CS420+500: Advanced Algorithm Design and Analysis

Lectures: March 29th + March 31st, 2017

Prof. Will Evans *Scribe: Andrew Adair Comminos*

— 2017-03-29 —

A Randomized Online Algorithm for Paging

Last lecture, we set up a cat/mouse scenario where the locations the mouse can hide are equivalent to different pages. The location the mouse is hiding is the page that is missing from the cache, and all other locations are in the cache. The cat's probe sequence is analagous to the page request sequence.

The deterministic mouse can be no better than $(m-1)$-competitive. The **random marking mouse (RMM)** algorithm, however, has no such bound.

How does RMM behave?

- Start at a random location.
- Wait, watching the cat.
  - When the cat probes a spot, the mouse marks the spot.
  - When the cat probes the mouse's location, the mouse *moves to a **random** unmarked spot.*
- If the mouse is now at the last unmarked spot, then it clears all marks.
  - (This is when a new phase begins.)

Claim: $E[\text{RMM}_{\text{cost}}(p_1 p_2 \dots p_n)] \leq O(\log m)\text{OPT}(p_1 p_2 \dots p_n)$

- (for all $p_1 p_2 \dots p_n$,where E is expected cost).

Proof: First, how might we go about measuring the expected cost of a cat probing?

- Initially, the probability that the mouse is at any spot is $\frac{1}{m}$.
- The first cat probe thus finds the mouse with probability $\frac{1}{m}$.
- Irrespective of whether the mouse is found or not, after the first probe the mouse is at one of the other $m-1$ spots with equal probability.
- We assume that the cat is smart- they won't probe the same spot twice.
  - This models the worst case for our marking scheme.
- The probability of the next probe finding the mouse is thus $\frac{1}{m-1}$.
- Again, irrespective of the outcome, the next probe will find the mouse with probability $\frac{1}{m-2}$.
  - This is because the mouse does not returned to spots that it has marked. Recall our "smart cat" assumption.
- This continues with subsequent probes in the same fashion.

Before continuing, we need to talk about **indicator random variables**.

- Let $X_i = \begin{cases} 0 & \text{if mouse not found on probe i} \\ 1 & \text{if mouse found on probe i} \end{cases}$
- This is an indicator random variable- it has value 1 if an event occurs, and a value 0 if an event does not occur.

Let us apply this concept to our problem as follows;

$$\begin{aligned}
&E[\#\text{ times RMM is found}] \\
&= E[\sum_i X_i] \\
&= E[X_1] + E[X_2] + \cdots + E[X_m] && \text{(by linearity of expectation)} \\
&= \frac{1}{m} + \frac{1}{m-1} + \frac{1}{m-2} + \cdots + \frac{1}{1} \\
&= H_m \approxeq \ln m && \text{(by sum of harmonic sequence)}
\end{aligned}$$

Since every spot is probed by the end of this sequence of probes (called a phase), the optimal mouse OPT must move at least once.

— 2017-03-31 —

Last time, we determined that for all page requests, the expected cost of the RMM algorithm was not that bad:

$$E[\text{RMM}(p_1 p_2 \dots p_n)] \leq O(\log m) \cdot \text{OPT}(p_1 p_2 \dots p_n)$$

Claim: For all mice A, deterministic or randomized, there exists a sequence of probes $p_1 p_2 \dots p_n$ such that:

$$E[A_{\text{COST}}(p_1 p_2 \dots p_n)] > (\log m) \cdot \text{OPT}(p_1 p_2 \dots p_n)$$

Proof: We need to prove the existence of a sequence of page requests that will make *any* mouse "look bad" compared to the optimal algorithm- it will cause A to move a factor of $\log m$ more frequently than OPT.

This is a proof of the existence of some sequence without actually providing the sequence.

If cat probes at random, then no matter what mouse A does, cat finds it with probability $\frac{1}{m}$. The expected # of times A must move over a sequence of t probes is thus $\frac{t}{m}$.

How many random cat probes until cat examines all m spots?

(this is called the coupon collector problem, as you're trying to collect all k of the breakfast cereal toys where each box has one of k toys with uniform probability)

Reducing this to the coupon collector problem implies that the solution to this problem involves $m \ln m$ probes.

Thus, **the optimal mouse (opt) moves only once every $m \ln m$ steps, while A moves $\frac{m \ln m}{m}$ times.**

Therefore, A faults $\geq (\ln m) \cdot \text{OPT}$.

Hash Functions

Universal Hash Functions:

- Fixed hash functions are a bad idea. Denial of service attacks may be performed by selecting a sequence with terrible, terrible collisions.
- Yet, we need to pick a fixed function- otherwise, we can't really use the hash.
- A **universal hash function** is a set of hash functions that will be used to pick a hash function from.

Formally, a set of hash functions H that map keys $U \to \{0, 1, \dots, m-1\}$ is **universal** if for all distinct keys $k, \ell \in U$, the number of hash functions $h \in H$ such that $h(k) = h(\ell)$ is small: at most $\frac{|H|}{m}$, where m is the size of the hash table.

- What does this mean? By choosing a hash function at random from a universal set of hash functions, we make it difficult for an attacker to guess what hash function to target for a denial-of-service attack. In addition, the property of the universal set means that the probability (over our choice of function) that two items collide is as small as we could hope.

We now shift to **chaining using universal hash functions**.

- Recall that chaining is used to probe in the presence of a collision by application of the hash function again.

Suppose we're trying to hash n keys into a table of size m.

$$\alpha = \frac{n}{m} = \text{load factor}$$

Theorem: Using a hash function $h \in_R H$ ($\in_R$ denotes chosen uniformly at random), for a key k:

$$E[n_{h(k)}] \leq \begin{cases} \alpha + 1 & \text{if } k \in T \\ \alpha & \text{if } k \notin T \end{cases}$$

Where n_i is the # of items in bucket i.

(recall $E[X] = \sum_y \text{Prob}[X = y] \cdot y$)

Proof:

Let $X_{k\ell} = 1$ if $h(k) = h(l)$, or zero otherwise.

Let $Y_k = \sum_{\ell \neq k, \ell \in U} X_{k\ell}$.

To be continued next lecture...