Basic algorithmic techniques for data streams

Mariano Zelke

Goethe-Universität Frankfurt am Main
Institut für Informatik
Arbeitsgruppe Theorie Komplexer Systeme

DEIS’10, Dagstuhl, Nov. 2010
Contents

Basic Definitions

Sampling
   General Idea
   Reservoir Sampling
   Sampling Applications

Advanced Sampling
   AMS Sampling
   Sliding Window Sampling

Count-Min Sketch
Data Stream Model

A data stream model stream is a sequence of elements \( m \) from a universe of size \( n \):

\[ a_1, a_2, a_3, \ldots, a_m \]

where each element \( a_i \in \{1, \ldots, n\} \).

The goal is to gain information about the stream, such as statistical information (median, frequency moments, ...), and the longest increasing subsequence, ...

However, algorithms are restricted to sequential access to items in the stream and limited memory, sublinear in \( m \) and \( n \).
Data Stream Model

► **Stream**: $m$ elements of some universe of size $n$
Data Stream Model

> **Stream**: $m$ elements of some universe of size $n$

$$a_1, a_2, a_3, \ldots, a_m \quad a_i \in \{1, \ldots, n\}$$
Data Stream Model

- **Stream**: $m$ elements of some universe of size $n$
  
  $a_1, a_2, a_3, \ldots, a_m$ \quad \quad a_i \in \{1, \ldots, n\}$

- **Goal**: gain information about stream
Stream: $m$ elements of some universe of size $n$

$a_1, a_2, a_3, \ldots, a_m \quad a_i \in \{1, \ldots, n\}$

Goal: gain information about stream

- statistical information (median, frequency moments, ...),
- longest increasing subsequence, ...
Data Stream Model

- **Stream**: \( m \) elements of some universe of size \( n \)
  
  \[
  a_1, a_2, a_3, \ldots, a_m \quad a_i \in \{1, \ldots, n\}
  \]

- **Goal**: gain information about stream
  
  statistical information (median, frequency moments, ...), longest increasing subsequence, ...

- **But**: algorithms are restricted to
Data Stream Model

- **Stream**: $m$ elements of some universe of size $n$
  
  $a_1, a_2, a_3, \ldots, a_m \quad a_i \in \{1, \ldots, n\}$

- **Goal**: gain information about stream
  
  statistical information (median, frequency moments, ...),
  longest increasing subsequence,...

- **But**: algorithms are restricted to
  
  - sequential access to items in stream
Data Stream Model

- **Stream**: $m$ elements of some universe of size $n$
  
  $a_1, a_2, a_3, \ldots, a_m \quad a_i \in \{1, \ldots, n\}$

- **Goal**: gain information about stream
  
  statistical information (median, frequency moments, ...),
  longest increasing subsequence,...

- **But**: algorithms are restricted to
  
  - sequential access to items in stream
  - limited memory, sublinear in $m$ and $n$
Sampling

General idea:

▶ sample (= select) items from stream according to some rule
▶ rule is randomized or deterministic
▶ use sampled items to get information about whole stream
⇒ good for streaming
Sampling

General idea:

- sample (= select) items from stream according to some rule
Sampling

General idea:

- sample (= select) items from stream according to some rule
- rule is randomized or deterministic
Sampling

General idea:

- sample (select) items from stream according to some rule
- rule is randomized or deterministic
Sampling

General idea:

- sample (= select) items from stream according to some rule
- rule is randomized or deterministic
- use sampled items to get information about whole stream
Sampling

General idea:
- sample (= select) items from stream according to some rule
- rule is randomized or deterministic
- use sampled items to get information about whole stream
- only need to store sampled items
  ⇒ good for streaming
Sample out a single item uniformly at random from the stream
Sample out a single item uniformly at random from the stream

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m$
Easy Starter

Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
Easy Starter

Sample out a single item uniformly at random from the stream

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m$

- Easy if $m$ is known in advance:
- Pick a random number $r \in \{1, 2, \ldots, m\}$
Easy Starter

Sample out a single item uniformly at random from the stream $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m$

- Easy if $m$ is known in advance:
  - Pick a random number $r \in \{1, 2, \ldots, m\}$
  - Go over the stream and snatch out sampled item
Easy Starter

Sample out a single item uniformly at random from the stream

\[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m\]

▶ Easy if \(m\) is known in advance:
▶ Pick a random number \(r \in \{1, 2, \ldots, m\}\)
▶ Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Easy Starter

Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
- Pick a random number \( r \in \{1, 2, \ldots, m\} \)
- Go over the stream and snatch out sampled item
Easy Starter

Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m\]

- Easy if \(m\) is known in advance:
  - Pick a random number \(r \in \{1, 2, \ldots, m\}\)
  - Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
- Pick a random number \( r \in \{1, 2, \ldots, m\} \)
- Go over the stream and snatch out sampled item
Easy Starter

Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Easy Starter

Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

Easy if \( m \) is known in advance:

- Pick a random number \( r \in \{1, 2, \ldots, m\} \)
- Go over the stream and snatch out sampled item
Easy Starter

Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

▶ Easy if \( m \) is known in advance:
▶ Pick a random number \( r \in \{1, 2, \ldots, m\} \)
▶ Go over the stream and snatch out sampled item
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

▶ Easy if \( m \) is known in advance:
▶ Pick a random number \( r \in \{1, 2, \ldots, m\} \)
▶ Go over the stream and snatch out sampled item \( a_r \)
Sample out a single item uniformly at random from the stream

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_r, \ldots, a_m \]

- Easy if \( m \) is known in advance:
  - Pick a random number \( r \in \{1, 2, \ldots, m\} \)
  - Go over the stream and snatch out sampled item \( a_r \)

But what if \( m \) is not known in advance?
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m\]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

current sample:
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

down
\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

take as sample

current sample: \[ a_1 \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[
a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m
\]

current sample: \(a_r\)

replace with probability 1/2
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ \downarrow \]

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

replace with probability 1/3

current sample: \( a_r \)
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

\[ \text{current sample: } a_r \]

replace with probability 1/4
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

replace with probability \(1/i\)

current sample: \(a_r\)
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

Current sample: \( a_r \)

Replace with probability \( 1/m \)
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

final sample: \[ a_r \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

current sample:

\[ Pr[\text{final sample } a_i] = \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length.

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

current sample: \( a_i \)

\[ Pr[\text{final sample } a_i] = \frac{1}{i} \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

current sample: \( a_i \)

\[ Pr[\text{final sample } a_i] = \frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

current sample: \( a_i \)

Pr[final sample \( a_i \)] = \( \frac{1}{i} \times (1 - \frac{1}{i+1}) \times (1 - \frac{1}{i+2}) \)
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

current sample: \( a_i \)

\[ Pr[\text{final sample } a_i] = \frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \left(1 - \frac{1}{i+2}\right) \times \ldots \times \left(1 - \frac{1}{m}\right) \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

final sample: \( a_i \)

\[ Pr[\text{final sample } a_i] = \frac{1}{i} \times \left( 1 - \frac{1}{i+1} \right) \times \left( 1 - \frac{1}{i+2} \right) \times \ldots \times \left( 1 - \frac{1}{m} \right) \]

\[ = \frac{1}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times \ldots \times \frac{m-1}{m} \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length.

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

final sample: \( a_i \)

\[ Pr[\text{final sample } a_i] = \frac{1}{i} \times (1 - \frac{1}{i+1}) \times (1 - \frac{1}{i+2}) \times \ldots \times (1 - \frac{1}{m}) \]

\[ = \frac{1}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times \ldots \times \frac{m-1}{m} \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

final sample: \[ a_i \]

\[ Pr[\text{final sample } a_i] = \frac{1}{i} \times (1 - \frac{1}{i+1}) \times (1 - \frac{1}{i+2}) \times \ldots \times (1 - \frac{1}{m}) \]
\[ = \frac{1}{m} \]
Reservoir Sampling

Sample out a single item uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_m \]

final sample: \( a_i \)

\[
Pr[\text{final sample } a_i] = \frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \left(1 - \frac{1}{i+2}\right) \times \ldots \times \left(1 - \frac{1}{m}\right) \\
= \frac{1}{m}
\]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m$

take into sample

current samples

Mariano Zelke Basic algorithmic techniques for data streams 6/12
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

\[ \text{sample with probability } 1/5 \]

current samples

\[ \begin{array}{cccc}
   a_1 & a_2 & a_3 & a_4 \\
\end{array} \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

sample with probability \( \frac{1}{5} \)

replace a sampled element uniformly at random

current samples

\[ a_1, a_2, a_3, a_4 \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

sample with probability 1/5

replace a sampled element uniformly at random

current samples

\[ \begin{array}{c}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
\end{array} \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ \downarrow \]

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current samples

\[ \begin{array}{cccc}
  a_1 & a_5 & a_3 & a_4 \\
\end{array} \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[
a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m
\]

current samples

\[
\begin{array}{cccc}
  a_1 & a_5 & a_3 & a_4 \\
\end{array}
\]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

sample with probability 1/6

current samples

- \( a_1 \)
- \( a_5 \)
- \( a_3 \)
- \( a_4 \)
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current samples

\[ \begin{array}{c|c|c|c}
    a_1 & a_5 & a_3 & a_4 \\
\end{array} \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]
Reservoir Sampling

Sample out **several items** uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

sample with probability \(1/i\)

\[ a_i \]

current samples \(a_{r_1}, a_{r_2}, a_{r_3}, a_{r_4}\)
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_{i}, \ldots, a_m \]

sample with probability \( \frac{1}{i} \)

replace a sampled element uniformly at random

current samples

\[ a_r_1, a_r_2, a_r_3, a_r_4 \]
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

sample with probability \( \frac{1}{i} \)

replace a sampled element uniformly at random

current samples

Memory usage for sampling \( k \) items:
Reservoir Sampling

Sample out several items uniformly at random from the stream without knowing its length

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

sample with probability \( \frac{1}{i} \)

replace a sampled element uniformly at random

current samples

Memory usage for sampling \( k \) items: \( k \cdot \log n \)
Sampling Applications

stream:
Sampling Applications

stream: ..............................................................
sampling

samples: ..............................................
Sampling Applications

stream: .................................................................
sampling
samples: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Goal: determine query selectivity on items of stream
(assume query to be invariant of stream order)
Sampling Applications

stream: .................................................. sampling
samples: ..............................................
determine selectivity of query on samples
Sampling Applications

stream: ................................................................................................................................

sampling

determine selectivity of query on samples

samples: .................................................................

To get $(1 \pm \varepsilon)$-estimate with probability $1 - \delta$: What sample size $s$?
Sampling Applications

To get $(1 \pm \varepsilon)$-estimate with probability $1 - \delta$: What sample size $s$?

Query selects $\frac{m}{c}$ stream items $\Rightarrow \text{Exp}[s^+] = \frac{s}{c}$
Sampling Applications

stream: .................................................................

sampling

samples: .................................................................

do not hallucinate.

determine selectivity of query on samples

To get $(1 \pm \varepsilon)$-estimate with probability $1 - \delta$: What sample size $s$?

Query selects $\frac{m}{c}$ stream items $\Rightarrow \text{Exp}[s^+] = \frac{s}{c}$

Chernoff-Hoeffding-ineq.: $\Pr[|s^+ - \text{Exp}[s^+]| > \varepsilon \cdot s^+] \leq e^{-\Theta(\varepsilon^2 s)}$
Sampling Applications

stream: .........................................................................................................................

sampling

determine selectivity of query on samples

to get \((1 \pm \varepsilon)\)-estimate with probability \(1 - \delta\): What sample size \(s\)?

Query selects \(\frac{m}{c}\) stream items \(\Rightarrow\) \(\text{Exp}[s^+] = \frac{s}{c}\)

Chernoff-Hoeffding-Ineq.: \(\Pr[|s^+ - \text{Exp}[s^+]| > \varepsilon \cdot s^+] \leq e^{-\Theta(\varepsilon^2 s)}\)

Sample \(O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta}\right)\) items
Sampling Applications

stream: .................................................................

sampling

samples: .................................................................
determine selectivity of query on samples

To get \((1 \pm \varepsilon)\)-estimate with probability \(1 - \delta\): What sample size \(s\)?

Query selects \(\frac{m}{c}\) stream items \(\Rightarrow \quad \text{Exp}[s^+] = \frac{s}{c}\)

Chernoff-Hoeffding-Ineq.: \(P_r[|s^+ - \text{Exp}[s^+]| > \varepsilon \cdot s^+] \leq e^{-\Theta(\varepsilon^2 s)}\)

Memory usage: \(O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta}\right)\) items \(\times \log n\) bits
Sampling Applications

stream: .................................................................

sampling

determine selectivity of query on samples

samples: .................................................................

To get \((1 \pm \varepsilon)\)-estimate with probability \(1 - \delta\): What sample size \(s\) ?

Query selects \(\frac{m}{c}\) stream items \(\Rightarrow\) \(\text{Exp}[s^+] = \frac{s}{c}\)

Chernoff-Hoeffding-Ineq.: \(\Pr[|s^+ - \text{Exp}[s^+]| > \varepsilon \cdot s^+] \leq e^{-\Theta(\varepsilon^2 s)}\)

Memory usage: \(\mathcal{O}\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta} \cdot \log n\right)\) bits
Sampling Applications

stream: .................................................................

sampling

tsamples:  ⋅ ⋅ ⋅ ⋅ ⋅ ⋅ ⋅ ⋅ ⋅ ⋅ ⋅ ⋅ ⋅
Sampling Applications

stream: .................................................................

samples: ............................................................

Goal: find the median of the stream
Sampling Applications

stream: ................................................................. sampling

samples: ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●●
sort samples

sorted samples: ● ≤ ● ≤ ● ≤ ● ≤ ● ≤ ● ≤ ● ≤ ● ≤ ● ≤ ● ≤
Sampling Applications

Pick median of samples as an estimate of stream’s median
Sampling Applications

stream: .................................................................

sampling

circular arrow

tsampling

samples:  

sort samples

circular arrow

tsampling

sorted

circular arrow

tsampling

samples:  

median of samples

Pick median of samples as an estimate of stream’s median

To get \((1 \pm \varepsilon)\)-estimate with probability \(1 - \delta\):

\[
sample \mathcal{O}\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta}\right) \text{ items}
\]
AMS Sampling

Stream $a_1, a_2, a_3, \ldots, a_m$

Frequency of an item: $f_i = \mid \{ j : a_j = i \} \mid$

$k$th frequency moment $F_k = \sum_{i=1}^{n} f_k^i$

Frequency moments provide useful statistics:

- $F_0$: number of distinct elements in stream
- $F_1$: length of stream, $m$
- $F_2$: size of self join
- $F_k$, $k \geq 2$: skew of distribution

Trivial determination of $F_k$: maintain counters for each $f_i$ $\Rightarrow \Omega(n)$ bits needed
AMS Sampling

Stream \ a_1, a_2, a_3, \ldots, a_m

Frequency of an item: \ f_i = \mid \{ j : a_j = i \} \mid

\text{kth frequency moment } F_k = \sum_{i=1}^{n} f_k i

Frequency moments provide useful statistics:

- \( F_0 \): number of distinct elements in stream
- \( F_1 \): length of stream, \( m \)
- \( F_2 \): size of self join
- For \( k \geq 2 \), \( F_k \): skew of distribution

Trivial determination of \( F_k \): maintain counters for each \( f_i \) \implies \Omega(n) \text{ bits needed}
AMS Sampling

[Alon, Matias, Szegedy '96]

Stream \( a_1, a_2, a_3, \ldots, a_m \)

Frequency of an item: \( f_i = |\{ j : a_j = i \}| \)
AMS Sampling

[Alon, Matias, Szegedy '96]

Stream \( a_1, a_2, a_3, \ldots, a_m \)

Frequency of an item: \( f_i = |\{j : a_j = i\}| \)

Example: stream \( 2, 3, 3, 2, 3, 1, 2, 3 \)

\( f_1 = 1, \quad f_2 = 3, \quad f_3 = 4 \)
AMS Sampling

Stream \( a_1, a_2, a_3, \ldots, a_m \)

Frequency of an item: \( f_i = |\{j : a_j = i\}| \)

\( k \)th frequency moment \( F_k = \sum_{i=1}^{n} f_i^k \)
AMS Sampling

Stream \( a_1, a_2, a_3, \ldots, a_m \)

Frequency of an item: \( f_i = |\{j : a_j = i\}| \)

\( k \)th frequency moment \( F_k = \sum_{i=1}^{n} f_i^k \)

Frequency moments provide useful statistics:

- \( F_0 \): number of distinct elements in stream
- \( F_1 \): length of stream, \( m \)
- \( F_2 \): size of self join
- \( F_k, k \geq 2 \): skew of distribution
AMS Sampling

[Alon, Matias, Szegedy '96]

Stream \( a_1, a_2, a_3, \ldots, a_m \)

Frequency of an item: \( f_i = |\{j : a_j = i\}| \)

\( k \)th frequency moment \( F_k = \sum_{i=1}^{n} f_i^k \)

Frequency moments provide useful statistics:

- \( F_0 \): number of distinct elements in stream
- \( F_1 \): length of stream, \( m \)
- \( F_2 \): size of self join
- \( F_k, k \geq 2 \): skew of distribution

Trivial determination of \( F_k \): maintain counters for each \( f_i \)
AMS Sampling

Stream \( a_1, a_2, a_3, \ldots, a_m \)

Frequency of an item: \( f_i = |\{j : a_j = i\}| \)

\[ F_k = \sum_{i=1}^{n} f_i^k \]

Frequency moments provide useful statistics:
- \( F_0 \): number of distinct elements in stream
- \( F_1 \): length of stream, \( m \)
- \( F_2 \): size of self join
- \( F_k, k \geq 2 \): skew of distribution

Trivial determination of \( F_k \): maintain counters for each \( f_i \)
\[ \Rightarrow \Omega(n) \text{ bits needed} \]
AMS Sampling (for estimating $F_k$)
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
AMS Sampling (for estimating $F_k$)
1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$

Example: stream $2, 3, 3, 2, 3, 1, 2, 3$

$a_j = 3$, $r = 3$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[m(r^k - (r - 1)^k)]$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[m(r^k - (r - 1)^k)]$$

$$= \left[ m(f_1^k - (f_1 - 1)^k) \right]$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate
   $$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[ m(r^k - (r - 1)^k) ]$$

$$= \left[ m(f_1^k - (f_1 - 1)^k) + m((f_1 - 1)^k - (f_1 - 2)^k) \right]$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[m(r^k - (r - 1)^k)] = \left[m(f_1^k - (f_1 - 1)^k) + m((f_1 - 1)^k - (f_1 - 2)^k) + \ldots + m(2^k - 1^k)\right]$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[m(r^k - (r - 1)^k)] = m(f_1^k - (f_1 - 1)^k) + m((f_1 - 1)^k - (f_1 - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k$$
AMS Sampling

AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[m(r^k - (r - 1)^k)] = \left[ m(f_1^k - (f_1 - 1)^k) + m((f_1 - 1)^k - (f_1 - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \\
m(f_2^k - (f_2 - 1)^k) + m((f_2 - 1)^k - (f_2 - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \right]$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[m(r^k - (r - 1)^k)] = \left[ m(f_1^k - (f_1 - 1)^k) + m((f_1 - 1)^k - (f_1 - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \\
+ m(f_2^k - (f_2 - 1)^k) + m((f_2 - 1)^k - (f_2 - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \\
\ldots \\
+ m(f_n^k - (f_n - 1)^k) + m((f_n - 1)^k - (f_n - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \right]$$
AMS Sampling

AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate
   \[ m(r^k - (r - 1)^k) \]

for $k \geq 1$:
\[
\text{Exp}[m(r^k - (r - 1)^k)] = \left[ m(f_1^k - (f_1 - 1)^k) + m((f_1 - 1)^k - (f_1 - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \\
+ m(f_2^k - (f_2 - 1)^k) + m((f_2 - 1)^k - (f_2 - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \\
\ldots \\
+ m(f_n^k - (f_n - 1)^k) + m((f_n - 1)^k - (f_n - 2)^k) + \ldots + m(2^k - 1^k) + m \cdot 1^k \right] \frac{1}{m}
\]
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[ m(r^k - (r - 1)^k) ] = f_1^k + f_2^k + f_3^k + \ldots + f_n^k$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j': j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

for $k \geq 1$:

$$\text{Exp}[m(r^k - (r - 1)^k)] = f_1^k + f_2^k + f_3^k + \ldots + f_n^k = F_k$$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{ j' : j' \geq j, a_{j'} = a_j \}|$
3. At the end of the stream calculate

$$m(r^k - (r-1)^k)$$

To get $(1 \pm \varepsilon)$-estimate of $F_k$ with probability $1 - \delta$: 
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = \left| \{ j' : j' \geq j, a_{j'} = a_j \} \right|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

To get $(1 \pm \varepsilon)$-estimate of $F_k$ with probability $1 - \delta$:

Run $O\left(\frac{n^{1-1/k}}{\varepsilon^2}\right)$ parallel instances, take average $A$.
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate

$$m(r^k - (r - 1)^k)$$

To get $(1 \pm \varepsilon)$-estimate of $F_k$ with probability $1 - \delta$:

Run $O\left(\frac{n^{1-1/k}}{\varepsilon^2}\right)$ parallel instances, take average $A$

$\Rightarrow$ Chebyshev-Ineq.: $Pr[|A - F_k| > \varepsilon \cdot F_k] \leq p < \frac{1}{2}$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate $m(r^k - (r - 1)^k)$

To get $(1 \pm \varepsilon)$-estimate of $F_k$ with probability $1 - \delta$:

Run $O\left(\frac{n^{1-1/k}}{\varepsilon^2}\right)$ parallel instances, take average $A$

$\Rightarrow$ Chebyshev-Ineq.: $Pr[|A - F_k| > \varepsilon \cdot F_k] \leq p < \frac{1}{2}$

Take median $M$ over $O\left(\log \frac{1}{\delta}\right)$ such averages
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j' : j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate
   \[ m(r^k - (r - 1)^k) \]

To get $(1 \pm \varepsilon)$-estimate of $F_k$ with probability $1 - \delta$:

Run $O \left( \frac{n^{1-1/k}}{\varepsilon^2} \right)$ parallel instances, take average $A$

$\Rightarrow$ Chebyshev-Ineq.: $Pr[|A - F_k| > \varepsilon \cdot F_k] \leq p < \frac{1}{2}$

Take median $M$ over $O \left( \log \frac{1}{\delta} \right)$ such averages

$\Rightarrow$ Chernoff-Ineq.: $Pr[|M - F_k| > \varepsilon \cdot F_k] \leq \delta$
AMS Sampling (for estimating $F_k$)

1. Pick random item $a_j$ from stream
2. Compute $r = |\{j': j' \geq j, a_{j'} = a_j\}|$
3. At the end of the stream calculate
   \[ m(r^k - (r - 1)^k) \]

To get $(1 \pm \varepsilon)$-estimate of $F_k$ with probability $1 - \delta$:

Run $O\left(\frac{n^{1-1/k}}{\varepsilon^2}\right)$ parallel instances, take average $A$
\[ \Rightarrow \text{Chebyshev-Ineq.: } Pr[|A - F_k| > \varepsilon \cdot F_k] \leq p < \frac{1}{2} \]

Take median $M$ over $O\left(\log \frac{1}{\delta}\right)$ such averages
\[ \Rightarrow \text{Chernoff-Ineq.: } Pr[|M - F_k| > \varepsilon \cdot F_k] \leq \delta \]

Memory consumption: $O\left(\frac{n^{1-1/k}}{\varepsilon^2} \cdot \log \frac{1}{\delta} \cdot (\log n + \log m)\right)$ bits
Sliding Window Sampling
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample: \(a_2\)
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample: \( a_2 \)

Current sample might be "far away" from actual item
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample: \( a_2 \)

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample: \( a_2 \)

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
  \[ \Rightarrow \] only consider items in a sliding window of size \( w \)
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
\[ \Rightarrow \text{only consider items in a sliding window of size } w \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Current sample might be "far away" from actual item
  - fine for some applications
  - for others only \( w \) recent items matter
    \[ \Rightarrow \text{only consider items in a sliding window of size } w \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
  \[ \Rightarrow \text{only consider items in a sliding window of size } w \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
  \[ \Rightarrow \] only consider items in a sliding window of size \( w \)
Sliding Window Sampling

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m$

Current sample might be "far away" from actual item
- fine for some applications
- for others only $w$ recent items matter
  $\Rightarrow$ only consider items in a sliding window of size $w$
Sliding Window Sampling

\[ a_1, \overbrace{a_2, a_3, a_4, a_5, a_6}^{a_1}, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
  \[ \Rightarrow \text{only consider items in a sliding window of size } w \]
Sliding Window Sampling

\[
\downarrow
\]

\[a_1, a_2, (a_3, a_4, a_5, a_6, a_7), a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m\]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \(w\) recent items matter
  \(\Rightarrow\) only consider items in a sliding window of size \(w\)
Sliding Window Sampling

\[
\begin{align*}
    a_1, a_2, a_3, \overbrace{a_4, a_5, a_6, a_7, a_8}^w, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m
\end{align*}
\]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
  \( \Rightarrow \) only consider items in a sliding window of size \( w \)
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, (a_5, a_6, a_7, a_8, a_9), a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
  \[ \Rightarrow \text{only consider items in a sliding window of size } w \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, \underbrace{a_6, a_7, a_8, a_9, a_{10}}_{\text{current sample}}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Current sample might be "far away" from actual item
- fine for some applications
- for others only \( w \) recent items matter
  \[ \Rightarrow \text{only consider items in a sliding window of size } w \]
Sliding Window Sampling

\[
a_1, a_2, a_3, a_4, a_5, (a_6, a_7, a_8, a_9, a_{10}), a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m
\]

Goal: sample an item uniformly from sliding window of size \(w\)
Sliding Window Sampling

Goal: sample an item uniformly from sliding window of size $w$

Trivial: - memorize whole window content $\Rightarrow w \cdot \log n$ bits
- impractical if $w$ is large
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, (a_6, a_7, a_8, a_9, a_{10}), a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Goal: sample an item uniformly from sliding window of size \( w \)

Trivial: - memorize whole window content \( \Rightarrow w \cdot \log n \) bits
  - impractical if \( w \) is large

Better idea:

Algorithm:
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Goal: sample an item uniformly from sliding window of size \( w \)

Trivial: - memorize whole window content \( \Rightarrow w \cdot \log n \) bits
- impractical if \( w \) is large

Better idea:
Algorithm: 1: For each \( a_i \) pick random value \( r_i \in (0, 1) \)
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, (a_6, a_7, a_8, a_9, a_{10}), a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Goal: sample an item uniformly from sliding window of size \( w \)

Trivial: - memorize whole window content \( \Rightarrow w \cdot \log n \) bits
- impractical if \( w \) is large

Better idea:

**Algorithm:** 1: For each \( a_i \) pick random value \( r_i \in (0, 1) \)
2: In window \( (a_{i-w+1}, \ldots, a_i) \) choose \( a_j \) with smallest \( r_j \)
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, \overline{a_6, a_7, a_8, a_9, a_{10}}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Goal: sample an item uniformly from sliding window of size \(w\)

Trivial: - memorize whole window content \(\Rightarrow w \cdot \log n\) bits
- impractical if \(w\) is large

Better idea:

**Algorithm:** 1: For each \(a_i\) pick random value \(r_i \in (0, 1)\)

2: In window \((a_{i-w+1}, \ldots, a_i)\) choose \(a_j\) with smallest \(r_j\)

3: Only maintain items in window whose \(r\)-value is minimal among subsequent \(r\)-values
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

**Algorithm:**

1. For each \( a_i \) pick random value \( r_i \in (0, 1) \)
2. In window \((a_{i-w+1}, \ldots, a_i)\) choose \( a_j \) with smallest \( r_j \)
3. Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
Sliding Window Sampling

\[ \underline{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m} \]

**Algorithm:**

1. For each \( a_i \) pick random value \( r_i \in (0, 1) \)
2. In window \( (a_{i-w+1}, \ldots, a_i) \) choose \( a_j \) with smallest \( r_j \)
3. Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

random value \( r_1 = 0.2 \)

**Algorithm:**

1: For each \( a_i \) pick random value \( r_i \in (0, 1) \)

2: In window \( (a_{i-w+1}, \ldots, a_i) \) choose \( a_j \) with smallest \( r_j \)

3: Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
Sliding Window Sampling

\[
\downarrow
\]
\[
(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m)
\]

random value \( r_1 = 0.2 \)

Algorithm: 1: For each \( a_i \) pick random value \( r_i \in (0, 1) \)
2: In window \((a_{i-w+1}, \ldots, a_i)\) choose \( a_j \) with smallest \( r_j \)
3: Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
**Sliding Window Sampling**

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

random value \( r_1 = 0.2 \)

**Algorithm:**

1: For each \( a_i \) pick random value \( r_i \in (0, 1) \)

2: In window \( (a_{i-w+1}, \ldots, a_i) \) choose \( a_j \) with smallest \( r_j \)

3: Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$
2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$
3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$
2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$
3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Sliding Window Sampling

Algorithm:
1. For each \(a_i\) pick random value \(r_i \in (0, 1)\)
2. In window \((a_{i-w+1}, \ldots, a_i)\) choose \(a_j\) with smallest \(r_j\)
3. Only maintain items in window whose \(r\)-value is minimal among subsequent \(r\)-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$
2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$
3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$

2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$

3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$
2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$
3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$
2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$
3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Algorithm: 1: For each \( a_i \) pick random value \( r_i \in (0, 1) \)

2: In window \( (a_{i-w+1}, \ldots, a_i) \) choose \( a_j \) with smallest \( r_j \)

3: Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$
   2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$
   3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$

2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$

3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
Sliding Window Sampling

Algorithm: 1: For each \( a_i \) pick random value \( r_i \in (0, 1) \)

2: In window \( (a_{i-w+1}, \ldots, a_i) \) choose \( a_j \) with smallest \( r_j \)

3: Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
**Sliding Window Sampling**

\[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m\]

Random value \(r_6 = 0.5\)

**Algorithm:**

1. For each \(a_i\) pick random value \(r_i \in (0, 1)\)
2. In window \((a_{i-w+1}, \ldots, a_i)\) choose \(a_j\) with smallest \(r_j\)
3. Only maintain items in window whose \(r\)-value is minimal among subsequent \(r\)-values
Sliding Window Sampling

Algorithm: 1: For each $a_i$ pick random value $r_i \in (0, 1)$

2: In window $(a_{i-w+1}, \ldots, a_i)$ choose $a_j$ with smallest $r_j$

3: Only maintain items in window whose $r$-value is minimal among subsequent $r$-values
**Sliding Window Sampling**

\[ a_1, (a_2, a_3, a_4, a_5, a_6), a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

random value \( r_6 = 0.5 \)

current sample

\[ a_2 \rightarrow a_6, 0.4 \rightarrow 0.5 \]

**Algorithm:**

1. For each \( a_i \) pick random value \( r_i \in (0, 1) \)
2. In window \((a_{i-w+1}, \ldots, a_i)\) choose \( a_j \) with smallest \( r_j \)
3. Only maintain items in window whose \( r \)-value is minimal among subsequent \( r \)-values
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[
\begin{array}{c}
\frac{a_x}{r_x} \\
\frac{a_y}{r_y}
\end{array}
\]

\[ \ldots \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[ a_x \]

\[ r_x \]

\[ a_y \]

\[ r_y \]

linked list \( \ell \)

Analysis: What is the length of \( \ell \)?
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Analysis: What is the length of \( \ell \) ?

Worst case: \( |\ell| = w \) \( \) But that is very unlikely.
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[ \begin{array}{c}
  a_x \\
r_x
\end{array} \quad \rightarrow \quad \begin{array}{c}
  a_y \\
r_y
\end{array} \quad \rightarrow \quad \ldots \]

linked list \( \ell \)

Analysis: What is the length of \( \ell \)?

\[ \text{Exp}[|\ell|] = \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[ \begin{array}{c}
    a_x \\
    r_x \\
\end{array} \quad \begin{array}{c}
    a_y \\
    r_y \\
\end{array} \quad \cdots \\
\]

linked list \( \ell \)

Analysis: What is the length of \( \ell \)?

\[ \text{Exp}[|\ell|] = \text{Pr}[a_i \in \ell] + \]

Mariano Zelke

Basic algorithmic techniques for data streams
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[ \overbrace{a_x \atop r_x} \quad \overbrace{a_y \atop r_y} \quad \ldots \]

linked list \( \ell \)

Analysis: What is the length of \( \ell \)?

\[ \text{Exp}[|\ell|] = \text{Pr}[a_i \in \ell] + \text{Pr}[a_{i-1} \in \ell] + \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[
\begin{array}{c}
\text{linked list } \ell \\
\end{array}
\]

Analysis: What is the length of \( \ell \)?

\[
\text{Exp}[|\ell|] = Pr[a_i \in \ell] + Pr[a_{i-1} \in \ell] + \ldots + Pr[a_{i-w+1} \in \ell]
\]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

Analysis: What is the length of \( \ell \)?

\[
\text{Exp}[|\ell|] = Pr[a_i \in \ell] + Pr[a_{i-1} \in \ell] + \ldots + Pr[a_{i-w+1} \in \ell]
\]

\[= 1 + \]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

### Analysis: What is the length of $\ell$?

\[
\text{Exp}[|\ell|] = \text{Pr}[a_i \in \ell] + \text{Pr}[a_{i-1} \in \ell] + \ldots + \text{Pr}[a_{i-w+1} \in \ell] \\
= 1 + \frac{1}{2} +
\]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[ \begin{array}{c}
a_x \\
r_x
\end{array} \quad \rightarrow \quad \begin{array}{c}
a_y \\
r_y
\end{array} \quad \rightarrow \quad \ldots
\]

linked list \( \ell \)

Analysis: What is the length of \( \ell \)?

\[
\text{Exp}[|\ell|] = \Pr[a_i \in \ell] + \Pr[a_{i-1} \in \ell] + \ldots + \Pr[a_{i-w+1} \in \ell]
\]

\[= 1 + \frac{1}{2} + \frac{1}{3} +\]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

current sample

\[ \text{linked list } \ell \]

Analysis: What is the length of \( \ell \)?

\[
\text{Exp}[|\ell|] = Pr[a_i \in \ell] + Pr[a_{i-1} \in \ell] + \ldots + Pr[a_{i-w+1} \in \ell] \\
= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{w}
\]
Sliding Window Sampling

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m$

current sample

\[ \text{linked list } \ell \]

Analysis: What is the length of $\ell$?

\[
\text{Exp}[|\ell|] = \text{Pr}[a_i \in \ell] + \text{Pr}[a_{i-1} \in \ell] + \ldots + \text{Pr}[a_{i-w+1} \in \ell]
\]

\[
= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{w} = \mathcal{O}(\log w)
\]
Sliding Window Sampling

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_i, \ldots, a_m \]

**Analysis:**

\[ \text{Exp[ memory usage ]} = \mathcal{O}(\log w \cdot \log n) \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]
Count-Min Sketch

[Cormode, Muthukrishnan ’04]

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Trivial solution: maintain \( n \) counters
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

2/\varepsilon

rand. hash funct. \( h_1 \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

rand. hash funct. \( h_1 \)

2/\( \varepsilon \)
Count-Min Sketch

\[ \text{rand. hash funct. } h_1 \]

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ 2/\varepsilon \]
Count-Min Sketch

[Cormode, Muthukrishnan '04]

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

rand. hash funct. \( h_1 \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ \text{rand. hash funct. } h_1 \]

\[ \frac{2}{\varepsilon} \]

[Count-Min Sketch][Cormode, Muthukrishnan '04]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ \text{rand. hash funct. } h_1 \]

\[ +1 \]

\[ \frac{2}{\varepsilon} \]
Count-Min Sketch

[Cormode, Muthukrishnan '04]

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

rand. hash funct. \( h_1 \)

\[ \frac{2}{\varepsilon} \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Random hash function \( h_1 \)

\[ \frac{2}{\varepsilon} \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Rand. hash funct. \( h_1 \)
Count-Min Sketch

\[ \hat{f}_i = \min_{j \in \{1, \ldots, m\}} (h_j(i) \mod M) + 1 \]

Point query: \( f_i = ? \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Give \( \hat{f}_i \) as estimate for \( f_i \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Give \( \hat{f_i} \) as estimate for \( f_i \)

Analysis:

\[ 2/\varepsilon \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Give \( \hat{f}_i \) as estimate for \( f_i \)

Analysis: \( \hat{f}_i \geq f_i \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Give \( \hat{f}_i \) as estimate for \( f_i \)

Analysis: \( \hat{f}_i \geq f_i \)

\[ \text{Exp}[\hat{f}_i] \leq f_i + \varepsilon \cdot m/2 \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Give \( \hat{f}_i \) as estimate for \( f_i \)

Analysis: \( \hat{f}_i \geq f_i \)

\[ \text{Exp}[\hat{f}_i] \leq f_i + \varepsilon \cdot m/2 \]

Markov-Ineq.: \( \text{Pr}[\hat{f}_i > f_i + \varepsilon \cdot m] \leq \frac{1}{2} \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

rand. hash funct. \( h_1 \)

\[ \frac{2}{\varepsilon} \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ \frac{2}{\varepsilon} \]

\[ d = \log \frac{1}{\delta} \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[
\begin{array}{c}
\text{rand. hash funct. } h_1 \\
\text{rand. hash funct. } h_2 \\
\text{rand. hash funct. } h_3 \\
\vdots \\
\text{rand. hash funct. } h_d
\end{array}
\]

\[
\begin{array}{c}
\text{2/}\epsilon
\end{array}
\]

\[
\begin{array}{c}
d = \\
\log \frac{1}{\delta}
\end{array}
\]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ \begin{aligned}
\text{rand. hash funct. } h_1 \\
\text{rand. hash funct. } h_2 \\
\text{rand. hash funct. } h_3 \\
\vdots \\
\text{rand. hash funct. } h_d
\end{aligned} \]

\[ 2/\varepsilon \]

\[ d = \log \frac{1}{\delta} \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ \text{rand. hash funct. } h_1 \]
\[ \text{rand. hash funct. } h_2 \]
\[ \text{rand. hash funct. } h_3 \]
\[ \vdots \]
\[ \text{rand. hash funct. } h_d \]

\[ d = \log \frac{1}{\delta} \]

\[ \frac{2}{\varepsilon} \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Random hash functions \( h_1, h_2, h_3, \ldots, h_d \)

\[ 2/\varepsilon \]

\[ d = \log \frac{1}{\delta} \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ \text{Point query: } f_i = ? \]
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

Point query: \( f_i = ? \)

Give \( \hat{f}_i := \text{minimum of } \square \)-values as estimate for \( f_i \)
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

\[ \text{Analysis: } Pr[\hat{f}_i > f_i + \varepsilon \cdot m] \leq \left(\frac{1}{2}\right)^d = \delta \]

Mariano Zelke Basic algorithmic techniques for data streams 10/12
Count-Min Sketch

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, \ldots, a_j, \ldots, a_m \]

**Analysis:** \( \Pr[\hat{f}_i > f_i + \varepsilon \cdot m] \leq \left(\frac{1}{2}\right)^d = \delta \)

**Memory consumption:** \( \mathcal{O}\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta} \cdot \log m + \log \frac{1}{\delta} \cdot \log n\right) \)
Recap

- Reservoir sampling for uniform selection
- AMS sampling for frequency moments
- Sliding window sampling
- Count-Min sketch for point queries
References