Randomness and Computation **Edinburgh INFR11089, Spring 2015**

Problem Set 1

Due: Friday, February 27, 4p.m. at the ITO

---

1. (a) Suppose you are given a coin for which the probability of HEADS, say p, is *unknown.* How can you use this coin to generate unbiased (i.e., $\mathbf{Pr}[\text{HEADS}] = \mathbf{Pr}[\text{TAILS}] = 1/2$) coin-flips? Give an algorithm for which the expected number of flips of the biased coin for extracting one unbiased coin-flip is no more than $1/[p(1-p)]$.

   (b) Let $a_1, \ldots, a_n$ be a list of n distinct numbers. We say that a_i and a_j are inverted if $i < j$ but $a_i > a_j$. The *Bubblesort* sorting algorithm swaps pairwise adjacent inverted numbers in the list until there are no more inversions, so the list is in sorted order. Suppose that the input to Bubblesort is a random permutation, equally likely to be any of the $n!$ permutations of n distinct numbers. Determine the expected number of inversions that need to be corrected by Bubblesort.

2. (a) Consider the following balls-and-bin game. We start with one black ball and one white ball in a bin. We repeatedly do the following: choose one ball from the bin uniformly at random, and then put the ball back in the bin with another ball of the same color. We repeat until there are n balls in the bin. Show that the number of white balls is equally likely to be any number between 1 and $n-1$.

   (b) Let Y be a nonnegative integer-valued random variable with positive expectation. Prove that
   $$\frac{(\mathbf{E}[Y])^2}{\mathbf{E}[Y^2]} \leq \mathbf{Pr}[Y \neq 0] \leq \mathbf{E}[Y].$$

3. A permutation on the numbers $[n] = \{1, \ldots, n\}$ can be represented as a function $\pi : [n] \to [n]$, where $\pi(i)$ is the position of i in the ordering given by the permutation. A *fixed point* of a permutation $\pi : [n] \to [n]$ is a value for which $\pi(x) = x$. Consider the following random experiment: We pick a permutation uniformly at random from the set of all permutations from $[n]$ to $[n]$. Let F be the random variable representing the number of fixed points of a permutation chosen uniformly at random.

   (a) Find the expectation of F.

   (b) Find the variance of F.

4. We have a function $F : \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, m-1\}$. We know that, for $0 \leq x, y \leq n-1$, $F((x+y) \mod n) = (F(x) + F(y)) \mod m$. The only way we have for evaluating F is to use a lookup table that stores the values of F. Unfortunately, an Evil Adversary has changed the value of $1/5$ of the table entries when we were not looking.

   Describe a simple randomized algorithm that, given an input z, outputs a value that equals $F(z)$ with probability at least $1/2$. Your algorithm should work for every value of z, regardless of what values the Adversary changed. Your algorithm should use as few lookups and as little computation as possible. Suppose you are allowed to repeat your initial algorithm three times. What should you do in this case? Justify your answer.

5. In this problem, we will analyze a simple algorithm to learn an unknown probability distribution from samples.

   A *discrete probability distribution* over the set $[n] = \{1, \ldots, n\}$ can be viewed as a function $p : [n] \to [0, 1]$. The number $p(i)$ represents "the probability the distribution p assigns to point i." Hence, we have that $p(i) \geq 0$ for all $i \in [n]$, and $\sum_{i=1}^{n} p(i) = 1$. For two distributions p, q over $[n]$ the *total variation distance* between p and q is the quantity $\mathrm{d_{TV}}(p, q) := \sum_{i=1}^{n} |p(i) - q(i)|$. ($\mathrm{d_{TV}}(p, q)$ represents a measure of the "closeness" between p and q.)

   In many scenarios we are interested in *learning* an *unknown* probability distribution from *samples*. In more detail, a *learning algorithm* is given access to a *sampling oracle* for p, i.e., a "black-box" with the following property: Every invocation of the oracle (query) yields an output $s \in [n]$ that is a random variable distributed according to p (i.e., $\mathbf{Pr}[s = j] = p(j)$ for all $j \in [n]$) and is independent of all previous outputs. For a given error parameter $0 < \epsilon < 1$, the goal of the learning algorithm is to output a *hypothesis distribution* h over $[n]$ such that with probability at least $2/3$ (over the samples obtained from the oracle) the following condition is satisfied: $\mathrm{d_{TV}}(p, h) \leq \epsilon$.

   Given m independent samples $s_1, \ldots, s_m$, drawn from distribution $p : [n] \to [0, 1]$, the *empirical distribution* $\widehat{p}_m : [n] \to [0, 1]$ is defined as follows: for all $i \in [n]$,

$$\widehat{p}_m(i) = \frac{|\{j \in [m] \mid s_j = i\}|}{m}.$$

   Consider the following algorithm:

   "Draw m samples from the oracle for p and output the distribution $h = \widehat{p}_m$."

   (a) For $i \in [n]$, let $N_i = |\{j \in [m] \mid s_j = i\}|$ denote the number of samples that "land" on point i. Show that $\mathbf{Var}[N_i] = mp(i)\left(1 - p(i)\right)$.

   (b) Show that $\mathbf{E}\left[|p(i) - \widehat{p}_m(i)|\right] \leq \sqrt{\frac{p(i)}{m}}$. Deduce as a consequence that

$$\mathbf{E}\left[\mathrm{d_{TV}}(p, \widehat{p}_m)\right] \leq \sqrt{\frac{n}{m}}.$$

   (Hint: Use (a) along with Jensen's inequality.)

   (c) Show that there exists a constant $C > 0$ such that if $m \geq Cn/\epsilon^2$ the above described algorithm satisfies $\mathrm{d_{TV}}(p, h) \leq \epsilon$ with probability at least $9/10$.