Assignment 7 - Key

1. Find a topological ordering for the graph in Figure 9.81

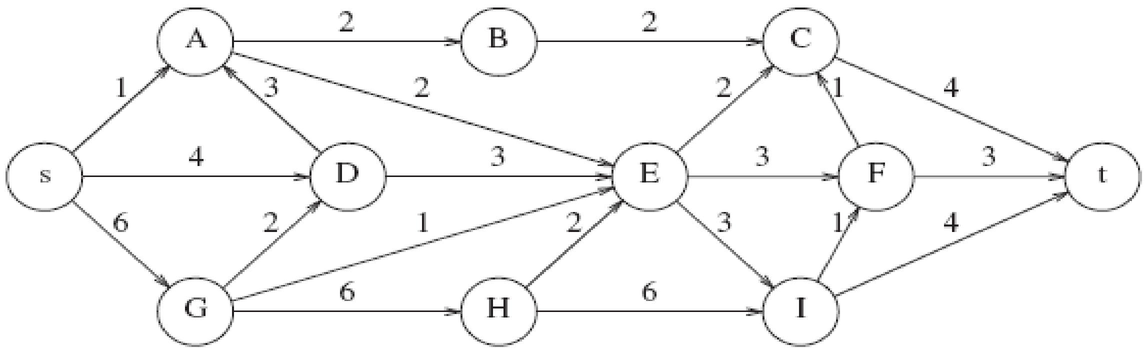
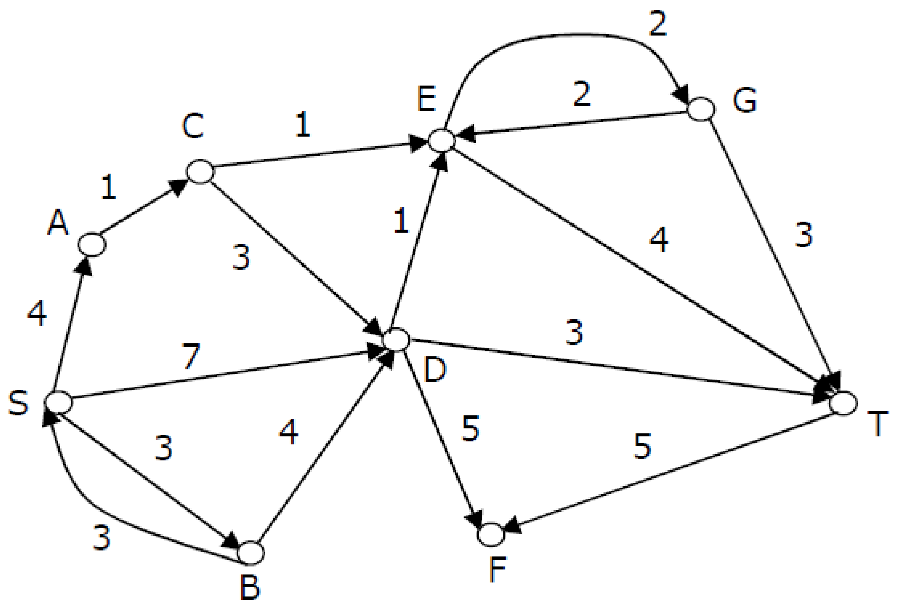


Figure 9.81 Graph used in Exercises 9.1 and 9.11

The following ordering is arrived at by using a queue and assumes that vertices appear on an adjacency list alphabetically. The topological order that results is then

s, G, D, H, A, B, E, I, F, C, t

2. Consider the directed graph shown in the figure below. Apply Dijkstra's algorithm with vertex S as source and find the shortest path to all other vertices. Show all steps. **[10 points]**



Initially make every distance ∞ other than S.

Step 1

Vertex	Known	Distance	Path
S	F	0	Null
A	F	∞	Null
B	F	∞	Null
C	F	∞	Null
D	F	∞	Null
E	F	∞	Null
F	F	∞	Null
G	F	∞	Null
T	F	∞	Null

Step 2

Vertex	Known	Distance	Path
S	**T**	**0**	**Null**
A	F	4	S
B	F	3	S
C	F	∞	Null
D	F	7	S
E	F	∞	Null
F	F	∞	Null
G	F	∞	Null
T	F	∞	Null

Step 3

Vertex	Known	Distance	Path
S	T	0	Null
A	F	4	S
B	**T**	**3**	**S**
C	F	∞	Null
D	F	7	S
E	F	∞	Null
F	F	∞	Null
G	F	∞	Null
T	F	∞	Null

Step 4

Vertex	Known	Distance	Path
S	T	0	Null
A	**T**	**4**	**S**
B	T	3	S
C	F	5	A
D	F	7	S
E	F	∞	Null
F	F	∞	Null
G	F	∞	Null
T	F	∞	Null

Step 5

Vertex	Known	Distance	Path
S	T	0	Null
A	T	4	S
B	T	3	S
C	**T**	**5**	**A**
D	F	7	S
E	F	6	C
F	F	∞	Null
G	F	∞	Null
T	F	∞	Null

Step 6

Vertex	Known	Distance	Path
S	T	0	Null
A	T	4	S
B	T	3	S
C	T	5	A
D	F	7	S
E	**T**	**6**	**C**
F	F	∞	Null
G	F	8	E
T	F	10	E

Step 7

Vertex	Known	Distance	Path
S	T	0	Null
A	T	4	S

B	T	3	S
C	T	5	A
D	**T**	**7**	**S**
E	T	6	C
F	F	12	D
G	F	8	E
T	F	10	E

Step 8

Vertex	Known	Distance	Path
S	T	0	Null
A	T	4	S
B	T	3	S
C	T	5	A
D	T	7	S
E	T	6	C
F	F	12	D
G	**T**	**8**	**E**
T	F	10	D

Step 9

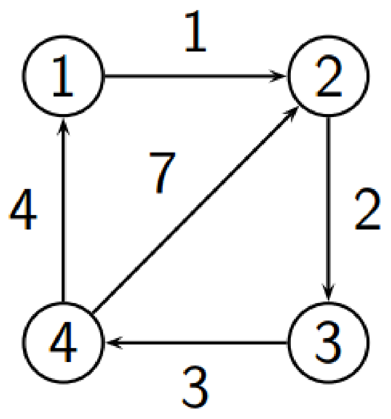
Vertex	Known	Distance	Path
S	T	0	Null
A	T	4	S
B	T	3	S
C	T	5	A
D	T	7	S
E	T	6	C
F	F	12	D
G	T	8	E
T	**T**	**10**	**D**

Step 10

Vertex	Known	Distance	Path
S	T	0	Null
A	T	4	S
B	T	3	S
C	T	5	A
D	T	7	S

E	T	6	C
F	**T**	**12**	**D**
G	T	8	E
T	T	10	D

3. Use Floyd-Warshall Algorithm to get all-pair shortest path for the following graph.



D(0)					***P(0)***				
	1	***2***	***3***	***4***		***1***	***2***	***3***	***4***
1	0	1	∞	∞	**1**	1	1	∞	∞
2	∞	0	2	∞	**2**	∞	2	2	∞
3	∞	∞	0	3	**3**	∞	∞	3	3
4	4	7	∞	0	**4**	4	4	∞	4

D(1)					***P(1)***				
	1	***2***	***3***	***4***		***1***	***2***	***3***	***4***
1	0	1	∞	∞	**1**	1	1	∞	∞
2	∞	0	2	∞	**2**	∞	2	2	∞

3	∞	∞	0	3	**3**	∞	∞	3	3
4	4	*5*	∞	0	**4**	4	*1*	∞	4

D(2)					***P(2)***				
	1	***2***	***3***	***4***		***1***	***2***	***3***	***4***
1	0	1	*3*	∞	**1**	1	1	*2*	∞
2	∞	0	2	∞	**2**	∞	2	2	∞
3	∞	∞	0	3	**3**	∞	∞	3	3
4	4	5	*7*	0	**4**	4	1	*2*	4

D(3)					***P(3)***				
	1	***2***	***3***	***4***		***1***	***2***	***3***	***4***
1	0	1	3	*6*	**1**	1	1	2	*3*
2	∞	0	2	*5*	**2**	∞	2	2	*3*
3	∞	∞	0	3	**3**	∞	∞	3	3
4	4	5	7	0	**4**	4	1	2	4

D(4)					***P(4)***				
	1	***2***	***3***	***4***		***1***	***2***	***3***	***4***
1	0	1	3	6	**1**	1	1	2	3
2	*9*	0	2	5	**2**	*4*	2	2	3
3	*7*	*8*	0	3	**3**	*4*	*1*	3	3
4	4	5	7	0	**4**	4	1	2	4

Final distance matrix

	1	***2***	***3***	***4***
1	0	1	3	6
2	9	0	2	5
3	7	8	0	3

4	4	5	7	0

Final Path Matrix

	1	***2***	***3***	***4***
1	1	1	2	3
2	4	2	2	3
3	4	1	3	3
4	4	1	2	4

4. The input is a list of league game scores (and there are no ties). If all teams have at least one win and a loss, we can generally prove, by a silly transitivity argument, that any team is better than any other. For instance, in the six-team league where everyone plays three games, suppose we have the following results: A beat B and C; B beat C and F; C beat D; D beat E; E beat A; F beat D and E. Then we can prove that A is better than F, because A beat B, who in turn beat F. Similarly, we can prove that F is better than A because F beat E and E beat A.

   Given a list of game scores and two teams X and Y, either find a proof (if one exists) that X is better than Y, or Y is better than X, or indicate that no proof of this form can be found.

[5 points]

i) Given the list of games create a directed graph, where each team is a vertex. There will be a directed edge from the team that won to the team that lost.
ii) Once the graph is constructed do the following,
    (1) If there is a path from X to Y but none from Y to X then declare X is better than Y.
    (2) If there is a path from Y to X but none from X to Y then declare Y is better than X
    (3) If there is a path from both X to Y and Y to X, then declare no proof of the form can be found.

5. A student needs to take a certain number of courses to graduate, and these courses have prerequisites that must be followed. Assume that all courses are offered every semester and that the student can take an unlimited number of courses per semester. Given a list of courses and their prerequisites, compute a schedule that requires the minimum number of semesters.

[5 points]

Given the list of courses create a directed acyclic graph. Each course is a vertex; if *X* is a prerequisite for *Y*, draw an edge from *X* to *Y* Find the longest path in the acyclic graph.

6. Give an algorithm to find a maximum spanning tree. Is this harder than finding a minimum spanning tree.

   Since the minimum spanning tree algorithm works for negative edge costs, an obvious solution is to replace all the edge costs by their negatives and use the minimum spanning tree algorithm. Alternatively, change the logic so that < is replaced by >, *min* by *max*, and vice versa.

7. Find all the articulation points in the graph in Figure 9.85. Show the depth-first spanning tree and the values of Num and Low for each vertex.

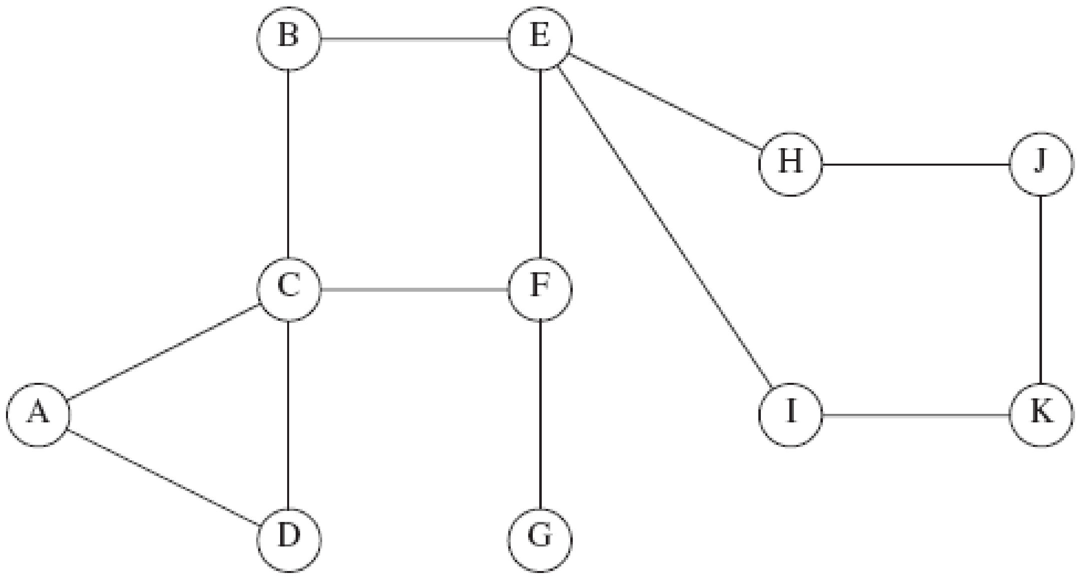
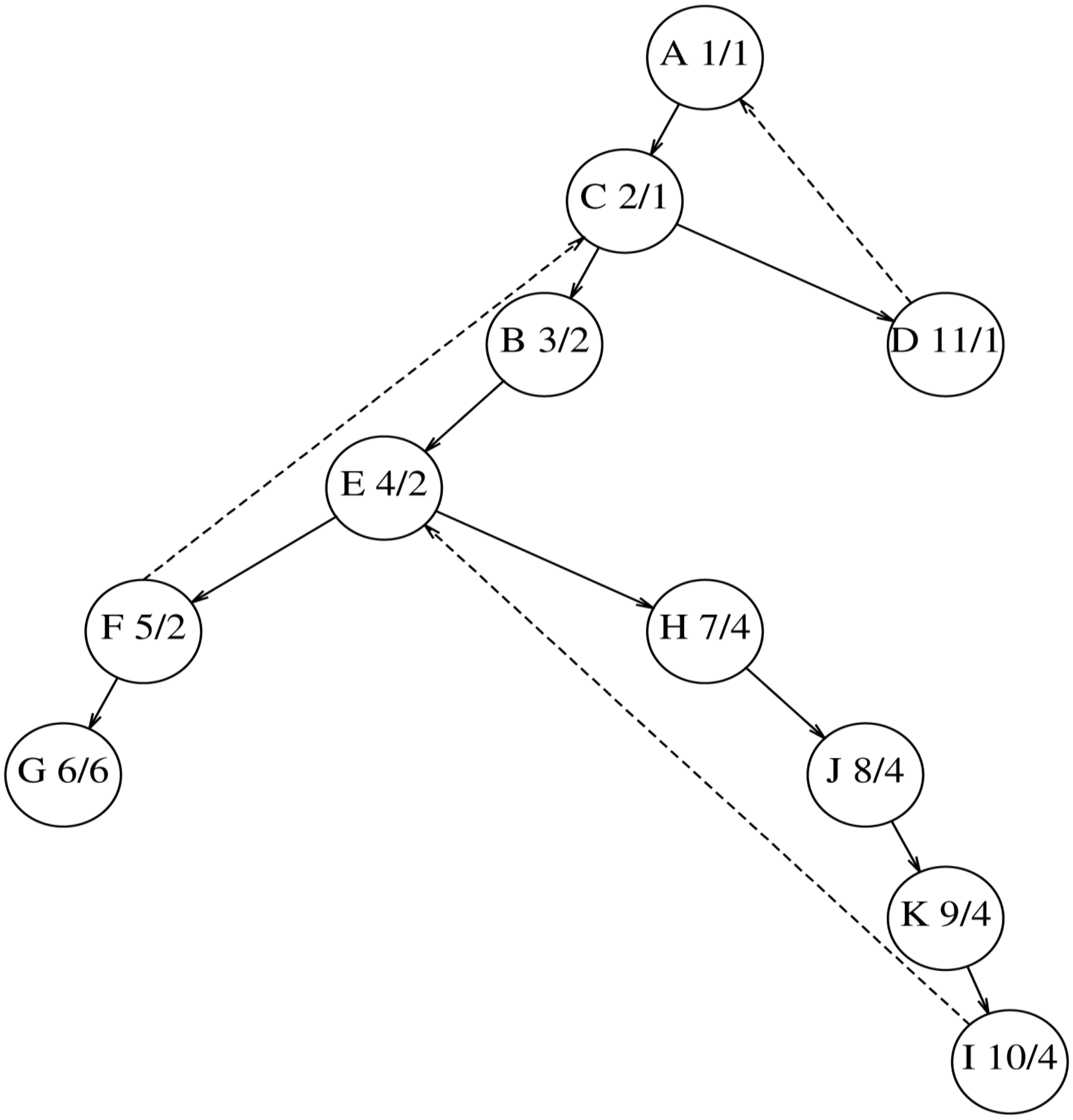


Figure 9.85 Graph used in Exercise 9.21

We start the depth-first search at *A* and visit adjacent vertices alphabetically. The articulation points are *C*, *E*, and *F*. *C* is an articulation point because $low[B] \geq num[C]$; *E* is an articulation point because *low*[*H*] ≥ *num*[*E*]; and *F* is an articulation point because *low*[*G*] ≥ *num*[*F*]. The depth-first spanning tree is shown in the following Figure.


A 1/1
C 2/1
B 3/2
D 11/1
E 4/2
F 5/2
H 7/4
G 6/6
J 8/4
K 9/4
I 10/4