CS420/MATH500: Advanced Algorithms

Lecture: March 27 and March 29, 2017

Prof. Will Evans *Scribe: Sikander Randhawa*

In these lectures we:

- Proved that any deterministic online algorithm for paging is $c \geq k$ competitive.
- Proved that LRU is k-competitive, and so is optimal.
- Defined the Randomized Marking Mouse (RMM) algorithm.
- Showed that in expectation, RMM is $\mathcal{O}(\log m)$ competitive.

1 Analysis of Deterministic Page Replacement Policies

In the previous lecture, we remarked that no deterministic page replacement policy could do better than k-competitive with the optimal algorithm which has knowledge about the future. Today we prove that this is true.

Before we begin, let us redefine what it means for an online algorithm to be k competitive, for convenience:

Definition 1. *An online algorithm $\mathcal{A}$ is c competitive if there exists b in $\mathbb{R}_{\geq 0}$ such that for any sequence of requests $p_1, \ldots, p_n$:*

$$f_{\mathcal{A}}(p_1, \ldots, p_n) \leq c \cdot f_{OPT}(p_1, \ldots, p_n) + b,$$

where f_A denotes the "cost" of running an algorithm A on the given sequence and OPT is the optimal algorithm for the problem in an offline setting.

Theorem 1. *If $\mathcal{A}$ is a deterministic online algorithm for paging, then $\mathcal{A}$ is $c \geq k$ competitive, where $k =$ cache-size.*

Proof. Observe that for the problem of page replacement given a sequence of requests, the cost function is number of faults an algorithm incurs on a sequence of requests.

To prove Theorem 1, it suffices to define a sequence for which any arbitrary deterministic paging algorithm, $\mathcal{A}$, incurs k times the number of faults as the furthest in the future algorithm (i.e. the optimal offline algorithm for page replacement).

Loosely speaking, this amounts to coming up with a sequence of page requests which performs poorly using $\mathcal{A}$ but at the same time performs well using OPT.

Assume that $\mathcal{A}$ and OPT have the same pages in the cache. Let us denote these pages as:

$$a_1, \ldots a_k.$$

Let us call a' any page that is not in the cache.

Now, let use construct the following sequence:

$$a'\ b_1\ b_2 \ldots$$

where the b_i's are the page that $\mathcal{A}$ evicted at step $i-1$. That is, b_1 is the page which $\mathcal{A}$ evicted after the request to a', b_2 is the page which $\mathcal{A}$ evicted on the request to b_1, and so on. Observe that the sequence $a', b_1, \ldots$ is contained in $a_1, \ldots, a_k, a'$.

Observation 2. *$\mathcal{A}$ faults on each request in the sequence*

$$a'\ b_1\ b_2 \ldots .$$

This is true because we repeatedly request the last page that we evicted. Obviously OPT must also fault on a' since a' is not in the cache when the request to a' was made. Let j be such that the request to b_j is the next fault that OPT makes.

Claim 3. $j \geq k$.

Proof. After a' is brought into the cache, OPT can ensure that the next $k-1$ pages in the sequence remain in the cache, because even if they were all different from a' they were all in the cache before a' was brought in. So, OPT could have simply evicted a page not in the next $k-1$ requests. □

So, OPT faults at most once per $k+1$ requests while $\mathcal{A}$ faults on each request. Hence,

$$f_{\mathcal{A}}(p_1, \ldots, p_n) \geq k f_{OPT}(p_1, \ldots, p_n).$$

□

Theorem 4. *LRU is k-competitive.*

Proof. To prove this, we must demonstrate that for any sequence of page-requests that the number of faults which LRU incurs is at most k the number of the furthest in the future algorithm (plus some constant).

Let $p_1, \ldots, p_n$ be any sequence of page requests. We partition this sequence into contiguous blocks, called phases, as follows:

- LRU faults on the first page of the phase.
- The page contains exactly k different pages.
- The phase is of maximal size.

Example 1. *Page Requests: ACDCCBCADAA, with $k = 3$.*

In the above example, ACDCC is phase 1, BCA is phase 2, and DAA... is phase 3.

Since each phase has k distinct pages, then LRU can fault at most k times per phase.

Now, we consider the performance of OPT. After the request for the first page of a phase, OPT must have that page in the cache. The remainder of the phase plus the first page of the next phase consists of k distinct pages. Therefore, OPT must fault on one of these pages.

It follows that we have:

$$\begin{cases} f_{LRU}(p_1, \ldots, p_n) \leq k \cdot \text{number of phases} \\ f_{OPT}(p_1, \ldots, p_n) \geq \text{number of phases} - 1. \end{cases}$$

Hence,

$$f_{LRU}(p_1, \ldots, p_n) \leq k \cdot f_{OPT}(p_1, \ldots, p_n) + 1$$

□

2 Randomized Online Algorithms

Here we present a new problem, in terms of a cat and mouse, and then we will reformulate this problem in terms of the page replacement problem we've already been discussing.

Cat and Mouse Problem

- Mouse hides in one of m hiding spots.
- Every time step, cat looks in one of the m places.
- If the mouse is there, the mouse must move to a different hiding place.

We relate this to page replacement as follows:

- **Cost** = number of mouse moves.
- **OPT** = minimum number of mouse moves.
- **m-1** = size of the cache.
- **m** pages.
- **Cat probes** = page request sequence.
- **Probe resulting in mouse move** = page fault.

Example 2. *Let the $m = 4$, and let $i \in \{1, 2, 3, 4\}$ denote the i-th hiding spot.*

In the above example $OPT(12341234) = 2$, and $OPT(12113412) = 1$.

Why? In the first example, since the cat checks each hiding spot twice by probing the hiding spots in consecutive order, the mouse must move once over the first probing of the four hiding spots, and again once over the second probing of the four hiding spots. In the second example, each spot is probed at least once, so the mouse will have to move at least once.

From our result above about deterministic page replacement policies, we know that a "deterministic" mouse can do no better that $(m-1)$-competitive.

We now present a randomized algorithm for this cat and mouse problem:

Randomized Marking Mouse

- Start at a random location.
- When cat probes a location, mark it.
- When cat probes mouses location, mouse moves to a random unmarked spot.
- If mouse is at last unmarked spot, clear marks [new phase begins].

Since we know we can really view the cat and mouse problem as a page replacement problem, then if we can show that the above algorithm achieves a competitive ratio better than $(m-1)$ then we will have found a better algorithm for page replacement.

Theorem 5. *For any sequences of page requests,* $p_1, \ldots, p_n$*,*

$$\mathbb{E}[RMM_{cost}(p_1 p_2 \ldots p_n)] \leq \mathcal{O}(\log m) \cdot OPT(p_1 p_2 \ldots p_n).$$

Proof. Each phase is of length m. We would like to demonstrate for any phase,

$$\mathbb{E}[\#\text{ of times mouse is found using RMM}] \leq \mathcal{O}(\log n)(\#\text{ of times mouse is found using OPT}).$$

If the mouse is "randomized", then the cat finds the mouse after the first probe with probability $\frac{1}{m}$. The mouse won't be in this location again, because it's been marked. So, on the next probe, the cat finds the mouse with probability $\frac{1}{m-1}$. Similarly, on probe i, the cat finds the mouse with probability $\frac{1}{m-i+1}$.

Let

$$X_i = \begin{cases} 0 & \text{if mouse not found on probe i} \\ 1 & \text{if mouse is found on probe i.} \end{cases}$$

Therefore,

$$\mathbb{E}[\#\text{ times RMM is found in a phase}] = \mathbb{E}[\sum_{i=1}^{m} X_i = \sum_{i=1}^{m} \mathbb{E}[X_i] = \sum_{i=1}^{m} \frac{1}{i} = \mathcal{O}\log(m).$$

Furthermore, the optimal mouse is found at least once per phase because each phase consists of probes to all possible locations. So, for any phase, we have

$$\mathbb{E}[\#\text{ of times mouse is found using RMM}] \leq \mathcal{O}(\log n)(\#\text{ of times mouse is found using OPT}),$$

as desired and our result follows. □