

Assignment 2 Solutions

1. This problem concerns the offline caching problem, when $k = 3$ and the only pages are a, b, c, d, and e. Consider the following request and eviction schedule, and trace through the exchange argument as defined in class as applied to this schedule.

Request:	e	d	c	b	c	d	a	c	e	a	e	a	e	c	a	e	d	a	e	b
Eviction:	-	-	-	c	d	e	d	-	a	e	a	c	-	a	c	-	b	-	-	e

Note that the above eviction schedule is not claimed to be optimal.

Answer: Throughout this answer, I will refer to the schedule produced above as S , the schedule produced by FIF as F , and the schedule produced by applying i iterations of the exchange argument on S as S'_i .

Let's go ahead and determine FIF's schedule first.

Request:	e	d	c	b	c	d	a	c	e	a	e	a	e	c	a	e	d	a	e	b
Eviction (FIF):	-	-	-	e	-	-	b	-	d	-	-	-	-	-	-	-	c	-	-	e

The first few requests go as expected, after which the memory contents on both S and F are: $\{e, d, c\}$. When b is requested, F evicts e, while S evicts c. As noted in class, to make S'_1 we need to instead evict e, but also need to look ahead to which ever happens first: S evicting e, or a request to c. Since the latter is what happens first (at time step 5), we look at what S evicts for c, which is page d. Hence our eviction schedule looks like as follows:

Request:	e	d	c	b	c	d	a	c	e	a	e	a	e	c	a	e	d	a	e	b
Eviction (S'_1):	-	-	-	e	d*	e*	d	-	a	e	a	c	-	a	c	-	b	-	-	e

Note that I've written d* and e* for time steps 5 and 6, since under the exchange procedure we're imagining that we've added e in replace of d. In reality, this really just means we've kept d, and pretended it's e for future evictions in S , which coincidentally happens in the next time step. Hence S'_1 becomes

Request:	e	d	c	b	c	d	a	c	e	a	e	a	e	c	a	e	d	a	e	b
Eviction (S'_1):	-	-	-	e	-	-	d	-	a	e	a	c	-	a	c	-	b	-	-	e

Now, the next time S'_1 and F differ is on time step 7, where d is evicted in S'_1 instead of b in FIF. Again, we look for which happens first: a request for which S'_1 evicts b, or a request for d. Since these happen to coincide, we can apply the exact swap. Hence we construct S'_2 as

Request:	e	d	c	b	c	d	a	c	e	a	e	a	e	c	a	e	d	a	e	b
Eviction (S'_2):	-	-	-	e	-	-	b	-	a	e	a	c	-	a	c	-	-	-	-	e

Our next difference happens at time step 9, where S'_2 evicts a while FIF evicts d. Hence, we look forward to the next time where S'_2 evicts d or there is a request to a, which happens at the very next time step. In this case, we construct S'_3 as follows:

Request:	e	d	c	b	c	d	a	c	e	a	e	a	e	c	a	e	d	a	e	b
Eviction (S'_3):	-	-	-	e	-	-	b	-	d	e*	a*	c*	-	a*	c*	-	-	-	-	e

Here I write e*, a*, c* again because we are pretending to evict e to make room for d, which we pretend to make room for by evicting a, and so on. This equivalently becomes:

Request:	e	d	c	b	c	d	a	c	e	a	e	a	e	c	a	e	d	a	e	b
Eviction (S'_3):	-	-	-	e	-	-	b	-	d	-	-	-	-	-	-	-	c	-	-	e

Assignment 2 Solutions

Since our schedule is now identical to FIF, we are done.

One nice thing about this exchange argument is that it worked for solutions that aren't already optimal. I guess I'd point out that this isn't always the case - note that sometimes in our proofs we've relied on the optimality of our given solutions in making certain 'exchanges', but we didn't really do that here aside from noting that the number of cache misses didn't increase by applying our exchanges.