Randomized paging algorithms

You have a fast memory (cache) of size k and a large slow memory. Given a sequence of page requests, serving a request r is done at cost 0 of the page is in the cache and at cost 1 if the page is not there: it has to be brought into the cache. The goal is to design an algorithm to minimize the number of page faults (requests that incur a cost). The algorithm chooses which page to evict from the cache to make room for the current request. The algorithm is online is its choice is done without knowledge of future requests.

Theorem 1 *No deterministic algorithm can be better than k-competitive.*

Proof: stay tuned.

Consider the following randomized algorithm called MARKING: when a request r arrives, if the page is in memory then mark it, and if it is not in memory, evict a random unmarked page from the cache, fetch r and mark it. When we need to do an eviction but all pages are marked, first we unmark all pages.

Theorem 2 *MARKING is $2H_k$-competitive*

Fix a sequence of n pages requests. Decompose into phases as follows. Phase 1 starts with request σ_1. A phase starting with requests σ_i ends just before request σ_j such that the set of pages requested at times $\{i, \ldots, j-1\}$ has size k but the set of pages requested at times $\{i, \ldots, j\}$ has size $k+1$. In each phase, it is easy to see that MARKING starts by unmarking every page, and ends with all pages marked. Consider the ith phase, and say that a request is *old* if the pages had already been requested in phase $i-1$, *new* otherwise; let m_i be the new requests and $k - m_i$ be the old requests.

During phase i, the expected number of page faults of MARKING is maximized if the new requests precede the old requests. Each new requests induces a page fault, for a total of m_i. The request to the jth distinct old page induces a page fault if and only if the page requested is not within the remaining $k - m_i - (j-1)$ pages still in the cache and still unmarked at this point. These pages are a random subset of the $k-(j-1)$ pages of phase $i-1$ that have not yet been requested, so the fault probability is $1-(k-m_i-(j-1))/(k-(j-1))$. From this it is easy to deduce that the expected cost of MARKING is at most $\sum_i m_i H_k$.

In phase $i-1$ and phase i, in total $k+m_i$ distinct pages are requested, so the optimal algorithm must pay at least m_i. Summing and taking the double counting into account, OPT must pay at least $(1/2)\sum_i m_i$.

Hence the theorem.

Theorem 3 *No randomized algorithm can be better than H_k-competitive.*

Proof: next time. (Basically, a universe of size $k+1$, p_j is the probability that the algorithm does not have page j in the cache, and build a sequence by requesting the page such that p_j is maximum.)