Distinct Elements in Streams: An Algorithm for the (Text) Book[*]

Sourav Chakraborty ⓡ
Indian Statistical Institute, India

N. V. Vinodchandran ⓡ
University of Nebraska-Linclon, USA

Kuldeep S. Meel
National University of Singapore, Singapore

Abstract

Given a data stream $\mathcal{A} = \langle a_1, a_2, \ldots, a_m \rangle$ of m elements where each $a_i \in [n]$, the Distinct Elements problem is to estimate the number of distinct elements in $\mathcal{A}$. Distinct Elements has been a subject of theoretical and empirical investigations over the past four decades resulting in space optimal algorithms for it. All the current state-of-the-art algorithms are, however, beyond the reach of an undergraduate textbook owing to their reliance on the usage of notions such as pairwise independence and universal hash functions. We present a simple, intuitive, sampling-based space-efficient algorithm whose description and the proof are accessible to undergraduates with the knowledge of basic probability theory.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases Distinct Elements Estimation, Streaming, Sampling

1 Introduction

We consider the fundamental problem of estimating the number of distinct elements in a data stream (Distinct Elements problem or the F_0 estimation problem). For a data stream $\mathcal{A} = \langle a_1, a_2, \ldots, a_m \rangle$, where each $a_i \in [n]$, $\mathsf{F}_0(\mathcal{A})$ is the number of distinct elements in $\mathcal{A}$: $\mathsf{F}_0(\mathcal{A}) = |\{a_1, a_2, \ldots, a_m\}|$. Since $\mathcal{A}$ is clear from the context, we use F_0 to refer to $\mathsf{F}_0(\mathcal{A})$.

▶ **Problem 1.** *Given a stream $\mathcal{A} = \langle a_1, a_2, \ldots, a_m \rangle$ of m elements where each $a_i \in [n]$, parameters ε, δ, output an (ε, δ)-approximation of $\mathsf{F}_0(\mathcal{A})$. That is, output c such that*

$$\Pr[(1-\varepsilon) \cdot \mathsf{F}_0(\mathcal{A}) \leq c \leq (1+\varepsilon) \cdot \mathsf{F}_0(\mathcal{A})] \geq 1 - \delta.$$

We are interested in streaming algorithms that uses $\mathsf{poly}(\log m, \log n, \frac{1}{\varepsilon}, \log \frac{1}{\delta})$ bits of memory wherein we use log to denote $\log_2$. The seminal work of Flajolet and Martin [9] provided the first algorithm assuming the existence of hash functions with full independence. Subsequent investigations relying on the usage of limited-independence hash functions have led to design of algorithms with optimal space complexity $O(\log n + \frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta})$. We defer detailed bibliographical remarks to Section 3. However, all the current space-efficient algorithms are beyond the reach of an undergraduate textbook due to their reliance on notions such as pairwise independence and universal hash functions.

[*] The earlier version of the paper, as published at ESA 2022, contained error in the proof of Claim 4. The current revised version fixes the error in the proof. The authors decided to forgo the old convention of alphabetical ordering of authors in favor of a randomized ordering, denoted by ⓡ. The publicly verifiable record of the randomization is available at `https://www.aeaweb.org/journals/policies/random-author-order/search`

We present a very simple algorithm for the F_0 estimation problem using a sampling strategy that only relies on basic probability for its analysis. In particular, it does not use universal hash functions. We believe that only using basic probability theory for the analysis makes the algorithm presentable to undergraduates right after the introduction of basic tail bounds. In addition, the simplicity of the code makes it appealing to be used in practical implementations. Our algorithm builds and refines ideas introduced in the recent work on estimating the size of the union of sets in the general setting of *Delphic sets* [13].

2 A Simple Algorithm

Algorithm 1 F_0-Estimator

```
Input Stream $\mathcal{A} = \langle a_1, a_2, \ldots, a_m \rangle$, $\varepsilon$, $\delta$
Initialize $p \leftarrow 1$; $\mathcal{X} \leftarrow \emptyset$; thresh $\leftarrow \lceil \frac{12}{\varepsilon^2} \log(\frac{8m}{\delta}) \rceil$
for $i = 1$ to $m$ do
    $\mathcal{X} \leftarrow \mathcal{X} \setminus \{a_i\}$
    With probability $p$, $\mathcal{X} \leftarrow \mathcal{X} \cup \{a_i\}$
    if $|\mathcal{X}| =$ thresh then
        Throw away each element of $\mathcal{X}$ with probability $\frac{1}{2}$
        $p \leftarrow \frac{p}{2}$
        if $|\mathcal{X}| =$ thresh then Output $\perp$
Output $\frac{|\mathcal{X}|}{p}$
```

The algorithm F_0-Estimator uses a simple sampling strategy. In order to keep the set of samples small, it makes sure that $\mathcal{X}$ does not grow beyond the value thresh by adjusting the sampling rate p accordingly. After all the elements of the stream are processed, it outputs $\frac{|\mathcal{X}|}{p}$ where p is the final sampling rate[1].

2.1 Theoretical Analysis

We present the theoretical analysis entirely based on first principles, which adds to its length. For readers who are familiar with randomized algorithms, the proof is standard.

We state the following well-known concentration bound, Chernoff bound, for completeness.

▶ **Fact 2.1** (Chernoff's Bound). *Let* $\mathsf{v}_1, \ldots, \mathsf{v}_k$ *be independent random variables taking values in* $\{0, 1\}$. *Let* $\mathsf{V} = \sum_{i=1}^{k} \mathsf{v}_i$ *and* $\mu = \mathbb{E}[\mathsf{V}]$. *Then, for* $\beta > 0$, $\Pr(|\mathsf{V} - \mu| \geq \beta\mu) \leq 2e^{-\frac{\beta^2 \mu}{2+\beta}}$

The following theorem captures the correctness and space complexity of F_0-Estimator.

▶ **Theorem 2.** *For any data stream* $\mathcal{A}$ *and any* $0 < \varepsilon, \delta < 1$, *the algorithm* F_0-Estimator *outputs an* (ε, δ)*-approximation of* $\mathsf{F}_0(\mathcal{A})$. *The algorithm uses* $O(\frac{1}{\varepsilon^2} \cdot \log n \cdot (\log m + \log \frac{1}{\delta}))$ *space in the worst case.*

Proof. The stated space complexity bound of the algorithm follows because, from the description, it is clear that the size of the set of samples kept by the algorithm is always $\leq$ thresh, and each item requires $\lceil \log_2 n \rceil$ bits to store.

[1] In an earlier version, it was wrongly claimed that every element seen so far is independently in $\mathcal{X}$ with equal probability p. That claim was erroneous but, fortunately, not used in the analysis. It is also worth remarking that $\frac{|\mathcal{X}|}{p}$ is not an unbiased estimator of F_0.

We give a formal proof of correctness below. Consider the following two events:

Error : 'The algorithm $\mathsf{F_0}$-$\mathsf{Estimator}$ does not return a value in the range $[(1-\varepsilon)\mathsf{F_0}, (1+\varepsilon)\mathsf{F_0}]$'
Fail : 'The algorithm $\mathsf{F_0}$-$\mathsf{Estimator}$ outputs $\bot$.'

We will bound $\Pr[\mathsf{Error}]$ by δ. Observe that $\Pr[\mathsf{Error}] \leq \Pr[\mathsf{Fail}] + \Pr[\mathsf{Error} \cap \overline{\mathsf{Fail}}]$. Theorem follows from Claim 3 and Claim 4. ◀

▷ Claim 3. $\Pr[\mathsf{Fail}] \leq \frac{\delta}{8}$

Proof. Let Fail_j denote the event that Algorithm 1 returns $\bot$ when $i = j$. Formally, Fail_j: '$|\mathcal{X}| = \mathsf{thresh}$ and none of the elements of $\mathcal{X}$ are thrown away at line 6 for $i = j$'. The probability that Fail_j happens is $\left(\frac{1}{2}\right)^{\mathsf{thresh}}$. Therefore,

$$\Pr[\mathsf{Fail}] \leq \sum_{j=1}^{m} \Pr[\mathsf{Fail}_j] \leq m \cdot \left(\frac{1}{2}\right)^{\mathsf{thresh}} \leq \frac{\delta}{8}$$ ◀

▷ Claim 4. $\Pr[\mathsf{Error} \cap \overline{\mathsf{Fail}}] \leq \frac{\delta}{2}$.

Proof of Claim 4

To bound $\Pr[\mathsf{Error} \cap \overline{\mathsf{Fail}}]$, we consider a relaxed version of Algorithm $\mathsf{F_0}$-$\mathsf{Estimator}$, which is stated as Algorithm 2. Algorithm 2 is nothing but $\mathsf{F_0}$-$\mathsf{Estimator}$ with line 8 removed. Observe that for a given input, the algorithm $\mathsf{F_0}$-$\mathsf{Estimator}$ behaves identically to Algorithm 2 as long as $|\mathcal{X}| < \mathsf{thresh}$ after each element of $\mathcal{X}$ is thrown away with probability $\frac{1}{2}$ (i.e., the event Fail does not happen). Now, we consider the following event:

Error_2 : 'The Algorithm 2 does not output a value in the range $[(1-\varepsilon)\mathsf{F_0}, (1+\varepsilon)\mathsf{F_0}]$.'

Observe that $\Pr[\mathsf{Error} \cap \overline{\mathsf{Fail}}] \leq \Pr[\mathsf{Error}_2]$. In Claim 6, we obtain the desired bound on $\Pr[\mathsf{Error}_2]$ and hence on $\Pr[\mathsf{Error} \cap \overline{\mathsf{Fail}}]$.

Algorithm 2

```
   Input Stream $\mathcal{A} = \langle a_1, a_2, \ldots, a_m \rangle$, $\varepsilon$, $\delta$
1: Initialize $p \leftarrow 1$; $\mathcal{X} \leftarrow \emptyset$; $\mathsf{thresh} \leftarrow \lceil \frac{12}{\varepsilon^2} \log(\frac{8m}{\delta}) \rceil$
2: for $i = 1$ to $m$ do
3:     $\mathcal{X} \leftarrow \mathcal{X} \setminus \{a_i\}$
4:     With probability $p$, $\mathcal{X} \leftarrow \mathcal{X} \cup \{a_i\}$
5:     if $|\mathcal{X}| = \mathsf{thresh}$ then
6:         Throw away each element of $\mathcal{X}$ with probability $\frac{1}{2}$
7:         $p \leftarrow \frac{p}{2}$
8: Output $\frac{|\mathcal{X}|}{p}$
```

To prove an upper bound on $\Pr[\mathsf{Error}_2]$ in Claim 6, we will consider the Algorithm 3. Algorithm 3 uses two subroutines: (1) $\mathsf{GenerateRandomBits}$ that returns a randomly generated array of $m+1$ bits, and (2) $\mathsf{FirstZeroIndex}$ returns the minimum of first index of array equal to 0 and one plus the size of array; we assume array is zero-indexed. In the following, we use S_i to denote $\{a_1, a_2, \ldots, a_i\}$ – distinct elements that appear in the first i items in the stream.

▷ Claim 5. The following property holds true in line 7 of Algorithm 3

Every element in S_i is in $\mathcal{Y}_k$ independently with probability 2^{-k}.

Algorithm 3

```
   Input Stream $\mathcal{A} = \langle a_1, a_2, \ldots, a_m \rangle$
1: for $k = 0$ to $m$ do
2:     Initialize $\mathcal{Y}_k \leftarrow \emptyset$;
3: for $i = 1$ to $m$ do
4:     $r \leftarrow \mathsf{GenerateRandomBits}(m+1)$
5:     for $k = 0$ to $m$ do
6:         $\mathcal{Y}_k \leftarrow \mathcal{Y}_k \setminus \{a_i\}$
7:         if $k \leq \mathsf{FirstZeroIndex}(r)$ then $\mathcal{Y}_k \leftarrow \mathcal{Y}_k \cup \{a_i\}$
```

Proof. The proof proceeds via induction on i.

Base Case: $\Pr[a_1 \in \mathcal{Y}_k] = \Pr[k \leq \mathsf{FirstZeroIndex}(r)] = 1/2^k$.
Inductive Step: Note that, by induction hypothesis, for all $a_\ell \in S_i \setminus a_i$, we have $a_\ell \in \mathcal{Y}_k$ independently with probability $1/2^k$. Now, for a_i, we have $a_i \notin \mathcal{Y}_k$ in line 6. In line 7, $\Pr[a_i \in \mathcal{Y}_k] = \Pr[k \leq \mathsf{FirstZeroIndex}(r)] = 1/2^k$ ◀

Let us use the random variables p_j and X_j to denote the value of p and the set $\mathcal{X}$ at the end of the loop iteration with $i = j$ (in Algorithm 2). Similarly, we will use the random variable $\mathsf{Y}_{k,j}$ to indicate the set $\mathcal{Y}_k$ at the end of loop iteration with $i = j$ (in Algorithm 3).

We can view Algorithm 2 as updating value of p and $\mathcal{X}$ as the elements of stream $\mathcal{A}$ are processed such that we have $(\mathsf{p}_j, \mathsf{X}_j) = (\mathsf{p}_j, \mathsf{Y}_{k,j})$ where $\mathsf{p}_j = 2^{-k}$. It is perhaps worth observing that the value of p in Algorithm 2 is always at least 2^{-m}, which is why we have k iterate over $[0, m]$ in Algorithm 3.

For any $j \in [1, m]$ and $k \in [0, m]$ and $a \in S_j$ let $\mathsf{r}^a_{k,j}$ denote the indicator random variable indicating whether a is in the set $\mathcal{Y}_k$ in line 7 for $i = j$ (in Algorithm 3). By Claim 5, the random variables $\{\mathsf{r}^a_{k,j}\}_{a \in S_j}$ are independent and for all $a \in S_j$ $\Pr[\mathsf{r}^a_{k,j} = 1] = 2^{-k}$.

$$\mathbb{E}\left[|\mathsf{Y}_{k,j}|\right] = \mathbb{E}\left[\sum_{a \in S_j} \mathsf{r}^a_{k,j}\right] = \sum_{a \in S_j} \Pr[\mathsf{r}^a_{k,j} = 1] = 2^{-k} \cdot |S_j| \leq 2^{-k} \cdot \mathsf{F}_0. \tag{1}$$

▷ Claim 6. $\Pr[\mathsf{Error}_2] \leq \frac{\delta}{2}$

Proof. We decompose $\Pr[\mathsf{Error}_2]$ based on the value of p at the end of Algorithm 2. To this end, let us define the event Bad_2: "The value of p at line 8 in Algorithm 2 is less than $\frac{\mathsf{thresh}}{4\mathsf{F}_0}$."

Let $\ell = \lfloor \log(\frac{4\mathsf{F}_0}{\mathsf{thresh}}) \rfloor$. Since every value of p can be expressed as power of 2, we have that $p < 2^{-\ell}$ if and only if $p < \frac{\mathsf{thresh}}{4\mathsf{F}_0}$. Observe that $\Pr[\mathsf{Error}_2] \leq \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Error}_2 \cap \overline{\mathsf{Bad}_2}]$.

Bounding $\Pr[\mathsf{Bad}_2]$

For $j \in [1, m]$, let $\mathsf{Bad}_{2,j}$ denote the event that 'jth iteration of the **for** loop in Algorithm 2 is the first iteration where the value of p goes below $2^{-\ell}$' i.e., $\mathsf{p}_{j-1} = 2^{-\ell}$ and $\mathsf{p}_j = 2^{-(\ell+1)}$. Therefore, $\Pr[\mathsf{Bad}_2] = \sum_{j=1}^m \Pr[\mathsf{Bad}_{2,j}]$. We will now compute $\Pr[\mathsf{Bad}_{2,j}]$ for a fixed j.

By definition of $\mathsf{Bad}_{2,j}$, we have $|\mathcal{X}| = \mathsf{thresh}$ and $p = 2^{-\ell}$ in line 5 of Algorithm 2 for $i = j$, i.e., $|\mathsf{Y}_{\ell,j}| = \mathsf{thresh}$. From Equation 1 we have $\mathbb{E}\left[|\mathsf{Y}_{\ell,j}|\right] \leq 2^{\ell} \cdot \mathsf{F}_0 \leq \frac{\mathsf{thresh}}{4}$. Thus, $\Pr[\mathsf{Bad}_{2,j}] \leq \Pr[|\mathsf{Y}_{\ell,j}| \geq \mathsf{thresh}] \leq 2e^{-\frac{9 \cdot \mathsf{thresh}}{20}} \leq \frac{\delta}{4m}$, where the second inequality follows from Chernoff Bound. Therefore, $\Pr[\mathsf{Bad}_2] \leq \frac{\delta}{4}$.

Bounding $\Pr[\mathsf{Error}_2 \cap \overline{\mathsf{Bad}_2}]$

Let us define the event $\mathsf{Error}_{2,q}$: ‘$\mathsf{p}_m = 2^{-q}$ and $\frac{|\mathsf{X}_m|}{2^{-q}} \notin [(1-\epsilon)\mathsf{F}_0, (1+\epsilon)\mathsf{F}_0]$ ’
Observe that $\Pr[\mathsf{Error}_{2,q}] \le \Pr[|\mathsf{Y}_{q,m}| \notin [(1-\epsilon)\cdot\frac{\mathsf{F}_0}{2^q}, (1+\epsilon)\cdot\frac{\mathsf{F}_0}{2^q}]]$.

$$
\begin{aligned}
\text{Therefore, } \Pr[\mathsf{Error}_2 \cap \overline{\mathsf{Bad}_2}] &\le \sum_{q=0}^{\ell} \Pr[\mathsf{Error}_{2,q}] \\
&\le \sum_{q=0}^{\ell} \Pr[|\mathsf{Y}_{q,m}| \notin [(1-\epsilon)\cdot\frac{\mathsf{F}_0}{2^q}, (1+\epsilon)\cdot\frac{\mathsf{F}_0}{2^q}]] \\
&\le \sum_{q=0}^{\ell} 2e^{-\frac{\varepsilon^2 \mathsf{F}_0}{3\cdot 2^q}} \qquad \text{[Using Equation 1 and Chernoff bound]} \\
&\le 4e^{-\frac{\varepsilon^2 \mathsf{F}_0}{3\cdot 2^\ell}} \le 4e^{-\frac{\varepsilon^2 \mathsf{thresh}}{12}} \le 4\cdot\left(\frac{\delta}{8}\right)^{\log e} \le \frac{\delta}{4} \qquad \blacktriangleleft
\end{aligned}
$$

3 Bibliographic Remarks

Distinct Elements problem (or F_0 estimation problem) is one of the most investigated problem in the data streaming model [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. While the Distinct Elements problem has a wide range of applications in several areas of computing, it was first investigated in the algorithms community by Flajolet and Martin [9]. They provided the first approximation under the assumption of the existence of hash functions with full independence. The seminal work of Alon, Matias, and Szegedy [1] that introduced the data streaming model of computation revisited this problem as a special case of F_k estimation problem and achieved space complexity of $O(\log n)$ for $\varepsilon > 1$ and constant δ. The first (ε, δ) approximation for Distinct Elements problem was Gibbson and Tirthpura who achieved $O(\frac{\log n}{\varepsilon^2})$ space complexity [10]. Bar-Yossef, Jayram, Kumar, Sivakumar and Trevisan improved the space complexity bound to $\tilde{O}(\log n + 1/\varepsilon^2)$ [2]. Subsequently, Kane, Nelson, and Woodruff achieved $O(\log n + 1/\varepsilon^2)$ which is optimal in n and ε [12]. All the above bounds are for a fixed confidence parameter δ, which can be amplified to achieve confidence bounds for arbitrary δ by simply running $\log(\frac{1}{\delta})$-estimators in parallel and returning the median. This incurs a multiplicative factor of $\log(\frac{1}{\delta})$. Błasiok designed an (ε, δ) approximation algorithm for F_0 estimation problem with space complexity of $O(\frac{1}{\varepsilon^2}\cdot\log\frac{1}{\delta} + \log n)$, thereby matching the lower bound in all the three parameters n, ε and δ [4]. As is expected, every subsequent improvement added to the complexity of the algorithm or the analysis, and a majority of these work remain beyond the reach of non-experts. A crucial technical ingredient for all the works mentioned above is their careful usage of limited-independence hash functions in order to make space $\mathsf{poly}(\log n)$. Monte Carlo-based approaches have been utilized in the context of size estimation of the union of sets, but their straightforward adaptation to the streaming setting did not seem to yield progress. Recently, a new sampling-based approach was proposed in the context of estimating the size of the union of sets in the streaming model that achieves space complexity with $\log m$-dependence [13]. The algorithm we presented adapts ideas from this work to the context of F_0 estimation.

Acknowledgments

We are deeply grateful to Donald E. Knuth for his thorough review, which not only enhanced the quality of this paper (including fixing several errors) but has also inspired us for higher standards. We owe sincere thanks to Arijit Ghosh, Mridul Nandi, Soumit Pal, Uddalok Sarkar, and anonymous reviewers for careful reading and pointing out critical errors in earlier versions. This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme (NRF-NRFFAI1-2019-0004), Singapore Ministry of Education Academic Research Fund Tier 1, Ministry of Education Singapore Tier 2 grant (MOE-T2EP20121-0011), and Amazon Faculty Research Awards. Vinod was supported in part by NSF CCF-2130608 and NSF HDR:TRIPODS-1934884 awards.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. `doi:10.1006/jcss.1997.1545`.

2 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Proc. of RANDOM*, pages 1–10, 2002. `doi:10.1007/3-540-45726-7\_1`.

3 Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of SODA*, pages 623–632, 2002. URL: `http://dl.acm.org/citation.cfm?id=545381.545464`.

4 Jaroslaw Blasiok. Optimal streaming and tracking distinct elements with high probability. In *Proc. of SODA*, 2018. `doi:10.1137/1.9781611975031.156`.

5 Joshua Brody and Amit Chakrabarti. A multi-round communication lower bound for gap hamming and some consequences. In *Proc. of CCC*, pages 358–368. IEEE Computer Society, 2009. `doi:10.1109/CCC.2009.31`.

6 Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities (extended abstract). In *Proc. of ESA*, pages 605–617, 2003. `doi:10.1007/978-3-540-39658-1\_55`.

7 Cristian Estan, George Varghese, and Michael E. Fisk. Bitmap algorithms for counting active flows on high-speed links. *IEEE/ACM Trans. Netw.*, 14(5):925–937, 2006. `doi:10.1145/1217709`.

8 Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156, 2007. URL: `https://doi.org/10.46298/dmtcs.3545`.

9 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985. `doi:10.1016/0022-0000(85)90041-8`.

10 Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proc. of SPAA*, pages 281–291, 2001. `doi:10.1145/378580.378687`.

11 Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. In *Proc. of FOCS*, pages 283–288, 2003. `doi:10.1109/SFCS.2003.1238202`.

12 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proc. of PODS*, pages 41–52, 2010. `doi:10.1145/1807085.1807094`.

13 Kuldeep S. Meel, N. V. Vinodchandran, and Sourav Chakraborty. Estimating the size of union of sets in streaming models. In *Proc. of PODS*, pages 126–137, 2021. `doi:10.1145/3452021.3458333`.