# Lecture 1: Distinct Elements and Frequency Moments in Data Streams

**Michael Kapralov**

**EPFL**

May 23, 2017

# Streaming model

Observe a (very long) stream of data, e.g. IP packets, tweets, search queries….

Task: maintain (approximate) statistics of the stream

# Streaming model

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

  Typically, assume $n$ is known, $i_j \in [n]$

- Small (sublinear) storage: typically $n^\alpha, \alpha < 1$ or $\log^{O(1)} n$

  Units of storage: bits, words or ‘data items’ (e.g., points, nodes/edges)

- Fast processing time per element

- Mostly randomized algorithms

  Randomness often necessary

In this lecture:

- Distinct elements
- Frequency moments (AMS sketch)

In this lecture:

- **Distinct elements**
- Frequency moments (AMS sketch)

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!

1 2 3 4 5 6 7 8 9 10

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

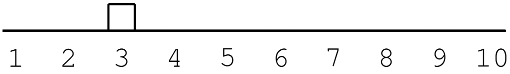
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

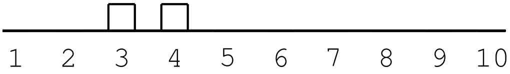
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

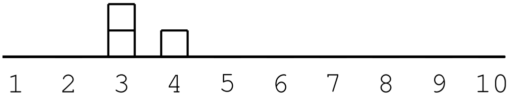
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

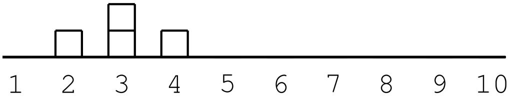
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

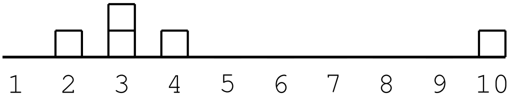
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!


3 4 3 2 10 3

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

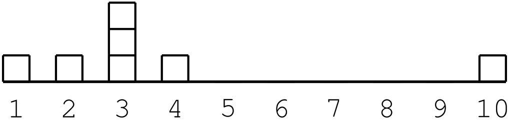
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

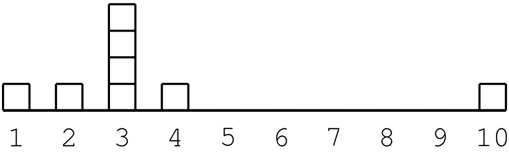
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

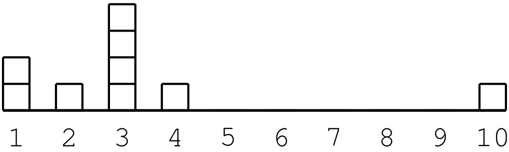
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

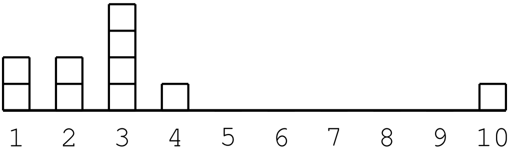
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

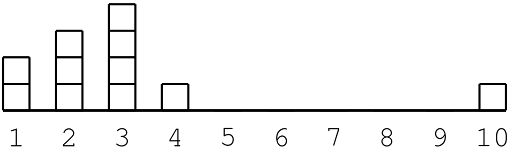
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

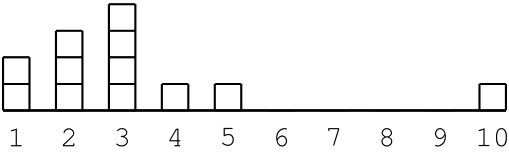
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

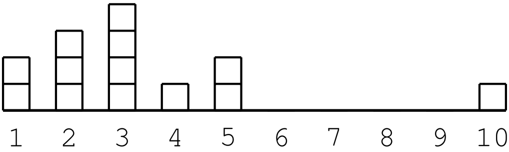
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

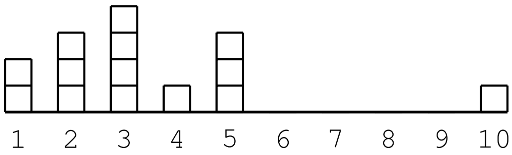
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

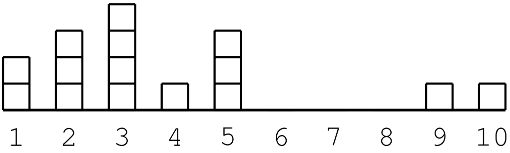
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

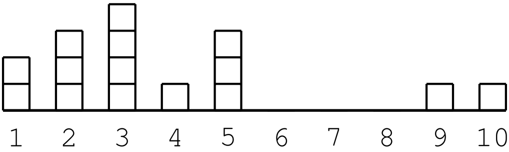
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

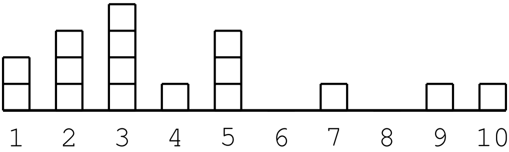
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

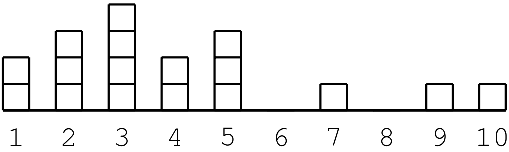
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

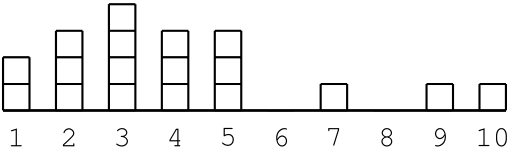
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

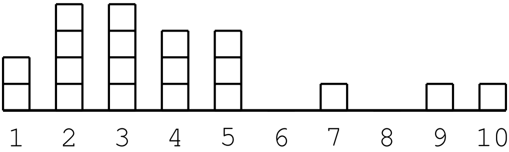
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\mathrm{poly}(n)}$

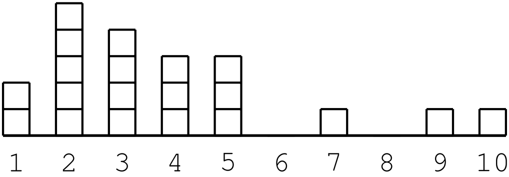
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

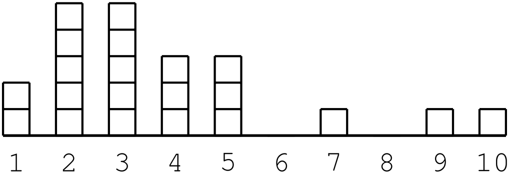
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2 3

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

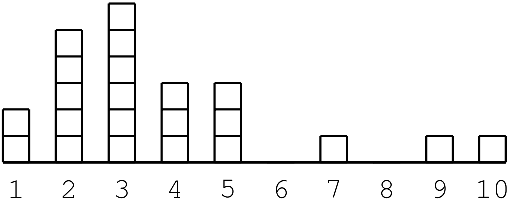
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2 3 3

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_{\text{poly}(n)}$

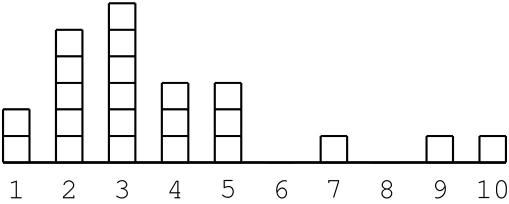
  Typically, assume $n$ is known, $i_j \in [n]$

- Output number of distinct elements seen

  (Approximately, randomness ok)

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!




#distinct elements=#`{1, 2, 3, 4, 5, 7, 9, 10}=8`

`3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2 3 3`

# Estimating number of IP flows through a router

Estimate the # of IP flows through a router

$$
\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} & \begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}
$$

# Estimating number of IP flows through a router


destination

source

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Estimating number of IP flows through a router


destination

source

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$
\begin{array}{c c}
 & \texttt{destination} \\
\texttt{source} & \begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
$$

# Estimating number of IP flows through a router


destination

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

source

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router

$$\begin{array}{cc} & \texttt{destination} \\ \texttt{source} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \end{array}$$

# Estimating number of IP flows through a router

$$\begin{array}{cc}
 & \text{destination} \\
\text{source} & \begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}$$

| Src | Dst |
| --- | --- |
| DATA | |

# Estimating number of IP flows through a router

$$
\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} & \begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
$$

# Estimating number of IP flows through a router


$$
\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\texttt{destination}} \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\
\texttt{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$
\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} &
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}
$$

# Estimating number of IP flows through a router

$$
\begin{array}{c|ccccccccc}
 & & & & \text{destination} & & & & & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\
\text{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$\begin{array}{cc} & \texttt{destination} \\ \texttt{source} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \end{array}$$

# Estimating number of IP flows through a router


destination

| source | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | **3** | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router

$$
\begin{array}{c c}
 & \text{destination} \\
\text{source} &
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}
$$

# Estimating number of IP flows through a router


$$
\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\texttt{destination}} \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\texttt{source} & 0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$
\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} & \begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
$$

# Estimating number of IP flows through a router

$$
\begin{array}{c}
\text{destination} \\
\end{array}
$$

$$
\begin{array}{c ccccccccc}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
\text{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & \color{red}{\mathbf{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

| Src | Dst |
| --- | --- |
| DATA | |

# Estimating number of IP flows through a router

$$
\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} &
\begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
$$

# Estimating number of IP flows through a router


destination

source

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | **3** | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | **2** | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router

$$\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} & \begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}$$

# Estimating number of IP flows through a router


destination

source

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | **4** | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | **2** | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$
\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} & \begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
$$

# Estimating number of IP flows through a router

$$
\begin{array}{c|c}
 & \text{destination} \\
\text{source} & \begin{matrix}
0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&\mathbf{1}&0&0\\
0&0&0&0&0&0&0&0&0\\
0&0&0&\mathbf{4}&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0\\
0&\mathbf{3}&0&0&0&0&0&\mathbf{1}&0\\
0&0&0&0&0&\mathbf{1}&0&0&0\\
0&0&\mathbf{1}&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0
\end{matrix}
\end{array}
$$

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} & \begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}$$

# Estimating number of IP flows through a router

$$
\begin{array}{cc}
 & \text{destination} \\
\text{source} &
\begin{array}{ccccccccc}
\color{red}{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
$$

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$\begin{array}{cc} & \texttt{destination} \\ \texttt{source} & \begin{array}{ccccccccc} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \end{array}$$

# Estimating number of IP flows through a router

$$\begin{array}{cc} & \texttt{destination} \\ \texttt{source} & \begin{matrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \end{array}$$

# Estimating number of IP flows through a router


$$
\begin{array}{cc}
 & \text{destination} \\
\text{source} & \begin{matrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}
$$

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


$$\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} & \begin{matrix}
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}$$

# Estimating number of IP flows through a router

$$
\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\texttt{destination}} \\
\hline
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & \color{red}{5} & 0 & 0 & 0 & 0 & 0 \\
\texttt{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

| Src | Dst |
|---|---|
| DATA | |

# Estimating number of IP flows through a router


| | destination | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | **5** | 0 | 0 | 0 | 0 | 0 |
| source | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | **4** | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Estimating number of IP flows through a router


$$
\begin{array}{cc}
 & \texttt{destination} \\
\texttt{source} &
\begin{array}{ccccccccc}
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{5} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
$$

# Estimating number of IP flows through a router


Estimate the # of IP flows through a router

destination

| source | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | **5** | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | **4** | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Estimating number of IP flows through a router


Estimate the # of IP flows through a router

destination

source

1

1

5

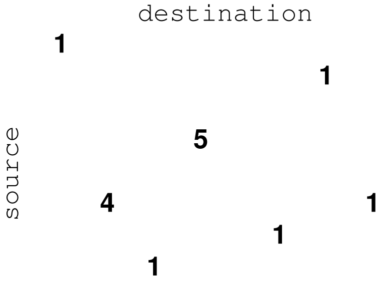
4 1

1

1

# Estimating number of IP flows through a router

Estimate the # of IP flows
through a router




Trivial: store all distinct `IP` pairs

Space complexity: $\Omega(n)$

# Estimating number of IP flows through a router


Estimate the # of IP flows through a router

destination

source

1

1

5

4 1

1

1

Trivial: store all distinct `IP` pairs

Space complexity: $\Theta(n)$

This lecture: solve in space $\log^{O(1)} n$

$\boxed{\text{Exponential improvement!}}$

# Estimating search statistics

Given a set of items as a stream (e.g. queries on `google.com` over a period of time)

`Geneva to NYC, coffee in Geneva, Geneva to NYC`

# Estimating search statistics

Given a set of items as a stream (e.g. queries on `google.com` over a period of time)

`Geneva to NYC, coffee in Geneva, Geneva to NYC`

Find the # of distinct items in the set

`Geneva to NYC, coffee in Geneva`

# Streaming model

| | Trivial | This lecture |
|---|---|---|
| Solution | `hash<string> h;` | HyperLogLog |
| Space | `# of distinct items` | $\log^{O(1)} n$ |

# Streaming model

| | Trivial | This lecture |
|---|---|---|
| Solution | `hash<string> h;` | HyperLogLog |
| Space | `# of distinct items` | $\log^{O(1)} n$ |

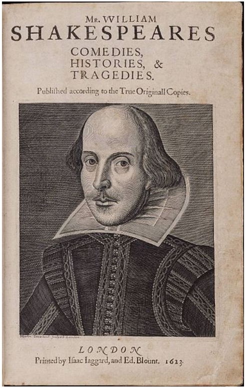
Are constants small?

# Streaming model

| | Trivial | This lecture |
|---|---|---|
| Solution | `hash<string> h;` | HyperLogLog |
| Space | `# of distinct items` | $\log^{O(1)} n$ |

Are constants small?

HyperLogLog: estimate Shakespeare's vocabulary using 128 bits of memory



# Streaming model

Widely used in practice for **scalable data analytics**

most frequent searches on `google.com`
over a time period

most frequent tweets

# Distinct elements problem

- Single pass over the data: $i_1, i_2, \ldots, i_n$

  integers between 1 and $\text{poly}(n)$

- Output $(1 \pm \varepsilon)$-approximation to # of distinct elements

  $(1-\varepsilon)\text{DE} \le \widehat{\text{DE}} \le (1+\varepsilon)\text{DE}$

- Small storage: will get $\log^{O(1)} n$

  Much better than storing all items!

- Success probability $\ge 1 - \delta$

Simpler goal: for a given $T > 0$, provide an algorithm ALG that, with probability $1 - \delta$:

- answers YES if DE $> (1 + \varepsilon) T$
- answers NO if DE $< (1 - \varepsilon) T$

Simpler goal: for a given $T > 0$, provide an algorithm ALG that, with probability $1 - \delta$:

- answers YES if DE $> (1+\varepsilon)T$
- answers NO if DE $< (1-\varepsilon)T$

To achieve the original goal, run in ALG with thresholds

$$T = 1, 1+\varepsilon, (1+\varepsilon)^2, \ldots, n$$

Simpler goal: for a given $T > 0$, provide an algorithm ALG that, with probability $1 - \delta$:

- answers YES if $\text{DE} > (1+\varepsilon)T$
- answers NO if $\text{DE} < (1-\varepsilon)T$

To achieve the original goal, run in ALG with thresholds

$$T = 1, 1+\varepsilon, (1+\varepsilon)^2, \ldots, n$$

- total space multiplied by $\log_{1+\varepsilon} n \approx \frac{1}{\varepsilon} \log n$
- failure probability multiplied by same factor

# Vector interpretation

$x \in \mathbb{R}^n$ 1 2 3 4 5 6 7 8 9 10

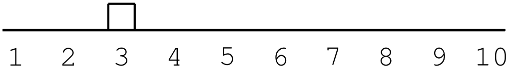
- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate DE($x$)

# Vector interpretation

$x \in \mathbb{R}^n$




3

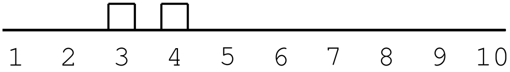
- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation

$x \in \mathbb{R}^n$



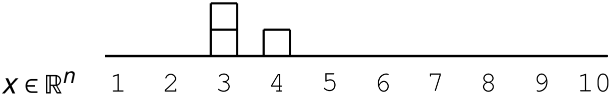

3 4

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation




3 4 3

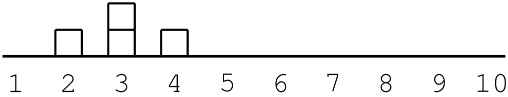
- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation

$x \in \mathbb{R}^n$



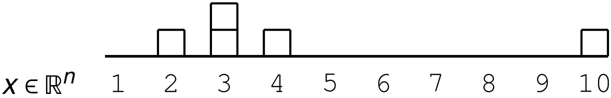

3 4 3 2

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



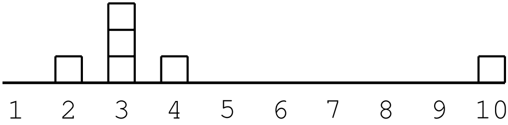

3 4 3 2 10

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation




$x \in \mathbb{R}^n$

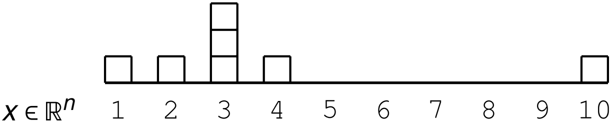
3 4 3 2 10 3

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



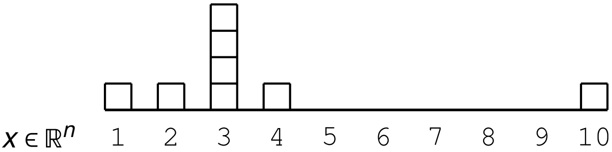

3 4 3 2 10 3 1

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



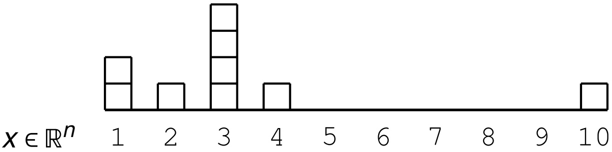

3 4 3 2 10 3 1 3

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



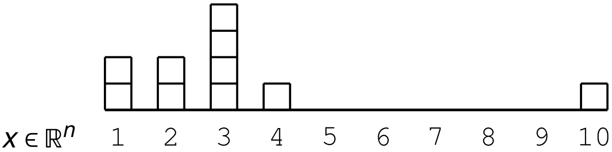

3 4 3 2 10 3 1 3 1

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



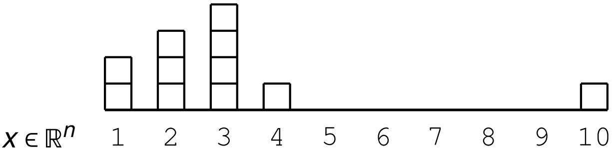

`3 4 3 2 10 3 1 3 1 2`

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



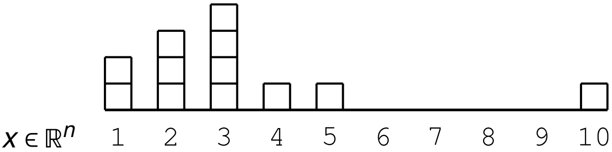

3 4 3 2 10 3 1 3 1 2 2

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



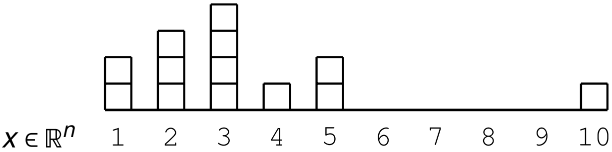

3 4 3 2 10 3 1 3 1 2 2 5

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



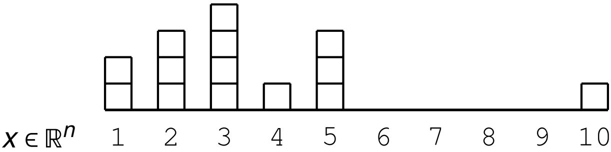

3 4 3 2 10 3 1 3 1 2 2 5 5

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



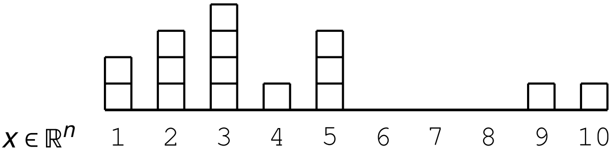

3 4 3 2 10 3 1 3 1 2 2 5 5 5

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



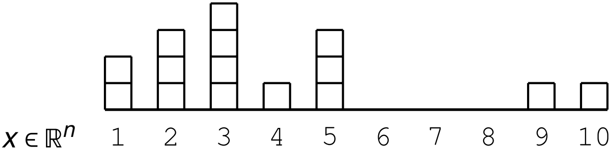

3 4 3 2 10 3 1 3 1 2 2 5 5 5 9

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate DE($x$)

# Vector interpretation



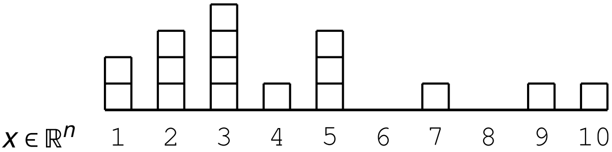

3 4 3 2 10 3 1 3 1 2 2 5 5 5 9

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



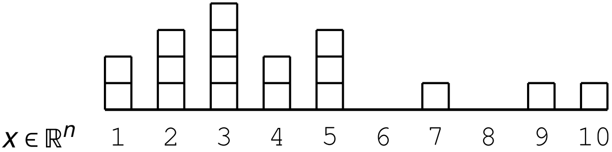

3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



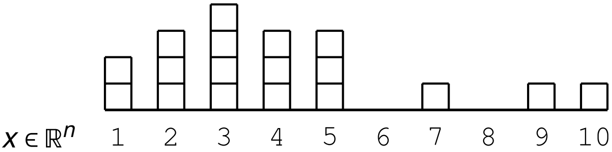

`3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4`

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



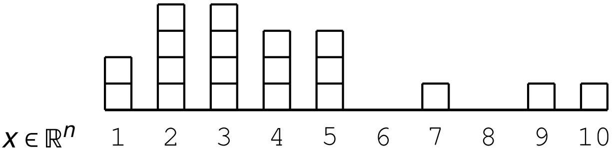

3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



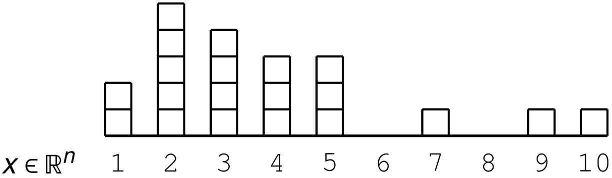

3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



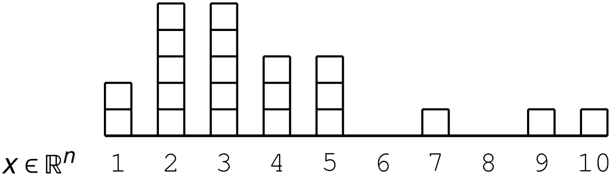

`3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2`

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



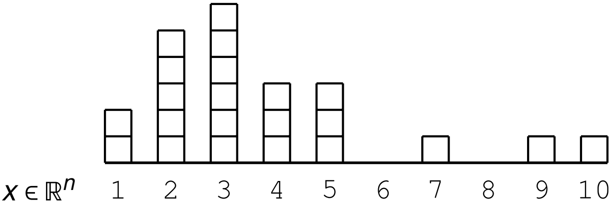

3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2 3

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation



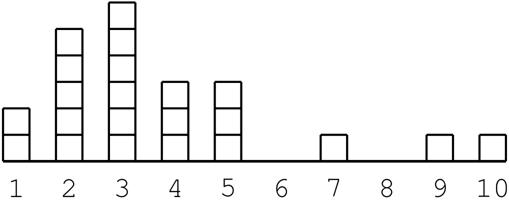

3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2 3 3

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

# Vector interpretation




#distinct elements=#{1, 2, 3, 4, 5, 7, 9, 10}=8

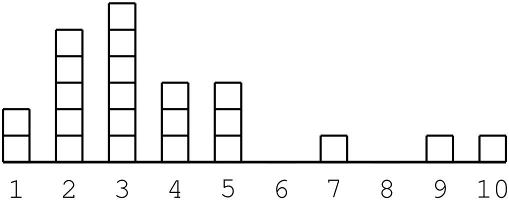
3 4 3 2 10 3 1 3 1 2 2 5 5 5 9 7 4 4 2 2 3 3

- Initially, $x = 0$
- Insertion of $i$ interpreted as

$$x_i := x_i + 1$$

- Want to estimate $\mathrm{DE}(x)$

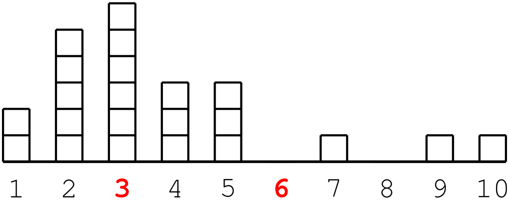
# Estimating DE($x$) – decision problem




- Choose a random set $S \subseteq [n]$ s.t. for each $i \in [n]$

$$\mathbf{Pr}[i \in S] = 1/T$$

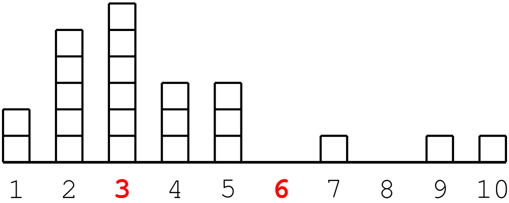
# Estimating DE($x$) – decision problem




- Choose a random set $S \subseteq [n]$ s.t. for each $i \in [n]$

$$\mathbf{Pr}[i \in S] = 1/T$$

# Estimating DE($x$) – decision problem

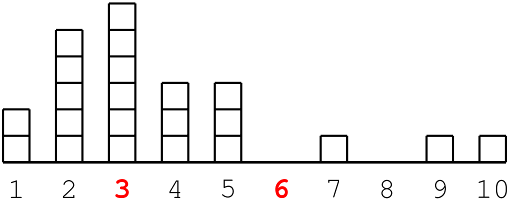



- Choose a random set $S \subseteq [n]$ s.t. for each $i \in [n]$

$$\mathbf{Pr}[i \in S] = 1/T$$

- Maintain $c_S := \sum_{i \in S} x_i$

# Estimating DE($x$) – decision problem




- Choose a random set $S \subseteq [n]$ s.t. for each $i \in [n]$

$$\mathbf{Pr}[i \in S] = 1/T$$

- Maintain $c_S := \sum_{i \in S} x_i$
- Estimation:
  - If $c_S > 0$, output YES
  - If $c_S = 0$, output NO

# Basic algorithm (decision problem)

## Algorithm:

- Choose a random set $S \subseteq [n]$ s.t. for each $i \in [n]$

$$\mathbf{Pr}[i \in S] = 1/T$$

- Maintain $c_S := \sum_{i \in S} x_i$
- Estimation:
  - If $c_S > 0$, output YES
  - If $c_S = 0$, output NO

## Analysis:

- For $T$ large enough: $\mathbf{Pr}[c_S = 0] = (1 - 1/T)^{\mathrm{DE}} \approx e^{-\mathrm{DE}/T}$
- So for small enough $\varepsilon$
  - If DE $> (1+\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] \approx e^{-(1+\varepsilon)} < 1/e - \varepsilon/3$
  - If DE $< (1-\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] \approx e^{-(1-\varepsilon)} > 1/e + \varepsilon/3$

# Full algorithm for decision problem

## Basic algorithm:

- If DE $> (1+\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] < 1/e - \varepsilon/3$
- If DE $< (1-\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] > 1/e + \varepsilon/3$

# Full algorithm for decision problem

## Basic algorithm:

- If $\mathrm{DE} > (1+\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] < 1/e - \varepsilon/3$
- If $\mathrm{DE} < (1-\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] > 1/e + \varepsilon/3$

## Full algorithm:

- Select sets $S_1, \ldots, S_k$, $k = O(\frac{1}{\varepsilon^2}\log(1/\delta))$
- Maintain counters $c_{S_j}, j \in [k]$
- $Z := \|\{j \in [k] : c_{S_j} = 0\}\|$
- If $Z < k/e$, say YES
- If $Z \geq k/e$, say NO

Space complexity? Correctness?

# Full algorithm for decision problem – space complexity

## Basic algorithm:

- If DE $> (1+\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] < 1/e - \varepsilon/3$
- If DE $< (1-\varepsilon)T$, then $\mathbf{Pr}[c_S = 0] > 1/e + \varepsilon/3$

## Full algorithm:

- Select sets $S_1, \ldots, S_k$, $k = O(\frac{1}{\varepsilon^2} \log(1/\delta))$
- $Z := \|\{j \in [k] : c_{S_j} = 0\}\|$
- If $Z < k/e$, say YES
- If $Z \geq k/e$, say NO

Space:

- Decision problem: $O(\frac{1}{\varepsilon^2} \log(1/\delta))$ numbers in $[0..n^{O(1)}]$
- Estimation: $O(\frac{1}{\varepsilon^3} \log n \log(1/\delta))$ numbers in $[0..n^{O(1)}]$

  (error probability $O(\delta \cdot \frac{1}{\varepsilon} \log n)$)

# Chernoff bound

## Theorem

*Let $Z_1, \ldots, Z_n$ be i.i.d. Bernoulli random variables with $\mathbf{E}[Z_i] = p$, and let $Z = \sum_{i=1}^n Z_i$. Then for every $\varepsilon \in (0,1)$*

$$\mathbf{Pr}\left[\left|\sum_{i=1}^n Z_i - \mathbf{E}[Z]\right| > \varepsilon \mathbf{E}[Z]\right] \le 2\exp(-\varepsilon^2 \mathbf{E}[Z]/3).$$

# How do we store the set $S$?

Choose a hash function

$$h : [n] \to [1 : T],$$

let

$$S = \{i \in [n] : h(i) = 1\}$$

# How do we store the set $S$?

Choose a hash function

$$h : [n] \to [1 : T],$$

let

$$S = \{i \in [n] : h(i) = 1\}$$

- How do we store $h$? :)

# How do we store the set $S$?

Choose a hash function

$$h : [n] \to [1 : T],$$

let

$$S = \{i \in [n] : h(i) = 1\}$$

- How do we store $h$? :)
- Use a pseudorandom number generator
  (e.g. Nisan's PRG)

# How do we store the set $S$?

Choose a hash function

$$h : [n] \to [1 : T],$$

let

$$S = \{i \in [n] : h(i) = 1\}$$

- How do we store $h$? :)
- Use a pseudorandom number generator (e.g. Nisan's PRG)

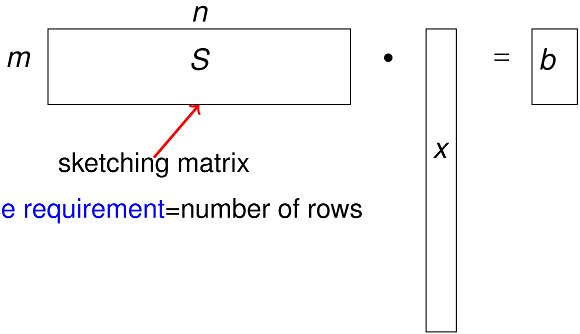
  or

- redo analysis (with slight modifications) for a pairwise independent $h$
- pairwise independent $h$ can be stored using $O(\log n)$ bits (think $ax + b \bmod p$)

Ex: redo analysis assuming that $h$ is pairwise independent only

# Linear sketching

Maintain $Sx$ for a matrix $S \in \mathbb{R}^{m \times n}$, $m$ small
$(m = O(\frac{1}{\varepsilon^3} \log^2 n \cdot \log(1/\delta)))$




space requirement=number of rows

So algorithm also works if some elements are deleted:

$$x_i := x_i - 1,$$

as long as $x \geq 0$ at the end of the stream.

# Optimal space bounds, practical algorithms

Asymptotically tight space: $O(\frac{1}{\varepsilon^2} + \log n)$ bits Kane-Nelson-Woodruff'10

Practical: Durand-Flajolet'03

And recent practical improvements:

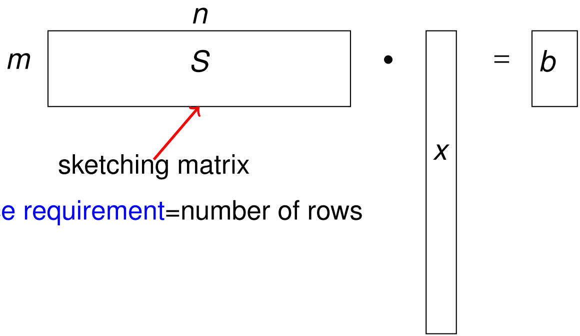
**HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm**


Stefan Heule
ETH Zurich and Google, Inc.
stheule@ethz.ch

Marc Nunkesser
Google, Inc.
marcnunkesser@google.com

Alexander Hall
Google, Inc.
alexhall@google.com

# Linear sketching




Later this week: more sketching algorithms for basic statistics, and then graph sketching

# Approximate $||x||_p$ for other $p$?

Approximate

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

in small space?

Note:

- $||x||_\infty = \max_{i \in [n]} |x_i|$
- $||x||_0 = \#$distinct elements in $x$

# Approximate $||x||_p$ for other $p$?

Approximate

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

in small space?

Note:

- $||x||_\infty = \max_{i\in[n]} |x_i|$
- $||x||_0 = \#$distinct elements in $x$

Frequency moments: $F_p = ||x||_p^p$.

# Approximate $||x||_p$ for other $p$?

Approximate

$$||x||_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

in small space?

Note:

- $||x||_\infty = \max_{i \in [n]} |x_i|$
- $||x||_0 = \#$distinct elements in $x$

Frequency moments: $F_p = ||x||_p^p$.

How much space is needed for $(1 \pm \varepsilon)$-approximation to $||x||_p$ for constant $\varepsilon$?

- $\log^{O(1)} n$ suffices for $p \in (0, 2]$
- $\Omega(n^{1-2/p})$ needed for $p > 2$.

In this lecture:

- Distinct elements
- Frequency moments (AMS sketch)

In this lecture:

- Distinct elements
- **Frequency moments (AMS sketch)**

# AMS sketch (Alon-Matias-Szegedy'96)

Goal: approximate

$$||x||_2 = \sqrt{\sum_{i \in [n]} x_i^2}$$

from a stream of increments/decrements to $x_i$.

# AMS sketch (Alon-Matias-Szegedy'96)

Goal: approximate

$$||x||_2 = \sqrt{\sum_{i \in [n]} x_i^2}$$

from a stream of increments/decrements to $x_i$.

Choose $r_1, \ldots, r_n$ to be i.i.d. r.v., with

$$\mathbf{Pr}[r_i = +1] = \mathbf{Pr}[r_i = -1] = 1/2.$$

# AMS sketch (Alon-Matias-Szegedy'96)

Goal: approximate

$$||x||_2 = \sqrt{\sum_{i \in [n]} x_i^2}$$

from a stream of increments/decrements to $x_i$.

Choose $r_1, \ldots, r_n$ to be i.i.d. r.v., with

$$\mathbf{Pr}[r_i = +1] = \mathbf{Pr}[r_i = -1] = 1/2.$$

Maintain

$$Z = \sum_{i=1}^{n} r_i x_i$$

under increments/decrements of $x$.

# AMS sketch (Alon-Matias-Szegedy'96)

Goal: approximate

$$||x||_2 = \sqrt{\sum_{i \in [n]} x_i^2}$$

from a stream of increments/decrements to $x_i$.

Choose $r_1, \ldots, r_n$ to be i.i.d. r.v., with

$$\mathbf{Pr}[r_i = +1] = \mathbf{Pr}[r_i = -1] = 1/2.$$

Maintain

$$Z = \sum_{i=1}^{n} r_i x_i$$

under increments/decrements of $x$.

Basic algorithm: output $Z^2$

# AMS sketch (Alon-Matias-Szegedy'96)

Goal: approximate

$$||x||_2 = \sqrt{\sum_{i \in [n]} x_i^2}$$

from a stream of increments/decrements to $x_i$.

Choose $r_1, \ldots, r_n$ to be i.i.d. r.v., with

$$\mathbf{Pr}[r_i = +1] = \mathbf{Pr}[r_i = -1] = 1/2.$$

Maintain

$$Z = \sum_{i=1}^{n} r_i x_i$$

under increments/decrements of $x$.

Basic algorithm: output $Z^2$

Want to claim that $Z^2$ is 'close' to $||x||_2^2$ with 'high probability'

# Alon-Matias-Szegedy – analysis (expectation)

Want to claim that $Z^2$ is 'close' to $||x||_2^2$ with 'high probability'

# Alon-Matias-Szegedy – analysis (expectation)

Want to claim that $Z^2$ is 'close' to $||x||_2^2$ with 'high probability'

Compute expectation of $Z^2$, then bound the variance

# Alon-Matias-Szegedy – analysis (expectation)

Want to claim that $Z^2$ is 'close' to $||x||_2^2$ with 'high probability'

$\boxed{\text{Compute expectation of } Z^2\text{, then bound the variance}}$

Expectation:

$$
\begin{aligned}
\mathbf{E}[Z^2] &= \mathbf{E}\left[ (\sum_{i=1}^{n} r_i x_i)^2 \right] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{E}[r_i r_j x_i x_j] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{E}[r_i r_j] x_i x_j \\
&= \sum_{i=1}^{n} x_i^2 + \sum_{i,j:i \neq j}^{n} \mathbf{E}[r_i]\mathbf{E}[r_j] x_i x_j \\
&= \sum_{i=1}^{n} x_i^2 \\
&= ||x||_2^2
\end{aligned}
$$