Prof. Dr. Susanne Albers
Dr. Dimitrios Letsios
Dario Frascaria
Lehrstuhl für Theoretische Informatik
Fakultät für Informatik
Technische Universität München

Problem set 3
May 4, 2015
Summer Semester 2015

Online and Approximation Algorithms

Due May 11, 2015 before class!

Exercise 1 (Dynamic List Update problem - 10 points)

We modify the list update problem by adding the operations INSERT(a) and DELETE(a). Let n be the current length of the list. Whenever INSERT(a) is called, the list is searched for element a. If a is already in the list, the costs for the operation are equal to its position. If not it is inserted to position $n+1$ of the list. In this case the cost is $n+1$. In addition we are allowed to move the inserted item to any position in the list. If DELETE(a) is called, the list is searched for element a. If a is found, the costs for the operation are equal to its position, otherwise the cost is n. Move-To-Front (MTF) moves the newly inserted element to the front of the list. Show that MTF is still 2-competitive.

Exercise 2 (List Update, Paid exchanges - 10 points)

(a) Show that any optimal offline algorithm for the list update problem cannot avoid using paid exchanges.
(Hint: 3 elements and 4 requests are sufficient.)

(b) Show that there exists an optimal offline algorithm that uses **only** paid exchanges.

Exercise 3 (RMARK - 10 points)

RMARK is the randomized online paging algorithm that works as follows: Initially all pages are unmarked. Whenever a page is requested it becomes marked. When a page is brought into the cache it replaces a randomly and uniformly chosen page from the set of unmarked pages that are in the cache. When all pages in the cache are marked and a cache fault occurs, all pages become unmarked. Prove that RMARK is H_k-competitive against oblivious adversaries when the total number of pages is $k+1$, where k is the size of the fast memory.

Exercise 4 (RMARK II - 10 points)

Build a sequence such that the cost of RMARK(with specific random choices) is greater than H_k times the cost of the optimal offline algorithm.
(Hint: There is an example with 4 pages and cache size equal to 2.)