Student Information

Full Name : Hasan Küreli
Id Number : 2580751

Answer 1

a)

$$deg(a) = 3, deg(b) = 3, deg(c) = 3, deg(d) = 2, deg(e) = 3, sum = 14$$

b)

Adjacency matrix, Columns and rows are a, b, c, d, e:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

There are 14 non-zero entries.

c)

Incidence matrix, Columns are edges: {a,b}, {a,c}, {a,e}, {b,c}, {b,e}, {c,d}, {e,d
Rows are vertices: a, b, c, d, e respectively.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

There are 21 zero entries.

d)

No. The adjacency matrix of a complete graph has all non-zero enties except for the diagonal. But if we try to remove any one of the vertices (just 1) and remove corresponding rows and columns in the matrix in part (b) we can't reach this form.

e)

No. Let's start giving colors blue and red for the vertices with the condition that no adjacent vertex has the same color. Color b red, a blue, c red, we can see it is not bipartite because b and c are adjacent but have the same color red. So, it is not bipartite.

f)

A directed graph might have each of its edges directed in two different ways. Since, G has 7 edges $2^7 = 128$ different directed graphs have G as their underlying undirected graph.

g)

a-b-c-d-e-a-c is the path.
The length is 6.

h)

1. Since G is a connected graph there is only 1 connected component.

i)

No. Because, G has 4 vertices with odd degree it should have 0.

j)

No. Because, G has 4 vertices with odd degree vertices it should have 0 or 2.

k)

Yes. The circuit is: a-e-d-c-b-a.

l)

Yes. The path is: a-e-d-c-b.

Answer 2

They are isomorphic.
Because we can find a one-to-one and onto function f from G to H such that:
$f(a) = a^{'}$
$f(b) = b^{'}$
$f(c) = c^{'}$

$f(d) = d^{'}$
$f(e) = e^{'}$
And all edges in G are also there in H using this function (a and b are adjacent so is f(a) and f(b)).

Answer 3

Visited: A set of the visited vertices which we will not consider again.
Unvisited: A set of unvisited vertices.
Distance: array that stores distance of each vertex.
Previous: array that stores the previous of each vertex.

0) We initialize distance of s to 0 and other to infinity. Add all elements to Unvisited.

Vertex	Distance	Previous
s	0	NULL
t	∞	NULL
u	∞	NULL
v	∞	NULL
w	∞	NULL
x	∞	NULL
y	∞	NULL
z	∞	NULL

Visited = {}
Unvisited= { s,t,u,v,w,x,y,z}

1) We check the unvisited neigbours of the unvisited vertex with minimum distance which is s and update their distance (if the new distance is lower) and previous. Move s to visited.

Vertex	Distance	Previous
s	0	NULL
t	∞	NULL
u	4	s
v	5	s
w	3	s
x	∞	NULL
y	∞	NULL
z	∞	NULL

Visited = {s}
Unvisited= { t,u,v,w,x,y,z}

2) We check the neigbours of the unvisited vertex with minimum distance which is w and update their distance and previous. Move w to visited.

Vertex	Distance	Previous
s	0	NULL
t	∞	NULL
u	4	s
v	5	s
w	3	s
x	11	w
y	∞	NULL
z	15	w

Visited = {s,w}
Unvisited= { t,u,v,x,y,z}

3) We check the neigbours of the unvisited vertex with minimum distance which is u and update their distance and previous. Move u to visited.

Vertex	Distance	Previous
s	0	NULL
t	∞	NULL
u	4	s
v	5	s
w	3	s
x	11	w
y	15	u
z	15	w

Visited = {s,w,u}
Unvisited= { t,v,x,y,z}

4) We check the neigbours of the unvisited vertex with minimum distance which is v and update their distance and previous. Move v to visited.

Vertex	Distance	Previous
s	0	NULL
t	∞	NULL
u	4	s
v	5	s
w	3	s
x	7	v
y	11	v
z	15	w

Visited = {s,w,u,v}
Unvisited= { t,x,y,z}

5) We check the neigbours of the unvisited vertex with minimum distance which is x and update their distance and previous. Move x to visited.

Vertex	Distance	Previous
s	0	NULL
t	∞	NULL
u	4	s
v	5	s
w	3	s
x	7	v
y	8	x
z	13	x

Visited = {s,w,u,v,x}
Unvisited= { t,y,z}

6) We check the neigbours of the unvisited vertex with minimum distance which is y and update their distance and previous. Move y to visited.

Vertex	Distance	Previous
s	0	NULL
t	17	y
u	4	s
v	5	s
w	3	s
x	7	v
y	8	x
z	12	y

Visited = {s,w,u,v,x,y}
Unvisited= { t,z}

7) We check the neigbours of the unvisited vertex with minimum distance which is z and update their distance and previous. Move z to visited.

Vertex	Distance	Previous
s	0	NULL
t	15	z
u	4	s
v	5	s
w	3	s
x	7	v
y	8	x
z	12	y

Visited = {s,w,u,v,x,y,z}
Unvisited= { t}
We have reached vertex t so we can stop.
The shortest path is found by following the previous:
s-v-x-y-z-t

distance is 15

Answer 4

a)

I will use Kruskal's algorithm.
I remove all the edges then start adding them from the edges with minimum weight.
There are multiple edges with weight 2. I add them by checking that they don't form any loops.
The graph looks like below after this:

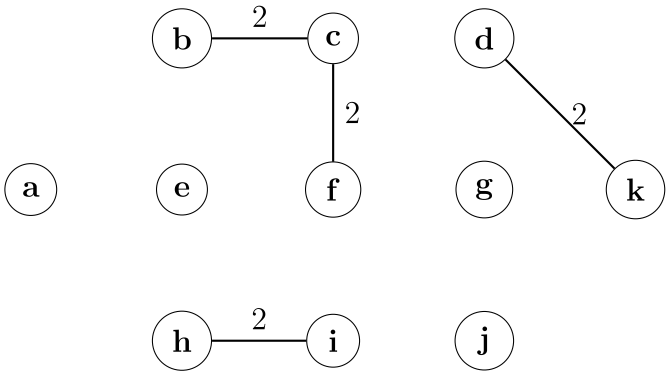

Figure 1: After some edges are added

And then I continue adding the remaining edges with minimum weight by checking the loop condition.
The order I add them will be : {b,c}, {c,f}, {h,i}, {d,k}, {c,d},{a,b},{f,j}, {f,i}, {e,f}, and lastly either {g,j} or {f,g} I choose {f,g}
Note: The question did not ask to draw all the steps so I did not show all steps clearly. But I just drew an intermediate step.

b)

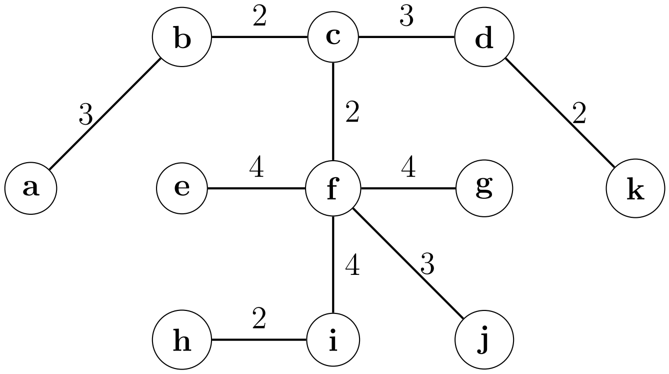

Figure 2: The minimum spanning tree

c)

No it is not. I said in part(a) that at the last step I can add two different edges {g,j} or {f,g} this choice will result in different minimum spanning trees. Hence, it is not unique.