CS369N: Problem Set #1

Due in class on Friday, October 23, 2009

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

(10 points) Recall that when $\text{cost}(A, z)$ measures the running time of an algorithm on an input, instance optimality asserts that an algorithm A should be as fast (up to a constant factor α) as every other correct algorithm A' on the input z. Obviously the strength of such a guarantee depends on how small α is. The point of this problem is to show that a necessary condition for an interesting guarantee is for α to be independent of the size of the program A'.

Precisely, consider an arbitrary search problem — i.e., given an instance, exhibit a correct solution or report than none exist — in which every correct algorithm takes at least linear time (e.g., from reading the input) and in which the correctness of a purported solution can be verified in linear time. Show that for every such problem there is an instance-optimal algorithm A in the sense that $\text{cost}(A, z) \leq f(|A'|) \cdot \text{cost}(A', z)$ for every input z and (correct) algorithm A', where $|A'|$ denotes the size (e.g., number of lines of code) of the algorithm A' and $f(|A'|)$ is some function that depends only on $|A'|$ and not on the length of z.

[Hint: consider enumerating all programs of at most a given length.]

Problem 2

Recall the original notion of instance-optimality discussed in Lecture #1 and the Fagin/Lotem/Naor paper (where the constant factor is independent of the program length, as per the previous problem).

(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 3

Recall from Lecture #2 the model of online paging with access graphs.

(a) (4 points) Consider the online paging problem in which the access graph G is a line on $k+1$ pages. (Recall that in every legal input, every page must be a neighbor of the previous one; the first page can be arbitrary.) Recall that LRU has competitive ratio 1 on this graph. Show that the competitive ratio on G of the FIFO paging algorithm is k, the size of the cache.

(b) (6 points) Consider the online paging problem in which the access graph G is a cycle on $k+1$ pages. Recall that LRU has competitive ratio k on this graph. Give an online algorithm that has competitive ratio $O(\log k)$ for G.

(c) (5 points) Suppose we change the cost model so that every cache hit costs 1 and every cache miss costs $m \geq 1$. Recall from Lecture #2 how we break a sequence σ into blocks $\sigma_1, \sigma_2, \ldots, \sigma_b$, where each σ_i is a maximal sequence in which only k distinct pages are requested. A simplistic and sequence-dependent model of locality is as follows: for a given σ, define its locality $L(\sigma)$ as its average block length (the number of requests divided by b). Note that this does make at least some intuitive sense as a locality measure. Prove that for every sequence σ with locality at least αm, the cost (in this new cost model) of LRU is at most $1 + \frac{k-1}{\alpha+1}$ times that of the optimal (furthest-in-future) algorithm.

(d) (Extra credit) Formulate a definition of a "graph-independent" algorithm (like FIFO or LRU but unlike the one you presumably designed in part (b)). Say whatever you can (examples, lemmas, conjectures) about whether or not there are better graph-independent algorithms than LRU.

Problem 4

Recall the Markov paging model from Lecture #2.

(a) (4 points) A degenerate type of Markov chain has the same outgoing distribution at every node — this represents the case where each page is an independent draw from a fixed distribution over the N pages. What is the optimal online algorithm in this case? (Assume you know this distribution in advance.) Prove your answer.

(b) (8 points) Now consider a general Markov chain. Design an algorithm that takes as input the description of a Markov chain on N pages, a cache size k, and computes the corresponding optimal online algorithm (that with minimum average page fault rate as the sequence length goes to infinity). Your algorithm does not need to run in polynomial time but you should prove that it eventually terminates with a correct answer. Estimate the time and space requirements of your algorithm.

[Hint: There are multiple ways to solve this, but it might help to think first along dynamic programming lines.]

(c) (8 points) Recall that the LPR algorithm needed to compute the following: given the pages requested so far and two distinct pages p, q, what is the probability that p's next request will precede q's? How would you efficiently compute (or at least closely approximate) this probability in time polynomial in the size of the given Markov Chain? Solve this problem for as big a class of Markov chains that you can.

Problem 5

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,

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