Homework 1 (due 2/24)

DS-563 / CD-543 @ Boston University

Spring 2023

Before you start. . .

Collaboration policy: 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: 4VYBJ6). 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 (up to 10 points each)

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

   Note: For the rest of this homework, please be very explicit what probabilistic inequalities you are using when you apply them.

2. When analyzing CountMin Sketch, 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 $2/3$, provides a good estimate

(i.e., one that is in the desired range)? How can you make the probability of a bad estimate at most $\delta \in (0, 1/3)$?

Hint: How about taking the median? How can you bound the probability that half or more of the estimates are outside of the desired range?

3. In our analysis of CountMin Sketch, we showed that we can achieve a stream fraction estimate that differs from the exact value by at most ϵ with probability at most δ, where δ and ϵ are small values close to 0. This required $O(\epsilon^{-1} \log(1/\delta))$ counters, which we referred to as $O(\log(1/\delta))$ rows of $O(\epsilon^{-1})$ buckets each.

   (a) Suppose now that you are allowed to have only one row. How big would it have to be to achieve the same guarantee on each estimate?

   (b) What if you can use two rows? That is, what if you final estimate is the minimum of estimates from two rows?

4. In Lecture 3, 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.)

   (a) Write full pseudocode for this algorithm.

   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.

   (b) Is this algorithm a linear sketch? Explain.

   (c) Is this algorithm adversarially robust? Explain.
   Hint: Is it randomized?

5. Suppose that your stream is a stream of numbers in $[n] = \{1, \ldots, n\}$ and $f(x)$ is the number of times an element x appears in the stream. Consider the following estimator for $F_p = \sum_{x \in [n]} (f(x))^p$, the p-th moment:

```
$x_\star \leftarrow$ a uniformly random element in $[n]$
$c \leftarrow 0$
for each element $x$ of the stream:
    if $x = x_\star$:
        $c \leftarrow c + 1$
output $n \cdot c^p$
```

   Answer the following questions and explain your reasoning for your answers:

   (a) What is the space usage of this estimator?

   (b) Is it biased?

   (c) Is it useful?

6. Recall that in our algorithm for estimating the number of distinct elements, we used a random function $h : X \to \mathbb{N}$ such that for any $x \in X$ and any $i \in \mathbb{N}$, $\Pr(h(x) = i) = 2^{-(i+1)}$. We now construct a function that can play the role of h, assuming that we have a hash function $g : X \to \{1, \ldots, 2^k\}$ with the following properties:

- $k = \lceil 2 \log m \rceil$ and m is (an upper bound on) the length of the input stream.
- g is pairwise independent.
- Each $g(x)$ is distributed uniformly on $\{1, \ldots, 2^k\}$.

Define $h'(x) = \max\{i \in \mathbb{N} : g(x) \text{ is divisible by } 2^i\}$ for all x.

(a) Is h' pairwise independent? Explain why.

(b) What is the probability distribution of each $h'(i)$ and how does it differ from the distribution of h that we used in class?

(c) Argue why using h' instead of h gives essentially the same results, especially for sufficiently large m.

7. Design streaming algorithms for the following problems and analyze their space complexity:

   (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.

8. (a) Read again the section titled "Collisions (the Birthday Paradox)" in the handout on probabilistic inequalities.

   (b) Go to `https://www.xe.com/currencytables/` or any similar webpage. Select an arbitrary pair of currencies and check their exchange rate yesterday. Look at the two least significant digits. (Example: If the exchange rate is `3.523432`, then these digits are `32`.) Then look at the exchange rate two days ago, three days ago, and so on, always restricting your attention to the two least significant digits. How many days do you have to look back to see two days on which the exchange rate's two least significant digits are the same.

   (c) Repeat this experiment for four more pairs of currencies. Tell us what the results of your experiments are. Do you think these results match the theory, assuming that the two least significant digits are distributed uniformly?

9. Consider an experiment in which an event $\mathcal{E}$ occurs with probability p. Why does it suffice to run the experiment independently $\Omega(1/p)$ times to see $\mathcal{E}$ occur at least once with constant probability? Formalize this argument.

   Hint: What is the probability that you repeat the experiment k times and it does not occur? You may find the following inequality useful: $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$.

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

11. **(Optional, no credit)** A k-wise independent hash function construction that we considered in one of the discussion sections was based on the following property:

> Let p be a prime and let $(x_i, y_i) \in \mathbb{Z}_p^2$, for $i \in \{0, \ldots, k\}$, be pairs such that all x_i's are different. There is exactly one polynomial $F(x) = \left(a_0 + \sum_{i=1}^{k} a_i x^i\right) \bmod p$, in which all a_i's belong to $\mathbb{Z}_p$, such that for all $i \in \{1, \ldots, k\}$, $F(x_i) = y_i$.

How can you use this property to distribute a secret $z \in \mathbb{Z}_p$ among $t < p - 1$ entities such that no subset of at most k of them can learn anything about it, but any group of $k + 1$ can recover it?

Hint: Pick a random polynomial F, defined as above, such that $F(0) = z$. Send $F(i)$ to the i-th entity.