Homework 1 (due 9/22)

DS-563 / CD-543 @ Boston University

Fall 2021

Before you start…

Collaboration policy:[1] You may verbally collaborate on required homework problems, however, you must write your solutions independently. If you choose to collaborate on a problem, you are allowed to discuss it with at most 4 other students currently enrolled in the class.

The header of each assignment you submit must include the field "Collaborators:" with the names of the students with whom you have had discussions concerning your solutions. A failure to list collaborators may result in credit deduction.

You may use external resources such as textbooks, lecture notes, and videos to supplement your general understanding of the course topics. You may use references such as books and online resources for well known facts. However, you must always cite the source.

You may **not** look up answers to a homework assignment in the published literature or on the web. You may **not** share written work with anyone else.

Submitting: Solutions should be submitted via Gradescope (entry code: BPDKV8). You are allowed to submit your solutions both in handwriting or typed. If you decide to hand-write your solutions, make sure they are as readable as possible. If you decide to submit a typed version, we suggest using LaTeX.

Grading: Whenever we ask for an algorithm (or bound), you may receive partial credit if the algorithm is not sufficiently efficient (or the bound is not sufficiently tight).

Questions

1. Read the handout about useful probabilistic inequalities on the course webpage. Which one is your favorite and why?

2. In class, we saw how the probability of getting a good estimate can be improved if an algorithm can provide an estimate that is too high with some probability, but never provides an estimate that is too low. (This was done by running multiple independent copies of the algorithm and returning the minimum of their estimates.) What can you do if your algorithm can err by providing an estimate that is too high or too low, but with probability $3/4$, provides an estimate in the acceptable range? How can you make the probability of error at most $\delta \in (0, 1/4)$?

[1] Based on the collaboration policy for CS332 (Mark Bun)

3. We now show a good hash function for the CountMin sketch for strings of length k on alphabet $A = \{0, \ldots, t-1\}$. Let $p \geq t$ be a prime number. Our hash function $h : A^k \to \{0, \ldots, p-1\}$ is

$$h(a_1 a_2 \ldots a_k) = \left(\sum_{i=1}^{k} z_i a_i \right) \bmod p,$$

where each z_i, $1 \leq i \leq k$, is selected independently and uniformly from $\{0, \ldots, p-1\}$. Show that for two different strings $a = a_1 a_2 \ldots a_k$ and $b = b_1 b_2 \ldots b_k$, the probability that $h(a) = h(b)$ is at most $1/p$.

Hint: Consider the last character on which the strings differ. Suppose it is the j-th character. If

$$(a_j - b_j) z' \equiv (a_j - b_j) z'' \mod p$$

for $z', z'' \in \{0, \ldots, p-1\}$, what is the relationship between z' and z''?

4. In Lecture 2, we saw how to find candidates for heavy hitters in a stream. (In short, we kept up to k elements with counts, and when we had k different elements, we discarded a single copy of each of them.) Let us refer to this algorithm as `FindHeavyCandidates`. Write full pseudocode for `FindHeavyCandidates`.

Note: You can assume basic data structures and any readable syntax is acceptable. If you borrow syntax from a programming language that is not very popular, please let us know what it is.

5. Recall that the CountMin sketch has two nice properties. First, it is easy to handle item deletions. Second, it is possible to compute separate sketches for disjoint subsets of your dataset and then combine them to obtain a sketch for the entire data set. Reply to the following questions about properties of `FindHeavyCandidates`. Explain your answers by giving a proof or showing a counterexample.

   (a) Can `FindHeavyCandidates` handle deletions?

   (b) Can `FindHeavyCandidates` be computed for disjoint subsets and then easily combined?

   (c) Is `FindHeavyCandidates` a linear sketch?

   (d) What is the probability of `FindHeavyCandidates` failing to output an element that has probability greater than $1/k$?

6. Design streaming algorithms for the following problems:

   (a) Suppose that the input stream is a sequence of updates to an initially empty multiset $S \subseteq \mathbb{Z}$. Each update is of the form either "insert a copy of x into S" or "delete a copy of x from S." You are promised that at the end of the stream, there will be exactly one element in S. Design a small space streaming algorithm that outputs this element.

   (b) Suppose that the input stream is a sequence of integers in the range $[n] = \{1, \ldots, n\}$ and you are promised that all of them appear exactly once, except for one of them that appears twice. Design a small space streaming algorithm that outputs the number that appears twice.

7. In the first discussion section, we have seen a sketch for approximating the second moment of data. More specifically, for each $x \in X$, let $f(x)$ be the number of times x appears in the data set. The quantity we want to estimate is

$$F_2 = \sum_{x \in X} f(x)^2.$$

The AMS sketch (where AMS $=$ Alon–Matias–Szegedy) equals $Y = \sum_{x \in X} h(x) \cdot f(x)$ where $h : X \to \{-1, 1\}$ is a random function in which each $h(x)$ is uniformly distributed on $\{-1, 1\}$.

(a) Assuming that h is fully random with all coordinates independent, show that Y^2 is an *unbiased estimator* for the second moment, i.e., $E[Y^2] = F_2$.

(b) **(Optional, no credit)** Under the same assumption, show that

$$\mathrm{Var}(Y^2) \leq 2F_2^2.$$

(c) Still under the same assumption and assuming the bound on variance, show how to use multiple independent copies of the AMS sketch to obtain an estimate Z such that $|Z - F_2| \leq \epsilon F_2$ with probability $99/100$ for any parameter $\epsilon \in (0, 1)$. How many copies of the sketch did you use?

(d) **(Optional, no credit)** Show that 4-independence of coordinates of h is sufficient for the analysis above. Show how to create 4-independent hash functions that can be stored in small space.

8. How much time (approximately) did you spend on this homework? Was is too easy/too hard?