Student Information

Full Name : Damlanur Yağdı
Id Number : 2522118

Answer 1

a)

Degrees of all nodes:
$a \to 3$
$b \to 3$
$c \to 3$
$d \to 2$
$e \to 3$
Hence, the sum of them = 14.

b)

Adjacency matrix representation of G

	a	b	c	d	e
a	0	1	1	0	1
b	1	0	1	0	1
c	1	1	0	1	0
d	0	0	1	0	1
e	1	1	0	1	0

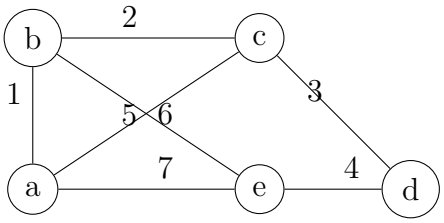
$$\downarrow$$

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

Hence, the number of non-zero entries in the adjacency matrix representation of $G = 14$.

c)

The rows of the matrix $[A_C]$ represent the number of nodes and the column of the matrix $[A_C]$ represent the number of branches in the given graph.
In the graph G, we have 5 nodes such as a, b, c, d, e.
There are also 7 branches, which we can represent as followings.



To avoid confusion, the branch between a and c = 6 , the branch between b and e = 5.

$[A_C]$=

	1	2	3	4	5	6	7
a	1	0	0	0	0	1	1
b	1	1	0	0	1	0	0
c	0	1	1	0	0	1	0
d	0	0	1	1	0	0	0
e	0	0	0	1	1	0	1

The number of zero entries in the incidence matrix = 21.

d)

According to the definition of the textbook, the complete graph of n vertices refers that an undirected graph with n vertices where each pair of vertices is connected by an edge. In the graph G, there is no complete graph of at least four vertices, since all of the vertices are not connected by an edge.
For example, if we choose the vertices a, b, c, and e since there is no edge between e and c there will not be a complete graph.

e)

According to the definition of the textbook, a bipartite graph refers to a graph with a vertex set that can be partitioned into subsets V_1 and V_2 so that each edge connects a vertex in V_1 and a vertex in V_2. The pair (V_1, V_2) is called a bipartition of V.
Let's create a vertex set for G that can be partitioned into two sets.

- Let's choose 3 vertices for V_1 and 2 vertices for V_2
  According to the definition of bipartite graphs, sets V_1 and V_2 should have the same size. Because in the graph G, every edge should be connected to a vertex in both V_1 and V_2. In this case, it can be clearly seen that every edge of G does not connect a vertex in V_1 and a vertex in V_2
  .
- We can create V_1 and V_2 with sizes (2,3) as well. Similarly, every edge of G does not connect a vertex in V_1 and a vertex V_2
  .

- Other options are to create two disjoint sets with the sizes (4,1) and (1,4), again every edge of G does not connect a vertex in V_1 and a vertex V_2.

Hence, G is not bipartite.

f)

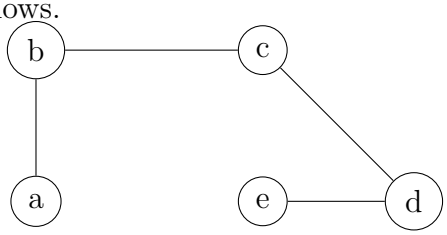
According to the definition of the textbook, the underlying undirected graph of a graph with directed edges refers to the undirected graph obtained by ignoring the directions of the edges.
To find the directed graphs, we can now consider all possibilities.
Every edge of G can be in two different forms. Let's consider the edge between a and b.

- First possibility is that the direction of the edge can be from a to b.
- Second possibility is that the direction of the edge can be from b to a.

Since G has 7 edges, there are 2 possibilities for 7 vertices. Hence there are 2^7 directed graphs that have G as their underlying undirected graph.

g)

The length of the simple longest path in G is 4, and the path is a, b, c, d, e. It can be visualized as follows.



h)

According to the definition of the textbook, a connected component of a graph G is a connected subgraph of G that is not a proper subgraph of another connected subgraph of G. That is, a connected component of a graph G is a maximal connected subgraph of G.
In this question, the number of connected components of G is 1, since every component can be extended until it becomes the graph G itself. Moreover, graph G is a subgraph of itself. So, the answer is 1.

i)

- Theorem 1 from section 10.5 of the textbook says that:

  A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has an even degree.

- Since the Graph G is a connected multigraph with 5 vertices, we should check if each vertex has even degrees.
- Degrees of the vertices:
  $a \to 3$
  $b \to 3$
  $c \to 3$
  $d \to 2$
  $e \to 3$
- We can clearly see that, vertices a, b, c, e have odd degrees.
- Therefore, Graph G does not have an Euler Circuit.

j)

- Theorem 1 from section 10.5 of the textbook says that:

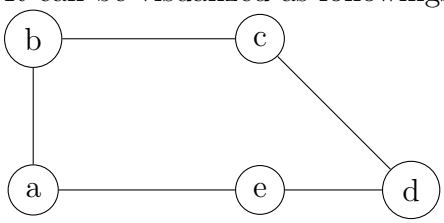
  A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree
- Since the Graph G is a connected multigraph with 5 vertices, we should check the degrees of each vertex.
- Degrees of the vertices:
  $a \to 3$
  $b \to 3$
  $c \to 3$
  $d \to 2$
  $e \to 3$
- We can clearly see that, 4 vertices have odd degrees, whereas 1 vertex has an even degree
- Therefore, Graph G does not have an Euler Path.

k)

- ORE'S THEOREM says that if G is a simple graph with n vertices with $n \geq 3$ such that $deg(u) + deg(v) \geq n$ for every pair of nonadjacent vertices and v in G, then G has a Hamilton circuit.
- G has 5 vertices and the degrees of the vertices are:
  $a \to 3$
  $b \to 3$

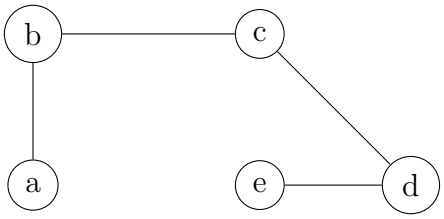
$c \to 3$
$d \to 2$
$e \to 3$

- Let's check Ore's Theorem's conditions for G.
- First, we have a simple graph with 5 vertices, which is ≥ 3.
- Second, we should check the sum of the degrees of nonadjacent vertices, which are $(a, d), (b, d), (c, e)$.
- $deg(a) + deg(d) = 5 \geq n = 5$
  $deg(b) + deg(d) = 5 \geq n = 5$
  $deg(c) + deg(e) = 6 \geq n = 5$
- Graph G meets the conditions of Ore's Theorem. That is why G has a Hamilton circuit.
- Hamiltonian circuit should visit every vertex once. It also begins and ends at the same vertex. Hence our circuit is $a - b - c - d - e - a$.
- It can be visualized as followings:



l)

- Hamiltonian path should visit every vertex once. Unlike the Hamiltonian circuit, it does not begin and end at the same vertex.
- Hence, our Hamiltonian path can be $a - b - c - d - e$.
- It can be visualized as followings:



Answer 2

If two graphs are isomorphic, then they have:

- Equal number of vertices.
- Equal number of edges.
- Same degree sequence.
- Same number of circuits of a particular length.

Let's check if G and H meet these conditions.
1-) There are 5 vertices in both graphs.
2-) There are 5 edges in both graphs.
3-) Each vertex in the graph G has degree 2. This statement is also true for the graph H.
4-) They have exactly the same circuit of a particular length, which is 5.

Moreover, when we define a bijective function $f(x)$ from G to H the mapping is :

$$f(a) = a'$$

$$f(b) = b'$$

$$f(c) = c'$$

$$f(d) = d'$$

$$f(e) = e'$$

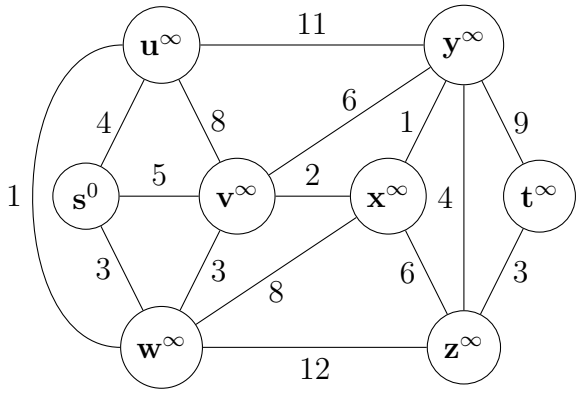
The above correspondence preserves adjacency as a is adjacent to b, and b is adjacent to c in G, whereas a' is adjacent to b', b' is adjacent to c' in H.
Similarly, it can be shown that adjacency is preserved for all vertices.
Therefore the graphs G and H are isomorphic.

Answer 3

In this question, I assigned infinity to the unvisited nodes. In each iteration, while visiting the neighbors, I wrote the weight of every visited vertex as an exponent to the vertex. If I had found a smaller distance to that neighbor I updated the weight of that vertex.

Step 0

s is our starting node. Hence I assigned 0 to it since the distance to itself is 0. This is our initial condition.

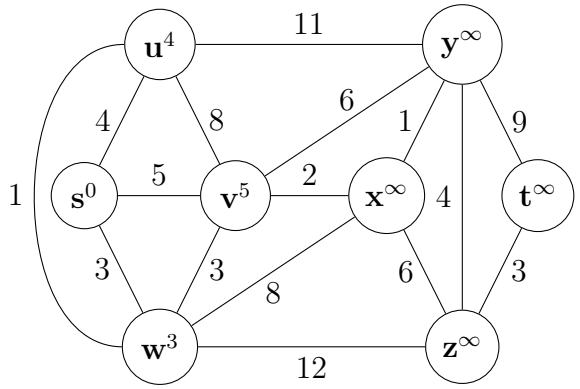


Step 1

The first step is visiting the neighbor nodes of s.

- The distance to u = 4.
- The distance to v = 5.
- The distance to w = 3.

Visited Nodes : s



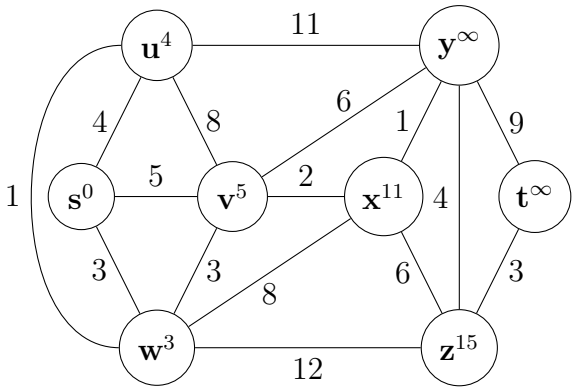
Step 2

The second step is visiting the neighbor nodes of w since it has the minimum weight among the neighbors of s.

- The distance to u = 4. (s)
- The distance to v = 5. (s)
- The distance to w = 3. (s)
- The distance to x = 11. (s, w)

- The distance to z = 15. (s, w)

Visited Nodes : s, w

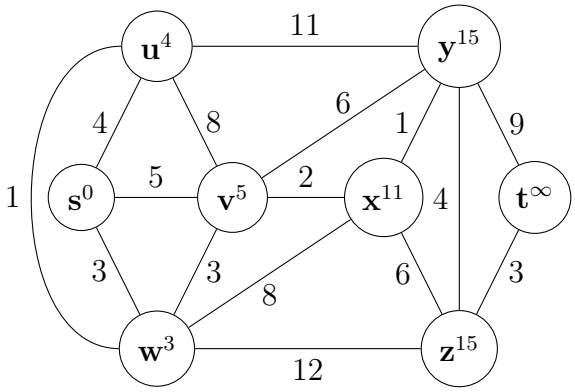


Step 3

The third step is visiting the neighbor nodes of u since it has the minimum weight among u and v, which are the neighbors of s that we did not visit yet.

- The distance to u = 4. (s)
- The distance to v = 5. (s)
- The distance to w = 3. (s)
- The distance to x = 11. (s, w)
- The distance to z = 15. (s, w)
- The distance to y = 15. (s, u)

Visited Nodes : s, w, u

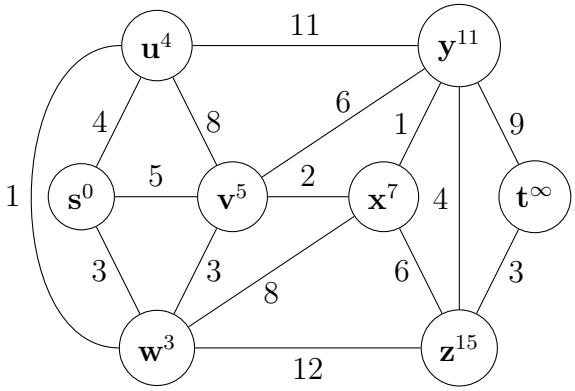


Step 4

The fourth step is visiting the neighbor nodes of v, which is the last neighbor of s that we did not visit yet.

- The distance to u = 4. (s)
- The distance to v = 5. (s)
- The distance to w = 3. (s)
- The distance to x = 7 (s,v) → We updated the weight of x since $7 \leq 11$
- The distance to z = 15 (s,w)
- The distance to y = 11 (s,v) → We updated the weight of y since $11 \leq 15$

Visited Nodes: s, w, u, v

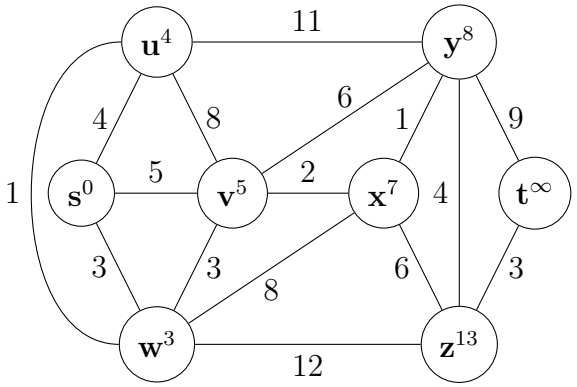


Step 5

The fifth step is visiting the neighbor nodes of x, since it has the minimum weight among the x, y, z

- The distance to u = 4. (s)
- The distance to v = 5. (s)
- The distance to w = 3. (s)
- The distance to x = 7 (s,v) → We updated the weight of x since $7 \leq 11$
- The distance to z = 13 (s,v,x) → We updated the weight of z since $13 \leq 15$
- The distance to y = 8 (s,v,x) → We updated the weight of y since $8 \leq 11$

Visited Nodes: s, w, u, v, x

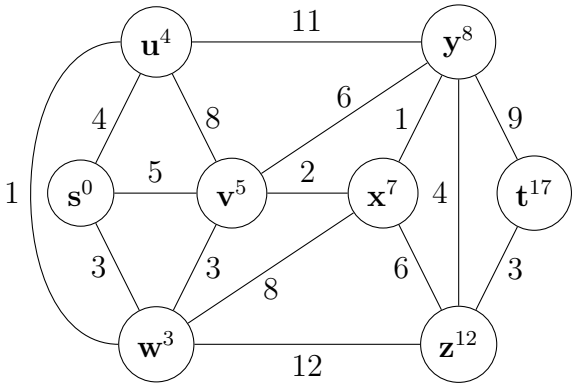


Step 6

The sixth step is visiting the neighbor nodes of y since it has the minimum weight among the y, z, that we did not visit yet.

- The distance to u = 4. (s)
- The distance to v = 5. (s)
- The distance to w = 3. (s)
- The distance to x = 7 (s,v)
- The distance to z = 12 (s,v,x,y) → We updated the weight of z since $12 \leq 13$
- The distance to y = 8 (s,v,x)
- The distance to t = 17 (s,v,x,y)

Visited Nodes : s, w, u, v, x, y

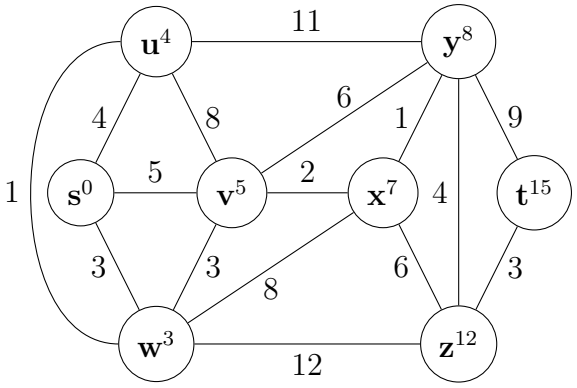


Step 7

The seventh step is visiting the neighbor nodes of z The distance costs and routes from node s are as follows;

- The distance to u = 4. (s)
- The distance to v = 5. (s)
- The distance to w = 3. (s)
- The distance to x = 7 (s,v)
- The distance to z = 12 (s,v,x,y)
- The distance to y = 8 (s,v,x)
- The distance to t = 15 (s,v,x,y,z) We updated the weight of t since $15 \leq 17$

Visited Nodes : s, w, u, v, x, y, z



Step 8

Since there are no unvisited neighbor nodes of t, we reached our conclusion.
By using Dijkstra's Algorithm, we found the shortest path from s to t, which is:

$$s \to v \to x \to y \to z \to t$$

The total weight of the shortest path is 15.

Answer 4

a)

By Kruskal's Algorithm,
1-)A minimum spanning tree is a subset of a graph with the same number of vertices as the graph and edges equal to the number of vertices -1.
2-) A minimum tree should not contain any circles.
Number of vertices = 11
Hence, we should find 10 edges.
Step-1) Considering this, we can create our minimum spanning tree starting from the edge with the least weight, which we can choose as the edge between b and c.

$$(b - c)$$

Step-2) After this point we will continue to choose the edges that have the least weight and not create any circles. For this time, let's say the edge between d and k.

$$(d - k)$$

Step-3)

$$(c - f)$$

Step-4)

$$(h - i)$$

Step-5) Now, we will choose edges that have the weight 3.

$$(a - b)$$

Step-6)

$$(c - d)$$

Step-7)

$$(f - j)$$

Step-8) Now, we will choose edges that have the weight 4.

$$(e - f)$$

Step-9)

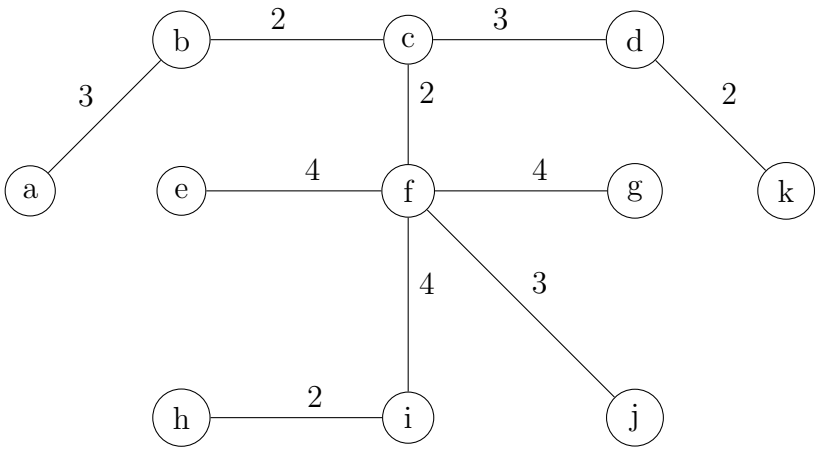
$$(f - g)$$

Step-10)

$$(f - i)$$

Now, we can stop here, since the number of edges = 10 = the number of vertices -1.

b)

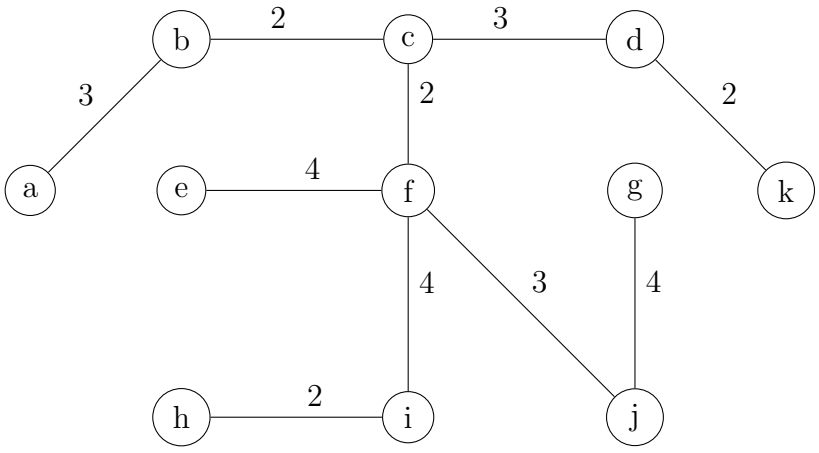


Weight of the spanning tree = $w(e) = 29$.

c)

The uniqueness of the minimum spanning tree depends on the weight of the edge. If all the weights are unique then we can have a unique minimum spanning tree (MST) to a graph. But if we have two different edges to a node with the same weight then we can choose both the edges from that node and therefore we will again check for minimum weighted edges. In the graph G weights of the edges are not unique which is why we can have another minimum tree to find if it is unique, let's review our steps.

- Until step 8, we finished the edges with weights 2 and 3. After this point, we added the edges with weight 4.
- While adding the edges with weight 4, we preferred the edges $(e-f), (f-g)$, and $(f-i)$.
- However we could choose the edge $(g-j)$ instead of the edge $(f-g)$.
- As a result of this, our minimum spanning tree would be as followings:



Weight of the spanning tree $= w(e) = 29$.

Although the weights of the two minimum spanning trees are the same, they are different from each other. Hence, the minimum spanning tree is not unique.