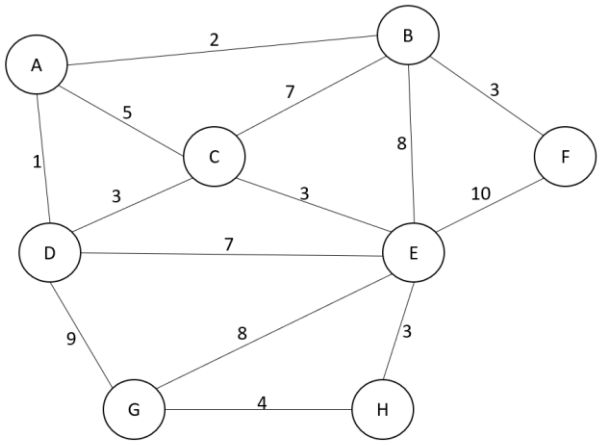
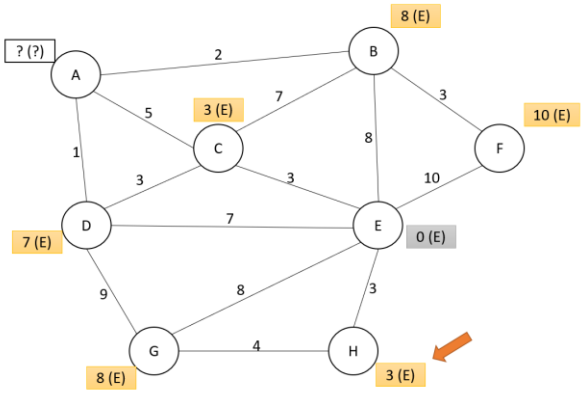
CS300 – HOMEWORK 5

QUESTION 1:

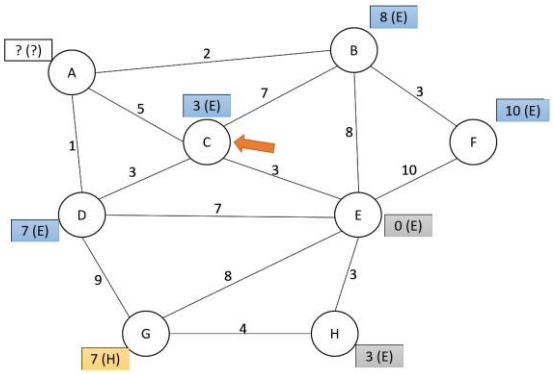
Trace the operation of Dijkstra's weighted shortest path algorithm for the following graph. Use vertex E as your start vertex.



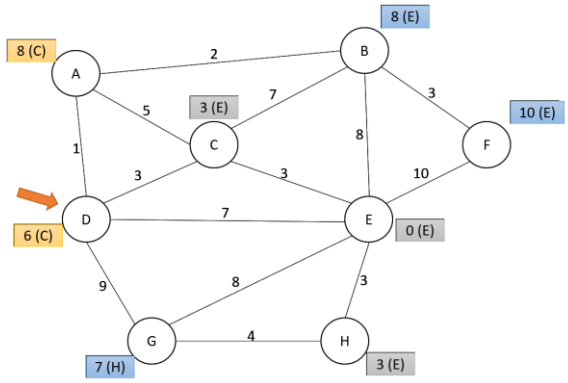
Iteration 1- Select vertex E to start and update distances of its adjacent vertices.



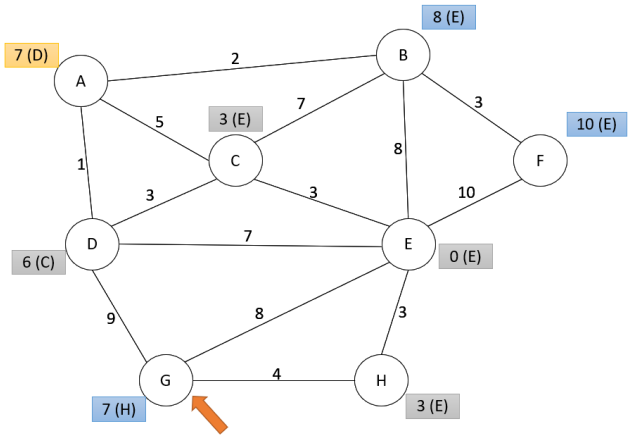
Iteration 2- Among the unknown vertices, select the one with the smallest distance. In this case, choose one of C or H, I chose H. Make necessary updates for the distances of the unknown vertices adjacent to H.



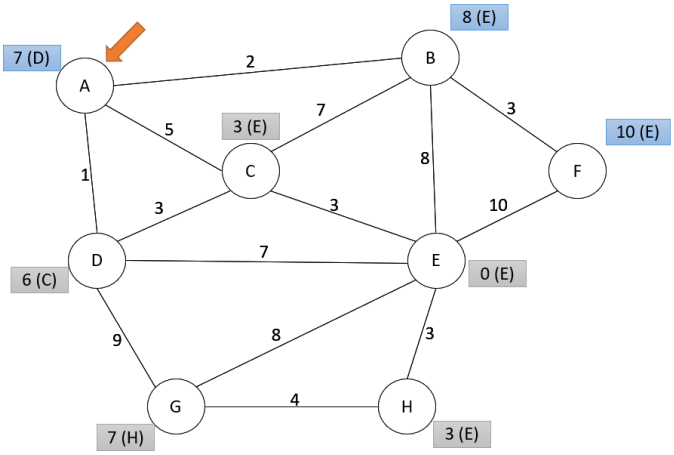
Iteration 3- Among the unknown vertices, select the one with the smallest distance. In this case, choose C. Make necessary updates for the distances of the unknown vertices adjacent to C.



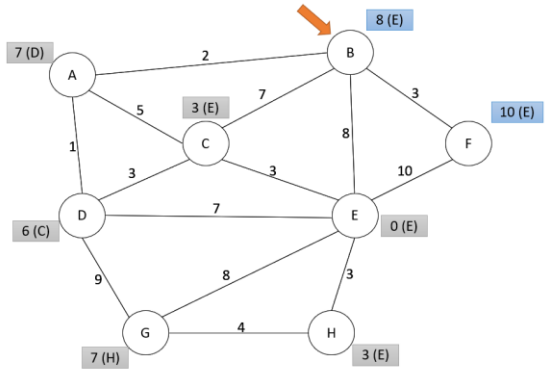
Iteration 4- Among the unknown vertices, select the one with the smallest distance. In this case, choose D. Make necessary updates for the distances of the unknown vertices adjacent to D.



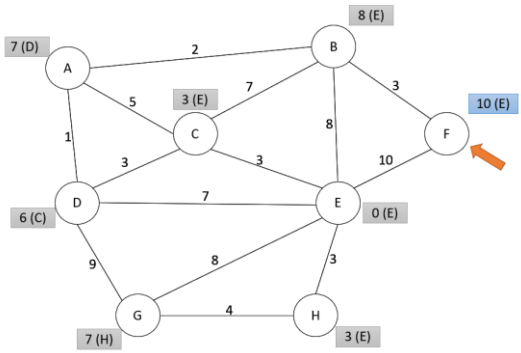
Iteration 5- Among the unknown vertices, select the one with the smallest distance. In this case, choose G. Make necessary updates for the distances of the unknown vertices adjacent to G. There is no need to update any vertex.



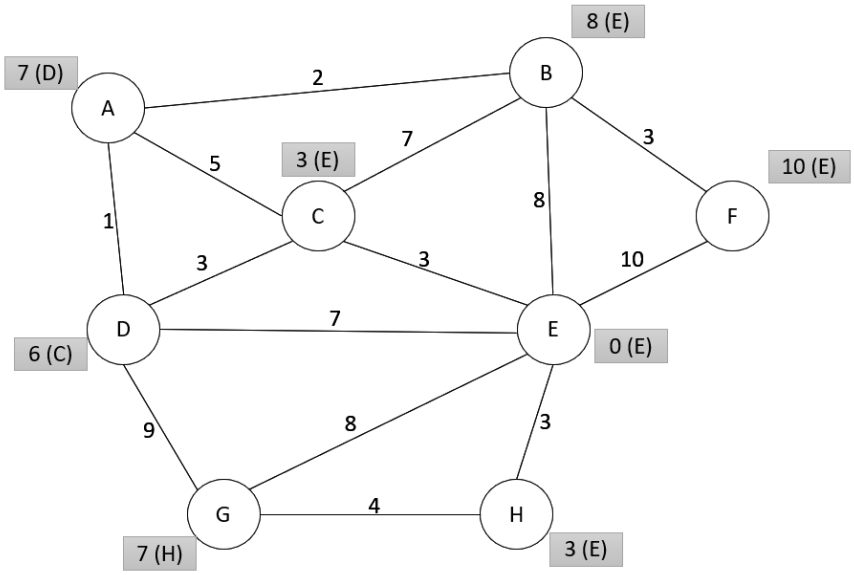
Iteration 6- Among the unknown vertices, select the one with the smallest distance. In this case, choose one of A or B, I chose B. Make necessary updates for the distances of the unknown vertices adjacent to B.



Iteration 7- Among the unknown vertices, select the one with the smallest distance. In this case, choose B. Make necessary updates for the distances of the unknown vertices adjacent to B. There is no need to update any vertex.

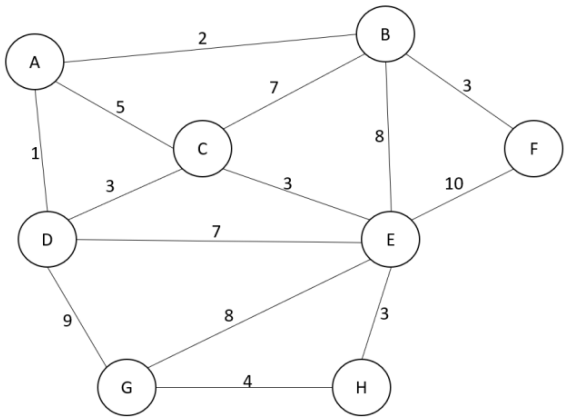


Iteration 7- Among the unknown vertices, select the one with the smallest distance. In this case, choose F. There is no other unknown vertex left. Algorithm completed.

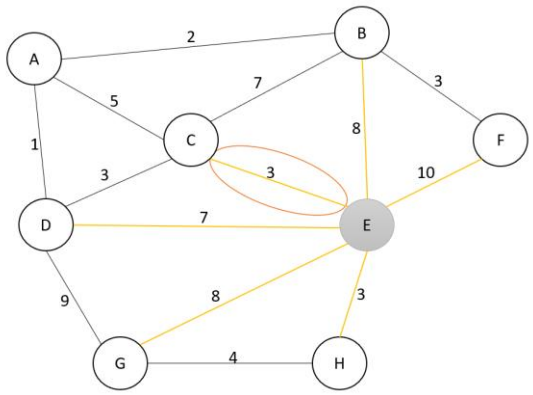


QUESTION 2:

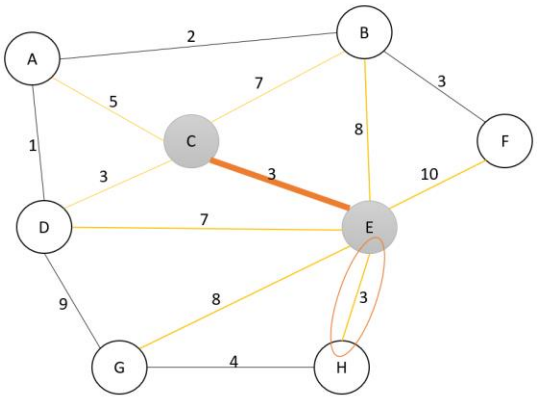
Trace the operation of Prim's minimum spanning tree algorithm for the following graph. Use vertex E as your start vertex.



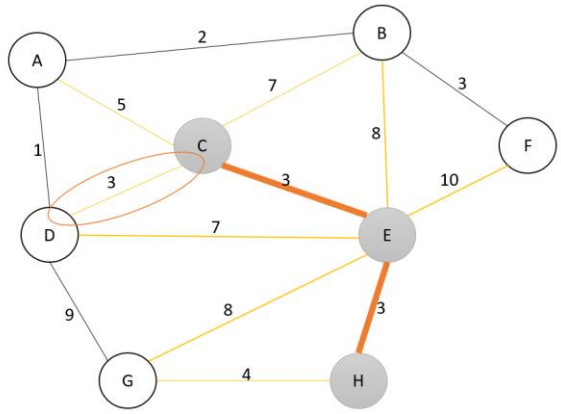
Iteration 1 – Start by marking vertex E as known. Then pick the edge with the lowest weight that connects a known vertex (a vertex in the tree) and an unknown vertex (a vertex that is not in the tree). In this case, pick (E,C) and add it to the tree. Mark C as known.



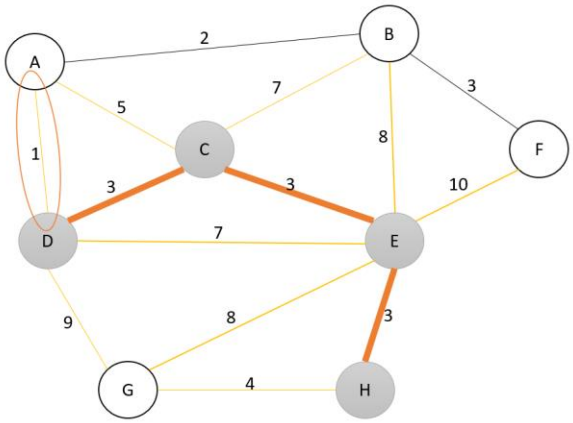
Iteration 2 - Pick the edge with the lowest weight that connects a known vertex (a vertex in the tree) and an unknown vertex (a vertex that is not in the tree). In this case, pick (E,H) and add it to the tree. Mark H as known.



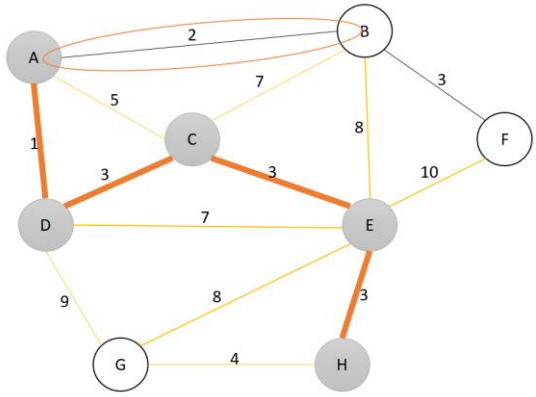
Iteration 3 - Pick the edge with the lowest weight that connects a known vertex (a vertex in the tree) and an unknown vertex (a vertex that is not in the tree). In this case, pick (C,D) and add it to the tree. Mark D as known.



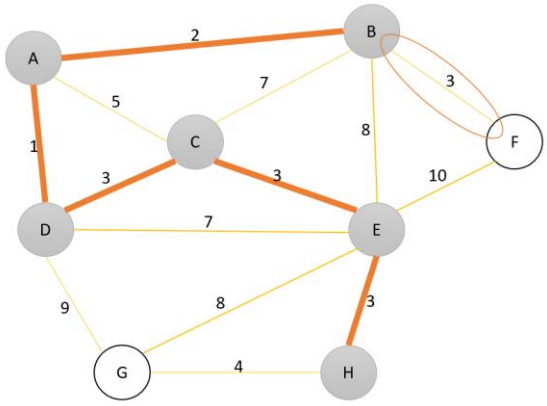
Iteration 4 - Pick the edge with the lowest weight that connects a known vertex (a vertex in the tree) and an unknown vertex (a vertex that is not in the tree). In this case, pick (D,A) and add it to the tree. Mark A as known.



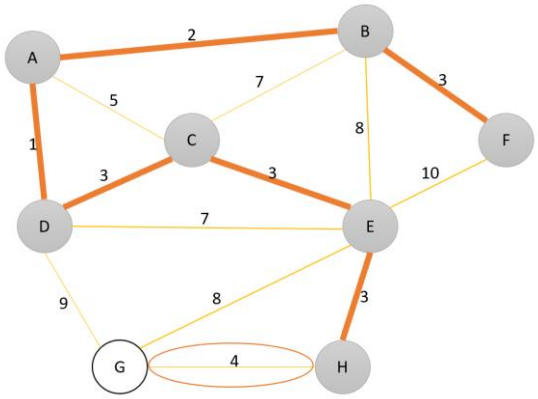
Iteration 5 - Pick the edge with the lowest weight that connects a known vertex (a vertex in the tree) and an unknown vertex (a vertex that is not in the tree). In this case, pick (A,B) and add it to the tree. Mark B as known.



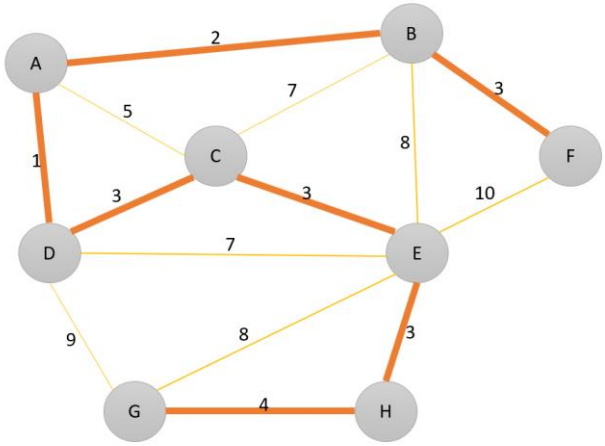
Iteration 6 - Pick the edge with the lowest weight that connects a known vertex (a vertex in the tree) and an unknown vertex (a vertex that is not in the tree). In this case, pick (B,F) and add it to the tree. Mark F as known.



Iteration 7 - Pick the edge with the lowest weight that connects a known vertex (a vertex in the tree) and an unknown vertex (a vertex that is not in the tree). In this case, pick (H,G) and add it to the tree. Mark G as known.

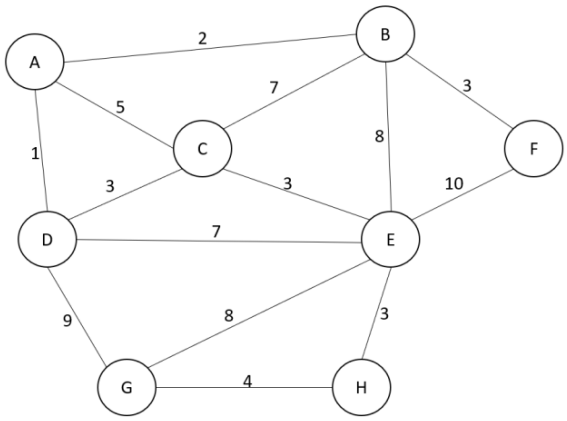


Since all vertices are included In the spanning tree, algorithm completed. Minimum spanning tree is as follows.



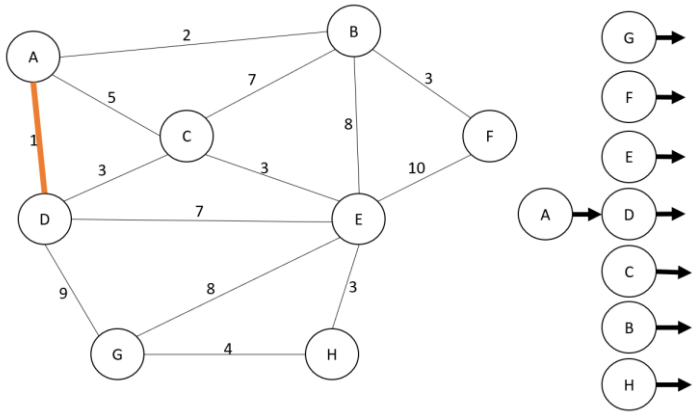
QUESTION 3:

Trace the operation of Kruskal's minimum spanning tree algorithm for the following graph.

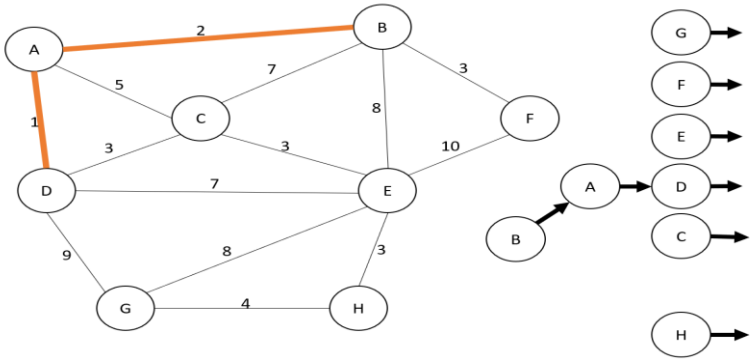


At the start of the operations, the graph is represented as a forest with every vertex representing a tree.

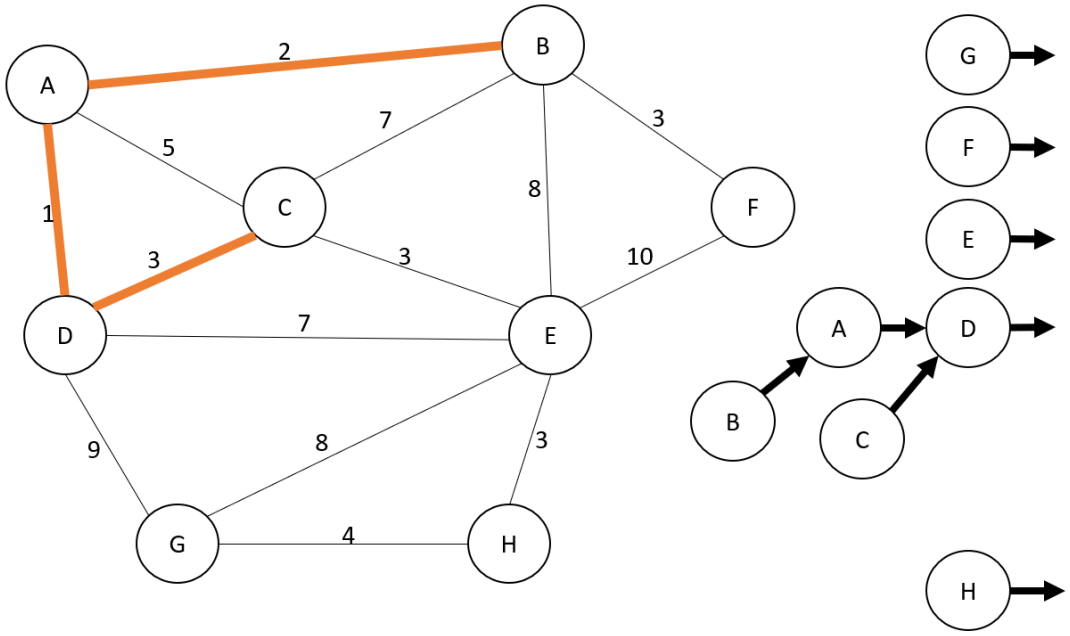
Iteration 1- Pick the edge with smallest weight, which is edge (A,D) with weight 1. Using union operation on them and including the edge in the minimum spanning tree does not cause a cycle. Thus add the edge to minimum spanning tree.



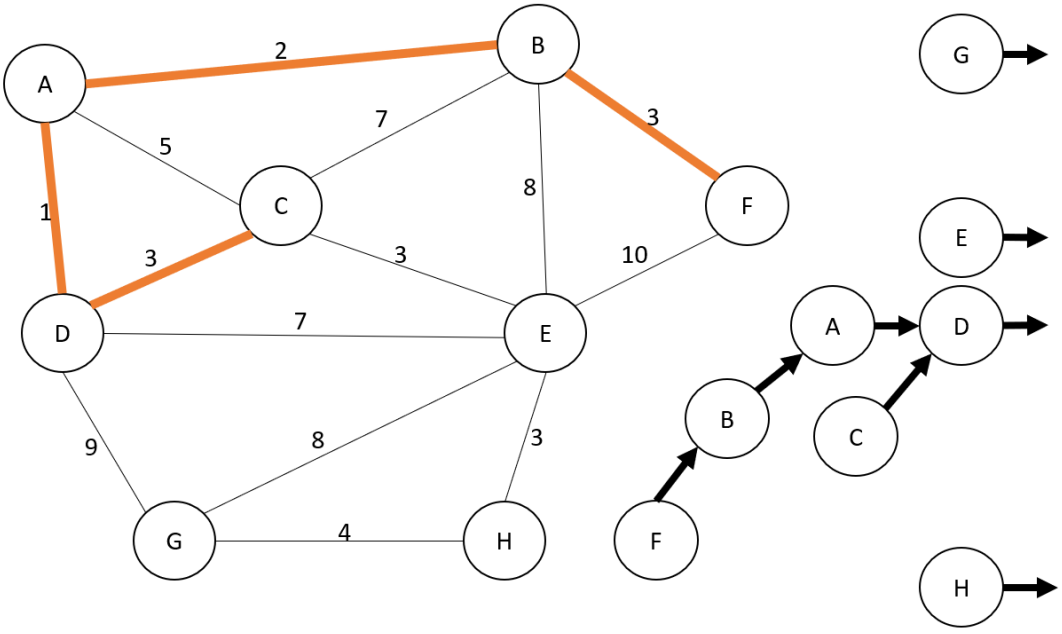
Iteration 2 – Pick the edge with the smallest weight, which is edge (A,B) with weight 2. Using union operation on them and including the edge in the minimum spanning tree does not cause a cycle. Thus add the edge to minimum spanning tree.



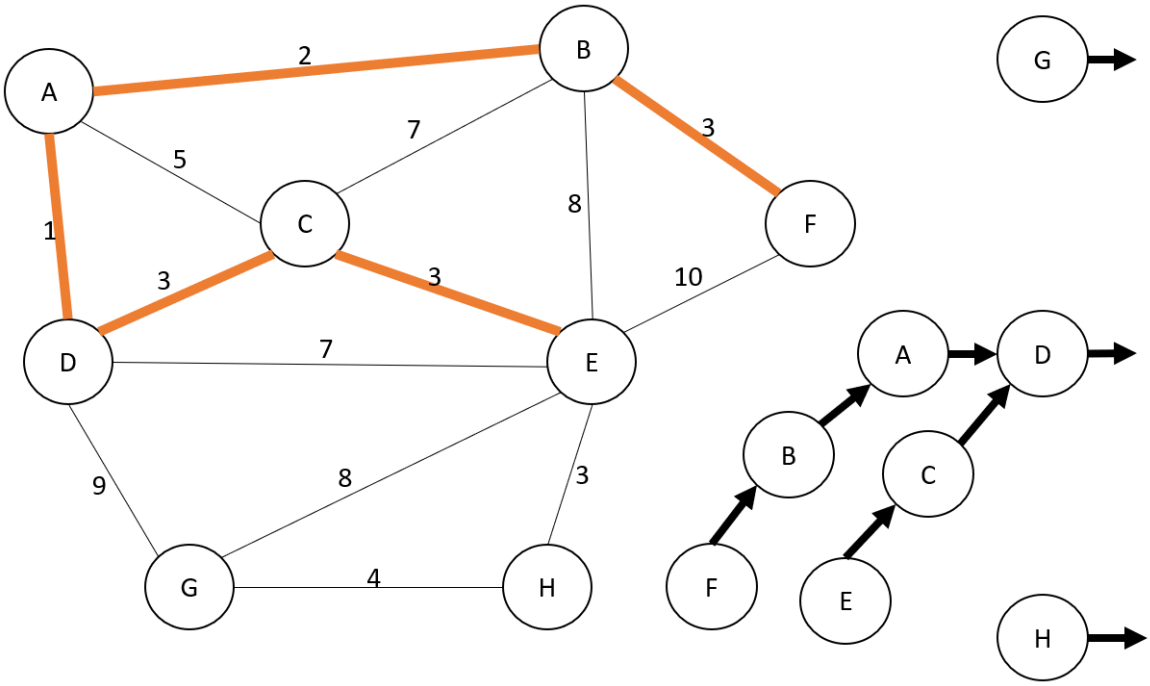
Iteration 3 – Pick the edge with the minimum weight, which is one of (D,C), (B,F), (E,C) or (E,H) with weights 3. First pick edge (D,C). Using union operation on them and including the edge in the minimum spanning tree does not cause a cycle. Thus add the edge to minimum spanning tree.



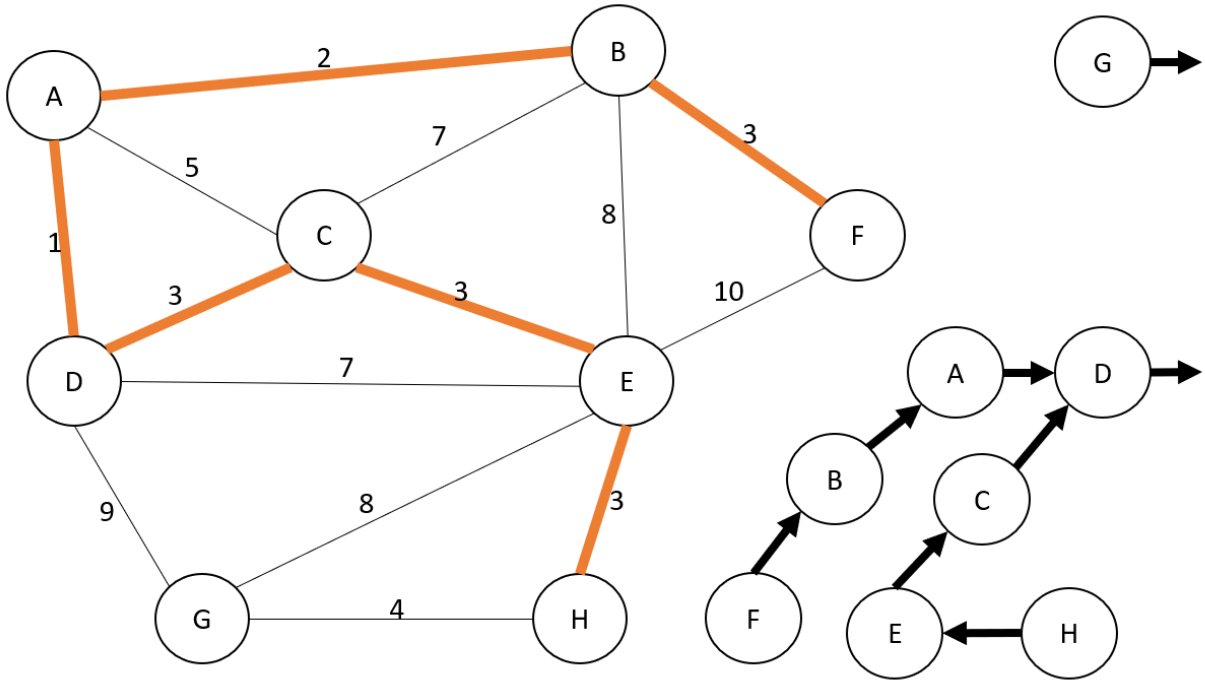
Iteration 4 – Pick (B,F) with weight 3. Using union operation on them and including the edge in the minimum spanning tree does not cause a cycle. Thus add the edge to minimum spanning tree.



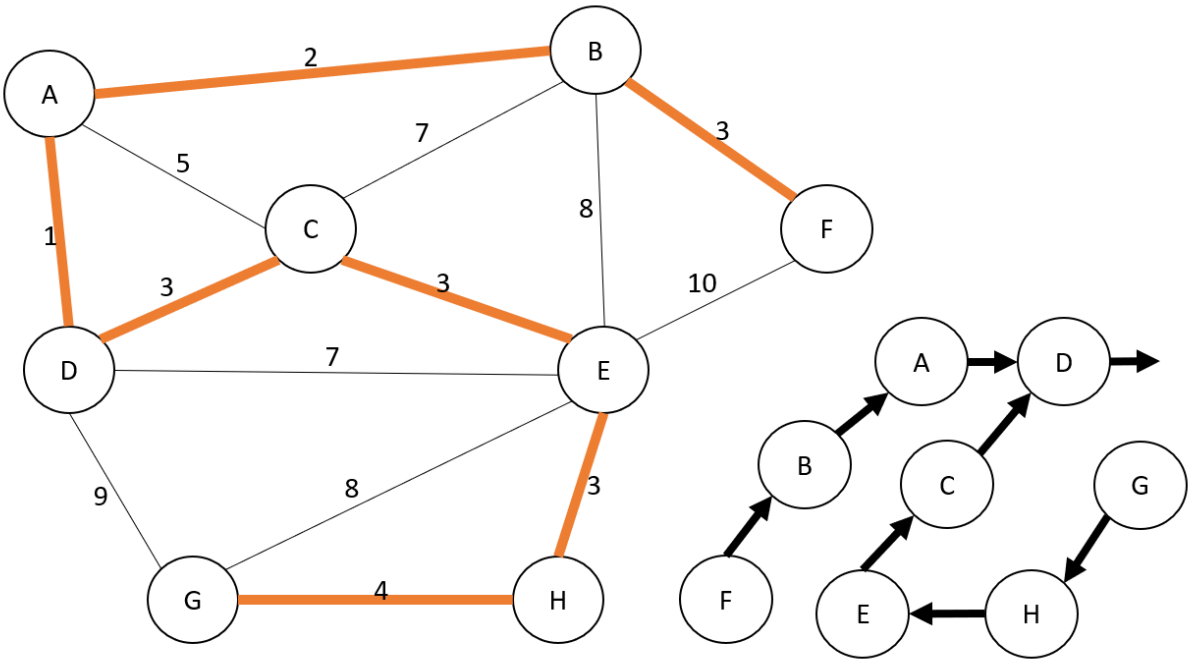
Iteration 5 – Pick (C,E) with weight 3. Using union operation on them and including the edge in the minimum spanning tree does not cause a cycle. Thus add the edge to minimum spanning tree.



Iteration 6 – Pick (E,H) with weight 3. Using union operation on them and including the edge in the minimum spanning tree does not cause a cycle. Thus add the edge to minimum spanning tree.



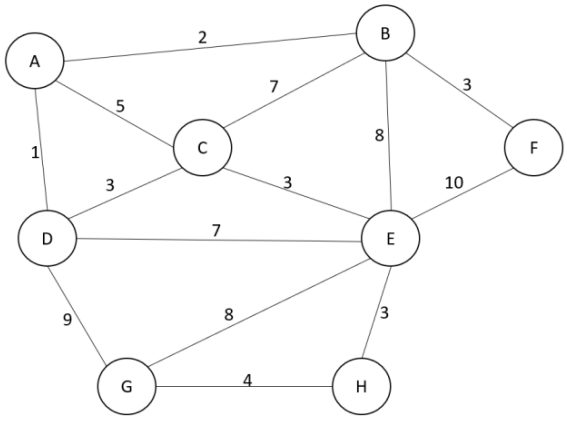
Iteration 7- Pick the edge with minimum weight, which is (H,G) with weight 4. Using union operation on them and including the edge in the minimum spanning tree does not cause a cycle. Thus add the edge to minimum spanning tree.



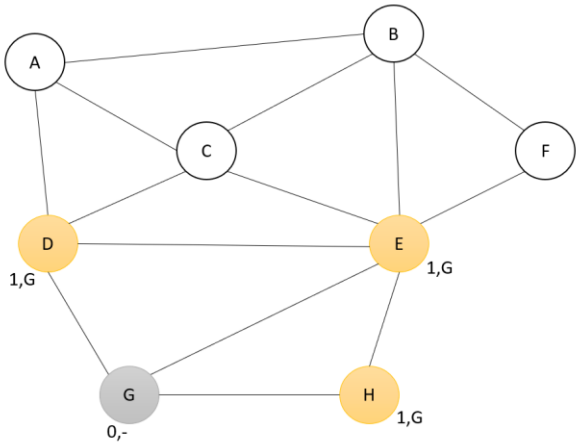
Since there is only one tree remaining and minimum spanning tree encompasses all the vertices, operation is completed.

QUESTION 4:

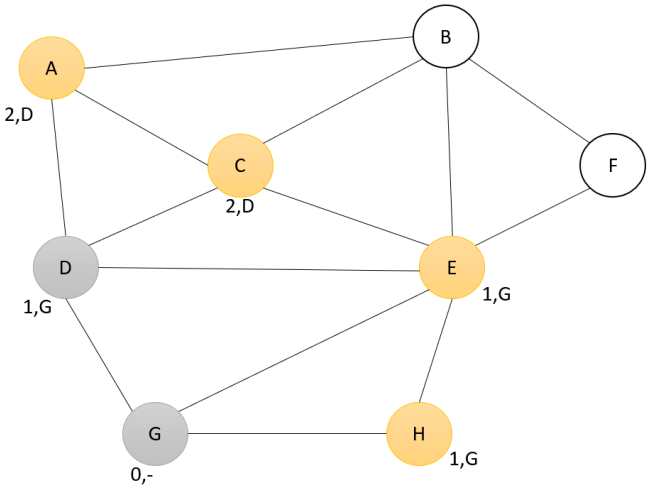
Find shortest unweighted path from G to all other vertices for the following graph. Use breadth-first search algorithm in your answer. Do NOT forget to show the trace.



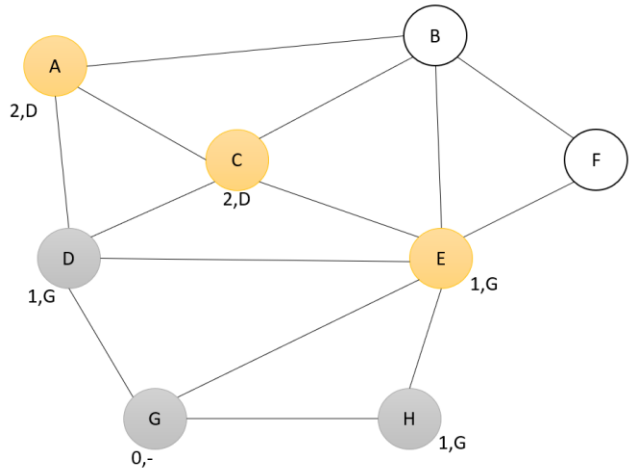
Iteration 1- Mark G as visited and set all unknown vertices adjacent to its distances as 1.



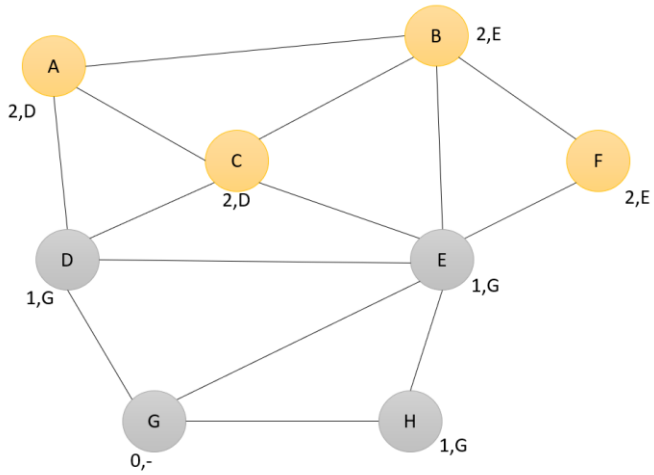
Iteration 2 – Mark D as visited and set all unknown vertices adjacent to its distances as 2.



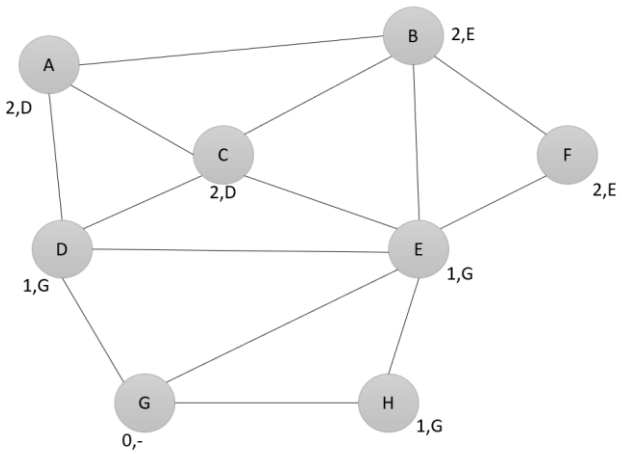
Iteration 3 - Mark H as visited.



Iteration 4 - Mark E as visited and set all unknown vertices adjacent to its distances as 2.

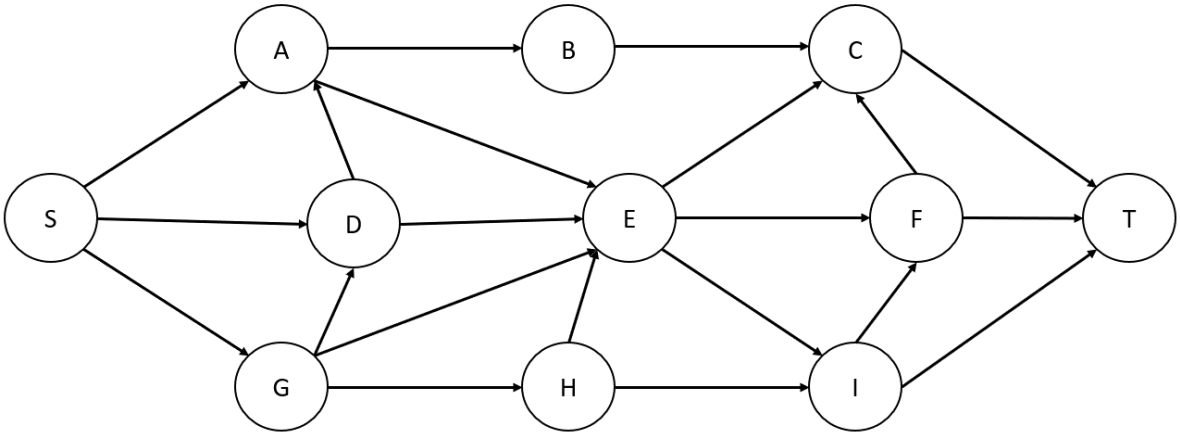


Iteration 5- since there are no unknown vertices left, mark all vertices as visited. All vertices have their shortest distance from G and path written next to them.

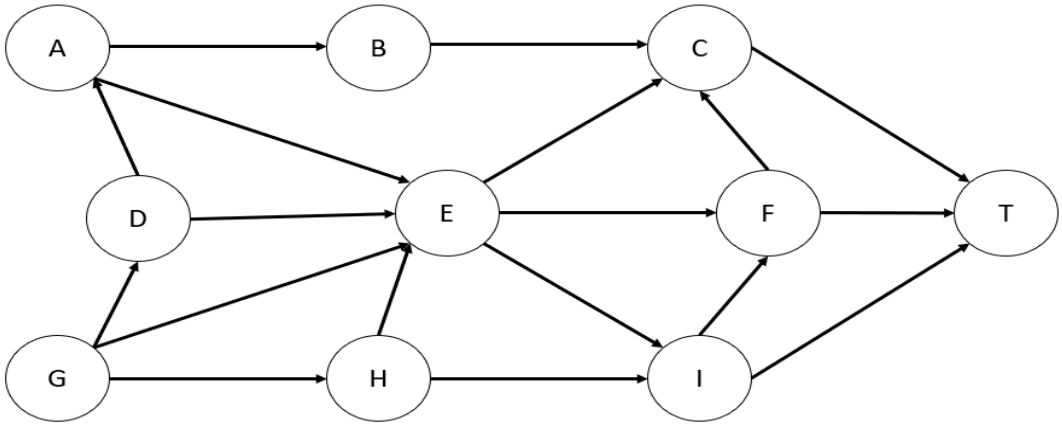


QUESTION 5:

Find a topological ordering of the following graph.

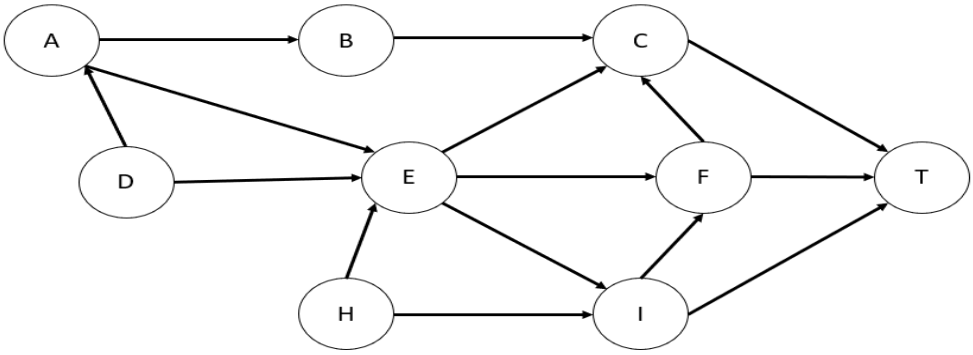


Iteration 1- Select a vertex with indegree 0, in this case, select S. Print it and remove it from the graph.



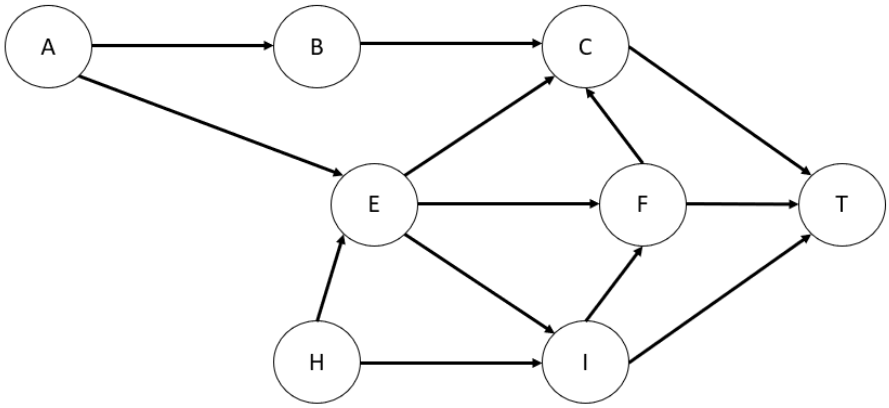
- S,

Iteration 2 – Select a vertex with indegree 0, in this case , select G. Print it and remove it from the graph.



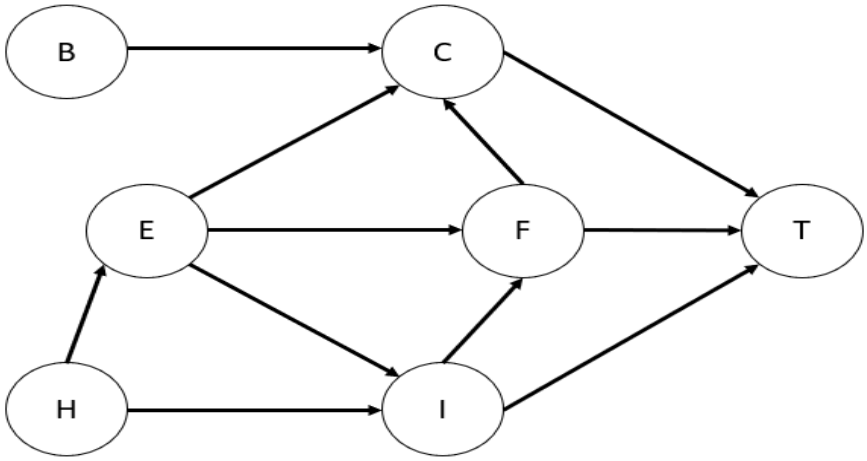
- S, G,

Iteration 3 – Select D, print it, and remove it from the graph.



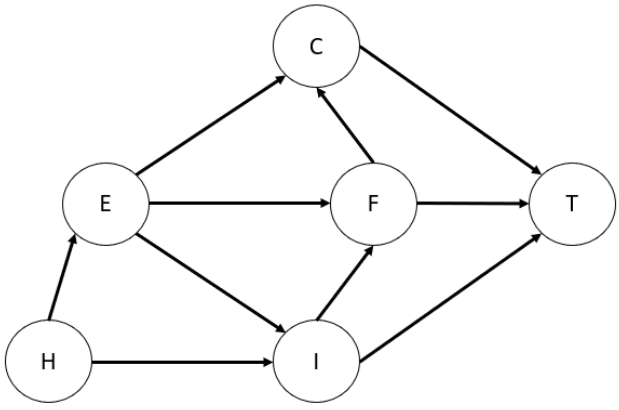
- S, G, D

Iteration 4 – Select A, print it, and remove it from the graph.



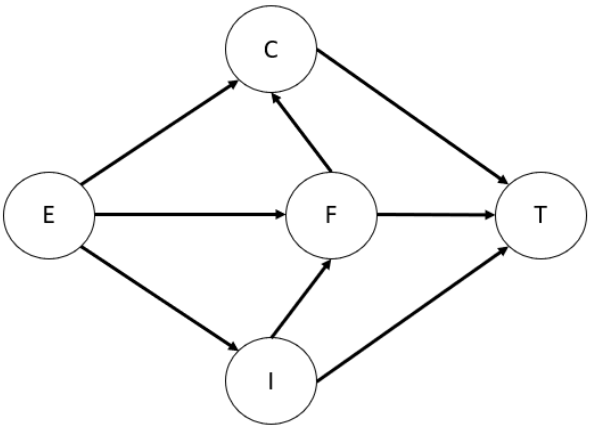
- S, G, D, A

Iteration 5 – Select B, print it, and remove it from the graph.



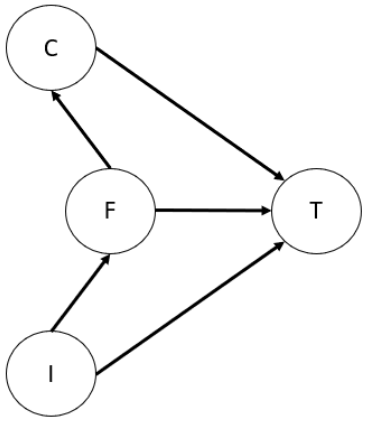
- S, G, D, A, B

Iteration 6 – Select H, print it, and remove it from the graph.



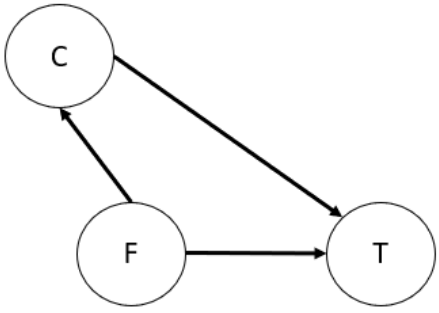
- S, G, D, A, B, H

Iteration 7- Select E, print it, and remove it from the graph.



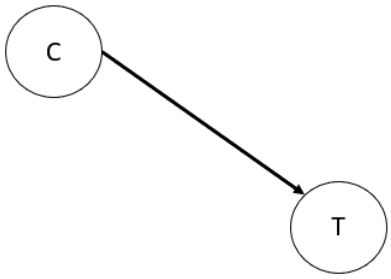
- S, G, D, A, B, H, E

Iteration 8- Select I, print it, and remove it from the graph.



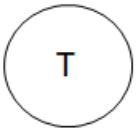
- S, G, D, A, B, H, E, I

Iteration 9- Select F, print it, and remove it from the graph.



- S, G, D, A, B, H, E, I, F

Iteration 10- Select C, print it, and remove it from the graph.



- S, G, D, A, B, H, E, I, F, C

Iteration 11 – Print T and remove it from the graph.

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

Is a valid topological sort of the graph.