DS 320 Algorithms for Data Science	**Lecture #8**
Spring 2023	Prof. Kira Goldner

Optimal Caching and Greedy Exchange

In the optimal caching problem, our computer has a main memory of size n, a cache of size k, and we are presented with a sequence of data $D = d_1, d_2, \ldots, d_m$ that we must process. When an item is not in the cache, we have a cache miss, and must bring the item into the cache and evict something else if the cache is full. Our goal is to give an algorithm that minimizes the number of misses.

Example 1:	a, b, c, b, c, a, b	$k = 2$	cache $= \{a, b\}$
Example 2:	$a, b, c, d, a, d, e, a, d, b, c$	$k = 3$	cache $= \{a, b, c\}$

Determine a cache maintenance algorithm by coming up with an eviction schedule.

A cache maintenance algorithm with an optimal greedy eviction schedule is the *Farthest-in-Future* Algorithm. When a new item needs to be brought into the cache, it greedily evicts the item that is needed the farthest into the future.

Goal: Prove that the Farthest-in-Future Algorithm is optimal. We'll call this schedule S_{FF}.

Definition 1. A schedule is *reduced* if it does the minimal amount of work necessary in a given step.

Lemma 1. *For every non-reduced schedule, there is an equally good reduced schedule (that brings in at most as many items as the original schedule).*

Prove this by construction.

Hint: You might *charge* a miss from one schedule to a miss in another schedule to show that it doesn't have any extra misses.

Proof. For any schedule S, define its reduced schedule $\bar{S}$: whenever S brings in some d that wasn't requested, $\bar{S}$ leaves d in main memory and only brings it in the next time that it's requested. Then the *miss* of $\bar{S}$ in that next step j when d is requested can be charged to step i in S. $\square$

Observation 1. *For any reduced schedule, the number of items that are brought in is exactly the number of misses.*

Proof by Greedy Exchange

Step 1: Label. Label your algorithm's solution ($A = \{a_1, a_2, \ldots, a_k\}$), and a general solution ($O = \{o_1, o_2, \ldots, o_m\}$).

Let S_{FF} be the schedule created by the farthest-in-future algorithm and let S be an arbitrary reduced schedule.

Step 2: Compare. Compare greedy with the other solution. Assume that they're not the same and isolate some difference.

Suppose they are the same through the first j items. Then we show in the following lemma, by an exchange argument, that we can modify them to be the same through the first $j+1$ items.

Step 3: Exchange. Swap the elements in in O without making the solution worse. Argue that swapping a finite number of times will result in A. Hence, greedy is just as good as *any* optimal or arbitrary solution.

Lemma 2. *Suppose S is a reduced schedule that makes the same eviction decisions as S_{FF} through the first j items in the sequence for some j. Then there exists a reduced schedule S' that makes the same eviction decisions as S_{FF} through the first $j+1$ items and incurs no more misses in total than S does in total.*

Prove this by constructing S'. *This is an* ***exchange*** *argument.*

a. What happens if the $j+1^{st}$ item is in cache?

Neither evicts (recall that S is reduced). Let $S' = S$.

b. What happens if the $j+1^{st}$ item isn't in cache, but S evicts the same item as S_{FF}?

Let $S' = S$.

c. What happens if the $j+1^{st}$ item isn't in cache, and S evicts a different item as S_{FF}? What should S' do?

Let S' match S_{FF} up to $j+1$, now S' and S's caches differ by one. Go to the next part.

d. How can you get S''s cache back to the same as S's without incurring more total misses?

From now on, S' does the same as S unless

- there's a request to $g \neq e$, f not in cache and S would evict e. Then S' evicts f.
- there's a request to f and S evicts e'. IF $e' \neq e$, then S' evicts e' and brings in e.

Now the caches and misses are equal. We need to transform S' to reduced.

e. How do we know that S' is a reduced schedule?

It may not be, but by Lemma 1, we can transform it to be.

f. Sanity check: Are all parts of the lemma true?

Matches S_{FF} through step $j+1$, same caches and misses as S, is reduced. Yep!

All together, this gives:

Proof. As S and S_{FF} are the same up to this point, they have the same cache. Upon the $j+1^{\text{st}}$ item, either (1) d_{j+1} is in cache, so neither evicts (S is reduced), so $S' = S$. Or, (2) S happens to make the same decision as S_{FF}, so $S' = S$. Or, (3) S evicts some f while S_{FF} evicts some e. Let S' match S_{FF} up to $j+1$. Now S' and S's caches differ by one. From now on, S' does the same as S unless

- there's a request to $g \neq e, f$ not in cache and S would evict e. Then S' evicts f.
- there's a request to f and S evicts e'. IF $e' \neq e$, then S' evicts e' and brings in e.

Now the caches and misses are equal. We need to transform S' to reduced. $\square$

Theorem 2. *S_{FF} incurs no more misses than any other schedule S^* and hence is optimal.*

By induction, for any m and algorithm S, there's a reduced schedule that incurs no more misses but makes the same evictions as S_{FF}.