Sketching for Data Streams

Michael Kapralov

EPFL

August 30, 2023

Streaming model (Alon, Matias, Szegedy'96)

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: $i_1, i_2, \ldots, i_N$

  Typically, assume N is known

- 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

Heavy hitters problem

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

  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!

1 2 3 4 5 6 7 8 9 10

Heavy hitters problem

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

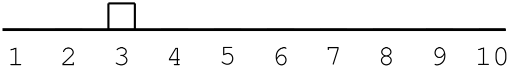
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3

Heavy hitters problem

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

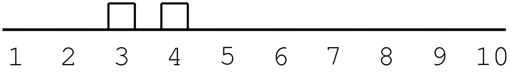
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4

Heavy hitters problem

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

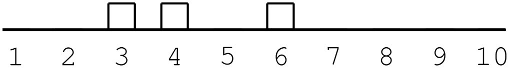
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6

Heavy hitters problem

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

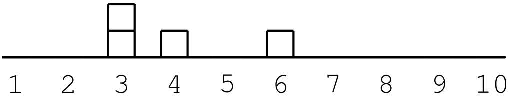
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3

Heavy hitters problem

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

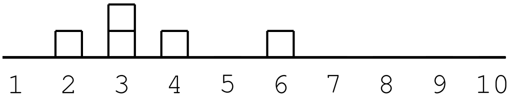
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2

Heavy hitters problem

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

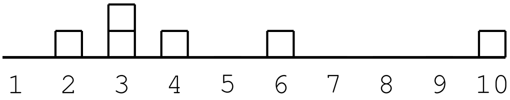
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10

Heavy hitters problem

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

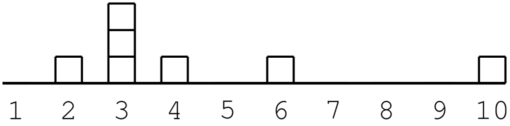
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3

Heavy hitters problem

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

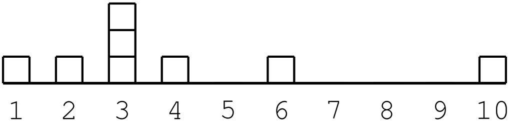
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1

Heavy hitters problem

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

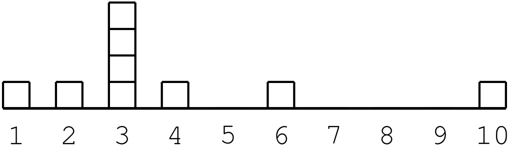
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3

Heavy hitters problem

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

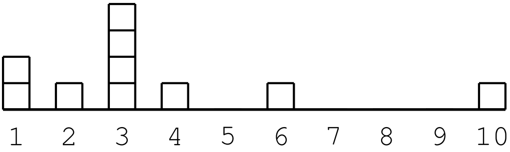
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1

Heavy hitters problem

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

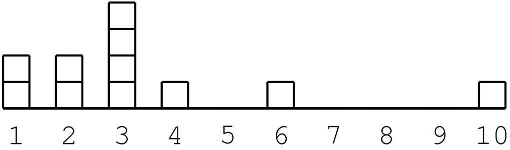
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2

Heavy hitters problem

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

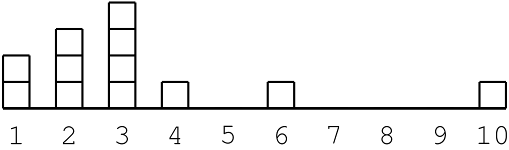
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2

Heavy hitters problem

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

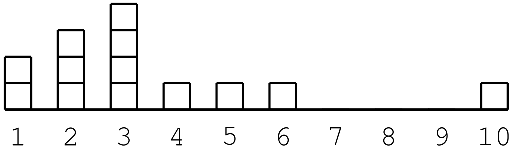
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5

Heavy hitters problem

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

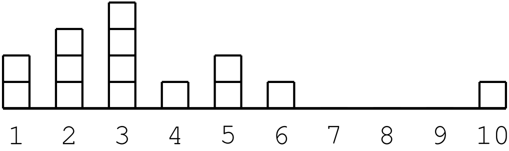
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5

Heavy hitters problem

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

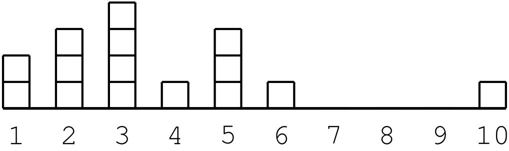
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5

Heavy hitters problem

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

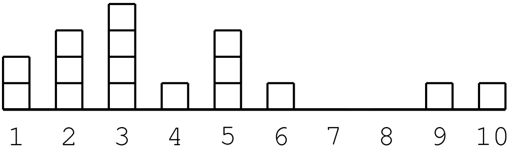
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9

Heavy hitters problem

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

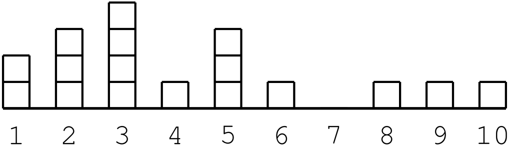
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8

Heavy hitters problem

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

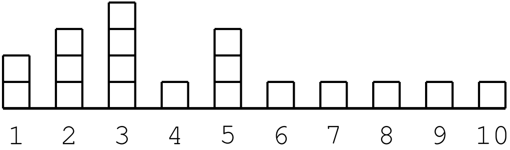
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7

Heavy hitters problem

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

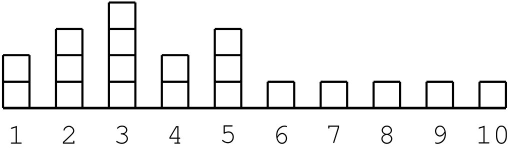
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4

Heavy hitters problem

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

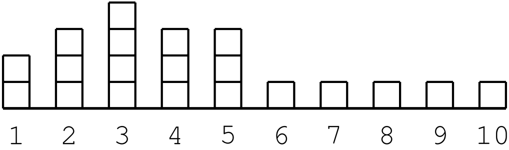
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4

Heavy hitters problem

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

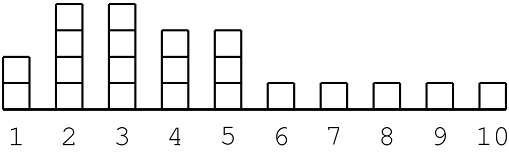
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2

Heavy hitters problem

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

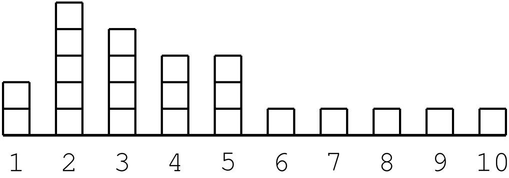
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2 2

Heavy hitters problem

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

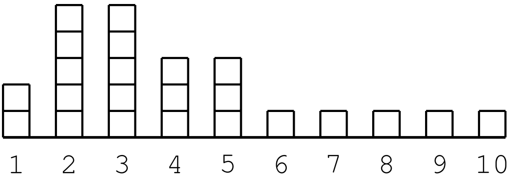
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2 2 3

Heavy hitters problem

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

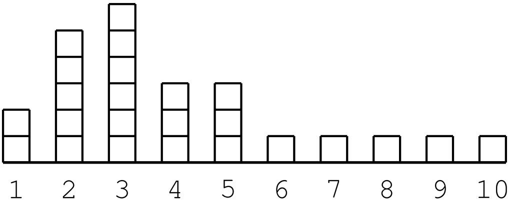
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2 2 3 3

Heavy hitters problem

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

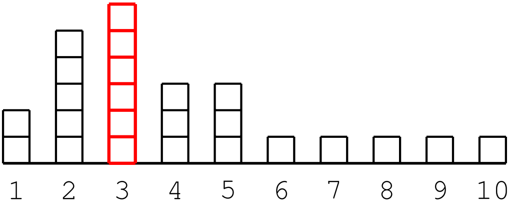
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2 2 3 3

Estimating IP flows through a router

Estimate the dominant 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 IP flows through a router

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

Estimating IP flows through a router

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

Src	Dst
DATA	

Estimating 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
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

Estimating IP flows through a router

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

Src	Dst
DATA	

Estimating 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 IP flows through a router

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

Src	Dst
DATA	

Estimating IP flows through a router

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

Estimating IP flows through a router

destination

$$
\text{source}\quad
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
$$

Src	Dst
DATA	

Estimating 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	**2**	0	0	0	0	0
source	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	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0

Estimating IP flows through a router

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

Src	Dst
DATA	

Estimating IP flows through a router

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

Estimating IP flows through a router

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

Src	Dst
DATA	

Estimating IP flows through a router

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

Estimating IP flows through a router

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

Src	Dst
DATA	

Estimating 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 IP flows through a router

$$
\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\texttt{destination}} \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
\texttt{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & \color{red}{\mathbf{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Src	Dst
DATA	

Estimating IP flows through a router

destination

$$\text{source}\quad\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

Estimating IP flows through a router

destination

	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
source	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 IP flows through a router

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

Estimating IP flows through a router

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

Src	Dst
DATA	

Estimating IP flows through a router

destination

	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
source	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

Estimating IP flows through a router

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

Src	Dst
DATA	

Estimating 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	**3**	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 IP flows through a router

$$
\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\texttt{destination}} \\
\hline
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
\texttt{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Src	Dst
DATA	

Estimating IP flows through a router

$$
\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\texttt{destination}} \\
\hline
 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\
\texttt{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Estimating 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 IP flows through a router

$$\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\texttt{destination}} \\
 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\
\texttt{source} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Src	Dst
DATA	

Estimating IP flows through a router

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

Estimating IP flows through a router

$$
\begin{array}{c|c}
 & \text{destination} \\
\hline
\text{source} &
\begin{matrix}
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{5} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
\end{array}
$$

Src	Dst
DATA	

Estimating IP flows through a router

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

Estimating IP flows through a router

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

Estimating IP flows through a router

Estimate the dominant IP flows through a router

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

Estimating IP flows through a router

Estimate the dominant IP flows through a router

destination

source

1

1

5

4 1

1

1

Estimating IP flows through a router

Estimate the dominant IP flows through a router

destination

source

1

1

5

4 1

1

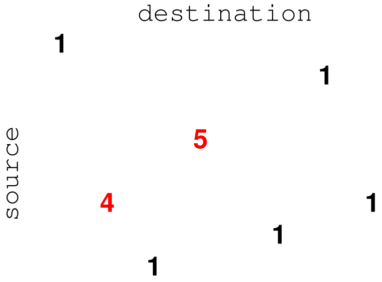
1

Trivial: store all distinct `IP` pairs

Space complexity: $\Theta(N)$

Estimating IP flows through a router

Estimate the dominant IP flows through a router



Trivial: store all distinct `IP` pairs

Space complexity: $\Theta(N)$

This lecture: solve in space $O(\log N)$

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

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 most frequent items in the set

`Geneva to NYC, coffee in Geneva`

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 most frequent items in the set

`Geneva to NYC, coffee in Geneva`

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 most frequent items in the set

`Geneva to NYC, coffee in Geneva`

	Trivial	This lecture
Solution	`hash<string> h;`	CountSketch
Space	`# of distinct items`	$O(\log N)$

Streaming model

	Trivial	This lecture
Solution	`hash<string> h;`	CountSketch
Space	`# of distinct items`	$O(\log N)$

Streaming model

	Trivial	This lecture
Solution	`hash<string> h;`	CountSketch
Space	`# of distinct items`	$O(\log N)$

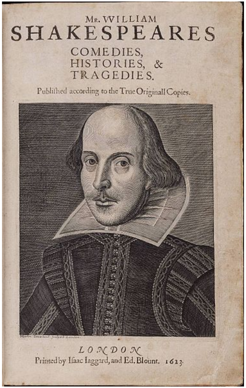
Are constants small?

Streaming model

	Trivial	This lecture
Solution	`hash<string> h;`	CountSketch
Space	`# of distinct items`	$O(\log N)$

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

Heavy hitters problem

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

  Assume N is known

- Output k most frequent items

  (Heavy hitters)

- Small storage: will get $O(k \log N)$

  Much better than storing all items!

Goal: design a small space data structure

$\textsc{FindTop}(S, k)$: returns top k most frequent items seen so far

Goal: design a small space data structure

FINDTOP(S, k): returns top k most frequent items seen so far

Useful to first design

POINTQUERY(S, i): processes stream, then for any query item i can return f_i=number of times item i appeared

Denote the number of times item i appears in the stream by f_i (frequency of i)

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$

Denote the number of times item i appears in the stream by f_i (frequency of i)

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$

$\boxed{\textsc{PointQuery}(S, i) \text{ in space } O(k \log N)?}$

Denote the number of times item i appears in the stream by f_i (frequency of i)

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$

$\boxed{\textsc{PointQuery}(S, i) \text{ in space } O(k \log N)?}$

Impossible in general...

Imagine a stream where all elements occur with about the same frequency

FindApproxTop(S, k, ε): returns set of k items such that $f_i \geq (1-\varepsilon) f_k$ for all reported i

ApproxPointQuery(S, i, ε): processes stream, then for any query item i can return approximation $\widehat{f_i} \in [f_i - \varepsilon f_k, f_i + \varepsilon f_k]$

$\text{FINDAPPROXTOP}(S, k, \varepsilon)$: returns set of k items such that $f_i \geq (1-\varepsilon) f_k$ for all reported i

$\text{APPROXPOINTQUERY}(S, i, \varepsilon)$: processes stream, then for any query item i can return approximation $\hat{f}_i \in [f_i - \varepsilon f_k, f_i + \varepsilon f_k]$

In this lecture: find most frequent (head) items **if they contribute the bulk of the stream** under some measure

In what follows: ApproxPointQuery in small space

Observe a stream of updates, maintain small space data structure

Task: after observing the stream, given $i \in \{1, 2, \ldots, m\}$, compute estimate $\widehat{f}_i$ of f_i

In what follows: ApproxPointQuery in small space

Observe a stream of updates, maintain small space data structure

Task: after observing the stream, given $i \in \{1, 2, \ldots, m\}$, compute estimate $\widehat{f}_i$ of f_i

To be specified:

- space complexity?
- quality of approximation?
- success probability?

1. Finding top k elements via (ApproxPointQuery)PointQuery
2. Basic version of ApproxPointQuery
3. ApproxPointQuery and the CountSketch algorithm

1. Finding top k elements via (ApproxPointQuery)PointQuery

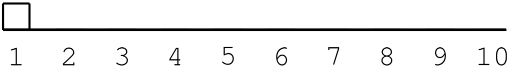
2. **Basic version of ApproxPointQuery**

3. ApproxPointQuery and the CountSketch algorithm

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$

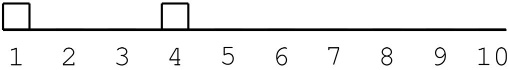
1 2 3 4 5 6 7 8 9 10

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



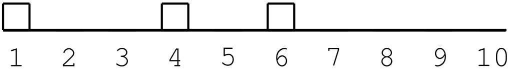
1

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



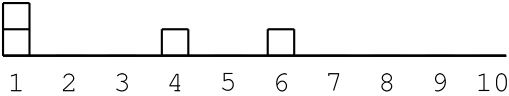
1 4

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



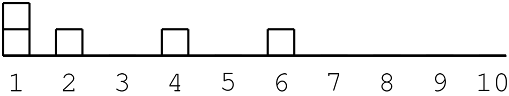
1 4 6

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



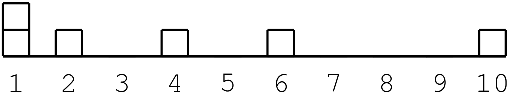
1 4 6 1

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



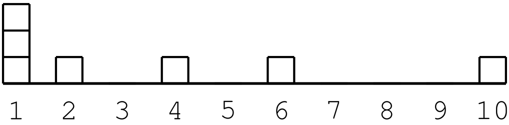
1 4 6 1 2

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



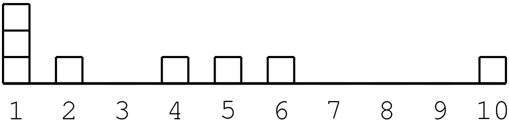
1 4 6 1 2 10

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



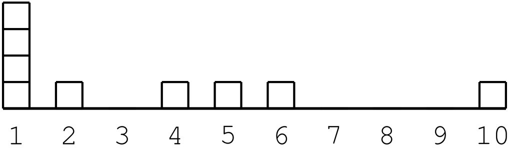
1 4 6 1 2 10 1

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



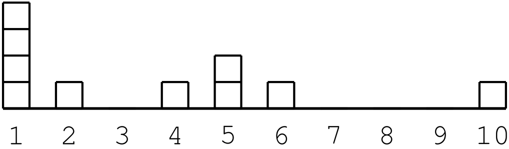
1 4 6 1 2 10 1 5

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



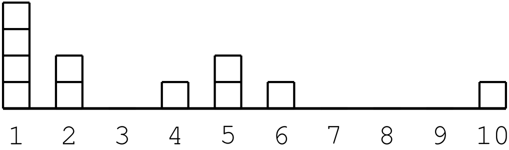
1 4 6 1 2 10 1 5 1

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



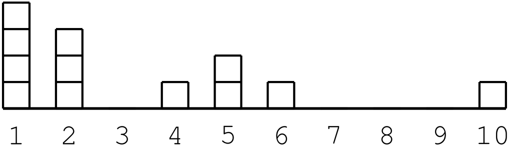
1 4 6 1 2 10 1 5 1 5

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



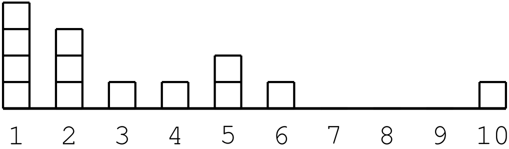
1 4 6 1 2 10 1 5 1 5 2

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



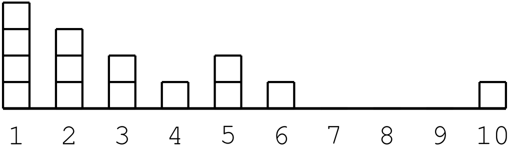
1 4 6 1 2 10 1 5 1 5 2 2

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



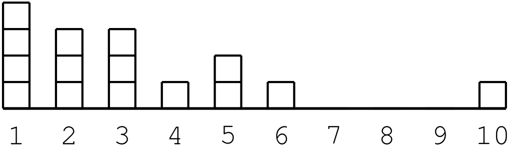
1 4 6 1 2 10 1 5 1 5 2 2 3

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



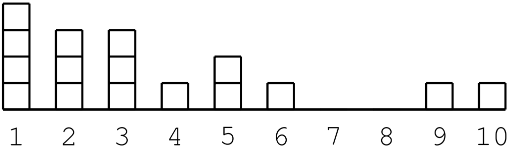
1 4 6 1 2 10 1 5 1 5 2 2 3 3

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



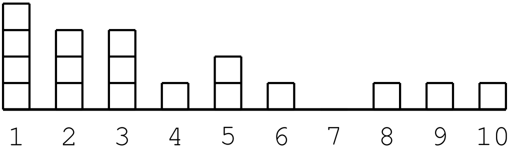
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



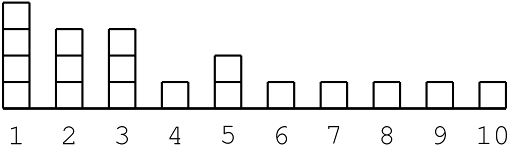
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



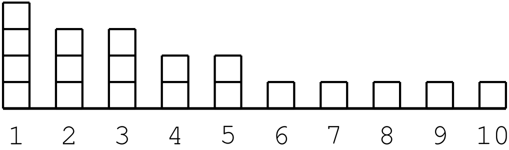
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9 8

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



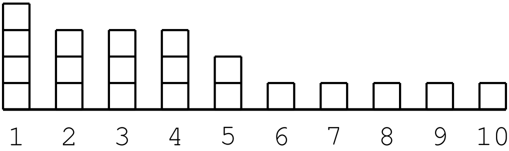
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9 8 7

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



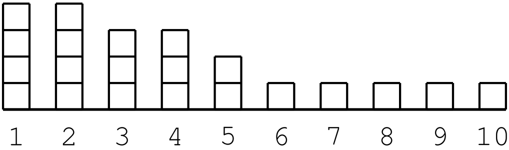
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9 8 7 4

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



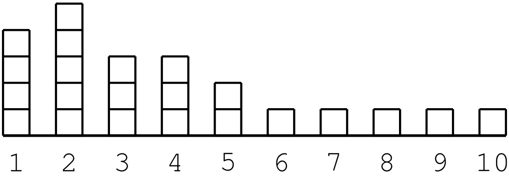
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9 8 7 4 4

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



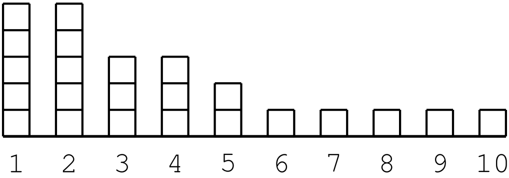
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9 8 7 4 4 2

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



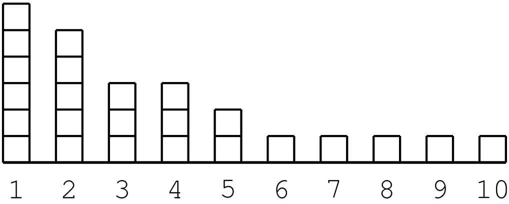
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9 8 7 4 4 2 2

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



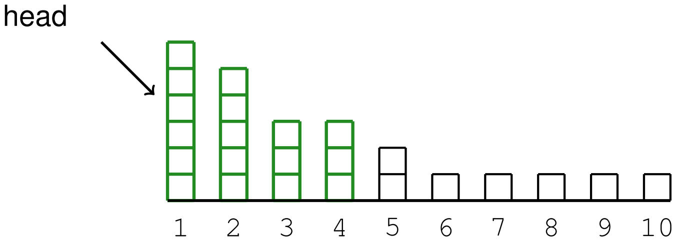
1 4 6 1 2 10 1 5 1 5 2 2 3 3 3 9 8 7 4 4 2 2 1

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



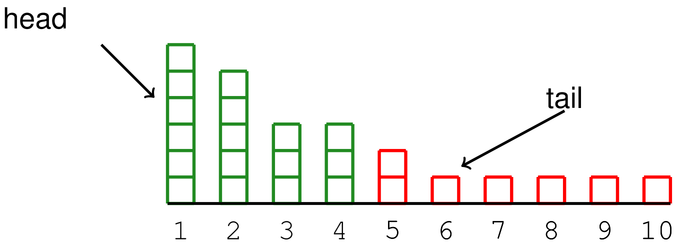
1 4 6 1 2 10 1 5 1 1 2 2 3 3 3 9 8 7 4 4 2 2 1 5

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



1 4 6 1 2 10 1 5 1 1 2 2 3 3 3 9 8 7 4 4 2 2 1 5

Assume elements are ordered by frequency: $f_1 \geq f_2 \geq \ldots \geq f_m$



1 4 6 1 2 10 1 5 1 1 2 2 3 3 3 9 8 7 4 4 2 2 1 5

Basic estimate

Will design a basic estimate with $O(1)$ space complexity, analyze precision

Basic estimate

Will design a basic estimate with $O(1)$ space complexity, analyze precision

Choose a hash function $s : [m] \to \{-1, +1\}$ uniformly at random

INITIALIZE

$C \leftarrow 0$

UPDATE(C, i)

$C \leftarrow C + s(i)$

Basic estimate

Will design a basic estimate with $O(1)$ space complexity, analyze precision

Choose a hash function $s : [m] \to \{-1, +1\}$ uniformly at random

INITIALIZE
$C \leftarrow 0$

UPDATE(C, i)
$C \leftarrow C + s(i)$

for every $p = 1, \ldots, N$ (every element in the stream)
UPDATE(C, i_p)
end for

Basic estimate

Will design a basic estimate with $O(1)$ space complexity, analyze precision

Choose a hash function $s:[m]\to\{-1,+1\}$ uniformly at random

INITIALIZE
$C \leftarrow 0$

UPDATE(C, i)
$C \leftarrow C + s(i)$

for every $p = 1,\ldots,N$ (every element in the stream)
UPDATE(C, i_p)
end for

ESTIMATE(C, i)
return $C \cdot s(i)$

$\boxed{\text{Show that } C\cdot s(i) \text{ is close to } f_i \text{ 'with high probability'?}}$

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$\boxed{\text{Show that } C \cdot s(i) \text{ is close to } f_i \text{ 'with high probability'?}}$

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

Show that $C \cdot s(i)$ is close to f_i 'with high probability'?

Two steps:

- show that $\mathbf{E}_s[C \cdot s(i)] = f_i$
  (so $C \cdot s(i)$ is an unbiased estimate of f_i)
- show that $\mathbf{Var}_s[C \cdot s(i)]$ is 'small'

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$C \cdot s(i) = \sum_{p=1}^{N} s(i_p) s(i)$$

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$C \cdot s(i) = \sum_{p=1}^{N} s(i_p) s(i) = \sum_{j \in [m]} f_j \cdot s(j) s(i)$$

Basic estimate:mean

UPDATE(C, i)

$C \leftarrow C + s(i)$

ESTIMATE(C, i)

return $C \cdot s(i)$

$$C \cdot s(i) = \sum_{p=1}^{N} s(i_p)s(i) = \sum_{j \in [m]} f_j \cdot s(j)s(i)$$

$$= f_i s(i)^2 + \sum_{j \in [m] \setminus i} f_j \cdot s(j)s(i)$$

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$
\begin{aligned}
C \cdot s(i) = \sum_{p=1}^{N} s(i_p)s(i) &= \sum_{j \in [m]} f_j \cdot s(j)s(i) \\
&= f_i s(i)^2 + \sum_{j \in [m] \setminus i} f_j \cdot s(j)s(i) \\
&= f_i + \sum_{j \in [m] \setminus i} f_j \cdot s(j)s(i) \quad \longleftarrow \text{random } \pm 1\text{'s}
\end{aligned}
$$

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$C \cdot s(i) \; = f_i + \sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)$$

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$\mathbf{E}[C \cdot s(i)] = f_i + \mathbf{E}[\sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)]$$

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$\begin{aligned}
\mathbf{E}[C \cdot s(i)] &= f_i + \mathbf{E}[\sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)] \\
&= f_i + \sum_{j \in [m] \setminus i} f_j \cdot \mathbf{E}[s(j)] \mathbf{E}[s(i)] \quad \text{(by independence of } s(i))
\end{aligned}$$

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$\begin{aligned}
\mathbf{E}[C \cdot s(i)] &= f_i + \mathbf{E}[\sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)] \\
&= f_i + \sum_{j \in [m] \setminus i} f_j \cdot \mathbf{E}[s(j)]\mathbf{E}[s(i)] \quad \text{(by independence of } s(i)\text{)} \\
&= f_i
\end{aligned}$$

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$\begin{aligned}
\mathbf{E}[C \cdot s(i)] &= f_i + \mathbf{E}[\sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)] \\
&= f_i + \sum_{j \in [m] \setminus i} f_j \cdot \mathbf{E}[s(j)] \mathbf{E}[s(i)] \quad \text{(by independence of } s(i)\text{)} \\
&= f_i
\end{aligned}$$

The mean is correct: our estimator is unbiased!

Basic estimate:mean

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$
\begin{aligned}
\mathbf{E}[C \cdot s(i)] &= f_i + \mathbf{E}[\sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)] \\
&= f_i + \sum_{j \in [m] \setminus i} f_j \cdot \mathbf{E}[s(j)] \mathbf{E}[s(i)] \quad \text{(by independence of } s(i)\text{)} \\
&= f_i
\end{aligned}
$$

$\boxed{\text{The mean is correct: our estimator is unbiased!}}$

Is the estimate $C \cdot s(i)$ close to f_i with high probability?

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

We have

$$C \cdot s(i) = f_i + \sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)$$

and

$$\mathbf{E}[C \cdot s(i)] = f_i.$$

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

We have

$$C \cdot s(i) = f_i + \sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)$$

and

$$\mathbf{E}[C \cdot s(i)] = f_i.$$

We need to bound

$$\begin{aligned} \mathbf{Var}(C \cdot s(i)) &= \mathbf{E}[(C \cdot s(i) - \mathbf{E}[C \cdot s(i)])^2] \\ &= \mathbf{E}[(C \cdot s(i) - f_i)^2] \\ &= \mathbf{E}\left[\left(\sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i)\right)^2\right] \end{aligned}$$

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$\begin{aligned}
(C \cdot s(i) - f_i)^2 &= \left(\sum_{j \in [m] \setminus i} f_j \cdot s(j) s(i) \right)^2 \\
&= \sum_{j \in [m] \setminus i} \sum_{j' \in [m] \setminus i} f_j f_{j'} \cdot s(j) s(j') \cdot s^2(i) \\
&= \sum_{j \in [m] \setminus i} \sum_{j' \in [m] \setminus i} f_j f_{j'} \cdot s(j) s(j')
\end{aligned}$$

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

$$
\begin{aligned}
\mathbf{E}[(C \cdot s(i) - f_i)^2] &= \mathbf{E}[\sum_{j \in [m] \setminus i} \sum_{j' \in [m] \setminus i} f_j f_{j'} \cdot s(j) s(j')] \\
&= \sum_{j \in [m] \setminus i} \sum_{j' \in [m] \setminus i} f_j f_{j'} \cdot \mathbf{E}[s(j) s(j')] \\
&= \sum_{j \in [m] \setminus i} f_j^2
\end{aligned}
$$

since

- $s(j)^2 = 1$ for all j
- $\mathbf{E}[s(j)s(j')] = \mathbf{E}[s(j)]\mathbf{E}[s(j')] = 0$ for $j \neq j'$.

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

We have proved that

$$\mathbf{Var}(C \cdot s(i)) = \mathbf{E}[(C \cdot s(i) - f_i)^2] = \sum_{j \in [m] \setminus i} f_j^2$$

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

We have proved that

$$\mathbf{Var}(C \cdot s(i)) = \mathbf{E}[(C \cdot s(i) - f_i)^2] = \sum_{j \in [m] \setminus i} f_j^2$$

By Chebyshev's inequality

$$\mathbf{Pr}\left[|C \cdot s(i) - f_i| \geq 8 \cdot \sqrt{\sum_{j \in [m] \setminus i} f_j^2} \right] \leq 1/64$$

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

We have proved that

$$\mathbf{Var}(C \cdot s(i)) = \mathbf{E}[(C \cdot s(i) - f_i)^2] = \sum_{j \in [m] \setminus i} f_j^2$$

By Chebyshev's inequality

$$\mathbf{Pr}\left[|C \cdot s(i) - f_i| \geq 8 \cdot \sqrt{\sum_{j \in [m] \setminus i} f_j^2} \right] \leq 1/64$$

So $C \cdot s(i)$ is close (?) to f_i with high probability

Basic estimate: variance

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

We have proved that

$$\mathbf{Var}(C \cdot s(i)) = \mathbf{E}[(C \cdot s(i) - f_i)^2] = \sum_{j \in [m] \setminus i} f_j^2$$

By Chebyshev's inequality

$$\mathbf{Pr}\left[|C \cdot s(i) - f_i| > \mathbf{8} \cdot \sqrt{\sum_{\mathbf{j} \in [\mathbf{m}] \setminus \mathbf{i}} \mathbf{f_j^2}} \right] \leq 1/64$$

$\boxed{\text{So } C \cdot s(i) \text{ is close } \textbf{(?)} \text{ to } f_i \text{ with high probability}}$

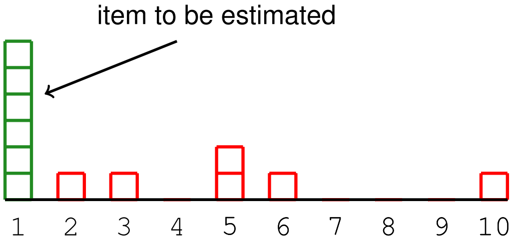
Basic estimate: summary

UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

Estimate f_i up to

$$8 \cdot \sqrt{\sum_{j \in [m] \setminus i} f_j^2}$$



Pro: works well for most frequent item, if other items are small

Basic estimate: summary

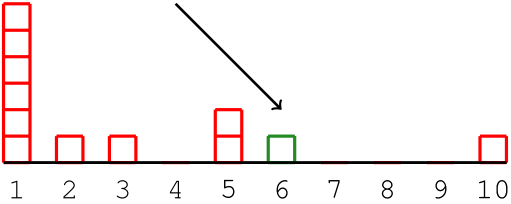
UPDATE(C, i)
$C \leftarrow C + s(i)$

ESTIMATE(C, i)
return $C \cdot s(i)$

Estimate f_i up to

$$8 \cdot \sqrt{\sum_{j \in [m] \setminus i} f_j^2}$$

item to be estimated



Pro: works well for most frequent item, if other items are small

Con: estimate for a small items contaminated by large items

1. Finding top k elements via (APPROX)POINTQUERY
2. Basic version of APPROXPOINTQUERY
3. APPROXPOINTQUERY and the COUNTSKETCH algorithm

1. Finding top k elements via (APPROX)POINTQUERY
2. Basic version of APPROXPOINTQUERY
3. **APPROXPOINTQUERY and the COUNTSKETCH algorithm**

ApproxPointQuery and CountSketch

CountSketch algorithm (Charikar, Chen, Farach-Colton'02)

Main ideas:

1. run basic estimate on subsampled/hashed stream
   (reduces variance)

ApproxPointQuery and CountSketch

CountSketch algorithm (Charikar, Chen, Farach-Colton'02)

Main ideas:

1. run basic estimate on subsampled/hashed stream (reduces variance)
2. aggregate independent estimates to boost confidence (take medians)

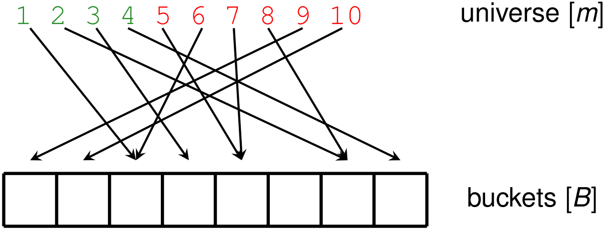
ApproxPointQuery and CountSketch

CountSketch algorithm (Charikar, Chen, Farach-Colton'02)

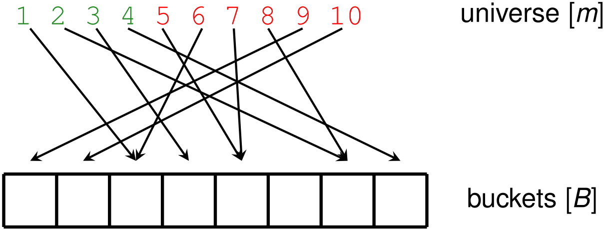
Main ideas:

1. run basic estimate on subsampled/hashed stream
   (reduces variance)

2. aggregate independent estimates to boost confidence
   (take medians)



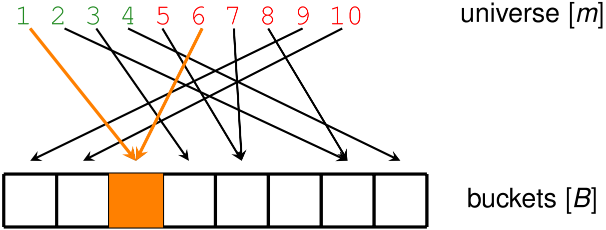
Hashing the items



Hashed into $B = 8$ buckets, get 8 subsampled streams

For item i its stream consists of $j \in [m]$ such that $h(j) = h(i)$

Hashing the items



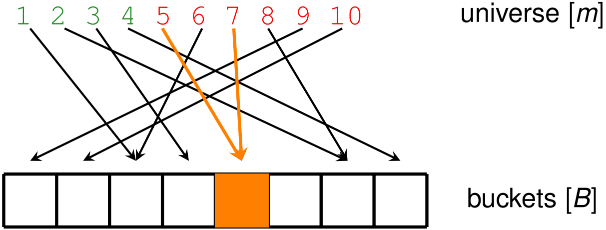
Hashed into $B = 8$ buckets, get 8 subsampled streams

For item i its stream consists of $j \in [m]$ such that $h(j) = h(i)$

For example,

- subsampled stream of item `1` is `{1, 6}`

Hashing the items



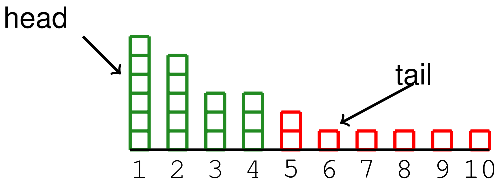
Hashed into $B = 8$ buckets, get 8 subsampled streams

For item i its stream consists of $j \in [m]$ such that $h(j) = h(i)$

For example,

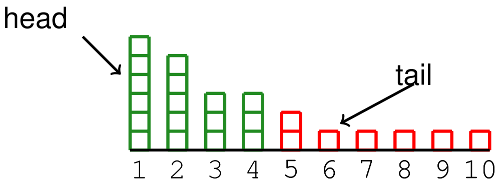
- subsampled stream of item 1 is {1, 6}
- subsampled stream of item 5 is {5, 7}

Note: hashing the universe $[m]$, not positions in the stream



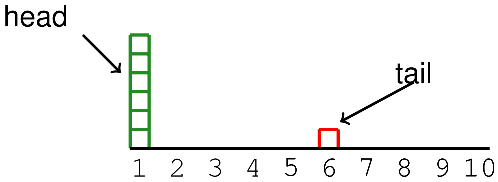
1 4 6 1 2 10 1 5 1 1 2 2 3 3 3 9 8 7 4 4 2 2 1 5

Note: hashing the universe $[m]$, not positions in the stream



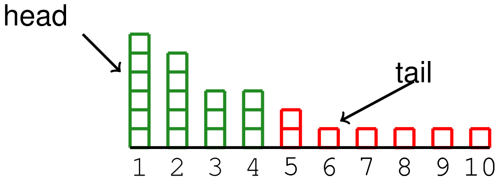
1 4 6 1 2 10 1 5 1 1 2 2 3 3 3 9 8 7 4 4 2 2 1 5

E.x. the subsampled stream of item 1 is {1, 6}



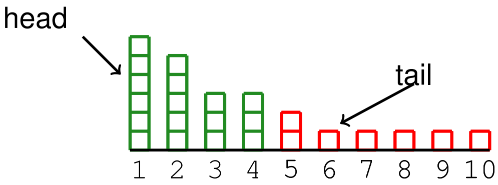
1 6 1 1 1 1 1

Note: hashing the universe $[m]$, not positions in the stream



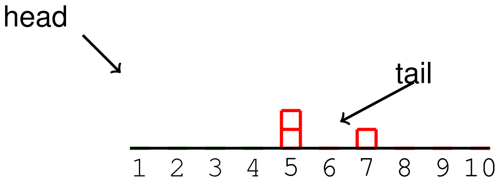
1 4 6 1 2 10 1 5 1 1 2 2 3 3 3 9 8 7 4 4 2 2 1 5

Note: hashing the universe [m], not positions in the stream



1 4 6 1 2 10 1 5 1 1 2 2 3 3 3 9 8 7 4 4 2 2 1 5

E.x. the subsampled stream of item 5 is {5, 7}



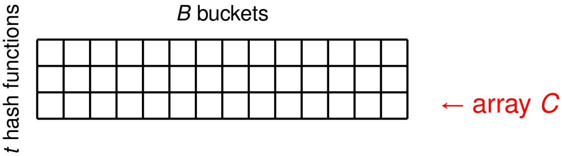
5 7 5

Final ApproxPointQuery

Choose

- t random hash functions $h_1, h_2, \ldots, h_t$ from items $[m]$ to $B \approx k$ buckets $\{1, 2, \ldots, B\}$

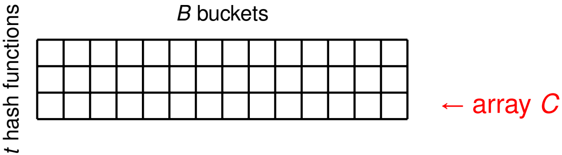
- t random hash functions $s_1, s_2, \ldots, s_t$ from items $[m]$ to $\{-1, +1\}$



Final ApproxPointQuery

Choose

- t random hash functions $h_1, h_2, \ldots, h_t$ from items $[m]$ to $B \approx k$ buckets $\{1, 2, \ldots, B\}$
- t random hash functions $s_1, s_2, \ldots, s_t$ from items $[m]$ to $\{-1, +1\}$



The algorithm runs t independent copies of basic estimate:

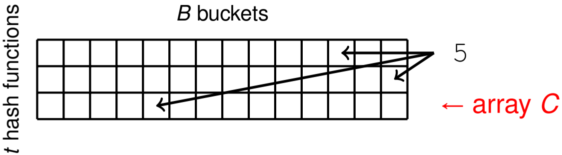
UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Final ApproxPointQuery

Choose

- t random hash functions $h_1, h_2, \dots, h_t$ from items $[m]$ to $B \approx k$ buckets $\{1, 2, \dots, B\}$
- t random hash functions $s_1, s_2, \dots, s_t$ from items $[m]$ to $\{-1, +1\}$



The algorithm runs t independent copies of basic estimate:

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

UPDATE(C, i)
for $r \in [1:t]$
$C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

UPDATE(C, i)

for $r \in [1:t]$

$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$

end for

ESTIMATE(C, i)

return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Lemma

If $B \geq 8 \max\left\{k, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_k)^2}\right\}$ *and* $t = O(\log N)$, *then for every* $i \in [m]$

$$|\text{ESTIMATE}(C, i) - f_i| \leq \varepsilon f_k$$

at every point in the stream whp.

UPDATE(C, i)
for $r \in [1:t]$
$C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Lemma

If $B \geq 8 \max\left\{k, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_k)^2}\right\}$ *and* $t = O(\log N)$*, then for every* $i \in [m]$

$$|\text{ESTIMATE}(C, i) - f_i| \leq \varepsilon f_k$$

at every point in the stream whp.

Space complexity is $O(B \log N)$

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Lemma

If $B \geq 8 \max\left\{k, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_k)^2}\right\}$ *and* $t = O(\log N)$, *then for every* $i \in [m]$

$$|\text{ESTIMATE}(C, i) - f_i| \leq \varepsilon f_k$$

at every point in the stream whp.

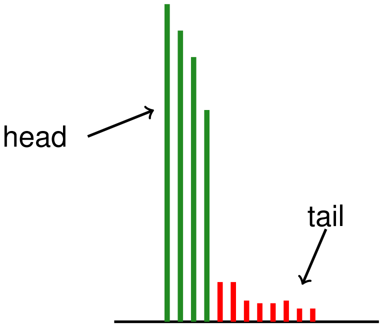
Space complexity is $O(B \log N)$

How large is B?

Space complexity

$$\text{Set } B = 8\max\left\{k, \frac{32\sum_{j\in TAIL} f_j^2}{(\varepsilon f_k)^2}\right\}$$

Note that $B = O(k/\varepsilon^2)$ if $\frac{1}{k}\sum_{j\in TAIL} f_j^2 = O(f_k^2)$



Note: if $B \geq k$, can detect elements with counts above $O\left(\sqrt{\frac{1}{B}\cdot\sum_{j\in TAIL} f_j^2}\right)$

Space complexity

Set $k = 1$. Suppose that `1` appears $\sqrt{N}$ times in the stream, and other $N - \sqrt{N}$ elements are distinct

Space complexity

Set $k = 1$. Suppose that `1` appears $\sqrt{N}$ times in the stream, and other $N - \sqrt{N}$ elements are distinct

Then $f_1 = \sqrt{N}$, $f_i = 1$ for $i = 2, N - \sqrt{N}$.

Set $B = 8 \max \left\{ 1, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_1)^2} \right\}$

Space complexity

Set $k = 1$. Suppose that `1` appears $\sqrt{N}$ times in the stream, and other $N - \sqrt{N}$ elements are distinct

Then $f_1 = \sqrt{N}$, $f_i = 1$ for $i = 2, N - \sqrt{N}$.

$$\text{Set } B = 8\max\left\{1, \frac{32\sum_{j\in TAIL} f_j^2}{(\varepsilon f_1)^2}\right\}$$

$$\boxed{\text{We have } \sum_{j\in TAIL} f_j^2 = N - \sqrt{N} \le N, \text{ and } f_1^2 = N}$$

Space complexity

Set $k = 1$. Suppose that `1` appears $\sqrt{N}$ times in the stream, and other $N - \sqrt{N}$ elements are distinct

Then $f_1 = \sqrt{N}$, $f_i = 1$ for $i = 2, N - \sqrt{N}$.

$$\text{Set } B = 8 \max\left\{1, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_1)^2}\right\}$$

$$\boxed{\text{We have } \sum_{j \in TAIL} f_j^2 = N - \sqrt{N} \leq N, \text{ and } f_1^2 = N}$$

$$\text{So } B = 8 \max\left\{1, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_1)^2}\right\} = O(1/\varepsilon^2) \text{ suffices}$$

Space complexity

Set $k = 1$. Suppose that `1` appears $\sqrt{N}$ times in the stream, and other $N - \sqrt{N}$ elements are distinct

Then $f_1 = \sqrt{N}$, $f_i = 1$ for $i = 2, N - \sqrt{N}$.

$$\text{Set } B = 8 \max\left\{1, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_1)^2}\right\}$$

$$\boxed{\text{We have } \sum_{j \in TAIL} f_j^2 = N - \sqrt{N} \le N, \text{ and } f_1^2 = N}$$

$$\text{So } B = 8 \max\left\{1, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_1)^2}\right\} = O(1/\varepsilon^2) \text{ suffices}$$

Remarkable, as `1` appears only in $\sqrt{N}$ positions out of N: a vanishingly small fraction of positions!

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Lemma

If $B \geq 8 \max\left\{k, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_k)^2}\right\}$ *and* $t \geq A \log N$ *for an absolute constant* $A > 0$*, then for every* $i \in [m]$

$$\left|\text{ESTIMATE}(C, i) - f_i\right| \leq \varepsilon f_k$$

with high probability.

(f_i is the frequency of i)

UPDATE(C, i)
for $r \in [1:t]$
$C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Variance of estimate for i from r-th row:

$$\sum_{j \neq i : \mathbf{h_r(j) = h_r(i)}} f_j^2$$

UPDATE(C, i)
for $r \in [1:t]$
  $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Variance of estimate for i from r-th row:

$$\sum_{j \neq i : \mathbf{h_r(j) = h_r(i)}} f_j^2$$

Show that

$$\sum_{j \neq i : \mathbf{h_r(j) = h_r(i)}} f_j^2 = O(1/B) \sum_{j \in TAIL, j \neq i} f_j^2$$

with high constant probability.

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2$$

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2$$

For each $r \in [1 : t]$ and each item $i \in [m]$ define three events:

- NO-COLLISIONS$_r(i)$ – i does not collide with any of the head items under hashing r

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2$$

For each $r \in [1:t]$ and each item $i \in [m]$ define three events:

- NO-COLLISIONS$_r(i)$ – i does not collide with any of the head items under hashing r
- SMALL-VARIANCE$_r(i)$ – i does not collide with too many of tail items under hashing r

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2$$

For each $r \in [1 : t]$ and each item $i \in [m]$ define three events:

- $\textsc{No-Collisions}_r(i)$ – i does not collide with any of the head items under hashing r
- $\textsc{Small-Variance}_r(i)$ – i does not collide with too many of tail items under hashing r
- $\textsc{Small-Deviation}_r(i)$ – success event from basic analysis

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2$$

For each $r \in [1:t]$ and each item $i \in [m]$ define three events:

- NO-COLLISIONS$_r(i)$ – i does not collide with any of the head items under hashing r
- SMALL-VARIANCE$_r(i)$ – i does not collide with too many of tail items under hashing r
- SMALL-DEVIATION$_r(i)$ – success event from basic analysis

Show that all three events hold simultaneously with probability strictly bigger than $1/2$ – so median gives good estimate

(No) collisions with head items

$\textsc{No-Collisions}_r(i)$:=event that

$$\{j \in HEAD \setminus i : h_r(j) = h_r(i)\} = \emptyset,$$

i.e. that i collides with none of top k elements under h_r.

(No) collisions with head items

$\textsc{No-Collisions}_r(i)$:=event that

$$\{j \in HEAD \setminus i : h_r(j) = h_r(i)\} = \emptyset,$$

i.e. that i collides with none of top k elements under h_r.

For every $j \neq i$ and every $r \in [1 : t]$

$$\mathbf{Pr}[h_r(i) = h_r(j)] \leq 1/B$$

(No) collisions with head items

$\text{NO-COLLISIONS}_r(i)$:=event that

$$\{j \in HEAD \setminus i : h_r(j) = h_r(i)\} = \emptyset,$$

i.e. that i collides with none of top k elements under h_r.

For every $j \neq i$ and every $r \in [1:t]$

$$\mathbf{Pr}[h_r(i) = h_r(j)] \leq 1/B$$

Suppose that $B \geq 8k$. Then by the union bound

$$\begin{aligned}\mathbf{Pr}[\text{NO-COLLISIONS}_r(i)] &\geq 1 - k/B \\ &\geq 1 - 1/8\end{aligned}$$

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2$$

For each $r \in [1:t]$ and each item $i \in [m]$ define three events:

- NO-COLLISIONS$_r(i)$ – i does not collide with any of the head items under hashing r
- SMALL-VARIANCE$_r(i)$ – i does not collide with too many of tail items under hashing r
- SMALL-DEVIATION$_r(i)$ – success event from basic analysis

Show that all three events hold simultaneously with probability strictly bigger than $1/2$ – so median gives good estimate

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ \mathbf{h_r(j) = h_r(i)}}} f_j^2$$

For each $r \in [1 : t]$ and each item $i \in [m]$ define three events:

- NO-COLLISIONS$_r(i)$ – i does not collide with any of the head items under hashing r
- **SMALL-VARIANCE$_r(i)$ – i does not collide with too many of tail items under hashing r**
- SMALL-DEVIATION$_r(i)$ – success event from basic analysis

Show that all three events hold simulaneously with probability strictly bigger than $1/2$ – so median gives good estimate

Small variance from tail elements

SMALL-VARIANCE$_r(i)$:=event that

$$\sum_{\substack{j \in TAIL, j \neq i \\ h_r(j)=h_r(i)}} f_j^2 \leq \frac{8}{B} \sum_{j \in TAIL} f_j^2$$

Small variance from tail elements

SMALL-VARIANCE$_r(i)$:=event that

$$\sum_{\substack{j \in TAIL, j \neq i \\ h_r(j)=h_r(i)}} f_j^2 \leq \frac{8}{B} \sum_{j \in TAIL} f_j^2$$

For every $i, j \in [m], i \neq j$ and $r \in [1:t]$

$$\mathbf{Pr}_{h_r}[h_r(i) = h_r(j)] = 1/B \quad (B \text{ is the number of buckets})$$

Small variance from tail elements

SMALL-VARIANCE$_r(i)$:=event that

$$\sum_{\substack{j \in TAIL, j \neq i \\ h_r(j)=h_r(i)}} f_j^2 \leq \frac{8}{B} \sum_{j \in TAIL} f_j^2$$

For every $i, j \in [m], i \neq j$ and $r \in [1:t]$

$$\mathbf{Pr}_{h_r}[h_r(i) = h_r(j)] = 1/B \quad (B \text{ is the number of buckets})$$

So by linearity of expectation

$$\mathbf{E}\left[\sum_{\substack{j \in TAIL, j \neq i \\ h_r(j)=h_r(i)}} f_j^2\right] = \sum_{j \in TAIL, j \neq i} f_j^2 \cdot \mathbf{Pr}_{h_r}[h_r(i) = h_r(j)]$$

Small variance from tail elements

SMALL-VARIANCE$_r(i)$:=event that

$$\sum_{\substack{j \in TAIL, j \neq i \\ h_r(j)=h_r(i)}} f_j^2 \leq \frac{8}{B} \sum_{j \in TAIL} f_j^2$$

For every $i, j \in [m], i \neq j$ and $r \in [1:t]$

$$\mathbf{Pr}_{h_r}[h_r(i) = h_r(j)] = 1/B \quad (B \text{ is the number of buckets})$$

So by linearity of expectation

$$\mathbf{E}\left[\sum_{\substack{j \in TAIL, j \neq i \\ h_r(j)=h_r(i)}} f_j^2\right] = \sum_{j \in TAIL, j \neq i} f_j^2 \cdot \mathbf{Pr}_{h_r}[h_r(i) = h_r(j)]$$

$$\leq \frac{1}{B} \sum_{j \in TAIL} f_j^2$$

We proved that

$$\mathbf{E}\left[\sum_{\substack{j\in \mathit{TAIL}, j\neq i\\ h_r(j)=h_r(i)}} f_j^2\right] \leq \frac{1}{B}\sum_{j\in \mathit{TAIL}} f_j^2$$

By Markov's inequality one has, for every i and every r,

$$\mathbf{Pr}[\textsc{Small-Variance}_r(i)] \geq 1 - 1/8$$

$\text{NO-COLLISIONS}_r(i)$ and $\text{SMALL-VARIANCE}_r(i)$: recap

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ h_r(j) = h_r(i)}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ h_r(j) = h_r(i)}} f_j^2$$

Conditioned on $\text{NO-COLLISIONS}_r(i)$ and $\text{SMALL-VARIANCE}_r(i)$

$\textsc{No-Collisions}_r(i)$ and $\textsc{Small-Variance}_r(i)$: recap

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ h_r(j) = h_r(i)}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ h_r(j) = h_r(i)}} f_j^2$$

Conditioned on $\textsc{No-Collisions}_r(i)$ and $\textsc{Small-Variance}_r(i)$

- first term is zero

$\text{NO-COLLISIONS}_r(i)$ and $\text{SMALL-VARIANCE}_r(i)$: recap

Consider contribution of head and tail items separately:

$$\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{\substack{j \in HEAD, j \neq i \\ h_r(j) = h_r(i)}} f_j^2 + \sum_{\substack{j \in TAIL, j \neq i \\ h_r(j) = h_r(i)}} f_j^2$$

Conditioned on $\text{NO-COLLISIONS}_r(i)$ and $\text{SMALL-VARIANCE}_r(i)$

- first term is zero
- second term is at most

$$\frac{8}{B} \sum_{j \in TAIL} f_j^2$$

Small deviation event

SMALL-DEVIATION$_r(i)$=event that

$$(C[r, h_r(i)] \cdot s_r(i) - f_i)^2 \leq 8\mathbf{Var}(C[r, h_r(i)] \cdot s_r(i)).$$

Small deviation event

$\textsc{Small-Deviation}_r(i)$=event that

$$(C[r,h_r(i)]\cdot s_r(i)-f_i)^2\leq 8\mathbf{Var}(C[r,h_r(i)]\cdot s_r(i)).$$

By Markov's inequality one has, for every i and every r,

$$\mathbf{Pr}[\textsc{Small-Deviation}_r(i)]\geq 1-1/8$$

$$\mathbf{Pr}[\textsc{Small-Variance}_r(i)] \geq 1 - 1/8$$

$$\mathbf{Pr}[\textsc{No-Collisions}_r(i)] \geq 1 - 1/8$$

$$\mathbf{Pr}[\textsc{Small-Deviation}_r(i)] \geq 1 - 1/8$$

So by the union bound

$$\mathbf{Pr}[\textsc{Small-Variance}_r(i) \text{ and } \textsc{No-Collisions}_r(i) \\ \text{ and } \textsc{Small-Deviation}_r(i)] \geq 5/8.$$

For every $p \in [1:N]$ let $f_i(p) :=$frequency of i up to position p

Lemma

If $B \geq 8 \max\left\{k, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_k)^2}\right\}$ *and* $t \geq A \log N$ *for an absolute constant* $A > 0$*, then with probability* $\geq 1 - 1/N^3$ *for every* $i \in [m]$

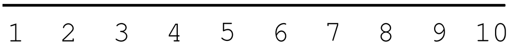
$$\left|\textsc{Estimate}(C, i) - f_i(p)\right| \leq \varepsilon f_k$$

at the end of the stream.

Remarks, related results, open problems

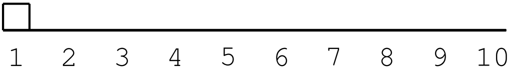
UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

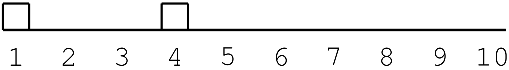
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1

UPDATE(C, i)
for $r \in [1:t]$
  $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

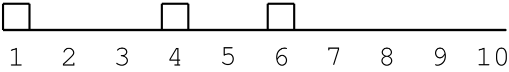
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4

UPDATE(C, i)
for $r \in [1 : t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

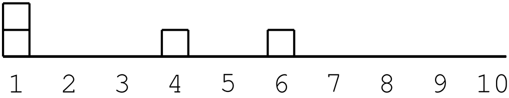
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

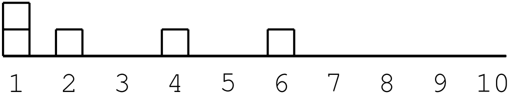
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

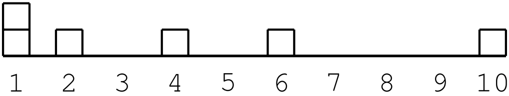
ESTIMATE(C, i)
return median$_r$ $\{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

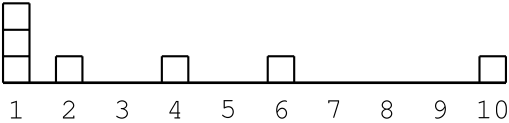
ESTIMATE(C, i)
return median$_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10

$\textsc{Update}(C, i)$
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

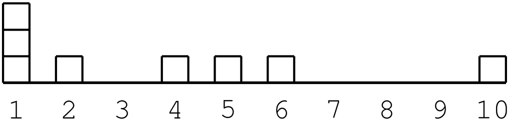
$\textsc{Estimate}(C, i)$
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

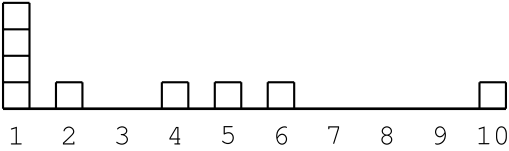
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

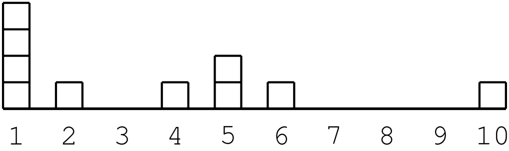
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5

UPDATE(C, i)

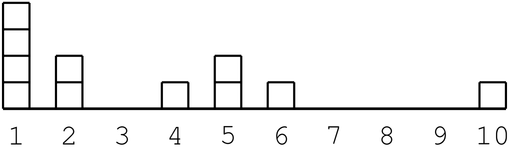
for $r \in [1:t]$

$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$

end for

ESTIMATE(C, i)

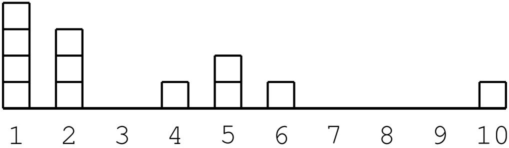
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2

UPDATE(C, i)
for $r \in [1:t]$
$C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

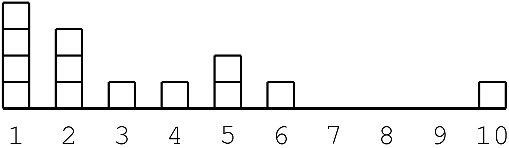
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3

UPDATE(C, i)

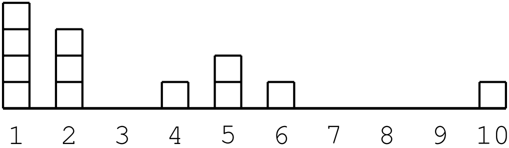
for $r \in [1 : t]$

$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$

end for

ESTIMATE(C, i)

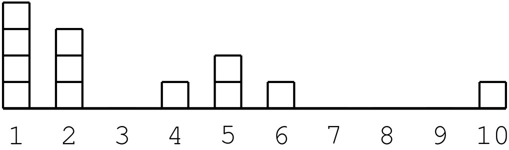
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3

UPDATE(C, i)
for $r \in [1:t]$
$C[r,h_r(i)] \leftarrow C[r,h_r(i)] + s_r(i)$
end for

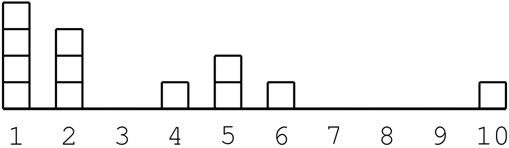
ESTIMATE(C, i)
return $\text{median}_r \{C[r,h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 $\boxed{-3}$

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

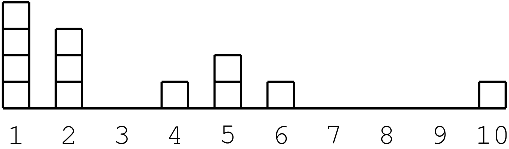
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 $\boxed{-3}$

UPDATE(C, i)
for $r \in [1 : t]$
$$C[r, h_r(i)] \leftarrow C[r, h_r(i)] - s_r(i)$$
end for

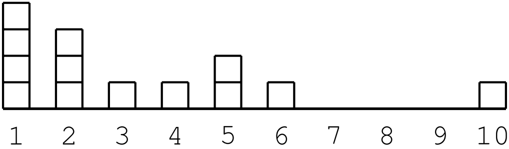
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 $\boxed{-3}$

UPDATE(C, i)
for $r \in [1:t]$
$$C[r,h_r(i)] \leftarrow C[r,h_r(i)] + s_r(i)$$
end for

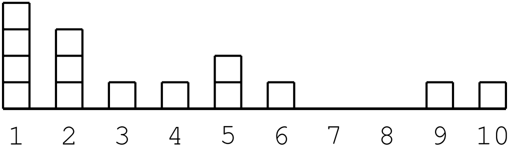
ESTIMATE(C, i)
return $\text{median}_r \{C[r,h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3

UPDATE(C, i)
for $r \in [1:t]$
$$C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$$
end for

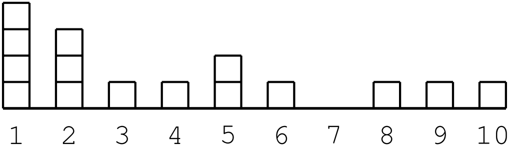
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

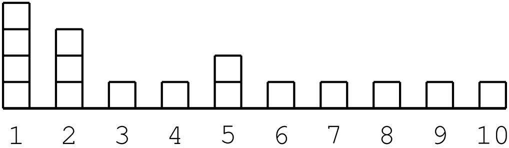
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9 8

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

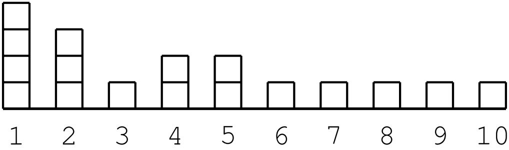
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9 8 7

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

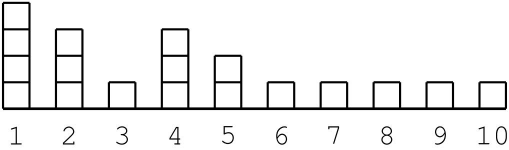
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9 8 7 4

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

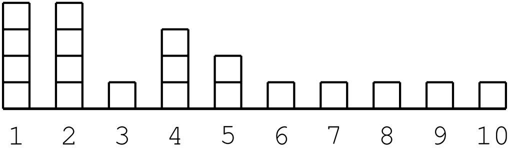
ESTIMATE(C, i)
return median$_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9 8 7 4 4

UPDATE(C, i)
for $r \in [1:t]$
$C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

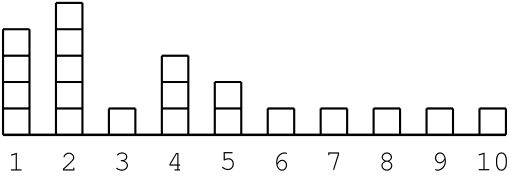
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9 8 7 4 4 2

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

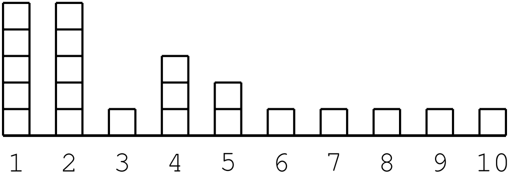
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9 8 7 4 4 2 2

UPDATE(C, i)
for $r \in [1:t]$
$\quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

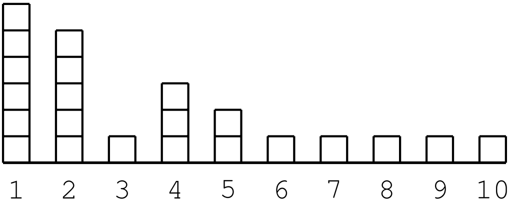
ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$



1 4 6 1 2 10 1 5 1 5 2 2 3 −3 3 9 8 7 4 4 2 2 1

UPDATE(C, i)
for $r \in [1:t]$
  $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

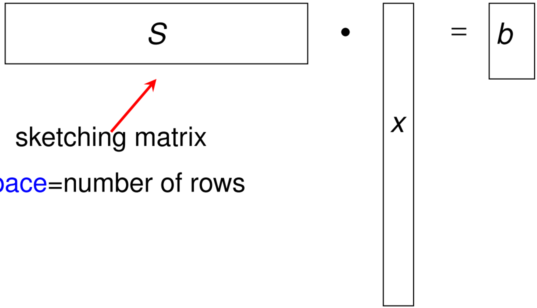


1 4 6 1 2 10 1 5 1 1 2 2 3 −3 3 9 8 7 4 4 2 2 1 5

UPDATE(C, i)
for $r \in [1:t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

ESTIMATE(C, i)
return median$_r \{C[r, h_r(i)] \cdot s_r(i)\}$

Sketching: take (randomized) linear measurements of the input

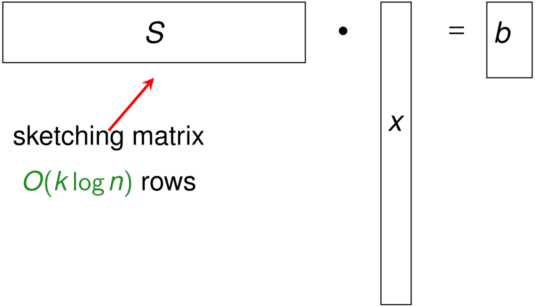


sketching matrix

space=number of rows

Easy to maintain sketch in dynamic streams
(insertions and deletions)

Sparse recovery



Let S be a CountSketch matrix with $O(k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ if $\hat{x} = \textsc{Est}(Sx)$, then whp

$$\|x - \hat{x}\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Sparse recovery

Let S be a CountSketch matrix with $O(k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ if $\widehat{x} = \textsc{Est}(Sx)$, then whp

$$\left\| x - \widehat{x} \right\|_\infty \leq \frac{1}{\sqrt{k}} \| x_{TAIL} \|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Sparse recovery

Let S be a CountSketch matrix with $O(k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ if $\hat{x} = \textsc{Est}(Sx)$, then whp

$$\|x - \hat{x}\|_\infty \le \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Observation 1: # of measurements is optimal for ℓ_∞/ℓ_2 guarantee above

(capacity of the Gaussian channel, i.e. Shannon-Hartley theorem, – see Do Ba, Indyk, Price, Woodruff'10)

Sparse recovery

Let S be a CountSketch matrix with $O(k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ if $\widehat{x} = \textsc{Est}(Sx)$, then whp

$$\|x - \widehat{x}\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Observation 1: # of measurements is optimal for ℓ_∞/ℓ_2 guarantee above

(capacity of the Gaussian channel, i.e. Shannon-Hartley theorem, – see Do Ba, Indyk, Price, Woodruff'10)

Observation 2: ℓ_2/ℓ_2 sparse recovery guarantee follows:

$$\|x - \widehat{x}\|_2 = O(1) \cdot \|x_{TAIL}\|_2.$$

Sparse recovery

Let S be a CountSketch matrix with $O(k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ if $\widehat{x} = \textsc{Est}(Sx)$, then whp

$$\|x - \widehat{x}\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Observation 1: # of measurements is optimal for ℓ_∞/ℓ_2 guarantee above

(capacity of the Gaussian channel, i.e. Shannon-Hartley theorem, – see Do Ba, Indyk, Price, Woodruff'10)

Sparse recovery

Let S be a CountSketch matrix with $O(k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ if $\widehat{x} = \textsc{Est}(Sx)$, then whp

$$\left\|x - \widehat{x}\right\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Observation 1: # of measurements is optimal for ℓ_∞/ℓ_2 guarantee above

(capacity of the Gaussian channel, i.e. Shannon-Hartley theorem, – see Do Ba, Indyk, Price, Woodruff'10)

Observation 2: ℓ_2/ℓ_2 sparse recovery guarantee follows:

$$\left\|x - \widehat{x}\right\|_2 = O(1) \cdot \min_{k-sparse\ x'} \|x - x'\|_2.$$

Sparse recovery

Let S be a CountSketch matrix with $O(\frac{1}{\varepsilon^2} k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ *if* $\widehat{x} = \textsc{Est}(Sx)$*, then whp*

$$\|x - \widehat{x}\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Observation 1: # of measurements is optimal for ℓ_∞/ℓ_2 guarantee above

(capacity of the Gaussian channel, i.e. Shannon-Hartley theorem, – see Do Ba, Indyk, Price, Woodruff'10)

Observation 2: ℓ_2/ℓ_2 sparse recovery guarantee follows:

$$\|x - \widehat{x}\|_2 = (1+\varepsilon) \cdot \min_{k-sparse\ x'} \|x - x'\|_2.$$

Sparse recovery

Let S be a COUNTSKETCH matrix with $O(k \log n)$ rows

Lemma

For every $x \in \mathbb{R}^n$ if $\widehat{x} = \textsc{Est}(Sx)$, then whp

$$\left\| x - \widehat{x} \right\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2.$$

(x_{TAIL} – x with largest k elements zeroed out)

Observation 3: if $\|x\|_0 \leq k$, then $x_{TAIL} = 0$ and $\textsc{Est}(Sx) = x$ whp

Exact sparse recovery: a k-sparse vector can be recovered from $O(k)$ linear measurements

CountSketch for matrices?

Input: r parties hold vectors $x_1, \dots, x_r \in \mathbb{R}^n$

each party sends $O(B \log n)$ bits to coordinator

(assume shared randomness)

CountSketch for matrices?

Input: r parties hold vectors $x_1, \ldots, x_r \in \mathbb{R}^n$

each party sends $O(B \log n)$ bits to coordinator

(assume shared randomness)

Output: find largest entries in $A = \sum_{i=1}^{r} x_i x_i^T$

more precisely, output approximation $\widehat{A}$

$$\left|\widehat{A_{ij}} - A_{ij}\right| = \frac{O(1)}{\sqrt{B}} \|A\|_F.$$

CountSketch for matrices?

Input: r parties hold vectors $x_1, \ldots, x_r \in \mathbb{R}^n$

each party sends $O(B \log n)$ bits to coordinator

(assume shared randomness)

Output: find largest entries in $A = \sum_{i=1}^{r} x_i x_i^T$

more precisely, output approximation $\widehat{A}$

$$\left|\widehat{A_{ij}} - A_{ij}\right| = \frac{O(1)}{\sqrt{B}} \|A\|_F.$$

$x_1 \in \mathbb{R}^n$

$x_2 \in \mathbb{R}^n$

$x_3 \in \mathbb{R}^n$ Coordinator $\longrightarrow$ $\widehat{A}$

$\vdots$

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

CountSketch for matrices?

Input: r parties hold vectors $x_1, \dots, x_r \in \mathbb{R}^n$

each party sends $O(B \log n)$ bits to coordinator

(assume shared randomness)

Output: find largest entries in $A = \sum_{i=1}^{r} x_i x_i^T$

more precisely, output approximation $\widehat{A}$

$$\left|\widehat{A_{ij}} - A_{ij}\right| = \frac{O(1)}{\sqrt{B}} \|A\|_F.$$

$x_1 \in \mathbb{R}^n$

$x_2 \in \mathbb{R}^n$

$x_3 \in \mathbb{R}^n \longrightarrow$ Coordinator $\longrightarrow \widehat{A}$

$\vdots$

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

Every party i sends $\textsc{CountSketch}(x_i x_i^T)$ into $O(B)$ buckets

(slow)

Define

$$A = \sum_{i=1}^{r} x_i x_i^T \in \mathbb{R}^{n \times n}.$$

Define

$$A = \sum_{i=1}^{r} x_i x_i^T \in \mathbb{R}^{n \times n}.$$

Hash function

$$h : [n] \times [n] \to [B]$$

and random signs

$$s : [n] \times [n] \to \{-1, +1\}.$$

Define

$$A = \sum_{i=1}^{r} x_i {x_i}^T \in \mathbb{R}^{n \times n}.$$

Hash function

$$h : [n] \times [n] \to [B]$$

and random signs

$$s : [n] \times [n] \to \{-1, +1\}.$$

$$(Sx)_b = \sum_{i,j \in [n] : h(i,j) = b} s(i,j) \cdot x_i x_j.$$

Define

$$A = \sum_{i=1}^{r} x_i x_i^T \in \mathbb{R}^{n \times n}.$$

Hash function

$$h : [n] \times [n] \to [B]$$

and random signs

$$s : [n] \times [n] \to \{-1, +1\}.$$

$$(Sx)_b = \sum_{i,j \in [n] : h(i,j) = b} s(i,j) \cdot x_i x_j.$$

$\textsc{CountSketch}(x_i x_i^T)$ takes n^2 time to compute...

Define

$$A = \sum_{i=1}^{r} x_i x_i^T \in \mathbb{R}^{n \times n}.$$

Hash function

$$h : [n] \times [n] \to [B]$$

and random signs

$$s : [n] \times [n] \to \{-1, +1\}.$$

$$(Sx)_b = \sum_{i,j \in [n] : h(i,j) = b} s(i,j) \cdot x_i x_j.$$

$\textsc{CountSketch}(x_i x_i^T)$ takes n^2 time to compute...

$\boxed{\text{Make hash functions 'separable'?}}$

CountSketch for matrices?

Input: r parties hold vectors $x_1, \ldots, x_r \in \mathbb{R}^n$

each party sends $O(B \log n)$ bits to coordinator

(assume shared randomness)

Output: find largest entries in $A = \sum_{i=1}^{r} x_i x_i^T$

more precisely, output approximation $\widehat{A}$

$$\left|\widehat{A_{ij}} - A_{ij}\right| = \frac{O(1)}{\sqrt{B}} \|A\|_F.$$

$x_1 \in \mathbb{R}^n$

$x_2 \in \mathbb{R}^n$

$x_3 \in \mathbb{R}^n \longrightarrow$ Coordinator $\longrightarrow \widehat{A}$

$\vdots$

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

CountSketch for matrices?

Input: r parties hold vectors $x_1, \dots, x_r \in \mathbb{R}^n$

each party sends $O(B \log n)$ bits to coordinator

(assume shared randomness)

Output: find largest entries in $A = \sum_{i=1}^r x_i x_i^T$

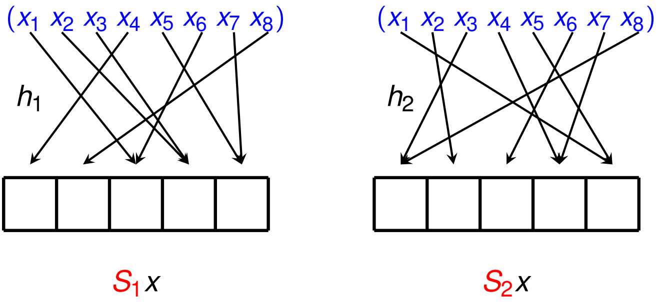
more precisely, output approximation $\widehat{A}$

$$\left| \widehat{A_{ij}} - A_{ij} \right| = \frac{O(1)}{\sqrt{B}} \|A\|_F.$$

$$\begin{tikzcd}
x_1 \in \mathbb{R}^n \arrow[ddr] & & \\
x_2 \in \mathbb{R}^n \arrow[dr] & & \\
x_3 \in \mathbb{R}^n \arrow[r] & \text{Coordinator} \arrow[r] & \widehat{A} \\
\vdots \arrow[ur] & & \\
x_r \in \mathbb{R}^n \arrow[uur] & &
\end{tikzcd}$$

Every party i sends $\textsc{SomeSketch}(x_i)$ into B buckets?

(fast)



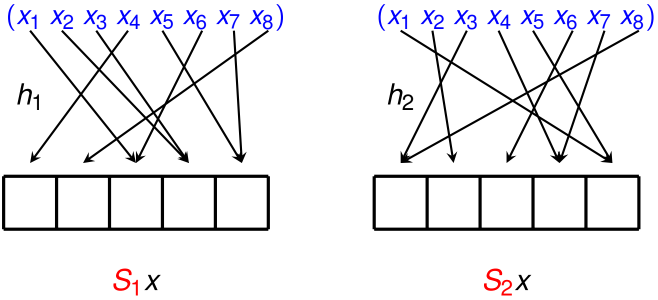
Take two independent instances of CountSketch: hash functions

$$h_1, h_2 : [n] \to [B],$$

random signs

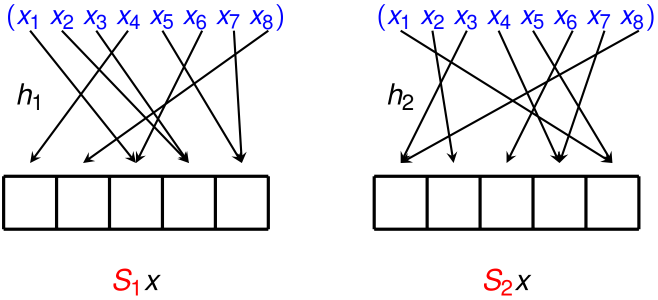
$$s_1, s_2 : [n] \to \{-1, +1\}$$

Tensor CountSketch_1 and CountSketch_2!



Define tensoring of CountSketch_1 and CountSketch_2:

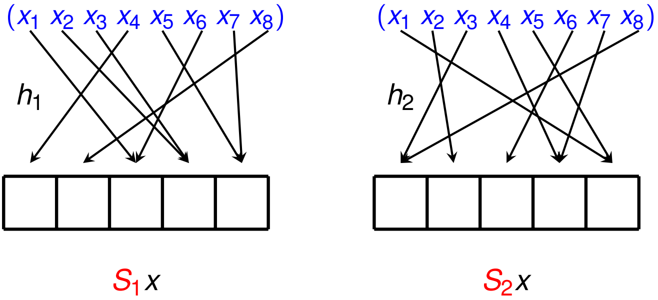
$$h(i,j) = h_1(i) + h_2(j) \pmod{B}.$$



Define tensoring of $\textsc{CountSketch}_1$ and $\textsc{CountSketch}_2$:

$$h(i,j) = h_1(i) + h_2(j) \pmod{B}.$$

and $s(i,j) = s_1(i) \cdot s_2(j)$.



Define tensoring of CountSketch$_1$ and CountSketch$_2$:

$$h(i,j) = h_1(i) + h_2(j) \pmod{B}.$$

and $s(i,j) = s_1(i) \cdot s_2(j)$.

$$\begin{aligned}(Sx)_b &= \sum_{i,j \in [n] : h(i,j) = b} s(i,j) \cdot x_i x_j \\ &= \sum_{i,j \in [n] : h_1(i) + h_2(j) = b} s_1(i) \cdot s_2(j) \cdot x_i x_j\end{aligned}$$

$$(Sx)_b = \sum_{i,j \in [n] : h_1(i) + h_2(j) = b} s_1(i) \cdot s_2(j) \cdot x_i x_j.$$

Can find Sx from $\textsc{CountSketch}_1(x)$ and $\textsc{CountSketch}_2(x)$ fast! (exercise)

Stronger analysis of CountSketch

The bound

$$\|x - \hat{x}\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2$$

is optimal for sketches with $O(k \log n)$ rows, for worst case x

Stronger analysis of CountSketch

The bound

$$\|x - \hat{x}\|_\infty \le \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2$$

is optimal for sketches with $O(k \log n)$ rows, for worst case x

If x is drawn from a distribution (e.g., power law, Zipfian), one can do better by about $\log n$ factor: Minton-Price'14

Stronger analysis of CountSketch

The bound

$$\|x - \hat{x}\|_\infty \leq \frac{1}{\sqrt{k}} \|x_{TAIL}\|_2$$

is optimal for sketches with $O(k \log n)$ rows, for worst case x

If x is drawn from a distribution (e.g., power law, Zipfian), one can do better by about $\log n$ factor: Minton-Price'14

Minton-Price'14 assumes uniformly random hashing. A very recent improvement:

Pseudorandom Hashing for Space-bounded Computation with Applications in Streaming

Praneeth Kacham* Rasmus Pagh† Mikkel Thorup‡ David P. Woodruff§

Abstract

We revisit Nisan's classical pseudorandom generator (PRG) for space-bounded computation (STOC 1990) and its applications in streaming algorithms. We describe a new generator, HashPRG, that can be thought of as a symmetric version of Nisan's generator over larger alphabets.

Non-asymptotic measurement complexity?

Good constants are achieved by ℓ_1-minimization and related (non-sublinear) methods. Get best of both worlds?

Non-asymptotic measurement complexity?

Good constants are achieved by ℓ_1-minimization and related (non-sublinear) methods. Get best of both worlds?

Ex: in LDPC codes, the Tanner graphs needs to be irregular to achieve capacity – similar effects here?

Non-asymptotic measurement complexity?

Good constants are achieved by ℓ_1-minimization and related (non-sublinear) methods. Get best of both worlds?

Ex: in LDPC codes, the Tanner graphs needs to be irregular to achieve capacity – similar effects here?

Information Theoretic Limits of Cardinality Estimation: Fisher Meets Shannon*

Seth Pettie
pettie@umich.edu
University of Michigan
Computer Science and Engineering
Ann Arbor, MI, USA

Dingyu Wang
wangdy@umich.edu
University of Michigan
Computer Science and Engineering
Ann Arbor, MI, USA

ABSTRACT

Estimating the *cardinality* (number of distinct elements) of a large multiset is a classic problem in streaming and sketching, dating back to Flajolet and Martin's classic Probabilistic Counting (PCSA) algorithm from 1983.

In this paper we study the intrinsic tradeoff between the *space complexity* of the sketch and its *estimation error* in the RANDOM ORACLE model. We define a new measure of efficiency for cardinality estimators called the *Fisher-Shannon* (Fish) number $\mathcal{H}/\mathcal{I}$. It captures the tension between the limiting Shannon entropy ($\mathcal{H}$) of the sketch and its normalized Fisher information ($\mathcal{I}$), which characterizes the variance of a statistically efficient, asymptotically unbiased estimator.

KEYWORDS

cardinality estimation, streaming algorithm

ACM Reference Format:
Seth Pettie and Dingyu Wang. 2021. Information Theoretic Limits of Cardinality Estimation: Fisher Meets Shannon. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451032

1 INTRODUCTION

Cardinality Estimation (aka *Distinct Elements* or F_0*-estimation*) is a fundamental problem in streaming/sketching, with widespread industrial deployments in databases, networking, and sensing.

Non-asymptotic measurement complexity?

Good constants are achieved by ℓ_1-minimization and related (non-sublinear) methods. Get best of both worlds?

Ex: in LDPC codes, the Tanner graphs needs to be irregular to achieve capacity – similar effects here?

Information Theoretic Limits of Cardinality Estimation: Fisher Meets Shannon*

Seth Pettie
pettie@umich.edu
University of Michigan
Computer Science and Engineering
Ann Arbor, MI, USA

Dingyu Wang
wangdy@umich.edu
University of Michigan
Computer Science and Engineering
Ann Arbor, MI, USA

ABSTRACT

Estimating the *ca*
multiset is a class
back to Flajolet ar
algorithm from 19
In this paper w
complexity of the
ORACLE model. W
nality estimators
captures the tens
of the sketch and
characterizes the
unbiased estimato

KEYWORDS

Peeling Close to the Orientability Threshold – Spatial Coupling in Hashing-Based Data Structures

Stefan Walzer*

Abstract

1 Introduction

Non-asymptotic measurement complexity?

Good constants are achieved by ℓ_1-minimization and related (non-sublinear) methods. Get best of both worlds?

Ex: in LDPC codes, the Tanner graphs needs to be irregular to achieve capacity – similar effects here?

Information Theoretic Limits of Cardinality Estimation: Fisher Meets Shannon*

Seth Pettie
pettie@umich.edu
University of Michigan
Computer Science and Engineering
Ann Arbor, MI, USA

Dingyu Wang
wangdy@umich.edu
University of Michigan
Computer Science and Engineering
Ann Arbor, MI, USA

ABSTRACT

Estimating the *ca*
multiset is a class
back to Flajolet ar
algorithm from 19
In this paper w
complexity of the
ORACLE model. W
nality estimators
captures the tens
of the sketch and
characterizes the
unbiased estimato

KEYWORDS

Peeling Close to the Orientability Threshold – Spatial Coupling in Hashing-Based Data Structures

Stefan Walzer*

Simple Set Sketching

Abstract

Learning-augmented sketching: learn the hash function h in COUNTSKETCH (and more!) from data

Adversarially robust sketching: what if x is chosen by an adversary with (partial) knowledge of the data structure?

Learning-augmented sketching: learn the hash function h in CountSketch (and more!) from data

Adversarially robust sketching: what if x is chosen by an adversary with (partial) knowledge of the data structure?

On the Robustness of CountSketch to Adaptive Inputs

Edith Cohen* Xin Lyu† Jelani Nelson‡ Tamás Sarlós§ Moshe Shechner¶ Uri Stemmer‖

March 1, 2022

Abstract

`CountSketch` is a popular dimensionality reduction technique that maps vectors to a lower dimension using randomized linear measurements. The sketch supports recovering ℓ_2-heavy hitters of a vector (entries with $v[i]^2 \geq \frac{1}{k}\|\boldsymbol{v}\|_2^2$). We study the robustness of the sketch in *adaptive* settings where input vectors may depend on the output from prior inputs. Adaptive settings arise in processes with feedback or with adversarial attacks. We show that the classic estimator is not robust, and can be attacked with a number of queries of the order of the sketch size. We propose a robust estimator (for a slightly modified sketch) that allows for quadratic number of queries in the sketch size, which is an improvement factor of $\sqrt{k}$ (for k heavy hitters) over prior work.

1 Introduction

Take (randomized) linear measurements of the input

$$S \cdot x = b$$

sketching matrix

space=number of rows

Distribution of the sketching matrix?

Distribution of the sketching matrix?

$$\begin{matrix} +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & +1 & 0 & 0 & -1 \\ 0 & 0 & 0 & +1 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Bernoulli(± 1) or Gaussian linear measurements on random subsets of the universe

The nonzeros are specified by the hash function $h : [n] \to [B]$

Can compute Sx in time $\mathrm{nnz}(x)$!

Distribution of the sketching matrix?

$$
\begin{matrix}
\mathbf{+1} & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{+1} & 0 \\
0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & +1 & 0 & 0 & -1 \\
0 & 0 & 0 & +1 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0
\end{matrix}
$$

Bernoulli(± 1) or Gaussian linear measurements on random subsets of the universe

The nonzeros are specified by the hash function $h : [n] \to [B]$

Can compute Sx in time $\mathrm{nnz}(x)$!

Distribution of the sketching matrix?

$$\begin{array}{cccccccccccc}
+1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\
0 & 0 & 0 & 0 & \mathbf{+1} & 0 & 0 & 0 & 0 & \mathbf{-1} & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & +1 & 0 & 0 & -1 \\
0 & 0 & 0 & +1 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Bernoulli(± 1) or Gaussian linear measurements on random subsets of the universe

The nonzeros are specified by the hash function $h : [n] \to [B]$

Can compute Sx in time $\mathrm{nnz}(x)$!

Distribution of the sketching matrix?

$$\begin{array}{cccccccccccc}
+1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\
0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & \mathbf{+1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & +1 & 0 & 0 & -1 \\
0 & 0 & 0 & +1 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Bernoulli(± 1) or Gaussian linear measurements on random subsets of the universe

The nonzeros are specified by the hash function $h : [n] \to [B]$

Can compute Sx in time $\mathrm{nnz}(x)$!

Distribution of the sketching matrix?

$$\begin{matrix} +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & 0 & 0 & \mathbf{+1} & 0 & 0 & \mathbf{-1} \\ 0 & 0 & 0 & +1 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Bernoulli(± 1) or Gaussian linear measurements on random subsets of the universe

The nonzeros are specified by the hash function $h : [n] \rightarrow [B]$

Can compute Sx in time $\mathrm{nnz}(x)$!

Distribution of the sketching matrix?

$$\begin{matrix}
+1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\
0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & +1 & 0 & 0 & -1 \\
0 & 0 & 0 & \mathbf{+1} & 0 & 0 & \mathbf{+1} & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

Bernoulli(± 1) or Gaussian linear measurements on random subsets of the universe

The nonzeros are specified by the hash function $h : [n] \rightarrow [B]$

Can compute Sx in time $\mathrm{nnz}(x)$!

Random restrictions (hashing)

What can we learn from Sx, where S is just random restrictions?

$$\begin{matrix}
+1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\
0 & +1 & +1 & 0 & +1 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\
+1 & 0 & 0 & +1 & +1 & +1 & 0 & +1 & 0 & 0 & 0 & 0 \\
+1 & +1 & 0 & 0 & +1 & +1 & 0 & 0 & +1 & 0 & 0 & +1 \\
0 & +1 & 0 & +1 & +1 & +1 & +1 & 0 & +1 & 0 & +1 & 0
\end{matrix}$$

Can learn $\|x\|_0$, i.e. number of nonzeros in x

Johnson-Lindenstrauss transform

Sketching matrix S = a row of i.i.d. Gaussians of unit variance

Johnson-Lindenstrauss transform

Sketching matrix S = a row of i.i.d. Gaussians of unit variance

Measures ℓ_2^2 norm of x in expectation:

$$\mathbb{E}\left[\|Sx\|_2^2\right] = \|x\|_2^2$$

Johnson-Lindenstrauss transform

Sketching matrix S = m rows of i.i.d. Gaussians of unit variance

$1/m$

Johnson-Lindenstrauss transform

Sketching matrix S = m rows of i.i.d. Gaussians of unit variance

$1/m$

Measures ℓ_2^2 norm of x with high probability:

$$\mathbb{P}\left[\|Sx\|_2^2 \not\approx \|x\|_2^2\right] = 1 - \exp(-\Omega(\varepsilon^2 m)$$

Johnson-Lindenstrauss transform

Sketching matrix S = m rows of i.i.d. Gaussians of unit variance $1/m$

Measures ℓ_2^2 norm of x with high probability:

$$\mathbb{P}\left[\|Sx\|_2^2 \not\approx \|x\|_2^2\right] = 1 - \exp(-\Omega(\varepsilon^2 m))$$

Downside: Sx takes $m \cdot n$ time to compute

(Faster) Johnson-Lindenstrauss transform

Subsampled randomized Hadamard transform

(Faster) Johnson-Lindenstrauss transform

Subsampled randomized Hadamard transform

Sketching matrix $S = P \cdot H \cdot D$

(Faster) Johnson-Lindenstrauss transform

Subsampled randomized Hadamard transform

Sketching matrix $S = P \cdot H \cdot D$

D=diagonal random sign matrix, H=Hadamard transform,
P=random sampling matrix

(Faster) Johnson-Lindenstrauss transform

Subsampled randomized Hadamard transform

Sketching matrix $S = P \cdot H \cdot D$

D=diagonal random sign matrix, H=Hadamard transform,
P=random sampling matrix

Sx can be computed in $O(m + n \log n)$ time

Frequency moments

The p-th frequency moment

$$F_p = \sum_{i \in [n]} f_i^p$$

Frequency moments

The p-th frequency moment

$$F_p = \sum_{i \in [n]} f_i^p$$

Theorem

Can approximate F_p for all $p \in [0,2]$ in polylogarithmic space, but need $\Omega(n^{1-2/p})$ space for all $p > 2$

Frequency moments

The p-th frequency moment

$$F_p = \sum_{i \in [n]} f_i^p$$

Theorem

Can approximate F_p for all $p \in [0,2]$ in polylogarithmic space, but need $\Omega(n^{1-2/p})$ space for all $p > 2$

Bar-Yossef et al. Information complexity approach to data stream lower bounds