CS381 **Homework 5** **Fall 2013**

Question 1. (20 points) Suppose that a connected undirected graph G does not contain a bridge, but that removing any edge from it results in a graph which contains one or more bridge(s). Assume $n \geq 3$. Prove that G has no more than $2n-3$ edges. (Recall that, by definition, in an undirected graph there is at most 1 edge between two vertices.)

Hint. Use the concept of a depth-first search tree.

Question 2. (20 points) Suppose you have software that implements an all-pairs shortest paths algorithm: It takes as input a directed graph G with positive costs associated with its edges, and its output is a matrix D where $D[v, w]$ is the cost of a shortest path from v to w. Explain how you would use that software for solving the version of the all-pairs shortest-paths problem where the positive costs are associated with the *vertices* rather than with the edges, hence the cost of a directed path is the sum of the costs of the vertices on that path. You are not allowed to change the internals of the software, the only control you have is over what input you give it, and how you obtain your answer from its output.

Question 3. (20 points) Let $G = (V, E)$ be an undirected graph that is connected and has positive costs associated with its edges. Let T denote a minimum-cost spanning tree of G (as computed by, for example, Kruskal's algorithm). Suppose that the cost of a path in G is defined to be the cost of the most expensive edge on that path (i.e., it is the max rather than the sum of the edge costs on the path).

- Prove that, for any pair of vertices x and y, the path from x to y that uses only edges of T is at least as cheap as any other path in G from x to y. In other words following edges of T produces a shortest path (hence there is no need to use edges outside of T).

Question 4. (20 points) Let $X = a_1 \dots a_n$ and $Y = b_1 \dots b_n$ be strings of length n each. Give a linear time algorithm for determining whether Y is a circularly rotated version of X.

Hint. This can be solved by a judicious use of pattern matching.

Question 5. (20 points) Give an $O(n)$ time algorithm that, given a string $X = a_0 \dots a_{n-1}$, determines whether there is a circular rotation of X that turns it into a palindrome (a palindrome is a string that is equal to its reverse, such as the string *amanaplanacanalpanama*). For example, if $X = amaamanaplanacanalpan$, then the answer is "yes" because a left circular rotation of X by three positions turns it into the palindrome *amanaplanacanalpanama*.

Hint. This can be solved by a judicious use of pattern matching.

Date due: Thursday October 17, 2013