Warm-up

Problem 1. Go through the "median-of-means" proof for the Morris counter, to prove the statement about the result.

Problem 2. Instead of using the "median-of-means" trick to boost the accuracy of the Morris counter, what happens if we were to do the opposite and use the "mean-of-medians"?

Problem solving

Problem 3. Analyse the "careful variant" of the Morris counter, where instead of doubling C with probability $1/C$, we multiply C by $1+\alpha$ with probability $1/(\alpha C)$.

a) Compute the expectation of C at the end of the algorithm.

b) Compute its variance, and conclude by Chebyshev.

c) Explain how you would set α to get a $(1+\varepsilon)$-factor estimate of the true count with probability at least $1-\delta$ using the median trick, and give the resulting space complexity, *almost* proving the theorem stated in the lecture (but with a multiplicative instead of additive $\log(1/\delta)$.

d) Explain how you would set α to get a $(1+\varepsilon)$-factor estimate of the true count with probability at least $1-\delta$ *without* using the median trick, and give the resulting space complexity, *actually* proving the theorem stated in the lecture (with an additive $\log(1/\delta)$).

e) How would you actually implement the increment step (i.e., how, given random uniform bits, would you "multiply C by $1+\alpha$ with probability $1/(\alpha C)$")?

Problem 4. Given a stream σ of length m and $\varepsilon \in [0,1]$, let

$$H_\varepsilon(\sigma) = \{j \in [n] : f_j \geq \varepsilon \cdot m\}$$

denote the set of *ε-heavy hitters* of σ. Modify the Misra–Gries algorithm to make it output a set $H \subseteq [n]$ such that $H_\varepsilon(\sigma) \subseteq H \subseteq H_{\varepsilon/2}(\sigma)$. *(That is, the algorithm outputs every ε-heavy hitter, and everything it outputs is at least an $(\varepsilon/2)$-heavy hitter.)* Your algorithm should be one-pass, and use space $O(\log(mn)/\varepsilon)$.

Problem 5. Consider the following "Bottom-k" algorithm for the Distinct Elements problem, where $k \geq 1$ is a parameter.

a) Show that, for $k = \Theta(1/\varepsilon^2)$, the above algorithm returns $(1 \pm \varepsilon)$-estimate of the number of distinct elements d, with probability at least 99%. (To do so, define X_i as the indicator that $h(a_i) < \frac{k}{(1+\varepsilon)d}$, and use Chebyshev on $\sum_i X_i$.)

b) What is the space complexity of the algorithm?

Pick a hash function $h\colon [n] \to [0,1]$ from a strongly universal hash family
▷ *Technically, from $h\colon [n] \to \{0, 1/N, 2/N \ldots, 1\}$ where $N = poly(n)$ is large enough to not have to worry about collisions.*
Set $z = (1, 1, \ldots, 1) \in \mathbb{R}^k$
for all $1 \leq i \leq m$ **do**
    Get new element $a_i \in [n]$ of the stream
    $z \leftarrow k$ smallest values among $z_1, \ldots, z_k, h(a_i)$
return $\hat{d} \leftarrow \frac{k}{\max(z_1, \ldots, z_k)}$

Advanced

Problem 6. Given a stream σ of length m where each element a_i is a vector in $\{-1, 1\}^d$, our goal is to estimate how large the mean vector

$$\bar{a} := \frac{1}{m} \sum_{i=1}^{m} a_i \in [-1, 1]^d$$

is: that is, to obtain a multiplicative estimate of $\|\bar{a}\|_2$ within a factor 2, with probability at least 99%. Assuming that the algorithm does not "pay" the cost of storing the random bits it uses, describe an approach to do this with very small space complexity. What is the number of "free random bits" your algorithm uses?