CS420+500: Advanced Algorithm Design and Analysis

Lectures: March 27 + 29, 2017

Prof. Will Evans *Scribe: Julieta Martinez*

In this lecture we:

- Discussed k-competitiveness of deterministic paging algorithms;
- Randomized marking algorihtms (the randomized marking mouse RMM).

Handouts (posted on webpage):

- No handouts were given in this lecture.

Reading: None. There are plenty of online references for competitive analysis of paging algorithms.

1 Competitive analysis of deterministic paging algorithms

Remember: A paging algorithm is a method that decides which parts of memory ("pages") should be kept in cache at any given point. Last lecture, we learned that worst-case analysis is not very helpful to analyze these algorithms, because we can always come up with an adversarial sequence that will make the algorithms fault on every page request. This motivates the use of *competitive analysis*, a technique that compares and obtains performance bounds with respect to an optimal algorithm, OPT.

Notation

- OPT is an optimal paging algorithm. OPT knows what future page requests will be.
- k is the number of pages that we can keep in cache.
- A c-competitive algorithm is an algorithm that does not fault c times more than OPT.

Theorem 1. *If* A *is a deterministic, c-competitive online algorithm, then $c \geq k$.*

Proof. The idea is to find a sequence that is bad for A and good for OPT.

Assume that A and OPT have the same pages in cache: $1, 2, 3, \ldots, k$.

- Request page $k + 1 = a_1$. A faults and evicts some page a_2.
- Request page $k + 2 = a_2$. A faults and evicts some page a_3.
- Request page $k + 3 = a_3$. A faults and evicts some page a_4.

- ...and so on.

In other words, A faults on every page request when the total number of different pages is at least $k+1$.

Our claim is that OPT faults on request $j+1$, and $j \geq k+1$. The proof is simple; OPT evicts the page p in cache that is requested furthest in the future. Since there are only $k+1$ different pages, the next k page requests can be kept in cache. We now know that

- OPT faults at most once every $k+1$ requests.
- A faults every page request.

From these two premises, it follows that

$$f_{\text{A}}(a_1, a_2, \ldots, a_n) \geq k \cdot f_{\text{OPT}}(a_1, a_2, \ldots, a_n) \tag{1}$$

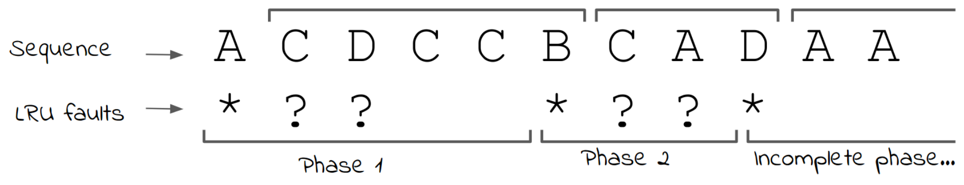
Now, we show that LRU is a k-competitive algorithm.

Theorem 2. LRU *is k-competitive*

Proof. Let $p_1, p_2, \ldots, p_n$ be any sequence of page requests.

We partition this sequence into contiguous subsequences (*aka*, phases), such that LRU faults on the first page of the phase and the phase contains exactly k different pages.

For example, if we let $k = 3$ and observe the following sequence:



It is clear that LRU faults at most k times per phase.

On the other hand, OPT must have the first page of the phase in cache; but, since the remainder of the phase plus the first page of the next phase consists of k different pages, OPT must fault at least once during these requests.

Now we know that

- $f_{\text{LRU}}(p_1, p_2, \ldots, p_n) \leq k \cdot (\#phases)$
- $f_{\text{OPT}}(p_1, p_2, \ldots, p_n) \geq \#phases - 1$

Which implies that LRU is k-competitive.

2 Randomized online algorithms for paging

We analyze the performance of a randomized algorithm for marking, called the randomized marking mouse (RMM). In RMM:

- The mouse hides in one of m hiding places
- Every time step, the cat looks into one of the m places
- if the mouse is there, it must move to a different place

In this analogy,

- The locations where the mouse can hide are m different pages
- The locations where the mouse is not hiding are pages in cache
- A cat probe sequence is a sequence of page requests

It follows from Theorem 1 that a deterministic mouse cannot be better than $(m-1)-$competitive.

We define the *cost* as the number of times the mouse moves.

The optimal algorithm, OPT, moves the minimum number of times as it is possible, assuming it knows where the cat will look in the future.

Now, imagine a *very* systematic cat, that keeps track of its probes.

- The mouse starts at a random location
- The cat first probes a random location
- When the cat probes a location, it marks it
- When the cat probes the mouse's location, the mouse moves to a random *unmarked* spot[1]
- If the mouse is at the last unmarked spot, the cat finds the mouse in the last probe and clears the marks (a new phase begins)

Claim 3. *For all* $p_1, p_2, \ldots, p_n$, $\mathbb{E}\left[\text{RMM}_{cost}(p_1, p_2, \ldots, p_n)\right] \leq O(\log m)\text{OPT}_{cost}(p_1, p_2, \ldots, p_n)$

Proof. Initially, the probability that the mouse is at any spot is $1/m$.
The first cat probe finds the mouse with probability $1/m$. Now, note that whether the mouse is found or not, the mouse is at any of the unmarked spots with probability $1/(m-1)$. Therefore

- In probe #2 the cat finds the mouse with probability $1/(m-1)$
- In probe #3 the cat finds the mouse with probability $1/(m-2)$
- In probe #4 the cat finds the mouse with probability $1/(m-3)$

[1] The mouse could move to a marked spot as well, and make the job of the cat harder. However, the mouse is nice enough to move to a marked spot to help us out with our analysis.

- ... and so on.

It follows that

$$\mathbb{E}\left[\# \, times \, \textsc{rmm} \, is \, found \, during \, a \, phase\right] = \tag{2}$$

$$\mathbb{E}\left[\sum_{i=1}^{m} X_i\right] = \mathbb{E}\left[X_1\right] + \mathbb{E}\left[X_2\right] + \cdots + \mathbb{E}\left[X_m\right] = \tag{3}$$

$$\frac{1}{m} + \frac{1}{m-1} + \frac{1}{m-2} + \cdots + \frac{1}{1} = H_m \simeq \ln m \tag{4}$$

Now, since OPT knows the future, we know that OPT will found at least once during the phase[2]. Thus, our claim is proven.

[2]We know it will be found *exactly* once, but we want to emphasize the performance bound on OPT, in contrast to that of RMM.