15-859FF: Coping with Intractability **CMU, Fall 2019**
HW #0 (out Monday Aug 26, 2019) **Due:** Friday Sep 6, 2019

This HW is going out a week before classes start so that you can get a feel for the course asap, and prepare accordingly. It's a short HW, with a short deadline (end of week #1 of classes), so that we can get to the real material for the course (and HW1) soon. These problems are solvable using ideas we cover in the first few lectures of our undergraduate Algorithms course, plus a basic probability course. Unless specified otherwise, all algorithms should run in poly-time. We're not asking you to optimize your runtimes, but in general please do so when possible (and reasonable); as algorithm designers, it's a good habit to strive for optimality.

Please solve the (non-exercise) problems without collaboration. You may discuss the exercises with others. Submission details (and corrections) will appear on the webpage and on Piazza, please check it regularly.

Exercises

Exercises are for fun and edification, please do not submit. But please do try to solve them, we may need ideas from there later in the course!

1. **(A Recurring Theme)** Define $T(0) = 0$ and $T(1) = 1$, and

$$T(n) = T(n-1) + T(n-2).$$

Prove that $T(n) \leq O(\phi^n)$ where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. Similarly, if $S(n) = S(n-1) + S(n-3)$, show that $S(n) \leq O(C^n)$, where C is a positive real root of the equation $x^3 = x^2 + 1$.

2. **(Random Questions)** Some of these are a bit non-trivial, but they are fun!

   (a) Given random variables (r.v.s) X, Y, show that $\mathbf{E}[X+Y] = \mathbf{E}[X] + \mathbf{E}[Y]$ and $\mathbf{E}[cX] = c\,\mathbf{E}[X]$ and $\mathbf{Var}(cX) = c^2\,\mathbf{Var}(X)$ for any constant c. If they are independent, then show that $\mathbf{E}[XY] = \mathbf{E}[X]\,\mathbf{E}[Y]$ and $\mathbf{Var}(X+Y) = \mathbf{Var}(X) + \mathbf{Var}(Y)$. Hence show that for independent $X_1, \ldots, X_n$, if each X_i has mean μ and variance σ^2, then $\frac{\sum_{i=1}^n X_i}{n}$ has mean μ and variance σ^2/n.

   (b) Let $x_1, x_2, \ldots, x_n$ be a random permutation of the numbers $[n] := \{1, 2, \ldots, n\}$. You scan the numbers from left to right. You have a buffer B of size b where $1 \leq b \leq n$, initially containing b copies of ∞. When you see x_i, if B contains a number bigger than x_i, then drop the largest number in B, and add in x_i to B. What is the expected total number of numbers you add to B during the scan?

   (c) Given a cycle C_n of length $n \geq 3$ whose nodes are numbered 0 through $n-1$, you start an unbiased random walk at node 0. At each step, when you are at node i, you go to $i+1$ with probability 1/2, and to $i-1$ with probability 1/2. (Both numbers are considered modulo n, of course.) You continue this walk until you visit all the nodes. What is the probability that the last node to be visited is node $n-1$?

   (d) An airplane in Politesville has n seats, and n passengers assigned to these seats. The first passenger to board gets confused, and sits down at a uniformly random seat. The rest of the passengers do the following: when they board, if their assigned seat is free they sit in it, else (being too polite) they choose a uniformly random empty seat and

sit in it. (a) What is the probability that the last person to board sits in their assigned seat? (b) What is the expected number of people who board to find their assigned seat occupied?

(e) You want to sample uniformly at random from the set of all n-bit strings that are *balanced*, i.e., that contain exactly $n/2$ 0's and $n/2$ 1's. You do the following: take a uniform random sample ω from the set of all n-bit strings. output ω if it is balanced, else reject, and sample again. Show that the expected number of times you sample an ω is $O(\sqrt{n})$. (Hint: Stirling's formula.) You should also try to show the expected number of samples is $\leq n+1$, by arguments from first principles.

3. **(Fast Fourier Transform)** Consider polynomials $P(x) = p_d x^d + p_{d-1}x^{d-1} + \ldots + p_1 x + p_0$ of degree at most d over a single variable x, having coefficients p_i in some field $\mathbb{F}$.

(a) One way to specify a degree-d polynomial is to give the $d+1$ coefficients. Another way is to fix some $d+1$ distinct points $a_0, a_1, \ldots, a_d \in \mathbb{F}$, and specify the value $P(a_i)$ of the polynomial at these $d+1$ points. Show that you can move from one representation to the other in $O(d^2)$ time. (Hint: see these notes if you need a reminder of how to do this.) Assume that basic arithmetic operations can be done in constant time.

(b) Given two $d+1$-tuples $\mathbf{p} = (p_0, p_1, \ldots, p_d)$ and $\mathbf{q} = (q_0, q_1, \ldots, q_d)$ in $\mathbb{F}^{d+1}$, define their *convolution* as the $(2d+1)$-tuple $\mathbf{r} = (r_0, r_1, \ldots, r_{2d})$ where

$$r_i := \textstyle\sum_{j=0}^{i} p_j q_{i-j}.$$

Check that you can easily compute the convolution of $\mathbf{p}, \mathbf{q}$ in time $O(d^2)$.

(c) The Fast Fourier Transform (FFT) of Cooley and Tukey uses a divide-and-conquer algorithm to compute convolutions in time $O(d \log d)$. Here are some notes; please read them—or your favorite source on FFTs.

(d) Given $\mathbf{p} \in \mathbb{R}^n$ and $\mathbf{q} \in \mathbb{R}^m$ where $m \leq n$, define the *cross-correlation* $\mathbf{r} \in \mathbb{R}^{n+m-1}$ as

$$r_i := \textstyle\sum_j p_j q_{j+i} \qquad \forall i = 1-m, \ldots, n-1.$$

Use the FFT to compute the cross-correlation in time $O(n \log m)$. *Note that convolution is defined for equal-length strings only!* You can use cross-correlation to solve several kinds of string matching problems fast: see Section 7 of these notes, for example.

Problems

1. **(Balls in Bins)** You throw n balls independently into n bins, with each ball being equally likely to fall into each bin.

(a) (Do not submit.) Let random variable (r.v.) X_i denote the load of bin i, i.e., the number of balls in bin i. Show that $\mathbf{E}[X_i] = 1$ and $\mathbf{Var}[X_i] = \frac{n-1}{n}$.

(b) Use Chebyshev's inequality to show that the load of the most loaded bin is $O(\sqrt{n})$ with probability at least 1/2; i.e., $\mathbf{Pr}[\max_{i \in [n]} X_i \leq c\sqrt{n}] \geq \frac{1}{2}$ for some constant c.

(c) Now use the Chernoff bound below to show that $\max_i X_i$ is at most $O(\log n)$ with probability at least $1 - 1/n^{100}$.

Here is a convenient Chernoff bound: *given independent* $\{0,1\}$*-valued r.v.s* $Y_1, \ldots, Y_m$*, define* $S := \sum_{i=1}^m Y_i$ *with* $\mu := \mathbf{E}[S]$*. Then for any* $\lambda \geq 0$*,*

$$\mathbf{Pr}[S \geq \mu + \lambda] \leq \exp(-\lambda^2/(2\mu + \lambda)).$$

Please be sure to verify the independence of the random variables you apply the bound to!

2. **(If You Cut Us...)** Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E, where each edge has a positive weight w_e. Let $|V| = n$. The input is a graph and an integer $K \in [0, n]$. A $(K, n-K)$*-partition* of G is a coloring of the vertices with two colors (red/blue) such that K nodes are colored red, and the remaining $n - K$ nodes are colored blue. Our goal is to find a $(K, n-K)$-partition where the total weight of *split* edges (edges with one endpoint red and the other blue) is minimized.

   (a) Design an $O(n^3)$ time algorithm to solve this problem when G is a binary tree. Prove the correctness of your algorithm and analyze its running time. Formally, a binary tree—for our purposes—is a tree rooted at some node r, and each node has at most two children.

   (b) Extend this algorithm to work for all trees, not just binary trees. (We know of at least two ways of doing this.) Again, the runtime should be $O(n^3)$, or $O(n^4)$ at the most.

3. **(Finding a Triangle)** Assume you have an algorithm $\mathcal{A}$ for matrix multiplication, that given two $n \times n$ matrices A, B, computes the product $C = AB$ in time $O(n^\omega)$.[1] For a graph $G = (V, E)$, a *triangle* in G is a set of three vertices $\{u, v, w\}$ such that all three edges $\{u, v\}, \{u, w\}, \{v, w\}$ belong to E.

   Show how to find a triangle in a graph G with $|V| = n$ (or report that none exists) in time $O(n^\omega)$.

[1] Observe that the naive implementation of matrix multiplication gives $\omega = 3$. A famous breakthrough of Strassen's gave $\omega = \log_2 7 = 2.8$, and the current best value known is $\omega = 2.372\ldots$. The only lower bound is $\omega \geq 2$, which comes from the fact that reading the input itself requires $Omega(n^2)$ time.