Homework 10

Out: 21 Apr. Due: 28 Apr.

Submit your solutions in pdf format on Gradescope by **5pm on Friday, April 28**. *Solutions may be written either in LaTeX (with either machine-drawn or hand-drawn diagrams) or* **legibly** *by hand. (The LaTeX source for this homework is provided in case you want to use it as a template.) Please be sure to begin the solution for each problem on a new page, and to tag each of your solutions to the correct problem! Per course policy, no late solutions will be accepted. Take time to write* **clear** *and* **concise** *answers; confused and long-winded solutions may be penalized. You are encouraged to form small groups (two to four people) to work through the homework, but you* **must** *write up all your solutions on your own. Depending on grading resources, we reserve the right to grade a random subset of the problems and check off the rest; so you are advised to attempt all the problems.*

1. The following question arose in our proof in class of Wald's Equation.

(a) Suppose X is a *non-negative* random variable with finite expectation μ. Show that $\mathrm{E}[|X-\mu|] \le 2\mu$.

(b) Show that the constant 2 in part (a) is optimal, by constructing, for any $\varepsilon > 0$, a non-negative random variable that achieves $\mathrm{E}[|X-\mu|] = (2-\varepsilon)\mu$.

(c) Show that if we remove the non-negativity assumption then $\mathrm{E}[|X-\mu|]$ can be unbounded.

2. Let $a = a_1 \dots a_n$ and $b = b_1 \dots b_n$ be two binary sequences of length n. A *longest common subsequence* (lcs) of a and b is a string s of maximum length such that s occurs as a (not necessarily contiguous) substring of both a and b. For example, a lcs of the strings 001001001 and 100101101 is 0010101.

Suppose now that the symbols of a and b are all chosen independently and u.a.r. from $\{0,1\}$. Let the r.v. X_n denote the length of a lcs of a, b. It can be shown quite easily that the expectation $\mu_n = \mathrm{E}[X_n]$ satisfies $\mu_n \sim \gamma n$ as $n \to \infty$, where $\gamma \approx 0.8$ is a constant (but you are not expected to do this!).

(a) Use Azuma's inequality to show that

$$\Pr[|X_n - \mu_n| \ge \lambda] \le 2\exp(-\lambda^2/cn)$$

for some constant $c > 0$ (which you should specify) and for any $\lambda > 0$. Be explicit in your definition of the martingale, and in your derivation of a bound on the martingale differences.

(b) Is the result of part (a) useful for this problem?

(c) How, if at all, does each of the following changes to the problem affect your bound of part (a)? Justify your answers rigorously.

(i) The symbols of a, b are not binary, but are chosen u.a.r. from an alphabet of size k.
(ii) The symbols of a, b are not independent.
(iii) There are three strings a, b, c instead of just two.

[continued on next page]

3. In the *bin packing* problem we are given n items of sizes $X_1, \ldots, X_n$, with $0 < X_i < 1$, and asked to find the minimum number of bins of capacity 1 into which the items can be packed. To analyze this problem in the average case, assume that the X_i are i.i.d. uniform on the continuous interval $[0, 1]$. Denote by B_n the minimum number of bins required to pack the n items. It can be shown that $\mu_n := \mathrm{E}[B_n] \sim \gamma n$ as $n \to \infty$, for some constant $\gamma > 0$. By defining a suitable martingale, show that B_n is concentrated in an interval of width $O(\sqrt{n})$ about μ_n. Be sure to carefully justify any bounds you need on the martingale differences. Does this result still hold if the distribution of the X_i is not uniform?

4. Consider the *two-dimensional* pattern matching problem, in which we are given as input a $n \times n$ binary matrix X and a $m \times m$ binary matrix Y, with $m < n$. Our goal is to determine whether Y occurs as a (contiguous) submatrix of X.

Describe how to modify the Karp/Rabin fingerprinting algorithm to handle this case. Your algorithm should have running time $O(n^2)$, assuming that arithmetic operations on $O(\log n)$-bit integers can be performed in constant time. 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.]