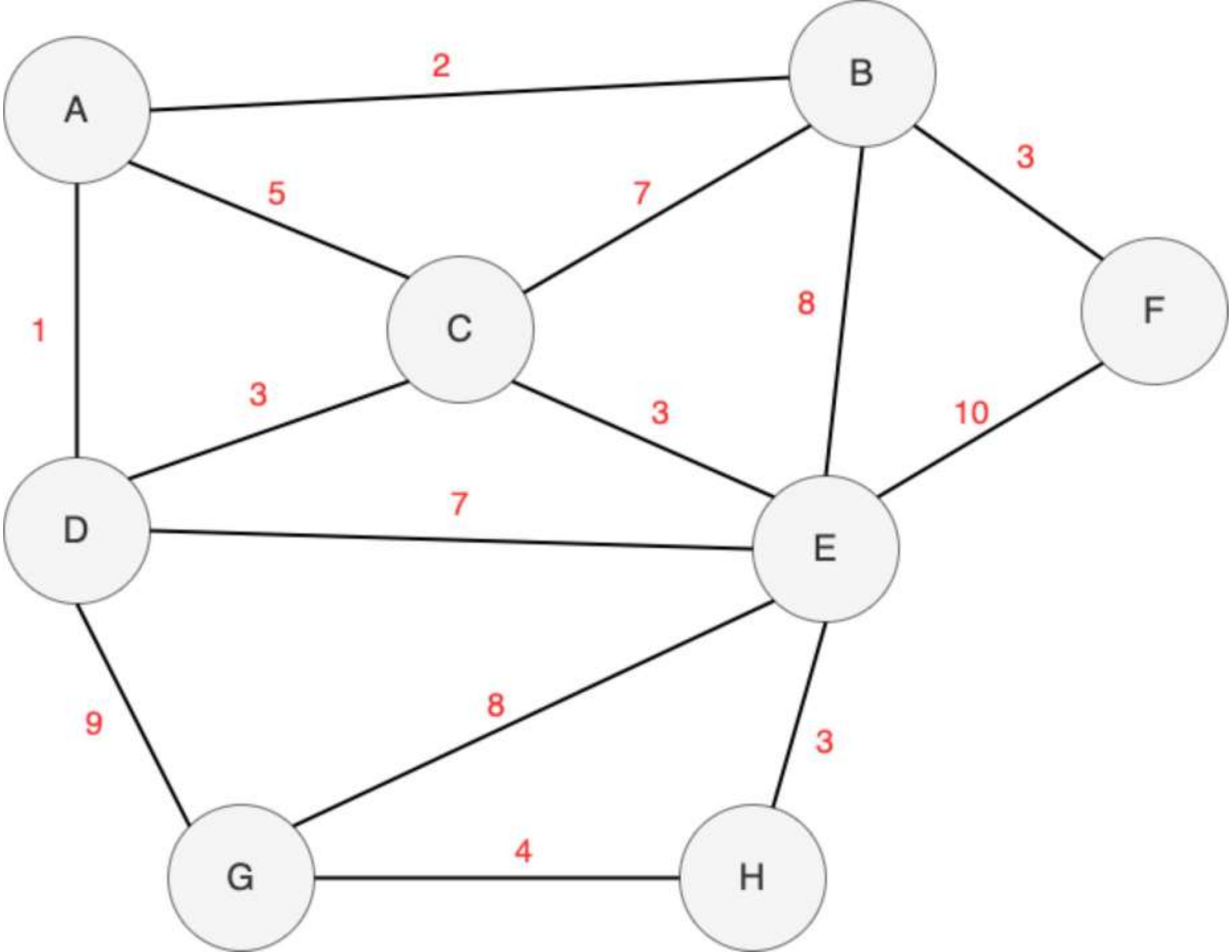
CS 300 HW5

Hamit Kartal 28404

QUESTION 1

Dijkstra Algorithm


A
B
C
D
E
F
G
H
2
5
1
7
3
8
3
3
10
7
9
8
3
4

Our starting vertex is E. So we can initialize our table.

	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3

Among unvisited vertexes, we choose the vertex with the minimum weight that is C with 3. When we are at C, we can go to A-B-D. For A, it is 3+5=8 better than NULL. So we change NULL to 8. For B, 7+3 =10 > 8. So we do not change B. For D, 3+3=6 < 7. So we change 7 to 6.

	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3
C	8	8	3	6	0	10	8	3

Among unvisited vertexes, we choose the vertex with the minimum weight that is H with 3. When we are at H, we can go to G. For G, it is 4+3=7 < 8. So we change 8 to 7.

	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3
C	8	8	3	6	0	10	8	3
H	8	8	3	6	0	10	7	3

Among unvisited vertexes, we choose the vertex with the minimum weight that is D with 6. When are at D, we can go to A-C-G. For A, it is 6+1=7 < 8. So we change 8 to 7. For C, it is 6+3=9 > 3. So we do not change C. For G, it is 6+9=15 > 7. So we do not change G.

	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3
C	8	8	3	6	0	10	8	3
H	8	8	3	6	0	10	7	3
D	7	8	3	6	0	10	7	3

Among unvisited vertexes, we choose the vertex with the minimum weight that is A with 7. When we are A, we can go to B-C-D. For B, it is 7+2=9 > 8. So we do not change B. For C, it is 7+5=12 > 3. So we do not change C. For D, it is 7+1=8 > 6. So we do not change D.

	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3
C	8	8	3	6	0	10	8	3
H	8	8	3	6	0	10	7	3
D	7	8	3	6	0	10	7	3
A	7	8	3	6	0	10	7	3

Among unvisited vertexes, we choose the vertex with the minimum weight that is G with 7. When we are A, we can go to D-H. For D, it is 7+9=16 > 6. So we do not change D. For H, it is 7+4=11 > 3. So we do not change H.

	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3
C	8	8	3	6	0	10	8	3
H	8	8	3	6	0	10	7	3
D	7	8	3	6	0	10	7	3
A	7	8	3	6	0	10	7	3
G	7	8	3	6	0	10	7	3

Among unvisited vertexes, we choose the vertex with the minimum weight that is B with 8. When we are B, we can go to A-C-F. For A, it is 8+2=10 > 7. So we do not change A. For C, it is 8+7=15 > 3. So we do not change C. For F, it is 8+3=11 > 10. So we do not change F.

	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3
C	8	8	3	6	0	10	8	3
H	8	8	3	6	0	10	7	3
D	7	8	3	6	0	10	7	3
A	7	8	3	6	0	10	7	3
G	7	8	3	6	0	10	7	3
B	7	8	3	6	0	10	7	3

Among unvisited vertexes, we choose the vertex with the minimum weight that is F with 10. When we are F, we can go to B. For B, it is 10+3=13 > 8. So we do not change B.
Since there are no left unvisited vertex; this is the final table.

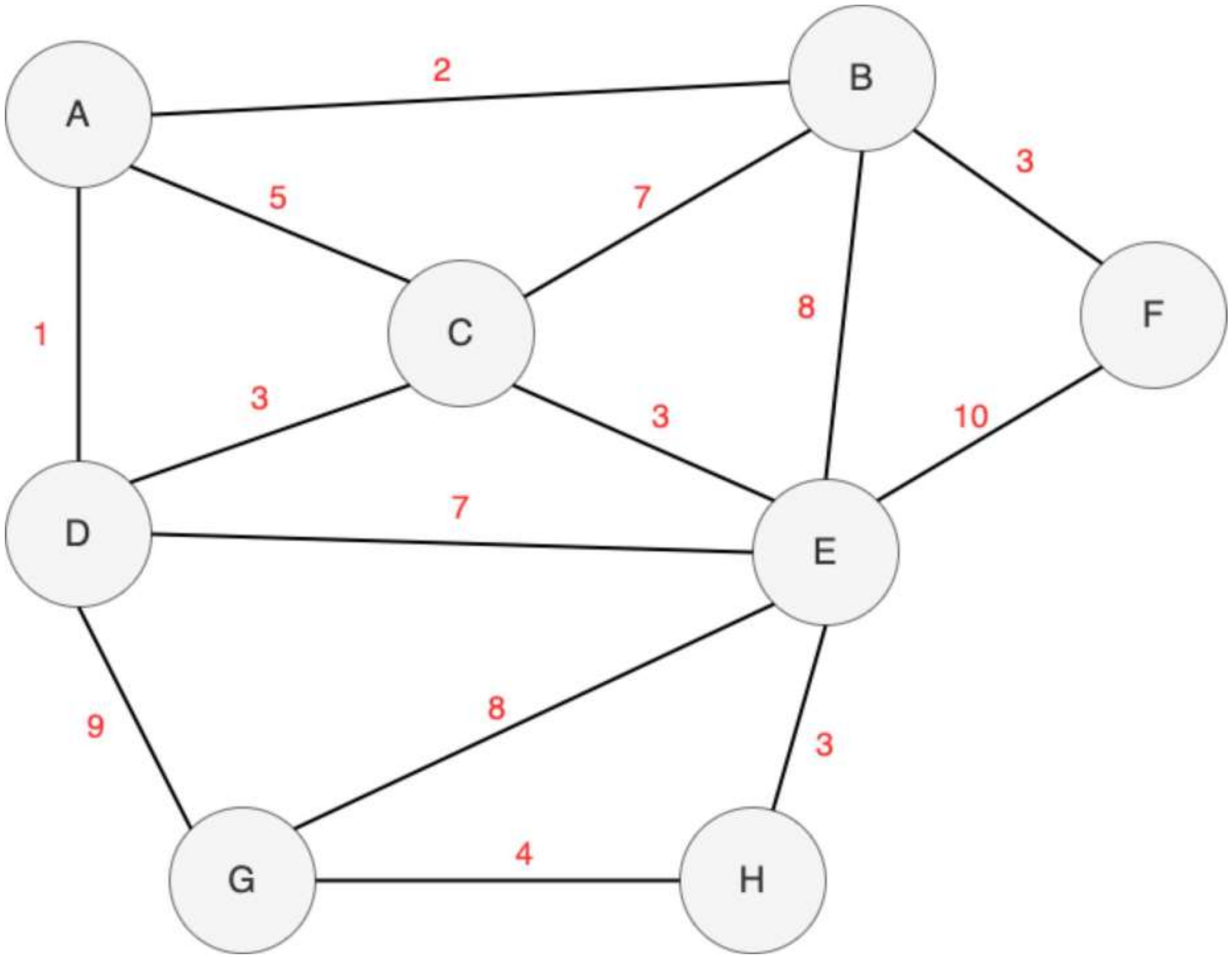
	A	B	C	D	E	F	G	H
E	NULL	8	3	7	0	10	8	3
C	8	8	3	6	0	10	8	3
H	8	8	3	6	0	10	7	3
D	7	8	3	6	0	10	7	3
A	7	8	3	6	0	10	7	3
G	7	8	3	6	0	10	7	3
B	7	8	3	6	0	10	7	3
F	7	8	3	6	0	10	7	3

Now, we can just consider the last row as shortest path from vertex E; as

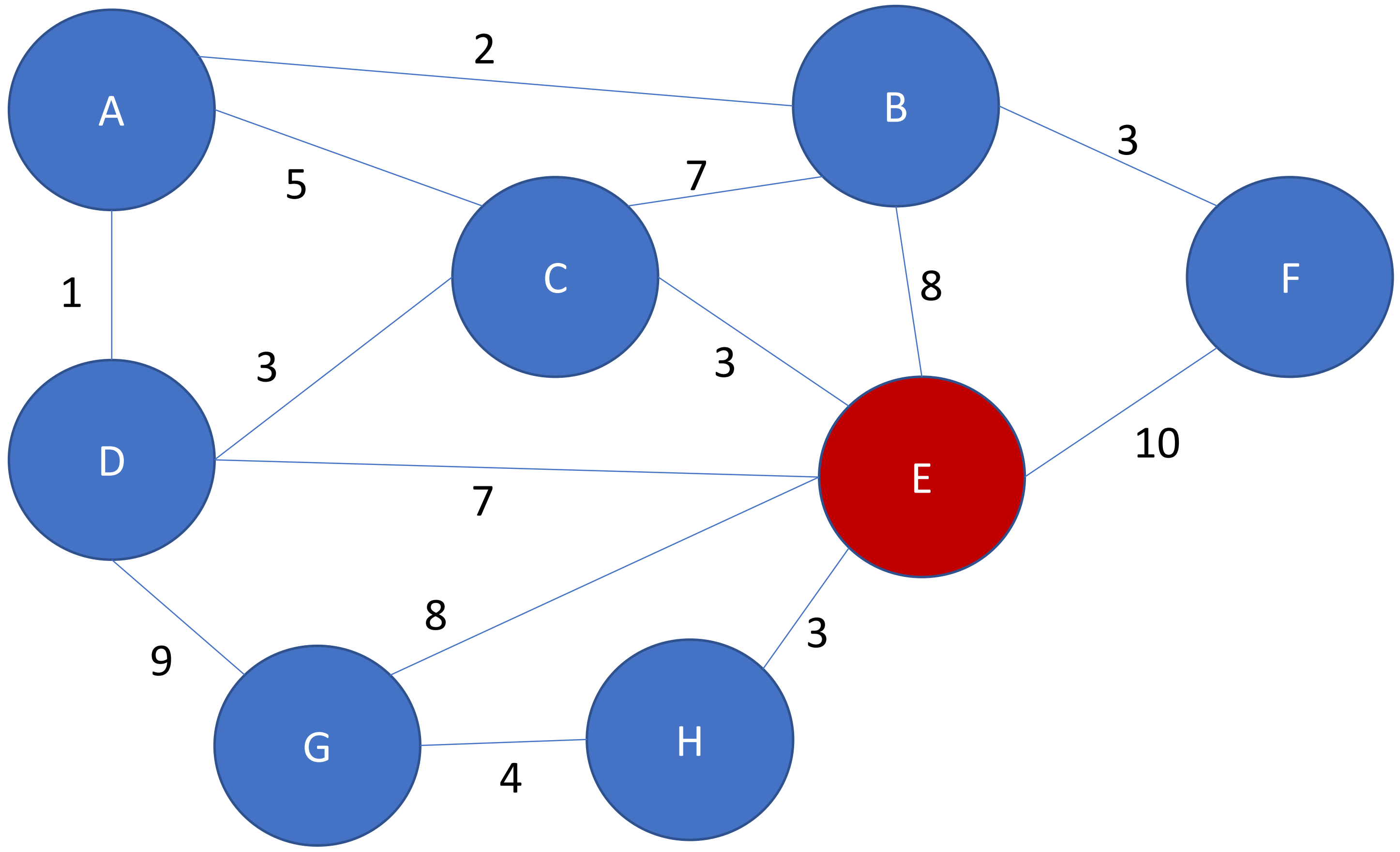
A	B	C	D	E	F	G	H
7	8	3	6	0	10	7	3

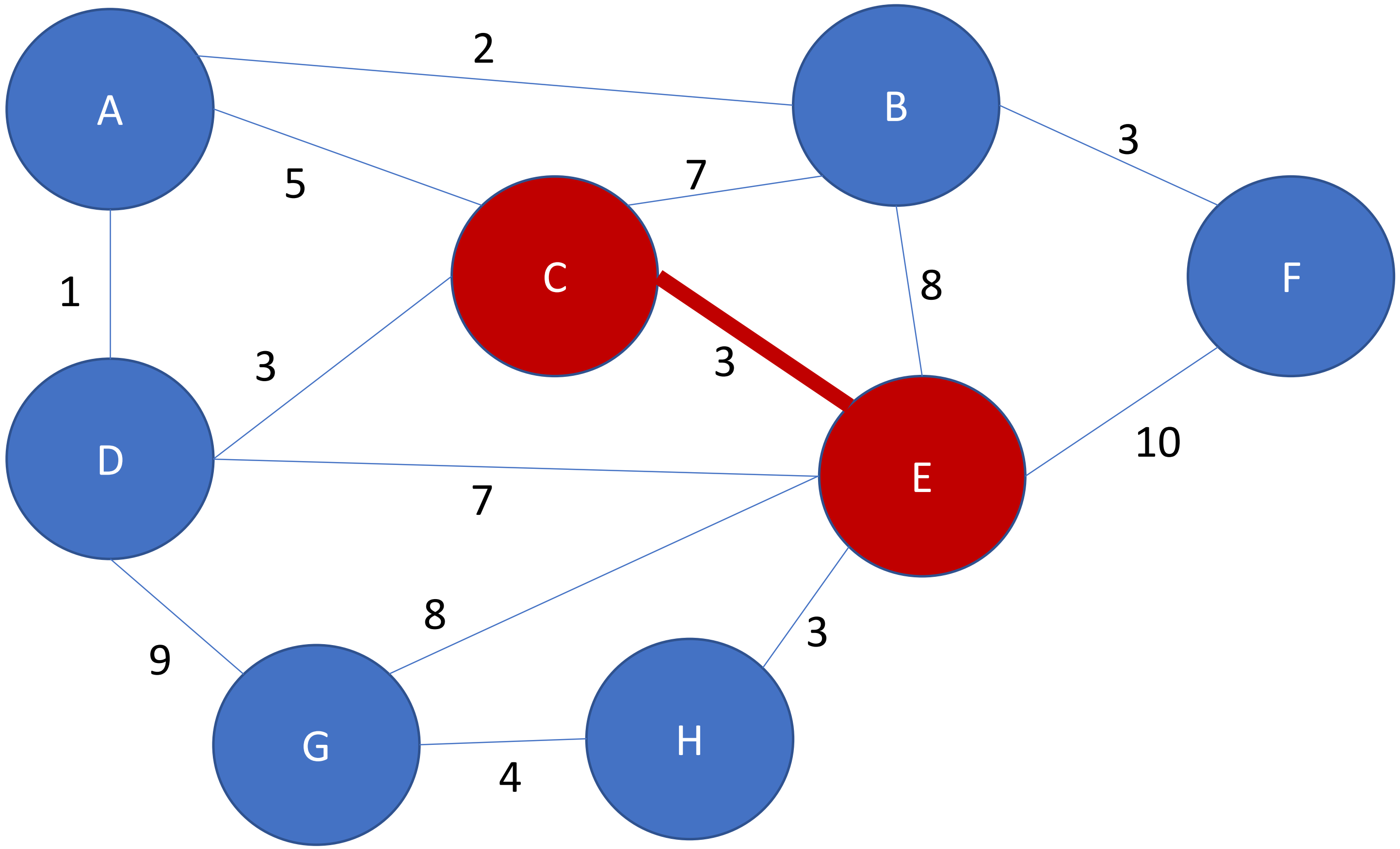
QUESTION 2

Prim Algorithm

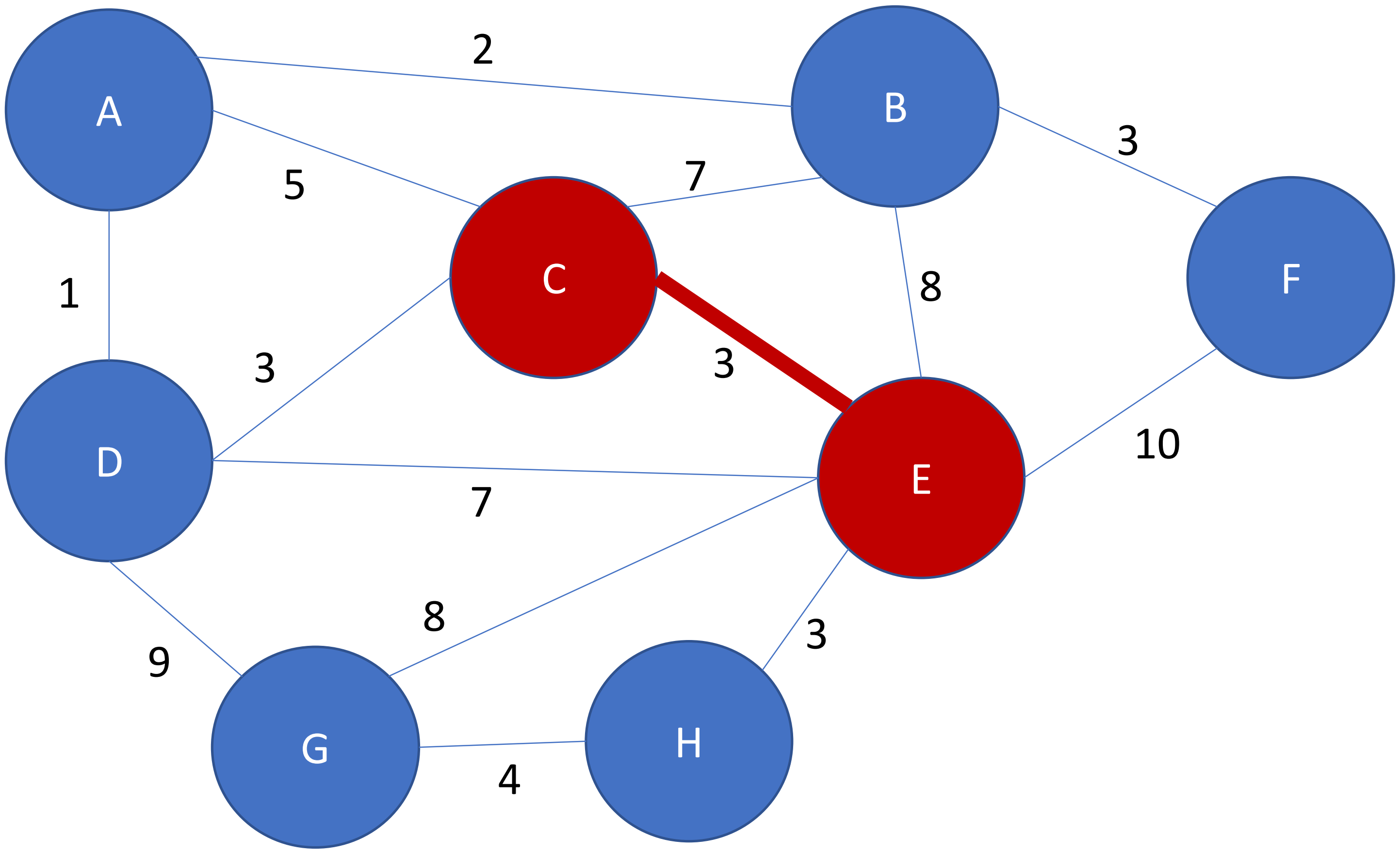


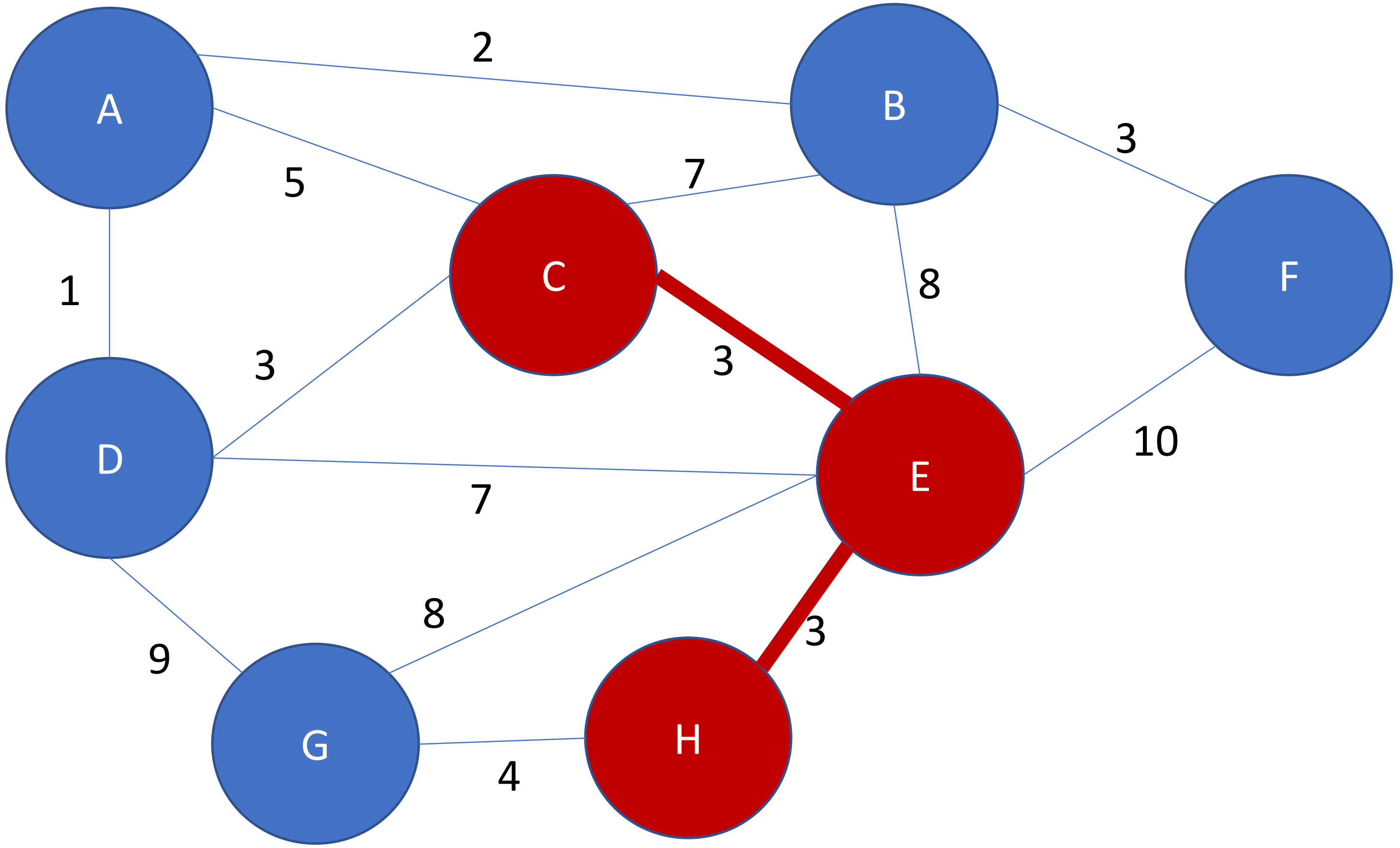
We choose E as our start vertex. Then we go through the edge with the minimum weight to an unvisited vertex from current MST. E-H and E-C are equally 3. So I choose C.



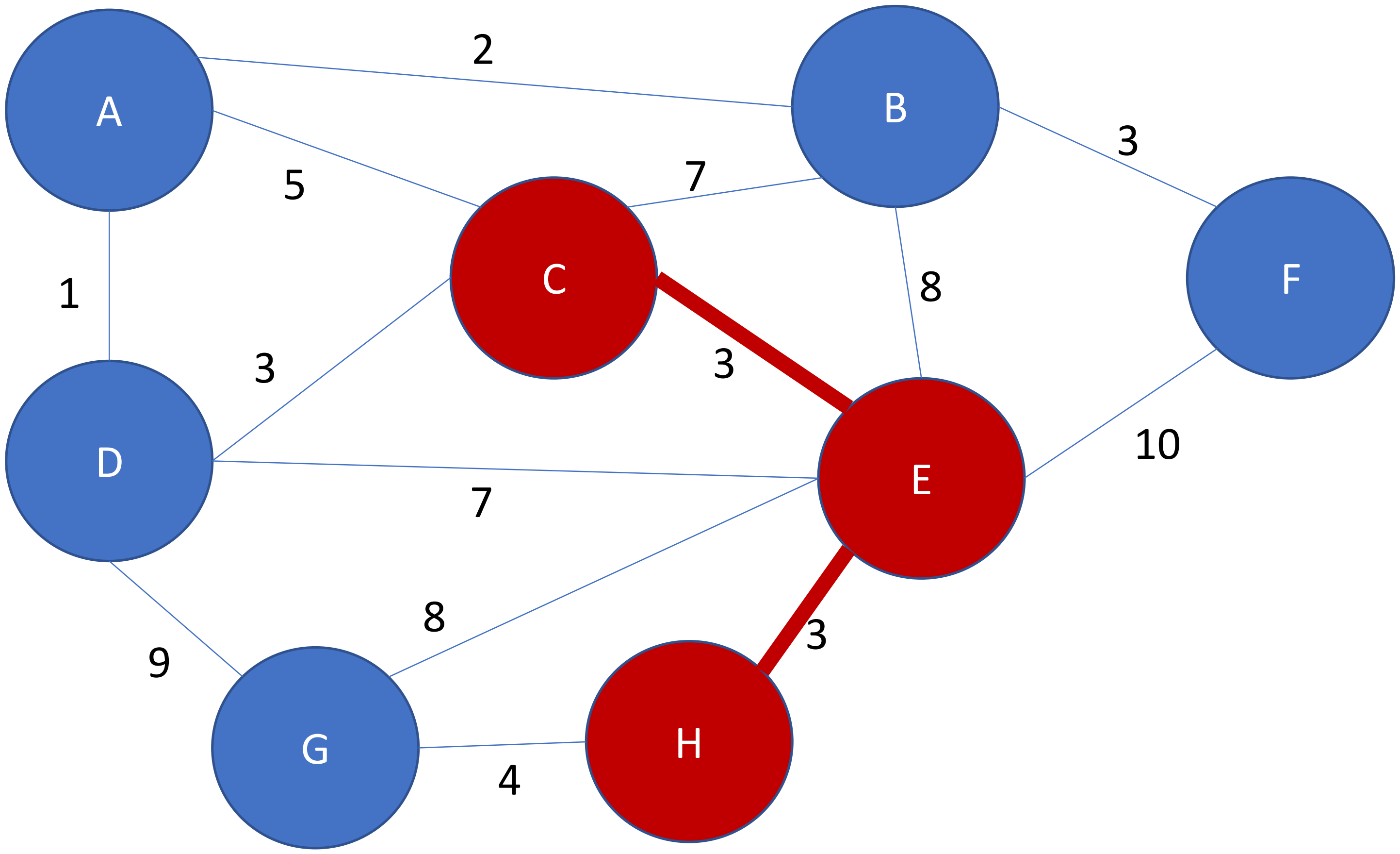

A
B
C
D
E
F
G
H
2
5
1
7
3
3
8
3
7
10
8
9
3
4

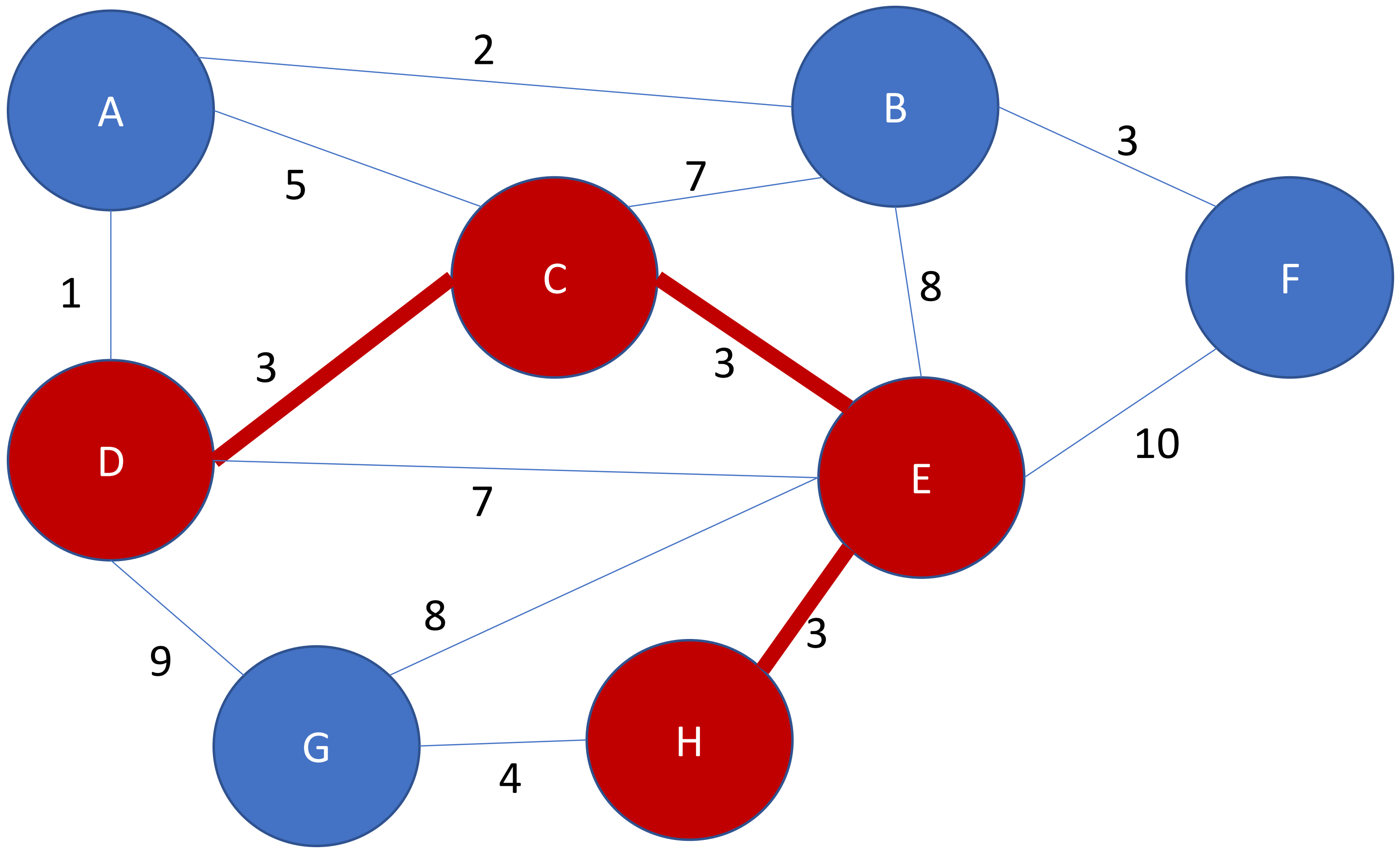
Now we go through the edge with minimum weight to an unvisited vertex from current MST. C-D and E-H are equally 3. So I choose H.



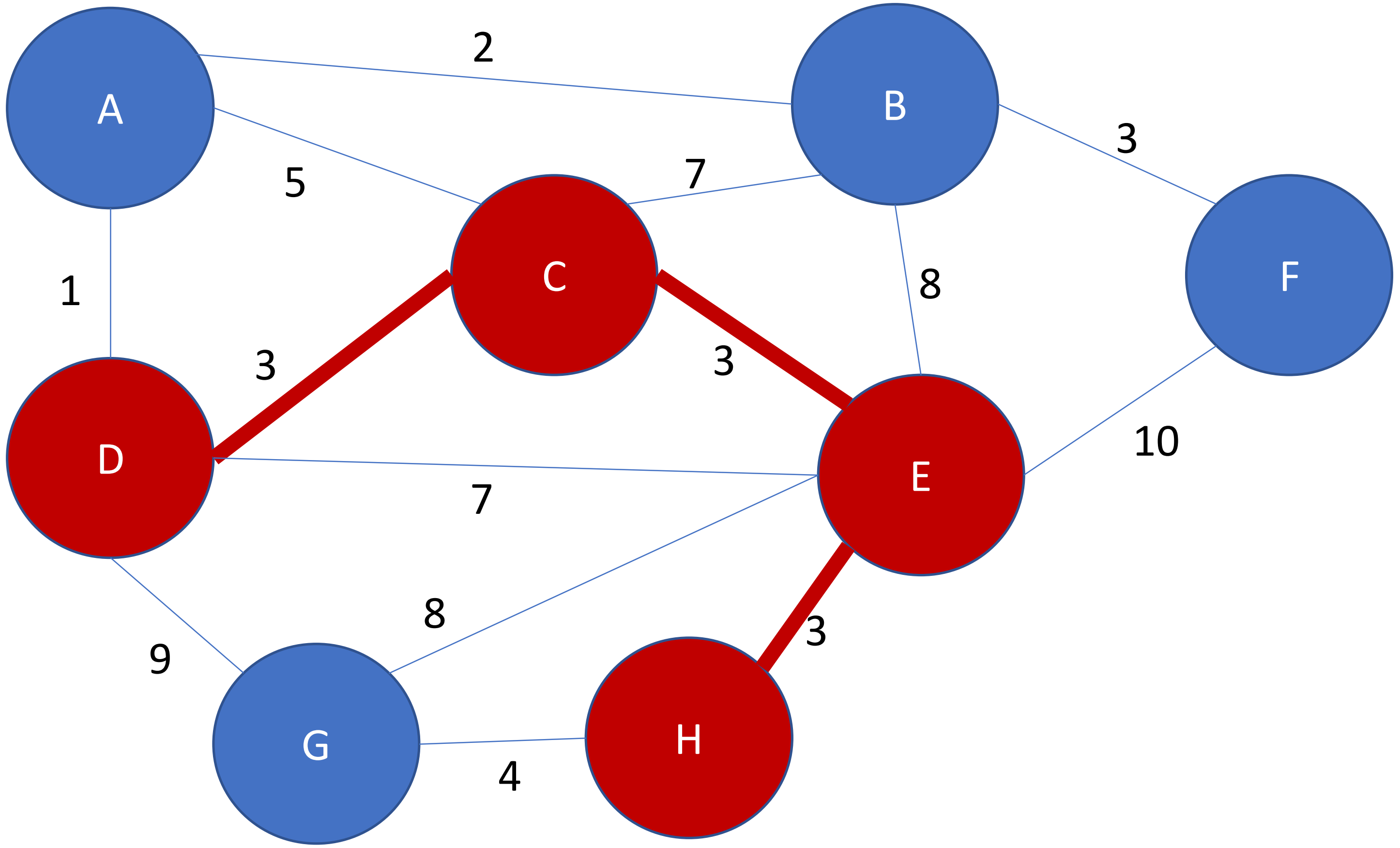

A
B
C
D
E
F
G
H
2
5
7
3
1
8
3
3
10
7
8
9
3
4

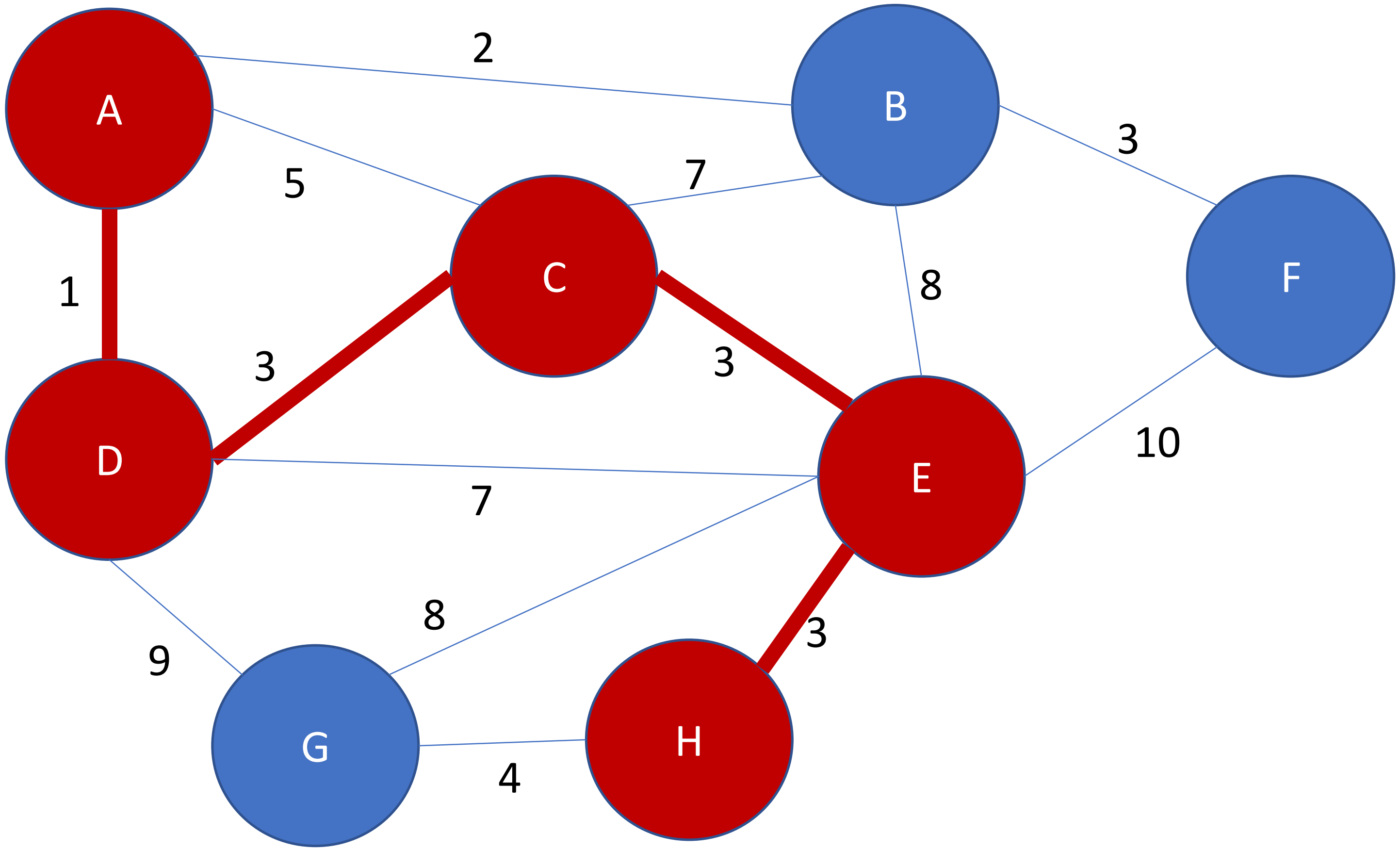
Now we go through the edge with the minimum weight to an unvisited vertex from current MST. C-D has the minimum weight.



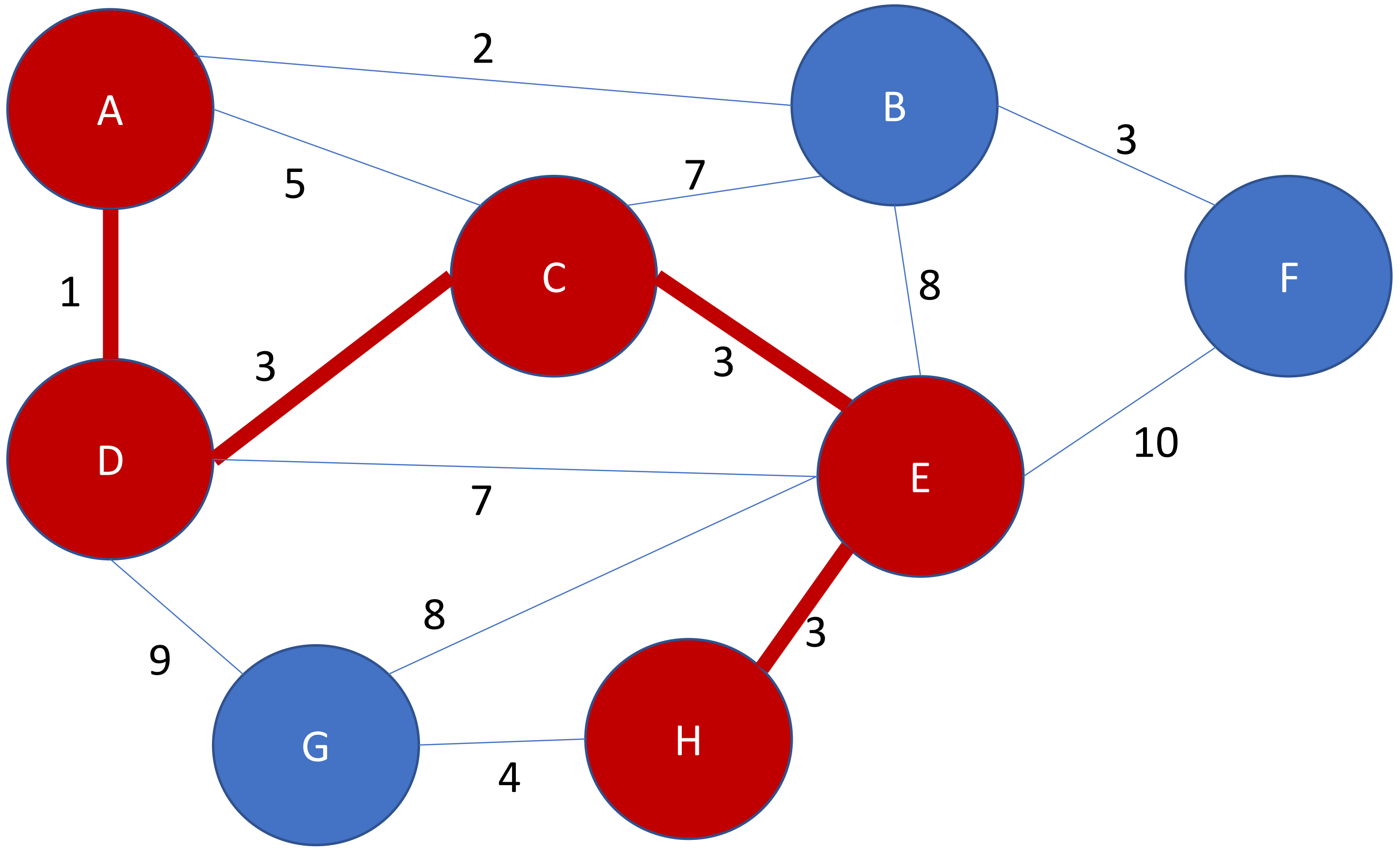

A
B
C
D
E
F
G
H
2
5
7
3
1
8
3
3
10
7
8
9
3
4

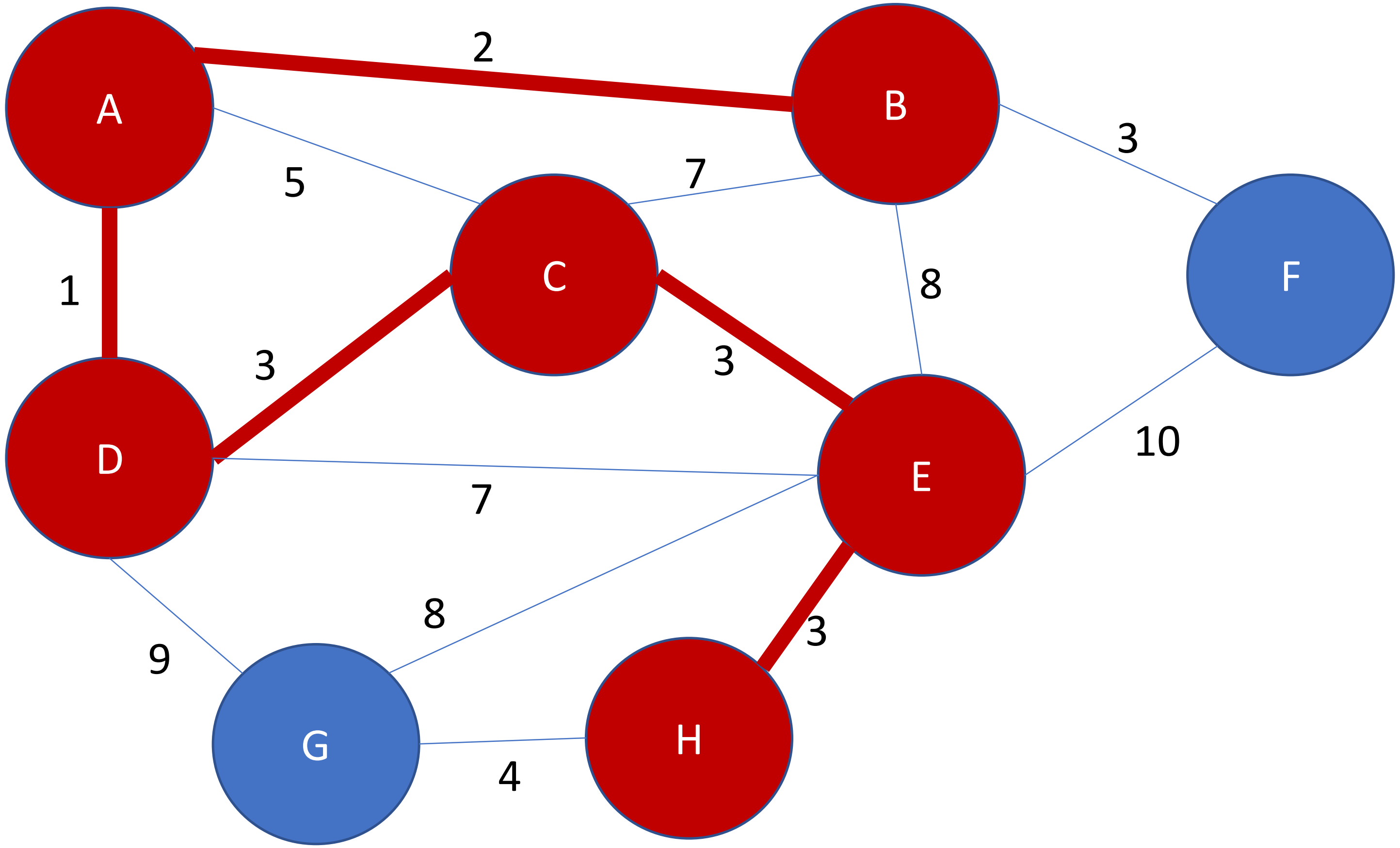
Now we go through the edge with the minimum weight to an unvisited vertex from current MST. A-D has the minimum weight.



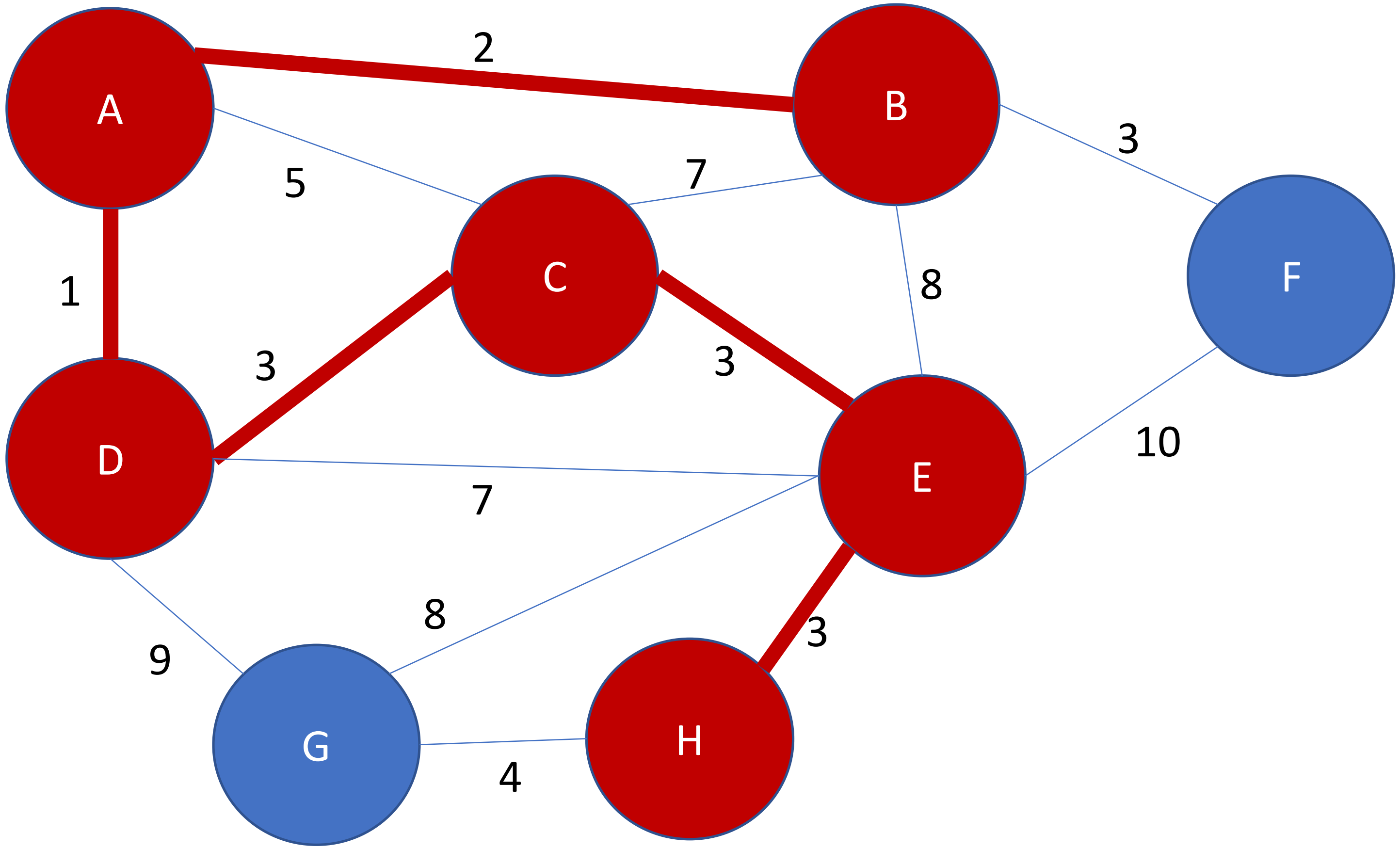

A
B
C
D
E
F
G
H
2
5
7
3
8
1
3
3
10
7
8
9
3
4

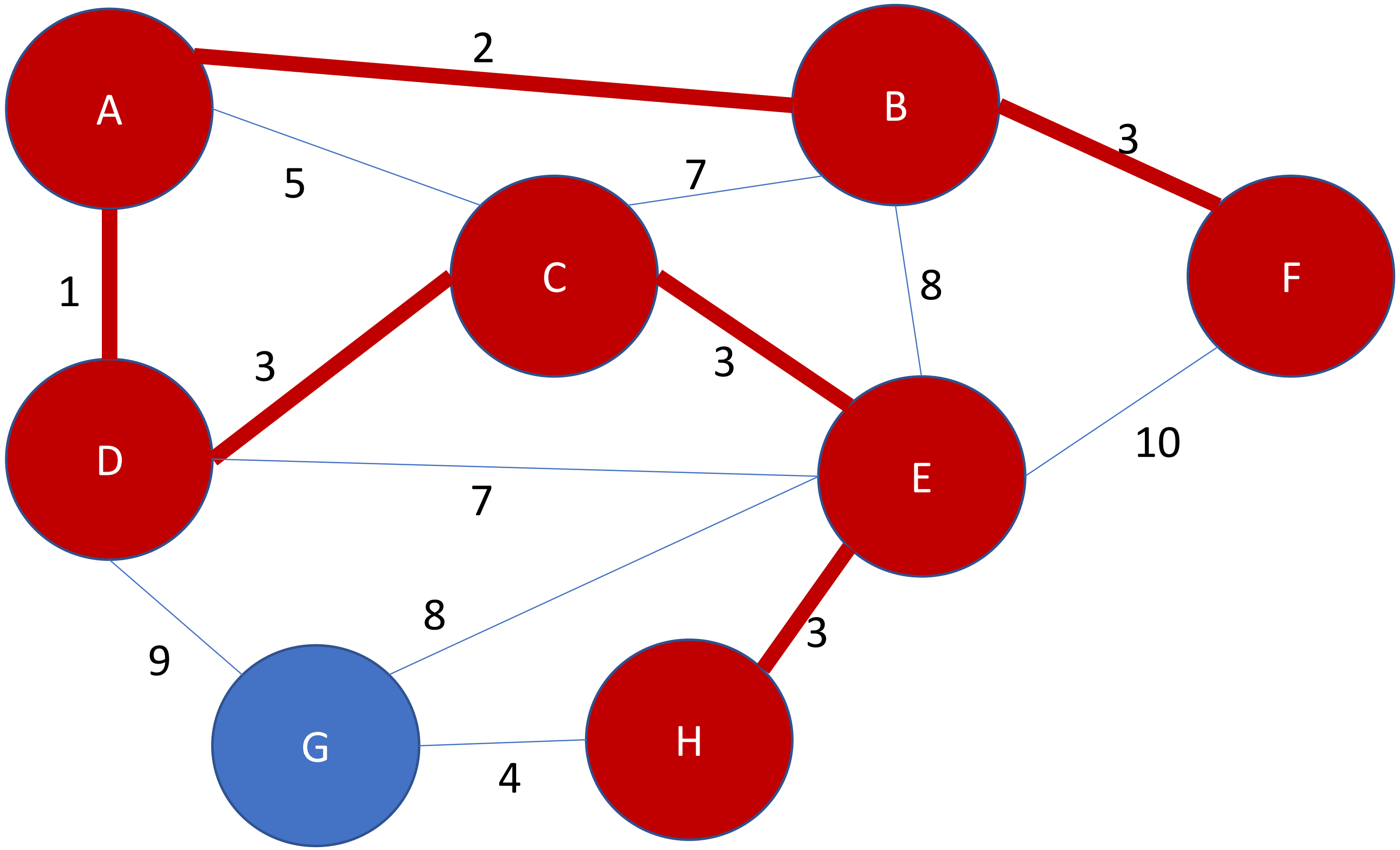
Now we go through the edge with the minimum weight to an unvisited vertex from current MST. A-B has the minimum weight.




A
B
C
D
E
F
G
H
2
5
7
3
1
3
3
8
10
7
8
9
3
4

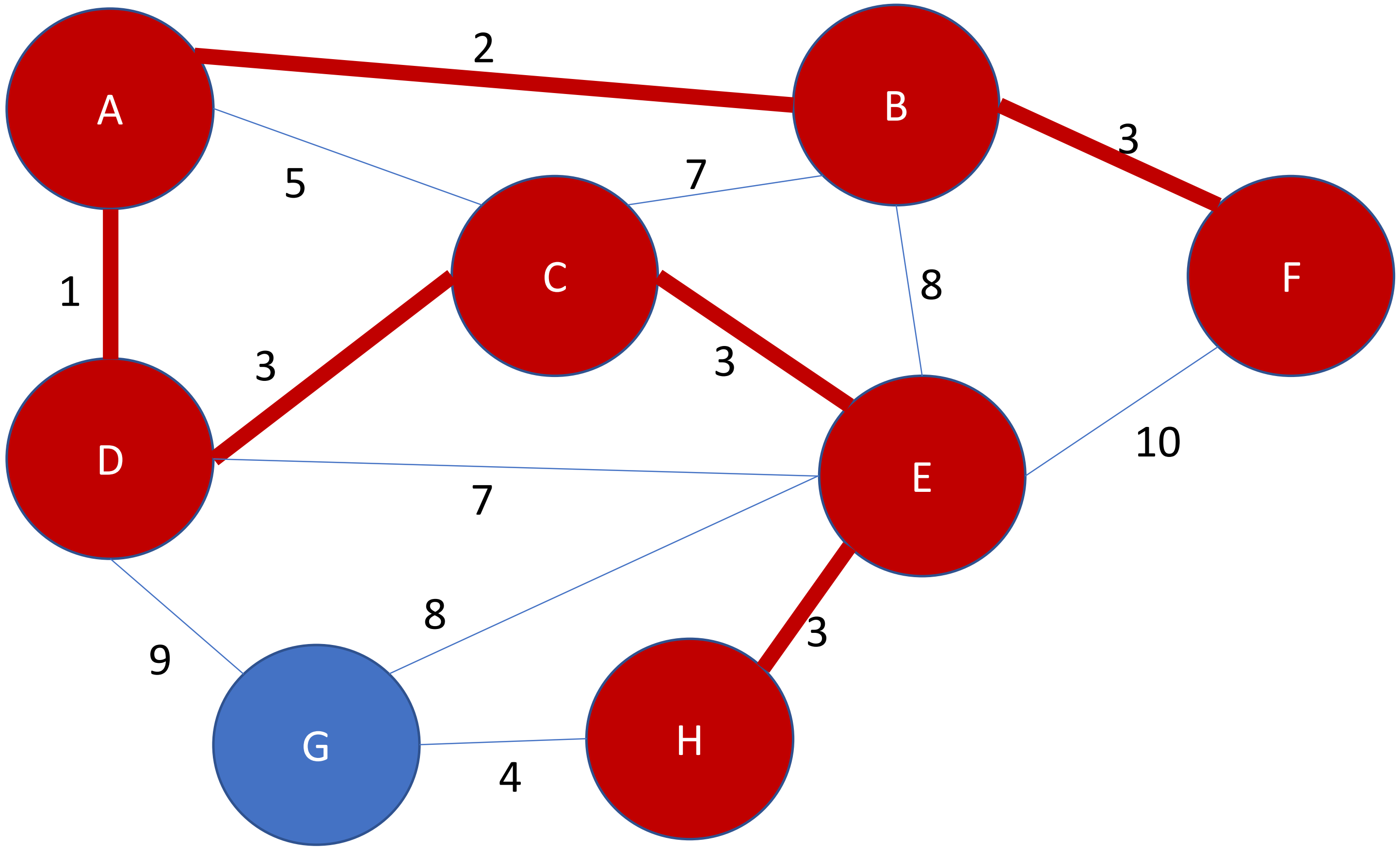
Now we go through the edge with the minimum weight to an unvisited vertex from current MST. B-F has the minimum weight.



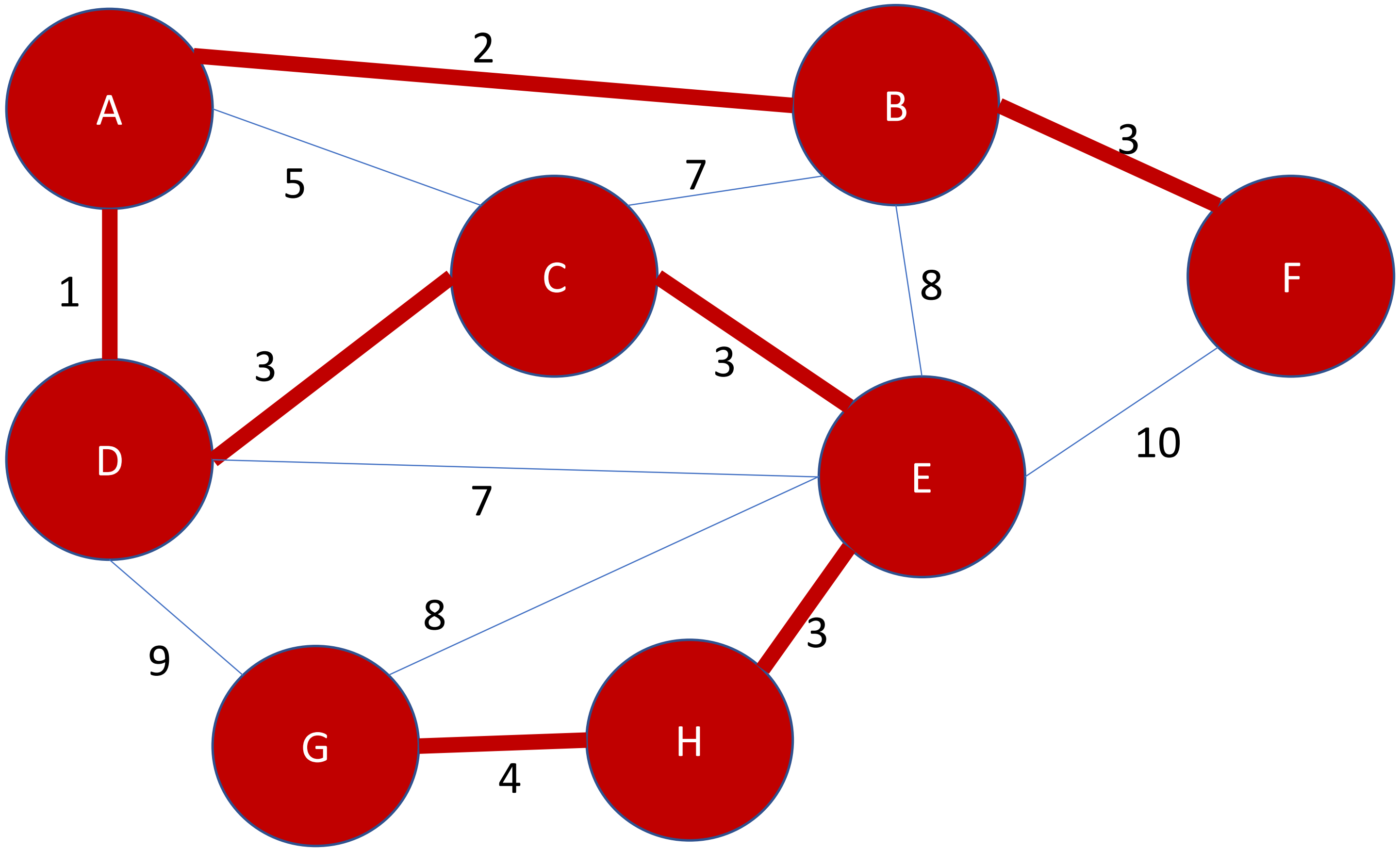


A
B
C
D
E
F
G
H
2
5
7
3
1
3
3
8
10
7
8
9
3
4

Sine the vertex G is the only vertex not spanned by current MST, we should consider the vertex G now.
To go to the vertex G, the H-G edge has the minimum weight.

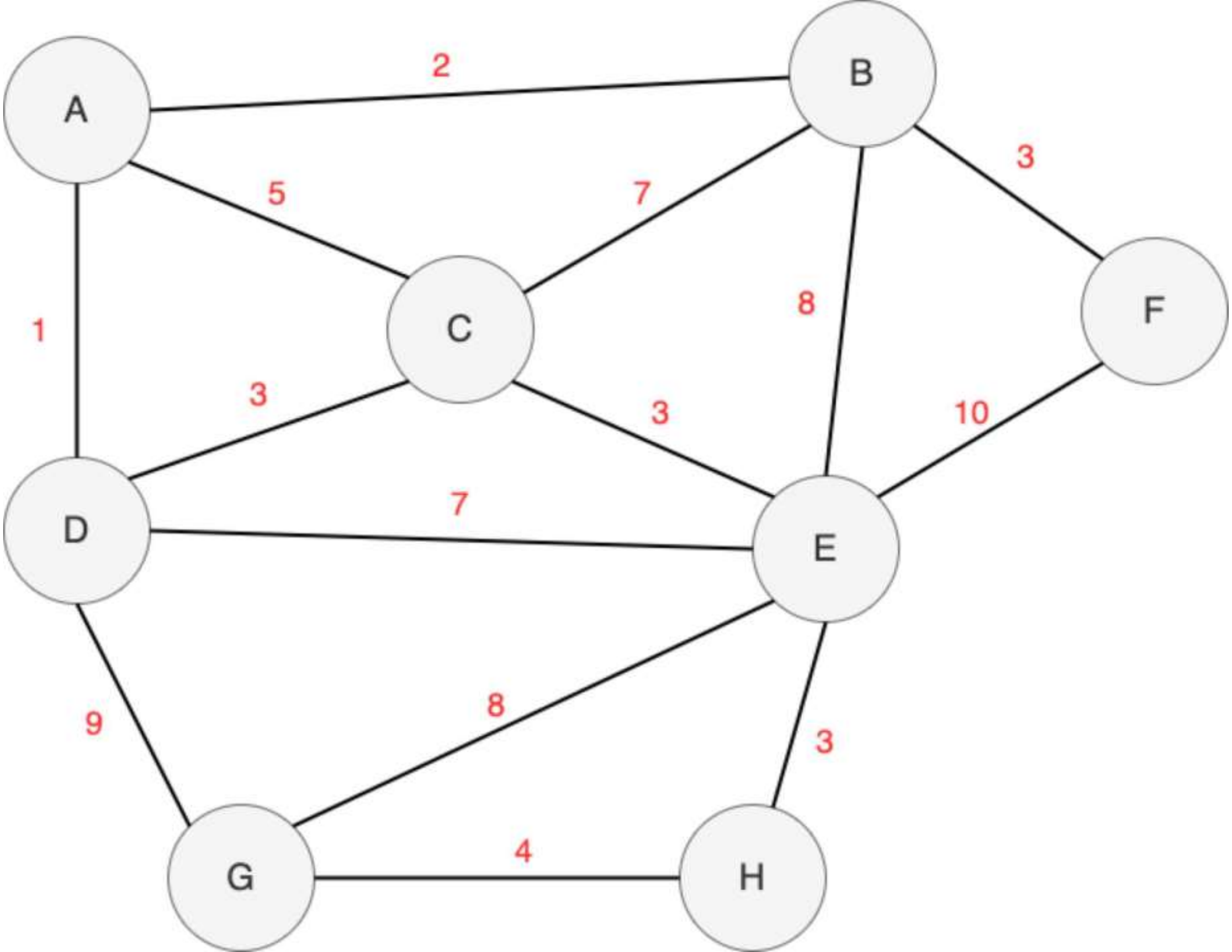


And here is the final MST:



QUESTION 3

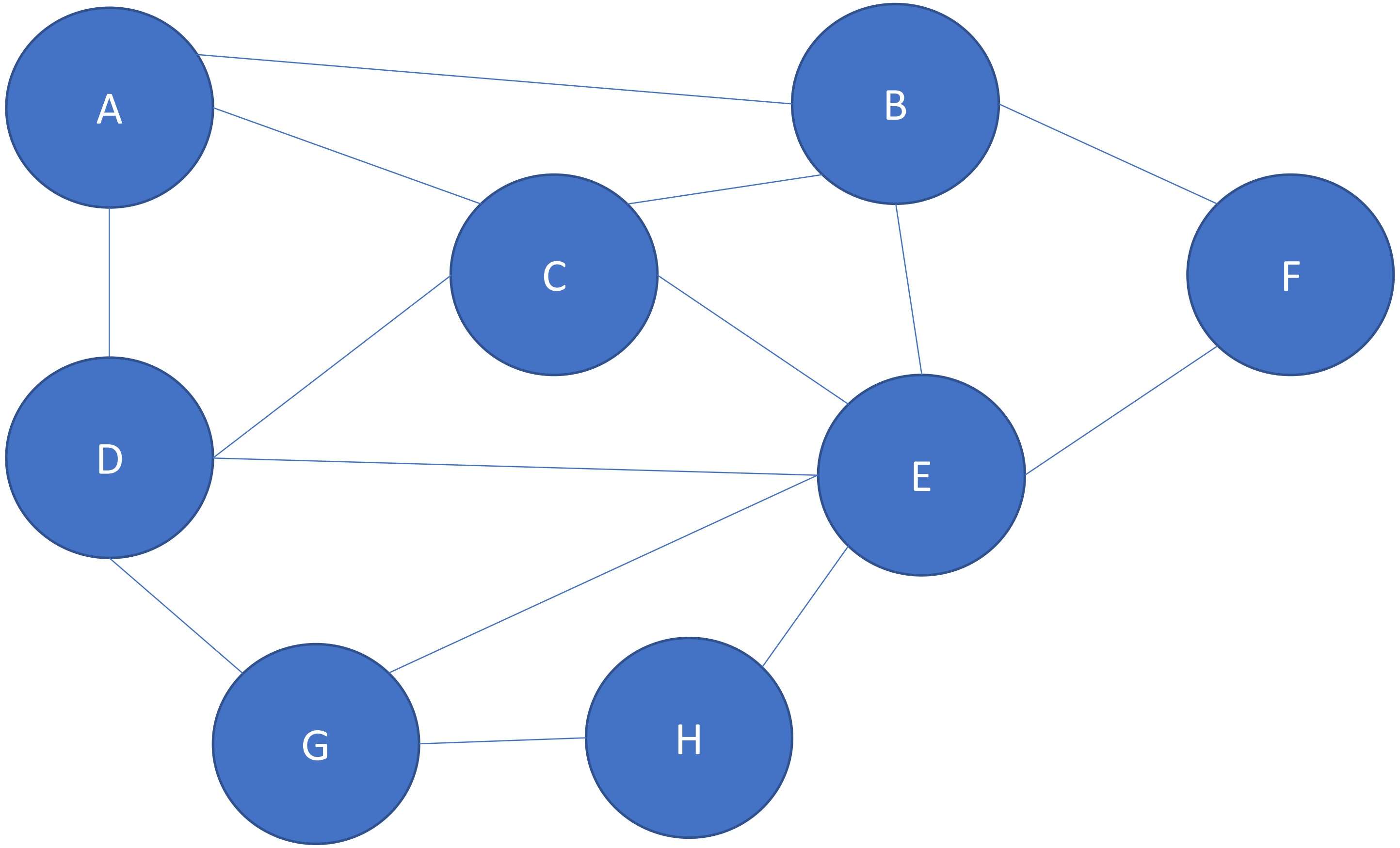
Kruskal Algorithm


A
B
C
D
E
F
G
H
2
5
1
7
3
8
3
3
10
7
9
8
3
4

For Kruskal algorithm, we need to initialize our edge-list in ascending order and states of vertexes (U: Unvisited; V:Visited)

A	B	C	D	E	F	G	H
U	U	U	U	U	U	U	U

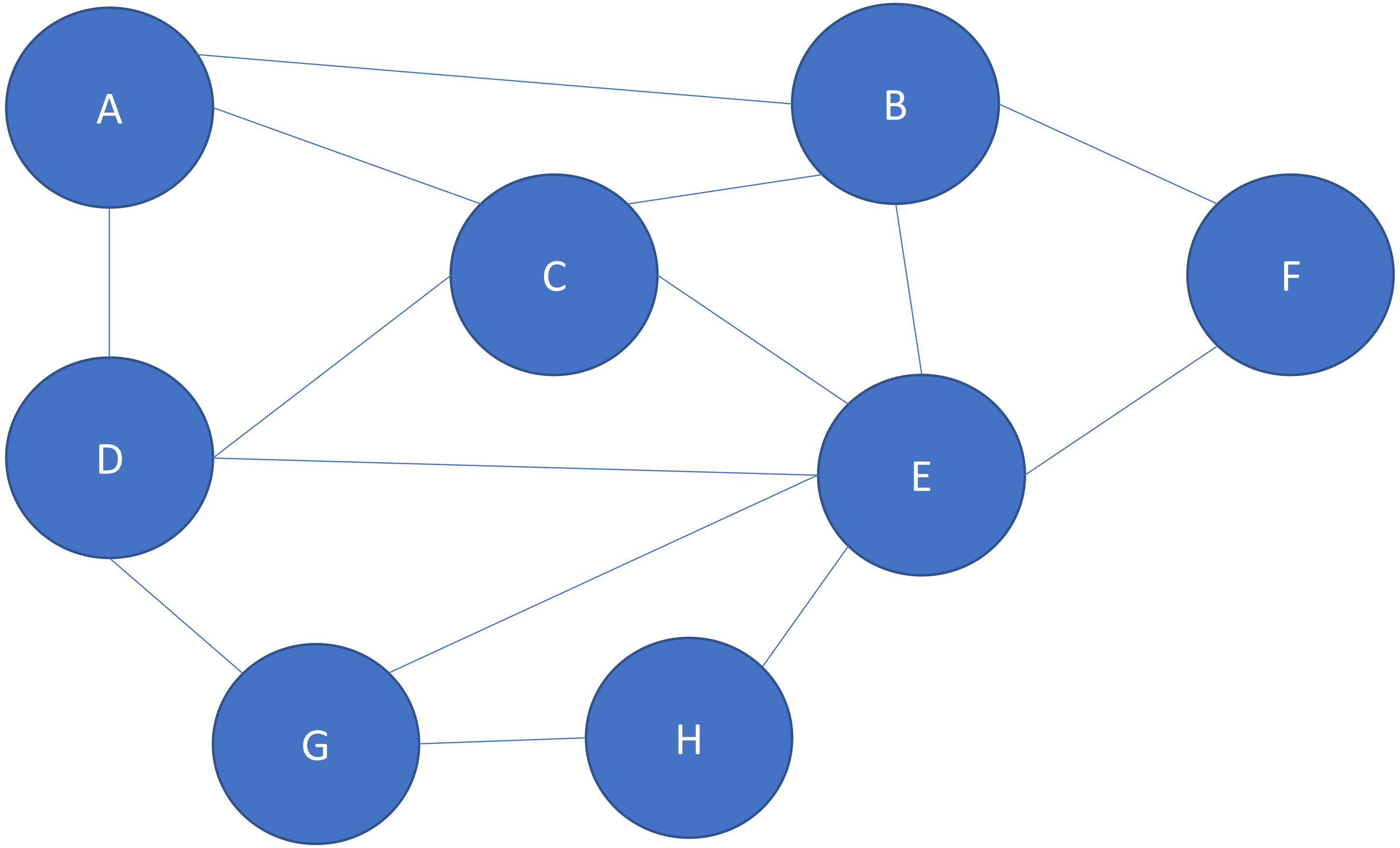
A-D: 1
A-B: 2
D-C: 3
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



We look at the A-D edge. A and D are both unvisited. So we can add A-D edge to MST.

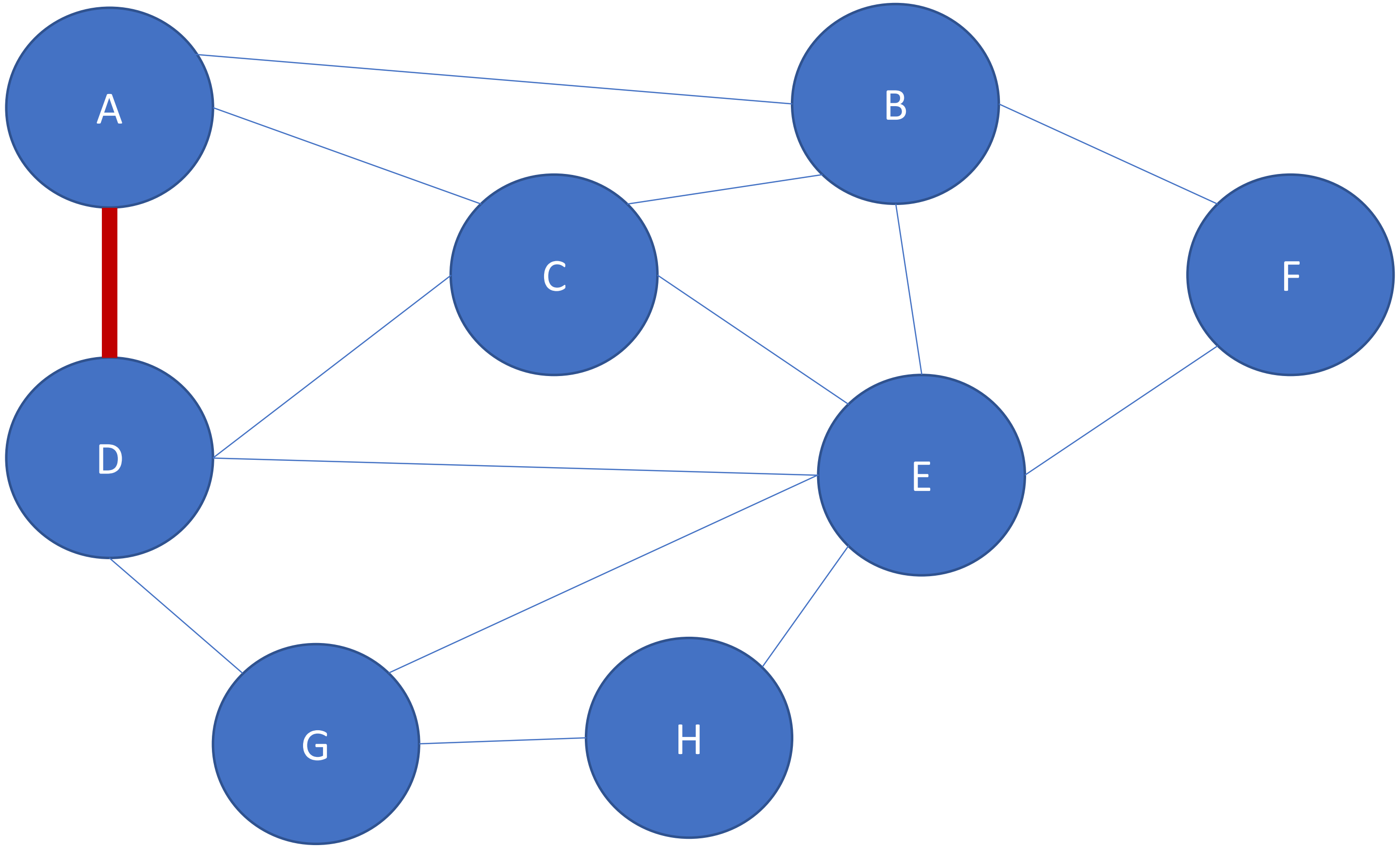
A	B	C	D	E	F	G	H
U	U	U	U	U	U	U	U

A-D: 1
A-B: 2
D-C: 3
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



A	B	C	D	E	F	G	H
V	U	U	V	U	U	U	U

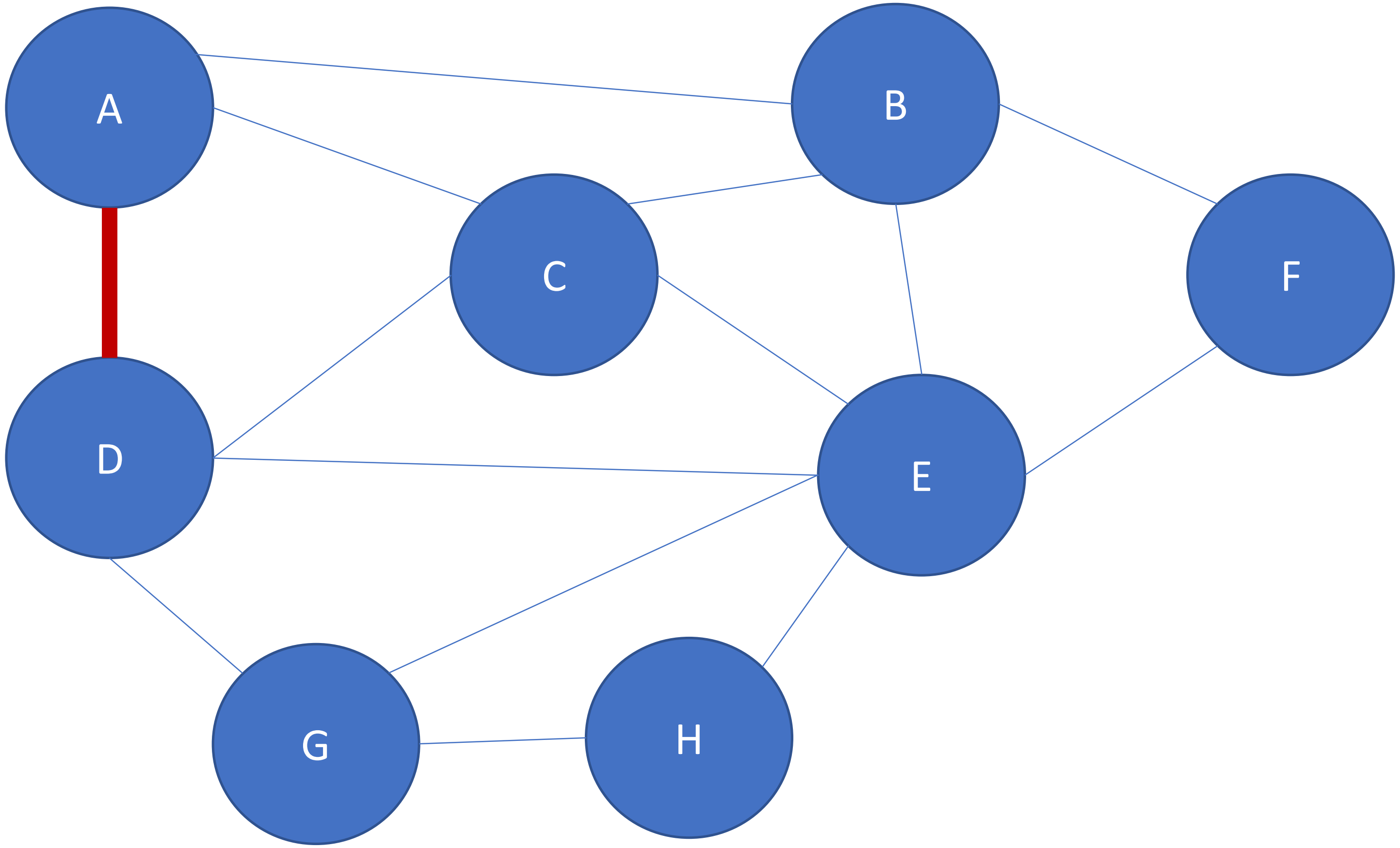
~~A-D: 1~~
A-B: 2
D-C: 3
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



We look at the A-B edge. A is visited but B is unvisited. So we can add A-B edge to MST.

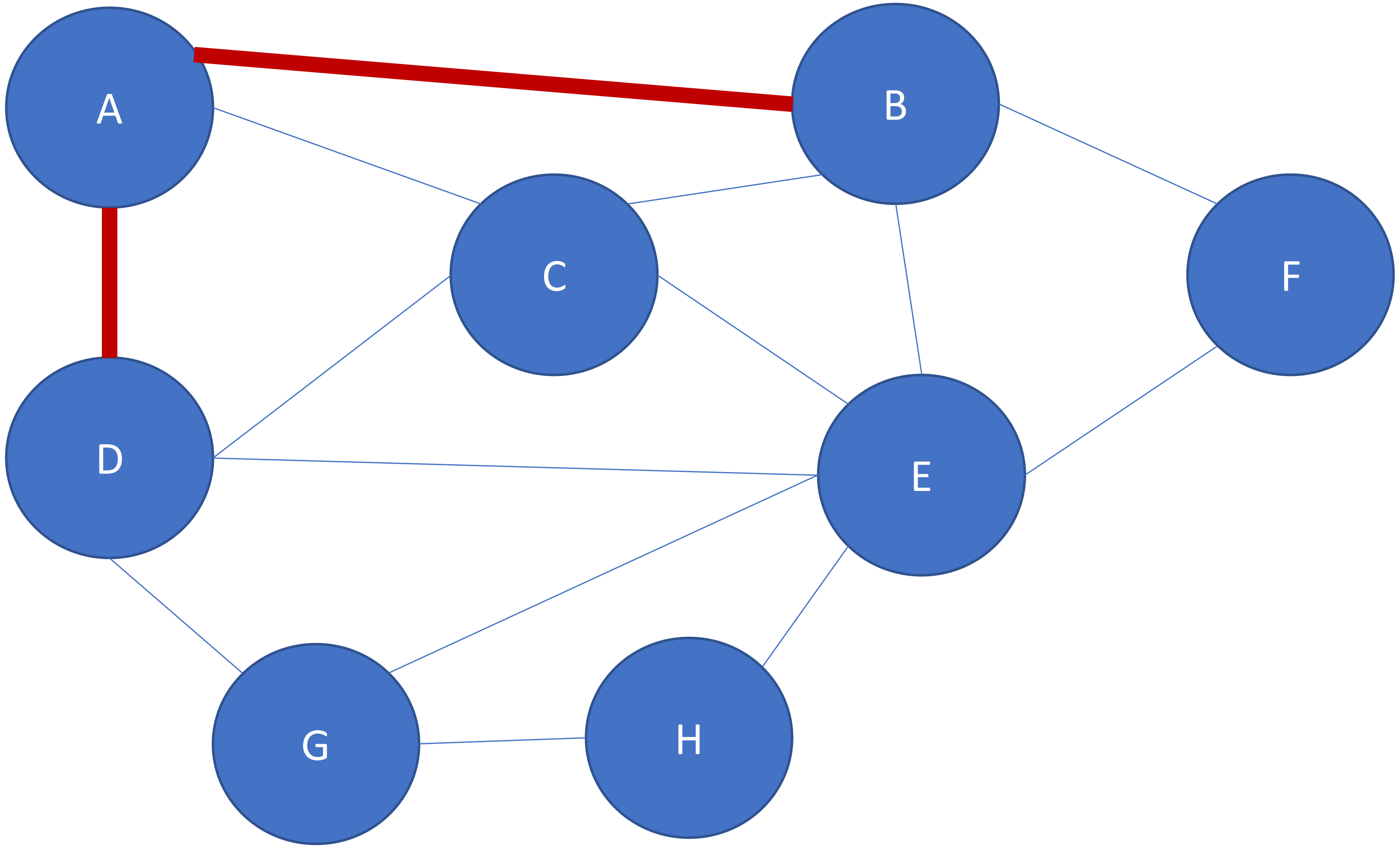
A	B	C	D	E	F	G	H
V	U	U	V	U	U	U	U

~~A-D: 1~~
A-B: 2
D-C: 3
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



A	B	C	D	E	F	G	H
V	V	U	V	U	U	U	U

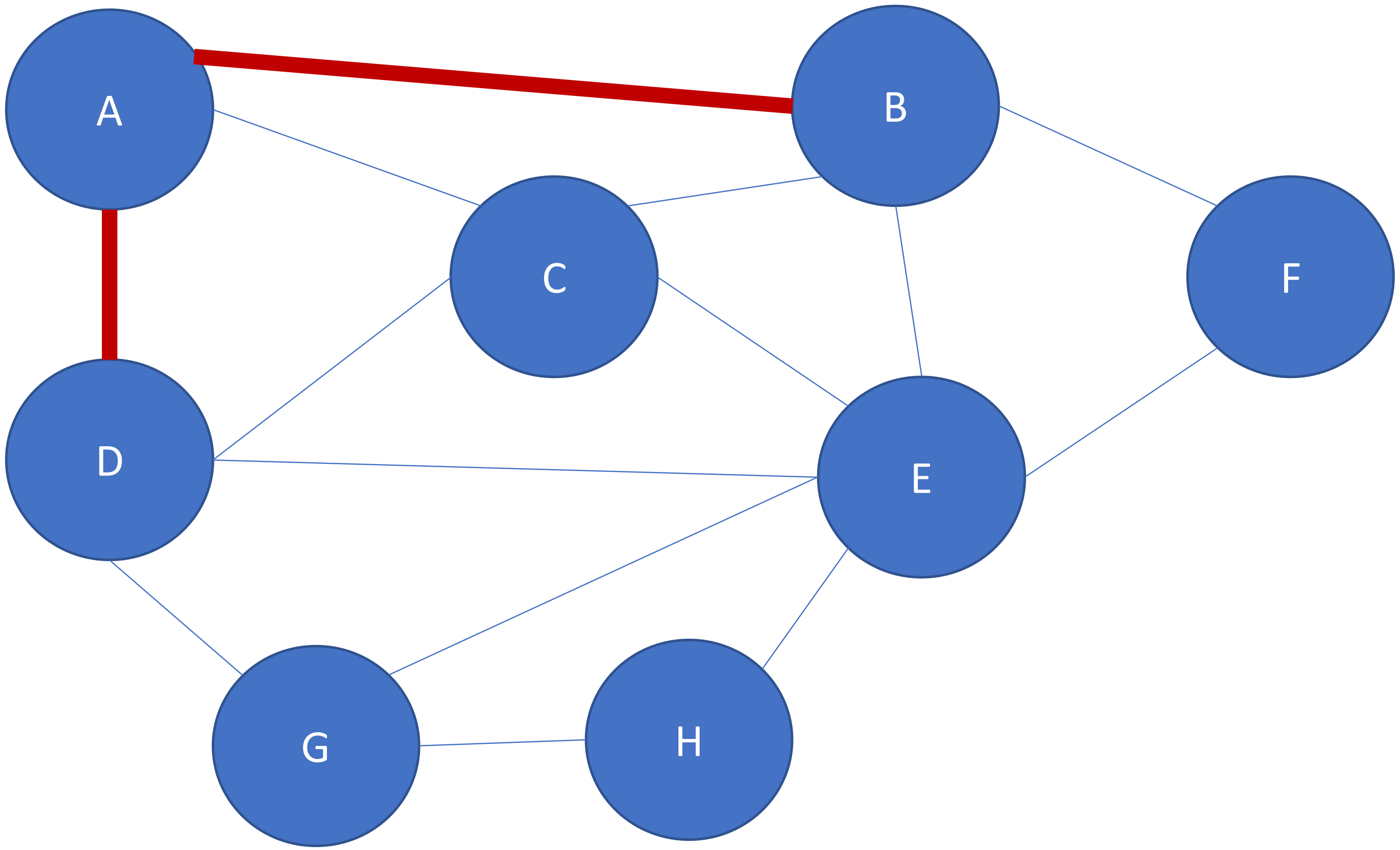
~~A-D: 1~~
~~A-B: 2~~
D-C: 3
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



We look at the D-C edge. D is visited but C is unvisited. So we can add D-C edge to MST.

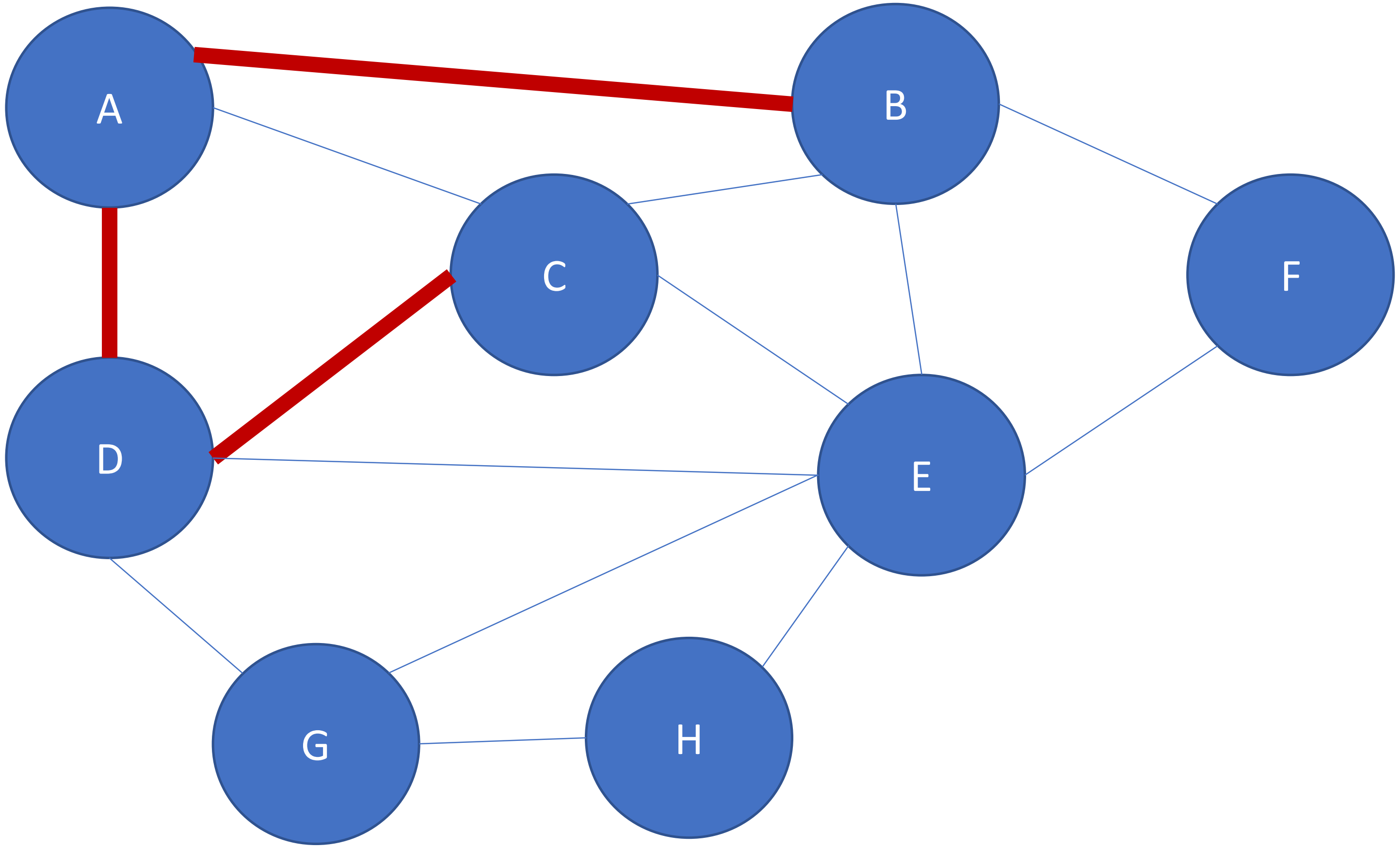
A	B	C	D	E	F	G	H
V	V	U	V	U	U	U	U

~~A-D: 1~~
~~A-B: 2~~
D-C: 3
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



A	B	C	D	E	F	G	H
V	V	V	V	U	U	U	U

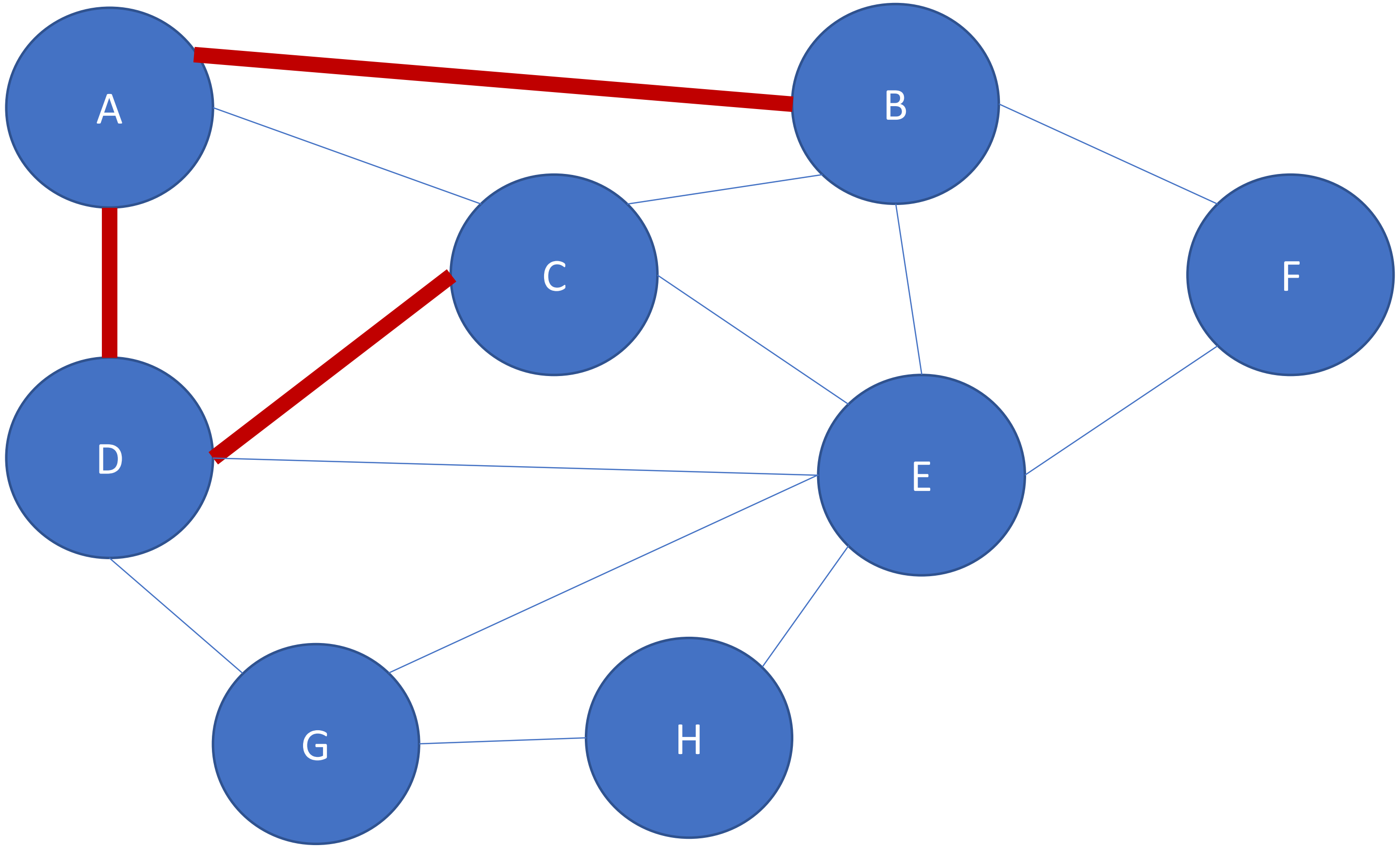
~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



We look at the B-F edge. B is visited but F is unvisited. So we can add B-F edge to MST.

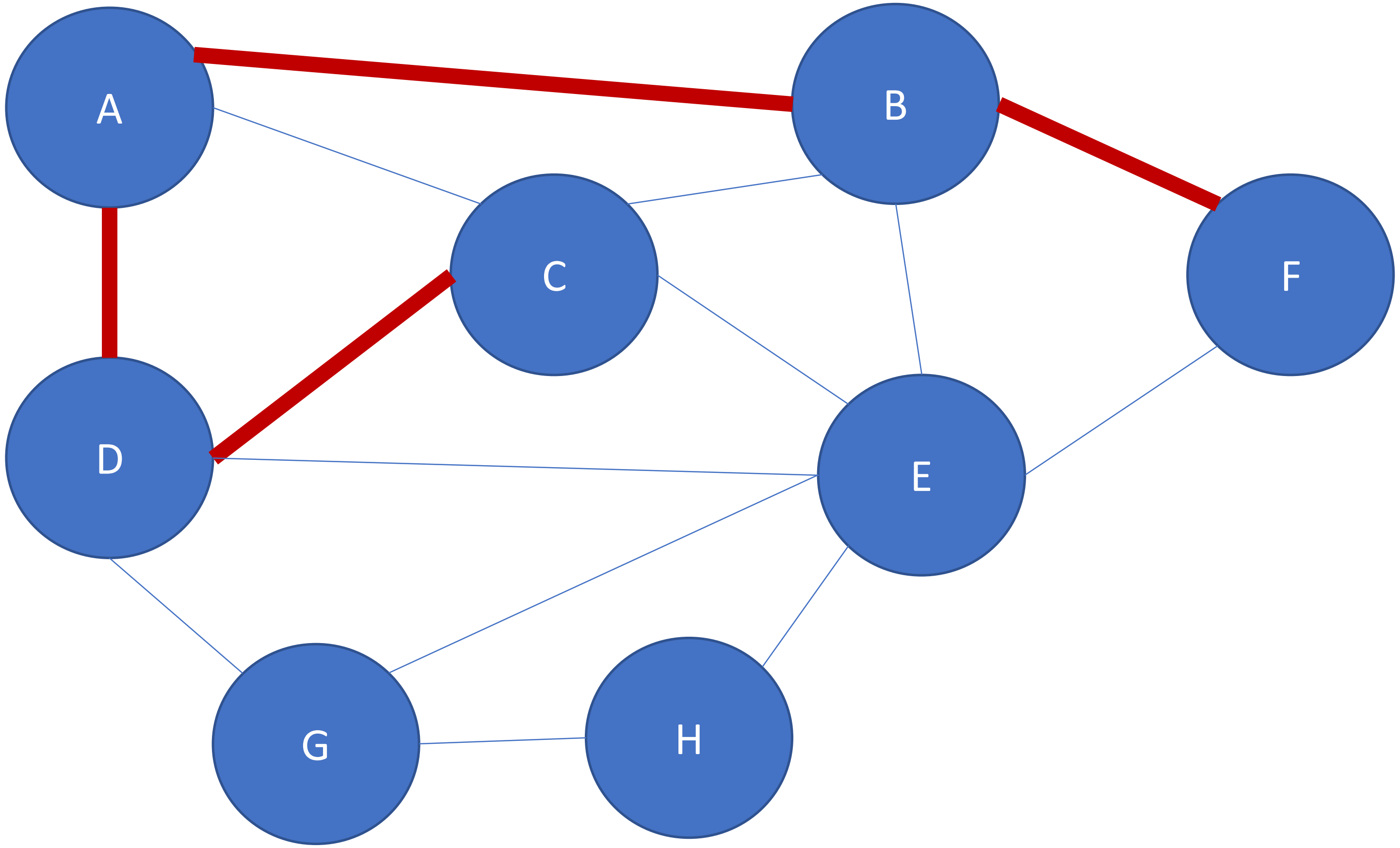
A	B	C	D	E	F	G	H
V	V	V	V	U	U	U	U

~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
B-F: 3
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



A	B	C	D	E	F	G	H
V	V	V	V	U	V	U	U

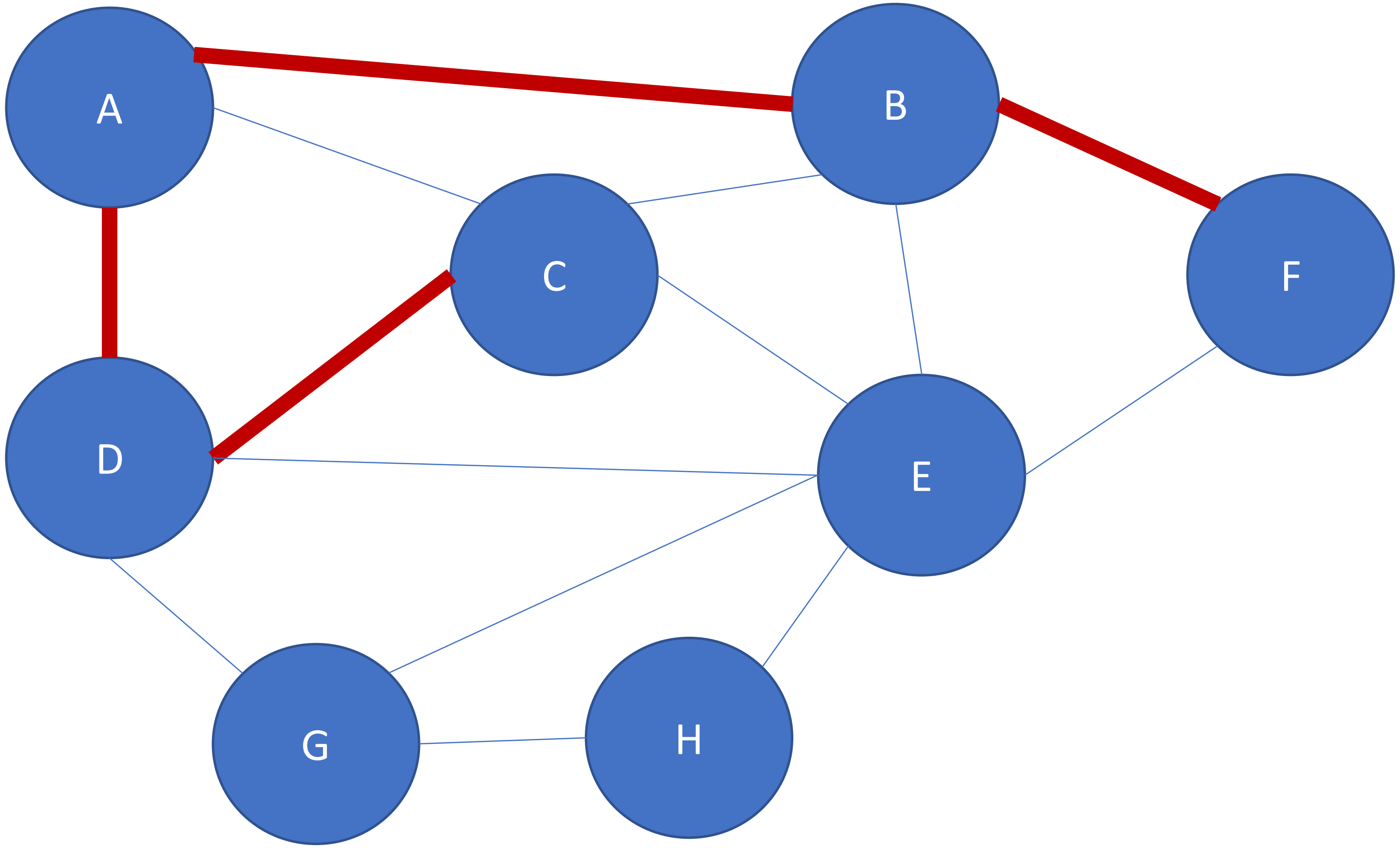
~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



We look at the C-E edge. C is visited but E is unvisited. So we can add C-E edge to MST.

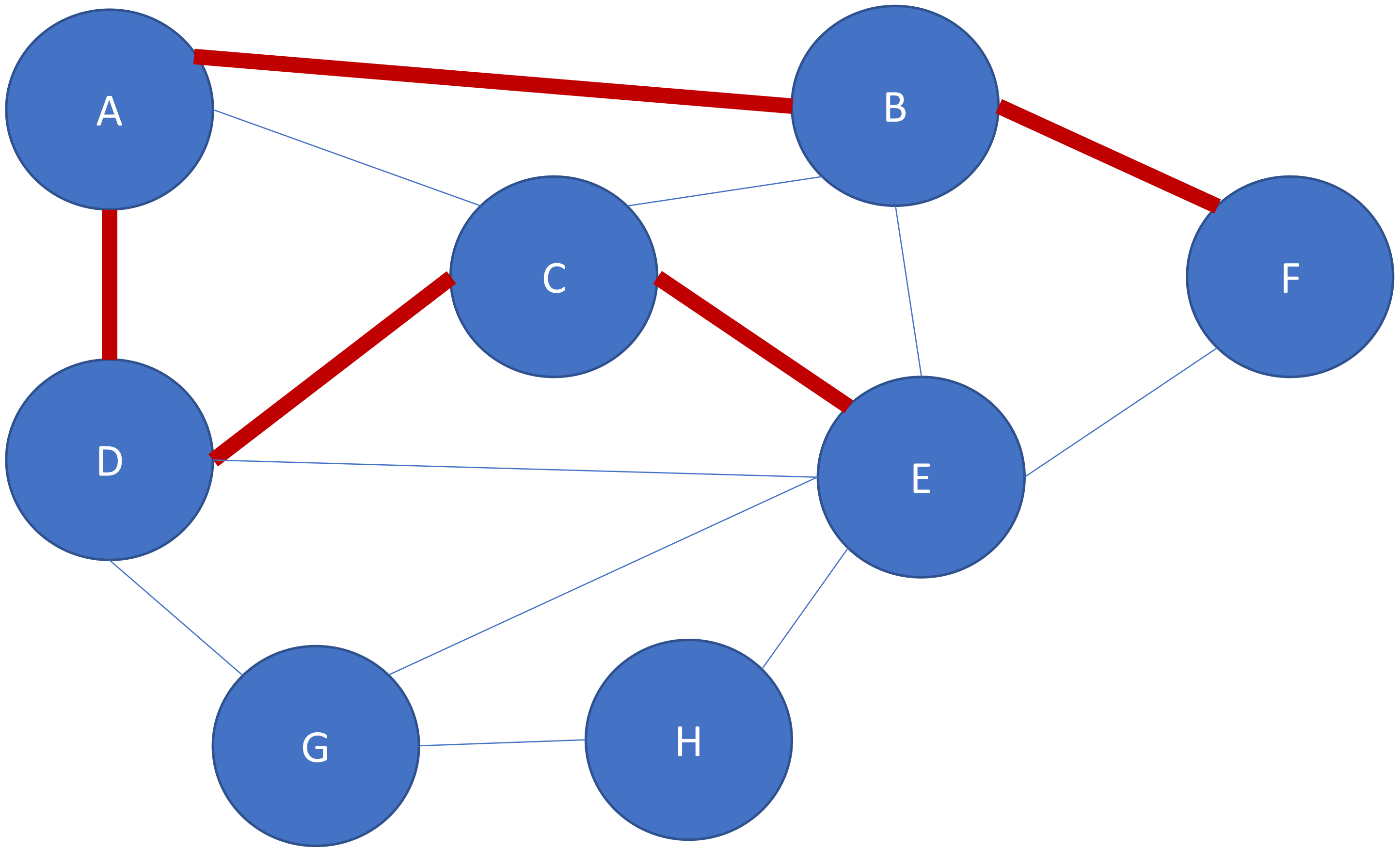
A	B	C	D	E	F	G	H
V	V	V	V	U	V	U	U

~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
C-E: 3
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



A	B	C	D	E	F	G	H
V	V	V	V	V	V	U	U

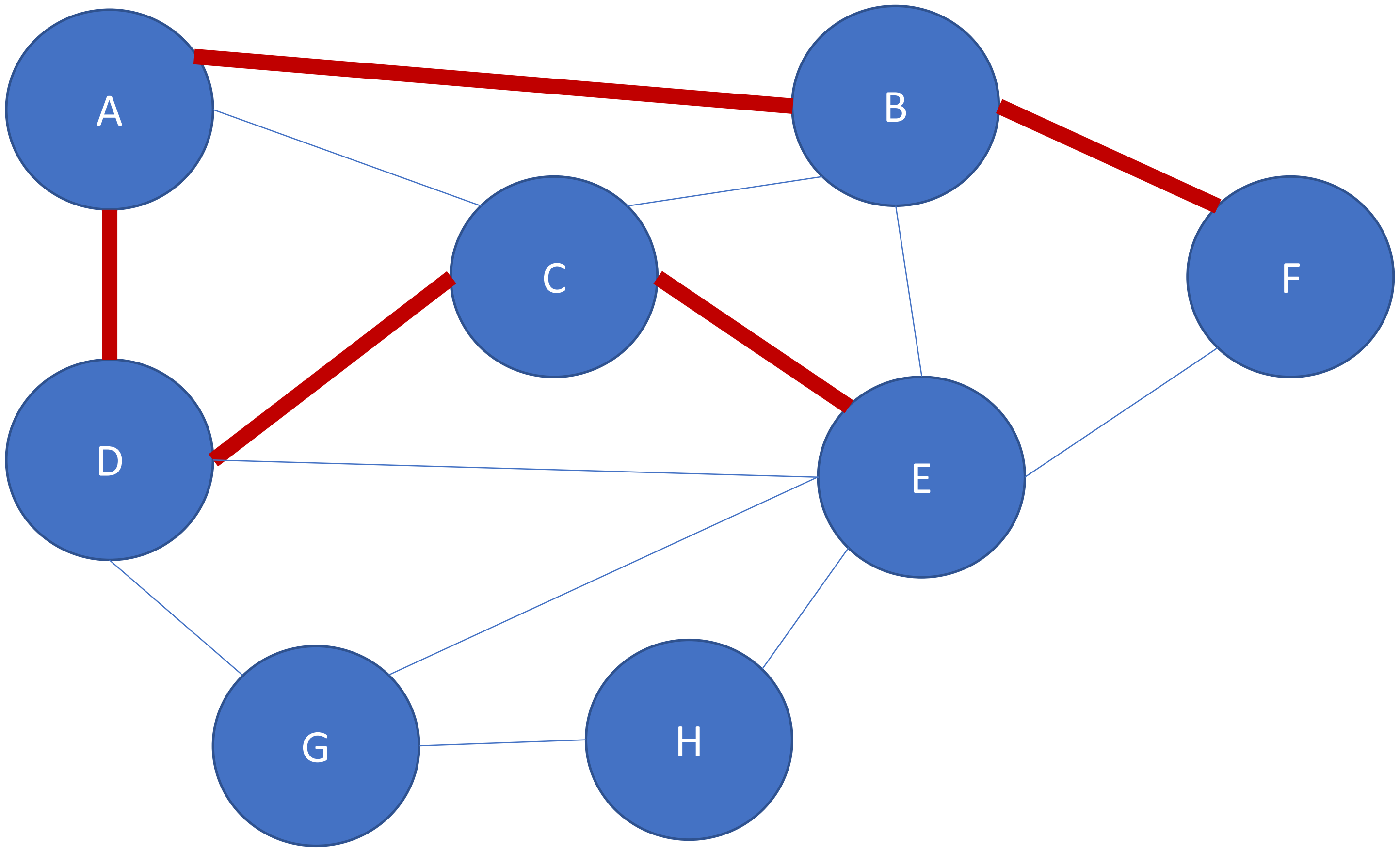
~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
~~C-E: 3~~
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



We look at the E-H edge. E is visited but H is unvisited. So we can add E-H edge to MST.

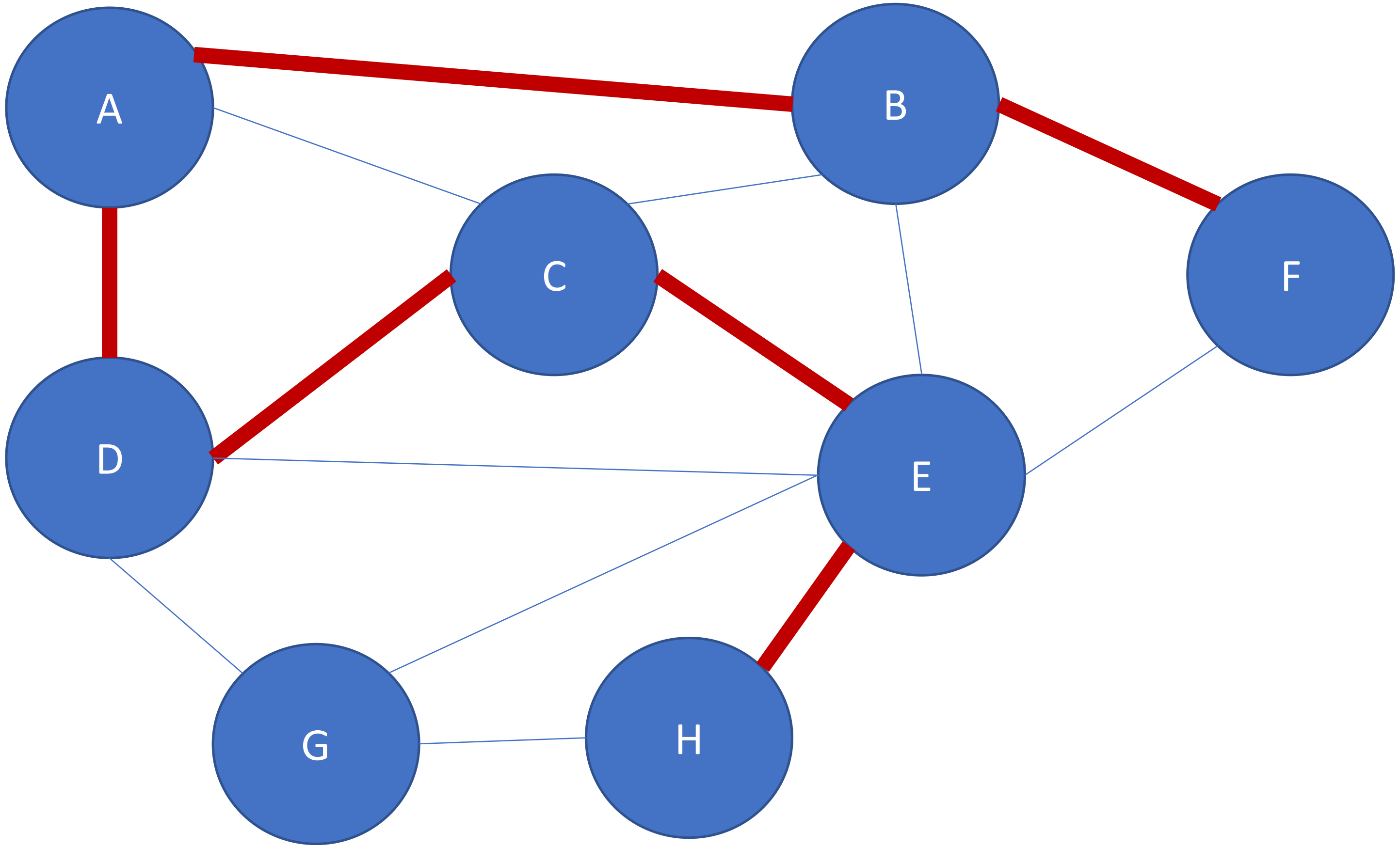
A	B	C	D	E	F	G	H
V	V	V	V	V	V	U	U

~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
~~C-E: 3~~
E-H: 3
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



A	B	C	D	E	F	G	H
V	V	V	V	V	V	U	V

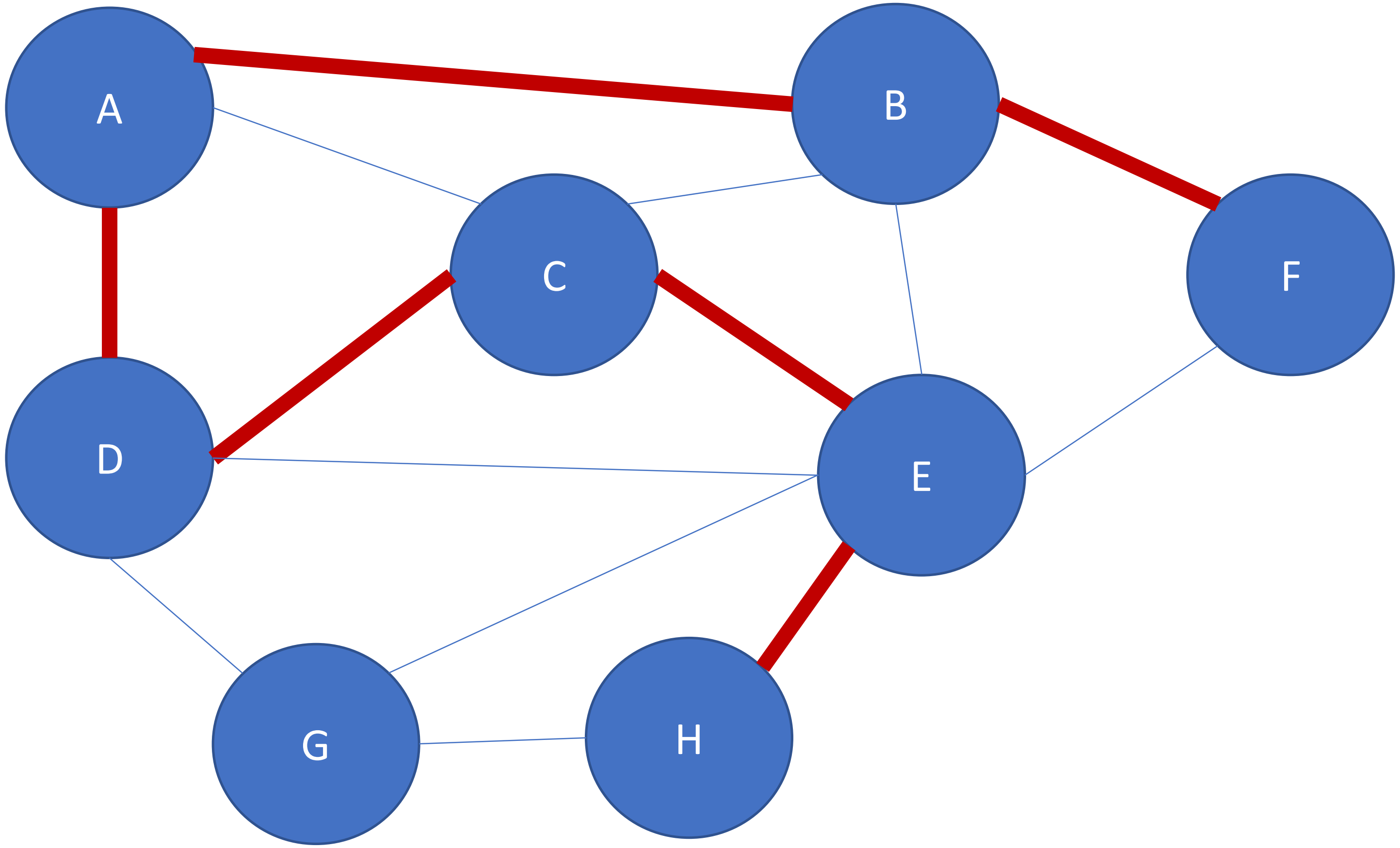
~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
~~C-E: 3~~
~~E-H: 3~~
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



We look at the G-H edge. H is visited but G is unvisited. So we can add G-H edge to MST.

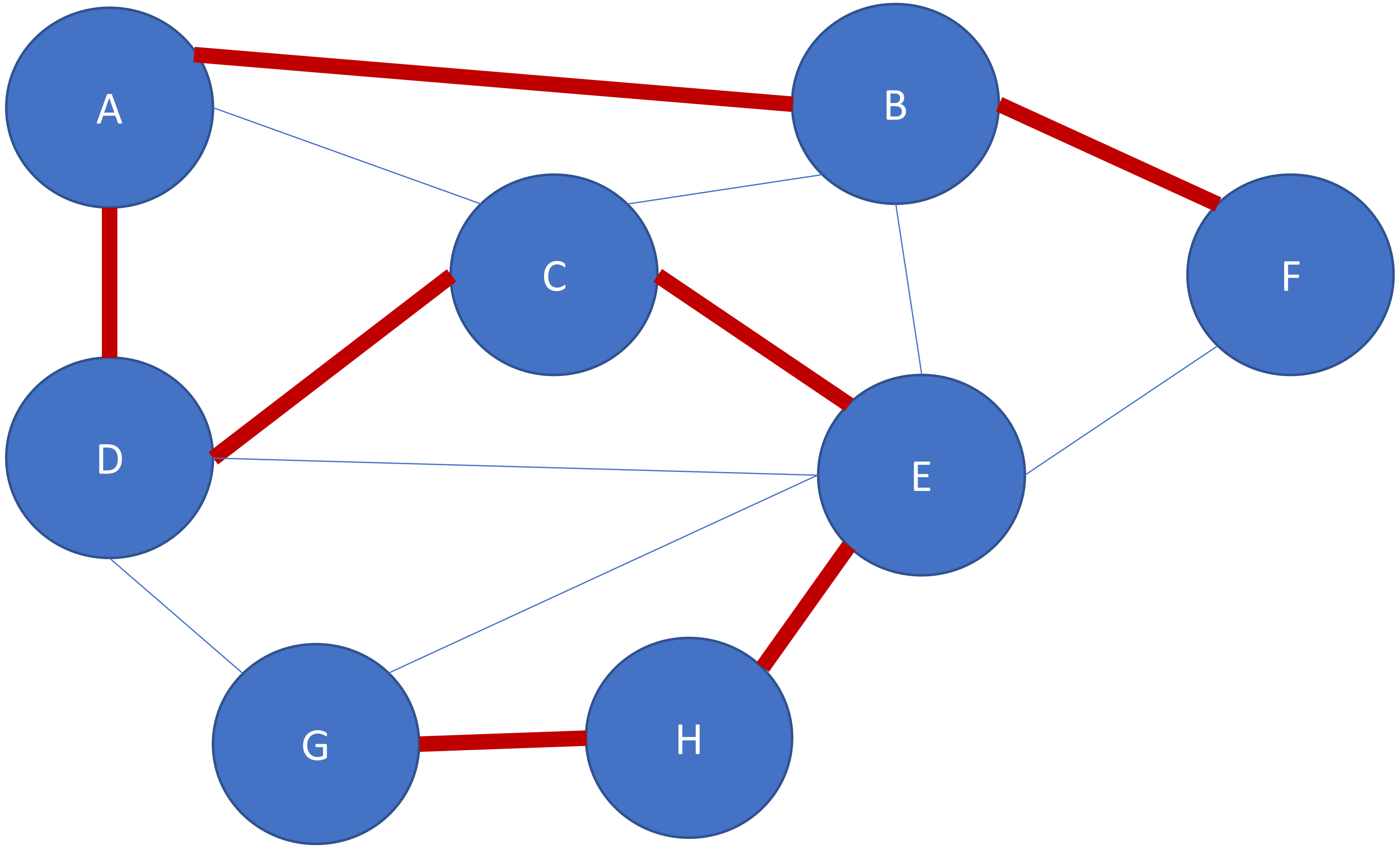
A	B	C	D	E	F	G	H
V	V	V	V	V	V	U	V

~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
~~C-E: 3~~
~~E-H: 3~~
G-H: 4
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



A	B	C	D	E	F	G	H
V	V	V	V	V	V	V	V

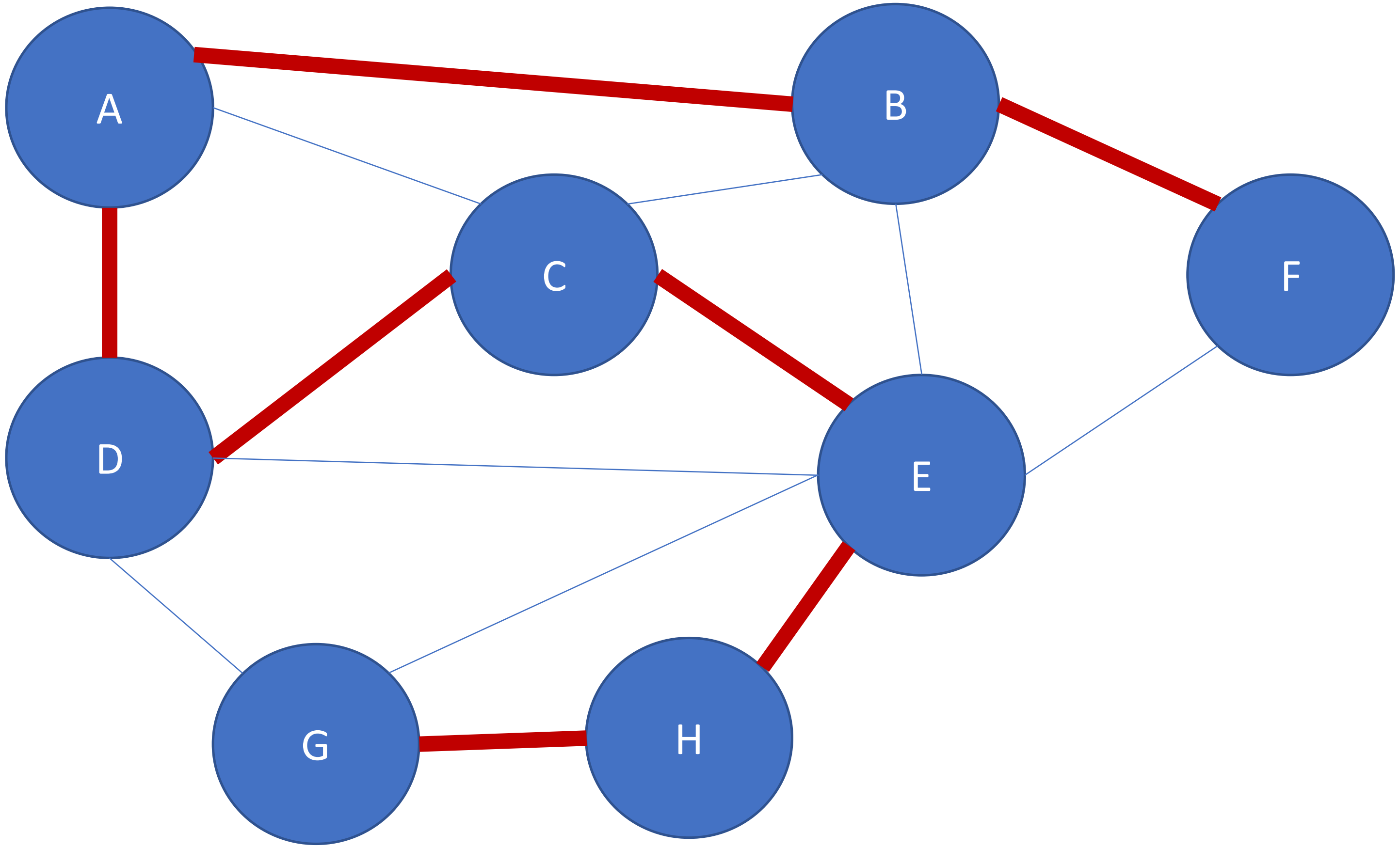
~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
~~C-E: 3~~
~~E-H: 3~~
~~G-H: 4~~
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10



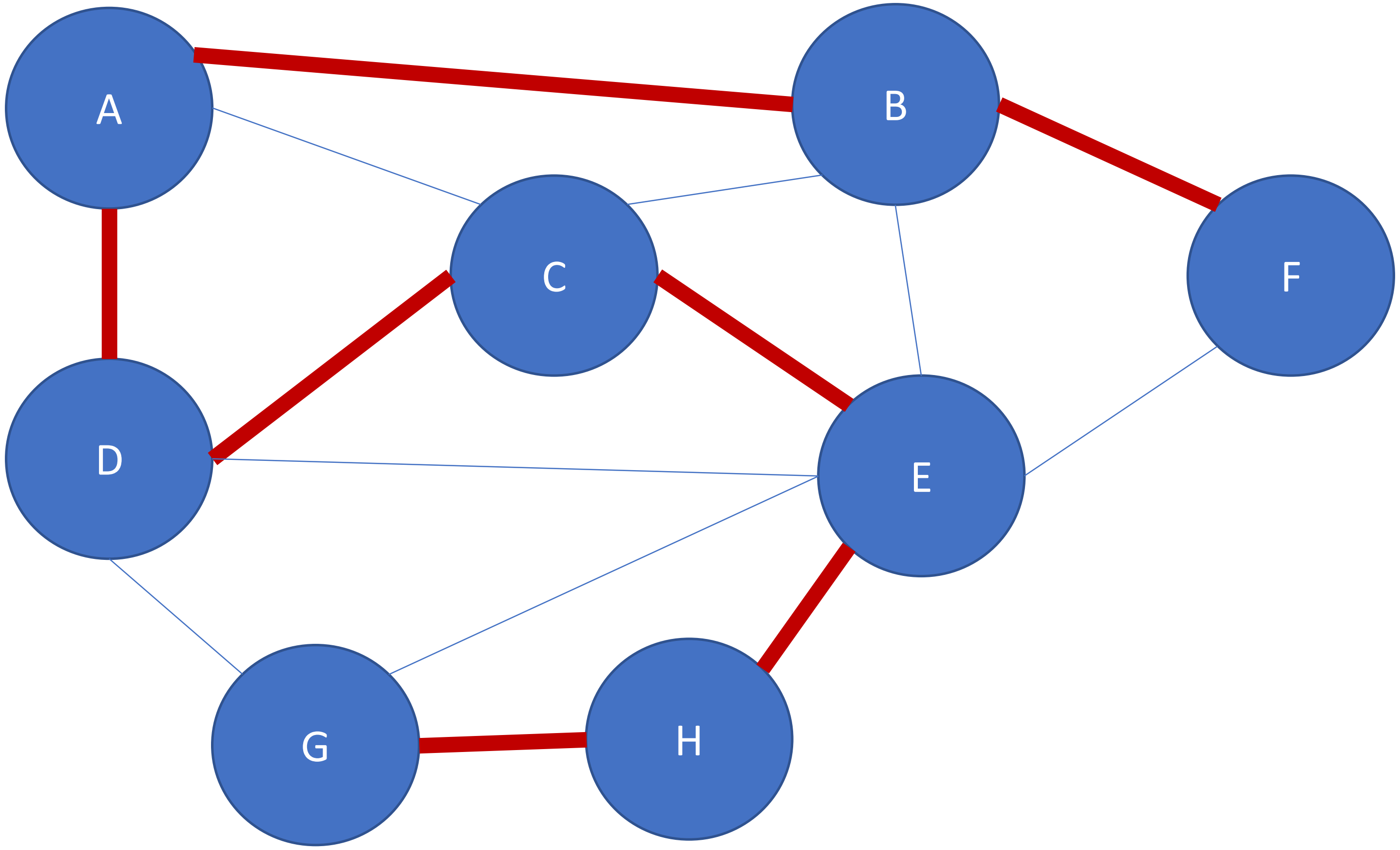
Since our current MST spans all the vertexes once, we are done.

A	B	C	D	E	F	G	H
V	V	V	V	V	V	V	V

~~A-D: 1~~
~~A-B: 2~~
~~D-C: 3~~
~~B-F: 3~~
~~C-E: 3~~
~~E-H: 3~~
~~G-H: 4~~
A-C: 5
D-E: 7
B-E: 8
G-E: 8
D-G: 9
E-F: 10

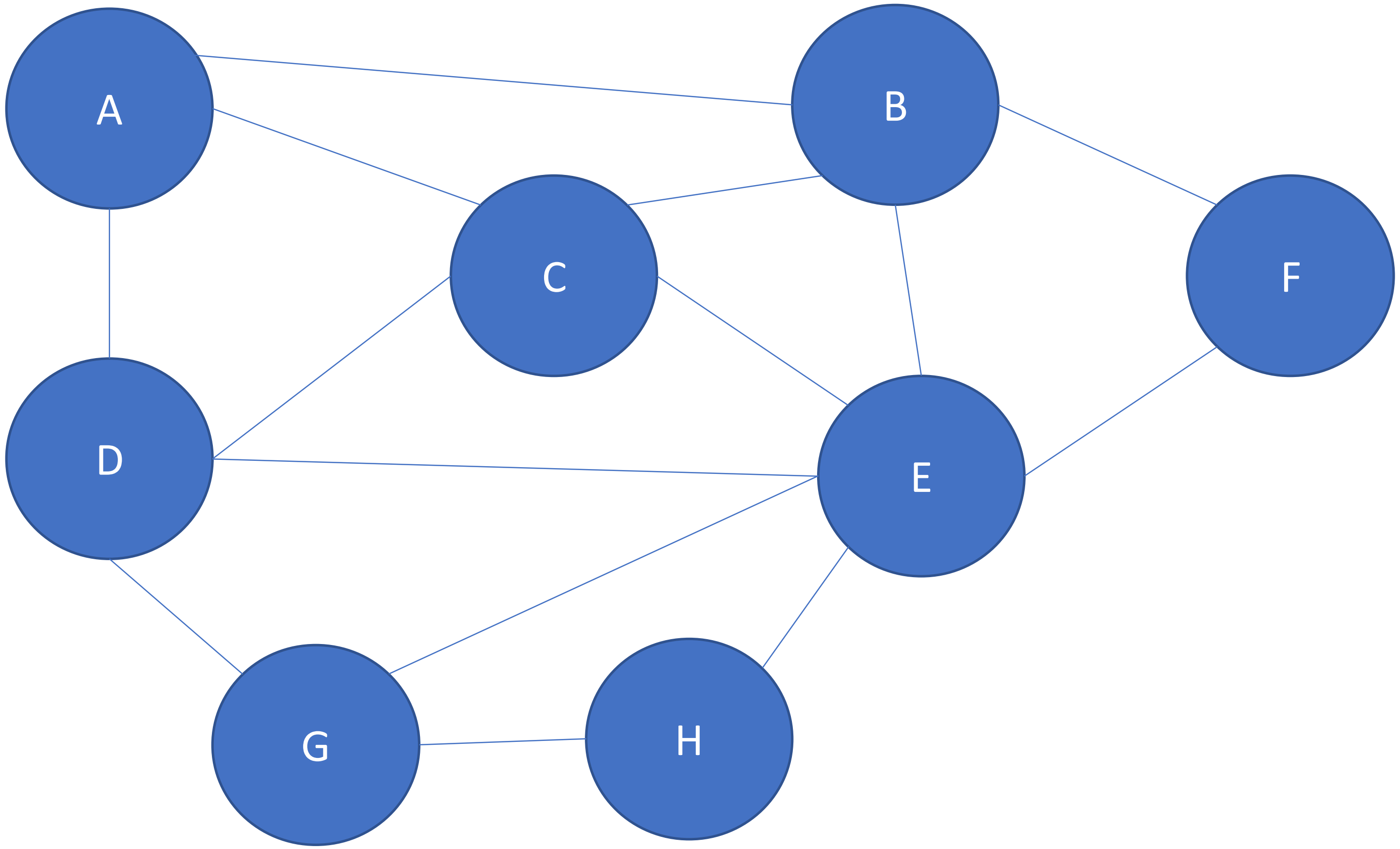


Final MST:

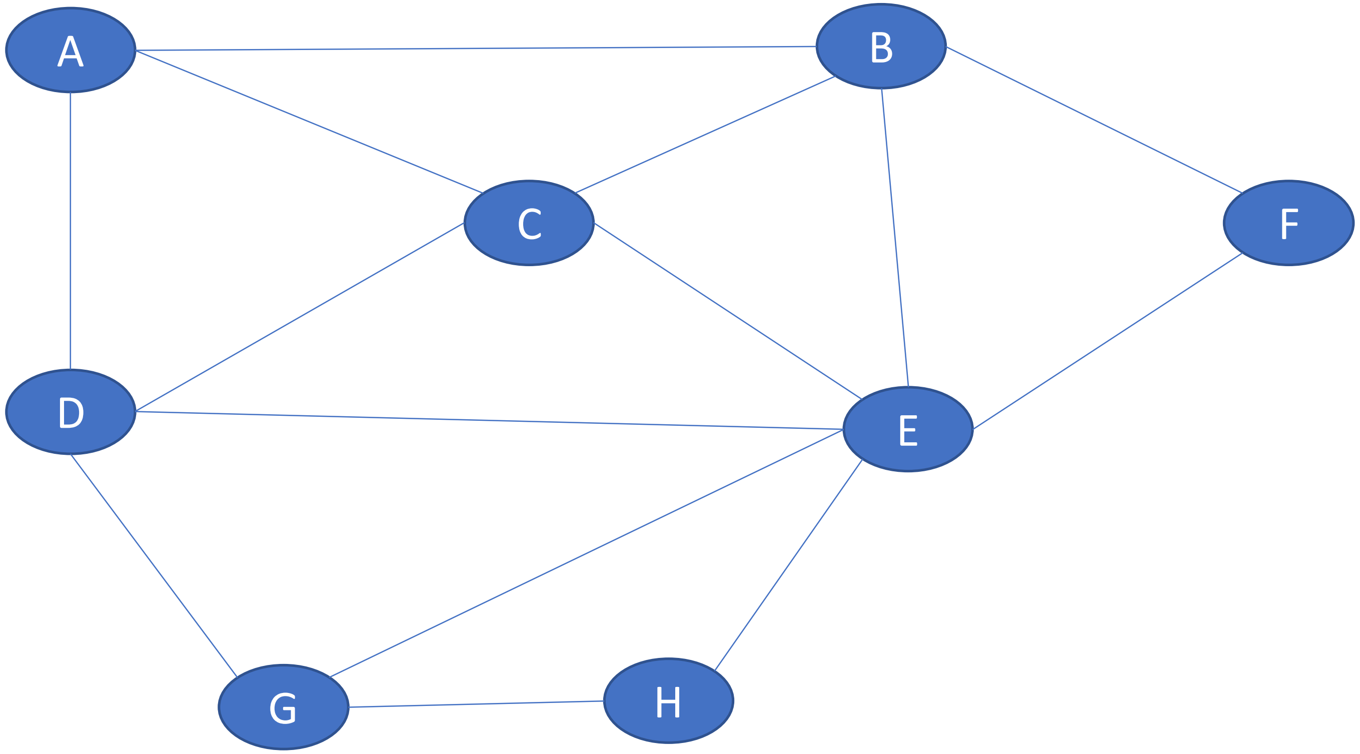


QUESTION 4

Unweighted Shortest Path

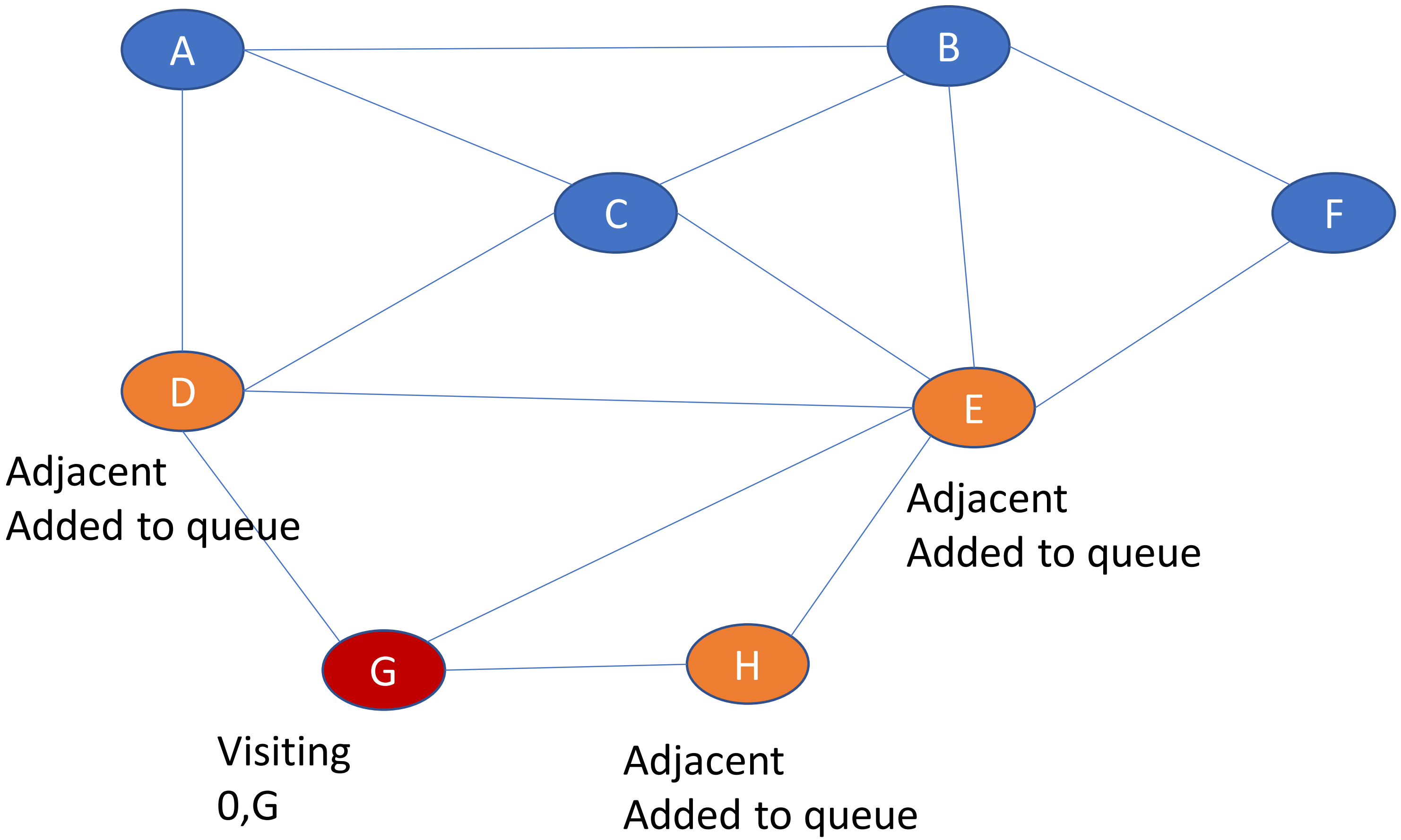


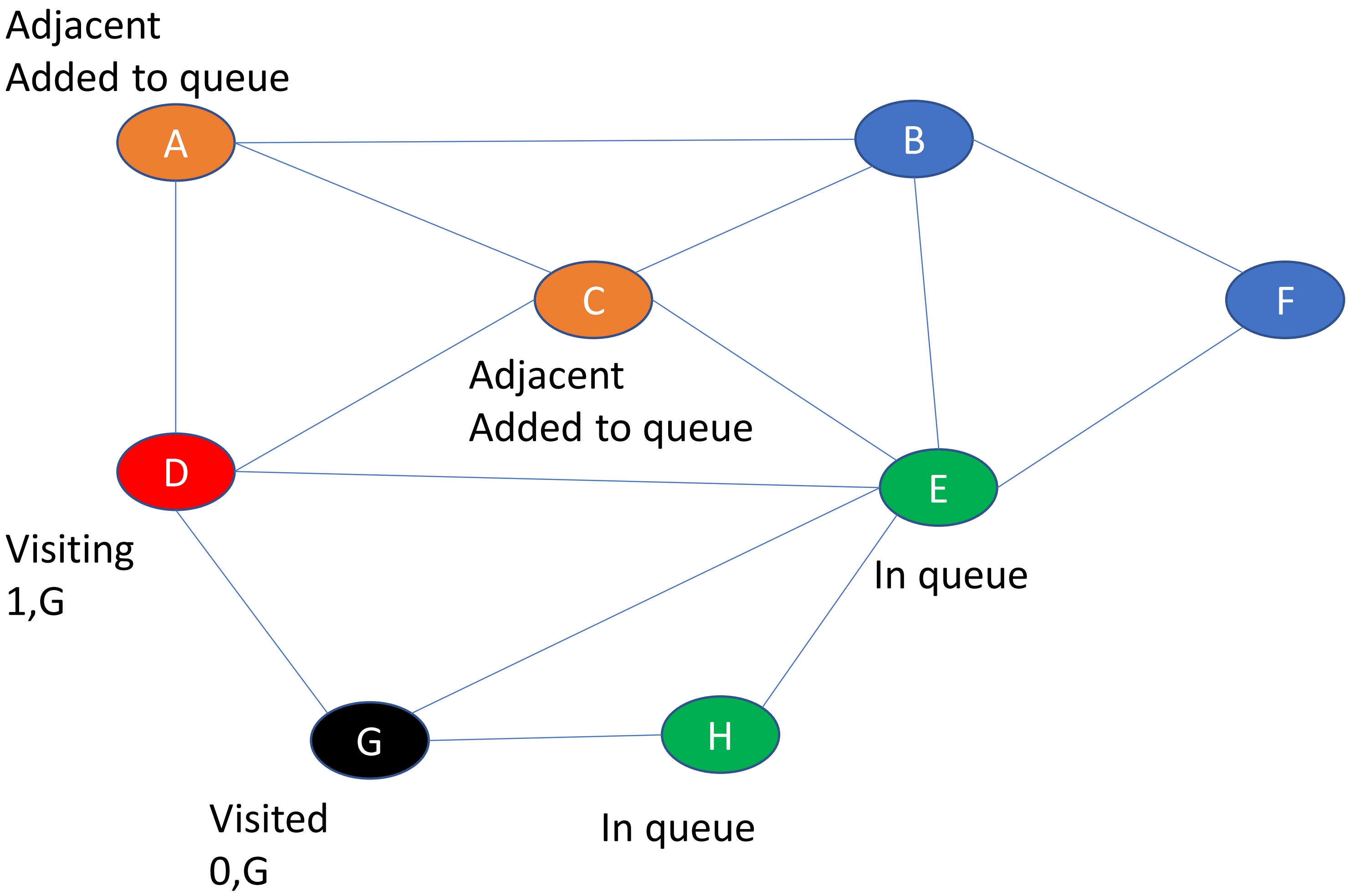
QUEUE:
G



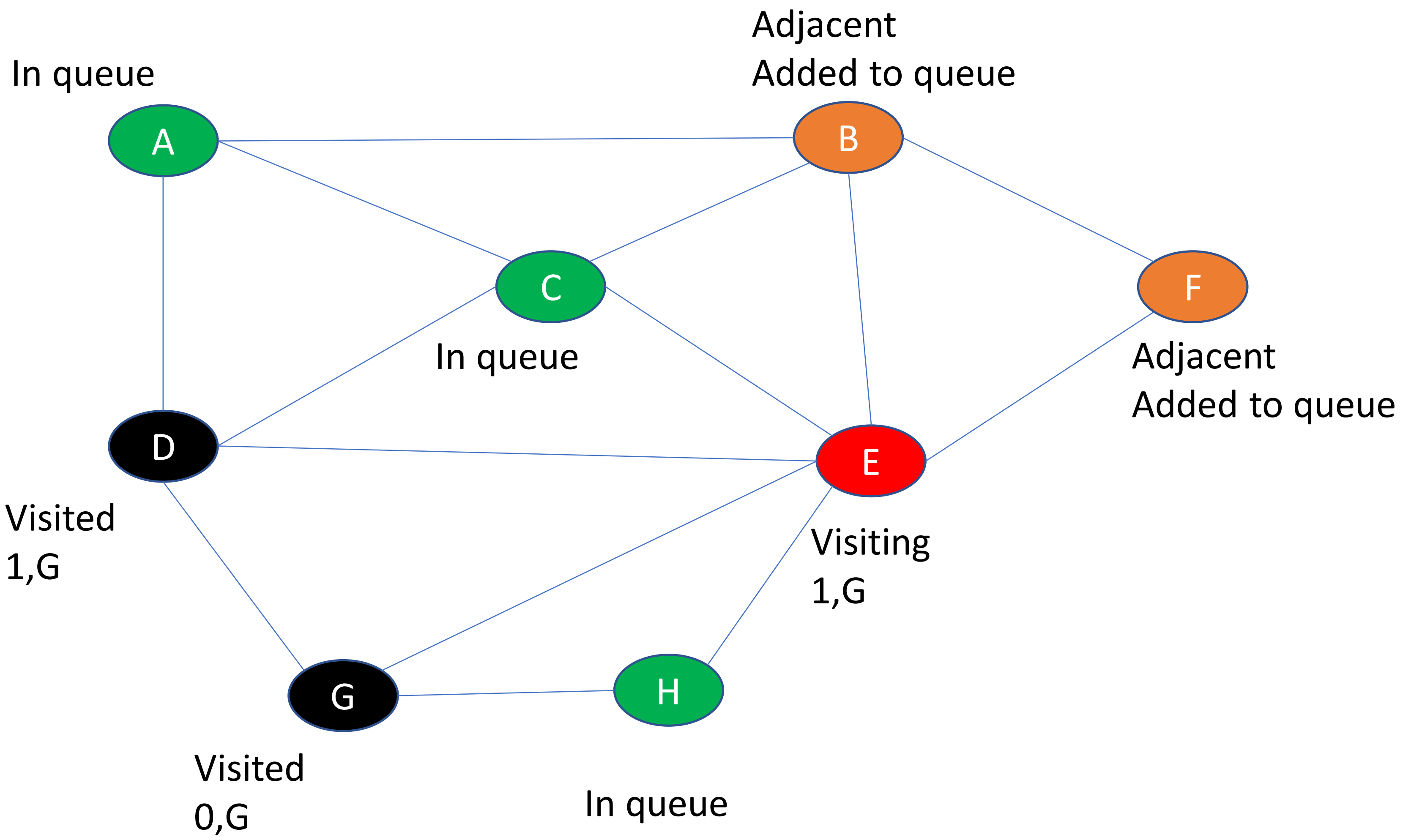
Starting vertex
Added to queue

QUEUE:
G <-
D
E
H

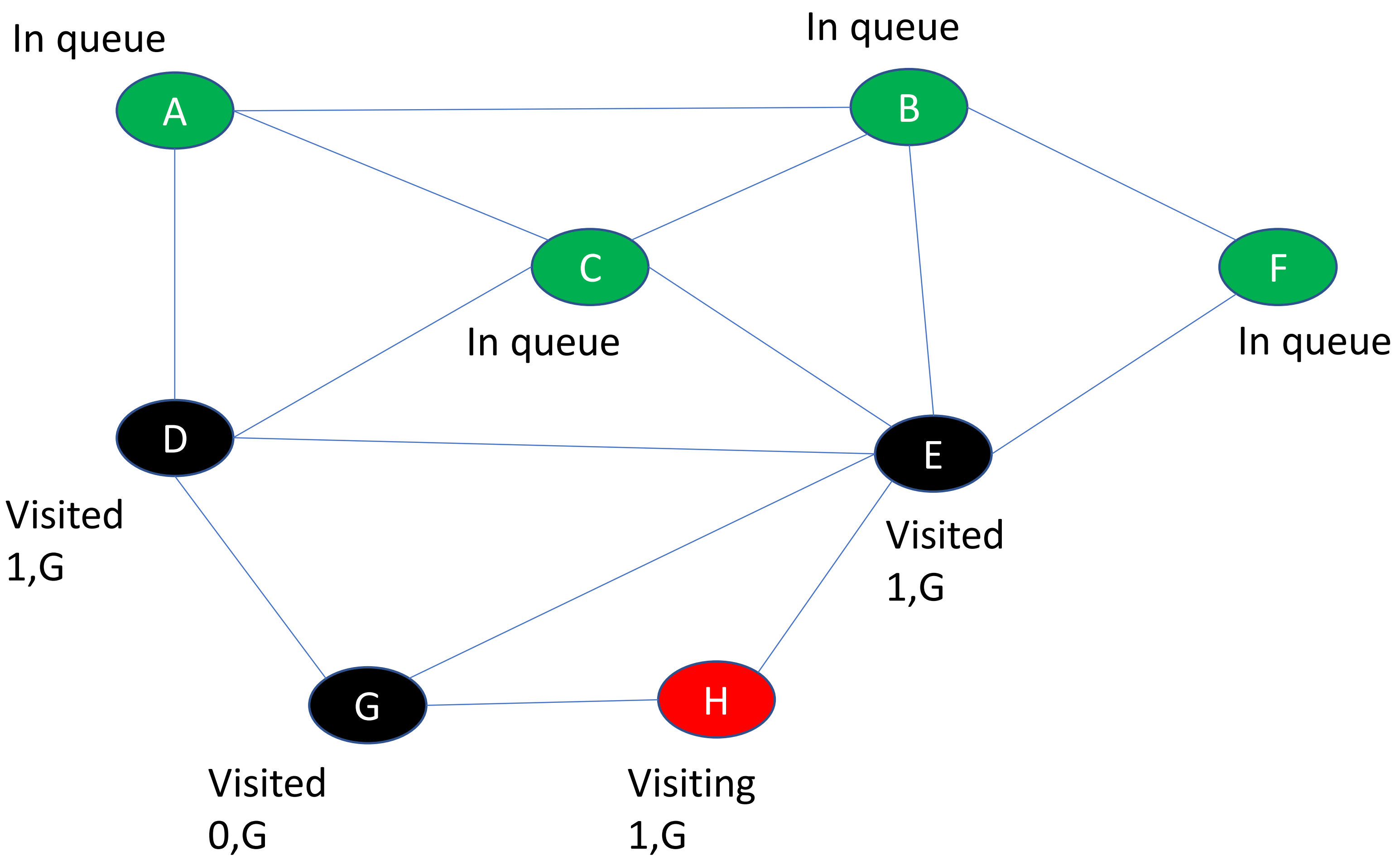

A
B
C
F
D
E
Adjacent
Added to queue
Adjacent
Added to queue
G
H
Visiting
0,G
Adjacent
Added to queue



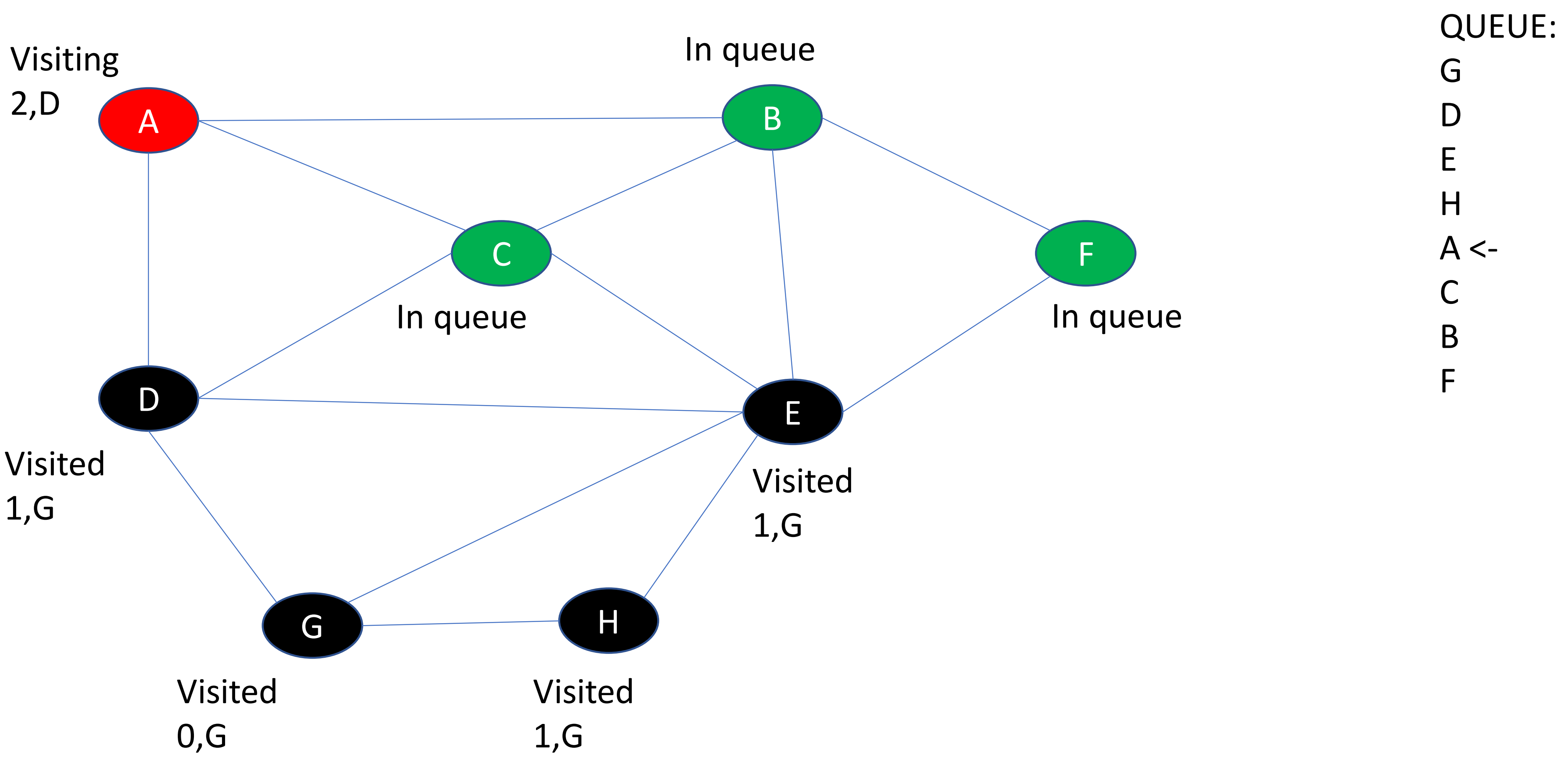
QUEUE:
G
D <-
E
H
A
C

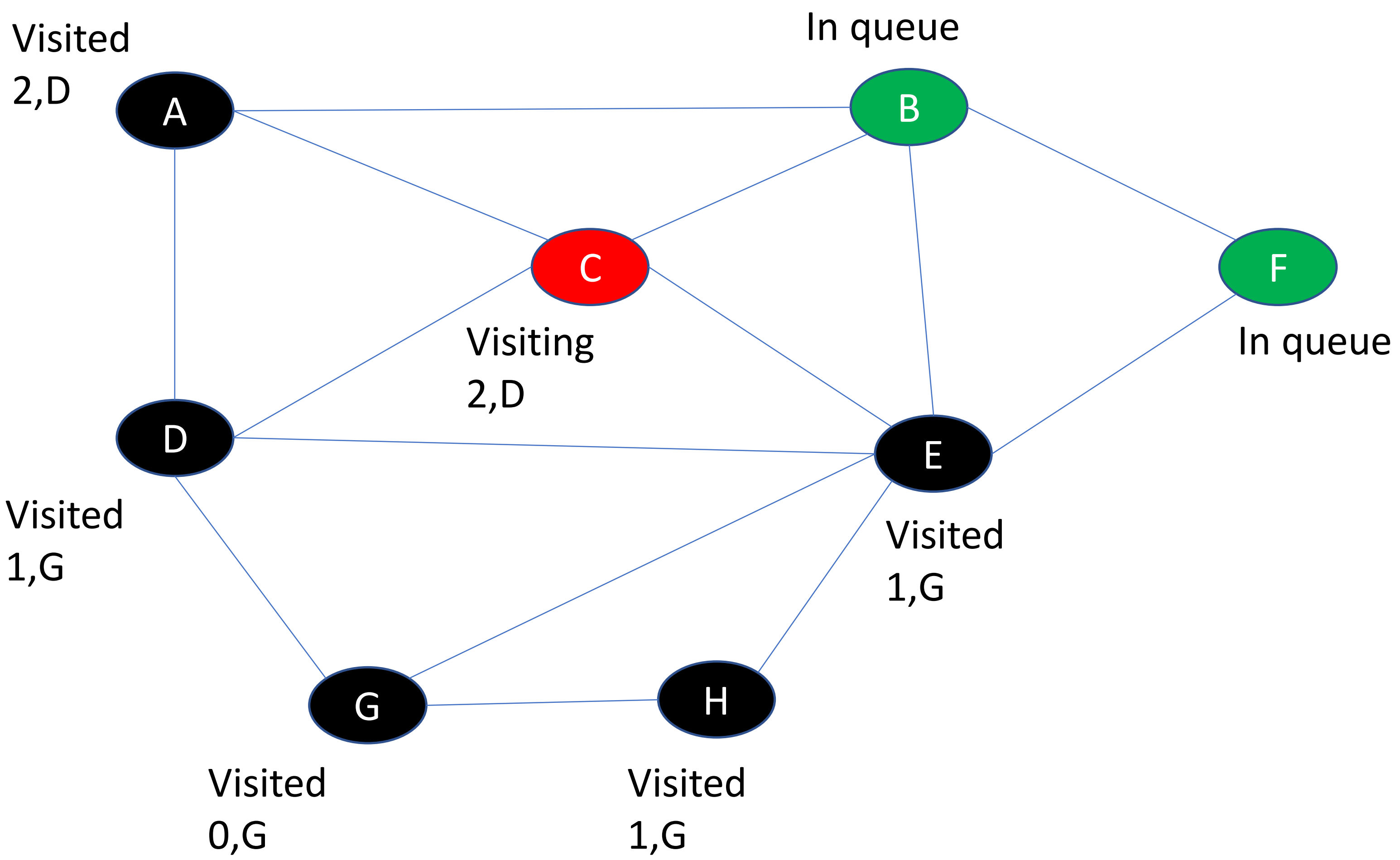


QUEUE:
G
D
E <-
H
A
C
B
F

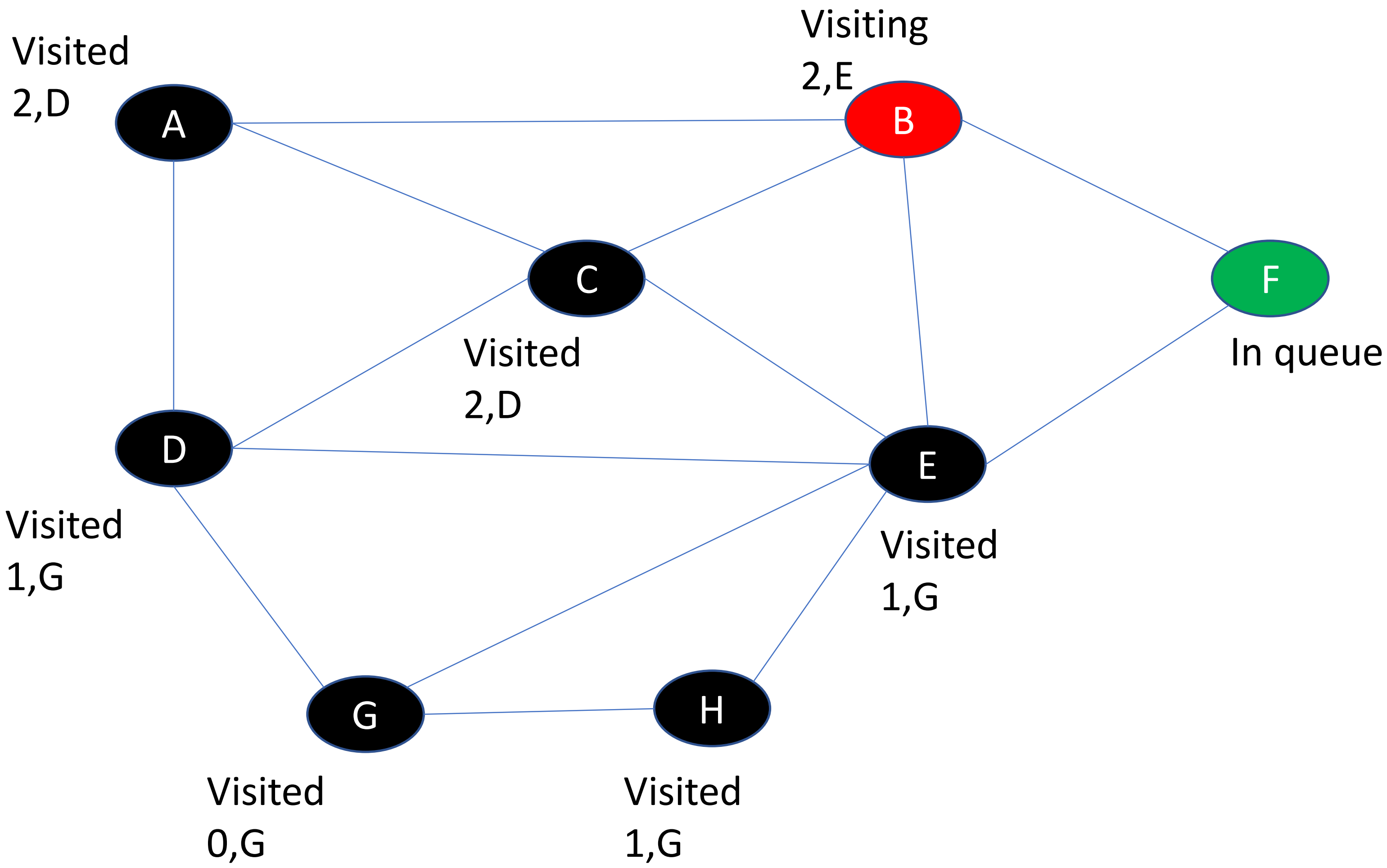


QUEUE:
G
D
E
H <-
A
C
B
F

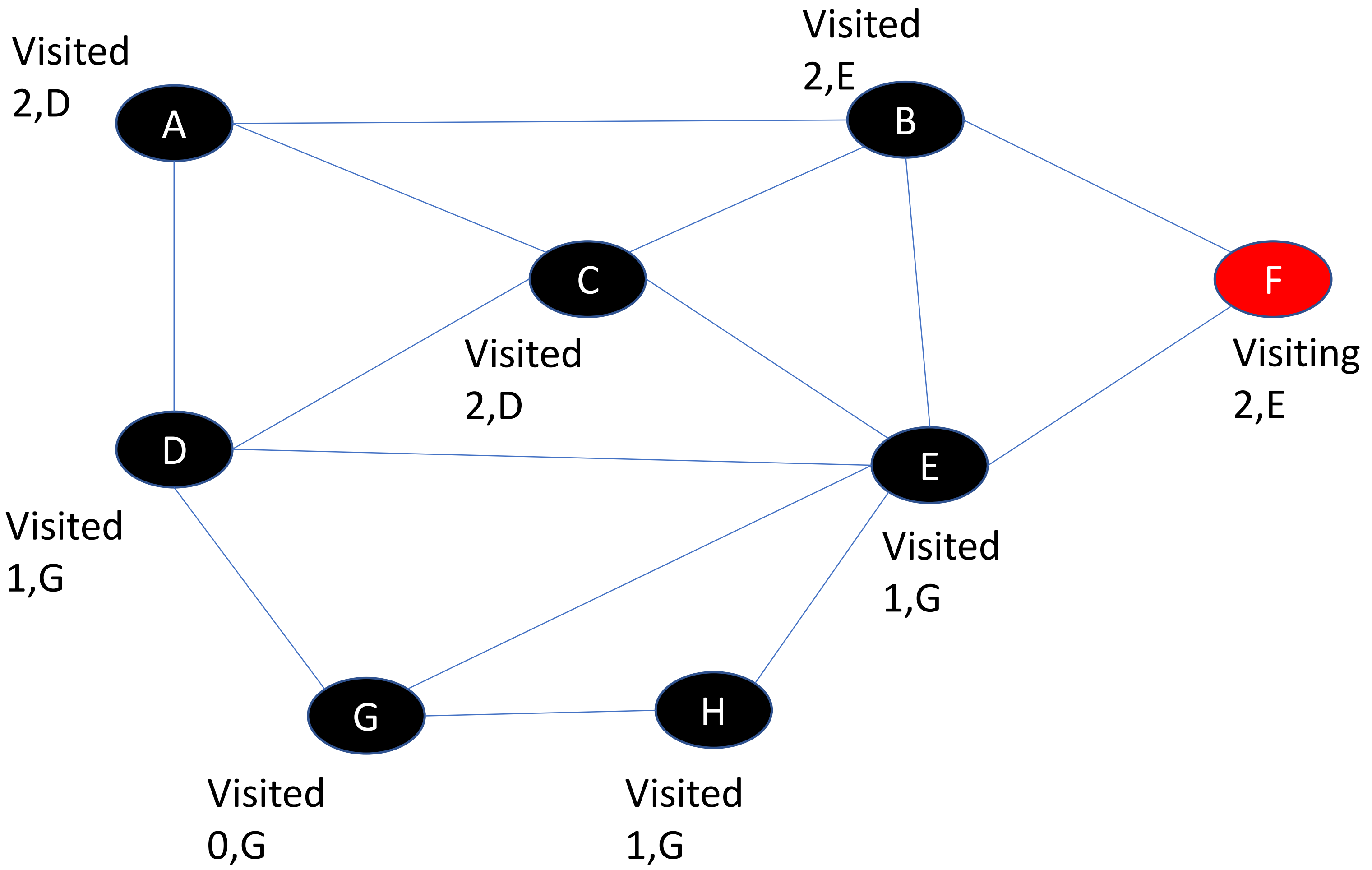

Visiting
2,D
A
In queue
B
C
In queue
F
In queue
D
Visited
1,G
E
Visited
1,G
G
H
Visited
0,G
Visited
1,G
QUEUE:
G
D
E
H
A <-
C
B
F



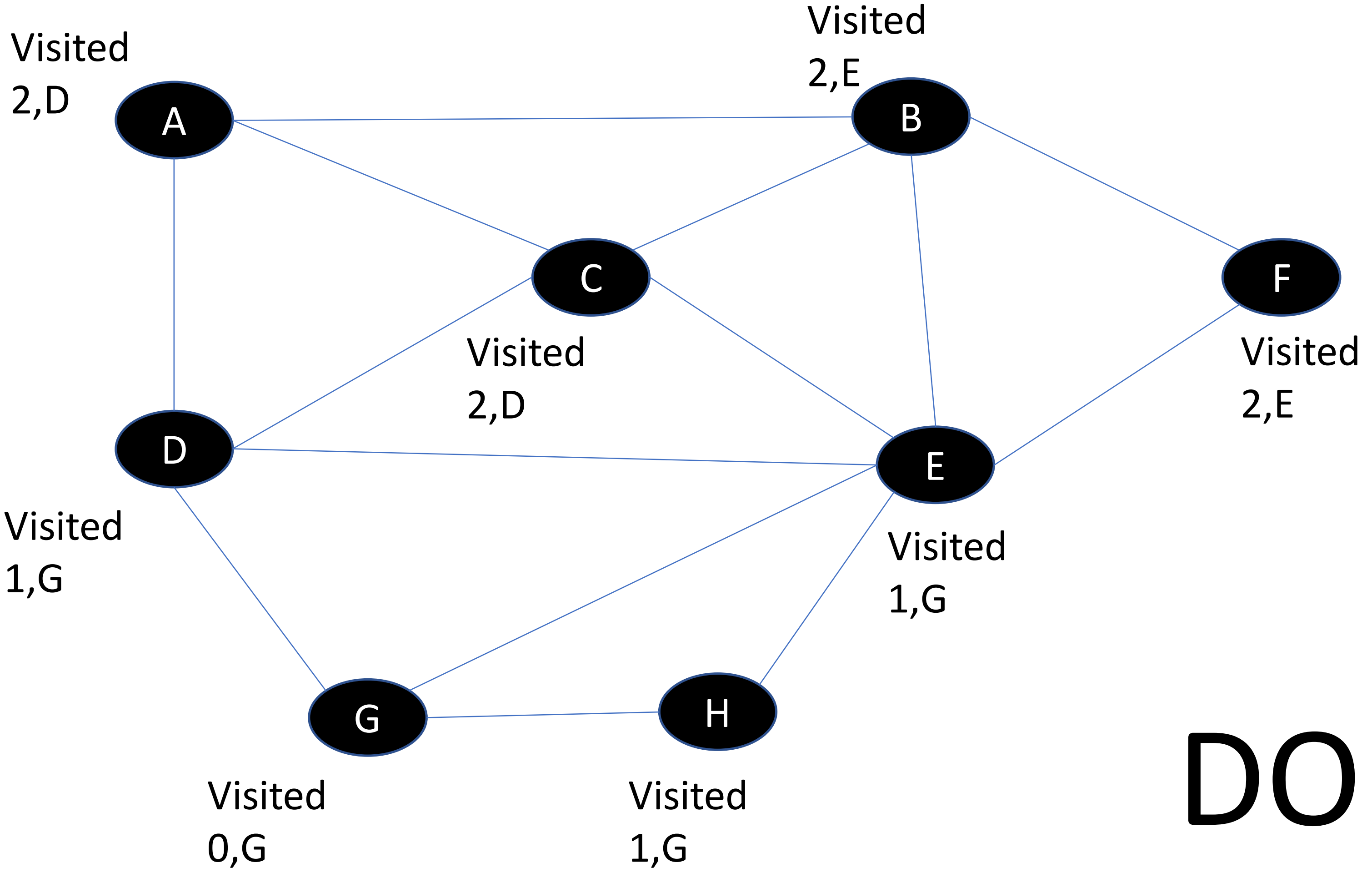
QUEUE:
G
D
E
H
A
C <-
B
F



QUEUE:
G
D
E
H
A
C
B <-
F



QUEUE:
G
D
E
H
A
C
B
F <-

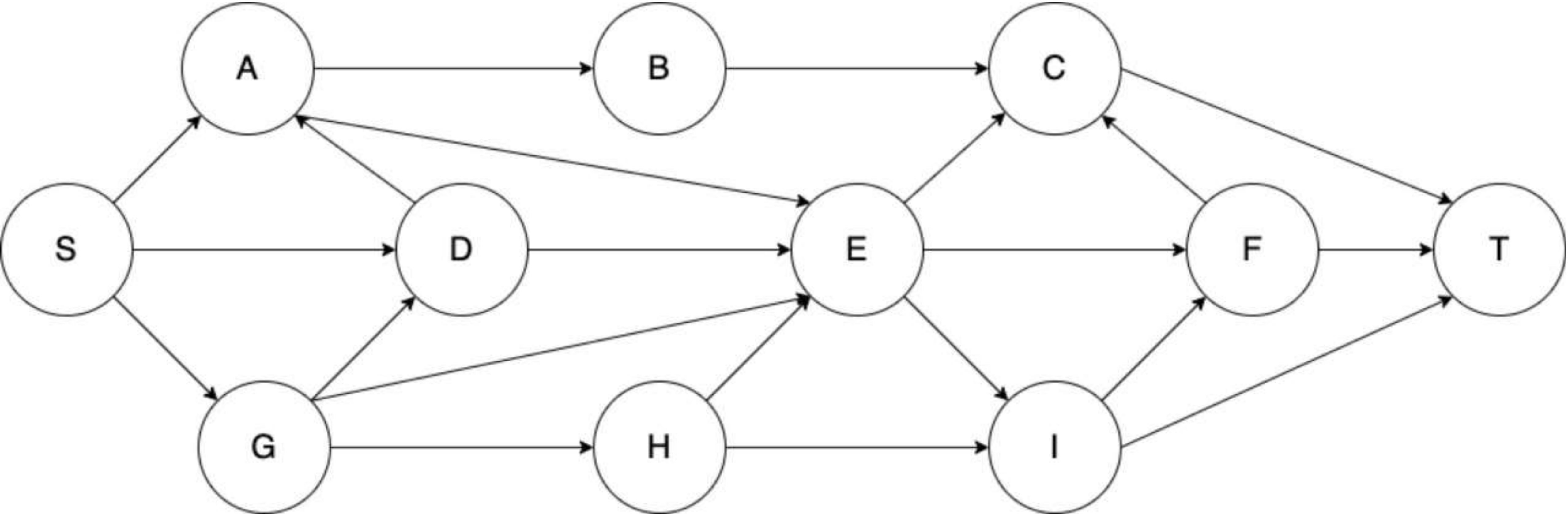


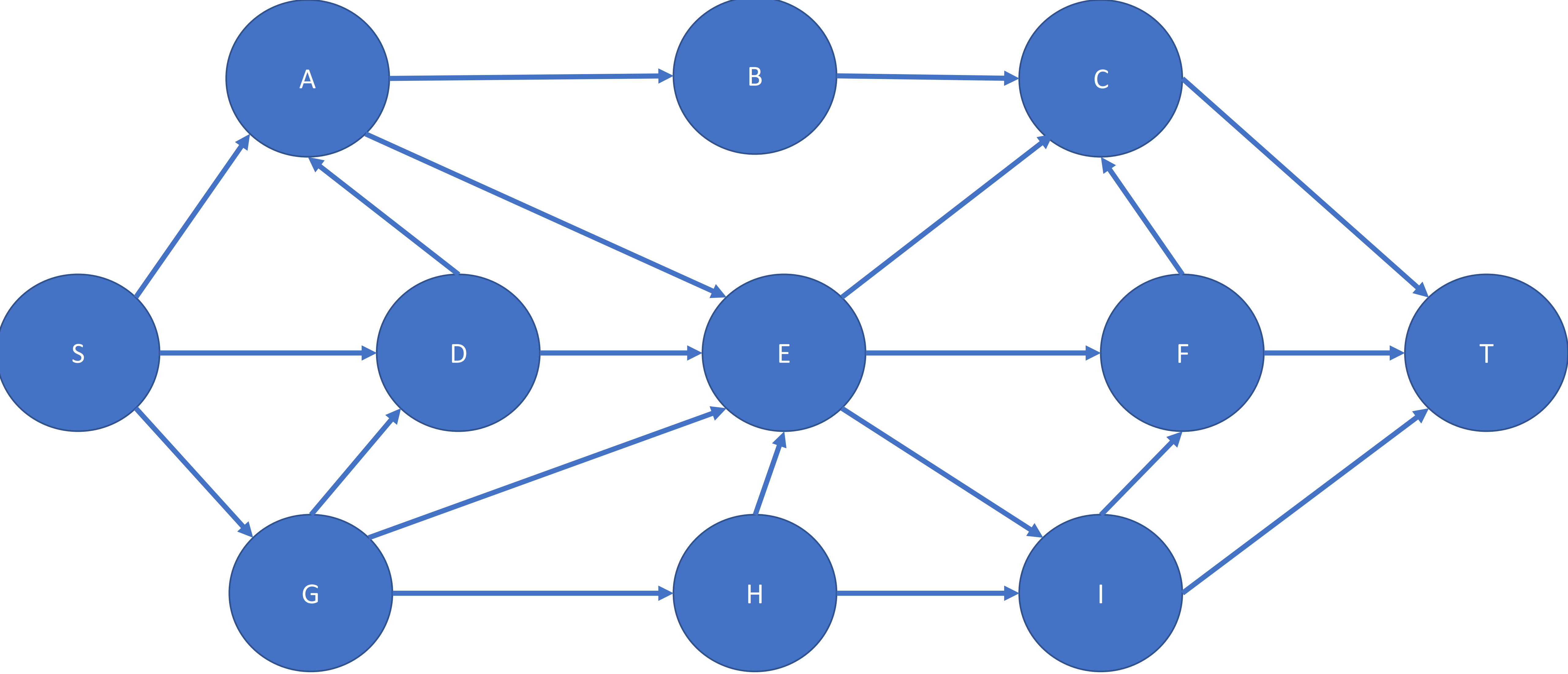
QUEUE:
G
D
E
H
A
C
B
F

DONE!

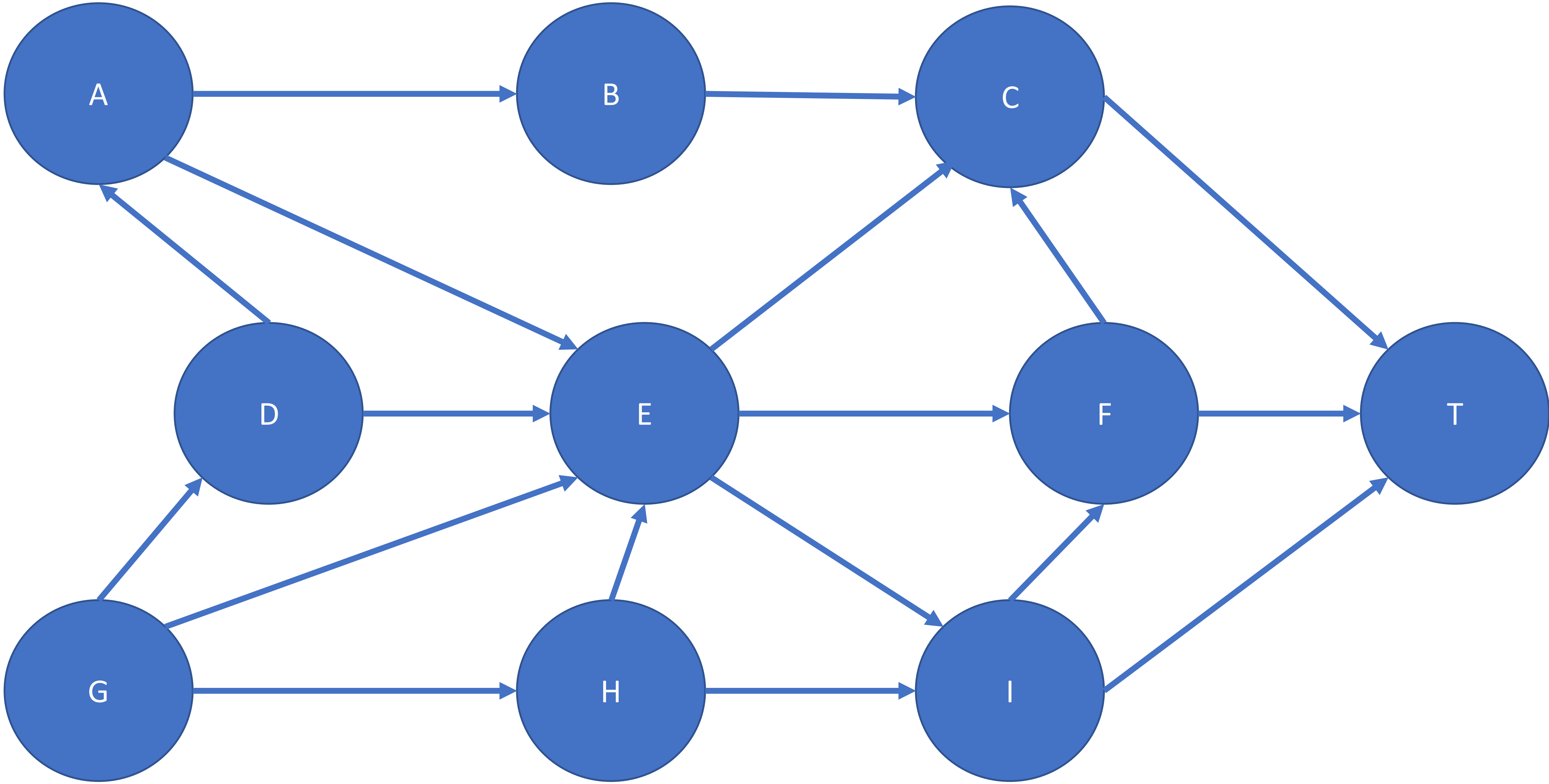
QUESTION 5

Topological order



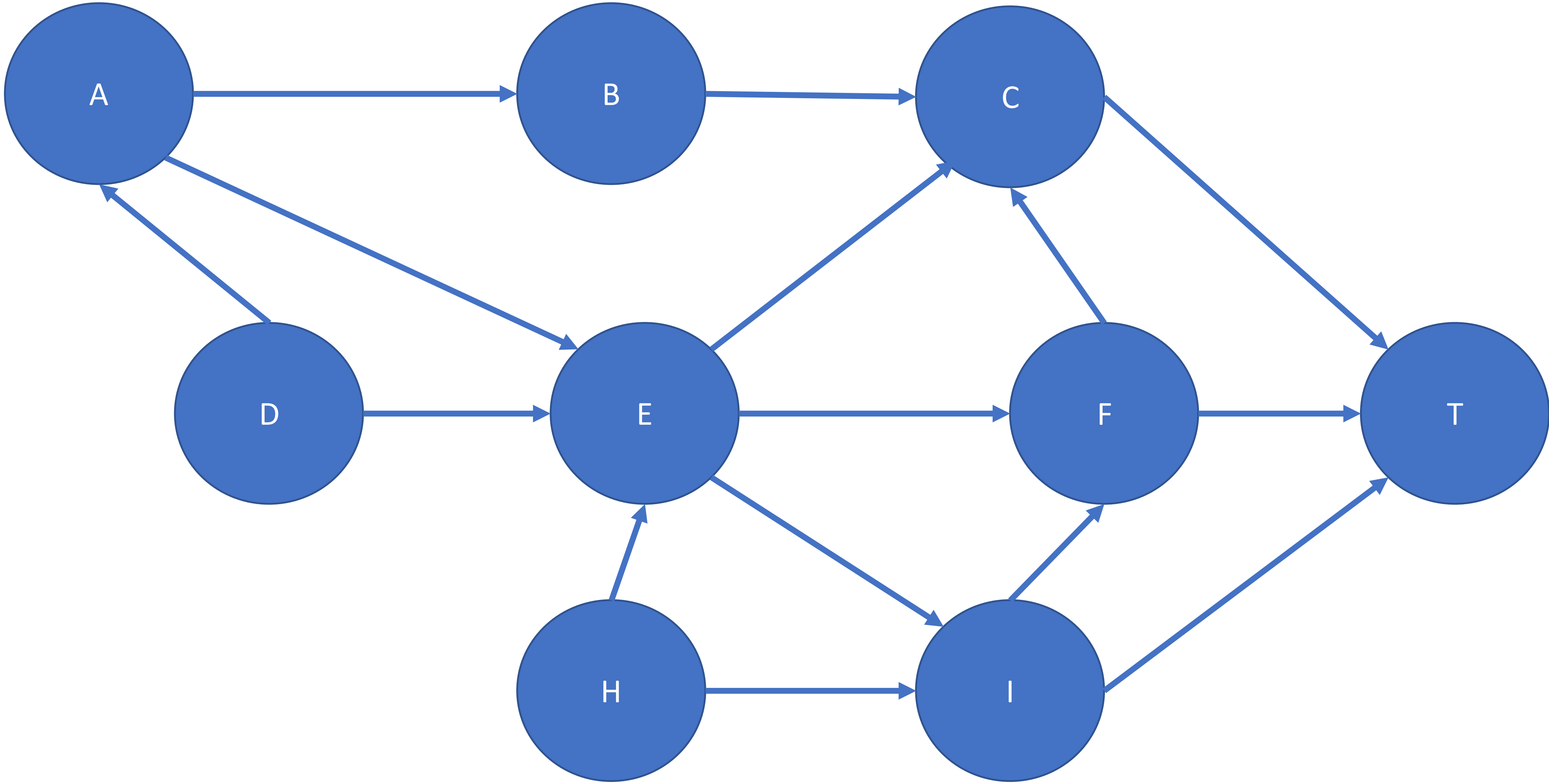

A
B
C
S
D
E
F
T
G
H
I

Select a vertex with indegree 0. In current graph, we select S. Add S to the order and remove from graph.



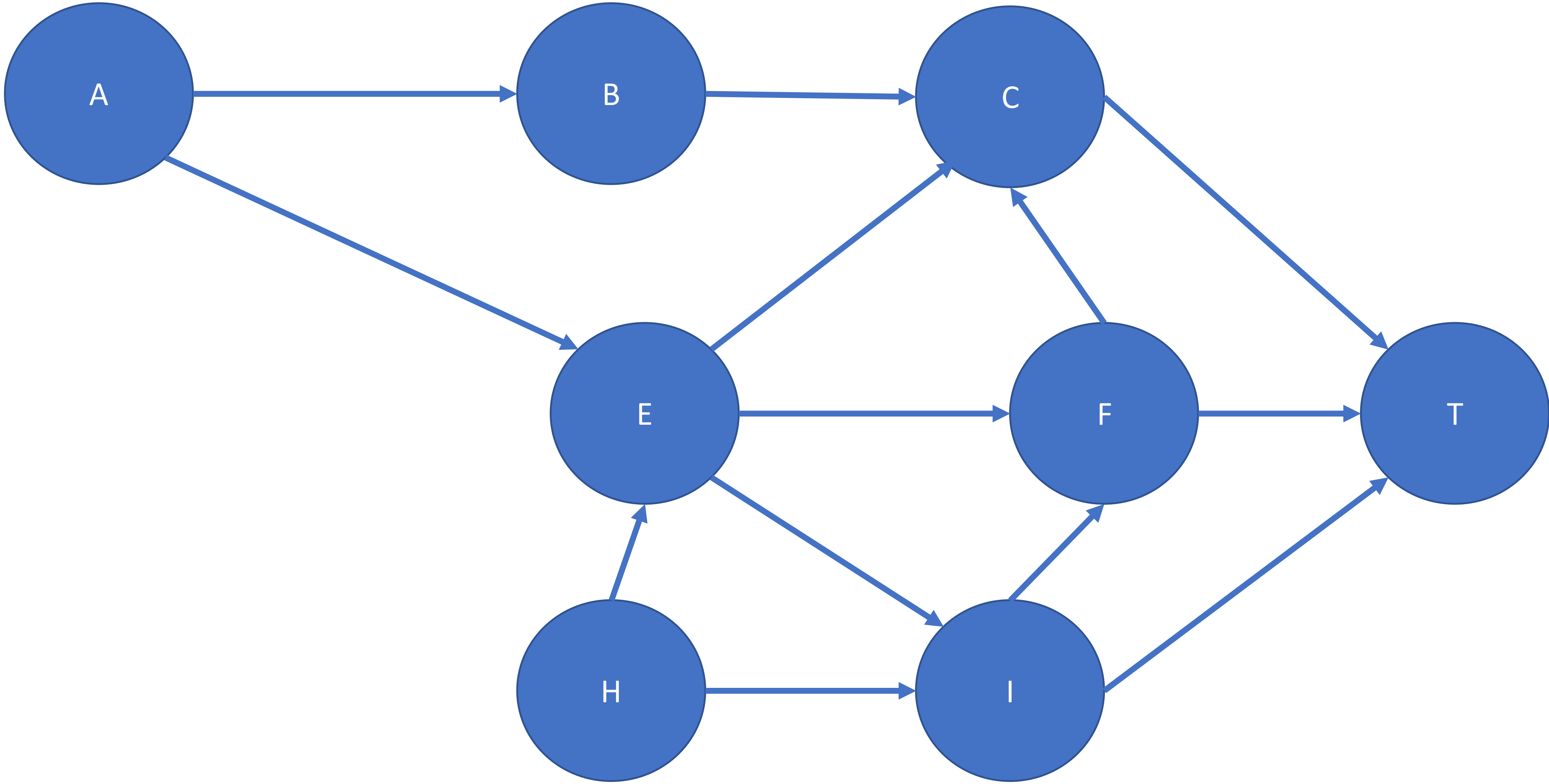
Order: S,

Select a vertex with indegree 0. In current graph, we select G. Add G to the order and remove from graph.



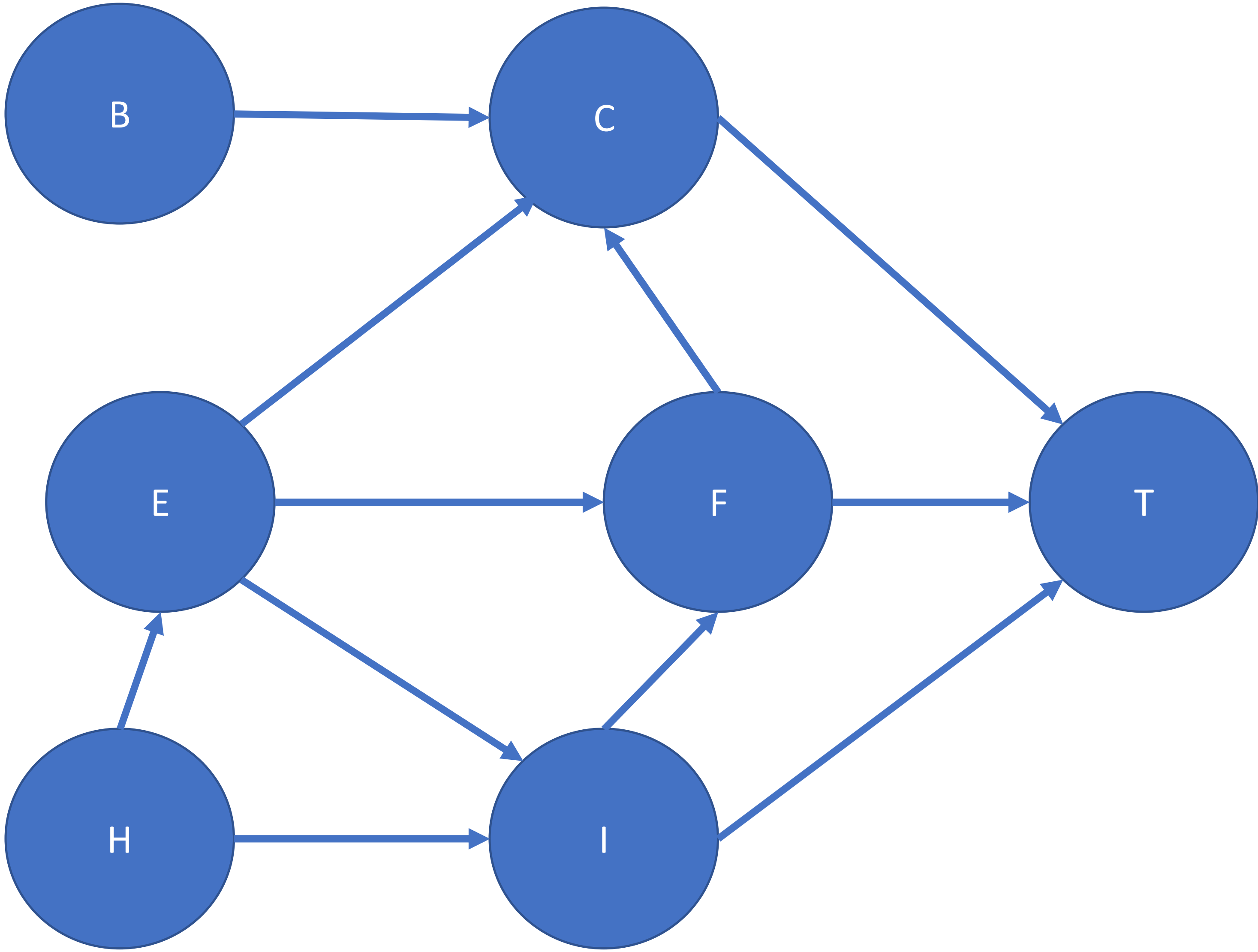
Order: S, G,

Select a vertex with indegree 0. In current graph, we select D. Add D to the order and remove from graph.



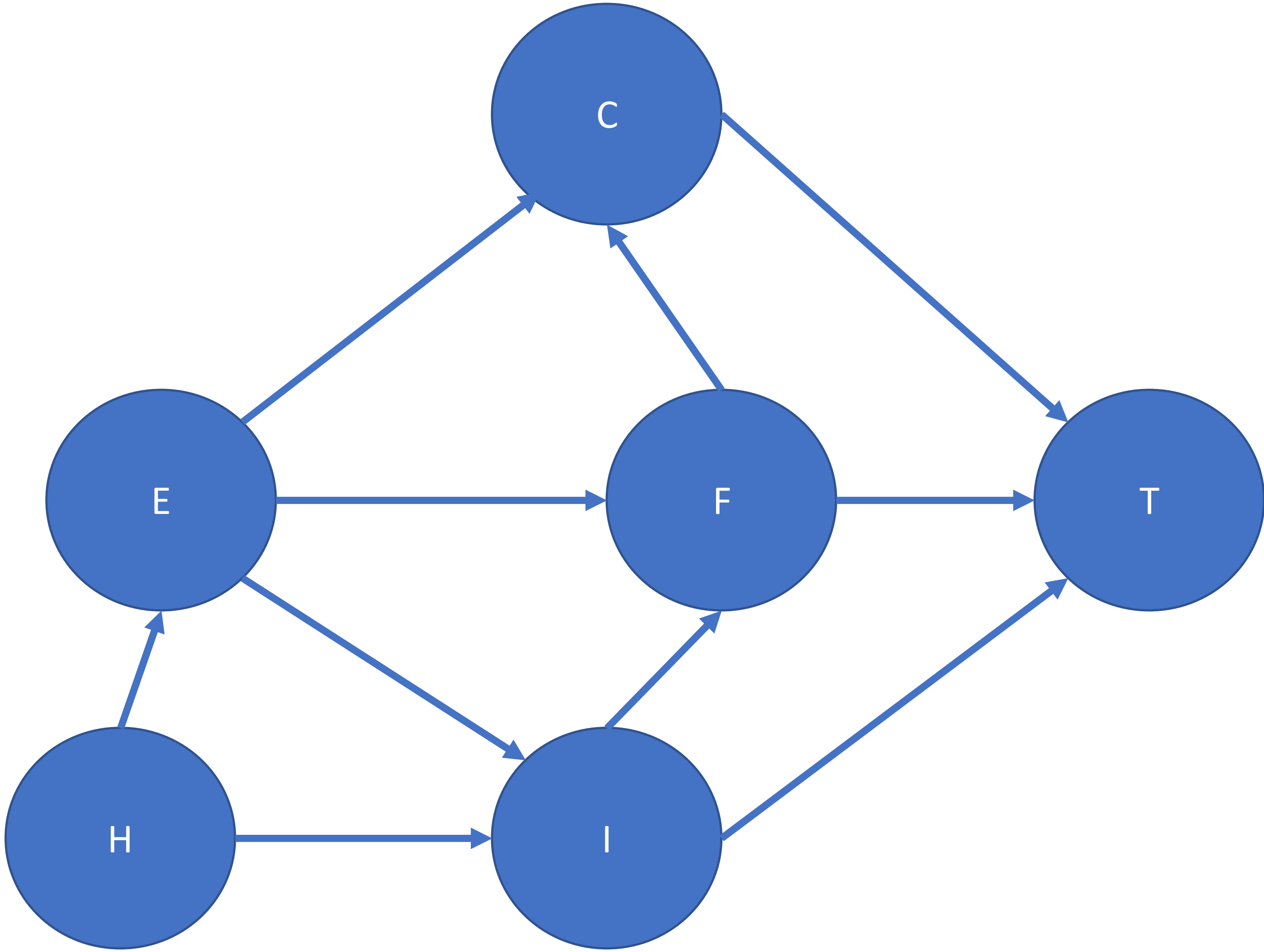
Order: S, G, D,

Select a vertex with indegree 0. In current graph, we select A. Add A to the order and remove from graph.



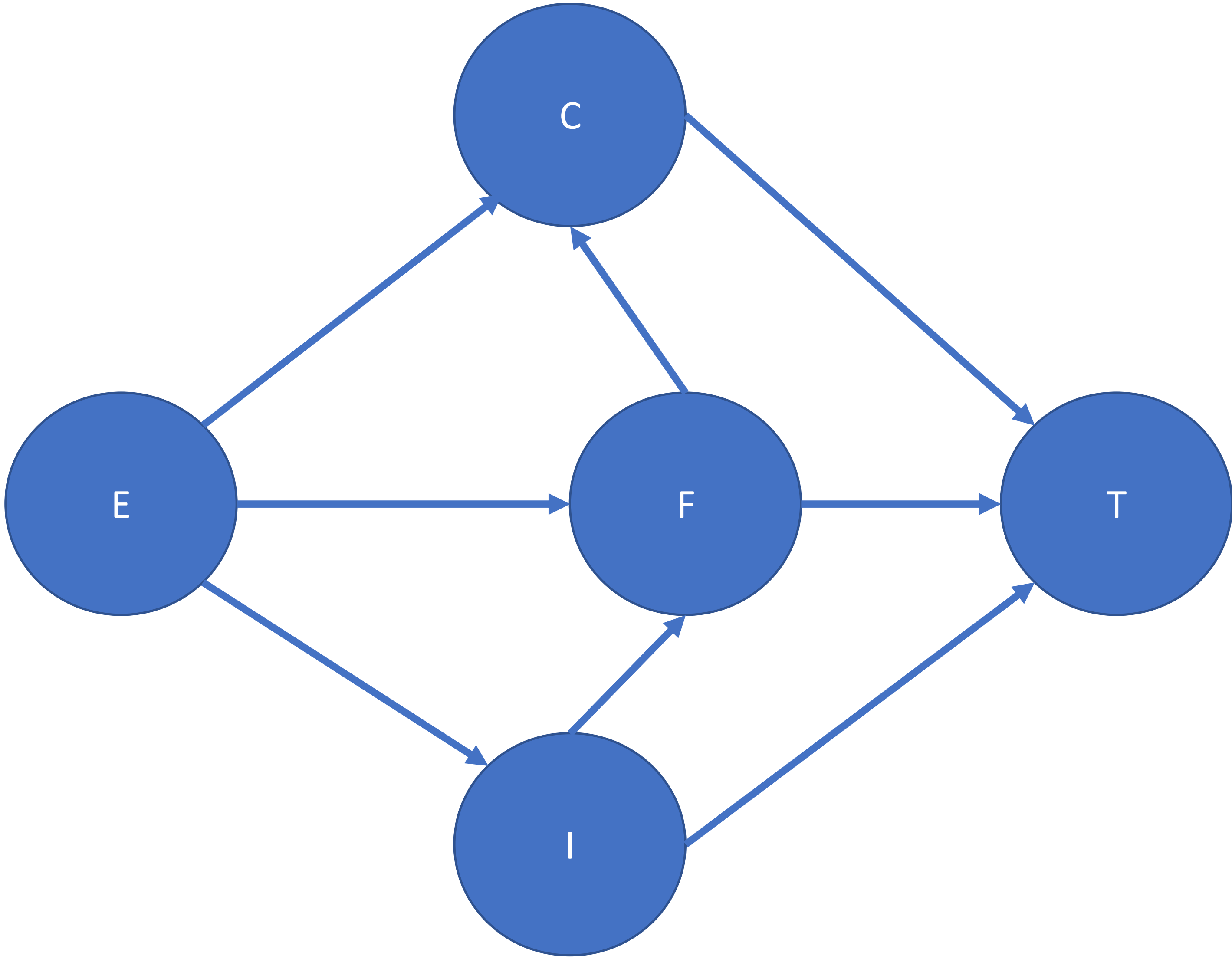
Order: S, G, D, A,

Select a vertex with indegree 0. In current graph, we select B. Add B to the order and remove from graph.



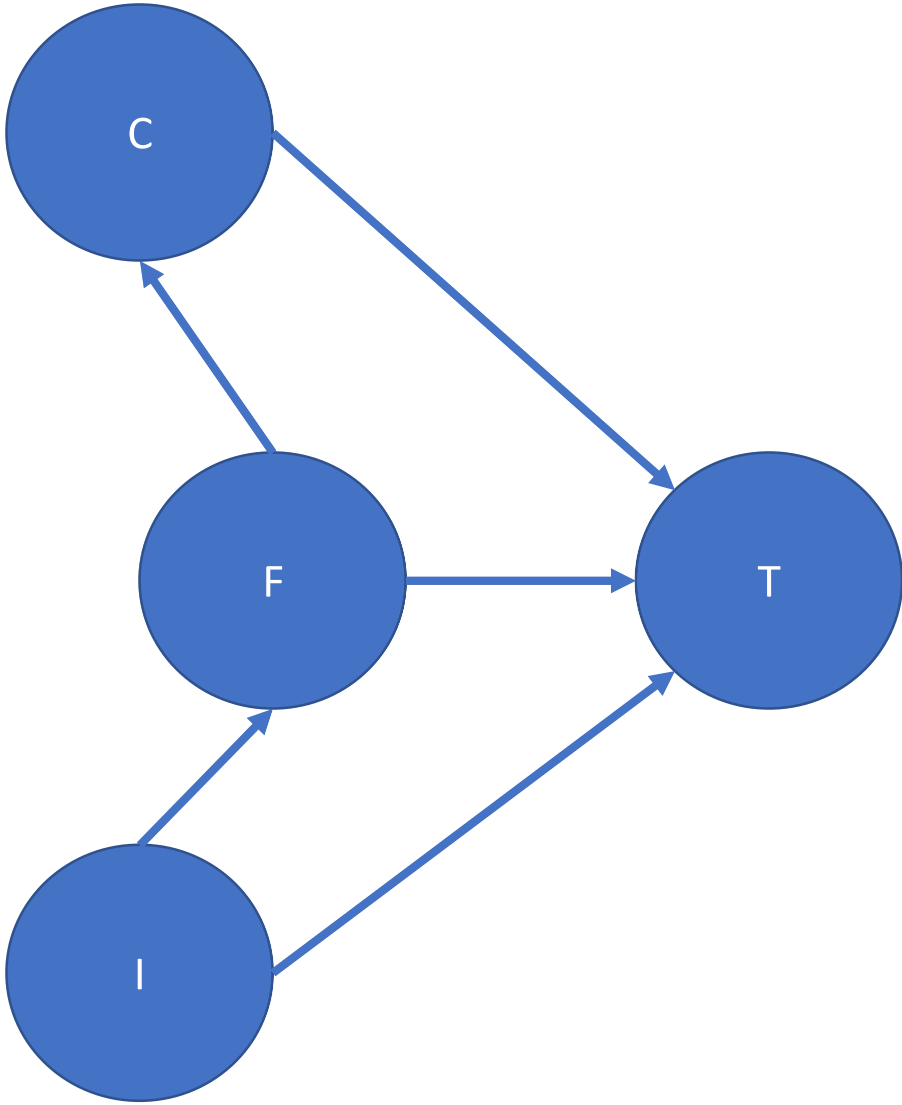
Order: S, G, D, A, B,

Select a vertex with indegree 0. In current graph, we select H. Add H to the order and remove from graph.



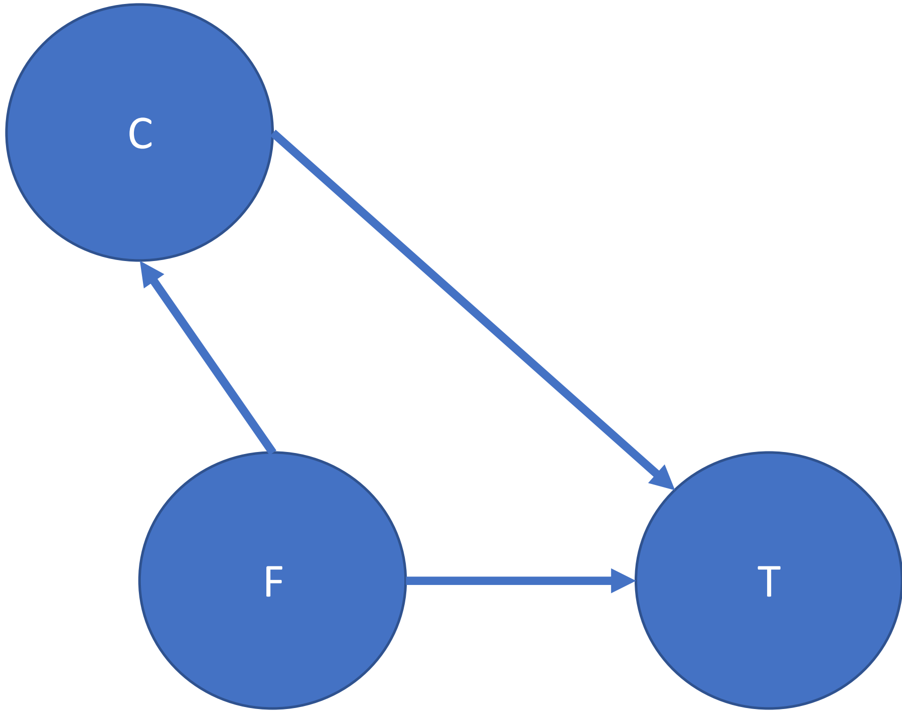
Order: S, G, D, A, B, H,

Select a vertex with indegree 0. In current graph, we select E. Add E to the order and remove from graph.



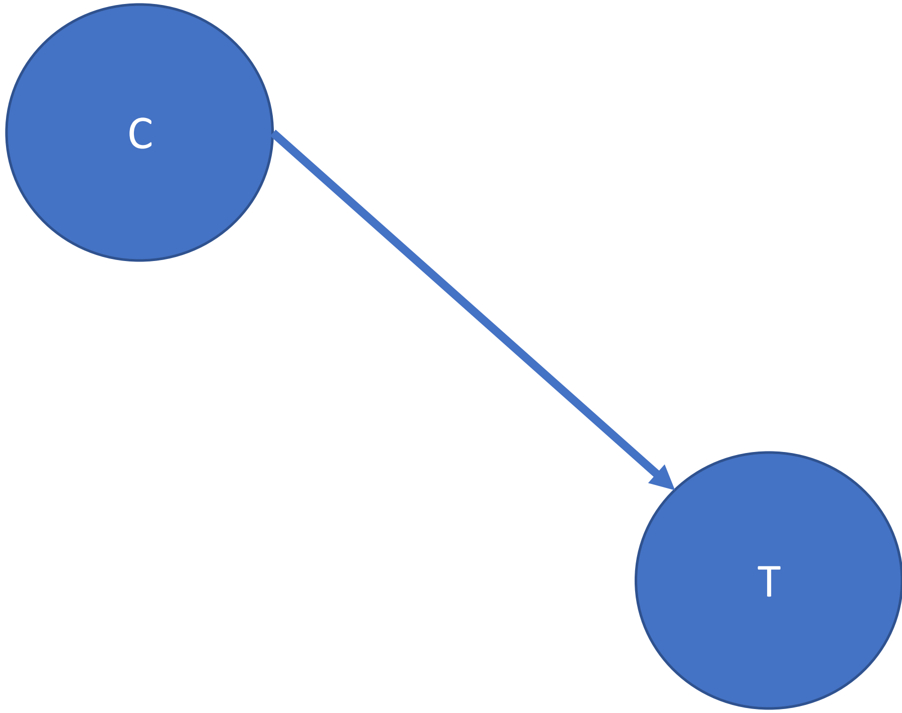
Order: S, G, D, A, B, H, E,

Select a vertex with indegree 0. In current graph, we select I. Add I to the order and remove from graph.



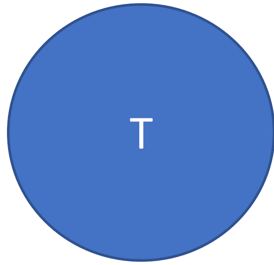
Order: S, G, D, A, B, H, E, I,

Select a vertex with indegree 0. In current graph, we select F. Add F to the order and remove from graph.



Order: S, G, D, A, B, H, E, I, F,

Select a vertex with indegree 0. In current graph, we select C. Add C to the order and remove from graph.



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

Select a vertex with indegree 0. In current graph, we select T. Add T to the order and remove from graph.

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

Order: S, G, D, A, B, H, E, I, F, C, T
is a valid order.