cs473: Algorithms Assigned: *Fri., Oct. 9, 2020*

Problem Set #5

Forbes/Chekuri Due: *Thu., Oct. 22, 2020 (5:00pm)*

All problems are of equal value.

1. **Sampling, Chebyshev vs Chernoff.** Suppose you want to estimate the average of n numbers via sampling, for example the average wealth of people in a town. The average can be very skewed by outliers — perhaps there are a few billionaires that will not make it to the sample but will clearly affect the average. However, we can obtain an accurate estimate if we assume that the numbers are within some limited range. Assume the input numbers $z_1, z_2, \ldots, z_n$ are from $[a, b]$ where $a, b \in \mathbb{R}$ with $a \leq b$. Suppose you sample k input numbers (with replacement) and output their average as the estimate for the true average $\alpha = (\sum_i z_i)/n$. Let X be the random variable denoting the output value.

   - Using Chebyshev's inequality, show that for $k \geq \frac{(b-a)^2}{\delta\epsilon^2}$, we have
   $$\Pr[|X - \alpha| \geq \epsilon] \leq \delta.$$
   - Using the Chernoff inequality, show that there exists a constant $c > 0$ such that for $k \geq \frac{c(b-a)^2 \log(2/\delta)}{\epsilon^2}$, we have
   $$\Pr[|X - \alpha| \geq \epsilon] \leq \delta.$$

2. **Hashing.** In this problem we consider yet another method for universal hashing. Suppose we are hashing from the universe $\mathcal{U} = \{0, 1, \ldots, 2^w - 1\}$ of w-bit strings to a hash table of size $m = 2^\ell$; that is, we are hashing w-bit *words* into ℓ-bit *labels*. To define our universal family of hash functions, we think of words and labels as *boolean vectors* of length w and ℓ, respectively, and we specify our hash function by choosing a random *boolean matrix*.

   For any $\ell \times w$ matrix M of 0s and 1s, define the hash function $h_M \colon \{0,1\}^w \to \{0,1\}^\ell$ by the boolean matrix-vector product
   $$h_M(x) = Mx \bmod 2 = \bigoplus_{i=1}^{w} M_i x_i = \bigoplus_{i \colon x_i = 1} M_i.$$
   where $\oplus$ denotes bitwise exclusive-or (that is, addition mod 2), M_i denotes the ith column of M, and x_i denotes the ith bit of x. Let $\mathcal{M} = \{h_m \mid M \in \{0,1\}^{w \times \ell}\}$ denote the set of all such random-matrix hash functions.

   For example, suppose $w = 8$ and $\ell = 4$. Let M be the $w \times \ell$ matrix
   $$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Then we can compute $h_M(173) = 12$ as follows:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

(a) Prove that $\mathcal{M}$ is a 2-universal family of hash functions.

(b) Prove that $\mathcal{M}$ is *not* uniform.

(c) Now consider a modification of the previous scheme, where we specify a hash function by a random matrix $M \in \{0,1\}^{\ell \times w}$ and an independent random offset vector $b \in \{0,1\}^{\ell}$:

$$h_{M,b}(x) = (Mx + b) \bmod 2 = \left(\bigoplus_{i=1}^{w} M_i x_i \right) \oplus b$$

Prove that the family $\mathcal{M}^+$ of all such functions is *strongly* universal (2-uniform).

(d) Prove that $\mathcal{M}^+$ is *not* 4-uniform.

(e) **[Extra credit]:** Prove that $\mathcal{M}^+$ is actually 3-uniform.

3. **Two-dimensional pattern matching.** In lecture we discussed the Karp-Rabin randomized algorithm for pattern matching. The power of randomization is seen by considering the *two-dimensional* pattern matching problem. The input consists of a $n \times n$ binary matrix T and a $m \times m$ binary matrix P. Our goal is to check if P occurs as a (contiguous) submatrix of T. Describe an algorithm that runs in $O(n^2)$ time assuming that arithmetic operation in $O(\log n)$-bit integers can be performed in constant time. This can be done via a modification of the Karp-Rabin algorithm. To achieve this, you will have to apply some ingenuity in figuring out how to update the fingerprint in only constant time for most positions in the array.

Hint: We can view an $m \times m$ matrix as an m^2-bit integer. Rather than computing its fingerprint directly, compute instead a fingerprint for each row first, and maintain these fingerprints as you move around.