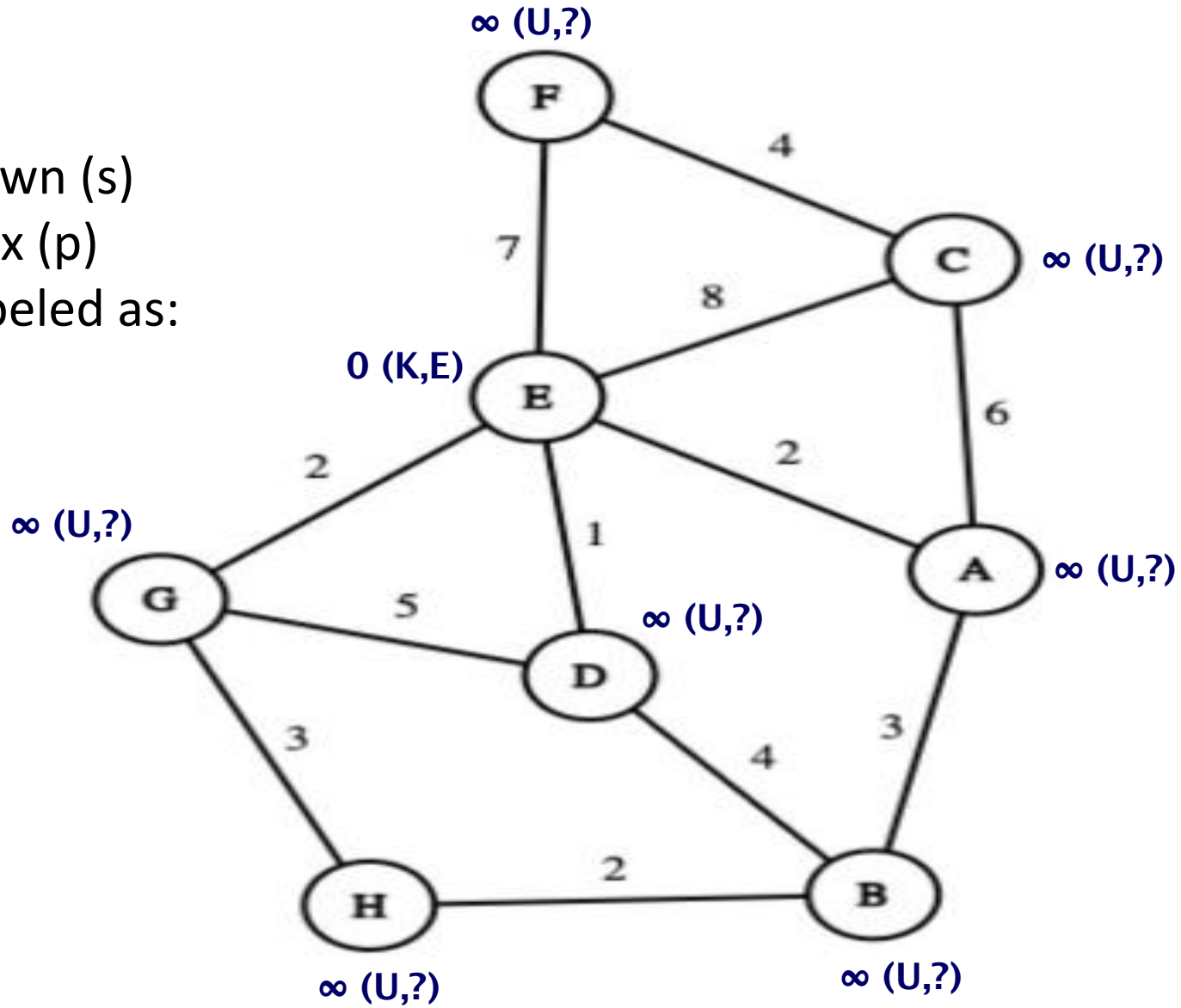
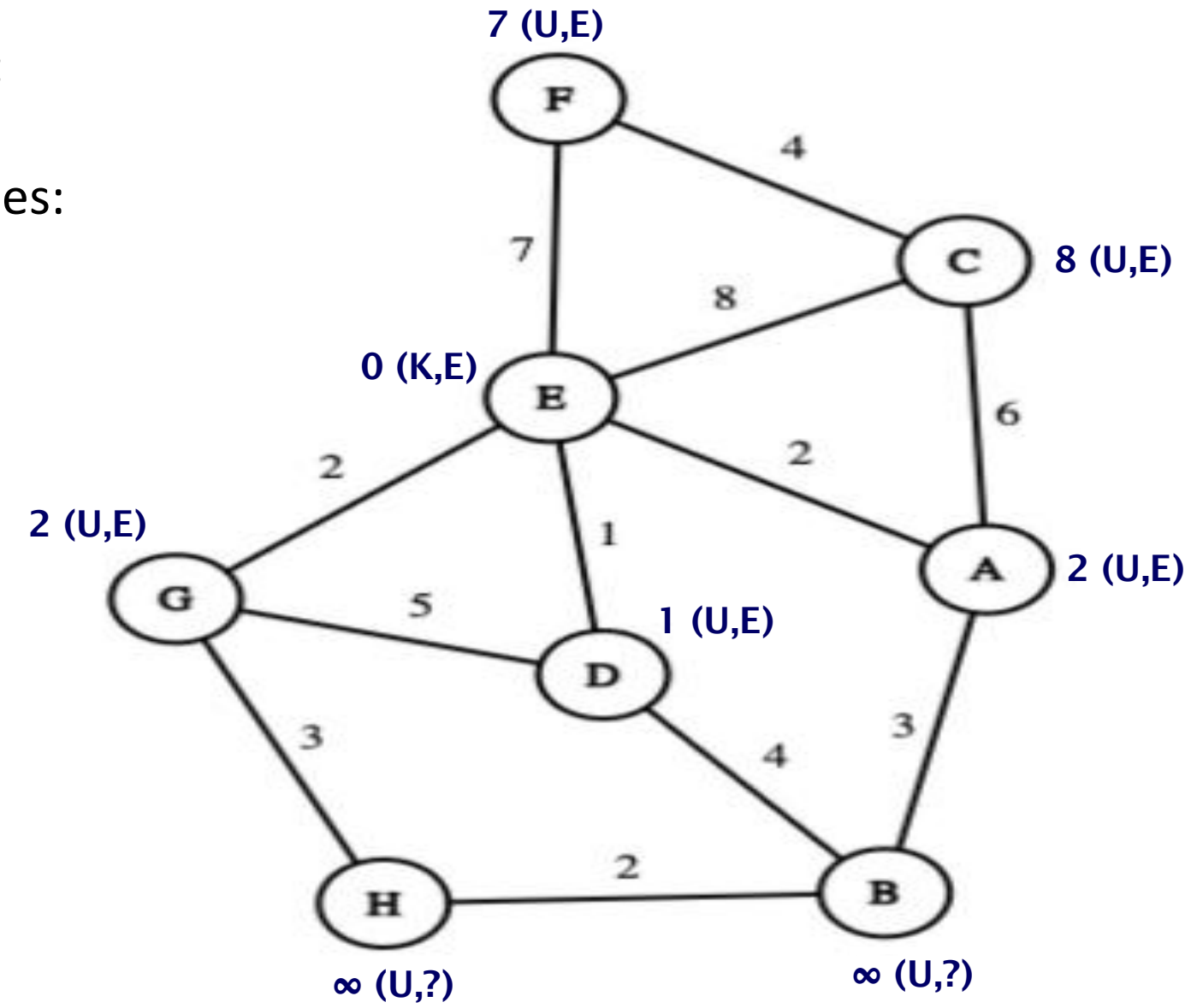
Question 1

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

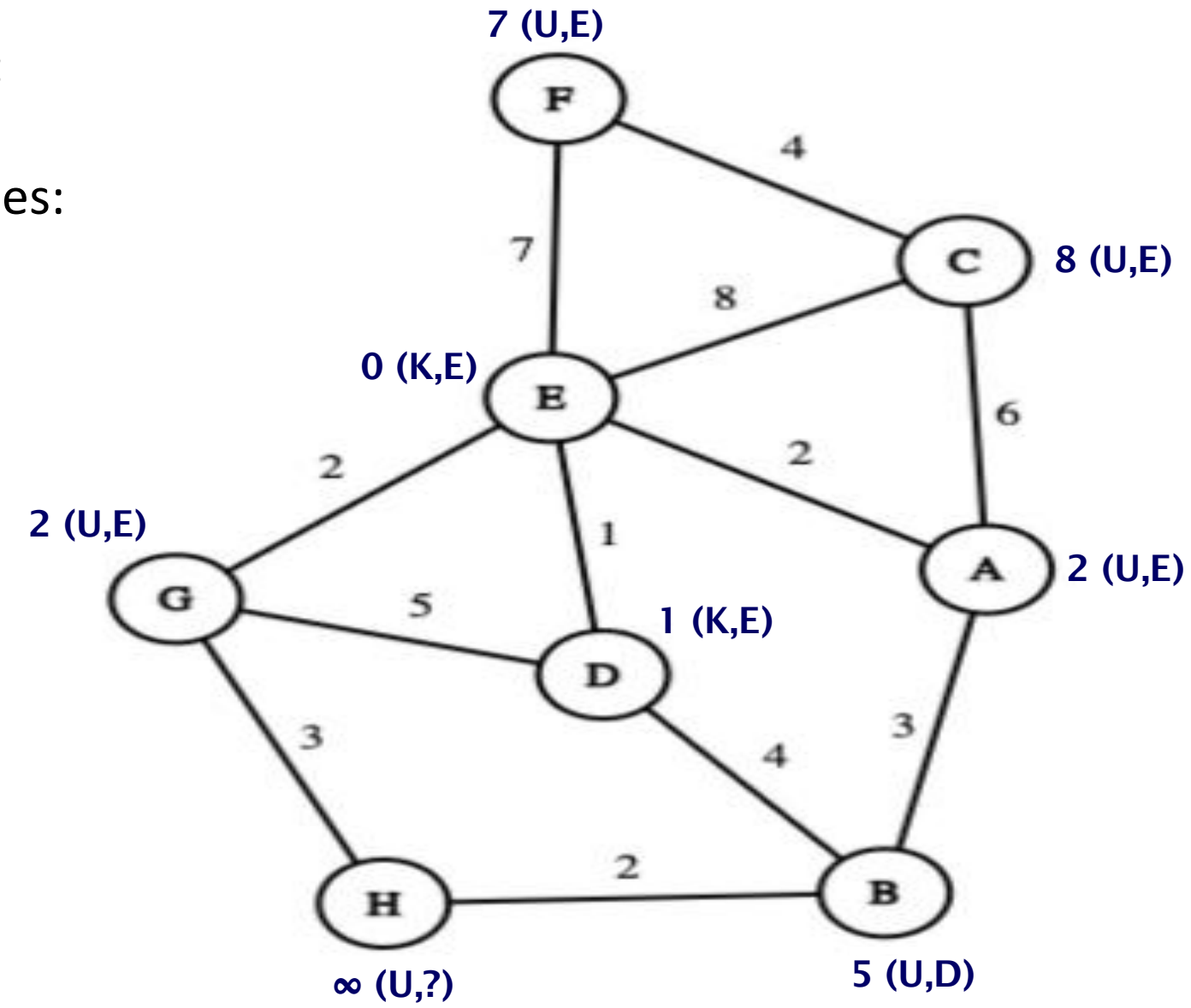
Parameters:
1 - Distance (d)
2 - Known/Unknown (s)
3 - Previous Vertex (p)
Every vertex is labeled as:
d (s,p)



Known Vertices:
- E
Unknown Vertices:
- A,B,C,D,F,G,H

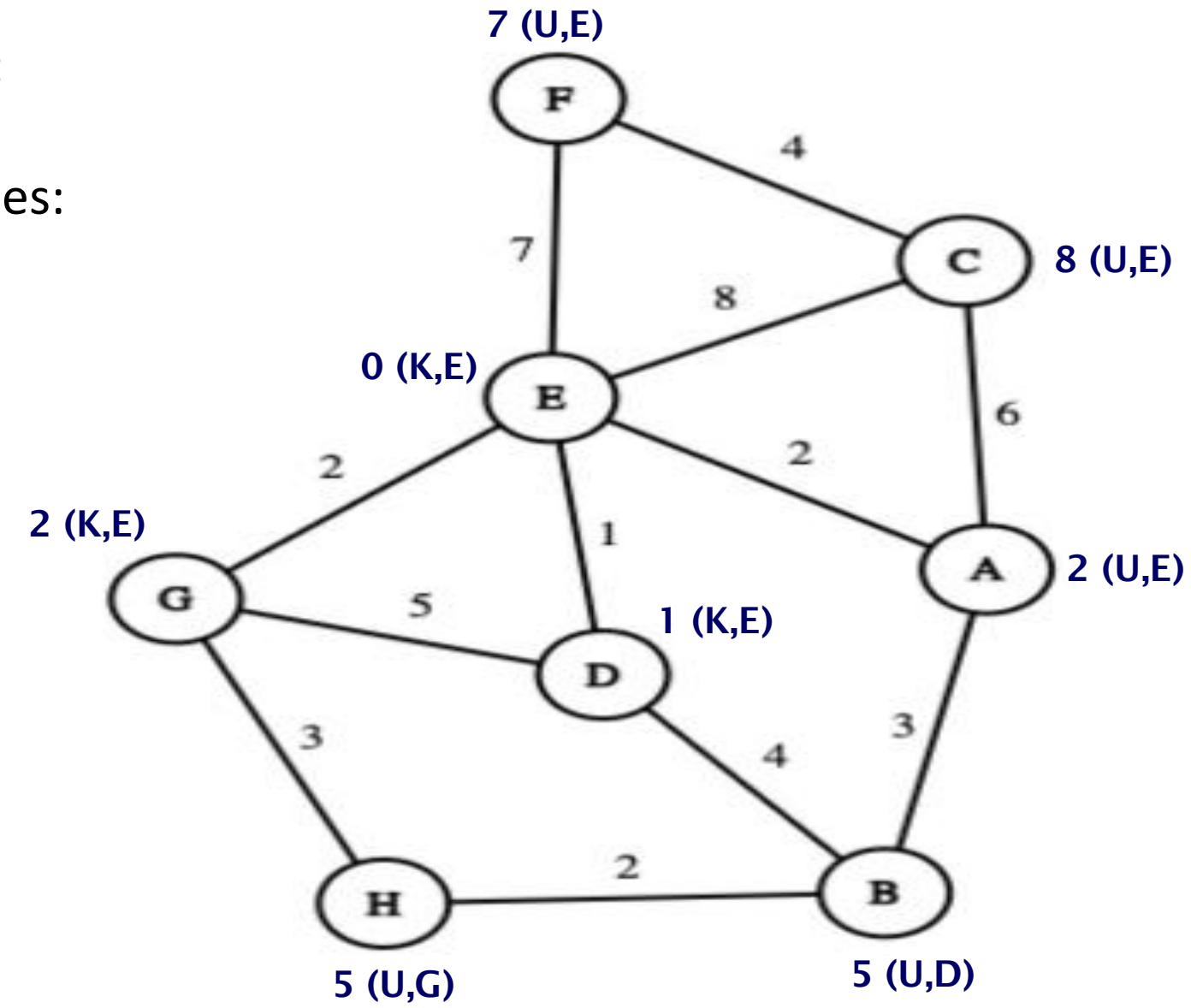


Known Vertices:
- E,D
Unknown Vertices:
- A,B,C,F,G,H



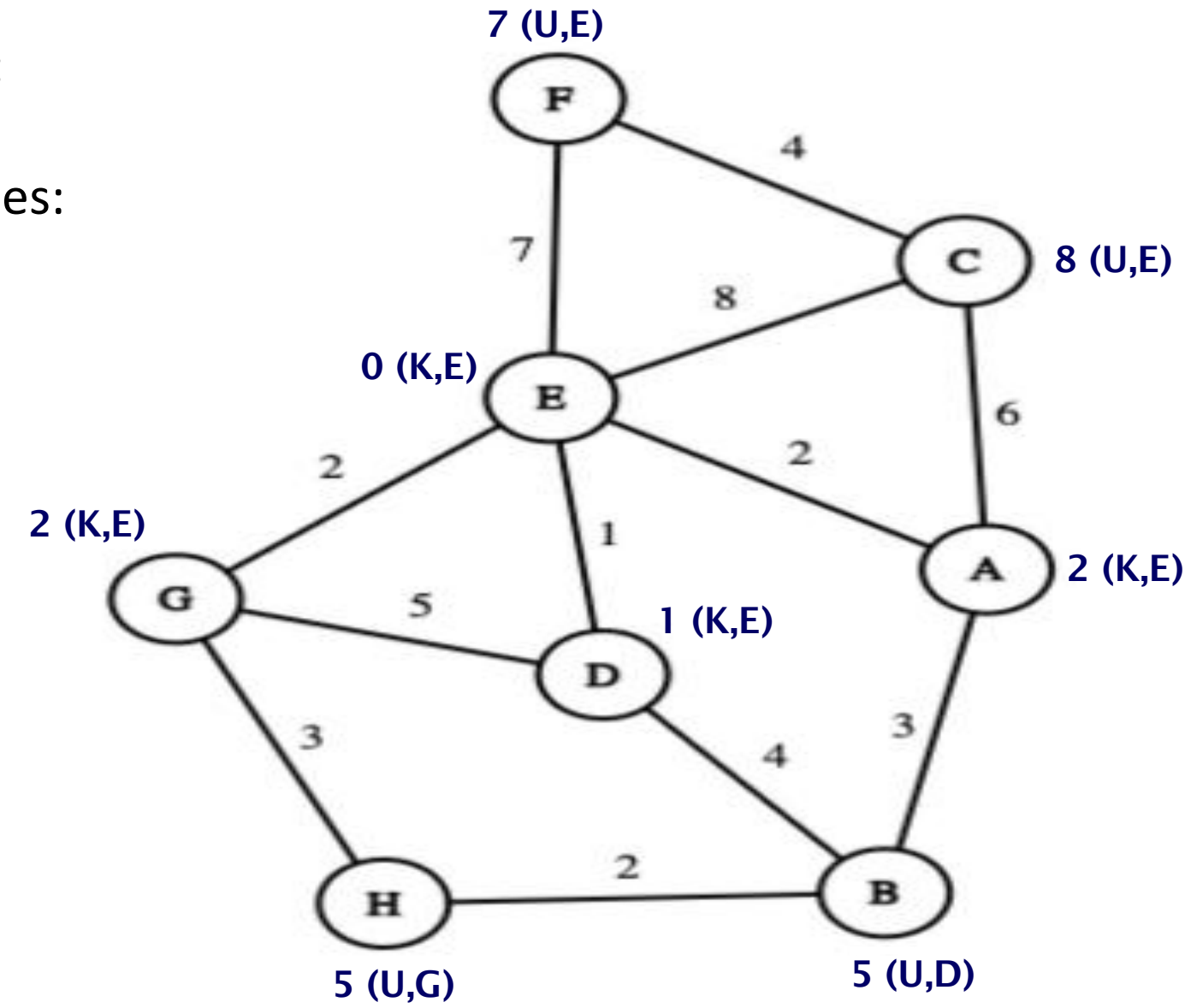
Known Vertices:
- E,D,G

Unknown Vertices:
- A,B,C,F,H



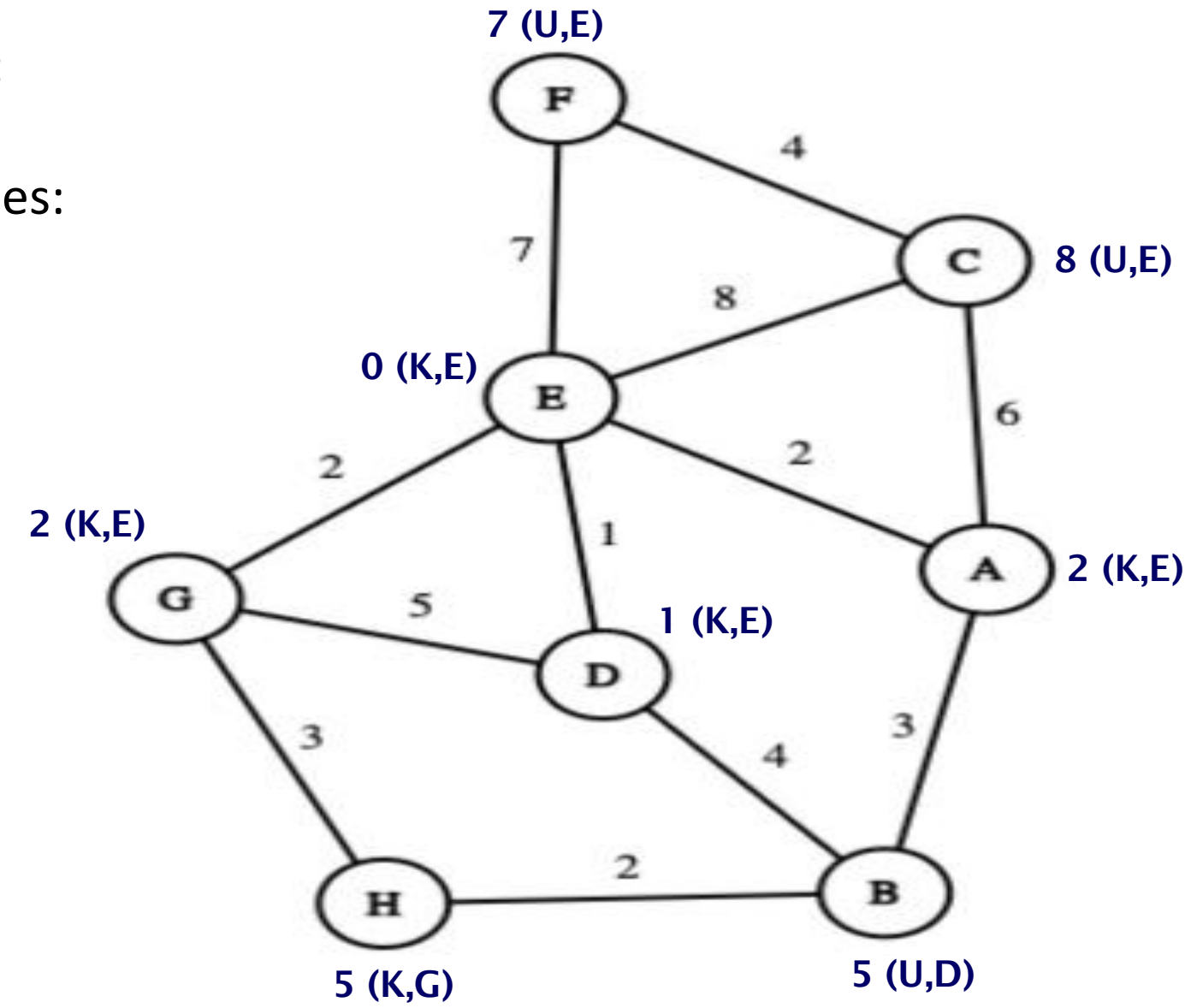
Known Vertices:
- E,D,G,A

Unknown Vertices:
- B,C,F,H

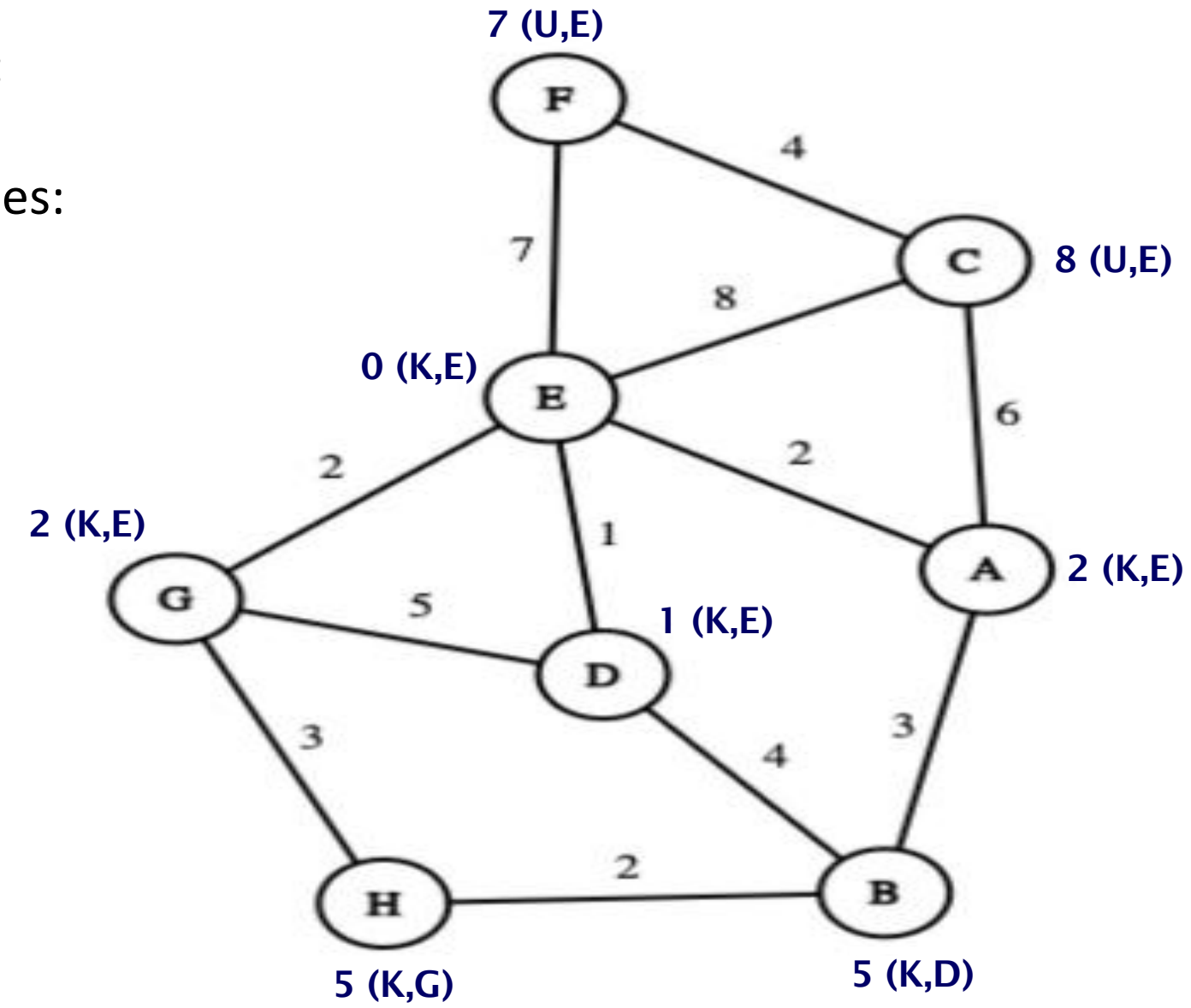


Known Vertices:
- E,D,G,A,H

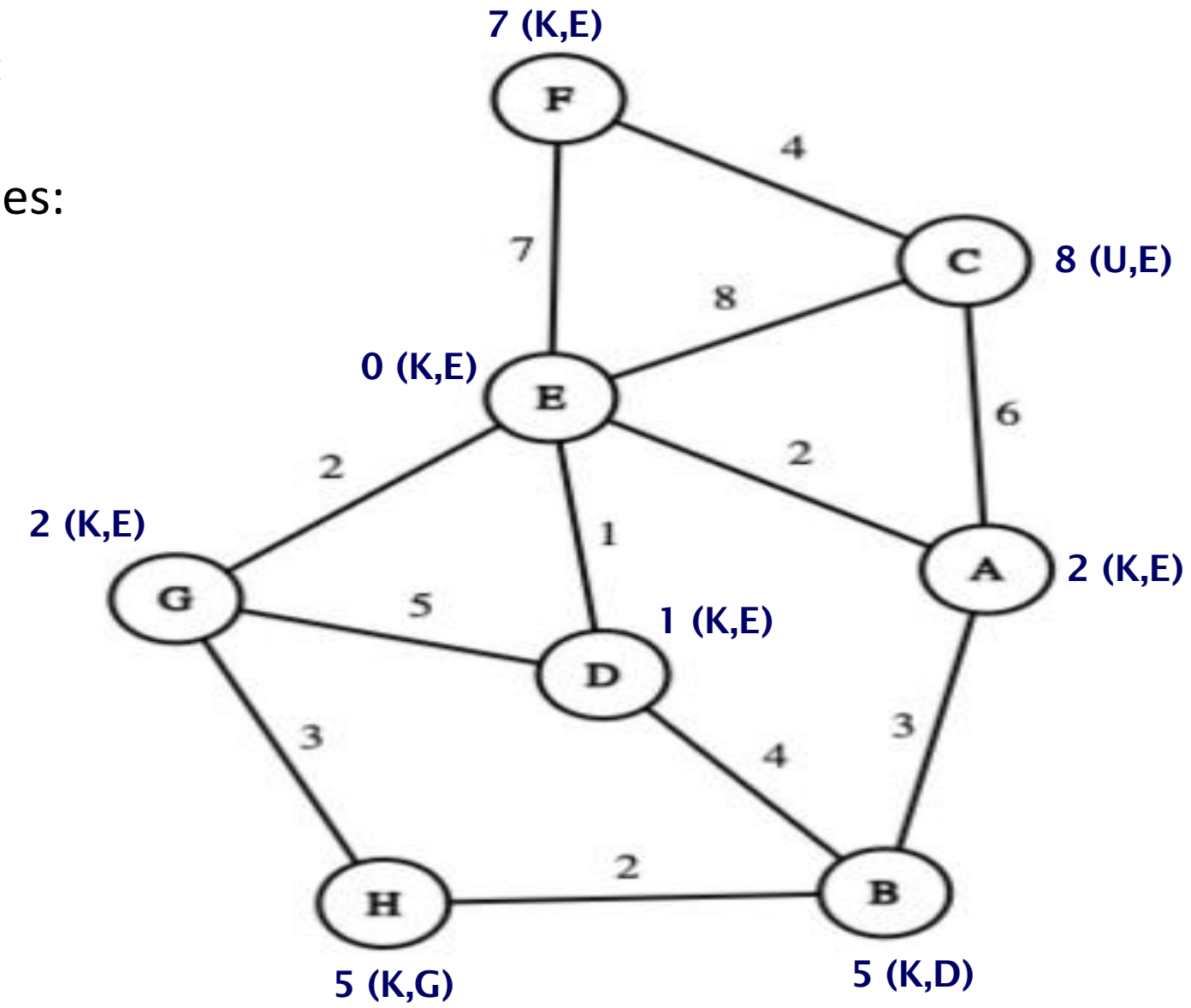
Unknown Vertices:
- B,C,F



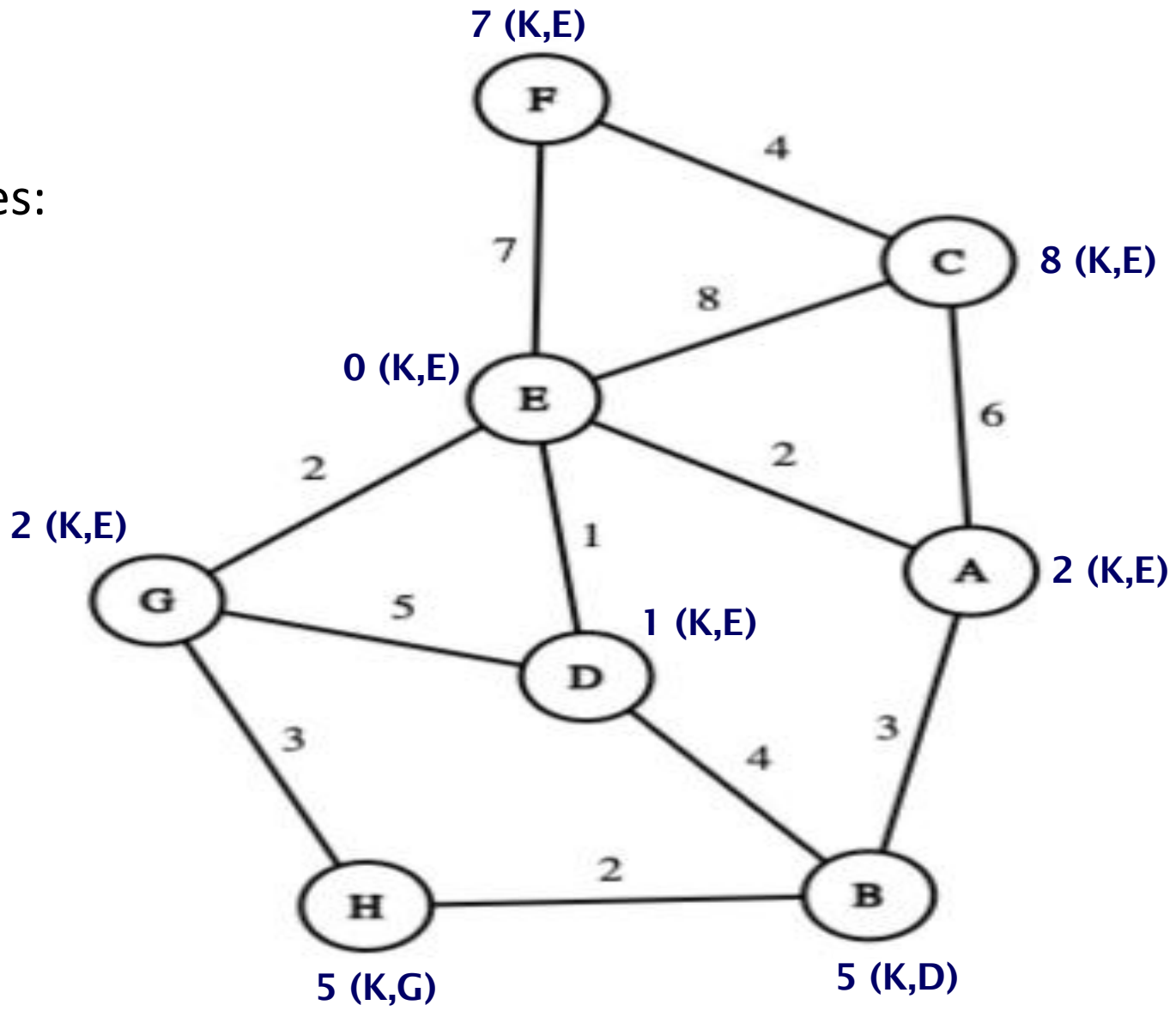
Known Vertices:
- E,D,G,A,H,B
Unknown Vertices:
- C,F



Known Vertices:
- E,D,G,A,H,B,F
Unknown Vertices:
- C



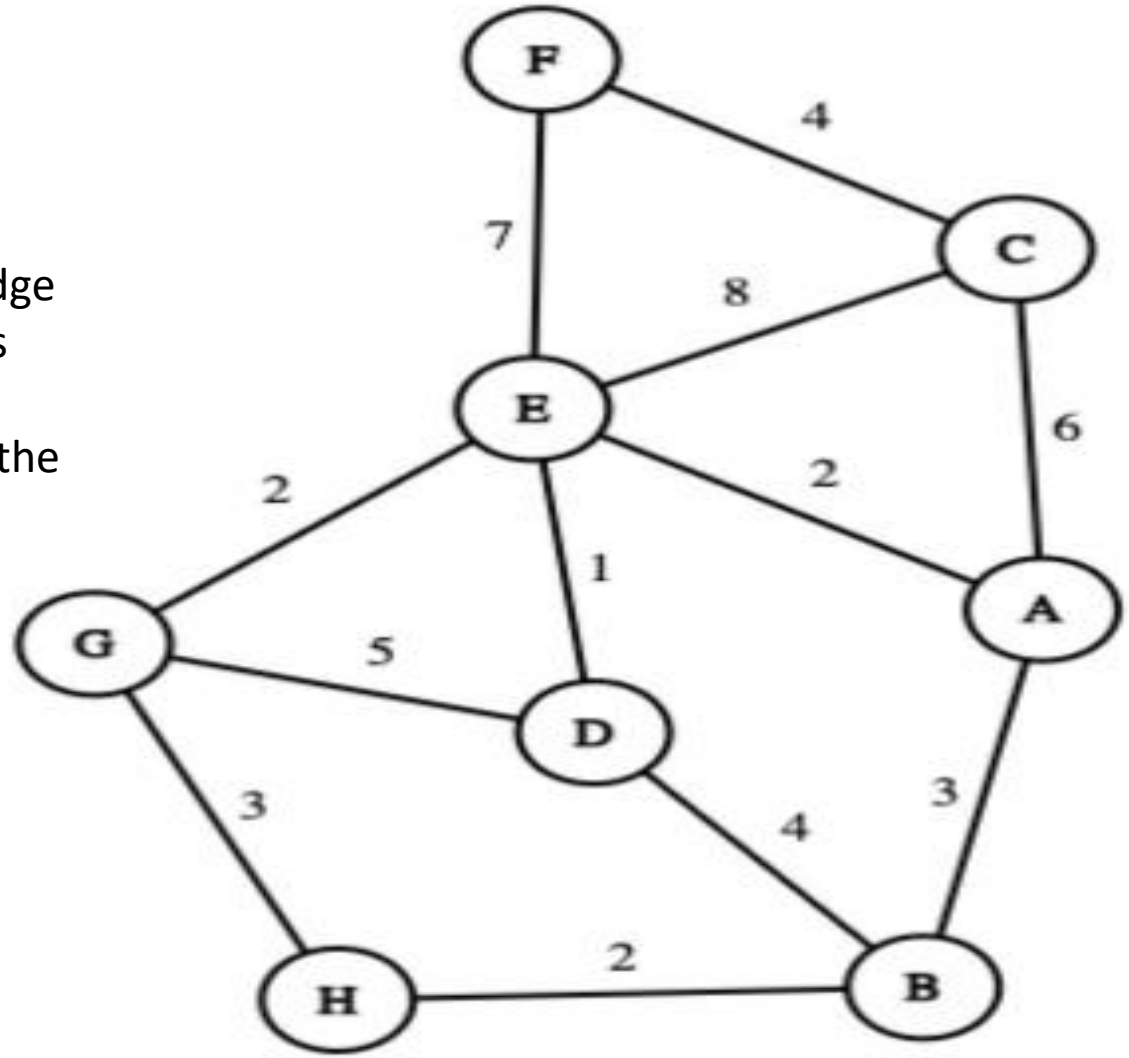
Known Vertices:
- E,D,G,A,H,B,F,C
Unknown Vertices:
- None



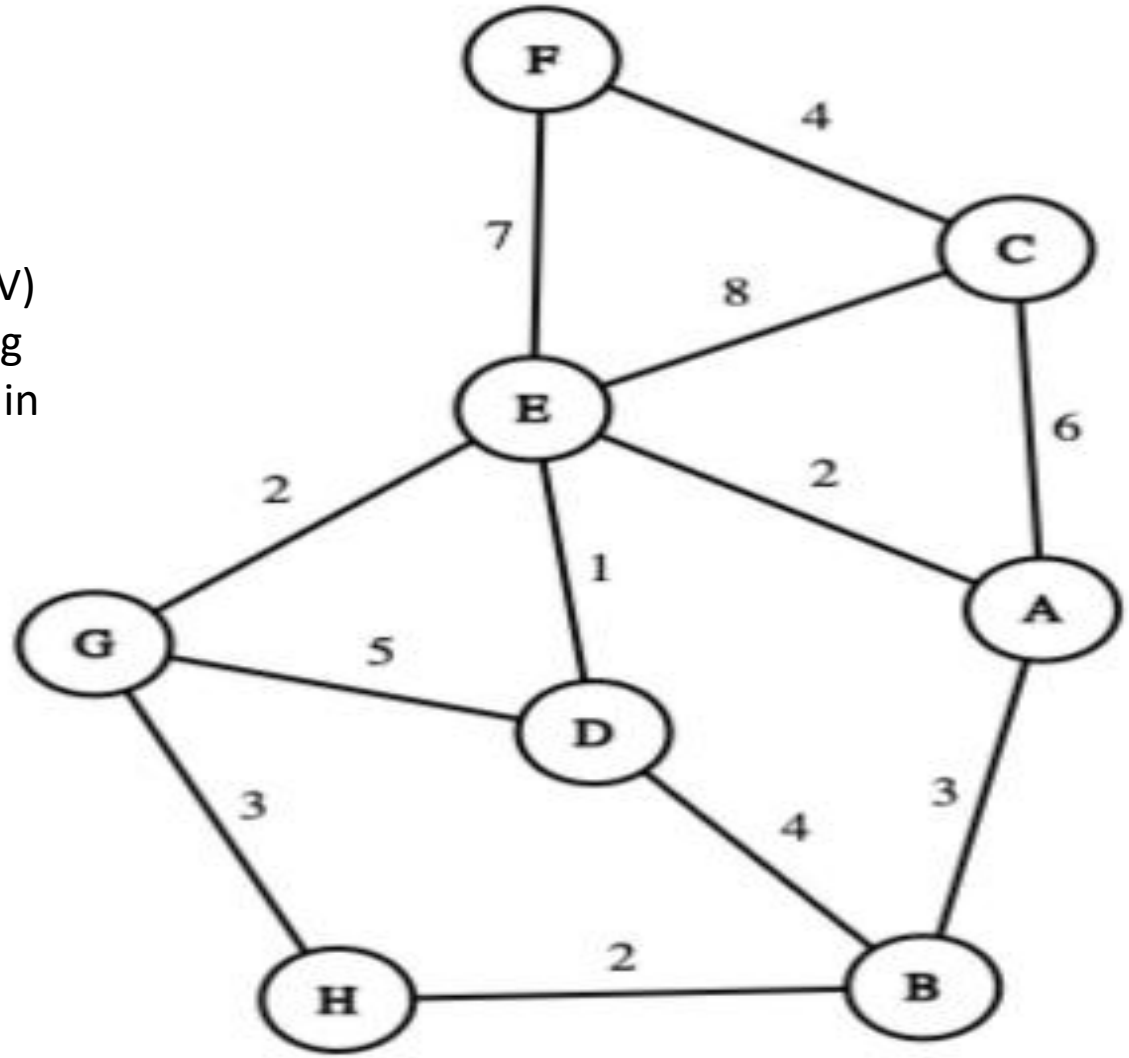
Question 2

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

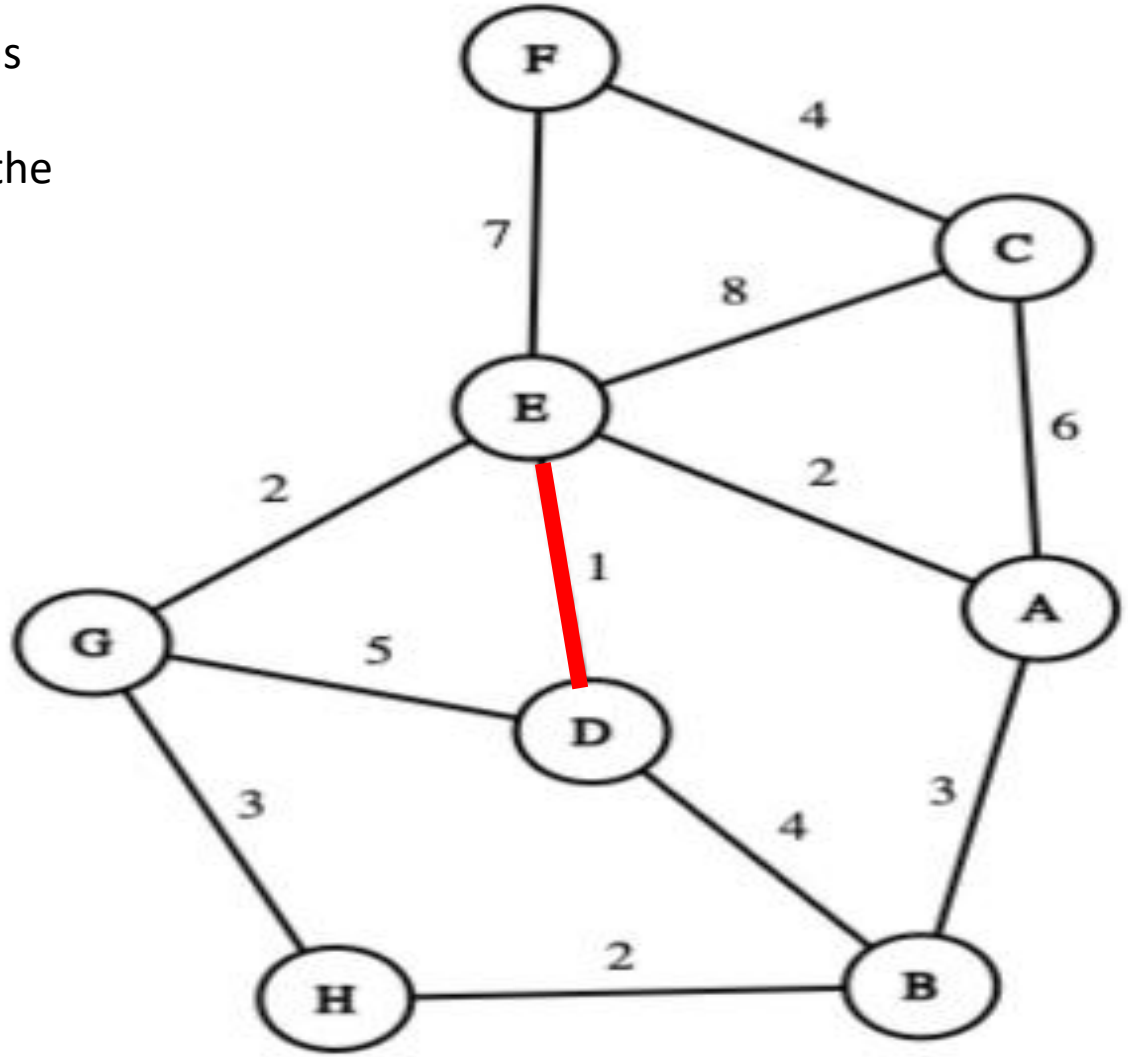
Prim’s Algorithm:
1 - Start with one of the vertices in the graph
2 - Grow the tree in successive stages
3 - At each stage, add an edge (U,V) to the tree if (U,V) has the smallest cost among all the edges such that U is in the tree and V is not and mark the edge as known



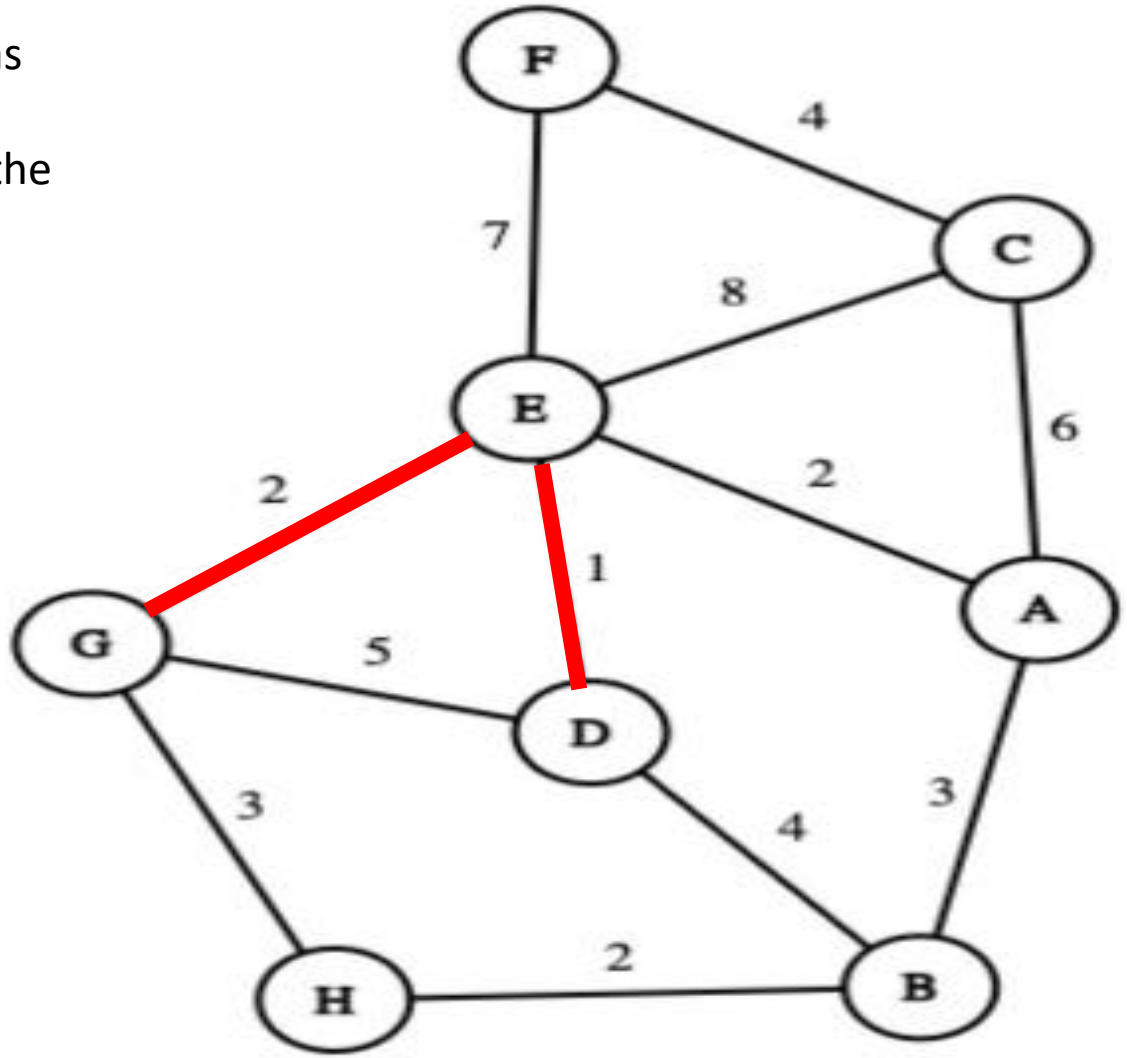
Prim's Algorithm:
1 - Start with E
2 - Grow the tree in successive stages
3 – At each stage, add an edge (U,V) to the tree if (U,V) has the smallest cost among all the edges such that U is in the tree and V is not and mark the edge as known



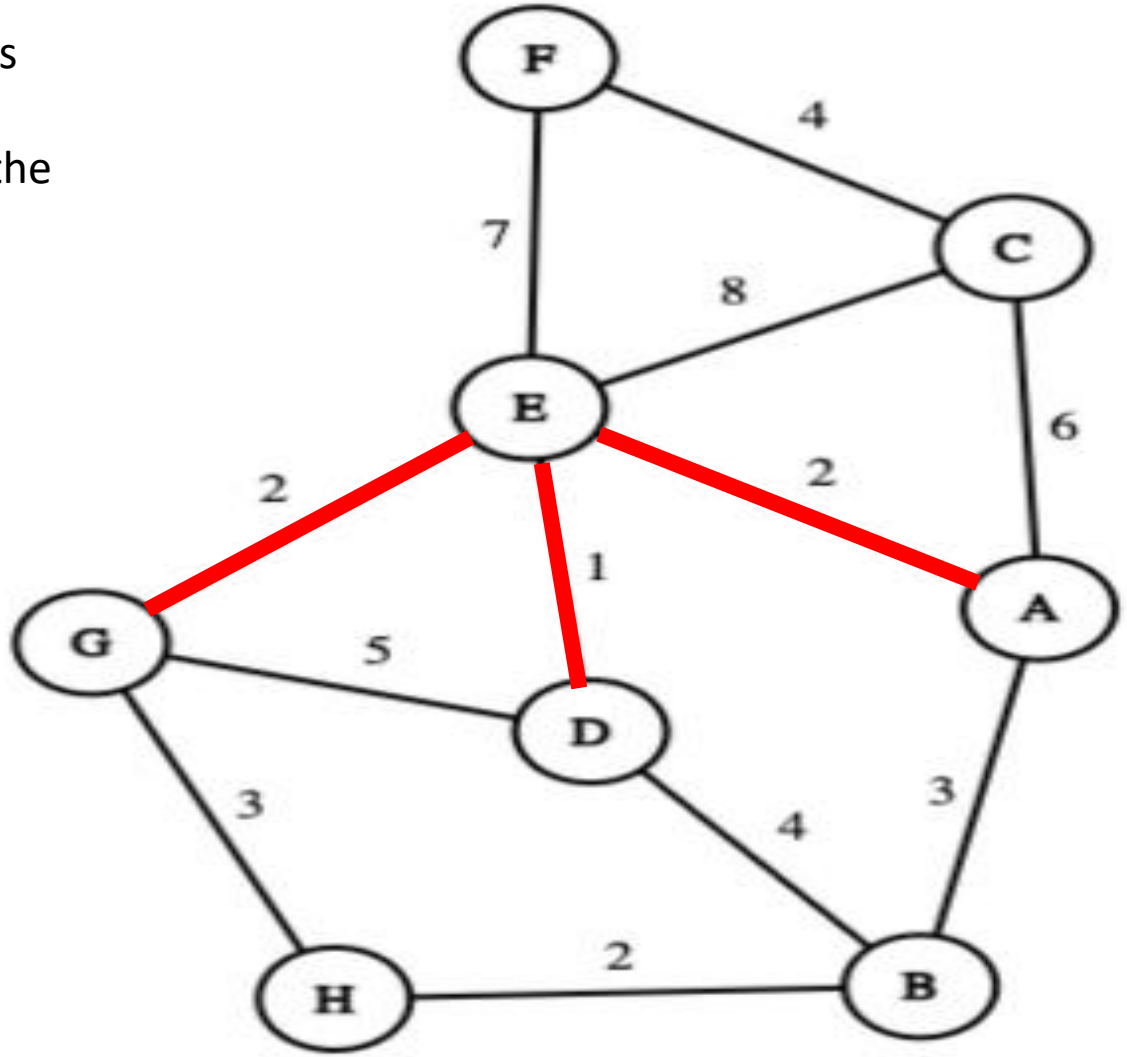
Prim's Algorithm:
- Add edge (E,D) as (E,D) has the smallest cost among all the edges such that u is in the tree and v is not



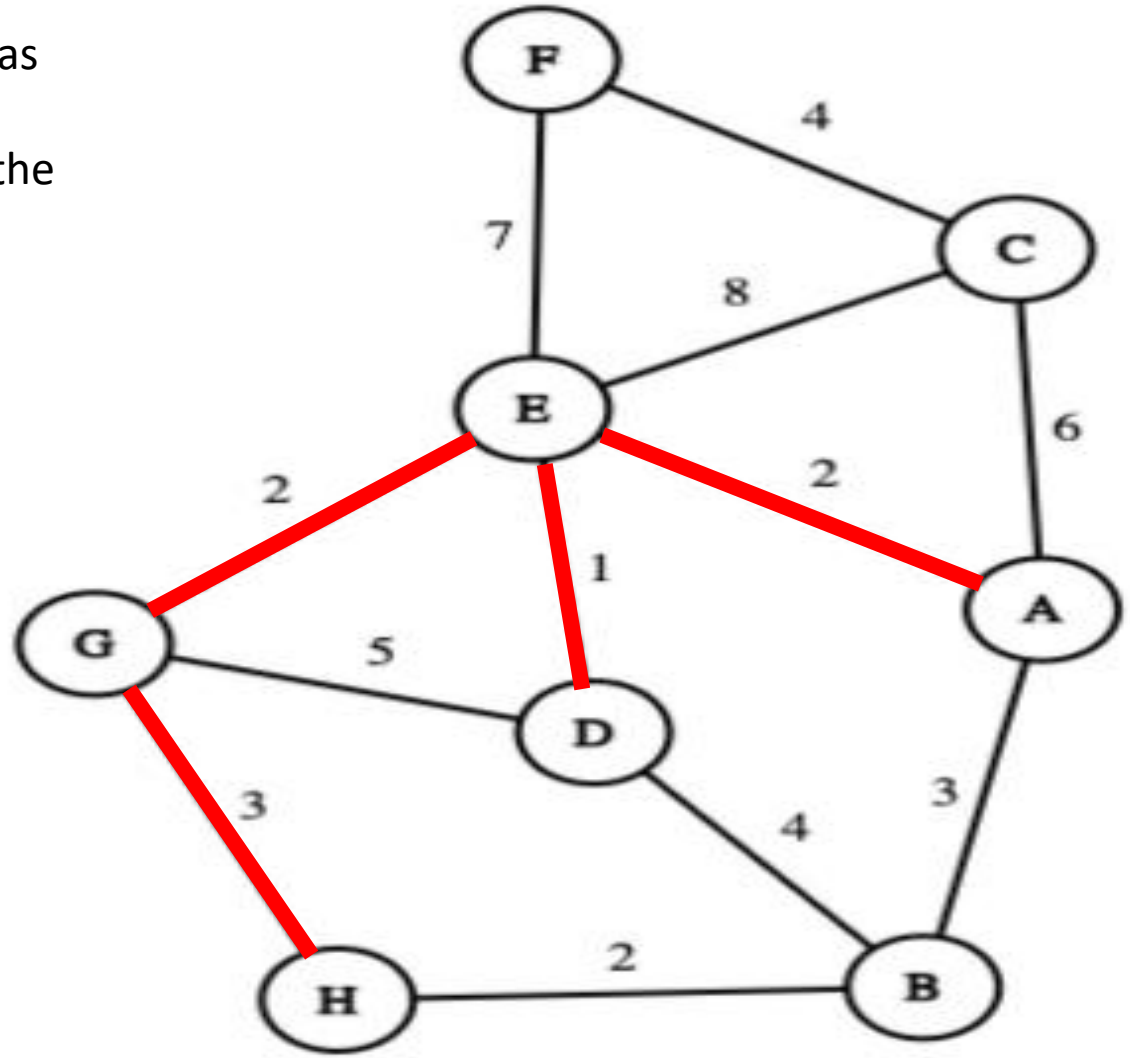
Prim's Algorithm:
- Add edge (E,G) as (E,G) has the smallest cost among all the edges such that u is in the tree and v is not



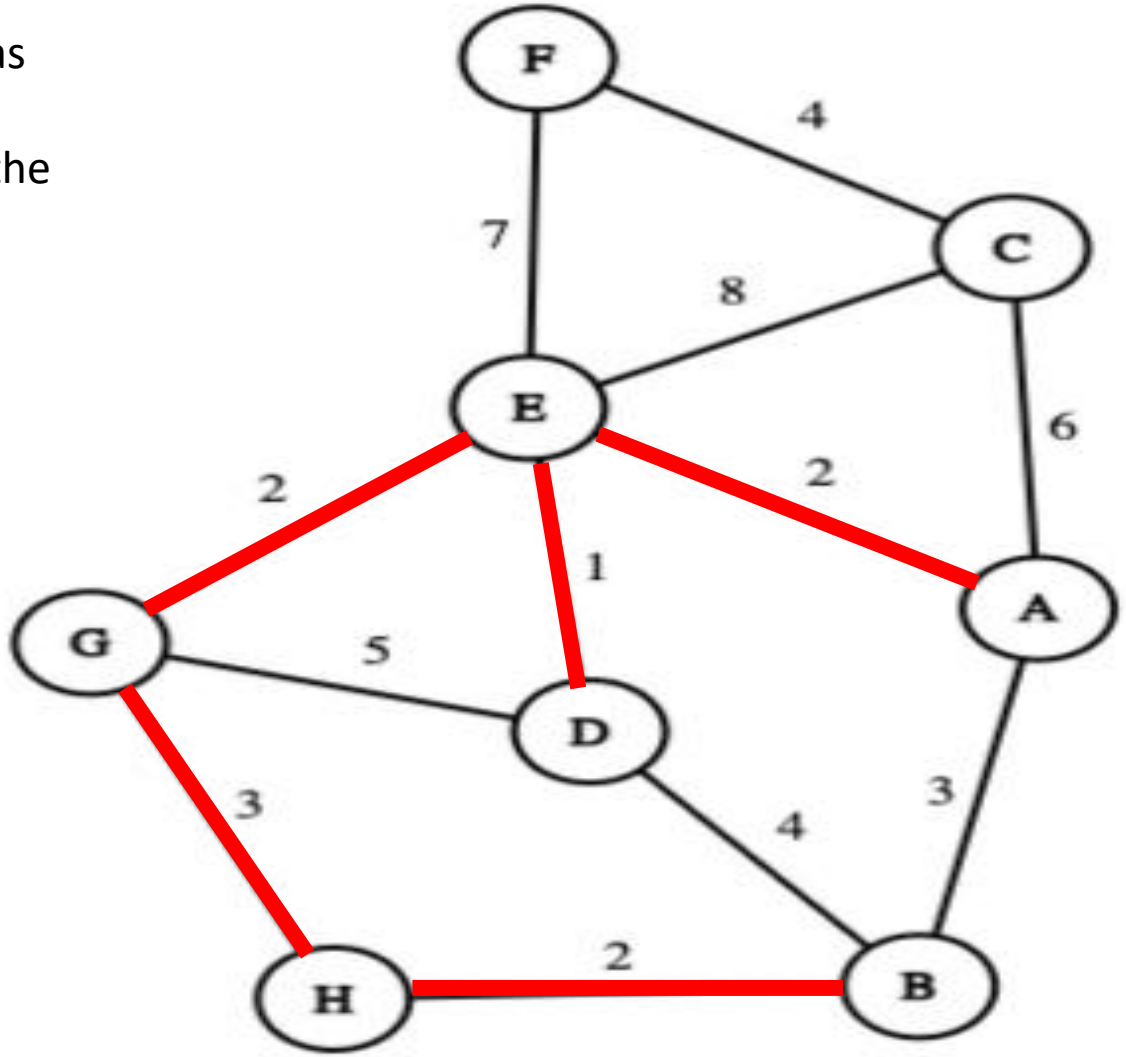
Prim's Algorithm:
- Add edge (E,A) as (E,A) has the smallest cost among all the edges such that u is in the tree and v is not



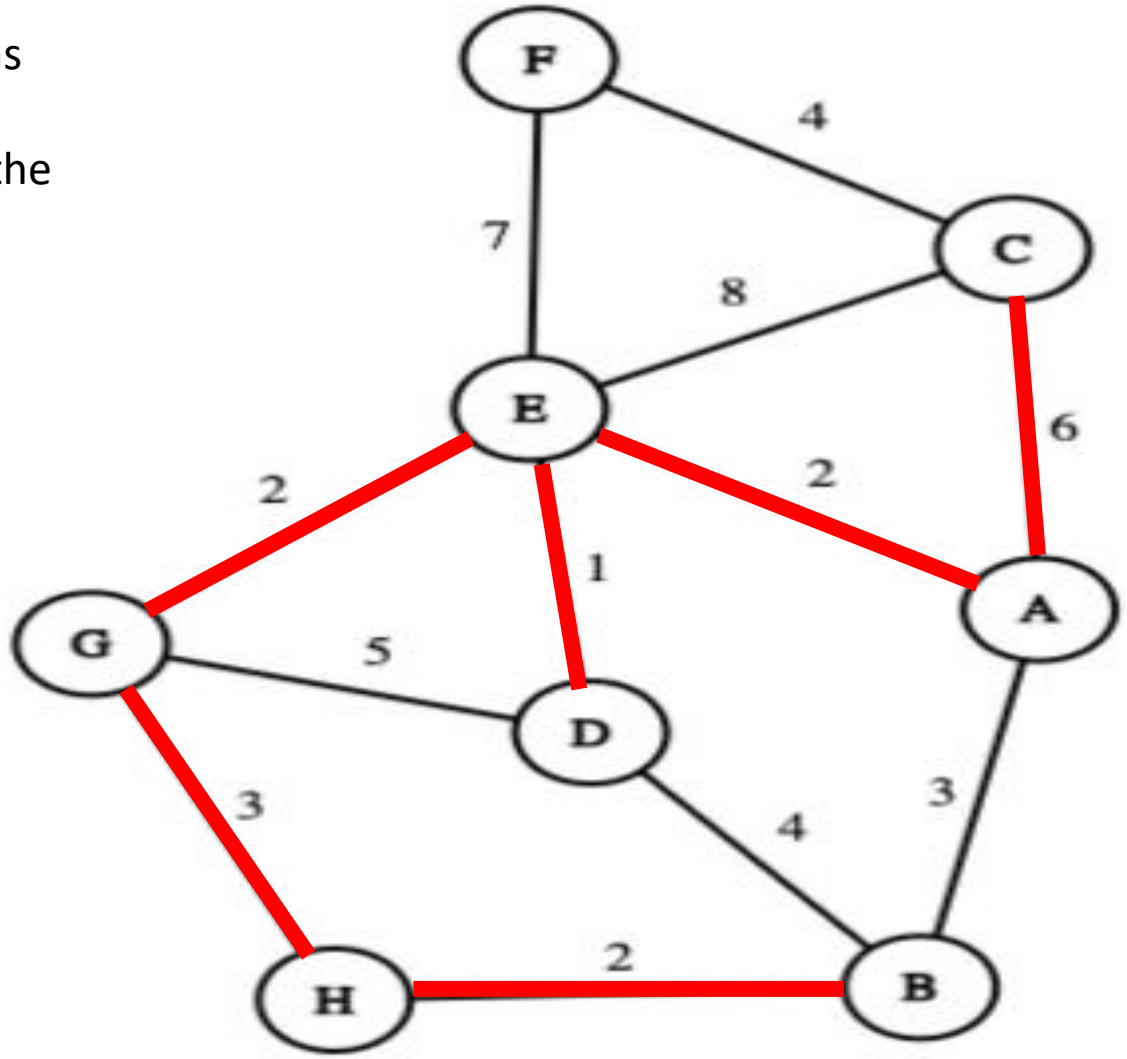
Prim's Algorithm:
- Add edge (G,H) as (G,H) has the smallest cost among all the edges such that u is in the tree and v is not



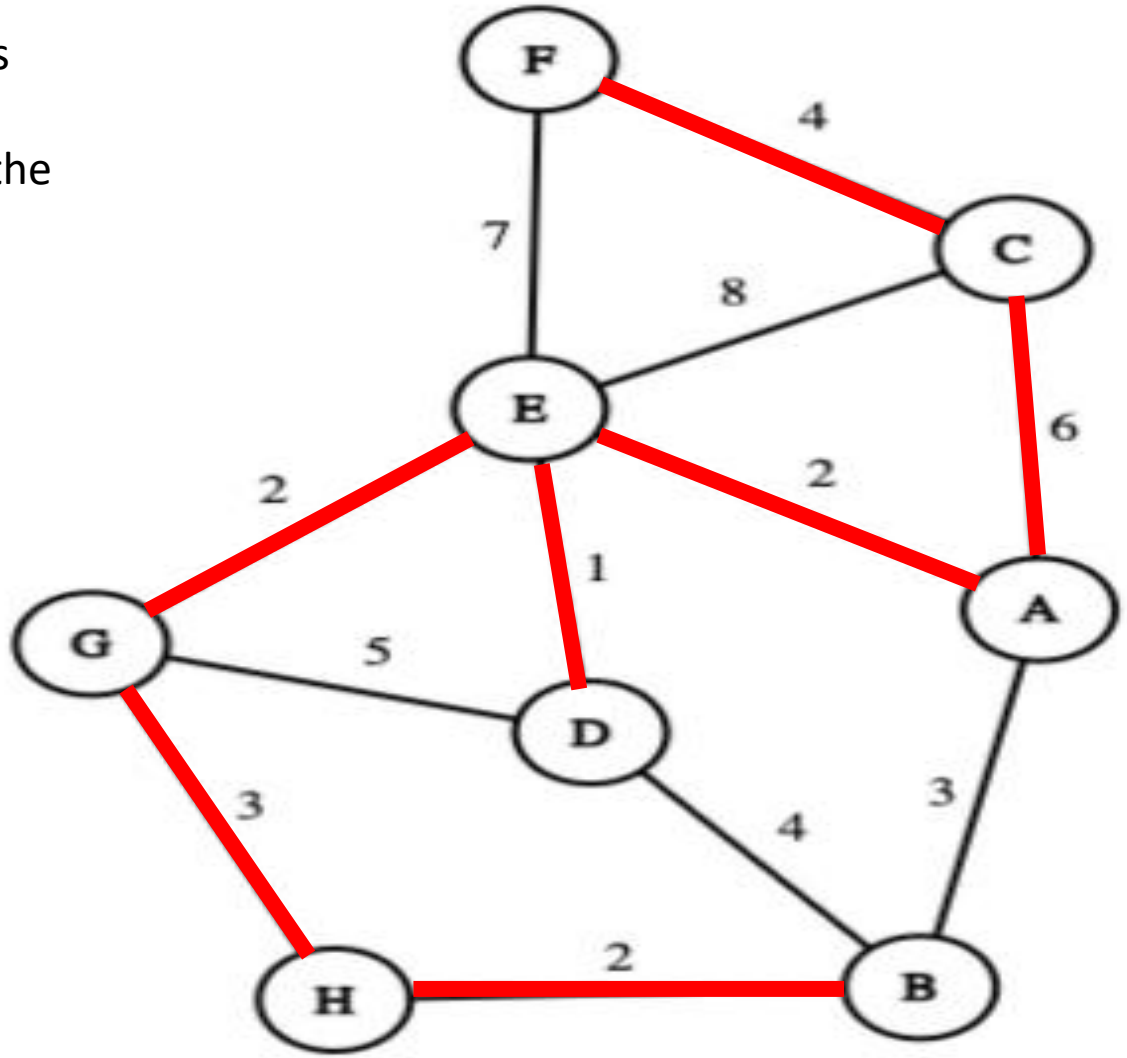
Prim's Algorithm:
- Add edge (H,B) as (H,B) has the smallest cost among all the edges such that u is in the tree and v is not



Prim's Algorithm:
- Add edge (A,C) as (A,C) has the smallest cost among all the edges such that u is in the tree and v is not



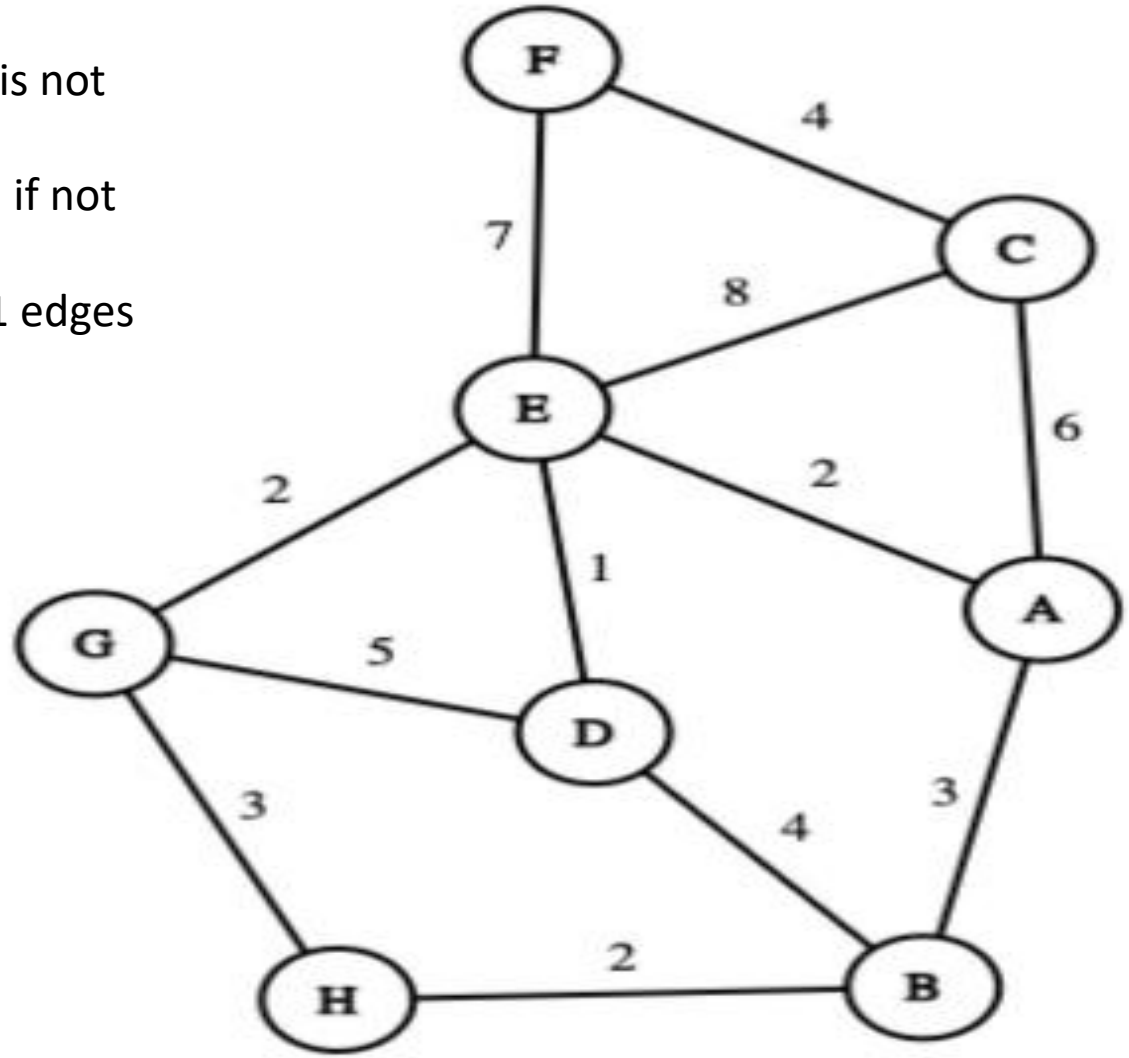
Prim's Algorithm:
- Add edge (C,F) as (C,F) has the smallest cost among all the edges such that u is in the tree and v is not
- All vertices have been marked as known, finalize



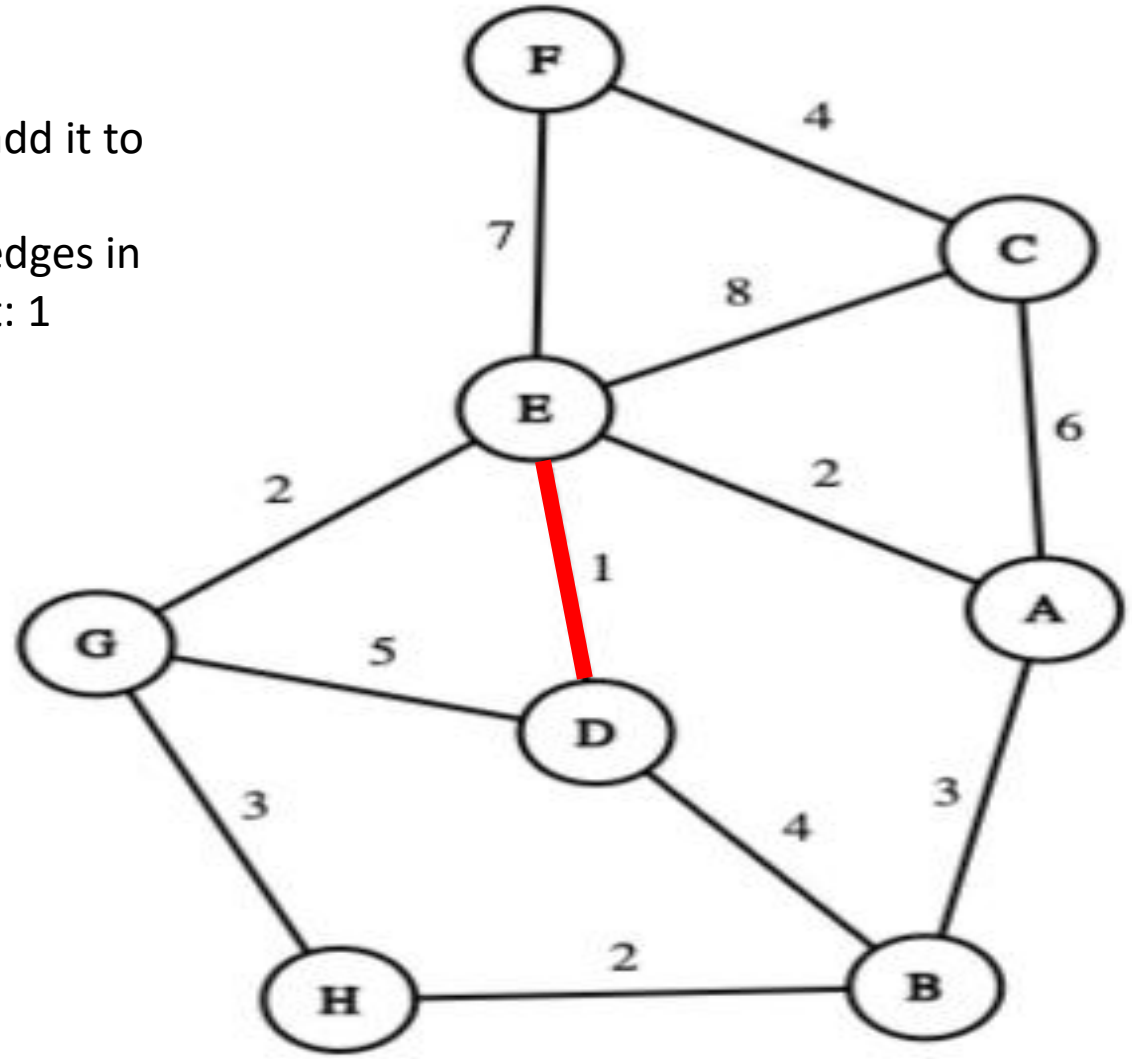
Question 3

Trace the Kruskal's minimum spanning tree algorithm on the graph in Figure 1

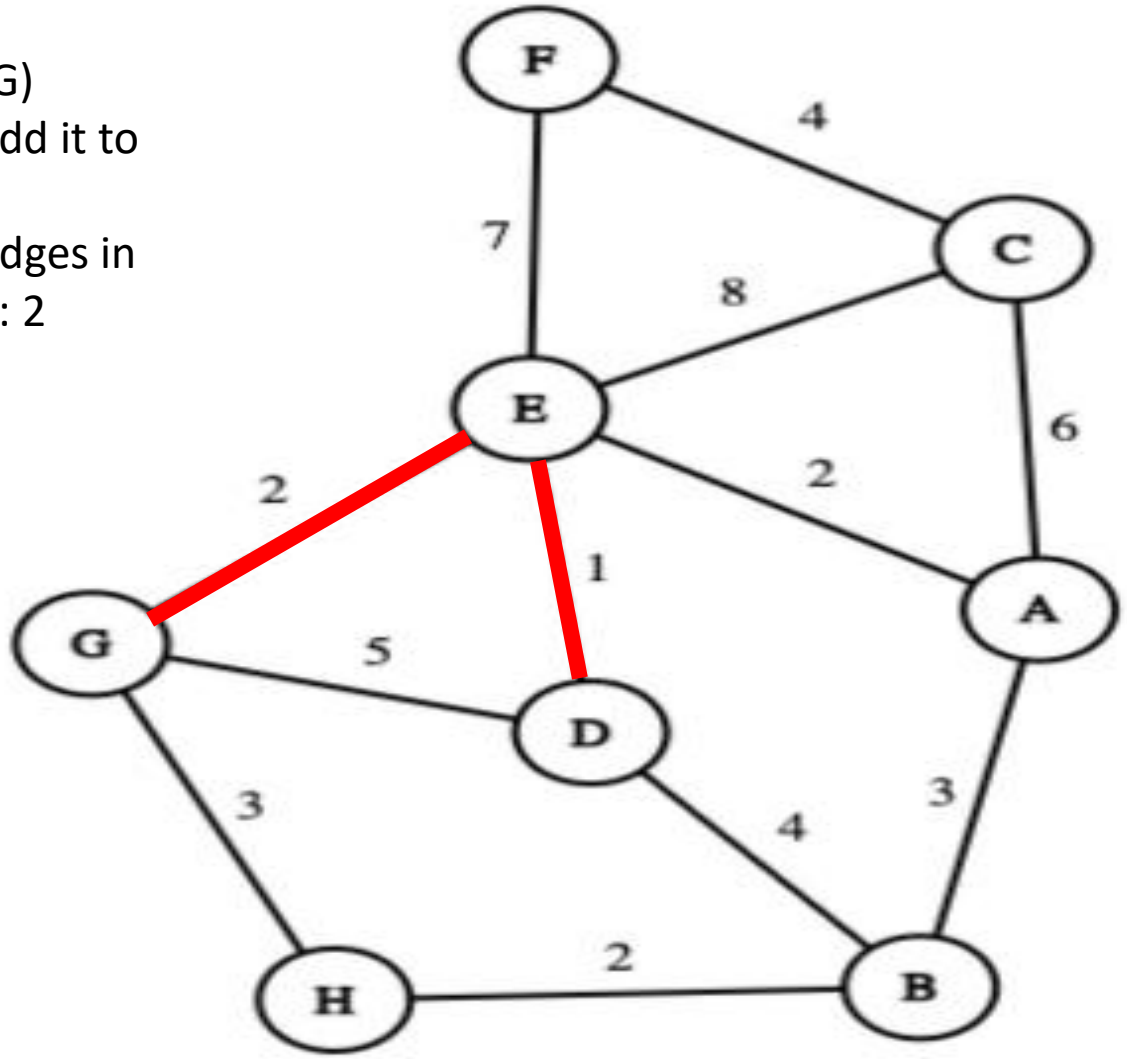
Kruskal's Algorithm:
1 - Pick smallest edge which is not yet in the tree
2 - Check if it creates a cycle, if not add it to the tree
3 - Repeat until there are V-1 edges in the tree



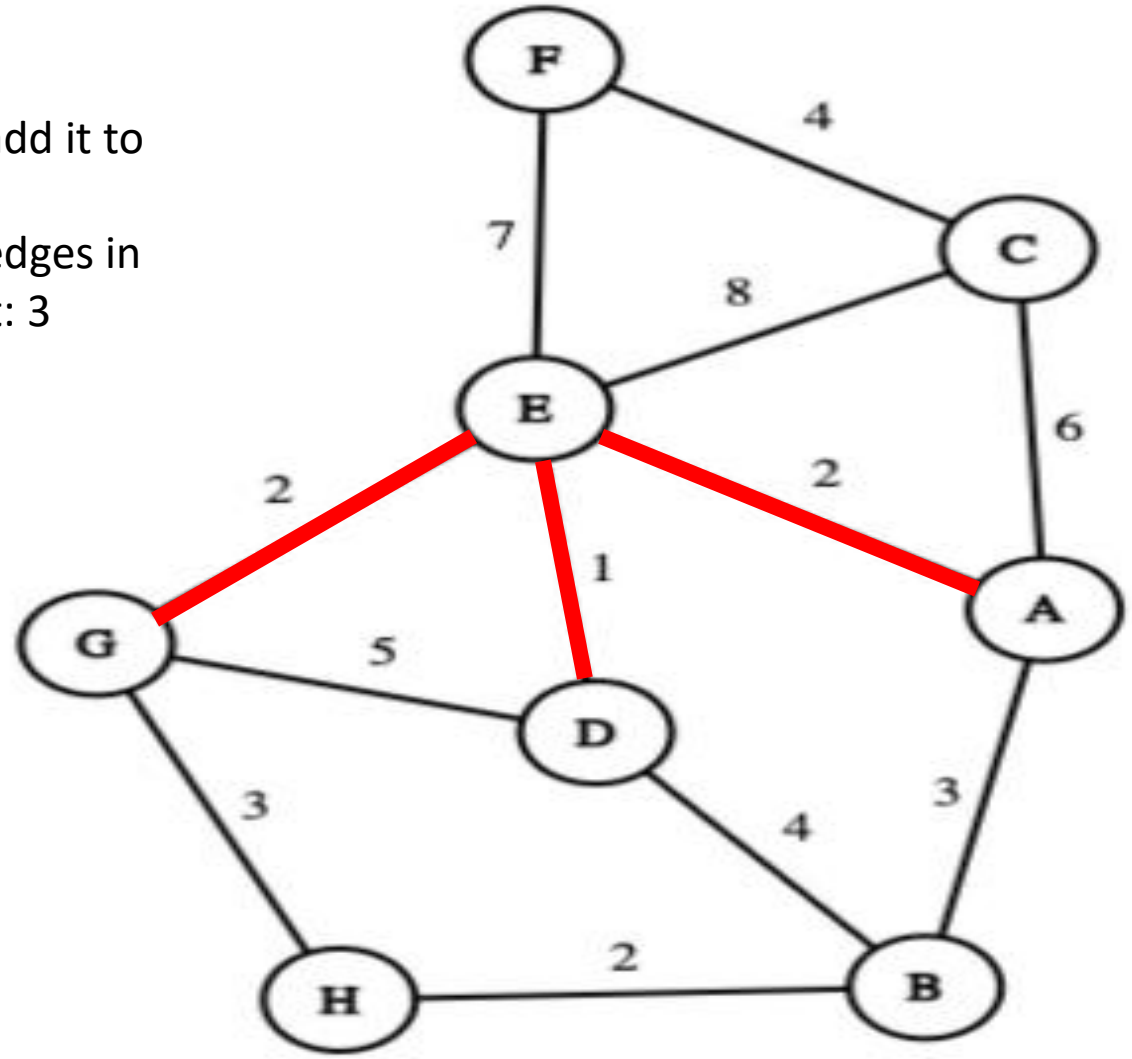
Kruskal's Algorithm:
1 - Pick E-D
2 - Does not create a cycle, add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 1



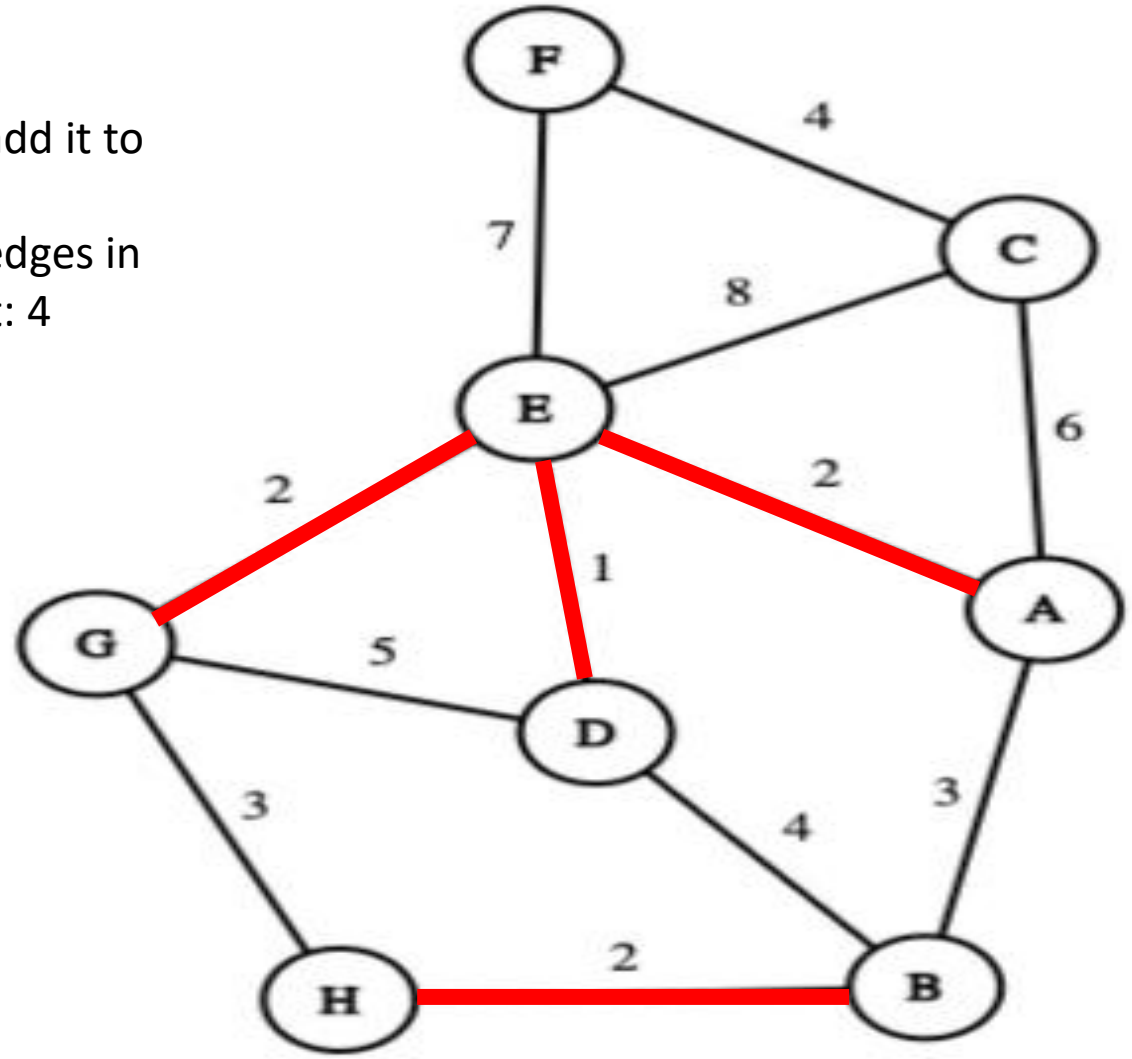
Kruskal's Algorithm:
1 - Pick E-G or E-A (Picked E-G)
2 - Does not create a cycle, add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 2



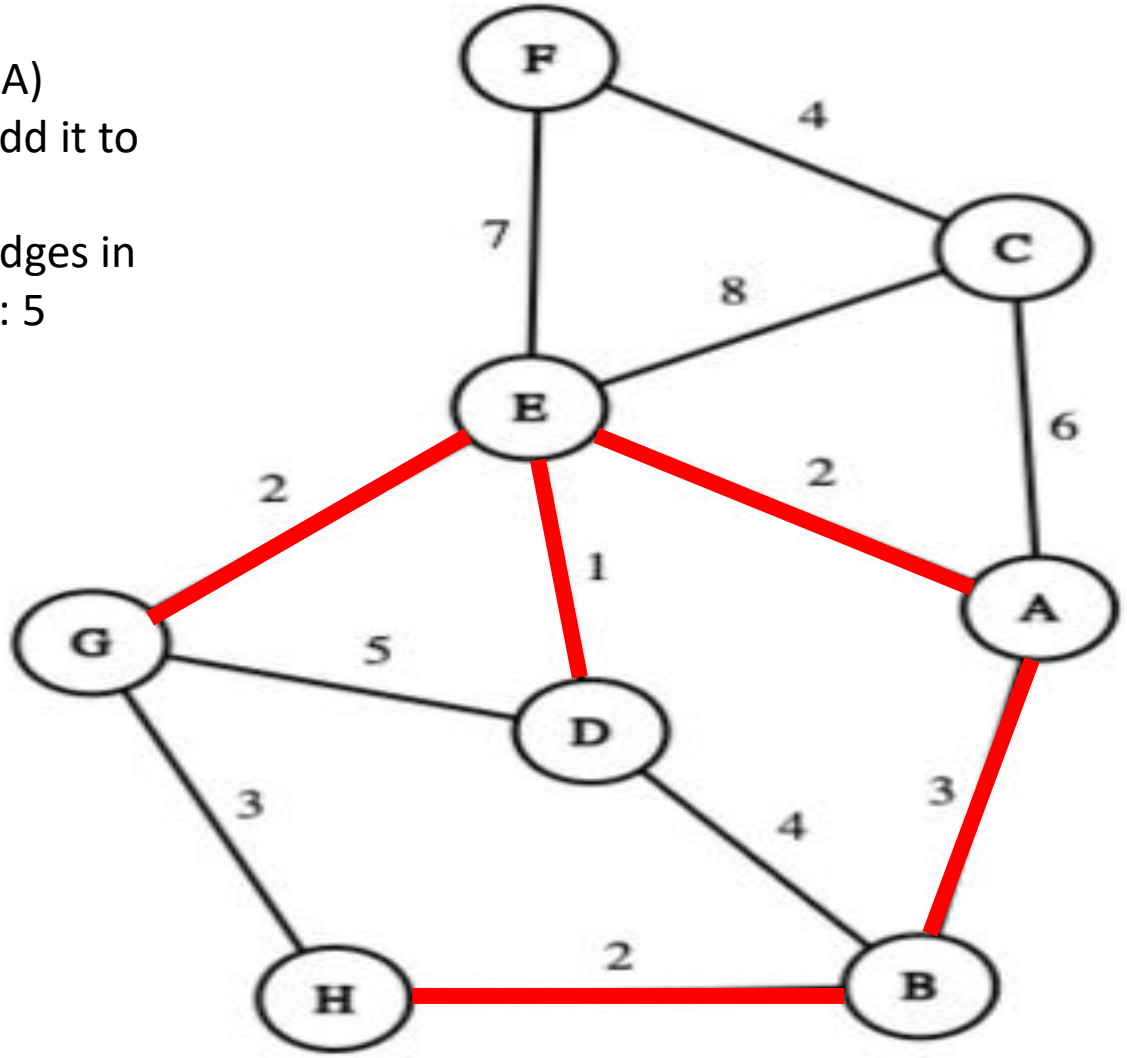
Kruskal's Algorithm:
1 - Pick E-A
2 - Does not create a cycle, add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 3



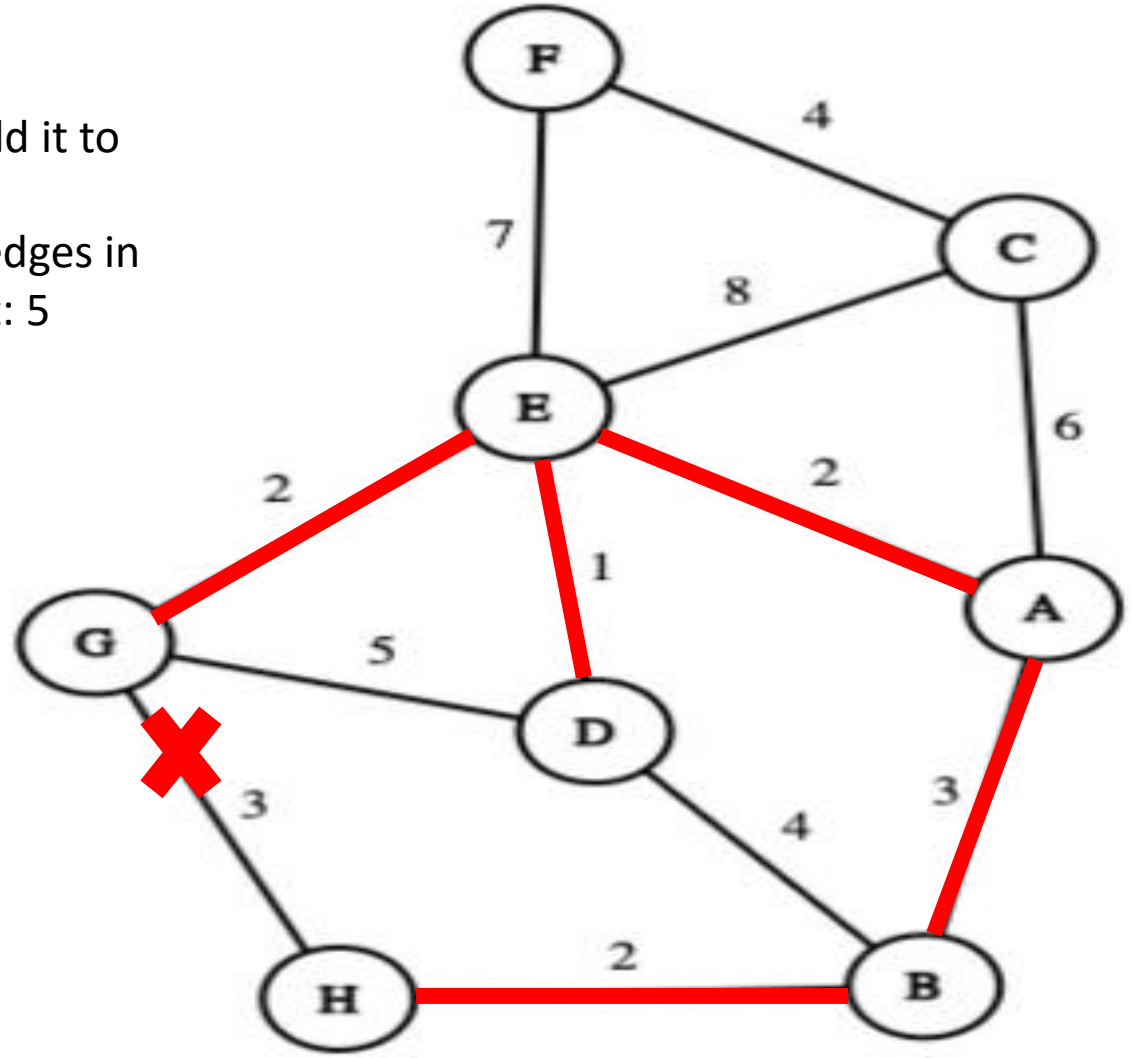
Kruskal's Algorithm:
1 - Pick H-B
2 - Does not create a cycle, add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 4



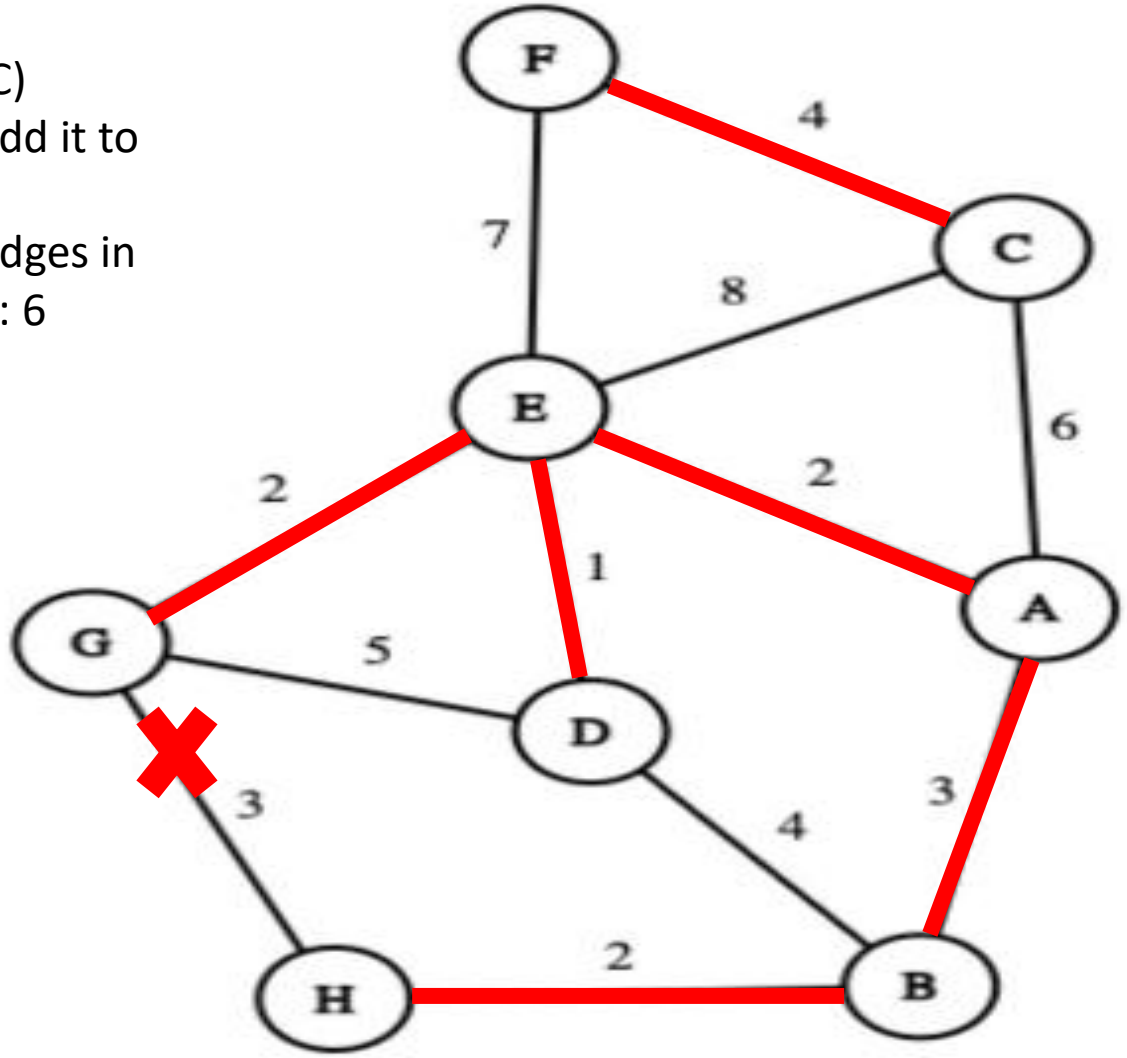
Kruskal's Algorithm:
1 - Pick B-A or G-H (Picked B-A)
2 - Does not create a cycle, add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 5



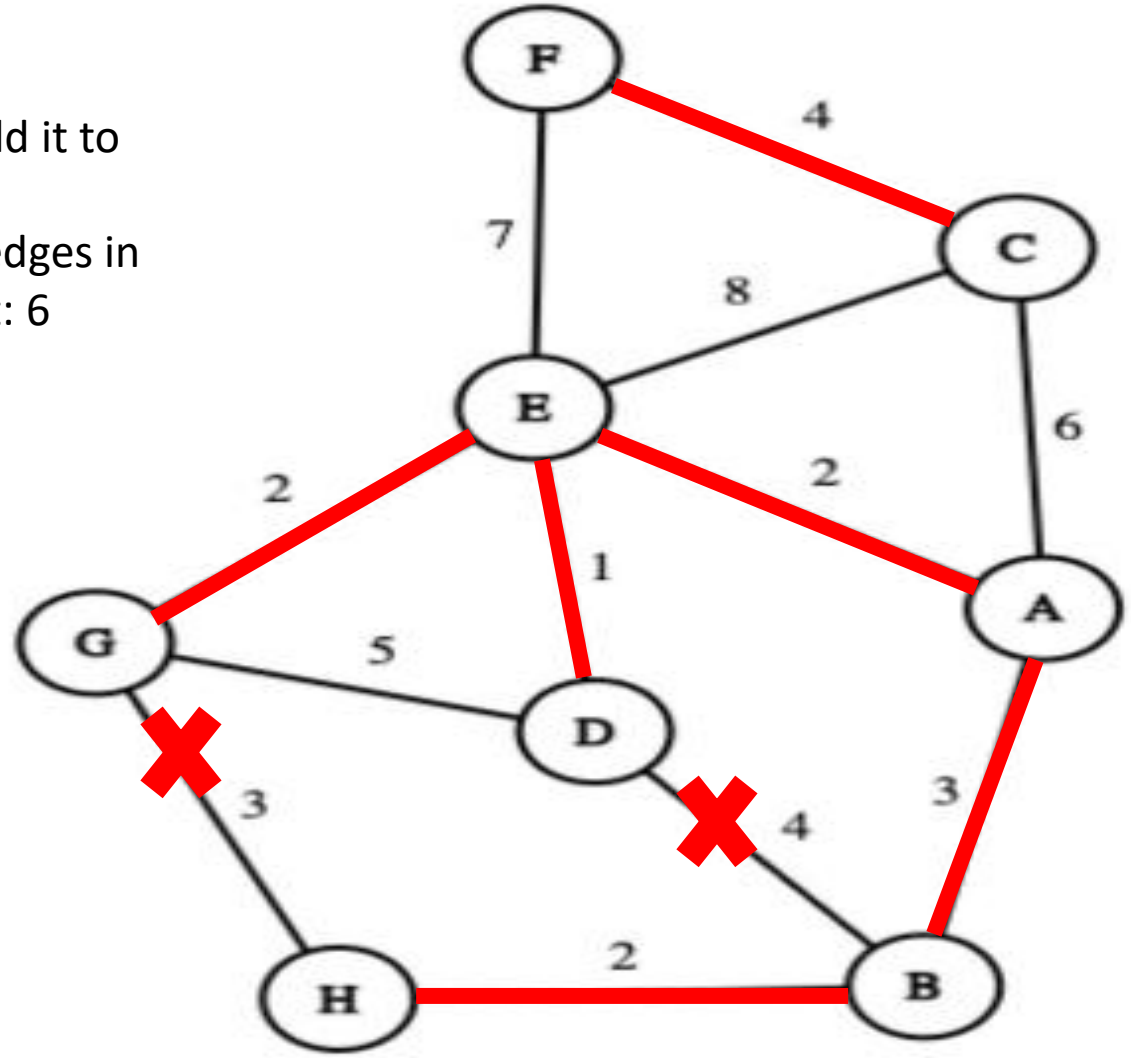
Kruskal's Algorithm:
1 - Pick G-H
2 - Creates a cycle, do not add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 5



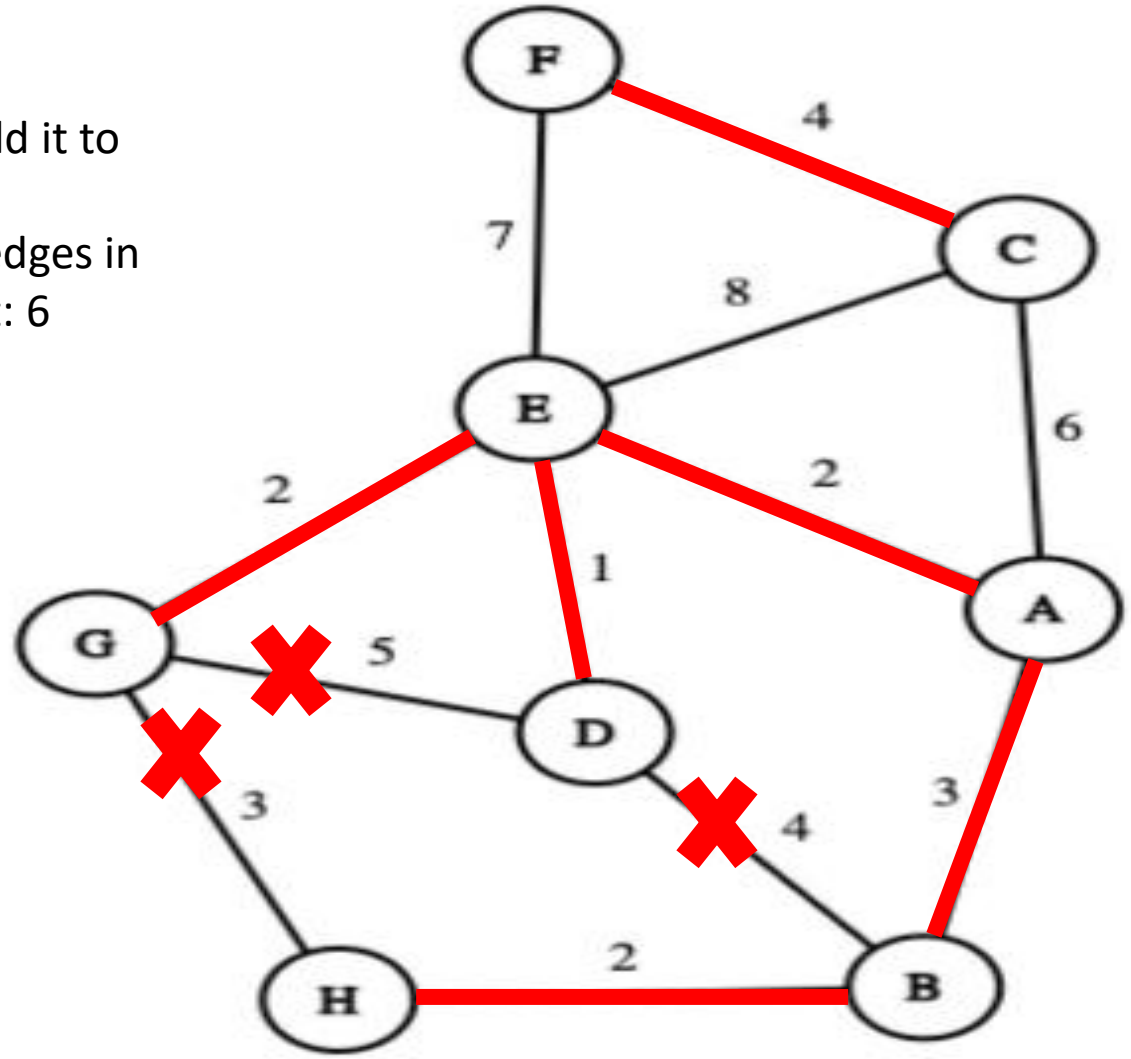
Kruskal's Algorithm:
1 - Pick F-C or D-B (Picked F-C)
2 - Does not create a cycle, add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 6



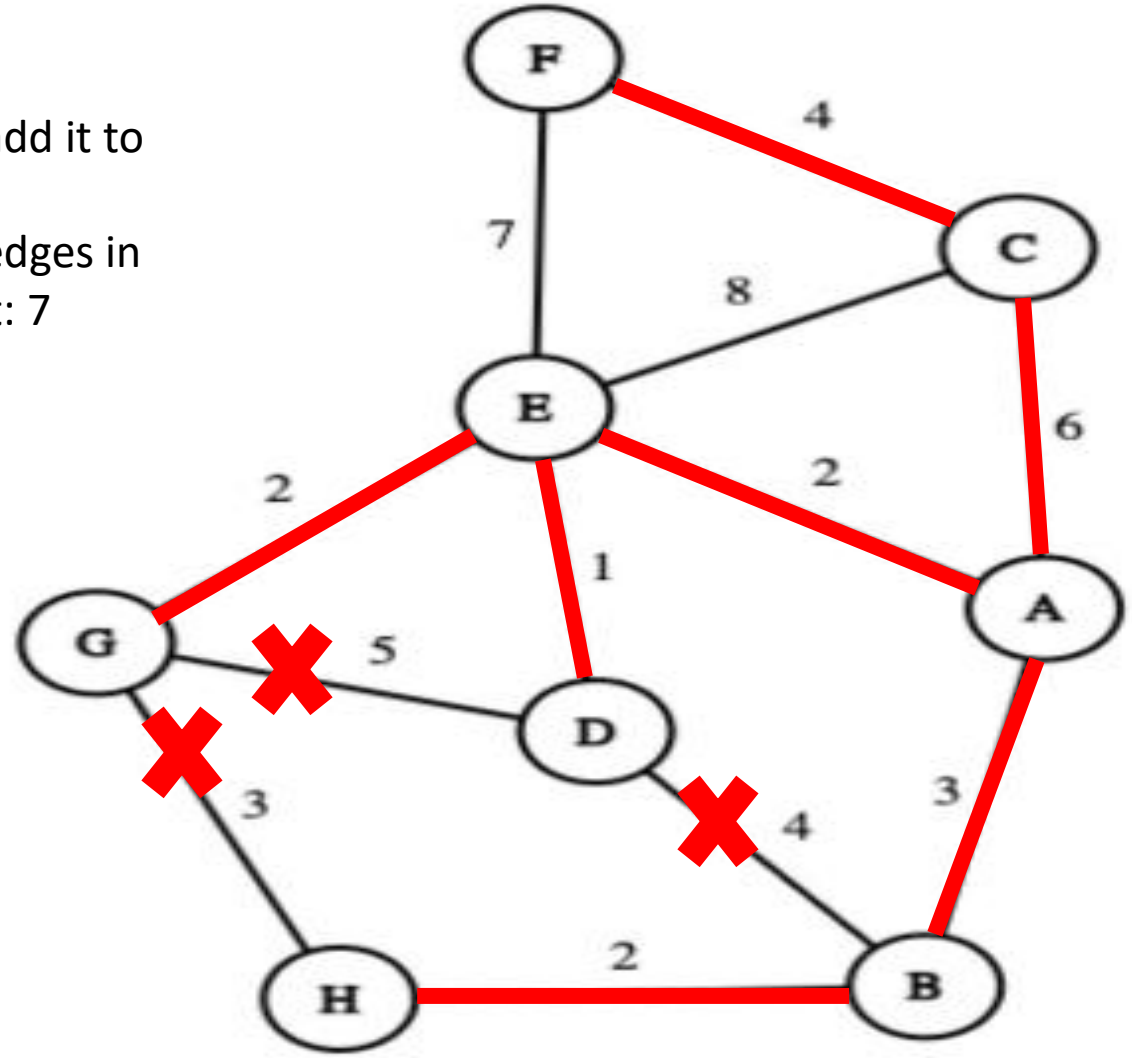
Kruskal's Algorithm:
1 - Pick D-B
2 - Creates a cycle, do not add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 6



Kruskal's Algorithm:
1 - Pick G-D
2 - Creates a cycle, do not add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 6



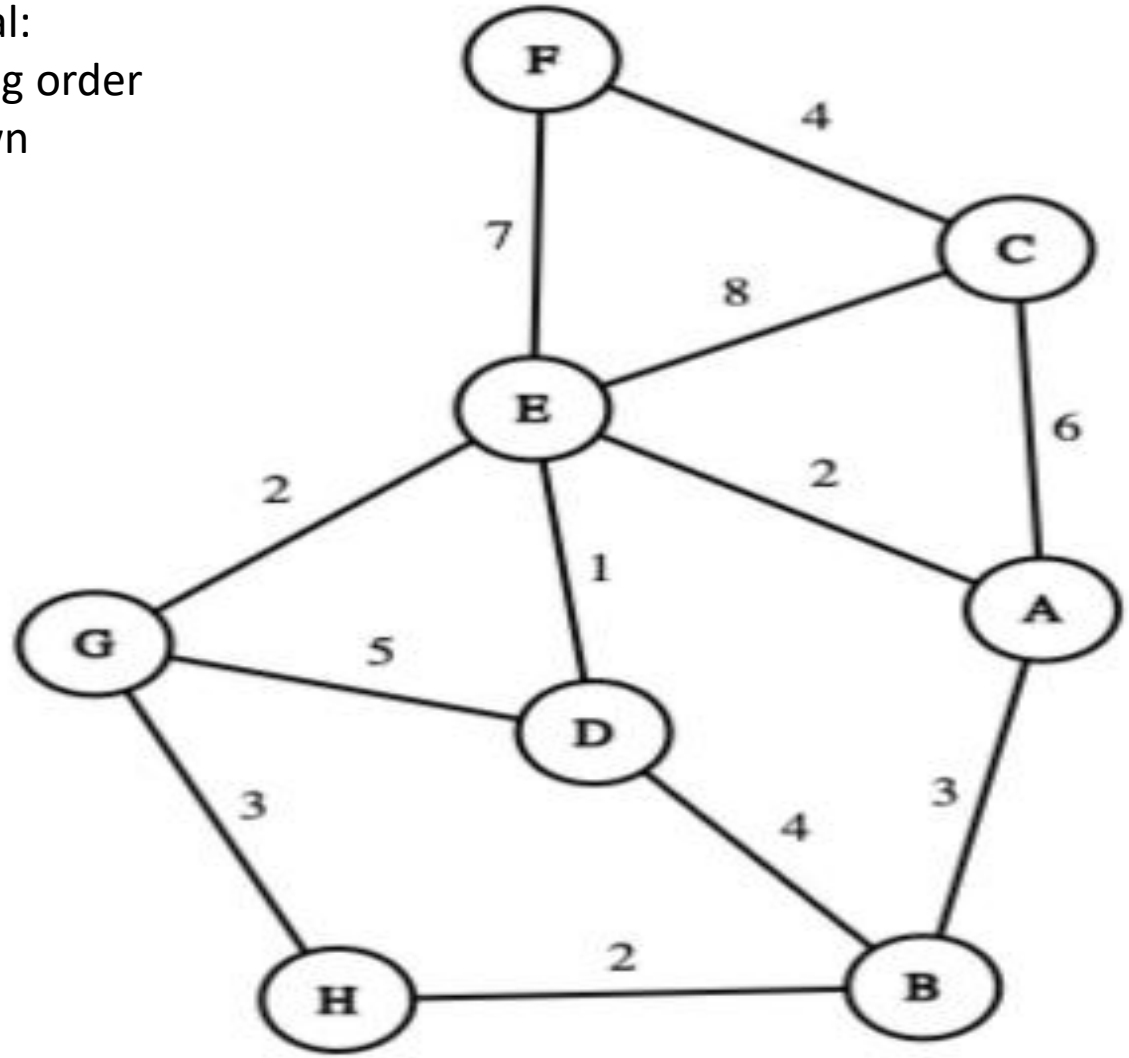
Kruskal's Algorithm:
1 - Pick C-A
2 - Does not create a cycle, add it to the tree
3 - Repeat until there are 7 edges in the tree. Current edge count: 7
Finalize the process



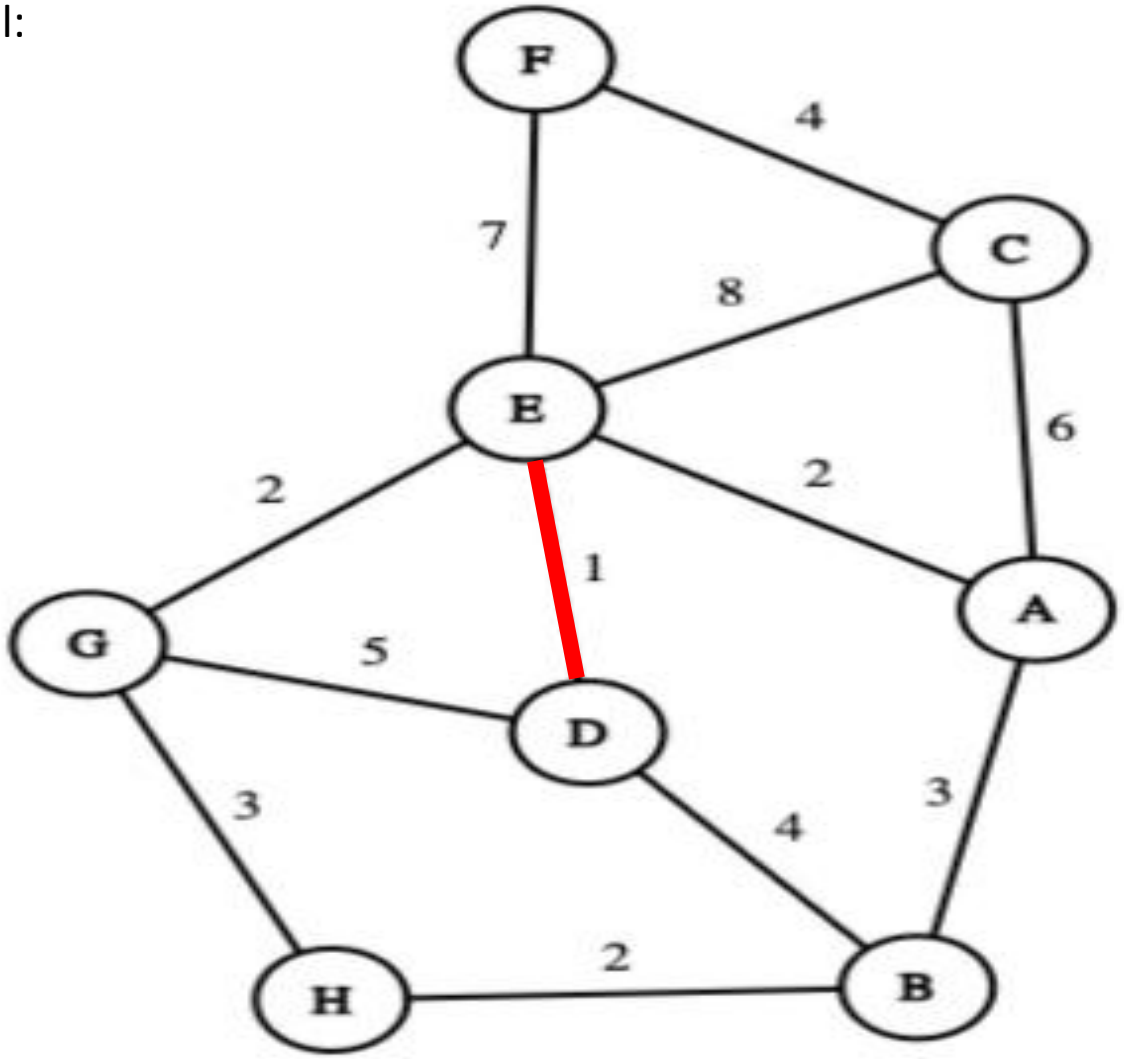
Question 4

Trace the breadth-first search traversal algorithm on the graph in Figure 1 starting from vertex E.

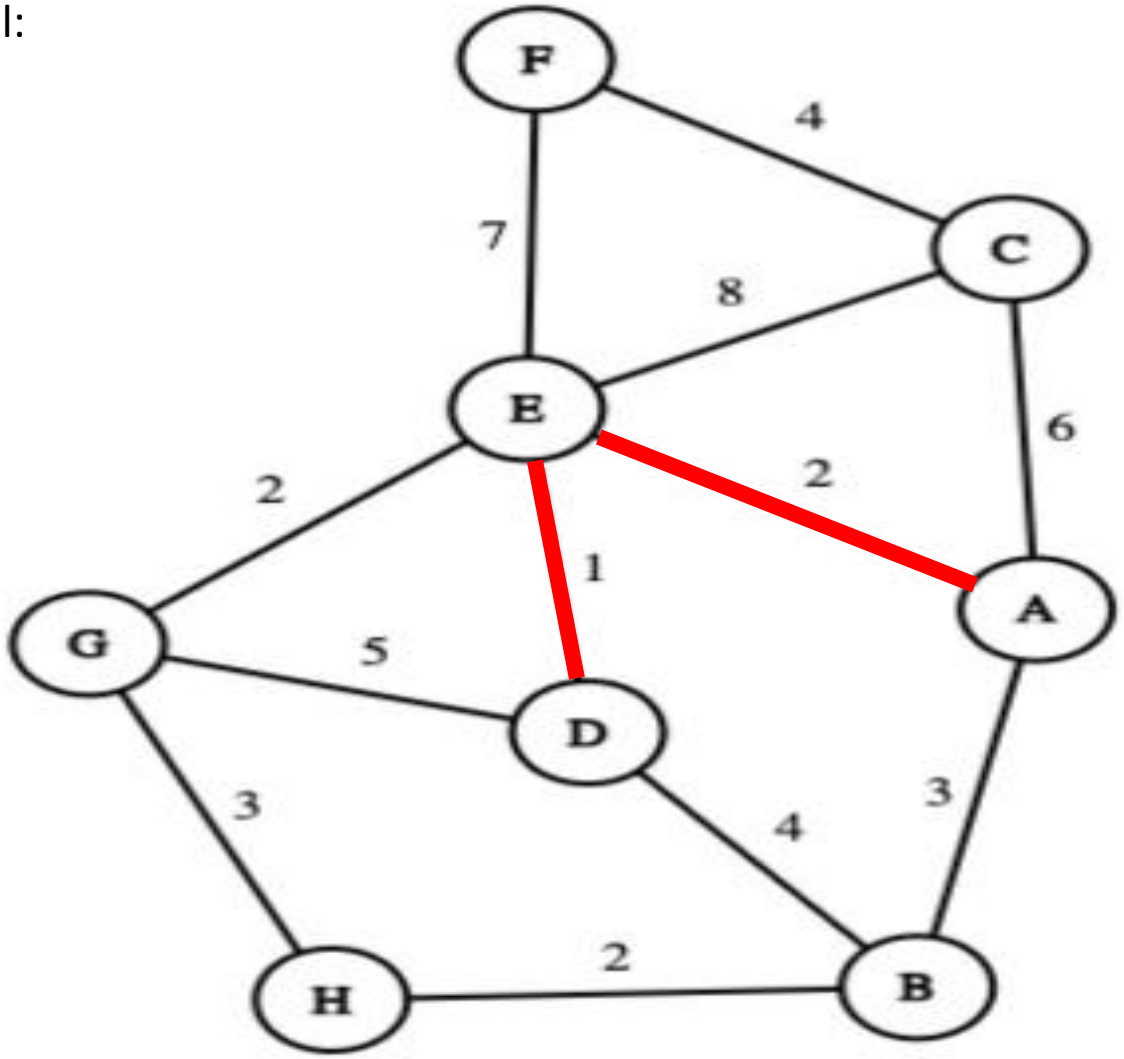
Breadth First Search Traversal:
Traverse vertices in increasing order while marking them as known
- Traverse vertices 1 away
- Traverse vertices 2 away
- Traverse vertices 3 away
…
- Traverse vertices V-1 away



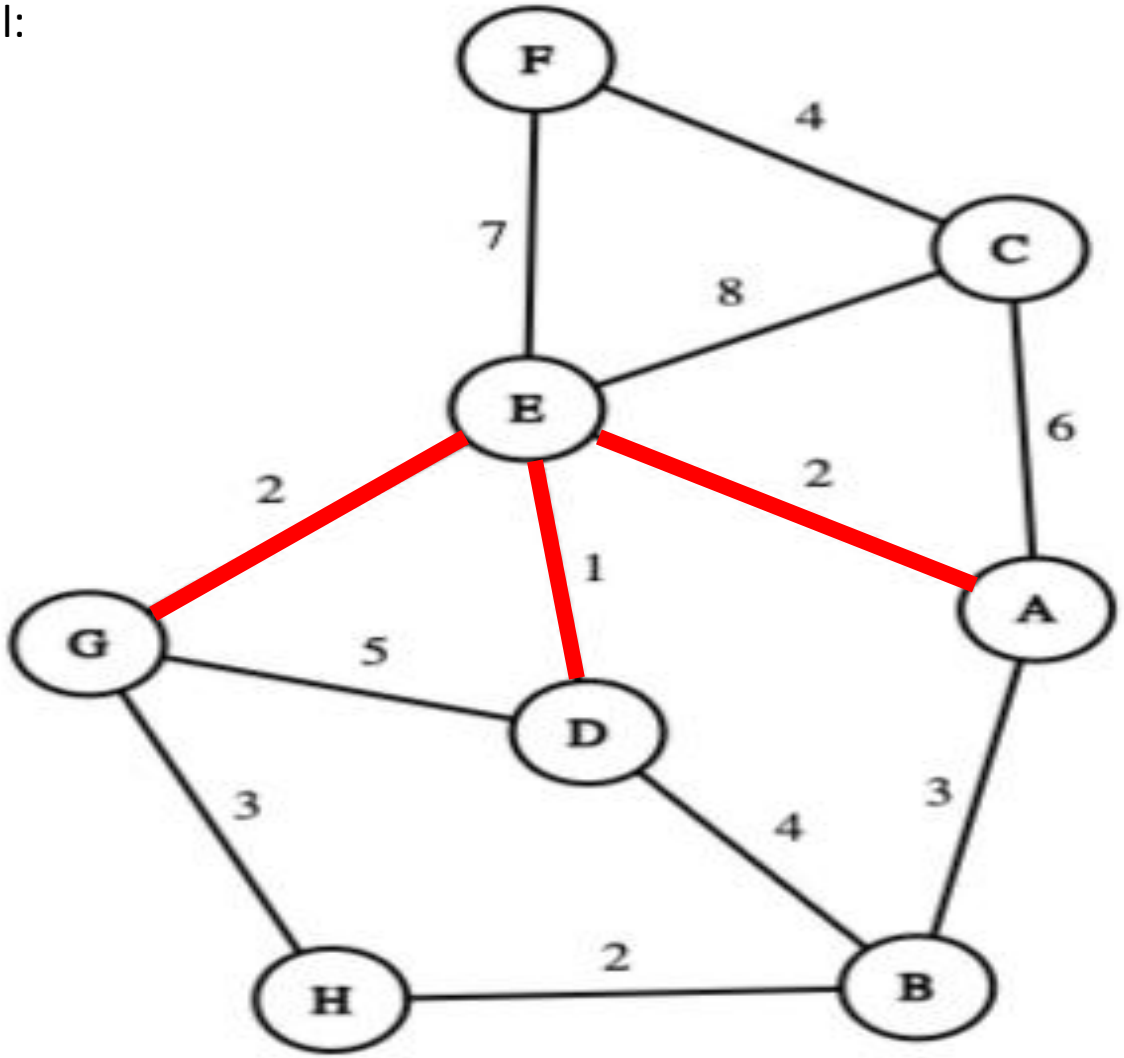
Breadth First Search Traversal:
- Traverse vertices 1 away



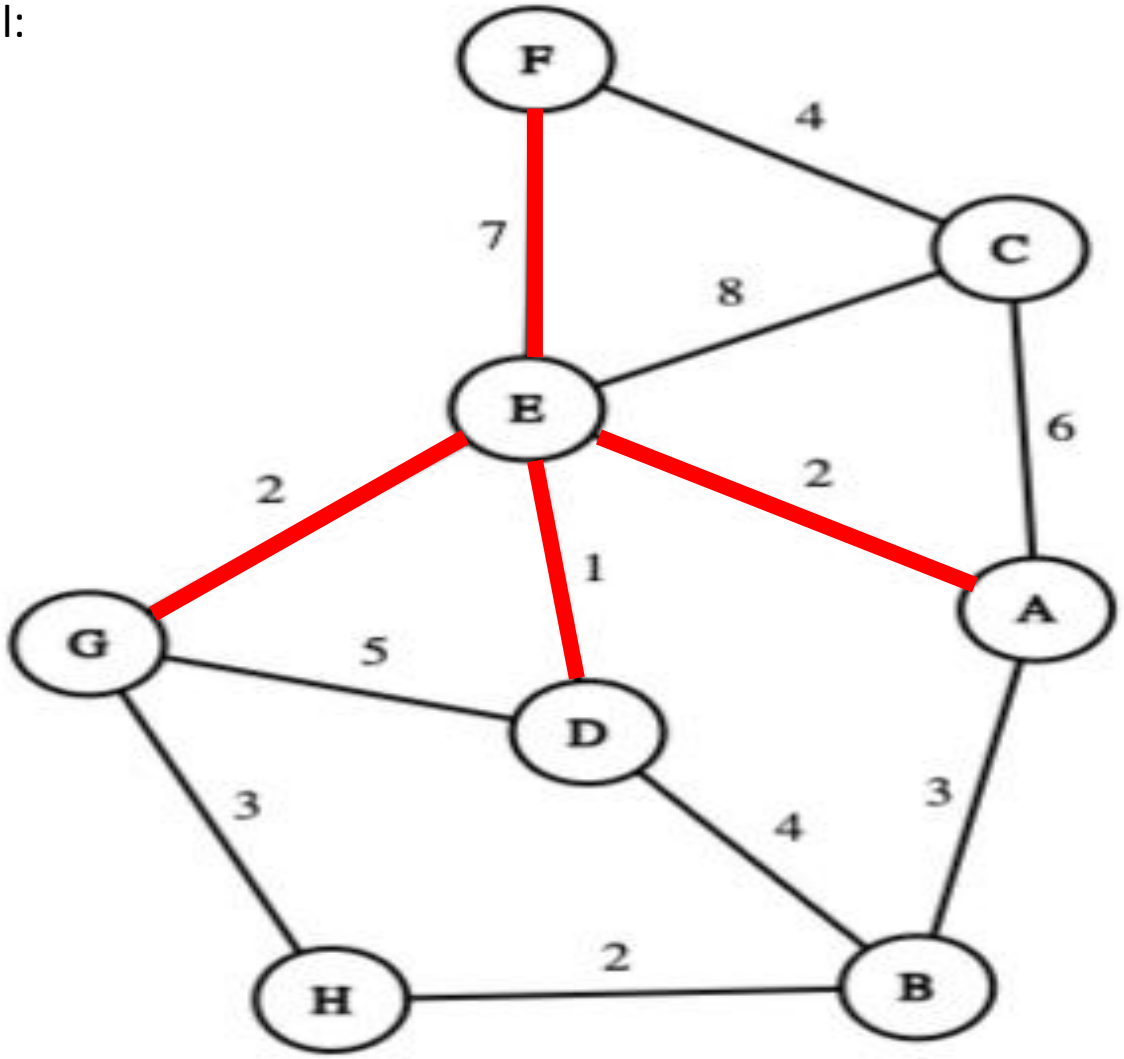
Breadth First Search Traversal:
- Traverse vertices 1 away



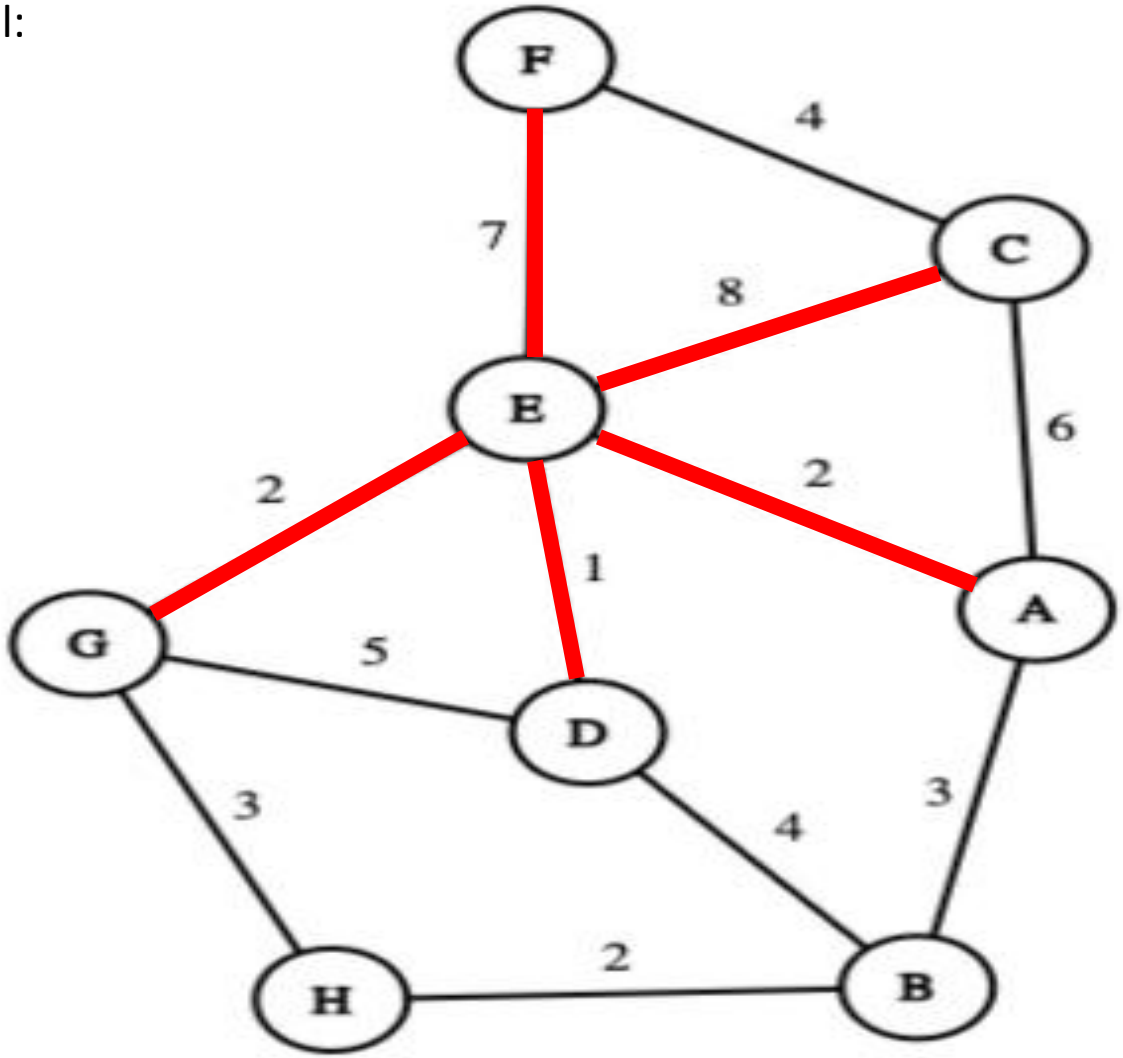
Breadth First Search Traversal:
- Traverse vertices 1 away



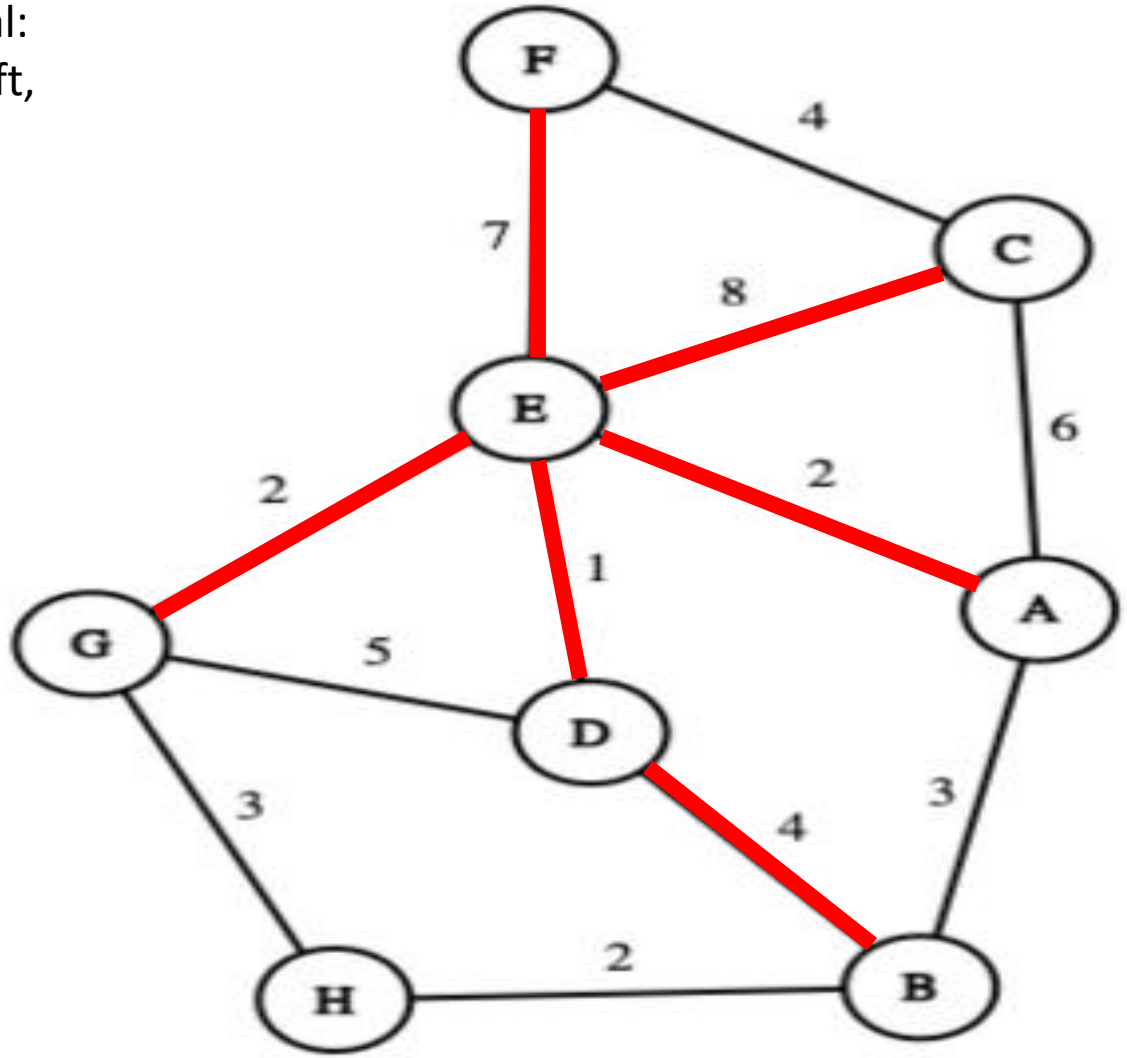
Breadth First Search Traversal:
- Traverse vertices 1 away



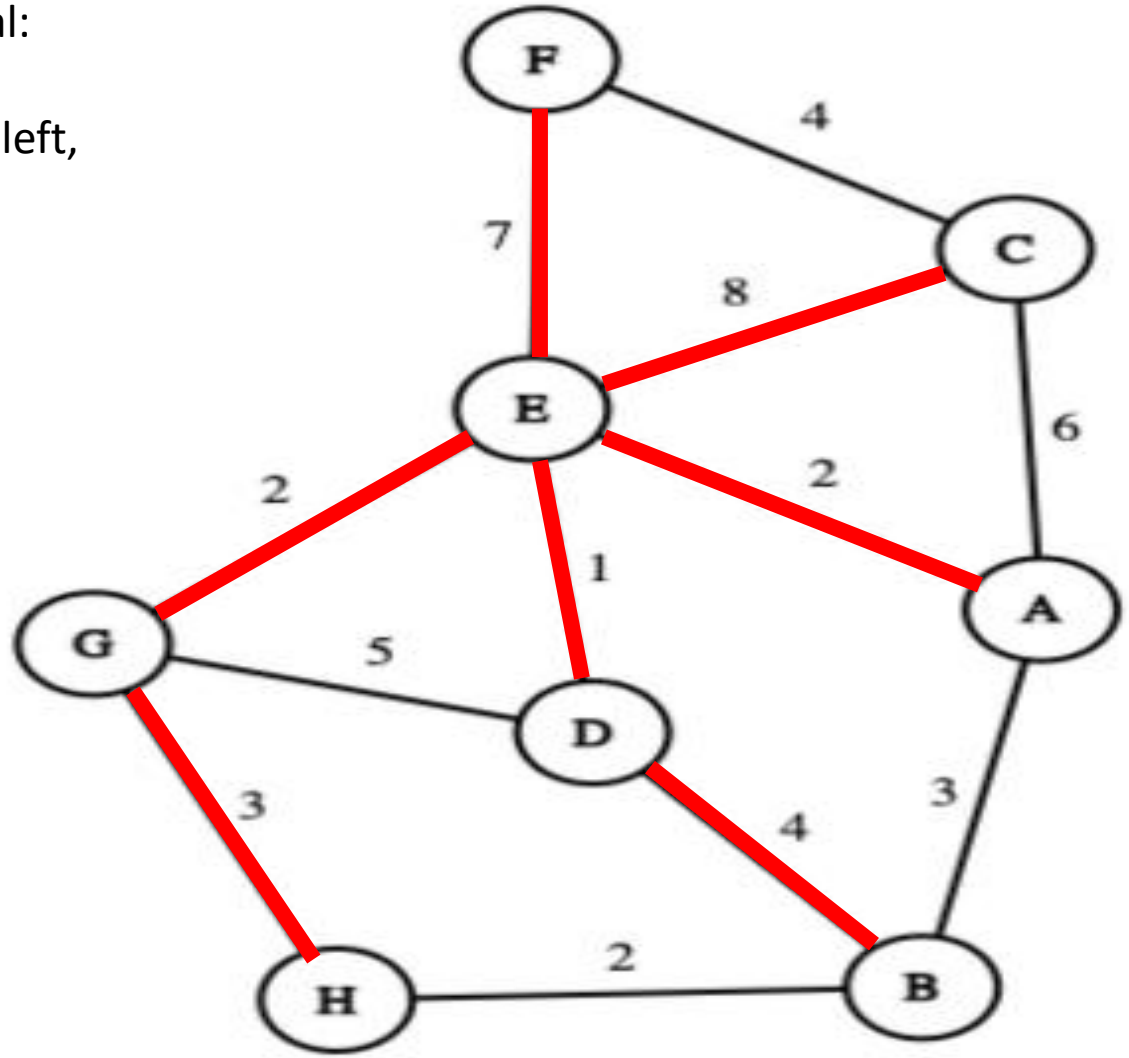
Breadth First Search Traversal:
- Traverse vertices 1 away



Breadth First Search Traversal:
- No more vertices 1 away left,
- Traverse vertices 2 away



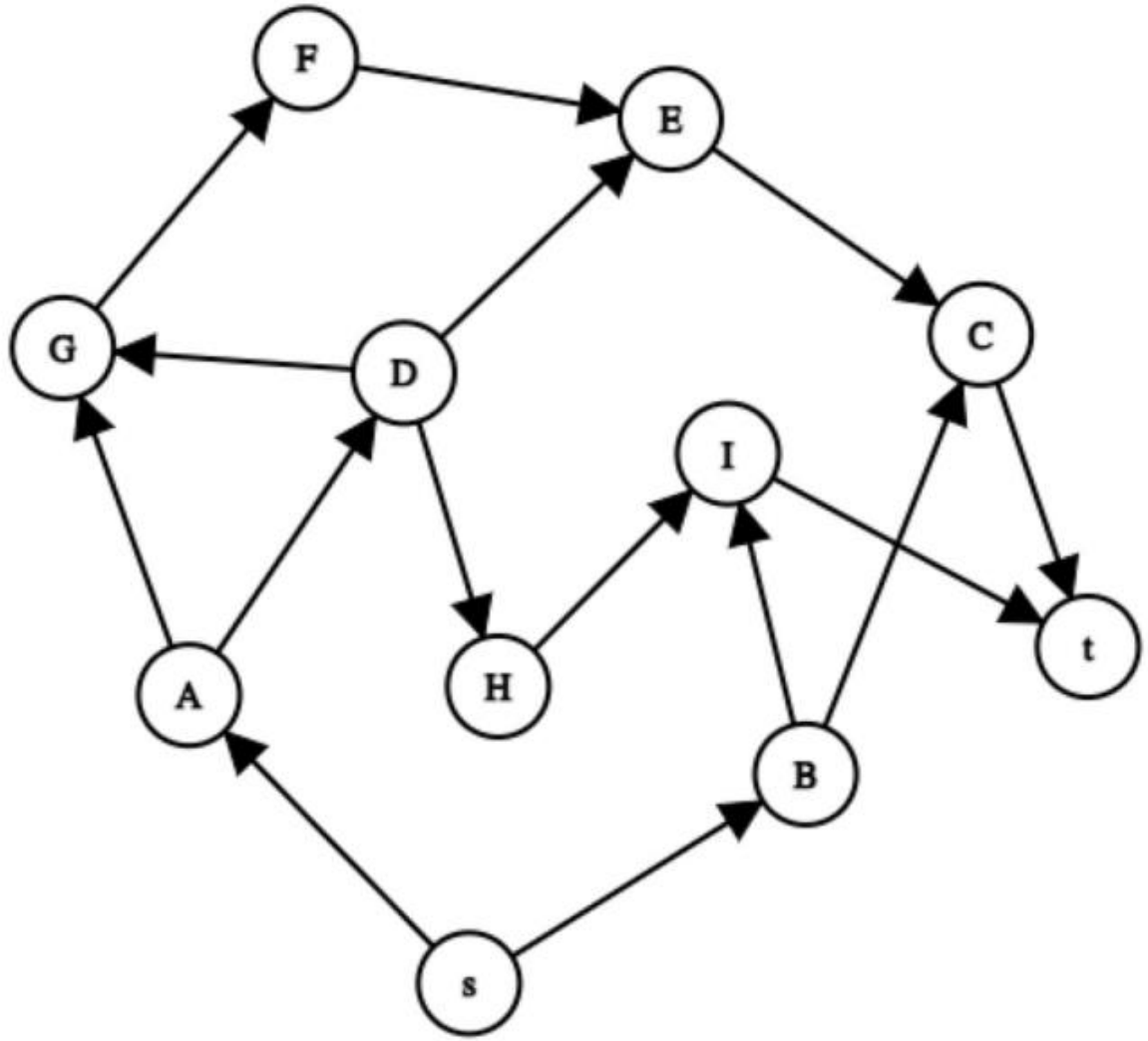
Breadth First Search Traversal:
- Traverse vertices 2 away
- No more unvisited vertices left, finalize



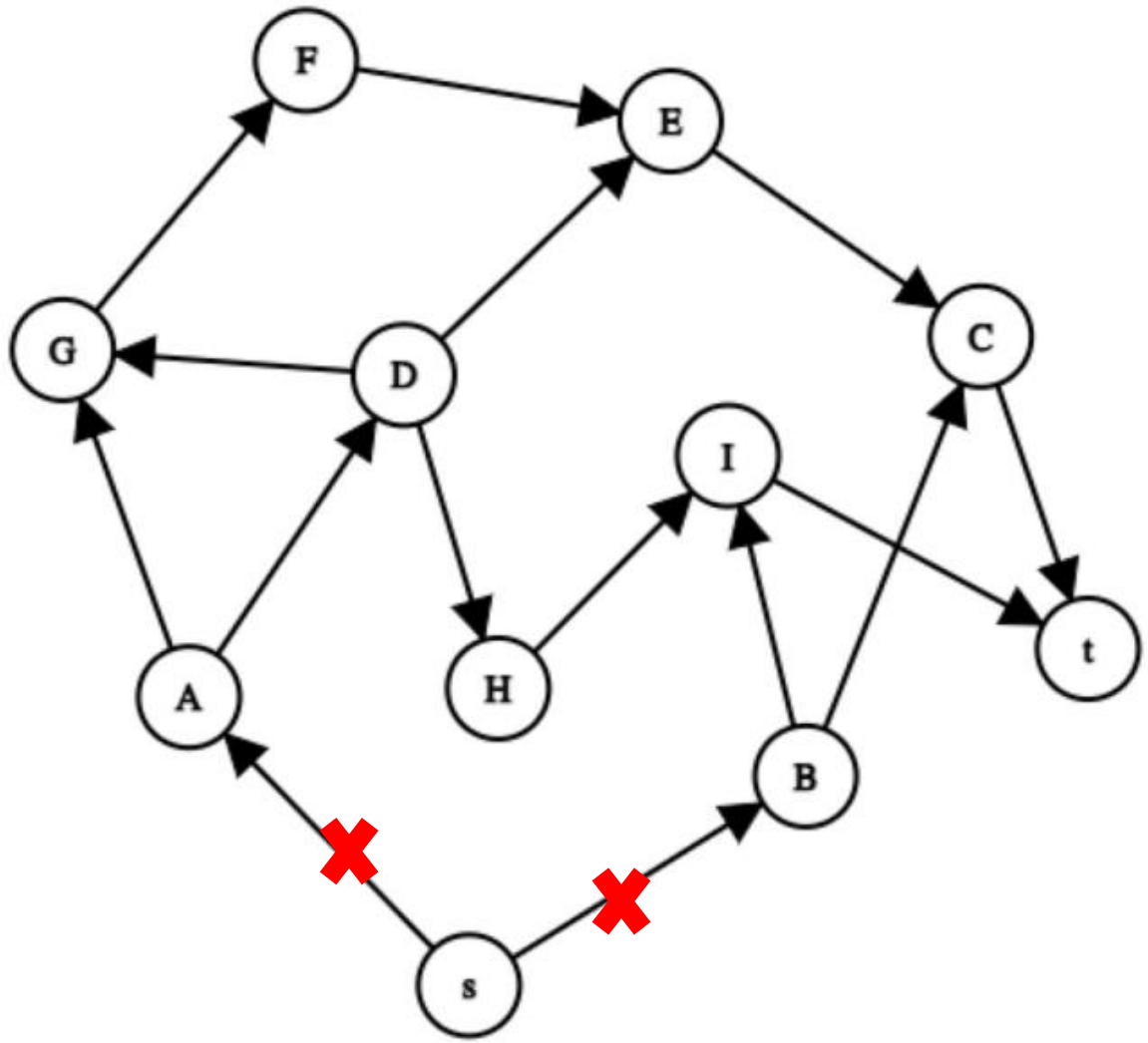
Question 5

Find a topological ordering of the graph in Figure 2.

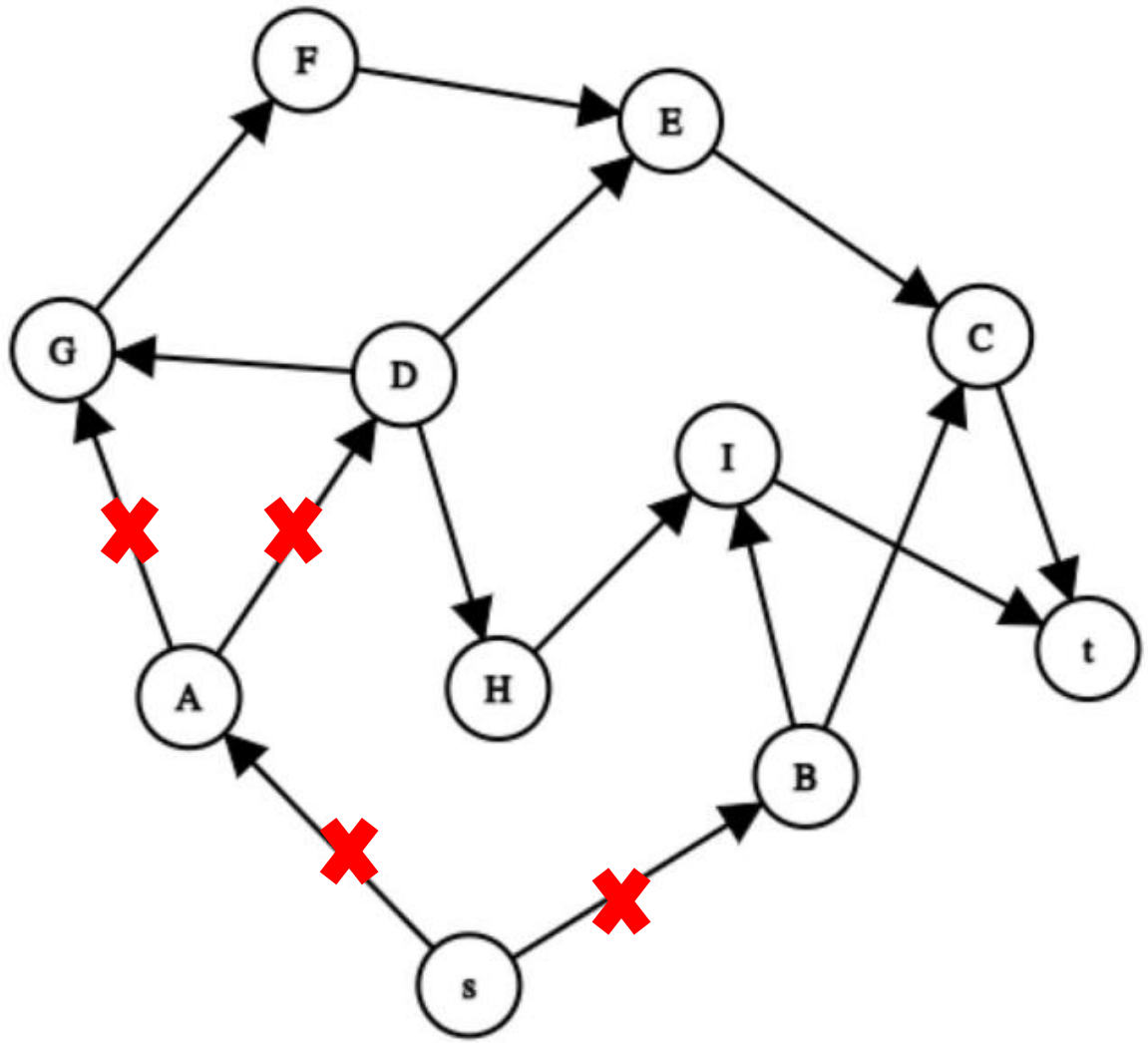
Topological Sort:
1 - Select any vertex with in-degree 0
2 - Print it out
3 - Remove it
4 - Repeat



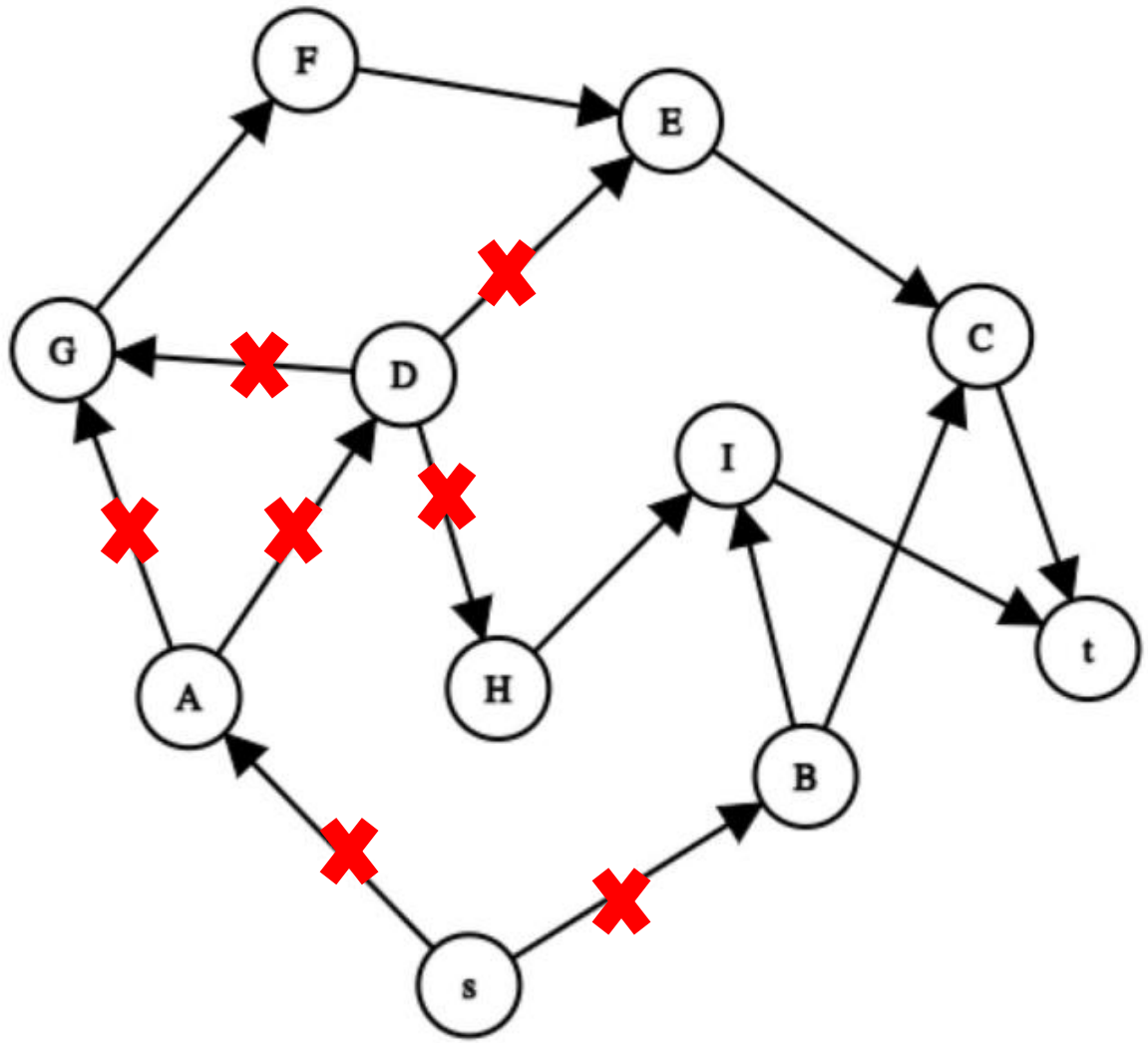
Topological Sort:
1 - Select s
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s



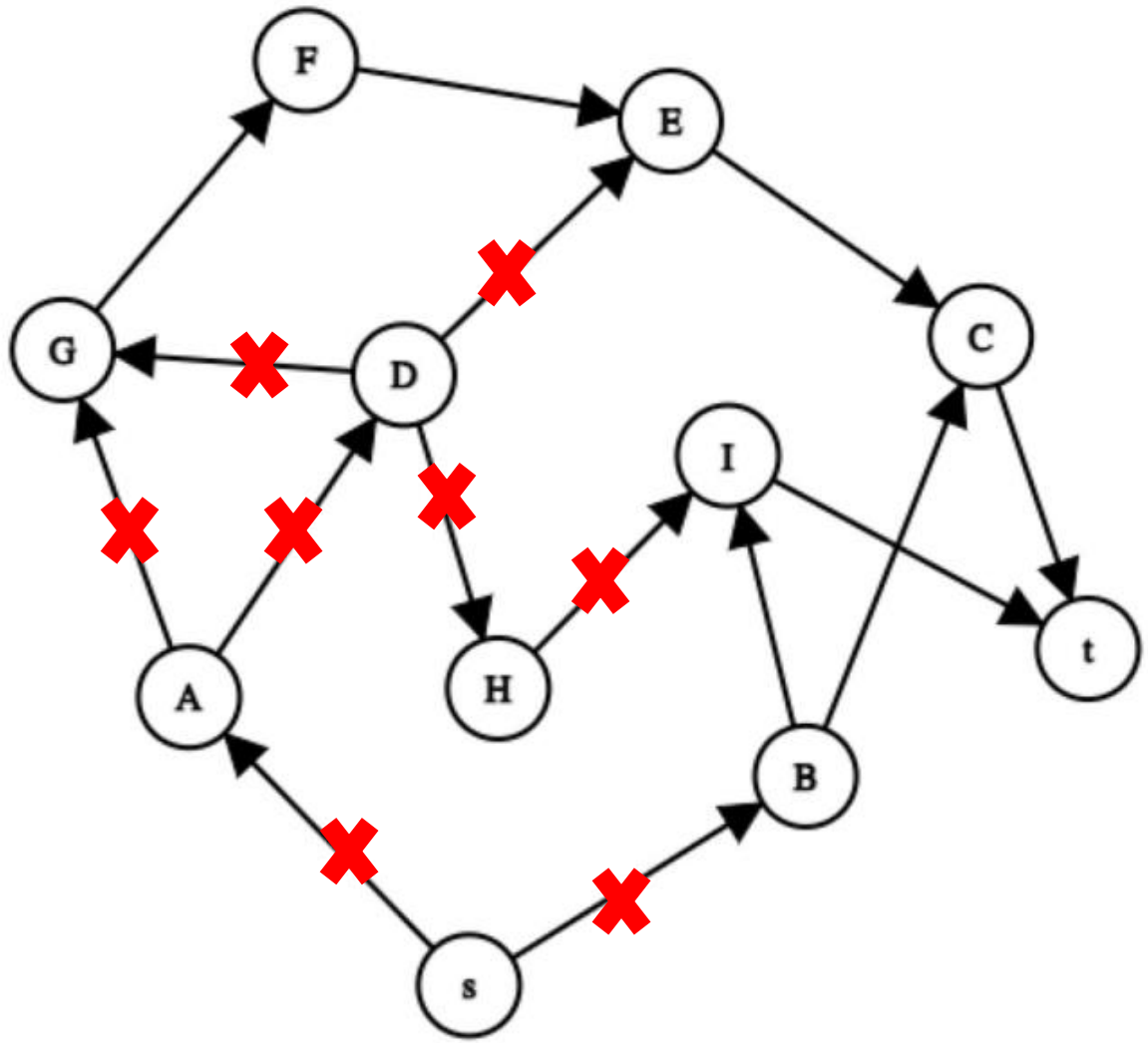
Topological Sort:
1 - Select A
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A



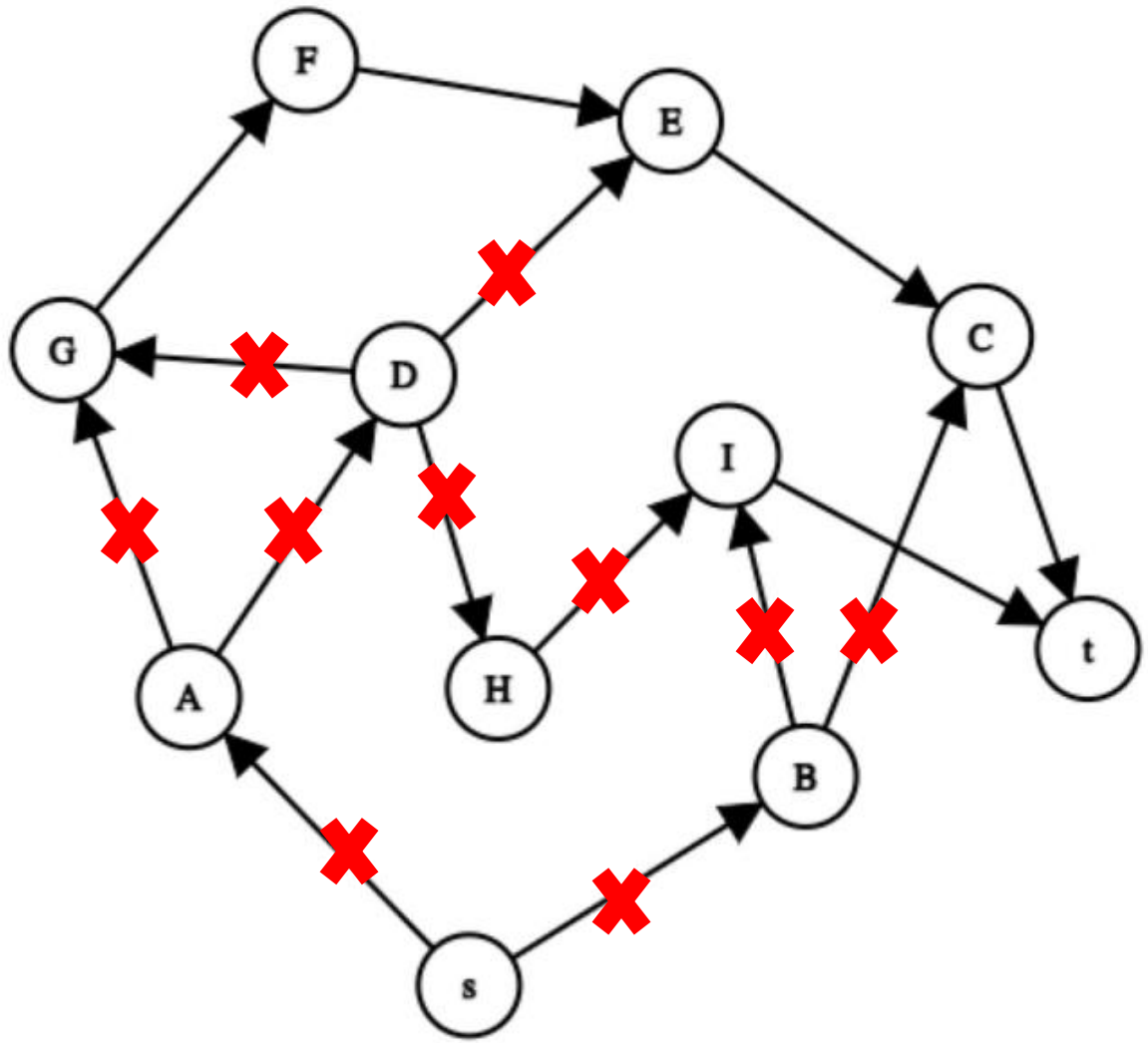
Topological Sort:
1 - Select D
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D



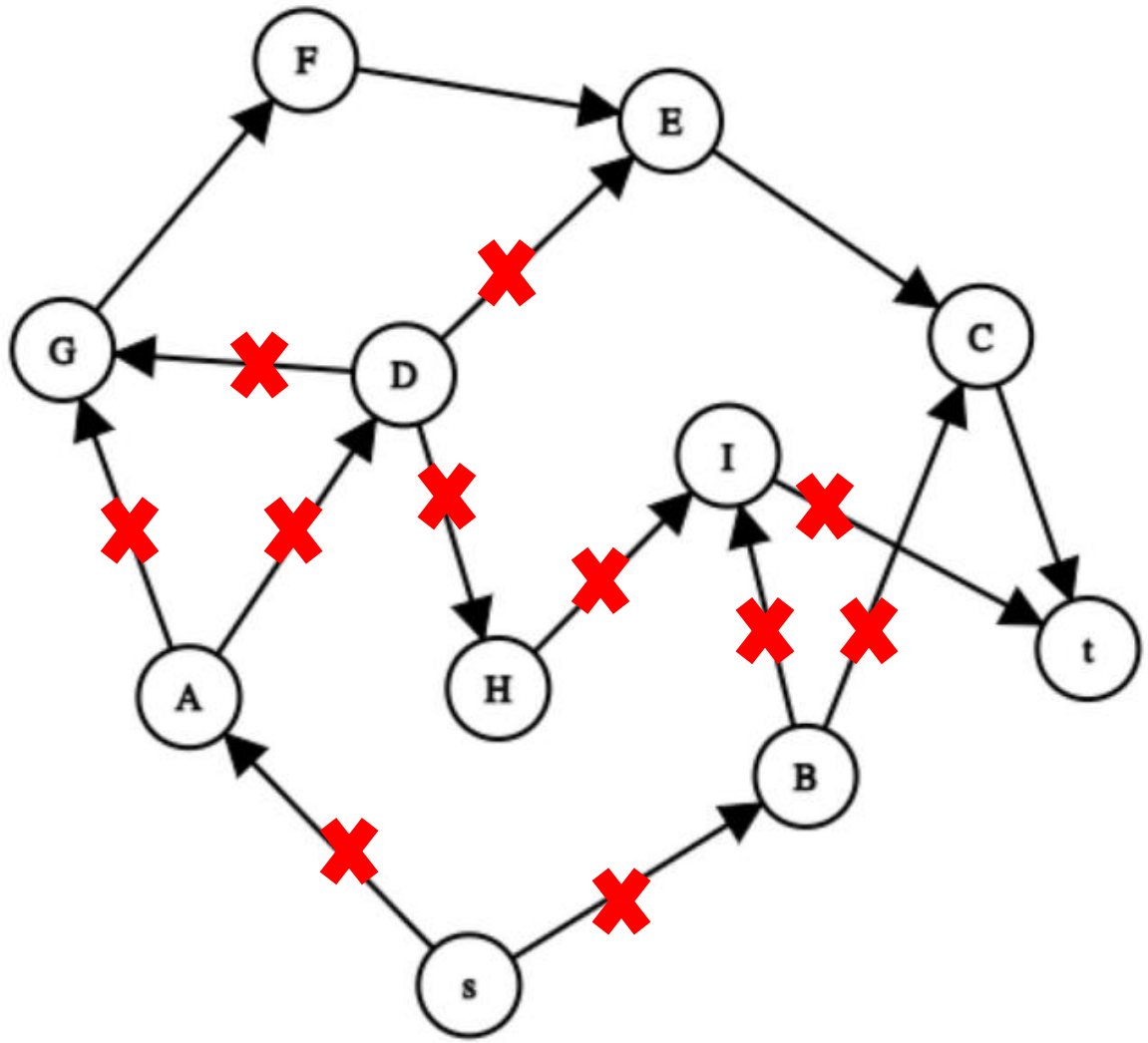
Topological Sort:
1 - Select H
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D H



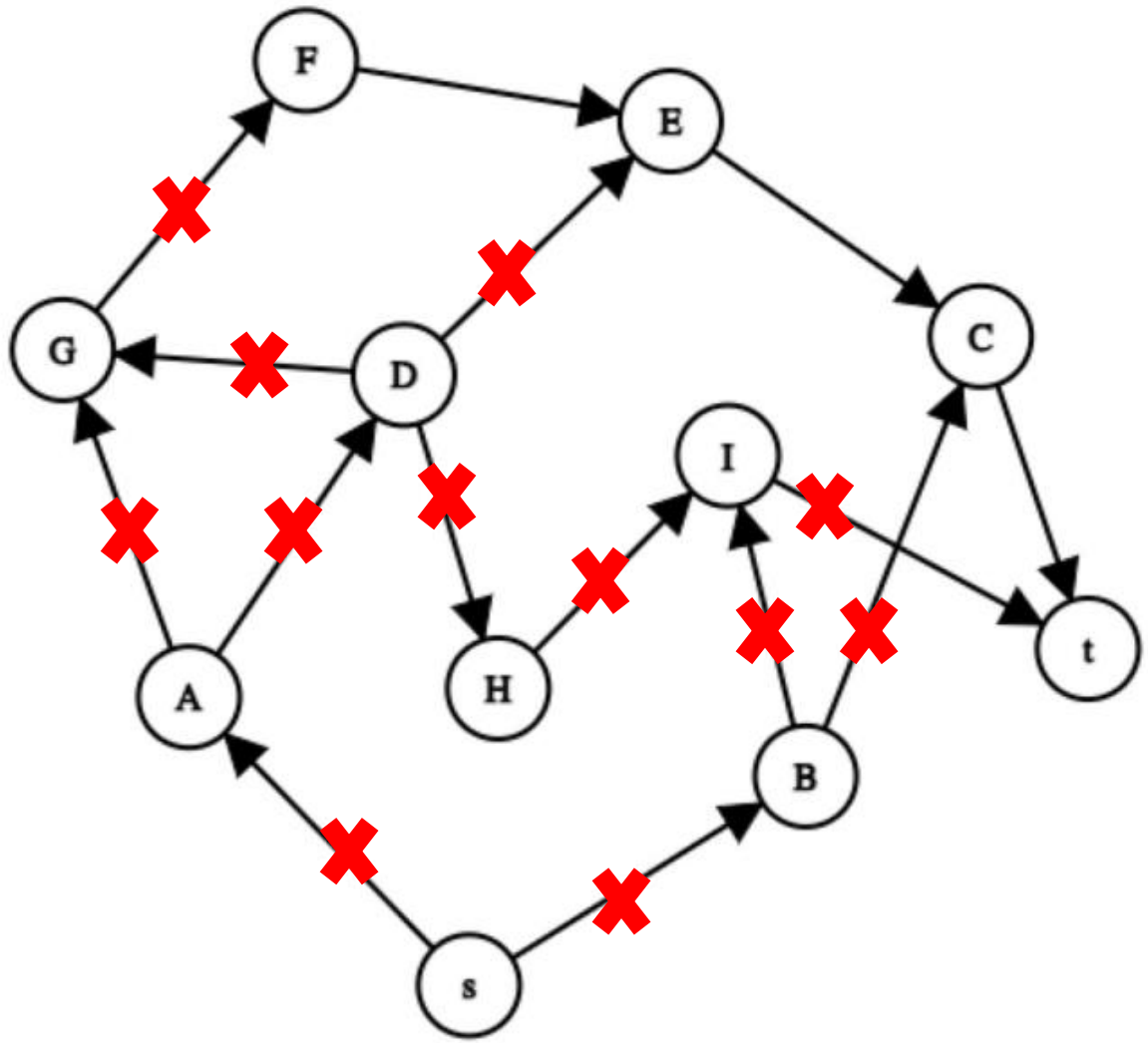
Topological Sort:
1 - Select B
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D H B



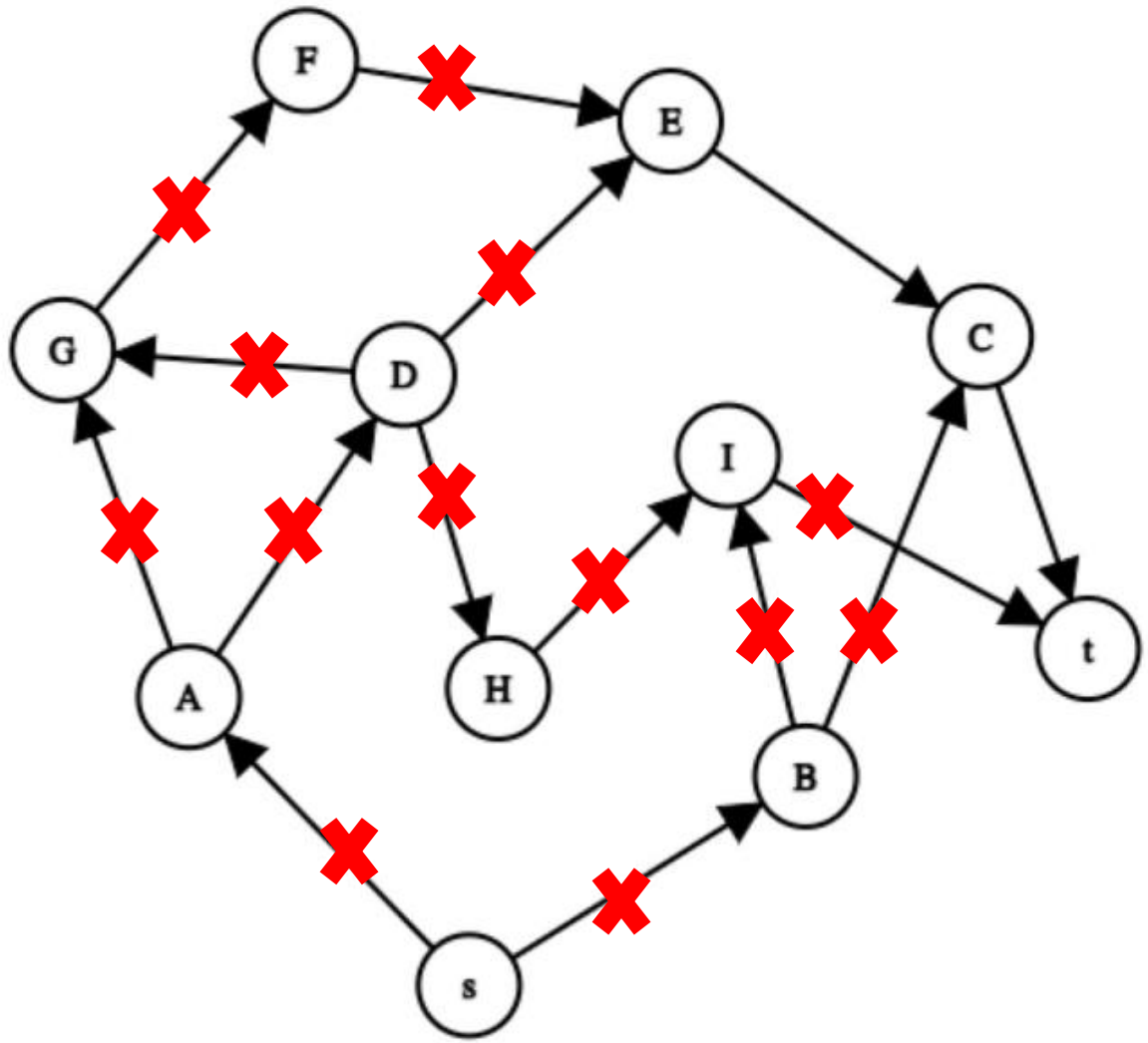
Topological Sort:
1 - Select I
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D H B I



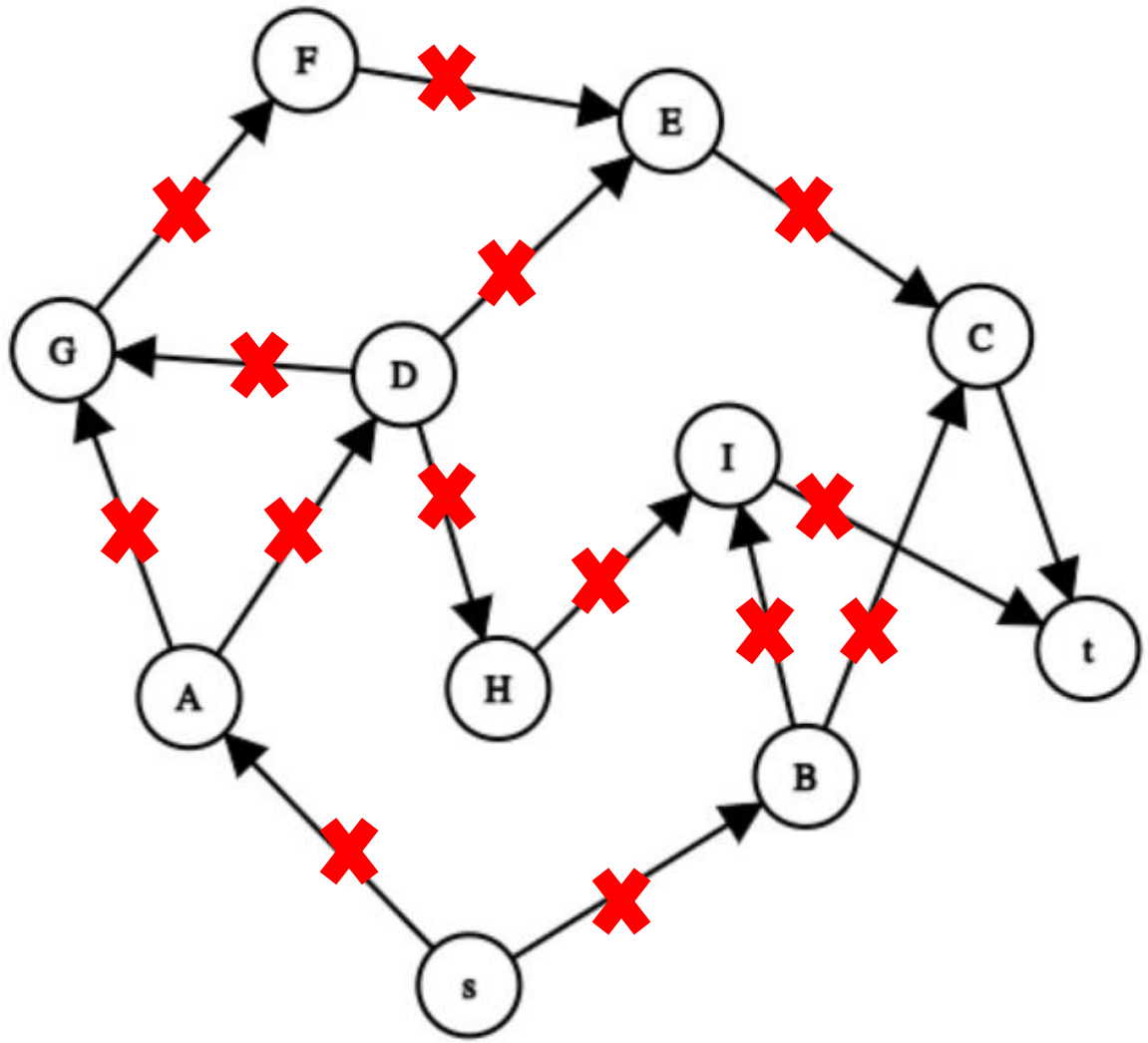
Topological Sort:
1 - Select G
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D H B I G



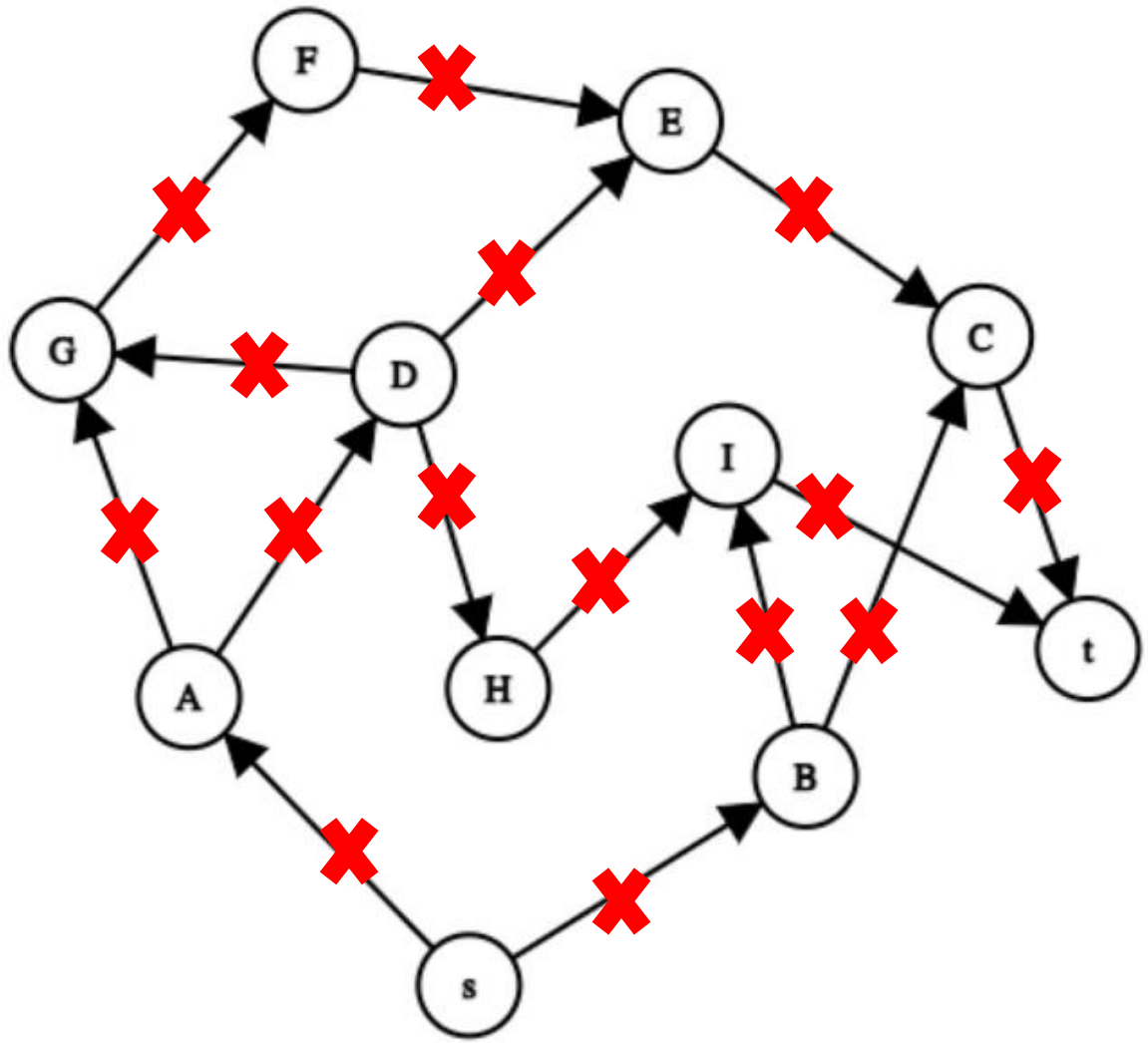
Topological Sort:
1 - Select F
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D H B I G F



Topological Sort:
1 - Select E
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D H B I G F E



Topological Sort:
1 - Select C
2 - Print it out
3 - Remove it
4 - Repeat
Current sequence:
s A D H B I G F E C



Topological Sort:
1 - Select t
2 - Print it out
3 - Remove it
4 - No more vertices left, finalize
Possible topological sort:
s A D H B I G F E C t