CS 141, Advanced Topics in Algorithms
Spring 2004
Homework 4b
Due Monday, February 16

1. **[20 Points] The Dynamic List Access Problem.** ***Carried over from the last homework set.*** In class we looked at the MTF online algorithm for the List Access Problem. Specifically, we looked at the **Static** List Access Problem where only FINDs were allowed - there were no INSERT and DELETE operations permitted in the request sequence.

   Now, consider the more general **Dynamic** List Access Problem which differs from the static version in three ways:

   (a) Any FIND operation may fail. That is, the item might not be found. In this case, the actual cost is $\ell + 1$ where ℓ was the length of the list at the time that the FIND was performed.

   (b) INSERT operations are permitted in the request sequence. An inserted object is placed at the end of the list. If the length of the list prior to the INSERT was ℓ, then the actual cost of this operation is $\ell + 1$. After inserting the element at cost $\ell + 1$, you may move it to any position earlier in the list at no cost (just like in a successful FIND operation).

   (c) DELETE operations are permitted in the request sequence. The cost of the DELETE is simply the cost of finding the object. (However, the potential function changes as a consequence of the DELETE, so a DELETE differs from a FIND in this way!)

   Describe a modification of the MTF algorithm to handle all three of these operations. Show that when your modified MTF algorithm is applied to the Dynamic List Access Problem, it still maintains a competitive ratio of 2. (That is, it is 2-competitive with respect to an optimal offline algorithm.)

2. **[15 Points] The Ski Rental Problem.** Professor I. Lai has a friend, Professor Sue Perfast, who is visiting him for the winter. Professor Perfast is a ski fanatic. Being the accommodating host that he is, Professor Lai agrees to go skiing with Professor Perfast whenever she wishes.

   Since he doesn't know in advance how many times he will be asked to go skiing, Professor Lai's dilemma is this: How many times should he rent skis before taking the plunge (so to speak) and buy a pair of skis? You may assume that it costs \$30 to rent a pair of skis for a day and \$300 to buy a pair of skis. *Moreover, you cannot buy the skis on the same day that you plan to use them – there isn't time for that! So, if you decide to purchase a pair of skis, there is a chance that you'll never use them again.*

Professor Lai needs an online algorithm! The algorithm gets a "request to go skiing". If Professor Lai has not already purchased a pair of skis on an earlier day, then he must rent. If he bought skis earlier, he will use them. Such an algorithm would rent for the first j times (for some j) and would then buy skis on the evening after the j^{th} ski trip. The objective, of course, is to minimize the total cost incurred. An offline algorithm would know in advance how many "requests to go skiing" there will be and could therefore decide to either rent all the time or buy before the first ski trip.

(a) Describe an online algorithm for the ski rental problem and prove that it is 2-competitive.

(b) Prove that no deterministic online algorithm for this problem can be better than 2-competitive.

3. **[20 Points] The Investment Problem!** You've been hired as Chief Algorithms Officer at the investment firm of Weil, Proffet, and Howe. ("Chief Algorithms Officer" is a real job title at several big companies now! Search Google on "chief algorithms officer" and you will find an astounding number of hits!) Here's your first task: Over the course of the next n days, you will be presented each day with a share price for a particular company's stock. Your company owns one share of that stock. You must sell your share sometime during that n day period. Your objective is to find the best sell date in that period. Once you sell on a given day, the game is over and you walk away with the value at which you sold the stock.

   Unfortunately, you get the share value data day-by-day. You must either sell at the price quoted that day or take your chances and see what you are offered subsequently. Fortunately, there is a known range on the value of that stock: The stock is guaranteed (or in real life, perhaps "confidently estimated") to stay in the range m to M, $m \leq M$. Moreover, if you haven't sold your stock at the end of the n days (when the offers end) you may cash it in for its low value of m.

   Let $\phi = \frac{M}{m}$ (this ratio is sometimes called the "global fluctuation ratio").

   (a) Describe a deterministic online algorithm for this problem. Your algorithm should be strictly $\sqrt{\phi}$ competitive. You may assume that you know m and M.

   (b) Assume that you are told ϕ but you are not given m nor M. Show that no deterministic online algorithm can be better than ϕ competitive in this case.

4. **[25 Points] LFD is an Optimal Offline Paging Algorithm.** In class we mentioned that LFD (Longest-Forward-Distance) is an optimal offline algorithm for the paging problem. Recall that the algorithm works as follows: On each page fault, the algorithm evicts the page from fast memory whose next request is furthest in the future.

   It may seem intuitive that LFD is optimal, but intuition on these problems is often misleading! Here we'll prove the optimality of LFD formally in two parts.

   (a) First, consider the following claim:

Claim 1 *Let OFF be any offline paging algorithm. Without loss of generality (as we saw in class!), OFF is a demand paging algorithm. Let σ be any request sequence. For any i, $1 \leq i \leq |\sigma|$, it is possible to construct another offline algorithm OFF_i that satisfies the following properties:*

i. OFF_i processes the first $i-1$ requests exactly as does OFF.

ii. If the i^{th} request in σ is a page fault, then OFF_i evicts from memory the page with the longest forward distance.

iii. $OFF_i(\sigma) \leq OFF(\sigma)$.

Prove this claim. (*Note:* To do so, you will want to specify how OFF_i behaves with respect to OFF after the i^{th} request.) Make sure that your proof is entirely rigorous.

(b) Now show how the above claim can be applied repeatedly to prove that LFD is optimal.