HS 2008 Prof. Dr. R. Wattenhofer / Raphael Eidenbenz / Roland Flury

Discrete Event Systems
Solution to Exercise 12

1 Bin Packing

The algorithm mentioned in the exercise is 2-competitive. The proof works as follows: Consider the bins in the order in which they were closed. Consider two consecutive bins i and $i+1$. Assume that the algorithm fills bin i up to level $x \leq 1$. The next item (the first to be put into bin $i+1$) must be of size *larger* than $1-x$. Otherwise, the algorithm would not have opened a new bin. Therefore, that any *two consecutive bins* must have total size strictly more than 1. Because the optimum must also use at least one bin to store this size, the algorithm requires at most twice as many bins.

To show that there is indeed a sequence which the optimum can serve using half the number of bins as the algorithm, assume the following input sequence:

$$\sigma = \underbrace{\frac{1}{2}, \frac{1}{n}, \frac{1}{2}, \frac{1}{n}, \frac{1}{2}, \frac{1}{n}, \ldots, \frac{1}{2}, \frac{1}{n}}_{2n \text{ items}}.$$

Clearly, our algorithm ALG needs to open a new bin after every other item. That is, the number of bins opened by ALG is $c(ALG) = n$. On the other hand, the optimal algorithm OPT can always put two items of size $1/2$ together, thus requiring $n/2$ bins. For all additional n items of size $1/n$, it merely requires an additional bin. Hence, $OPT = n/2 + 1$.

The competitive ratio c for sequence σ is therefore

$$c \geq \frac{ALG}{OPT} = \frac{n}{\frac{n}{2}+1} = \frac{2n}{n+1}.$$

For large n, this asymptotically tends to 2.

2 Paging

a) i) FIFO (First-in/First-out): Replace the page that has been in the cache longest.

The FIFO strategy is 3-*competitive*. In order to prove this, observe that on any consecutive input subsequence containing 3 or fewer distinct page references, FIFO incurs 3 or fewer page faults. Now, consider a 3-*phase partition* of the input sequence σ. A 3-phase partition is defined as follows: Phase 0 is the empty sequence. For every $i \geq 1$, phase i is the maximal sequence following phase $i-1$ that contains at most 3 distinct page requests; that is, if it exists phase $i+1$ begins on the request that constitutes the 4th distinct page request since the start of the ith phase.

By the above observation, it is clear that FIFO incurs at most 3 page faults for any phase $i \geq 1$, because it cannot fault twice on the same page. For any $i \geq 1$, let q be the first request of phase i and consider the input sequence starting with the second request of phase i up to and including the first request of phase $i+1$. The optimal algorithm OPT has 2 pages (not including q, and there are 3 requests (except if phase i is the last phase) in this sequence not counting q, so that OPT must incur at least 1 page fault in this phase.

Combining the two bounds, we have

$$ALG(\sigma) \leq 3 \cdot OPT(\sigma) + \alpha,$$

where $\alpha \leq 3$ is the maximal number of page faults incurred by FIFO during the last phase.

ii) LFU (Least Frequently Used): Replace the page that has been requested the least since entering the fast memory.

The LFU strategy is *not competitive*. Consider the following request sequence:

$$\sigma = p_1 \ , \ p_1 \ , \ p_2 \ , \ p_2 \ , \ p_3 \ , \ p_4 \ , \ p_3 \ , \ p_4 \ , \ p_3 \ , \ \ldots$$

In this sequence, LFU keeps on exchanging p_3 and p_4 ad infinitum, while the optimum can keep these two pages in the cache.

iii) LIFO (Last-in/First-out): Replace the page most recently moved to the cache.

For the same reason as LFU, the LIFO strategy is also *not competitive*. Consider the following request sequence:

$$\sigma = p_1 \ , \ p_2 \ , \ p_3 \ , \ p_4 \ , \ p_3 \ , \ p_4 \ , \ p_3 \ , \ \ldots$$

In this sequence, LIFO keeps on exchanging p_3 and p_4 ad forever. It is therefore not competitive.

iv) LRU (Least Recently Used): When eviction is necessary, replace the page whose most recent request was the earliest.

Like the FIFO strategy, LRU is 3-*competitive*. The reason is that (like FIFO), LRU has the property that on any consecutive input subsequence containing 3 or fewer distinct page references, it incurs at most 3 page faults. The remainder of the proof is then equivalent to the FIFO case.

v) FWF (Flush When Full): Whenever there is a page fault and there is no space left in the cache, evict *all* pages currently in the cache.

The FWF algorithm is also 3-*competitive*. Consider the first phase, i.e., the first consecutive input subsequence containing 3 distinct page references. Clearly, FWF does not operate a flush. Now, consider the subsequent phase. In this phase, there can be at most one flush and hence, three page faults. Similarly, in every subsequent phase, there can be at most one flush and three page faults. From this observation, the proof follows like in the FIFO case.

b) We prove the following theorem:

Theorem 2.1. *There exists no deterministic online paging algorithm ALG with a competitive ratio better than* 3.

Proof. Assume that there are 4 pages, $p_1, \ldots, p_4$. We prove that there is an arbitrarily long request sequence σ for which $|\sigma| = ALG(\sigma) \geq 3 \cdot OPT(\sigma)$. Without loss of generality, assume that ALG initially holds p_1, p_2, and p_3 in its cache. We define a "cruel" request sequence σ inductively: $r_1 = p_{k+1}$, and r_{i+1} is defined to be the unique page not in ALG's cache just after servicing the request sequence $r_1, \ldots, r_i$. In other words, the adversary always requests the one page that ALG does not have in its cache. Clearly, σ can be made arbitrarily long and ALG has a page fault on each request in σ, hence $|\sigma| = ALG(\sigma)$.

We now show, however, that it is possible to serve every request sequence σ with at most $\sigma/3$ page faults. Specifically, consider an imaginary offline algorithm LFD that knows the future and always evicts the one page from the cache whose *next request is latest.* Suppose that for servicing the ith request, r_i, LFD evicts the page p. By the definition of LFD, and since there are 4 pages in all, it must be that all the pages in the cache (except perhaps r_i) must be requested prior to the next request of p. Hence, LFD has a page fault at most once every 3 requests.

Clearly, it holds that $OPT(\sigma) \leq LFD(\sigma) \leq \sigma/3$, and the theorem therefore follows because of $|\sigma| = ALG(\sigma)$. □