Assignment 14

R-7.1 Draw a simple, connected, undirected, weighted graph with 8 vertices and 16 edges, each with unique edge weights. Identify one vertex as a "start" vertex and illustrate a running of Dijkstra's shortest path algorithm on this graph.

R-7-7 Draw a simple, connected, undirected, weighted graph with 8 vertices and 16 edges, each with unique edge weights. Illustrate the execution of Kruskal's algorithm on this graph. (Note there is only one minimum spanning tree for this graph.)

R-7.8 Repeat the previous problem for the Prim-Jarvik algorithm.

R-7-9 Repeat the previous problem for Baruvka's algorithm.

Consider the following potential MST algorithms based on the generic MST algorithm. Which, if any, successfully computes a MST? **Hint**: to show that an algorithm does not compute an MST, all you need to do is find a counterexample. If it does, you need to argue why based on the cycle property and/or the partition property.

a.

```
Algorithm MST-a(G, w)
    T ← edges in E sorted in nonincreasing order of edge weights
    for each e ∈ T do {each e is taken in nonincreasing order by weight }
        if T – {e} is a connected graph then
            T ← T – {e}     {remove e from T}

    return T
```

b.

```
Algorithm MST-b(G, w)
    T ← { }
    for each e ∈ E do { e is taken in arbitrary order }
        if T ∪ {e} has no cycles then
            T ← T ∪ {e}     {add e to T}

    return T
```

c.

```
Algorithm MST-c(G, w)
    T ← { }
    for each e ∈ E do { e is taken in arbitrary order }
        T ← T ∪ {e}     {add e to T}
        if T now has a cycle C then
            if e' is the edge of C with the maximum weight then
                T ← T – {e'}     {remove e' to T}

    return T
```

C-3-28 Describe how to implement the locator-based method *before*(l) as well as the locator-based method *closestBefore*(k) in a dictionary realized using a skip-list. What are the expected running times of the two locator-based methods in your implementation?

C-5.1 A native Australian named Anatjari wishes to cross a desert carrying only a single water bottle. He has a map that marks all the watering holes along the way. Assuming he can walk *k* miles on one bottle of water, design an efficient algorithm for determining where Anatjari should refill his bottle in order to make as few stops possible. Argue why your algorithm is correct.
Do this with two different assumptions:

1. Assume the watering holes are all located along a road
2. Assume the watering holes are spread over the whole desert, i.e., there is no road.