Advanced Algorithm Design: Marking Algorithm

Lectured by Prof. Moses Charikar
Transcribed by Borislav Hristov*

February 20th, 2013

1 Marking Algorithm

At the end of the last lecture we briefly sketched a probabilistic algorithm for caching which is $\log k$ comparative. Here is the algorithm in details; we assume there is an extra bit per every page which we will use as indicator if the page has been recently requested.

- initially all pages are unmarked
- when page *p* is requested if *p* is in the cache mark it (set the bit);otherwise:
- if $\nexists$ unmarked page, then unmark all pages
- evict random unmarked page
- bring *p* into cache and mark it

Let's examine more closely the *"unmark all pages"* step. We can divide the request sequence σ into phases based on when this unmarking happens. The idea is similar to the one we used in last class to divide the requested sequence for the deterministic algorithm into groups of k distinct pages. Here, we make the following

Claim 1. Deterministic marking algorithm is $k-$comparative.

The proof is pretty straightforward. For all pages to be marked at the end of the phase it means that we have k distinct pages in the cache, all marked, and now the first element x of the next phase is a different one and we have to unmark it (deterministically). But that page x is certainly

*borislav@cs.princeton.edu

a miss for the optimal algorithm as well. Therefore, any deterministic marking algorithm is k-comperative.

However, this is a very crude bound. If we use some probabilistic reasoning for the ramdomized algortihm we can derive a better bound.

Claim 2. The marking algorithm is $\log k-$comparative.

Let's consider the union of two consecutive phases $i-1$ and i and denote with m_i the number of misses the optimal algorithm makes in those two phases. Therefore, we have the following two lower bounds for OPT:

$\Omega(OPT) \geq m_2 + m_4 + m_6 + ... + m_{2r}$
$\Omega(OPT) \geq m_3 + m_5 + m_7 + ... + m_{2r+1}$ which combined give us
$2\Omega(OPT) \geq \sum_i m_i$

Now, consider phase i, phase $i-1$ just happened. The worst case scenario for the probabilistic algorithm is if all m_i evicts happen at the beginning of phase i and they evict pages we will request later. Let x_1 be the first distinct page we request after the m_i evicts. The probability that x_1 is in the cache is $\frac{k-m_i}{k}$ because we evicted randomly m_i pages and x_1 could have been one of them. Thus, the probability of faulting on the request for x_1 is $\frac{m_i}{k}$. Similarly, the probability of faulting on the request for the next distinct page x_2 ($x_1 \neq x_2$) is $\frac{m_i}{k-1}$ and so on. Thus, for phase i

$$E[\#faults] = m_i + \frac{m_i}{k} + \frac{m_i}{k-1} + \frac{m_i}{k-2} + ... + \frac{m_i}{k-m_i} \leq m_i(1 + \frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{k-m_i}) \leq m_i H_k$$

$$E[total\#faults] \leq \sum_i m_i H_k \leq 2\Omega(OPT) H_k$$

Therefore the marking algorithm is $2H_k-$comparative.

2 Yao's principle

Let's discuss a general mechanism to prove lower bounds for randomized algorithms against oblivious adversaries known as Yao's principle. Let R be a randomized algorithm and C_r its comparative ratio against oblivious adversaries. Let P be a probability distribution over an input (request sequence) σ and let C_A^P be the expected comparative ratio of a deterministic algorithm A on distribution P. Then, Yao's principle states that:

$$\inf_R C_R = \sup_P \inf_A C_A^P$$

The right side means to take the worst possible distribution and have the best possible deterministic algorithm do it. From the above we can also derive:

$$\inf_R C_R \geq \inf_A C_A^P$$

Now, let's use this principle to analyze the lower bound for the marking algorithm. Construct σ at random, uniformly from $\{1, 2, ..., k+1\}$ where k is the size of the cache. We claim that this distribution of $k+1$ pages is bad and any deterministic algorithm looks bad on it. Observe that the

probability of faulting on the $i-th$ request is $\frac{1}{k+1}$. Thus, $E[\#faults] = \frac{\|\sigma\|}{k+1}$. Suppose we have seen i pages. The probability of seeing a new page is exactly $Pr[new\,page] = \frac{k+1-i}{k+1}$ and therefore the number of steps to see the i-th new page is:
$E[\#steps\ to\ see\ ith\ new\ page] = 1/Pr[new\ page] = \frac{k+1}{k+1-i}$
Therefore:
$E[\#steps\ to\ see\ all\ pages] = \sum_i \frac{k+1}{k+1-i} = (k+1)H_{k+1} = \Theta(k \log k)$ (Note: this is how you do the coupon collector problem)

So, we have $E[OPT] \leq O(\frac{\|\sigma\|}{k \log k})$ and $E[\#faults\ any\ det\ alg] = \frac{\|\sigma\|}{k+1}$ which give us a lower bound of $\Omega(\log k)$.

3 Experts' advice

Now, in the second part of the lecture, we will consider a different problem. Let n be the number of experts predicting whether a given stock will go up or down. The goal is to use their input and construct an algorithm so that we minimize the difference in the number of mistakes we do compared to the best performing expert. Here is one algorithm which uses weighted majority:

- initialize all experts' weights to 1
- given prediction $x_i \in (0, 1)$ choose the weighted majority, that is:

  if $\sum_{i\, x_i=1} w_i > \sum_{i\, x_i=0} w_i$ then choose 1; otherwise choose 0.
- if an expert is wrong, reduce its weight by a half, i.e:
  if the correct answer is l and $x_i \neq l$ then $w_i \leftarrow w_i/2$

Analysis. Suppose our algorithm makes M mistakes and the best expert makes m. Let w be the sum of all weights. Initially $w = n$. When we make a mistake it must be the case that at least half of the mass was on the wrong side. But every expert on the wrong side will have its weight halved so the total weight will decrease by a factor of at least $\frac{3}{4}$. Thus, at the end the final weight will be $\leq n(\frac{3}{4})^M$. The best expert's weight at the end will be $(\frac{1}{2})^m$ because he made m mistakes. So, we have:

$(\frac{1}{2})^m \leq n(\frac{3}{4})^M \Leftrightarrow n2^n \geq (\frac{4}{3})^M \Leftrightarrow log_2 n + m \geq M log_2 \frac{4}{3}$ which gives us:

$M \leq 2.4(m + log_2 n)$

It is true that this resembles a learning problem and perhaps different learning approaches may yield better performance. But the paradigm will be useful in other settings, i.e when "experts" are replaced by "algorithms". In next class, we will analyze a different algorithm, one in which we take the advice of one expert chosen probabilistically and proportionally to its weight.