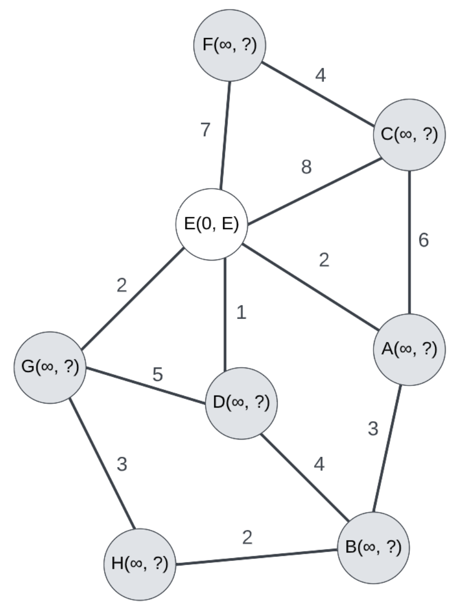
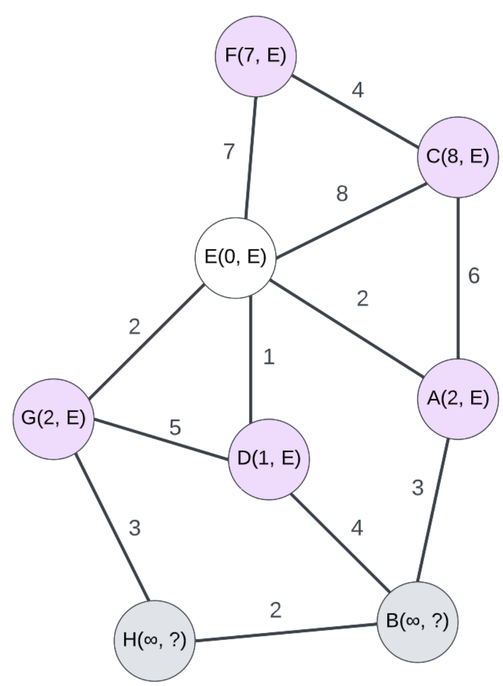
Question 1: Trace the Dijkstra's weighted shortest path algorithm on the graph given in Figure 1. Use vertex E as your start vertex.

Distance values of vertices adjacent to starting vertex E are placed.

Vertex	Shortest distance from E	Previous Vertex
E	0	
A	∞	
B	∞	
C	∞	
D	∞	
G	∞	
H	∞	
F	∞	



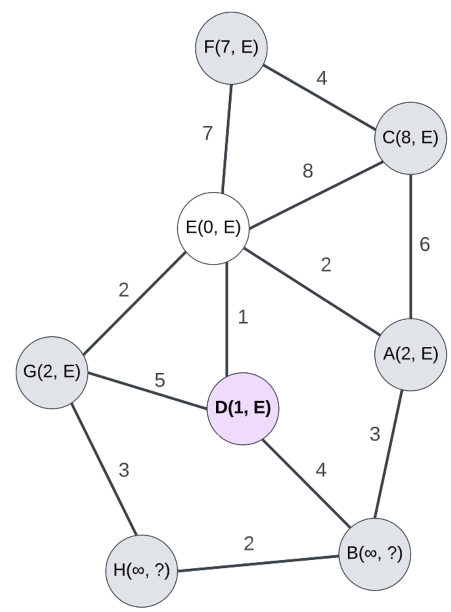
Vertex	Shortest distance from E	Previous Vertex
E	0	
A	∞	
B	∞	
C	∞	
D	∞	
G	∞	
H	∞	
F	∞	



The vertex with the minimum distance value is picked, which is vertex D and then the distance value(s) of the adjacent vertices to vertex D is/are placed.

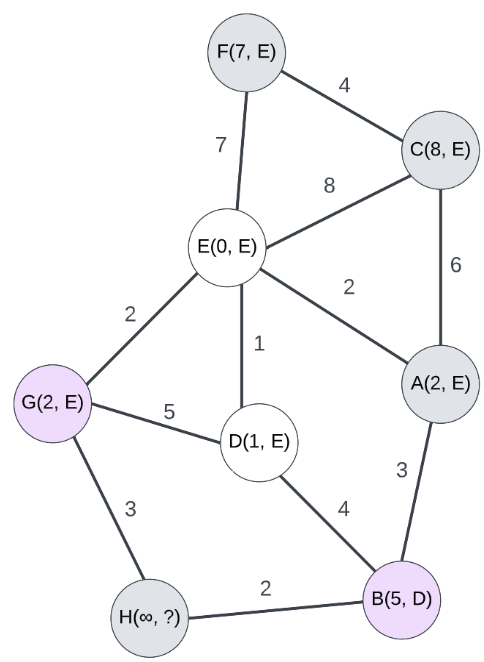
1. Vertex G: (2, E) < 1+5 = (6, G). Since 6 = 6 vertex 6 stays the same
2. Vertex B: 1+4 = (5, D) < (∞,?), B changes to 5,D

Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	2	E
B	∞	
C	8	E
D	1	E
G	2	E
H	∞	
F	7	E



Visited = [E]　　　　　　　　Selected minimum = [D]

Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	2	E
B	5	D
C	8	E
D	**1**	**E**
G	2	E
H	∞	
F	7	E

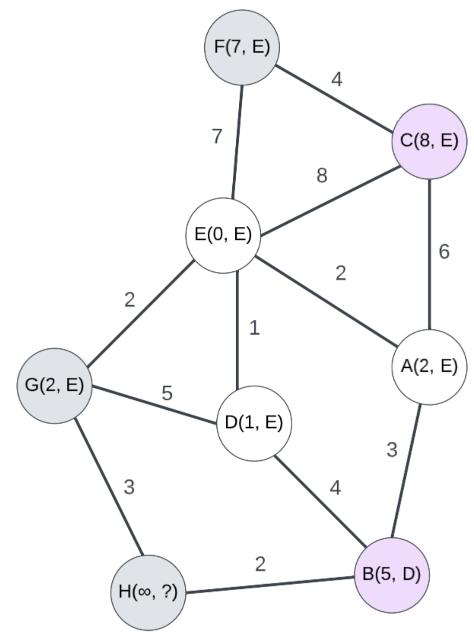


Visited = [E, D]　　　　　　Selected minimum = [A]

All operations related to vertex D are finished, the remaining vertices are painted back to gray. Since vertex A has the minimum distance value it is painted white and A's adjacent vertices B and C are painted purple.

1. Vertex B: 2 + 3 = (**5, A**)= (**5, D**) Since 5 = 5 vertex B remains the same
2. Vertex C: 2 + 6 = (**8, A)**= (**8, E**) Since 8 = 8 vertex C remains the same

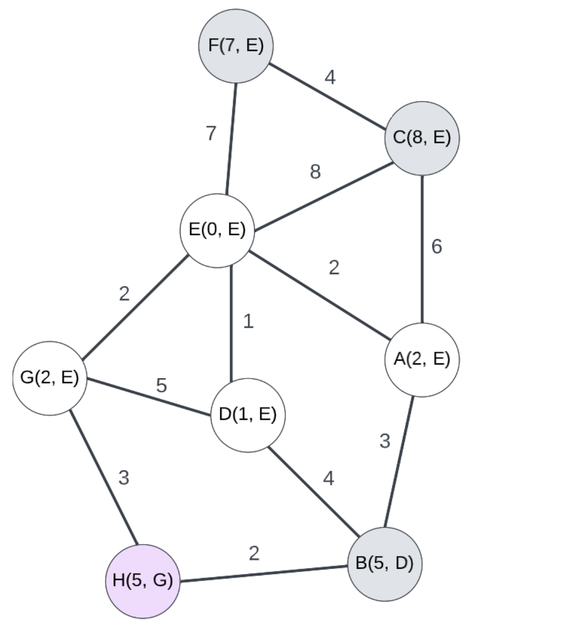
Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	**2**	**E**
B	5	D
C	8	E
D	**1**	**E**
G	2	E
H	∞	
F	7	E



Visited = [E, D, A]   Selected minimum = [G]

All operations related to vertex A are finished, the remaining vertices are painted back to gray. Since vertex G has the minimum distance value it is painted white and G's adjacent vertex H is painted purple. Since Vertex H: 2+3 = 5 < ∞, H changes to (5,G)

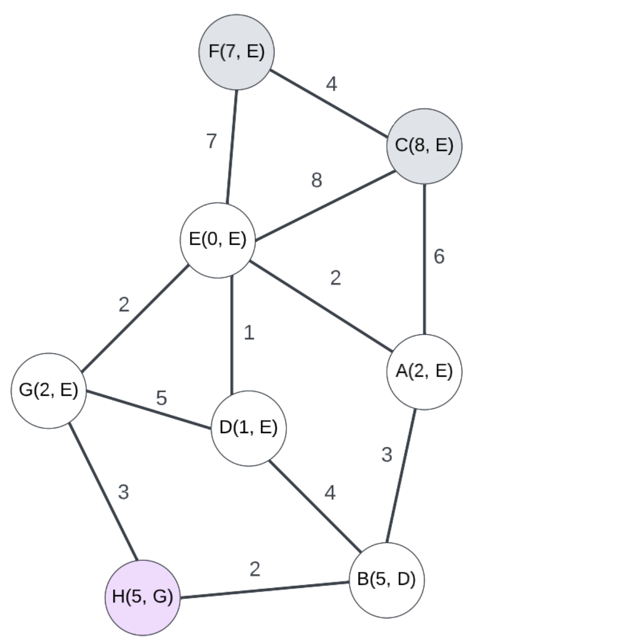
Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	**2**	**E**
B	5	D
C	8	E
D	**1**	**E**
G	**2**	**E**
H	5	G
F	7	E



Visited = [E, D, A, G] Selected minimum = [B]

All operations related to vertex G are finished, the remaining vertices are painted back to gray. Since vertex B has the minimum distance value it is painted white and B's adjacent vertex H is painted purple. Vertex H: Since 5+2 > 5, H remains unchanged.

Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	**2**	**E**
B	**5**	**D**
C	8	E
D	**1**	**E**
G	**2**	**E**
H	5	G
F	7	E

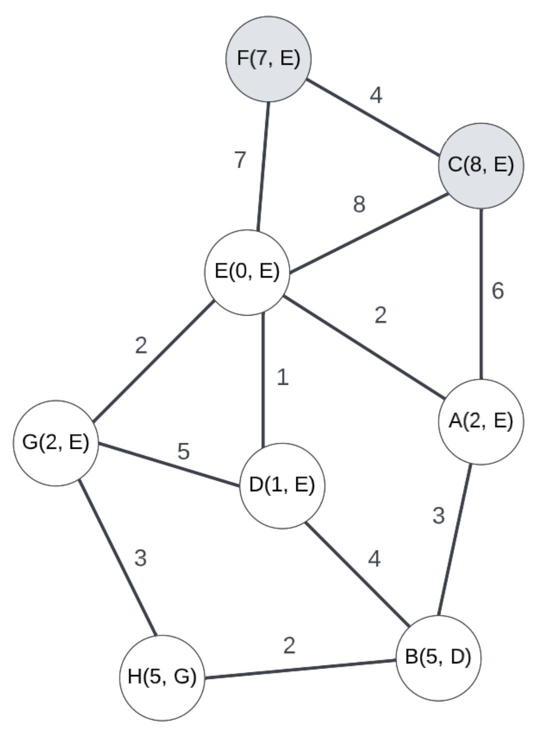


Visited = [E, D, A, G, B] Selected minimum = [H]

All operations related to vertex B are finished, the remaining vertices are painted back to gray.

Since vertex H has the minimum distance value it is painted white and H’s adjacent vertices are G and B which are already traces, so nothing happens.

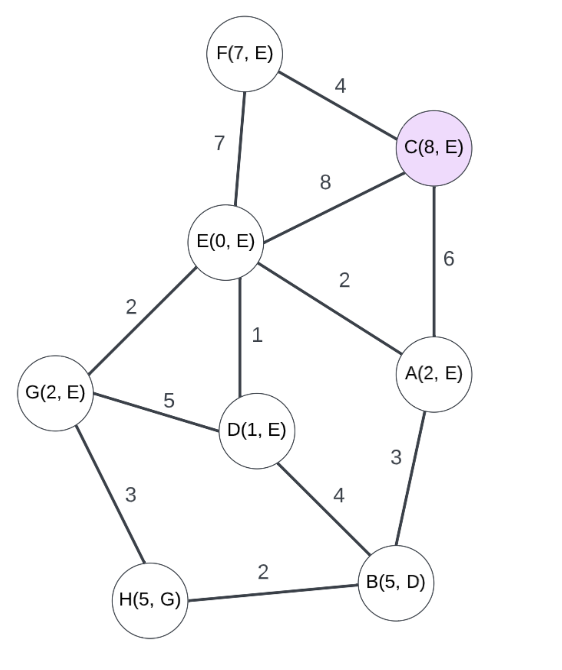
Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	**2**	**E**
B	**5**	**D**
C	8	E
D	**1**	**E**
G	**2**	**E**
H	**5**	**G**
F	7	E



Visited = [E, D, A, G, B, H]   Selected minimum = [F]

All operations related to vertex H are finished. Since vertex F has the minimum distance value it is painted white and F’s adjacent vertex C is painted purple. Vertex H: Since 7+4 > 8, C remains unchanged.

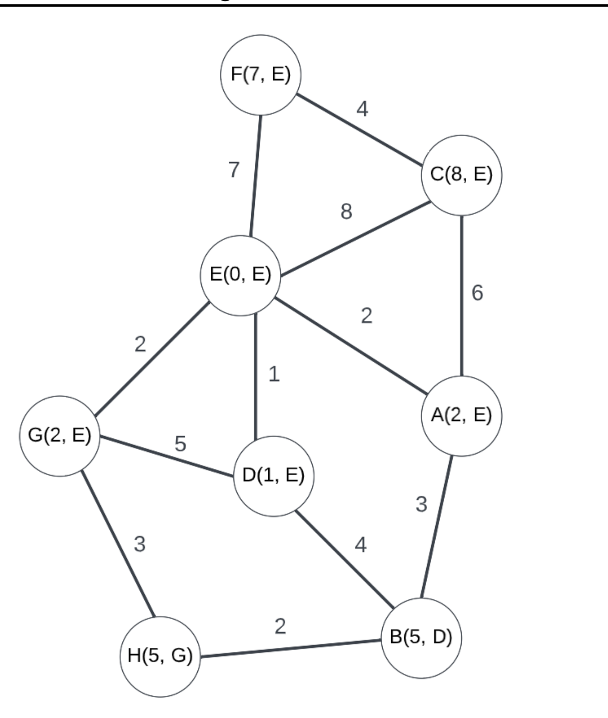
Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	**2**	**E**
B	**5**	**D**
C	8	E
D	**1**	**E**
G	**2**	**E**
H	**5**	**G**
F	**7**	**E**



Visited = [E, D, A, G, B, H, F]   Selected minimum = [C]

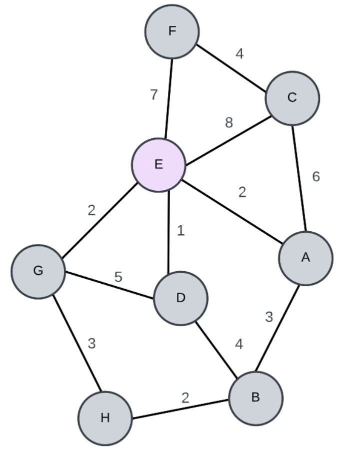
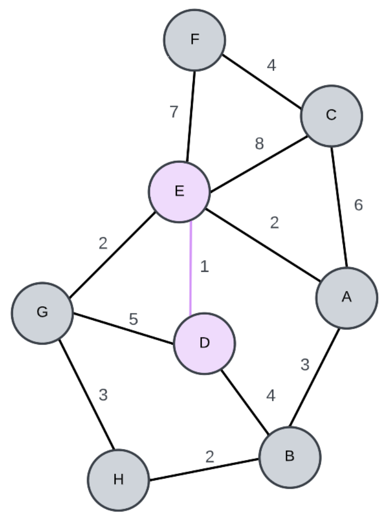
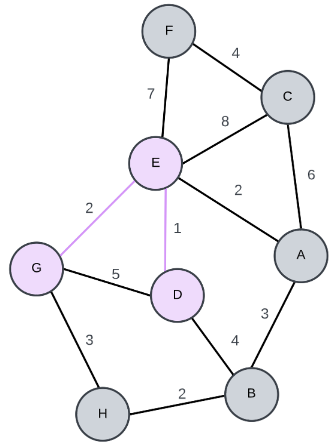
All operations related to vertex F are finished. Since vertex C has the minimum distance value it is painted white since all the other vertices are traced. The algorithm is over.

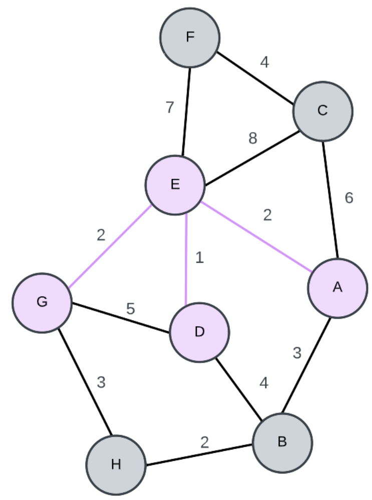
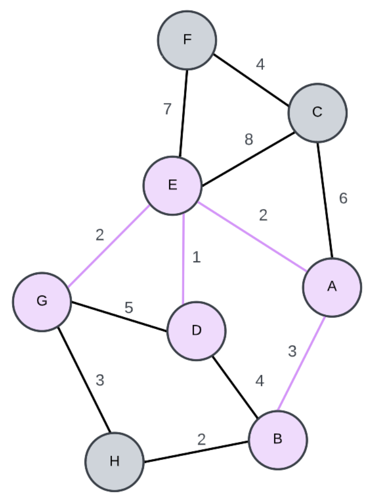
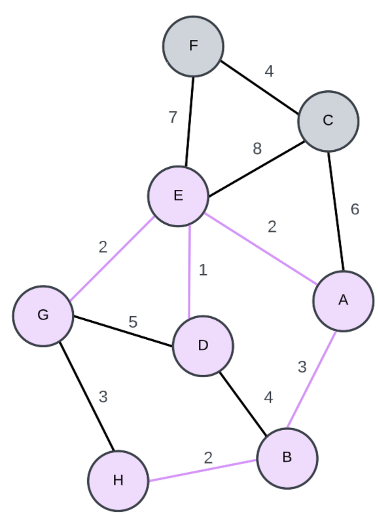
Vertex	Shortest distance from E	Previous Vertex
E	**0**	
A	**2**	**E**
B	**5**	**D**
C	**8**	**E**
D	**1**	**E**
G	**2**	**E**
H	**5**	**G**
F	**7**	**E**

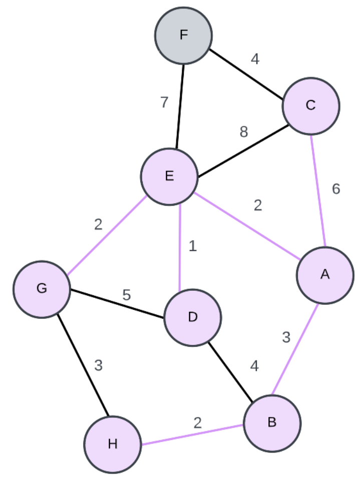
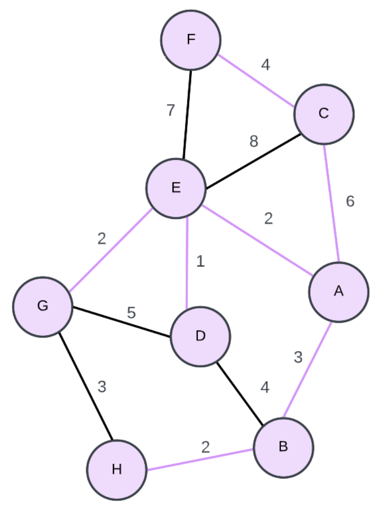
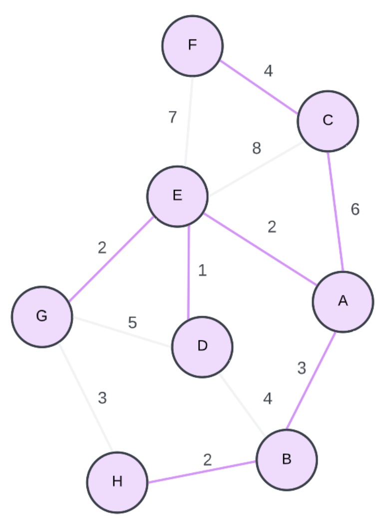


Visited = [E, D, A, G, B, H, F, C]
All vertices are traced.

Question 2: Trace the Prim's minimum spanning tree algorithm on the graph in Figure 1. Use vertex E as your start vertex.

We start the algorithm from vertex E so it is painted in purple. MST = E	
We chose the minimum distance edge which is connected to E and also which would not make a loop. D (DE) is chosen and painted purple together with the connecting edge. MST = E, D	
We chose the minimum distance edge which is connected to E or D and also which would not make a loop. G (GE) is chosen and painted purple together with the connecting edge. MST = E, D, G	

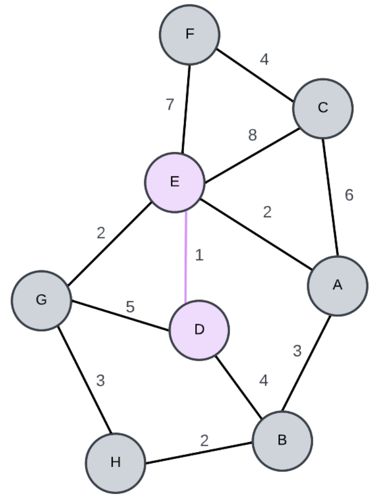
We chose the minimum distance edge which is connected to E, D, or G and also which would not make a loop. A (AE) is chosen and painted purple together with the connecting edge. MST = E, D, G, A	
We chose the minimum distance edge which is connected to E, D, G, or A and also which would not make a loop. B (BA) is chosen and painted purple together with the connecting edge. MST = E, D, G, A, B	
We chose the minimum distance edge which is connected to E, D, G, A or B and also which would not make a loop. H (HB) is chosen and painted purple together with the connecting edge. MST = E, D, G, A, B, H	

We chose the minimum distance edge which is connected to E, D, G, A, B or H and also which would not make a loop. C (CA) is chosen and painted purple together with the connecting edge. MST = E, D, G, A, B, H, C	
We chose the minimum distance edge which is connected to E, D, G, A, B, H or C and also which would not make a loop. F (FC) is chosen and painted purple together with the connecting edge. MST = E, D, G, A, B, H, C, F	
The final version is as such: MST = E, D, G, A, B, H, C, F	

Question 3: Trace the Kruskal’s minimum spanning tree algorithm.

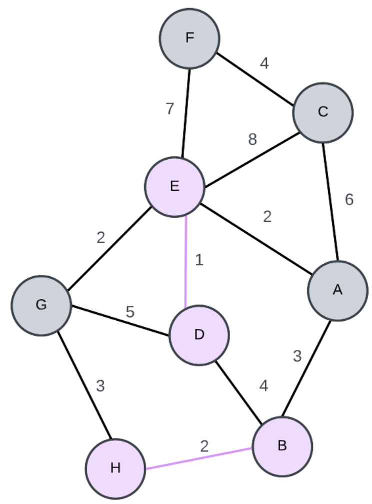
We start the algorithm by choosing the edge with the minimum distance value which is the edge between E and D. (If the distance values are the same either one can be picked)

Path: E-D



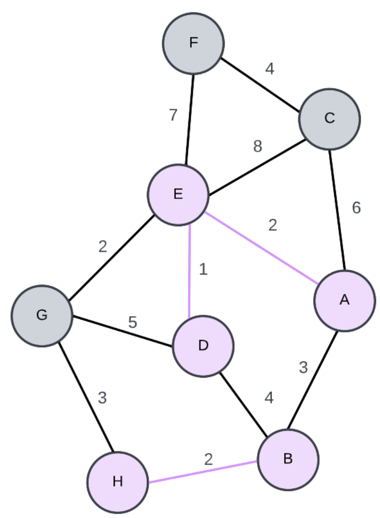
We start the algorithm by choosing the edge with the minimum distance value which is the edge between H and B. (If the distance values are the same either one can be picked)

Path: E-D, H-B

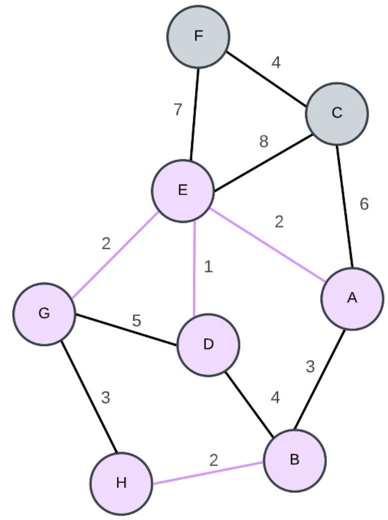


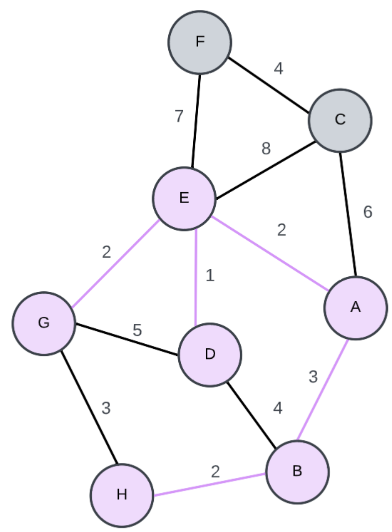
We start the algorithm by choosing the edge with the minimum distance value which is the edge between E and A. (If the distance values are the same either one can be picked)

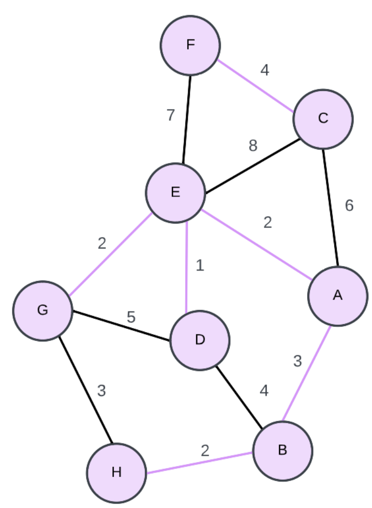
Path: E-D, H-B, E-A

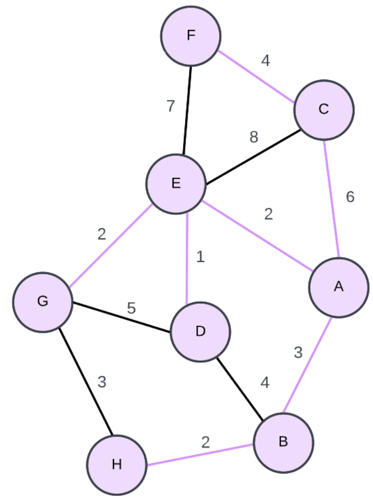
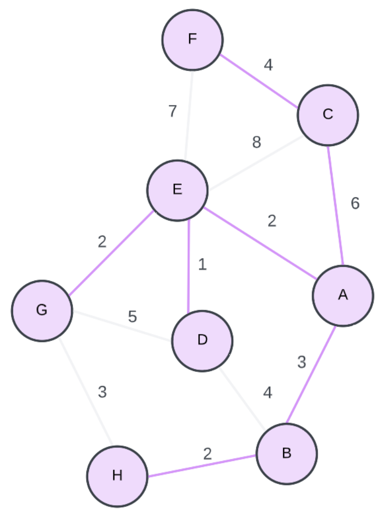


<table>
<tr>
<td>We start the algorithm by choosing the edge with the minimum distance value which is the edge between E and G. (If the distance values are the same either one can be picked)

Path: E-D, H-B, E-A, E-G</td>
<td>
</td>
</tr>
<tr>
<td>We start the algorithm by choosing the edge with the minimum distance value which is the edge between A and B. (If the distance values are the same either one can be picked)

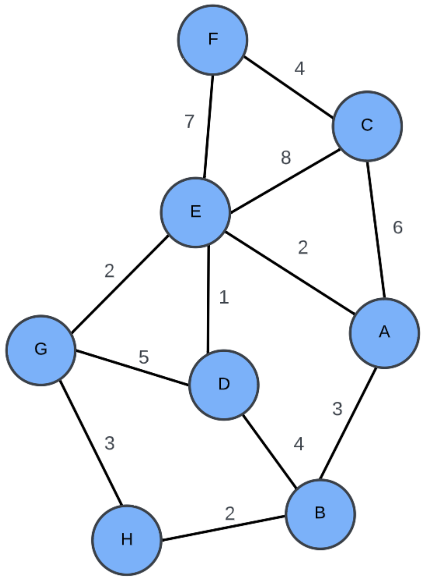
Path: E-D, H-B, E-A, E-G, A-B</td>
<td>
</td>
</tr>
<tr>
<td>We start the algorithm by choosing the edge with the minimum distance value which is the edge between F and C. (If the distance values are the same either one can be picked)

Path: E-D, H-B, E-A, E-G, A-B, F-C</td>
<td>
</td>
</tr>
</table>

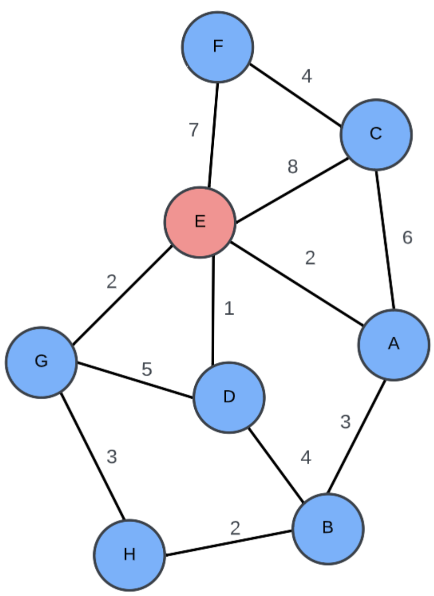
We start the algorithm by choosing the edge with the minimum distance value which is the edge between C and A. (If the distance values are the same either one can be picked) Path: E-D, H-B, E-A, E-G, A-B, F-C, C-A	
The final version is as such: Path: E-D, H-B, E-A, E-G, A-B, F-C, C-A	

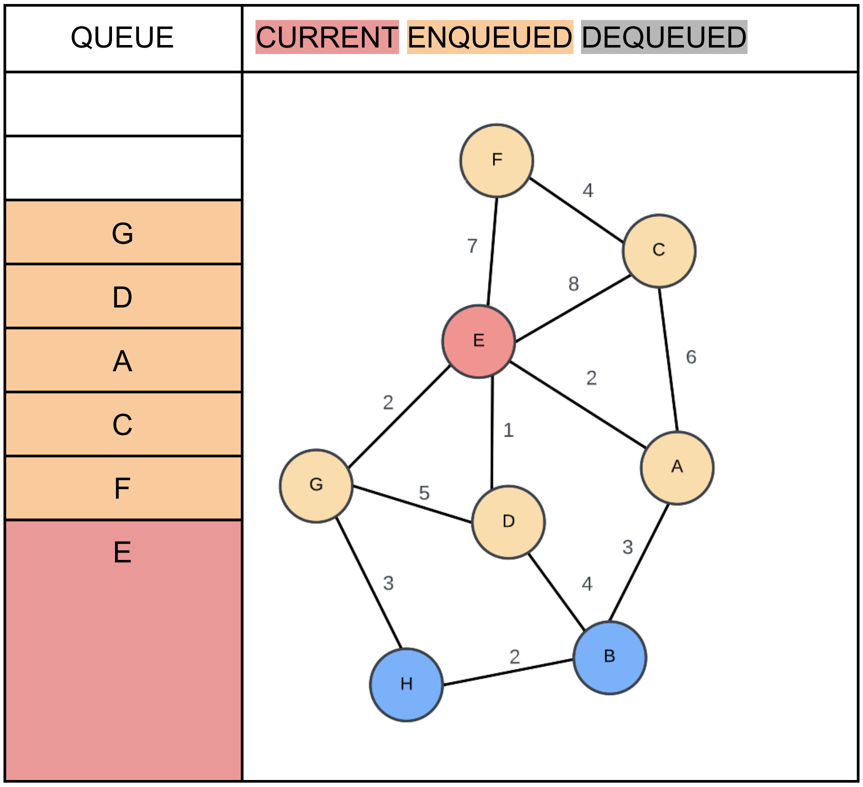
Question 4: Trace the breadth-first search traversal algorithm.

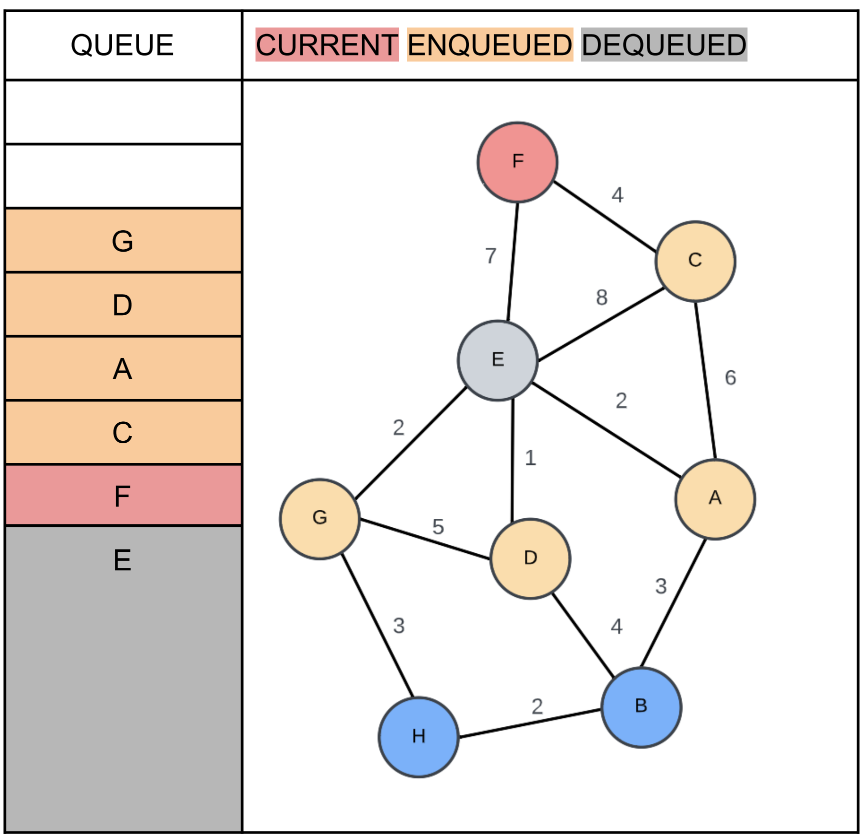
QUEUE	CURRENT ENQUEUED DEQUEUED

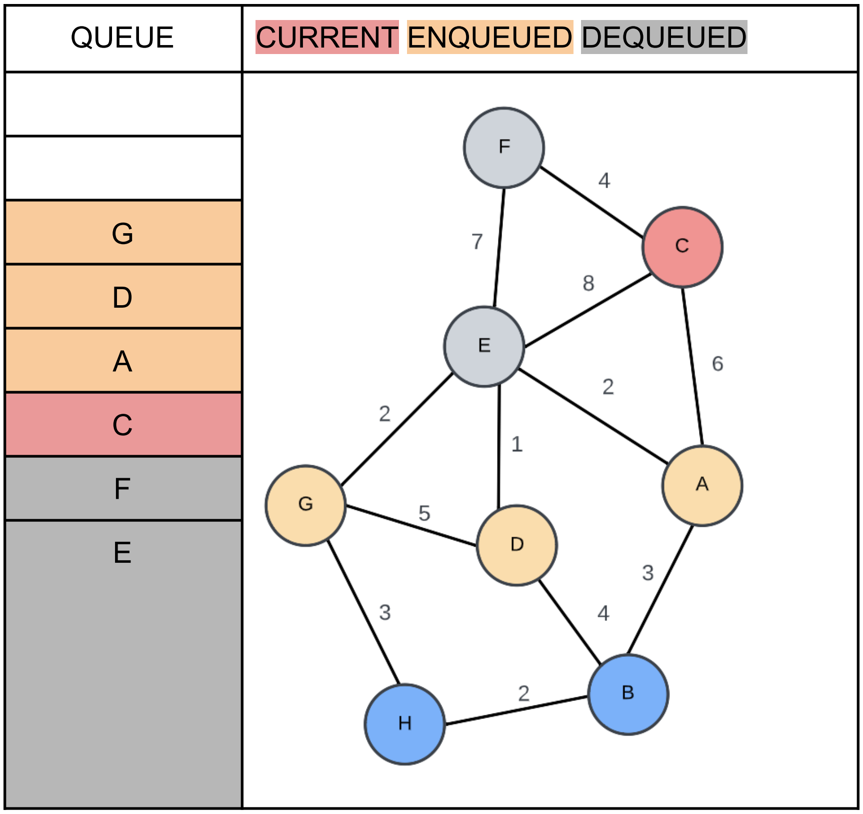


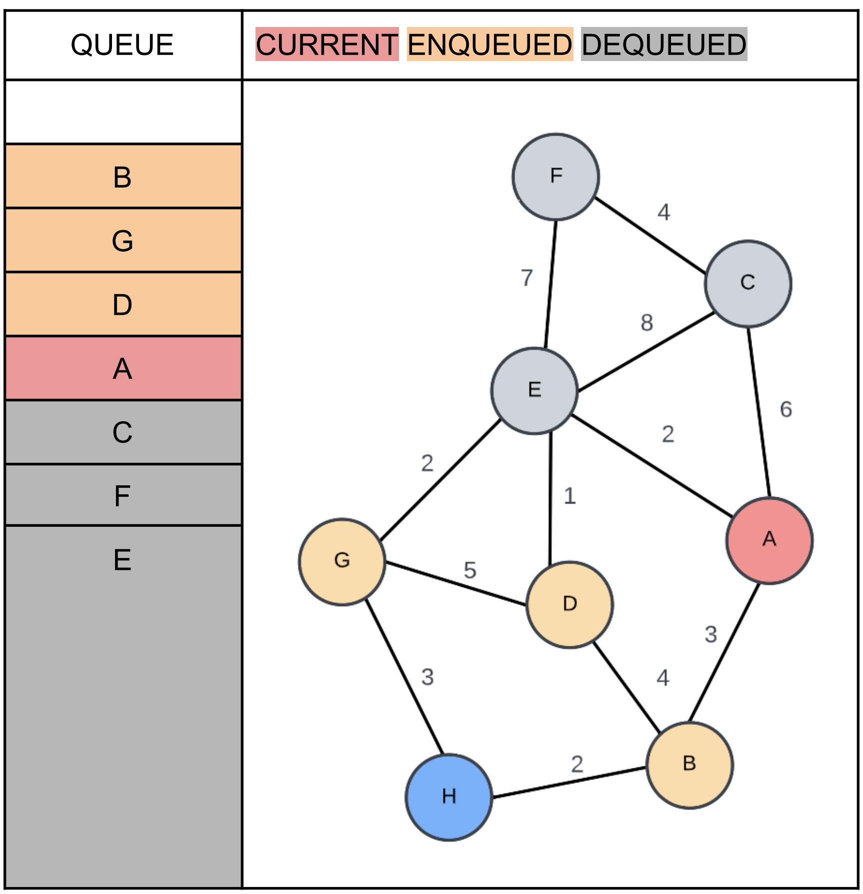
QUEUE	CURRENT ENQUEUED DEQUEUED
E	

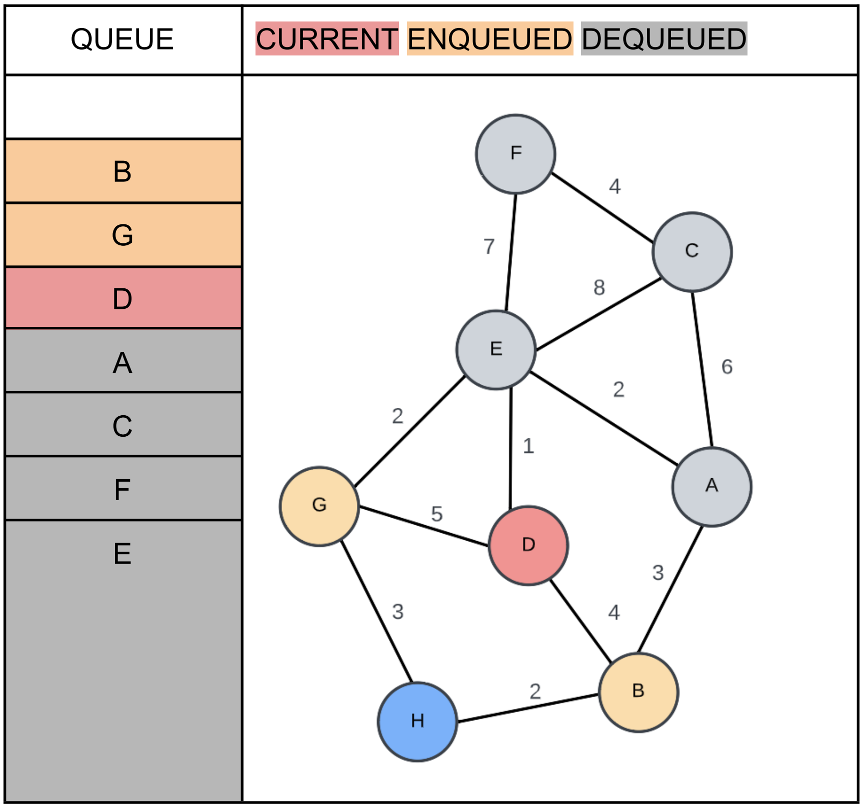


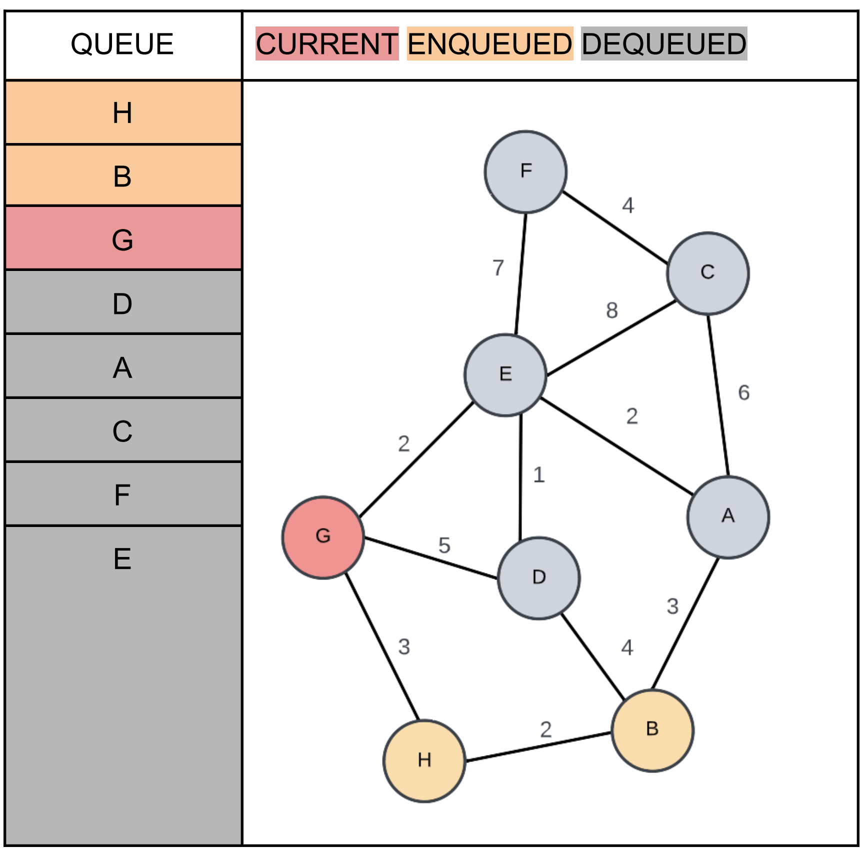

QUEUE
CURRENT
ENQUEUED
DEQUEUED
G
D
A
C
F
E
F
C
E
G
D
A
H
B
4
7
8
6
2
2
1
5
3
3
4
2


QUEUE
CURRENT
ENQUEUED
DEQUEUED
G
D
A
C
F
E
F
C
E
G
D
A
H
B
4
7
8
6
2
2
1
5
3
3
4
2


QUEUE
CURRENT ENQUEUED DEQUEUED
G
D
A
C
F
E
F
C
E
G
D
A
H
B
4
7
8
6
2
2
1
5
3
3
4
2

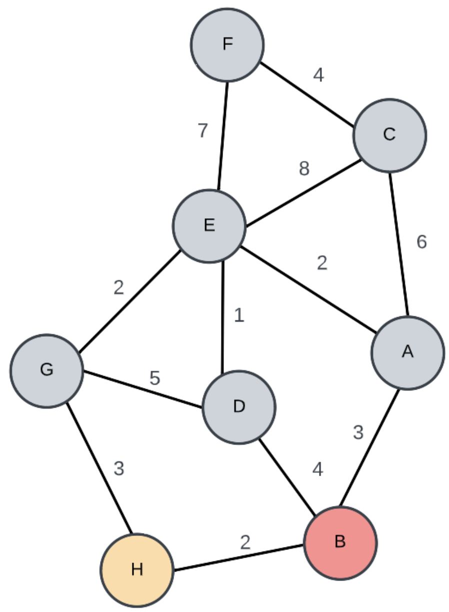

QUEUE
CURRENT ENQUEUED DEQUEUED
B
G
D
A
C
F
E
F
C
E
G
D
A
H
B
4
7
8
6
2
2
1
5
3
3
4
2


QUEUE
CURRENT ENQUEUED DEQUEUED
B
G
D
A
C
F
E
F
C
E
A
G
D
H
B
4
7
8
6
2
2
1
5
3
4
3
2


QUEUE
CURRENT ENQUEUED DEQUEUED
H
B
G
D
A
C
F
E
F
C
E
A
G
D
H
B
4
7
8
6
2
2
1
5
3
4
3
2

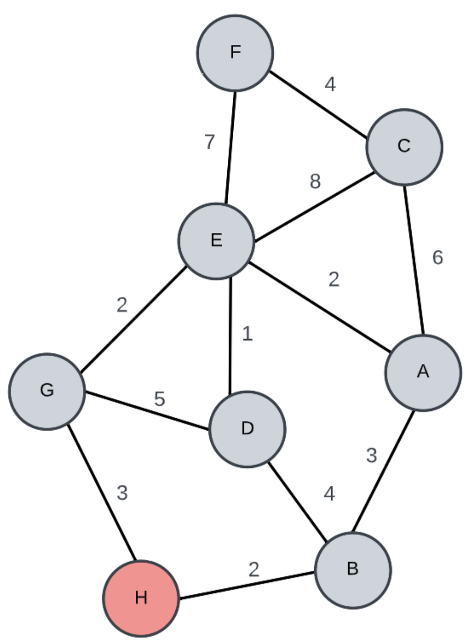
QUEUE
H
B
G
D
A
C
F
E

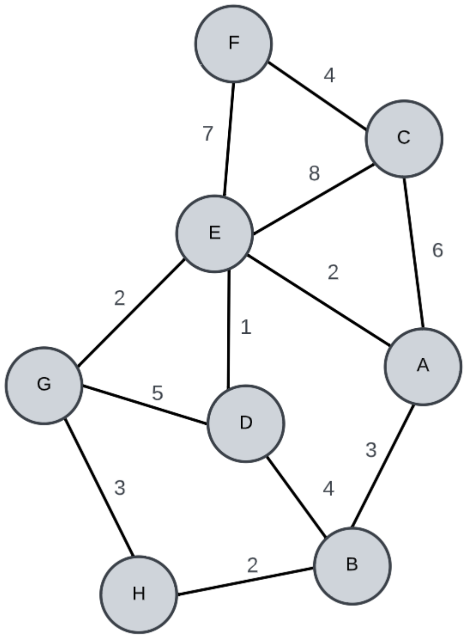
CURRENT ENQUEUED DEQUEUED



QUEUE
H
B
G
D
A
C
F
E

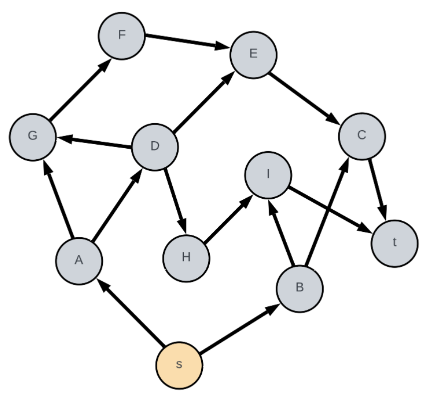
CURRENT ENQUEUED DEQUEUED



<table>
<tr><th>QUEUE</th><th>CURRENT ENQUEUED DEQUEUED</th></tr>
<tr><td>H</td><td rowspan="8">
</td></tr>
<tr><td>B</td></tr>
<tr><td>G</td></tr>
<tr><td>D</td></tr>
<tr><td>A</td></tr>
<tr><td>C</td></tr>
<tr><td>F</td></tr>
<tr><td>E</td></tr>
</table>

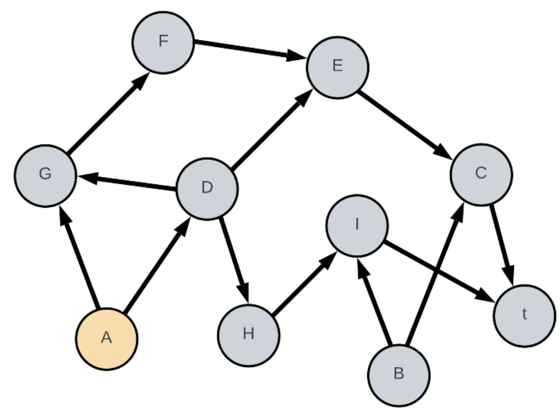
Question 5: Find a topological ordering of the graph in Figure 2.

<table>
<tr><td>Select vertex with in-degree 0, which is
vertex s.
Print it out.
Remove it.
Repeat.

Result = [s]</td><td>
</td></tr>
</table>

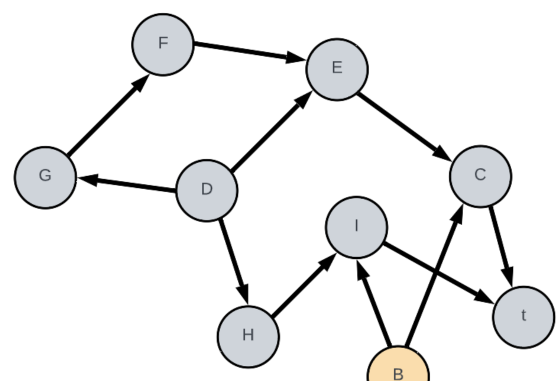
Select vertex with in-degree 0, which is **vertex A**.
Print it out.
Remove it.
Repeat.

Result = [s, A]



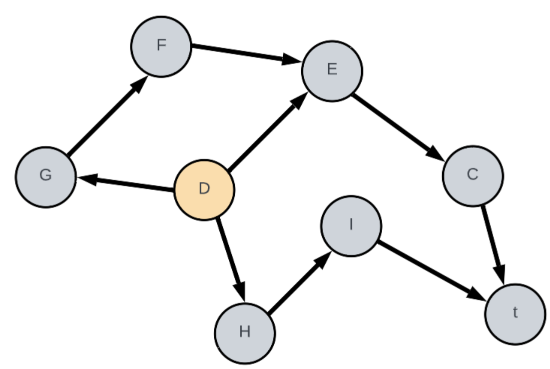
Select vertex with in-degree 0, which is **vertex B**.
Print it out.
Remove it.
Repeat.

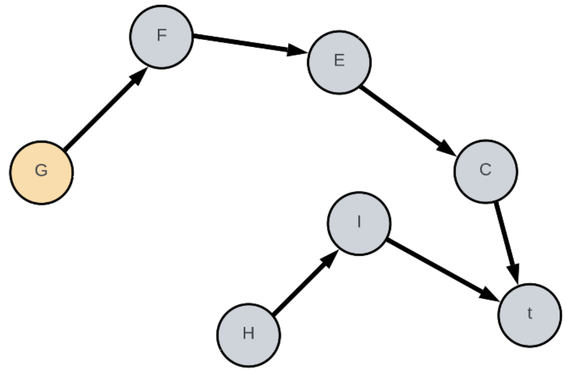
Result = [s, A, B]

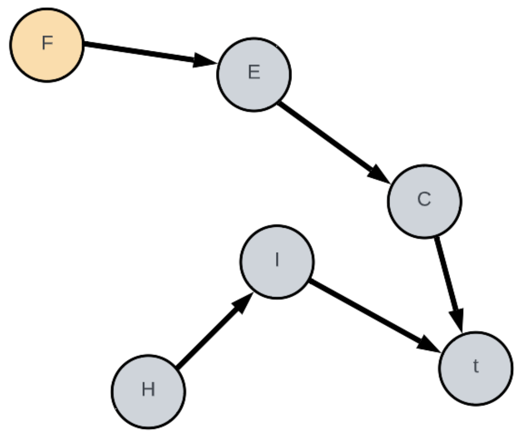


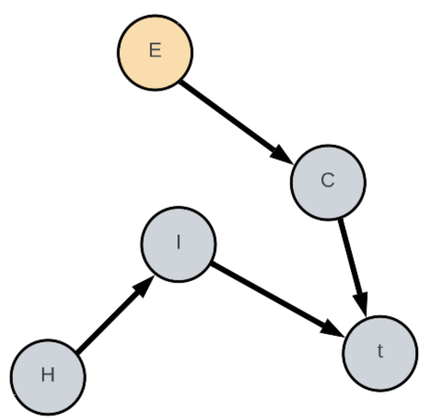
Select vertex with in-degree 0, which is **vertex D**.
Print it out.
Remove it.
Repeat.

Result = [s, A, B, D]



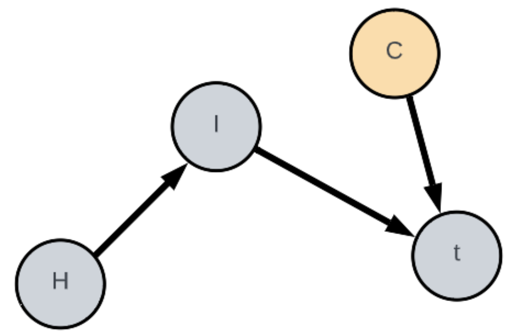
Select vertex with in-degree 0, which is **vertex G**. Print it out. Remove it. Repeat. Result = [s, A, B, D, G]	

Select vertex with in-degree 0, which is **vertex F**. Print it out. Remove it. Repeat. Result = [s, A, B, D, G, F]	

Select vertex with in-degree 0, which is **vertex E**. Print it out. Remove it. Repeat. Result = [s, A, B, D, G, F, E]	

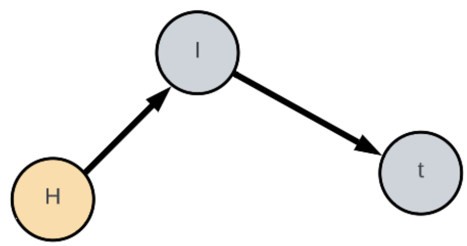
Select vertex with in-degree 0, which is **vertex C**.
Print it out.
Remove it.
Repeat.

Result = [s, A, B, D, G, F, E, C]



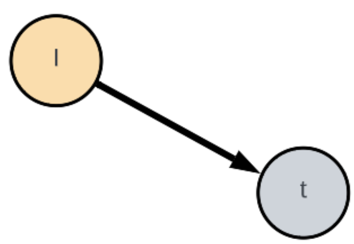
Select vertex with in-degree 0, which is **vertex H**.
Print it out.
Remove it.
Repeat.

Result = [s, A, B, D, G, F, E, C, H]



Select vertex with in-degree 0, which is **vertex I**.
Print it out.
Remove it.
Repeat.

Result = [s, A, B, D, G, F, E, C, H, I]



Select vertex with in-degree 0, which is **vertex t**. Print it out. Remove it. Repeat. Result = [s, A, B, D, G, F, E, C, H, I, t]