Annotations in Data Streams

Amit Chakrabarti[*] Graham Cormode[†] Andrew McGregor[‡] Justin Thaler [§]

September 17, 2010

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, 26]. 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 [32, 34, 39]. A key issue is that we

[*]Dartmouth College. ac@cs.dartmouth.edu
[†]AT&T Labs–Research. graham@research.att.com
[‡]Microsoft Research, Silicon Valley. amcgreg@microsoft.com
[§]Harvard University, School of Engineering and Applied Sciences. jthaler@seas.harvard.edu

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, 30] 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* i.e., annotation that only depends on data that has arrived before the annotation is written. 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, 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 [11] 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 a $O(v \log m)$-space algorithm that uses $(\phi^{-1} h \log m)$ bits of online annotation and returns exactly the tokens whose frequency exceeds ϕm. 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. Additionally, we generalize the protocol for F_k to any *frequency-based function*, which is a function of the form $\sum_{i\in[n]} g(f_i)$ for some $g : \mathbb{N}_0 \to \mathbb{N}_0$. We obtain $(n^{2/3} \log n, n^{2/3} \log n)$ prescient protocols and $(n^{3/4} \log n, n^{3/4} \log n)$ online protocols for this important class of functions, as well as improved schemes for functions *based on low-frequencies* and for skewed data streams.

- *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, is triangle-free or bipartite, 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. Finally, for any h, v such that $hv \geq n^2$, we present $(h \log n, v \log n)$ protocols for determining whether a graph is connected or bipartite, or contains a perfect matching.

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, 18, 19]. 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, 25]. 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. [23] 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. [21] 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, 31]. The work of Aaronson and Widgerson [1] and Klauck [30] are particularly relevant. They resolve the MA complexity of two-party set disjointness — we extend some of their techniques to our streaming model. Lastly, recent work has considered which computations which can be verified relatively efficiently while permitting multiple rounds of interaction between the parties [24].

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 [38]. Work on stream outsourcing studies the problem of verifying that a claimed "grouping" corresponds to the input data [39]. 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 [32] 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 $\mathcal{P}$, using a public random string R, eventually outputting a bit $\mathrm{out}(\mathcal{P};\mathbf{x},R,\mathfrak{h})$. To clarify, R is not known to Merlin at the time he writes $\mathfrak{h}$. An MA protocol is δ-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 $\Pr_R[\mathrm{out}(\mathcal{P};\mathbf{x},R,\mathfrak{h}(\mathbf{x})) = 0] \leq \delta$.
2. If $f(\mathbf{x}) = 0$ then $\forall \mathfrak{h}'$ $\Pr_R[\mathrm{out}(\mathcal{P};\mathbf{x},R,\mathfrak{h}') = 1] \leq \delta$.

We define $\mathrm{err}(\mathcal{P})$ to be the minimum δ such that the above conditions are satisfied. We also define the *help cost* $\mathrm{hcost}(\mathcal{P})$ to be the maximum length of $\mathfrak{h}$, over all $\mathbf{x}$, and the *verification cost* $\mathrm{vcost}(\mathcal{P})$ 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 $\mathcal{P}$ to be $\mathrm{cost}(\mathcal{P}) = \mathrm{hcost}(\mathcal{P}) + \mathrm{vcost}(\mathcal{P})$. We then define the δ-error MA-complexity of f as $\mathrm{MA}_\delta(f) = \min\{\mathrm{cost}(\mathcal{P}) : \mathcal{P}$ is an MA protocol for f with $\mathrm{err}(\mathcal{P}) \leq \delta\}$. 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 ith message is only a function of the first i inputs. To make this precise we need to amend the definition of δ-error: An online-MA protocol is δ-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 $\Pr_R[\mathrm{out}(\mathcal{P};\mathbf{x},R,\mathfrak{h}_1(x_1),\mathfrak{h}_2(x_1,x_2),\ldots,\mathfrak{h}_t(x_1,\ldots,x_t)) = 0] \leq \delta$.
2. If $f(\mathbf{x}) = 0$ then $\forall \mathfrak{h}' = (\mathfrak{h}'_1,\mathfrak{h}'_2,\ldots,\mathfrak{h}'_t)$ $\Pr_R[\mathrm{out}(\mathcal{P};\mathbf{x},R,\mathfrak{h}') = 1] \leq \delta$.

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

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 $\mathbf{x} \in \mathcal{U}^t$ for some universe $\mathcal{U}$, 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., $\{a : |a - f(\mathbf{x})| \leq \varepsilon f(\mathbf{x})\}$.

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: $\mathfrak{h}_1 : \mathcal{U}^t \to \{0,1\}^*, \ldots, \mathfrak{h}_t : \mathcal{U}^t \to \{0,1\}^*$. Let $\mathfrak{h}(\mathbf{x}) := (\mathfrak{h}_1(\mathbf{x}), \ldots, \mathfrak{h}_t(\mathbf{x}))$. We then consider a randomized protocol, $\mathcal{A}$, 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 δ-error if $\Pr_R[\mathrm{out}(\mathcal{A};\mathbf{x},R,\mathfrak{h}) \notin C(f(\mathbf{x}))] \leq \delta$ and $\Pr_R[\mathrm{out}(\mathcal{A};\mathbf{x},R,\mathfrak{h}') \neq \perp] \leq \delta$ for any $\mathfrak{h}' = (\mathfrak{h}'_1,\mathfrak{h}'_2,\ldots,\mathfrak{h}'_t) \neq \mathfrak{h}(\mathbf{x})$. We define $\mathrm{err}(\mathcal{A})$ to be the minimum δ such that the above conditions are satisfied. We define the *help cost* $\mathrm{hcost}(\mathcal{A})$ to be the maximum length of $\sum_i |\mathfrak{h}_i|$, over all $\mathbf{x}$, and the *verification cost* $\mathrm{vcost}(\mathcal{A}) = w$. We say that $\mathcal{A}$ and $\mathfrak{h}$ forms an (h,v) *prescient scheme* if $\mathrm{hcost}(\mathcal{A}) = O(h+1)$, $\mathrm{vcost}(\mathcal{A}) = O(v+1)$ and $\mathrm{err}(\mathcal{A}) < 1/3$.

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

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 $\text{BF}_q(r,\mathbf{f}) := \prod_{j=1}^{n}(r-j)^{f_j}$. To make fingerprints, we choose q based on an *a priori* bound m on $\|\mathbf{f}\|_1$.

Lemma 2.1. *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 $\text{BF}_q(r,\mathbf{f}(A))$ 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" $\Pr_{r\in_R\mathbb{F}_q}[\text{BF}_q(r,\mathbf{f}') = \text{BF}_q(r,\mathbf{f}(A))] \leq m/q$.*

Proof. To compute the fingerprint in streaming fashion, express $\text{BF}_q(r,\mathbf{f}(A)) = \prod_{i=1}^{m}(r-a_i)$. The bound on the collision probability follows from the fact that for any $\mathbf{f} \in \mathbb{Z}_+^n$, the polynomial $\text{BF}_q(X,\mathbf{f}) \in \mathbb{F}_q[X]$ has degree at most $\|\mathbf{f}\|_1$. □

This fingerprint implies a prescient protocol for a multi-set inclusion problem:

Lemma 2.2. *Let $A \subset \mathcal{U}$ be a set of size n and let $B \subset \mathcal{U}$ 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 $B' = A$.*

Proof. Note that we may assume that $t \geq n$ otherwise $A \neq B'$ and this is easy to certify. The helper annotates each $a \in A$ with the multiplicity, f_a, of a in B. This allows the verifier to incrementally construct a fingerprint of the set $\{(a,i) : a \in A, i \in [f_a]\}$. The prover annotates the jth occurrence of $b \in B$ with j, which allows the verifier to incrementally construct a fingerprint of the set $\{(a,i) : a \in B', i \in [f_a]\}$. The verifier accepts if the two fingerprints match. □

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

Proof. For the lower bound, we use the given online MA protocol $\mathcal{Q}$ to build a randomized one-way INDEX protocol $\mathcal{Q}'$. Let $h = \text{hcost}(\mathcal{Q})$. Let $\mathcal{B}(n,p)$ denote the binomial distribution with parameters n and p, and let k be the smallest integer such that $X \sim \mathcal{B}(k, 1/3) \Rightarrow \Pr[X > k/2] \leq 2^{-h}/3$. A standard tail estimate gives $k = \Theta(h)$. Let $a(x,R)$ denote the message that Alice sends in $\mathcal{Q}$ 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 $\mathcal{Q}'$, 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, $\text{cost}(\mathcal{Q}') \leq k \cdot \text{vcost}(\mathcal{Q}) = O(\text{hcost}(\mathcal{Q})\,\text{vcost}(\mathcal{Q}))$. We claim that $\mathcal{Q}'$ is a $\frac{1}{3}$-error protocol for INDEX whence, by a standard lower bound (see, e.g., Ablayev [2]), $\text{cost}(\mathcal{Q}') = \Omega(N)$.

To prove the claim, consider the case when $x_i = 1$. By the correctness of $\mathcal{Q}$ 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 $\mathcal{Q}'$ 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 2.1, the error probability is at most $h/q < 1/3$. □

Remark. *The above lower bound argument in fact shows that an online MA protocol $\mathcal{P}$ for an* arbitrary *two-party communication problem f satisfies* $\text{hcost}(\mathcal{P})\,\text{vcost}(\mathcal{P}) = \Omega(\text{R}^{\rightarrow}(f))$. *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 3.2. *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: Given string x, we form the stream over $[2n]$ by placing $a_j = 2j - x_j$ in the stream for each j. Further, given the integer index i, we place i copies of $2n$ and $m - i$ copies of 1. Consequently, the median of this length $2m$ stream is $2i - x_i$, from which the value of x_i can be recovered. To complete the proof, observe that any streaming scheme to solve this problem would imply a communication protocol with the same cost; and that all players can perform this reduction online without extra space or annotation.

□

Notice that in the above scheme the information computed by the verifier is independent of ρ, the rank of the desired element. Therefore these algorithms work even when ρ is revealed at the end of the stream.

4 Frequent Items, Frequency Moments, and Generalizations

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 kth 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 [12].

4.1 Frequent Items

We provide results on finding exact and approximate frequent items. 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 4.1. *There exists a $(\phi^{-1}\log m, \phi^{-1}\log m)$ 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 problem must have $hv = \Omega(n)$.*

Proof. For the upper bound, consider a binary tree $\mathcal{T}$ whose leaves are the elements of the universe $[n]$. We will specify a witness set W of size $O(\phi^{-1})$ to identify to identify all leaves i with $f_i > T$; we base W on the concept of *Hierarchical Heavy Hitters* (*HHH*s) [14]. Below, we refer to the set of Hierarchical Heavy Hitters as H.

We define H inductively, beginning with the leaves and working our way to the root r. We include a leaf in H if its frequency exceeds T. Let v be a node at distance l from r (i.e. at level l of $\mathcal{T}$), and assume inductively that we have determined all *HHH*s at levels greater than l. Let H_u denote the set of descendants of u that have been included in H, and associate each node u with the set of elements at the leaves of the subtree rooted at u. Call this set $L(u)$ where $L(u) = \{i\}$ if u is the ith leaf. Finally, define $S(u) := L(u) \setminus \left(\cup_{v\in H_u} L(v)\right)$. Intuitively, $S(u)$ is the set of leaves in $L(u)$ that have not already contributed their frequency to an *HHH* descendant of u. Define the *conditioned frequency* (alternatively, the *conditioned subtree count*) of u as $g(u) := \sum_{i\in S(u)} f_i$; we include u in H if $g(u) > T$. Observe there are at most ϕ^{-1} items in H since $T = \phi m$: each leaf contributes its frequency to $g(u)$ for exactly one $u \in H$, and therefore $|H|T \leq \sum_{u\in H} g(u) \leq m$.

We now define our witness set W as all leaves i in H in addition to all nodes u such that u's parent is in H but u is not in H. Observe that each node $u \in W$ is witness to the fact that no leaves $i \in S(u)$ can have $f_i > T$; we also include the root r in W to account for any leaves that are not descendants of any node in H. The sets $S(u)$ for $u \in W$ form a partition of $[n]$. Note that $|W| = O(\phi^{-1})$ since $|H| \leq \phi^{-1}$.

This leads to two schemes for the problem. In the first, prescient scheme, the helper lists all nodes $u \in W$ sorted by the natural order on nodes, and the verifier remembers this information. The verifier may then compute the conditioned frequency of each such u using space $O(|W|\log n) = O(\phi^{-1}\log n)$: each time an item i appears in the stream, the verifier determines the unique $u \in W$ such that $i \in S(u)$ (u is simply the ancestor of i in W farthest from the root), and increments $g(u)$. The verifier checks that $g(i) > T$ for all leaf nodes $i \in W$, and that $g(u) \leq T$ for all internal nodes in W and outputs $\perp$ otherwise.

The second, online scheme is more involved. In the online setting, it is no longer possible for the verifier to track the conditioned frequency of each node in W while observing the stream. However, it *is* possible for the verifier to track (fingerprints of) a related quantity for each node v: namely the *unconditioned frequency* (alternatively, *unconditioned subtree count*), which we now define.

For each node u in $\mathcal{T}$, recall that $L(u)$ denotes the leaves of the subtree rooted at u, and H_u denotes the descendants of u that are in H. Define the unconditioned frequency of u as $f(u) = \sum_{i \in L(u)} f_i$; one may think of $f(u)$ as the frequency of u in the stream, even though u is an internal node in $\mathcal{T}$. Observe that there is a simple relationship between the conditioned and unconditioned counts of u, namely $g(u) = f(u) - \sum_{v \in H_u} g(v)$. The verifier may exploit this relationship to force the helper to provide the true conditioned frequencies for each node $u \in W$.

The $2n-1$ nodes in the tree are divided into v groups of h such that $hv \geq 2n$, as in the simple protocol for INDEX (Theorem 3.1). While observing the stream, the verifier keeps a fingerprint of the vector of unconditioned frequencies of each group. In doing so, the verifier is essentially treating each entry i in the stream as an update to (the unconditioned counts of) the $\log n$ ancestors of i in $\mathcal{T}$.

For any node u, let $\mathbf{v}(u)$ denote the vector corresponding to u's group. After the stream is seen, the helper provides the witness set W, beginning with the leaves in W and working level by level towards the root. While processing the reported nodes in W, the verifier will modify (the fingerprints of) each group's vector in such a way that when each node $u \in W$ is presented, the entry corresponding to u in $\mathbf{v}(u)$ will equal $g(u)$. For each internal node u in W, the helper also presents the parent of u, and the conditioned subtree counts for both nodes. The verifier can ensure that the conditioned frequency of u, $g(u)$, is below T, indicating that no other nodes in the subtree of u have significant (conditioned) frequency.

The verifier will identify certain nodes u as being members of H, when $g(u)$ exceeds T. When processing $(u, g(u))$, the verifier treats this as a "deletion" of $g(u)$ occurrences of u and of each ancestor v of u (note that fingerprints can handle deletions as well as insertions). That is, for each ancestor v of u, the verifier subtracts $g(u)$ from the appropriate entry of $\mathbf{v}(v)$ by modifying the fingerprint of $\mathbf{v}(v)$ accordingly. As a result, when each node $u \in W$ is presented by the helper, the entry corresponding to u in $\mathbf{v}(u)$ is equal to $f(u) - \sum_{v \in H_u} f(v) = g(u)$.

The helper is further required to follow each pair $(u, g(u))$ with all the entries of the vector $\mathbf{v}(u)$ (i.e. the group containing u, accounting for all deletions that the verifier has simulated so far). If the helper does not faithfully provide the vector $\mathbf{v}(u)$, a fingerprint of the claimed vector will fail to match the verifier's fingerprint with high probability. Consequently, the helper is forced to provide the true conditioned frequencies of each node u in W.

In total, the verifier requires space $v \log n$ to maintain v fingerprints, and the helper needs to provide $\min\{O(|W|h), n\}$ items and (conditioned) counts, 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 $\perp$.

The lower bound follows from the hardness of INDEX: a string x of length n (assume n is odd) induces a stream A, by placing $a_j = 2j - x_j$ in the stream for each j. Then, given index i, we place $n-1$ copies of $2i$ in the stream. Observe that if $x_i = 0$ then $f_{2i} > m/2$, else $f_{2i} < m/2$. Thus, determining the frequent items for $T = \frac{1}{2}m$ solves INDEX, proving the bound.

JT comment: If we wanted a lower bound that applied to prescient protocols as well, we could simply reduce from Disjoint instead of Index. The reduction from Disjoint is equally simple, and we wouldn't need to full power of Theorem 4.5 – we'd just need Klauck's 2003 result that the MA

complexity of two-party Disjoint is $\Omega(\sqrt{n})$. □

In many cases, it suffices to find a set of *approximate* frequent items: these include all items with $f_i > \phi m$ and no items with $f_i < (\phi - \varepsilon)m$ for parameters ε, ϕ. Solutions to this problem in the traditional streaming model are often based on "sketch" algorithms. We define an *integer linear sketch* broadly as any summary $\mathbf{v}$ which can be computed as $\mathbf{v} = S\mathbf{f}(A)$ for a "sketch matrix" S with integral entries. Such sketches include instantiations of the Johnson-Lindenstrauss transform [27], Count-Sketch [13], and Count-Min [16]. Each stream token i increments $\mathbf{v}$ by $S\mathbf{e}_i$, where $\mathbf{e}_i$ is the vector that is 1 in location i and 0 elsewhere. The sketch can be fingerprinted: each update multiplies the fingerprint by $\mathrm{BF}_q(r, S\mathbf{e}_i)$. This observation means that the helper can annotate (parts of) $\mathbf{v}$ at the end of the stream, for verification. However, to define an efficient scheme, we also need to show: (1) the verifier can compute $S\mathbf{e}_i$ in small space, so S must have a compact representation; and (2) the verifier must be able to extract the result from $\mathbf{v}$ in a streaming fashion, in space sublinear in the size of the *sketch*.

We use ideas from verifying exact frequent items to build a scheme for verifying approximate frequent items via sketching. Count-Sketch [13] defines a "basic" sketch of length w via two pairwise independent hash functions $b_\ell : [n] \to [w]$, and $c_\ell : [n] \to \{-1, +1\}$. The sketch vector $\mathbf{v}$ is defined by $\mathbf{v}_{\ell,j} = \sum_{i: b_\ell(i)=j} f_i c_\ell(i)$, and the basic estimate of the frequency of i is $\hat{f}_{i,\ell} = c_\ell(i)\mathbf{v}_{\ell, b_\ell(i)}$. This satisfies $|\hat{f}_{i,\ell} - f_i| = O((F_2/w)^{1/2})$ with constant probability; to reduce the error probability, the median of the basic estimates from d basic sketches with different hash functions is taken, $\hat{f}_i = \mathrm{median}_{1 \le \ell \le d} \hat{f}_{i,\ell}$. Count-Min is essentially Count-Sketch with $c_\ell(i) := 1$ for all ℓ. It promises $|\hat{f}_{i,\ell} - f_i| = O(F_1/w)$ [16].

Theorem 4.2. *There exists an $(s \log n \log m, \log m)$ online scheme to verify the approximate frequent items found by Count-Sketch or Count-Min sketches of size s.*

Proof. Given a threshold T, the set of approximate frequent items are $\{i : \hat{f}_i > T\}$. To deal with the presence of error in the frequency estimates obtained from Count-Sketch or Count-Min sketches, we use a simplified version of the witness-tree approach from Theorem 4.1 to ensure that no items are omitted. This simplified witness set W' is based on (approximate) unconditioned frequencies instead of conditioned frequencies, and will have size $O(\phi^{-1} \log n)$, larger than W by a $\log n$ factor.

Specifically, consider an expanded set of items that includes the set of tree nodes u in $\mathcal{T}$ and their corresponding unconditioned frequencies $f(u)$ (recall $f(u)$ is the sum of the frequencies of all leaves in $L(u)$, the subtree rooted at u). The helper and verifier now keep a sketch $\mathbf{v}^k$ for each level k of the tree, to obtain *estimated* unconditioned frequencies $\hat{f}(u)$ for each node u in the tree. We henceforth assume that $\hat{f}(u) = f(u) \pm \varepsilon m$; when using sketches with $d = O(\log n)$, this holds for each i with probability at least $1 - 1/16n$, and so it holds over *all* $2n$ frequencies with probability at least $7/8$.

The simplified witness set W', given threshold T, consists of all leaves i with $\hat{f}_i > T$ in addition to pairs of nodes (u, v) such that u is the child of v, and $\hat{f}(u) \le T$ but $\hat{f}(v) > T$. Here, each pair $(u, v) \in W$ is witness to the fact that no leaves $i \in L(u)$ can have $f_i > T$. The sets $L(u)$ for such u together with $\{i : f_i > T\}$ cover all of $[n]$. Further, there can be at most ϕ^{-1} such nodes v at any level of the binary tree, as the sum of $\hat{f}(v)$ is at most $(1+\varepsilon)m$. This bounds the size of this witness set to $|W'| = O(\phi^{-1} \log n)$ if $\varepsilon < \frac{\phi}{2}$.

The verifier can validate this witness set W over the full set of nodes and their estimated unconditioned frequencies as follows. By presenting the set of nodes v in W in order of $\min L(v)$, the verifier can ensure that the nodes identified do cover all of $[n]$ as required (and hence that no high frequency items are omitted). If the helper provides for each node $v \in W$ the information about v contained in the sketch, as $(v, \hat{f}_v, \hat{f}_{v,1}, \ldots \hat{f}_{v,d})$ the verifier can check that $\hat{f}_v$ is above or below T as appropriate. The verifier ensures that $\hat{f}_v$ is derived correctly from the d values of $\hat{f}_{v,\ell}$ (using $O(d)$ working space). The verifier also incrementally builds a fingerprint of the set $B = \{(v, \ell, \hat{f}_{v,\ell})\}$. At the end of the annotation, the helper lists the entries of each sketch $\mathbf{v}^k_{\ell,j}$ in order and tags each entry with the set of v's for which it has been used to make an estimate. The verifier builds a fingerprint of the tuples $(v, \ell, c_\ell(v)\mathbf{v}^k_{\ell,b_\ell(v)})$, and checks that it matches the fingerprint of B (this is essentially an instance of the multiset equality protocol in Lemma 2.2). The verifier fingerprints also the (untagged) sketch to check it matches the verifier's fingerprinted sketch built from the input stream.

The total amount of annotation is $O(s \log n)$ sketch entries, from the $\log n$ sketches of size s. The verifier needs to remember d estimated frequencies (to verify their median) and $O(\log n)$ fingerprinted sketches (one for each level). We mention that if $\phi \gg \varepsilon$, then the verifier only needs to inspect a small fraction of the sketch entries to verify the frequent items. In this case, one can obtain a tradeoff: write the sketch as an array of $h \times v$ entries, so that $hv \geq s$. The verifier can create v fingerprints each summarizing h entries of the sketch. To verify, the helper modifies the above algorithm to only present those blocks of h entries which include a value that needs to be seen by the verifier. In total, to verify $O(|W'|)$ approximate frequencies requires verifying $O(\phi^{-1} d \log n)$ entries, giving an $(\phi^{-1} h \log m \log^2 n, v \log m)$ online scheme. □

Other algorithms find all items i such that $\hat{f}_i \geq \phi F_2^{1/2}$.These can also be adapted to our setting using similar ideas, and verified in logarithmic space with annotation proportional to the sketch size.

4.2 Protocols for 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. [33] and Shamir [36].

Theorem 4.3. *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 $p_{a,b}$ is the Lagrange polynomial

$$p_{a,b}(X,Y) = \prod_{i \in [h] \setminus \{a\}} (X-i)(a-i)^{-1} \cdot \prod_{j \in [v] \setminus \{b\}} (Y-j)(b-j)^{-1}.$$

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 in a small amount of time. For $h = v = \sqrt{n}$, this takes a constant number of arithmetic operations per update without affecting the asymptotic space cost. $\square$

Numerous problems such as computing Hamming distance and Inner Product, and approximating F_2 and F_∞, can be solved using F_k as a primitive or using related techniques.

Approximate F_2. F_2 can be approximated up to a $1+\varepsilon$ factor from an integer linear sketch of size $O(1/\varepsilon^2)$. In particular, if $\mathrm{CS}_w(A)$ denotes a length-w Count-Sketch vector of the stream A built using 4-wise independent hash functions, then $F_2(\mathrm{CS}_w(A))$ estimates $F_2(A)$ with relative error $\varepsilon = w^{-1/2}$ with constant probability [37]. Thus, if the verifier and helper have access to a source of public randomness to define the hash functions used by the sketch (or we extend the model to allow the verifier to send the description of the randomly chosen hash functions to the helper at the start of the protocol), the above F_2 scheme yields a $(\varepsilon^{-2\alpha}\log m, \varepsilon^{2\alpha-2}\log m)$ online scheme for any $\alpha \in [0,1]$. This follows from the combination of the algebrization approach with the observation that the verifier can track linear updates to their sketch efficiently.

Approximate F_∞. Recall that $F_\infty = \max_i f_i$ and note that $F_\infty^t \le F_t \le nF_\infty^t$. Hence, if $t = \log n/\log(1+\varepsilon)$, then $(F_t)^{1/t}$ is at most a factor $1+\varepsilon$ from F_∞. This yields a $((\frac{1}{\varepsilon}\log n)^2 h \log m,\, (\frac{1}{\varepsilon}\log n) v \log m)$ online scheme for approximating F_∞ for any h, v such that $hv \ge n$.

Inner Product and Hamming Distance. Consider a stream consisting of length N binary string $\mathbf{x}$ followed by length N binary string $\mathbf{y}$. Exact computation of F_2 implies online schemes for certain functions of $\mathbf{x}$ and $\mathbf{y}$. For example, the inner-product $\mathbf{x}\cdot\mathbf{y}$ is $(F_2(x+y) - F_2(x) - F_2(y))/2$ and the Hamming distance is $|\{i : x_i = 1\}| + |\{i : y_i = 1\}| - 2\mathbf{x}\cdot\mathbf{y}$. Hence we get $(N^\alpha \log N, N^{1-\alpha}\log N)$ online schemes for both functions, for any $\alpha \in [0,1]$. Alternately, the above approach can be used to directly generate protocols for these problems with the same bounds.

Convex Hull on a 2D Grid. Consider an instance of the convex hull problem where all input points P fall on the intersection points of a two-dimensional grid defining g possible point locations. Let C be the convex hull of a stream of points. Then there exists a $((|C| + g^\alpha)\log n, (|C| + g^{1-\alpha})\log n)$ online protocol to report the convex hull. The helper provides the claimed hull C', which the verifier can store exactly, and verify that it is indeed convex. Define $c(C)$ as the set of (grid) points contained within a convex shape C, and observe that it is easy to enumerate (but not store) $c(C)$ in space $O(|C|)$. The verifier then must establish that $C' \subseteq P$, and that $P \subseteq c(C')$. Both these subset tests can be verified efficiently. Consider sets X, Y, represented as characteristic vectors x, y s.t. $x_i = 1$ iff $i \in X$ and 0 otherwise. Then

$$X \subseteq Y \iff F_2(y-x) = |Y| - |X|.$$

Consequently, the helper can run the above protocol on vectors over the g grid points to allow the verifier to agree that the two containments hold. As described, this protocol requires that P should contain no duplicate points; we explain later how to use our protocol for frequency-based functions to handle the situation when each point in P is duplicated a small number of times.

4.3 Lower Bounds on Frequency Moments

We next present lower bounds on the trade-off possible for computation of F_k.

Theorem 4.4. *Any (h,v) scheme (online or prescient) that exactly computes F_k requires $hv = \Omega(n)$ and any (h,v) scheme that approximates F_k up to a constant factor (online or prescient) 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 ith 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 [30], and reductions from $\text{DISJ}_{n,2}$ or $\text{DISJ}_{n,O(n^{1/k})}$ establish the first and second parts of Theorem 4.4 respectively in a straightforward manner. The next theorem also resolves a question of Feigenbaum et al. [21].

Theorem 4.5. *Let $\mathcal{P}$ be an ε-error MA protocol for $\text{DISJ}_{n,t}$, where $\varepsilon \leq 1/3$. Then* $\text{hcost}(\mathcal{P}) \cdot \text{vcost}(\mathcal{P}) = \Omega(n/t^4)$. *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 $\mathcal{X}_1 \times \cdots \times \mathcal{X}_t$, where each $\mathcal{X}_i \subseteq \{0,1\}^n$ 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 4.6 (Alon-Matias-Szegedy [4], generalizing Razborov [35])**.** *There exists a distribution μ 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 $h = \text{hcost}(\mathcal{P})$ and $v = \text{vcost}(\mathcal{P})$. An input $\mathbf{x} \in A$ is said to be *covered* by a message $\mathfrak{h}$ from Merlin if $\Pr_R[\text{out}(\mathcal{P};\mathbf{x},R,\mathfrak{h}) = 0] \leq \varepsilon$. 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 $\mathcal{P}$ 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 $\mathcal{P}'$ 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 μ' denote the distribution μ conditioned on $G \cup B$. Note that, by condition (1) of Lemma 4.6,

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

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

$$\sum_{i=1}^{2^c} \mu'(T_i \cap B) \leq 2^{-2h} \tag{2}$$

and

$$\mu'\left(A \setminus \bigcup_{i=1}^{2^c} T_i\right) \leq 2^{-2h} \tag{3}$$

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 4.6 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 $hv = \Omega(c) = \Omega(n/t^4)$. □

4.4 Frequency-Based Functions

It is natural to ask whether the F_2 algorithm of Theorem 4.3 generalizes to more complicated functions. We demonstrate that this is indeed the case by presenting non-trivial algorithms for the class of all *frequency based functions*. A frequency based function is any function $\mathfrak{f}$ on frequency vectors $\mathbf{f} = (f_1, \ldots, f_n)$ of the form $\mathfrak{f}(\mathbf{f}) = \sum_{i\in[n]} g(f_i)$ for some $g : \mathbb{N}_0 \to \mathbb{N}_0$. We assume $g(x) \le n^c$ for some constant c, so that all values in the range of g and $\mathfrak{f}$ require $O(\log n)$ bits to represent. If there are constants C_1 and C_2 such that $g(x) = C_1$ for all $x \ge C_2$, then we say $\sum_{i\in[n]} g(f_i)$ is *based on low frequencies*.[1]

Frequency-based functions have a number of important special cases, including frequency moments, F_0 (the number of distinct items in the stream), and point and range queries on the frequency distribution, and can also be used to compute F_∞, the highest frequency in the frequency vector. These functions occupy an important place in the streaming world: Alon, Matias, and Szgedy asked for a precise characterization of which frequency-based functions can be approximated efficiently in the standard streaming model in their seminal paper [4], and Braverman and Ostrovsky [9] recently gave a zero-one law for approximating monotonically increasing functions of frequencies that are zero at the origin – this can be contrasted with our result that in the annotation model, *all* frequency-based functions have non-trivial exact protocols.

Theorem 4.7. *Suppose $m = \Theta(n)$. Let $\mathfrak{f}$ be any frequency-based function. Then $\mathfrak{f}$ has an $(n^{2/3}\log n, n^{2/3}\log n)$ prescient protocol, and an $(n^{3/4}\log n, n^{3/4}\log n)$ online protocol. Additionally, if $\mathfrak{f}$ is based on low-frequencies, then $\mathfrak{f}$ has an $(n^{2/3}\log n, n^{2/3}\log n)$ online protocol.*

Proof. We first describe the prescient protocol. Let A be the input stream. As in the F_2 algorithm, we shall work in the field $\mathbb{F}_q$ for a sufficiently large prime q, and we map the length n vector $\mathbf{f}(A)$ into an $h \times v$ matrix $(f(x,y))_{x\in[h],y\in[v]}$, where h and v are parameters to be specified later. Let $\tilde{f}_1(X,Y) \in \mathbb{F}_q[X,Y]$ be the unique polynomial satisfying $\deg_X(\tilde{f}_1) = h-1$, $\deg_Y(\tilde{f}_1) = v-1$ and $\tilde{f}_1(x,y) = f(x,y)$ for all $(x,y) \in [h]\times[v]$. The verifier picks a random $r \in \mathbb{F}_q$, and maintains a sketch consisting of the v quantities $\tilde{f}_1(r,1), \ldots, \tilde{f}_1(r,v)$ as the stream is read.

The goal is to compute $\sum_{x,y\in[h]\times[v]} g(\tilde{f}_1(x,y))$. As a first attempt at generalizing the F_2 protocol, we could have the helper send a polynomial to the verifier claimed to be

$$s_1(X) = \sum_{y=1}^{v} \tilde{g}_1 \circ \tilde{f}_1(X,y),$$

where $\tilde{g}_1$ is defined through interpolation as the unique degree-m polynomial that agrees with g on $[m] := \{0,1,\ldots,m\}$. Then the verifier can compute $h(\mathbf{f}(A)) = \sum_{x\in[h]} s_1(x)$, and we can ensure the polynomial sent by the helper is as claimed in much the same manner as the F_2 protocol. Below, we refer to this protocol as the *polynomial-agreement protocol*. The problem with this approach is that $\tilde{g}_1 \circ \tilde{f}_1$ has degree mh, and therefore s_1 requires up to mh words to represent—it would be more efficient for the helper to just repeat the stream in sorted order!

The solution is to reduce the degree of $\tilde{g}_1$ by removing the *heavy hitters* from A with the aid of the prover. That is, we run the prescient heavy hitters protocol of Theorem 4.1 to determine $H := \sum_{i\in S} g(f_i)$

[1]In full generality, we can obtain improves protocols for functions for which $C_2 = o(n^{1/12})$.

where $S = \{i : f_i \geq n^k\}$. Here, k is a parameter to be specified later. Note this requires communication $O(\frac{m}{n^k}\log n) = O(n^{1-k}\log n)$ since $m = \Theta(n)$ by assumption. Intuitively, H represents the contribution of the heavy hitters to the frequency-based function, and the verifier then "removes" these items from the stream by setting $f_i = 0$ for all $i \in S$. Assuming without loss of generality that $g(0) = 0$, this ensures that the removed items do not contribute to the sum $\sum_{i\in[n]} g(f_i)$. The verifier and helper then run the polynomial-agreement protocol on the *modified* frequency vector. We emphasize that the protocol just described consists of a *single* message from the helper to the verifier; the helper may send his message for the Heavy Hitter protocol, appended with his message for the polynomial-agreement protocol.

More precisely, we may view the new frequency vector $\mathbf{f}$ (with heavy hitters removed) as defining a function $\tilde{f} :\in \mathbb{F}_q[X,Y]$ that agrees with the modified frequency vector on the domain $[h]\times[v]$. Now define $\tilde{g}(W,Z)$ through interpolation as the polynomial with degree n^k in each variable that agrees with g on $[n^k]$. Finally, define

$$s(X) = \sum_{y=1}^{v} \tilde{g}\circ\tilde{f}(X,y)$$

Note s has degree at most hn^k. The prover's message to the verifier consists of a polynomial $s' : \mathbb{F}_q \to \mathbb{F}_q$, which also has degree at most hn^k and can therefore be specified with $O(hn^k \log n)$ bits.

The remainder of this proof is in line with the F_2 protocol. The helper claims $s' = s$; if s' is as claimed then verifier can compute $\mathfrak{f}(\mathbf{f}(A))$ via the identity $\mathfrak{f}(\mathbf{f}(A)) = \sum_{x\in[h]} s(x) + H$. So it suffices check $s = s'$.

To accomplish this, the verifier evaluates $s'(r)$ from the supplied annotation and computes $s(r) = \sum_{y\in[v]} \tilde{g}(\tilde{f}(r,y))$ 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 we can set the parameters h, v, and k of the above protocol to achieve hcost $=$ vcost $= O(n^{2/3}\log n)$. The communication cost is $O(n^{1-k}\log n)$ bits for the Heavy Hitters protocol plus $O(hn^k\log n)$ bits for sending the polynomial s'. The space cost is $O(n^{1-k}\log n)$ bits for the Heavy Hitters protocol and $O(v\log n)$ bits for the verifier's sketch. Setting $k = \frac{1}{3}$, $h = n^{\frac{1}{3}}$, and $v = n^{\frac{2}{3}}$ has the desired costs.

A subtlety is that we also have to account for the cost of storing g, since the verifier must be able evaluate $s(r) = \sum_{y\in[v]} \tilde{g}(\tilde{f}(r,y))$. In the protocol just described, we assume that g has a succinct implicit description; this is indeed the case for important examples such as F_0, F_∞, and point and range queries on the frequency distribution that are described subsequently.

In order to achieve an $(n^{3/4}\log n, n^{3/4}\log n)$ online protocol for $\mathfrak{f}$, observe that the only place where the protocol described above used prescience was to identify Heavy Hitters. In our online protocol, we simply substitute the online Heavy Hitter protocol of Theorem 4.1 with parameter $\alpha \in [0,1]$ in place of the prescient version. In this case, the communication cost of the protocol is $O(n^{1-k}n^{\alpha}\log n)$ bits for the Heavy Hitters protocol and $O(hn^k\log n)$ bits for sending the polynomial s'. The space cost is $O(n^{1-\alpha}\log n)$ bits for the Heavy Hitters protocol and $O(v\log n)$ bits for the verifier's sketch. Balancing these costs by setting $k = \frac{1}{2}$, $\alpha = \frac{1}{4}$ $h = n^{\frac{1}{4}}$, and $v = n^{\frac{3}{4}}$ gives the desired costs.

Finally, we describe how to achieve an online $(n^{2/3}\log n, n^{2/3}\log n)$ protocol if $\mathfrak{f}$ is based on low-frequencies. Suppose there is are constants C_1 and C_2 such that $g(x) = C_1$ for all $x \geq C_2$. Then we may obviate the need for a Heavy Hitters protocol entirely: while observing the stream, the verifier keeps a buffer of the $n^{2/3}$ most recent items observed, and "collapses down" any items appearing more than C_2 times in the buffer to an instance occuring exactly C_2 times. It is easy to see that $\mathfrak{f}$ is the same for the collapsed stream as for the original stream, since $\mathfrak{f}$ is based on low frequencies. And as a result of the collapsing, no item in the filtered stream has frequency higher than $O(n^{1/3})$. Therefore, a protocol similar to the polynomial-agreement protocol applied to the collapsed stream yields the desired bounds. $\square$

Theorem 4.7 provides protocols for the following problems:

- F_0, the number of items with non-zero count. This follows by observing that F_0 is equivalent to computing $\sum_{i \in [u]} g(f_i)$ for $g(0) = 0$ and $g(x) = 1$ for $x > 0$. Since F_0 is based on low frequencies, we achieve an $(n^{2/3} \log n, n^{2/3} \log n)$ online protocol.
- More generally, we can compute functions on the inverse distribution, i.e. queries of the form "how many items occur exactly k times in the stream" by setting, for any fixed j, $g(j) = 1$ and $g(i) = 0$ for $i \neq j$. One can build on this to compute, e.g. the number of items which occurred between j and j' times, the median of this distribution, etc. If j is a constant, as in the case of 'rarity' ($j = 1$) [17] we achieve an $(n^{2/3} \log n, n^{2/3} \log n)$ online protocol. Otherwise, we achieve an $(n^{2/3} \log n, n^{2/3} \log n)$ prescient protocol and an $(n^{3/4} \log n, n^{3/4} \log n)$ online protocol.
- We obtain a protocol for $F_\infty = \max_i f_i$, with a little more work. The helper first claims a lower bound lb on F_∞ by providing the index of an item with frequency F_∞, which the verifier checks by running the INDEX protocol from Theorem 3.1. Then the verifer runs the above protocol with $g(i) = 0$ for $i \leq lb$ and $g(i) = 1$ for $i > lb$; if $\sum_{i \in [u]} g(f_i) = 0$, then the verifier is convinced no item has frequency higher than lb, and concludes that $F_\infty = lb$. We therefore achieve an $(n^{2/3} \log n, n^{2/3} \log n)$ prescient protocol and an $(n^{3/4} \log n, n^{3/4} \log n)$ online protocol for F_∞ (or an $(n^{2/3} \log n, n^{2/3} \log n)$ online protocol in the case that F_∞ is at most a constant).

Handling Duplicates via Frequency-Based Functions. We mention that the protocol of Theorem 4.7 can be used to solve problem of verifying a Convex Hull on a 2D Grid, even in the presence of duplicate input points. The same approach allows duplicate edges to be handled in the subsequent protocols for graph computations.

Specifically, the convex hull protocol from Section 4.2 must check that the input points are contained within the claimed convex hull C'. It exploited the fact that for sets X, Y represented as indicator vectors x, y, it holds that $X \subseteq Y \iff F_2(y - x) = |Y| - |X|$, allowing the verifier to check that $C' \subseteq P$ and $P \subseteq c(C')$ using Theorem 4.3. More generally, for any vector y such that $y_i > 0$ if and only if $i \in Y$, it holds that $X \subseteq Y \iff y_i - x_i \geq 0$ for all i. Assume b is a known upper bound on $F_\infty(y)$, which corresponds to the maximum number of times any point in Y is duplicated. Let $\tilde{g}$ be defined through interpolation as the polynomial of degree $2b$ over the finite field $\mathbb{F}_p$ such that $\tilde{g}(x) = 0$ for $0 \leq x \leq b$, and $\tilde{g}(x) = 1$ for $-b-1 \leq x(\bmod p) < 0$. Then $\sum_i \tilde{g}(y_i - x_i) = 0$ if and only if $X \subseteq Y$. Applying the polynomial-agreement protocol from within the proof of Theorem 4.7 under this definition of $\tilde{g}$, we obtain a $(kh \log n, v \log n)$ protocol for checking $X \subseteq Y$ for any $hv \geq n^2$. This yields a a $((|C| + kg^\alpha) \log n, (|C| + g^{1-\alpha}) \log n)$ online protocol to report the convex hull, where g is the number of possible point locations.

4.5 Frequency-Based Functions for Skewed Streams

In practice, the frequency distributions of data streams are often skewed, in the sense that a small number of frequent items make up a large portion of the stream. We observe that, if the stream is sufficiently skewed, so that there are few heavy hitters, we can achieve a more efficient frequency-based function protocol. To see this, notice that in the protocol of Theorem 4.7, after the prover tells the verifier the heavy hitters, the verifier only needs to know an *approximate* upper bound on the F_∞ of the stream A' obtained from the original stream A by deleting all the heavy hitters. That is, the prover only needs to convince the verifier that he has presented "enough" of the true heavy hitters (and their exact frequencies) so that $F_\infty(A') \leq b$ for some upper bound $b = \Theta(n^k)$—then we may define $\tilde{g}$ to agree with g on $[b]$, so that the degree of $\tilde{g}$ remains $O(n^k)$.

Therefore, it suffices to force the prover to identify enough heavy hitters so that $F_\infty(A') \leq b$ for some upper bound $b = \Theta(n^k)$. Observe that if there are not many heavy items, this can be accomplished efficiently by having the prover send a list L of heavy hitters and their frequencies (proving the frequencies are truthful via L parallel INDEX queries) and then appending a proof of an approximate upper bound (within factor $1+\varepsilon$) as per Section 4.2 on the quantity F_∞ of A'.

It suffices to let ε be any positive constant in order to achieve $b = O(n^k)$. If there are fewer than l items with frequency greater than n^k, the INDEX queries, if they are online, require annotation $O(lh \log n)$ and space $O(v \log n)$, while the approximate F_∞ protocol requires annotation $O(h \log^3 n)$ and space $O(v \log^2 n)$. In what follows, we will choose l to be polynomial in n, so we will obtain a $(lh \log n, v \log^2 n)$ scheme for identifying the set of heavy hitters and an upper bound ub on $F_\infty(A')$.

For concreteness, we will analyze the costs of our improved protocol under the assumption that the frequencies of items in the stream follow a zipfian distribution, so that the ith largest frequency is (at most) mi^{-z} for parameter z. Setting this equal to n^k and rearranging, we obtain that there are at most $(m/n^k)^{1/z}$ heavy hitters to identify.

Therefore, if $m = \Theta(n)$, we can (with high probability) reduce the cost of the heavy hitters sub-protocol within the scheme of Theorem 4.7 to $(n^{(1-k)/z} h \operatorname{poly}\log n, v \operatorname{poly}\log n)$. Adding in the annotation cost of sending the polynomial $\tilde{g} \circ \tilde{f}$, and the space cost of storing the verifier's sketch, the entire protocol therefore requires $\tilde{O}(n^{(1-k)/z} h + hn^k)$ and space $\tilde{O}(v)$. Balancing exponents by setting $k = \frac{1}{z+1}$, $h = n^{\frac{1}{2} + \frac{1}{2(z+1)}}$, and $v = n/h$, we obtain an $(n^{\frac{1}{2} + \frac{1}{2(z+1)}} \operatorname{poly}\log n, n^{\frac{1}{2} + \frac{1}{2(z+1)}} \operatorname{poly}\log n)$ protocol.

For example, if $z = 2$, we obtain an $(n^{2/3} \operatorname{poly}\log n, n^{2/3} \operatorname{poly}\log n)$ online protocol, which essentially matches the cost of our online protocol for functions based on low-frequencies, but applies to any frequency-based function. If $z = 3$, we obtain an $(n^{5/8} \operatorname{poly}\log n, n^{5/8} \operatorname{poly}\log n)$ online protocol.

Finally, we present a more efficient prescient protocol. If we use prescient INDEX protocols rather than online ones, our heavy hitters protocol only requires annotation $\tilde{O}(l + h_1)$ and space $\tilde{O}(l + v_1)$, provided $h_1 v_1 \geq n$. Hence, the entire protocol has communication cost $\tilde{O}(n^{(1-k)/s} + h_1 + h_2 n^k)$ and space cost $\tilde{O}(n^{(1-k)/s} + v_1 + v_2)$, where $h_1 v_1 = h_2 v_2 = n$. Assume $1 < s \leq 2$. Then setting $k = \frac{2-s}{2+s}$, $h_1 = v_1 = n^{1/2}$, $h_2 = n^{z/(2+z)}$, and $v_2 = n^{2/(2+z)}$, we obtain an $(n^{2/(2+z)} \operatorname{poly}\log n, n^{2/(2+z)} \operatorname{poly}\log n)$ protocol. For example, if $z = 2$, we obtain a $(n^{1/2} \operatorname{poly}\log n, n^{1/2} \operatorname{poly}\log n)$ prescient protocol. For $z > 2$, protocols with the same cost follow by setting $k = 0$, $h_1 = h_2 = v_1 = v_2 = n^{1/2}$.

5 Graph Problems

In this section we consider computing properties of graphs on n nodes, determined by a stream of m edges [20, 26]. We present tight results for testing connectivity of sparse graphs, determining bipartiteness, determining if a bipartite graph has a perfect matching, and counting triangles. Our bipartite perfect matching result achieves optimal tradeoffs up to logarithmic factors.

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, 10, 28]. 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 4.5, shows that this is best possible.

Theorem 5.1. *Any (h,v) scheme for counting triangles must have $hv = \Omega(n^2)$.*

Proof. We show a reduction from an instance $\textsc{disj}_{(n^2/9),2}$. We represent the instance as $n/3 \times n/3$ matrices X, Y in the natural way. We proceed to construct a graph that has a triangle iff $X_{i,j} = Y_{i,j} = 1$ for some $i, j \in [n/3]$. The nodes are partitioned into sets U, V, W so that $|U| = |V| = |W| = n/3$. Insert edges $\{(u_i, w_i) : i \in [n/3]\} \cup \{(u_i, v_j) : X_{i,j} = 1\} \cup \{(w_i, v_j) : Y_{i,j} = 1\}$. There is a triangle (u_i, v_j, w_i) iff $X_{i,j} = Y_{i,j} = 1$, and there is no other way to form a triangle. The result follows from Theorem 4.5. □

We now outline an online scheme with vcost $= O(\log n)$ and 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 [22] 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 5.2. *There exists a $(n^2, \log n)$ online scheme for matrix multiplication.*

Proof. By the result of Kimbrel and Sinha [29], the verifier can check $AB = C$ by picking r uniformly from $\mathbb{F}_q$ and checking that $A(B\mathbf{r}^\top) = C\mathbf{r}^\top$ for vector $\mathbf{r} = (r^1, \ldots, r^n)$. This fails to distinguish different matrices with probability at most n/q. Rather than computing $A(B\mathbf{r}^\top)$ and $C\mathbf{r}^\top$ explicitly, the verifier will compare fingerprints of $C\mathbf{r}^\top$ and $AB\mathbf{r}^\top$. These are computed as $\mathbf{s}C\mathbf{r}^\top$ and $\mathbf{s}AB\mathbf{r}^\top$, for a vector $\mathbf{s} = (s^1, \ldots, s^n)$ where s is picked uniformly from $\mathbb{F}_q$. This also fails with probability at most n/q. We observe that (1) $\mathbf{s}C\mathbf{r}^\top = \sum_{i,j} s^i r^j C_{i,j}$ can be computed easily whatever order the entries of C are presented in. (2) $\mathbf{s}AB\mathbf{r}^\top = (\mathbf{s}A)(B\mathbf{r}^\top)$ is the inner product of two n-dimensional vectors, and that $(\mathbf{s}A)_i = \sum_j s^j A_{i,j}$ and $(B\mathbf{r}^\top)_i = \sum_j r^j B_{j,i}$. Therefore, if the helper presents the ith column of A followed by the ith row of B for each i in turn, the verifier can easily compute $\mathbf{s}AB\mathbf{r}^\top$, in $O(\log q)$ space. Picking $q \geq 6n$ ensures that the verifier is fooled with probability at most $1/3$, and the total space used by the verifier to store r, s and intermediate values is $O(\log n)$. □

With this primitive, arbitrary matrix products $(A_\ell A_{\ell-1} \ldots A_2 A_1)$ 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^ℓ are verified with $O(n^2 \log \ell)$ annotation.

Theorem 5.3. *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 5.2 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 $\{(u, v, w) : w \neq u, v\}$. 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 4.3, we obtain the following theorem.

Theorem 5.4. *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 two online schemes for testing whether a bipartite graph has a perfect matching. Our first scheme is efficient for sparse graphs, while our second achieves *optimal* tradeoffs between hcost and vcost for dense graphs, up to logarithmic factors. Graph matchings have been considered in the stream model [20, 40] 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 $O(n^{1+\alpha}\log n)$ annotation, the answer can be verified in $O(n^{1-\alpha}\log n)$ space. This is shown to be best possible by combining a reduction from [20] coupled with Theorem 3.1.

Theorem 5.5. *There exists an* $(m, \log n)$ *online scheme for bipartite perfect matching, as well as an* $(n^{1+\alpha}\log n, n^{1-\alpha}\log n)$ *online scheme for any* $0 \leq \alpha \leq 1$. *Any* (h, v) *online scheme for bipartite perfect matching requires* $hv = \Omega(n^2)$.

Proof. We begin by presenting the $(m, \log n)$ scheme. We consider the general case, where there may be nodes in $[n]$ with no incident edges, which are to be ignored for the matching. If there is a perfect matching M, the annotation lists all edges in M, and the degree of all nodes in $[n]$. Let x be the characteristic vector that has 1 in the vth coordinate iff the degree of v is non-zero, y be the vector of node frequencies in M. The verifier can use fingerprints to ensure that the claimed degree sequence is correct, and that x matches y.

If the graph does not have a perfect matching, Hall's theorem provides a witness. Let $L \cup R$ be a bipartition of the graph, then there exists $L' \subset L$ such that $|L'| > |\Gamma(L')|$, the neighbors of L'. The helper lists for each node: its degree; whether it is in L or R; and whether it is in L', $\Gamma(L')$, or neither. Then the helper presents each edge (u, v), along with the same information on each node. Using Lemma 2.2, the verifier can ensure that the sets are consistent with a constant number of fingerprints. It remains to check that each edge is allowable and $|L'| > |\Gamma(L')|$.

Our $(n^{1+\alpha}\log n, n^{1-\alpha}\log n)$ scheme follows the same conceptual outline as the above: if G has a perfect matching, the helper provides the matching, while if G has no perfect matching, the helper demonstrates this via Hall's theorem.

If there is a perfect matching M, the annotation lists all edges in M, followed by a proof that $M \subseteq E$. More specifically, for any $hv \geq n^2$, the discussion of finding a convex hull in Section 4.2 describes how to obtain a $v\log n$-space protocol using annotation $h\log n$ for showing $M \subseteq E$, assuming no duplicate edges. This can be extended to a $(kh\log n, v\log n)$ protocol if edges may be duplicated up to k times. The helper uses this protocol to demonstrate $M \subseteq E$, and the verifier checks that M is a matching by comparing a fingerprint of M to one of the set $\{1, 2, \ldots, n\}$.

If the graph does not have a perfect matching, let $L \cup R$ be a bipartition of the graph, as before, and let $L' \subset L$ such that $|L'| > |\Gamma(L')|$. We will use the $(n^{1+\alpha}\log n, n^{1-\alpha}\log n)$ online protocol for integer $n \times n$ matrix-vector multiplication described in subsequent work [15, Theorem 4]. The verifier must check that (1) L is a bipartition of n; (2) $L' \subseteq L$; and (3) $|L'| > |\Gamma(L')|$. Let $x \in \{0,1\}^n$ be the indicator vector of L, and let A be the adjacency matrix of G, i.e. $A_{ij} = 1$ if there is an edge between i and j in G and $A_{ij} = 0$ otherwise. (1) is equivalent to the condition $x^T A x = 0$, which can be checked using integer matrix-vector multiplication to verify Ax, followed by an inner-product protocol to verify $x^T A x$. (2) can be checked trivially while the helper specifies L by requiring the nodes of L' to be marked. To check (3), notice that $|\Gamma(L')|$ is equal to the number of non-zero entries in the vector Ax. This can be computed while the verifier checks (1), and that $|\Gamma(L')| < |L'|$.

The result is a $(kn^{1+\alpha}\log n, n^{\alpha}\log n)$ online protocol for $0 \le \alpha \le 1$. We remark that the above two protocols also provide $(m,1)$ and $(n^{1+\alpha}\log n, n^{1-\alpha}\log n)$ schemes for proving bipartiteness, respectively; proving non-bipartitness can by accomplished simply by providing an odd cycle C (the verifier checks C is odd, C is a cycle, and $C \subseteq E$). $\square$

Connectivity. The problem of determining if a graph is connected was considered in the standard stream model [20, 26] and the multi-pass W-stream model [19]. 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 [20] with Theorem 3.1. We also achieve tradeoffs between hcost and vcost for dense graphs, obtaining an $(n^{1+\alpha}\log n, n^{1-\alpha}\log n)$ online protocol.

Theorem 5.6. *There exists an $(m,\log n)$ online scheme for connectivity, as well as an $(n^{1+\alpha}\log n, n^{1-\alpha}\log n)$ online scheme for any $0 \le \alpha \le 1$. Any (h,v) scheme (online or prescient) for connectivity requires $hv = \Omega(n)$ even when $m = O(n)$.*

Proof of Theorem 5.6. We begin with the $(m,\log n)$ protocol. If the graph is connected then there exists a spanning tree T directed towards the root and an injective labeling of the nodes $f : V \to [|V|]$ such that each non-root node with label j is linked to exactly one node with label $> j$. The helper outputs such a function f, and the verifier ensures that it is an injection. Then each (directed) edge (u,v) in T and its labels $f(u) > f(v)$ is presented in decreasing order of $f(u)$. The verifier checks this order, and ensures that it is consistent with f via fingerprinting. The helper must also list all edges, so that the verifier can ensure that all T edges are from the input.

If the graph is not connected the annotation presents a bipartition of the graph. Each node is presented along with its label, and each is presented along with the corresponding node labels. The verifier uses fingerprinting to ensure no edge is omitted, and the multiset protocol of Lemma 2.2 to ensure that the node labels are consistent.

The $(n^{1+\alpha}\log n, n^{1-\alpha}\log n)$ scheme follows the same conceptual outline as above: if G is connected, the helper demonstrates this by providing a spanning tree; if G is disconnected, the helper identifies a bipartition of the graph. In the first case, the helper provides a set of edges T claimed to be a spanning tree, and the verifier must check (1) T is spanning and (2) $T \subseteq E$. (1) is accomplished as in the $(m,1)$ case, by appropriate labelling of the $O(n)$ edges, with $O(n)$ annotation. The description of the convex hull protocol of Section 4.2 explains how to check (2) with space $n^{1-\alpha}\log n$ and annotation $n^{1+\alpha}\log n$.

If G is disconnected, the helper presents a set $L \subset V$, $L \neq V$, and claims that L is disconnected from $V \setminus L$. Let A be the adjacency matrix of G, and let $x \in \{0,1\}^n$ be the indicator function of L. To check that L is as claimed, it suffices for the verifier to compute Ax, and check that the each non-zero entry of Ax corresponds to vertices in L (intuitively, this means the set L' of vertices at distance one from L is contained in L). The first step uses the integer matrix-vector multiplication protocol of [15, Theorem 4]. This allows the verifier to ensure that the set $\{i : (Ax)_i \neq 0\}$ matches L, via fingerprints.

For the lower bound, we reduce an instance of $\text{DISJ}_{n,2}$ to connectivity of a graph with $O(n)$ edges over nodes $v_{0,0} \ldots v_{3,n}$: create edges $(v_{j,0}, v_{j,i})$ for $j \in \{0,2,3\}$ and $i \in [n]$. Then if $x_i = 1$, add edge $(v_{0,i}, v_{1,i})$, else add edge $(v_{1,i}, v_{2,i})$; and if $y_i = 1$, add edge $(v_{1,i}, v_{3,i})$ else add edge $(v_{2,i}, v_{3,i})$. The resulting graph is connected only if x and y are not disjoint. The result follows from Theorem 4.5. $\square$

Here, we have considered undirected graphs. The technique extends to checking (strong) connectivity of directed graphs, but we omit the details for brevity.

Subsequent Work. Subsequent work has further studied the protocols for graph computations in this model [15]. In particular, it is observed that given any deterministic RAM algorithm with running time R, there exists a $(m+R, \log n)$ protocol which simulates the algorithm in the annotation model. This implies alternate proofs for the existence of $(m,1)$ protocols for bipartite perfect matchings, bipartiteness, and connectivity.

Acknowledgements:

We thank Yael Gertner, Sampath Kannan, and Mahesh Viswanathan for sharing [23]. 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 Symposium on Theory of Computing*, pages 731–740, 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 Symposium on Foundations of Computer Science*, pages 540–549, 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 Symposium on Foundations of Computer Science*, pages 337–347, 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 Symposium on Discrete Algorithms*, pages 623–632, 2002.

[7] P. Beame and D.-T. Huynh-Ngoc. On the value of multiple read/write streams for approximating frequency moments. In *IEEE Symposium on Foundations of Computer Science*, 2008.

[8] P. Beame, T. S. Jayram, and A. Rudra. Lower bounds for randomized read/write stream algorithms. In *ACM Symposium on Theory of Computing*, pages 689–698, 2007.

[9] V. Braverman and R. Ostrovsky. Zero-one frequency laws. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 281–290, New York, NY, USA, 2010. ACM.

[10] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *ACM Principles of Database Systems*, pages 253–262, 2006.

[11] A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *ACM Symposium on Theory of Computing*, pages 641–650, 2008.

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

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

[14] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in data streams. In *International Conference on Very Large Data Bases*, pages 464–475, 2003.

[15] G. Cormode, M. Mitzenmacher, and J. Thaler. Streaming graph computations with a helpful advisor. In *European Symposium on Algorithms*, 2010.

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

[17] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Computer Science*, pages 323–334, 2002.

[18] C. Demetrescu, B. Escoffier, G. Moruz, and A. Ribichini. Adapting parallel algorithms to the w-stream model, with applications to graph problems. In *MFCS*, pages 194–205, 2007.

[19] C. Demetrescu, I. Finocchi, and A. Ribichini. Trading off space for passes in graph streaming problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 714–723, 2006.

[20] 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.

[21] J. Feigenbaum, S. Kannan, and J. Zhang. Annotation and computational geometry in the streaming model. Technical Report YALEU/DCS/TR-1249, Yale University, 2003.

[22] R. Freivalds. Fast probabilistic algorithms. In *MFCS*, pages 57–69, 1979.

[23] Y. Gertner, S. Kannan, and M. Viswanathan. NP and streaming verifiers. *Manuscript*, 2002.

[24] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: Interactive proofs for muggles. In *ACM Symposium on Theory of Computing*, pages 113–122, 2008.

[25] M. Grohe, A. Hernich, and N. Schweikardt. Randomized computations on large data sets: tight lower bounds. In *ACM Principles of Database Systems*, pages 243–252, 2006.

[26] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.

[27] W. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[28] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *International Conference on Computing and Combinatorics*, pages 710–716, 2005.

[29] 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.

[30] H. Klauck. Rectangle size bounds and threshold covers in communication complexity. In *IEEE Conference on Computational Complexity*, pages 118–134, 2003.

[31] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[32] F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios. Proof-infused streams: Enabling authentication of sliding window queries on streams. In *VLDB*, pages 147–158, 2007.

[33] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[34] S. Papadopoulos, Y. Yang, and D. Papadias. Cads: Continuous authentication on data streams. In *VLDB*, pages 135–146, 2007.

[35] A. Razborov. On the distributional complexity of disjontness. In *International Colloquium on Automata, Languages and Programming*, pages 249–253, 1990.

[36] A. Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[37] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[38] P. A. Tucker, D. Maier, L. M. L. Delcambre, T. Sheard, J. Widom, and M. P. Jones. Punctuated data streams, 2005.

[39] K. Yi, F. Li, M. Hadjieleftheriou, G. Kollios, and D. Srivastava. Randomized synopses for query assurance on data streams. In *IEEE International Conference on Data Engineering*, 2008.

[40] M. Zelke. Weighted matching in the semi-streaming model. In *STACS*, pages 669–680, 2008.

[41] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley (Reading MA), 1949.