Verification of the CVM algorithm with a Functional Probabilistic Invariant

E. Karayel[1], J. Watt[2], D. Khu[2], K. Meel[4,5], Y. K. Tan[2,3]

[1]Technical University of Munich, Germany

[2]Institute for Infocomm Research (I^{2}R), A*STAR, Singapore

[3]Nanyang Technological University, Singapore

[4]Georgia Institute of Technology, Atlanta, GA, USA

[5]University of Toronto, Canada

September 27, 2025

Table of Contents

1 Introduction to the CVM algorithm

2 Summary of Results

3 Verification using Functional Probabilistic Invariants

4 Sketch of the Original Proof

Table of Contents

1 Introduction to the CVM algorithm

2 Summary of Results

3 Verification using Functional Probabilistic Invariants

4 Sketch of the Original Proof

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$
- Algorithm has only limited mutable state: $\mathcal{O}(\varepsilon^{-2} \ln(\delta^{-1} l) b)$

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$
- Algorithm has only limited mutable state: $\mathcal{O}(\varepsilon^{-2} \ln(\delta^{-1} l) b)$
- Approximation of the count of distinct elements

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$
- Algorithm has only limited mutable state: $\mathcal{O}(\varepsilon^{-2} \ln(\delta^{-1} l) b)$
- Approximation of the count of distinct elements
- Randomized

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$
- Algorithm has only limited mutable state: $\mathcal{O}(\varepsilon^{-2} \ln(\delta^{-1} l) b)$
- Approximation of the count of distinct elements
- Randomized
- Small error with high probability

$$\mathcal{P}\left(|X - |A|| \geq \varepsilon |A|\right) \leq \delta$$

for choosable $\varepsilon > 0$, $\delta > 0$

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$
- Algorithm has only limited mutable state: $\mathcal{O}(\varepsilon^{-2} \ln(\delta^{-1} l) b)$
- Approximation of the count of distinct elements
- Randomized
- Small error with high probability

  $$\mathcal{P}\left(|X - |A|| \geq \varepsilon |A|\right) \leq \delta$$

  for choosable $\varepsilon > 0$, $\delta > 0$
- Suprisingly simple compared to previous algorithms for the same problem

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$
- Algorithm has only limited mutable state: $\mathcal{O}(\varepsilon^{-2} \ln(\delta^{-1} l) b)$
- Approximation of the count of distinct elements
- Randomized
- Small error with high probability

  $$\mathcal{P}\left(|X - |A|| \geq \varepsilon |A|\right) \leq \delta$$

  for choosable $\varepsilon > 0$, $\delta > 0$
- Suprisingly simple compared to previous algorithms for the same problem (see e.g. article in Quanta Magazine[7])

What is the CVM algorithm?

- Name after initals of S. Chakraborty, N. V. Vinodchandran, K. S. Meel
- Streaming Algorithm (one sequential pass): $a_1, \ldots, a_l$
- Algorithm has only limited mutable state: $\mathcal{O}(\varepsilon^{-2} \ln(\delta^{-1} l) b)$
- Approximation of the count of distinct elements
- Randomized
- Small error with high probability

  $$\mathcal{P}\left(|X - |A|| \geq \varepsilon |A|\right) \leq \delta$$

  for choosable $\varepsilon > 0$, $\delta > 0$
- Suprisingly simple compared to previous algorithms for the same problem (see e.g. article in Quanta Magazine[7])
- No hashing

Illustration of Streaming Algorithms

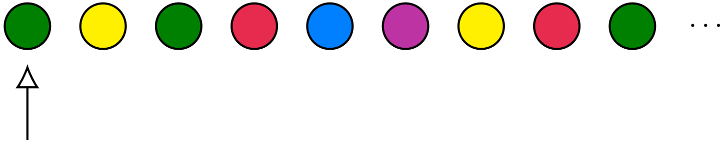
$\boxed{a_1}\boxed{a_1}\boxed{a_2} \quad \cdots \quad \boxed{a_l}$

The CVM Algorithm

Input: Stream elements $a_1, \ldots, a_l$, $0 < \varepsilon$, $0 < \delta < 1$.
Output: A cardinality estimate R for set $A = \{a_1, \ldots, a_l\}$

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**
  $b \xleftarrow{\$} \mathrm{Ber}(p)$ ▷ random bit b from the Bernoulli distribution
  if b **then** ▷ insert a_i if b is true (with prob. p)
    $\chi \leftarrow \chi \cup \{a_i\}$
  else ▷ remove a_i otherwise
    $\chi \leftarrow \chi - \{a_i\}$
  if $|\chi| = n$ **then** ▷ subsample if buffer χ is full
    $\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$
    $p \leftarrow \frac{p}{2}$
  if $|\chi| = n$ **then return** $\perp$ ▷ fail if χ remains full
return $\frac{|\chi|}{p}$ ▷ estimate cardinality of A

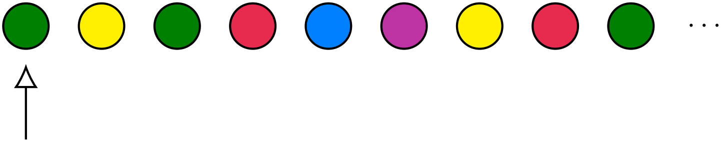
Illustration of the Algorithm

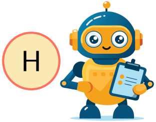


$$\chi = \{ \qquad \}$$

$$p = 1, n = 4$$

Illustration of the Algorithm

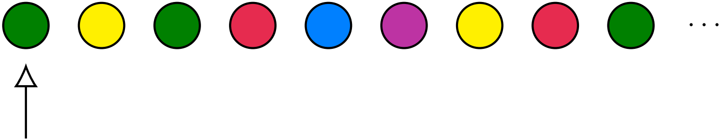


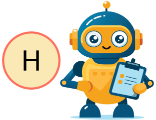


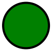
$\chi = \{ \quad \}$

$p = 1, n = 4$

Illustration of the Algorithm

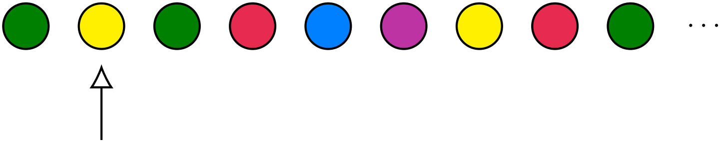


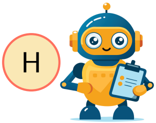


$\chi = \{$  $\}$

$p = 1, n = 4$

Illustration of the Algorithm

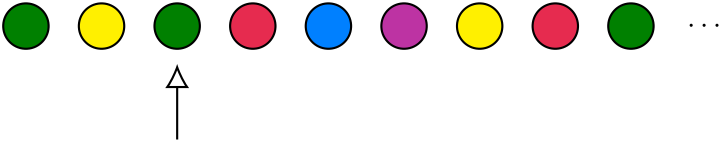


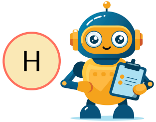


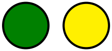
$\chi = \{ \qquad \}$

$p = 1, n = 4$

Illustration of the Algorithm

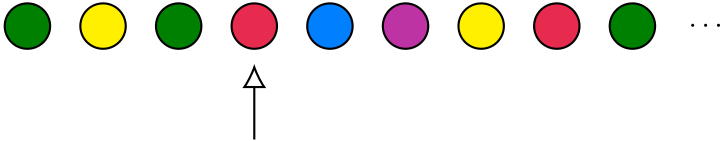


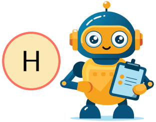


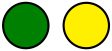
$\chi = \{$  $\}$

$p = 1, n = 4$

Illustration of the Algorithm

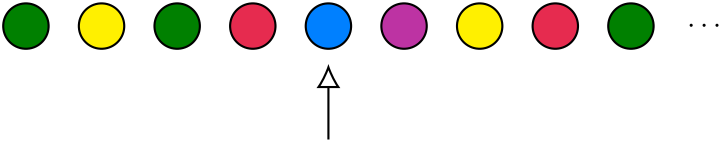


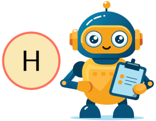


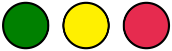
$\chi = \{$  $\}$

$p = 1, n = 4$

Illustration of the Algorithm

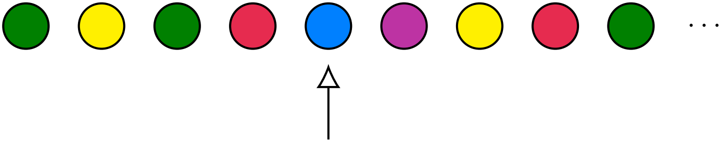


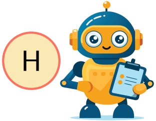


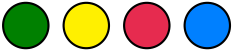
$\chi = \{$  $\}$

$p = 1, n = 4$

Illustration of the Algorithm

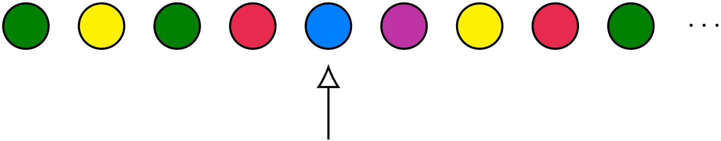


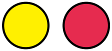


$\chi = \{$  $\}$

$p = 1, n = 4$

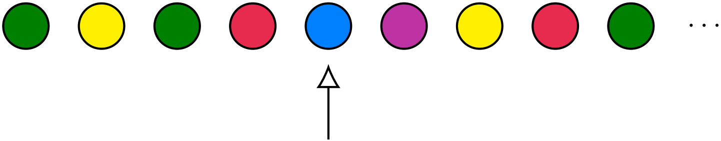
Illustration of the Algorithm

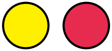


$\chi = \{$  $\}$

$p = 1, n = 4$

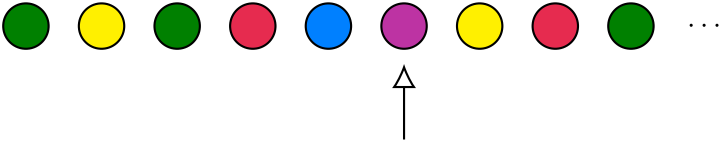
Illustration of the Algorithm

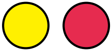


$\chi = \{$  $\}$

$p = \frac{1}{2}, n = 4$

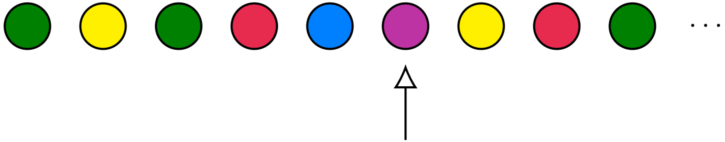
Illustration of the Algorithm

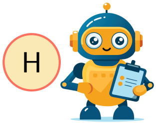


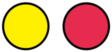
$\chi = \{$  $\}$

$p = \frac{1}{2}, n = 4$

Illustration of the Algorithm

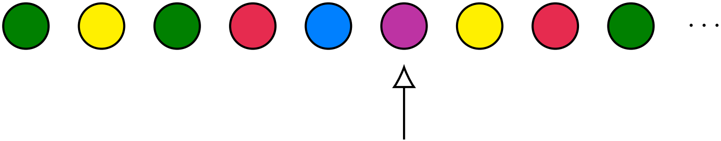


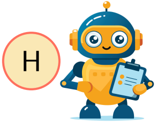


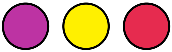
$\chi = \{$  $\}$

$p = \frac{1}{2}, n = 4$

Illustration of the Algorithm

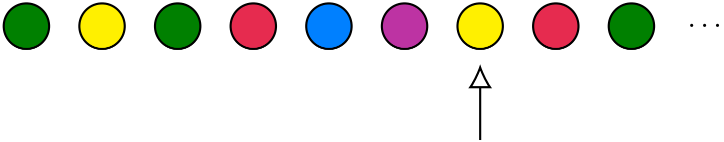


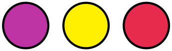


$\chi = \{$  $\}$

$p = \frac{1}{2}, n = 4$

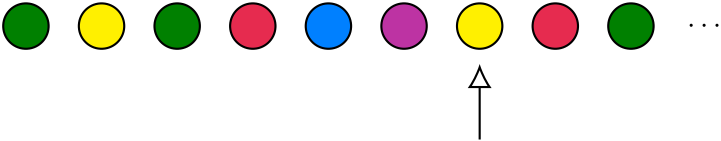
Illustration of the Algorithm

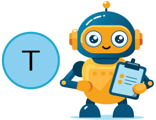


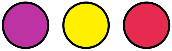
$\chi = \{$  $\}$

$p = \frac{1}{2}, n = 4$

Illustration of the Algorithm

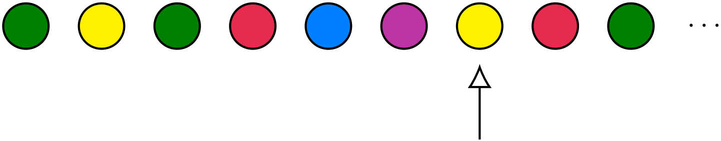




$\chi = \{$  $\}$

$p = \frac{1}{2}, n = 4$

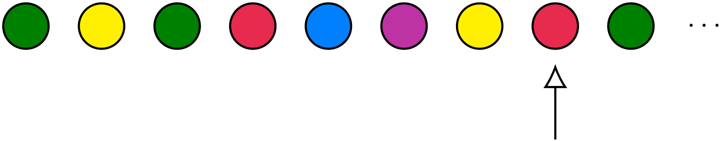
Illustration of the Algorithm



$\chi = \{$  $\}$

$p = \frac{1}{2}, n = 4$

Illustration of the Algorithm



$\chi = \{ \qquad \qquad \}$

$p = \frac{1}{2}, n = 4$

The CVM Algorithm

Input: Stream elements $a_1, \ldots, a_l$, $0 < \varepsilon$, $0 < \delta < 1$.
Output: A cardinality estimate R for set $A = \{a_1, \ldots, a_l\}$

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**
  $b \xleftarrow{\$} \mathrm{Ber}(p)$ ▷ random bit b from the Bernoulli distribution
  if b **then** ▷ insert a_i if b is true (with prob. p)
    $\chi \leftarrow \chi \cup \{a_i\}$
  else ▷ remove a_i otherwise
    $\chi \leftarrow \chi - \{a_i\}$
  if $|\chi| = n$ **then** ▷ subsample if buffer χ is full
    $\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$
    $p \leftarrow \frac{p}{2}$
  if $|\chi| = n$ **then return** $\perp$ ▷ fail if χ remains full
return $\frac{|\chi|}{p}$ ▷ estimate cardinality of A

Comparison with Previous Algorithms

Year	Author(s)	Space	Remarks
2001	Bar-Yossef et al. [1]	$\tilde{O}(\ln(\delta^{-1})(\varepsilon^{-2} + b))$	M. H.
2007	HLL [4]	$O(\ln(\delta^{-1}\varepsilon^{-2} \ln b)$	O. M. H.
2010	Kane et al. [5]	$O(\ln(\delta^{-}1)(\varepsilon^{-2} + b))$	H.
2020	Błasiok [2]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E.
2022	Karayel [6]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E. M.
2023	Chakraborty et al. [3]	$O(\varepsilon^{-2} \ln(\delta^{-1} l) b)$	-

Abbreviations:

- M: Supports merging of sketches.
- H: Relies on hash functions.
- E: Relies on expander graphs.
- O: Random oracle model.

Comparison with Previous Algorithms

Year	Author(s)	Space	Remarks
2001	Bar-Yossef et al. [1]	$\tilde{O}(\ln(\delta^{-1})(\varepsilon^{-2} + b))$	M. H.
2007	HLL [4]	$O(\ln(\delta^{-1}\varepsilon^{-2} \ln b)$	O. M. H.
2010	Kane et al. [5]	$O(\ln(\delta^{-}1)(\varepsilon^{-2} + b))$	H.
2020	Błasiok [2]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E.
2022	Karayel [6]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E. M.
2023	Chakraborty et al. [3]	$O(\varepsilon^{-2} \ln(\delta^{-1} l) b)$	-

Abbreviations:

- M: Supports merging of sketches.
- H: Relies on hash functions.
- E: Relies on expander graphs.
- O: Random oracle model.

Comparison with Previous Algorithms

Year	Author(s)	Space	Remarks
2001	Bar-Yossef et al. [1]	$\tilde{O}(\ln(\delta^{-1})(\varepsilon^{-2}+b))$	M. H.
2007	HLL [4]	$O(\ln(\delta^{-1}\varepsilon^{-2}\ln b)$	O. M. H.
2010	Kane et al. [5]	$O(\ln(\delta^{-}1)(\varepsilon^{-2}+b))$	H.
2020	Błasiok [2]	$O(\varepsilon^{-2}\ln(\delta^{-1})+b)$	H. E.
2022	Karayel [6]	$O(\varepsilon^{-2}\ln(\delta^{-1})+b)$	H. E. M.
2023	Chakraborty et al. [3]	$O(\varepsilon^{-2}\ln(\delta^{-1}l)b)$	-

Abbreviations:

- M: Supports merging of sketches.
- H: Relies on hash functions.
- E: Relies on expander graphs.
- O: Random oracle model.

Comparison with Previous Algorithms

Year	Author(s)	Space	Remarks
2001	Bar-Yossef et al. [1]	$\tilde{O}(\ln(\delta^{-1})(\varepsilon^{-2} + b))$	M. H.
2007	HLL [4]	$O(\ln(\delta^{-1}\varepsilon^{-2} \ln b)$	O. M. H.
2010	Kane et al. [5]	$O(\ln(\delta^-1)(\varepsilon^{-2} + b))$	H.
2020	Błasiok [2]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E.
2022	Karayel [6]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E. M.
2023	Chakraborty et al. [3]	$O(\varepsilon^{-2} \ln(\delta^{-1} l) b)$	-

Abbreviations:

- M: Supports merging of sketches.
- H: Relies on hash functions.
- E: Relies on expander graphs.
- O: Random oracle model.

Comparison with Previous Algorithms

Year	Author(s)	Space	Remarks
2001	Bar-Yossef et al. [1]	$\tilde{O}(\ln(\delta^{-1})(\varepsilon^{-2}+b))$	M. H.
2007	HLL [4]	$O(\ln(\delta^{-1}\varepsilon^{-2}\ln b)$	O. M. H.
2010	Kane et al. [5]	$O(\ln(\delta^{-}1)(\varepsilon^{-2}+b))$	H.
2020	Błasiok [2]	$O(\varepsilon^{-2}\ln(\delta^{-1})+b)$	H. E.
2022	Karayel [6]	$O(\varepsilon^{-2}\ln(\delta^{-1})+b)$	H. E. M.
2023	Chakraborty et al. [3]	$O(\varepsilon^{-2}\ln(\delta^{-1}l)b)$	-

Abbreviations:

- M: Supports merging of sketches.
- H: Relies on hash functions.
- E: Relies on expander graphs.
- O: Random oracle model.

Comparison with Previous Algorithms

Year	Author(s)	Space	Remarks
2001	Bar-Yossef et al. [1]	$\tilde{O}(\ln(\delta^{-1})(\varepsilon^{-2} + b))$	M. H.
2007	HLL [4]	$O(\ln(\delta^{-1}\varepsilon^{-2} \ln b)$	O. M. H.
2010	Kane et al. [5]	$O(\ln(\delta^{-}1)(\varepsilon^{-2} + b))$	H.
2020	Błasiok [2]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E.
2022	Karayel [6]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E. M.
2023	Chakraborty et al. [3]	$O(\varepsilon^{-2} \ln(\delta^{-1} l) b)$	-

Abbreviations:

- M: Supports merging of sketches.
- H: Relies on hash functions.
- E: Relies on expander graphs.
- O: Random oracle model.

Comparison with Previous Algorithms

Year	Author(s)	Space	Remarks
2001	Bar-Yossef et al. [1]	$\tilde{O}(\ln(\delta^{-1})(\varepsilon^{-2} + b))$	M. H.
2007	HLL [4]	$O(\ln(\delta^{-1}\varepsilon^{-2} \ln b)$	O. M. H.
2010	Kane et al. [5]	$O(\ln(\delta^{-}1)(\varepsilon^{-2} + b))$	H.
2020	Błasiok [2]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E.
2022	Karayel [6]	$O(\varepsilon^{-2} \ln(\delta^{-1}) + b)$	H. E. M.
2023	Chakraborty et al. [3]	$O(\varepsilon^{-2} \ln(\delta^{-1} l) b)$	-

Abbreviations:

- M: Supports merging of sketches.
- H: Relies on hash functions.
- E: Relies on expander graphs.
- O: Random oracle model.

Table of Contents

1 Introduction to the CVM algorithm

2 Summary of Results

3 Verification using Functional Probabilistic Invariants

4 Sketch of the Original Proof

Summary of Results

Summary of Results

- Verification of Correctness

Summary of Results

- Verification of Correctness
- Discovery of a new more succinct proof

Summary of Results

- Verification of Correctness
- Discovery of a new more succinct proof, which is based on a *new* proof technique for randomized algorithms.

Summary of Results

- Verification of Correctness
- Discovery of a new more succinct proof, which is based on a *new* proof technique for randomized algorithms.
- We call it: *Functional Probabilistic Invariants*

Summary of Results

- Verification of Correctness using the original and new method
- Discovery of a new more succinct proof (1003 lines vs. 2634 lines), which is based on a *new* proof technique for randomized algorithms.
- We call it: *Functional Probabilistic Invariants*

Summary of Results

- Verification of Correctness using the original and new method
- Discovery of a new more succinct proof (1003 lines vs. 2634 lines), which is based on a *new* proof technique for randomized algorithms.
- We call it: *Functional Probabilistic Invariants*
- Discovery of an unbiased total variant of the algorithm. (Application of our new technique.)

Table of Contents

1 Introduction to the CVM algorithm

2 Summary of Results

3 Verification using Functional Probabilistic Invariants

4 Sketch of the Original Proof

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**
    $b \xleftarrow{\$} \mathrm{Ber}(p)$
    if b **then**
        $\chi \leftarrow \chi \cup \{a_i\}$
    else
        $\chi \leftarrow \chi - \{a_i\}$
    if $|\chi| = n$ **then**
        $\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$
        $p \leftarrow \frac{p}{2}$
    if $|\chi| = n$ **then return** $\perp$
return $\frac{|\chi|}{p}$

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**
  $b \xleftarrow{\$} \mathrm{Ber}(p)$
  if b **then**
    $\chi \leftarrow \chi \cup \{a_i\}$
  else
    $\chi \leftarrow \chi - \{a_i\}$

Step 1

  if $|\chi| = n$ **then**
    $\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$
    $p \leftarrow \frac{p}{2}$
  if $|\chi| = n$ **then return** $\perp$
return $\frac{|\chi|}{p}$

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$

for $i \leftarrow 1$ to l **do**

$b \xleftarrow{\$} \mathrm{Ber}(p)$

if b **then**

$\chi \leftarrow \chi \cup \{a_i\}$

else

$\chi \leftarrow \chi - \{a_i\}$

Step 1

if $|\chi| = n$ **then**

$\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$

$p \leftarrow \frac{p}{2}$

Step 2

if $|\chi| = n$ **then return** $\perp$

return $\frac{|\chi|}{p}$

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**

$b \xleftarrow{\$} \mathrm{Ber}(p)$
if b **then**
$\chi \leftarrow \chi \cup \{a_i\}$
else
$\chi \leftarrow \chi - \{a_i\}$

Step 1

if $|\chi| = n$ **then**
$\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$
$p \leftarrow \frac{p}{2}$

Step 2

~~**if** $|\chi| = n$ **then return** $\perp$~~

return $\frac{|\chi|}{p}$

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$

for $i \leftarrow 1$ to l **do**

$b \xleftarrow{\$} \mathrm{Ber}(p)$

if b **then**

$\chi \leftarrow \chi \cup \{a_i\}$

else

$\chi \leftarrow \chi - \{a_i\}$

Step 1

if $|\chi| = n$ **then**

$\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$

$p \leftarrow \frac{p}{2}$

Step 2

~~**if** $|\chi| = n$ **then return** $\perp$~~

return $\frac{|\chi|}{p}$

Estimate

Giry Monad

We can represent randomized algorithms using the Giry monad:

- Primitive random operations, e.g., $\mathrm{Ber}(p)$
- Return operation $\mathrm{return}\, x$
- Sequential compositon $m \gg= f$

Giry Monad

We can represent randomized algorithms using the Giry monad:

- Primitive random operations, e.g., $\mathrm{Ber}(p)$
- Return operation $\mathrm{return}\, x$
- Sequential compositon $m \ggg= f$

Fact

For an event E:

$$\mathcal{P}_{m \ggg= f}(E) \quad = \quad \int_m P_{f(x)}(E)\, dx$$

Giry Monad

We can represent randomized algorithms using the Giry monad:

- Primitive random operations, e.g., $\mathrm{Ber}(p)$
- Return operation $\mathrm{return}\, x$
- Sequential compositon $m \ggg f$

Fact

For an event E:

$$\mathcal{P}_{m \ggg f}(E) \quad = \quad \int_m P_{f(x)}(E)\, dx = \sum_x P_{f(x)}(E) P_m(x)$$

Giry Monad

We can represent randomized algorithms using the Giry monad:

- Primitive random operations, e.g., $\mathrm{Ber}(p)$
- Return operation $\mathrm{return}\, x$
- Sequential compositon $m \ggg f$

Fact

For an event E:

$$
\begin{aligned}
\mathcal{P}_{m \ggg f}(E) &= \int_m P_{f(x)}(E)\, dx = \sum_x P_{f(x)}(E) P_m(x) \\
\mathrm{E}_{m \ggg f}[g] &= \int_m \mathrm{E}_{f(x)}[g]\, dx
\end{aligned}
$$

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**

$b \xleftarrow{\$} \mathrm{Ber}(p)$
if b **then**
$\chi \leftarrow \chi \cup \{a_i\}$
else
$\chi \leftarrow \chi - \{a_i\}$

Step 1

if $|\chi| = n$ **then**
$\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$
$p \leftarrow \frac{p}{2}$

Step 2

~~**if** $|\chi| = n$ **then return** $\perp$~~

return $\frac{|\chi|}{p}$

Estimate

Our algorithm in monadic notation

$$
\begin{aligned}
\mathrm{init} \quad &\gg\!= \quad \mathrm{step}_1\, a_1 \gg\!= \mathrm{step}_2 \gg\!= \mathrm{step}_1\, a_2 \gg\!= \mathrm{step}_2 \cdots \\
&\gg\!= \quad \mathrm{step}_1\, a_l \gg\!= \mathrm{step}_2 \gg\!= \mathrm{estimate} \\
\mathrm{init} \quad &= \quad \mathrm{return}\,(1, \emptyset)
\end{aligned}
$$

Our algorithm in monadic notation

$$\begin{aligned}
\text{init} \quad &\gg\!= \quad \text{step}_1\, a_1 \gg\!= \text{step}_2 \gg\!= \text{step}_1\, a_2 \gg\!= \text{step}_2 \cdots \\
&\gg\!= \quad \text{step}_1\, a_l \gg\!= \text{step}_2 \gg\!= \text{estimate} \\
\text{init} \quad &= \quad \text{return}\,(1, \emptyset)
\end{aligned}$$

Example invariant:

$$\mathrm{E}\left[\frac{\mathrm{I}(s \in \chi)}{p}\right] = 1$$

for all s that are present in the stream.

Verifying the invariant: Induction step for Step 2

Assumption

$\mathrm{E}_m \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$

Goal

$L := \mathrm{E}_{m \gg= \mathrm{step}_2} \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$

$$L = \mathrm{E}_{m \gg= \mathrm{step}_2} \left[\frac{\mathrm{I}(s \in \chi)}{p} \right]$$

Verifying the invariant: Induction step for Step 2

Assumption

$$\mathrm{E}_m \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$$

Goal

$$L := \mathrm{E}_{m \gg= \mathrm{step}_2} \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$$

$$L = \int_m \left(\textbf{if } |\chi| = n \textbf{ then } \int_{\mathrm{subs.}(\chi)} \frac{\mathrm{I}(s \in \tau)}{p/2} \, \mathrm{d}\tau \textbf{ else } \frac{\mathrm{I}(s \in \chi)}{p} \right) \mathrm{d}\sigma$$

Verifying the invariant: Induction step for Step 2

Assumption

$$\mathrm{E}_m\left[\frac{\mathrm{I}(s\in\chi)}{p}\right] = 1$$

Goal

$$L := \mathrm{E}_{m \ggg= \mathrm{step}_2}\left[\frac{\mathrm{I}(s\in\chi)}{p}\right] = 1$$

$$L = \int_m \left(\textbf{if } |\chi| = n \textbf{ then } \frac{2}{p}\int_{\mathrm{subs.}(\chi)} \mathrm{I}(s \in \tau)\,\mathrm{d}\tau \textbf{ else } \frac{\mathrm{I}(s \in \chi)}{p}\right)\,\mathrm{d}\sigma$$

Verifying the invariant: Induction step for Step 2

Assumption

$\mathrm{E}_m \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$

Goal

$L := \mathrm{E}_{m \gg= \mathrm{step}_2} \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$

$$L = \int_m \left(\textbf{if } |\chi| = n \textbf{ then } \frac{2}{p} \frac{\mathrm{I}(s \in \chi)}{2} \textbf{ else } \frac{\mathrm{I}(s \in \chi)}{p} \right) \, \mathrm{d}\sigma$$

Verifying the invariant: Induction step for Step 2

Assumption

$\mathrm{E}_m\left[\frac{\mathrm{I}(s\in\chi)}{p}\right] = 1$

Goal

$L := \mathrm{E}_{m \gg= \mathrm{step}_2}\left[\frac{\mathrm{I}(s\in\chi)}{p}\right] = 1$

$$L = \int_m \left(\textbf{if } |\chi| = n \textbf{ then } \frac{\mathrm{I}(s\in\chi)}{p} \textbf{ else } \frac{\mathrm{I}(s\in\chi)}{p}\right) \mathrm{d}\sigma$$

Verifying the invariant: Induction step for Step 2

Assumption

$$\mathrm{E}_m\left[\frac{\mathrm{I}(s{\in}\chi)}{p}\right] = 1$$

Goal

$$L := \mathrm{E}_{m \ggg= \mathrm{step}_2}\left[\frac{\mathrm{I}(s{\in}\chi)}{p}\right] = 1$$

$$L = \int_m \frac{\mathrm{I}(s \in \chi)}{p}\,\mathrm{d}\sigma$$

Verifying the invariant: Induction step for Step 2

Assumption

$\mathrm{E}_m \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$

Goal

$L := \mathrm{E}_{m \ggg= \mathrm{step}_2} \left[\frac{\mathrm{I}(s \in \chi)}{p} \right] = 1$

$$L = 1$$

Observations

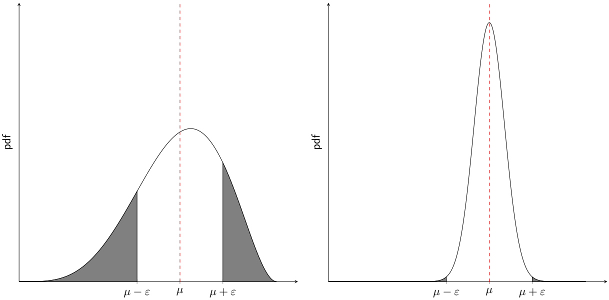
- Expected result is the count of distinct elements.

Observations

- Expected result is the count of distinct elements.
- We want to establish concentration

Observations

- Expected result is the count of distinct elements.
- We want to establish concentration



Observations

- Expected result is the count of distinct elements.
- We want to establish concentration
- Can extend the above recursive analysis to more complex expressions:

Observations

- Expected result is the count of distinct elements.
- We want to establish concentration
- Can extend the above recursive analysis to more complex expressions:

$$\mathrm{E}\left[h\left(\frac{\mathrm{I}(s \in \chi)}{p}\right)\right] \leq h(1)$$

for every non-negative concave function h and stream elements s.

Observations

- Expected result is the count of distinct elements.
- We want to establish concentration
- Can extend the above recursive analysis to more complex expressions:

$$\mathrm{E}\left[\prod_{s \in S} h\left(\frac{\mathrm{I}(s \in \chi)}{p}\right)\right] \leq h(1)^{|S|}$$

for every non-negative concave function h and subset of stream elements S.

Observations

- Expected result is the count of distinct elements.
- We want to establish concentration
- Can extend the above recursive analysis to more complex expressions:

$$\mathrm{E}\left[\prod_{s\in S} h\left(\frac{\mathrm{I}(s\in\chi)}{p}\right)\right] \leq h(1)^{|S|}$$

  for every non-negative concave function h and subset of stream elements S.
- $\Rightarrow$ Approximation Guarantee

$$\mathcal{P}\left(|X - |A|| \geq \varepsilon|A|\right) \leq \delta$$

Observations

- Expected result is the count of distinct elements.
- We want to establish concentration
- Can extend the above recursive analysis to more complex expressions:

$$\mathrm{E}\left[\prod_{s \in S} h\left(\frac{\mathrm{I}(s \in \chi)}{p}\right)\right] \leq h(1)^{|S|}$$

  for every non-negative concave function h and subset of stream elements S.
- $\Rightarrow^*$ Approximation Guarantee

$$\mathcal{P}\left(|X - |A|| \geq \varepsilon |A|\right) \leq \delta$$

- More details in the paper.

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)
- *As far as we can tell*, this technique is new.

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)
- *As far as we can tell*, this technique is new.
- The new proof opens up the design-space of the algorithm.

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)
- *As far as we can tell*, this technique is new.
- The new proof opens up the design-space of the algorithm.
- For example we can select the subsampling ratio dynamically.

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)
- *As far as we can tell*, this technique is new.
- The new proof opens up the design-space of the algorithm.

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)
- *As far as we can tell*, this technique is new.
- The new proof opens up the design-space of the algorithm.
- More interestingly: It is possible to use a different subsampling operation.

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)
- *As far as we can tell*, this technique is new.
- The new proof opens up the design-space of the algorithm.
- More interestingly: It is possible to use a different subsampling operation.
- Select a random nf-subset (where $\frac{1}{2} \leq f < 1$, $nf \in \mathbb{Z}$).

Observations II

- The new proof is much shorter than the original proof (1003 lines vs. 2634 lines.)
- *As far as we can tell,* this technique is new.
- The new proof opens up the design-space of the algorithm.
- More interestingly: It is possible to use a different subsampling operation.
- Select a random nf-subset (where $\frac{1}{2} \leq f < 1$, $nf \in \mathbb{Z}$).
- $\Rightarrow$ Unbiased algorithm

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**

$b \xleftarrow{\$} \mathrm{Ber}(p)$
if b **then**
$\chi \leftarrow \chi \cup \{a_i\}$
else
$\chi \leftarrow \chi - \{a_i\}$

Step 1

if $|\chi| = n$ **then**
$\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$
$p \leftarrow \frac{p}{2}$

Step 2

~~**if** $|\chi| = n$ **then return** $\perp$~~

return $\frac{|\chi|}{p}$

Estimate

Another slide with the algorithm

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{3l}{\delta}\right) \right\rceil$
for $i \leftarrow 1$ to l **do**

$b \xleftarrow{\$} \mathrm{Ber}(p)$
if b **then**
$\chi \leftarrow \chi \cup \{a_i\}$
else
$\chi \leftarrow \chi - \{a_i\}$

Step 1

if $|\chi| = n$ **then**
$\chi \xleftarrow{\$} \mathrm{subsample}'(\chi)$
$p \leftarrow \frac{p}{2}$

Step 2

return $\frac{|\chi|}{p}$

Estimate

Table of Contents

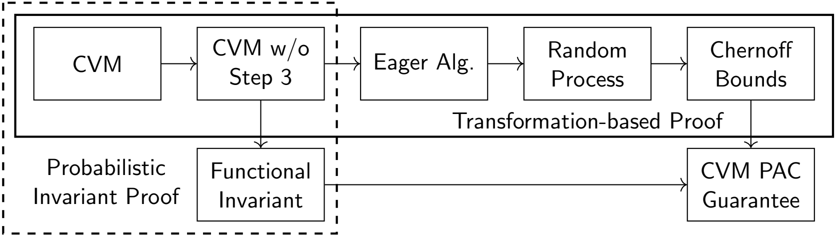
1 Introduction to the CVM algorithm

2 Summary of Results

3 Verification using Functional Probabilistic Invariants

4 Sketch of the Original Proof

Original Proof



CVM

CVM → CVM w/o Step 3 → Eager Alg. → Random Process

$\chi \leftarrow \{\}, p \leftarrow 1, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$

for $i \leftarrow 1$ to l **do**

  $b \xleftarrow{\$} \mathrm{Ber}(p)$

  if b **then**

    $\chi \leftarrow \chi \cup \{a_i\}$

  else

    $\chi \leftarrow \chi - \{a_i\}$

  if $|\chi| = n$ **then**

    $\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$

    $p \leftarrow \frac{p}{2}$

  if $|\chi| = n$ **then return** $\bot$

return $\frac{|\chi|}{p}$

CVM Algorithm II

CVM → **CVM w/o Step 3** → Eager Alg. → Random Process

$\chi \leftarrow \{\}, k \leftarrow 0, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$

for $i \leftarrow 1$ to l **do**

 $b \xleftarrow{\$} \mathrm{Ber}(2^{-k})$

 if b **then**

  $\chi \leftarrow \chi \cup \{a_i\}$

 else

  $\chi \leftarrow \chi - \{a_i\}$

 if $|\chi| = n$ **then**

  $\chi \xleftarrow{\$} \mathrm{subsample}(\chi)$

  $k \leftarrow k + 1$

return $2^k|\chi|$

Eager Algorithm

CVM → CVM w/o Step 3 → **Eager Alg.** → Random Process

$\chi \leftarrow \{\}, k \leftarrow 0, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$

$b[i,j] \xleftarrow{\$} \mathrm{Ber}(1/2)$ for $i, j \in \{1, \cdots, l\}$

for $i \leftarrow 1$ to l **do**

  if $b[i,1] = b[i,2] = \cdots = b[i,k] = 1$ **then**

    $\chi \leftarrow \chi \cup \{a_i\}$

  else

    $\chi \leftarrow \chi - \{a_i\}$

  if $|\chi| = n$ **then**

    $\chi \leftarrow \{a \in \chi \mid b[\mathrm{last\_index}(a), k+1] = 1\}$

    $k \leftarrow k + 1$

return $2^k|\chi|$

Eager Algorithm

$$\boxed{\mathrm{CVM}} \rightarrow \boxed{\text{CVM w/o Step 3}} \rightarrow \boxed{\text{Eager Alg.}} \rightarrow \boxed{\text{Random Process}}$$

$\chi \leftarrow \{\}, k \leftarrow 0, n = \left\lceil \frac{12}{\varepsilon^2} \ln\left(\frac{6l}{\delta}\right) \right\rceil$

$b[i,j] \xleftarrow{\$} \mathrm{Ber}(1/2)$ for $i,j \in \{1, \cdots, l\}$

for $i \leftarrow 1$ to l **do**

  if $b[i,1] = b[i,2] = \cdots = b[i,k] = 1$ **then**

    $\chi \leftarrow \chi \cup \{a_i\}$

  else

    $\chi \leftarrow \chi - \{a_i\}$

  if $|\chi| = n$ **then**

    $\chi \leftarrow \{a \in \chi \mid b[\mathrm{last\_index}(a), k+1] = 1\}$

    $k \leftarrow k + 1$

return $2^k|\chi|$

Random Process

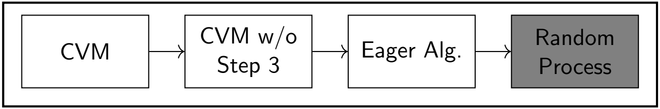
CVM $\rightarrow$ CVM w/o Step 3 $\rightarrow$ Eager Alg. $\rightarrow$ **Random Process**

Observation

The eager algorithm preserves the invariant:

$$s \in \chi \leftrightarrow \text{For all } j < k\text{:}\, b[\mathrm{last\_index}(s), j] = 1$$

Random Process



Observation

The eager algorithm preserves the invariant:

$$s \in \chi \leftrightarrow \text{For all } j < k\text{:}b[\mathrm{last\_index}(s), j] = 1$$

Non-deterministic algorithm:

$b[i,j] \xleftarrow{\$} \mathrm{Ber}(1/2)$ for $i, j \in \{1, \cdots, l\}$
$k \leftarrow \{0, \ldots, k_{\max}\}$ $\triangleright$ Non-deterministc step.
$\chi \leftarrow \{s \in A \mid b[\mathrm{last\_index}(s), j] = 1 \text{ for all } j < k\}$
return $2^k|\chi|$

Conclusion/Summary

- Verification of the CVM Algorithm

Conclusion/Summary

- Verification of the CVM Algorithm
- New Technique for the Verfication of Randomized Algorithms

Conclusion/Summary

- Verification of the CVM Algorithm
- New Technique for the Verfication of Randomized Algorithms
- New unbiased/total variant of the CVM Algorithm

Conclusion/Summary

- Verification of the CVM Algorithm
- New Technique for the Verfication of Randomized Algorithms
- New unbiased/total variant of the CVM Algorithm
- Development of a Library for Negative Association

Conclusion/Summary

- Verification of the CVM Algorithm
- New Technique for the Verfication of Randomized Algorithms
- New unbiased/total variant of the CVM Algorithm
- Development of a Library for Negative Association
  - A generalization of the concept of independence

Conclusion/Summary

- Verification of the CVM Algorithm
- New Technique for the Verfication of Randomized Algorithms
- New unbiased/total variant of the CVM Algorithm
- Development of a Library for Negative Association
    - A generalization of the concept of independence
    - Example: Indicator random variables of an idealized Bloom-Filter

Conclusion/Summary

- Verification of the CVM Algorithm
- New Technique for the Verfication of Randomized Algorithms
- New unbiased/total variant of the CVM Algorithm
- Development of a Library for Negative Association
  - A generalization of the concept of independence
  - Example: Indicator random variables of an idealized Bloom-Filter
- Thank You!

Conclusion/Summary

- Verification of the CVM Algorithm
- New Technique for the Verfication of Randomized Algorithms
- New unbiased/total variant of the CVM Algorithm
- Development of a Library for Negative Association
  - A generalization of the concept of independence
  - Example: Indicator random variables of an idealized Bloom-Filter
- Thank You!
- Artwork:  By vecteezy.com.

References I

Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan.
Counting distinct elements in a data stream.
In José D. P. Rolim and Salil P. Vadhan, editors, *RANDOM*, volume 2483 of *LNCS*, pages 1–10. Springer, 2002.

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

Sourav Chakraborty, N. V. Vinodchandran, and Kuldeep S. Meel.
Distinct elements in streams: An algorithm for the (text) book.
CoRR, abs/2301.10191, 2023.

References II

Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier.
Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm.
Discrete Mathematics & Theoretical Computer Science, DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07), Jan 2007.

Daniel M. Kane, Jelani Nelson, and David P. Woodruff.
An optimal algorithm for the distinct elements problem.
In Jan Paredaens and Dirk Van Gucht, editors, *PODS*, pages 41–52. ACM, 2010.

References III

Emin Karayel.
Formalization of randomized approximation algorithms for frequency moments.
In June Andronick and Leonardo de Moura, editors, *ITP*, volume 237 of *LIPIcs*, pages 21:1–21:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Steve Nadis.
Computer scientists invent an efficient new way to count, 2024.
Accessed: 2025-01-14.