CSCI 270 Lecture 15: Optimal Caching

A cache can store n items. There is a sequence of m requests $d_1, d_2, ..., d_m$ known in advance. If an item is requested which is not in the cache, it must be brought into the cache, resulting in a **cache miss**. You may only bring something into the cache when it is requested. The goal is to minimize the number of cache misses.

$k = 2$, initial contents = 'ab'

requests: a b c b c a a b

- What greedy criteria might work for this problem?
- Can you find any counter-examples to these algorithms?
- How is this problem different than the standard version of Caching?
- What must happen at request $k + 1$?

FIF	...				
	Request k	f	o	e	...
	Request $k+1$	d_{k+1}	o	e	...
	...				

OPT	...				
	Request k	f	o	e	...
	Request $k+1$	f	d_{k+1}	e	...
	...				

At request $k + 1$, FIF kicks out f, while OPT kicks out o. Otherwise, the cache contents are completely identical.

Events that can cause our different cache contents to be relevant:

1. OPT kicks out f.
2. Cache miss on o.
3. Cache miss on f. This can't be the first event, because there must first be a cache miss on o.

We will refer to the first of these events as having occured at request j.

Case 1: OPT removes f.

OPT	...				
	Request k	f	o	e	...
	Request $k+1$	f	d_{k+1}	e	...
	...				
	Request $j-1$	f	d_{k+1}	e	...
	Request j	d_j	d_{k+1}	e	...
	...				

We want to make an exchange, thereby transforming OPT into OPT'. We want OPT' to have the same number of cache misses as OPT. We also want OPT' to be identical to FIF through $k+1$ requests.

OPT'	...				
	Request k	f	o	e	...
	Request $k+1$	d_{k+1}	o	e	...
	...				
	Request $j-1$	d_{k+1}	o	e	...
	Request j				...
	...				

What should OPT' do at request j?

Case 2: There is a request on o

OPT	Request k	f	o	e	...
	Request $k+1$	f	d_{k+1}	e	...
	...				
	Request $j-1$	f	d_{k+1}	e	...
	Request j	f	d_{k+1}	o	...
	...				

OPT has to kick something out for o_2, we will say it kicks out the here-to-fore unmentioned e.

We want to make an exchange, thereby transforming OPT into OPT'. We want OPT' to have the same number of cache misses as OPT. We also want OPT' to be identical to FIF through $k+1$ requests.

OPT'	Request k	f	o	e	...
	Request $k+1$	d_{k+1}	o	e	...
	...				
	Request $j-1$	d_{k+1}	o	e	...
	Request j				...
	...				

- What would we like OPT' to do this at this request?
- Does this break any rules?

We will delay the exchange until an event occurs. OPT' has e, but OPT has f. The cache contents are otherwise identical. The possible events are:

- Something (x) is removed for e. We now plan to kick out x at the next event, instead of e. We avoid a cache miss, meaning that OPT wasn't even optimal.
- f is removed for something. We kick out e instead. We avoid a cache miss.
- f is requested. Now we kick out e as originally planned. We simply delayed our cache miss: it occurs for both solutions, but at different times.