An Embarrassingly Parallel Optimal-Space Cardinality Estimation Algorithm

Emin Karayel
Department of Computer Science, Technische Universität München, Germany

Abstract

In 2020 Błasiok (ACM Trans. Algorithms 16(2) 3:1-3:28) constructed an optimal space streaming algorithm for the cardinality estimation problem with the space complexity of $\mathcal{O}(\varepsilon^{-2}\ln(\delta^{-1}) + \ln n)$ where ε, δ and n denote the relative accuracy, failure probability and universe size, respectively. However, his solution requires the stream to be processed sequentially. On the other hand, there are algorithms that admit a merge operation; they can be used in a distributed setting, allowing parallel processing of sections of the stream, and are highly relevant for large-scale distributed applications. The best-known such algorithm, unfortunately, has a space complexity exceeding $\Omega(\ln(\delta^{-1})(\varepsilon^{-2}\ln\ln n + \ln n))$. This work presents a new algorithm that improves on the solution by Błasiok, preserving its space complexity, but with the benefit that it admits such a merge operation, thus providing an optimal solution for the problem for both sequential and parallel applications. Orthogonally, the new algorithm also improves algorithmically on Błasiok's solution (even in the sequential setting) by reducing its implementation complexity and requiring fewer distinct pseudo-random objects.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Theory of computation → Pseudorandomness and derandomization; Theory of computation → Distributed algorithms; Theory of computation → Sketching and sampling

Keywords and phrases Distinct Elements, Distributed Algorithms, Randomized Algorithms, Expander Graphs, Derandomization, Sketching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.35

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2307.00985` [26]

Supplementary Material
Software: `https://isa-afp.org/entries/Distributed_Distinct_Elements.html` [25]
Software: `https://isa-afp.org/entries/Expander_Graphs.html` [27]

Acknowledgements Thanks to the anonymous reviewers for their careful review and helpful comments and to Tobias Nipkow for many helpful discussions.

1 Introduction

In 1985 Flajolet and Martin [14] introduced a space-efficient streaming algorithm for the estimation of the count of distinct elements in a stream $a_1, ..., a_m$ whose elements are from a finite universe U. Their algorithm does not modify the stream, observes each stream element exactly once and its internal state requires space logarithmic in $n = |U|$. However, their solution relies on the model assumption that a given hash function can be treated like a random function selected uniformly from the family of all functions with a fixed domain and range. Despite the ad-hoc assumption, their work spurred a large number of publications[1], improving the space efficiency and runtime of the algorithm. In 1999 Alon et al. [4] identified

[1] Pettie and Wang [33, Table 1] summarized a comprehensive list.

a solution that avoids the ad-hoc model assumption. They use 2-independent families of hash functions, which can be seeded by a logarithmic number of random bits in $|U|$ while retaining a restricted set of randomness properties. Their refined solution was the first rigorous Monte-Carlo algorithm for the problem. Building on their work, Bar-Yossef et al. in 2002 [6], then Kane et al. in 2010 [24] and lastly, Błasiok in 2020 [9][2] developed successively better algorithms achieving a space complexity of $\mathcal{O}(\varepsilon^{-2}\ln(\delta^{-1}) + \ln n)$, which is known to be optimal [23, Theorem 4.4].

Table 1 Important cardinality estimation algorithms.

Year, Author	Space Complexity	Merge
1981, Flajolet and Martin	$\mathcal{O}(\varepsilon^{-2}\ln n)$ for constant δ [a)]	Yes
1999, Alon et al.	$\mathcal{O}(\ln\ln n)$ for $\delta = 2(\varepsilon+1)^{-1}$	Yes
2002, Bar-Yossef et al.[b)]	$\mathcal{O}(\ln(\delta^{-1})(\varepsilon^{-2}\ln\ln n + \mathrm{poly}(\ln(\varepsilon^{-1}), \ln\ln n)\ln n))$ [c)]	Yes
2010, Kane et al.	$\mathcal{O}(\ln(\delta^{-1})(\varepsilon^{-2} + \ln n))$	No
2020, Błasiok	$\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$	No
This work	$\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$	Yes

a) Random oracle model.
b) Algorithm 2 from the publication.
c) The notation $\mathrm{poly}(a, b)$ stands for a term polynomial in a and b.

These algorithms return an approximation Y of the number of distinct elements $|A|$ (for $A := \{a_1, \ldots, a_m\}$) with relative error ε and success probability $1 - \delta$, i.e.:

$$\mathcal{P}(|Y - |A|| \leq \varepsilon\,|A|) \geq 1 - \delta$$

where the probability is only over the internal random coin flips of the algorithm but holds for all inputs.

Unmentioned in the source material is the fact that it is possible to run the older algorithms by Alon et al. and Bar-Yossef et al. in a parallel mode of operation. This is due to the fact that the algorithms make the random coin flips only in a first initialization step, proceeding deterministically afterwards and that the processing step for the stream elements is commutative. For example, if two runs for sequences a and b of the algorithm had been started with the same coin flips, then it is possible to introduce a new operation that merges the final states of the two runs and computes the state that the algorithm would have reached if it had processed the concatenation of the sequences a and b sequentially.

This enables processing a large stream using multiple processes in parallel. The processes have to communicate at the beginning and at the end to compute an estimate. The communication at the beginning is to share random bits, and the communication at the end is to merge the states. Because there is no need for communication in between, the speed-up is optimal with respect to the number of processes, such algorithms are also called embarrassingly parallel [15, Part 1]. This mode of operation has been called the distributed streams model by Gibbons and Tirthaputra [17]. Besides the distributed streams model, such a merge operation allows even more varied use cases, for example, during query processing in a Map-Reduce pipeline [11]. Figure 1 illustrates two possible modes of operation (among many) enabled by a merge function.

[2] An earlier version of Błasiok's work was presented in the ACM-SIAM Symposium on Discrete Algorithms in 2018. [8]

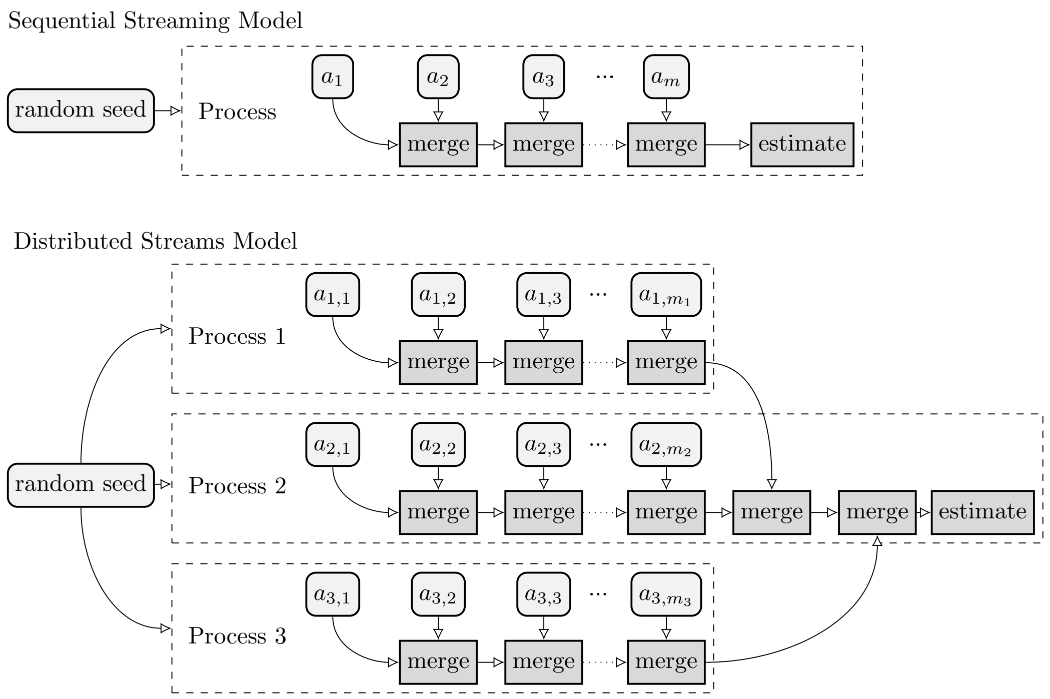

Figure 1 Example use cases for cardinality estimation algorithms that support merge.

However, an extension with such a merge operation is not possible for the improved algorithms by Kane et al. and Błasiok. This is because part of their correctness proof relies inherently on the assumption of sequential execution, in particular, that the sequence of states is monotonically increasing, which is only valid in the sequential case. This work introduces a new distributed cardinality estimation algorithm which supports a merge operation with the same per-process space usage as the optimal sequential algorithm by Błasiok: $\mathcal{O}(\varepsilon^{-2}\ln(\delta^{-1}) + \ln n)$. Thus the algorithm in this work has the best possible space complexity in *both* the sequential and distributed streaming model.[3] (Table 1 provides a summary of the algorithms mentioned here.)

The main idea was to modify the algorithm by Błasiok into a history-independent algorithm. This means that the algorithm will, given the same coin-flips, reach the same state independent of the order in which the stream elements arrive, or more precisely, independent of the execution tree as long as its nodes contain the same set of elements. This also means that the success event, i.e., whether an estimate computed from the state has the required accuracy, only depends on the set of distinct stream elements encountered (during the execution tree) and the initial random coin flips. As a consequence and in contrast to previous work, the correctness proof does not rely on bounds on the probability of certain events over the entire course of the algorithm, but can be established independent of past events.

Błasiok uses a pseudo-random construction based on hash families, expander walks, an extractor based on Parvaresh-Vardy codes [21] and a new sub-sampling strategy [9][Lem. 39]. I was able to build a simpler stack that only relies on hash families and a new two-stage expander graph construction, for which I believe there may be further applications. To summarize – the solution presented in this work has two key improvements:

[3] That the complexity is also optimal for the distributed setting is established in Section 7.

- Supports the sequential *and* distributed streaming model with optimal space.
- Requires fewer pseudo-random constructs, i.e., only hash families and expander walks.

The next section introduces notation. After that, I present new results on expander walks (Section 3) needed in the new pseudo-random construction and a self-contained presentation of the new algorithm and its correctness proof (Sections 5 and 6). Concluding with a discussion of its optimality in the distributed setting (Section 7) and a discussion of open research questions (Section 8). It is worthwhile noting that an extended version of this work [26] is available, which includes more background and more detailed proofs.

The results obtained in this work have also been formally verified [25] using the proof assistant Isabelle [31]. Isabelle has been used to verify many [1] advanced results from mathematics (e.g. the prime number theorem [12]) and computer science (e.g. the Cook-Levin theorem [5]). For readers mainly interested in the actual results, the formalization can be ignored as the theorems all contain traditional mathematical proofs. Nevertheless, Table 3 references the corresponding formalized fact for every theorem in this work.

2 Notation and Preliminaries

General constants are indicated as $C_1, C_2, \cdots$ etc. Their values are fixed throughout this work and are summarized in Table 2. For $n \in \mathbb{N}$, let us define $[n] := \{0, 1, \ldots, n-1\}$. The notation $[P]$ for a predicate P denotes the Iverson bracket, i.e., its value is 1 if the predicate is true and 0 otherwise. The notation $\operatorname{ld} x$ (resp. $\ln x$) stands for the logarithm to base 2 (resp. e) of $x \in \mathbb{R}_{>0}$. The notations $\lfloor x \rfloor$ and $\lceil x \rceil$ represent the floor and ceiling functions: $\mathbb{R} \to \mathbb{Z}$. For a probability space Ω, the notation $\mathcal{P}_{\omega \sim \Omega}(F(\omega))$ is the probability of the event: $\{\omega | F(\omega)\}$. And $\mathbb{E}_{\omega \sim \Omega}(f(\omega))$ is the expectation of f if ω is sampled from the distribution Ω, i.e., $\mathbb{E}_{\omega \sim \Omega}(f(\omega)) := \int_\Omega f(\omega)\, d\omega$. Similarly, $\mathbb{V} f = \mathbb{E}(f - \mathbb{E} f)^2$. For a finite non-empty set S, $U(S)$ is the uniform probability space over S, i.e., $\mathcal{P}(\{x\}) = |S|^{-1}$ for all $x \in S$. (Usually, we will abbreviate $U(S)$ with S when it is obvious from the context.) All probability spaces mentioned in this work will be discrete, i.e., measurability will be trivial.

All graphs in this work are finite and are allowed to contain parallel edges and self-loops. For an ordering of the vertices of such a graph, it is possible to associate an adjacency matrix $A = (a_{ij})$, where a_{ij} is the count of the edges between the i-th to the j-th vertex. We will say it is undirected d-regular if the adjacency matrix is symmetric and all its row (or equivalently) column sums are d. Such an undirected d-regular graph is called a λ-expander if the second largest absolute eigenvalue of its adjacency matrix is at most $d\lambda$.

Given an expander graph G, we denote by $\operatorname{Walk}(G, l)$, the set of walks of length l. For a walk $w \in \operatorname{Walk}(G, l)$ we write w_i for the i-th vertex and $w_{i,i+1}$ for the edge between the i-th and $(i+1)$-th vertex. Because of the presence of parallel edges, two distinct walks may have the same vertex sequence. As a probability space $U(\mathrm{Walk(G, l)})$ corresponds to choosing a random starting vertex and performing an $(l-1)$-step random walk.

3 Chernoff-type estimates for Expander Walks

The following theorem has been shown implicitly by Impagliazzo and Kabanets [22, Th. 10]:

▶ **Theorem 1** (Impagliazzo and Kabanets). *Let $G = (V, E)$ be a λ-expander graph and f a boolean function on its vertices, i.e.: $f : V \to \{0, 1\}$ s.t. $\mu = \mathbb{E}_{v \sim U(V)} f(v)$, $6\lambda \le \mu$ and $2\lambda < \varepsilon < 1$ then:*

$$\mathcal{P}_{w \sim \operatorname{Walk}(G,l)} \left(\sum\nolimits_{i \in [l]} f(w_i) \ge (\mu + \varepsilon) l \right) \le \exp(-l D(\mu + \varepsilon || \mu + 2\lambda))$$

Especially, the restriction $\mu \geq 6\lambda$ in the above result causes technical issues since usually one only has an upper bound for μ. The result follows in Impagliazzo and Kabanets work as a corollary from the application of their main theorem [22][Thm. 1] to the hitting property established by Alon et al. [3, Th. 4.2] in 1995. It is easy to improve Theorem 1 by using an improved hitting property:

▶ **Theorem 2** (Hitting Property for Expander Walks). *Let $G = (V, E)$ be a λ-expander graph and $W \subseteq V$, $I \subseteq [l]$ and let $\mu := \frac{|W|}{|V|}$ then:*

$$\mathcal{P}_{w \sim \mathrm{Walk}(G,l)} \left(\bigwedge_{i \in I} w_i \in W \right) \leq (\mu(1-\lambda)+\lambda)^{|I|} \leq (\mu+\lambda)^{|I|}$$

The above theorem for the case where $I = [l]$ is shown by Vadhan [37, Theorem 4.17]. The extended, full version [26] of this manuscript describes how to extend Vadhan's proof to the case where $I \subset [l]$. With the previous result, it is possible to obtain a new, improved version of Theorem 1:

▶ **Theorem 3** (Improved version of Theorem 1). *Let $G = (V, E)$ be a λ-expander graph and f a boolean function on its vertices, i.e.: $f : V \to \{0,1\}$ s.t. $\mu = \mathbb{E}_{v \sim U(V)} f(v)$ and $\mu + \lambda \leq \gamma \leq 1$ then:*

$$\mathcal{P}_{w \sim \mathrm{Walk}(G,l)} \left(\sum_{i \in [l]} f(w_i) \geq \gamma l \right) \leq \exp\left(-lD\left(\gamma || \mu+\lambda\right)\right)$$

Proof. This follows from Theorem 2 and the generalized Chernoff bound [22][Thm. 1]. ◀

Impagliazzo and Kabanets approximate the divergence $D(\gamma||\mu+\lambda)$ by $2(\gamma-(\mu+\lambda))^2$. In this work, we are interested in the case where $\mu+\lambda \to 0$, where such an approximation is too weak, so we cannot follow that approach. (Note that $D(\gamma||\mu+\lambda)$ can be arbitrarily large, while $(\gamma-(\mu+\lambda))^2$ is at most 1.) Instead, we derive a bound of the following form:

▶ **Lemma 4.** *Let $G = (V, E)$ be a λ-expander graph and f a boolean function on its vertices, i.e.: $f : V \to \{0,1\}$ s.t. $\mu = \mathbb{E}_{v \sim U(V)} f(v)$ and $\mu + \lambda \leq \gamma < 1$ then:*

$$\mathcal{P}_{w \sim \mathrm{Walk}(G,l)} \left(\sum_{i \in [l]} f(w_i) \geq \gamma l \right) \leq \exp\left(-l(\gamma \ln((\mu+\lambda)^{-1}) - 2e^{-1})\right)$$

Proof. The result follows from Theorem 3 and the inequality: $D(\gamma||p) \geq \gamma \ln(p^{-1}) - 1$ for $0 < \gamma < 1$ and $0 < p < 1$. ◀

An application for the above inequality, where the classic Chernoff-bound by Gillman [18] would not be useful, is establishing a failure probability for the repetition of an algorithm that already has a small failure probability. For example, if an algorithm has a failure probability of δ^*, then it is possible to repeat it $\mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln((\delta^*)^{-1})}\right)$-times to achieve a failure probability of δ. (This is done in Section 6.) Another consequence of this is a deviation bound for unbounded functions with a sub-gaussian tail bound:

▶ **Lemma 5** (Deviation Bound). *Let $G = (V, E)$ be a λ-expander graph and $f : V \to \mathbb{R}_{\geq 0}$ s.t. $\mathcal{P}_{v \sim U(V)}(f(v) \geq x) \leq \exp(-x(\ln x)^3)$ for $x \geq 20$ and $\lambda \leq \exp(-l(\ln l)^3)$ then*

$$\mathcal{P}_{w \sim \mathrm{Walk}(G,l)} \left(\sum_{i \in [l]} f(w_i) \geq C_1 l \right) \leq \exp(-l)$$

where $C_1 := e^2 + e^3 + (e-1) \leq 30$.

Note that the class includes sub-gaussian random variables but is even larger. The complete proof is in Appendix A. The proof essentially works by approximating the function f using the Iverson bracket: $f(x) \leq \Sigma_k [e^k \leq f(x) \leq e^{k+1}] e^{k+1}$ and establishing bounds on the frequency of each bracket. For large k this is established using the Markov inequality, and for small k the previous lemma is used. The result is a stronger version of a lemma established by Błasiok [9][Lem. 36], and the proof in this work is heavily inspired by his.

4 Explicit Pseudo-random Constructions

4.1 Strongly explicit expander graphs

For the application in this work, it is necessary to use *strongly explicit expander graphs.* For such a graph, it is possible to sample a random walk without having to represent the graph in memory. Moreover, sampling a random walk from a d-regular graph G with n-vertices is possible using a random sample from $[nd^{l-1}]$, i.e., we can map such a number to a walk algorithmically, such that the resulting distribution corresponds to the distribution from $\mathrm{Walk}(G, l)$ – this allows the previously mentioned two-stage construction.

A possible construction for strongly explicit expander graphs for every vertex count n and spectral bound λ is described by Murtagh et al. [29][Thm. 20, Apx. B]. Note that the degree d in their construction only grows polynomially with λ^{-1}, hence $\ln(d(\lambda)) \in \mathcal{O}(\ln(\lambda^{-1}))$. We will use the notation $\mathcal{E}([n], \lambda, l)$ for the sample space of random walks of length l in the described graph over the vertex set $[n]$. The same construction can also be used on arbitrary finite vertex sets S, if it is straightforward to map $[|S|]$ to S algorithmically. Thus we use the notation $\mathcal{E}(S, \lambda, l)$ for such S. Importantly $|\mathcal{E}(S, \lambda, l)| = |S|\, d(\lambda)^{l-1}$. Thus a walk in such a graph requires $\mathcal{O}(\mathrm{ld}\,|S| + l\,\mathrm{ld}(\lambda^{-1}))$ bits to represent.

4.2 Hash Families

Let us introduce the notation: $\mathcal{H}_k([n], [2^c])$ for the k-independent hash-family [39] from $[n]$ to $[2^c]$. Note that $\mathrm{ld}\,(|\mathcal{H}_k([n], [2^c])|) \in \mathcal{O}(k(c + \ln n))$.

For our application, we will need a second family with a geometric distribution (as opposed to uniform) on the range, in particular such that $\mathcal{P}(f(a) \geq k) = 2^{-k}$. A straightforward method to achieve that is to compose the functions of the hash family $\mathcal{H}_k([2^d], [2^d])$ with the function that computes the number of trailing zeros of the binary representation of its input $[2^d] \to [d]$. We denote such a hash family with $\mathcal{G}_k([2^d])$ where the range is $[d+1]$. Such a hash family is also one for a domain $[n] \subseteq [2^d]$, and hence we can extend the notation: $\mathcal{G}_k([n])$. Note that: $\mathcal{P}_{f \sim G_k([n])}(f(a) \geq k) = 2^{-k}$ for all $k \leq \lceil \mathrm{ld}\, n \rceil$ and also $\mathrm{ld}\,(|\mathcal{G}_k([n])|) \in \mathcal{O}(k \ln n)$.

5 The Algorithm

Because of all the distinct possible execution models, it is best to present the algorithm as a purely functional data structure with four operations:

$$\begin{aligned} &\mathrm{init} : () \to \mathrm{seed} && \mathrm{single} : [n] \to \mathrm{seed} \to \mathrm{sketch} \\ &\mathrm{merge} : \mathrm{sketch} \to \mathrm{sketch} \to \mathrm{sketch} && \mathrm{estimate} : \mathrm{sketch} \to \mathbb{R} \end{aligned}$$

The init step should be called only once globally – it is the only random operation – its result forms the seed and must be the same during the entire course of the algorithm. The operation single returns a sketch for a singleton set corresponding to its first argument. The operation merge computes a sketch representing the union of its input sketches and the operation estimate returns an estimate for the number of distinct elements for a given sketch.

The algorithm will be introduced in two successive steps. The first step is a solution that works for $(\ln n)^{-1} \leq \delta < 1$. The sketch requires only $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln\ln n)$, but the initial coin flips require $\mathcal{O}(\ln n + \ln(\varepsilon^{-1})^2 + \ln(\delta^{-1})^3)$ bits. For $\delta \geq (\ln n)^{-1}$ this is already optimal. In the second step (Section 6) a black-box vectorization of the previous algorithm will be needed to achieve the optimal $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$ space usage for all $0 < \delta < 1$.

For this entire section let us fix a universe size $n > 0$, a relative accuracy $0 < \varepsilon < 1$, a failure probability $(\ln n)^{-1} \leq \delta < 1$ and define:

$$l := \lceil C_6 \ln(2\delta^{-1}) \rceil \qquad b := 2^{\lceil \mathrm{ld}(C_4\varepsilon^{-2}) \rceil}$$

$$k := \lceil C_2 \ln b + C_3 \rceil \qquad \lambda := \min\left(\frac{1}{16}, \exp(-l(\ln l)^3)\right)$$

$$\Psi := \mathcal{G}_2([n]) \times \mathcal{H}_2([n], [C_7 b^2]) \times \mathcal{H}_k([C_7 b^2], [b]) \qquad \Omega := \mathcal{E}(\Psi, \lambda, l)$$

The implementation of the operations is presented in Algorithm 1. Note that these are

Algorithm 1 Algorithm for $\delta > (\ln n)^{-1}$.

```
function init() : $\Omega$
    return random $U(\Omega)$

function compress($(B,q) : \mathcal{S}$) : $\mathcal{S}$
    while $\sum_{i \in [l], j \in [b]} \lfloor \mathrm{ld}(B[i,j]+2) \rfloor > C_5 b l$
        $q \leftarrow q + 1$
        $B[i,j] \leftarrow \max(B[i,j]-1, -1)$ for $i \in [l], j \in [b]$
    return $(B,q)$

function single($x : U, \omega : \Omega$) : $\mathcal{S}$
    $B[i,j] \leftarrow -1$
    for $i \in [l]$
        $B[i, h(g(x))] = f(x)$ where $(f,g,h) = \omega_i$
    return compress$(B,0)$

function merge($(B_a,q_a) : \mathcal{S}, (B_b,q_b) : \mathcal{S}$) : $\mathcal{S}$
    $q \leftarrow \max(q_a, q_b)$
    $B[i,j] \leftarrow \max(B_a[i,j]+q_a-q, B_b[i,j]+q_b-q)$ for $i \in [l], j \in [b]$
    return compress$(B,q)$

function estimate($(B,q) : \mathcal{S}$) : $\mathbb{R}$
    for $i \in [l]$
        $s \leftarrow \max(0, \max\{B[i,j]+q \mid j \in [b]\} - \mathrm{ld}\, b + 9)$
        $p \leftarrow |\{j \in [b] | B[i,j]+q \geq s\}|$
        $Y_i \leftarrow 2^s \ln(1-pb^{-1})(\ln(1-b^{-1}))^{-1}$
    return median$(Y_0, \ldots, Y_{l-1})$
```

functional programs and pass the state as arguments and results; there is no global (mutable) state. The sketch consists of two parts (B,q). The first part is a two-dimensional table of sizes b and l. The second part is a single natural number, the cut-off level. The function compress is an internal operation and is not part of the public API. It increases the cut-off level and decreases the table values if the space usage is too high.

5.1 History-Independence

As mentioned in the introduction, this algorithm is history-independent, meaning that given the initial coin flips, it will reach the same state no matter in which permutation or frequency the stream elements are encountered. More precisely, the final state only depends on the set of encountered distinct elements over the execution tree and the initial coin flips, but not the shape of the tree. Informally, this is easy to see because the chosen cut-off level is the smallest possible with respect to the size of the values in the bins, and that property is maintained because the values in the bins are monotonically increasing with respect to the set of elements in the execution tree. Nevertheless, let us prove the property more rigorously. Let $\omega \in \Omega$ be the initial coin flips. Then there is a function $\tau(\omega, A)$ fulfilling the equations:

$$\begin{aligned} \text{single}(\omega, x) &= \tau(\omega, \{x\}) && (1) \\ \text{merge}(\tau(\omega, A), \tau(\omega, B)) &= \tau(\omega, A \cup B) && (2) \end{aligned}$$

The function τ is defined as follows:

$$\begin{aligned} \tau_0((f,g,h),A) &:= j \to \max\{f(a) \mid a \in A \wedge h(g(a)) = j\} \cup \{-1\} \\ \tau_1(\psi, A, q) &:= j \to \max\{\tau_0(\psi, A) - q, -1\} \\ \tau_2(\omega, A, q) &:= (i,j) \to \tau_1(\omega_i, A, q)[j] \\ q(\omega, A) &:= \min\left\{q \geq 0 \,\middle|\, \textstyle\sum_{i \in [l], j \in [b]} \lfloor \mathrm{ld}(\tau_2(\omega, A, q)[i,j] + 2) \rfloor \leq C_5 b l\right\} \\ \tau_3(\omega, A, q) &:= (\tau_2(\omega, A, q), q) \\ \tau(\omega, A) &:= \tau_3(\omega, A, q(\omega, A)) \end{aligned}$$

The function τ_0 describes the values in the bins if there were no compression, i.e., when $q = 0$. The function τ_1 describes the same for the given cut-off level q. Both are with respect to the selected hash functions $\psi = (f, g, h)$. The function τ_2 represents the state of all tables based on a seed for the expander. The next function τ_3 represents the entire state, which consists of the tables and the cut-off level. The function q represents the actual cut-off level that the algorithm would choose based on the values in the bins. Finally, the full state is described by the function τ for a given seed ω and set of elements A.

▶ **Lemma 6.** *Equations 1 and 2 hold for all $\omega \in \Omega$ and $\emptyset \neq A \subset [n]$.*

Proof. Let us also introduce the algorithms merge_1 and single_1. These are the algorithms merge and single but without the final compression step.

The following properties follow elementarily[4] from the definition of τ, s and the algorithms:

(i) $\tau(\omega, A) = \text{compress}(\tau_3(\omega, A, q))$ for all $0 \leq q \leq q(\omega, A)$
(ii) $\tau_3(\omega, A_1 \cup A_2, \max(q(\omega, A_1), q(\omega, A_2))) = \text{merge}_1(\tau(\omega, A_1), \tau(\omega, A_2))$
(iii) $\tau_3(\{x\}, 0) = \text{single}_1(\omega, x)$
(iv) $q(\omega, A_1) \leq q(\omega, A_2)$ if $A_1 \subseteq A_2$
(v) $q(A) \geq 0$

To verify Eq. 1 we can use i, iii and v and to verify Eq. 2 we use i, ii taking into account that $\max(q(\omega, A_1), q(\omega, A_2)) \leq q(\omega, A_1 \cup A_2)$ because of iv. ◀

[4] The verification relies on the semi-lattice properties of the max operator, as well as its translation invariance (i.e. $\max(a + c, b + c) = \max(a, b) + c$).

5.2 Overall Proof

Because of the argument in the previous section, $\tau(\omega, A)$ will be the state reached after any execution tree over the set A and the initial coin flips, i.e., $\omega \in \Omega$. Hence for the correctness of the algorithm, we only need to show that:

▶ **Theorem 7.** *Let $\emptyset \neq A \subseteq [n]$ then $\mathcal{P}_{\omega \in U(\Omega)} (\text{estimate}(\tau(\omega, A)) - |A| > \varepsilon\,|A|) \leq \delta$.*

Proof: Postponed. This will be shown in two steps: First, we want to establish that the cut-off threshold q will be equal to or smaller than $q_{\max} := \max(0, \lceil \text{ld}\,|A| \rceil - \text{ld}\, b)$ with high probability. And if the latter is true, then the estimate will be within the desired accuracy with high probability. For the second part, we verify that the estimation step will succeed with high probability for all $0 \leq q \leq q_{\max}$. (This will be because the sub-sampling threshold s in the estimation step will be $\geq q_{\max}$ with high probability.)

For the remainder of this section, let $\emptyset \neq A \subset [n]$ be fixed and we will usually omit the dependency on A. For example, we will write $\tau(\omega)$ instead of $\tau(\omega, A)$. Then, we can express the decomposition discussed above using the following chain:

$$\begin{aligned}&\mathcal{P}_{\omega\in\Omega} (|\text{estimate}(\tau(\omega)) - |A|| > \varepsilon\,|A|) \leq \\ &\mathcal{P}_{\omega\in\Omega} (\exists q \leq q_{\max}.\, |\text{estimate}(\tau_2(\omega, q)) - |A|| > \varepsilon\,|A| \vee q(\omega) > q_{\max}) \leq \frac{\delta}{2} + \frac{\delta}{2}\end{aligned} \tag{3}$$

Thus we only have to show the following two inequalities:

- $\mathcal{P}_{\omega\in\Omega} (q(\omega) > q_{\max}) \leq \frac{\delta}{2}$
- $\mathcal{P}_{\omega\in\Omega} (\exists q \leq q_{\max}.\, |\text{estimate}(\tau_2(\omega, q)) - |A|| > \varepsilon\,|A|) \leq \frac{\delta}{2}$

The first will be shown in the following subsection, and the next in the subsequent one.

5.3 Cut-off Level

This subsection proves that the cut-off level will be smaller than or equal to $q_{\max}$. This is the part where the tail estimate for sub-gaussian random variables over expander walks (Lemma 5) is applied:

▶ **Lemma 8.** $\mathcal{P}_{\omega\in\Omega} (q(\omega) > q_{\max}) \leq \frac{\delta}{2}$

Proof. Let us make a few preliminary observations:

$$\lfloor \text{ld}(x + 2) \rfloor \leq \text{ld}(x + 2) \leq (c + 2) + \max(x - 2^c, 0) \text{ for } (-1) \leq x \in \mathbb{R} \text{ and } c \in \mathbb{N}. \tag{4}$$

This can be verified using case distinction over $x \geq 2^c + 2$.

$$\mathbb{E}_{f \sim \mathcal{G}_2([n])} \max(f(a) - q_{\max} - 2^c, 0) \leq 2^{-q_{\max}} 2^{-2^c} \text{ for all } a \in [n] \text{ and } c \in \mathbb{N} \tag{5}$$

Note that this relies on the fact f is geometrically distributed.

$$|A|\, b^{-1} 2^{-q_{\max}} \leq 1 \tag{6}$$

This follows from the definition of $q_{\max}$ via case distinction.

To establish the result, we should take into account that $q(\omega)$ is the smallest cut-off level q fulfilling the inequality: $\sum_{i\in[l], j\in[b]} \lfloor \text{ld}(\tau_2(\omega, q)[i, j] + 2) \rfloor \leq C_5 bl$. In particular, if the inequality is true for $q_{\max}$, then we can conclude that $q(\omega)$ is at most $q_{\max}$, i.e.:

$$\mathcal{P}_{\omega\in\Omega} (q(\omega) > q_{\max}) = \mathcal{P}_{\omega\in\Omega} \left(\sum_{i\in[l], j\in[b]} \lfloor \text{ld}(\tau_2(\omega, q_{\max})[i, j] + 2) \rfloor > C_5 bl \right) \tag{7}$$

Let us introduce the random variable X over the seed space Ψ. It describes the space usage of a single column of the table B:

$$X(\psi) := \sum_{j\in[b]} \lfloor \mathrm{ld}(\tau_1(\psi, q_{\max})[j] + 2)\rfloor$$

Which can be approximated using Eq. 4 as follows:

$$X(\psi) \le \sum_{j\in[b]} c+2+\max(\tau_1(\psi, q_{\max})[j] - 2^c, 0) = \sum_{j\in[b]} c+2+\max(\tau_0(\psi)[j] - q_{\max} - 2^c, 0)$$

for all $0 \le c \in \mathbb{N}$. Hence:

$$\begin{aligned}
&\mathcal{P}_{\psi\sim\Psi}\left(X(\psi) \ge (c+3)b\right) \le \mathcal{P}_{\psi\sim\Psi}\left(\textstyle\sum_{j\in[b]} \max(\tau_0(\psi)[j] - q_{\max} - 2^c, 0) \ge b\right) \le \\
&\mathcal{P}_{(f,g,h)\sim\Psi}\left(\textstyle\sum_{j\in[b]} \max\{f(a) - q_{\max} - 2^c \mid a \in A \wedge h(g(a)) = j\} \cup \{0\} \ge b\right) \le \\
&\mathcal{P}_{(f,g,h)\sim\Psi}\left(\textstyle\sum_{a\in A} \max(f(a) - q_{\max} - 2^c, 0) \ge b\right) \le \\
&b^{-1} \sum_{a\in A} \mathbb{E}_{(f,g,h)\sim\Psi} \max(f(a) - q_{\max} - 2^c, 0) \le b^{-1} |A| \, 2^{-q_{\max}} 2^{-2^c} \le 2^{-2^c}
\end{aligned}$$

where the third and second-last inequality follow from Eq. 5 and 6. It is straightforward to conclude from the latter that for *all* $20 \le x \in \mathbb{R}$:

$$\mathcal{P}_{\psi\sim\Psi}\left(\frac{X(\psi)}{b} - 3 \ge x\right) \le \mathcal{P}_{\psi\sim\Psi}\left(X(\psi) \ge b(\lfloor x\rfloor + 3)\right) \le \exp(-2^{\lfloor x\rfloor} \ln 2) \le e^{-x(\ln x)^3}$$

Hence, it is possible to apply Lemma 5 on the random variables $b^{-1}X(\psi) - 3$ obtaining:

$$\mathcal{P}_{\omega\in\Omega}\left(\textstyle\sum_{i\in[l]} b^{-1}X(h(\omega, i)) - 3 \ge C_1 l\right) \le \exp(-l) \le \frac{\delta}{2}$$

This lemma now follows using $C_5 \ge C_1 + 3$ and that $\sum_{i\in[l]} X(h(\omega, i)) \le C_5 bl$ implies $q(\omega) \le q_{\max}$ as discussed at the beginning of the proof (Eq. 7). ◀

5.4 Accuracy

Let us introduce the random variables:

$$\begin{aligned}
t(f) &:= \max\{f(a) \mid a \in A\} - \mathrm{ld}\, b + 9 & s(f) &:= \max(0, t(f)) \\
p(f, g, h) &:= |\{j \in [b] \mid \tau_1((f, g, h), 0)[j] \ge s(f)\}| & Y(f, g, h) &:= 2^{s(f)} \rho^{-1}(p(f, g, h))
\end{aligned}$$

where $\rho(x) := b(1 - (1 - b^{-1})^x)$ – the expected number of hit bins when x balls are thrown into b bins. Note that the definitions t, p and Y correspond to the terms within the loop in the estimate function under the condition that the approximation threshold q is 0. In particular: $\mathrm{estimate}(\tau_3(\omega, 0)) = \mathrm{median}_{i\in[l]} Y(\omega_i)$ for $\omega \in \Omega$. Moreover, we denote by $R(f)$ the set of elements in A whose level is above the sub-sampling threshold, i.e.: $R(f) := \{a \in A \mid f(a) \ge s(f)\}$. The objective is to show that the individual estimates obtained in the loop in the estimate function (assuming $q = 0$) have the right accuracy and that the threshold $s \ge q_{\max}$ with high probability, i.e.:

$$\mathcal{P}_{\psi\sim\Psi}\left(|Y(\psi) - |A|| > \varepsilon |A| \vee s(f) < q_{\max}\right) \le \frac{1}{16} \tag{8}$$

In Lemma 14 this will be generalized to $0 \le q \le q_{\max}$. To be able to establish a bound on the above event, we need to check the likelihood of the following 4 events:

- The computed sub-sampling threshold $s(f)$ is approximately $\mathrm{ld}(|A|)$.
- The size of the sub-sampled elements $R(f)$ is a good approximation of $2^{-s(f)}|A|$.
- There is no collision during the application of g on the sub-sampled elements $R(f)$.
- The count of elements above the sub-sampling threshold in the table is close to the expected number $\rho(R(f))$ (taking collisions due to the application of h into account).

Then it will be possible to conclude that one of the above must fail if the approximation is incorrect. More formally:

$$E_1(\psi) :\leftrightarrow 2^{-16}b \le 2^{-t(f)}|A| \le 2^{-1}b \qquad E_2(\psi) :\leftrightarrow \left| |R(f)| - 2^{-s(f)}|A| \right| \le \tfrac{\varepsilon}{3}2^{-s(f)}|A|$$

$$E_3(\psi) :\leftrightarrow \forall a \ne b \in R(f). g(a) \ne g(b) \qquad E_4(\psi) :\leftrightarrow |p(\psi) - \rho(|R(f)|)| \le \tfrac{\varepsilon}{12}|R(f)|$$

for $\psi = (f, g, h) \in \Psi$. The goal is to show all four events happen simultaneously w.h.p.:

$$\mathcal{P}_{\psi\sim\Psi}(\neg E_1(\psi) \vee \neg E_2(\psi) \vee \neg E_3(\psi) \vee \neg E_4(\psi)) \le \frac{1}{16} \tag{9}$$

which can be shown by verifying: $\mathcal{P}_{\psi\sim\Psi}\left(\bigwedge_{j<i} E_j(\psi) \wedge \neg E_i(\psi)\right) \le 2^{-6}$ for each $i \in \{1, \dots, 4\}$. Let us start with the $i = 1$ case:

▶ **Lemma 9.** $\mathcal{P}_{\psi\in\Psi}(\neg E_1(\psi)) \le 2^{-6}$

Proof. For $X(f) = \max\{f(a) \mid a \in A\}$ it is possible to show:

$$\mathcal{P}_{(f,g,h)\sim\Psi}(X(f) < \mathrm{ld}(|A|) - k - 1) \le 2^{-k} \qquad \mathcal{P}_{(f,g,h)\sim\Psi}(X(f) > \mathrm{ld}(|A|) + k) \le 2^{-k}$$

using the proof for the F_0 algorithm by Alon et al. [4][Proposition 2.3]. The desired result follows taking $k = 7$ and that $t(f) = X(f) - \mathrm{ld}\, b + 9$. ◀

The following lemma is the interesting part of the proof in this subsection. In previous work, the sub-sampling threshold is obtained using a separate parallel algorithm, which has the benefit that it is straightforward to verify that $|R(f)|$ approximates $2^{-s}|A|$. The drawback is, of course, additional algorithmic complexity and an additional independent hash function. However, in the solution presented here, the threshold is determined from the data to be sub-sampled itself, which means it is not possible to assume independence. The solution to the problem is to show that $|R(f)|$ approximates $2^{-s}|A|$ with high probability for *all* possible values $s(f)$ assuming E_1.

▶ **Lemma 10.** $L := \mathcal{P}_{\psi\sim\Psi}(E_1(\psi) \wedge \neg E_2(\psi)) \le 2^{-6}$

Proof. Let $r(f, t) := |\{a \in A \mid f(a) \ge t\}|$ and $t_{\max}$ be maximal, s.t. $2^{-16}b \le 2^{-t_{\max}}|A|$. Then $2^7 \le \frac{\varepsilon^2}{9}2^{-16}b \le \frac{\varepsilon^2}{9}2^{-t_{\max}}|A|$. Hence: $2^{7+t_{\max}-t} \le \frac{\varepsilon^2}{9}2^{-t}|A| = \frac{\varepsilon^2}{9}\,\mathbb{E}\, r(f, t)$. Thus:

$$2^{7+t_{\max}-t}\,\mathbb{V}\, r(f, t) \le 2^{7+t_{\max}-t}\,\mathbb{E}\, r(f, t) \le \frac{\varepsilon^2}{9}(\mathbb{E}\, r(f, t))^2$$

for all $0 < t \le t_{\max}$. (This may be a void statement if $t_{\max} \le 0$.) Hence:

$$\mathcal{P}_{(f,g,h)\in\Psi}\left(\exists t. 0 < t \le t_{\max} \wedge |r(f, t) - \mathbb{E}\, r(\cdot, t)| > \frac{\varepsilon}{3}\,\mathbb{E}\, r(\cdot, t)\right) \le$$

$$\sum_{t=1}^{t_{\max}} \mathcal{P}_{(f,g,h)\in\Psi}\left(|r(f, t) - \mathbb{E}\, r(\cdot, t)| > \sqrt{2^{7+t_{\max}-t}\,\mathbb{V}\, r(f, t)}\right) \le \sum_{t=1}^{t_{\max}} 2^{-7-t_{\max}+t} \le 2^{-6}$$

Note that the predicate $E_2(\psi)$ is always true if $s(f) = 0$ because, in that case, there is no sub-sampling, i.e., $|R(f)| = |A|$. On the other hand if $s(f) > 0$, then $s(f) = t(f) \leq t_{\max}$ assuming $E_1(\psi)$. Hence:

$$\begin{aligned} L &\leq \mathcal{P}_{(f,g,h)}\left(s(f) > 0 \wedge E_1(f,g,h) \wedge \neg E_2(f,g,h)\right) \\ &\leq \mathcal{P}_{(f,g,h)}\left(0 < t(f) \leq t_{\max} \wedge \left||R(f)| - 2^{-t(f)}|A|\right| > \tfrac{\varepsilon}{3}2^{-t(f)}|A|\right) \\ &\leq \mathcal{P}_{(f,g,h)}\left(0 < t(f) \leq t_{\max} \wedge \left|r(f,t(f)) - 2^{-t(f)}|A|\right| > \tfrac{\varepsilon}{3}2^{-t(f)}|A|\right) \leq 2^{-6} \end{aligned}$$

where the last step follows from the previous equation. ◀

$$\text{Note that: } E_1(f,g,h) \wedge E_2(f,g,h) \to |R(f)| \leq \frac{2}{3}b \text{ for } (f,g,h) \in \Psi \tag{10}$$

▶ **Lemma 11.** $L := \mathcal{P}_{\psi\sim\Psi}(E_1(\psi) \wedge E_2(\psi) \wedge \neg E_3(\psi)) \leq 2^{-6}$

Proof. Using Eq. 10 we can conclude:

$$\begin{aligned} L &\leq \mathcal{P}_{(f,g,h)\sim\Psi}\left(|R(f)| \leq b \wedge (\exists a < b \in R(f).g(a) = g(b))\right) \\ &\leq \int_{\mathcal{G}_2([n])} [|R(f)| \leq b]\, \mathcal{P}_{g\sim\mathcal{H}_2([n],[C_7b^2])}(\exists a < b \in R(f).g(a) = g(b))\, df \\ &\leq \int_{\mathcal{G}_2([n])} [|R(f)| \leq b] \sum_{a<b\in R(f)} \mathcal{P}_{g\sim\mathcal{H}_2([n],[C_7b^2])}(g(a) = g(b))\, df \\ &\leq \int_{\mathcal{G}_2([n])} \frac{b(b-1)}{2C_7b^2}\, df \leq \frac{1}{2C_7} = 2^{-6}. \end{aligned}$$

◀

▶ **Lemma 12.** $L := \mathcal{P}_{\psi\sim\Psi}(E_1(\psi) \wedge E_2(\psi) \wedge E_3(\psi) \wedge \neg E_4(\psi)) \leq 2^{-6}$

Proof. Let $\tilde{R}(f,g,h) = \{i \in [C_7b^2] \mid f(a) \geq t(f) \wedge g(a) = i \wedge a \in A\}$ denote the indices hit in the domain $[C_7b^2]$ by the application of g on the elements above the sub-sampling threshold. If $E_3(f,g,h)$, then $\left|\tilde{R}(f,g,h)\right| = |R(f)|$ and if $E_1(f,g,h) \wedge E_2(f,g,h)$,then $|R(f)| \leq b$ (see Eq. 10). Recalling that $p(\psi)$ is the number of bins hit by the application of k-independent family from $\tilde{R}(\psi) \subseteq [C_7b^2]$ to $[b]$ we can apply Lemma 20. This implies:

$$\begin{aligned} &\mathcal{P}_{(f,g,h)\sim\Psi}\left(\textstyle\bigwedge_{i\in\{1,2,3\}} E_i(f,g,h) \wedge |p(f,g,h) - \rho(|R(f)|)| \geq \frac{\varepsilon}{12}|R(f)|\right) \leq \\ &\mathcal{P}_{\psi\sim\Psi}\left(\left|\tilde{R}(\psi)\right| \leq b \wedge \left|p(\psi) - \rho\left(\left|\tilde{R}(\psi)\right|\right)\right| \geq \frac{\varepsilon}{12}\left|\tilde{R}(\psi)\right|\right) \leq \\ &\mathcal{P}_{\psi\sim\Psi}\left(\left|\tilde{R}(\psi)\right| \leq b \wedge \left|p(\psi) - \rho\left(\left|\tilde{R}(\psi)\right|\right)\right| \geq 9b^{-1/2}\left|\tilde{R}(\psi)\right|\right) \leq 2^{-6} \end{aligned}$$

where we used, that $b \geq 9^2 12^2 \varepsilon^{-2}$ (i.e. $C_4 >= 9^2 12^2$). ◀

▶ **Lemma 13.** *Equation 8 is true.*

Proof. Let us start by observing that $E_1(\psi) \wedge E_2(\psi) \wedge E_4(\psi) \to |A^*(\psi) - |A|| \leq \varepsilon|A|$. This is basically an error propagation argument. First note that by using Eq. 10: $p(f,g,h) \leq \rho(R(f)) + \frac{\varepsilon}{12}|R(f)| \leq \rho(\frac{2}{3}b) + \frac{1}{12}|R(f)| \leq \frac{41}{60}b$. Moreover, using the mean value theorem:

$$\left|\rho^{-1}(p(f,g,h)) - |R(f)|\right| = (\rho^{-1})'(\xi)\, |p(f,g,h) - \rho(|R(f)|)| \leq \tfrac{\varepsilon}{3}|R(f)|$$

for some ξ between $\rho(|B(f)|)$ and $p(f,g,h)$ where we can approximate $(\rho^{-1})'(\xi) < 4$. Hence:

$$\begin{aligned}\left|\rho^{-1}(p(f,g,h)) - 2^{-s(f)}|A|\right| &\leq \left|\rho^{-1}(p(f,g,h)) - |R(f)|\right| + \left||R(f)| - 2^{-s(f)}|A|\right| \\ &\leq \frac{\varepsilon}{3}|R(f)| + \left||R(f)| - 2^{-s(f)}|A|\right| \\ &\leq \left(\frac{2\varepsilon}{3} + \frac{\varepsilon^2}{9}\right)2^{-s(f)}|A| \leq \varepsilon 2^{-s(f)}|A|\end{aligned}$$

It is also possible to deduce that $E_1(f,g,h) \to t(f) \geq \lceil \mathrm{ld}(|A|)\rceil - \mathrm{ld}\, b \to s(f) \geq q_{\max}$. Using Lemma 9 to 12 we can conclude that Equation 9 is true. And the implications derived here show that then Equation 8 must be true as well. ◀

To extend the previous result to the case: $q \leq q_{\max}$, let us introduce the random variables:

$$t_c(\psi, q) := \max\{\tau_1(\psi,q)[j] + q \mid j \in [b]\} - \mathrm{ld}\, b + 9 \qquad s_c(\psi,q) := \max(0, t_c(\psi,q))$$
$$p_c(\psi,q) := |\{j \in [b] \mid \tau_1(\psi,q)[j] + q \geq s_c(\psi,q)\}| \qquad Y_c(\psi,q) := 2^{s_c(\psi,q)}\rho^{-1}(p_c(\psi,q))$$

These definitions t_c, p_c and Y_c correspond to the terms within the loop in the estimate function for arbitrary q.

▶ **Lemma 14.** $\mathcal{P}_{\psi\sim\Psi}\left(\exists q \leq q_{\max}.\, |Y_c(\psi,q) - |A|| > \varepsilon|A|\right) \leq \frac{1}{16}$

Proof. It is possible to see that $t_c(\psi,q) = t(\psi)$ if $q \leq t(\psi)$. This is because $\tau_1(\psi,q) + q$ and $\tau_1(\psi,0)$ are equal except for values strictly smaller than q. With a case distinction on $t(\psi) \geq 0$ it is also possible to deduce that $s(\psi,q) = s(\psi)$ if $q \leq s(\psi)$. Hence: $p_c(\psi,q) = p(\psi)$ and $Y_c(\psi,q) = Y(\psi)$ (for $q \leq s(\psi)$). Thus this lemma is a consequence of Lemma 13. ◀

▶ **Lemma 15.** $L := \mathcal{P}_{\omega\in\Omega}\left(\exists q \leq q_{\max}.\, |\mathrm{estimate}(\tau_2(\omega,q)) - |A|| > \varepsilon|A|\right) \leq \frac{\delta}{2}$

Proof. Because the median of a sequence will certainly be in an interval, if more than half of the elements are in it, we can approximate the left-hand side as:

$$\begin{aligned} L &\leq \mathcal{P}_{\omega\in\Omega}\left(\exists q \leq q_{\max}.\, \sum_{i\in[l]}[|Y(\omega_i,q) - |A|| > \varepsilon|A|] \geq \frac{l}{2}\right) \\ &\leq \exp\left(-l\left(\frac{1}{2}\ln\left(\left(\frac{1}{16}+\frac{1}{16}\right)^{-1}\right) - 2e^{-1}\right)\right) \leq \exp\left(-\frac{l}{4}\right) \leq \frac{\delta}{2}\end{aligned}$$

The second inequality follows from Lemma 4 and 14 as well as $\lambda \leq \frac{1}{16}$. ◀

Proof of Theorem 7. Follows from Lemma 8 and the previous lemma, as well as the reasoning established in Equation 3. ◀

5.5 Space Usage

It should be noted that the data structure requires an efficient storage mechanism for the levels in the bins. We need to store the table values in a manner in which the number of bits required for a value x is proportional to $\ln x$. A simple strategy would be to store each value using a prefix-free universal code and concatenating the encoded variable-length bit strings.[5] A well-known universal code for positive integers is the Elias-gamma code, which

[5] Note that a vector of prefix-free values can be decoded even if they are just concatenated.

requires $2\lfloor \operatorname{ld} x \rfloor + 1$ bits for $x \geq 1$ [13]. Since, in our case, the values are integers larger or equal to (-1), they can be encoded using $2\lfloor \operatorname{ld}(x+2) \rfloor + 1$ bits.[6] In combination with the condition established in the compress function of Algorithm 1 the space usage for the table is thus $(2C_5 + 1)bl \in \mathcal{O}(bl) \subseteq \mathcal{O}(\ln(\delta^{-1})\varepsilon^2)$. Additionally, the approximation threshold needs to be stored. This threshold is a non-negative integer between 0 and $\operatorname{ld} n$ requiring $\mathcal{O}(\ln \ln n)$ bits to store. In summary, the space required for the sketch is $\mathcal{O}(\ln(\delta^{-1})\varepsilon^2 + \ln \ln n)$. For the coin flips, we need to store a random choice from Ω, i.e., we need to store $\ln(|\Omega|)$ bits. The latter is in $\mathcal{O}(\ln(|\Omega|)) \subseteq \mathcal{O}(\ln(|\Psi|) + l \ln(\lambda^{-1})) \subseteq \mathcal{O}(\ln n + \ln(\varepsilon^{-1})^2 + \ln(\delta^{-1})^3)$. Overall the total space for the coin flips and the sketch is $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n + \ln(\delta^{-1})^3)$.

6 Extension to small failure probabilities

The data structure described in the previous section has a space complexity that is close but exceeds the optimal $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$. The main reason this happens is that, with increasing length of the random walk, the spectral gap of the expander is increasing as well – motivated by the application of Lemma 5 in Subsection 5.3, with which we could establish that the cut-level could be shared between all tables. A natural idea is to restrict that.

If δ^{-1} is smaller than $\ln n$ the term $(\ln(\delta^{-1}))^3$ in the complexity of the algorithm is not a problem because it is dominated by the $\ln n$ term. If it is larger, we can split the table into sub-groups and introduce multiple cut-levels. Hence a single cut-level would be responsible for a smaller count of tables, and thus the spectral gap would be lower. (See also Figure 2).

A succinct way to precisely prove the correctness of the proposal is to repeat the previous algorithm, which has only a single shared cut-level, in a black-box manner for the same universe size and accuracy but for a higher failure probability. The seeds of each repetition are selected again using an expander walk. Here the advantage of Lemma 4 is welcome, as the inner algorithm needs to have a failure probability depending on n – the natural choice is $(\ln n)^{-1}$. The length of the walk of the inner algorithm matches the number of bits of the cut-level $\mathcal{O}(\ln \ln n)$. The repetition count of the outer algorithm is then $\mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln \ln n}\right)$.

▶ **Theorem 16.** *Let $n > 0$, $0 < \varepsilon < 1$ and $0 < \delta < 1$. Then there exists a cardinality estimation data structure for the universe $[n]$ with relative accuracy ε and failure probability δ with space usage $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$.*

Proof. If $\delta^{-1} < \ln n$, then the result follows from Theorem 7 and the calculation in Subsection 5.5. Moreover, if $n < \exp(e^5)$, then the theorem is trivially true, because there is an exact algorithm with space usage $\exp(e^5) \in \mathcal{O}(1)$. Hence we can assume $e^5 \leq \ln n \leq \delta^{-1}$. Let Ω^*, single*, merge* and estimate* denote the seed space and the API of Algorithm 1 for the universe $[n]$, relative accuracy ε and failure probability $\delta^* := (\ln n)^{-1}$. Moreover, let $m := \left\lceil 4\frac{\ln(\delta^{-1})}{\ln \ln n} \right\rceil$ – the plan is to show that with these definitions Algorithm 2 fulfills the conditions of this theorem. Let $\nu(\theta, A)[i] := \tau^*(\theta_i, A)$ for $i \in [m]$ and $\theta \in \Theta := U(\mathcal{E}(\Omega^*, \delta^*, m))$. Then it is straightforward to check that:

$$\operatorname{single}(\theta, x) = \nu(\theta, \{x\}) \qquad \operatorname{merge}(\nu(\theta, A), \nu(\theta, B)) = \nu(\theta, A \cup B)$$

for $x \in [n]$ and $\emptyset \neq A, B \subseteq [n]$ taking into account Lemma 6. Hence the correctness follows if: $\mathcal{P}_{\theta \in \Theta}(|\operatorname{estimate}(\nu(\theta, A)) - |A|| > \varepsilon |A|) \leq \delta$. Because the estimate is the median of the individual estimates, this is true if at least half of the individual estimates are in the desired range. Similar to the proof of Lemma 15 we can apply Lemma 4. This works if

[6] There are more sophisticated strategies for representing a sequence of variable-length strings that allow random access. [7]

$$\exp\left(-m\left(\frac{1}{2}\ln\left((\delta^*+\delta^*)^{-1}\right)-2e^{-1}\right)\right)\le\delta$$

which follows from $m \ge 4\ln(\delta^{-1})(\ln\ln n)^{-1}$ and $\ln\ln n \ge 5$. The space usage for the seed is: $\ln|\Theta| \in \mathcal{O}(\ln n+\ln(\varepsilon^{-1})^2+(\ln((\delta^*)^{-1}))^3+m\ln((\delta^*)^{-1})) \subseteq \mathcal{O}(\ln n+\ln(\varepsilon^{-1})^2+\ln(\delta^{-1}))$. The space usage for the sketch is: $\mathcal{O}(m\ln((\delta^*)^{-1})\varepsilon^{-2}+m\ln\ln n) \subseteq \mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2}+\ln\ln n)$. ◀

Algorithm 2 Algorithm for $0<\delta<(\ln n)^{-1}$.

```
function init() : $\Theta$
    return random $U(\Theta)$

function single($x : U, \theta : \Theta$) : $\mathcal{S}$
    $D[i] = \text{single}^*(x, \theta_i)$ for $i \in [m]$
    return $D$

function merge($D_a : \mathcal{S}, D_b : \mathcal{S}$) : $\mathcal{S}$
    $D[i] \leftarrow \text{merge}^*(D_a[i], D_b[i])$ for $i \in [m]$
    return $D$

function estimate($D : \mathcal{S}$) : $\mathbb{R}$
    $Y_i \leftarrow \text{estimate}^*(D[i])$ for $i \in [m]$
    return median($Y_0, \ldots, Y_{m-1}$)
```

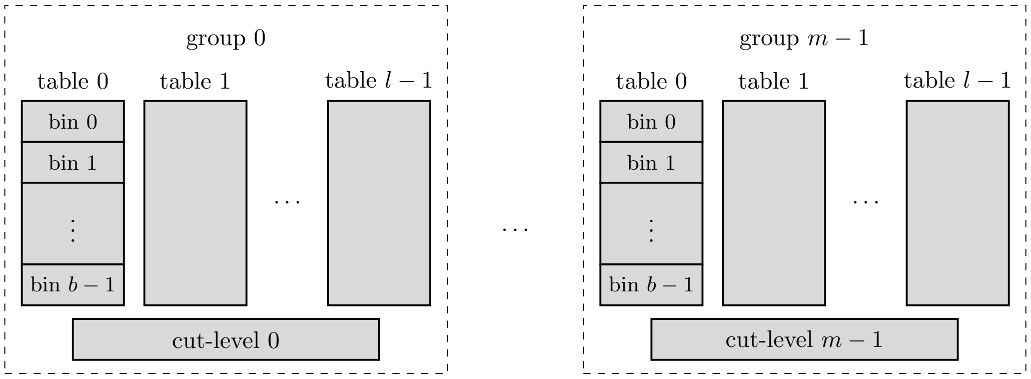

Figure 2 Schematic representation of the states of Algorithm 2 with $m \in \mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln\ln n}\right)$ repetitions of the inner algorithm. The inner algorithm uses $b \in \mathcal{O}(\varepsilon^{-2})$ bins and $l \in \mathcal{O}(\ln\ln n)$ tables.

7 Optimality

The optimality of the algorithm introduced by Błasiok [9] follows from the lower bound established by Jayram and Woodruff [23, Theorem 4.4]. The result (as well as its predecessors [4, 40]) follows from a reduction to a communication problem. This also means that their theorem is a lower bound on the information the algorithm needs to retain between processing successive stream elements.

An immediate follow-up question to Theorem 16 is whether the space usage is also optimal in the distributed setting. Let us assume there are p processes, each retaining m stream elements, and they are allowed to communicate at the beginning, before observing the stream elements, and after observing all stream elements. Even with these relaxed constraints, the number of bits that each process will need to maintain will be the same as the minimum number of bits of a sequential streaming solution. This follows by considering a specific subset of the input set where except for process 0, the stream elements on all the other processes are equal to the last stream element of process 0. In particular, the information the processes $1, 2, \ldots, p-1$ have is 0 bits from the perspective of process 0. If our distributed hypothetical algorithm is correct, it can only be so if the worst-case space usage per process is $\Omega(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$.

8 Conclusion

A summary of this work would be that for the space complexity of cardinality estimation algorithms, there is no gap between the distributed and sequential streaming models. Moreover, it is possible to solve the problem optimally (in either model) with expander graphs and hash families without using code-based extractors (as they were used in previous work). The main algorithmic idea is to avoid using a separate rough estimation data structure for quantization (cut-off); instead, the cut-off is guided by the space usage. During the estimation step at the end, an independent rough estimate is still derived, but it may be distinct from the cut-off reached at that point. This is the main difference between this solution and the approach by Kane et al. [24]. The main mathematical idea is to take the tail estimate based on the Kullback-Leibler divergence for random walks on expander graphs, first noted by Impagliazzo and Kabanets [22, Th. 10] seriously. With which, it is possible to achieve a failure probability of δ using $\mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln((\delta^*)^{-1})}\right)$ repetitions of an inner algorithm with a failure probability $\delta^* > \delta$. Note that the same cannot be done with the standard Gillman-type Chernoff [18] bounds. This allows the two-stage expander construction that we needed. As far as I can tell, this strategy is new and has not been used before.

An interesting question is whether the two-stage expander construction can somehow be collapsed into a single stage. For that, it is best to consider the following non-symmetric aggregate:

$$\mathcal{P}_{\omega \in \mathcal{E}(\mathcal{E}(S, \exp(-l(\ln l)^3), l), \exp(-l/m), m)} \left(\sum_{i \in [m]} \left[\sum_{j \in [l]} X(\omega_{ij}) \geq C_1 \right] \geq \frac{m}{2} \right) \leq \exp(-\mathcal{O}(lm))$$

where X may be an unbounded random variable with, e.g., sub-gaussian distribution. Indeed, the bound on the count of too-large cut-off values from Algorithm 2 turns out to be a tail estimate of the above form. I tried to obtain such a bound using only a single-stage expander walk but did not succeed without requiring too large spectral gaps, i.e., with $\lambda^{-1} \in \mathcal{O}(1)$ for $m \ll l$. There is a long list of results on more advanced Chernoff bounds for expander walks [2, 28, 30, 34, 35, 38] and investigations into more general aggregation (instead of summation) functions [10, 16, 19, 20, 32, 36], but I could not use any of these results/approaches to avoid the two-stage construction. This suggests that either there are more advanced results to be found or multi-stage expander walks are inherently more powerful than single-stage walks.

References

1 Archive of Formal Proofs. https://isa-afp.org. Accessed: 2023-07-03.

2 Rohit Agrawal. Samplers and Extractors for Unbounded Functions. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:21, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.59.

3 Noga Alon, Uriel Feige, Avi Wigderson, and David Zuckerman. Derandomized graph products. *computational complexity*, 5:60–75, 1995. doi:10.1007/BF01277956.

4 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.

5 Frank J. Balbach. The cook-levin theorem. *Archive of Formal Proofs*, January 2023. , Formal proof development. URL: https://isa-afp.org/entries/Cook_Levin.html.

6 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-45726-7_1.

7 Daniel K. Blandford and Guy E. Blelloch. Compact dictionaries for variable-length keys and data with applications. *ACM Trans. Algorithms*, 4(2), May 2008. doi:10.1145/1361192.1361194.

8 Jarosław Błasiok. Optimal streaming and tracking distinct elements with high probability. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2432–2448, 2018. doi:10.1137/1.9781611975031.156.

9 Jarosław Błasiok. Optimal streaming and tracking distinct elements with high probability. *ACM Trans. Algorithms*, 16(1):3:1–3:28, 2020. doi:10.1145/3309193.

10 Gil Cohen, Noam Peri, and Amnon Ta-Shma. Expander random walks: A fourier-analytic approach. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1643–1655, New York, NY, USA, 2021. doi:10.1145/3406325.3451049.

11 Jeffrey Dean and Sanjay Ghemawat. Mapreduce: A flexible data processing tool. *Commun. ACM*, 53(1):72–77, January 2010. doi:10.1145/1629175.1629198.

12 Manuel Eberl and Lawrence C. Paulson. The prime number theorem. *Archive of Formal Proofs*, September 2018. , Formal proof development. URL: https://isa-afp.org/entries/Prime_Number_Theorem.html.

13 P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

14 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985. doi:10.1016/0022-0000(85)90041-8.

15 Ian Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Addison-Wesley Longman Publishing Co., Inc., USA, 1995.

16 Ankit Garg, Yin Tat Lee, Zhao Song, and Nikhil Srivastava. A matrix expander chernoff bound. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1102–1114, New York, NY, USA, 2018. doi:10.1145/3188745.3188890.

17 Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '01, pages 281–291, 2001. doi:10.1145/378580.378687.

18 David Gillman. A chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998. doi:10.1137/S0097539794268765.

19 Louis Golowich. A new berry-esseen theorem for expander walks. *Electron. Colloquium Comput. Complex.*, TR22, 2022.

20 Louis Golowich and Salil Vadhan. Pseudorandomness of expander random walks for symmetric functions and permutation branching programs. In *Proceedings of the 37th Computational Complexity Conference*, CCC '22, Dagstuhl, Germany, 2022. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2022.27.

21 Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), July 2009. doi:10.1145/1538902.1538904.

22 Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In Maria Serna, Ronen Shaltiel, Klaus Jansen, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 617–631, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-15369-3_46.

23 T. S. Jayram and David P. Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Trans. Algorithms*, 9(3), June 2013. doi:10.1145/2483699.2483706.

24 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 41–52, New York, 2010. doi:10.1145/1807085.1807094.

25 Emin Karayel. Distributed distinct elements. *Archive of Formal Proofs*, April 2023. , Formal proof development. URL: https://isa-afp.org/entries/Distributed_Distinct_Elements.html.

26 Emin Karayel. An embarrassingly parallel optimal-space cardinality estimation algorithm, 2023. arXiv:2307.00985.

27 Emin Karayel. Expander graphs. *Archive of Formal Proofs*, March 2023. , Formal proof development. URL: https://isa-afp.org/entries/Expander_Graphs.html.

28 Pascal Lezaud. Chernoff-type bound for finite Markov chains. *The Annals of Applied Probability*, 8(3):849–867, 1998. doi:10.1214/aoap/1028903453.

29 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Deterministic Approximation of Random Walks in Small Space. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.42.

30 Assaf Naor, Shravas Rao, and Oded Regev. Concentration of markov chains with bounded moments. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 56(3):2270–2280, 2020. doi:10.1214/19-AIHP1039.

31 Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, first edition, 2002.

32 Daniel Paulin. Concentration inequalities for Markov chains by Marton couplings and spectral methods. *Electronic Journal of Probability*, 20:1–32, 2015. doi:10.1214/EJP.v20-4039.

33 Seth Pettie and Dingyu Wang. Information theoretic limits of cardinality estimation: Fisher meets shannon. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 556–569, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451032.

34 Shravas Rao. A hoeffding inequality for markov chains. *Electronic Communications in Probability*, 24:1–11, 2019. doi:10.1214/19-ECP219.

35 Shravas Rao and Oded Regev. A sharp tail bound for the expander random sampler, 2017. arXiv:1703.10205.

36 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 655–670, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40328-6_45.

37 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

38 Roy Wagner. Tail estimates for sums of variables sampled by a random walk. *Comb. Probab. Comput.*, 17(2):307–316, March 2008. doi:10.1017/S0963548307008772.

39 Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981. doi:10.1016/0022-0000(81)90033-7.

40 David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 167–175, USA, 2004. Society for Industrial and Applied Mathematics.

A Proof of Lemma 5

▶ **Lemma 5** (Deviation Bound). *Let $G = (V, E)$ be a λ-expander graph and $f : V \to \mathbb{R}_{\geq 0}$ s.t. $\mathcal{P}_{v\sim U(V)}(f(v) \geq x) \leq \exp(-x(\ln x)^3)$ for $x \geq 20$ and $\lambda \leq \exp(-l(\ln l)^3)$ then*

$$\mathcal{P}_{w\sim \text{Walk}(G,l)}\left(\textstyle\sum_{i\in[l]} f(w_i) \geq C_1 l\right) \leq \exp(-l)$$

where $C_1 := e^2 + e^3 + (e-1) \leq 30$.

Proof. Let $\mu_k := \mathbb{E}_{v\sim V}[e^k \leq f(v)] \leq \exp(-e^k k^3)$ for $k \geq 3$. We will show

$$L_k := \mathcal{P}_{w\sim \text{Walk}(G,l)}\left(\sum_{i\in[l]}[e^k \leq f(w_i)] \geq l e^{-k} k^{-2}\right) \leq \exp(-l-k+2) \text{ for all } k \geq 3 \quad (11)$$

by case distinction on the range of k:

Case $k \geq \max(\ln l, 3)$: In this case the result follows using Markov's inequality. Note that the random walk starts from and remains in the stationary distribution, and thus for any index $i \in [l]$ the distribution of the i-th walks step w_i will be uniformly distributed over V, hence:

$$\begin{aligned} L_k &\leq e^k k^2 l^{-1}\, \mathbb{E}_{w\sim \text{Walk}(G,l)} \textstyle\sum_{i\in[l]}[e^k \leq f(w_i)] = e^k k^2\, \mathbb{E}_{v\sim V}[e^k \leq f(v)] \\ &\leq e^k k^2 \exp(-e^k k^3) = \exp(k + 2\ln k - e^k k^3) \leq \exp(2k - e^k(k^2+2)) \\ &\leq \exp(2k - e^k k^2 - e^k - e^k) \leq \exp(-l-k+2) \end{aligned}$$

Here we use that $k^3 \geq k^2 + 2$ and $e^k \geq k$ for $k \geq 3$ and $e^k \geq l$.

Case $3 \leq k < \ln l$: Then we have

$$\begin{aligned} L_k &\leq \exp\left(-l(e^{-k}k^{-2}\ln((\mu_k+\lambda)^{-1}) - 2e^{-1})\right) \quad \text{using Lemma 4} \\ &\leq \exp\left(-l(e^{-k}k^{-2}(e^k k^3 - \ln 2) - 2e^{-1})\right) \leq \exp\left(-l(k - e^{-k}k^{-2}\ln 2 - 2e^{-1})\right) \\ &\leq \exp\left(-l(k-1)\right) \leq \exp\left(-l-k+2\right) \end{aligned}$$

Concluding the proof of Eq. 11.

Note that:

$$\begin{aligned} \sum_{i\in[l]} f(w_i) &\leq e^2 l + \sum_{i\in[l]}\sum_{k\geq 2} e^{k+1}[e^k \leq f(w_i) < e^{k+1}] \\ &\leq e^2 l + \sum_{i\in[l]}\left(\sum_{k\geq 2} e^{k+1}[e^k \leq f(w_i)] - \sum_{k\geq 2} e^{k+1}[e^{k+1} \leq f(w_i)]\right) \\ &\leq (e^2+e^3)l + (e-1)\sum_{i\in[l]}\left(\sum_{k\geq 3} e^k[e^k \leq f(w_i)]\right) \end{aligned}$$

Hence:

$$\begin{aligned}\mathcal{P}_{w\sim\text{Walk}(G,l)}\left(\sum_{i\in[l]} f(w_i) \geq C_1 l\right) &\leq \mathcal{P}_{w\sim\text{Walk}(G,l)}\left(\sum_{k\geq 3, i\in[l]} e^k[e^k \leq f(w_i)] \geq l\right)\\ &\leq \mathcal{P}_{w\sim\text{Walk}(G,l)}\left(\bigvee_{k\geq 3}\sum_{i\in[l]}[e^k \leq f(w_i)] \geq le^{-k}k^{-2}\right)\\ &\leq \sum_{k\geq 3} L_k \leq \sum_{k\geq 3}\exp(-l-k+2) \leq \exp(-l).\end{aligned}$$ ◀

B Balls and Bins

Let $\Omega = U([r] \to [b])$ be the uniform probability space over the functions from $[r]$ to $[b]$ for $b \geq 1$ and $0 \leq r \leq b$ and let $X(\omega) = |\omega([r])|$ be the size of the image of such a function. This models throwing r balls into b bins independently, where X is the random variable counting the number of hit bins. Moreover, let $E_i(\omega) = \{\omega \mid i \in \omega([r])\}$ be the event that the bin i was hit. Note that $X(\omega) = \sum_{i\in[b]} E_i(\omega)$. And we want to show that

$$\mathbb{E}_{\omega\sim\Omega} X(\omega) = b\left(1-\left(1-\frac{1}{b}\right)^r\right) \qquad\qquad \mathbb{V}_{\omega\sim\Omega} X(\omega) \leq \frac{r(r-1)}{b}$$

▶ **Lemma 17.** $\mathbb{E}_{\omega\sim\Omega} X(\omega) = b\left(1-\left(1-\frac{1}{b}\right)^r\right)$

The proof is available in the full version [26].

▶ **Lemma 18.** $\mathbb{V}_{\omega\sim\Omega} X(\omega) \leq \frac{r(r-1)}{b}$

The proof is available in the full version [26]. The above is a stronger version of the result by Kane et al. [24][Lem. 1]. Their result has the restriction that $r \geq 100$ and a superfluous factor of 4.

Interestingly, it is possible to obtain a similar result for k-independent balls into bins. For that let Ω' be a probability space of functions from $[r]$ to $[b]$ where

$$\mathcal{P}_{\omega\sim\Omega'}\left(\bigwedge_{i\in I}\omega(i) = x(i)\right) = r^{-|I|}$$

for all $I \subset [r]$, $|I| \leq k$ and all $x : I \to [b]$. As before let us denote $X'(\omega) := |\omega([r])|$ the number of bins hit by the r balls. Then the expectation (resp. variance) of X' approximates that of X with increasing independence k, more precisely:

▶ **Lemma 19.** *If $\varepsilon \leq e^{-2}$ and $k \geq 1 + 5\ln(b\varepsilon^{-1})(\ln(\ln(b\varepsilon^{-1})))^{-1}$ then:*

$$|\mathbb{E}_{\omega'\in\Omega'} X'(\omega') - \mathbb{E}_{\omega\in\Omega} X(\omega)| \leq \varepsilon r \qquad\qquad |\mathbb{V}_{\omega'\in\Omega'} X'(\omega') - \mathbb{V}_{\omega\in\Omega} X(\omega)| \leq \varepsilon^2 .$$

This has been shown[7] by Kane et al. [24][Lem. 2]. The proof relies on the fact that $X = \sum_{i\in[b]}\max(1, Y_i)$ where Y_i denotes the random variable that counts the number of balls in bin i. It is possible to show that $\mathbb{E}(Y_i)^j = \mathbb{E}(Y_i')^j$ for all $j \leq k$ (where Y_i' denotes the same notion over Ω'). Their approach is to approximate $\max(1, \cdot)$ with a polynomial g of

[7] Without the explicit constants mentioned in here.

degree k. Since $\mathbb{E}\, g(Y_i) = \mathbb{E}\, g(Y_i')$ they can estimate the distance between $\mathbb{E}\, X$ and $\mathbb{E}\, X'$ by bounding the expectation of each approximation error: $g(Y_i) - \max(1, Y_i)$. Obviously, larger degree polynomials (and hence increased independence) allow better approximations. The reasoning for the variance is analogous.

▶ **Lemma 20.** *If $k \geq C_2 \ln b + C_3$ then:*

$$L := \mathcal{P}_{\omega' \in \Omega'}\left(|X'(\omega') - \rho(r)| > 9b^{-1/2}r\right) \leq 2^{-6}$$

Proof. This follows from Lemma 17, 18 and the previous lemma for $\varepsilon = \min(e^{-2}, b^{-1/2})$ in particular: $\mathbb{V}\, X' \leq \mathbb{V}\, X + \frac{1}{b} \leq \frac{r^2}{b}$ and hence:

$$\begin{aligned} L &\leq \mathcal{P}_{\omega' \in \Omega'}\left(|X'(\omega') - \mathbb{E}\, X'| + |\mathbb{E}\, X' - \rho(r)| \geq 9b^{-1/2}r\right) \\ &\leq \mathcal{P}_{\omega' \in \Omega'}\left(|X'(\omega') - \mathbb{E}\, X'| + b^{-1/2}r \geq 9b^{-1/2}r\right) \\ &\leq \mathcal{P}_{\omega' \in \Omega'}\left(|X'(\omega') - \mathbb{E}\, X'| \geq 8b^{-1/2}r\right) \\ &\leq \mathcal{P}_{\omega' \in \Omega'}\left(|X'(\omega') - \mathbb{E}\, X'| \geq 8\sqrt{\mathbb{V}\, X'}\right) \leq 2^{-6} \end{aligned}$$

where the last line follows from Chebychev's inequality. ◀

C Table of Constants

Table 2 Table of Constants.

Constant	References	Constant	References
$C_1 := e^2 + e^3 + (e-1)$	Lemma 5	$C_2 := \frac{15}{2}$	Lemma 19
$C_3 := 16$	Lemma 19	$C_4 := 3^2 2^{23}$	Lemma 9 and 12
$C_5 := \lceil C_1 + 3 \rceil = 33$	Lemma 8	$C_6 := 4$	Lemma 15
$C_7 := 2^5$	Lemma 11		

D Formalization

As mentioned in the introduction the proofs in this work have been machine-checked using Isabelle. They are available [25, 27] in the AFP (Archive of Formal Proofs) [1] – a site hosting formal proofs verified by Isabelle. Table 3 references the corresponding facts in the AFP entries. The first column refers to the lemma in this work. The second is the corresponding name of the fact in the formalization. The formalization can be accessed in two distinct forms: As a source repository with distinct theory files, as well as two "literate-programming-style" PDF documents with descriptive text alongside the Isabelle facts (optionally with the proofs). The latter is much more informative. The third column of the table refers to the file name [8] of the corresponding source file, while the last column contains the reference of the AFP entry, including the section in the PDF versions.

[8] `Distributed_Distinct_Elements` is abbreviated by `DDE` and `Without` with `WO`.

Table 3 Reference to the formal entities.

Lemma	Formalized Entity	Theory	Src.
Thm. 1	This theorem from Impagliazzo and Kabanets was stated for motivational reasons and is never used in any of the following results, hence it is not formalized.		
Thm. 2	**theorem** *hitting-property*	`Expander_Graphs_Walks`	[27, §9]
Thm. 3	**theorem** *kl-chernoff-property*	`Expander_Graphs_Walks`	[27, §9]
Lem. 4	**lemma** *walk-tail-bound*	`DDE_Tail_Bounds`	[25, §5]
Lem. 5	**lemma** *deviation-bound*	`DDE_Tail_Bounds`	[25, §5]
Lem. 6 (1)	**lemma** *single-result*	`DDE_Inner_Algorithm`	[25, §6]
Lem. 6 (2)	**lemma** *merge-result*	`DDE_Inner_Algorithm`	[25, §6]
Lem. 8	**lemma** *cutoff-level*	`DDE_Cutoff_Level`	[25, §8]
Lem. 9	**lemma** *e-1*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 10	**lemma** *e-2*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 11	**lemma** *e-3*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 12	**lemma** *e-4*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 13	**lemma** *accuracy-without-cutoff*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 14	**lemma** *accuracy-single*	`DDE_Accuracy`	[25, §9]
Lem. 15	**lemma** *estimate-result-1*	`DDE_Accuracy`	[25, §9]
Thm. 7	**lemma** *estimate-result*	`DDE_Accuracy`	[25, §9]
Thm. 16 (1)	**theorem** *correctness*	`DDE_Outer_Algorithm`	[25, §10]
Thm. 16 (2)	**theorem** *space-usage*	`DDE_Outer_Algorithm`	[25, §10]
Thm. 16 (3)	**theorem** *asymptotic-space-complexity*	`DDE_Outer_Algorithm`	[25, §10]
Lem. 17	**lemma** *exp-balls-and-bins*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 18	**lemma** *var-balls-and-bins*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 19 (1)	**lemma** *exp-approx*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 19 (2)	**lemma** *var-approx*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 20	**lemma** *deviation-bound*	`DDE_Balls_And_Bins`	[25, §4]