Sublinear Time and Space Algorithms 2016B – Lecture 9
More Lower Bounds and Algorithms for Sequences*

Robert Krauthgamer

1 Set Disjointness and Approximating ℓ_∞-norm

Problem definition: The inputs are $x, y \in \{0,1\}^n$ and the goal is to determine whether the cardinality of $\{i \in [n] : x_i = y_i = 1\}$ is one or zero.

We can view x, y as subsets of $[n]$, and the goal is to decide if the two sets intersect (exactly once) or are disjoint. This is sometimes called the unique intersection property.

Theorem 1 [Kalyanasundaram and Schnitger, 1992] and [Razborov, 1992]: The communication complexity (with unbounded number of rounds) of Set Disjointness in $\{0,1\}^n$ is $\Omega(n)$, even with shared randomness.

Stated without proof.

Corollary 2: Every randomized streaming algorithm that approximates ℓ_∞-norm in $\mathbb{R}^n$ within factor 2.99 requires $\Omega(n)$ bits.

Proof: Was seen in class.

2 Multiparty Disjointness and ℓ_p-norm

Problem definition: There are t players, with respective inputs $x^{(1)}, \ldots, x^{(t)} \in \{0,1\}^n$ and the goal is to determine whether

- for all $i \neq j$, $\{i \in [n] : x^{(i)} = x^{(j)} = 1\} = \emptyset$; or
- there is $k \in [n]$ such that for all $i \neq j$, $\{i \in [n] : x_i \wedge y_i = 1\} = \{k\}$.

(It may be easier to think of it as set intersection $\left|x^{(i)} \wedge x^{(j)}\right|$.)

We usually consider the model where all messages are written on a blackboard that is seen by all players (equivalently, it is broadcasted to all players without counting it n times).

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

Theorem 3 [Gronemeier, 2009], following [Bar-Yossef, Jayram, Kumar and Sivakumar, 2002] and [Chakrabarti, Khot and Sun, 2003]: The communication complexity (with unbounded number of rounds) of t-party Set Disjointness in $\{0,1\}^n$ is $\Omega(n/t)$, even with shared randomness.

Stated without proof.

Remarks:
(a) It follows that at least one player has to send $\Omega(n/t^2)$ bits.
(b) The bound holds even in the one-way model, where the messages go first from Player 1 to 2, then from Player 2 to 3, and so forth.

Corollary 4: Every streaming algorithm that 2-approximates the ℓ_p-norm, for $p > 2$, in $\mathbb{R}^n$, requires $\Omega(n^{1-2/p})$ bits of storage.

Remark: Holds even for insertions-only streams.

Proof: Was seen in class.

3 Application 3 of point queries: Range Queries

Problem Definition: Let $x \in \mathbb{R}^n$ be the frequency vector of the input stream, and let $\varepsilon \in (0,1)$ be a parameter known in advance.

Given a range query $[i,j]$ (where $i,j \in [n]$), report an estimate for $\sum_{l=i}^{j} x_l$ that with high (constant) probability is within additive error $\varepsilon \|x\|_1$.

Observe there are $O(n^2)$ possible queries, which include the n possible point queries. The challenge is to avoid accumulation of error from the different coordinates.

Theorem: There is a randomized streaming algorithm for ℓ_1 range queries in $\mathbb{R}^n$ that has storage requirement of $O(\text{poly}(\varepsilon^{-1} \log n))$ words.

Proof sketch: Was seen in class.

4 Streaming Algorithms for LIS

Problem Definition: The input is a stream of numbers/letters (e.g., a string) of length n. The goal is to compute the length of the longest increasing subsequence (LIS).

Observation: the LIS is easily computed by the Patience Sorting algorithm, which is just dynamic programming, where each table entry $P(i)$ stores the smallest letter a such that there is an increasing subsequence of length i ending with a.

Theorem 5 [Gopalan-Jayram-Kumar-K.'07]: Every randomized streaming algorithm for computing LIS requires storage $\Omega(n)$.

Proof: Was seen in class.

Theorem 6 [Gopalan-Jayram-Kumar-K.'07]: There is a deterministic streaming algorithm that computes $(1+\varepsilon)$-approximation for the LIS of a stream using storage of $O(\sqrt{n/\varepsilon})$ words.

Proof: Was seen in class.

Remark: The $O(\sqrt{n})$ storage is optimal for deterministic algorithms [Gal-Gopalan'07, Ergun-Jowhari'08] but it is open whether randomized algorithms can do better.

Exer: Show that a similar argument works to estimate the distance to monotonicity, i.e., the minimum number of letter deletions that make the stream increasing, which can be described as $n - \text{LIS}$.

In fact, for this problem the best algorithm known is deterministic with approximation $1+\varepsilon$ using polylogarithmic space.

Theorem 7 [Naumovitz-Saks'15]: There is a deterministic streaming algorithm that computes $(1+\varepsilon)$-approximation for the distance to monotonicity using storage of $O(\varepsilon^{-2}\log^5 n)$ words.

5 Testing Monotonicity

Problem definition:

Input: A list $\vec{x} = (x_1, \ldots, x_n)$ of numbers.

Definition: A list is called *monotone* if it is increasing (WLOG strictly). It is called ε-close to (being) monotone if its distance to monotonicity is at most εn (i.e., it can be made monotone by deleting at most ε-fraction of the entries). Otherwise, it is called ε-far from monotone.

Goal: Determine whether $\vec{x}$ is monotone or ε-far from monotone (called testing).

Theorem 8 [Ergun-Kannan-Kumar-Rubinfeld-Viswanathan'98]: There is a randomized algorithm that tests whether an input list is monotone (i.e., determines WHP whether the list is monotone or ε-far from monotone) and runs in time $O(\frac{1}{\varepsilon}\log n)$.

Main idea: do binary search for a random element contained in the list

Algorithm TestMonotonicity:

1. Repeat the following $2/\varepsilon$ times: Choose a random index $i \in [n]$ and perform a binary search in $\vec{x}$ for the value x_i.

2. Accept if all binary searches succeed (find x_i and the search path contains no out-of-order pair). Otherwise, reject.

Analysis: Was seen in class.

Exer: Is there an ε-far (but not 2ε-far) input on which the binary search fails with probability $\gg \varepsilon$?

Exer: Is there an ε-far list for which a single iteration fails with probability $O(\varepsilon)$ (meaning the analysis cannot be improved).