Algorithm Foundations of Data Science and Engineering

Lecture 4: Sketch for Data Streaming

YANHAO WANG

DaSE @ ECNU
(for course related communications)
yhwang@dase.ecnu.edu.cn

Sep. 27, 2021

Outline

Streaming Data

Item Frequencies

- Deterministic Algorithm
- Randomized Algorithm

Motivations

Sensor data

Most of the algorithms are mining a database, where all our data is available when and if we want it. But...

Motivations

Sensor data

Most of the algorithms are mining a database, where all our data is available when and if we want it. But...

- Imagine a temperature sensor bobbing about in the ocean, sending back to a base station a reading of the surface temperature each hour (not very interesting since the data rate is so low).

Motivations

Sensor data

Most of the algorithms are mining a database, where all our data is available when and if we want it. But...

- Imagine a temperature sensor bobbing about in the ocean, sending back to a base station a reading of the surface temperature each hour (not very interesting since the data rate is so low).
- Given the GPS sensor, and let it report surface height instead of temperature. If it sends a 4-byte real number every tenth of a second, then it produces 3.5 megabytes per day (not very interesting since # sensors is only one).

Motivations

Sensor data

Most of the algorithms are mining a database, where all our data is available when and if we want it. But...

- Imagine a temperature sensor bobbing about in the ocean, sending back to a base station a reading of the surface temperature each hour (not very interesting since the data rate is so low).
- Given the GPS sensor, and let it report surface height instead of temperature. If it sends a 4-byte real number every tenth of a second, then it produces 3.5 megabytes per day (not very interesting since # sensors is only one).
- To learn something about ocean behavior, we might want to deploy a million sensors (is not very many since there would be one for every 150 square miles), each sending back a stream, at the rate of ten per second. Now we have 3.5 terabytes arriving every day.

Motivations Cont'd

Image data

- Satellites often send down to earth streams consisting of many terabytes of images per day.
- Surveillance cameras produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at intervals like one second.
- London has six million such cameras, each producing a stream.

Motivations Cont'd

Image data

- Satellites often send down to earth streams consisting of many terabytes of images per day.
- Surveillance cameras produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at intervals like one second.
- London has six million such cameras, each producing a stream.

Internet traffic

- A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its output.
- Normally, the job of it is to transmit data and not to retain it or query it.
- But there is a tendency to put more capability into the switch, e.g., the ability to detect denial-of-service attacks or the ability to reroute packets based on information about congestion in the network.

Motivations Cont'd

Internet traffic

- Web sites receive streams of various types. Many interesting things can be learned from these streams.
- For example, an increase in queries like "sore throat" enables us to track the spread of viruses.
- A sudden increase in the click rate for a link could indicate some news connected to that page, or it could mean that the link is broken and needs to be repaired.

Motivations Cont'd

Internet traffic

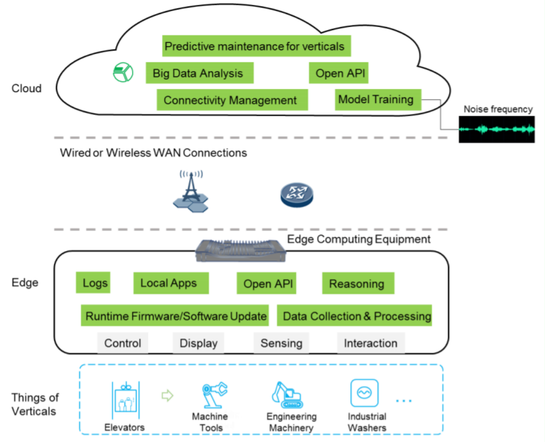
- Web sites receive streams of various types. Many interesting things can be learned from these streams.
- For example, an increase in queries like "sore throat" enables us to track the spread of viruses.
- A sudden increase in the click rate for a link could indicate some news connected to that page, or it could mean that the link is broken and needs to be repaired.

Many applications

Videos, email messages, webpages, chats, search queries, shopping history, GPS trials, financial transactions, stock exchange data, electricity consumption, Astronomy, Physics, medical imaging, weather measurements, maps, telephony data, audio tracks and songs, etc.

Motivations Cont'd

Edge Computing



Streaming data

Characters

A sequence $\sigma =< a_1, a_2, \cdots, a_m, \cdots >$, where the elements of the sequence are drawn from the universe $[n] := \{1, 2, \cdots, n\}$.

- Continuously arriving
- Large volume
- High speed

Streaming data

Characters

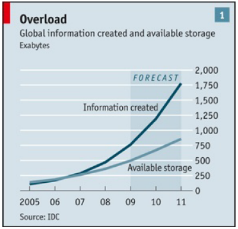
A sequence $\sigma =< a_1, a_2, \cdots, a_m, \cdots >$, where the elements of the sequence are drawn from the universe $[n] := \{1, 2, \cdots, n\}$.

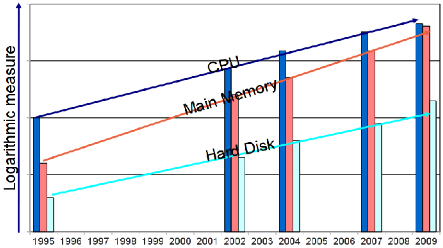
- Continuously arriving
- Large volume
- High speed

Algorithms for data stream

- Our central goal will be to process the input stream using a small amount of space s.
- We do not have random access to the tokens. We can only scan the sequence following the given order in p passes (some "small" integer p, for example $p = 1$).

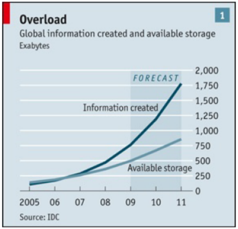
The need for streaming data algorithm

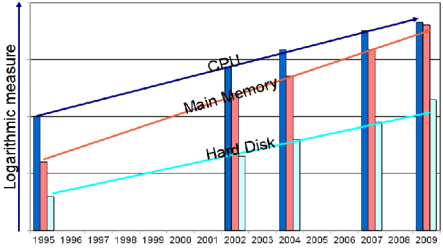




Data growth

The need for streaming data algorithm

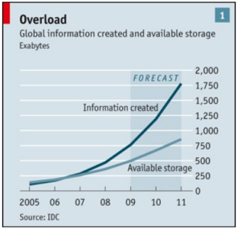


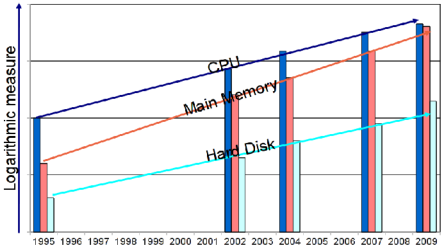


Data growth

- IDC reports that enterprise data stores will grow an average of 60% annually.

The need for streaming data algorithm





Data growth

- IDC reports that enterprise data stores will grow an average of 60% annually.
- Moore law is the observation that the number of transistors in a dense integrated circuit doubles approximately every two years.

Streaming algorithm VS. traditional algorithm Cont'd

Sketch

A sketch $C(\sigma)$ of some data σ with respect to some function ϕ is a compression of σ that allows us to compute or approximately compute ϕ given access only to $C(\sigma)$.

Streaming algorithm VS. traditional algorithm Cont'd

Sketch

A sketch $C(\sigma)$ of some data σ with respect to some function ϕ is a compression of σ that allows us to compute or approximately compute ϕ given access only to $C(\sigma)$.

- There exist exact solutions for some trivial tasks, such as count items, sum values, sample, find min or max.

Streaming algorithm VS. traditional algorithm Cont'd

Sketch

A sketch $C(\sigma)$ of some data σ with respect to some function ϕ is a compression of σ that allows us to compute or approximately compute ϕ given access only to $C(\sigma)$.

- There exist exact solutions for some trivial tasks, such as count items, sum values, sample, find min or max.
- For non-trivial task, different problems maybe design different sketches to solve them.

Streaming algorithm VS. traditional algorithm Cont'd

Sketch

A sketch $C(\sigma)$ of some data σ with respect to some function ϕ is a compression of σ that allows us to compute or approximately compute ϕ given access only to $C(\sigma)$.

- There exist exact solutions for some trivial tasks, such as count items, sum values, sample, find min or max.
- For non-trivial task, different problems maybe design different sketches to solve them.
- We are expected to give a bound for the estimation.

Streaming algorithm VS. traditional algorithm Cont'd

Sketch

A sketch $C(\sigma)$ of some data σ with respect to some function ϕ is a compression of σ that allows us to compute or approximately compute ϕ given access only to $C(\sigma)$.

- There exist exact solutions for some trivial tasks, such as count items, sum values, sample, find min or max.
- For non-trivial task, different problems maybe design different sketches to solve them.
- We are expected to give a bound for the estimation.

	Traditional algorithm	Streaming algorithm
Type	finite & static	infinite, dynamic & high speed
Storage	hard disk	memory & space limitation
Efficiency	not real time	real time & ad-hoc
Return	accurate	approximate result

Approximation and randomization

- Many things are hard to compute exactly over a stream

Approximation and randomization

- Many things are hard to compute exactly over a stream
  - Is the count of all items the same in two different streams?
  - Requires linear space to compute exactly.

Approximation and randomization

- Many things are hard to compute exactly over a stream
  - Is the count of all items the same in two different streams?
  - Requires linear space to compute exactly.
- Approximation: find an answer correct within some factor
  - Find an answer that is within 10% of correct result
  - More generally, a $(1 \pm \epsilon)$ factor approximation

Approximation and randomization

- Many things are hard to compute exactly over a stream
  - Is the count of all items the same in two different streams?
  - Requires linear space to compute exactly.
- Approximation: find an answer correct within some factor
  - Find an answer that is within 10% of correct result
  - More generally, a $(1 \pm \epsilon)$ factor approximation
- Randomization: allow a small probability of failure
  - Answer is correct, except with probability 1 in 10,000
  - More generally, success probability $(1 - \delta)$

Approximation and randomization

- Many things are hard to compute exactly over a stream
  - Is the count of all items the same in two different streams?
  - Requires linear space to compute exactly.
- Approximation: find an answer correct within some factor
  - Find an answer that is within 10% of correct result
  - More generally, a $(1 \pm \epsilon)$ factor approximation
- Randomization: allow a small probability of failure
  - Answer is correct, except with probability 1 in 10,000
  - More generally, success probability $(1 - \delta)$
- Approximation and Randomization: (ϵ, δ)-approximations

The quality of an algorithm's answer

Algorithm

We shall typically seek to compute only an approximation of the true value of $\phi(\sigma)$, because many basic functions can be provably not be computed exactly using sublinear space.

The quality of an algorithm's answer

Algorithm

We shall typically seek to compute only an approximation of the true value of $\phi(\sigma)$, because many basic functions can be provably not be computed exactly using sublinear space.

- Relative version: let $\mathcal{A}(\sigma)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input σ; note that this is a random variable. Let ϕ be the function that $\mathcal{A}$ is supposed to compute. We say that the algorithm $(\epsilon, \delta)-$approximation ϕ if we have

$$P\left[|\frac{\mathcal{A}(\sigma)}{\phi(\sigma)} - 1| > \epsilon\right] \leq \delta, \text{ i.e., } P\left[|\mathcal{A}(\sigma) - \phi(\sigma)| > \epsilon \cdot \phi(\sigma)\right] \leq \delta.$$

The quality of an algorithm's answer

Algorithm

We shall typically seek to compute only an approximation of the true value of $\phi(\sigma)$, because many basic functions can be provably not be computed exactly using sublinear space.

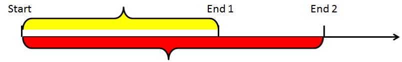
- Relative version: let $\mathcal{A}(\sigma)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input σ; note that this is a random variable. Let ϕ be the function that $\mathcal{A}$ is supposed to compute. We say that the algorithm $(\epsilon, \delta)-$approximation ϕ if we have

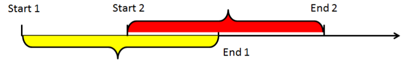
$$P\Big[|\frac{\mathcal{A}(\sigma)}{\phi(\sigma)} - 1| > \epsilon\Big] \leq \delta, \text{ i.e., } P\Big[|\mathcal{A}(\sigma) - \phi(\sigma)| > \epsilon \cdot \phi(\sigma)\Big] \leq \delta.$$

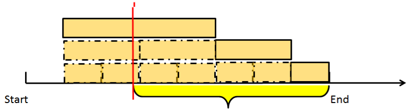
- Absolute version: In the above setup, the algorithm $(\epsilon, \delta)-$additively-approximation ϕ if we have

$$P[|\mathcal{A}(\sigma) - \phi(\sigma)| > \epsilon] \leq \delta.$$

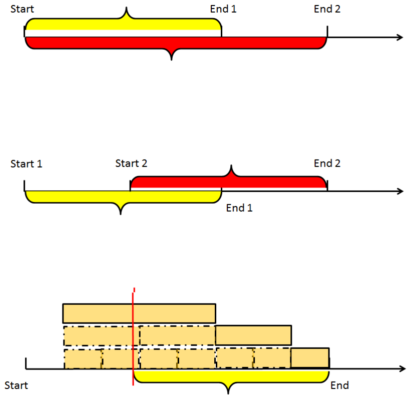
Streaming data modeling







Streaming data modeling

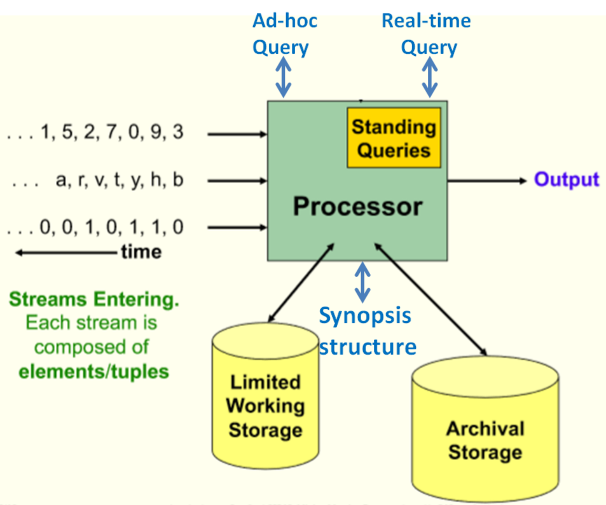


Model

- Landmark model:
- Sliding window model:
- Exponential histogram model:

Framework of streaming data mining

Framework



Item frequencies

We have a stream $\sigma =< a_1, a_2, \cdots , a_m >$, with each $a_i \in [n]$ and this implicitly defines a frequency vector

$$f = (f_1, f_2, \cdots , f_n).$$

Note that $\sum_{i=1}^{n} f_i = m$.

Item frequencies

We have a stream $\sigma =< a_1, a_2, \cdots , a_m >$, with each $a_i \in [n]$ and this implicitly defines a frequency vector

$$f = (f_1, f_2, \cdots , f_n).$$

Note that $\sum_{i=1}^{n} f_i = m$.

- Majority problem: if $\exists j : f_j > \frac{m}{2}$, then output j, otherwise $\perp$.

Item frequencies

We have a stream $\sigma =< a_1, a_2, \cdots , a_m >$, with each $a_i \in [n]$ and this implicitly defines a frequency vector

$$f = (f_1, f_2, \cdots , f_n).$$

Note that $\sum_{i=1}^{n} f_i = m$.

- Majority problem: if $\exists j : f_j > \frac{m}{2}$, then output j, otherwise $\perp$.
- General problem
  - Given a parameter k, output the set $\{j : f_j \geq \frac{m}{k}\}$.
  - Alternatively, given a parameter ψ, output the set $\{j : f_j > \psi m\}$.

Item frequencies

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$ and this implicitly defines a frequency vector

$$f = (f_1, f_2, \cdots, f_n).$$

Note that $\sum_{i=1}^{n} f_i = m$.

- Majority problem: if $\exists j : f_j > \frac{m}{2}$, then output j, otherwise $\perp$.
- General problem
  - Given a parameter k, output the set $\{j : f_j \geq \frac{m}{k}\}$.
  - Alternatively, given a parameter ψ, output the set $\{j : f_j > \psi m\}$.
- For example, $\sigma =< a, b, a, c, c, a, b, d >$, we have $f_a = 3$, $f_b = 2$, $f_c = 2$, $f_d = 1$.
  - For $k = 4$, the frequent items are a, b, c.
  - And for $\psi = 0.3$, the frequent item is a.

Frequency estimation problem

Problem definition

The task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.

Frequency estimation problem

Problem definition

The task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.

Alternatively, there are set of queries for the frequency estimation problem.

Frequency estimation problem

Problem definition

The task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.

Alternatively, there are set of queries for the frequency estimation problem.

- Point query: For $i \in [n]$, estimate f_i;

Frequency estimation problem

Problem definition

The task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.

Alternatively, there are set of queries for the frequency estimation problem.

- Point query: For $i \in [n]$, estimate f_i;
- Range query: For $i, j \in [n]$, estimate $f_i + f_{i+1} + \cdots + f_j$;

Frequency estimation problem

Problem definition

The task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.

Alternatively, there are set of queries for the frequency estimation problem.

- Point query: For $i \in [n]$, estimate f_i;
- Range query: For $i, j \in [n]$, estimate $f_i + f_{i+1} + \cdots + f_j$;
- Quantile query: For $\phi \in [0, 1]$ find j with $f_1 + \cdots + f_j \approx \phi m$;

Frequency estimation problem

Problem definition

The task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.

Alternatively, there are set of queries for the frequency estimation problem.

- Point query: For $i \in [n]$, estimate f_i;
- Range query: For $i, j \in [n]$, estimate $f_i + f_{i+1} + \cdots + f_j$;
- Quantile query: For $\phi \in [0, 1]$ find j with $f_1 + \cdots + f_j \approx \phi m$;
- Heavy hitter problem: For $\phi \in [0, 1]$, find all i with $f_i \geq \phi m$.

Frequency estimation problem

Problem definition

The task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.

Alternatively, there are set of queries for the frequency estimation problem.

- Point query: For $i \in [n]$, estimate f_i;
- Range query: For $i, j \in [n]$, estimate $f_i + f_{i+1} + \cdots + f_j$;
- Quantile query: For $\phi \in [0, 1]$ find j with $f_1 + \cdots + f_j \approx \phi m$;
- Heavy hitter problem: For $\phi \in [0, 1]$, find all i with $f_i \geq \phi m$.

In this lecture, we focus on the tasks of finding the frequent items from an input streaming.

Outline

Streaming Data

Item Frequencies

Deterministic Algorithm

Randomized Algorithm

$\psi-$frequent items

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i|a_i = j\}|$. The exact $\psi-$frequent items comprise the set $\{j|f_j > \psi m\}$.

$\psi-$frequent items

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i|a_i = j\}|$. The exact $\psi-$frequent items comprise the set $\{j|f_j > \psi m\}$.

- For example, $\sigma =< a, b, a, c, c, a, b, d >$, we have $f_a = 3$, $f_b = 2$, $f_c = 2$, $f_d = 1$. For $\psi = 0.2$, the frequent item is a, b, and c.

$\psi-$frequent items

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i|a_i = j\}|$. The exact $\psi-$frequent items comprise the set $\{j|f_j > \psi m\}$.

- For example, $\sigma =< a, b, a, c, c, a, b, d >$, we have $f_a = 3$, $f_b = 2$, $f_c = 2$, $f_d = 1$. For $\psi = 0.2$, the frequent item is a, b, and c.
- Alternatively, we define an approximate version of $\psi-$frequent items.

$\psi-$frequent items

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i|a_i = j\}|$. The exact $\psi-$frequent items comprise the set $\{j|f_j > \psi m\}$.

- For example, $\sigma =< a, b, a, c, c, a, b, d >$, we have $f_a = 3$, $f_b = 2$, $f_c = 2$, $f_d = 1$. For $\psi = 0.2$, the frequent item is a, b, and c.
- Alternatively, we define an approximate version of $\psi-$frequent items.
  - The $\epsilon-$approximate frequent items problem is to return a set of items F so that for all items $j \in F$, $\widehat{f_j} > (\psi - \epsilon)m$, and there is no $j \notin F$ such that $\widehat{f_j} > \psi m$.

$\psi-$frequent items

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i|a_i = j\}|$. The exact $\psi-$frequent items comprise the set $\{j|f_j > \psi m\}$.

- For example, $\sigma =< a, b, a, c, c, a, b, d >$, we have $f_a = 3$, $f_b = 2$, $f_c = 2$, $f_d = 1$. For $\psi = 0.2$, the frequent item is a, b, and c.
- Alternatively, we define an approximate version of $\psi-$frequent items.
  - The $\epsilon-$approximate frequent items problem is to return a set of items F so that for all items $j \in F$, $\widehat{f_j} > (\psi - \epsilon)m$, and there is no $j \notin F$ such that $\widehat{f_j} > \psi m$.
  - That is, $\epsilon-$approximate frequent items problem is to process a stream s.t., given any j, an $\widehat{f_j} \le f_j \le \widehat{f_j} + \epsilon m$

Misra-Gries summary

The Misra-Gries summary is a simple algorithm that solves the frequent items problem. It can be viewed as a generalization of Majority to track multiple frequent items.

Misra-Gries summary

The Misra-Gries summary is a simple algorithm that solves the frequent items problem. It can be viewed as a generalization of Majority to track multiple frequent items.

Misra-Gries Algorithm

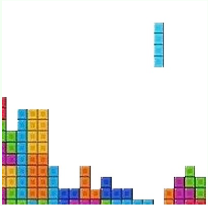
$m \leftarrow 0$; $F \leftarrow \emptyset$;
For each i;
  $m \leftarrow m + 1$;
  if $i \in F$;
    $c_i \leftarrow c_i + 1$;
  else
    $F \leftarrow F \cup \{i\}$;
    $c_i \leftarrow 1$;
  if $|F| \geq k$;
  for all $j \in F$;
    $c_j \leftarrow c_j - 1$;
    if $c_j = 0$;
      $F \leftarrow F \backslash \{j\}$;
Return a if $a \in F$;

Example of Misra-Gries summary

Given the input streaming a, b, a, c, d, e, a, d, and $k = 3$, i.e., three counters.

Example of Misra-Gries summary

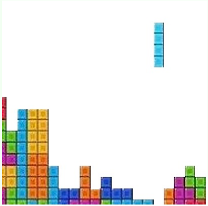
Given the input streaming a, b, a, c, d, e, a, d, and $k = 3$, i.e., three counters.



Tetris is a tile-matching puzzle video game designed by game designer Alexey Pajitnov.

Example of Misra-Gries summary

Given the input streaming a, b, a, c, d, e, a, d, and $k = 3$, i.e., three counters.



Tetris is a tile-matching puzzle video game designed by game designer Alexey Pajitnov.

input	operation	result
a	add	$F = \{(a, 1)\}$
b	add	$F = \{(a, 1), (b, 1)\}$
a	update	$F = \{(a, 2), (b, 1)\}$
c	add delete	$F = \{(a, 2), (b, 1), (c, 1)\}$ $F = \{(a, 1)\}$
d	add	$F = \{(a, 1), (d, 1)\}$
e	add delete	$F = \{(a, 1), (d, 1), (e, 1)\}$ $F = \{\}$
a	add	$F = \{(a, 1)\}$
d	add	$F = \{(a, 1), (d, 1)\}$

Analysis of Misra-Gries summary

Theorem I

Minus operation is triggered at most $\frac{m}{k}$.
Proof:
Counters will decrease k over all for every minus operation. Finally, the sum of counters is at least 0. Thus, the minus operation is triggered at most $\frac{m}{k}$.

Analysis of Misra-Gries summary

Theorem I

Minus operation is triggered at most $\frac{m}{k}$.

Proof:

Counters will decrease k over all for every minus operation. Finally, the sum of counters is at least 0. Thus, the minus operation is triggered at most $\frac{m}{k}$.

Theorem II

All $\psi-$frequent item will be detected by Misra-Gries algorithm when $k = \lceil \frac{1}{\psi} \rceil$.

Proof:

Since minus operation is triggered at most $\frac{m}{k}$, a counter c_j satisfies $c_j \leq f_j \leq c_j + \frac{m}{k}$. For each $j \notin F$, we have $c_j = 0$, i.e., $f_j \leq \frac{m}{k} \leq \psi m$. Thus, all items with $f_j > \psi m$ will be detected by Misra-Gries algorithm.

Outline

Streaming Data

Item Frequencies
- Deterministic Algorithm
- Randomized Algorithm

Problem formulation

Point query

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i | a_i = j\}|$. For a fixed item a, we want to query its frequency f_a.

Problem formulation

Point query

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i|a_i = j\}|$. For a fixed item a, we want to query its frequency f_a.

- The query item is unknown for a system. Thus, we need to compute the frequencies for all items.

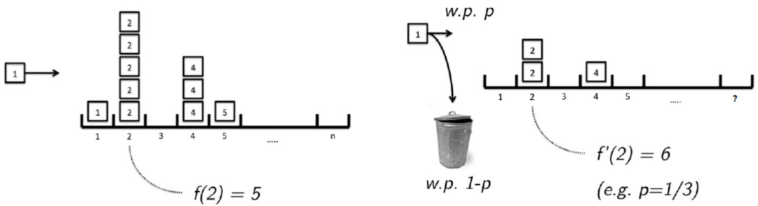
Problem formulation

Point query

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$, the frequency of an item j is $f_j = |\{i | a_i = j\}|$. For a fixed item a, we want to query its frequency f_a.

- The query item is unknown for a system. Thus, we need to compute the frequencies for all items.
- We employ the approximate and randomized algorithm to do that.
  - Naive sampling;
  - Count sketch;
  - Count-min sketch.

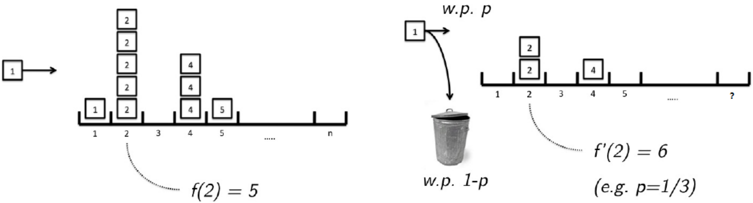
Naive approach



Sampling

Let $\sigma = < a_1, a_2, \cdots, a_m >$ be a stream, with each $a_i \in [n]$.

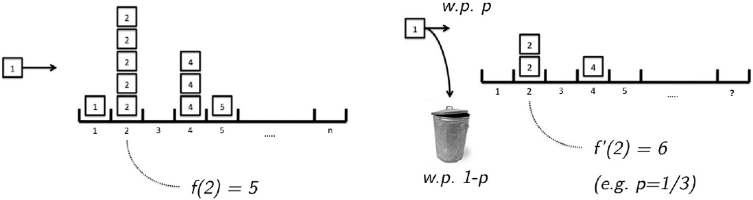
Naive approach



Sampling

Let $\sigma = < a_1, a_2, \cdots, a_m >$ be a stream, with each $a_i \in [n]$.Pick each element a_i from σ independently into our set $\widehat{\sigma}$ with probability $p = \frac{M}{m}$.

Naive approach



Sampling

Let $\sigma =< a_1, a_2, \cdots, a_m >$ be a stream, with each $a_i \in [n]$.Pick each element a_i from σ independently into our set $\widehat{\sigma}$ with probability $p = \frac{M}{m}$.

Let r.v. $X_i = \begin{cases} 1, & \text{element } a_i \text{ is picked;} \\ 0, & \text{otherwise.} \end{cases}$

Analysis of naive approach

Analysis

- The number of picked elements is $\sum_{i=1}^{m} X_i$. Its expectation is

$$E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{M}{m} = M.$$

Analysis of naive approach

Analysis

- The number of picked elements is $\sum_{i=1}^{m} X_i$. Its expectation is

$$E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{M}{m} = M.$$

- g_j be the frequency of j in $\widehat{\sigma}$, that is $g_j = \sum_{i:a_i=j} X_i$. Thus, $E[g_j] = pf_j$.

Analysis of naive approach

Analysis

- The number of picked elements is $\sum_{i=1}^{m} X_i$. Its expectation is

$$E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{M}{m} = M.$$

- g_j be the frequency of j in $\widehat{\sigma}$, that is $g_j = \sum_{i:a_i=j} X_i$. Thus, $E[g_j] = pf_j$.
- If we define $\widehat{f_j} = \frac{g_j}{p}$, we have $E[\widehat{f_j}] = \frac{E[g_j]}{p} = \frac{pf_j}{p} = f_j$.

Analysis of naive approach

Analysis

- The number of picked elements is $\sum_{i=1}^{m} X_i$. Its expectation is

$$E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{M}{m} = M.$$

- g_j be the frequency of j in $\widehat{\sigma}$, that is $g_j = \sum_{i:a_i=j} X_i$. Thus, $E[g_j] = pf_j$.
- If we define $\widehat{f_j} = \frac{g_j}{p}$, we have $E[\widehat{f_j}] = \frac{E[g_j]}{p} = \frac{pf_j}{p} = f_j$.
- In terms of Chernoff bound, we have $P(|\widehat{f_j} - f_j| \geq \epsilon f_j) = P(|g_j - pf_j| \geq \epsilon pf_j) = P(|\frac{g_j}{f_j} - p| \geq \epsilon p) \leq 2\exp\left(-p\epsilon^2/4\right) < \delta$.

Analysis of naive approach

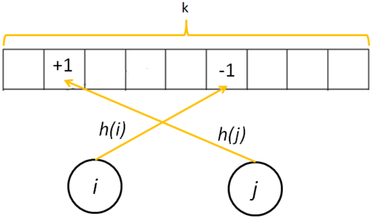
Analysis

- The number of picked elements is $\sum_{i=1}^{m} X_i$. Its expectation is

$$E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{M}{m} = M.$$

- g_j be the frequency of j in $\widehat{\sigma}$, that is $g_j = \sum_{i:a_i=j} X_i$. Thus, $E[g_j] = pf_j$.
- If we define $\widehat{f_j} = \frac{g_j}{p}$, we have $E[\widehat{f_j}] = \frac{E[g_j]}{p} = \frac{pf_j}{p} = f_j$.
- In terms of Chernoff bound, we have $P(|\widehat{f_j} - f_j| \geq \epsilon f_j) = P(|g_j - pf_j| \geq \epsilon pf_j) = P(|\frac{g_j}{f_j} - p| \geq \epsilon p) \leq 2\exp\left(-p\epsilon^2/4\right) < \delta$.
- Thus the space requirement is therefore $M = O(\frac{m \log 1/\delta}{\epsilon^2})$, which is related to the size of streaming data. Thus, the approach is inefficient.

Basic count sketch



Algorithm

1: $C[1...k] \leftarrow \overleftarrow{0}$, where $k = \frac{3}{\epsilon^2}$;
2: Choose a random hash function $h : [n] \to [k]$ (uniformly);
3: Choose a random hash function $g : [n] \to \{-1, 1\}$ (uniformly);
Process item (i, c), where $c = 1$:
4: $C[h(i)] \leftarrow C[h(i)] + cg(i)$;
Output:
5: On query a, report $\widehat{f_a} = g(a)C[h(a)]$;

Example of basic count sketch

Suppose we are processing the input $< a, b, c, a, b, a >$. With three counters, there is no need to hash $a : 3, b : 2, c : 1$.

Example of basic count sketch

Suppose we are processing the input $< a, b, c, a, b, a >$. With three counters, there is no need to hash $a : 3, b : 2, c : 1$.
Let's suppose however that we have just two, and $h(a) = h(b) \neq h(c)$. There are eight possible functions $g : \{a, b, c\} - > \{+1, -1\}$. Here is a table of the outcomes.

Example of basic count sketch

Suppose we are processing the input $< a, b, c, a, b, a >$. With three counters, there is no need to hash $a : 3, b : 2, c : 1$.
Let's suppose however that we have just two, and $h(a) = h(b) \neq h(c)$. There are eight possible functions $g : \{a, b, c\} - > \{+1, -1\}$. Here is a table of the outcomes.

g(a)	g(b)	g(c)	$h(a)$ or $h(b)$	$h(c)$
+	+	+	+3 + 2 = 5	1
+	+	-	+3 + 2 = 5	- 1
+	-	+	+3 - 2 = 1	1
+	-	-	+3 - 2 = 1	- 1
-	+	+	-3 + 2 = -1	1
-	+	-	-3 + 2 = -1	- 1
-	-	+	-3 - 2 = -5	1
-	-	-	-3 - 2 = -5	-1

Example of basic count sketch Cont'd

g(a)	g(b)	g(c)	$h(a)$ or $h(b)$	$h(c)$
$+$	$+$	$+$	$+3 + 2 = 5$	1
$+$	$+$	-	$+3 + 2 = 5$	- 1
$+$	-	$+$	$+3$ - $2 = 1$	1
$+$	-	-	$+3$ - $2 = 1$	- 1
-	$+$	$+$	-3 $+ 2 =$ -1	1
-	$+$	-	-3 $+ 2 =$ -1	- 1
-	-	$+$	-3 - $2 =$ -5	1
-	-	-	-3 - $2 =$ -5	-1

Example of basic count sketch Cont'd

g(a)	g(b)	g(c)	$h(a)$ or $h(b)$	$h(c)$
+	+	+	+3 + 2 = 5	1
+	+	-	+3 + 2 = 5	- 1
+	-	+	+3 - 2 = 1	1
+	-	-	+3 - 2 = 1	- 1
-	+	+	-3 + 2 = -1	1
-	+	-	-3 + 2 = -1	- 1
-	-	+	-3 - 2 = -5	1
-	-	-	-3 - 2 = -5	-1

For computing f_a, $f_a = \frac{(5+5+1+1)-(-5-5-1-1)}{8} = 3$.

Example of basic count sketch Cont'd

g(a)	g(b)	g(c)	$h(a)$ or $h(b)$	$h(c)$
$+$	$+$	$+$	$+3 + 2 = 5$	1
$+$	$+$	-	$+3 + 2 = 5$	- 1
$+$	-	$+$	$+3$ - $2 = 1$	1
$+$	-	-	$+3$ - $2 = 1$	- 1
-	$+$	$+$	-3 $+ 2 =$ -1	1
-	$+$	-	-3 $+ 2 =$ -1	- 1
-	-	$+$	-3 - 2 $=$ -5	1
-	-	-	-3 - 2 $=$ -5	-1

For computing f_a, $f_a = \frac{(5+5+1+1)-(-5-5-1-1)}{8} = 3$.

For computing f_b, $f_b = \frac{(5+5-1-1)-(-5-5+1+1)}{8} = 2$.

Example of basic count sketch Cont'd

g(a)	g(b)	g(c)	$h(a)$ or $h(b)$	$h(c)$
$+$	$+$	$+$	$+3 + 2 = 5$	1
$+$	$+$	-	$+3 + 2 = 5$	- 1
$+$	-	$+$	$+3$ - 2 $= 1$	1
$+$	-	-	$+3$ - 2 $= 1$	- 1
-	$+$	$+$	-3 $+ 2 =$ -1	1
-	$+$	-	-3 $+ 2 =$ -1	- 1
-	-	$+$	-3 - 2 $=$ -5	1
-	-	-	-3 - 2 $=$ -5	-1

For computing f_a, $f_a = \frac{(5+5+1+1)-(-5-5-1-1)}{8} = 3.$
For computing f_b, $f_b = \frac{(5+5-1-1)-(-5-5+1+1)}{8} = 2.$
For computing f_c, $f_c = \frac{(1+1+1+1)-(-1-1-1-1)}{8} = 1.$

Basic count sketch analysis

Analysis

- Fix an arbitrary a and consider the output $X = \widehat{f_a}$ on query a.

Basic count sketch analysis

Analysis

- Fix an arbitrary a and consider the output $X = \widehat{f_a}$ on query a.
- What happens if $h(j) = h(a)$ for $j \in [n]$?

Basic count sketch analysis

Analysis

- Fix an arbitrary a and consider the output $X = \widehat{f_a}$ on query a.
- What happens if $h(j) = h(a)$ for $j \in [n]$? **Collision**!!!

Basic count sketch analysis

Analysis

- Fix an arbitrary a and consider the output $X = \widehat{f_a}$ on query a.
- What happens if $h(j) = h(a)$ for $j \in [n]$? **Collision**!!!
- For each token $j \in [n]$, let

$$Y_j = \begin{cases} 1, & \text{if } h(j) = h(a); \\ 0, & \text{otherwise.} \end{cases}$$

where Y_j is a r.v., which is the indicator for the event $h(j) = h(a)$.

Basic count sketch analysis

Analysis

- Fix an arbitrary a and consider the output $X = \widehat{f_a}$ on query a.
- What happens if $h(j) = h(a)$ for $j \in [n]$? **Collision**!!!
- For each token $j \in [n]$, let

$$Y_j = \begin{cases} 1, & \text{if } h(j) = h(a); \\ 0, & \text{otherwise.} \end{cases}$$

  where Y_j is a r.v., which is the indicator for the event $h(j) = h(a)$.
- Token j contributes to $C[h(a)]$ iff $h(j) = h(a)$, and the amount of the contribution is its frequency f_j times associated with the random sign $g(j)$.

Basic count sketch analysis Cont'd

Analysis

$$\widehat{f_a} = g(a)C[h(a)] = g(a)\sum_{j=1}^{n} f_j g(j) Y_j = f_a + g(a)\sum_{j\in[n]\setminus\{a\}} f_j \cdot g(j) \cdot Y_j.$$

Basic count sketch analysis Cont'd

Analysis

$$\widehat{f_a} = g(a)C[h(a)] = g(a)\sum_{j=1}^{n} f_j g(j) Y_j = f_a + g(a)\sum_{j\in[n]\setminus\{a\}} f_j \cdot g(j) \cdot Y_j.$$

- Note that g and h are independent, we have

$$E[g(j)\cdot Y_j] = E[g(j)]\cdot E[Y_j] = 0 \cdot E[Y_j] = 0.$$

- Furthermore,

$$E[\widehat{f_a}] = f_a + g(a)\sum_{j\in[n]\setminus\{a\}} f_j \cdot E(g(j)\cdot Y_j) = f_a,$$

that is, the output is an unbiased estimator for f_a.

Count sketch analysis Cont'd

Analysis

We still need to show that X is unlikely to deviate too much from its means.

Count sketch analysis Cont'd

Analysis

We still need to show that X is unlikely to deviate too much from its means. Thus, we need to analyze its variance.

Count sketch analysis Cont'd

Analysis

We still need to show that X is unlikely to deviate too much from its means.Thus, we need to analyze its variance.
We therefore compute $Var(\widehat{f_a})$

$$Var(\widehat{f_a}) = g(a)^2 Var\Big[\sum_{j\in[n]\setminus\{a\}} f_j \cdot g(j) \cdot Y_j\Big]$$

$$= g(a)^2 E\Big[\sum_{j\in[n]\setminus\{a\}} f_j \cdot g(j) \cdot Y_j\Big]^2 - g(a)^2 \Big(E\Big[\sum_{j\in[n]\setminus\{a\}} f_j \cdot g(j) \cdot Y_j\Big]\Big)^2$$

$$= g(a)^2 E\Big[\sum_{j\in[n]\setminus\{a\}} f_j \cdot g(j) \cdot Y_j\Big]^2$$

$$= E\Big[\sum_{j\in[n]\setminus\{a\}} f_j^2 Y_j^2 + \sum_{i\neq j\in[n]\setminus\{a\}} f_i f_j g(i) g(j) Y_i Y_j\Big]$$

Count sketch analysis Cont'd

- For $j \in [n] \setminus \{a\}$, we have

$$E[Y_j^2] = E[Y_j] = P[h(j) = h(a)] = \frac{1}{k}.$$

- For $i \neq j \in [n]$, we have

$$E[g(i) \cdot g(j) \cdot Y_i \cdot Y_j] = E[g(i)]E[g(j)]E[Y_i \cdot Y_j] = 0 \cdot E[Y_i \cdot Y_j] = 0.$$

Thus, $Var(\widehat{f_a})$ is computed as

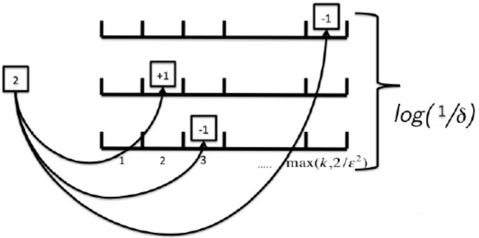
$$Var(\widehat{f_a}) = E\Big[\sum_{j \in [n] \setminus \{a\}} f_j^2 Y_j^2 + \sum_{i \neq j \in [n] \setminus \{a\}} f_i f_j g(i) g(j) Y_i Y_j\Big]$$
$$= \sum_{j \in [n] \setminus \{a\}} \frac{f_i^2}{k} + 0 = \frac{\|f\|_2^2 - f_a^2}{k} \doteq \frac{\|f_{-a}\|_2^2}{k}$$

Count sketch analysis Cont'd

Using Chebyshev's inequality

$$\leq P[|\widehat{f_a} - f_a| \geq \epsilon \|f_{-a}\|_2]$$
$$\leq \frac{Var(X)}{\epsilon^2 \|f_{-a}\|_2^2} = \frac{1}{k\epsilon^2} = \frac{1}{3}.$$

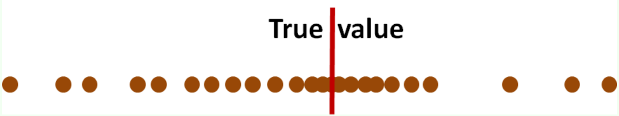
Boosting the probability (Tug of war)



Boosting the probability

Tug of war

1: $C[1...t][1...k] \leftarrow \overleftarrow{0}$, where $k = \frac{3}{\epsilon^2}$ and $t = O(\log(1/\delta))$;
2: Choose t random hash functions $h_1, \cdots, h_t : [n] \rightarrow [k]$ (uniformly);
3: Choose t random hash functions $g_1, \cdots, g_t : [n] \rightarrow \{-1, 1\}$ (uniformly);
Process item (j, c), where $c = 1$:
4: for i to t do $C[i][h_i(j)] \leftarrow C[i][h_i(j)] + cg_i(j)$;
Output:
5: On query a, report $\widehat{f_a} = \text{median}_{1 \le i \le t} g_i(a) C[i][h_i(a)]$;



Boosting the probability Cont'd

Analysis

Define

$$Y_i = \begin{cases} 1, & \text{if } |\widehat{f_a} - f_a| \geq \epsilon \|f\|_2; \\ 0, & \text{otherwise.} \end{cases}$$

Boosting the probability Cont'd

Analysis

Define

$$Y_i = \begin{cases} 1, & \text{if } |\widehat{f_a} - f_a| \geq \epsilon \|f\|_2; \\ 0, & \text{otherwise.} \end{cases}$$

- For $k = O(1/\epsilon^2)$, we have $P(Y_i = 1) < \frac{1}{3}$.

Boosting the probability Cont'd

Analysis

Define

$$Y_i = \begin{cases} 1, & \text{if } |\widehat{f_a} - f_a| \geq \epsilon \|f\|_2; \\ 0, & \text{otherwise.} \end{cases}$$

- For $k = O(1/\epsilon^2)$, we have $P(Y_i = 1) < \frac{1}{3}$.
- Note that $\mu = E(\sum_i Y_i) \leq \frac{t}{3}$. Then by the Chernoff bound,

$$P(\sum_i Y_i > \frac{t}{2}) \leq P(\sum_i Y_i > (1 + \frac{1}{2})\mu) \leq \exp\left(-\mu(1/2)^2/4\right),$$

$$\exp\left(-t/48\right) \leq \exp\left(-\mu(1/2)^2/4\right) < \delta$$

thus, we have $t = O(\log 1/\delta)$.

Boosting the probability Cont'd

Analysis

Define

$$Y_i = \begin{cases} 1, & \text{if } |\widehat{f_a} - f_a| \geq \epsilon \|f\|_2; \\ 0, & \text{otherwise.} \end{cases}$$

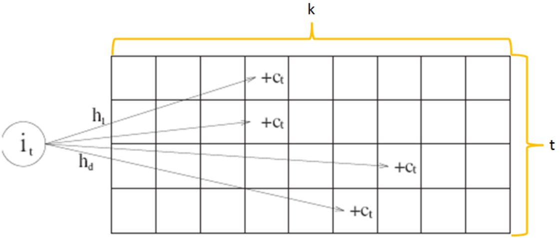
- For $k = O(1/\epsilon^2)$, we have $P(Y_i = 1) < \frac{1}{3}$.
- Note that $\mu = E(\sum_i Y_i) \leq \frac{t}{3}$. Then by the Chernoff bound,

$$P(\sum_i Y_i > \frac{t}{2}) \leq P(\sum_i Y_i > (1 + \frac{1}{2})\mu) \leq \exp(-\mu(1/2)^2/4),$$

$$\exp(-t/48) \leq \exp(-\mu(1/2)^2/4) < \delta$$

  thus, we have $t = O(\log 1/\delta)$.
- Finally, we can get an $(\epsilon, \delta)-$approximation in space complexity $O(\frac{\log 1/\delta}{\epsilon^2})$ counters.

Count min or Cormode-Muthukrishnan sketch

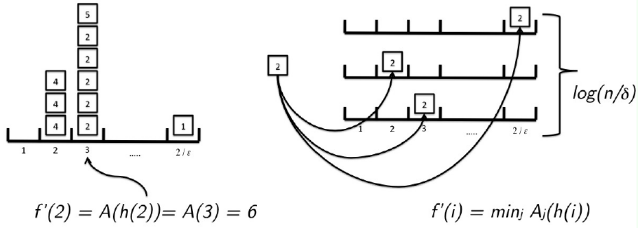


Algorithm

1: $C[1...t][1...k] \leftarrow \overleftarrow{0}$, where $k = \frac{2}{\epsilon}$ and $t = \lceil \log(1/\delta) \rceil$;
2: Choose t independent hash functions $h_1, h_2, \cdots, h_t : [n] \to [k]$;
Process item (j, c), where $c = 1$:
3: for $i = 1$ to t do $C[i][h_i(j)] \leftarrow C[i][h_i(j)] + c$;
Output:
4: On query a, report $\widehat{f_a} = \min_{1 \le i \le t} C[i][h_i(a)]$;

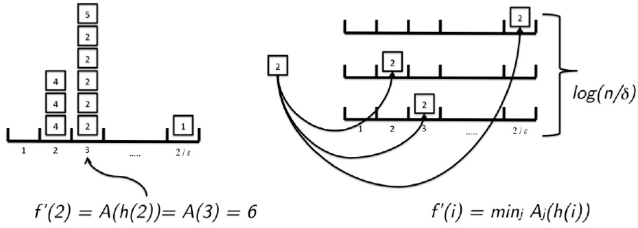
CM sketch analysis

Analysis



CM sketch analysis

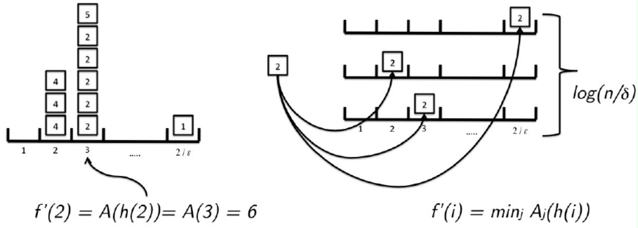
Analysis



- Clearly, for each i, we immediately have $f(a) \leq count[i, h_i(a)]$. However, the bound may be poor.

CM sketch analysis

Analysis



- Clearly, for each i, we immediately have $f(a) \leq count[i, h_i(a)]$. However, the bound may be poor.
- To get a better estimator, we will take the minimum over all the rows in count.

CM sketch analysis cont.

Analysis

- For a fixed a, we now analyze the collision in one such counter, say in $count[i, h_i(a)]$. Let r.v. X_i denote this collision.

CM sketch analysis cont.

Analysis

- For a fixed a, we now analyze the collision in one such counter, say in $count[i, h_i(a)]$. Let r.v. X_i denote this collision.
- For $j \in [n] \setminus \{a\}$, let

$$Y_{i,j} = \begin{cases} 1, & \text{if } h_i(j) = h_i(a); \\ 0, & \text{otherwise.} \end{cases}$$

  be the indicator of the event $h_i(j) = h_i(a)$. Notice that j makes a contribution to the counter iff $Y_{i,j} = 1$ (Note that $E(Y_{i,j}) = \frac{1}{k}$).

CM sketch analysis cont.

Analysis

- For a fixed a, we now analyze the collision in one such counter, say in $count[i, h_i(a)]$. Let r.v. X_i denote this collision.
- For $j \in [n] \setminus \{a\}$, let

$$Y_{i,j} = \begin{cases} 1, & \text{if } h_i(j) = h_i(a); \\ 0, & \text{otherwise.} \end{cases}$$

  be the indicator of the event $h_i(j) = h_i(a)$. Notice that j makes a contribution to the counter iff $Y_{i,j} = 1$ (Note that $E(Y_{i,j}) = \frac{1}{k}$).
- Thus, we have $X_i = \sum_{j \in [n] \setminus \{a\}} f_j Y_{i,j}$. By linearity of expectation,

$$E[X_i] = X_i = \sum_{j \in [n] \setminus \{a\}} \frac{f_j}{k} = \frac{\|f\|_1 - f_a}{k} = \frac{\|f_{-a}\|_1}{k}.$$

CM sketch analysis cont.

Analysis

- For a fixed a, we now analyze the collision in one such counter, say in $count[i, h_i(a)]$. Let r.v. X_i denote this collision.
- For $j \in [n] \setminus \{a\}$, let
$$Y_{i,j} = \begin{cases} 1, & \text{if } h_i(j) = h_i(a); \\ 0, & \text{otherwise.} \end{cases}$$
be the indicator of the event $h_i(j) = h_i(a)$. Notice that j makes a contribution to the counter iff $Y_{i,j} = 1$ (Note that $E(Y_{i,j}) = \frac{1}{k}$).
- Thus, we have $X_i = \sum_{j \in [n] \setminus \{a\}} f_j Y_{i,j}$. By linearity of expectation,
$$E[X_i] = X_i = \sum_{j \in [n] \setminus \{a\}} \frac{f_j}{k} = \frac{\|f\|_1 - f_a}{k} = \frac{\|f_{-a}\|_1}{k}.$$
- Since each $f_j \geq 0$, we have $X_i \geq 0$, and we can apply Markov's inequality to get (by choosing the value of k)
$$P[X_i \geq \epsilon \|f\|_1] \leq P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{\|f_{-a}\|_1}{k\epsilon \|f_{-a}\|_1} = \frac{1}{2}.$$

CM sketch analysis cont.

Analysis

- The above probability is for one counter. We have t such counters, mutually independent. The excess in the output $\widehat{f_a} - f_a$, is the minimum of excesses X_i over all $i \in [t]$. Thus

$$P[\widehat{f_a} - f_a \geq \epsilon \|f_{-a}\|_1] = P[\min\{X_1, \cdots, X_t\} \geq \epsilon \|f_{-a}\|_1]$$

$$= \Pi_{i=1}^{t} P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{1}{2^t}.$$

CM sketch analysis cont.

Analysis

- The above probability is for one counter. We have t such counters, mutually independent. The excess in the output $\widehat{f_a} - f_a$, is the minimum of excesses X_i over all $i \in [t]$. Thus

$$P[\widehat{f_a} - f_a \geq \epsilon \|f_{-a}\|_1] = P[\min\{X_1, \cdots, X_t\} \geq \epsilon \|f_{-a}\|_1]$$
$$= \Pi_{i=1}^{t} P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{1}{2^t}.$$

- Using our choice of t, this probability is at most δ. Thus, we have shown that, with high probability,

$$f_a \leq \widehat{f_a} \leq f_a + \epsilon \|f_{-a}\|_1.$$

CM sketch analysis cont.

Analysis

- The above probability is for one counter. We have t such counters, mutually independent. The excess in the output $\widehat{f}_a - f_a$, is the minimum of excesses X_i over all $i \in [t]$. Thus

$$P[\widehat{f}_a - f_a \geq \epsilon \|f_{-a}\|_1] = P[\min\{X_1, \cdots, X_t\} \geq \epsilon \|f_{-a}\|_1]$$
$$= \Pi_{i=1}^{t} P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{1}{2^t}.$$

- Using our choice of t, this probability is at most δ. Thus, we have shown that, with high probability,

$$f_a \leq \widehat{f}_a \leq f_a + \epsilon \|f_{-a}\|_1.$$

- Thus the space requirement is therefore $M = O(\frac{\log 1/\delta}{\epsilon})$ counters.

Take-home messages

- Data streaming
- Deterministic algorithm
- Randomized algorithm
  - Naive sampling
  - Count sketch
  - Count min sketch