15-750: Algorithms in the Real World **CMU, Spring 2023**
HW #3 (out Tuesday Feb 21, 2023) **Due:** Tuesday March 14, 2023

Please limit yourself to groups of two (or three) people. We adhere to a "whiteboard" collaboration policy: you should discuss in your group on whiteboards (or equivalents) but then should go away and write down your solutions yourself. You must not share written work. **You must write down the names of the person(s) with whom you collaborate.** In general, you should solve the homework problems using only to material we refer you to (or put on the course page), and not books or other online resources. If you have reason to use any resources we point you to (except the course notes), please cite them.

A word of advice: please try to solve the problem by yourself before working with your group. That is the way to improve your problem-solving skills.

Homeworks will be due at 11:59pm on the due date on `gradescope`. Corrections and changes will appear on the course webpage and on Piazza; please check them regularly.

Problems

1. **(A Simple Pairwise Uniform Hash Family.)** Recall that a family $\mathcal{H}$ of hash functions $\{0, 1, \ldots, \mathcal{U}-1\} \to \{0, 1, \ldots, n-1\}$ is said to be *pairwise uniform* (or *pairwise independent*) if, for any two distinct items $0 \le x \ne y < \mathcal{U}$, when you choose $h \in \mathcal{H}$ uniformly at random, the pair $(h(x), h(y))$ is equally likely to be any of the n^2 possibilities.

   Assume $\mathcal{U} = 2^u$ and $n = 2^\ell$ for some integers $u, \ell > 0$. Here is one way to get a pairwise uniform family in which each hash function is named using only $O(u + \ell)$ bits.

   For a string $a \in \{0,1\}^{\ell+u-1}$, let M_a denote the $\ell \times u$ matrix formed as follows: Start with M_a empty. Consider the path that travels up the first column of M_a (bottom to top), then takes a right-turn at the top-left entry and travels all along the first row. Fill these entries of M_a with the string a (in that order). Then, fill in each remaining empty entry as follows: travel "northwest" (diagonally up-left) until you hit an already-filled-in-entry, and copy that. For example, if $\ell = 3, u = 5$ and $a = 0110110$ then M_a would be

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

   Finally, for $b \in \{0,1\}^\ell$, define the hash function $h_{a,b}$ by

$$h_{a,b}(x) = Mx + b \pmod 2,$$

   where x is treated as a base-2 length-u bitstring.

   Prove that $\mathcal{H} = \{h_{a,b} : a \in \{0,1\}^{\ell+u-1},\ b \in \{0,1\}^\ell\}$ is pairwise uniform.

2. **Faster Approximate Matrix Multiplication By "Preconditioning". Part 1.** Matrix multiplication is a very common primitive, but is an expensive operation. It turns out that we can do these multiplications faster if we are happy with approximate answers! We will now use the randomized dimension reduction procedure (the "JL Lemma") to do this. In a nutshell, we want to show that the random $K \times D$ matrix S used in the JL Lemma is very likely to ensure

$$AB \approx AS^\intercal SB \tag{1}$$

and moreover that the product on the right can be computed much faster than the one on the left (because S has few rows). The reason (1) will be true is: matrix multiplications can be viewed as a whole bunch of dot products, so if we show that the JL Reduction approximately maintains dot products, we are in good shape.

You may use the following version of the JL Lemma, which we mostly proved in class:

Theorem 1 (JL Lemma.). *Let C be a set of n vectors in $\mathbb{R}^D$. For some $K = O(\frac{\log(n/\delta)}{\varepsilon^2})$, the following holds: Suppose we pick a random $K \times D$ matrix where each entry is independently ± 1 with probability $1/2$ each, then we multiply this matrix by $1/\sqrt{K}$ and call the result S. Then*

$$\mathbf{Pr}[S \text{ ``}\varepsilon\text{-preserves'' the length of } x \text{ for all } x \in C] \geq 1 - \delta.$$

Here we say that S ``ε-preserves the length of x'' if $(1-\varepsilon)\|x\| \leq \|Sx\| \leq (1+\varepsilon)\|x\|$.

(a) Suppose that in the setting of the above Theorem, we also wanted to ε-preserve all pairwise distances (so $(1-\varepsilon)\|x-y\| \leq \|Sx - Sy\| \leq (1+\varepsilon)\|x-y\|$ for all pairs $x, y \in C$). Why are basically still okay ("up to constant factors")? What if we wanted to also ε-preserve the lengths of $x + y$ for all pairs $x, y \in C$? What if we also wanted to ε-preserve the length of $\frac{x}{\|x\|} \pm \frac{y}{\|y\|}$ for all pairs $x, y \in C$?

(b) Suppose $u, v \in \mathbb{R}^D$ are unit vectors ($\|u\| = \|v\| = 1$), and that the lengths of $u, v, u+v, u-v$ are all ε-preserved by matrix S. Show that their dot product is also $O(\varepsilon)$-preserved; i.e., for some constant c we have

$$u \cdot v - c\varepsilon \leq (Su) \cdot (Sv) \leq u \cdot v + c\varepsilon.$$

(You might like to remember some basic tricks like $\|w\|^2 = w \cdot w$ and $u \cdot v = u^\intercal v$ when u, v are column vectors.)

3. **Faster Approximate Matrix Multiplication By "Preconditioning". Part 2.**

(a) Let A be a matrix with D columns and at most n rows. Let B be a matrix with D rows and at most n columns. Note that the ij-entry of AB is the dot product between two D-dimensional vectors (not necessarily unit-length). Show that for any fixed pair i, j and for $K = O(\log(n/\delta)/\varepsilon^2)$, when we pick S at random,

$$\mathbf{Pr}[|(AB)_{ij} - (AS^\intercal SB)_{ij}| \leq \varepsilon \|A_{i\star}\| \|B_{\star j}\|] \geq 1 - \delta,$$

where $A_{i\star}$ (respectively, $B_{\star j}$) is the ith row of A (respectively, jth column of B).

(b) Complete the argument by showing that

$$\mathbf{Pr}\left[\|AB - AS^\intercal SB\|_F \leq \varepsilon \|A\|_F \|B\|_F\right] \geq 1 - \delta.$$

where $\|M\|_F$ denotes the "Frobenius norm" of matrix M, defined by $\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$.

(c) Assuming we use the standard basic matrix multiplication method, what is the runtime of computing AB versus $AS^\intercal SB$? Take note of why the improvement is considerable if $n, D \gg K$.

4. **(Discovering A Projection Matrix.)** Let A be a $D \times D$ symmetric matrix. We say that A is a *projection matrix* if all of its eigenvalues are either 0 or 1. We further say it is a *rank-K projection matrix* if it has exactly K eigenvalues that are 1. As you may convince yourself, this geometrically means that there is some certain K-dimensional subspace, and A's action on a vector v is to project it into that subspace.

   Given as input a $D \times D$ symmetric matrix A that happens to be a rank-K projection matrix, describe and analyze an $O(KD^2)$-time algorithm for discovering an orthonormal basis of the relevant K-dimensional subspace (in other words, some K orthogonal unit eigenvectors of A with corresponding eigenvalues equal to 1). Your algorithm should be able to do this without knowing what K is (but ignore the case $K = 0$).

   (Don't worry about "numerical accuracy" at all in this problem; just assume that you can do basic mathematical operations on numbers (like computing square-roots, or testing if a number is 0) in time $O(1)$. Also, you may assume that: (i) you can pick a Gaussian random variable in time $O(1)$; (ii) if you pick D Gaussians independently and stack them into a vector u, then u is "rotationally symmetric"; (iii) in particular, if you fix in advance any particular subspace H of dimension less than D, and then you choose u, the probability that u lands exactly in H is literally 0.)