Due: March 28th (Thursday), 6pm, submit to courseworks.

You may use "standard" arguments without a formal proof—either cite from class or add one sentence showing the main idea.

Problem 1: ℓ_1 LSH with constant time and large space

Prove that the lower bound we saw in Lecture 7 is tight, i.e., that there is a (randomized) DS for the *decision* version of $(1+\epsilon, r)$-ANN under the ℓ_1 norm over $\{0,1\}^d$, with space $s = n^{O(1/\epsilon^2)}$, and query time $t = O(1)$.

(Hint: Instead of just projecting to k random coordinates $\in [d]$ as we saw in class, consider the hash function that samples k independent n-bit strings $z_i \in_R \{0,1\}^d$, each distributed as $z_i \sim Bin(d, 1/2r)$ (i.e., each coordinate of z_i is 1 w.p $1/2r$ and 0 otherwise), and then maps each point $x \in \{0,1\}^d$ to the k inner-products (mod 2) of x with the z_i's, in other words, to the vector $(\langle x, z_1\rangle, \ldots, \langle x, z_k\rangle) \in \{0,1\}^k$. Show that, for $k = O(\lg n/\epsilon^2)$, this is a valid LSH function for $(1+\epsilon, r)$-ANN, similar to the DS we saw in class. You may use without a proof the Chernoff bound: $\Pr[\sum_{i=1}^k X_i \notin (1 \pm \beta)k\mu] \leq e^{-\Omega(k\mu\beta^2)}$ for i.i.d boolean random variables X_i with $E[X_i] = \mu$).

Problem 2: Dynamization with deletions for decomposable problems

P1) Recall that we saw in class how to dynamize any static ANN data structure in a *black-box*, with only $O(\lg n)$ overhead in query and (amortized) update time. However, this worked only for *insertion* of points. Why does this method fail for deletions ?

P2) Show that any static DS for a decomposable search problem (like ANN) with preprocessing time (and hence space) bounded by $p(n)$ and query time $t(n)$, can be *fully dynamized* (i.e., support both insetion and deletions) with worst-case time $t_u = \tilde{O}(\sqrt{n} \cdot p(n)/n)$, and query time $t_q = \tilde{O}(\sqrt{n} \cdot t(n))$.

(Hint: Show to to "fix" the exponential block construction from class using larger blocks. Make use of an additional Dictionary data structure that maintains information about which update element is stored in which block. Note that if we have blocks of size B, then rebuilding a block costs $p(B)$, and querying all blocks takes $O(t(n) \cdot (n/B)))$.

Problem 3: Asymmetric CC of Lopsided Set Disjointness

Complete the proof from class, showing that, for every $k = O(1), \delta > 0$ and large enough n, it holds that $ACC(LSD(k,n)) \geq (\Omega(\delta k \lg(n), n^{1-2\delta}))$. Use the Richness Lemma proved in class.

Problem 4: Final Project Proposal

Write a $<$ 1-page project proposal for your final project (either research-based or implementation-based, as discussed in the initial course guidelines and in the course webpage), in which you concretely describe the scope, goals and questions you wish to study or

experiment/implement in your project. You can consult the list of suggested related research projects posted on the course website. We encourage you to be creative and inquisitive – if you can think about generalizations, interesting connections between DS we saw in class, or new techniques and applications, feel free to suggest concrete ideas. Your proposal should convey the impression that you are well prepared and converged on your project, but deliberation between up to 2-3 topics is OK. The TA and Instructor are available for appointments, should you need to consult us.

If you wish to collaborate in pairs for your final project, you must clearly explain the reason you think your project is of larger scope/effort/ambition. For implementation projects, depending on scope, we will allow up to 3 collaborators per project.