# Annotations in Data Streams

Amit Chakrabarti[1], Graham Cormode[2], and Andrew McGregor[3]

[1] Dartmouth College, `ac@cs.dartmouth.edu`
[2] AT&T Labs–Research, `graham@research.att.com`
[3] University of Massachusetts, Amherst, `mcgregor@cs.umass.edu`

**Abstract.** The central goal of data stream algorithms is to process massive streams of data using *sublinear* storage space. Motivated by work in the database community on outsourcing database and data stream processing, we ask whether the space usage of such algorithms be further reduced by enlisting a more powerful "helper" who can *annotate* the stream as it is read. We do not wish to blindly trust the helper, so we require that the algorithm be convinced of having computed a correct answer. We show upper bounds that achieve a non-trivial tradeoff between the amount of annotation used and the space required to verify it. We also prove lower bounds on such tradeoffs, often nearly matching the upper bounds, via notions related to Merlin-Arthur communication complexity. Our results cover the classic data stream problems of selection, frequency moments, and fundamental graph problems such as triangle-freeness and connectivity. Our work is also part of a growing trend — including recent studies of multi-pass streaming, read/write streams and randomly ordered streams — of asking more complexity-theoretic questions about data stream processing. It is a recognition that, in addition to practical relevance, the data stream model raises many interesting theoretical questions in its own right.

## 1 Introduction

The data stream model has become a popular abstraction when designing algorithms that process network traffic and massive data sets [4, 21]. The computational restrictions that define this model are severe: algorithms must use a relatively small amount of working memory and process input in whatever order it arrives. This captures constraints in high-throughput data processing settings. For example, network monitoring often requires (near) real-time response to anomalies and hence traffic must be processed as it arrives, rather than being stored and processed offline. For massive data sets stored in external memory, being able to process the data in any order avoids the I/O bottlenecks that arise with algorithms that assume random access. Unfortunately, while some problems admit efficient streaming algorithms, many others provably require a lot of working memory or multiple passes over the data, which is typically not feasible.

This paper considers the potential for off-loading stream computation to a more powerful "helper" so that single pass, small-space stream computation is possible even for such "hard" functions. The additional power of the helper can arise in a variety of situations, e.g., multiple processing units, special purpose hardware, or a third party who provide a commercial stream processing service. This last case has recently garnered attention in the context of outsourcing database processing [27, 29, 34]. A key issue is that we do not want to blindly trust the helper: hardware faults or outright deception by a third-party would lead to incorrect results. So our protocols must have sufficient information contained in the help to allow the "verifier" to be convinced that they have obtained the correct answer. We think of this help as annotations augmenting the original stream. Our goal is to design protocols so that the verifier finds the correct answer with an honest helper, and is likely not fooled by a dishonest helper. The primary metrics are the amount of annotations provided by the helper and the amount of working space used by the verifier.

Our approach is naturally related to Interactive Proofs and Merlin-Arthur communication protocols [1, 5, 25] but differs in two important regards. Firstly, the verifier must process both the original data and the advice provided by the helper under the usual restrictions of the data stream model. Secondly, we focus on annotations that can be provided *online*. Note that in Merlin-Arthur communication, it is assumed that the helper is omniscient and that the advice he provides can take into account data held by any of the players. In the stream model, this would correspond to *prescience* where the annotation in the stream at position $t$ may depend on data that is yet to arrive. In contrast we are primarily interested in designing algorithms with online annotation, i.e., annotation that only depends on data that has arrived before the annotation is written. This corresponds to a helper who sees the data concurrently with the verifier.

**Our Contributions:** We first formally define the relevant models: traditional and online Merlin-Arthur communication, and streaming models with either prescient or online annotations. We then investigate the complexity of a range of problems in these models, including selection, frequency moments, and graph problems such as triangle-counting and connectivity. Estimating frequency moments in particular has become a canonical problem when exploring variants of the data stream model such as random order streams [10] and read/write streams [7]. Our results include:

- *Selection.* The problem of finding the median of $m$ values in the range $[n]$ highlights the difference between prescient and online annotation. For any $h, v$ such that $hv \geq m$ we present an $O(v \log m)$-space algorithm that uses $O(h \log m \log n)$ bits of online annotation. Furthermore, we show that this trade-off is optimal up to polylogarithmic factors. In contrast, a trivial $O(\log mn)$ space algorithm can verify $O(\log n)$ bits of prescient annotation.
- *Frequency Moments and Frequent Items.* We next consider properties of $\{f_i\}_{i \in [n]}$ where $f_i$ is the frequency of the token "$i$". For any $h, v$ such that $hv \geq n$, we present an $O(h \log m)$-space algorithm that uses $(\phi^{-1} v \log m)$ bits of online annotation and returns exactly the tokens whose frequency exceeds $\phi m$. We also show an $O(\log m)$ space algorithm that uses $O(\varepsilon^{-1} \log^2 m)$ bits of online annotation and returns a set of tokens containing $\{i : f_i \geq \phi m\}$ and no elements from $\{i : f_i \leq (\phi - \varepsilon) m\}$. This algorithm relies on a powerful way that annotation can be used in conjunction with sketch-based algorithms. For any $h, v$ such that $hv \geq n$, we present an $O(kv \log m)$-space algorithm that uses $O(k^2 h \log m)$ bits of online annotation and computes $F_k = \sum_i f_i^k$ exactly ($k \in \mathbb{Z}_+$). The trade-off is optimal up to polylogarithmic factors even if the algorithm is allowed to use prescient annotation. To prove this we present the first Merlin-Arthur communication bounds for multi-party set-disjointness.

– *Graph Problems.* For graphs defined by streams of $m$ edges on $n$ nodes, we show that only $O(\log n)$ space is needed by the verifier to determine whether a graph is connected, contains a perfect matching, or is triangle-free, with annotation proportional to the input size. We show that our algorithms are optimal in many cases. For any $h, v$ such that $hv \geq n^3$, we also present an $\tilde{O}(v)$ space algorithm for counting triangles that uses $\tilde{O}(h)$ bits of annotation where $\tilde{O}$ hides poly-logarithmic factors.

**Related Work:** When multiple passes over the input are allowed, it is natural to consider annotations that can be written to the "input tape" and are available to the stream algorithm in subsequent passes [3,14,15]. The read/write stream model, which provides both multiple passes and multiple working tapes, can be viewed as a natural extension of the multi-pass annotation model [7, 8, 20]. However, such annotations are of no use if only a single pass over the input is allowed.

Few examples of prior work have explicitly considered annotations that are provided by an (untrusted) third party. Gertner et al. [19] showed that the set of languages recognized by a verifier with logarithmic space given annotation polynomial in the input size is exactly NP. In contrast, our focus is on the case where the annotation is (sub)linear in the input size and can be provided online; the distinction between prescient and online annotation was not relevant in their results because with polynomial annotation, the entire input could be repeated. Feigenbaum et al. [17] observe that a logarithmic space verifier can check a linear space annotation for the disjointness problem. In communication complexity, the role of non-deterministic advice has been studied more extensively, see e.g., [5, 26]. Recent works of Aaronson and Widgerson [1] and Klauck [25] are particularly relevant. They resolve the MA complexity of two-party set disjointness — we extend some of their techniques to our streaming model.

There has also been more applied work which implicitly defines annotation protocols. The notion of *stream punctuations* are, in our terminology, simple prescient annotations, indicating facts such as that there are no more tuples relevant to timestamp $t$ in the remainder of the stream [33]. Work on stream outsourcing studies the problem of verifying that a claimed "grouping" corresponds to the input data [34]. They solve exact and approximate versions of the problem by using a linear amount of annotation. Lastly, work on *proof infused streams* answers various selection and aggregation queries over sliding windows [27] with logarithmic space and linear annotation. However, a critical difference is that this work requires that the helper and verifier agree on a one-way hash function, for which it is assumed the helper cannot find collisions. Our results are in a stronger model without this assumption.

## 2 Models and Definitions

### 2.1 Communication Models

Let $f : X_1 \times \cdots \times X_t \to \{0,1\}$ be a function, where each $X_i$ is a finite set. This naturally gives a $t$-player number-in-hand communication problem, where Player $i$ holds an input $x_i \in X_i$ and the players wish to output $f(x_1, \ldots, x_t)$ correctly, with high probability.

**MA Communication:** We first consider a variant of this communication model. A Merlin-Arthur protocol (henceforth, "MA protocol") for $f$ is one that involves the usual $t$ players, plus a "super-player," called Merlin, who knows the entire input $\mathbf{x} = (x_1, \ldots, x_t)$. The protocol works as follows: first Merlin deterministically writes a help message $\mathfrak{h}$ on the blackboard, and then Players 1 through $t$ run a randomized protocol $\mathscr{P}$, using a public random string $R$, eventually outputting a bit $\mathrm{out}(\mathscr{P}; \mathbf{x}, R, \mathfrak{h})$. To clarify, $R$ is not known to Merlin at the time he writes $\mathfrak{h}$. An MA protocol is $\delta$-error if there exists a function $\mathfrak{h} : X_1 \times \ldots \times X_t \to \{0,1\}^*$, such that:

1. If $f(\mathbf{x}) = 1$ then $\Pr_R[\mathrm{out}(\mathscr{P}; \mathbf{x}, R, \mathfrak{h}(\mathbf{x})) = 0] \le \delta$.
2. If $f(\mathbf{x}) = 0$ then $\forall \mathfrak{h}'$ $\Pr_R[\mathrm{out}(\mathscr{P}; \mathbf{x}, R, \mathfrak{h}') = 1] \le \delta$.

We define $\mathrm{err}(\mathscr{P})$ to be the minimum $\delta$ such that the above conditions are satisfied. We also define the *help cost* $\mathrm{hcost}(\mathscr{P})$ to be the maximum length of $\mathfrak{h}$, over all $\mathbf{x}$, and the *verification cost* $\mathrm{vcost}(\mathscr{P})$ to be the maximum number of bits communicated by Players 1 through $t$ over all $\mathbf{x}$ and $R$. Finally, we define the *cost* of $\mathscr{P}$ to be $\mathrm{cost}(\mathscr{P}) = \mathrm{hcost}(\mathscr{P}) + \mathrm{vcost}(\mathscr{P})$. We then define the $\delta$-error MA-complexity of $f$ as $\mathrm{MA}_\delta(f) = \min\{\mathrm{cost}(\mathscr{P}) : \mathscr{P} \text{ is an MA protocol for } f \text{ with} \mathrm{err}(\mathscr{P}) \le \delta\}$. Further, we define $\mathrm{MA}(f) = \mathrm{MA}_{1/3}(f)$.

**Online-MA Communication:** We also consider a variant of the above model, specific to *one-way protocols* (i.e., protocols where the players speak once each, in increasing order), where Merlin constructs $t$ help messages $\mathfrak{h}_1, \ldots, \mathfrak{h}_t$ so that the $i$th message is only a function of the first $i$ inputs. To make this precise we need to amend the definition of $\delta$-error: An online-MA protocol is $\delta$-error if there exists a family of functions $\mathfrak{h}_i : X_1 \times \ldots \times X_i \to \{0,1\}^*$, such that:

1. If $f(\mathbf{x}) = 1$ then $\Pr_R[\mathrm{out}(\mathscr{P}; \mathbf{x}, R, \mathfrak{h}_1(x_1), \mathfrak{h}_2(x_1, x_2), \ldots, \mathfrak{h}_t(x_1, \ldots, x_t)) = 0] \le \delta$.
2. If $f(\mathbf{x}) = 0$ then $\forall \mathfrak{h}' = (\mathfrak{h}'_1, \mathfrak{h}'_2, \ldots, \mathfrak{h}'_t)$ $\Pr_R[\mathrm{out}(\mathscr{P}; \mathbf{x}, R, \mathfrak{h}') = 1] \le \delta$.

The message $\mathfrak{h}_i$ is revealed privately to the $i$th player. We define the help cost, $\mathrm{hcost}(\mathscr{P})$, to be the maximum length of $\sum_{i \in [t]} |\mathfrak{h}_i|$. We define $\mathrm{err}(\mathscr{P}), \mathrm{vcost}(\mathscr{P})$, and $\mathrm{cost}(\mathscr{P})$ as for MA. Define $\mathrm{MA}^{\rightarrow}_{\delta}(f) = \min\{\mathrm{cost}(\mathscr{P}) : \mathscr{P}$ is an online MA protocol for $f$ with $\mathrm{err}(\mathscr{P}) \le \delta\}$ and write $\mathrm{MA}^{\rightarrow}(f) = \mathrm{MA}^{\rightarrow}_{1/3}(f)$.

### 2.2 Data Stream Models

The annotated data-stream models are most conveniently defined relative to the above communication models. Again we consider the computation of a function $f$ on a $t$-tuple $\mathbf{x} \in \mathscr{U}^t$ for some universe $\mathscr{U}$, e.g., $\{0,1\}$ or $[n]$. The main difference from the communication model is that we further insist that the message sent by player $i$ must be computed with limited memory and only sequential access to $x_i$ and $\mathfrak{h}_i$. Without advice, this is equivalent to the usual definition of the single-pass data stream model. We will also consider non-Boolean functions $f$ and a notion of approximation: we say $f$ is computed correctly if the answer returned is in some pre-defined set $C(f(\mathbf{x}))$, e.g., $\{a : |a - f(\mathbf{x})| \le \varepsilon f(\mathbf{x})\}$.

**Stream Model with Prescient Annotations:** In the context of the stream model we consider the help $\mathfrak{h}$ provided by Merlin to be decomposed into $t$ (deterministic) functions that map the input to binary help strings: $\mathfrak{h}_1 : \mathscr{U}^t \to \{0,1\}^*, \ldots, \mathfrak{h}_t : \mathscr{U}^t \to \{0,1\}^*$. Let $\mathfrak{h}(\mathbf{x}) := (\mathfrak{h}_1(\mathbf{x}), \ldots, \mathfrak{h}_t(\mathbf{x}))$. We then consider a randomized protocol, $\mathscr{A}$, with oracle access to a random string $R$, where Player $i$ computes a message of size at most

$w$ given only $w$ bits of working memory and only sequential access to the bit stream $\langle x_i, \mathfrak{h}_i(\mathbf{x})\rangle$. The output of this protocol is allowed to include the special symbol $\perp$ if the verifier is not convinced of the validity of the annotation. Such a protocol is said be $\delta$-error if $\Pr_R[\mathrm{out}(\mathscr{A};\mathbf{x},R,\mathfrak{h}) \notin C(f(\mathbf{x}))] \leq \delta$ and $\Pr_R[\mathrm{out}(\mathscr{A};\mathbf{x},R,\mathfrak{h}') \neq \perp] \leq \delta$ for any $\mathfrak{h}' = (\mathfrak{h}'_1, \mathfrak{h}'_2, \ldots, \mathfrak{h}'_t) \neq \mathfrak{h}(\mathbf{x})$. We define $\mathrm{err}(\mathscr{A})$ to be the minimum $\delta$ such that the above conditions are satisfied. We define the *help cost* $\mathrm{hcost}(\mathscr{A})$ to be the maximum length of $\sum_i |\mathfrak{h}_i|$, over all $\mathbf{x}$, and the *verification cost* $\mathrm{vcost}(\mathscr{A}) = w$. We say that $\mathscr{A}$ and $\mathfrak{h}$ forms an $(h,v)$ *prescient scheme* if $\mathrm{hcost}(\mathscr{A}) = O(h+1)$, $\mathrm{vcost}(\mathscr{A}) = O(v+1)$ and $\mathrm{err}(\mathscr{A}) < 1/3$.

**Stream Model with Online Annotations:** For online annotations we insist that the $i$th help function is only a function of $(x_1, \ldots, x_i)$. The other definitions are as above. We say that $\mathscr{A}$ and $\mathfrak{h}$ form an $(h,v)$ *online scheme* as above if $\mathrm{hcost}(\mathscr{A}) = O(h+1)$, $\mathrm{vcost}(\mathscr{A}) = O(v+1)$ and $\mathrm{err}(\mathscr{A}) < 1/3$.

### 2.3 Preliminary Lemmas

In multiple places we make use of basic fingerprinting techniques which enable a verifier to test whether two large streams represent the same object using small space. Let $\mathbb{Z}_+$ denote the set of non-negative integers, and let $\mathbb{F}_q$ denote the finite field with $q$ elements (whenever it exists). Let $A = \langle a_1, \ldots, a_m\rangle$ denote a data stream, with each $a_i \in [n]$. Then $A$ implicitly defines a frequency distribution $\mathbf{f}(A) := (f_1, \ldots, f_n)$, where $f_j = |\{i \in [m] : a_i = j\}|$. Fingerprints are formed by computations over $\mathbb{F}_q$, as $\mathrm{BF}_q(r,\mathbf{f}) := \prod_{j=1}^{n}(r-j)^{f_j}$. To make fingerprints, we choose $q$ based on an *a priori* bound $m$ on $\|\mathbf{f}\|_1$.

**Lemma 1.** *Let $q \geq m$ be a prime, and choose $r$ uniformly at random from $\mathbb{F}_q$. Given an input stream $A$ of length $m$, the fingerprint $\mathrm{BF}_q(r,\mathbf{f}(A))$ can be computed using $O(\log q)$ storage. Suppose $\mathbf{f}' \in \mathbb{Z}_+^n$ is a vector with $\mathbf{f}' \neq \mathbf{f}(A)$ and $\|\mathbf{f}'\|_1 \leq m$. Then the "collision probability" $\Pr_{r \in_R \mathbb{F}_q}[\mathrm{BF}_q(r,\mathbf{f}') = \mathrm{BF}_q(r,\mathbf{f}(A))] \leq m/q$.*

The proof of this fact, along with other proofs, is deferred to the full version. This fingerprint implies a prescient protocol for a multi-set inclusion problem:

**Lemma 2.** *Let $A \subset \mathscr{U}$ be a set of size $n$ and let $B \subset \mathscr{U}$ be multi-set of size $t$. Let $B'$ be the set formed by removing all duplicate elements from $B$. Then, given a stream which begins with the elements of $A$ followed by the elements of $B$, there is a $(t\log t, \log t)$ prescient scheme that establishes whether $B' = A$.*

## 3 Warm-Up: Index and Selection

In this section, we present an online scheme for the SELECTION problem: Given desired rank $\rho \in [m]$, output an item $a_k$ from the stream $A = \langle a_1, \ldots, a_m\rangle \in [n]^m$, such that $|\{i : a_i < a_k\}| < \rho$ and $|\{i : a_i > a_k\}| \leq m - \rho$. We assume $m = \Theta(n)$ to simplify the statement of bounds. An easy $(\log m, \log m)$ prescient scheme is for the helper to give an answer $s$ as annotation at the start of the stream. The verifier need only count how many items in the stream are (a) smaller than $s$ and (b) greater than $s$. The verifier returns $s$

if the rank of $s$ satisfies the necessary conditions. Next, we present (almost) matching upper and lower bounds when only online annotation is allowed.

To do this, we first consider the online MA complexity of the communication problem of INDEX: Alice holds a string $x \in \{0,1\}^N$, Bob holds an integer $i \in [N]$, and the goal is for Bob to output $\text{INDEX}(x,i) := x_i$. The lower bound for SELECTION will follow from the lower bound for INDEX and a key idea for the SELECTION upper bound follows from the communication protocol for INDEX.

**Theorem 1 (Online MA complexity of INDEX).** *Let $h$ and $v$ be integers such that $hv \geq N$. There is a online MA protocol $\mathscr{P}$ for* INDEX*, with* $\text{hcost}(\mathscr{P}) \leq h$ *and* $\text{vcost}(\mathscr{P}) = O(v \log h)$*; and any online MA protocol $\mathscr{Q}$ for* INDEX *must have* $\text{hcost}(\mathscr{Q})\,\text{vcost}(\mathscr{Q}) = \Omega(N)$*. So, in particular,* $\text{MA}^{\rightarrow}(\text{INDEX}) = \tilde{\Theta}(\sqrt{N})$.

*Proof.* For the lower bound, we use the given online MA protocol $\mathscr{Q}$ to build a randomized one-way INDEX protocol $\mathscr{Q}'$. Let $h = \text{hcost}(\mathscr{Q})$. Let $\mathscr{B}(n,p)$ denote the binomial distribution with parameters $n$ and $p$, and let $k$ be the smallest integer such that $X \sim \mathscr{B}(k,1/3) \Rightarrow \Pr[X > k/2] \leq 2^{-h}/3$. A standard tail estimate gives $k = \Theta(h)$. Let $a(x,R)$ denote the message that Alice sends in $\mathscr{Q}$ when her random string is $R$, and let $b(\mathfrak{a},i,\mathfrak{h})$ be the bit Bob outputs upon receiving message $\mathfrak{a}$ from Alice and $\mathfrak{h}$ from Merlin. In the protocol $\mathscr{Q}'$, Alice chooses $k$ independent random strings $R_1,\ldots,R_k$ and sends Bob $a(x,R_1),\ldots,a(x,R_k)$. Bob then outputs 1 iff there exists a $h$-bit string $\mathfrak{h}$ such that $\text{MAJORITY}\,(b(a(x,R_1),i,\mathfrak{h}),\ldots,b(a(x,R_k),i,\mathfrak{h})) = 1$. Clearly, $\text{cost}(\mathscr{Q}') \leq k \cdot \text{vcost}(\mathscr{Q}) = O(\text{hcost}(\mathscr{Q})\,\text{vcost}(\mathscr{Q}))$. We claim that $\mathscr{Q}'$ is a $\frac{1}{3}$-error protocol for INDEX whence, by a standard lower bound (see, e.g., Ablayev [2]), $\text{cost}(\mathscr{Q}') = \Omega(N)$.

To prove the claim, consider the case when $x_i = 1$. By the correctness of $\mathscr{Q}$ there exists a suitable help message $\mathfrak{h}$ from Merlin that causes $\Pr[b(a(x,R),i,\mathfrak{h}) = 0] \leq 1/3$. Thus, by construction and our choice of $k$, the probability that Bob outputs 0 in $\mathscr{Q}'$ is at most $2^{-h}/3$. Now suppose $x_i = 0$. Then, *every* possible message $\mathfrak{h}$ from Merlin satisfies $\Pr[b(a(x,R),i,\mathfrak{h}) = 1] \leq 1/3$. Arguing as before, and using a union bound over all $2^h$ possible messages $\mathfrak{h}$, we see that Bob outputs 1 with probability at most $2^h \cdot 2^{-h}/3 = \frac{1}{3}$.

The upper bound follows as a special case of the two-party set-disjointness protocol in [1, Theorem. 7.4] since the protocol there is actually online. We give a more direct protocol which establishes intuition for our SELECTION result. Write Alice's input string $x$ as $x = y^{(1)} \cdots y^{(v)}$, where each $y^{(j)}$ is a string of at most $h$ bits, and fix a prime $q$ with $3h < q < 6h$. Let $y^{(k)}$ be the substring that contains the desired bit $x_i$. Merlin sends Bob a string $z$ of length at most $h$, claiming that it equals $y^{(k)}$. Alice picks a random $r \in \mathbb{F}_q$ and sends Bob $r$ and the strings $\text{BF}_q(r,y^{(1)}),\ldots,\text{BF}_q(r,y^{(v)})$, thus communicating $O(v \log q) = O(v \log h)$ bits. Bob checks if $\text{BF}_q(r,z) = \text{BF}_q(r,y^{(k)})$, outputting 0 if not. If the check passes, Bob assumes that $z = y^{(k)}$, and outputs $x_i$ from $z$ under this assumption. By Lemma 1, the error probability is at most $h/q < 1/3$.

*Remark 1.* The above lower bound argument in fact shows that an online MA protocol $\mathscr{P}$ for an *arbitrary* two-party communication problem $f$ satisfies $\text{hcost}(\mathscr{P})\,\text{vcost}(\mathscr{P}) = \Omega(\text{R}^{\rightarrow}(f))$. Thus, $\text{MA}^{\rightarrow}(f) = \Omega(\sqrt{\text{R}^{\rightarrow}(f)})$ where $\text{R}^{\rightarrow}(f)$ is the one-way, randomized communication complexity of $f$.

**Theorem 2.** *For any $h,v$ s.t. $hv \geq m$ there is a $(h \log m, v \log m)$ online scheme for* SELECTION *and any $(h,v)$ online scheme for* SELECTION *must have $hv = \Omega(m)$.*

*Proof.* Conceptually, the verifier builds a vector $\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{Z}_+^n$ where $r_k = |\{j \in [m] : a_j < k\}|$. This is done by inducing a new stream $A'$ from the input stream $A$: each token $a_j$ in $A$ causes virtual tokens $a_j + 1, a_j + 2, \ldots, n$ to be inserted into $A'$. Then $\mathbf{r} = \mathbf{f}(A')$; note that $\|\mathbf{r}\|_1 = O(m^2)$. As in the INDEX protocol, the vector $\mathbf{r}$ is arranged into $v$ subvectors of dimension $h$, and the verifier retains only fingerprints — based on a prime $q = O(m^2)$ — on each subvector. After the stream is seen, the helper claims that the answer is $s$, by providing the values of $r_i$ for all $i$ in the subvector containing $s$. The verifier fingerprints the provided block, and outputs $s$ if it agrees with their stored fingerprint, otherwise it returns $\perp$. For the lower bound, we use a standard reduction from the INDEX problem and this is deferred until the full version.

## 4 Frequency Moments and Frequent Items

In this section we consider properties of $\mathbf{f} = \{f_i : i \in [n]\}$ where $f_i$ is the frequency of the token "$i$" in the stream. In particular, the $k$th frequency moment is defined as $F_k = \sum_{i\in[n]} f_i^k$ and the frequent items are defined as the set $\{i : f_i > T\}$, for some threshold $T$. Both problems have a long history in the data streams literature. It is well known that in the traditional data stream model, exact computation of $F_k$ ($k \neq 1$) requires $\Omega(n)$ space. Even constant factor approximation requires $\Omega(n^{1-2/k})$ space [11].

**Frequent Items.** We prove results on finding exact and approximate frequent items. The approximate result relies on a powerful way that annotation can be used in conjunction with sketch based algorithms (such as Count-Sketch [12] and Count-Min [13]) and we expect this will have other applications. The approximate case is more complicated than the exact case and further discussion is deferred to the full version.

A prescient helper can list the set of claimed frequent items, along with their frequencies, for the verifier to check against the stream. But we must also ensure that the helper is not able to omit any items that exceed the threshold. Our result shows a compact witness set for the exact case, which leads to online schemes for the exact and approximate versions of the problem.

**Theorem 3.** *There exists a $(\phi^{-1}\log^2 m, \phi^{-1}\log^2 m)$ prescient scheme and a $(\phi^{-1}n^{\alpha}\log m, n^{1-\alpha}\log m)$ online scheme ($\alpha \in [0,1]$) for finding $\{i : f_i > T := \phi m\}$. Any $(h,v)$ online scheme for this must have $hv = \Omega(n)$.*

*Proof.* The lower bound follows from the hardness of INDEX and we omit the simple reduction from this presentation. For the upper bound consider a binary tree whose leaves are the elements of the universe $[n]$. Associate each node $v$ with the set of elements at the leaves of the the subtree rooted at $v$. Call this set $S(v)$ where $S(u) = \{i\}$ if $u$ is the $i$th leaf. Let $g(v) = \sum_{i\in S(v)} f_i$. Note that if $u$ is a node and $v$ is any ancestor of $u$, then $g(u) \leq g(v)$. Now observe that there is a witness set of size $O(\phi^{-1}\log n)$ to identify all leaves $i$ with $f_i > T$: this consists of the set $W$ of all such $i$s in addition to pairs of nodes $(u,v)$ such that $u$ is the child of $v$, and $g(u) \leq T$ but $g(v) > T$. Here, each pair $(u,v) \in W$ is witness to the fact that no leaves $i \in S(u)$ can have $f_i > T$. The sets $S(u)$ for such $u$ together with $\{i : f_i > T\}$ form a partition of $[n]$. Further, there can be at most $\phi^{-1}$ such nodes $v$ at any level of the binary tree, as the sum of $g(v)$ is at most $m$. This bounds the size of this witness set to $|W| = O(\phi^{-1}\log n)$. This leads to two schemes for the problem.

In the first,prescient scheme, the helper lists the members of $W$ and their corresponding frequencies. The verifier remembers this information, and ensures that it agrees with the frequencies in the stream. Assuming $m = \Omega(n)$ gives hcost = vcost = $\phi^{-1}\log^2 m$. In the second, online scheme, the $2n-1$ nodes in the tree are divided into $v$ groups of $h$ such that $hv \geq 2n$. The verifier keeps a fingerprint of the frequency vector of each group. After the stream is seen, the helper provides the witness set $W$, sorted by the natural order on nodes, plus the frequency vector of all groups containing items named in $W$. This totals $\min\{O(|W|h), n\}$ items, yielding a $(\min\{n\log m, h\phi^{-1}\log m\}, v\log m)$ online scheme. A subtlety here is that the output size can exceed the verifier's memory, so the verifier may output a partial result before returning $\perp$.

**Frequency Moments.** We now show a family of algorithms that exhibit an optimal verification/annotation trade-off for the exact computation of $F_k$. Our algorithm is inspired by the "algebrization" results of Aaronson and Wigderson [1] but the key idea can be traced back to classic interactive proof protocols of Lund et al. [28] and Shamir [31].

**Theorem 4.** *Suppose $h$ and $v$ are positive integers with $hv \geq n$. Then, for integers $k \geq 1$, there exists a $(k^2 h\log m, kv\log m)$ online scheme for computing $F_k$ exactly.*

*Proof.* Let $A$ be the input stream. We map the length $n$ vector $\mathbf{f}(A)$ into an $h \times v$ matrix $(f(x,y))_{x\in[h],y\in[v]}$, using any canonical bijection between $[n]$ and $[h]\times[v]$. Pick a prime $q \geq \max\{m^k, 3kh\}$; since $m \geq n$, this can be done while ensuring that $\log q = O(k\log m)$. We shall work in the field $\mathbb{F}_q$, which is safe because $q$ exceeds the maximum possible value of $F_k(A)$. Let $\tilde{f}(X,Y) \in \mathbb{F}_q[X,Y]$ be the unique polynomial satisfying $\deg_X(\tilde{f}) = h-1$, $\deg_Y(\tilde{f}) = v-1$ and $\tilde{f}(x,y) = f(x,y)$ for all $(x,y) \in [h]\times[v]$. The verifier picks a random $r \in \mathbb{F}_q$. As the stream is read, the verifier maintains a sketch consisting of the $v$ quantities $\tilde{f}(r,1),\ldots,\tilde{f}(r,v)$. Clearly, this sketch fits in $O(v\log q)$ bits of storage.

At the end of the stream, the annotator provides a polynomial $s'(X) \in \mathbb{F}_q[X]$ that is claimed to be equal to $s(X) := \sum_{y\in[v]} \tilde{f}(X,y)^k$, which has degree at most $k(h-1)$, thus using $O(kh\log q)$ bits of annotation. The verifier evaluates $s'(r)$ from the supplied annotation and computes $s(r) = \sum_{y\in[v]} \tilde{f}(r,y)^k$ from his sketch, checks that $s'(r) = s(r)$ and outputs $\perp$ if not. If the check passes, the verifier outputs $\sum_{x\in[h]} s'(x)$ as the final answer. Clearly, this answer is correct if the annotation was honest. Further, the verifier is fooled only if $s' \neq s$, but $s'(r) = s(r)$; the probability of this is at most $k(h-1)/q \leq \frac{1}{3}$, by choice of $q$.

It remains to show that the sketch can be computed incrementally in $O(v\log q)$ space. To maintain each $\tilde{f}(r,y)$ for $y \in [v]$, note that upon reading a new token $i \in [n]$ that maps to $(a,b) \in [h]\times[v]$, the necessary update is of the form $\tilde{f}(r,y) \leftarrow \tilde{f}(r,y) + p_{a,b}(r,y)$, where $p_{a,b}(X,Y) = \prod_{i\in[h]\setminus\{a\}}(X-i)(a-i)^{-1} \cdot \prod_{j\in[v]\setminus\{b\}}(Y-j)(b-j)^{-1}$. Since $p_{a,b}(r,y) = 0$ for any $y \in [v]\setminus\{b\}$, the verifier need only update the single value $\tilde{f}(r,b)$, by adding $p_{a,b}(r,b)$, upon reading this token. Note that using a table of $O(v)$ appropriate precomputed values, this update can be computed efficiently. For $h = v = \sqrt{n}$, this takes a constant number of arithmetic operations per update.

Numerous problems such as computing Hamming distance and Inner Product, and approximating $F_2$ and $F_\infty$, can be solved using $F_k$ as a primitive or using related techniques. We defer discussion to the full version. We next present lower bounds on the trade-off possible for computation of $F_k$.

**Theorem 5.** *Any $(h,v)$ scheme that exactly computes $F_k$ requires $hv = \Omega(n)$ and any $(h,v)$ scheme that approximates $F_k$ up to a constant factor requires $hv = \Omega(n^{1-5/k})$.*

These bounds are based on bounds we prove on the MA complexity of $\text{DISJ}_{n,t} : \{0,1\}^{nt} \to \{0,1\}$, the $t$-party communication problem defined as follows. The input is a $t \times n$ Boolean matrix, with Player $i$ holding the $i$th row, for $i \in [t]$. The desired output is $\wedge_{i=1}^{t} \vee_{j=1}^{n} \neg x_{ij}$, i.e., 1 iff the subsets of $[n]$ represented by the rows are disjoint. We call an input $\mathbf{x} = (x_{ij})_{i\in[t],j\in[n]}$ *valid* if every column of $\mathbf{x}$ has weight either 0 or 1 or $t$, and at most one column has weight $t$. Note that $\text{DISJ}_{n,t}$ is naturally related to frequency moments: for any valid input $\mathbf{x}$, $F_k(S) \geq t^k$ if $\text{DISJ}_{n,t}(\mathbf{x}) = 0$ and $F_k(S) \leq n$ if $\text{DISJ}_{n,t}(\mathbf{x}) = 1$ where $S$ is the multi-set $\{j : x_{ij} = 1\}$. The next theorem, a generalization of a result by Klauck [25], and reductions from $\text{DISJ}_{n,2}$ or $\text{DISJ}_{n,O(n^{1/k})}$ establish the first and second parts of Theorem 5 respectively in a straightforward manner. The next theorem also resolves a question of Feigenbaum et al. [17].

**Theorem 6.** *Let $\mathscr{P}$ be an $\varepsilon$-error MA protocol for $\text{DISJ}_{n,t}$, where $\varepsilon \leq 1/3$. Then* $\text{hcost}(\mathscr{P}) \cdot \text{vcost}(\mathscr{P}) = \Omega(n/t^4)$. *In particular,* $\text{MA}(\text{DISJ}_{n,t}) = \Omega(\sqrt{n}/t^2)$.

*Proof.* A rectangle is defined as a subset of inputs of the form $\mathscr{X}_1 \times \cdots \times \mathscr{X}_t$, where each $\mathscr{X}_i \subseteq \{0,1\}^n$ is a subset of all possible inputs for Player $i$. In deterministic communication protocols, the inverse image of any transcript of such a protocol must be a rectangle. Let $A = \text{DISJ}_{n,t}^{-1}(1)$ and $B = \text{DISJ}_{n,t}^{-1}(0)$.

**Lemma 3 (Alon-Matias-Szegedy [4], generalizing Razborov [30]).** *There exists distribution $\mu$ over valid inputs with 1) $\mu(A) = \mu(B) = 1/2$ and 2) $\mu(T \cap B) = (2e)^{-1}\mu(T \cap A) - t2^{-n/2t^4}$ for each rectangle $T$.* □

Assume $t = \omega(n^{1/4})$ since otherwise the bound is trivial. Put $h = \text{hcost}(\mathscr{P})$ and $v = \text{vcost}(\mathscr{P})$. An input $\mathbf{x} \in A$ is said to be *covered* by a message $\mathfrak{h}$ from Merlin if $\Pr_R[\text{out}(\mathscr{P};\mathbf{x},R,\mathfrak{h}) = 0] \leq \varepsilon$. By correctness, every such input must be covered, so there exists a help message $\mathfrak{h}^*$ that covers every input in a set $G \subseteq A$, with $\mu(G) \geq 2^{-h}\mu(A) = 2^{-h-1}$. Fix Merlin's message in $\mathscr{P}$ to $\mathfrak{h}^*$ and amplify the correctness of the resulting randomized Merlin-free protocol by repeating it $O(h)$ times and taking the majority of the outputs. This gives us a randomized protocol $\mathscr{P}'$ for $\text{DISJ}_{n,t}$ with communication cost $c = O(hv)$ whose error, on every input in $G \cup B$, is at most $2^{-2h}$. Let $\mu'$ denote the distribution $\mu$ conditioned on $G \cup B$. Note that, by condition (1) of Lemma 3,

$$\forall \mathbf{x} \in \{0,1\}^{nt} : \quad \text{either } \mu'(\mathbf{x}) = 0 \text{ or } \mu(\mathbf{x}) \leq \mu'(\mathbf{x}) \leq 2\mu(\mathbf{x})\,. \tag{1}$$

By fixing the random coins of $\mathscr{P}'$ we can obtain a deterministic protocol $\mathscr{Q}$, for $\text{DISJ}_{n,t}$, such that $\text{err}_{\mu'}(\mathscr{Q}) \leq 2^{-2h}$ and $\text{cost}(\mathscr{Q}) = c$. By the rectangle property, there exist disjoint rectangles $T_1, T_2, \ldots, T_{2^c}$ such that $\text{out}(\mathscr{Q};\mathbf{x}) = 1$ iff $\mathbf{x} \in \bigcup_{i=1}^{2^c} T_i$. Therefore

$$\sum_{i=1}^{2^c} \mu'(T_i \cap B) \leq 2^{-2h} \qquad (2) \qquad \text{and} \qquad \mu'\left(A \setminus \bigcup_{i=1}^{2^c} T_i\right) \leq 2^{-2h} \qquad (3)$$

By (1), $\mu'(A) = \mu'(G) \geq \mu(G) \geq 2^{-h-1}$. Using (1), and a rearrangement of (3):

$$\sum_{i=1}^{2^c} \mu(T_i \cap A) \;\geq\; \frac{1}{2}\sum_{i=1}^{2^c} \mu'(T_i \cap A) \;\geq\; \frac{1}{2}\left(\mu'(A) - 2^{-2h}\right) \;\geq\; 2^{-h-3}\,.$$

Suppose $c \le n/5t^4$ and $n$ is large enough. Applying condition (2) of Lemma 3 we get $\sum_{i=1}^{2^c} \mu(T_i \cap B) \ge 2^{-h-3}/(2e) - 2^c t 2^{-n/2t^4} \ge 2^{-h-6}$. However, by (1) and (2), we have $\sum_{i=1}^{2^c} \mu(T_i \cap B) \le 2^{-2h}$, a contradiction. Hence $hv = \Omega(c) = \Omega(n/t^4)$.

# 5 Graph Problems

In this section we consider computing properties of graphs on $n$ nodes, determined by a stream of $m$ edges [16, 21]. We present tight results for testing connectivity of sparse graphs, determining if a bipartite graph has a perfect matching, and counting triangles. Almost all proofs are deferred to the full version.

**Triangles via Matrix Multiplication.** Estimating the number of triangles in a graph has received significant attention because of its relevance to database query planning (knowing the degree of transitivity of a relation is useful when evaluating relational queries) and investigating structure properties of the web-graph [6,9,23]. In the absence of annotation, any single pass algorithm to determine if there is a non-zero number of triangles requires $\Omega(n^2)$ bits of space [6]. We show that the answer can be verified with $O(n^2)$ annotation in logarithmic space. The following theorem, proved using ideas from [6] coupled with Theorem 6, shows that this is best possible.

**Theorem 7.** *Any $(h,v)$ scheme for counting triangles must have $hv = \Omega(n^2)$.*

We now outline an online scheme with $\text{vcost} = O(\log n)$ and $\text{hcost} = O(n^2)$. A major subroutine of our algorithm is the verification of matrix multiplication in our model. That is, given $n \times n$ matrices $A, B$ and $C$, verify that $AB = C$. Our technique extends the classic result of Frievalds [18] by showing that if the helper presents the results in an appropriate order, the verifier needs only $O(\log n)$ bits to check the claim. Note that this much annotation is necessary if the helper is to provide $C$ in his stream.

**Theorem 8.** *There exists a $(n^2, \log n)$ online scheme for matrix multiplication.*

With this primitive, arbitrary matrix products $A_\ell, A_{\ell-1} \dots A_2 A_1$ are verified with $O(\ell n^2)$ annotation by verifying $A_{2,1} := A_2 A_1$, then $A_{3,2,1} := A_3 A_{2,1}$, etc. Matrix powers $A^\ell$ are verified with $O(n^2 \log \ell)$ annotation.

**Theorem 9.** *There is a $(n^2, \log n)$ online scheme for counting triangles.*

*Proof.* Denote the graph adjacency matrix by $A$, with $A_{i,i} := 0$. The helper lists $A_{v,w}$ and $A^2_{v,w}$ for all pairs $(v,w)$ in some canonical order. The verifier computes $\sum_{v,w} A_{v,w} A^2_{v,w}$ as the number of triangles. The verifier uses fingerprints to check that $A$ matches the original set of edges, and the protocol in Theorem 8 to ensure that $A^2$ is as claimed.

We also show that it is possible to trade-off the computation with the helper in a "smooth" manner. The approach is based on an observation of Bar-Yossef et al. [6]: The frequency moments of a derived stream can be expressed in terms of the number of triples of nodes with exactly $\{0,1,2,3\}$ edges between them. In small space we can induce a length $m(n-2)$ stream by replacing each edge $(u,v)$ by the set of triples $\{(u,v,w) : w \ne u,v\}$. It follows that the number of triangles can be expressed in terms of the frequency moments of this derived stream, as $(F_3 - 2F_2 + F_1)/12$. By using the protocol of Theorem 4, we obtain the following theorem.

**Theorem 10.** *There is a* $(n^{3\alpha}, n^{3-3\alpha})$ *online scheme for counting triangles* $(\alpha \in [0,1])$.

**Bipartite Perfect Matchings.** We now present an online scheme for testing whether a bipartite graph has a perfect matching. Graph matchings have been considered in the stream model [16,35] and it can be shown that any single pass algorithm for determining the exact size of the maximum matching requires $\Omega(n^2)$ space. We show that we can off-load this computation to the helper such that, with only $O(n^2)$ annotation, the answer can be verified in $O(\log n)$ space. This is shown to be best possible by combining a reduction from [16] coupled with Theorem 1.

**Theorem 11.** *There exists a* $(m, \log n)$ *online scheme for bipartite perfect matching and any* $(h, v)$ *online scheme for bipartite perfect matching requires* $hv = \Omega(n^2)$.

**Connectivity.** The problem of determining if a graph is connected was considered in the standard stream model [16, 21] and the multi-pass W-stream model [15]. In both models, it can be shown that any constant pass algorithm without annotations needs $\Omega(n)$ bits of space. In our model, the helper can convince a verifier with $O(\log n)$ space whether a graph is connected with only $O(m)$ annotation. This is the best possible for sparse graphs where $m = O(n)$ by combining a reduction from [16] with Theorem 1.

**Theorem 12.** *There exists a* $(m, \log n)$ *online scheme for connectivity and any* $(h, v)$ *online scheme for connectivity requires* $hv = \Omega(n)$ *even when* $m = O(n)$.

**Acknowledgements:** We thank Yael Gertner, Sampath Kannan, and Mahesh Viswanathan for sharing [19]. We also thank Sudipto Guha and T. S. Jayram for helpful discussions.

## References

1. S. Aaronson and A. Wigderson. Algebrization: a new barrier in complexity theory. In *ACM STOC*, 2008.
2. F. Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science*, 175(2):139–159, 1996.
3. G. Aggarwal, M. Datar, S. Rajagopalan, and M. Ruhl. On the streaming model augmented with a sorting primitive. In *IEEE FOCS*, 2004.
4. N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
5. L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory (preliminary version). In *IEEE FOCS*, 1986.
6. Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *ACM-SIAM SODA*, 2002.
7. P. Beame and D.-T. Huynh-Ngoc. On the value of multiple read/write streams for approximating frequency moments. In *IEEE FOCS*, 2008.
8. P. Beame, T. S. Jayram, and A. Rudra. Lower bounds for randomized read/write stream algorithms. In *ACM STOC*, 2007.
9. L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *ACM PODS*, 2006.
10. A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *ACM STOC*, 2008.

11. A. Chakrabarti, S. Khot, and X. Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE CCC*, 2003.
12. M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.
13. G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.
14. C. Demetrescu, B. Escoffier, G. Moruz, and A. Ribichini. Adapting parallel algorithms to the w-stream model, with applications to graph problems. In *MFCS*, 2007.
15. C. Demetrescu, I. Finocchi, and A. Ribichini. Trading off space for passes in graph streaming problems. In *ACM-SIAM SODA*, 2006.
16. J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.
17. J. Feigenbaum, S. Kannan, and J. Zhang. Annotation and computational geometry in the streaming model. Technical Report YALEU/DCS/TR-1249, Yale University, 2003.
18. R. Freivalds. Fast probabilistic algorithms. In *MFCS*, 1979.
19. Y. Gertner, S. Kannan, and M. Viswanathan. NP and streaming verifiers. *Manuscript*, 2002.
20. M. Grohe, A. Hernich, and N. Schweikardt. Randomized computations on large data sets: tight lower bounds. In *ACM PODS*, 2006.
21. M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, 1999.
22. W. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.
23. H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *COCOON*, 2005.
24. T. Kimbrel and R. K. Sinha. A probabilistic algorithm for verifying matrix products using $o(n^2)$ time and $\log_2 n + o(1)$ random bits. *Inf. Process. Lett.*, 45(2):107–110, 1993.
25. H. Klauck. Rectangle size bounds and threshold covers in communication complexity. In *IEEE CCC*, 2003.
26. E. Kushilevitz and N. Nisan. *Communication Complexity*. CUP, 1997.
27. F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios. Proof-infused streams: Enabling authentication of sliding window queries on streams. In *VLDB*, 2007.
28. C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.
29. S. Papadopoulos, Y. Yang, and D. Papadias. Cads: Continuous authentication on data streams. In *VLDB*, 2007.
30. A. Razborov. On the distributional complexity of disjontness. In *ICALP*, 1990.
31. A. Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.
32. M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *ACM-SIAM SODA*, 2004.
33. P. A. Tucker, D. Maier, L. M. L. Delcambre, T. Sheard, J. Widom, and M. P. Jones. Punctuated data streams, 2005.
34. K. Yi, F. Li, M. Hadjieleftheriou, G. Kollios, and D. Srivastava. Randomized synopses for query assurance on data streams. In *IEEE ICDE*, 2008.
35. M. Zelke. Weighted matching in the semi-streaming model. In *STACS*, pages 669–680, 2008.