Time lower bounds for nonadaptive turnstile streaming algorithms

Kasper Green Larsen[*] Jelani Nelson[†] Huy L. Nguyễn[‡]

July 8, 2014

Abstract

We say a turnstile streaming algorithm is *non-adaptive* if, during updates, the memory cells written and read depend only on the index being updated and random coins tossed at the beginning of the stream (and not on the memory contents of the algorithm). Memory cells read during *queries* may be decided upon adaptively. All known turnstile streaming algorithms in the literature are non-adaptive.

We prove the first non-trivial update time lower bounds for both randomized and deterministic turnstile streaming algorithms, which hold when the algorithms are non-adaptive. While there has been abundant success in proving space lower bounds, there have been no non-trivial update time lower bounds in the turnstile model. Our lower bounds hold against classically studied problems such as heavy hitters, point query, entropy estimation, and moment estimation. In some cases of deterministic algorithms, our lower bounds nearly match known upper bounds.

1 Introduction

In the turnstile streaming model of computation [Mut05] there is some vector $v \in \mathbb{R}^n$ initialized to $\vec{0}$, and we must provide a data structure that processes coordinate-wise updates to v. An update of the form (i, Δ) causes the change $v_i \leftarrow v_i + \Delta$, where $\Delta \in \{-M, \ldots, M\}$. Occasionally our data structure must answer queries for some function of v. In many applications n is extremely large, and thus it is desirable to provide a data structure with space consumption much less than n, e.g. polylogarithmic in n. For example, n may be the number of valid IP addresses ($n = 2^{128}$ in IPv6), and v_i may be the number of packets sent from source IP address i on a particular link. A query may then ask for the support size of v (the "distinct elements" problem [FM85]), which was used for example to estimate the spread of the Code Red worm after filtering the packet stream based on the worm's signature [EVF06, Moo01]. Another query may be the ℓ_2 norm of v [AMS99], which was used by AT&T as part of a packet monitoring system [KSZC03, TZ12]. In some examples there is more than one possible query to be asked; in "point query" problems a query is some $i \in [n]$ and the data structure must output v_i up to some additive error, e.g. $\varepsilon\|v\|_p$ [CCFC04, CM05]. Such point query data structures are used as subroutines in the heavy hitters problem, where informally the

[*]Aarhus University. `larsen@cs.au.dk`. Supported by Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84.

[†]Harvard University. `minilek@seas.harvard.edu`. Supported by NSF CAREER award CCF-1350670.

[‡]Princeton University. `hlnguyen@cs.princeton.edu`. Supported in part by NSF CCF-0832797, an IBM Ph.D. fellowship and a Siebel Scholarship.

goal is to output all i such that v_i is "large". If the data structure is linearly composable (meaning that data structures for v and v' can be combined to form a data structure for $v - v'$), heavy hitters data structures can be used for example to detect trending topics in search engine query streams [Gooa, Goob, CCFC04]. In fact the point query structure [CCFC04] has been implemented in the log analysis language Sawzall at Google [PDGQ05].

Coupled with the great success in providing small-space data structures for various turnstile streaming problems has been a great amount of progress in proving space lower bounds, i.e. theorems which state that *any* data structure for some particular turnstile streaming problem must use space (in bits) above some lower bound. For example, tight or nearly tight space lower bounds are known for the distinct elements problem [AMS99, Woo04, Woo07, JKS08], ℓ_p norm estimation [AMS99, BYJKS04, CKS03, Woo04, KNW10a, Gro09, Jay09, JW13, Woo07, JKS08], heavy hitters [JST11], entropy estimation [CCM10, KNW10a], and several other problems.

While there has been much previous work on understanding the space required for solving various streaming problems, much less progress has been made regarding update time complexity: the time it takes to process an update in the stream. This is despite strong motivation, since in several applications the data stream may be updated at an extremely fast rate; for example, [TZ12] reported in their application for ℓ_2-norm estimation that their system was constrained to spend only 130 nanoseconds per packet to keep up with high network speeds. Of course, without any space constraint fast update time is trivial: store v in memory explicitly and spend constant time to add Δ to v_i after each update. Thus an interesting data structural issues arises: how can we simultaneously achieve small space and low update time for turnstile streaming data structures? Surprisingly, very little is understood about this question for any turnstile streaming problem. For some problems we have very fast algorithms (e.g. constant update time for distinct elements [KNW10b], and also for ℓ_2 estimation [TZ12]), whereas for others we do not (e.g. super-constant time for ℓ_p estimation for $p \neq 2$ [KNPW11] and heavy hitters problems [CCFC04, CM05]), and we do not have proofs precluding the possibility that a fast algorithm exists. Indeed, the only previous update time lower bounds for streaming problems are those of Clifford and Jalsenius [CJ11] and Clifford, Jalsenius, and Sach [CJS13] for streaming multiplication and streaming Hamming distance computation, respectively. These problems are significantly harder than e.g. ℓ_p estimation (the lower bounds proved apply even for super-linear space usage) and there appears to be no way of extending their approach to obtain lower bounds for the problems above. Also, the time lower bounds in [CJ11, CJS13] are for the sum of the update and query time, whereas in many applications such as network traffic monitoring, updates are much more frequent and thus it is more acceptable for queries to be slower. Lastly but importantly, the problems considered in [CJ11, CJS13] were not in the turnstile model.

A natural model for upper bounds in the streaming literature (and also for data structures in general), is the word RAM model: basic arithmetic operations on machine words of w bits each cost one unit of time. In the data structure literature, one of the strongest lower bounds that can be proven is in the cell probe model [MP69, Fre78, Yao81], where one assumes that the data structure is broken up into S words each of size w bits and cost is only incurred when the user reads a memory cell from the data structure (S is the space). Lower bounds proven in this model hold against any algorithm operating in the word RAM model. In recent years there has been much success in proving data structure lower bounds in the cell probe model (see for example [Păt08, Lar13]) using techniques from communication complexity and information theory. Can these techniques be imported to obtain time lower bounds for clasically studied turnstile streaming problems?

Question 1. *Can we use techniques from cell probe lower bound proofs to lower bound the update time complexity for classical streaming problems?*

Indeed the lower bounds for streaming multiplication and Hamming distance computation were proved using the *information transfer* technique of Pătraşcu and Demaine [PD06], but as mentioned, this approach appears useless against the turnstile streaming problems above.

There are a couple of obvious problems to attack for Question 1. For example, for many streaming problems one can move from, say, 2/3 success probability to $1 - \delta$ success probability by running $\Theta(\log(1/\delta))$ instantiations of the algorithm in parallel then outputting the median answer. This is possible whenever the output of the algorithm is numerical, such as for example distinct elements, moment estimation, entropy estimation, point query, or several other problems. Note that doing so increases both the space and update time complexity of the resulting streaming algorithm by a $\Theta(\log(1/\delta))$ factor. Is this necessary? It was shown that the blowup in *space* is necessary for several problems by Jayram and Woodruff [JW13], but absolutely nothing is known about whether the blowup in time is required. Thus, any non-trivial lower bound in terms of δ would be novel.

Another problem to try for Question 1 is the heavy hitters problem. Consider the ℓ_1 heavy hitters problem: a vector v receives turnstile updates, and during a query we must report all i such that $|v_i| \geq \varepsilon\|v\|_1$. Our list is allowed to contain some false positives, but not "too false", in the sense that any i output should at least satisfy $|v_i| \geq (\varepsilon/2)\|v\|_1$. Algorithms known for this problem using non-trivially small space, both randomized [CM05] and deterministic [GM07,NNW14], require $\tilde{\Omega}(\log n)$ update time. Can we prove a matching lower bound?

Our Results. In this paper we present the first update time lower bounds for a range of classical turnstile streaming problems. The lower bounds apply to a restricted class of randomized streaming algorithms that we refer to as randomized *non-adaptive* algorithms. We say that a randomized streaming algorithm is *non-adaptive* if:

- Before processing any elements of the stream, the algorithm may toss some random coins.
- The memory words read/written to (henceforth jointly referred to as probed) on any update operation (i, Δ) are completely determined from i and the initially tossed random coins.

Thus a non-adaptive algorithm may not decide which memory words to probe based on the current contents of the memory words. Note that in this model of non-adaptivity, the random coins can encode e.g. a hash function chosen (independent of the stream) from a desirable family of hash functions and the update algorithm can choose what to probe based on the hash function. It is only the input-specific contents of the probed words that the algorithm may not use to decide which words to probe. To the best of our knowledge, all the known algorithms for classical turnstile streaming problems are indeed *non-adaptive*, in particular *linear sketches* (i.e. maintaining Πv in memory for some $r \times n$ matrix Π, $r \ll n$). We further remark that our lower bounds only require the update procedure to be non-adaptive and still apply if adaptivity is used to answer queries.

For the remainder of the paper, for ease of presentation we assume that the update increments are (possibly negative) integers bounded by some $M \leq \text{poly}(n)$ in magnitude, and that the number of updates in the stream is also at most $\text{poly}(n)$. We further assume the trans-dichotomous model [FW93, FW94], i.e. that the machine word size w is $\Theta(\log n)$ bits. This is a natural assumption, since typically the streaming literature assumes that basic arithmetic operations on a value $|v_i|$, the index of the current position in the stream, or an index i into v can be performed in constant time.

We prove lower bounds for the following types of queries in turnstile streams. For each problem listed, the query function takes no input (other than point query, which takes an input $i \in [n]$). Each query below is accompanied by a description of what the data structure should output.

- ℓ_1 **heavy hitter:** return an index i such that $|v_i| \geq \|v\|_\infty - \|v\|_1/2$.
- **Point query:** given i at query time, returns $v_i \pm \|v\|_1/2$.
- ℓ_p/ℓ_q **norm estimation** $(1 \leq q \leq p \leq \infty)$**:** returns $\|v\|_p \pm \|v\|_q/2$.
- **Entropy estimation.** returns a 2-approximation of the entropy of the distribution which assigns probability $v_i/\|v\|_1$ to i for each $i \in [n]$.

The lower bounds we prove for non-adaptive streaming algorithms are as follows ($n^{-O(1)} \leq \delta \leq 1/2 - \Omega(1)$ is the failure probability of a query):

- Any randomized non-adaptive streaming algorithm for point query, ℓ_p/ℓ_q estimation with $1 \leq q \leq p \leq \infty$, and entropy estimation, must have worst case update time

  $$t_u = \Omega\left(\frac{\log(1/\delta)}{\sqrt{\log n \log(eS/t_u)}}\right).$$

  We also show that any *deterministic* non-adaptive streaming algorithm for the same problems must have worst case update time $t_u = \Omega(\log n / \log(eS/t_u))$.

- Any randomized non-adaptive streaming algorithm for ℓ_1 heavy hitters, must have worst case update time

  $$t_u = \Omega\left(\min\left\{\sqrt{\frac{\log(1/\delta)}{\log(eS/t_u)}}, \frac{\log(1/\delta)}{\sqrt{\log t_u \cdot \log(eS/t_u)}}\right\}\right).$$

  Any deterministic non-adaptive streaming algorithm for the same problem must have worst case update time $t_u = \Omega\left(\frac{\log n}{\log(eS/t_u)}\right)$.

Remark 2. The deterministic bound above for point query matches two previous upper bounds for point query [NNW14], which use error-correcting codes to yield deterministic point query data structures. Specifically, for space $S = O(\log n)$, our lower bound implies $t_u = \Omega(\log n)$, matching an upper bound based on random codes. For space $O((\log n / \log \log n)^2)$, our lower bound implies $t_u = \Omega(\log n / \log \log n)$, matching an upper bound based on Reed-Solomon codes. Similarly, the deterministic bound above for deterministic ℓ_2/ℓ_1 norm estimation matches the previous upper bound for this problem [NNW14], showing that for the optimal space $S = \Theta(\log n)$, the fastest query time of non-adaptive algorithms is $t_u = \Theta(\log n)$.

These deterministic upper bounds are also in the cell probe model. In particular, the point query data structure based on random codes and the norm estimation data structure require access to combinatorial objects that are shown to exist via the probabilistic method, but for which we do not have explicit constructions. The point query structure based on Reed-Solomon codes can be implemented in the word RAM model with $t_u = \tilde{O}(\log n)$ using fast multipoint evaluation of polynomials. This is because performing an update, in addition to accessing $O(\log n/\log\log n)$ memory cells of the data structure, requires evaluating a degree-$O(\log n/\log\log n)$ polynomial on $O(\log n/\log\log n)$ points to determine which memory cells to access (see [NNW14] for details).

Remark 3. The best known randomized upper bounds are $S = t_u = O(\log(1/\delta))$ for point query [CM05] and ℓ_p/ℓ_p estimation for $p \leq 2$ [TZ12, KNPW11]. For entropy estimation the best upper bound has $S = t_u = \tilde{O}((\log n)^2 \log(1/\delta))$ [HNO08]. For ℓ_1 heavy hitters the best known upper bound (in terms of S and t_u) has $S = t_u = O(\log(n/\delta))$.

In addition to being the first non-trivial turnstile update time lower bounds, we also managed to show that the update time has to increase polylogarithmically in $1/\delta$ as the error probability δ decreases, which is achieved with the typical reduction using $\log(1/\delta)$ independent copies.

Our lower bounds can also be viewed in another light. If one is to obtain constant update time algorithms for the above problems, then one has to design algorithms that are *adaptive*. Since all known upper bounds have non-adaptive updates, this would require a completely new strategy to designing turnstile streaming algorithms.

Technique. As suggested by Question 1, we prove our lower bounds using recent ideas in cell probe lower bound proofs. More specifically, we use ideas from the technique now formally known as *cell sampling* [GM03, PTW10, Lar12]. This technique derives lower bounds based on one key observation: if a data structure/streaming algorithm probes t memory words on an update, then there is a set C of t memory words such that at least m/S^t updates probe only memory words in C, where m is the number of distinct updates in the problem (for data structure lower bound proofs, we typically consider queries rather than updates, and we obtain tighter lower bounds by forcing C to have near-linear size).

We use this observation in combination with the standard one-way communication games typically used to prove streaming space lower bounds. In these games, Alice receives updates to a streaming problem and Bob receives a query. Alice runs her updates through a streaming algorithm for the corresponding streaming problem and sends the resulting Sw bit memory footprint to Bob. Bob then answers his query using the memory footprint received from Alice. By proving communication lower bounds for *any* communication protocol solving the one-way communication game, one obtains space lower bounds for the corresponding streaming problem.

At a high level, we use the cell sampling idea in combination with the one-way communication game as follows: if Alice's non-adaptive streaming algorithm happens to "hash" her updates such that they all probe the same t memory cells, then she only needs to send Bob the contents of those t cells. If $t < S$, this gives a communication saving over the standard reduction above. We formalize this as a general sketch-compression theorem, allowing us to compress the memory footprint of any non-adaptive streaming algorithm at the cost of increasing the error probability. This general theorem has the advantage of allowing us to re-use previous space lower bounds that have been proved using the standard reduction to one-way communication games, this time however obtaining lower bounds on the update time. We demonstrate these ideas in Section 2 and also give a more formal definition of the classic one-way communication game.

2 Sketch Compression

In the following, we present a general theorem for compressing non-adaptive sketches. Consider a streaming problem in which we are to maintain an n_0-dimensional vector v. Let U be the *update domain*, where each element of U is a pair $(i, \Delta) \in [n] \times \{-M, \ldots, M\}$ for some $M = \text{poly}(n)$. We interpret an update $(i, \Delta) \in U$ as having the effect $v[i] \leftarrow v[i] + \Delta$. Initially all entries of v are 0. We also define the *query domain* $Q = \{q_1, \ldots, q_r\}$, where each $q_i \in Q$ is a function $q_i : \mathbb{Z}^n \to \mathbb{R}$. With one-way communication games in mind, we define the input to a streaming problem as consisting of two parts. More specifically, the *pre-domain* $D_{pre} \subseteq U^a$ consists of sequences of a update operations. The *post-domain* $D_{post} \subseteq \{U \cup Q\}^b$ consists of sequences of b updates and/or queries. Finally, the *input domain* $D \subseteq D_{pre} \times D_{post}$ denotes the possible pairings of a initial updates followed by b intermixed queries and updates. The set D defines a streaming problem P_D.

We say that a randomized streaming algorithm for a problem P_D uses S words of space if the maximum number of memory words used when processing any $d \in D$ is S. Here a memory word consists of $w = \Theta(\log n)$ bits. The worst case update time t_u is the maximum number of memory words read/written to upon processing an update operation for any $d \in D$. The error probability δ is defined as the maximum probability over all $d \in D$ and queries $q_i \in d \cap Q$, of returning an incorrect results on query q_i after the updates preceding it in the sequence d.

A streaming problem P_D of the above form naturally defines a one-way communication game: on an input $(d_1, d_2) \in D$, Alice receives d_1 (the first a updates) and Bob receives d_2 (the last b updates and/or queries). Alice may now send a message to Bob based on her input and Bob must answer all queries in his input as if streaming through the concatenated sequence of operations $d_1 \circ d_2$. The error probability of a communication protocol is defined as the maximum over all $d \in D$ and $q_i \in \{d \cap Q\}$, of returning an incorrect results on q_i when receiving d as input.

Traditionally, the following reduction is used:

Theorem 4. *If there is a randomized streaming algorithm for P_D with space usage S and error probability δ, then there is a public coin protocol for the corresponding one-way communication game in which Alice sends Sw bits to Bob and the error probability is δ.*

Proof. Alice simply runs the streaming algorithm on her input and sends the memory image to Bob. Bob continues the streaming algorithm and outputs the answers. ■

Recall from Section 1 that a randomized streaming algorithm is *non-adaptive* if:

- Before processing any elements of the stream, the algorithm may toss some random coins.
- The memory words read/written to (henceforth jointly referred to as probed) on any update operation (i, Δ) is completely determined from i and the initially tossed random coins.

We show that for non-adaptive algorithms, one can efficiently reduce the communication by increasing the error probability. We require some additional properties of the problem however: we say that a streaming problem P_D is *permutation invariant* if:

- For any *permutation* $\pi : [n] \to [n]$, it holds that $\pi(q_i(v)) = (\pi(q_i)(\pi(v)))$ for all $q_i \in Q$. Here $\pi(v)$ is the n-dimensional vector with value $v[i]$ in entry $\pi[i]$, $\pi(q_i)$ maps all indices (if any) in the definition of the query q_i wrt. π and $\pi(q_i(v))$ maps all indices in the answer $q_i(v)$ (if any) wrt. π.

Observe that point query, ℓ_p estimation, entropy estimation and heavy hitters all are permutation invariant problems. For point query, we have $\pi(q_i(v)) = q_i(v)$ since answers contain no indices, but $\pi(q_i)$ might differ from q_i since queries are defined from indices. For ℓ_p estimation and entropy estimation, we simply have $\pi(q_i(v)) = q_i(v)$ and $\pi(q_i) = q_i$ since neither queries or answers involve indices. For heavy hitters we have $\pi(q_i) = q_i$ (there is only one query), but we might have that $\pi(q_i(v)) \neq q_i(v)$ since the answer to the one query is an index. We now have the following:

Theorem 5. *If there is a randomized non-adaptive streaming algorithm for a permutation invariant problem P_D with $a \leq \sqrt{n}$, having space usage S, error probability δ, and worst case update time $t_u \leq (1/2)(\log n/\log(eS/t_u))$, then there is a private coin protocol for the corresponding one-way communication game in which Alice sends at most $a \log e + t_u a \log(eS/t_u) + \log a + \log\log(en/a) + 2t_u w + 1$ bits to Bob and the error probability is $2e^a \cdot (eS/t_u)^{t_u a}\delta$.*

Before giving the proof, we present the two main ideas. First observe that once the random choices of a non-adaptive streaming algorithm have been made, there must be a large collection of indices $I \subseteq [n]$ for which all updates (i, Δ), where $i \in I$, probe the same small set of memory words (there are at most $\binom{S}{t_u}$ distinct sets of t_u words to probe). If all of Alice's updates probed only the same set of t_u words, then Alice could simply send those words to Bob and we would have reduced the communication to $t_u w$ bits. To handle the case where Alice's updates probe different sets of words, we make use of the permutation invariant property. More specifically, we show that Alice and Bob can agree on a collection of k permutations of the input indices, such that one of these permutes all of Alice's updates to a new set of indices that probe the same t_u memory cells. Alice can then send this permutation to Bob and they can both alter their input based on the permutation. Therefore Alice and Bob can solve the communication game with $\log k + t_u w$ bits of communication. The permutation of indices unfortunately increases the error probability as we shall see below. With these ideas in mind, we now give the proof of Theorem 5.

Proof (of Theorem 5). Alice and Bob will permute their communication problem and use the randomized non-adaptive streaming algorithm on this transformed instance to obtain an efficient protocol for the original problem.

By Yao's principle, we have that the randomized complexity of a private coin one-way communication game with error probability δ equals the complexity of the best deterministic algorithm with error probability δ over the worst distribution. Hence we show that for any distribution μ on $D \subseteq D_{pre} \times D_{post}$, there exists a deterministic one-way communication protocol with $t_u a \log S + \log a + \log\log n + t_u w$ bits of communication and error probability $2e^a \cdot (eS/t_u)^{t_u a}\delta$. We let μ_1 denote the marginal distribution over D_{pre} and μ_2 the marginal distribution over D_{post}.

Let $\mu = (\mu_1, \mu_2)$ be a distribution on D. Define from this a new distribution $\gamma = (\gamma_1, \gamma_2)$: pick a uniform random permutation π of $[n]$. Now draw an input d from μ and permutate all indices of updates (and queries if defined for such) using the permutation π. The resulting sequence $\pi(d)$ is given to Alice and Bob as before, which defines the new distribution $\gamma = (\gamma_1, \gamma_2)$. We use $A \sim \mu_1$ to denote the r.v. providing Alice's input drawn from distribution μ_1 and $\pi(A) \sim \gamma_1$ denotes the random variable providing Alice's transformed input. We define $B \sim \mu_2$ and $\pi(B) \sim \gamma_2$ symmetrically.

Recall we want to solve the one-way communication game on A and B. To do this, first observe that by fixing the random coins, the randomized non-adaptive streaming algorithm gives a non-adaptive and deterministic streaming algorithm that has error probability δ and space usage S

under distribution γ. Before starting the communication protocol on A and B, Alice and Bob both examine the algorithm (it is known to both of them). Since it is non-adaptive and deterministic, they can find a set of t_u memory words C, such that at least

$$\frac{n}{\binom{S}{t_u}} > \frac{n}{(eS/t_u)^{t_u}} \geq \sqrt{n} \geq a$$

indices $i \in [n]$ satisfy that any update (i, Δ) probes only memory words in C. We let I^C denote the set of all such indices (again, I^C is known to both Alice and Bob). Alice and Bob also agree on a set of permutations $\{\rho_1, \ldots, \rho_k\}$ (we determine a value for k later), such that for any set of at most a indices, I', that can occur in Alice's updates, there is at least one permutation ρ_i where:

- $\rho_i(j) \in I^C$ for all $j \in I'$
- Let $I(A)$ denote the (random) indices of the updates in A. Then the probability that the non-adaptive and deterministic protocol errs on input $\rho_i(A)$, conditioned on $I(A) = I'$, is at most $2e^a \cdot (eS/t_u)^{t_u a} \cdot \varepsilon_{I(A)=I'}$ where $\varepsilon_{I(A)=I'}$ is the error probability of the deterministic and non-adaptive streaming algorithm on distribution γ, conditioned on $I(A) = I'$.

Again, this set of permutations is known to both players. The protocol is now simple: upon receiving A, Alice finds the index i of a permutation ρ_i satisfying the above for the indices $I(A)$. She then sends this index to Bob and runs the deterministic and non-adaptive algorithm on $\rho_i(A)$. She forwards the addresses and contents of all memory words in C as well. This costs a total of $\log k + |C|(w + \log S(n)) \leq \log k + 2t_u(n)w$ bits. Note that no words outside C are updated during Alice's updates. Bob now remaps his input B according to ρ_i and runs the deterministic and non-adaptive streaming algorithm on his updates and queries. Observe that for each query $q_j \in B$, Bob will get the answer $\rho_i(q_j)(\rho_i(v))$ if the algorithm does not err, where v is the "non-permuted" vector after processing all of Alice's updates A and all updates in B preceeding the query q_j. For each such answer, he computes $\rho_i^{-1}(\rho_i(q_j)(\rho_i(v)))$. Since P_D is permutation invariant, we have $\rho_i^{-1}(\rho_i(q_j)(\rho_i(v))) = \rho_i^{-1}(\rho_i(q_j(v))) = q_j(v)$. The final error probability (over μ) is hence at most $2e^a \cdot (eS/t_u)^{t_u a} \cdot \delta$ since $\mathbb{E}_{I'}\, \varepsilon_{I(A)=I'} = \delta$.

We only need a bound on k. For this, fix one set I' of at most a indices in $[n]$ and consider drawing k uniform random permutations. For each such random permutation Γ, note that $\Gamma(I')$ is distributed as $I(\pi(A))$ conditioned on $I(A) = I'$. Hence, the expected error probability when using the map Γ (expectation over choice of Γ) is precisely $\varepsilon_{I(A)=I'}$. We also have

$$\mathbb{P}(\Gamma(I') \subseteq I^C) = \frac{\binom{|I^C|}{|I'|}}{\binom{n}{|I'|}} \geq \left(\frac{|I^C|}{en}\right)^{|I'|} \geq e^{-a} \cdot \left(\frac{eS}{t_u}\right)^{-t_u a}$$

By Markov's inequality and a union bound, we have both $\Gamma(I') \subseteq I^C$ and error probability at most $2e^a \cdot (eS/t_u)^{t_u a} \cdot \varepsilon_{I(A)=I'}$ with probability at least $e^{-a} \cdot (eS/t_u)^{-t_u a}/2 \stackrel{\text{def}}{=} p$ over the choice of Γ. Thus if we pick $k = (1/p)a \log(en/a) > (1/p) \cdot \log \binom{n}{a}$ and use that $1 + x \leq e^x$ for all real x, then setting the probability that all permutations Γ chosen fail to have the desired failure probability and $\Gamma(I') \subseteq I^C$ is at most $(1-p)^k < 1/\binom{n}{a}$. Thus by a union bound over all $\binom{n}{a}$ size-a subsets of indices, we can conclude that the desired set of permutations exists. ■

3 Implications

In this section, we present the (almost) immediate implications of Theorem 5. All these result follow by re-using the one-way communication game lower bounds originally proved to obtain space lower bounds of heavy hitters. Consider the generic protocol in Figure 1. For different streaming problems, we will show the different ways to implement the function $\mathit{Check}(t, j)$ using the updates and queries of the problems, where $\mathit{Check}(t, j)$ is supposed to be able to tell if t is equal to i_j with failure probability δ. We show first that if this is the case, we obtain a lower bound on the update time t_u. The lower bounds then follow from the implementation of $\mathit{Check}(t, j)$.

Alice chooses a indices $i_1, \dots, i_a$ randomly without replacement in $[n]$.
Alice performs updates $v[i_j] \leftarrow v[i_j] + C^j$ for $j = 1, \dots, a$, where C is a large constant.
Alice sends her memory state to Bob.
for j from a down to 1 **do** ▷ Bob decodes $i_a, \dots, i_1$ from Alice's message.
    for all $t \in [n]$ **do**
        if Check(t,j) **then**
            Bob declares $i_j = t$.
            Bob performs the update $v[i_j] \leftarrow v[i_j] - C^j$
        end if
    end for
end for

Figure 1: The communication protocol for Alice to send a random numbers in $[n]$ to Bob.

Theorem 6. *Any randomized non-adaptive streaming algorithm that can implement all the* $\mathit{Check}(t, j)$ *calls using no more than* $k \leq n^{O(1)}$ *queries with failure probability* $n^{-\Omega(1)} \leq \delta \leq 1/2 - \Omega(1)$ *each, must have worst case update time* $t_u = \Omega\left(\min\left\{\sqrt{\frac{\log 1/\delta}{\log(eS/t_u)}}, \frac{\log 1/\delta}{\sqrt{\log k \log(eS/t_u)}}\right\}\right)$.

Proof. We first prove the lower bound for the case where $\sqrt{\frac{\log 1/\delta}{\log(eS/t_u)}}$ is the smaller of the two terms. This is the case at least from $n^{-\Omega(1)} \leq \delta \leq k^{-3}$. Assume for contradiction that such a randomized non-adaptive algorithm exists with $t_u = o(\sqrt{\log(1/\delta)/\log(eS/t_u)})$. Set $a = c_0 t_u$ for a sufficiently large constant c_0. We invoke Theorem 5 to conclude that Alice can send her memory state to Bob using $a \log e + t_u a \log(eS/t_u) + \log a + \log\log(en/a) + 2t_u w + 1$ bits while increasing the failure probability of Bob's k queries to

$$2e^a \cdot (eS/t_u)^{t_u a} \delta \leq (eS/t_u)^{c_0 t_u^2} \delta^{1-o(1)} \leq \delta^{1-o(1)} \leq k^{-2}$$

each. By a union bound, the answer to all Bob's queries are correct with probability at least $1 - 1/k \geq 9/10$, so Bob can recover $i_1, \dots, i_a$ with probability $9/10$. By Fano's inequality, Alice must send Bob at least $\Omega(H(i_1, \dots, i_a)) = \Omega(a \log(n/a)) \geq c_0 c_1 t_u \log n$ bits for some constant c_1 (where c_1 does not depend on c_0). But the size of her message was

$$a \log e + t_u a \log(eS/t_u) + \log a + \log\log(en/a) + 2t_u w + 1 \leq c_0 t_u^2 \log(eS/t_u) + 2t_u w + o(t_u \log n).$$

We assumed $t_u = o\left(\sqrt{\log(1/\delta)/\log(eS/t_u)}\right) = o\left(\sqrt{\log n/\log(eS/t_u)}\right)$ and thus the above is bounded by $2t_u w + o(t_u \log n)$. Setting c_0 high enough, we get that $2w \geq (c_0 c_1 \log n)/2$ and thus we have reached a contradiction.

For the case $k^{-3} < \delta < 1/2 - \Omega(1)$, observe that we can decrease the failure probability to k^{-3} by increasing the space, update time and query time by a factor $\alpha = \Theta(\log_{1/\delta} k)$: simply run α independent copies of the streaming algorithm and return the majority answer on a query. Hence the lower bound becomes

$$\Omega\left(\frac{\sqrt{\frac{\log k}{\log(eS\alpha/(t_u\alpha))}}}{\alpha}\right) = \Omega\left(\frac{\log 1/\delta}{\sqrt{\log k \log(eS/t_u)}}\right).$$

■

Theorem 7. *Any deterministic non-adaptive streaming algorithm that can implement $Check(t, j)$ must have worst case update time*

$$t_u = \Omega\left(\frac{\log n}{\log(eS/t_u)}\right).$$

Proof. The proof is similar to that of Theorem 6. Assume for contradiction there is a deterministic non-adaptive streaming algorithm with $t_u = o(\log n/\log(eS/t_u))$. Choose $a = c_0 t_u$ for sufficiently large constant c_0. By Theorem 5, Alice can send her memory state to Bob using

$$\begin{aligned} a\log e + t_u a\log(eS/t_u) + \log a + \log\log(en/a) + 2t_u w + 1 &\le c_0 t_u^2 \log(eS/t_u) + 2t_u w + o(t_u \log n) \\ &\le 2t_u w + o(t_u \log n) \end{aligned}$$

bits and Bob can still answer every query correctly. Since Bob can recover $i_1, \ldots, i_a$, Alice must send Bob at least $H(i_1, \ldots, i_a) = \Omega(a\log(n/a)) = c_0 c_1 t_u \log n$ for some constant c_1 (independent of c_0). Setting c_0 large enough gives $2w \ge (c_0 c_1 \log n)/2$, thus deriving the contradiction. ■

Corollary 8. *Any randomized non-adaptive streaming algorithm for point query, ℓ_p estimation with $1 \le p \le 2$, and entropy estimation with failure probability $n^{-\Omega(1)} \le \delta \le 1/2 - \Omega(1)$ must have worst case update time*

$$t_u = \Omega\left(\frac{\log(1/\delta)}{\sqrt{\log n \log(eS/t_u)}}\right).$$

Any deterministic non-adaptive streaming algorithm for the same problems must have worst case update time $t_u = \Omega(\log n/\log(eS/t_u))$.

Proof. We just need to show implementations of $Check(t, j)$ that use $n^{O(1)}$ queries in total and apply Theorem 6 to get the stated bound for randomized algorithms. The stated bound for deterministic algorithms follows from applying Theorem 7 instead.

3.1 Applications to specific problems

Point query. We can implement each $Check(t, j)$ by simply querying for the value of $v[t]$ and check if the returned value is at least $C^j/3$. By the guarantee of the algorithm, if it does not fail, the returned value is within a factor 3 from the right value and thus, $Check(t, j)$ can correctly tell if $t = v_j$. Thus we run a total of $na = n^{O(1)}$ queries to implement all $Check(t, j)$'s.

ℓ_p/ℓ_q **estimation.** We can implement $Check(t, j)$ as follows.

- $v[t] \leftarrow v[t] - C^j$
- Check if $\|v\|_p$ is at most $C^j/3$
- $v[t] \leftarrow v[t] + C^j$

By the guarantee of the algorithm, if it does not fail, the estimated ℓ_p/ℓ_q norm is at most $3C^{j-1} < C^j/3$ if $t = i_j$ and it is greater than $C^j/3$ if $t \neq i_j$. Thus, $Check(t, j)$ can correctly tell if $t = v_j$. Again we used $n^{O(1)}$ queries in total.

Entropy estimation. We can implement $Check(t, j)$ as follows.

- $v[t] \leftarrow v[t] - C^j$
- Check if the entropy is at most $1/3$
- $v[t] \leftarrow v[t] + C^j$

Consider the case the algorithm does not fail. First, if $t = i_j$ then the entropy is at most

$$\sum_{i=1}^{j} \frac{C^i(C-1)}{C^{j+1}-C} \log \frac{C^{j+1}-C}{C^i(C-1)} \leq \sum_{i=1}^{j-1} \frac{C^i(C-1)}{C^{j+1}-C} \log 2C^{j-i} + \frac{C^j(C-1)}{C^{j+1}-C} \log \frac{C^{j+1}-C}{C^j(C-1)}$$
$$\leq O(\frac{\log C}{C-1}) + \frac{C^j(C-1)}{C^{j+1}-C} \cdot \frac{C^j-C}{C^j(C-1)} \leq O(\frac{\log C}{C-1}) < 1/10$$

On the other hand, if $t \neq i_j$ then after the first operation, $\|v\|_1 \leq \frac{2C^{j+1}-C^j-C}{C-1} < 3C^j$ so the entropy is at least the binary entropy of whether the index i_j is picked, which is greater than $H(1/3) > 0.9$. Thus, by the guarantee of the algorithm, $Check(t, j)$ correctly tells if $t = v_j$ using $n^{O(1)}$ queries. ■

Corollary 9. *Any randomized non-adaptive streaming algorithm for ℓ_1 heavy hitters with failure probability $n^{-O(1)} \leq \delta \leq 1/2 - \Omega(1)$ must have worst case update time*

$$t_u = \Omega\left(\min\left\{\sqrt{\frac{\log(1/\delta)}{\log(eS/t_u)}}, \frac{\log(1/\delta)}{\sqrt{\log t_u \cdot \log(eS/t_u)}}\right\}\right).$$

Any deterministic non-adaptive streaming algorithm for the same problems must have worst case update time $t_u = \Omega\left(\frac{\log n}{\log(eS/t_u)}\right)$.

Proof. We use a slightly different decoding procedure for Bob. Instead of running $Check(t, j)$ all indices $t \in [n]$, in iteration j, Bob can simply query for the heavy hitter to find i_j. Note that in iteration j, we have $\|v\|_1 = \frac{C^{j+1}-C}{C-1} < 2C^j = 2v[i_j]$ so if the algorithm is correct, Bob will find the index i_j. We have thus implemented all the $Check(t, j)$'s using only a queries. Recall from the proof of Theorem 6 that $a = O(t_u)$. ■

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[BYJKS04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

[CCFC04] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[CCM10] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for estimating the entropy of a stream. *ACM Transactions on Algorithms*, 6(3), 2010.

[CJ11] Raphaël Clifford and Markus Jalsenius. Lower bounds for online integer multiplication and convolution in the cell-probe model. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 593–604, 2011.

[CJS13] Raphaël Clifford, Markus Jalsenius, and Benjamin Sach. Tight cell-probe bounds for online hamming distance computation. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 664–674, 2013.

[CKS03] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

[CM05] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[EVF06] Cristian Estan, George Varghese, and Michael E. Fisk. Bitmap algorithms for counting active flows on high-speed links. *IEEE/ACM Trans. Netw.*, 14(5):925–937, 2006.

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[Fre78] Michael L. Fredman. Observations on the complexity of generating quasi-Gray codes. *SIAM J. Comput.*, 7:134–146, 1978.

[FW93] Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *J. Comput. Syst. Sci.*, 47(3):424–436, 1993.

[FW94] Michael L. Fredman and Dan E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *J. Comput. Syst. Sci.*, 48(3):533–551, 1994.

[GM03] Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, pages 332–344, 2003.

[GM07] Sumit Ganguly and Anirban Majumder. CR-precis: A deterministic summary structure for update data streams. In *ESCAPE*, pages 48–59, 2007.

[Gooa] Google. Google Trends. `http://www.google.com/trends`.

[Goob] Google. Google Zeitgeist. `http://www.google.com/press/zeitgeist.html`.

[Gro09] Andre Gronemeier. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 505–516, 2009.

[HNO08] Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 489–498, 2008.

[Jay09] T. S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of AND. In *Proceedings of the 12th International Workshop on Randomization and Computation (RANDOM)*, pages 562–573, 2009.

[JKS08] T. S. Jayram, Ravi Kumar, and D. Sivakumar. The one-way communication complexity of Hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

[JST11] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 49–58, 2011.

[JW13] T. S. Jayram and David P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms*, 9(3):26, 2013.

[KNPW11] Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 745–754, 2011.

[KNW10a] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1161–1178, 2010.

[KNW10b] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 41–52, 2010.

[KSZC03] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *Internet Measurement Comference*, pages 234–247, 2003.

[Lar12] Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 293–301, 2012.

[Lar13] Kasper Green Larsen. *Models and Techniques for Proving Data Structure Lower Bounds*. PhD thesis, Aarhus University, 2013.

[Moo01] David Moore, 2001. `http://www.caida.org/research/security/code-red/`.

[MP69] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, 1969.

[Mut05] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[NNW14] Jelani Nelson, Huy L. Nguyễn, and David P. Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Lin. Alg. Appl.*, 441:152–167, January 2014. Preliminary version in RANDOM 2012.

[Păt08] Mihai Pătraşcu. *Lower bound techniques for data structures*. PhD thesis, Massachusetts Institute of Technology, 2008.

[PD06] Mihai Pătraşcu and Erik D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35(4):932–963, 2006.

[PDGQ05] Rob Pike, Sean Dorward, Robert Griesemer, and Sean Quinlan. Interpreting the data: Parallel analysis with Sawzall. *Scientific Programming*, 13(4):277–298, 2005.

[PTW10] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 805–814, 2010.

[TZ12] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput.*, 41(2):293–331, 2012.

[Woo04] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 167–175, 2004.

[Woo07] David P. Woodruff. *Efficient and Private Distance Approximation in the Communication and Streaming Models*. PhD thesis, Massachusetts Institute of Technology, 2007.

[Yao81] Andrew Chi-Chih Yao. Should tables be sorted? *J. ACM*, 28:615–628, 1981.