CS 224 Advanced Algorithms — Fall 2014

Problem Set 2

Due: 11:59pm, Wednesday, September 24th
Submit to: `cs224-f14-assignments@seas.harvard.edu`
Solution maximum page limit: 5 pages

See homework policy at `http://people.seas.harvard.edu/~minilek/cs224/hmwk.html`

Problem 1: (20 points) Suppose we have random variables $\delta_1, \ldots, \delta_n$, independent and each either 0 or 1 with $\mathbb{E}\,\delta_i = p$. Write $X = \sum_{i=1}^n \delta_i$ and $\mu = \mathbb{E}\,X = np$. The Chernoff bound states that for any $\varepsilon > 0$

$$\mathbb{P}(X > (1+\varepsilon)\mu) < \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{\mu} \tag{1}$$

and for $0 < \varepsilon < 1$

$$\mathbb{P}(X < (1-\varepsilon)\mu) < \left(\frac{e^{-\varepsilon}}{(1-\varepsilon)^{1-\varepsilon}}\right)^{\mu} \tag{2}$$

(a) (3 points) Show that for any $0 < \varepsilon < 1/2$, Eqs. (1) and (2) imply that

$$\mathbb{P}(|X-\mu| > \varepsilon\mu) < 2e^{-C\varepsilon^2\mu} \tag{3}$$

for some universal constant $C > 0$. **Hint:** Taylor's theorem.

(b) (12 points) One could also argue via the moment method. By Markov's inequality,

$$\mathbb{P}(|X-\mu| > \varepsilon\mu) = \mathbb{P}((X-\mu)^{2\ell} > (\varepsilon\mu)^{2\ell}) < \left(\frac{1}{\varepsilon\mu}\right)^{2\ell} \cdot \mathbb{E}(X-\mu)^{2\ell} \tag{4}$$

for any positive integer ℓ. Bound $\mathbb{E}(X-\mu)^{2\ell}$ for a specific ℓ, which when plugged into Eqn. (4), yields Eq. (3). You may use without proof that for any positive integer m,

$$\left(\frac{m}{e}\right)^m < m! \leq m^m$$

(c) (5 points) How would you modify your argument from part (b) to handle the case when not all the $\mathbb{E}\,\delta_i$ are the same? That is, $\delta_i \in \{0,1\}$, but $\mathbb{E}\,\delta_i = p_i$.

Problem 2: (20 points) In *simple tabulation hashing*, we write a key $x \in [u]$ in base $u^{1/c}$ (assume $u^{1/c}$ is an integer and power of 2) so that $x = \sum_{i=0}^{c-1} x_i \cdot u^{i/c}$. We call the x_i "characters". We then allocate c completely random lookup tables $H_0, \ldots, H_{c-1}$ each of size $[u^{1/c}]$. Then for each $y \in [u^{1/c}]$ we set $H_i(y)$ uniformly at random from $[m]$ (assume also m is a power of 2). Then to hash x, we define (where $\oplus$ denotes bitwise XOR)

$$h(x) = H_0(x_0) \oplus H_1(x_1) \oplus \cdots \oplus H_{c-1}(x_{c-1}).$$

(a) (5 points) Fix $c, m \geq 1$. Show that the family $\mathcal{H}$ of all such hash functions is 3-wise independent.

(b) (5 points) Show that $\mathcal{H}$ is not 4-wise independent.

(c) (10 points) Imagine using such an $\mathcal{H}$ to implement hashing with chaining, as in Lecture 3. Show that if $m > n^{1.01}$ then with probability at least 2/3, every linked list in the hash table has size $O(1)$. **Hint:** show that if a subset T of the n items is "large", then $\mathcal{H}$ behaves completely randomly on some "somewhat large" subset T' of T.

Problem 3: (10 points) A problem that was encountered by the Altavista search engine (and still arises today) is that of near-duplicate document detection in information retrieval. When a user issues a query several web pages might seem to be good matches, however many of those pages might be (nearly) mirrors of some primary source. Search engines for obvious reasons don't then want to flood you with a page of search results all mirroring the same content. They thus wanted a quick-and-dirty way to compare two web pages for similarity.

Altavista's solution, invented by Broder [1], worked as follows. Treat each web page A as a "bag of words", that is, a set of elements in some dictionary D (those elements are simply the words that appear in the document). Given two documents (sets) A and B, define the *Jaccard similarity coefficient* $J(A, B)$ as $|A \cap B|/|A \cup B|$. Notice this is 0 if they contain disjoint words, and 1 if they contain exactly the same set of words. Computing $J(A, B)$ between all pairs of search result pages A, B could be slow. Also storing $J(A, B)$ for every pair of web pages on the Internet would take too much space (the square of the number of web pages on the Internet!). Let $|D| = n$. Broder proposed picking a uniformly random permutation $\pi : [n] \to [n]$ and, for each page A, storing $h(A) = \min_{x \in A} \pi(x)$. It is easy to then see $J(A, B) = \mathbb{P}(h(A) = h(B))$. The storage overhead of storing $h(A)$ for each A is quite small, since it is one extra word, whereas we are storing the entire web page already.

(a) (5 points) Suppose in pre-processing we pick k such random permutations pairwise independently, $\pi_1, \ldots, \pi_k$. That is to say, if s_i is the binary representation of π_i as a bitstring of length $t = \lceil \lg(n!) \rceil$, we can view s_i as an integer in $[2^t]$. Then $s_i = h(i)$ for h drawn from a 2-wise independent family mapping $[n]$ to $[2^t]$. We then store $h_i(A) = \min_{x \in A} \pi_i(A)$ for every web page A and for each $1 \leq i \leq k$. Given this setup, devise a scheme that, given pages A, B, produces a value Z such that

$$\mathbb{P}(|Z - J(A, B)| > \varepsilon) < 1/3. \tag{5}$$

The smaller $k = k(\varepsilon)$ your scheme requires, the better.

(b) (5 points) Suppose we want high probability of success and not just 1/3. That is, for $0 < \delta < 1$ we want $\mathbb{P}(|Z - J(A, B)| > \varepsilon) < \delta$. What must k be in your scheme from part (a) to guarantee this? What if the s_i were fully independent? Now imagine a different scheme: let $Z_1, Z_2, \ldots, Z_t$ be independent estimators each satisfying Eq. (5). Show that for some $t = O(\log(1/\delta))$,

$$\mathbb{P}(|(\underset{1 \leq i \leq t}{median}\, Z_i) - J(A, B)| > \varepsilon) < \delta.$$

If you want to read more about how to relax the assumption that π is a totally random permutation, see [2, 3, 4] for definitions and constructions of "min-wise hash families".

Problem 4: (1 point) How much time did you spend on this problem set? If you can remember the breakdown, please report this per problem. (sum of time spent solving problem and typing up your solution)

References

[1] Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of SEQUENCES*, pages 21–29, 1997.

[2] Andrei Z. Broder, Moses Charikar, Michael Mitzenmacher. A derandomization using min-wise independent permutations. *J. Discrete Algorithms* 1(1), pages 11–20, 2003.

[3] Piotr Indyk. A Small Approximately Min-Wise Independent Family of Hash Functions. *J. Algorithms* 38(1), pages 84–90, 2001.

[4] Mihai Pătraşcu, Mikkel Thorup. The Power of Simple Tabulation Hashing. *J. ACM* 59(3): 14, 2012.