Space Complexity of Hierarchical Heavy Hitters in Multi-Dimensional Data Streams*

John Hershberger
Mentor Graphics Corp.
8005 SW Boeckman Road
Wilsonville, OR 97070
john_hershberger@mentor.com

Nisheeth Shrivastava & Subhash Suri
Computer Science Department
University of California at Santa Barbara
Santa Barbara, CA 93106
{nisheeth, suri}@cs.ucsb.edu

Csaba D. Tóth
Dept. of Mathemetics
MIT
Cambridge, MA 02139
toth@math.mit.edu

ABSTRACT

Heavy hitters, which are items occurring with frequency above a given threshold, are an important aggregation and summary tool when processing data streams or data warehouses. Hierarchical heavy hitters (HHHs) have been introduced as a natural generalization for hierarchical data domains, including multi-dimensional data. An item x in a hierarchy is called a *ϕ-HHH* if its frequency *after discounting the frequencies of all its descendant hierarchical heavy hitters* exceeds ϕn, where ϕ is a user-specified parameter and n is the size of the data set. Recently, single-pass schemes have been proposed for computing *ϕ-HHHs* using space roughly $O(\frac{1}{\phi}\log(\phi n))$. The frequency estimates of these algorithms, however, hold only for the *total frequencies* of items, and not the discounted frequencies; this leads to *false positives* because the discounted frequency can be significantly smaller than the total frequency. This paper attempts to explain the difficulty of finding hierarchical heavy hitters with better accuracy. We show that a single-pass deterministic scheme that computes *ϕ-HHHs* in a d-dimensional hierarchy with any approximation guarantee must use $\Omega(1/\phi^{d+1})$ space. This bound is tight: in fact, we present a data stream algorithm that can report the *ϕ-HHHs* without false positives in $O(1/\phi^{d+1})$ space.

1 Introduction

Finding frequent items, or *heavy hitters*, in a single pass over a data set is a classical problem that has received considerable renewed interest recently due to its applications in "online analytical processing" (OLAP) and real-time monitoring systems. An unprecedented amount of data is constantly being generated and collected: retail transactions, measurements by scientific instruments, IP packet logs, telephone call records, etc. At the same time, there is a growing need to make decisions and infer interesting patterns in real time. An important step in this direction is to compute, in a single pass, aggregates and summaries that capture interesting features of the data. These summaries are used both to detect unusual events and to direct subsequent more refined queries [3, 8, 13].

Heavy hitters, also called "icebergs" queries, are one of the most basic summaries—they report items with frequency above a specified threshold. Heavy hitters have obvious applications in IP networking (nodes responsible for a large fraction of the traffic), databases (high-frequency attribute values), fraud detection, etc. For a given threshold ϕ, where $0 < \phi < 1$, a simple counter-based algorithm of Misra and Gries [12] returns a list of $O(1/\phi)$ items including every item whose frequency is above ϕn in a single pass, using $O(1/\phi)$ space. Essentially the same algorithm has recently been reinvented by Karp, Shenker, and Papadimitriou [10] and Demaine, López-Ortiz, and Munro [7]. This algorithm, however, may return false positives, and requires a second pass over the data to determine the exact set of heavy hitters. Nevertheless, it maintains approximate frequencies that are no greater than the true frequencies and at most ϕn smaller. An algorithm of Manku and Motwani [11] provides an ε-approximation guarantee: For any $\varepsilon \in (0, 1]$, it reports all items with frequency above ϕn but no item with frequency below $(1-\varepsilon)\phi n$, using $O(\frac{1}{\varepsilon\phi}\log\varepsilon\phi n)$ space. In fact, this approximation guarantee can also be achieved by the Misra–Gries algorithm using only $O(\frac{1}{\varepsilon\phi})$ space (run the algorithm with parameter $\varepsilon\phi$, and then report all items whose approximate frequency is above $(1-\varepsilon)\phi n$ [2]). Finally, Charikar, Chen, and Farach-Colton [4] and Estan and Varghese [9] present hash- and sample-based randomized schemes for estimating the frequencies of all items. Thus, a variety of schemes using sampling, counting, or hashing are known for finding heavy hitters or estimating their frequencies with a guaranteed error bound.

In many applications of data streams, the data is hierarchically organized, and a *flat* summary like heavy hitters does not adequately reflect the structure inherent in the data. For instance, database transactions can be viewed at different levels of detail: grouped by time (hours, days, weeks, months), or grouped by geography (store, city, state, country); IP addresses have a natural 32 bit hierarchy; port

*The research of the last three authors was partially supported by National Science Foundation grants CCR-0049093 and IIS-0121562. Research by the fourth author was done at UC Santa Barbara.

numbers may be grouped by service type, and so on. Consequently, it is often useful to look for heavy hitters not just at the *highest* level of detail, or even at the *same* level of detail, but rather at different levels of detail. For example, a single web server in a computer science department may be a heavy hitter for network traffic. It would be interesting to know whether the computer science department as a whole is a heavy hitter only because of the web server, or *even* without it. Similarly, no single machine in a research group may be a heavy hitter, but together they may form a heavy hitter entity (at a higher level in the network prefix hierarchy).

Many application data are also multi-dimensional. For instance, database transactions typically have several independent significant attributes: time, location, volume, etc. IP packets typically have five or more fields of interest for monitoring applications: source, destination, protocol, and two port numbers. The interesting patterns among such data often emerge only by considering all or multiple dimensions simultaneously.

Hierarchical heavy hitters (HHH) were introduced as a natural generalization of heavy hitters to hierarchical and multi-dimensional data domains by Estan and Varghese [9] and Cormode et al. [5, 6]. Following Cormode et al. [6], we imagine the d-dimensional data organized as a lattice, where the input tuples form the lowest level of the lattice. Each internal node of the lattice represents the *aggregation* of all its descendant leaves. Thus, the frequency of an internal node x is the cumulative frequency of all the descendant leaves of x. *Discounted frequencies* and *ϕ-hierarchical heavy hitters* (*ϕ-HHHs*) are defined recursively: the discounted frequency of x counts only the descendant leaves not included in any *ϕ-HHH* that is a strict descendant of x; an item x is a *ϕ-HHH* if its discounted frequency exceeds ϕn.

Hierarchical heavy hitters are natural and powerful constructs, but they involve subtle difficulties. First, an item's discounted frequency potentially depends on many descendants: the contributions of many descendants may need to be subtracted if they become heavy, but not otherwise. This is further complicated by the fact that a stream-processing algorithm can maintain only approximate frequencies. Thus, it seems difficult to control the error in estimating the discounted frequency without storing too much data. Second, in multi-dimensional data, the descendant items (aggregates) can overlap. The best way to see this is through geometry: Each lattice node corresponds to a rectangular box, and a parent node can have multiple descendants intersecting in complex ways. In computing the discounted frequency of x, we cannot simply subtract the frequencies of all heavy descendants (even if we knew them exactly). Instead we need to know the *non-overlapping* count of the descendant items, which again seems difficult to determine without storing large amounts of data.

All current methods for computing the *ϕ-HHHs* in one pass fail to bound the *discounted frequency* of the items. Although Cormode et al. [5, 6] and Estan, Savage, and Varghese [8] define hierarchical heavy hitters using the discounted frequency, their algorithms output items that are guaranteed only to have *total frequency* above the threshold. Using total frequency instead of discounted frequency leads to *false positives* because the former can be significantly larger than the latter. Indeed, Cormode et al. [6] state that "we do not know of ways to accurately approximate the discounted count," and resort to a weaker definition of hierarchical heavy hitters in order to get space-efficient algorithms.

Our contribution

In this paper, we formalize the difficulty of computing true *ϕ-HHHs* and prove lower bounds on the space complexity of algorithms that compute them. We focus primarily on *deterministic* algorithms, although one of our bounds also applies to randomized algorithms. We prove lower bounds under two different models. For streams of 1-dimensional data, we give an $\Omega(1/\phi^2)$ space lower bound for *any* (deterministic or randomized) algorithm, using an information-theoretic argument.

To prove lower bounds for streams of multi-dimensional data and to establish stronger space bounds, we limit our discussion to a simple model of deterministic algorithms, which we call the *node frequency* model. In this model, an algorithm with space bound s is allowed s distinct counters, and each counter maintains the frequency of a node in the hierarchy. The node frequency model captures all the counter-based algorithms, including [7, 10, 11, 12], as well as algorithms for hierarchical heavy hitters that build on those algorithms [5, 8]. While the multi-dimensional HHH scheme of Cormode et al. [6] does not strictly fit in this model, we will show that our lower bound construction in fact applies to it as well.

We show that any single-pass deterministic scheme that computes *ϕ-HHHs* for d-dimensional data in the node frequency model with any bounded approximation guarantee must use $\Omega(1/\phi^{d+1})$ space. Our basic lower bound construction shows that if we want all items with discounted frequency above ϕn to be reported as *ϕ-HHHs*, and no item with discounted frequency below $(1 - \frac{1}{2^d})\phi n$ to be reported, then the space lower bound holds. We then generalize this basic construction to any fixed degree of accuracy. In particular, for any fixed $\varepsilon \in [0, 1)$, separating true hierarchical heavy hitters from those with discounted frequency at most $(1 - \varepsilon)\phi n$ requires $\Omega(1/\phi^{d+1})$ memory.

Finally, we show that our lower bounds are asymptotically tight: we exhibit a deterministic data stream algorithm (in the node frequency model) that can compute *ϕ-HHHs* with constant approximation error, using $O(1/\phi^{d+1})$ memory. The constant in the asymptotic notation is quite large, and the algorithm is not intended to be practical; rather it is meant to show that our lower bound is asymptotically tight.

Thus, our main result provides a sharp contrast between the complexity of non-hierarchical and hierarchical heavy hitters. Non-hierarchical heavy hitters can be determined with absolute error at most εn, for any $\varepsilon < \phi$, using space $O(1/\varepsilon)$. But computing *ϕ-HHH*s requires at least $\Omega(1/\phi^{d+1})$ space in the worst case, even for a fixed accuracy level.

2 Hierarchical Domains and Heavy Hitters

The input to our heavy hitter problem is a stream S of tuples $\langle e_1, e_2, \ldots, e_d \rangle$ from a d-dimensional hierarchical domain D. To build intuition, we first consider the case $d = 1$. A one-dimensional hierarchical domain is naturally modeled by a tree. The *elements* of the domain are represented by the leaves of a rooted, balanced b-ary tree T. We say that b is the *branching factor* of the hierarchy D. Each non-leaf node of T represents the set of all its descendant leaves, and the nodes at the same level of T form a partition of the domain. For two nodes v and w of the tree, we say that $v \preceq w$ if v is a strict descendant of w or $v = w$. If $v \neq w$, then $v \prec w$.

The IP address hierarchy provides an illustrative example of these concepts. The elements of this domain are the 32-bit addresses of individual machines. The root, denoted $*$, represents the entire address space, and intermediate nodes represent network prefixes, denoting hierarchical subnets. Thus, all descendants of a node v share a common address prefix, associated with v.

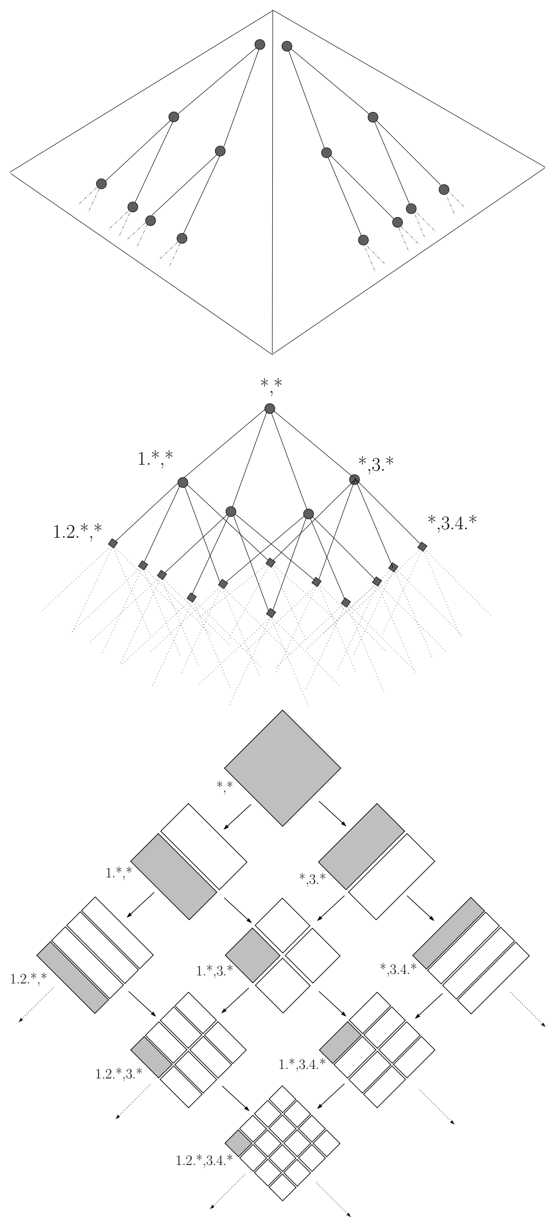


Figure 1. The top figure shows two 1-D hierarchies; the middle figure shows the 2-D hierarchy generated by their cross product; the bottom figure shows the 2-D hierarchy as an arrangement of boxes. The shaded boxes form a lattice.

A d-dimensional hierarchical domain $D = \langle D_1, D_2, ..., D_d\rangle$ is a cross product of d one-dimensional hierarchical domains. The cross product $T = \langle T_1, T_2, \ldots, T_d\rangle$ of the corresponding trees is the d-dimensional hierarchy. Each node of the hierarchy is a d-tuple $\langle v_1, \ldots, v_d\rangle$, where v_i is a node of the tree T_i. We say that $\langle v_1, \ldots, v_d\rangle \preceq \langle w_1, \ldots, w_d\rangle$ if $v_i \preceq w_i$ in all dimensions $i = 1, \ldots, d$. The elements of the domain D are the leaves of this hierarchy, that is, the nodes at *level zero*. *Level* ℓ of the hierarchy consists of all the nodes at distance ℓ from level 0. There are altogether $L = 1 + \sum_i h_i$ distinct levels in the hierarchy, where h_i is the height of tree T_i. If $d \geq 2$, then nodes at the same level may correspond to intersecting subsets of the domain. For example, the pairs of IP addresses $\langle \mathrm{src}, \mathrm{dst}\rangle$ form a 2D hierarchy, and the nodes $\langle 1.*, *\rangle$ and $\langle *, 3.*\rangle$ in Figure 1 are at the same level, with a non-empty intersection $\langle 1.*, 3.*\rangle$.

In iceberg query computations, the hierarchy is often modeled as a lattice. The nodes of a d-dimensional hierarchy containing a specific element of the domain form a *lattice* with respect to the relation $\preceq$. All lattices for distinct elements of a domain are isomorphic, and the lattice structure characterizes the multidimensional hierarchy. A vertex of the lattice corresponds to all nodes whose level is the same d-tuple $[\ell_1, \ell_2, \ldots, \ell_d]$ in the d trees, and so these nodes form a partition of the domain. Two quantities of this lattice will be relevant in our analysis: the *size* of the lattice, which is $H = \Pi_{i=1}^{d}(1 + h_i)$, and the size of the *largest anti-chain*, which is at most $a = \frac{H}{\max_j(1+h_j)}$.

2.1 Hierarchical heavy hitters

The elements of the data stream correspond to the leaf nodes of the hierarchy. A leaf element x is called a *ϕ-heavy hitter*, for a parameter $0 < \phi \leq 1$, if the frequency of x in the stream exceeds ϕn, where n is the size of the stream seen so far. In hierarchical domains, we also wish to identify groups of elements that form a heavy hitter, even though no single element in the group is heavy. The natural groups correspond to the nodes in the hierarchy. The frequency of a non-leaf node v is defined to be the sum of the frequencies of all the descendant leaves of v.

Simply reporting all the nodes with frequency above ϕn, however, is both verbose and imprecise. It is verbose because every ancestor of a heavy node is automatically classified as heavy, and so the number of reported heavy hitters can blow up by a factor of H, the lattice size. It is imprecise because each heavy hitter carries very little new information: we cannot tell whether it is heavy only because of a descendant, or because it is a new heavy group.

Motivated by these considerations, Estan, Savage, and Varghese [8] and Cormode et al. [5] introduced the following natural notion of hierarchical heavy hitters: a non-leaf node is defined to be a *ϕ-hierarchical heavy hitter*, denoted *ϕ-HHH*, if its frequency, after discounting the frequencies of all its descendant hierarchical heavy hitters, exceeds ϕn. Formally, the *discounted frequency* of a node v with respect to a subset X of the hierarchy is defined as:

$$g_X(v) = \\ |\{e \in S : e \in v \text{ but } e \notin w \text{ for any } w \prec v, w \in X\}|.$$

Definition 1. A set of nodes X is called *ϕ-hierarchical heavy hitters* (*ϕ-HHHs*) if, for every node v of the hierarchy, we have $v \in X$ if and only if $g_X(v) \geq \phi n$.

In the on-line setting of data streams, with limited storage, we must allow a certain degree of approximation in computing heavy hitters; otherwise, the algorithm can be forced to use memory proportional to the size of the domain [1]. We use the following definition for ε-approximate hierarchical heavy hitters.

Definition 2. A set of nodes X is called *ε-approximate HHHs*, where $0 < \varepsilon < 1$, if for every node v of the hierarchy, we have

- $g_X(v) \geq \phi n$ implies $v \in X$, and
- $v \in X$ implies $g_X(v) \geq (1 - \varepsilon)\phi n$.

That is, every node with discounted frequency at least ϕn is reported, and no reported node has discounted frequency less than $(1 - \varepsilon)\phi n$.[1] The space-efficient schemes of [5, 6, 8] satisfy only the first condition in this definition, and not the second one. Thus, the items reported as *ϕ-HHHs* by those algorithms are not guaranteed to have discounted frequency above any approximation threshold.

2.2 A Geometric Framework

Our lower bound is best described in a geometric framework. We think of the d-tuples as "points" in a d-dimensional space, and the nodes of the hierarchy as "rectangular boxes." Consider the 1-dimensional hierarchy (tree). The leaves represent the points; each non-leaf node represents a contiguous interval; and the root represents the interval containing all the points. The intervals at any level of the hierarchy are pairwise disjoint and they form a partition of the domain. A node's interval is properly contained in the interval of each of its ancestors. Thus, the geometric representation of the 1-dimensional lattice is a system of nested intervals.

In d dimensions, the geometric representation of our hierarchy is the cross product of d such 1-dimensional structures. The leaves of the hierarchy are points as before. But each non-leaf node is a d-dimensional rectangular box, which is the cross product of d 1-dimensional intervals. The box for a node is properly contained in the boxes of its ancestors. However, unlike in one dimension, the boxes at the same level of the hierarchy are *not disjoint*.

In this framework, our *ϕ-HHH* problem can be formulated as follows: Given a stream of d-dimensional points, and a system of rectangular boxes defined by the hierarchical structure of the domain, let the frequency of a box be the number of input points contained in the box. A box is called ϕ-heavy if its frequency exceeds ϕn, where n is the number of points in the stream. The *discounted frequency* of a box x (relative to a set of boxes X) is the number of points that lie in x but *not* in any strict descendant box of x that belongs to X. A set of boxes X forms *ε-approximate ϕ-HHHs* if every box with discounted frequency above ϕn is in X, and the discounted frequency of every box in X is above $(1 - \varepsilon)\phi n$.

2.3 The Model for Lower Bounds

We prove lower bounds under two different models. In the case of 1-dimensional streams, we are able to give an $\Omega(1/\phi^2)$ space lower bound for *any* (deterministic or randomized) algorithm. This lower bound, which is largely information theoretic, shows that computing 1-dimensional *ϕ-HHHs* with any guaranteed level of accuracy requires at least $\Omega(1/\phi^2)$ space.

In order to extend these lower bounds to multi-dimensional streams and to show stronger space bounds, we limit our discussion to a simple (deterministic) model of algorithms, which we call the *node frequency* model. In this model, an algorithm with space bound s is allowed s distinct counters, which it can use to maintain the frequencies of points inside any boxes (nodes in the hierarchy). The algorithm can reallocate its counters arbitrarily during the run. (When the algorithm deletes a counter, all memory of that box's frequency is lost. When the algorithm starts a new counter for box b mid-stream, then only the future points of the stream falling inside b can be tracked.)

The node frequency model captures all the counter-based algorithms, including [7, 10, 11, 12], as well as algorithms for hierarchical heavy hitters that use those algorithms [5, 8]. The multi-dimensional hierarchical heavy hitter scheme of Cormode et al. [6] does not strictly fit in our model, because it also maintains an *offset counter* for each box it tracks. This offset counter, however, is the frequency of another node (a grandchild), and we will show that this algorithm is also subject to our lower bounds.

3 A Lower Bound for One-Dimensional Hierarchies

In this section, we show that *any* algorithm that computes ε-approximate *ϕ-HHHs* in 1-dimensional streams requires $\Omega(1/\phi^2)$ space in the worst case. Our argument is based on information theory, and therefore applies to any (deterministic or randomized) algorithm. It bounds the *bit complexity* of the necessary space, whereas all our later upper and lower bounds hold in the RAM model, where each counter is stored in $O(1)$ space.

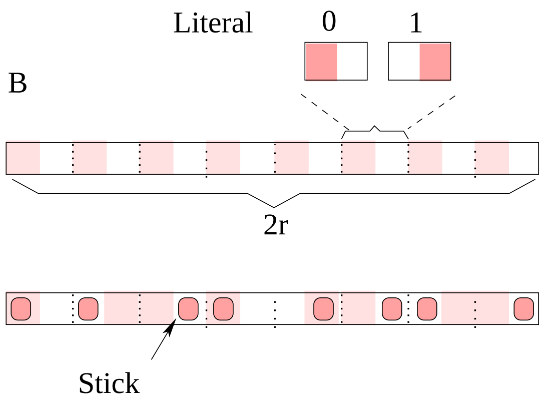


Figure 2. An illustration of the 1-dimensional lower bound construction. The top figure shows r sub-intervals of B, each containing a *literal*; the bottom figure shows a possible configuration of literals and sticks.

Consider a one-dimensional binary hierarchy. Let r be a power of two, and consider an interval B of length $2r$ in this hierarchy. We partition B into r sub-intervals, each of length two, and call them *literals*. Imagine the following 2-phase

[1] Cormode et al. [5] and Manku and Motwani [11] define an ε-approximate heavy hitter to have frequency at least $(\phi - \varepsilon)n$. We find our definition more convenient for our analysis.

data stream. In the first phase, either the left half or the right half subinterval of each literal is populated with $3\phi n/r$ points. Depending on whether the left half or the right half is populated, we assign the literal an *orientation* 0 or 1 (see Figure 2). In the second phase, an adversary chooses either the left or the right half in each literal and populates it with ϕn points: the intervals populated in the second phase are called the *sticks*; observe that stick intervals are clearly ϕ-*HHHs*. The algorithm, however, must decide whether any of the largest three intervals (B and its children) is a ϕ-*HHH* or not. Their status depends on which intervals the adversary chooses for sticks and, as we show below, determining it requires $\Omega(r)$ space.

Suppose that an algorithm A uses at most $0.01r$ bits of space. Then, at the end of phase one, it must *encode* the orientations of the r literal boxes in $0.01r$ bits. Because the orientation of each literal is 0 or 1, the sequence of r literal bits can be thought of as a vertex of the hypercube $\{0,1\}^r$. Because there are 2^r distinct orientation sequences, at least $2^{0.99r}$ of them must map to a single $(0.01r)$-bit code. Using a standard upper bound on the volume of a Hamming ball of radius $r/3$ in $\{0,1\}^r$, we know that this ball contains fewer than $2^{0.99r}$ vertices. We conclude that there are two points of $\{0,1\}^r$ at Hamming distance at least $r/3$ mapped to the same $(0.01r)$-bit code. These two sequences of literal orientations are indistinguishable for A.

Let σ_1 and σ_2 denote two orientation sequences with Hamming distance greater than $r/3$ that are mapped to the same code. In the first phase of the data stream, an adversary sets the orientations of the literals either according to σ_1 or σ_2, and then it chooses the distribution of the sticks by one of the two sequences.

If the distribution of the literals and the sticks coincide, then the discounted frequency of B is 0, in which case B should not be reported as a ϕ-*HHH*. But if the literals and the sticks have different distributions, then the points in at least $r/3$ literals are not covered by heavy sticks. In this case, the discounted frequency of B is at least $\frac{r}{3} \cdot 3 \cdot \phi n/r = \phi n$, and so B (or possibly one of its children if all the uncovered literals lie on the left or on the right side of B) is a ϕ-*HHH*. Because the algorithm cannot distinguish between these two cases, it reports incorrectly in at least one case.

This shows that any algorithm requires $\Omega(r)$ bits for this construction. By making r disjoint copies of the interval B in the first phase, and then using only one of them to complete the construction in the second phase, we obtain the claimed $\Omega(r^2)$ space lower bound. We choose r from the interval $(1/8\phi, 1/4\phi]$, so that the total number of points used in all r copies of B is $r \cdot 3\phi n + r\phi n = 4r\phi n \leq n$.

To apply this lower bound to randomized algorithms, we use a non-oblivious adversary argument. The adversary picks an orientation sequence uniformly at random from among the 2^r possibilities, feeds this to the randomized algorithm, and then non-obliviously chooses the stick orientation sequence. We argue that the randomized algorithm's probability of correctly determining whether B or one of its children is a ϕ-*HHH* is no more than $\frac{1}{2} + 2^{-0.07r}$

A randomized algorithm can be regarded as an initial random choice among a collection of deterministic algorithms, followed by deterministic execution of the chosen algorithm. (The initial algorithm choice is essentially the same as producing a sequence of random bits that will be used by a standard randomized algorithm whenever it needs to make a random choice.) In our problem, the adversary chooses an orientation sequence σ_1 uniformly at random and the randomized algorithm encodes it into $0.01r$ bits. Importantly, after σ_1 has been encoded the adversary knows the deterministic algorithm that was used to perform the encoding. Thus the adversary can compute the set of orientation sequences that map to the same code.

We call a code *heavy* if the number of sequences that map to it is larger than the size of a radius-$(r/3)$ Hamming ball, which is less than $2^{0.92r}$. Since the average code is mapped to by $2^{0.99r}$ sequences, the chance that a randomly chosen sequence does *not* map to a heavy code is less than $2^{-0.07r}$. If the algorithm maps σ_1 to a heavy code, then the adversary picks another sequence σ_2 that maps to the chosen code and whose Hamming distance from σ_1 is at least $r/3$. The adversary chooses the stick sequence from $\{\sigma_1, \sigma_2\}$ uniformly at random; since the two sequences are indistinguishable to the algorithm, its chance of deciding correctly whether B or one of its children is a ϕ-*HHH* is only $1/2$.

THEOREM 3.1. *Any deterministic or randomized* 1*-dimensional data stream algorithm that computes ϕ-HHHs with any fixed approximation error requires $\Omega(1/\phi^2)$ space in the worst case.*

4 Multi-Dimensional Lower Bound Construction

In this section, we present the main result of our paper: the $\Omega(1/\phi^{d+1})$ space lower bound in the node frequency model. We begin with a basic construction that gives the main intuition behind the lower bound. This construction shows that, for 2-dimensional streams, computing $1/4$-approximate ϕ-*HHHs* requires $\Omega(1/\phi^2)$ space in the node frequency model. In Section 4.2, we show how to extend this simple construction to establish the optimal space lower bound of $\Omega(1/\phi^3)$. We then extend the construction and show that $\Omega(1/\phi^3)$ space is required even for ε-approximation of the ϕ-*HHHs*, for any ε, $0 < \varepsilon < 1$. Finally, we generalize our lower bounds to d-dimensional streams, for all $d \geq 2$, and establish the general lower bound of $\Omega(1/\phi^{d+1})$.

4.1 The Basic Construction

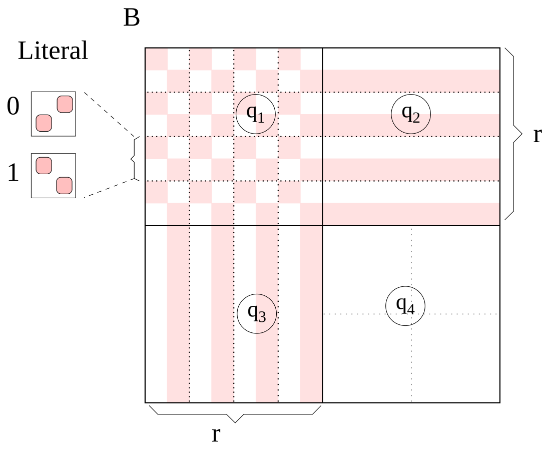
In this section, we describe our basic construction in two dimensions. We assume that the input stream is a set of two-dimensional points, drawn from a square box B, which is hierarchically composed with branching factor $b = 2$. Figure 3 shows the main components of our construction. The four box quadrants q_1, q_2, q_3 and q_4 are the grandchildren of the root box B. The pairs of neighboring boxes, namely, $q_1 \cup q_2$, $q_3 \cup q_4$, $q_1 \cup q_3$, and $q_2 \cup q_4$, are the children of B in the hierarchy.

Suppose ϕ is the user-specified heavy hitter parameter, and $\varepsilon = 1/4$ is our approximation parameter. A correct ϕ-*HHH* algorithm must report every box whose discounted frequency is above ϕn, and no box whose discounted frequency is below $\frac{3}{4}\phi n$. We will show that any algorithm in the node frequency model that uses less than $\Omega(1/\phi^2)$ space can be forced to *misclassify* the box B.

We work with two integer parameters $m = \Theta(\phi n)$ and $r = \Theta(1/\phi)$, where we assume that r is a power of 2. The parameter m is chosen so that $m < \frac{3}{4}\phi n$, but $2m \geq \phi n$; that is, a box with m points is below the heavy hitter threshold, but two such boxes combined exceed the threshold.

There are four important components in our construction: literals, diagonals, sticks, and the uniform box. These are boxes with carefully chosen frequencies. These objects arise by dividing the quadrant q_2 into r congruent horizontal slabs and the quadrant q_3 into r congruent vertical slabs. The extensions of these slabs partition the quadrant q_1 into an $r \times r$ regular grid. The four types of boxes and their point population are given as follows.

- **Literal**: The *literal boxes* are the $r^2 - r$ *non-diagonal* grid boxes in the quadrant q_1. Each literal box is divided into four congruent boxes, two of which are populated with $m/(2r^2 - 2r)$ points each. The pair of non-empty subboxes of a literal can have one of two configurations, called the *orientation* of the literal, as shown in Figure 3. We refer to one of these orientations as 0 and the other as 1.



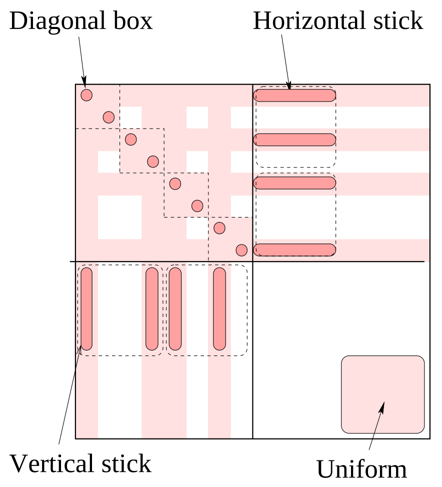


Figure 3. An illustration of the basic lower bound construction. The top figure shows the four quadrants of the box B; the bottom figure shows the four key components of the construction: literals, diagonals, sticks, and the uniform box.

- **Diagonal**: The *diagonal boxes* are the r grid boxes that lie on the main diagonal of q_1. Every diagonal box is also divided into four congruent boxes; the two subboxes along the diagonal are populated with m points each. Thus, each diagonal box is heavy, and does not contribute to the discounted frequency of B.

- **Stick**: We divide each of the r horizontal slabs of q_2 and vertical slabs of q_3 into four congruent pieces (bisecting horizontally and vertically), out of which a randomly chosen box adjacent to quadrant q_1 is called a *stick*. Every stick is populated with m points. Thus, no single stick, or slab, is heavy, but two adjacent slabs together are heavy. The sticks also do not contribute to the discounted frequency of the box B.

- **Uniform**: The quadrant q_4 is divided into four congruent squares, and the lower right square is populated with m points. We note that no vertical (horizontal) box smaller than $q_2 \cup q_4$ ($q_3 \cup q_4$) can span both the sticks and the uniform box. The population of the uniform box will always contribute to the discounted frequency of B.

We begin with the simple observation that changing the orientation of a literal has no effect on the frequency of any box not contained in the literal.

Proposition 4.1. *If the orientation of a literal L flips, then the frequency of any box R in the hierarchy that is not contained in L remains the same.*

Proof. If the box R is disjoint from L or contains L, then the flip clearly has no effect on the frequency of R. If the two boxes intersect, then in exactly one coordinate i we have $L_i \prec R_i$; that is, the extent of R in the ith coordinate contains the extent of L. The intersection $L \cap R$ consists of one complete row or column of L. Whether the orientation of L is 0 or 1, a row or column of L always contributes $m/(2r^2 - 2r)$ points. Thus the frequencies of $L \cap R$ and R are unaffected. $\square$

The following fact is an easy consequence of the node frequency model.

Proposition 4.2. *For any $k \in \mathbb{N}$, storing the orientation of k literal boxes requires at least $\Omega(k)$ space in the node frequency model.*

We now discuss the implications of the construction described above. In the quadrant q_1, only the diagonal boxes are heavy; the total frequency of all the literal boxes is m, which is less than $\frac{3}{4}\phi n$. In the quadrants q_2 and q_3, none of the sticks is heavy on its own, but pairs of adjacent sticks form heavy boxes. In the quadrant q_4, the total number of points is m, and so there is no heavy box there. Because of the hierarchical structure of the domain, any box that intersects two quadrants must have the full extent of B in at least one dimension.

Next, while no horizontal stick is heavy on its own, when combined with m points in the diagonal box, it forms a heavy row. Similarly, each vertical stick forms a heavy column when combined with a diagonal box. The only portion of the stream that is not covered by these heavy boxes lies either in the uniform box of quadrant q_4 or in the literal boxes of quadrant q_1. Because of the hierarchical structure, B is the smallest box intersecting (and containing) both q_1

and q_4. Thus, whether or not B is a ϕ-*HHH* depends on how many of the $\Theta(r^2)$ literal boxes are discounted. We argue that knowing this requires an algorithm to maintain the orientations of these literal boxes, which gives the desired space lower bound.

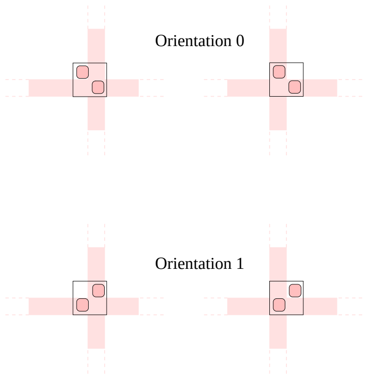


Figure 4. The orientations of a literal box, and its intersection with a heavy row or column.

Each literal intersects exactly one row and one column defined by the sticks. Figure 4 shows the combinatorially different ways this intersection occurs, depending on the orientations of literal boxes and the positions of the sticks. In each case, either all or half of the $m/(r^2 - r)$ points of the literal box are covered by the intersecting row and column. Since the row and column are heavy boxes, either all or half of the points in the literal are discounted from B. If all the points of all the literal boxes are covered by heavy rows and columns, then the discounted frequency of B is just m, which is the population of the uniform box. On the other hand, if only half of the points in the literal boxes are covered, then the discounted frequency of B is $g(B) = m + (r^2 - r) \cdot m/(2r^2 - 2r) = 3m/2$, which exceeds ϕn for appropriate choice of m.

We can now describe the data stream for our construction. The stream of input points arrives in two phases: in the first phase, we populate the quadrant q_1 including all the literal boxes with orientations specified below; in the second phase, we populate the diagonal boxes, the sticks, and the uniform box.

The orientations of the literal boxes, and the positions of the sticks, are determined by an adversary who can choose one of the following two variants of the data stream: either 10% of the literal boxes are fully covered by heavy stick boxes (and 90% are half covered), or 90% of the literal boxes are fully covered (and 10% are half covered). In the first case, the discounted frequency of B is $g(B) = m + \frac{9}{10} \cdot \frac{m}{2} = \frac{29}{20}m$, which exceeds ϕn, for $m = \frac{20}{29}\phi n$. Thus, in this case, B should be reported as a ϕ-*HHH*.

In the second case, however, the discounted frequency of B is $g(B) = m + \frac{1}{10}(r^2 - r) \cdot m/(2r^2 - 2r) = \frac{21}{20}m < \frac{3}{4}\phi n$, and so B should not be reported as a ϕ-*HHH*, for the given approximation error $\varepsilon = 1/4$. However, in the node frequency model of algorithms, in order for an algorithm to distinguish between these two cases, it must store the orientations of at least 10% of the literal boxes, which by Proposition 4.2 requires $\Omega(r^2)$ memory.

The only remaining part of the construction is to argue that $r^2 = \Omega(1/\phi^2)$. The total number of points in our construction (literal boxes, sticks, diagonals, and the uniform box) is $m + 2rm + 2rm + m = 2m(2r+1)$. Because $m = \frac{20}{29}\phi n$ and the stream size is n, the constraint $2m(2r+1) \leq n$ implies that $2r + 1 \leq 29/(40\phi)$. Thus, we can safely choose r to be the highest power of two for which $r \leq \frac{1}{4\phi}$, and so $r = \Theta(1/\phi)$. We have established the following result.

Lemma 4.3. *Any data stream algorithm in the node frequency model that computes approximate ϕ-HHHs in two dimensions with approximation error better than $\varepsilon = 1/4$ requires $\Omega(1/\phi^2)$ space.*

We note that this lower bound is *not* information theoretic. An algorithm that has access to all $r^2 - r$ literals can compute a summary of size $O(r)$ that can later be used to determine whether 10% or 90% of the literals are fully covered by the sticks. The binary matrix representing the pattern of the literals is an outer product of two binary r-vectors (the eventual stick patterns) perturbed by $r^2/10$ bits of noise. An algorithm based on data compression can extract the two r-vectors whose outer product determines 90% of the literal orientations. If the algorithm remembers these vectors and forgets the 10% noise superimposed on the literals, this is enough to determine whether B is a ϕ-*HHH* when the sticks arrive. However, this two-vector summary of the literal orientations only serves to show the *existence* of techniques that are outside the node frequency model. It does not suggest any general-purpose algorithm to circumvent the lower bound—indeed, the representation is *specifically* tailored to foil our lower bound construction, and is useless for other distributions of input points.

In the following sections, we improve the lower bound to $\Omega(1/\phi^3)$ by using multiple copies of the basic construction and also generalize the construction to d dimensions, $d > 2$.

4.2 An $\Omega(1/\phi^3)$ lower bound in two dimensions

We create r disjoint copies of the box B in our basic construction. In the first phase of the data stream, all $r^2(r-1)$ literal boxes are populated. In the second phase, the adversary chooses one of these copies for the rest of the construction. Because the algorithm cannot predict *which* copy of B will be used in the second phase, it is forced to keep in memory the orientations of $\Omega(r^2)$ literal boxes in each of the r copies, which requires $\Omega(r^3)$ total space.

The first phase of the stream uses rm points, m points for each of the r copies; the rest of the construction requires $m(4r+1)$ points as before. Thus, the total number of points in the construction is $(5r+1)m \leq n$. This condition is still satisfied if we set $m = \frac{20}{29}\phi n$ and $r \leq \frac{1}{4\phi}$ as in the previous subsection. We have the following theorem.

Theorem 4.4. *Any data stream algorithm in the node frequency model that computes approximate ϕ-HHHs in two dimensions with approximation error better than $\varepsilon = 1/4$ requires $\Omega(1/\phi^3)$ space.*

4.3 Lower bound for arbitrary approximation error ε

Our construction so far depends on the approximation error $\varepsilon = 1/4$. In this subsection, we show that the lower bound holds for any fixed $1/4 \le \varepsilon \le 1-\phi$. That is, even if the algorithm is required only to distinguish between boxes with discounted frequency at least ϕn and those with frequency at most $(1-\varepsilon)\phi n$, the space lower bound is still $\Omega(1/\phi^3)$.

In order to prove a lower bound for ε-approximation, we use a hierarchy in which the branching factor depends on ε. Specifically, we use $b = \Theta(1/(1-\varepsilon))$. In our construction, we assume that b is even and $b \ge 4$. The construction is similar to the one for binary hierarchies, however, the diagonal and uniform boxes are no longer necessary (the use of those boxes would slightly improve the dependence of b on ε). We first describe a construction in a bounding box B that yields a lower bound of $\Omega((1-\varepsilon)/\phi^2)$, which is then extended to the $\Omega(1/\phi^3)$ bound by using $\Theta(1/\phi(1-\varepsilon))$ disjoint copies of B.

The root box B has b^2 grandchildren, and each of the four quadrants (q_1, q_2, q_3, and q_4) contains $(b/2)^2$ grandchildren of B. Similarly to our basic construction, we define literal boxes in quadrant q_1, and sticks in quadrants q_2 and q_3. We use two parameters, r and m. The parameter r is a power of b to be fixed later. We choose m such that $m < (1-\varepsilon)\phi n$ but $\phi n \le (b/2)m$. That is, a box with discounted frequency m is not an ϕ-*HHH*, but $(b/2)m$ such boxes combined in a box of the hierarchy is already a ϕ-*HHH*. For any branching factor $b > 2/(1-\varepsilon)$, the choice $m = 2\phi n/b$ satisfies these conditions.

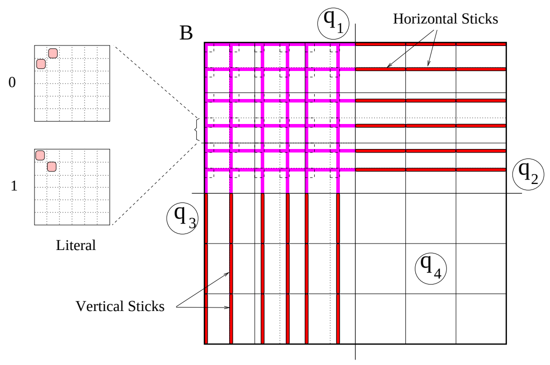


Figure 5. Schematic picture of our lower bound construction with branching factor $b = 6$. We chose $r = 2$ in this illustration for simplicity, even though r is a power of b in our construction.

The configuration of literals and sticks is similar to our construction for the the binary hierarchy. One key difference is that these boxes now lie within the b^2 grandchild boxes of B.

Literals: We divide each of the $(b/2)^2$ grandchild boxes of B lying in quadrant q_1 into $r \times r$ squares. These $b^2r^2/4$ boxes are the *literals*. We subdivide each literal L box into $r \times r$ subsquares. We populate two of the 2×2 subsquares in the upper left corner of L, each with $2m/(r^2b)$ points. The two subsquares with non-zero population can have two configurations, which are the possible *orientations* of the literal box (see Figure 5). Observe that Propositions 4.1 and 4.2 hold for the literal boxes in this setting, too.

Sticks: Divide the $(b/2)^2$ grandchildren boxes of B lying in quadrant q_2 (resp., q_3) into r horizontal (vertical) slabs. We divide each slab S into r horizontal (vertical) subslabs. The *stick* of S is one of the two uppermost (leftmost) subslabs within S. The horizontal (vertical) extension of a stick intersects $b/2$ slabs in q_2 (q_3). We choose the same subslab to be a stick in each of these $b/2$ horizontally (vertically) aligned slabs, and so the extension of every stick contains $b/2$ sticks. Each stick is populated with m points.

A heavy *row* (*column*) is the horizontal (vertical) extension of a stick in q_2 (q_3) to the full extent of B. Since b is the branching factor, the heavy row (column) is the parent box of a stick. No single stick is heavy on its own, but a heavy row (column) contains at least ϕn points. Within every grandchild box of B lying in quadrant q_2 and q_3, no stick or slab is heavy on its own, but the union of b consecutive slabs is already heavy.

Similarly to our construction for $b = 2$, the input stream of points arrives in two phases. In the first phase, the $b^2r^2/4$ literal boxes are populated with $4m/(r^2b)$ points each. In the second phase, the $2(b/2)^2r = b^2r/2$ sticks are populated with m points each. An adversary chooses the orientations of the literals and the positions of the sticks such that either all the literal boxes are half covered by heavy rows and columns, or only a $2/b$ fraction of the literal boxes are half covered and the rest are fully covered.

In the first case the discounted frequency of B is $g(B) = (b^2r^2/4) \cdot 2m/(r^2b) = bm/2$, and so B should be reported as a ϕ-*HHH*. In the second case, $g(B) = (2/b) \cdot (b^2r^2/4) \cdot 2m/(r^2b) = m$, and so B is not a ϕ-*HHH*. To distinguish between these two cases in the node frequency model, any deterministic algorithm must store the orientations of at least a $2/b$ fraction of the literal boxes. By Proposition 4.2, this requires $\Omega(br^2)$ memory.

It remains to show that $br^2 = \Omega((1-\varepsilon)/\phi^2)$. Recall that $2/(1-\varepsilon) < b \le 4/(1-\varepsilon)$ and $m = 2\phi n/b$. The total weight of all the literal boxes is $(b/2)^2r^2 \cdot 4m/br^2 = bm$, and the weight of all the sticks is $(b^2r/2) \cdot m = b^2rm/2$. The total number of points is $(br+2)bm/2 = (br+2)\phi n$, which must be no greater than n. We choose $r = \Theta(1/b\phi) = \Theta((1-\varepsilon)/\phi)$, for $0 \le \varepsilon \le 1-\phi$. This gives $br^2 = \Omega((1-\varepsilon)/\phi^2)$.

Finally, notice that $2m$ data points arrived in the first phase and $b^2rm/2$ points in the second phase. By combining $\Theta(b^2r)$ disjoint copies of the box B so that the literal boxes of all copies are populated in the first phase but only one copy is completed in the second phase, we obtain a lower bound of $\Omega(b^3r^3) = \Omega(\phi^{-3})$.

4.4 The lower bound in d dimensions

In this section, we present the d-dimensional lower bound construction for a fixed approximation error $\varepsilon = 1/2^d$. Our construction can be extended to an arbitrary ε following the scheme described in the previous section.

The points in the stream are drawn from a d-dimensional box B. This box is partitioned into 2^d congruent parts $q_1, q_2, \ldots, q_{2^d}$ (bisect along each dimension). Among these 2^d boxes, only $d+2$ will have non-zero frequency of points. The box q_1 is further partitioned into r^d congruent pieces, where r is a power of two that is $\Theta(1/\phi)$. The r boxes along the main diagonal are the *diagonal boxes*; in each diagonal box, we place two points of weight m each at two opposite corners. All other boxes are *literal boxes*. Each literal box has $m/(r^d - r)$ points, which are placed in one of two distinct *orientations*. Only half of the 2^d congruent subboxes of a literal are populated, each with $2^{1-d}m/(r^d - r)$ points. The populated boxes are chosen such that the neighbor of every populated box is empty, and the populated subboxes form one of the two possible d-dimensional checkerboard patterns in each literal box.

The *sticks* are located in the d subboxes of B that are adjacent to q_1. We assign a unique coordinate axis to each of those subboxes. Each box's axis is chosen among the $d-1$ axes parallel to the hyperplane separating the box from q_1. Consider the box q_i adjacent to q_1. We partition q_i into r congruent slabs by hyperplanes orthogonal to its assigned axis. In each slab, we choose a stick of population m as follows: Partition the slab into four congruent pieces by a hyperplane orthogonal to the assigned axis of q_i and a hyperplane parallel to the side common to q_i and q_1. We choose the stick randomly from the two subboxes adjacent to q_1. Finally, the *uniform* box is a box of population m such that its projection in all d directions is disjoint from all the sticks. We now argue that a deterministic algorithm can decide whether or not B is an ε-approximate hierarchical heavy hitter only if it knows the orientations of at least a constant fraction of all literal boxes.

As in the 2-dimensional construction, no stick is heavy on its own, because $m < (1-\varepsilon)\phi n = (1-2^{-d})\phi n$, but each stick belongs to a slab that is a ϕ-*HHH* when extended to q_1, because it contains half of the weight of a diagonal box $(m + m \geq \phi n)$. In each literal, either all the points are covered by stick slabs or a 2^{1-d} fraction of the points is uncovered, depending on the orientation of the literal. A correct algorithm must distinguish between data streams in which either 90% of the literal boxes are fully covered or 10% of them are fully covered. In the first case, the discounted count of B is $g(B) = \frac{1}{10} \cdot (r^d - r)2^{1-d}m/(r^d - r) + m = \frac{1+5\cdot 2^d}{5\cdot 2^d}m$, while in the second case, it is $g(B) = \frac{9+5\cdot 2^d}{5\cdot 2^d}m$. By setting $m = \frac{5\cdot 2^d}{9+5\cdot 2^d}\phi n$, the status of box B will be different in the two cases. This establishes a lower bound of $\Omega(1/\phi^d)$.

We can extend the lower bound to $\Omega(1/\phi^{d+1})$ by using r disjoint copies of q_1. The total number of points is $r \cdot m + 2rm + drm + m = ((d+3)r + 1)m$, which must be no more than n. Thus, we can choose $r = \Theta(\phi^{-1})$. This establishes a lower bound of $\Omega(r^{-(d+1)}) = \Omega(\phi^{-(d+1)})$ for d-dimensional hierarchies.

Also, using branching factors that grow with $1/\varepsilon$, as discussed in the previous section, the lower bound can be generalized to an arbitrary level of approximation $0 < \varepsilon \leq 1-\phi$. We summarize the main result of this section in the following theorem.

THEOREM 4.5. *Any data stream algorithm in the node frequency model that computes ε-approximate ϕ-HHHs in d dimensions for any ε, $0 < \varepsilon < 1-\phi$, requires $\Omega(1/\phi^{d+1})$ space in the worst case.*

5 The Upper Bound

In this section, we describe a data structure of size $O(1/\phi^{d+1})$ for computing approximate ϕ-*HHHs*. This indicates that our lower bounds in the previous section are asymptotically tight in terms of ϕ. In essence, we show that if the frequency of any node of the hierarchy is known within an additive error of $\phi^{d+1}n/a^d$, where a is the size of the maximal antichain in the hierarchy, then we can determine ϕ-*HHHs* with fixed approximation error.

We estimate the frequency of nodes (boxes) by using the hierarchical heavy hitter algorithm of Cormode et al. [6], which is, in turn, based on the *lossy counting* scheme of Manku and Motwani [11] and the lattice structure of the hierarchy described in our Subsection 2.1. By using the Misra–Gries [12] algorithm instead of the Manku–Motwani scheme, the algorithm in [6] can maintain a data structure (*summary*) of size $O(\frac{H}{\alpha})$ that can estimate the frequency of any node in the hierarchy with absolute error at most αn, for $\alpha \in (0,1)$. In particular, this summary maintains an approximate frequency count for a set of *fringe nodes*: these are the nodes whose frequency is above $\alpha n/2$ but all of whose children nodes have frequency less than αn. For the nodes *below* the fringe, the summary does not maintain any information: the frequency estimate for these nodes is zero. The frequencies of nodes *above* the fringe are approximated from the frequencies of the fringe nodes and from additional correctional counts stored in the summary.

We use the scheme of [6] with the approximation parameter value $\alpha = \varepsilon\phi^{d+1}/a^d$. We inductively process the ancestors of the fringe nodes in a bottom-up scan of the hierarchy. We compute the approximate discounted frequency of each node with an error of at most $\varepsilon\phi n$, and put it in a set Y of ϕ-*HHHs* if this value is above ϕn.

The discounted frequency of a node B is the difference of the frequency $f(B)$ and the frequency $\mathrm{dis}(B)$ of the union of all descendants of B in Y. The algorithm in [6] provides a fairly good approximation of $f(B)$ since $\alpha \ll \phi$. In the full paper we show that $\mathrm{dis}(B)$ can be expressed as the sum of $(a/\phi)^d$ node frequencies, each multiplied by ± 1. Since each node frequency is known to within an additive error of αn, the approximation error for $\mathrm{dis}(B)$ is $\varepsilon\phi n$.

6 Conclusions

Data streams have emerged as an important paradigm for processing on-line data. In many such applications, finding heavy hitters, or frequent items, is a fundamental problem. Hierarchical heavy hitters (HHH) were introduced as a natural generalization of heavy hitters to hierarchical and multi-dimensional data domains by Estan and Varghese [9] and Cormode et al. [5, 6]. The algorithms proposed in [5, 6, 8] guarantee only that the *total* frequency, and not the *discounted* frequency, of reported items is above the threshold, creating the false positives problem. In order to overcome this shortcoming, Cormode et al. [5, 6] and Estan, Savage, and Varghese [8] solve a weaker form of ϕ-*HHH*s, where an item's frequency is distributed among its ancestors. However, they leave open the problem of determining the true ϕ-*HHH*s with a space-efficient algorithm.

Our paper resolves this question by proving lower bounds on the space complexity of algorithms that find HHHs. Our first result is quite general: it shows an $\Omega(1/\phi^2)$ space lower bound for *any* (deterministic or randomized) algorithm, even for 1-dimensional data streams. Thus, the space-efficient schemes in [5, 6], which use roughly $O(\frac{H}{\phi}\log \phi n)$ space, cannot be expected to produce guaranteed quality hierarchical heavy hitters, even in one dimension. This seems surprising because several deterministic and space-efficient algorithms are known for approximating the (non-hierarchical) heavy hitters with arbitrary accuracy. The lower bound shows that hierarchical heavy hitters are fundamentally more difficult to estimate.

We show that the memory requirement grows *exponentially* with d. That is, to find ϕ-*HHHs* with any fixed accuracy requires space $\Omega(1/\phi^{d+1})$. In on-line stream processing, the node frequencies are known only approximately due to the limited space. Interestingly, our construction shows that the lower bound for ϕ-*HHHs* holds even if the node frequencies are known *exactly*.

Our multi-dimensional lower bound operates in the node frequency model. This is a simple and restrictive model, but it is general enough to include many of the (deterministic) stream algorithms known for finding heavy hitters. Unfortunately, the node frequency model does not apply directly to the multi-dimensional HHH algorithm of Cormode et al. [6]. Their scheme is based on lossy counting, which *does* fall into this model, but it also uses an additional counter to deal with the problem of overlapping descendants. This secondary counter is not (quite) a node frequency, and so the model does not strictly apply. However, it is easy to argue that our lower bound construction works for their algorithm as well. For simplicity, consider our lower bound construction in two dimensions. Imagine that the input stream uniformly populates the literal boxes with some fixed orientation. After the first phase, there are $\Theta(1/\phi^2)$ identical literal boxes. Because the algorithm has only $O(1/\phi)$ memory, it does not have counters for at least $\Omega(1/\phi^2)$ literals. If the counter for a literal is deleted, the compensating count passed to the grandparent is $m/(r^2 - r)$. Most importantly, the compensating count passed to the grandparent *does not depend on the orientation* of the literal. Therefore, if we feed another instance of the stream to this algorithm, where only the orientations of some of these $O(\phi^{-2} - \phi^{-1})$ literals are flipped, the algorithm cannot detect the switch. But, as our analysis shows, switching the orientation of the literals can change the status of B as a possible ϕ-*HHH*, and so the algorithm in [6] cannot correctly identify whether B is an approximate hierarchical heavy hitter or not.

Several natural open problems are suggested by our work. In particular, it would be interesting to explore what can be *provably* accomplished by a space-efficient scheme, for example, one using space roughly linear in the accuracy level. Our lower bound construction shows that the estimated discounted frequency of *one particular item* can be arbitrarily off target. Can this be the case with most of the items reported as ϕ-*HHH*s, or is it possible to identify at least a *constant fraction* of the true hierarchical heavy hitters?

In our lower bound construction, the error occurs for an item whose discounted frequency is near the threshold. Can it be shown that all items with discounted frequency significantly above the threshold can be found accurately?

References

[1] N. Alon, Y. Matias, M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.* **58** (1999), 137–147.

[2] A. Arasu and G. Manku. Approximate counts and quantiles over sliding windows. In *Proc. 23rd PODS*, 2004, ACM Press, pp. 286–296.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *21st PODS*, 2002, ACM Press, pp. 1–16.

[4] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *29th Proc. ICALP*, LNCS, Springer-Verlag, 2002, pp. 693–703.

[5] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in data streams. In *Proc. 29th Conf. on Very Large Data Bases*, 2003.

[6] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. in *Proc. ACM SIGMOD*, 2004.

[7] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *Proc. 10th European Sympos. Algorithms*, LNCS 2461, 2002, pp. 348–360.

[8] C. Estan, S. Savage, and G. Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proc. of ACM SIGCOMM*, 2003.

[9] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proc. 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001, pp. 75–80.

[10] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems* **28** (1) (2003), 51–55.

[11] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. 28th Conf. Very Large Data Bases*, 2002, pp. 346–357.

[12] J. Misra and D. Gries. Finding repeated elements. *Sci. Comput. Programming* **2** (1982), 143–152.

[13] S. Muthukrishnan. Data streams: Algorithms and applications. Preprint, 2003.