CS 514: Advanced Algorithms II – Sublinear Algorithms **Rutgers: Fall 2021**

Problem Set 2

Due: 11:59PM, November 9, 2021

Problem 1. Recall the uniform distribution testing problem from Lecture 5: We are given sample access to a distribution μ on domain $[n]$ and our goal is to decide whether μ is the uniform distribution U_n on $[n]$ or it is ε-far from U_n in total variation distance, i.e., $\Delta_{\text{tvd}}(\mu, U_n) \geq \varepsilon$. We showed that for any *constant* $\varepsilon \in (0,1)$, this problem can be solved with constant probability using $O(\sqrt{n})$ samples. Our goal now is to prove $\Omega(\sqrt{n})$ samples are also necessary for solving this problem for some *constant* $\varepsilon \in (0,1)$.

(a) Define $\mathcal{D}_{\mathsf{no}}$ as the distribution over family of distributions on $[n]$ as follows (elements of $\mathcal{D}_{\mathsf{no}}$ are itself distributions, i.e., when we sample from $\mathcal{D}_{\mathsf{no}}$, we obtain a distribution):

- A sample distribution μ from $\mathcal{D}_{\mathsf{no}}$ is obtained by sampling a subset $S \subseteq [n]$ of size $(n/2)$ uniformly at random, and letting $\mu(i) = \frac{2}{n}$ for all $i \in S$ and $\mu(j) = 0$ for $j \notin S$ (i.e., distribution μ is uniform over S and has no mass in $[n] - S$).

Prove that any distribution $\mu \sim \mathcal{D}_{\mathsf{no}}$ is ε-far from uniform in total variation distance for some fixed constant $\varepsilon \in (0,1)$. **(5 points)**

(b) Consider any algorithm A that uses $o(\sqrt{n})$ samples for uniformity testing: Prove that A *cannot* distinguish between U_n or a distribution $\mu \sim \mathcal{D}_{\mathsf{no}}$ with probability of success at least $2/3$. Conclude that any algorithm for uniformity testing with some constant $\varepsilon \in (0,1)$ with probability of success at least $2/3$ requires $\Omega(\sqrt{n})$ samples. **(15 points)**

Problem 2. Recall the problem of sparse recovery on $\mathbb{F}_2$ we studied in Lecture 6: Given a k-sparse vector $x \in \mathbb{F}_2^n$, i.e., $||x||_0 = k$, design a matrix A such that we can recover x *uniquely* from $A \cdot x$ (note that the computation is done over $\mathbb{F}_2$). In the class, we saw that we can find an A with $O(k \cdot \log(n/k))$ rows for this task and proved that $\Omega(k \cdot \log(n/k))$ rows are also necessary. However, our algorithm required *exponential* time (in k) for constructing A and also recovering x from $A \cdot x$.[1]

To bypass this, we considered the easier task of designing a polynomial-time randomized algorithm that outputs a matrix A such that with probability at least $9/10$, for *each* fixed (unknown) k-sparse x, we can recover x from $A \cdot x$ – note that this is a weaker guarantee than our previous algorithm in terms of recovery that simultaneously worked for all k-sparse $x \in \mathbb{F}_2^n$. In Lecture 6, we designed an algorithm for this problem with $O(k \cdot \log(n) \cdot \log k)$ rows, which is sub-optimal. The goal of this question is to obtain optimal bounds for this problem.

Design a polynomial-time randomized algorithm that outputs a matrix A with $O(k \cdot \log(n/k))$ rows such that with probability at least $9/10$, for any fixed (unknown) k-sparse x, we can recover x from $A \cdot x$.

(20 points)

Hint: Use the approach in Lecture 6 by designing a matrix B that can recover a *constant* fraction of 1 entries of x using $O(k \cdot \log(n/k))$ rows, with some "potential" error, namely, allowing for recovering wrong values (as long as their numbers is small). Specifically, one should be able to recover a vector $y \in \mathbb{F}_2^n$ from $B \cdot x$ such that $||y - x||_0$ is at most, say, $k/4$.

Show that another *independent* copy of the matrix in the previous part with a constant factor smaller number of rows can be used to recover a constant fraction of 1's in the vector $y - x$ now (again with some error).

[1] For larger fields, we also saw an algorithm with polynomial time for constructing A (taking a Vandermonde matrix) and even though we did not covered it in the class, we claimed one can also recover x from $A \cdot x$ in polynomial time.

Continue this to recover all the vector x, by reducing the number of rows for each copy of B with a small factor, so that the total number of rows form a geometric series and thus still adds up to $O(k \cdot \log(n/k))$.

Bonus part: Design a poly-time randomized algorithm for this problem with $O(k \cdot \log(n/k))$ rows that guarantees a "for-all" recovery, i.e., with constant probability, say, $9/10$, one can recover *every* $x \in \mathbb{F}_2^n$ from $A \cdot x$ in polynomial time. **(+25 points)**

Problem 3. Given a set of numbers S and a number $x \in S$, the *rank* of x is defined to be the number of elements in S that have value at most x:

$$\mathsf{rank}(x, S) = |\{y \in S : y \leq x\}|.$$

Given a parameter $\varepsilon \in (0, 1/2]$, we say that an element $x \in S$ is an ε-approximate element of rank r if

$$(1-\varepsilon) \cdot r \leq \mathsf{rank}(x, S) \leq (1+\varepsilon) \cdot r.$$

Suppose we are given a stream of numbers $S = s_1, s_2, \ldots, s_n$, where $s_i \in [m]$ for $1 \leq i \leq n$, and assume that all s_i's are *distinct*. Our goal is to design an $O(\varepsilon^{-2} \log m \log n)$ space streaming algorithm for retrieving an ε-approximate element for any given rank value.

(a) Consider the following algorithm for computing an ε-approximate median: sample $O(\varepsilon^{-2} \log n)$ numbers from the stream uniformly at random (with repetition) and return the median of the samples.

Show that this algorithm returns an ε-approximate median with probability at least $1 - 1/\text{poly}(n)$ and uses $O(\varepsilon^{-2} \cdot \log m \cdot \log n)$ bits of space. **(10 points)**

(b) We now extend the previous algorithm to compute an ε-approximate element of rank r for any $r \in [n]$.

Consider the following algorithm for this problem. Let $t = \lceil 24\varepsilon^{-2} \log n \rceil$. If $r \leq t$, then simply maintain a list T of r smallest elements seen in the stream, and output the largest element in T at the end of the stream. Otherwise, choose each element in the stream with probability t/r, and maintain the t smallest sampled values in a list T. At the end of the stream, output the *largest* number in T.

Show this algorithm returns an ε-approximate element of rank r with probability at least $1 - 1/\text{poly}(n)$ and uses $O(\varepsilon^{-2} \cdot \log m \cdot \log n)$ bits of space. **(10 points)**

Problem 4. Suppose we are given a stream of numbers $e_1, \ldots, e_n$ from a universe $[m]$ which defines a *frequency vector* $f \in \mathbb{N}^m$, namely, f_i denotes the number of times $i \in [m]$ has appeared in the stream. In this language, the distinct element problem we studied in Lecture 7 corresponds to estimating $||f||_0$.

In this question, we consider algorithms for another problem related to the frequency vector, namely, pointwise estimation of f. In particular, the streaming algorithm needs to output a data structure such that at the end of the stream, for any given $i \in [m]$, with probability at least $1 - \delta$, we can recover $\widetilde{f}_i$ from this data structure where:

$$f_i \leq \widetilde{f}_i \leq f_i + \varepsilon \cdot ||f||_1.$$

(a) The standard solution for this problem is called the **count-min sketch**. Let $a = 10 \ln(1/\delta)$ and $b = \frac{4}{\varepsilon}$. Pick a pairwise-independent hash functions $h_1, \ldots, h_a : [m] \to [b]$. Throughout the stream, compute $a \cdot b$ counters:

$$c_{p,q} = |\{e_i \text{ in the stream} \mid h_p(e_i) = q\}|.$$

At the end of the stream, given any $i \in [m]$, return

$$\widetilde{f}_i = \min_{p \in [a]} \left(c_{p,q} \text{ where } q = h_p(i) \right).$$

Prove that count-min sketch described above solves the given problem and analyze its space complexity.

(10 points)

(b) For a frequency vector f and $\phi \in (0,1)$, a *ϕ-heavy hitter* of f is any element $i \in [m]$ such that $f_i \geq \phi \cdot ||f||_1$. Design a streaming algorithm that given a stream $e_1, \ldots, e_n$ from universe $[m]$ (defining the frequency vector f) and parameters $\phi, \varepsilon, \delta \in (0, 1/2)$, outputs a list of at most $\frac{2}{\phi}$ numbers such that with probability $1 - \delta$:

(i) every ϕ-heavy hitter of f belongs to this list,

(ii) and no element which is *not* a $(\phi - \varepsilon)$-heavy hitter in f is reported in this list.

The space complexity of your algorithm should be poly-logarithmic in $n, m, (1/\delta)$, and polynomial in ϕ and ε. **(10 points)**

Problem 5. We considered communication complexity in Lecture 8. In this question, we will prove a communication lower bound for another fundamental communication problem, namely, the **inner product (IP)** problem. In this problem, Alice is given a vector $x \in \mathbb{F}_2^n$ and Bob is given $y \in \mathbb{F}_2^n$; their goal is to compute $\langle x, y \rangle$, namely, the inner product of x and y (over $\mathbb{F}_2$).

Our goal is to prove a lower bound on the *(public coin) randomized* communication complexity of this problem, namely, show that $R(IP) = \Omega(n)$. We will prove this by lower bounding $D_\mu(IP)$ over some distribution μ and then apply (the easy direction of) Yao's minimax principle (for simplicity, we consider distributional communication complexity that succeed with probability at least 99/100 instead of 2/3).

The distribution μ we work with is simply the uniform distribution over all pairs $(x, y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, i.e., the input of Alice and Bob is chosen independently and uniformly at random.

(a) Suppose π is a deterministic protocol that solves IP over μ with probability of success at least 0.99 using c bits of communication. Prove that there exists a transcript Π in π such that: (i) conditioned on the transcript of protocol π being Π, the protocol outputs the correct answer with probability at least 0.9, and (ii) the probability that transcript Π is communicated by the players is at least 2^{-c+10}.

(7 points)

(b) Consider the matrix $M \in \{-1, 1\}^{2^n \times 2^n}$ with rows and columns indexed by vectors in $\mathbb{F}_2^n$ such that $M_{x,y} = (-1)^{IP(x,y)}$ (this is "sort of" the communication matrix of IP).

Use the previous part to argue that there should be a combinatorial rectangle $A \times B \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^n$ in M with $|A| \cdot |B| \geq 2^{2n-c+10}$ such that

$$\left| \sum_{(x,y) \in A \times B} M_{x,y} \right| \geq 0.7 \cdot |A| \cdot |B|,$$

namely, $A \times B$ is "almost monochromatic". **(6 points)**

(c) Prove that for $c < n/100$, no such (almost monochromatic) combinatorial rectangle as described in the previous part exist in M. Use this to conclude the lower bound.

Hint: Observe that M is simply the well-known Hadamard matrix of dimension 2^n and so this property can be proven from known discrepancy bounds for Hadamard matrices. **(6 points)**

Problem 6 (Bonus problem). For any communication problem $f : \{0,1\}^n \to \{0,1\}^n \to \{0,1\}$, define $R_{pri}(f)$ to be the *private-coin* communication complexity of f, namely, the minimum number of bits needed to solve f using any private-coin protocol with probability of success at least 2/3 (thus $R_{pri}(f) \geq R(f)$).

Prove that for any function f, $R_{pri}(f) = \Omega(\log D(f))$ where $D(f)$ is the deterministic communication complexity of the problem defined in Lecture 8. You will receive partial credit even if you only prove this for one-way protocols. **(+25 points)**