Sublinear Time and Space Algorithms 2022B – Problem Set 3

Robert Krauthgamer

Due: June 20, 2022

General instructions: Please keep your answers short and easy to read. You can use results, calculations or notation seen in class without repeating them, unless asked explicitly to redo them.

1. Using the notation seen in class for Euclidean MST of $P \subset [\Delta]^d$, prove that $\mathrm{MST}_T(P) \geq \frac{1}{2}\mathrm{MST}(P)$. (Assume here that the quadtree is fixed and not randomly shifted.)

2. Consider the frequency-vector model, where the stream contains additive updates to a vector $x \in \mathbb{R}^n$ whose coordinates are integers bounded by $\mathrm{poly}(n)$.

   Explain how to $(1+\epsilon)$-approximate $\sum_{i<j}(x_i + x_j)^2$ by a streaming algorithm with storage requirement $(\epsilon^{-1}\log n)^{O(1)}$ bits.

   Remark: As done in class, do not count storage of the algorithm's random coins.

3. Let $y \in \mathbb{R}^n$ be the frequency vector of an input stream in the turnstile model (i.e., allowing insertions and deletions), and suppose its coordinates are integers in the range $[-n^2, n^2]$.

   Design a linear sketch that detects whether $|\,\mathrm{supp}(y)| = 1$ using storage requirement of $O(1)$ words (i.e., $O(\log n)$ bits), not counting storage of the algorithm's random coins. Its success probability should be at least $1 - 1/n$.

   Hint: Use a variant of the AMS sketch with large random coefficients.

Extra credit:

4. Design a streaming algorithm for the bichromatic matching problem (aka earthmover distance), where the input is a set of colored points $P \subset [\Delta]^d$, half of them are blue and half are red, i.e., $P = R \cup B$, and the goal is to compute a minimum-weight perfect matching between R and B.

   Hint: Use a randomly shifted quadtree (as seen in class), and for each level i estimate $\|x^{(i)}\|_1$.

   For simplicity, assume a randomized streaming algorithm that $(1+\varepsilon)$-approximates the ℓ_1 norm (of the frequency vector x), for $\varepsilon = 0.1$, using storage $s(n) = \mathrm{polylog}(n)$. Such algorithms are known, although we did not see it in class.