Name: Georges Hatem

CS590 Homework Assignment 13: Application Exercises

Due Date: April 24, 2022

Problem 15.6.23:

This can be accomplished by modifying Dijkstra's algorithm. Instead of representing the shortest path from a to u, the label D[u] represents the maximum bandwidth of any path from a to u. The maximum bandwidth for path from a through u to a vertex z adjacent to u is min{D[u], w((u, z))} so that the relaxation step updates D[z] to max{D[z], min{D[u], w((u, z))}}.

The algorithm is as follows:

MaxBandwidth (G, a, b)

Input: Weighted graph G and two distinguished vertices a and b
Output: Maximum bandwidth over all paths between a and b

Initialize the labels D[a] to infinity and D[u] to 0 for each vertex u! = a in G

```
Let there be a priority queue Q that contains
all the vertices of G using the D labels as keys

while (Q != NULL)

    u ← Q.removeMaxElement()

        if (u == y)

            return D[u]

        else

            for each vertex z, adjacent to u such
            that z E Q do

                bandwidth ← min {D[u], w(u,z)}

                if (bandwidth > D[z])

                    D[z] <-bandwidth

                    change the key of z in Q to
                    D[z]
```

Explanation:

1) No value is returned in the algorithm because the algorithm will find the vertex.
2) The label D[a] is initialized to infinity because the minimum value is found for D[a].
3) The label D[u] is initialized to zero as it is required to find the maximum value.

The running time of the algorithm is the same as Dijkstra's Algorithm. In this scenario, the adjacency list is being used. So, the running time complexity is O((n+m)log(n)) if the priority queue is being implemented as a heap, and O(n^2) if the priority queue is being implemented as an unsorted sequence. (Or even O(n log n + m) using a fancier data structure to implement the priority queue.)