Question 1. (20 points) Suppose that you face a version of the network flow problem in which vertices as well as edges have capacities, i.e., if a vertex v has capacity $c(v)$ then the flow into it cannot exceed $c(v)$. Edge capacities have the usual meaning (as explained in class). Explain how an algorithm for solving the version of network flow presented in class (in which only edges have capacities) can be used to solve this new version of the problem.

Question 2. (40 points) Given a string $x = a_1 \cdots a_n$, where $n = 2^q$ for some integer q, we seek an $O(n)$ time algorithm for determining whether x consists of the concatenation of copies of a string α (if there are many such possible answers α, then the algorithm should find the shortest one). In other words, the problem is to compute the shortest α such that $x = \alpha\alpha \cdots \alpha$. For example, if $x = abababab$ then the answer is $\alpha = ab$ rather than $abab$. Note that it is possible that $\alpha = x$.

1. (20 points) Give an $O(n)$ time algorithm that makes use of the linear time algorithm for pattern matching.
2. (20 points) Give an $O(n)$ time recursive algorithm that solves the problem directly, without using pattern matching as a subroutine.

Question 3. (40 points) Let G be an n-vertex, m-edge connected undirected graph that is also biconnected, where $n > 20$. Give an $O(n + m)$ time algorithm that removes edges from G so it ends up with fewer than cn edges for some constant c, while preserving the property that G is biconnected. Note that it is not required to remove as many edges as possible from G, only that G ends up with fewer than cn edges. Make sure you state clearly what c is, and prove that G ends up with less than cn edges and remains biconnected.

Date due: November 27, 2012