Name: Georges Hatem

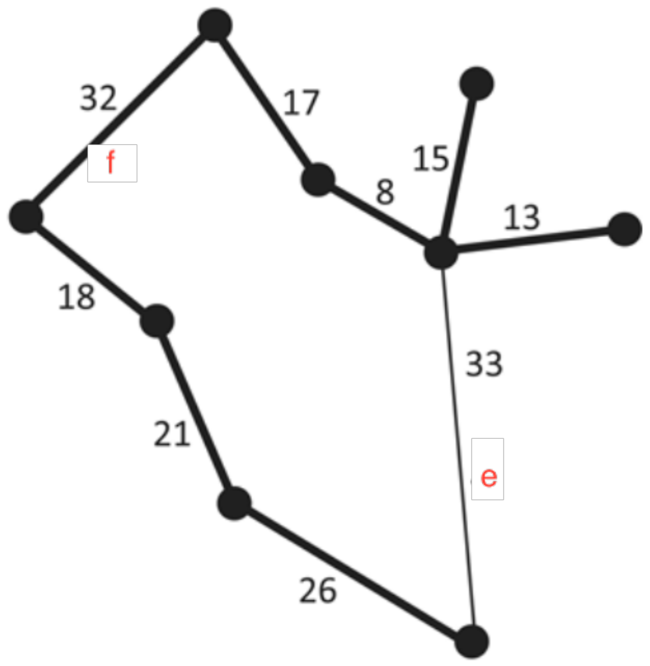
CS590 Homework Assignment 13: Minimum Spanning Trees Creativity Exercises

Due Date: April 24, 2022

Problem 15.6.12:

Before starting the proof, let's illustrate the following:

Suppose we have the following Graph (G) below (remember that per the exercise Graph (G) is a weighted, connected undirected simple Graph – connected means that the Graph is fully enclosed) {This picture below is taken from our Zybook just for illiustration purposes and since it is much better than drawing it here in Word (I just changed the e and f locations to make e the highest weight edge in G as considered in the exercise}:



(b)

This Graph above is weighted, connected, and undirected simple graph. As you can see, since the graph is connected, we are always going to be missing one edge of the graph when forming the minimum spanning tree (T), and this edge is the edge with the highest weight on it (in this case f). This is the reason why:

Since this is a connected graph, the number of edges cannot be lower than the number of vertices. Going by Kruska's Algorithm, we perform the following steps to get the minimum spanning tree (T):

1) Each vertex in Graph (G) has its own clusters. When 2 vertices combine, they become part of the same cluster. So, multiple vertices that are combined together share the same cluster.
2) The Algorithm starts by adding the lowest weight edges to the minimum spanning tree (T) and increasing if the 2 vertices of the edge are not in the same cluster.
3) If the 2 vertices of the edge we are trying to add into T are in the same cluster, then this edge will not be added to the spanning Tree (T).

Since we have a connected graph (G), the Graph (G) is closed just like the graph above, and so the number of edges cannot be lower than the number of vertices. Suppose we have the number of edges and vertices equal like the example in the Graph in the picture that we have above, by Kruska's Algorithm, the 2 vertices for the edge f are already in the same cluster when we get to edge f (since edge f is the edge with the highest weight), then edge f will not be added to the minimum spanning tree (T). What I mean by this is since we are given a connected graph (G), the number of edges cannot be lower than the number of vertices. If the number of edges and vertices are equal, then one of the edges is not going to be added to the minimum spanning tree (T) since the last edge in Graph (G) to be added into the minimum spanning tree (T) will have its 2 vertices already in the same cluster, and this edge that will not be added into the minimum spanning tree (T) is the edge with the highest weight in Graph (G). If we have more edges than vertices, than the number of edges that will not be added into the minimum spanning tree (T) is the edge with the highest weight in

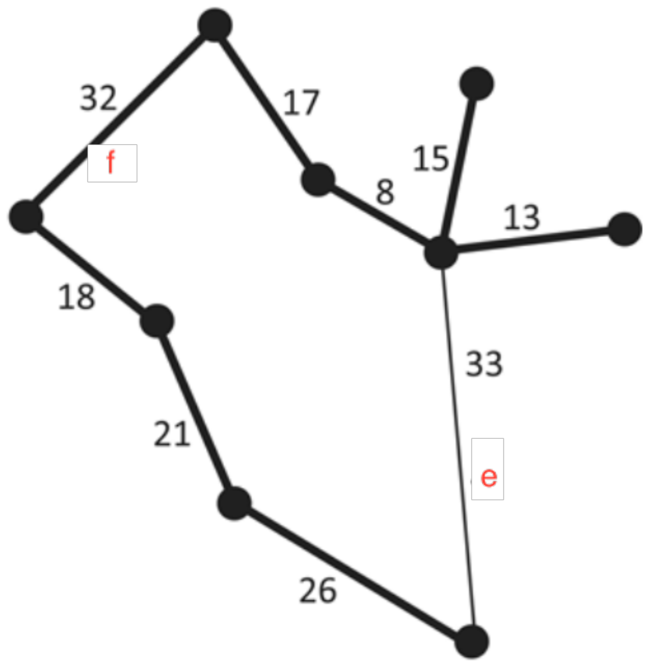
Graph (G) plus the number of edges that are higher than the vertices. This means that if the number of edges is higher than the number of vertices by 1, we can say that the number of edges that will not be added to the minimum spanning tree (T) is 1 (highest weight in Graph (G)) + 1 (number of edges higher than the number of vertices) = 2. So, in the case described in the previous sentence, 2 edges in Graph (G) will not be added to the minimum spanning tree (T).

From the analysis described above, since Graph (G) is a weighted, connected, undirected simple graph, the number of edges in the Graph (G) cannot be lower than the number of vertices in the Graph (G), and since the edge with the highest weight gets visited last, it will not be added to the minimum spanning tree (T). Therefore, considering we have a weighted, connected, undirected simple Graph (G), the edge with the highest weight in Graph (G) will not be added to the minimum spanning tree (T).

This proves the claim that there is no minimum spanning tree (T) of G that contains a largest weight edge of G (e).

Let's prove that claim in a different way:

Using same Graph (G) above as example:



(b)

Suppose that the minimum spanning tree of the weighted, connected graph (G) is T. So, T is a spanning tree that is connected acyclic subgraph of G that contains all the vertices of G. Now, let's suppose that e, which is the largest -weight edge in G, is in the minimum spanning tree (T).

Suppose, for the sake of contradiction, that there is an edge, f, whose weight is less than e's weight, and let’s suppose that edge, f, is not part of the minimum spanning tree (T) but it exists in Graph (G). Let’s remove e from the minimum spanning tree (T) and replace f by e. This will result in a spanning tree (T’), whose total weight is less than the total weight of T. But, the existence of such a minimum spanning tree (T’) contradict the fact that T is a minimum spanning tree. This means that we need to move weight numbers from Graph (G) into the minimum spanning tree in increasing weight order (from lower to higher). And so, this means that edge f (which has a lower weight than edge e) will be added into the minimum spanning tree (T) before adding edge e.

Now, the exercise is saying that the edge e is the highest weight edge in the Graph (G). Graph (G) is a weighted, connected, undirected simple graph. Since Graph (G) is a connected graph, this means that number of edges cannot be lower than the number of vertices. Since the number of edges cannot be lower than the number of vertices, this means that one or more edges cannot be added to the minimum spanning tree. If we have the number of edges same as the number of vertices, then 1 edge from Graph (G) will not be added to the minimum spanning tree (T). If the number of edges is higher than the number of vertices, then the number of edges from Graph (G) that will not be added into the minimum spanning tree (T) is 1 + (number of edges – number of vertices). Since the edges are added into T in weight increasing order, this means that the highest weight edge (in this case e) will always not be added to the minimum spanning tree (T).

From the above, considering Graph (G) a weighted, connected, undirected, simple graph and e is the largest-weight in G, we proved from above that the claim that there is

no minimum spanning tree of G that contains e is correct.