CPSC 536N: Algorithms That Matter (Term 2, 2016-17)
Assignment 2
Due: $\leq (1+\epsilon) \cdot$ (Monday, April 3rd)

Your solution must be written up in LaTeX. You may work in groups of at most 3.

Question 1: In the lecture on Count-Sketch, we discussed k-sparse recovery. Let the input vector to Count-Sketch be x. We showed that, with probability $\geq 1 - 1/n$, we can reconstruct a vector $\tilde{x}$ such that

$$\|\tilde{x} - x\|_\infty \leq \frac{\epsilon}{\sqrt{k}} \cdot \mathrm{err}_2^k(x).$$

This requires using a Count-Sketch table with $w = 3k/\epsilon^2$ columns and $O(\log n)$ rows. In particular, if x is k-sparse, then $\mathrm{err}_2^k(x) = 0$.

In our application to ℓ_0-sampling, we required a slightly different guarantee — we need to be able to detect if x is k-sparse or not.

Design a modification to Count-Sketch so that, with probability $\geq 1 - 1/n$,

- if x is k-sparse then the output is x,
- if x is not k-sparse then the output is "DENSE".

As before, you should use a Count-Sketch table that takes $O(k)$ columns and $O(\log n)$ rows.

Question 2: In class we saw an LSH scheme for the c-Approximate Nearest Neighbor problem for n binary strings of length d under the Hamming distance. The scheme was based on a reduction to the (c, r)-Near Neighbor[1] problem: basically we run $O(\log_c \Delta)$ copies of the Near Neighbor data structure, then we do binary search to find the Nearest Neighbor.

It turns out that there is a neat trick that can directly solve the c-Approximate Nearest Neighbor problem *without* using a reduction to the (c, r)-Near Neighbor problem. The trick is based on *random permutations of the coordinates*, and is sketched the paper Similarity Estimation Techniques From Rounding Algorithms by Moses Charikar. Describe and analyze the scheme in your own words.

Question 3: Consider the following matching problem (that has both a streaming and online flavor to it). The vertices are known ahead of time (not that it matters much). At each time step, an edge e arrives, together with a positive weight w_e for that edge. We assume that every edge weight *is a power of 2*. The goal is to construct a matching of large weight.

(Unlike the online matching results we saw in class, we will not assume that the graph is bipartite, but this won't make much difference.)

Consider the following algorithm. Initially M is empty. When a edge e arrives, check if w_e is *strictly greater* than the total weight of edges currently in M that share an endpoint with e (if any). If so, remove those edges from M and replace them with e.

Let ALG be the weight of the matching produces by the algorithm, and let OPT be the weight of the maximum weight matching. Prove that $ALG \geq \frac{1}{c} \cdot OPT$ for some positive constant c. How small can you make c?

[1] In older papers, this is called the PLEB problem.

Question 4: QTWIDKTA (Question To Which I Don't Know The Answer)

Let us consider the competitive ratio of the RANKING algorithm for online bipartite matching, as a function of $n = |S| = |T|$. For simplicity, assume that the graph has a perfect matching.

- The Birnbaum-Mathieu paper claims that RANKING has a competitive ratio of $\geq 1 - (1 - \frac{1}{n+1})^n$. This is slightly *worse* than $1 - 1/e$.
- The Karp-Vazirani-Vazirani and Goel-Mehta papers claim that RANKING has a competitive ratio of $\geq 1 - (1 - \frac{1}{n})^n$ (I think). This is slightly *better* than $1 - 1/e$.
- The Devanur-Jain-Kleinberg paper claims that RANKING has a competitive ratio of $\geq 1 - 1/e$ exactly.

Can you tweak the analysis of the Devanur-Jain-Kleinberg paper so that, for each finite n, it achieves a competitive ratio $c_n > (1 - 1/e)$? (Say, $c_n = 1 - (1 - \frac{1}{n})^n$?) Of course, we must have $\lim_{n \to \infty} c_n = 1 - 1/e$.