Student Information

Full Name : Kerem Karabacak
Id Number : 2644417

Answer 1

a)

The degree of a node in an undirected graph is the number of edges incident with it. Therefore $deg(a) = 3, deg(b) = 3, deg(c) = 3, deg(d) = 2, deg(e) = 3$. Total is 14.

b)

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$ Number of total non-zero entries is 14.

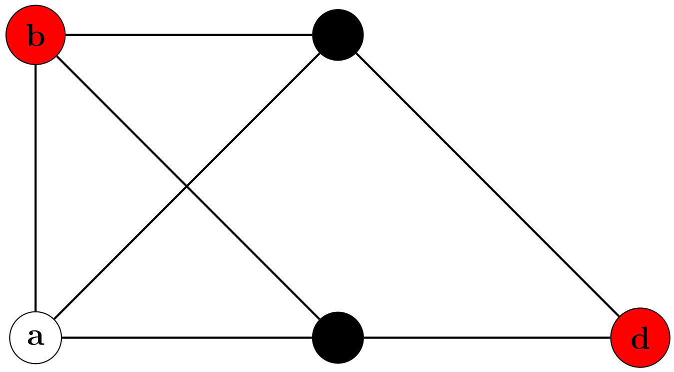
c)

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$ Number of total zero entries is 21.

d)

A complete graph with n nodes has $\frac{n(n-1)}{2}$ edges. Our graph has 5 nodes and 7 edges, so it is not complete graph. Also we can say that there is no such a edge for every paired-nodes. Now consider 4-node graphs. If we take out 1 node from our graph it will be a 4-node graph. When we do it we will lose edges as much as degree of node which we took out. If a 4-node graph is complete then there should be 6 edges(from the formula.). We need to take out a node with degree of 1, but there is no such a node. Therefore there is no complete subgraph in this graph.

e)



We can use coloring method. Two adjacent nodes should be different color, but as you can see there is no way to do it. Therefore this graph is not bipartite.

f)

The undirected graph that results from ignoring directions of edges is called the underlying undirected graph. This direction can be two different direction, so for all edges we can have 2 different case and we have 7 edges. If we apply product rule from counting: $2^7 = 128$ different directed graphs' underlying graph can be our graph.

g)

Simple path means that there is no repeated nodes in path, so simple longest path's length can be at most n, where n is the number of nodes in the graph. There is such a path in our graph : d-c-a-b-e. Length is 5.

h)

A graph is connected if there is a path between every pair of vertices. A node by itself is a connected component(5). We have a connected component for each edge(7). Except a-b-d all 3-node subgraphs are connected components(9). All 4-node subgraphs are connected components(5). Finally, our graph is connected as well(1). Therefore 27 connected components are available.

i)

According to theorem if a graph has Euler circuit, then all of its nodes' degrees are even, but this is not the case for our graph. Therefore there is no Euler circuit for this graph.

j)

According to theorem if a graph has Euler path, then it has 0 or 2 odd degree node(s), but our graph has 4 odd-degree nodes. Therefore there is no Euler path for this graph.

k)

We need to visit every vertex and return back to our starting position. There is such a path : b - c - d - e - a - b

l)

We need to visit every vertex again, but no need to return back our statring position. There is such a path as well : b - c - a - e - d

Answer 2

First, we can look graph invariant. G and H both have 5 nodes, 5 edges, and all nodes' degrees are 2. So these graphs might be isomorphic. Now, we should define a function from V_1 to V_2 which is one-to-one and onto(f). If this is the case then this graphs are isomorphic. We can use adjacency matrices.
$V_1 = a, b, c, d, e$
$V_2 = a^{'}, b^{'}, c^{'}, d^{'}, e^{'}$

$A_G =$

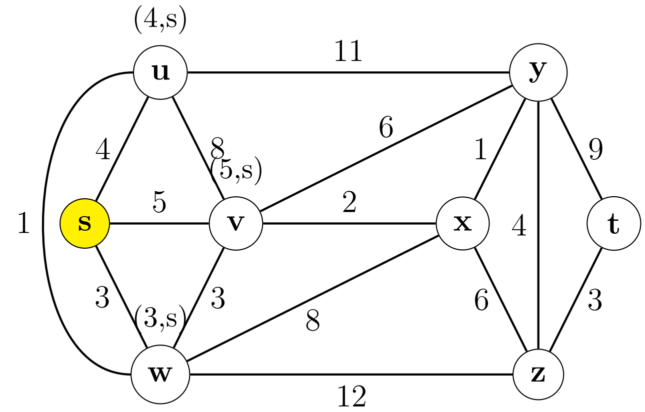
$$\begin{array}{c c} & \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$A_H =$

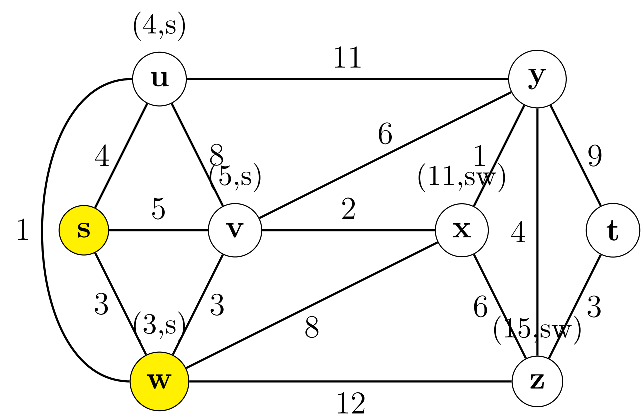
$$\begin{array}{c c} & \begin{array}{ccccc} a^{'} & b^{'} & c^{'} & d^{'} & e^{'} \end{array} \\ \begin{array}{c} a^{'} \\ b^{'} \\ c^{'} \\ d^{'} \\ e^{'} \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

We manage to find same-formed adjacency matrices. That means we can map vertexes:
$f(a) = a^{'}, f(b) = b^{'}, f(c) = c^{'}, f(d) = d^{'}, f(e) = e^{'}$.
Since we have a bijective function we can say that these graphs are isomorphic.

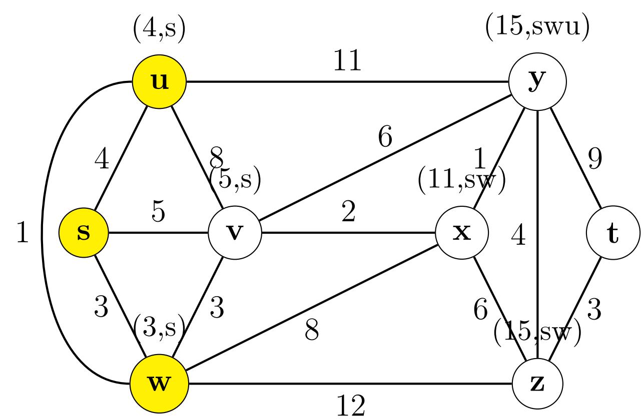
Answer 3



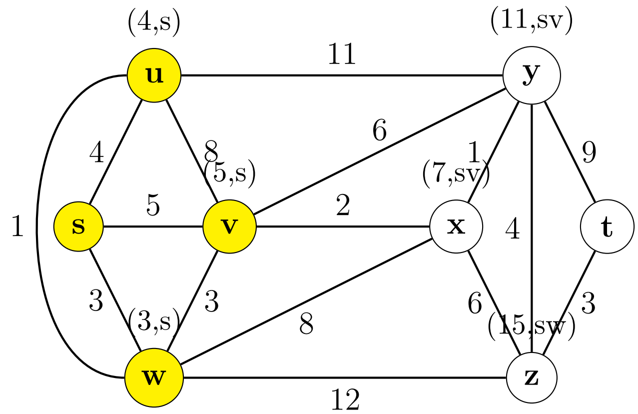
At first we are checking s node's unvisited neighbours(u,v,w). For the next step we need to choose the smallest distance one which is w.



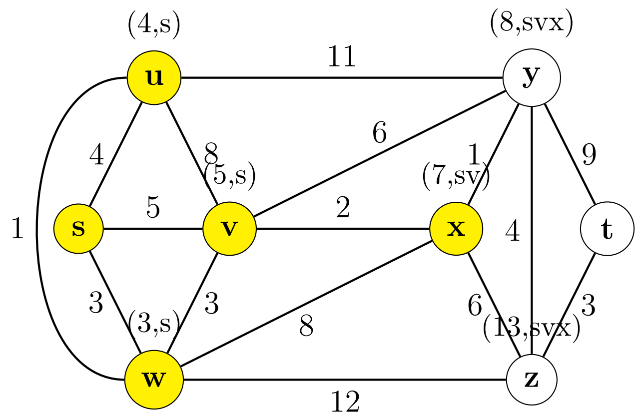
Now we are checking w node's unvisited neighbours(u,v,x,z). u node's distance is still 4, so no need to updatee. v node's distance is 3+3 = 6, but it's bigger than its current value, so no need to update. x node's current distance is ∞, so we should update this one. 3+ 8 = 11. Again z's current distance is ∞, so we must update it to 3+12 = 15. For the next step we are choosing smallest distance one which is u.



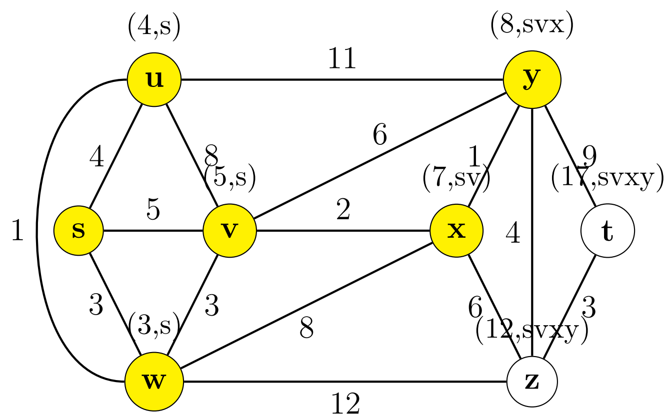
Now we should consider the u's unvisited neighbours(v,y).y's currenct distance is ∞. We need to update this to 4+11=15. If we reach v with this path our distance is 4+8 = 12,but v's current distance is smaller, so no need to update v. For the next step we should choose v which has smaller distance.



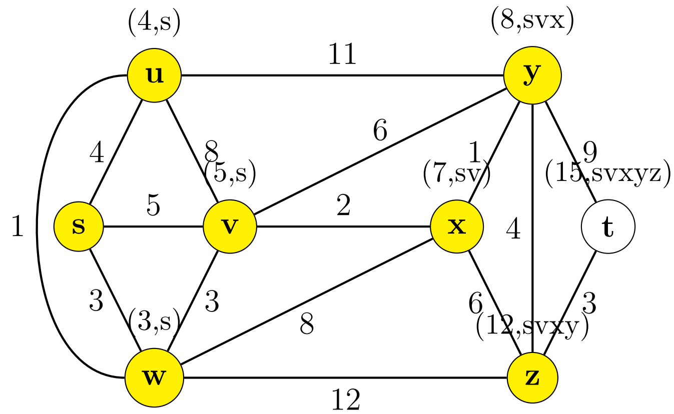
For this step we should consider y and x. x's distance need to be updated. 5+2 = 7<11. Also y's distance need to be updated. 5+6=11<15. For the next step we are moving to x because its distance is smaller than y's.



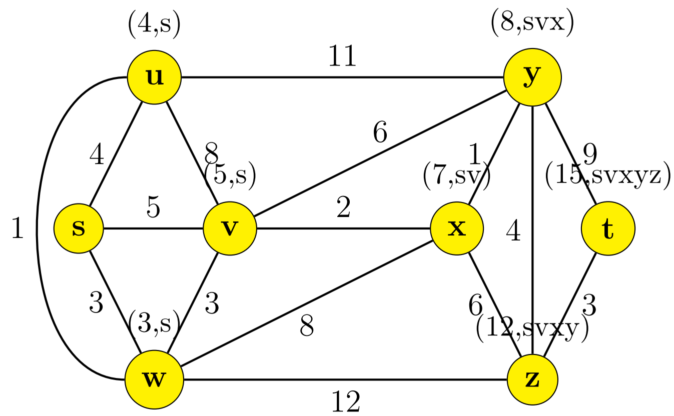
We should consider y and z. y's distance need to be updated. 7+1=8<11. Also z's distance need to be updated. 7+6=13<15. We are moving to y for next step because its distance is smaller than z's.



Neighbours are t and z. z's distance need to be updated. 8+4=12<13. t's current value is ∞. 8+9=17< ∞. For the next step our node is z.



Only unvisited negihbour is t. Distance value need to be updated. 12+3=15<17.



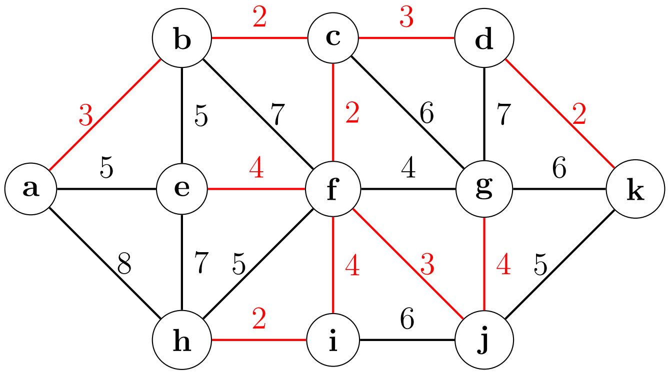
Therefore our last graph is looking like this. The shortest path from s to t is s-v-x-y-z-t and our cost is 15.

Answer 4

I am going to use Prim's algorithm to solve this problem.

Pick Order:
1)a-b 2)b-c 3)c-f 4)f-j 5) c-d 6)d-k 7)j-g 8)f-e 9) f-i 10)i-h

At first I chose an arbiray node a. The cheapest edge is a-b and the second one is b-c. c-f is the third one. While I am picking I am considering previouse nodes' costs. Until now newest one has the cheapest. For the fourth one we can choose either f-j or c-d. I chose f-j. Now it's c-d's turn and d-k. Now we should choose from previous ones, which j-g, and another previous one f-e. Cheapest one is f-i and finally i-h and we touched all of the nodes. Therefore minimum spannig tree is done.



Minimum spanning tree is not unique. We can give a counter example. After j-k pick, we had two different options. f-g or g-j. Both are valid and the cheapest picks. If we change g-j pick with f-g we can have a different minimum spanning tree. Therefore we have more than one minimum spanning tree.