ECS175: Algorithm Analysis and Design
Practice Midterm 2

1. A graph can include many negative-weight cycles. In the arbitrage problem from Homework 5, you gave an algorithm to find one negative-weight cycle. It would would be more profitable to find all of the negative-weight cycles, but unfortunately on a worst-case graph this might require time $\Omega(2^n)$, where $n = |V|$. Describe how to construct such a worst-case graph, for any given n.

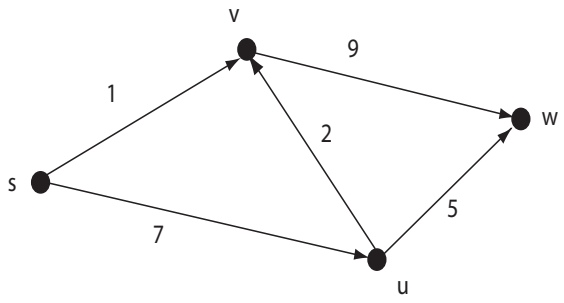
2. We implemented Dijkstra's algorithm using a heap for the priority queue. One can also implement a priority queue simply as a linked list, with the elements sorted in order of priority. To perform an INSERT (v, Q)or DECREASE-KEY(v, Q), we have to locate v in the list.

   a) What are the asymptotic times required for the operations EXTRACT-MIN(Q), INSERT(v, Q), and DECREASE-KEY(v, Q) when the priority queue is implemented as a linked list? How about when the priority queue is implemented as a heap?

   b) What is the overall running time of Dijkstra's algorithm with the queue implemented as a linked list?

3. We are given a graph G with weights on the edges, and we define the *capacity* of a path to be the *minimum* of the weights on the edges of the path (notice that this is *different* from the length of a path as defined in Chapter 24).

   Given a particular vertex s, we want to find the capacity of a maximum-capacity path from s to every other vertex v. Let us call this value $c(v)$. For instance, on the graph below we would find $c(v) = 2, c(w) = 5$.



   a) Prove that if $s, w, \ldots, u, v$ is a maximum-capacity path from s to v, that the maximum-capacity path from s to u, followed by the edge (u, v), is a maximum-capacity path to v. Is it true that every sub-path of a maximum-capacity path is a maximum-capacity path?

   b) One way to solve this problem is by using dynamic programming. We consider paths of length one, then paths of length two, etc., up until paths of length $n - 1$. Let $c(v, i)$ denote the capacity of the maximum-capacity path of length i from s to v. We can use the recursive formulation of the problem:

$$c(v, i) = \max\{c(v, i-1), \max_{u \in V, u \neq v}\{\min\{w(u, v), c(u, i-1)\}\}$$

   That is, we try every vertex u other than v, and we check the capacity of the path formed by taking the maximum-capacity path from s to u of length $i - 1$ and then taking edge (u, v). We choose whichever of these has the greatest capacity, or we stick with $c(v, i - 1)$, whichever is greatest.

   Describe how to use this recursive formulation to give a dynamic programming algorithm, and analyze the running time of your algorithm.