**DS 320 Algorithms for Data Science** **Lecture #8**
**Spring 2026** Prof. Kira Goldner and Jeffrey Considine

# 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. We assume that the cache is initially empty. Our goal is to give an algorithm that minimizes the number of misses.

| | | |
|---|---|---|
| Example 1: | $a, b, c, b, c, a, b$ | $k = 2$ |
| Example 2: | $a, b, c, d, a, d, e, a, d, b, c$ | $k = 3$ |

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

- If the requested item is already in the cache, nothing is done.
- Otherwise, the missing item is brought into the cache (after possibly evicting another item).

**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* bringing in an item in one schedule to bringing in an item in another schedule to show that it doesn't bring in any more items than the original schedule.

*Proof.* For any schedule $S$, define its reduced schedule $\bar{S}$: if at time $i$, $S$ 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 (say time $j$). Then the work of bringing in $d$ in that next step $j$ of $\bar{S}$ when $d$ is requested can be charged to step $i$ in $S$. □

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

## A Greedy Algorithm for Optimal Caching

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 $A_{FF}$.

## 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 $A_{FF}$ 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.

After $k$ exchanges, $O$ has been transformed into $A$ without increasing the cost.

## The Exchange Step

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

**Lemma 2.** *Suppose $S$ is a reduced schedule that makes the same eviction decisions as $A_{FF}$ 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 $A_{FF}$ 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 $A_{FF}$?

Let $S' = S$.

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

Suppose $S$ evicts some $y$ while $A_{FF}$ evicts some $x$. Let $S'$ match $A_{FF}$ up to $j+1$, and now $S'$ and $S$'s caches differ by one. Suppose $A_{FF}$ evicts some item $x$ and $S$ evicts some item $y$.

**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 $w \neq x$, $y$ not in cache and $S$ would evict $x$. Then $S'$ evicts $y$.
- there's a request to $y$ and $S$ evicts $z$. IF $z \neq x$, then $S'$ evicts $z$ and brings in $x$.

How do we know there's not a request to $x$? Well, $x$ was evicted because it's seen farthest in the future, so certainly $y$ or $w \neq x, y$ is requested before $x$.

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. The schedule construction used to prove Lemma 1 only adjusts the schedule after the first non-reduced step, so it won't affect the initial $j+1$ steps. (Also $A_{FF}$ will always be reduced so the reduction never conflicts there either.)

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

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

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

**Theorem 2.** *$A_{FF}$ incurs no more misses than any other schedule $O$ and hence is optimal.*

*Proof.* We use induction on the following claim: There exists a reduced schedule that makes the same eviction decisions as $A_{FF}$ through the first $i$ items in the sequence incurs no more misses in total than $O$ does in total for all $0 \leq i \leq m$.

**Base case:** At the $i = 0$th step, the schedules vacuously made the same decisions through the first 0 items, and our schedule is identical to $O$.

**Inductive hypothesis:** For some $i$, suppose $S$ is a reduced schedule that makes the same eviction decisions as $A_{FF}$ through the first $i$ items in the sequence and incurs no more misses in total than $O$ does in total.

**Inductive step:** By the inductive hypothesis and Lemma 2, the we can construct $S$ that makes the same eviction decisions as $A_{FF}$ through the first $i+1$ items in the sequence and incurs no more misses in total than $O$ does in total.

**Conclusion:** After $m$ steps, $O$ is identical to $A_{FF}$ and incurs no more misses in total than $O$ does in total, hence $A_{FF}$ must be optimal. □