Fall 2014, CS 598: Algorithms for Big Data

Homework 1
Due: 09/11/2013

Instructions and Policy: Each student should write up their own solutions independently. You need to indicate the names of the people you discussed a problem with; ideally you should discuss with no more than two other people.

Solve as many problems as you can.

You need to typeset your solutions in Latex. Please write clearly and concisely - clarity and brevity will be rewarded. Refer to known facts as necessary. Your job is to convince me that you know the solution, as quickly as possible.

Problem 1 In the first lecture we analyzed the following counter algorithm which counted up to n using much less than $O(\log n)$ bits of space: initialize a counter X to 1, and for every increment instruction, increment X with probability $1/2^X$. By averaging many such estimators, we obtained a $(1+\epsilon)$-approximation to n with good probability. Here we will investigate a different way to obtain a good approximation. Imagine we still initialize X to 1, but we increment it with probability $1/(1+a)^X$ instead. (Note: our estimator for n would have to change from $2^X - 1$ to something else; figure out what!)

Question: How small must a be so that our estimate $\hat{n}$ of n satisfies $|n - \hat{n}| \leq \epsilon n$ with at least $9/10$ probability when we return the output of a single estimator instead of averaging many estimators as in class? Also derive a bound $S = S(n)$ on the space (in bits) so that this algorithm uses at most S space with at least $9/10$ probability by the end of the n increments.

Problem 2 In class we saw that there is no deterministic algorithm using less than n bits of space can count the the number of distinct elements in a stream *exactly* where the universe of elements is from $\{1, 2, \ldots, n\}$. See Jelani Nelson's notes on distinct elements (linked from the course web page) for an argument which shows that any deterministic algorithm with $o(n)$ bits cannot approximate the number of distinct elements to within a factor of 2. Show that there is no randomized algorithm for the distinct elements problem using $o(n)$ bits of space which solves the problem *exactly* with at least $99/100$ probability on any input. Thus, any $o(n)$ space algorithm must be *both* randomized *and* approximate. *Hint:* Using Yao's minmax principle, it is sufficient to describe a distribution over inputs such that any *deterministic* algorithm that uses $o(n)$ space has an error of at least $1/100$ on that distribution (the algorithm can make use of the knowledge of the distribution).

Problem 3 This problem is on pairwise independence.

- Let $X_1, X_2, \ldots, X_k$ be independent binary random variables where $\Pr[X_i = 0] = \Pr[X_i = 1] = 1/2$. Given $S \subseteq \{1, 2, \ldots, k\}, S \neq \emptyset$ define a random variables $Y_S = \oplus_{i \in S} X_i$. Prove that the collection $\{Y_S \mid S \subseteq \{1, 2, \ldots, k\}, S \neq \emptyset\}$ is a pairwise independent family of random variables.

- Suppose $\mathcal{H}$ is a 2-universal family from $[n]$ to $[n^3]$. Prove that the probability that a random h from $\mathcal{H}$ is injective is at least $(1 - 1/n)$.