CS264: Homework #2

Due by midnight on Thursday, January 26, 2017

Instructions:

(1) Form a group of 1-3 students. You should turn in only one write-up for your entire group. See the course site for submission instructions.

(2) Please type your solutions if possible and feel free to use the LaTeX template provided on the course home page.

(3) Students taking the course for a letter grade should complete all exercises and problems. Students taking the course pass-fail only need to complete the exercises.

(4) Write convincingly but not excessively. Exercise solutions rarely need to be more than 1-2 paragraphs. Problem solutions rarely need to be more than a half-page (per part), and can often be shorter.

(5) 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. (Exception: feel free to use your undergraduate algorithms textbook.) Cite any sources that you use, and make sure that all your words are your own.

(6) If you discuss solution approaches with anyone outside of your team, you must list their names on the front page of your write-up.

(7) Exercises are worth 5 points each. Problem parts are labeled with point values.

(8) No late assignments will be accepted.

Lecture 3 Exercises

Exercise 6

Prove that for every cache size $k \geq 1$ and every page sequence σ, $\text{cost}(LRU, k+1, \sigma) \leq \text{cost}(LRU, k, \sigma)$.

[As usual, $\text{cost}(A, k, \sigma)$ denotes the number of page faults incurred by the paging algorithm A on the page sequence σ when the cache size is k.]

Exercise 7

Recall the resource augmentation guarantee: for every cache size k (for LRU) and $h \leq k$ (for the offline optimal algorithm),

$$\text{cost}(LRU, k, \sigma) \leq \frac{k}{k-h+1} \cdot \text{cost}(OPT, h, \sigma).$$

Prove a matching lower bound. That is, for every choice of k and $h \leq k$, every constant $\alpha < \frac{k}{k-h+1}$, and every deterministic paging algorithm A, there exists a sequence σ such that $\text{cost}(A, k, \sigma) > \alpha \cdot \text{cost}(OPT, h, \sigma)$.

Exercise 8

Suppose we try to interpolate between online and offline algorithms using the notion of *lookahead*. Precisely, an ℓ-*lookahead* paging algorithm makes each eviction decision using only knowledge of past requests and the ℓ next requests. (Online algorithms are the $\ell = 0$ case, offline algorithms the $\ell = +\infty$ case.) Prove that for every finite $\ell \geq 0$ and cache size $k \geq 1$, every deterministic ℓ-lookahead algorithm has competitive ratio at least k.

[Interpretation: this result gives yet another sense in which worst-case competitive analysis gives misleading information about online paging algorithms — it suggests that any finite amount of lookahead is completely useless.]

Lecture 4 Exercises

Exercise 9

Prove that the model of locality introduced in this lecture has no implications for the competitive ratio of an algorithm. Precisely, prove that for every concave function f with $f(2) = 2$ and $\lim_{\ell \to \infty} f(\ell) = N$ (where N is the total number of pages), every cache size k, and every deterministic online algorithm A, the competitive ratio of A on the subset of sequences legal for f is at least k.

Exercise 10

Recall the proof of the first lower bound (of $\alpha_f(k)$) given in this lecture. Given a concave function f with $f(2) = 2$, the proof considered a page sequence σ that consists of an arbitrarily large number of phases. Each phase consists of the same sequence of page requests: m_2 requests in a row for page p_1, m_3 requests in a row for page p_2, ..., and finally m_k requests in a row for page p_{k-1}. (Recall that m_j denotes the number of (consecutive) values of i for which $f(i) = j$, and that m_j is nondecreasing in j by concavity of f.) Prove that the sequence σ conforms to f.

Problems

Problem 3

Recall the online paging problem from Lecture #3. We proved that every deterministic paging algorithm has competitive ratio at least k, where k is the cache size. We considered only deterministic algorithms, because randomized paging algorithms are rarely used in practice. But the latter are fun to think about from a theoretical standpoint. The goal of this problem is to prove that randomized paging algorithms can have exponentially better competitive ratios than deterministic ones.

When a randomized paging algorithm suffers a page fault and has a full cache, it can use an arbitrary probability distribution to choose which page to evict from its cache. The competitive ratio of such an algorithm A is defined according to its expected number of page faults:

$$\max_{\sigma} \frac{\mathbf{E}[\text{cost}(A, \sigma)]}{\text{cost}(OPT, \sigma)},$$

where the expectation is over the random flips of the algorithm A.[1]

We study the following randomized paging algorithm. The algorithm keeps track of the blocks of the input sequence, as defined in Lecture #3 (maximal sequences with requests to only k distinct pages). When the algorithm incurs a page fault and the cache is not full (as in the first block), it simply brings the requested page into the cache. If the algorithm incurs a page fault in the block σ_i with a full cache, then among the pages in the cache that have not yet requested in σ_i, it chooses one uniformly at random to evict.

[1] This is sometimes called the *oblivious adversary* model: an "adversary" commits to the worst sequence σ she can think of, oblivious to the outcomes of the coin flips that will later be tossed by A. Positive results are much harder to come by with *adaptive adversaries*, which are permitted to define the input sequence σ online, adapting to the previous random choices of the algorithm.

(a) (2 points) Prove that the algorithm is well defined, meaning that when there is a page fault in block σ_i with a full cache, there is at least one page in the cache that has not yet been requested in the block σ_i.

(b) (2 points) Prove that the algorithm can only fault on a page p in a block σ_i on the first request to p in σ_i.

(c) (2 points) Consider a block σ_i with $i > 1$. Let X_i and Y_i denote the k distinct pages requested in σ_{i-1} and σ_i, respectively. Call the pages of $X_i \cap Y_i$ and $Y_i \setminus X_i$ *old* and *new* pages, respectively. Prove that, on the first request to a new page, the algorithm incurs a fault (with probability 1).

(d) (8 points) Consider again a block σ_i with $i > 1$. Let t_i denote the number of new pages ($t_i = |Y_i \setminus X_i|$). Among the old pages, suppose p is the jth one to be requested in the block σ_i (for some $j \in \{1, 2, \ldots, k - t_i\}$). Prove that the probability (over the algorithm's random choices) that the algorithm incurs a fault on the first request to p in block σ_i is at most

$$\frac{t_i}{k - j + 1}.$$

[Hints: Argue that the worst case is when all requests to new pages precede all requests to old pages. Prove (carefully!) that the cache slots occupied by the new pages and the already-requested $j - 1$ old pages are precisely the slots occupied by these old pages at the beginning of the block, plus a random subset of the other $k - j + 1$ slots of size t_i.]

(e) (4 points) Prove that the expected total number of page faults incurred by the algorithm over all blocks after the first is at most $\mathcal{H}_k \cdot \sum_{i>1} t_i$, where $\mathcal{H}_k$ denotes the kth Harmonic number $\sum_{j=1}^{k} \frac{1}{j}$.[2]

(f) (5 points) Prove that the offline optimal algorithm incurs at least $\frac{1}{2} \sum_{i>1} t_i$ page faults.

[Hint: for each $i > 1$, how many distinct page requests are there in the blocks σ_{i-1} and σ_i combined?]

(g) (2 points) Conclude that, modulo an additive term from the first block, $\mathbf{E}[\text{cost}(A, k, \sigma)] \leq 2\mathcal{H}_k \cdot \text{cost}(OPT, k, \sigma)$, for every cache size k and input σ.

(h) (6 points extra credit) Consider the following simpler randomized paging algorithm: whenever there is a page fault and the cache is full, evict one the cache's k pages uniformly at random. Is this algorithm also $O(\log k)$-competitive in expectation? Prove your answer.

Problem 4

Note: This problem makes use of Markov's inequality, the simplest large deviation inequality. See, for example, Lecture #18 from the instructor's CS261 course (notes posted on the Web site).

In Lecture #3 we discussed the idea of weakening the offline optimal algorithm to obtain smaller competitive ratios. Another idea is to strengthen the online algorithm. One way of doing this is to provide the algorithm with partial information about the input z; such algorithms are called *semi-online*.

(a) (3 points) Consider the following problem: you have a single bin with capacity 1, and items with positive sizes $s_1, s_2, \ldots$ arrive online, one-by-one. When an item arrives, you have to choose whether or not to put it in the bin. The sum of the sizes of the items in the bin must be at most 1, and you are not allowed to take items out of the bin. The objective function is to maximize the sum of the sizes of the items packed in the bin.[3]

Prove that for every constant $c > 0$, there is no deterministic algorithm with competitive ratio at least c.

[2] Note that $\mathcal{H}_k$ is roughly $\ln k$ (as seen by approximating the sum with the integral $\int_1^k \frac{dx}{x}$).

[3] Note that with maximization objective like this, competitive ratios will be at most 1, the closer to 1 the better.

(b) (5 points) Suppose that you are told 1 bit of information before any items arrive: whether or not there will be at least one item that has size at least $\frac{1}{2}$. Show that, with this information, there is a deterministic online algorithm with competitive ratio (at least) $\frac{1}{2}$.

(c) (8 points) The previous part motivates a general definition. By a *b-bit oracle*, we mean a function a from problem instances to $\{0,1\}^b$. An *online algorithm with b bits of advice* receives the bits $a(z)$ in advance of the input z, where a is a b-bit oracle. (In the previous part, $b = 1$.)

It turns out that there is a close connection between randomized online algorithms and online algorithms with advice. For example, consider the online paging problem. Fix a cache size k and a finite universe U of all pages; we consider page request sequences of length n, with $n \to \infty$. (k and U are constants, independent of n.) Let A be a randomized online paging algorithm (without advice) that is c-competitive in expectation (cf., Problem 3). We can imagine that A makes all of the coin flips that it might need in advance (to be examined as needed during A's execution), and use $r \in \{0,1\}^\ell$ to denote the outcome of these flips (where ℓ can depend on n and is big enough to describe all of the randomness needed). Note that, given r, A is a deterministic algorithm; we write $\text{cost}(A, r, \sigma)$ for the performance of the randomized algorithm A on a given input σ for a given random string r.

Prove that, for every n, there exists random strings $r_1, r_2, \ldots, r_t \in \{0,1\}^\ell$ such that:

(i) there are constants α, d (independent of n) such that $t \leq \alpha n^d$;

(ii) for every input σ of length n, there exists an index $i \in \{1, 2, \ldots, t\}$ such that $\text{cost}(A, r_i, \sigma) \leq 2c \cdot \text{cost}(OPT, \sigma)$.

[Hints: use Markov's inequality to upper the probability (over the random string) that the cost incurred by the online algorithm is more than $2c$ times that of the offline optimal. Repeat this experiment a polynomial number of times, and then take a union bound over all possible sequences of length n.]

(d) (4 points) Conclude that there is a deterministic online algorithm with $O(\log n)$ bits of advice that is $O(\log k)$-competitive on length-n input sequences.