CS369N: Problem Set #1

Due in class on Thursday, October 20, 2011

Instructions:

(1) Students taking the course for a letter grade should attempt all of the following 5 problems; those taking the course pass-fail should attempt at least 3 of them.

(2) Some of these problems are difficult. I highly encourage you to start on them early and discuss them extensively with your fellow students. If you don't solve a problem to completion, write up what you've got: partial proofs, lemmas, high-level ideas, counterexamples, and so on. This is not an IQ test; we're just looking for evidence that you've thought long and hard about the material.

(3) You may refer to your course notes, and to the textbooks and research papers listed on the course Web page *only*. You cannot refer to textbooks, handouts, or research papers that are not listed on the course home page. Cite any sources that you use, and make sure that all your words are your own.

(4) Collaboration on this homework is *strongly encouraged*. However, your write-up must be your own, and you must list the names of your collaborators on the front page.

(5) No late assignments will be accepted.

Problem 1

Recall the original notion of instance-optimality discussed in Lecture #1 and the Fagin/Lotem/Naor paper.

(a) (3 points) Prove that there is no instance-optimal algorithm for searching a sorted array for a given element (in the comparison model).

(b) (3 points) Prove that there is no instance-optimal algorithm for sorting a given array (in the comparison model).

(c) (4 points) Generalize your argument from (b) as much as possible. In other words, define properties of a computational problem that are sufficient to infer that there is no instance-optimal algorithm for it.

(d) (Extra credit) Recall that Afshani/Barbay/Chan faced a similar issue to (a)–(c) but worked around it by evaluating a competing algorithm on multiple inputs (in their case, all $n!$ permutations of a given point set) and aggregating the results by either averaging or taking the worst case (over permutations). Say whatever you can (examples, proposed definitions, lemmas, conjectures) about whether or not some analogous type of workaround seems possible for sorting and searching problems.

Problem 2

The point of this problem is to apply the "order-oblivious" version of instance optimality from the Afshani/Barbay/Chan paper to the online bin packing problem. This is an example of circumventing competitive analysis by changing the analysis framework rather than by introducing a model of the data.

In the online bin packing problem, items with sizes between 0 and 1 arrive online. At any point you can open up a new bin, which has capacity 1. Every item that arrives must be placed in a bin, subject to the capacity constraints. Previous items cannot be moved or removed. The objective is to minimize the number of bins opened by the end of the item sequence. A *greedy online algorithm* is one that only opens a new bin as a last resort (when the current item doesn't fit in any existing bins).

(a) (4 points) The *first fit (FF) algorithm* is the greedy online algorithm that packs the current item into the earliest bin in which it fits, or opens a new bin for the item if there is currently no room for it. Show that this algorithm is order-oblivious instance-optimal among greedy online algorithms, in the sense that for every (unordered) set S of items and every greedy online algorithm B:

$$\max_{\pi}\{\text{cost}(FF, \pi(S))\} \leq \max_{\pi}\{\text{cost}(B, \pi(S))\},$$

where $\text{cost}(A, z)$ denotes the number of bins used by algorithm A on input z, $\pi(S)$ denotes an ordering of the items in S, and each of the max operations ranges over all such orderings.

(b) (4 points) Prove that not every greedy online algorithm is order-oblivious instance-optimal in the sense of (a).

(c) (7 points) Identify the minimum value of $\alpha \geq 1$ for which the following statement is true: for every two greedy online algorithms A and B and every set S of items,

$$\max_{\pi}\{\text{cost}(A, \pi(S))\} \leq \alpha \cdot \max_{\pi}\{\text{cost}(B, \pi(S))\}.$$

Problem 3

This problem introduces an algorithm comparison tool that is a weak form of instance optimality, and applies it to online paging algorithms. Fix a set of N pages and a cache size k. Let I_n denote the page request sequences of length n. For two online algorithms A and B, write $A \leq_n B$ if there is a perfect matching M of I_n with itself (i.e., a bijection) such that $\text{cost}(A, z) \leq \text{cost}(B, M(z))$ for every $z \in I_n$. In other words, for every input z to A we can find an equally bad input $M(z)$ of the same length for B.

(a) (4 points) Suppose that $A \leq_n B$ for every n. Does this have any implications for an average-case analysis of the performance of A and B?

(b) (6 points) Recall the algorithms Flush-When-Full (FWF) and Least Recently Used (LRU) from Lecture #3. Prove that it is *not* the case that $FWF \leq_n LRU$ for every n.

(c) (10 points) An online paging algorithm is *lazy* if it only evicts a page on a cache miss. Note that FIFO and LRU are lazy, while FWF is not. Prove that for every two lazy online paging algorithms A and B, $A \leq_n B$ and $B \leq_n A$ for every n.

[This fact explains why it is difficult to separate the performance of different paging algorithms.]

Problem 4

Fix a set of pages N and a cache size k. As in Lecture #4, we use a function $f(\cdot)$ to impose locality of reference, as follows: a request sequence is *legal for f* if and only if in every window of ℓ consecutive page requests, there are requests for at most $f(\ell)$ *distinct* pages. We obviously assume that $f(\ell)$ is an integer between 1 and ℓ (for each ℓ); we also assume that f is nondecreasing and concave, in the sense of Lecture #4.

(a) (5 points) Prove that this model of locality has no implications for the competitive ratio of an algorithm. Precisely, prove that for every algorithm A and function f with $f(2) \geq 2$ and $\lim_{\ell\to\infty} f(\ell) = |N|$, the competitive ratio of A on sequences legal for f is the same as its competitive ratio for arbitrary sequences.

(b) (5 points + 5 extra credit points) For algorithms A and B and a function f, define $A \leq_n^f B$ as in Problem 3 except with I_n replaced by the subset of length-n sequences that are legal for f. Exhibit a function f that separates the FIFO and LRU algorithms in the sense that: (i) $LRU \leq_n^f FIFO$ for every n; and (ii) it is not the case that $FIFO \leq_n^f LRU$ for every n.

[You'll get 5 points if (ii) holds, and 5 extra credit points if you prove that (i) holds as well.]

Problem 5

Recall from Lecture #4 the definition of a selfish routing network, of an equilibrium flow, and of the price of anarchy. For a given network G with continuous and nondecreasing edge cost functions and a traffic rate r between a source s and sink t, let $\pi(G, r)$ denote the ratio of the costs of equilibrium flows at rate r and rate $r/2$. By the resource augmentation result from lecture, the price of anarchy in the network G at rate r is at most $\pi(G, r)$.

(a) (8 points) Prove a "loosely competitive" guarantee using the above resource augmentation bound: for every G and r, and for at least a constant fraction of the traffic rates $\hat{r}$ in $[r/2, r]$, the price of anarchy in G at traffic rate $\hat{r}$ is $O(\log \pi(G, r))$.

(b) (7 points) Prove that for every constant K, there exists a network G with continuous, nondecreasing edge cost functions and a traffic rate r such that the price of anarchy in G is at least K for every traffic rate $\hat{r} \in [r/2, r]$.