Distinct Elements in Streams and the Klee's Measure Problem

Sourav Chakraborty
Indian Statistical Institute

(Joint work with Kuldeep Meel [2] and N. V. Vinodchandran [3])

[2] University of Toronto
[3] University of Nebraska, Lincoln

with Addendum from: Don Knuth

Corresponding publications: PODS-21, PODS-22, ESA-22

A real life problem

How to keep a count (or approximate count) on the number of DISTINCT persons visiting your webpage?

A real life problem

How to keep a count (or approximate count) on the number of DISTINCT persons visiting your webpage?

Resource Minimization Objectives:

- Should not use too much space
- Should not take too much time to update your database

Data Streams

Goal: Computation over a stream of items

- An item of the stream arrives & stored in temporary memory.
- The item is accessed using the allowed set of operations and some of the information is stored in a working space.
- The item then disappears and the next item arrives.

Computational Task : Estimate/Compute some interesting functions over the entire data stream.

Data Streams

Goal: Computation over a stream of items

- An item of the stream arrives & stored in temporary memory.
- The item is accessed using the allowed set of operations and some of the information is stored in a working space.
- The item then disappears and the next item arrives.

Computational Task : Estimate/Compute some interesting functions over the entire data stream.

Resource Minimization Objectives:

- Space complexity - The (worst case) amount of space required.
- Update-time complexity - The (worst case) amount of time required to process any item.

Set Streams: When the items are sets

Union Size Estimation

Given a stream of sets $S_1, \ldots, S_m$, all $S_i \subseteq \Omega$ and two parameters ε, δ
Output an estimate $\mathcal{E}$ such that with probability $\geq (1 - \delta)$

$$(1 - \varepsilon) \left| \bigcup_{i=1}^{m} S_i \right| \leq \mathcal{E} \leq (1 + \varepsilon) \left| \bigcup_{i=1}^{m} S_i \right|$$

Set Streams: When the items are sets

Union Size Estimation

Given a stream of sets $S_1, \ldots, S_m$, all $S_i \subseteq \Omega$ and two parameters ε, δ
Output an estimate $\mathcal{E}$ such that with probability $\geq (1-\delta)$

$$(1-\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right| \leq \mathcal{E} \leq (1+\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right|$$

The special case when $|S_i| = 1$ is the well-investigated Distinct Elements Problem Estimation Problem.

Set Streams: When the items are sets

Union Size Estimation

Given a stream of sets $S_1, \ldots, S_m$, all $S_i \subseteq \Omega$ and two parameters ε, δ
Output an estimate $\mathcal{E}$ such that with probability $\geq (1 - \delta)$

$$(1-\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right| \leq \mathcal{E} \leq (1+\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right|$$

The special case when $|S_i| = 1$ is the well-investigated Distinct Elements Problem Estimation Problem.

- How are the sets given?
- How are the sets accessed?

Set Streams: When the items are sets

Union Size Estimation

Given a stream of sets $S_1, \ldots, S_m$, all $S_i \subseteq \Omega$ and two parameters ε, δ
Output an estimate $\mathcal{E}$ such that with probability $\geq (1-\delta)$

$$(1-\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right| \leq \mathcal{E} \leq (1+\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right|$$

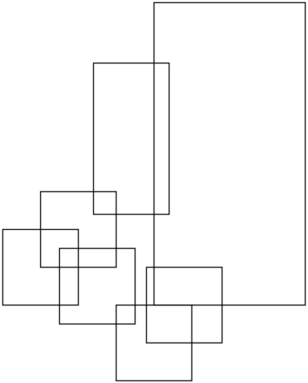
The special case when $|S_i| = 1$ is the well-investigated Distinct Elements Problem Estimation Problem.

- How are the sets given?
- How are the sets accessed?

Lets come back to these formalizations later.

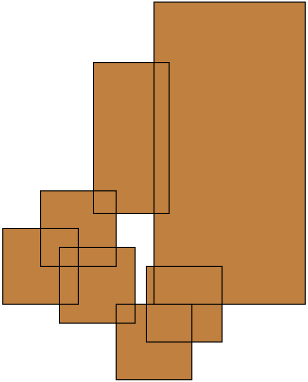
Example: Klee's Measure Problem

- Estimate the union of axis-parallel rectangles in $\mathbb{R}^d$.



Example: Klee's Measure Problem

- Estimate the union of axis-parallel rectangles in $\mathbb{R}^d$.



Example: Klee's Measure Problem

- Estimate the union of axis-parallel rectangles in $\mathbb{R}^d$.
(Discrete version: so count the number of integer points)

- Every $S_i = [a_{i,1}, b_{i,1}] \times [a_{i,2}, b_{i,2}] \ldots [a_{i,d}, b_{i,d}]$ where $a_{i,j} \leq \Delta$; $b_{i,j} \leq \Delta$

- Goal: Output an estimate $\mathcal{E}$ such that with probability $\geq (1-\delta)$

$$(1-\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right| \leq \mathcal{E} \leq (1+\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right|$$

Example: Klee's Measure Problem

- Estimate the union of axis-parallel rectangles in $\mathbb{R}^d$.
  (Discrete version: so count the number of integer points)

- Every $S_i = [a_{i,1}, b_{i,1}] \times [a_{i,2}, b_{i,2}] \ldots [a_{i,d}, b_{i,d}]$ where $a_{i,j} \leq \Delta$; $b_{i,j} \leq \Delta$

- Goal: Output an estimate $\mathcal{E}$ such that with probability $\geq (1-\delta)$

$$(1-\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right| \leq \mathcal{E} \leq (1+\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right|$$

- Some ways to access the sets?
  - Know the size of S_i: $\mathcal{O}(d \log |\Delta|)$
  - Sample uniformly at random elements from S_i: $\mathcal{O}(d \log |\Delta|)$
  - For any element $x \in \mathbb{R}^d$, check if $x \in S_i$: $\mathcal{O}(d \log |\Delta|)$

Example: Klee's Measure Problem

- Estimate the union of axis-parallel rectangles in $\mathbb{R}^d$.
  (Discrete version: so count the number of integer points)

- Every $S_i = [a_{i,1}, b_{i,1}] \times [a_{i,2}, b_{i,2}] \ldots [a_{i,d}, b_{i,d}]$ where $a_{i,j} \leq \Delta$; $b_{i,j} \leq \Delta$

- Goal: Output an estimate $\mathcal{E}$ such that with probability $\geq (1-\delta)$

$$(1-\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right| \leq \mathcal{E} \leq (1+\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right|$$

- Some ways to access the sets?
  - Know the size of S_i: $\mathcal{O}(d \log |\Delta|)$
  - Sample uniformly at random elements from S_i: $\mathcal{O}(d \log |\Delta|)$
  - For any element $x \in \mathbb{R}^d$, check if $x \in S_i$: $\mathcal{O}(d \log |\Delta|)$

- Lot of work done, most recently by Tirthapura-Woodruff (2012), Vahrenhold (2007), Indyk-Woodruff (2005)

- **Open Problem**: Show Klee's Measure Problem can be done with space and update-time complexity $\tilde{O}(poly(d, \log |\Delta|))$.

Special Case of Union Estimation: Distinct Elements Problem

Input A data stream $\mathcal{D} =< a_1, a_2, \ldots a_m >$ where $a_i \in [n]$

Output Compute the number of Distinct elements in $\mathcal{D}$. Formally,
$F = |\{a_1, a_2, \ldots a_m\}|$

Special Case of Union Estimation: Distinct Elements Problem

Input A data stream $\mathcal{D} =< a_1, a_2, \ldots a_m >$ where $a_i \in [n]$

Output Compute the number of Distinct elements in $\mathcal{D}$. Formally,

$F = |\{a_1, a_2, \ldots a_m\}|$

Example: $\mathcal{D} =< 1, 1, 2, 1, 4, 1, 1, 1 >$ $\quad F = 3$

Special Case of Union Estimation: Distinct Elements Problem

Input A data stream $\mathcal{D} =< a_1, a_2, \ldots a_m >$ where $a_i \in [n]$

Output Compute the number of Distinct elements in $\mathcal{D}$. Formally, $F = |\{a_1, a_2, \ldots a_m\}|$

Example: $\mathcal{D} =< 1, 1, 2, 1, 4, 1, 1, 1 >$ $\quad F = 3$

- Our focus: (ε, δ)-approximation

$$\Pr\left[(1-\varepsilon)F \leq \text{Est} \leq (1+\varepsilon)F\right] \geq 1-\delta$$

Special Case of Union Estimation: Distinct Elements Problem

Input A data stream $\mathcal{D} =< a_1, a_2, \ldots a_m >$ where $a_i \in [n]$

Output Compute the number of Distinct elements in $\mathcal{D}$. Formally,
$F = |\{a_1, a_2, \ldots a_m\}|$

Example: $\mathcal{D} =< 1, 1, 2, 1, 4, 1, 1, 1 >$ $\quad F = 3$

- Our focus: (ε, δ)-approximation

$$\Pr[(1-\varepsilon)F \leq \text{Est} \leq (1+\varepsilon)F] \geq 1-\delta$$

Naive Solution Maintain a large hash table.
Worst-case space complexity of $\mathcal{O}(n)$

Objective Optimize space and update time complexity
Update Time: Time to process each element of the stream

Rich History of work

- Flajolet and Martin (1985), Alon, Matias, and Szegedy (1996), Bar-Yossef, Jayram, Kumar, Sivakumar and Trevisan (2001), . . ., Kane, Nelson, and Woodruff (2010), . . ., Błasiok (2019), . . .

Crowning Jewel Optimal (time and) space complexity: $O\left(\log n + \frac{1}{\varepsilon^2} \cdot \log 1/\delta\right)$

Rich History of work

- Flajolet and Martin (1985), Alon, Matias, and Szegedy (1996), Bar-Yossef, Jayram, Kumar, Sivakumar and Trevisan (2001), ..., Kane, Nelson, and Woodruff (2010), ..., Błasiok (2019), ...

Crowning Jewel Optimal (time and) space complexity: $O\left(\log n + \frac{1}{\varepsilon^2} \cdot \log 1/\delta\right)$

Limitations Practically efficient algorithms are beyond graduate classroom
Theoretically efficient algorithms can be taught in graduate classroom but don't work in practice

Rich History of work

- Flajolet and Martin (1985), Alon, Matias, and Szegedy (1996), Bar-Yossef, Jayram, Kumar, Sivakumar and Trevisan (2001), . . ., Kane, Nelson, and Woodruff (2010), . . ., Błasiok (2019), . . .

Crowning Jewel Optimal (time and) space complexity: $O\left(\log n + \frac{1}{\varepsilon^2} \cdot \log 1/\delta\right)$

Limitations Practically efficient algorithms are beyond graduate classroom
Theoretically efficient algorithms can be taught in graduate classroom but don't work in practice

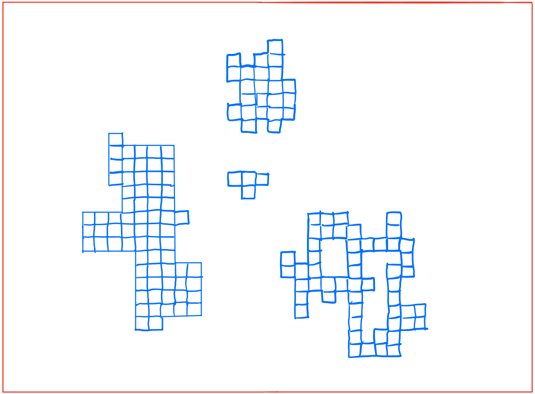
Theorem (C-Vinodchandran-Meel (ESA'22))

A simple algorithm with time and space complexity of $O(\frac{1}{\varepsilon^2} \cdot \log n \cdot (\log m + \log 1/\delta))$.

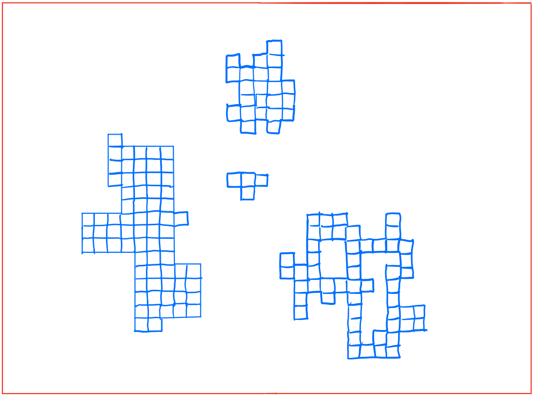
Remark: The description and algorithm requires only basic data structures and knowledge of elementary probability theory (Chernoff Bound), and can be easily taught in an undergraduate course, and the algorithm is practically efficient.

The paper is just five pages (including abstract and bibliographical remarks)
Knuth (May 23): " Ever since I saw it, a few days ago, I've been unable to resist trying to explain the ideas to just about everybody I meet."

How to estimate the volume of an object?

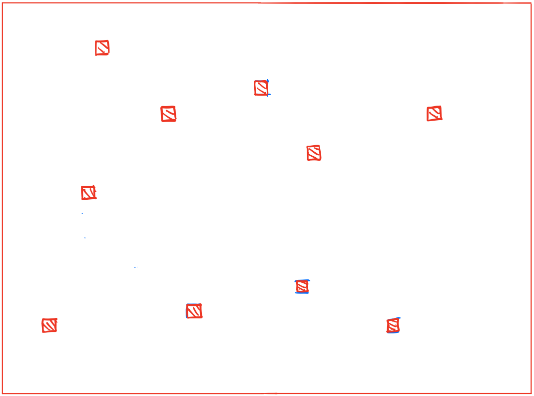


How to estimate the volume of an object?



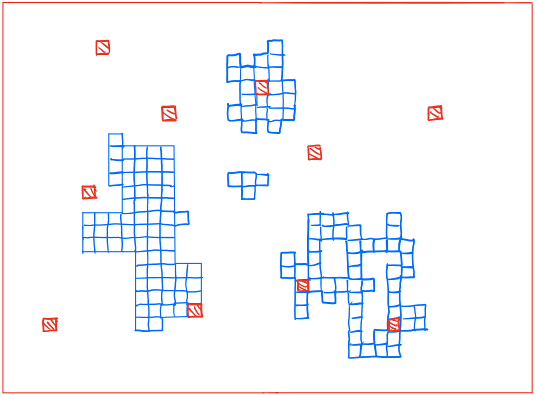
- Pick random points from the universe and measure the fraction of points falling inside the region.

How to estimate the volume of an object? Natural Approach 1



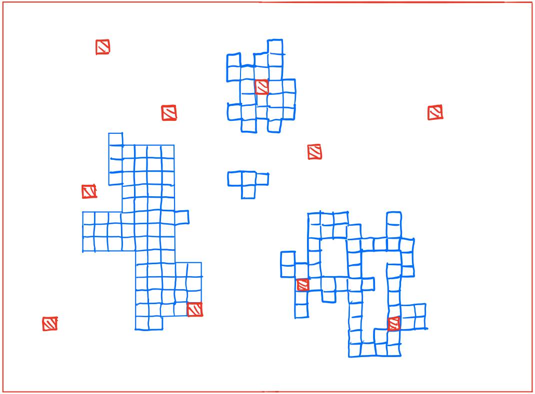
- Pick random points from the universe and measure the fraction of points falling inside the region.

How to estimate the volume of an object? Natural Approach 1



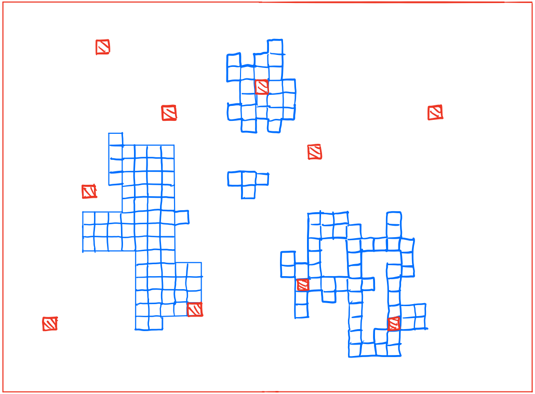
- Pick random points from the universe and measure the fraction of points falling inside the region.

How to estimate the volume of an object? Natural Approach 1



- Pick random points from the universe and measure the fraction of points falling inside the region.
- CATCH: Works when the size of region is large compared to universe.

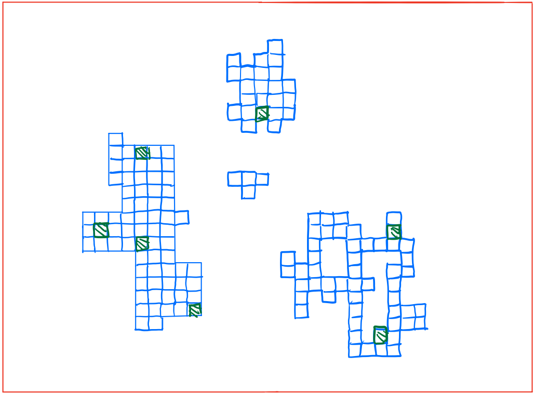
How to estimate the volume of an object? Natural Approach 1



- Pick random points from the universe and measure the fraction of points falling inside the region.
- CATCH: Works when the size of region is large compared to universe.

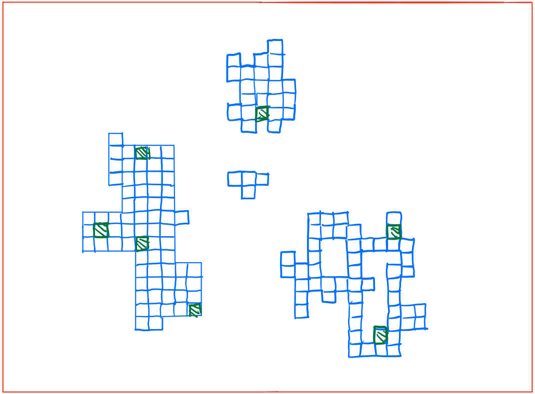
NOT GOOD for our purpose.

How to estimate the volume of an object? Natural Approach 2



- Pick every element in the region independently with probability p.

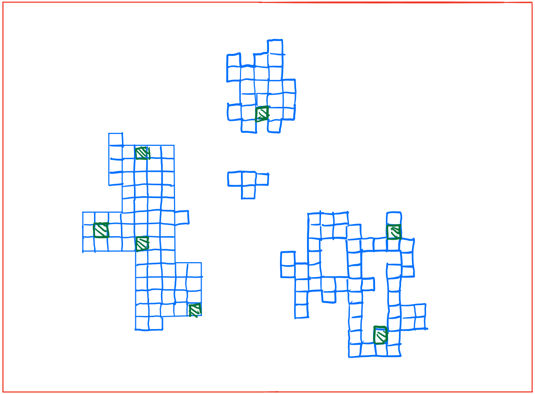
How to estimate the volume of an object? Natural Approach 2



- Pick every element in the region independently with probability p.

  The expected number of elements picks $= p \times$ (size of the region).

How to estimate the volume of an object? Natural Approach 2



- Pick every element in the region independently with probability p.

  The expected number of elements picks $= p \times$ (size of the region).
  So the estimate can be $=$ (number of elements picked) $\times \frac{1}{p}$

  The estimation is good if $p \geq \frac{1}{\epsilon^2(\text{size of the region})}$

Core Idea

Theorem (C-Vinodchandran-Meel (ESA'22))

A simple *algorithm with time and space complexity of* $O(\frac{1}{\varepsilon^2} \cdot \log n \cdot (\log m + \log 1/\delta))$.

Core Idea If we pick every ball in a bucket with probability p in our bucket and we end up k balls in the bucket, then $\frac{k}{p}$ is a good estimate of the number of balls in the bucket.

Key Ingredients - I

Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

---

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;

2: **for** $i = 1$ to m **do**

3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

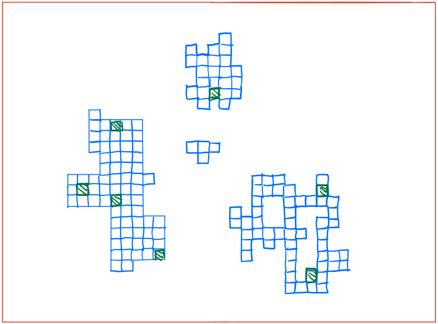
Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

Initialize $\mathcal{B} \leftarrow \emptyset$;

for $i = 1$ to m **do**

  With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$



Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

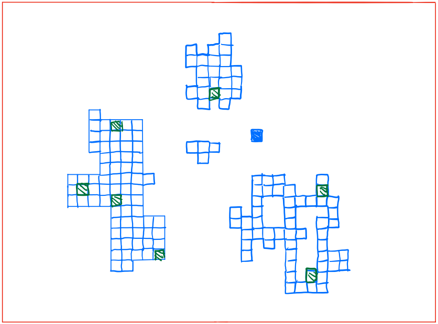
Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;

2: **for** $i = 1$ to m **do**

3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$



Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

---

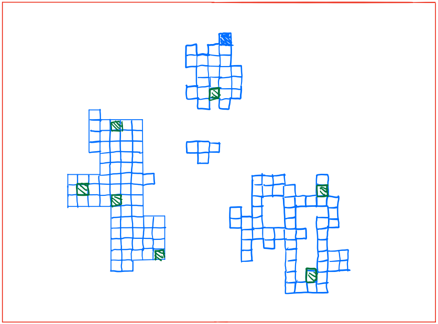
Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;

2: **for** $i = 1$ to m **do**

3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$



Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Algorithm NaiveSampler

```
Input Stream D = ⟨a_1, a_2, ..., a_m⟩, p
Initialize B ← ∅;
for i = 1 to m do
    With probability p, B ← B ∪ {a_i}
```

Example: $\mathcal{D} =< 1, 1, 2, 1, 4, 1, 1, 1 >$

Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

---

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;
2: **for** $i = 1$ to m **do**
3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Example: $\mathcal{D} =< 1, 1, 2, 1, 4, 1, 1, 1 >$

Challenge Elements that repeat more often are more likely to be sampled

Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m} \{a_i\}$ independently with prob. p

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

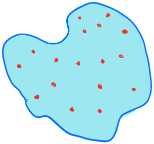
1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;

2: **for** $i = 1$ to m **do**

3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Example: $\mathcal{D} =< 1, 1, 2, 1, 4, 1, 1, 1 >$

Challenge Elements that repeat more often are more likely to be sampled



Key Ingredients - I

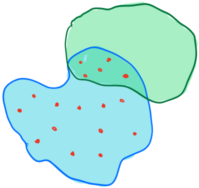
Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

---

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p
1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;
2: **for** $i = 1$ to m **do**
3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Challenge Elements that repeat more often are more likely to be sampled



Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

---

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

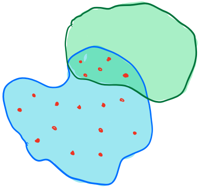
Initialize $\mathcal{B} \leftarrow \emptyset$;

for $i = 1$ to m **do**

  With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Challenge Elements that repeat more often are more likely to be sampled

Solution **Throw it Away** is All You Need



Key Ingredients - I

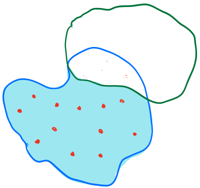
Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;
2: **for** $i = 1$ to m **do**
3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Challenge Elements that repeat more often are more likely to be sampled
Solution **Throw it Away** is All You Need



Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

---

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

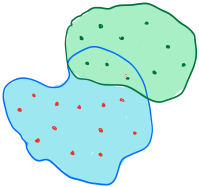
1: **Initialize** $\mathcal{B} \leftarrow \emptyset$;

2: **for** $i = 1$ to m **do**

3: With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Challenge Elements that repeat more often are more likely to be sampled

Solution **Throw it Away** is All You Need



Key Ingredients - I

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Algorithm NaiveSampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

Initialize $\mathcal{B} \leftarrow \emptyset$;
for $i = 1$ to m **do**
  With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Challenge Elements that repeat more often are more likely to be sampled
Solution **Throw it Away** is All You Need

Algorithm Sampler

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, p

Initialize $\mathcal{B} \leftarrow \emptyset$;
for $i = 1$ to m **do**
  $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
  With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

Observation Whether an element $x \in \mathcal{B}$ or not only depends on whether x was picked when it appeared last time

Key Ingredients - II

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Idea 2 Determine *just the right* value of p?

- Too large p, $|\mathcal{B}|$ is too large
- Too small p, $\frac{|\mathcal{B}|}{p}$ is not a good estimator

Key Ingredients - II

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Idea 2 Determine *just the right* value of p?

- Too large p, $|\mathcal{B}|$ is too large
- Too small p, $\frac{|\mathcal{B}|}{p}$ is not a good estimator

Our Idea: SAMPLE the SAMPLED elements, when needed

Key Ingredients - II

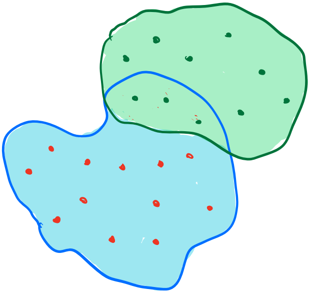
Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Idea 2 Determine *just the right* value of p?

- Too large p, $|\mathcal{B}|$ is too large
- Too small p, $\frac{|\mathcal{B}|}{p}$ is not a good estimator

Our Idea: SAMPLE the SAMPLED elements, when needed

Currently: Every element is picked independently with probability p.



Key Ingredients - II

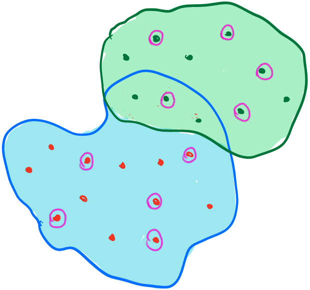
Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Idea 2 Determine *just the right* value of p?

- Too large p, $|\mathcal{B}|$ is too large
- Too small p, $\frac{|\mathcal{B}|}{p}$ is not a good estimator

Our Idea: SAMPLE the SAMPLED elements, when needed

For each sampled point pick it with probability $\frac{1}{2}$.



Key Ingredients - II

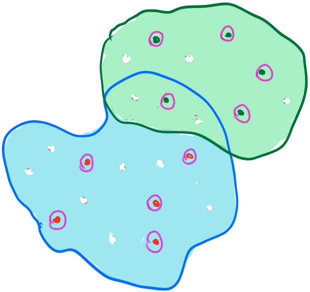
Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Idea 2 Determine *just the right* value of p?

- Too large p, $|\mathcal{B}|$ is too large
- Too small p, $\frac{|\mathcal{B}|}{p}$ is not a good estimator

Our Idea: SAMPLE the SAMPLED elements, when needed

Effectively: Every point is picked with probability $\frac{p}{2}$.



Key Ingredients - II

Idea 1 Sample every element of $\bigcup_{i=1}^{i=m}\{a_i\}$ independently with prob. p

Idea 2 Determine *just the right* value of p?

- Too large p, $|\mathcal{B}|$ is too large
- Too small p, $\frac{|\mathcal{B}|}{p}$ is not a good estimator

---

Algorithm Adaptive Estimator

```
Input Stream D = ⟨a_1, a_2, ..., a_m⟩, ε, δ
Initialize B ← ∅; thresh ← (12/ε^2) log(8m/δ); p ← 1
for i = 1 to m do
    B ← B \ {a_i}
    With probability p, B ← B ∪ {a_i}
    if |B| = thresh then
        Throw away each element of B with probability 1/2
        p ← p/2
Output |B|/p
```

Input Stream $\mathcal{D} = \langle a_1, a_2, \ldots, a_m \rangle$, ε, δ

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2}\log(\frac{8m}{\delta})$; $p \leftarrow 1$

for $i = 1$ to m **do**

  $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$

  With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$

  if $|\mathcal{B}| =$ thresh **then**

    Throw away each element of $\mathcal{B}$ with probability $\frac{1}{2}$

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

Output $\frac{|\mathcal{B}|}{p}$

Algorithm Adaptive Estimator

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2} \log(\frac{8m}{\delta})$; $p \leftarrow 1$
for $i = 1$ to m **do**
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
    With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$
    while $|\mathcal{B}| =$ thresh **do**
        For each element of $\mathcal{B}$ throw away the element independently with probability $\frac{1}{2}$
        $p \leftarrow \frac{p}{2}$
Output $\frac{|\mathcal{B}|}{p}$

Algorithm Adaptive Estimator

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2} \log(\frac{8m}{\delta})$; $p \leftarrow 1$
for $i = 1$ to m **do**
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
    With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$
    while $|\mathcal{B}| =$ thresh **do**
        For each element of $\mathcal{B}$ throw away the element independently with probability $\frac{1}{2}$
        $p \leftarrow \frac{p}{2}$
Output $\frac{|\mathcal{B}|}{p}$

Bad_i: The value of p at iteration i is less than $\frac{1}{\varepsilon^2 \cdot \mathrm{F}_0}$.

Algorithm Adaptive Estimator

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2} \log(\frac{8m}{\delta})$; $p \leftarrow 1$
for $i = 1$ to m **do**
  $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
  With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$
  while $|\mathcal{B}| =$ thresh **do**
    For each element of $\mathcal{B}$ throw away the element independently with probability $\frac{1}{2}$
    $p \leftarrow \frac{p}{2}$
Output $\frac{|\mathcal{B}|}{p}$

Bad_i: The value of p at iteration i is less than $\frac{1}{\varepsilon^2 \cdot \text{F}_0}$.

Claim 2 $\Pr[\text{Bad}_i] \leq \frac{\delta}{4 \cdot m}$

- For p to fall below $\frac{1}{\varepsilon^2 \cdot \text{F}_0}$, it should be the case that if every element is sampled with $p = \frac{1}{\varepsilon^2 \cdot \text{F}_0}$, we would have $|\mathcal{B}| \geq$ thresh
- Apply Chernoff bound on sum of i.i.d. indicator variables

Algorithm Adaptive Estimator

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2} \log(\frac{8m}{\delta})$; $p \leftarrow 1$
for $i = 1$ to m **do**
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
    With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$
    while $|\mathcal{B}| =$ thresh **do**
        For each element of $\mathcal{B}$ throw away the element independently with probability $\frac{1}{2}$
        $p \leftarrow \frac{p}{2}$
Output $\frac{|\mathcal{B}|}{p}$

Bad_i: The value of p at iteration i is less than $\frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$.

Claim 2 $\Pr[\mathsf{Bad}_i] \leq \frac{\delta}{4 \cdot m}$

- For p to fall below $\frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$, it should be the case that if every element is sampled with $p = \frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$, we would have $|\mathcal{B}| \geq$ thresh
- Apply Chernoff bound on sum of i.i.d. indicator variables

Claim 3 $\Pr[\mathsf{Bad} = \bigcup_i \mathsf{Bad}_i] \leq \frac{\delta}{4}$

Algorithm Adaptive Estimator

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2}\log(\frac{8m}{\delta})$; $p \leftarrow 1$
for $i = 1$ to m **do**
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
    With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$
    while $|\mathcal{B}| =$ thresh **do**
        For each element of $\mathcal{B}$ throw away the element independently with probability $\frac{1}{2}$
        $p \leftarrow \frac{p}{2}$
Output $\frac{|\mathcal{B}|}{p}$

Bad$_i$: The value of p at iteration i is less than $\frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$.

Claim 2 $\Pr[\text{Bad}_i] \leq \frac{\delta}{4 \cdot m}$

- For p to fall below $\frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$, it should be the case that if every element is sampled with $p = \frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$, we would have $|\mathcal{B}| \geq$ thresh
- Apply Chernoff bound on sum of i.i.d. indicator variables

Claim 3 $\Pr[\text{Bad} = \bigcup_i \text{Bad}_i] \leq \frac{\delta}{4}$

Error: The output of the algorithm is not in the range $[(1-\varepsilon)\mathsf{F}_0, (1+\varepsilon)\mathsf{F}_0]$.

Algorithm Adaptive Estimator

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2}\log(\frac{8m}{\delta})$; $p \leftarrow 1$
for $i = 1$ to m **do**
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
    With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$
    while $|\mathcal{B}| =$ thresh **do**
        For each element of $\mathcal{B}$ throw away the element independently with probability $\frac{1}{2}$
        $p \leftarrow \frac{p}{2}$
Output $\frac{|\mathcal{B}|}{p}$

Bad$_i$: The value of p at iteration i is less than $\frac{1}{\varepsilon^2 \cdot \mathrm{F}_0}$.

Claim 2 $\Pr[\mathrm{Bad}_i] \leq \frac{\delta}{4 \cdot m}$

- For p to fall below $\frac{1}{\varepsilon^2 \cdot \mathrm{F}_0}$, it should be the case that if every element is sampled with $p = \frac{1}{\varepsilon^2 \cdot \mathrm{F}_0}$, we would have $|\mathcal{B}| \geq$ thresh
- Apply Chernoff bound on sum of i.i.d. indicator variables

Claim 3 $\Pr[\mathrm{Bad} = \bigcup_i \mathrm{Bad}_i] \leq \frac{\delta}{4}$

Error: The output of the algorithm is not in the range $[(1-\varepsilon)\mathrm{F}_0, (1+\varepsilon)\mathrm{F}_0]$.

Claim 3 $\Pr[\mathrm{Error} \cap \overline{\mathrm{Bad}}] \leq \frac{\delta}{4}$

- Apply Chernoff bound on sum of i.i.d. indicator variables

Algorithm Adaptive Estimator

Initialize $\mathcal{B} \leftarrow \emptyset$; thresh $\leftarrow \frac{12}{\varepsilon^2}\log(\frac{8m}{\delta})$; $p \leftarrow 1$
for $i = 1$ to m **do**
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a_i\}$
    With probability p, $\mathcal{B} \leftarrow \mathcal{B} \cup \{a_i\}$
    while $|\mathcal{B}| =$ thresh **do**
        For each element of $\mathcal{B}$ throw away the element independently with probability $\frac{1}{2}$
        $p \leftarrow \frac{p}{2}$
Output $\frac{|\mathcal{B}|}{p}$

Bad$_i$: The value of p at iteration i is less than $\frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$.

Claim 2 $\Pr[\text{Bad}_i] \leq \frac{\delta}{4 \cdot m}$

- For p to fall below $\frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$, it should be the case that if every element is sampled with $p = \frac{1}{\varepsilon^2 \cdot \mathsf{F}_0}$, we would have $|\mathcal{B}| \geq$ thresh
- Apply Chernoff bound on sum of i.i.d. indicator variables

Claim 3 $\Pr[\text{Bad} = \bigcup_i \text{Bad}_i] \leq \frac{\delta}{4}$

Error: The output of the algorithm is not in the range $[(1-\varepsilon)\mathsf{F}_0, (1+\varepsilon)\mathsf{F}_0]$.

Claim 3 $\Pr[\text{Error} \cap \overline{\text{Bad}}] \leq \frac{\delta}{4}$

- Apply Chernoff bound on sum of i.i.d. indicator variables

Lemma 1 $\Pr[\text{Error}] \leq \frac{\delta}{2}$

Well, here we are

Distinct Elements in Streams: An Algorithm for the (Text) Book, ESA 2022

Algorithm Distinct-element-estimator

```
Input Stream 𝒟 = ⟨a_1, a_2, ..., a_m⟩, ε, δ
Initialize p ← 1; ℬ ← ∅; thresh ← (12/ε^2) log(8m/δ)
for i = 1 to m do
    ℬ ← ℬ \ {a_i}
    With probability p, ℬ ← ℬ ∪ {a_i}
    while |ℬ| = thresh do
        Throw away each element of ℬ with probability 1/2
        p ← p/2
Output |ℬ|/p
```

Theorem (C-Vinodchandran-Meel (ESA'22))

A simple *algorithm with time and space complexity of* $O(\frac{1}{\varepsilon^2} \cdot \log n \cdot (\log m + \log 1/\delta))$.

What is so nice about the algorithm?

- It is simple and can be followed by students with only a knowledge of Chernoff bounds.

What is so nice about the algorithm?

- It is simple and can be followed by students with only a knowledge of Chernoff bounds.
- The algorithm is a sampling-based algorithm.

What is so nice about the algorithm?

- It is simple and can be followed by students with only a knowledge of Chernoff bounds.
- The algorithm is a sampling-based algorithm.
  - This makes the algorithm much faster in implementation compared to hashing-based algorithms.

What is so nice about the algorithm?

- It is simple and can be followed by students with only a knowledge of Chernoff bounds.
- The algorithm is a sampling-based algorithm.
  - This makes the algorithm much faster in implementation compared to hashing-based algorithms.
  - It can be extended to the Union of Sets Estimation Problem.

Beyond the Singleton Setting: Delphic Sets (*know thyself*)

- The algorithm naturally extends to setting where every element a_i is replaced by $S_i \subseteq [n]$ belonging to Delphic family of sets and we are interested in computing $|\cup S_i|$

Cardinality	: Know the size of S_i
Sample	: Sample uniformly at random elements from S_i
Membership	: For an element $x \in [n]$, check if $x \in S_i$

Beyond the Singleton Setting: Delphic Sets (*know thyself*)

- The algorithm naturally extends to setting where every element a_i is replaced by $S_i \subseteq [n]$ belonging to Delphic family of sets and we are interested in computing $|\cup S_i|$

Cardinality	: Know the size of S_i
Sample	: Sample uniformly at random elements from S_i
Membership	: For an element $x \in [n]$, check if $x \in S_i$

Importance of Delphic Sets in Practice

- Estimation of the number of solutions of a DNF Formula
- Klee's Measure Problem: Volume of d-dimensional rectangles
- Test Coverage Estimation Problem

Delphic Sets Union Estimation Problem in Streaming Setting

Union Size Estimation

Given a stream of sets $S_1, \ldots, S_m$, all $S_i \subseteq \Omega$ and two parameters ε, δ
Output an estimate $\mathcal{E}$ such that with probability $\geq (1 - \delta)$

$$(1-\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right| \leq \mathcal{E} \leq (1+\varepsilon)\left|\bigcup_{i=1}^{m} S_i\right|$$

- Representation Size of each set: $O(\log |\Omega|)$
- Actions supported in $O(\log |\Omega|)$ space and time:
  - Know the size of S_i,
  - Sample uniformly at random elements from S_i,
  - For an element $x \in \Omega$, check if $x \in S_i$

Our Main Theorem (PODS 21)

Theorem

For any stream of DELPHIC *sets* $S_1, \ldots, S_m$ *and any parameter* ε *and* δ*, there is a very simple algorithm that* (ε, δ)*-estimates* $|\bigcup_{i=1}^{m} S_i|$ *with*

- *Update-time complexity :* $\tilde{O}(\log^2(m/\delta) \cdot \varepsilon^{-2} \cdot \log|\Omega|)$
- *Space complexity :* $O(\log(m/\delta) \cdot \varepsilon^{-2} \log|\Omega|)$.

Our Main Theorem (PODS 21)

Theorem

For any stream of DELPHIC sets $S_1, \ldots, S_m$ and any parameter ε and δ, there is a very simple algorithm that (ε, δ)-estimates $|\bigcup_{i=1}^{m} S_i|$ with

- *Update-time complexity :* $\tilde{O}(\log^2(m/\delta) \cdot \varepsilon^{-2} \cdot \log|\Omega|)$
- *Space complexity :* $O(\log(m/\delta) \cdot \varepsilon^{-2} \log|\Omega|)$.

Note: If one is only interested in space complexity and not on update-time complexity, simple F_0 estimation by Kane, Nelson and Woodruff (2010) give $O(\frac{1}{\varepsilon^2} + \log|\Omega|)$ bound and this is also lower bound for space complexity.

The main idea is the same as the Singleton setting

The main idea is the same as the Singleton setting

- At any point of time, say after k items (sets) $S_1, \ldots, S_k$ we store a random subset $\mathcal{B}$ of $\cup_{i=1}^{k} S_i$ such that,

The main idea is the same as the Singleton setting

- At any point of time, say after k items (sets) $S_1, \ldots, S_k$ we store a random subset $\mathcal{B}$ of $\cup_{i=1}^{k} S_i$ such that,

Criteria 1 Each element of $\cup_{i=1}^{k} S_i$ is in $\mathcal{B}$ independently with probability p,

The main idea is the same as the Singleton setting

- At any point of time, say after k items (sets) $S_1, \ldots, S_k$ we store a random subset $\mathcal{B}$ of $\cup_{i=1}^{k} S_i$ such that,

Criteria 1 Each element of $\cup_{i=1}^{k} S_i$ is in $\mathcal{B}$ independently with probability p,

Criteria 2 p is small enough so that the number of elements selected is not too large, and

The main idea is the same as the Singleton setting

- At any point of time, say after k items (sets) $S_1, \dots, S_k$ we store a random subset $\mathcal{B}$ of $\cup_{i=1}^{k} S_i$ such that,

Criteria 1 Each element of $\cup_{i=1}^{k} S_i$ is in $\mathcal{B}$ independently with probability p,

Criteria 2 p is small enough so that the number of elements selected is not too large, and

Criteria 3 With high probability $p \geq \frac{\log(1/\delta)}{\varepsilon^2 |\cup_{i=1}^{k} S_i|}$.

The main idea is the same as the Singleton setting

- At any point of time, say after k items (sets) $S_1, \ldots, S_k$ we store a random subset $\mathcal{B}$ of $\cup_{i=1}^{k} S_i$ such that,

Criteria 1 Each element of $\cup_{i=1}^{k} S_i$ is in $\mathcal{B}$ independently with probability p,

Criteria 2 p is small enough so that the number of elements selected is not too large, and

Criteria 3 With high probability $p \geq \frac{\log(1/\delta)}{\varepsilon^2 |\cup_{i=1}^{k} S_i|}$.

- Then $|\mathcal{B}|/p$ helps to estimate the union.

The main idea is the same as the Singleton setting

- At any point of time, say after k items (sets) $S_1, \ldots, S_k$ we store a random subset $\mathcal{B}$ of $\cup_{i=1}^{k} S_i$ such that,

Criteria 1 Each element of $\cup_{i=1}^{k} S_i$ is in $\mathcal{B}$ independently with probability p,

Criteria 2 p is small enough so that the number of elements selected is not too large, and

Criteria 3 With high probability $p \geq \frac{\log(1/\delta)}{\varepsilon^2 |\cup_{i=1}^{k} S_i|}$.

- Then $|\mathcal{B}|/p$ helps to estimate the union.

Except one more trick: How do you do even handle the first set?

The main idea is the same as the Singleton setting

- At any point of time, say after k items (sets) $S_1, \ldots, S_k$ we store a random subset $\mathcal{B}$ of $\cup_{i=1}^{k} S_i$ such that,

Criteria 1 Each element of $\cup_{i=1}^{k} S_i$ is in $\mathcal{B}$ independently with probability p,

Criteria 2 p is small enough so that the number of elements selected is not too large, and

Criteria 3 With high probability $p \geq \frac{\log(1/\delta)}{\varepsilon^2 |\cup_{i=1}^{k} S_i|}$.

- Then $|\mathcal{B}|/p$ helps to estimate the union.

Except one more trick: How do you do even handle the first set?

How to sample each element of the set S_1 independently with probability p?

Same Algorithm (nearly) works

Algorithm Delphic-Union

```
Initialize $\mathcal{B} \leftarrow \emptyset; p \leftarrow 1$
thresh $\leftarrow 3 \cdot \left(\frac{\log(2m/\delta)}{\varepsilon^2}\right)$
for $i = 1$ to $m$ do
    for all $s \in \mathcal{B}$ do
        if $s \in S_i$ then remove $s$ from $\mathcal{B}$
    For each element of $S_i$: with probability $p$ add it to $\mathcal{B}$.
    while $|\mathcal{B}| \geq$ thresh do
        Update $p = p/2$
        Throw away each element of $\mathcal{B}$ with probability $1/2$
Output $\frac{|\mathcal{B}|}{p}$
```

Same Algorithm (nearly) works

Algorithm Delphic-Union

Initialize $\mathcal{B} \leftarrow \emptyset; p \leftarrow 1$
thresh $\leftarrow 3 \cdot \left(\frac{\log(2m/\delta)}{\varepsilon^2}\right)$
for $i = 1$ to m **do**
    for all $s \in \mathcal{B}$ **do**
        if $s \in S_i$ **then** remove s from $\mathcal{B}$
    For each element of S_i: with probability p add it to $\mathcal{B}$.
    while $|\mathcal{B}| \geq$ thresh **do**
        Update $p = p/2$
        Throw away each element of $\mathcal{B}$ with probability $1/2$
Output $\frac{|\mathcal{B}|}{p}$

Challenge *For each element of S_i: with probability p add it to $\mathcal{B}$.*

Same Algorithm (nearly) works

Algorithm Delphic-Union

Initialize $\mathcal{B} \leftarrow \emptyset; p \leftarrow 1$
thresh $\leftarrow 3 \cdot \left(\frac{\log(2m/\delta)}{\varepsilon^2}\right)$
for $i = 1$ to m **do**
    for all $s \in \mathcal{B}$ **do**
        if $s \in S_i$ **then** remove s from $\mathcal{B}$
    For each element of S_i: with probability p add it to $\mathcal{B}$.
    while $|\mathcal{B}| \geq$ thresh **do**
        Update $p = p/2$
        Throw away each element of $\mathcal{B}$ with probability $1/2$
Output $\frac{|\mathcal{B}|}{p}$

Challenge *For each element of S_i: with probability p add it to $\mathcal{B}$.*

- $N_i \leftarrow \text{Bin}(|S_i|, p)$
- Draw N_i distinct elements from S_i by drawing $N_i \log N_i \log(\frac{2m}{\delta})$ samples

Same Algorithm (nearly) works

Algorithm Delphic-Union

```
Initialize $\mathcal{B} \leftarrow \emptyset; p \leftarrow 1$
thresh $\leftarrow 3 \cdot \left(\frac{\log(2m/\delta)}{\varepsilon^2}\right)$
for $i = 1$ to $m$ do
    for all $s \in \mathcal{B}$ do
        if $s \in S_i$ then remove $s$ from $\mathcal{B}$
    For each element of $S_i$: with probability $p$ add it to $\mathcal{B}$.
    while $|\mathcal{B}| \geq$ thresh do
        Update $p = p/2$
        Throw away each element of $\mathcal{B}$ with probability $1/2$
Output $\frac{|\mathcal{B}|}{p}$
```

Challenge *For each element of S_i: with probability p add it to $\mathcal{B}$.*

- $N_i \leftarrow \text{Bin}(|S_i|, p)$
- Draw N_i distinct elements from S_i by drawing $N_i \log N_i \log(\frac{2m}{\delta})$ samples

One Last thing: What if N_i is too large? (Update time complexity)

- Well, just update p to $p/2$ and resample $N_i \leftarrow \text{Bin}(N_i, 1/2)$ until $N_i <$ thresh

Union Estimation for DELPHIC sets

Algorithm Final Algorithm

```
Initialize 𝓑 ← ∅; p ← 1;  thresh ← 3 · (log(2m/δ) / ε²)
for i = 1 to m do
    for all s ∈ 𝓑 do
        if s ∈ S_i then remove s from 𝓑
    N_i ← Bin(|S_i|, p)
    while |𝓑| + N_i ≥ thresh  do
        N_i ← Bin(N_i, 1/2) and p ← p/2
        Throw away each element of 𝓑 with probability 1/2
    Pick N_i distinct elements of S_i randomly and add them to 𝓑.
Output |𝓑| / p
```

Union Estimation for DELPHIC sets

Algorithm Final Algorithm

Initialize $\mathcal{B} \leftarrow \emptyset$; $p \leftarrow 1$; thresh $\leftarrow 3 \cdot \left(\frac{\log(2m/\delta)}{\varepsilon^2}\right)$
for $i = 1$ to m **do**
  for all $s \in \mathcal{B}$ **do**
    if $s \in S_i$ **then** remove s from $\mathcal{B}$
  $N_i \leftarrow \mathrm{Bin}(|S_i|, p)$
  while $|\mathcal{B}| + N_i \geq$ thresh **do**
    $N_i \leftarrow \mathrm{Bin}(N_i, 1/2)$ and $p \leftarrow p/2$
    Throw away each element of $\mathcal{B}$ with probability $1/2$
  Pick N_i distinct elements of S_i randomly and add them to $\mathcal{B}$.
Output $\frac{|\mathcal{B}|}{p}$

Theorem

For any stream of DELPHIC sets $S_1, \ldots, S_m$ and any parameter ε and δ, there is a very simple algorithm that (ε, δ)-estimates $|\bigcup_{i=1}^m S_i|$ with

- *Update-time complexity :* $\tilde{O}(\log(m/\delta) \cdot \varepsilon^{-2} \cdot \log(m/\delta) \cdot \log|\Omega|)$
- *Space complexity :* $O(\log(m/\delta) \cdot \varepsilon^{-2} \log|\Omega|)$.

Some implications of our result

Theorem

There is a very simple algorithm that takes in input a stream of Delphic sets $S_1, \ldots, S_m$, parameters ε and δ, and provides (ε, δ)-estimate of $|\bigcup_{i=1}^{m} S_i|$

- *Update-time complexity :* $\tilde{O}(\log^2(m/\delta) \cdot \varepsilon^{-2} \cdot \log n)$
- *Space complexity :* $O(\log(m/\delta) \cdot \varepsilon^{-2} \cdot \log n)$.

- **Klee's Measure Problem** Estimate union of axis-parallel rectangles in $\mathbb{R}^d$. Our algorithm gives the first efficient algorithm with linear dependence on the dimension d — a long standing open problem. (PODS-21, PODS-22)
- **Model Counting for DNF** Count the number of DNF solutions. Our algorithm (nearly) matches the optimal bounds (in non-streaming setting!) The practical implementation (after engineering improvements) achieves nearly 100× speed up over prior state of the art. (IJCAI-23)
- **Coverage Estimation Problem** Critical for software testing: estimate amount of coverage achieved with certain set of "test vectors". We out-perform all currently used techniques in practice. (ICSE-22)

Wrapping up ...

Conclusion A simple algorithm for element distinctness that is sampling based.

The algorithm generalizes to obtain estimates of union of DELPHIC sets.

The algorithm solves the Klee's Measure problem.

The algorithm can be used even in non-streaming set-up to design practically efficient algorithms.

Further Work (PODS 22, APPROX 24) Algorithm for Delphic sets without dependence on stream size (m). We have the update-time and space complexity of $\tilde{O}(\log^2(|\Omega|/\delta) \cdot \epsilon^{-2})$.

Donald E. Knuth modified the algorithm to obtain a unbiased F_0 estimator for the Distinct Element problem.

Open Problem Optimal algorithm for Delphic sets?

These slides are available at `tinyurl.com/streaming-talk`
Knuth's Note: `https://cs.stanford.edu/~knuth/papers/cvm-note.pdf`