Streaming Algorithms for Measuring H-Impact

Priya Govindan
Rutgers University
Piscataway, NJ, USA
priyagn@cs.rutgers.edu

Morteza Monemizadeh
Rutgers University
Piscataway, NJ, USA
mortezam@cs.rutgers.edu

S. Muthukrishnan
Rutgers University
Piscataway, NJ, USA
muthu@cs.rutgers.edu

ABSTRACT

We consider publication settings with positive user feedback, such as, users publishing tweets and other users retweeting them, friends posting photos and others liking them or even authors publishing research papers and others citing these publications. A well-accepted notion of “impact” for users in these settings is the H-Index, which is the largest k such that at least k publications have k or more (positive) feedback.

We study how to calculate H-index on large streams of user publications and feedback. If all the items can be stored, H-index of a user can be computed by sorting. We focus on the streaming setting where as is typical, we do not have space to store all the items.

We present the first known streaming algorithm for computing the H-index of a user in the cash register streaming model using space poly(1/ε, log(1/δ), log n); this algorithm provides an additive ε approximation. For the aggregated model where feedback for a publication is collated, we present streaming algorithms that use much less space, either only dependent on ε and even a small constant. We also address the problem of finding “heavy hitters” users in H-index without estimating everyones’ H-index. We present randomized streaming algorithms for finding 1 + ε approximation to heavy hitters that uses space poly(1/ε, log(1/δ), log n) and succeeds with probability at least 1 − δ. Again, this is the first sublinear space algorithm for this problem, despite extensive research on heavy hitters in general. Our work initiates study of streaming algorithms for problems that estimate impact or identify impactful users.

1. INTRODUCTION

Consider a system that lets users u_i publish items p(i,j) and have other users respond to the items. These could be for example users in Twitter. Users publish their tweets or embed hashtags, and the others may respond by retweeting the publication or posting with the same hashtag handle and so on. Likewise, a user in Facebook may post a story or picture, and their friends can “like” it. In a more familiar setting, researchers publish their results and others cite the publications. In any publication setting, these responses may be positive or negative to varying extents or even just neutral or mix of these responses. Our research is on the positive impact of users, and hence, for the purposes of our research here, we will only focus on the positive responses. 1

Our focus is on how to measure users’ impact. There is a vast theory of studying influence of individual user actions on web-pages, posts, etc and also the cumulative impact of all of a user’s publications [14, 13, 12]. We adopt a very simple and fundamental notion of impact, well-known in academic world, called the H-index [7]. For each item p(i,j) published by user i, say the number of positive responses is R(i,j). Then, the H-index of user i is the largest k such that at least k publications p_1, p_2, ..., p_k have k or more (positive) feedback, that is, R(i_1,j_1, i_2,j_2, ..., i_k,j_k) ≥ k, for 1 ≤ k ≤ k. This is one of the most popular measures of impact in academia, it arose to popularity in the Physics community and is now widespread, with Google Scholar cited frequently by researchers. One of its attractions is that it is parameter-free, for example, it does not focus on the total responses for top 10 cited publications or the total number of publications with at least 10 citations each etc., where 10 is a parameter. It combines the number of publications and the number of responses (citations) for each in a nifty way, to eke out an impact measure. There is vast literature on extensions of H-index, for example, normalizing for areas, years of publication, different transformations like at least k publications have total of k^2 or more (positive) feedback , or applying it to groups of users, and so on [2, 4, 3]. See [1] for a discussion of the research on H-index.

We veer from prior research on H-index, in not studying its applicability or in extending its definition, but in computing it. If we have memory to store every response r(k,i,j) for each publication p(i,j), then it is easy to compute the H-index of user i. For example, one can calculate R(i,j) by aggregating over k for each (i,j), sorting based on R(i,j)'s, and determining H-index by considering them in say the decreasing order. However, in applications, the publications and responses arrive online over time as they are composed and released. Further, the number of publications and responses may be very large, and certainly they are prohibitive if we have a large number of users like the popular publication forums. Motivated by this, we study the streaming versions of the problem of computing the H-index.

1.1 Our Contribution

Our main results are very small space streaming algorithms for computing the H-index. We consider both aggregated and unaggregated models of streaming, and not only study computing H–index

1Our discussion can be extended to the setting when the responses are positive to different extent or when responses are negative only or when responses can be a mix of positive and negative, etc. This will lead to notions of impact of users different from the one we study here.
per user, but also computing the heavy-hitter users, that is, users who have a large H-index without explicitly computing each users’ H-index. In more detail, our results are as follows.

- **(Aggregate Streaming Model)** We consider a stream of numbers $R(i,j)$ in arbitrary order of $i$’s and $j$’s, and we design an algorithm for computing H-index of each user $i$. We present a streaming algorithm that uses only $O(e^{-1} \log e^{-1})$ space and returns an $1-e$ approximation to the H-index of a user. Further if there is a mild lower bound $\Omega(e^{-3} \log n)$ on the ultimate H-index and the publications are considered in a random order, we present an improved algorithm that uses only $O(1)$ storage, independent of the number of publications $j$’s, number of feedbacks $R(i,j)$’s or the approximation factor $\epsilon$. These are some of the most efficient streaming algorithms one could hope for, not even using $O(\log n)$ words that is standard for a lot of streaming analysis over signals of size $n$.

- **(Cash Register Streaming Model)** In the so called “cash register” streaming model, we get a stream of responses $r(k, i, j)$, that is, the responses arrive unaggregated. We present randomized streaming algorithms for estimating the H-index to $\epsilon$ additive error using space poly$(1/\epsilon, \log(1/\delta), \log n)$ and with success probability at least $1-\delta$. This is the first sublinear space algorithm for this problem.

- **(Heavy Hitters in H-index)** We address a data mining problem, given a cash register stream of publications and responses, identify the users whose H-index is large. For user $i$, let their H-index be denoted $h(i)$. We formulate this as a “heavy hitters” problem, that is output set $S$ of users such that for each user $i \in S$, $h(i) \geq eR$, where $R$ is the total number of responses. In the cash-register streaming model, we present randomized streaming algorithms for finding $1+\epsilon$ approximation to heavy hitters using space poly$(1/\epsilon, \log(1/\delta), \log n)$ that succeeds with probability at least $1-\delta$. Again, this is the first sublinear space algorithm for this problem, despite extensive research on heavy hitters in general.

We use many of the tools that have been developed in streaming research over the past few years. This includes exponential bucketing methods, distinct or $L_0$ sampling, group testing with a subset of items to find the heavy hitting ones, etc. We adopt them to calculate H-index, but in the process we significantly improve the results one can get. For example, while using exponential bucketing, we prove that the bucket counters can be reused for our problem and as a result get streaming algorithms that use very little space, sub-logarithmic or constant space even, while typical use of these techniques for other streaming problems use logarithmic space or more. In application of group testing to find heavy hitters, there are no decoding methods for finding the one heavy hitter in H-index among a set of users: we design a new algorithm for this problem using the properties of H-index. Our work initiates study of streaming algorithms for problems that estimate impact or identify impactful users. There are many variations of H-index and other impact measures for users, and streaming algorithms for calculating them per user, or mining anomalous users on such impact measures is a rich area for further study.

2. PRELIMINARIES

In this section, we present the definition of H-index and the different data stream models.

2.1 H-Index

The H-index was suggested in 2005 by Jorge E. Hirsch as a tool for determining theoretical physicists’ relative quality and is sometimes called the Hirsch index or Hirsch number.

**Definition 1** (H-Index [7]). Let $V \in \mathbb{N}^n$ be a vector of natural numbers of dimension $n$. We denote by $h^*(V)$ the maximum number $i \in [n]$ such that the number of elements of $V$ that are greater than or equal to $i$ is at least $i$, i.e.,

$$h^*(V) = \arg \max_{i \in [n]} \{ |j \in [n] : V[j] \geq i | \},$$

where we denote $[n] = \{ 1, 2, \ldots, n \}$. In other words, for the sorted vector $V'$ of $V$ in descending order, h-index of $V'$ (and $V$) is defined as $h^*(V) = h^*(V') = \max_{i \in [n]} \{ \min(V'[i]), i \}$.

We let $H(V) = \{ |V[i] : V[i] \geq h^*(V) | \}$ be the support of the H-index of the vector $V$ which is defined as the set of values $V[i]$ that are greater than or equal to the H-index $h^*(V)$. We drop $V$ and $V'$ from $h^*(V)$ and $h^*(V')$ when it is clear from the context.

**Example 2.** Let $V' = [5, 5, 6, 5, 5, 6, 5, 5, 5]$ be a vector of 10 numbers. Let $V' = [5, 6, 5, 6, 5, 5, 5, 5, 5]$ be the sorted variant of $V$. Then, for the H-index $h^*(V)$ of $V$ which is the same as $h^*(V')$ we have $h^*(V) = h^*(V') = 5 = \max_{i \in [10]} \{ \min(V'[i], i) \}$, but $H(V) = H(V') = \emptyset$.

2.2 Author, Paper and Citation Model

In this work, we consider publication settings that allows users to give positive feedback on other users’ activities. Examples of such a setting include tweets and re-tweeting, Facebook posts and likes and academic papers and citations. We define this setting in the general terms of authors, papers and citations.

Let $A$ be a set of authors and $P$ be a set of papers. Each paper $p \in P$ has a citation number $c_p$ which is the number of papers in $P$ that cite the paper $p$. As is common in science we assume a paper $p$ cannot cite itself.

We denote the set of papers of an $a \in A$ by $P_a \subseteq P$. Similarly, we denote the set of authors of a $p \in P$ by $A_p \subseteq A$. For the sake of simplicity, we assume that there is a natural number $x$ for the maximum number of authors that a paper $p$ can have, that is, $|A_p| \leq x$. We represent a paper $p \in P$ by a tuple $(p, a_1^p, \ldots, a_x^p, c_p)$ for $y \leq x$, where $a_1^p, \ldots, a_x^p$ and $c_p$ are the authors and the citation number of the paper, respectively. Thus, from now on, we consider a paper $p \in P$ or $(p, a_1^p, \ldots, a_x^p, c_p) \in P$, interchangeably.

2.3 Aggregate and Cash-Register Streams

Let $V \in \mathbb{N}^n$ be a vector of natural numbers of dimension $n$. Let $h^*(V)$ be the H-index of $V$. We say $S$ is an aggregate stream of the vector $V$ if $S$ is a (possibly adversarially ordered) stream of the elements of the underlying vector $V$. We say $S$ is a randomly ordered aggregate stream if $S$ is a random order chosen uniformly at random from the set of all permutations (or orders) of the elements of the underlying vector $V$. We say $S$ is a cash register stream if $S$ is a stream of updates to the underlying vector $V$, where an update at time $t$ is a pair $(i, z)$ for a natural number $z$ that replaces the entity $V[i]$ by $V[i] + z$.

For the data model described in Section 2.2, the aggregate streaming model would consist of one tuple $(p, a_1^p, \ldots, a_x^p, c_p)$ for each paper $p$ in the dataset, where $c_p$ is the number of citations of the paper. In the cash register model for the above data model, each tuple $(p, a_1^p, \ldots, a_x^p, c_p)$ corresponds to the $i$th update to the number of citations of paper $p$, such that $c_p = \sum_i c_p^i$. Note that in Section 3, for the sake of simplicity we assume the case when there is only

---

2The constant is just 6 as our proof shows.
one author in the stream. This can easily be extended to papers with multiple authors and computing H-index for each author.

We say a streaming algorithm \( A \) is a multiplicative \((\epsilon, \delta, s)\)-approximation algorithm for a stream \( S \) (either aggregate or cash register) if over one pass over \( S \), \( A \) uses a space \( s \) to report an estimator \( h(V) \) such that

\[
\Pr[h^*(V) - h(V)] \leq \epsilon h^*(V) \geq 1 - \delta.
\]

Similarly, we say a streaming algorithm \( A \) is an additive \((\epsilon, \delta, s)\)-approximation algorithm for a stream \( S \) (either aggregate or cash register) if one pass over \( S \), \( A \) uses a space \( s \) to report an estimator \( h(V) \) such that

\[
\Pr[h^*(V) - h(V)] \leq \epsilon n \geq 1 - \delta.
\]

2.4 \( \ell_0 \)-Sampler

We define an \( \ell_0 \)-Sampler as follows.

**Definition 3. \( (\ell_0 \text{-Sampler}) \) [5, 11] Let \( 0 < \delta < 1 \) be a parameter. Let \( S = (a_1, t_1), \ldots, (a_n, t_n) \) be a stream of updates of an underlying vector \( x \in \mathbb{R}^n \) where \( a_i \in [n] \) and \( t_i \in \mathbb{R} \). The \( i \)-th update \((a_i, t_i)\) updates the \( a_i \)-th element of \( x \) using \( x(a_i) = x(a_i) + t_i \). An \( \ell_0 \)-Sampler algorithm for \( x \neq 0 \) returns \( \text{FAIL} \) with probability at most \( \delta \). Else, with probability \( 1 - \delta \), it returns an element \( j \in [n] \) such that the probability that \( j \)-th element is returned is \( \Pr[j] = \frac{|x(j)|}{\sum_{i \in [n]} |x(i)|} \).

Here, \( \ell_0(x) = \sum_{i \in [n]} |x(i)|^0 \) is the (so-called) \( \ell_0 \)-norm of \( x \) that counts the number of non-zero entries.

**Lemma 4 ([9]).** Let \( 0 < \delta < 1 \) be a parameter. Then, there exists a linear sketch-based algorithm for \( \ell_0 \)-sampling using \( O(\log^2 n \log \delta^{-1}) \) bits of space.

3. H-INDEX IN STREAMS

In this section we present our algorithms to compute H-index of a single user in different streaming settings, namely aggregate, random order and cash-register models. In Section 3.1, we consider the aggregate streaming model, where the input stream is the number of likes or retweets of different posts of a user. In Section 3.2, we assume that the stream of number of positive feedback is given in a random order. In a cash-register streaming model the input is not the total number of likes or retweets per post, but a sequence of updates when a new like or retweet occurs. In Section 3.3, we present our algorithm to compute H-index of a user when the input is a cash-register model.

3.1 Aggregate Streaming Model

In this section, we first develop a streaming algorithm that using a space \( 2\epsilon^{-1} \log n \) words reports an \( (1 \pm \epsilon) \) estimator of the H-index of a stream. A similar approach was used by Govindan et al. [6] to compute H-index of core number of nodes in an effort to compute k-cores in graphs. Later we improve the space of this algorithm down to \( 6\epsilon^{-1} \log 3e^{-1} \) words. Formally, we first prove the following theorem. The pseudocode of the algorithm and the correctness proof of the algorithm are given after it.

**Theorem 5 (Exponential Histogram).** Let \( 0 < \epsilon < 1 \) be a parameter. Let \( V \in \mathbb{N}^n \) be a vector of natural numbers of dimension \( n \) whose H-index is \( h^*(V) \). Let \( S \) be an arbitrarily (possibly adversarially ordered) stream of the elements of the underlying vector \( V \). Then, there exists a one-pass deterministic streaming algorithm (Algorithm 1) that using space \( 2\epsilon^{-1} \log n \) words (where each word consists of \( \log n \) bits) reports an estimator \( h(V) \) such that

\[
(1 - \epsilon)h^*(V) \leq h(V) \leq h^*(V).
\]

Here for the simplicity of the exposition, we assume \( n = (1 + \epsilon)^k \) is a power of \( (1 + \epsilon) \) and we observe that \( n \) is a trivial upper bound for the H-index \( h^*(V) \) of an underlying vector \( V \). We consider \( \log_{1 + \epsilon} n \) guesses of \((1 + \epsilon)^k \) for \( h^*(V) \). For each guess \((1 + \epsilon)^k \), we design a counter \( c_i \) that counts the number of elements of the stream \( S \) that are greater than or equal to \((1 + \epsilon)^k \). At the end of the stream, we find the greatest guess \((1 + \epsilon)^k \) whose counter is at least \((1 + \epsilon)^k \), but the counter of the subsequent guess \((1 + \epsilon)^{k+1} \) is less than \((1 + \epsilon)^{k+1} \).

**Algorithm 1: Exponential Histogram**

**Input:** Stream \( S \) of elements of \( V \).
1. Initialize counters \( c_1 = 0 \) to zero for \( i \in [\log_{1 + \epsilon} n] \).
2. For step \( t \) that the element \( S_t \) arrives
   - For \( i = 1 \) to \( \log_{1 + \epsilon} n \)
     - If \( S_t \geq (1 + \epsilon)^i \), then \( c_i = c_i + 1 \)
3. Let \( h(V) = (1 + \epsilon)^i \) be the greatest threshold such that \( c_i \geq (1 + \epsilon)^i \) and \( c_{i+1} < (1 + \epsilon)^{i+1} \).
4. Output \( h(V) \).

**Proof.** Observe that there exists an index \( i \in [\log_{1 + \epsilon} n] \) such that \((1 + \epsilon)^i \leq h^*(V) < (1 + \epsilon)^{i+1} \). Let us consider two counters \( c_i \) and \( c_{i+1} \). The counter \( c_i \) corresponds to the number of elements of the stream \( S \) that are greater than or equal to \((1 + \epsilon)^i \). Similarly, the counter \( c_{i+1} \) corresponds to the number of elements of the stream \( S \) that are greater than or equal to \((1 + \epsilon)^{i+1} \). Observe that since \((1 + \epsilon)^i \leq h^*(V) < (1 + \epsilon)^{i+1} \), we have \( c_i \leq h^*(V) \leq (1 + \epsilon)^i = h(V) \). Therefore,

\[
(1 - \epsilon)h^*(V) \leq h(V) \leq (1 + \epsilon)^i \leq h^*(V) .
\]

The space of the algorithm is \( \log_{1 + \epsilon} n \leq 2\epsilon^{-1} \log n \) words, where each word consists of \( \log n \) bits.

Next we improve the space of this algorithm to \( 6\epsilon^{-1} \log 3e^{-1} \) words by observing that we do not need to keep the counters for all the guesses at each step of the stream, but only a shifting window of \( x = O(1/\epsilon) \) consecutive counters.

**Theorem 6 (Shifting Window).** Let \( 0 < \epsilon < 1 \) be a parameter. Let \( V \in \mathbb{N}^n \) be a vector of natural numbers of dimension \( n \) whose H-index is \( h^*(V) \). Let \( S \) be an arbitrarily (possibly adversarially ordered) stream of the elements of the underlying vector \( V \). Then, there exists a one-pass deterministic streaming algorithm (Algorithm 2) that using space \( 6\epsilon^{-1} \log 3e^{-1} \) words (where each word consists of \( \log n \) bits) reports an estimator \( h(V) \) such that

\[
(1 - \epsilon)h^*(V) \leq h(V) \leq h^*(V) .
\]

We maintain a window of \( x = O(1/\epsilon) \) consecutive counters \( c_1, c_{1+1}, \ldots, c_{\log_{1 + \epsilon} n} \) where the counter \( c_{i+1} \) corresponds to the revealed elements of the stream that are greater than or equal to \((1 + \epsilon)^{i+1} \). At every step \( t \) of the stream if the number of elements that are greater than or equal to \((1 + \epsilon)^{i+1} \) becomes at least \((1 + \epsilon)^{i+1} \), we shift the window of counters by one, i.e., we delete the counter \( c_i \), create a new counter \( c_{i+1+x+1} \), and set \( i = i + 1 \).
We prove that the number of elements of the stream that are greater than or equal to $(1 + e)^{i+1}$, but not counted by the counter $c_{i+1}$ (mainly because they have been revealed before the counter $c_{i+1}$ is created) is at most $e(1 + e)^{i+1}$ which is an $e$-fraction of the H-index of the stream if the H-index of the stream is at least $(1 + e)^{i+1}$.

**Algorithm 2: Shifting Window**

**Input:** Stream $S$ of elements of $V$.

1. Let $X = \{0, 1, 2, \cdots, r = \lceil \log_{1+e} e^{-1} \rceil \}$ and $i = 1$.
2. Initialize counters $c_0 = 0$ for each $j \in X$.
3. For step $t$ that the element $S_t$ arrives
   - For each $j \in X$ if $S_t \geq (1 + e)^{i+1}$, then $c_{i+1} = c_{i+1} + 1 $.
   - If $c_{i+1} \geq (1 + e)^{i+1}$, then $c_{i+1} = c_{i+1}$.
   - Delete the counter $c_i$.
   - Let $i = i + 1$.
   - Initialize new counter $c_{i+1} = 0$.
4. Let $h(V) = (1 + e)^{i+1}$ be the greatest threshold for $j \in X$ such that $c_{i+1} \geq (1 + e)^{i+1}$ and $c_{i+1} < (1 + e)^{i+1}$.
5. Output $h(V)$.

**Proof.** For the sake of simplicity we assume $e^{-1} = (1 + e)^{i+1}$ is a power of $(1 + e)$. Thus, $r = \log_{1+e} e^{-1} = i$. Observe that for the H-index $h^*(V)$ of the vector $V \in \mathbb{N}^n$, there exists a natural number $k$ for which we have $(1 + e)^{k+1} \leq h^*(V) < (1 + e)^{k+1}$. Let us set $k = i + 1 - \ell$, $x = (1 + e)^{i+1}$ and so $\frac{x}{e} = (1 + e)^{i+1}$. Observe that $x \cdot (1 + e)^{-1} = (1 + e)^{i+1} - (1 + e)^{i+1} \leq h^*(V) < (1 + e)^{i+1}$. Indeed, for each number $j \in X$, there exists a counter $c_j$ that
   - is initialized when the counter $c_{i+1}$ is deleted. That in turn, occurs when $c_{i+1} \geq (1 + e)^{i+1}$. Note that the case $j = 1$ is an exception which starts by seeing the first element of the stream $S$.
   - is deleted when for the subsequent counter $c_{i+1}$, we have $c_{i+1} \geq (1 + e)^{i+1}$. Observe that at the same time the new counter $c_{i+1}$ is created and set to zero.

Observe that since $j \in Y$, we must have $j + \ell, j + \ell - 1, j + \ell - 2, \cdots$ and $j + \ell = Y$ are decoupled and zero recursively as follows. Corresponding to each number $j \in Y$, we define the stream $S_j$ which starts when the counter $c_j$ is created and set to zero and ends when the counter $c_j$ is deleted. The stream $S_j$ corresponds to the sub-vector $V_j$ of the vector $V$ including those entries of the vector $V$ that are revealed in the stream $S_j$. In the remainder of the proof, we assume that the H-index of the sub-vector $V_j$ is $h^*(V_j) = h^*(S_j)$.

**Claim 7.** Let $j \in Y$ be an index in $Y$. Let $c_j$ be the counter corresponding to the index $j$. At the time when $c_j$ is deleted, there are at least $(1 + e)^{j+1}$ and at most $(1 + e)^{j+1}$ numbers in the union of the sub-streams $S_k \cup S_j$ and $S_j \cup S_{j+1}$ that are greater than or equal to $(1 + e)^{j+1}$. Moreover,

$$(1 + e)^{j+1} \leq c_{j+1} \leq (1 + e)^{j+2}.$$
That is, 
\[ c_{j+1} \leq \sum_{j-x \leq \ell} T_{j-x} \leq c_{j+1} \cdot (1 + \epsilon(1 + \epsilon)) \leq c_{j+1} \cdot (1 + 2\epsilon(1 + \epsilon)) \leq c_{j+1} \cdot (1 + 3\epsilon). \]

By replacing \( \epsilon \) with \( \epsilon/3 \) we finish the proof of this claim. \( \square \)

Similar to the proof of Claim 7 we can prove the general case.

**Claim 8.** Let \( j \in Y \) be an index in \( Y \). Let \( c_j \) be the counter corresponding to the index \( j \). At the time \( t_{j+\ell} \) of the stream \( S \) when the counter \( c_{j+\ell} \) for \( 0 \leq \ell \leq \xi \) is deleted, there are at least \( (1 + \epsilon)^{1+\ell} \) numbers from the beginning of the stream \( S \) that time that are greater than or equal to \( (1 + \epsilon)^{1+\ell} \). Moreover, \( (1 + \epsilon)^{1+\ell} \leq c_{j+\ell} \leq (1 + \epsilon)^{1+\ell+1} \) of the stream seen so far.

Having the above claims, we observe that at the time when every counter \( c_j \) is deleted, the next counter \( c_{j+1} \) is a \((1+\epsilon)\)-approximation of the H-index of the stream seen so far.

The space of the algorithm is \( \log_{1+\epsilon} e^{-1} \leq 2e^{-1} \log e^{-1} \) words. Since in the proofs of Claims 7 and 8 we need to replace \( \epsilon \) by \( \epsilon/3 \), the space of the algorithm will be \( 6e^{-1} \log 3e^{-1} \) words where each word consists of \( \log n \) bits.

### 3.2 Random Order Stream

In this section we show that if a stream \( S \) of items is uniformly randomly ordered, we can develop an \((1+\epsilon)\)-estimator for the H-index of the stream \( S \) using much smaller space. In particular, if we know a mild lower bound \( O(e^{-3} \log \log(n)) \) for the H-index of the stream, we only need six words (where each word consists of \( \log n \) bits) to \((1+\epsilon)\)-estimate the H-index of the stream. Without having such a mild lower bound, the space of our new streaming algorithm is \( O(e^{-1} \log e^{-1}) \) words where each word consists of \( O(\log \log \log n) \) bits. On the other hand, the streaming algorithm in Theorem 6 uses \( O(e^{-1} \log e^{-1}) \) words where each word in that algorithm consists of \( \log n \) bits. Formally, we state this main result in theorem below.

**Theorem 9 (Random Order Stream).** Let \( S \) be a randomly ordered stream of elements of an underlying vector \( V \in \mathbb{N}^n \) whose H-index is \( h^*(V) \). Let \( 0 < \epsilon, \delta < 1 \) and \( \beta = 150e^{-3} \log \log n \). Then, there exists a one-pass streaming algorithm that reports an estimator \( h(V) \) such that \( \Pr[|h^*(V) - h(V)| \leq \epsilon h^*(V)] \geq 1 - \delta. \)

- **Let** \( \beta/e \leq h^*(V) \) be a lower bound for \( h^*(V) \). Then, the space usage of this algorithm is six words, where a word is a binary vector of length \( \log n \) bits.
- **If** \( h^*(V) \) is upper bounded by \( \beta/e \), then the space usage of this algorithm is \( 6e^{-1} \log 3e^{-1} \) words each of one consists of \( \log(\beta/e) \) bits.

We first give an overview of the algorithm. We runs two subroutines corresponding two different cases for \( h^*(V) \) in parallel. The first case is when we have a lower bound \( \beta/e \) for \( h^*(V) \). The second case is when \( h^*(V) \) is upper bounded by \( \beta/e \). For the latter case we simply run the streaming algorithm of Theorem 5 that needs a space of \( O(e^{-1} \log e^{-1}) \) words where each word consists of \( \log(\beta/e) \) bits. Note that in the latter case the H-index is upper bounded by \( \beta/e \) and hence each word requires \( \log(\beta/e) \) bits.

For the former case, we consider \( \log_{1+\epsilon}(n/\beta) \) guesses for \( h^*(V) \) such that the \( i \)-th guess corresponds to \( n/(1+\epsilon)^i \leq h^*(V) \leq n/(1+\epsilon)^{i+1} \). For the \( i \)-th guess, we look at a window \( W_i \) of the stream \( S \) of size \( \beta(1+\epsilon)^{i+1} \) \((2+\epsilon)\) and count the number of elements of this window that are greater than or equal to \( n/(1+\epsilon)^i \). We should mention that every two windows \( W_{i-1} \) and \( W_i \) corresponding to two consecutive guesses \( i-1 \) and \( i \) overlap. Indeed, let \( Z = \{r_0 = 1, r_1, \ldots, r_i, \ldots, r_{\log_{1+\epsilon}(n/\beta)}\} \), where \( r_i = r_{i-1} + \beta/(1+\epsilon)^i \). The window \( W_i \) contains all the elements of the interval \( [r_{i-1}, r_i] \) of the stream \( S \). Observe that \( W_i \) contains \( \beta(1+\epsilon)^{i+1} + \beta(1+\epsilon)^i = \beta(1+\epsilon)^{i+1} + (2+\epsilon) \) elements of the stream \( S \). Let \( x = \beta(2+\epsilon)/(1+\epsilon) \). Then we accept the \( i \)-th guess and report \( n/(1+\epsilon)^i \) as an \((1+\epsilon)\)-approximation of \( h^*(V) \) if \( (1+\epsilon)x \) elements of this window are greater than or equal to \( n/(1+\epsilon)^i \).

We analyze the algorithm in three steps. Suppose that the \( i \)-th guess is correct, that is, \( n/(1+\epsilon)^i \leq h^*(V) \leq n/(1+\epsilon)^{i+1} \). First, we show that if we ignore the elements in the interval \( [1, r_{i-1}] \) of the stream \( S \) then we lose only \( \epsilon \)-fraction of \( h^*(V) \). Second, we observe that in expectation \( x \) elements of the window \( W_i \) are greater than or equal to \( n/(1+\epsilon)^i \). Finally, we use a Chernoff concentration bound for the hypergeometric distribution (i.e., sampling without replacement) to prove that with high probability (i.e., with probability at least \( 1 - \delta \) for \( 0 < \delta < 1 \)), \((1+\epsilon)x \) elements of the window \( W_i \) are greater than or equal to \( n/(1+\epsilon)^i \).

The Pseudocode of the algorithm is given in below.

**Algorithm 3: Random Order Stream**

**Input:** Stream \( S \) of elements of \( V \).
1. Let \( \beta = 150e^{-3} \log \log n \).
2. Let \( h_1 \) be the output of Algorithm 2 with input \((S, \beta)\).
   (We stop Algorithm 2 if the reported H-index of this algorithm is greater than \( \beta \)).
3. Let \( h_2 \) be the output of Algorithm 4 with input \((S)\).
4. \( h(V) = \max(h_1, h_2) \).

**Algorithm 4: Sampling Without Replacement**

**Input:** Stream \( S \) of elements of \( V \).
1. Let \( \beta = 150e^{-3} \log \log n \), \( x = \beta(2+\epsilon)/(1+\epsilon) \), \( k = 0 \), \( c = c' = 0 \).
2. For \( i = 0, 1, 2, \ldots \), \( [\log_{1+\epsilon}(n/\beta)] \)
3. Let \( r' = r = r + \beta/(1+\epsilon)^i \).
4. While \( r' < k \leq r \)
   5. If \( S_k > n/(1+\epsilon)^i \), then \( c = c + 1 \).
   6. If \( S_k > n/(1+\epsilon)^{i+1} \), then \( c' = c' + 1 \).
   7. Let \( k = k + 1 \).
8. If \( (1-\epsilon)/3 \leq x \leq (1+\epsilon)x \), then
9. \( \text{Output } n/(1+\epsilon)^i \) and quit.
10. Else
11. \( c = c' \) and \( c' = 0 \).
12. Output \( c = 0 \).

Next we prove Theorem 9. First we consider the case that the H-index \( h^*(V) \) is upper bounded by \( \beta/e \). In this case we invoke Algorithm 2 that returns \((1+\epsilon)\)-approximation of \( h^*(V) \) using \( 6e^{-1} \log 3e^{-1} \) words where each word consists of \( \log h^*(V) \leq \log(\beta/e) = 9 \log(e^{-1}) \log \log n \) bits.

Next we consider the case that the H-index \( h^*(V) \) is lower bounded by \( \beta/e \). We prove the following lemma.
**Lemma 10** (H-index is not very small). Let us assume that $h^*(V) \geq \beta/e$. Then, Algorithm 4 approximates the H-index $h^*(V)$ within $(1 \pm \epsilon)$-approximation.

We prove this lemma using a series of lemmas that we prove first. Let $i \in [\log_{1+\epsilon} n]$ be the guess for $h^*(V)$ such that $\frac{n}{1 + \epsilon} \leq h^*(V) \leq \frac{n}{1 - \epsilon}$. The first lemma shows that if we take a random sample subset of size $t_i = \beta(1 + \epsilon)^{-1}(2 + \epsilon)$, the number of elements in this set that are greater than or equal to $\frac{n}{1 + \epsilon}$ is sharply concentrated.

**Lemma 11** (Sampling Concentration Bound). Let $i \in [\log_{1+\epsilon} n]$ be a guess for $h^*(V)$ such that $\frac{n}{1 + \epsilon} \leq h^*(V) \leq \frac{n}{1 - \epsilon}$. Let $T_i$ be a random sample subset of size $t_i = \beta(1 + \epsilon)^{-1}(2 + \epsilon)$ from the indices of the vector $V$ taken uniformly at random without replacement. Then $T_i' = \{j \in T_i : V[j] \geq \frac{n}{1 + \epsilon}\}$ and $c = |T_i'|$.

- Then, for $t_i = \beta(1 + \epsilon)^{-1}(2 + \epsilon)$ we have
  \[\Pr[(1 - \epsilon/3)x \leq c \leq (1 + \epsilon)x] \geq 1 - \delta.\]

- Noisy Sampling: Suppose that due to a noise (e.g., previously incorrect guesses of $h^*(V)$) up to $c$ elements of $V$ that are at least $\frac{n}{1 + \epsilon}$ have been deleted. Then, for $t_i = \beta(1 + \epsilon)^{-1}(2 + \epsilon)$ and the same definition for $T_i'$ and $c$ we have
  \[\Pr[(1 - \epsilon/3)x \leq c \leq (1 + \epsilon)x] \geq 1 - \delta.\]

Proof. First, we prove the first claim. We define a random variable $X_i$ which corresponds to the number of indices $j \in T_i$ for which $V[j] \geq \frac{n}{1 + \epsilon}$. Observe that $X_i$ has the hypergeometric distribution $X_i \sim H(n, h^*(V), t_i)$ with the probability mass function (pmf)

\[\Pr[X_i = k] = \binom{h^*(V)}{k} \binom{n - h^*(V)}{t_i - k} \binom{t_i}{k},\]

and the expectation $E[X_i] = p_i t_i$ where $p_i = h^*(V)/n$.

Then, for $0 < \epsilon < 1$ and $t_i = \frac{4(2/\epsilon)^2}{c^2 p_i}$ we have the following tail bound for $X_i$,

\[\Pr[X_i \leq E[X_i]] \geq \exp(-\epsilon^2 E[X_i]/4) = \delta.\]

Recall that $\frac{1}{1 - \epsilon} \leq p_i = h^*(V)/n \leq \frac{1}{1 + \epsilon}$ which means that $\frac{1}{1 - \epsilon} \leq E[X_i] \leq \frac{1}{1 + \epsilon}$. Thus, $x \leq E[X_i] \leq (1 + \epsilon)x$.

From the above tail bound for $X_i$, with probability at least $1 - \delta$, we have $\Pr[X_i \leq E[X_i]] \leq |X_i| \leq (1 + \epsilon) E[X_i]$. Now we replace $x$ by $e/\epsilon$ to obtain

\[(1 - \epsilon/3)x \leq X_i \leq (1 + \epsilon)^2 x \leq (1 + \epsilon)x.\]

Next, we prove the second claim. Let $V'$ be the new vector after removing up to $c$ elements of $V$ that are at least $\frac{n}{1 + \epsilon}$. Then, since $(1 - \epsilon) h^*(V') \leq h^*(V') \leq h^*(V)$, we have

\[\frac{n}{(1 + \epsilon)^{2}} \leq (1 - \epsilon) \cdot \frac{n}{(1 + \epsilon)} \leq h^*(V') \leq \frac{n}{(1 + \epsilon)^{1-}\epsilon}.\]

Similar to the first claim, since $\frac{1}{1 + \epsilon} \leq p_i = h^*(V)/n \leq \frac{1}{1 - \epsilon}$, we obtain $x \leq E[X_i] \leq (1 + \epsilon)^2 x$.

Recall that with probability at least $1 - \delta$, we have $\Pr[X_i \leq E[X_i]] \leq X_i \leq (1 + \epsilon) E[X_i]$. Now we replace $x$ by $e/\epsilon$ to obtain

\[(1 - \epsilon/3)x \leq X_i \leq (1 + \epsilon)^2 x \leq (1 + \epsilon)x.\]

Next we show that if we ignore the first $t_{i-1}$ elements of the randomly ordered stream $S$, the H-index of the remaining part of the stream is at least $(1 - \epsilon) h^*(V)$.

**Lemma 12** (First Sample is a Tiny Noise). Suppose that $h^*(V) \geq \frac{n}{1 + \epsilon} \geq \frac{1 + \epsilon}{\epsilon^2} \beta$. Let $V'$ be the vector after sampling and removing $t_{i-1}$ elements from $V$ uniformly at random without replacement. Then, for $h^*(V')$ we have $h^*(V') \geq (1 - \epsilon) h^*(V)$.

Proof. We first compute an upper bound on $t_{i-1}$. Recall that $T = \{t_0 = 1, \ldots, t_i, \ldots, t_{[\log_{1+\epsilon} n]}\}$, where $T_i = t_{i-1} + 1$. The window $W_t$ contains all the elements in the interval $[t_1, t_{i-1}]$ of the stream $S$. Recall that $W_t$ contains $\beta(1 + \epsilon)^{-1} + \beta(1 + \epsilon)^{2} = \beta(1 + \epsilon)^{-1} + 2 + \epsilon$ elements of the stream $S$. Also we have $x = \beta(2 + \epsilon)/(1 + \epsilon)$. Thus,

\[t_{i-1} = \beta \sum_{j=0}^{i-1} (1 + \epsilon)^j = \frac{1}{(1 - \epsilon)} = \frac{\beta (1 + \epsilon)}{\epsilon},\]

Thus, if $E[Y_i] \leq h^*(V) - \frac{1}{(1 + \epsilon)^{2}} \beta$ elements of the stream, by the Taylor series $\frac{1}{1 - x} = \sum_{x=0}^{\infty} x^3$ for $x \in (-1, 1)$.

Let us define a random variable $Y_i$ which corresponds to the number of indices $j \in [t_i - 1]$ for which $V[j] \geq \frac{n}{1 + \epsilon}$. Once again, $Y_i$ has the hypergeometric distribution $Y_i \sim H(n, h^*(V), t_i - 1)$. Therefore, similar to the random variable $X_i$ in Lemma 11, we can prove that with probability $1 - \delta$, the random variable $Y_i$ is sharply concentrated around its expectation

\[E[Y_i] = t_{i-1} \cdot h^*(V)/n \leq \frac{(1 + \epsilon)^{2} \beta}{\epsilon} \cdot \frac{1}{(1 + \epsilon)^{-1}},\]

that is, $(1 - \epsilon) E[Y_i] \leq Y_i \leq (1 + \epsilon) E[Y_i]$.

Thus, if $\frac{n}{1 + \epsilon} \leq h^*(V) \leq \frac{1}{1 - \epsilon}$, then we have

\[Y_i \leq (1 + \epsilon) \cdot \frac{\beta (1 + \epsilon)}{\epsilon} \cdot \frac{h^*(V)}{n} \leq \frac{\beta (1 + \epsilon)^{2} \beta}{\epsilon} \cdot \frac{n}{(1 + \epsilon)^{2}} \frac{(1 + \epsilon)^{2} \beta}{\epsilon},\]

Therefore, if we ignore the first $t_{i-1}$ elements of the uniformly ordered stream $S$, in the rest of the stream $S$ we have at least $n(1 + \epsilon)^{2} \beta$ elements that are at least $n(1 + \epsilon)^{2} \beta$.

Recall that $h^*(V) \geq \frac{n}{1 + \epsilon} \geq \frac{1}{(1 + \epsilon)^{2}} \beta$. Let $V'$ be the vector after seeing at most $1/(1 + \epsilon)^{2} \beta$ elements of the stream $S$ that are at least $n/(1 + \epsilon)^{2} \beta$. We then have

\[h^*(V') \geq h^*(V) - \frac{1}{(1 + \epsilon)^{2}} \beta \geq (1 - \epsilon) \cdot \frac{n}{(1 + \epsilon)^{2}} \geq (1 - \epsilon) h^*(V) .\]

Now we apply the union bound to show that all the guesses that are far from $h^*(V)$ fail and a guess which is close to $h^*(V)$ will be well-approximated.

**Lemma 13** (Union Bound). Let $\beta = 36e^{-2 \ln(2/\epsilon^3)}$. Then, with probability at least $1 - \delta$ we have
Theorem 14 (Unbiased Sampling). Let $0 < \epsilon < 1$ be a parameter. Let $V \in \mathbb{N}^n$ be a vector of natural numbers of dimension $n = (1 + \epsilon)^k$ whose $H$-index is $h^*(V)$ where $k \in \mathbb{N}$. Let $S$ be a stream of updates $(i_1, z)$ of the underlying vector $V$. Then, there exists a one-pass randomized streaming algorithm (Algorithm 6) that reports an estimator $\hat{h}(V)$.

- **Multiplicative Error:** Let $h^*(V) \geq \beta$ be a lower bound for $h^*(V)$. If this algorithm uses $x$ instances of $\ell_0$-Sampler, where $x = \frac{\beta}{e^2} \cdot \ln(2/\delta)$, then $(1 - \epsilon)h^*(V) \leq \hat{h}(V) \leq h^*(V)$.

- **Additive Error:** Suppose we do not have a lower bound for $h^*(V)$. If this algorithm uses $x$ instances of $\ell_0$-Sampler where $x = \frac{\beta}{2e} \ln(2/\delta)$, then $h^*(V) - \epsilon \cdot n \leq \hat{h}(V) \leq h^*(V) + \epsilon \cdot n$.

The success probability of this algorithm is $1 - \delta$.

The outline of the algorithm is as follows. We sample entries of the vector $V$ (almost) uniformly at random. Given a stream $S$ of updates to the vector $V$, we keep track of changes to these samples to find an unbiased estimator for the $H$-index $h^*(V)$ whose expectation is $h^*(V)$. We show that with high probability the estimator is sharply concentrated around its expectation $h^*(V)$.

Algorithm 5: Unbiased Sampling

**Input:** Stream $S$ of updates $(i, z)$ to an underlying $n$-dimensional vector $V$ and a lower bound $\beta$ for $h^*(V) \geq \beta$.

1. Let $X$ be the output set of $x$ instances of $\ell_0$-Sampler.
   (X is a uniformly sample set of non-zero indices of $V$.)
2. Let $y$ be an $(1 \pm \epsilon)$-approximation of the number of non-zero indices of $V$ using algorithm of [10].
3. For $i = 0$ to $[\log_{1+\epsilon} n]$
4. Let $R_i = \{ j \in X : V[j] \geq (1 + \epsilon)^i \}$.
5. Let $r_i = |R_i| \cdot y/x$.
6. Let $z$ be the greatest $(1 + \epsilon)^j$ s.t. $r_j \geq (1 + \epsilon)^j (1 - \epsilon)$.
7. Output $z$.

Proof. We first prove the multiplicative error bound. Suppose that $(1 + \epsilon)^i \leq h^*(V) \leq (1 + \epsilon)^{i+1}$. Observe that $z$ is a random variable which depends on random variable $|R_i|$. In the following we prove that $E[|R_i| \cdot x] \geq \frac{x}{\beta}$, what yields $E[z] = (1 + \epsilon)^t$.

Let $X = \{ a_1, \ldots, a_{|R_i|}, a_k \}$ be the set of indices of $x$ instances of $\ell_0$-Sampler return. Here we assume that none of the $a_k$ instances of $\ell_0$-Sampler and the algorithm of [10] fail which can be easily guaranteed by a simple union bound argument. Recall that $R_i = \{ j \in X : V[j] \geq (1 + \epsilon)^i \}$ and $r_i = |R_i| \cdot y/x$. Let us define an indicator random variable $I_j$ corresponding to the $j$-th instance of the $\ell_0$-Sampler which is one if $V[a_j] \geq h^*(V) \geq (1 + \epsilon)^i$ and zero otherwise. Let $I = \sum_{j=1}^{k} I_j$. Recall that $\ell_0$-Sampler takes samples with replacement. Observe that $E[I] = Pr[I = 1] \geq \frac{h^*(V)}{y} \geq
A spectrally. Let $a(|x|) \geq x \cdot \frac{1+e^{i}}{g}$ which means that $E[Z_i] \geq (1 + e^{i})$. We use the Chernoff bound which is stated in below.

**Lemma 15 (Chernoff Inequality).** [8] Let $Z_1, \ldots, Z_m$ denote $m$ independent random variables such that $0 \leq Z_i \leq 1$ for $1 \leq i \leq m$. For $Z = \sum_{i=1}^{m} Z_i$ we get

$$\Pr[|Z - E[Z]| \geq eE[Z] \leq 2e^{2} \left( \frac{e^{2}E[Z]}{3} \right)^{1/3}.$$ 

Using the Chernoff bound for $x = 3e^{2} \log(2/\delta) - \frac{1}{n}$ we have

$$\Pr[|Z - E[Z]| \geq eE[Z] = \Pr[|Z_1 - E[Z_1| \geq e(1 + e^{1})] = \Pr[|Z_1 - E[Z_1| \geq eE[Z_1|] \leq 2e^{2} \left( \frac{e^{2}E[Z_1]}{3} \right)^{1/3} \leq \delta.$$ 

Now we proof the second claim of this theorem. To this end, we use the additive Hoeffding inequality which is given below.

**Lemma 16 (Additive Hoeffding Inequality).** [8] Let $Y_1, \ldots, Y_m$ denote $m$ independent random variables such that $0 \leq Y_i \leq M$ for $1 \leq i \leq n$. For $Y = \sum_{i=1}^{m} Y_i$ we get

$$\Pr[|Y - E[Y]| \geq 1] \leq 2e^{2} \left( \frac{mM^{2}}{3M^{2}} \right).$$ 

Using the additive Hoeffding inequality for $x = 3e^{2} \log(2/\delta)$ and $M = 1$ we have

$$\Pr[|Z - E[Z]| \geq e\eta] = \Pr[|Z_1 - E[Z_1| \geq e\eta] = \Pr[|Z_1 - E[Z_1| \leq 2e^{2} \left( \frac{e^{2}E[Z_1]}{3} \right)^{1/3} \leq \delta,$$

for $x \leq \eta \leq n$ and $0 < \delta < 1$. □

4. **HEAVY HITTERS H-INDICES**

Let $A$ and $P$ be sets of authors and papers, where we assume every author and paper is a number from domains $|A|$ and $|P|$, respectively. Let $a \in A$ be an author for which we show the set of its papers as $Pa \subseteq P$. Let $p \in P$ be a paper for which we show the set of its authors as $Ap \subseteq A$. We represent a paper $p \in P$ by a tuple $(p, a_{p1}, \ldots, a_{pn}, \epsilon_{p})$ for $y \leq x$, where $a_{p1}, \ldots, a_{pn}$ and $\epsilon_{p}$ are the authors and the citation number of the paper $p$, respectively.

Let $S$ be a stream of papers/tuples $(p, a_{p1}, \ldots, a_{pn}, \epsilon_{p})$. Suppose that $h^*(S) = \sum_{a \in S} h^*(a)$ is the sum of H-indices of authors whose papers are given in $S$. Here we first develop a streaming algorithm that distinguishes the following two cases:

1. **1-Heavy Hitter:** The case when $S$ has only one heavy hitter. That is, there exists an author $a_1$ in the stream $S$ whose H-index $h(a_1) \geq (1 - e) \cdot h^*(S)$.

2. **Noisy Stream/Heavy Hitters:** There is no author $a_1$ whose H-index $h(a_1) \geq (1 - e) \cdot h^*(S)$. That includes the cases when either $S$ does not have any heavy hitter or it has more than one heavy hitter. We call the former case noisy stream $S$ and the latter case noisyheavy hitters.

We develop Algorithm 7 to distinguish between these two cases. Later, we use this algorithm as a primitive to develop a streaming algorithm that detects up to $e$ heavy hitters of a stream $S$.

4.1 **1-Heavy Hitter of H-indices**

**Theorem 17 (1-Heavy Hitter of H-indices).** Let $S$ be a stream of papers $(p, a_{p1}, \ldots, a_{pn}, \epsilon_{p})$, where $\{a_{p1}, \ldots, a_{pn}\} \subseteq A$ and $\epsilon_{p}$ are the authors and the citation number of the paper $p \in P$, respectively. Let $0 < e, \delta < 1$. Let $h^*(S) = \sum_{a \in S} h^*(a)$ be the sum of H-indices of authors whose papers are given in $S$. Then, Algorithm 7 uses space $2e^{-2} \log(\log(n)\delta^{-1})$ words and with probability at least $1 - \delta$ distinguishes the following cases.

1. There exists an author $a_1$ in the stream $S$ whose H-index $h(a_1) \geq (1 - e) \cdot h^*(S)$.

2. There is no author $a_1$ whose H-index $h(a_1) \geq (1 - e) \cdot h^*(S)$.

**Algorithm 7: 1-Heavy Hitter**

**Input:** Stream $S$ of tuples/papers $(p, a_{p1}, \ldots, a_{pn}, \epsilon_{p})$, where $a_{p1}, \ldots, a_{pn}$ and $\epsilon_{p}$ are the authors and the citation number of the paper $p$, respectively.

1. Let $s = 2 \log(\log(n)\delta^{-1})$.
2. Initialize counters $c_i = 0$ to zero for $i \in [\log_2(1 + e) \cdot n]$.
3. For each paper $p \in S$ and each $i \in [\log_2(1 + e) \cdot n]$.
4. If $\epsilon_{p} = (1 + e) \cdot i$, then.
5. Let $c_i = c_i + 1$.
6. Let $T_{c_i}$ be a sample of $c_i$ papers ps.t. $\epsilon_{p} = (1 + e) \cdot i$.
7. Let $h(S) = (1 + e) \cdot i$ be the greatest threshold s.t.
8. $c_i \geq (1 + e) \cdot i$ and $c_i+1 < (1 + e) \cdot i+1$.
9. Let $i^*$ be the $i$ for which $h(S) = (1 + e) \cdot i$.
10. If $(1 - e)$ fraction of papers in $T_{c_i}$ has author $a_1$.
11. Return $a_1$ with $h(S)$; otherwise return FAIL.

The algorithm is similar to Algorithm 1. The only difference here is that for every threshold $(1 + e) \cdot i$ in Step 4 we sample a set $T_{c_i}$ of $s = 2 \log(\log(n)\delta^{-1})$ papers. At the end of the stream once we found the greatest threshold for which $c_i \geq (1 + e) \cdot i$ and $c_i+1 < (1 + e) \cdot i+1$, we look at its corresponding sample set $T_{c_i}$ of papers and check if there is an author $a_1$ who is the author of at least $(1 - e)$-fraction of the papers in $T_{c_i}$. If that is the case, we return that author and whose H-index which is $(1 - e)$-approximated by the H-index $h(S)$; otherwise we return the FAIL symbol.

Theorem 17 is proved using Theorem 5 and a simple application of Chernoff bound and the union bound argument for counters $c_i$ where $i \in [\log_2(1 + e) \cdot n]$.

4.2 **Multiple Heavy Hitters of H-indices**

**Theorem 18 (Heavy Hitters of H-indices).** Let $S$ be a stream of papers $(p, a_{p1}, \ldots, a_{pn}, \epsilon_{p})$, where $\{a_{p1}, \ldots, a_{pn}\} \subseteq A$ and $\epsilon_{p}$ are the authors and the citation number of the paper $p \in P$, respectively. Let $0 < e, \delta < 1$. Let $h^*(S) = \sum_{a \in S} h^*(a)$ be the sum of H-indices of authors whose papers are given in $S$. Then, Algorithm 8 with probability at least $1 - \delta$ outputs a set $A' = \{(a_1, h(a_1)), \ldots, (a_r, h(a_r))\}$ of authors for $0 \leq r \leq 1/e$ such that for each author $a_1 \in A'$ and his/her reported H-index $h(a_1)$ we have $h^*(a_1) \geq e \cdot h^*(S)$ and $(1 - e) \cdot h^*(a_1) \leq h(a_1) \leq (1 + e) \cdot h^*(S)$. The space usage of this algorithm is $O(e^{-1} \log(\log(n)\delta^{-1})) \cdot \log(e^{-1}\delta^{-1})$. 


The algorithm is an extension of a known heavy hitter algorithm. Indeed, suppose the distribution of H-indices of authors whose papers are given in a streaming fashion is heavy-tail. We hash the authors using a pair-wise independent hash function into O(ε⁻¹) buckets and treat the sub-stream of papers of those authors that are hashed into every bucket i as a sub-stream $S_i$ of papers of one author for which we run the streaming algorithm of Theorem 5 to find the H-index of the substream $S_i$. We prove that with high probability the authors that have heavy hitter H-indices are hashed into different buckets and the sum of H-indices of non-heavy hitter authors are evenly distributed among the buckets such that each bucket that corresponds to a heavy hitter author receives a noise of at most ε-factor of the H-index of that author. A simple sampling test from each bucket then finds those buckets that correspond to heavy hitter authors. The Pseudocode of the algorithm is given in below and the proof of the correctness of the algorithm is given after it.

**Algorithm 8: Heavy Hitters**

**Input:** Stream $S'$ of tuples/papers $(p_i, a_i^1, \ldots, a_i^n, c_p)\), where $a_i^1, \ldots, a_i^n$ and $c_p$ are the authors and the citation number of the paper $p$, respectively.

1. Let $H_j : [A] \rightarrow [\ell = 2e^{-2}]$ for $j \in [x = \log(\epsilon^{-1} \ell^{-1})]$ be an independently sample function from a set of pair-wise independent hash functions.
2. Let $B[x,t]$ be a $x \times t$ array buckets.
3. For every paper $(p_i, a_i^1, \ldots, a_i^n, c_p) \in S$
4. For each author $a_i^j$ in the paper $p_i$ and each $j \in [x]$
5. Hash $(p_i, a_i^1, \ldots, a_i^n, c_p)$ into the $B[j, H_j(a_i^j)]$.
6. For $j \in [x]$ and $k \in [t]$.
7. Let $S_{i,j,k}$ be papers hashed into bucket $B[j, k]$.
8. Invoke Algorithm 1-Heavy Hitter with input $S_{i,j,k}$.
9. If the output of Step 7 is an author $a_i$ with $H(a_i)$
10. Output $a_i$ and H-index $H(a_i)$ as a heavy hitter.

**Proof:** Recall that $A = \{a_1, \ldots, a_n\}$ is the set of authors. We say an author $a_i \in A$ is a heavy hitter author if for the H-index of each author $a_i \in A$ we have

$$h^*(a_i) \geq e \cdot h^*(A) = e \cdot \sum_{a_i \in A} h^*(a_i)$$

Without lost of generality suppose that the first $r$ authors in $A$ for $r \leq 1/\epsilon$ are the heavy hitter authors, that is, $X = \{a_1, \ldots, a_r\} \subseteq A$ is the set of heavy hitter authors. Let us fix a heavy hitter author $a_i \in X$. Let $I_{i,j,k}$ be an indicator random variable which corresponds to the event that another author $a_k \in A$ for $a_k \neq a_i$ (including other heavy hitter authors) is hashed into the same bucket as $B[j, H_j(a_i)]$. Observe that since $H_j$ is a pairwise independent hash function we have $E[I_{i,j,k}] = \Pr(H_j(a_k) = H_j(a_i)) \leq \frac{1}{\ell}$. We define a random variable $J_{i,j} = \sum_{k=1, k \neq i}^{\ell} I_{i,j,k} \cdot h^*(a_k)$ for the H-index noise of the bucket $B[j, H_j(a_i)]$. Observe that by linearity of expectation we obtain

$$E[J_{i,j}] = \sum_{k=1, k \neq i}^{\ell} h^*(a_k) \cdot E[I_{i,j,k}] \leq \frac{1}{\ell} \cdot (h^*(A) - h^*(a_i))$$

Now we use the Markov inequality

$$\Pr(h^*(B[H_j(a_i)]) \geq (1 + e)h^*(a_i)) \leq \Pr(J_{i,j} \geq e^2h^*(A)) = \Pr(J_{i,j} \geq eE[J_{i,j}]) \leq e$$

for $t = \frac{1}{\ell}$. We say an event $BAD_j$ occurs if for each hash function $H_j$ for $j \in [x]$, the H-index of the bucket $B[H_j(a_i)]$ is more than $1 + e$ that is, $h^*(B[H_j(a_i)]) \geq (1 + e)h^*(a_i)$. Therefore, for $x = \log(\frac{1}{\epsilon \ell})$ hash functions $H_1, \ldots, H_x$, $\Pr(BAD_j) \leq e^x = e\delta$.

Observe that there can be at most $1/\epsilon$ heavy hitter authors (i.e., $|X| \leq 1/\epsilon$). We define an event $BAD$ if one of bad events $BAD_i$ for $i \in [r]$ occurs. Thus, using the union bound we obtain $\Pr(BAD) \leq \sum_{i \in [r]} \Pr(BAD_i) \leq r \cdot e\delta \leq \delta$.

\[\square\]

5. EXTENSIONS AND CONCLUDING REMARKS

We have initiated the study of streaming algorithms for problems that estimate impact or identify impactful users, and focused on the H-index of users. There are many variations of the H-index, including those that take publication dates, response timestamps, respondent’s H-index, and so on into account; there are also variations based on different functions of the number of responses with respect to the number of publications like $k$ publications with a total of $k^2$ responses each; heavy hitters that are users with H-index that are large in the count or in square of the counts (so called L2 heavy hitters); etc. While our techniques will extend naturally to some of these problems, there is much that remains to be done in obtaining the best bounds for these streaming problems.

Acknowledgements

This work was supported by the NSF IIS 1247696, NSF CCF 1535878, NSF IIS 1447793 and NSF CCF 1161151.

6. REFERENCES


