CS 182
Advanced Topics in Algorithms
Spring 2003
Problem Set 4b
Due Tuesday, February 18 in class

1. **[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.

2. **[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 latest.

   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 ALG be any offline paging algorithm. Without loss of generality (as we saw in class!), ALG 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 ALG_i that satisfies the following properties:*

   i. ALG_i processes the first $i-1$ requests exactly as does ALG.

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

iii. $ALG_i(\sigma) \leq ALG(\sigma)$.

Prove this claim. (*Note:* To do so, you will want to specify how ALG_i behaves with respect to ALG 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.

3. **[15 Points] LFU and LIFO are not Competitive!** Prove that LFU and LIFO are not competitive online paging algorithms. LFU, Least-Frequently-Used, is a demand paging algorithm which evicts the page which has been used least since entering fast memory. LIFO, Last-In-First-Out, evicts the page that was most recently moved into fast memory. To show that these algorithms are not competitive, you must show that there is no constant c for which these algorithm are c-competitive.