A Theorist's Toolkit **CMU 15-751*, Fall 2016**

PROBLEM SET 3

Due: Noon, Monday Oct. 3, email the pdf to toolkit2016homework@gmail.com

---

Homework policy: Exactly the same as last time.

Note: In this homework you may assume that the following basic operations on n-bit integers can be performed in poly(n) time: addition, subtraction, multiplication, and integer-division (which produces the integer quotient and remainder).

---

1. (a) Consider the following algorithmic task:

   Input: a rational number $0 \le x < 4$, written in base 2, with n bits of precision.

   Desired output: a rational number y such that $|y - \sin x| \le 2^{-n}$.

   Show that this problem can be solved in poly(n) time. (Your algorithm should be presented in pseudocode and/or English, with the RAM model in mind. You should use the Taylor series for $\sin x$, which you can look up if you forget it.)

   (b) Show that a rational approximation $\widehat{\pi}$ of the number π satisfying $|\widehat{\pi} - \pi| \le 2^{-n}$ can be computed in poly(n) time. For this you should use the following facts: π is the unique solution of $\sin x = 0$ for $x \in [2, 4]$; and, $\sin x$ is a decreasing function on the interval $[2, 4]$.

2. The Gaussian, $\boldsymbol{Z} \sim N(0, 1)$, is probably the greatest random variable, but another great random variable is the *exponential* random variable. We say that $\boldsymbol{S}$ is an *exponential random variable of mean* 1, written $\boldsymbol{S} \sim \text{Exp}(1)$, if $\boldsymbol{S}$ has probability density function

$$f(s) = \begin{cases} e^{-s} & \text{if } s \ge 0, \\ 0 & \text{if } s < 0. \end{cases}$$

   We trust you can verify for yourself that $f(s)$ really is a valid pdf (i.e., $\int_0^\infty e^{-s} ds = 1$) and that $\mathbf{E}[\boldsymbol{S}] = 1$ (i.e., $\int_0^\infty s e^{-s} dy = 1$).

   (a) Suppose $\boldsymbol{U}$ is a uniformly random real in the interval $(0, 1]$. Let $\boldsymbol{S} = \ln(1/\boldsymbol{U})$. Show that $\boldsymbol{S} \sim \text{Exp}(1)$. (Hint: analyze $\mathbf{Pr}[\boldsymbol{S} \le t]$ for all $t \in \mathbb{R}$.)

   (b) Consider the following "algorithm":

   - Let $\boldsymbol{U}_1, \boldsymbol{U}_2$ be independent uniformly random reals in $(0, 1]$.
   - Let $\boldsymbol{\theta} = 2\pi \cdot \boldsymbol{U}_1$.
   - Let $\boldsymbol{S} = \ln(1/\boldsymbol{U}_2)$ and let $\boldsymbol{R} = \sqrt{2\boldsymbol{S}}$.
   - Let $\boldsymbol{X} = \boldsymbol{R}\cos\boldsymbol{\theta}$ and $\boldsymbol{Y} = \boldsymbol{R}\sin\boldsymbol{\theta}$.

   Show that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent standard Gaussians.

   (c) Let $\Delta \subset \mathbb{R}^n$ be the "standard simplex"; i.e., the set of vectors $w = (w_1, \ldots, w_n)$ with nonnegative coordinates adding up to 1. We wish to pick a uniformly random vector $\boldsymbol{W}$ in Δ. Show that the following method does the job:

   - Pick $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n \sim \text{Exp}(1)$, all independent.

- Let $\boldsymbol{L} = \boldsymbol{Y}_1 + \cdots + \boldsymbol{Y}_n$.
- Output $\boldsymbol{W} = \frac{1}{\boldsymbol{L}}(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n)$.

(Remark: In Wednesday's lecture, we came pretty close to describing a similar algorithm for choosing a uniformly random vector on the unit sphere in $\mathbb{R}^n$. The level of mathematical rigor you use for this problem should be about the same as was used in that lecture.)

3. Consider the fundamental algorithm underlying all of Linear Algebra: *Gaussian Elimination*. Recall that the goal of Gaussian Elimination, given an $m \times n$ input matrix A, is to "reduce" A to an upper-triangular matrix U by performing elementary row operations (i.e., adding multiples of one row to another row). As you'll recall from your Linear Algebra class, this algorithm generates a sequence of matrices $A_0 = A$, A_1, $\ldots$, A_r, where A_k is of the form

$$A_k = \begin{bmatrix} B & C \\ 0 & D \end{bmatrix},$$

where B is a $k \times k$ upper-triangular, nonsingular matrix. To obtain A_{k+1}, we:

- Choose a nonzero element of D (the "pivot"). Permute the rows and columns of the matrix such that the pivot moves to D_{11}.
- "Clear out" all entries in the first column of D (except D_{11}) by adding an appropriate multiple of the first row of D to make the entries 0.

The procedure proceeds until we get an A_r of the form

$$A_r = \begin{bmatrix} B & C \\ 0 & 0 \end{bmatrix}.$$

Here r is the rank of A.

Let's henceforth assume $m = n$; i.e., A is square. Since all of the elementary row operations correspond to multiplication by a (pretty simple) lower-triangular matrix, this procedure factorizes A as $A = PLUQ$, where L is lower-triangular, U is upper-triangular, and P and Q are permutation matrices. If A is nonsingular (i.e., $r = n$) then it's easy to see you only need to do column permutations and therefore one gets a factorization $A = PLU$.

This problem asks you to analyze the computational complexity of Gaussian Elimination in the Word RAM model. Assume that the initial $n \times n$ matrix A contains integer entries (positive or negative) that fit into a single w-bit word. You may also assume for simplicity that A is nonsingular and that you never need to do any permutations (i.e., that the diagonal entries encountered are always nonzero). You may also forget about keeping track of L and only worry about producing the final matrix $U \in \mathbb{Q}^{n \times n}$. (For example, perhaps you only want to know the determinant of A, which would be the product of U's diagonal elements.)

Assuming all this, you should give a careful proof that the algorithm can be carried out in poly(n) time. (Hint: the main difficulty in this problem is reasoning about the representation size of the rational numbers occurring throughout the algorithm. Try to show that each such number is the ratio of two determinants of submatrices of the original A.)

Solve 2 out of the following 3 problems.

4. Consider the following task: The input is the $n \times n$ adjacency matrix of a directed graph G on vertex set $[n]$. The task is to determine if there is a path from vertex #1 to vertex #n. Show that there is a circuit (family) solving this problem which uses $\text{poly}(n)$ size (specifically, $\text{poly}(n)$ AND/OR gates of arbitrary fan-in) and $O(\log n)$ depth. (Hint: first try to determine if there is a length-2 path from vertex #1 to vertex #n.)

5. The 3SUM problem is the following: The input consists of three length-n arrays A, B, C, each storing an integer between $-n^3$ and n^3. The task is to find three indices $i, j, k \in [n]$ such that $A[i] + B[j] + C[k] = 0$, or correctly report that no such indices exist. Show that in the Word RAM model (with word size $w = \Theta(\log n)$), this problem can be solved in $O(n^2)$ time.

6. Consider the Word RAM model. Suppose we wish to "allocate" and use an "array" of n words. An underspecified aspect of the Word RAM model is whether we can assume that all memory words are initially set to 0^w (the w-bit word of all zeros), or whether — as in many real computers — memory words are initialized to unknown "junk" strings in $\{0,1\}^w$. The point of this problem is to show that it doesn't really matter.

   More precisely, develop a "length-n array" data structure that works even in the "junk" model. You may assume $w = \log_2 n$. Your data structure should support three operations:

   - **Initialize**: Initializes the data structure.
   - **Read**(i) (for $0 \le i < n$): If the ith word of the array has previously been written to, return that value. Otherwise, return "uninitialized".
   - **Write**(i, x) (for $0 \le i < n$, $0 \le x < n$): Write x into the ith word of the array.

   All three operations must take $O(1)$ time in the worst case. Furthermore, your algorithm must use at most $O(n)$ space.