Qualifying Examination

Theoretical Computer Science

Thursday, March 10, 2016

Part I: Algorithms

Name	

Problem	Maximum Points	Points Earned	Grader
1	25		
2	25		
3	25		
4	25		
Total	100		

Instructions:

1. This is a closed book exam.
2. The exam is from 9:30am–4:30pm and has four problems of 25 points each. Read all the problems carefully to see the order in which you want to tackle them.
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: Consider a minimization problem, where for an instance I, its optimal value is non-negative and is denoted by $\mathrm{opt}(I)$. You are given a polynomial time approximation algorithm `alg`, for this optimization problem, that can compute a cn approximation to the given instance I, where $n = |I|$ denotes the size of the given input instance, and c is a fixed constant.

A polynomial time algorithm `reducer` is a (f, g)-*reduction*, if given any instance K, for which `alg` provides an α approximation, the algorithm `reducer` generates from K a new instance K' of size $f(|K|)$, such that `alg`, when run on K', provides a $g(\alpha)$ approximation to $\mathrm{opt}(K')$. Furthermore, luckily, we have that $\mathrm{opt}(K') = \mathrm{opt}(K)$.

1. Consider a (f_1, g_1)-reduction, for $f_1(x) = 2x$, and $g_1(x) = 1 + x/2$. Show how to compute a constant approximation to the optimal value of the given instance I. What is the running time of your algorithm?

2. Consider a (f_2, g_2)-reduction, for $f_2(x) = x^2$, and $g_2(x) = 1 + \ln x$. Show how to compute a constant approximation to the optimal value of the given instance I. What is the running time of your algorithm?

3. Consider (f_3, g_3)-reduction, for $f_3(x) = x^2$, and $g_3(x) = 1 + \sqrt{x}$. Show how to compute a constant approximation to the optimal value of the given instance I. What is the running time of your algorithm?

Problem 2: Let $T = (V, E)$ be a free tree (a connected undirected graph with no cycles) where every node v of T has a label $\ell(v)$ from some finite alphabet. Describe an efficient algorithm to find the longest path in T such that the labels on the path form a palindrome when concatenated along the path. A polynomial-time algorithm will get you siginifcant portion of the credit so focus on that first. Full credit for a solution that is reasonably fast.

Problem 3: Let G be a simple regular undirected graph; by simple we mean that there are no parallel edges any by regular we meant that all node degrees are the same. A cycle cover is a collection of vertex disjoint cycles that span the vertices of the graph. It is easy to see that a 2-regular graph has a cycle cover, in fact G itself is a cycle cover.

- Suppose G is 2^k-regular for some $k > 1$. Prove that there is a 2^{k-1}-regular subgraph of G.

- Using the above prove that every 2^k-regular graph has a cycle cover.
- *Hard:* Prove that every regular even-degree graph has a cycle cover.

Problem 4: Suppose we have a set system with ground set $\mathcal{U}$ and a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. A set cover is subset $\mathcal{S}' \subset \mathcal{S}$ such that $\cup_{A \in \mathcal{S}'} A = \mathcal{U}$. Consider the problem of finding the largest number of *disjoint* set covers in a given set system. That is, we want to find $\mathcal{S}'_1, \ldots, \mathcal{S}'_h$ such that for $1 \leq j \leq h$, $\mathcal{S}'_j$ is a set cover, and for $1 \leq i < j \leq h$, $\mathcal{S}'_i \cap \mathcal{S}'_j = \emptyset$. For each $e \in \mathcal{U}$ let k_e be the number of sets from $\mathcal{S}$ that contain e. Clearly $k = \min_{e \in \mathcal{U}} k_e$ is an upper bound on the number of disjoint set covers. Describe a randomized algorithm that outputs $\Omega(k/\log n)$ disjoint set covers in expectation where $n = |\mathcal{U}|$. Note that the bound is interesting only if $k = \Omega(\log n)$. As a corollary prove that there always exist $\Omega(k/\log n)$ disjoint set covers.