15-451 Algorithms, Spring 2019
Practice Problems

Recap of this week's lectures:

- Streaming
- Picking a random prime, and the density of primes
- String equality testing and Karp-Rabin Fingerprinting

Streaming.

Sampling: Given a number k, you want to maintain a random sample of size k from the stream. I.e., for each $n \geq k$, the set you have at time n should be a random subset of the prefix $a_{[1:n]}$, each of the $\binom{n}{k}$ subsets of size k from this prefix should be equally likely.

1. For $k = 1$, show that the algorithm: pick the first element. When faced with the n^{th} element, with prob. $1/n$ discard the element in your hand and pick the new element, and with prob. $1 - 1/n$ keep the element in hand.

2. Give an algorithm for general k. (What would you do when faced with the n^{th} element? With what probability should you pick this element? Which element should you drop?)

Missing Numbers: Suppose I give you a stream of $n - 1$ elements, which contains all the numbers from 1 thru n *except one of them.* (The numbers *do not* appear in sorted order.) Clearly you can figure out the missing number by storing all $n - 1$ numbers and looking for

the missing number. How can you output the missing number with only $O(\log n)$ space? What if there are two missing numbers: can you again use only $O(\log n)$ space?

Hashing, Fingerprinting, etc.

In `min-hashing` we created an estimator with "one-sided error": our estimate was always an *over*estimate. I.e., for the target value v we created a random variable X such that $\mathbf{Pr}[X \geq v] = 1$. Suppose $\mathbf{E}[X] = \mu$.

1. Show that $\mathbf{Pr}[X \geq 2\mu] \leq \frac{1}{2}$. ("The probability of one estimate being too large is at most 50%.")

2. Use this to show that if we take k independent copies $X_1, X_2, \ldots, X_k$ of the r.v. X, then $\mathbf{Pr}[min_{i=1}^{k}(X_i) \geq 2\mu] \leq 2^{-k}$.

3. Show that $k = \lg(1/\delta)$ gives $2^{-k} = \delta$. (If we want error probability 2^{-100}, take the minimum of 100 independent estimates.)

Many Patterns: You are given a set of patterns $P_1, P_2, \ldots, P_k$ of equal length (all of them having length n) and a text T of length m. Give an algorithm to find all the locations i such that some pattern P_j occurs as a substring of T starting at location i. The expected runtime should be $O(kn + m)$, and the probability of error is at most 0.01. [1]

[1] Assume you can do arithmetic operations on numbers of size $O(\log(kmn))$ in constant time, even modulo a prime.