Homework 1

Algorithms for Big Data: CS498 ABD/ABG, Spring 2020
Due: Wednesday at 10pm, 16th September 2020

Instructions and Policy:

- Each homework can be done in a group of size at most two. Only one homework needs to be submitted per group. However, we recommend that each of you think about the problems on your own first.
- Homework needs to be submitted in pdf format on Gradescope. See `https://courses.engr.illinois.edu/cs374/fa2018/hw-policies.html` for more detailed instructions on Gradescope submissions.
- Follow academic integrity policies as laid out in student code. You can consult sources but cite all of them including discussions with other classmates. Write in your own words. See the site mentioned in the preceding item for more detailed policies.

Problem 1. Reservoir sampling.

- Suppose we used reservoir sampling on a stream σ_1 of length m_1 and stored a k-sample (with replacement) S_1. We also used reservoir sampling on another stream σ_2 of length m_2 and stored a k-sample S_2. Show how you can obtain a k-sample (with replacement) S for the stream $\sigma = \sigma_1 \cdot \sigma_2$ (the concatenated stream) from S_1 and S_2 and knowing the values m_1, m_2. How would you generalize if you had h streams $\sigma_1, \sigma_2, \ldots, \sigma_h$ of lengths $m_1, m_2, \ldots, m_h$ and associated k-samples $S_1, S_2, \ldots, S_h$?
- Now suppose you want to do weighted sampling without replacement in the same setting as above. Recall from lecture notes the scheme where the algorithm picked a random θ value for each item and stored the k items with the largest values of $\theta^{1/w}$ seen in the stream. Show how this scheme can be used to obtain a sample from S_1 and S_2. You need to point out what needs to be stored along with the sample.

The idea of this problem is to review reservoir sampling and point out the notion of a small "sketch" of a stream that can be stored and combined with sketches of other streams to create a "sketch" of the concatenated stream.

Problem 2. Sampling, Chebyshev vs Chernoff. Suppose you want to estimate the average of n numbers via sampling, for example the average wealth of people in a town. The average can be very skewed by outliers — perhaps there are a few billionaires that will not make it to the sample but will clearly affect the average. However, we can obtain an accurate estimate if we assume that the numbers are within some limited range. Assume the input numbers $z_1, z_2, \ldots, z_n$ are from $[a, b]$ where $a, b \in \mathbb{R}$ with $a \leq b$. Suppose you sample k input numbers (with replacement) and output their average as the estimate for the true average $\alpha = (\sum_i z_i)/n$. Let X be the random variable denoting the output value.

- Using Chebyshev's inequality, show that for $k \geq \frac{(b-a)^2}{\delta \epsilon^2}$, we have

$$\mathrm{P}[|X - \alpha| \geq \epsilon] \leq \delta.$$

- Using the Chernoff inequality, show that there exists a constant $c > 0$ such that for $k \geq \frac{c(b-a)^2 \log(2/\delta)}{\epsilon^2}$, we have

$$\mathrm{P}[|X - \alpha| \geq \epsilon] \leq \delta.$$

Problem 3. Quick Sort. Given an array A of n numbers (which we assume are distinct for simplicity), the algorithm picks a pivot x uniformly at random from A and computes the rank of x. If the rank of x is between $n/4$ and $3n/4$ (call such a pivot a good pivot), it behaves like the normal QuickSort in partitioning the array A and recursing on both sides. If the rank of x does not satisfy the desired property (the pivot picked is not good), the algorithm simply repeats the process of picking a pivot until it finds a good one. Note that in principle the algorithm may never terminate!

- Write a formal description of the algorithm.
- Prove that the expected run time of this algorithm is $O(n \log n)$ on an array on n numbers.
- Prove that the algorithm terminates in $O(n \log n)$ time with high probability.

Problem 4. Pairwise Independence. Suppose we want to generate N pairwise independent random variables in the range $\{0, 1, 2, \ldots, M-1\}$. We will assume that N and M are powers of 2 and let $N = \{0,1\}^n$ and $M = \{0,1\}^m$ (hence $n = \log N$ and $m = \log M$). We saw a scheme in the lecture using mn bits. Here we will revisit that scheme in a different way and then see how it can be made more randomness-efficient.

Pick a uniformly random matrix $A \in \{0,1\}^{m \times n}$ and a random vector $b \in \{0,1\}^m$. Then for a vector $v \in \{0,1\}^n$, set $X_v = Av + b \mod 2$ (by this we mean component-wise mod 2).

- Suppose we pick A and b uniformly at random. Show that under this scheme, for all $w \in \{0,1\}^n$ where $w \neq \vec{0}$ and for all $\gamma \in \{0,1\}^m$,

$$\mathrm{P}_A[Aw = \gamma \bmod 2] = \frac{1}{2^m}.$$

  Why does this guarantee that X_u and X_v are independent for $u \neq v$ and $u \neq \vec{0}, v \neq \vec{0}$?

- We can make the following improvement. A matrix A is *Toeplitz* if the entries along each diagonal are constant, i.e., $A_{i,j} = A_{i+1,j+1}$. Suppose we choose A uniformly at random from the set of Toeplitz matrices whose entries are in $\{0,1\}$. Show that under this scheme, it is also the case that for all $w \in \{0,1\}^n$ where $w \neq \vec{0}$ and for all $\gamma \in \{0,1\}^m$,

$$\mathrm{P}_A[Aw = \gamma \bmod 2] = \frac{1}{2^m}.$$

- How many random bits do we need to generate a random Toeplitz matrix $A \in \{0,1\}^{m \times n}$ and how much storage (in bits) do you need to store it implicitly? Express this as a function of N and M and compare with the number of random bits needed for the case when A is picked as a random $\{0,1\}$ matrix.

Problem 5. Probabilistic counter. In lecture we analyzed probabilistic counting: 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 and space usage was $O(\log \log n)$. In this problem you will investigate a minor modification. Imagine we still initialize X to 1, but we increment it with probability $1/(1+a)^X$ for some fixed $a > 0$. (Note that your estimator for n would have to change from $2^X - 1$ to something else.)

How small must a be so that our estimate $\tilde{n}$ of n satisfies $|\tilde{n} - n| \leq \epsilon n$ with at least 9/10 probability when we return the output of a single estimator instead of averaging many estimators we did in the lecture? 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.

Additional exercises (not to be submitted)

Problem 6. In class, we proved a powerful tail inequality called the "(multiplicative) Chernoff bound" that we will use time and time again. In this exercise, we rewrite the Chernoff inequality in a convenient form that is a little more interpretable and easier to apply.

Recall the Chernoff inequality, as follows. Let $X_1, X_2, \ldots, X_n \in [0,1]$ be n independent, non-negative, and uniformly bounded random variables. Let

$$\mu = \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{E}[X_i]$$

be the expected value of the sum. The Chernoff inequality states that for any $\delta > 0$, we have

$$\mathrm{P}[\textstyle\sum_{i=1}^{n} X_i \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \text{ and } \mathrm{P}[\textstyle\sum_{i=1}^{n} X_i \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu},$$

where for the second inequality we also further assume $\delta < 1$.

Now let $X_1, \ldots, X_n$ and μ be as above.

- Show that for $x \geq 0$ sufficiently small, we have

$$x - (1+x)\ln(1+x) \leq -\frac{x^2}{3}.$$

  Hint: Consider the Taylor expansion $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \cdots$ *for* $x \in (-1, 1]$.

- Show that for $\epsilon \in [0,1]$,

$$\mathrm{P}\left[\sum_{i=1}^{n} X_i \geq (1+\epsilon)\mu\right] \leq e^{-\epsilon^2 \mu/3}.$$

- Show that for $x \in [0,1]$, we have

$$x + (1-x)\ln(1-x) \geq \frac{x^2}{2}.$$

- Show that for $\epsilon \in [0,1]$,

$$\mathrm{P}\left[\sum_{i=1}^{n} X_i \leq (1-\epsilon)\mu\right] \leq e^{-\epsilon^2\mu/2}.$$

Problem 7. Exercises 2 and 3 from HW 4 of the 2016 algorithms course (`https://courses.engr.illinois.edu/cs473/fa2016/Homework/hw4.pdf`)