Solutions to Assignment 1

CS4SH3

February 4, 2002

1. Consider the following list:

$$\underbrace{1,2,3\ldots,i,i+1}_{1},\underbrace{1,2,3,\ldots,i-1}_{2},\underbrace{i+2}_{3},\underbrace{1,2,3,\ldots,i,i+1,i+2}_{4}$$

   If we have a cache of size $i+1$, then we incur $i+1$ faults in segment 1 (because the cache is empty), then we have $i-1$ hits in segment 2, then we have another page fault in segment 3 so we evict 1, and in segment 4 we lag behind by 1 all the way, so we incur $i+2$ page faults. Hence, we incur $i+1+1+i+2=2i+4$ page faults in total.

   Suppose now that we have a cache of size i. Then we incur $i+1$ page faults in segment 1, then we have $i-1$ page faults in segment 2, and one page fault in segment 3, hence $2i+2$ page faults before starting segment 4. When segment 4 starts, we already have pages 1 through $i-1$ in the stack, so we have hits, and then when $i+1$ is requested, we have a fault, and when $i+2$ is requested we have a hit, and hence only one fault in segment 4. Therefore, we have $2i+3$ page faults with a cache of size i.

   To understand this solution, make sure that you keep track of the contents of the cache after each of the four segments has been processed. Note that i has to be at least 3.

2. Let σ be any request sequence, $\sigma=p_1,p_2,\ldots,p_n$, and let $\mathrm{LRU}_i(\sigma)$ be the number of faults that LRU incurs on σ with a cache of size i. We are going to show that for all i and σ, the following property holds:

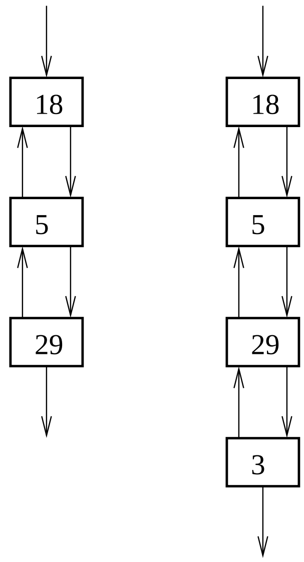
$$\mathrm{LRU}_i(\sigma) \geq \mathrm{LRU}_{i+1}(\sigma) \tag{1}$$

   To show that for any $i<j$, $\mathrm{LRU}_i(\sigma)\geq\mathrm{LRU}_j(\sigma)$, we use (1) repeatedly as follows:

$$\mathrm{LRU}_i(\sigma)\geq\mathrm{LRU}_{i+1}(\sigma)\geq\mathrm{LRU}_{i+2}(\sigma)\geq\ldots\geq\mathrm{LRU}_j(\sigma)$$

   Thus, LRU does not incur Belady's Anomaly.

   To show (1) we define a property of doubly-linked lists which we call embedding. We say that a doubly-linked list of size i can be **embedded** in another doubly-linked list of size $i+1$, if the two doubly-linked lists are identical, except that the longer one has one more item, which is the last one. For example:



Here, the doubly-linked list of size 3 on the left can be embedded in the doubly-linked list of size 4 on the right.

Claim: After each step of processing a sequence of requests, the doubly-linked list of LRU_i can be embedded in the doubly-linked list of LRU_{i+1}.

We prove this claim by induction on the number of steps. Basis Case: if $n = 1$, then both LRU_i and LRU_{i+1} incur a fault and bring in p_1. Induction Step: suppose that the claim holds after step n; we show that it also holds after step $n + 1$. If page p_{n+1} is in the cache of LRU_i, then it is also in the cache of LRU_{i+1} (by the induction hypothesis), so they both have a hit, so they both move p_{n+1} to the beginning of their lists, and the embedding property is preserved.

Suppose that LRU_i has a fault. There are now two subcases: if LRU_{i+1} also has a fault, then they both bring in page p_{n+1} from the slow memory, and put it at the beginning of their lists, and evict the LRU page; hence the embedding property is preserved. If LRU_{i+1} does not have a fault, then p_{n+1} must have been the LRU page of LRU_{i+1} (since it was not in the cache of LRU_i). It follows, since LRU_i has a fault, that both LRU_i and LRU_{i+1} moved p_{n+1} to the top of their lists. By the induction hypothesis, the remaining items are the same, so the embedding property is preserved in this case as well.

Now we use the claim to prove (1). It follows from the embedding claim, that at each step, if LRU_{i+1} has a fault, so does LRU_i, and hence LRU_{i+1} cannot have more faults than LRU_i.

3. We will show that if ALG is any paging algorithm (on-line or off-line), then on any sequence of requests σ, $\mathrm{ALG}(\sigma) \geq \mathrm{OPT}(\sigma)$. As usual, $\mathrm{ALG}(\sigma)$ denotes the number of page faults of ALG on the sequence of requests σ. We assume throughout that all algorithms are working with a cache of a fixed size k.

   We need to prove the following claim:

Claim: Let ALG be any paging algorithm. Let $\sigma = p_1, p_2, \ldots, p_n$ be any request sequence. Then, it is possible to construct an off-line algorithm ALG_i that satisfies the following three properties:

(a) ALG_i processes the first $i-1$ requests of σ exactly as ALG does,

(b) if the i-th request results in a page fault, ALG_i evicts from the cache the page with the "longest forward distance",

(c) $\mathrm{ALG}_i(\sigma) \leq \mathrm{ALG}(\sigma)$

We show first how to use the **Claim** to prove that OPT is in fact the optimal algorithm. Let $\sigma = p_1, p_2, \ldots, p_n$ be once again any sequence of requests. By tha claim, we know that:

$$\mathrm{ALG}_1(\sigma) \leq \mathrm{ALG}(\sigma)$$

Applying the claim agin, we get:

$$\mathrm{ALG}_2(\sigma) \leq \mathrm{ALG}_1(\sigma)$$

continuing this way, we finally obtain that:

$$\mathrm{ALG}_n(\sigma) \leq \mathrm{ALG}_{n-1}(\sigma)$$

Note that ALG_n acts just like OPT on σ, and therefore $\mathrm{ALG}_n(\sigma) = \mathrm{OPT}(\sigma)$, and hence $\mathrm{OPT}(\sigma) \leq \mathrm{ALG}(\sigma)$, and we are done.

We now prove the claim:

Proof of Claim: Divide σ into three segments as follows:

$$\sigma = \sigma_1, p_i, \sigma_2$$

Since on σ_1, ALG_i acts just like ALG, it follows that $\mathrm{ALG}_i(\sigma_1) = \mathrm{ALG}(\sigma_1)$, and also, they both do or don't incur a page fault on p_i. If they don't, then let ALG_i continue behaving just like ALG for the rest of σ, so that $\mathrm{ALG}_i(\sigma) = \mathrm{ALG}(\sigma)$.

However, if they do incur a page fault, ALG_i evicts the LFD page from its cache, and replaces it with p_i. If ALG also evicts the same page (the LFD page), then again, let ALG_i behave just like ALG for the rest of σ, so that $\mathrm{ALG}_i(\sigma) = \mathrm{ALG}(\sigma)$.

Finally, suppose that they both incur a fault at p_i, but ALG evicts some page q and ALG_i evicts some page p, and $p \neq q$. If both $p, q \notin \sigma_2$, then let ALG_i behave just like ALG, except the slots with p and q are interchanged (that is, when ALG evicts from the slot that contained q, ALG_i evicts from the slot that contained p, and when ALG evicts from a slot that contained p, ALG_i evicts from the slot that contained q).

If $q \in \sigma_2$ but $p \notin \sigma_2$, then again let ALG_i, when forced with an eviction, act just like ALG with the two slots interchanged. Note that in this case it may happen that $\mathrm{ALG}_i(\sigma_2) < \mathrm{ALG}(\sigma_2)$, since ALG evicted q, which is going to be requested again, but ALG_i did not.

Otherwise, we can assume that ALG_i evicts page p and ALG evicts page q, $p \neq q$, and:

$$\sigma_2 = p_{i+1}, \ldots, q, \ldots, p, \ldots, p_n$$

Assume that the q shown above is the earliest instance of q in σ_2. As before, let ALG_i act just like ALG with the two slots interchanged. We know for sure that ALG will have a fault at q. Suppose ALG does not have a fault at p; then, ALG never evicted p, so ALG_i never evicted q, so ALG_i did not have a fault at q. Therefore, $\text{ALG}(\sigma_2) \leq \text{ALG}_i(\sigma_2)$.

4. For this exercise assume that the cache is of size k. Otherwise the claim is not true: for example, suppose that we have a cache of size 1, and the following sequences:

$$1, 2, 1, 2$$

Then, in that sequence of 4 requests there are only 2 distinct requests, yet with a cache of size 1, there would be 4 faults, for *any* page-replacement algorithm.

With a cache of size k, LRU is never going to evict a page during this consecutive subsequence, once the page has been requested. Thus, each distinct page request can only cause one fault. Same goes for FIFO. Thus, they are both conservative algorithms.

However, it is possible that half-way through the consecutive subsequence, the cache of FWF is going to get full, and FWF is going to evict everybody. Hence, FWF may have more than one page fault on the same page during this consecutive subsequence.