Technische Universität München
Fakultät für Informatik
Lehrstuhl für Theoretische Informatik
Prof. Dr. Susanne Albers
Jens Quedenfeld

Winter Semester 2017/18
Problem Set 2
October 30, 2017

Online and Approximation Algorithms

Due November 6, 2017 at 10:00

Exercise 1 (Marking Algorithm – 10 points)

Consider a sequence of requests σ for pages from a memory system with a fast memory of size k. A *k-phase partition* of σ is obtained as follows: we partition the request sequence into phases such that each phase is the maximal sequence containing k pairwise distinct pages that follows the previous one, except possibly the last phase which contains requests to at most k different pages.
Given a k-phase partition of σ, we define a *marking* of the pages requested as follows. At the beginning of a phase, all pages are unmarked. During the phase, a page is marked upon the first request to it. Recall that an online paging algorithm is a *marking algorithm*, if it never evicts a marked page.

(a) Prove that every marking algorithm is k-competitive.

(b) Prove that FIFO is not a marking algorithm.

Exercise 2 (Demand Paging – 10 points)

Paging algorithms that do not evict pages unless there is a page fault are called *demand paging*. Prove that any paging algorithm can be modified to be demand paging without increasing the overall number of memory replacements on any request sequence.

Exercise 3 (Amortized Analysis – 10 points)

Suppose that we serve a sequence of n requests, where the cost for serving i-th request is

$$c_i = \begin{cases} i, & \text{if } i \text{ is an exact power of 2} \\ 1, & \text{otherwise} \end{cases}$$

What is the worst-case cost for serving the requests? Use amortized analysis with an appropriate potential function in order to show that the total cost is upper bounded by $3n$.

Exercise 4 (Implementing a Queue using two Stacks – 10 points)

Implement a FIFO queue by using two stacks with the operations PUSH, POP and IS_EMPTY.

(a) How can you implement the queue operations ENQUEUE and DEQUEUE? Write pseudo code.

(b) Show that you algorithm is correct.

(c) Analyze the amortized runtime.