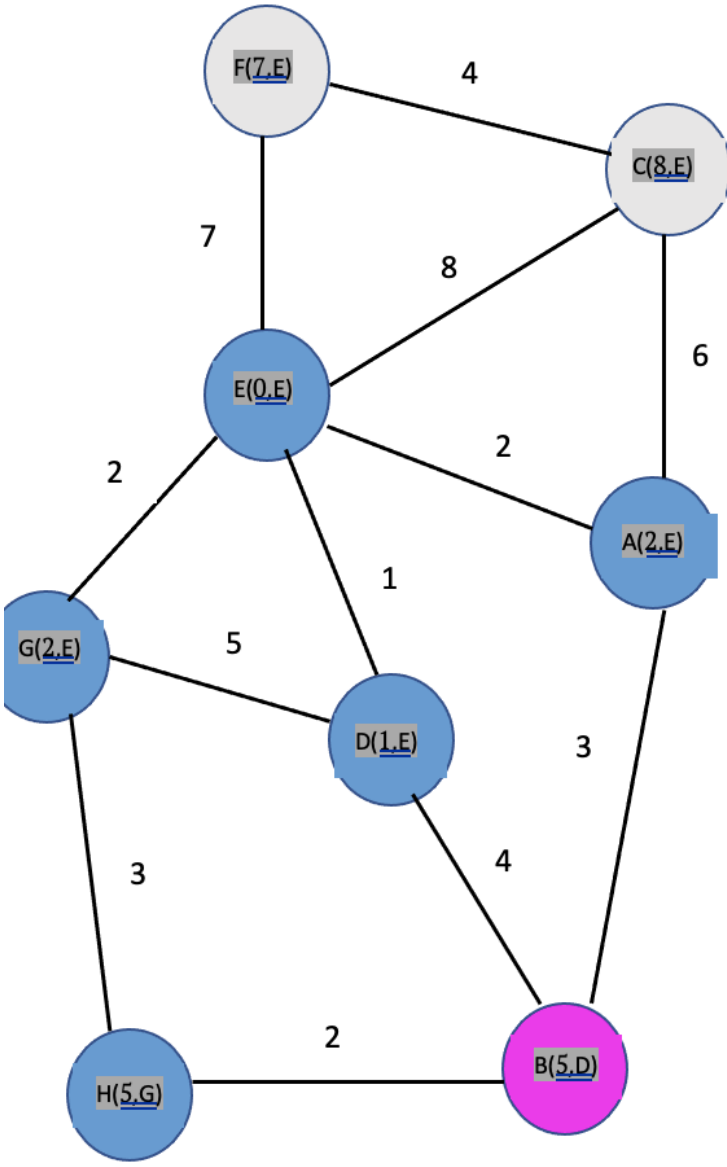
4. After the operations related to vertex *D* are finished, all remaining neighboring vertices are painted back to gray, the vertex **(*A*)** with the smallest value among the **unknown vertices** whose length value is calculated is painted **blue(known),** and the vertices adjacent to this vertex (*A*) are painted pink and the lengths for these neighboring vertices are calculated**(B,C).** At this stage the length value of vertex *B and C* is calculated, and no update is made because their length value would be same when going over vertex *A* to final destination Vertex E.



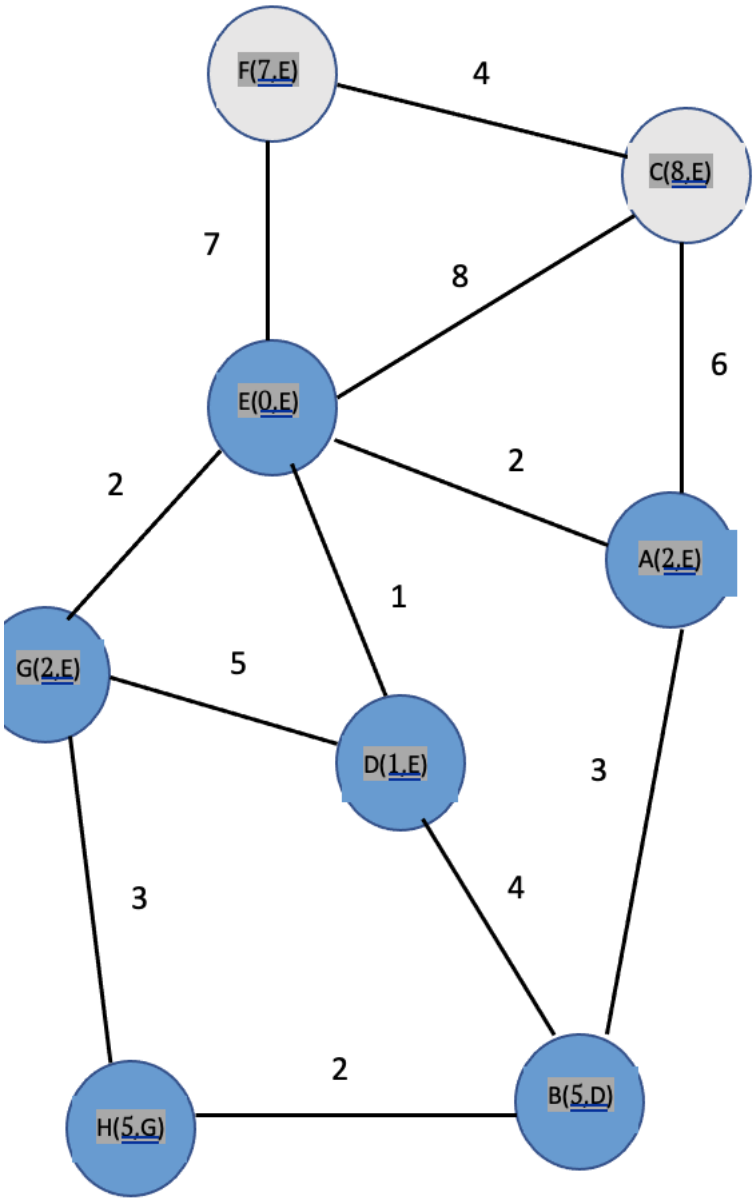
5. After the operations related to vertex *A* are finished, all remaining neighboring vertices are painted back to gray, the vertex **(*G*)** with the smallest value among the **unknown vertices** whose length value is calculated is painted blue, and the vertices adjacent to this vertex **(*G*)** are painted pink and the lengths for these neighboring vertices are calculated**(H).**No update is being made other than calculating distance for Vertex H, because with G becoming a known vertices doesn't change any other vertices smallest path to Vertex E.



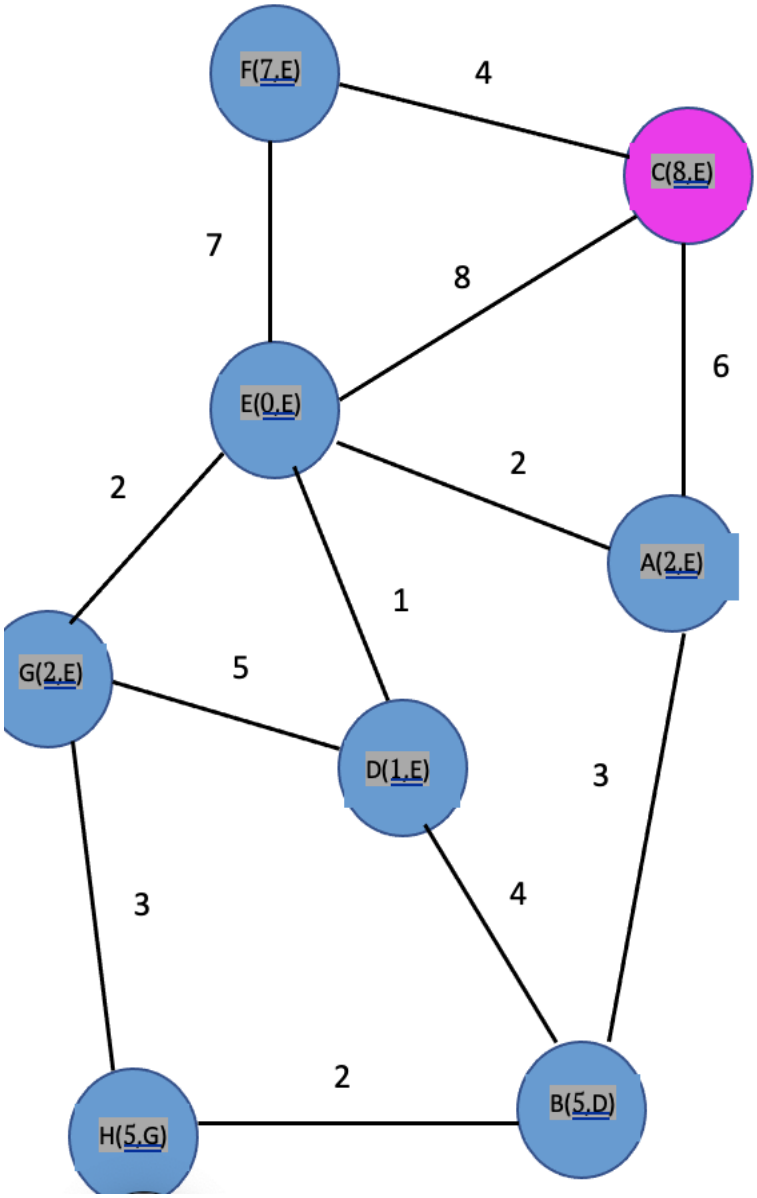
6. After the operations related to vertex *G* are finished, all remaining neighboring vertices are painted back to gray, the vertex **(*H*)** with the smallest value among the **unknown vertices** whose length value is calculated is painted blue, and the vertices adjacent to this vertex **(*H*)** are painted pink and the lengths for these neighboring vertices are calculated**(B).**The length of shortest path to Vertex B is not updated becuse its shortest path to Vertex E would be longer going over Vertex H.



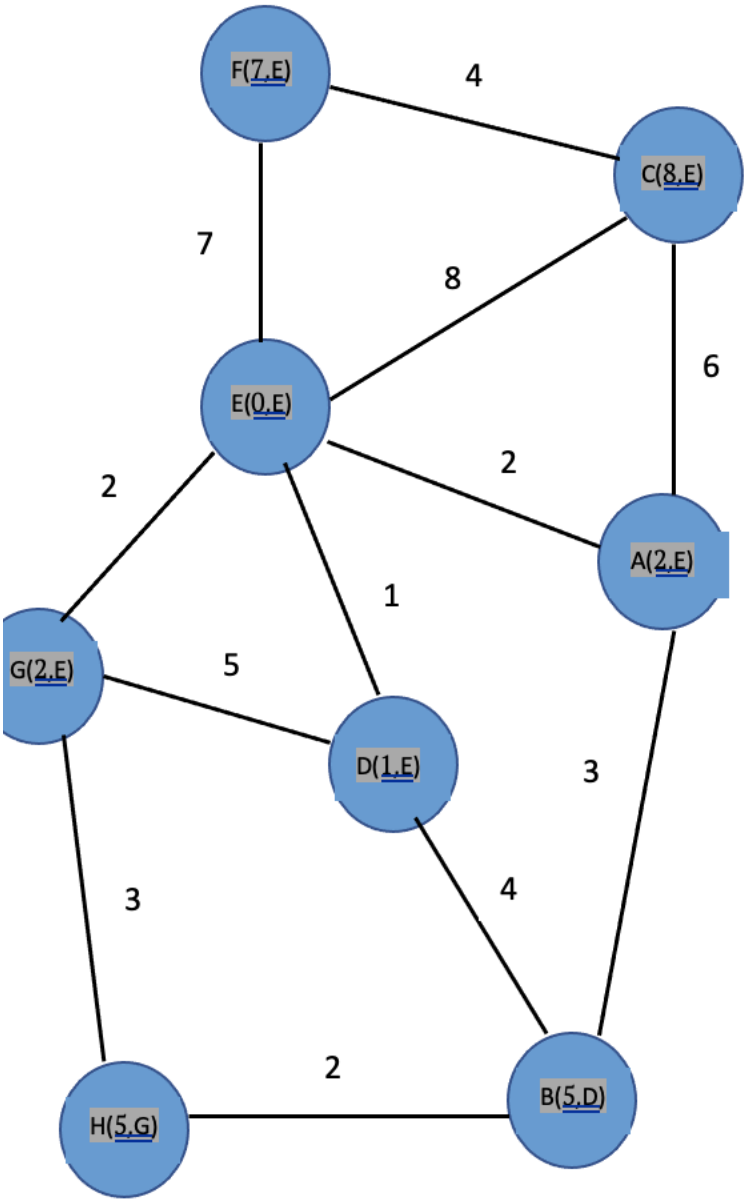
7. After the operations related to vertex *H* are finished, all remaining neighboring vertices are painted back to gray, the vertex **(*B*)** with the smallest value among the u**nknown vertices** whose length value is calculated is painted blue, and this vertex **(*B*)** does not have any **unknown adjacent vertices** son no action can be taken.



8. After the operations related to vertex *B* are finished, all remaining neighboring vertices are painted back to gray, the vertex (*F*) with the smallest value among the **unknown vertices** whose length value is calculated is painted blue, and the vertices adjacent to this vertex (*F*) are painted pink and the lengths for these neighboring vertices are calculated(C). The length of shortest path to Vertex C is not updated becuse it would be longer going over Vertex F to final destination Vertex E.

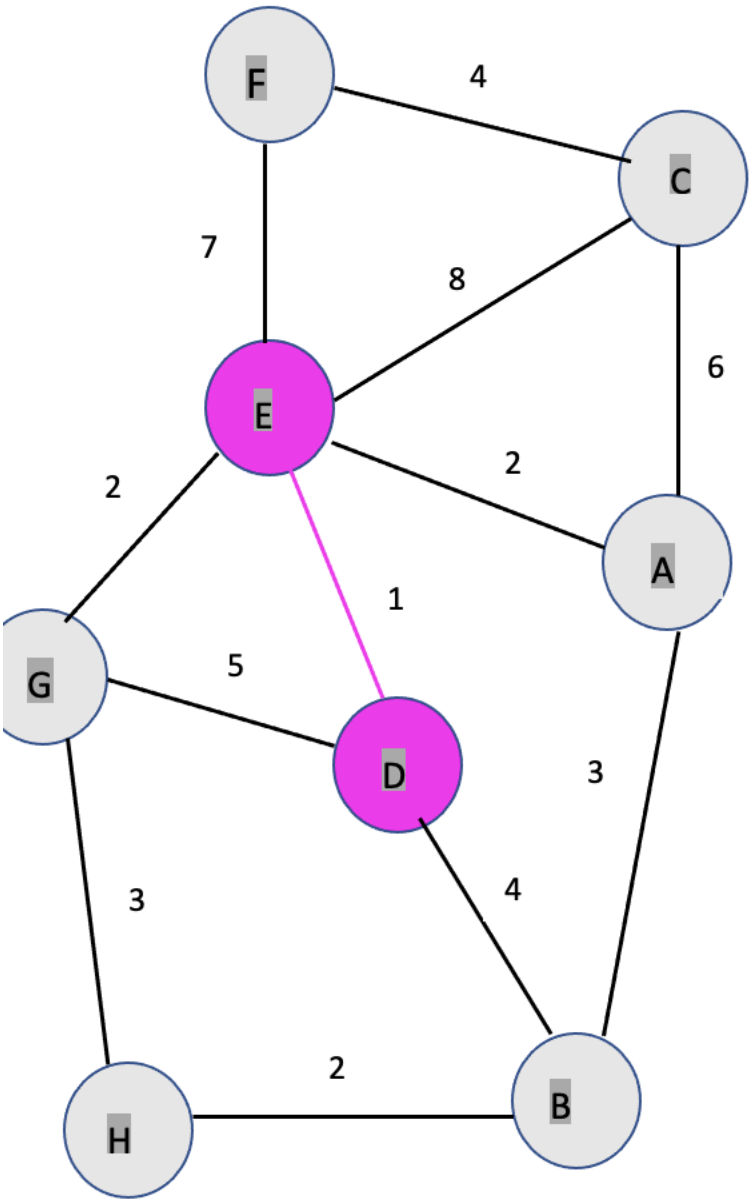


9. The only remaining vertex (*C*) is painted blue, and the process is completed. All verticies are now labeled known.

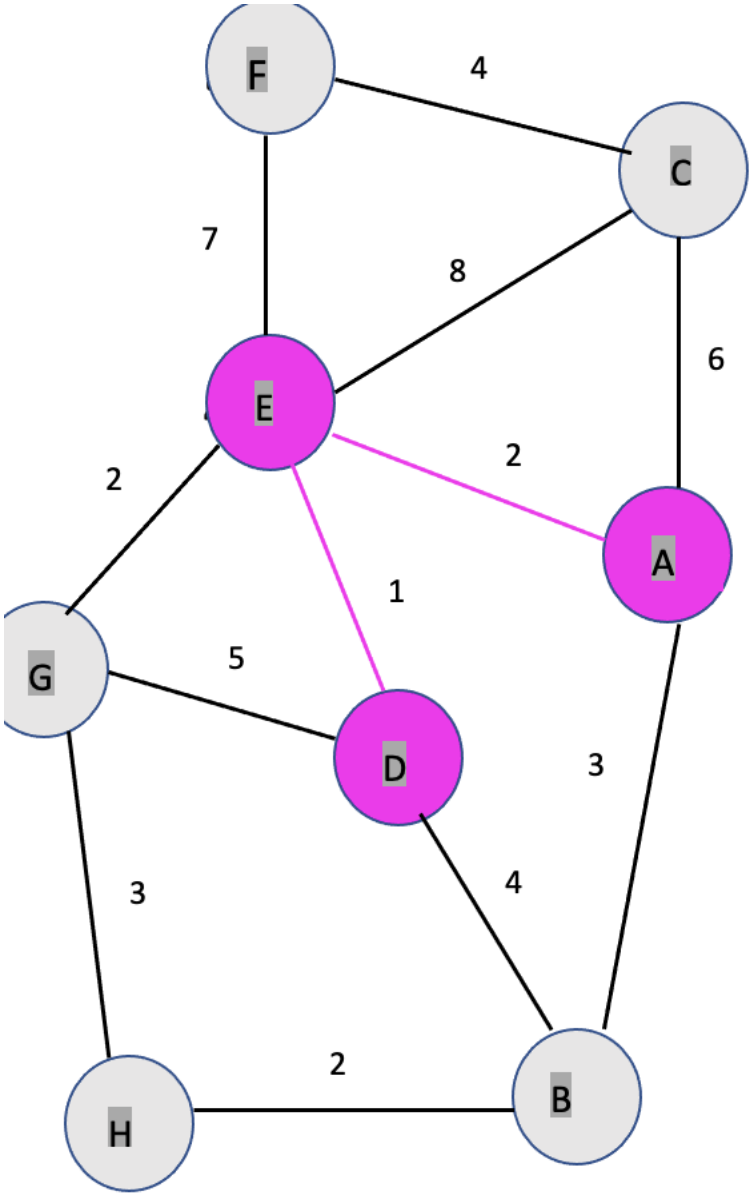


Question 2:

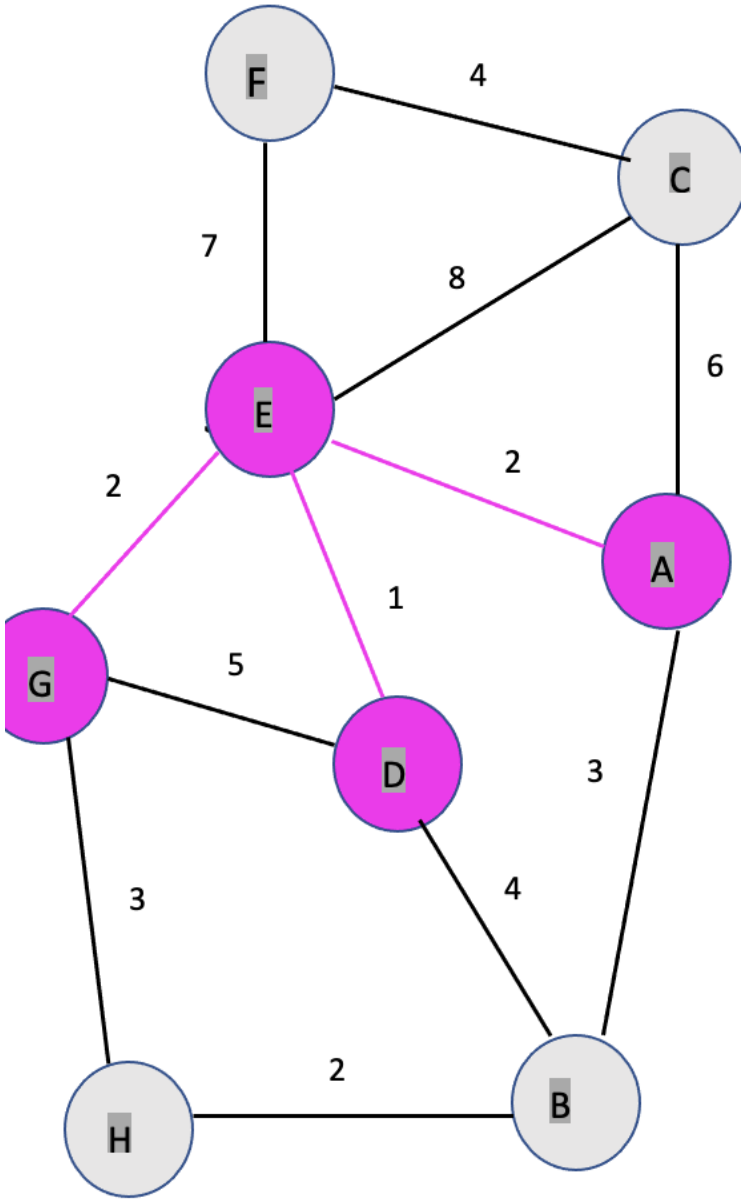
1. **Prim's Algorithm** is started to be applied from vertex *E*. Vertex ***E*** is painted in pink and the edge that has the smallest cost from the adjacent vertices**(E-D)** is painted in pink and painted together with the vertex **(D)** it connects to.



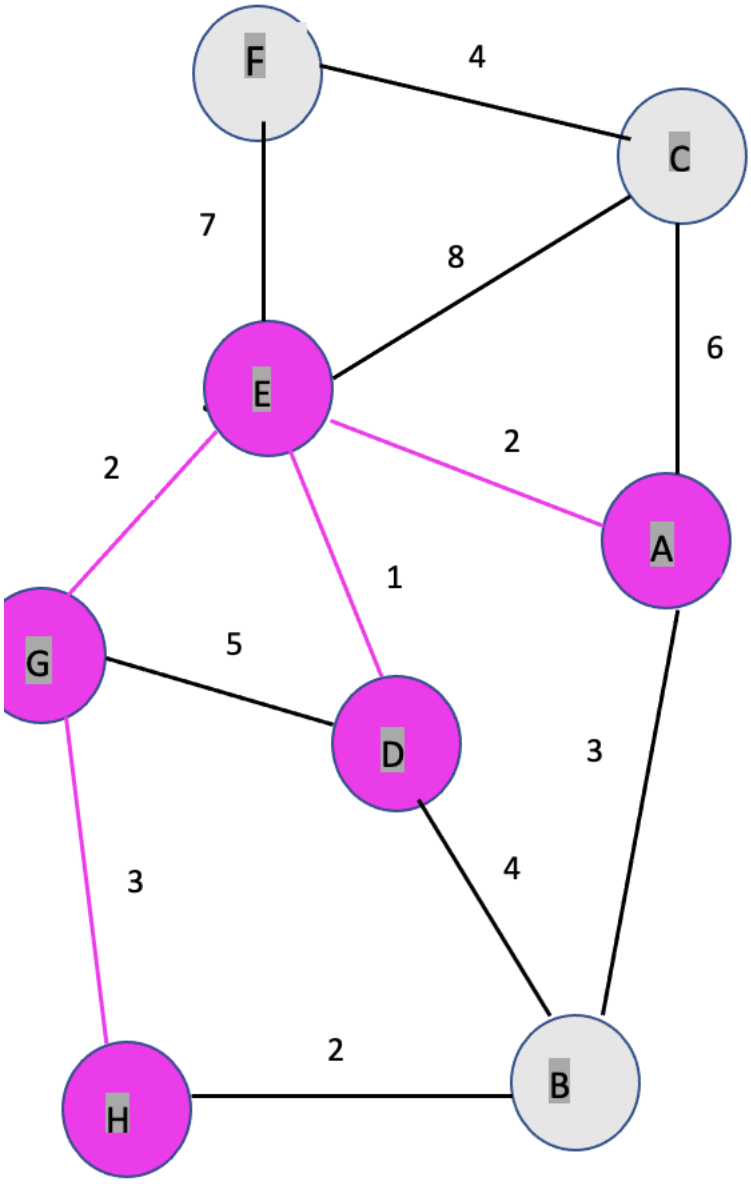
2. After the *D* vertex is painted pink, the edge that has one of the smallest edge cost and has the other end connected to an unpainted vertex is selected**(E-A)**and painted together with the vertex **(*A*)** to which it is connected.



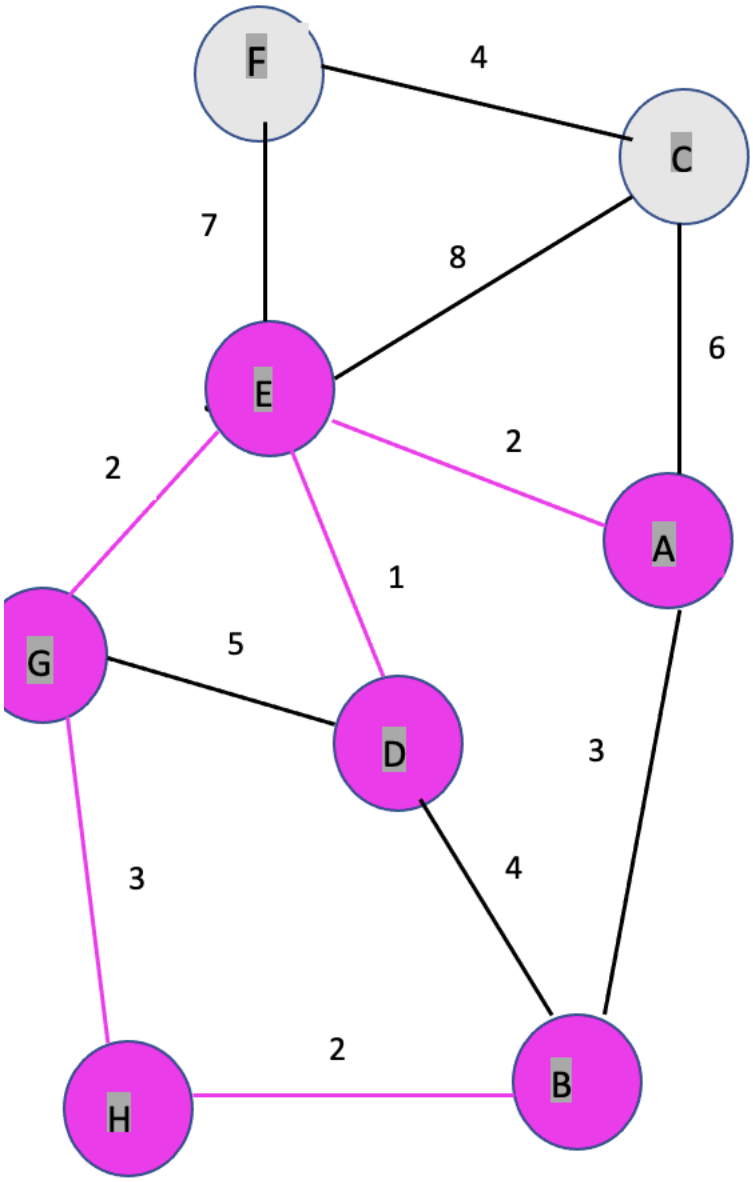
3. After the *A* vertex is painted pink, the edge that has one of the smallest edge cost and has the other end connected to an unpainted vertex is selected **(*E*-*G*)** and painted together with the vertex to which it is connected (**G**).



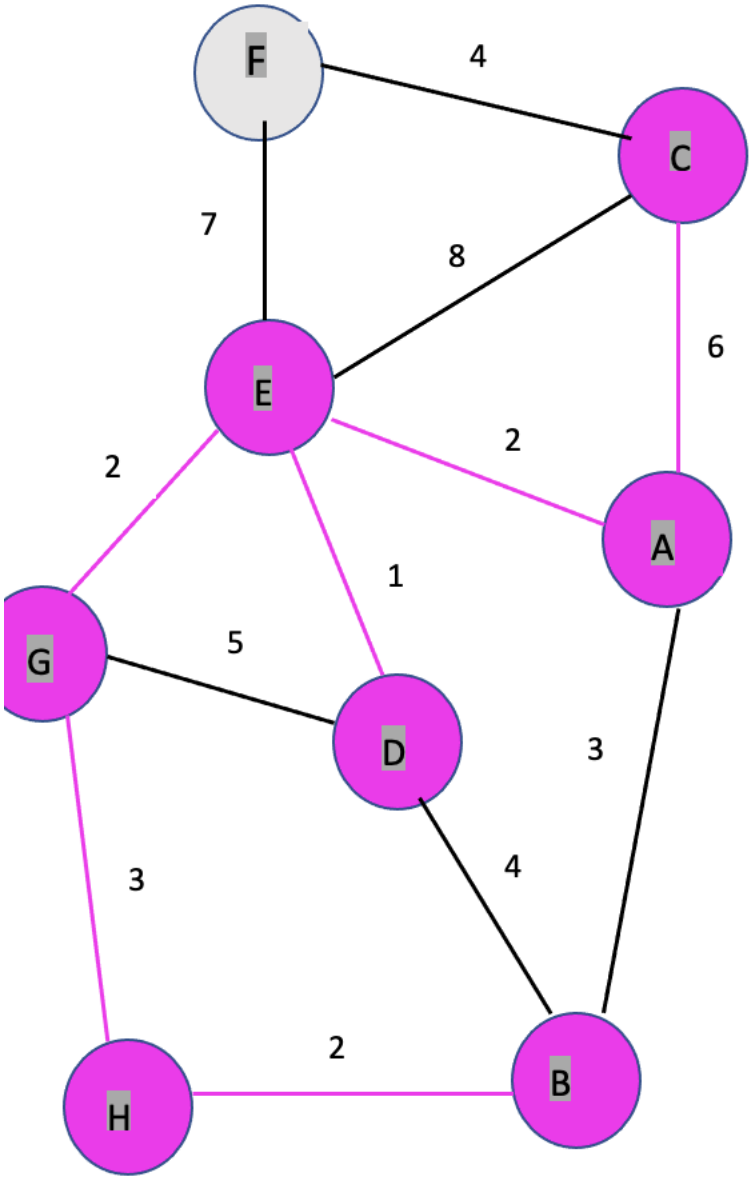
4. After the *G* vertex is painted pink, the edge that has one of the smallest edge cost and has the other end connected to an unpainted vertex is selected **(G-*H*)** and painted together with the vertex to which it is connected **(H)**.



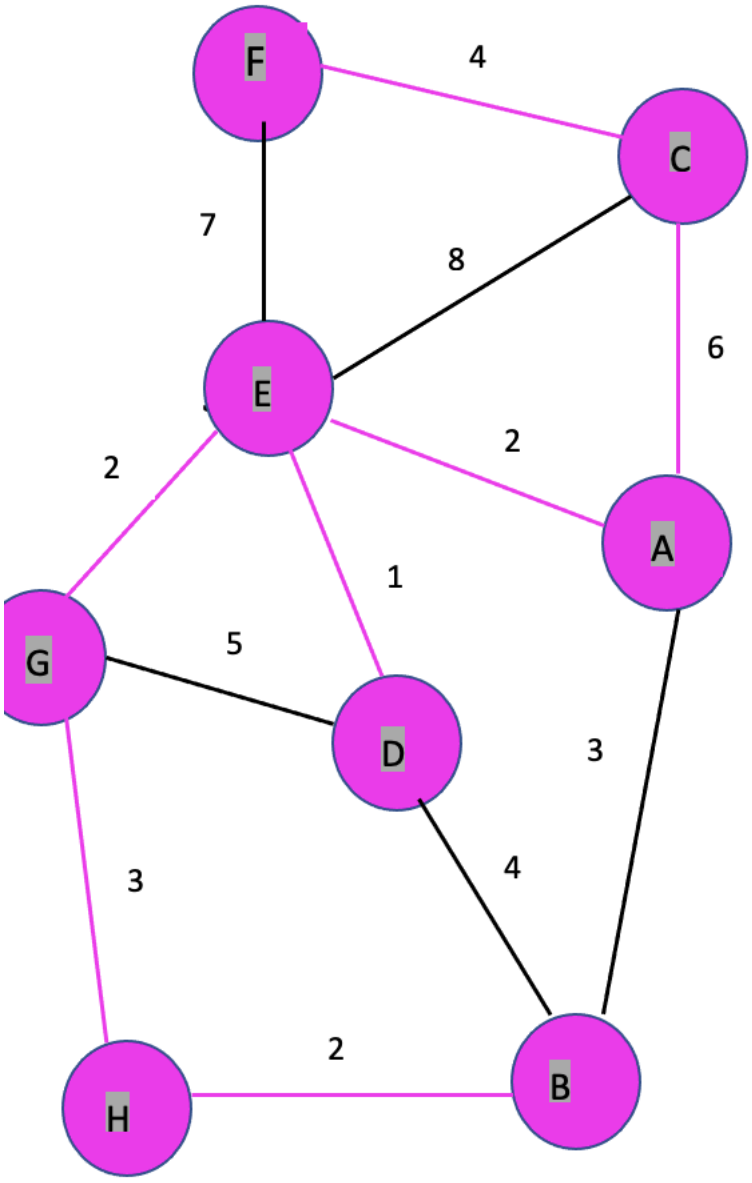
5. After the *H* vertex is painted pink, the edge that has one of the smallest edge cost and has the other end connected to an unpainted vertex is selected **(H-*B*)** and painted together with the vertex to which it is connected**(B).**



6. After the *B* vertex is painted pink, the edge that has one of the smallest edge cost and has the other end connected to an unpainted vertex is selected **(A-*C*)** and painted together with the vertex to which it is connected**(C).** I could have chosen the edge A-B but it would have caused a cycle so I didn't.



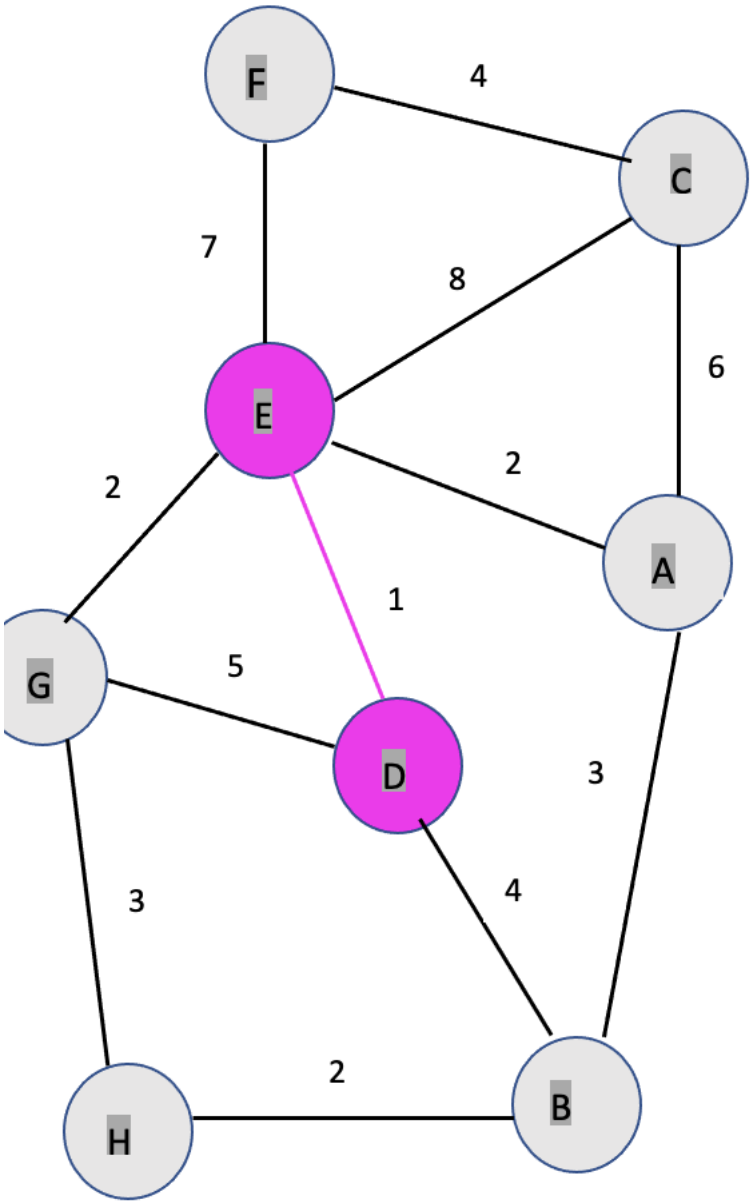
7. After the *C* vertex is painted pink, the edge that has one of the smallest edge cost and has the other end connected to an unpainted vertex is selected ***(F-C)*** and painted together with the vertex to which it is connected. And MST completed.



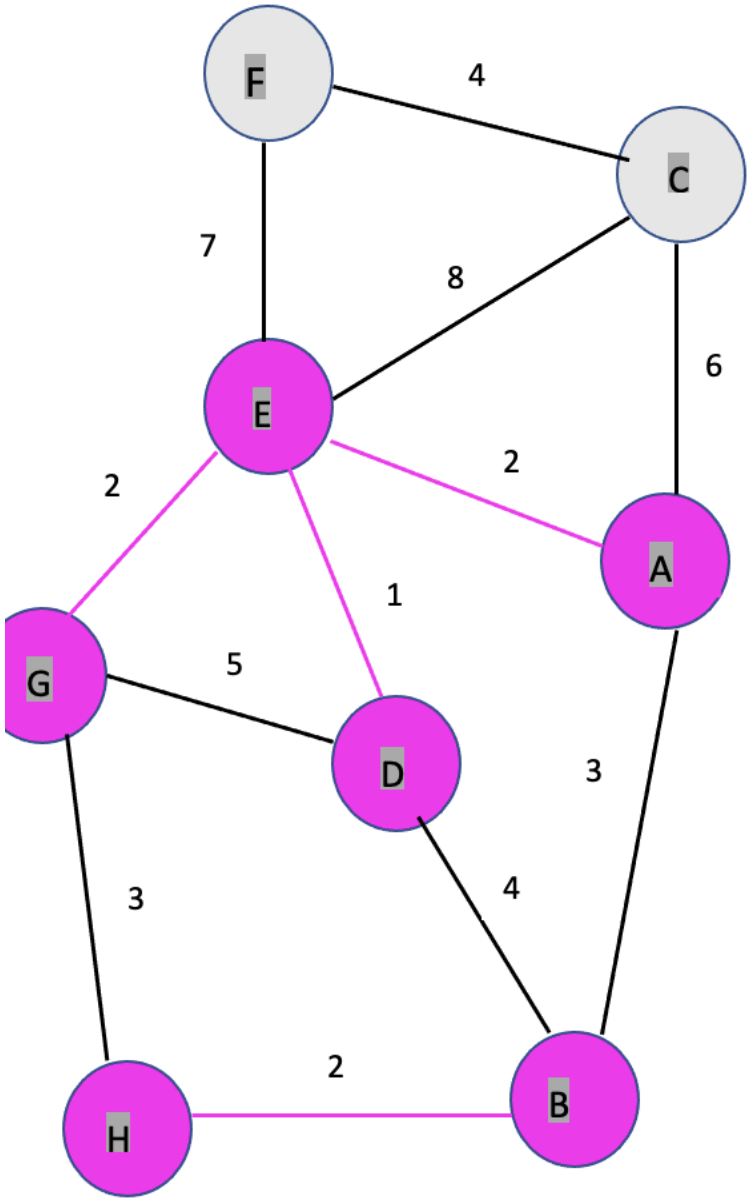
This is the final tree

Question 3:

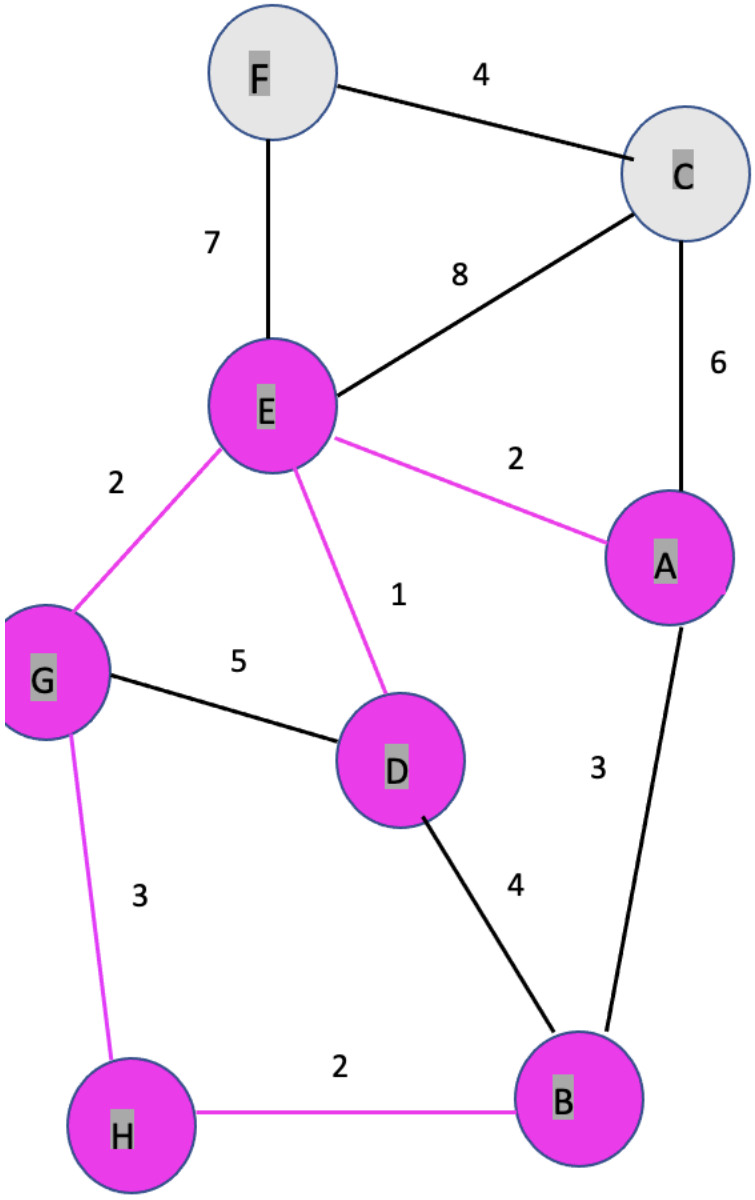
1. According to Kruskal's algorithm, edges are considered from smallest to largest, and if edge doesn't cause a cyle we add it to the tree. According to Figure 1, the edge with the lowest length value is between ***E-D*** and this edge and the vertex to which it is attached are painted pink.



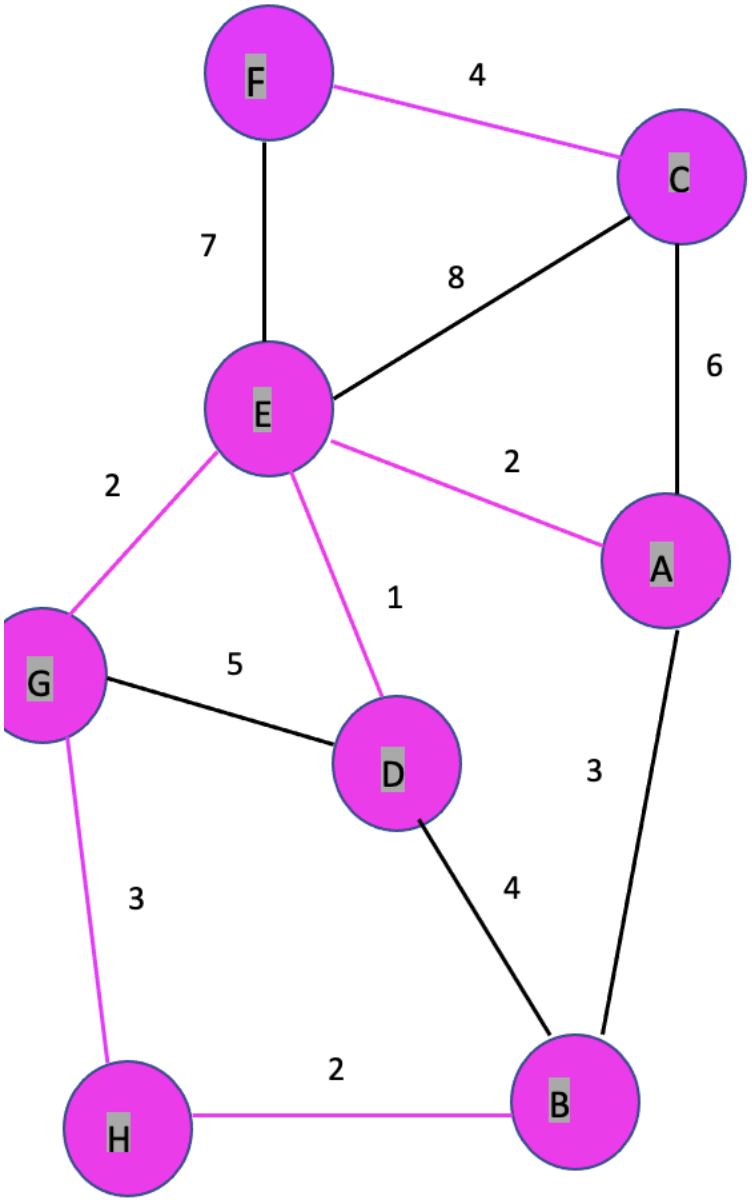
2. After the edge between *E-D* is painted, the edges with the smallest length values (***E-G, E-A, H- B)*** are painted together with the connected vertices, respectively, since it doesn't cause a cycle we can add it to the tree.



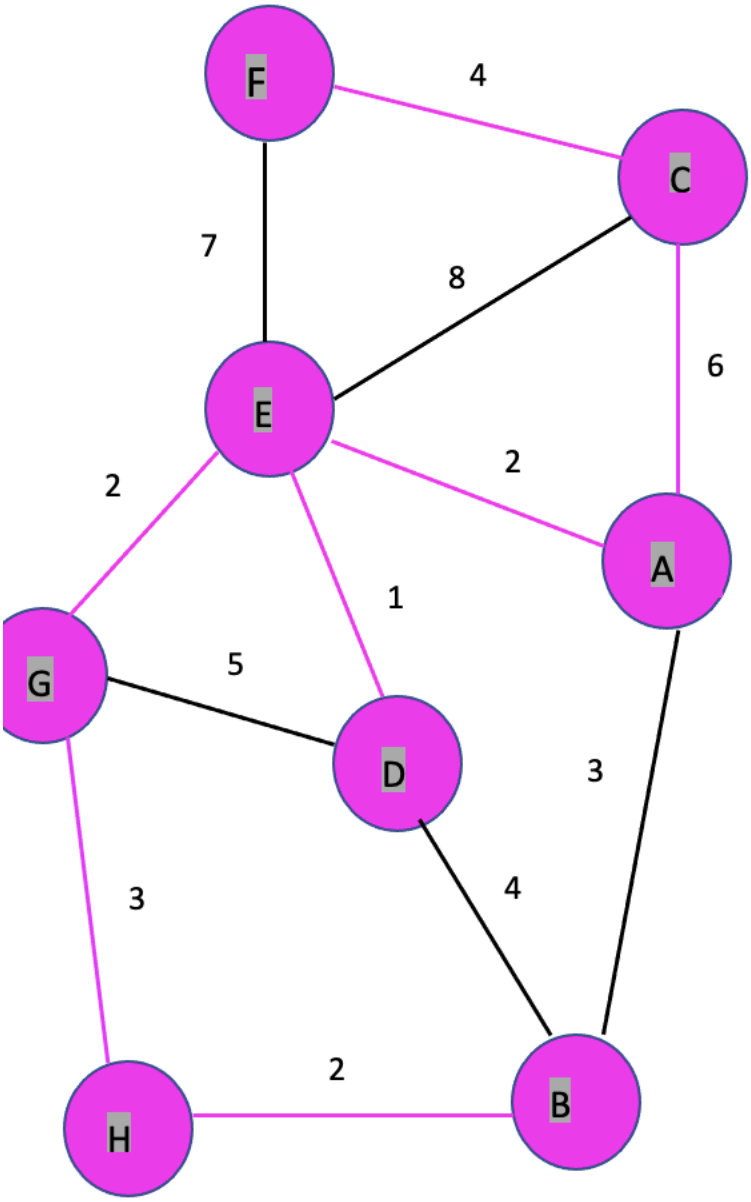
3. After the edges between *E-G*, *E-A*, *H- B* is painted, the edge with the smallest length values **(*G-H*)** is painted together with the connected vertex , respectively, since it doesn't cause a cycle we can add it to the tree.



4. After the edge between *G-H* is painted, the edge with the smallest length values **(*F-C*)** is painted together with the connected vertex, respectively, since it doesn't cause a cycle we can add it to the tree.



5. After the edge between *F-C* is painted, the edge with the smallest length value **(*A*-*C*)** is painted together with the connected vertex. And the algorithm is completed.

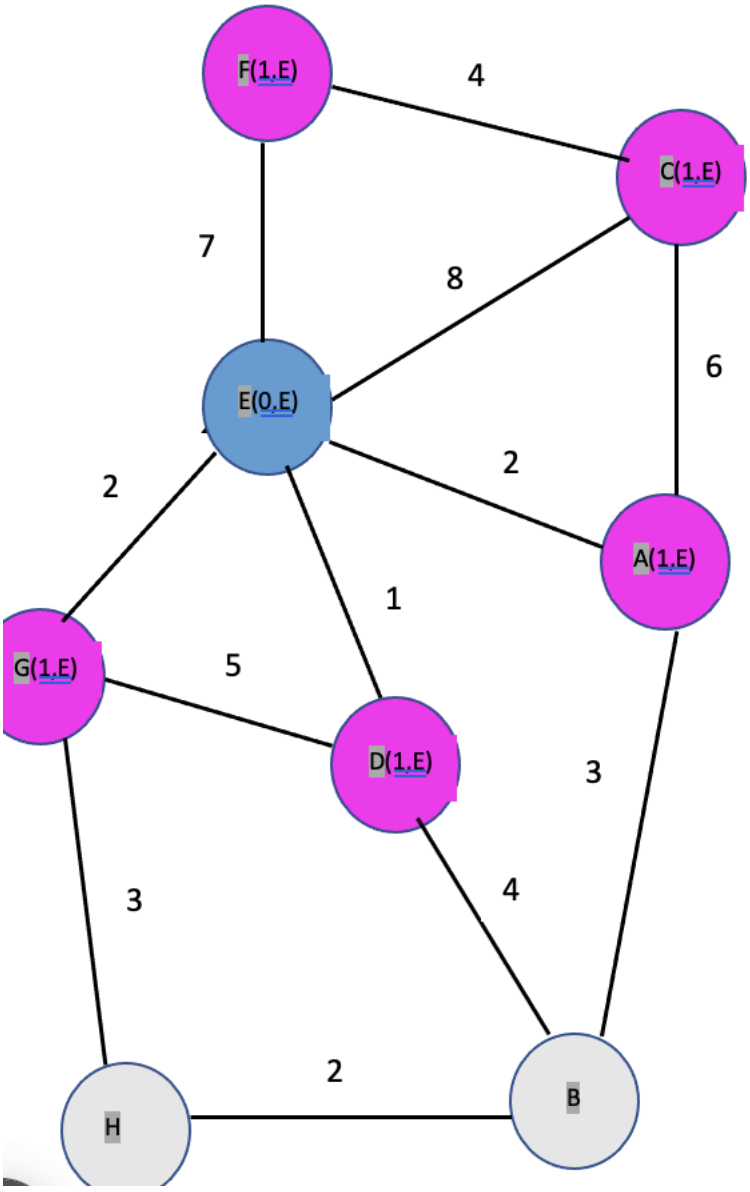


This is final tree.

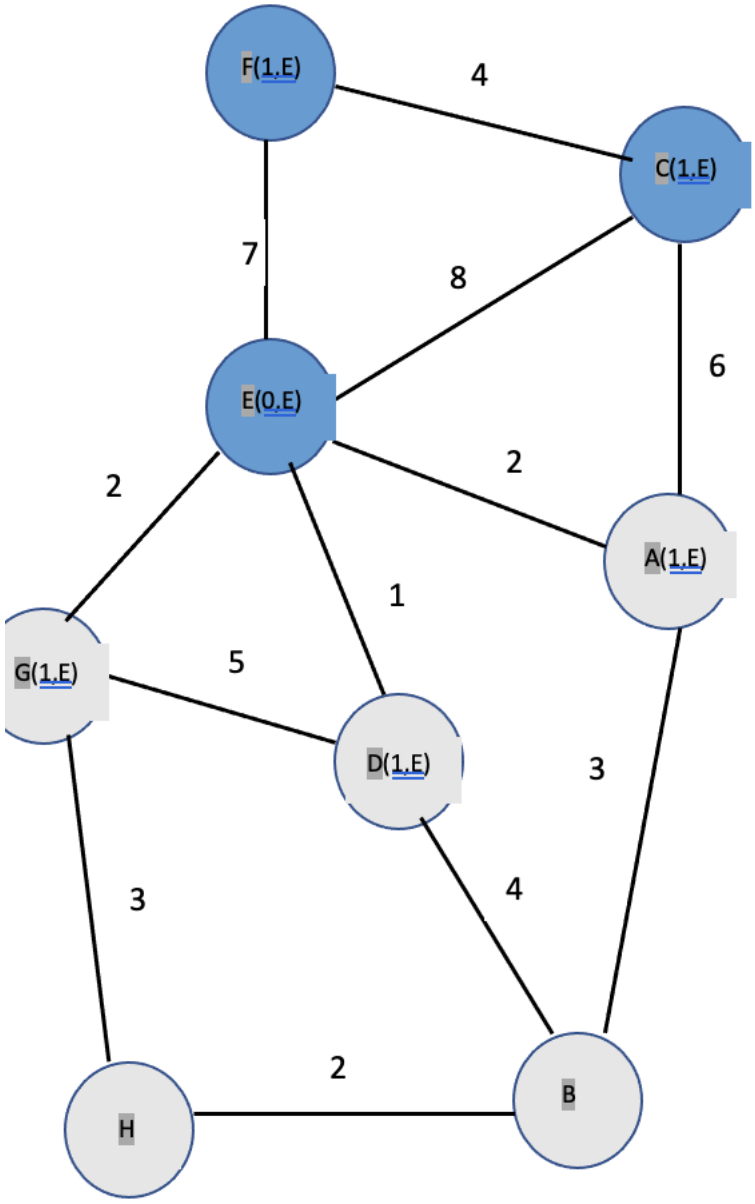
Question 4:

The BFS algorithm starts by enqueuing the root vertex and marking it as visited. Then, it dequeues the next vertex from the queue, examines all the unvisited adjacents of that vertex, and enqueues any that it finds. This process continues until the queue is empty, at which point all the reachable vertices in the graph have been visited.

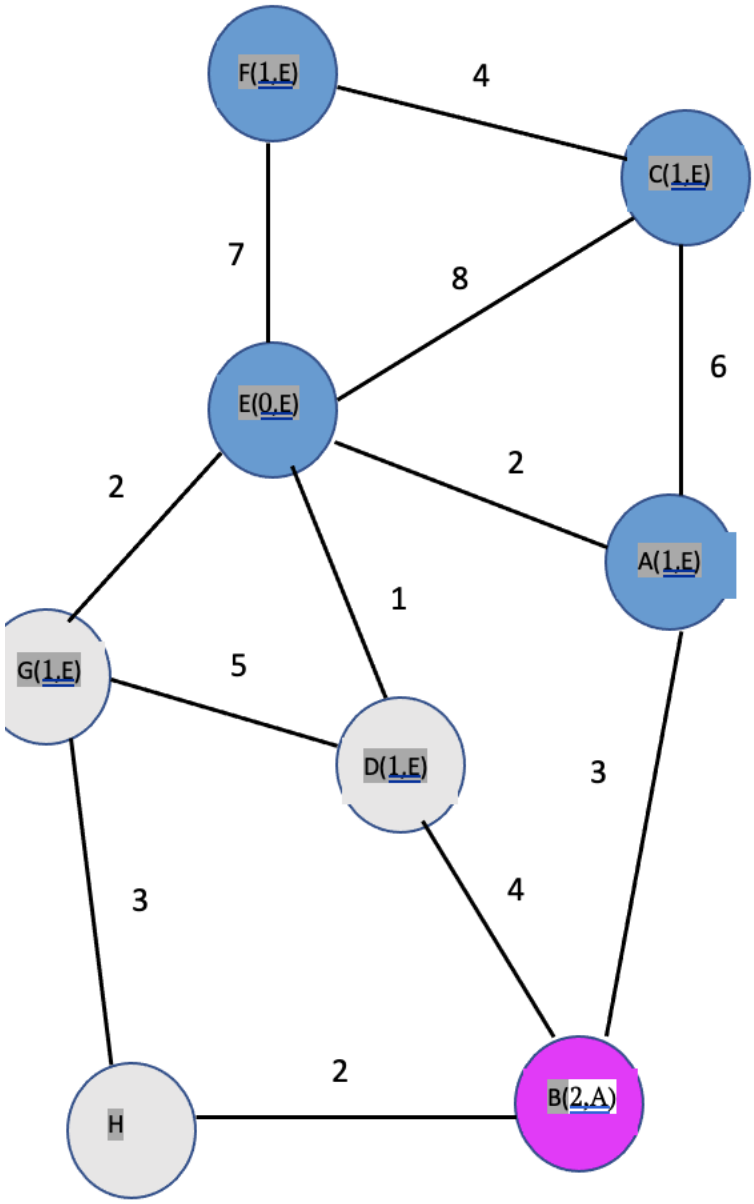
1. The starting vertex E has enqueued the queue, then E is dequeued and painted blue. Then for each vertex adjacent to vertex E is enqueued according to its distance to Vertex E and is painted pink.



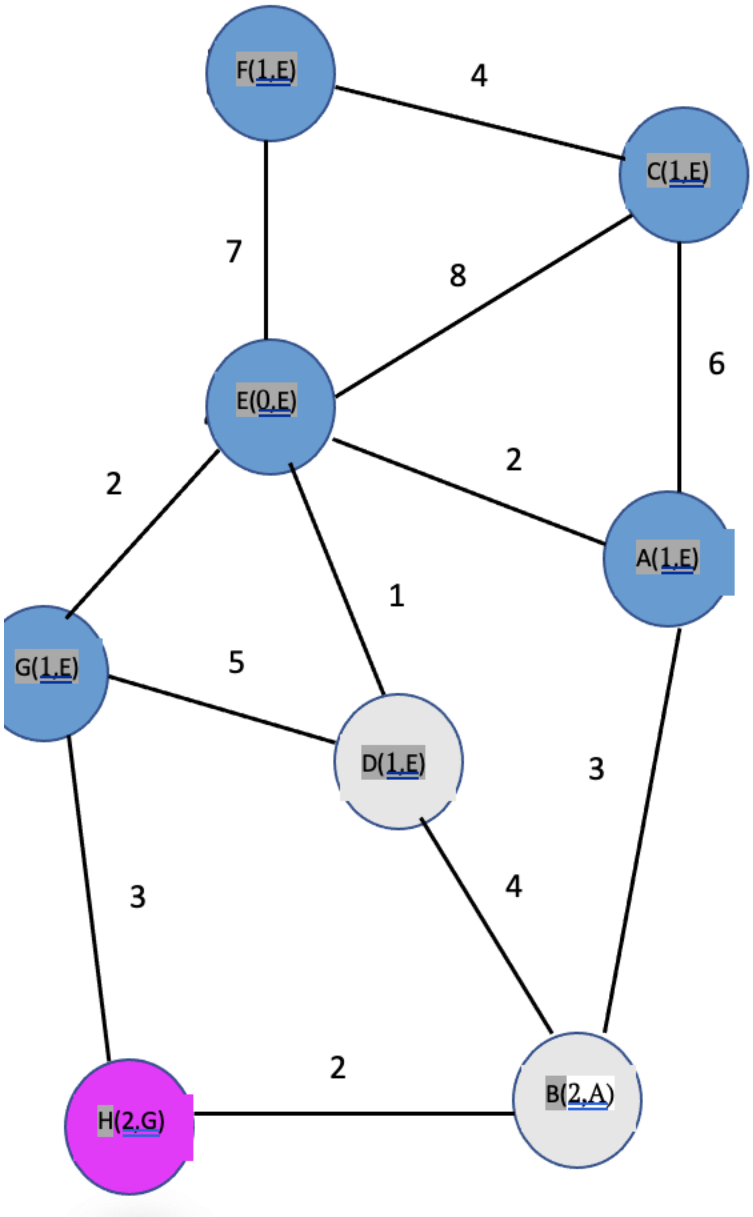
2. Vertex F after that Vertex C is dequeued and is painted blue. Because for both vertexes they don't have any adjacent vertices that have the distance field equal to infinity we don't do any enqueue action.



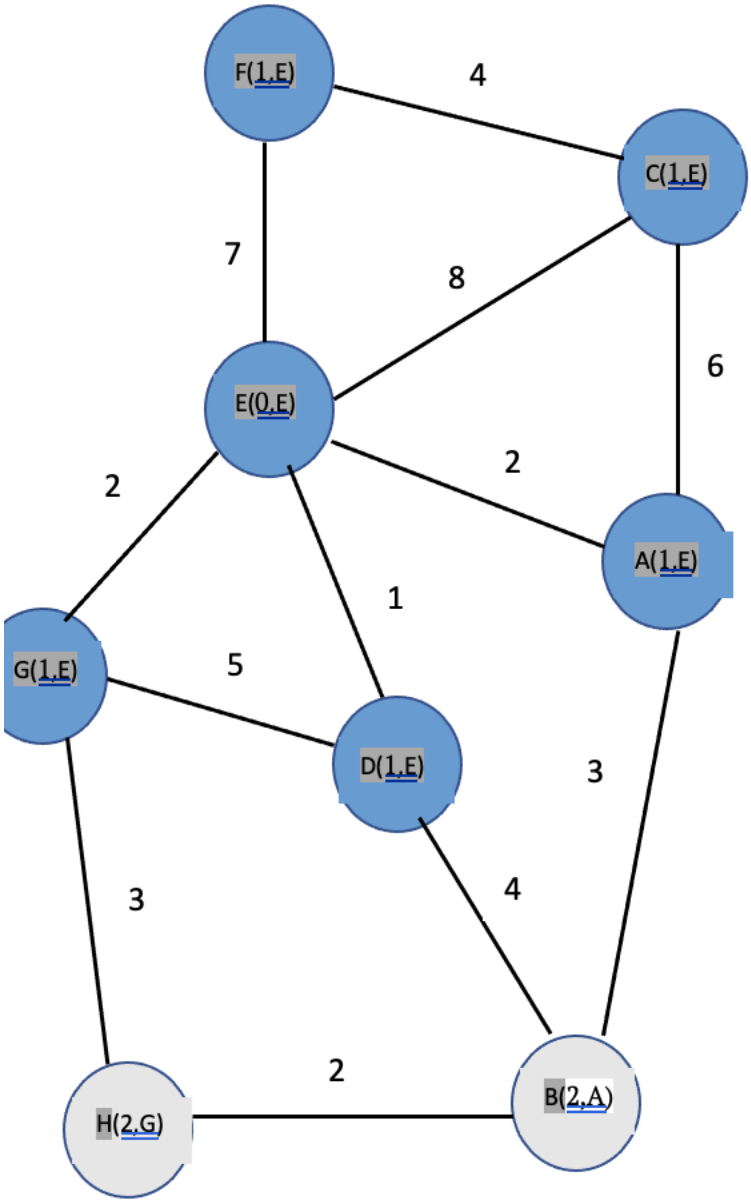
3. Vertex A is dequeued and is painted blue. Then for each vertex adjacent to vertex A is enqueued according to their distance to Vertex A and is painted pink.



4. Vertex G is dequeued and is painted blue. Then for each vertex adjacent to vertex G is enqueued according to their distance to Vertex G and is painted pink.

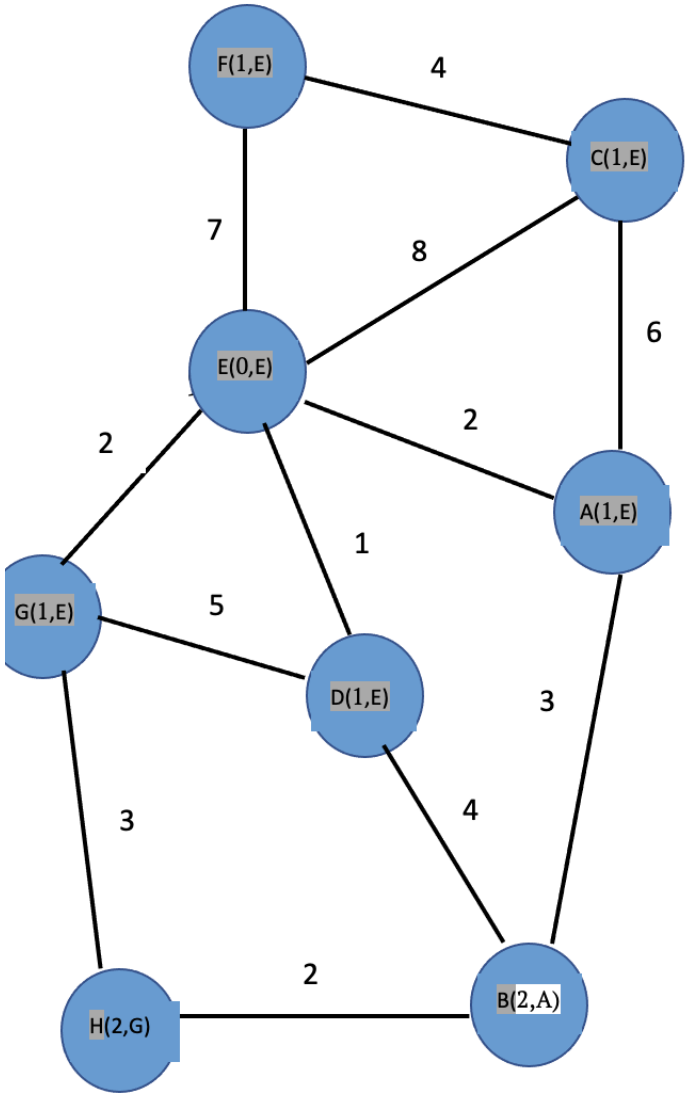


5. Vertex D is dequeued and is painted blue. Because Vertex D doesn't have any adjacent vertices that have the distance field equal to infinity we don't do any enqueue action.



6. Vertex B and Vertex H is dequeued in order and is painted blue. Because Vertex D doesn't have any adjacent vertices that have the distance field equal to infinity we don't do any enqueue action. And algorithm is completed.

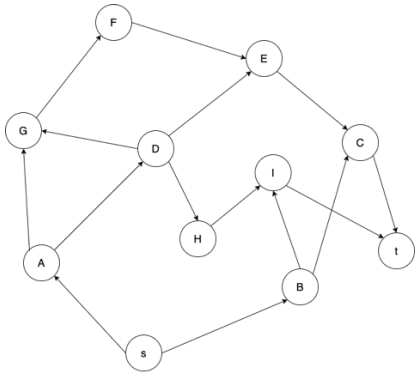
ORDER: *E-F-C-A-G-D-B-H*

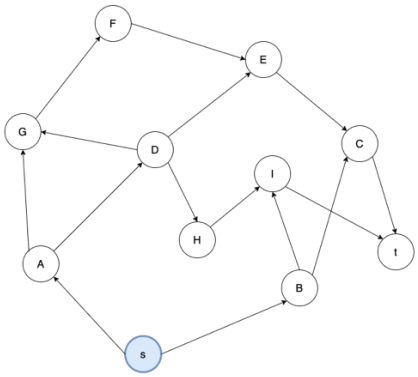


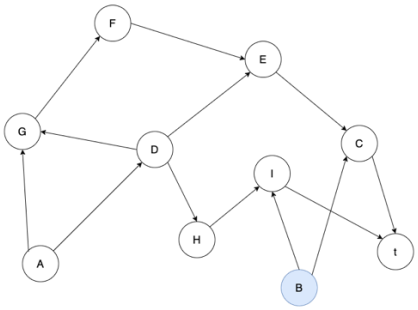
Question 5:

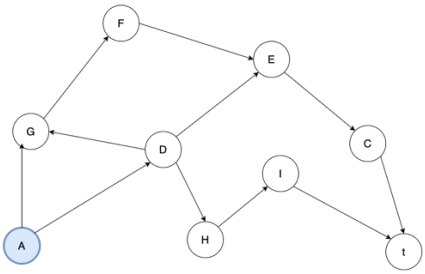
First, an unreachable vertex(**in degree =0**) is selected and printed. Then this vertex is removed from the graph and the same operation is continued until there is no vertex left in the graph. For the Topological sort there can be arbitrary choices of which vertex to remove and print because there can be multiple vertexes that has **in degree** value equal to zero. So the sort is not unique.

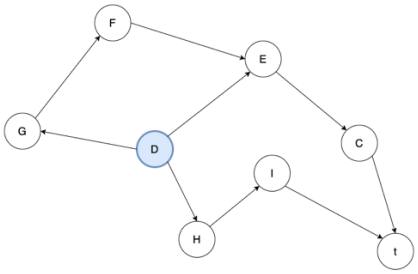
Order: S-B-A-D-G-H-F-I-E-C-T

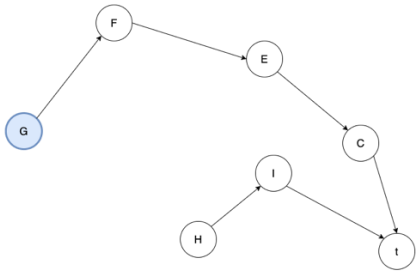

F
E
G
D
C
I
H
t
A
B
s

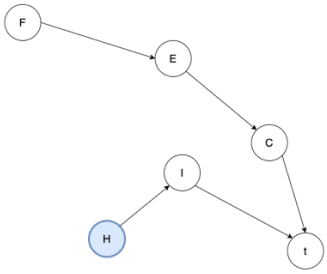

F
E
G
D
C
I
H
t
A
B
s

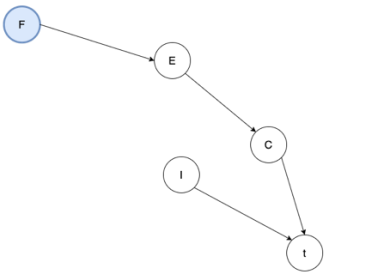

F
E
G
D
C
I
H
t
A
B

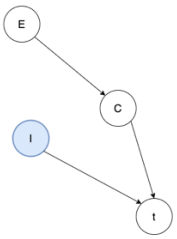

F
E
G
D
C
I
H
t
A

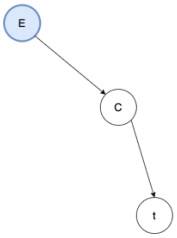

F
E
G
D
C
I
H
t


F
E
G
C
I
H
t


F
E
C
I
H
t


F
E
C
I
t


E
C
I
t


E
C
t


C
t


t