Qualifying Examination
Theoretical Computer Science
Monday, March 5, 2018
Part I: Algorithms

Instructions:

1. This is a closed book exam.
2. The exam has four problems worth 25 points each. Read all the problems carefully to see the order in which you want to tackle them. You have all day (9am–5pm) to solve the problems.
3. Write clearly and concisely. You may appeal to some standard algorithms/facts from text books unless the problem explicitly asks for a proof of that fact or the details of that algorithm.
4. If you cannot solve a problem, to get partial credit write down your main idea/approach in a clear and concise way. For example you can obtain a solution assuming a clearly stated lemma that you believe should be true but cannot prove during the exam. However, please do not write down a laundry list of half baked ideas just to get partial credit.

May the force be with you.

Problem 1: We are given a set S of n real numbers, where each element of S is assigned a color. We want to store S in a data structure to answer the following type of queries: given a query interval $[q_a, q_b]$, find the color that occurs the most frequently among the elements in $S \cap [q_a, q_b]$.

Describe a near-$O(n)$-space data structure that supports such queries in close to $O(\sqrt{n})$ time, ignoring logarithmic factors. *Hint*: divide into $\sqrt{n}$ groups of $\sqrt{n}$ elements.

Problem 2: Let $G = (V, E)$ be directed graph, where each edge $e \in E$ has non-negative integer edge length $\ell(e)$. The *minimum mean cycle* is a cycle C that minimizes the ratio $\ell(C)/|C|$, where $\ell(C) = \sum_{e \in E(C)} \ell(e)$ is the total length of the edges in C, and $|C|$ is the number of edges in C. Describe a polynomial-time algorithm to compute the minimum mean cycle.

Problem 3: Let $G = (V, E)$ be an undirected graph with non-negative edge costs $c : E \to \mathbb{R}_+$. Given a subset $S \subseteq V$ of nodes called terminals, a tree $T = (V_T, E_T)$ that is a subgraph of G is a *Steiner tree* for S if $S \subseteq V_T$. A node in $V_T \setminus S$ is called a *Steiner node*. Finding a minimum cost Steiner tree is an NP-Hard problem.

(a) Let $|S| = k$. Design an algorithm for the Steiner tree problem that runs in time $n^{O(k)}$ where $n = |V|$. *Hint:* How many Steiner nodes can there be with degree at least 3?

(b) Design a fixed-parameter tractable (FPT) algorithm for the Steiner tree problem, i.e., an algorithm that runs in time $f(k)n^c$ where c is a fixed constant independent of k and $f(k)$ is some function of k. There is an algorithm that runs in $\alpha^k n^c$ for fixed constants α, c. *Hint:* use dynamic programming.

Problem 4: You are given an undirected graph G with n vertices and m edges, and positive weights on the edges. This graph represents a network of computers. Your task is to deploy DNS servers on all of these computers. At each round, you can start a single server on one of the computers on the network.

When such a server starts, it broadcasts to the network that it is online, and all the clients for which it is a better server get reassigned to it. Here, for a client c, and servers s_1, s_2, the server s_1 is *better* than s_2 if $d_G(c, s_1) < d_G(c, s_2)$, where $d_G(\cdot, \cdot)$ is the shortest path metric of G. Specifically, the clients reassigned to such a new server are all the clients for which the new server is the closest among all the servers running. Unfortunately, reassigning a client to a new server is expensive.

In particular, given a permutation of the vertices of the graph, the *cost* of the permutation is the total number of clients reassignments that happened when deploying the servers in the order specified by the permutation.

(a) Describe a randomized algorithm that computes a permutation of cost $O(n \log n)$ (in expectation or with high probability) – prove the upper bound on the number of reassignments.

(b) Prove that the cost of $\Omega(n \log n)$ is a lower bound for certain graphs for all possible permutations.

(c) Given a permutation computed by your algorithm, describe how to compute all the reassignments for all the clients. How fast is your algorithm? [Faster is better.]