Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen (LEA)
Prof. Dr. Susanne Albers
Moritz Fuchs

Spring Semester 2014
Problem set 2
April 16, 2014

Online and approximation algorithms

Due April 23, 2014 before class!

Exercise 1 (Demand Paging - 10 points)

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

Exercise 2 (Marking Algorithm - 10 points)

In a marking algorithm we partition the request sequence σ into phases s.t. every phase i for $i \geq 1$ is the maximal sequence following phase $i-1$ that contains at most k distinct pages. That means, that the first request following phase i is distinct from all pages in phase i. We assume that phase 0 is the empty sequence. This kind of partition is called k-phase partition.
We associate with each page p a mark bit m_p. If $m_p = 1$ we say that p is marked, otherwise we say that p is unmarked. At the beginning of every phase all pages become unmarked. Whenever an unmarked page p is accessed, we mark it. Marked pages are never evicted from the cache, i.e. once a page was requested during a phase it will not leave the cache until the next phase.

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

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

Exercise 3 (Toy blocks - 10 points)

Assume we have n toy blocks in a box and are allowed to perform three types of actions:

1. Put a block onto the tower (or onto the ground if no tower exists). This costs 1 time step.
2. Knock one block off the tower and put it back into the box. This costs 1 time step.
3. Knock k blocks off the tower and put them back into the box. This costs k time steps.

(a) How much time does one action take at most?

(b) How much time (amortized) is needed to perform a sequence of m actions?
Hint: Use a potential function!

Exercise 4 (Least frequently used - 10 points)

Recall, that least frequently used (LFU) is the online paging algorithm that replaces the page that has been used least since it entered the cache.
Prove that LFU is not c-competitive for any $c > 0$.