CS 368 (Stanford, Spring 2022) Homework 1

Due: **April 17, 2022, 11:59pm**

Policy: You are permitted to discuss and collaborate on the homework but you must write up your solutions on your own or as a group of two. Furthermore, you need to cite your collaborators and/or any sources that you consulted. No late submissions are allowed. There will be no late days. All homework submissions are subject to the Stanford Honor Code. For all assignments, we are allowing group submissions for groups of 1 or 2.
Submission: We will use Gradescope for homework submissions. You must typeset your write-up (we recommend LaTeX via Overleaf). If you are working as a group of two, only one group member needs to submit the assignment. When submitting, please remember to add all group member names on Gradescope.
Length of submissions: Please include as much of the calculations that show that you understand everything that is going through the answer. As a rule of thumb after you have solved the problem, try to identify what are the main steps taken and critical points of a proof and include them. Unnecessary long answers to questions will be penalized. The points next to each question are indicative of the hardness/length of the proof.

1 Estimating F_3 using Complex Numbers (30 pt.)

In class we saw a particular way of producing estimates of frequency moments $F_k = \sum_{i=1}^{n} f_i^k$ and we briefly explored whether different estimators are possible. In this problem, you will see how one can use the field of complex numbers to achieve this. Let $\mathcal{R}_k = \{x \in \mathbb{C} | x^k = 1\}$ be the set of k-roots of unity. For simplicity, we will focus on the case of $k = 3$. The proposed basic estimator works as follows:

1. For each $i \in [m]$ we pick independently a uniform random element $x_i \in \mathcal{R}_3$.
2. We form the random variable $Z = \sum_{i=1}^{n} f_i x_i$, by adding x_i to Z each time we come across element $i \in [m]$.
3. We estimate F_3 as $\mathrm{Re}\ Z^3$.

One can think of the mapping $i \mapsto x_i$ as hash function, that instead of mapping to the 2-roots of unity $\{-1, +1\}$ (in the original AMS scheme) maps to the 3-roots. You will analyze properties of this estimator:

(a) **(10 pt.)** Show that for any element $i \in [m]$, $\mathsf{E}[x_i^j] = \mathsf{E}[\bar{x}_i^j] = \begin{cases} 0 & \text{if } 1 \le j < 3 \\ 1 & \text{if } j = 3 \end{cases}$.

(b) **(10 pt.)** Show that $\mathsf{E}[\mathrm{Re}\ Z^3] = F_3$. *Hint: compute first* $\mathsf{E}[Z^3]$.

(c) **(10 pt.)** Show that $\mathrm{Var}[\mathrm{Re}\ Z^3] = O(F_2^3)$. *Hint: use the multinomial expansion*

2 Estimating the Join Size using Sketches (20 pt.)

A scientist at BigDatabase Inc. has observed that the second frequency moment F_2 is the size of a self-join: the join of a relation in a database with itself. In fact, one can design a sketch that can scan a relation in one pass (i.e., in streaming fashion) such that, based on the sketches of two different relations, we can estimate the size of their join. This scientist has tasked you to design such a sketch and show that it works (with the full analysis including a high probability guarantee).

Recall that for two relations (i.e., tables in a database) $r(A, B)$ and $s(A, C)$, with a common attribute (i.e., column) A, we define the join $r \bowtie s$ to be a relation consisting of all tuples (a, b, c) such that $(a, b) \in r$ and $(a, c) \in s$. Therefore, if $f_j^{(r)}$ and $f_j^{(s)}$ denote the frequencies of j in the first columns (i.e., "A"-columns) of r and s, respectively, and j can take values in $[n]$, then the size of the join is $\sum_{j=1}^{n} f_j^{(r)} f_j^{(s)}$. Let $S = \sqrt{\sum f_i^{(s)^2}}$ and $R = \sqrt{\sum f_i^{(r)^2}}$. When estimating the error of your design, please give an error guarantee relative to RS.

3 A $(1 \pm \epsilon)$-factor Approximation Algorithm (40 pt.)

Assume a data stream $x_1, \ldots, x_n \in [m]$ where $[m]$ denotes the set $\{1, \ldots, m\}$ for some large m. For simplicity, you may assume that m is a power of 2. Let F_0 denote the number of distinct elements in the data stream. Consider the following algorithm. Note c is some constant to be chosen in your analysis and zeros(j) is the number of trailing 0's in the binary representation of j for $j \in [m]$.

Algorithm 1

Choose $h : [m] \to [m]$ from a pairwise independent family of hash functions;
$z \leftarrow 0$;
$B \leftarrow \emptyset$;
for *each* x_i **do**
  if zeros$(h(x_i)) \geq z$ **then**
    $B \leftarrow B \cup \{(x_i, \text{zeros}(h(x_i))\}$
  while $|B| \geq c/\epsilon^2$ **do**
    $z \leftarrow z + 1$;
    Remove all (x_i, z') where $z' < z$ from B;
Return $\tilde{F}_0 = |B| \cdot 2^z$

Let $\mathcal{F}$ denote the event that Algorithm 1 returns an answer $\tilde{F}_0$ that is not within the $1 \pm \epsilon$ multiplicative factor of the true number of distinct elements F_0 and $\Pr(\mathcal{F})$ be the failure probability. We show that $\Pr(\mathcal{F})$ can be upper bounded by looking at two cases based on the final value of z, call this z^*, at the termination of Algorithm 1. Note $0 \leq z^* \leq \log_2 m$. The

failure probability can be written:

$$\Pr(\mathcal{F}) = \Pr(\mathcal{F} \wedge z^* = 0) + \Pr(\mathcal{F} \wedge z^* \geq 1)\,.$$

$\Pr(\mathcal{F} \wedge z^* = 0)$ is the failure probability in the case $z^* = 0$ and $\Pr(\mathcal{F} \wedge z^* \geq 1)$ is the failure probability in the case $z^* \geq 1$.

Please use the following random variables in your analysis. For each $j \in [m]$ and integer $r \geq 0$, we define:

$$X_{r,j} = \begin{cases} 1, \text{ if zeros}(h(j)) \geq r \\ 0, \text{ otherwise} \end{cases}$$
$$Y_r = \sum_{j:f_j>0} X_{r,j}\,,$$

where f_j is the number of occurrences of element j in the given data stream.

(a) **(10 pt.)** Show that $\Pr(\mathcal{F} \wedge z^* = 0) = 0$.

(b) **(10 pt.)** For some s to be chosen (just for the analysis and not used in the algorithm), show that $\Pr(\mathcal{F} \wedge z^* \geq 1) \leq \frac{\mathrm{E}[Y_{s-1}]}{c/\epsilon^2} + \sum_{r=1}^{s-1} \frac{\mathrm{Var}(Y_r)}{(\epsilon F_0/2^r)^2}$. In your derivation, you may assume $s \in (1, \log_2 m]$. *Hint: use Chebyshev's inequality and Markov's inequality.*

(c) **(10 pt.)** Show $\Pr(\mathcal{F} \wedge z^* \geq 1) < \frac{2^s}{\epsilon^2 F_0} + \frac{F_0 \epsilon^2}{c 2^{s-1}}$. Let s be the unique integer such that $\frac{12}{\epsilon^2} \leq \frac{F_0}{2^s} < \frac{24}{\epsilon^2}$ (why does such an s exist?). Choose c so that $\Pr(\mathcal{F} \wedge z^* \geq 1)$ is at most some constant strictly less than $\frac{1}{2}$.

(d) **(10 pt.)** Conclude that $\Pr(\mathcal{F})$ is at most some constant strictly less than $\frac{1}{2}$. Using an idea discussed in class, improve the success probability to $1 - \delta$ for any $\delta > 0$. What is the total space used by the modified algorithm?

4 Data Streams with Insertions and Deletions (50 pt.)

Assume a data stream $(x_1, \Delta_1), \ldots, (x_n, \Delta_n)$ where $x_i \in [m] = \{1, \ldots, m\}$ and $\Delta_i \in \{\pm 1\}$. $\Delta_i = +1$ corresponds to an insertion of the element x_i and $\Delta_i = -1$ corresponds to a deletion of the element x_i. Let F_0 denote the number of distinct elements with counts not equal to 0, i.e., $F_0 = \#\{j | \sum_{i:x_i=j} \Delta_i \neq 0\}$. We show that we can compute $\tilde{F}_0 \in [(1-\epsilon)F_0, (1+\epsilon)F_0]$ with probability at least $1 - \delta$ using poly $\left(\log \frac{1}{\delta}, \frac{1}{\epsilon}, \log n, \log m\right)$ space. We use poly$(\cdot)$ to denote some polynomial function of its arguments. We note that there are other, more sophisticated, approaches that improve on the space usage of the scheme you will analyze in this problem.

To approximately compute F_0, we will make use of the following subroutines. Roughly speaking, our approach uses Subroutine A_1 at the top level, which in turn uses Subroutine A_2.

Subroutine A_1

Input: A data stream (with insertions and deletions) and threshold T. ;
Output: With probability at least $1-\delta_1$, it returns 'YES' if $F_0 > (1+\epsilon_1)T$ and 'NO' if $F_0 < (1-\epsilon_1)T$, and uses poly $\left(\log \frac{1}{\delta_1}, \frac{1}{\epsilon_1}, \log n, \log m\right)$ space.
If $F_0 \in [(1-\epsilon_1)T, (1+\epsilon_1)T]$, it returns either 'YES' or 'NO'.

Subroutine A_2

Input: A data stream (with insertions and deletions). ;
Output: It returns 'ZERO' if $F_0 = 0$ and 'NOT ZERO' otherwise with probability at least $1-\delta_2$ using poly $\left(\log \frac{1}{\delta_2}, \log m, \log n\right)$ space.

We will show that each of these subroutines can be implemented. In part a, we show that Subroutine A_2 can be implemented. In parts b - d, we show that the following algorithm, Algorithm 2 for some R and g to be determined, implements A_1.

Algorithm 2

Input: A data stream and threshold T. ;
Output: Sample R subsets $S_1, \ldots, S_R$ of $[m]$ where each element is included in each subset with probability $\frac{1}{T}$ independently. Let $\mathcal{S}_i$ be the elements of the stream that are in S_i. Run Subroutine A_2 on $\mathcal{S}_1, \ldots, \mathcal{S}_R$ in parallel. If the fraction of 'ZERO' outcomes out of R outcomes is at most $g(T, \epsilon_1)$, return 'YES'. Otherwise, return 'NO'.

For a randomly generated set S where each element is included with probability $\frac{1}{T}$, let $F_0|_S$ be the number of distinct elements of the data stream restricted to the subset S. Note that to obtain such a random set S, we choose a random hash function $h : [m] \to [0,1]$ and only consider those elements that hash to those values less than $\frac{1}{T}$. For the purpose of this problem, we assume there exists a family of hash functions that hash elements to $[0,1]$ uniformly randomly and ignore space requirements for it.

(a) **(10 pt.)** Design and analyze an algorithm that implements A_2. *Hint: Use the AMS sketch.*

(b) **(10 pt.)** Let p be the probability that $F_0|_S = 0$ when $F_0 > (1+\epsilon_1)T$ and q be the probability that $F_0|_S = 0$ when $F_0 < (1-\epsilon_1)T$. (These probabilities are over the random choice of S). Show that $p \leq \left(1-\frac{1}{T}\right)^{(1+\epsilon_1)T}$ and $q \geq \left(1-\frac{1}{T}\right)^{(1-\epsilon_1)T}$.

(c) **(10 pt.)** Show that $q - p = \Omega(\epsilon_1)$. For simplicity, assume $T \geq \frac{3}{2}$.[1]
Hint: The Taylor expansion $\alpha^x = e^{x \ln \alpha} = \sum_{i=0}^{\infty} \frac{(x \ln \alpha)^i}{i!}$ *might be useful here.*

(d) **(10 pt.)** Choose an appropriate R (in terms of ϵ_1 and δ_1) and g (in terms of T and ϵ_1) and show that Algorithm 2 implements A_1.
Hint: Use the additive version of Chernoff bound in this handout:
`https://www.cs.cmu.edu/~avrim/Randalgs11/lectures/lect0124.pdf`.

(e) **(10 pt.)** Using Subroutines A_1 and A_2, design and analyze an algorithm that for any $\delta > 0$, returns an estimate $\tilde{F}_0 \in [(1-\epsilon)F_0, (1+\epsilon)F_0]$ with probability at least $1 - \delta$ and uses poly $\left(\log \frac{1}{\delta}, \frac{1}{\epsilon}, \log n, \log m\right)$ space.
Hint: Consider values of T *of the form* $(1+\epsilon/c)^i$ *for some constant* $c \geq 1$*, and integers* i.

[1] Since the answer F_0 to our original problem is an integer, it suffices to consider T of the form $(1+\epsilon/c)^i$ (for some $c \geq 1$, and integer i) that are bounded away from 1, say at least $\frac{3}{2}$, for sufficiently small ϵ. We will not treat this here.