Finding Frequent Items in Data Streams

Linus Seelinger

December 9, 2014
Problem

- **Aim:** Find most frequent elements in data stream
- **Restrictions:**
  - Only 1 pass
  - Limited memory
- **Application:** Search engines, most frequent queries
- **Naive Solutions:**
  - Sort stream $\rightarrow$ Multiple passes, memory
  - Counters for all distinct elements $\rightarrow$ Memory
Problem

- **Aim:** Find most frequent elements in data stream
- **Restrictions:**
  - Only 1 pass
  - Limited memory
- **Application:** Search engines, most frequent queries
- **Naive Solutions:**
  - Sort stream → Multiple passes, memory
  - Counters for all distinct elements → Memory
Problem

- **Aim:** Find most frequent elements in data stream
- **Restrictions:**
  - Only 1 pass
  - Limited memory
- **Application:** Search engines, most frequent queries
- **Naive Solutions:**
  - Sort stream → Multiple passes, memory
  - Counters for all distinct elements → Memory
Problem

- **Aim:** Find most frequent elements in data stream
- **Restrictions:**
  - Only 1 pass
  - Limited memory
- **Application:** Search engines, most frequent queries
- **Naive Solutions:**
  - Sort stream → Multiple passes, memory
  - Counters for all distinct elements → Memory
Problem

- **Aim**: Find most frequent elements in data stream
- **Restrictions**:
  - Only 1 pass
  - Limited memory
- **Application**: Search engines, most frequent queries
- **Naive Solutions**:
  - Sort stream → Multiple passes, memory
  - Counters for all distinct elements → Memory
Problem

- **Aim:** Find most frequent elements in data stream
- **Restrictions:**
  - Only 1 pass
  - Limited memory
- **Application:** Search engines, most frequent queries
- **Naive Solutions:**
  - Sort stream → Multiple passes, memory
  - Counters for all distinct elements → Memory
Aim: Find most frequent elements in data stream

Restrictions:
- Only 1 pass
- Limited memory

Application: Search engines, most frequent queries

Naive Solutions:
- Sort stream → Multiple passes, memory
- Counters for all distinct elements → Memory
Problem

- **Aim:** Find most frequent elements in data stream
- **Restrictions:**
  - Only 1 pass
  - Limited memory
- **Application:** Search engines, most frequent queries
- **Naive Solutions:**
  - Sort stream → Multiple passes, memory
  - Counters for all distinct elements → Memory
Stream of elements $S = q_1, \ldots, q_n$

- $q_i \in O = \{o_1, \ldots, o_m\}$
- $o_i$ occurs $n_i$ times in $S$
- Order $o_i$ so that $n_1 \geq n_2 \geq \ldots \geq n_m$
Stream of elements $S = q_1, \ldots, q_n$

$q_i \in O = \{o_1, \ldots, o_m\}$

$o_i$ occurs $n_i$ times in $S$

Order $o_i$ so that $n_1 \geq n_2 \geq \ldots \geq n_m$
Stream of elements \( S = q_1, \ldots, q_n \)

\( q_i \in O = \{o_1, \ldots, o_m\} \)

\( o_i \) occurs \( n_i \) times in \( S \)

Order \( o_i \) so that \( n_1 \geq n_2 \geq \ldots \geq n_m \)
Input

- Stream of elements $S = q_1, \ldots, q_n$
- $q_i \in O = \{o_1, \ldots, o_m\}$
- $o_i$ occurs $n_i$ times in $S$
- Order $o_i$ so that $n_1 \geq n_2 \geq \ldots \geq n_m$
Problem (formal)

\textit{FindCandidateTop}
- Input: Stream \( S \) and \( k, l \in \mathbb{N} \)
- Output: \( l \) elements containing the \( k \) most frequent elements

\textit{FindApproxTop}
- Input: Stream \( S \), \( k \in \mathbb{N} \) and \( \varepsilon \in \mathbb{R} \)
- Output: \( k \) elements with frequencies

\[ n_i > (1 - \varepsilon)n_k \]
Problem (formal)

\textbf{FindCandidateTop}
- Input: Stream $S$ and $k, l \in \mathbb{N}$
- Output: $l$ elements containing the $k$ most frequent elements

\textbf{FindApproxTop}
- Input: Stream $S$, $k \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$
- Output: $k$ elements with frequencies

\[ n_i > (1 - \varepsilon) n_k \]
Problem (formal)

**FindCandidateTop**
- **Input:** Stream $S$ and $k, l \in \mathbb{N}$
- **Output:** $l$ elements containing the $k$ most frequent elements

**FindApproxTop**
- **Input:** Stream $S$, $k \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$
- **Output:** $k$ elements with frequencies $n_i > (1 - \varepsilon) n_k$
Problem (formal)

**FindCandidateTop**
- Input: Stream $S$ and $k, l \in \mathbb{N}$
- Output: $l$ elements containing the $k$ most frequent elements

**FindApproxTop**
- Input: Stream $S$, $k \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$
- Output: $k$ elements with frequencies

$$n_i > (1 - \varepsilon)n_k$$
Problem (formal)

\textit{FindCandidateTop}

- \textbf{Input:} Stream $S$ and $k, l \in \mathbb{N}$
- \textbf{Output:} $l$ elements containing the $k$ most frequent elements

\textit{FindApproxTop}

- \textbf{Input:} Stream $S$, $k \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$
- \textbf{Output:} $k$ elements with frequencies

\[ n_i > (1 - \varepsilon) n_k \]
Problem (formal)

*FindCandidateTop*
- **Input:** Stream $S$ and $k, l \in \mathbb{N}$
- **Output:** $l$ elements containing the $k$ most frequent elements

*FindApproxTop*
- **Input:** Stream $S$, $k \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$
- **Output:** $k$ elements with frequencies

\[ n_i > (1 - \varepsilon)n_k \]
Algorithm
Data Structure - First Step

- Hashing function $s : O \rightarrow \{+1, -1\}$
- Counter $c \in \mathbb{Z}$
- Adding $q \in S$: $c += s[q]$
- Estimating $q \in O$: return $c \cdot s[q]$

Properties:

- $\mathbb{E}[c \cdot s[q]] = n_q$
- Variance extremely large
- Strong effect of frequent elements
Hashing function $s : O \rightarrow \{+1, -1\}$
Counter $c \in \mathbb{Z}$
Adding $q \in S$: $c += s[q]$
Estimating $q \in O$: return $c \cdot s[q]$

Properties:
- $\mathbb{E}[c \cdot s[q]] = n_q$
- Variance extremely large
- Strong effect of frequent elements
Data Structure - First Step

- **Hashing function** \( s : O \rightarrow \{+1, -1\} \)
- **Counter** \( c \in \mathbb{Z} \)
  - Adding \( q \in S \): \( c += s[q] \)
  - Estimating \( q \in O \): return \( c \cdot s[q] \)

Properties:

- \( \mathbb{E}[c \cdot s[q]] = n_q \)
- Variance extremely large
- Strong effect of frequent elements
Data Structure - First Step

- Hashing function $s : O \rightarrow \{+1, -1\}$
- Counter $c \in \mathbb{Z}$
- Adding $q \in S$: $c += s[q]$
  - Estimating $q \in O$: return $c \cdot s[q]$

Properties:

- $\mathbb{E}[c \cdot s[q]] = n_q$
- Variance extremely large
- Strong effect of frequent elements
Data Structure - First Step

- Hashing function $s : O \rightarrow \{+1, -1\}$
- Counter $c \in \mathbb{Z}$
- Adding $q \in S$: $c += s[q]$
- Estimating $q \in O$: return $c \cdot s[q]$

Properties:

- $\mathbb{E}[c \cdot s[q]] = n_q$
- Variance extremely large
- Strong effect of frequent elements
Data Structure - First Step

- Hashing function $s : O \rightarrow \{+1, -1\}$
- Counter $c \in \mathbb{Z}$
- Adding $q \in S$: $c += s[q]$
- Estimating $q \in O$: return $c \cdot s[q]$

Properties:

- $\mathbb{E}[c \cdot s[q]] = n_q$
- Variance extremely large
- Strong effect of frequent elements
Data Structure - First Step

- Hashing function \( s : O \rightarrow \{+1, -1\} \)
- Counter \( c \in \mathbb{Z} \)
- Adding \( q \in S \): \( c += s[q] \)
- Estimating \( q \in O \): return \( c \cdot s[q] \)

Properties:

- \( \mathbb{E}[c \cdot s[q]] = n_q \)
- Variance extremely large
- Strong effect of frequent elements
Data Structure - First Step

- Hashing function $s : O \rightarrow \{+1, -1\}$
- Counter $c \in \mathbb{Z}$
- Adding $q \in S$: $c += s[q]$
- Estimating $q \in O$: return $c \cdot s[q]$

Properties:

- $\mathbb{E}[c \cdot s[q]] = n_q$
- Variance extremely large
- Strong effect of frequent elements
Data Structure - Second Step

- Independent hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Counters $c_1, \ldots, c_t \in \mathbb{Z}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $c_i += s_i[q]$
- Estimating $q \in O$: return median$_i \{c_i \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[c_i \cdot s_i[q]] = n_q$
- Lower variance for mean or median! $\rightarrow$ Stochastic independence!
- Strong effect of frequent items
Data Structure - Second Step

- Independent hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Counters $c_1, \ldots, c_t \in \mathbb{Z}$
  - Adding $q \in S$: For $i = 1, \ldots, t$: $c_i += s_i[q]$
  - Estimating $q \in O$: return median$_i\{c_i \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[c_i \cdot s_i[q]] = n_q$
- Lower variance for mean or median! $\rightarrow$ Stochastic independence!
- Strong effect of frequent items
Data Structure - Second Step

- Independent hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Counters $c_1, \ldots, c_t \in \mathbb{Z}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $c_i += s_i[q]$
  - Estimating $q \in O$: return median$_i\{c_i \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[c_i \cdot s_i[q]] = n_q$
- Lower variance for mean or median! $\rightarrow$ Stochastic independence!
- Strong effect of frequent items
Data Structure - Second Step

- Independent hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}
- Counters $c_1, \ldots, c_t \in \mathbb{Z}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $c_i += s_i[q]$
- Estimating $q \in O$: return median$_i\{c_i \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[c_i \cdot s_i[q]] = n_q$
- Lower variance for mean or median! $\rightarrow$ Stochastic independence!
- Strong effect of frequent items
Data Structure - Second Step

- Independent hashing functions $s_1, \ldots, s_t : O \to \{+1, -1\}$
- Counters $c_1, \ldots, c_t \in \mathbb{Z}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $c_i += s_i[q]$  
- Estimating $q \in O$: return median$_i\{c_i \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[c_i \cdot s_i[q]] = n_q$
- Lower variance for mean or median! → Stochastic independence!
- Strong effect of frequent items
Data Structure - Second Step

- Independent hashing functions \( s_1, \ldots, s_t : O \rightarrow \{+1, -1\} \)
- Counters \( c_1, \ldots, c_t \in \mathbb{Z} \)
- Adding \( q \in S \): For \( i = 1, \ldots, t \): \( c_i += s_i[q] \)
- Estimating \( q \in O \): return median\( i \{ c_i \cdot s_i[q] \} \)

Properties:

- \( \mathbb{E}[c_i \cdot s_i[q]] = n_q \)
- Lower variance for mean or median! \( \rightarrow \) Stochastic independence!
- Strong effect of frequent items
Data Structure - Second Step

- Independent hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Counters $c_1, \ldots, c_t \in \mathbb{Z}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $c_i += s_i[q]$ 
- Estimating $q \in O$: return median$_i\{c_i \cdot s_i[q]\}$ 

Properties:

- $\mathbb{E}[c_i \cdot s_i[q]] = n_q$
- Lower variance for mean or median! $\rightarrow$ Stochastic independence!
- Strong effect of frequent items
Data Structure - Final Step

- Indep. hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Indep. hashing functions $h_1, \ldots, h_t : O \rightarrow \{1, \ldots, b\}$
- Array of Counters $c \in \mathbb{Z}^{t \times b}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $h_i[q] += s_i[q]$
- Estimating $q \in O$: return median$_i \{h_i[q] \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[h_i[q] \cdot s_i[q]] = n_q$
- Effect of frequent elements limited
Data Structure - Final Step

- Indep. hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Indep. hashing functions $h_1, \ldots, h_t : O \rightarrow \{1, \ldots, b\}$
- Array of Counters $c \in \mathbb{Z}^{t \times b}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $h_i[q] += s_i[q]$
- Estimating $q \in O$: return median$_i\{h_i[q] \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[h_i[q] \cdot s_i[q]] = n_q$
- Effect of frequent elements limited
Data Structure - Final Step

- Indep. hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Indep. hashing functions $h_1, \ldots, h_t : O \rightarrow \{1, \ldots, b\}$
- Array of Counters $c \in \mathbb{Z}^{t \times b}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $h_i[q] += s_i[q]$
- Estimating $q \in O$: return median$_i\{h_i[q] \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[h_i[q] \cdot s_i[q]] = n_q$
- Effect of frequent elements limited
Data Structure - Final Step

- Indep. hashing functions $s_1, \ldots, s_t : O \to \{+1, -1\}$
- Indep. hashing functions $h_1, \ldots, h_t : O \to \{1, \ldots, b\}$
- Array of Counters $c \in \mathbb{Z}^{t \times b}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $h_i[q] += s_i[q]$
- Estimating $q \in O$: return median$_i\{h_i[q] \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[h_i[q] \cdot s_i[q]] = n_q$
- Effect of frequent elements limited
Data Structure - Final Step

- Indep. hashing functions $s_1, \ldots, s_t : O \to \{+1, -1\}$
- Indep. hashing functions $h_1, \ldots, h_t : O \to \{1, \ldots, b\}$
- Array of Counters $c \in \mathbb{Z}^{t \times b}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $h_i[q] += s_i[q]$
- Estimating $q \in O$: return $\text{median}_i \{h_i[q] \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[h_i[q] \cdot s_i[q]] = n_q$
- Effect of frequent elements limited
Data Structure - Final Step

- Indep. hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Indep. hashing functions $h_1, \ldots, h_t : O \rightarrow \{1, \ldots, b\}$
- Array of Counters $c \in \mathbb{Z}^{t \times b}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $h_i[q] += s_i[q]$
- Estimating $q \in O$: return median$_i \{h_i[q] \cdot s_i[q]\}$

Properties:

- $\mathbb{E}[h_i[q] \cdot s_i[q]] = n_q$
- Effect of frequent elements limited
Data Structure - Final Step

- Indep. hashing functions $s_1, \ldots, s_t : O \rightarrow \{+1, -1\}$
- Indep. hashing functions $h_1, \ldots, h_t : O \rightarrow \{1, \ldots, b\}$
- Array of Counters $c \in \mathbb{Z}^{t \times b}$
- Adding $q \in S$: For $i = 1, \ldots, t$: $h_i[q] += s_i[q]$
- Estimating $q \in O$: return median$_i\{h_i[q] \cdot s_i[q]\}$

Properties:
- $\mathbb{E}[h_i[q] \cdot s_i[q]] = n_q$
- Effect of frequent elements limited
Definition (CountSketch)

Data structure containing these hashing functions, the array $c$ and the operations $Add$ and $Estimate$. 

→ Algorithm?
Definition (CountSketch)

Data structure containing these hashing functions, the array $c$ and the operations $Add$ and $Estimate$.

→ Algorithm?
Algorithm

- CountSketch \( C \) for estimation, with \( b, t \) chosen appropriately
- Heap \( H \) for top \( k \) elements (with counters)

For each \( q \) in data stream \( q_1, \ldots, q_n \):

1. Add(\( C, q \))
   - If \( q \in H \): Increment its count
2. else: If Estimate(\( C, q \)) greater than smallest element in \( H \), add \( q \) to \( H \) and remove smallest element
Algorithm

- CountSketch $C$ for estimation, with $b, t$ chosen appropriately
- Heap $H$ for top $k$ elements (with counters)

For each $q$ in data stream $q_1, \ldots, q_n$:

1. Add($C, q$)
   - If $q \in H$: Increment its count
2. else: If Estimate($C, q$) greater than smallest element in $H$, add $q$ to $H$ and remove smallest element
Algorithm

- CountSketch $C$ for estimation, with $b$, $t$ chosen appropriately
- Heap $H$ for top $k$ elements (with counters)

For each $q$ in data stream $q_1, \ldots, q_n$:

1. Add($C, q$)
   
   If $q \in H$: Increment its count

2. else: If Estimate($C, q$) greater than smallest element in $H$, add $q$ to $H$ and remove smallest element
Algorithm

- CountSketch $C$ for estimation, with $b, t$ chosen appropriately
- Heap $H$ for top $k$ elements (with counters)

For each $q$ in data stream $q_1, \ldots, q_n$:

1. Add($C, q$)
   - If $q \in H$: Increment its count
2. else: If Estimate($C, q$) greater than smallest element in $H$, add $q$ to $H$ and remove smallest element
Algorithm

- CountSketch $C$ for estimation, with $b, t$ chosen appropriately
- Heap $H$ for top $k$ elements (with counters)

For each $q$ in data stream $q_1, \ldots, q_n$:

1. Add$(C, q)$
   - If $q \in H$: Increment its count
2. else: If Estimate$(C, q)$ greater than smallest element in $H$, add $q$ to $H$ and remove smallest element
Analysis - Expectation and Variance
**Expectation**

**Lemma**

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q \]

**Proof.**

\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_q's_i[q'] \]

\[
\mathbb{E}[h_i[q]s_i[q]]
= n_q + \mathbb{E} \left[ s_i[q] \sum_{q' \in A_i[q]} n_q's_i[q'] \right]
= n_q + \sum_{q' \in A_i[q]} \mathbb{E}[n_q's_i[q]] \mathbb{E}[s_i[q']] = 0
\]
Expectation

Lemma

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q \]

Proof.

\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'}s_i[q'] \]

\[
\begin{align*}
\mathbb{E}[h_i[q]s_i[q]] &= n_q + \mathbb{E} \left[ s_i[q] \sum_{q' \in A_i[q]} n_{q'}s_i[q'] \right] \\
&= n_q + \sum_{q' \in A_i[q]} \mathbb{E}[n_{q'}s_i[q]] \mathbb{E}[s_i[q']] \\
&= n_q
\end{align*}
\]
Expectation

Lemma

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q \]

Proof.

\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \]

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q + \mathbb{E} \left[ s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \right] \]

\[ \text{pairwise independence} \]

\[ \text{=} \]

\[ n_q + \sum_{q' \in A_i[q]} \mathbb{E}[n_{q'} s_i[q]] \mathbb{E}[s_i[q']] = 0 \]

\[ = n_q \]
Expectation

Lemma

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q \]

Proof.

\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'}s_i[q'] \]

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q + \mathbb{E} \left[ s_i[q] \sum_{q' \in A_i[q]} n_{q'}s_i[q'] \right] \]

pairwise independence

\[ = n_q + \sum_{q' \in A_i[q]} \mathbb{E}[n_{q'}s_i[q]] \mathbb{E}[s_i[q']] \]

\[ = 0 \]

\[ = n_q \]
Expectation

**Lemma**

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q \]

**Proof.**

\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'}s_i[q'] \]

\[
\mathbb{E}[h_i[q]s_i[q]] = n_q + \mathbb{E}\left[ s_i[q] \sum_{q' \in A_i[q]} n_{q'}s_i[q'] \right]
\]

pairwise independence

\[
= n_q + \sum_{q' \in A_i[q]} \mathbb{E}[n_{q'}s_i[q]]\mathbb{E}[s_i[q']] = 0
\]

\[ = n_q \]
Variance

**Lemma**

\[
\text{Var}(h_i[q]s_i[q]) = v_i[q] := \sum_{q' \in A_i[q]} n_{q'}^2
\]
\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \]

\[
\text{Var}[h_i[q]s_i[q]] = \mathbb{E}[(h_i[q]s_i[q] - n_q)^2]
\]

\[
= \mathbb{E} \left[ \left( s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \right)^2 \right]
\]

\[
= s_i[q]^2 \sum_{q', q'' \in A_i[q]} \mathbb{E}[n_{q'} s_i[q'] n_{q''} s_i[q'']] 
\]

pairwise independence

\[
= \sum_{q' \in A_i[q]} \mathbb{E} \left[ n_{q'}^2 s_i[q']^2 \right] 
\]

\[
+ \sum_{q' \neq q'' \in A_i[q]} \mathbb{E}[n_{q'} n_{q''} s_i[q']] \mathbb{E}[s_i[q'']] = 0
\]

\[
= \sum_{q' \in A_i[q]} n_{q'}^2 = v_i[q]
\]
\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \]

\[ \text{Var}[h_i[q]s_i[q]] = \mathbb{E}[(h_i[q]s_i[q] - n_q)^2] \]

\[ = \mathbb{E} \left( \left( s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \right)^2 \right) \]

\[ = s_i[q]^2 \sum_{q', q'' \in A_i[q]} \mathbb{E}[n_{q'} s_i[q'] n_{q''} s_i[q'']] \]

\(=1\) pairwise

\[ = \sum_{q' \in A_i[q]} \mathbb{E} \left[ n_{q'}^2 s_i[q']^2 \right] \]

\(=1\) independence

\[ + \sum_{q' \neq q'' \in A_i[q]} \mathbb{E}[n_{q'} n_{q''} s_i[q']] \mathbb{E}[s_i[q'']] \]

\(=0\)

\[ = \sum_{q' \in A_i[q]} n_{q'}^2 = v_i[q] \]
\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \]

\[
\operatorname{Var}[h_i[q]s_i[q]] = \mathbb{E}[(h_i[q]s_i[q] - n_q)^2] \\
= \mathbb{E} \left[ \left( s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \right)^2 \right] \\
= s_i[q]^2 \sum_{q', q'' \in A_i[q]} \mathbb{E}[n_{q'} s_i[q'] n_{q''} s_i[q'']] \\
\overset{\text{pairwise independence}}{=} \sum_{q' \in A_i[q]} \mathbb{E} \left[ n_{q'}^2 s_i[q']^2 \right] = 1 \\
+ \sum_{q' \neq q'' \in A_i[q]} \mathbb{E}[n_{q'} n_{q''} s_i[q']] \mathbb{E}[s_i[q'']] = 0 \\
= \sum_{q' \in A_i[q]} n_{q'}^2 = v_i[q] \]
\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \]

\[
\begin{align*}
\text{Var}[h_i[q]s_i[q]] &= \mathbb{E}[(h_i[q]s_i[q] - n_q)^2] \\
&= \mathbb{E} \left[ \left( s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \right)^2 \right] \\
&= s_i[q]^2 \sum_{q', q'' \in A_i[q]} \mathbb{E}[n_{q'} s_i[q'] n_{q''} s_i[q'']] \\
&\underset{\text{pairwise independence}}{=} \sum_{q' \in A_i[q]} \mathbb{E} \left[ n_{q'}^2 s_i[q']^2 \right] \\
&\underset{\text{independence}}{=} \sum_{q' \neq q'' \in A_i[q]} \mathbb{E}[n_{q'} n_{q''} s_i[q'] \mathbb{E}[s_i[q'']]] \\
&= \sum_{q' \in A_i[q]} n_{q'}^2 = v_i[q]
\end{align*}
\]
\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \]

\[
\text{Var}[h_i[q]s_i[q]] = \mathbb{E}[(h_i[q]s_i[q] - n_q)^2]
\]

\[
= \mathbb{E} \left[ \left( s_i[q] \sum_{q' \in A_i[q]} n_{q'} s_i[q'] \right)^2 \right]
\]

\[
= s_i[q]^2 \sum_{q' \in A_i[q]} \mathbb{E}[n_{q'} s_i[q'] n_{q''} s_i[q'']] \\
= \sum_{q' \in A_i[q]} \mathbb{E} \left[ n_{q'}^2 s_i[q']^2 \right] \\
+ \sum_{q' \neq q'' \in A_i[q]} \mathbb{E}[n_{q'} n_{q''} s_i[q']] \mathbb{E}[s_i[q'']] = 0
\]

\[
= \sum_{q' \in A_i[q]} n_{q'}^2 = v_i[q]
\]
\[ h_i[q]s_i[q] = n_q + s_i[q] \sum_{q' \in A_i[q]} n_q's_i[q'] \]

\[
\text{Var}[h_i[q]s_i[q]] = \mathbb{E}[(h_i[q]s_i[q] - n_q)^2]
\]
\[
= \mathbb{E}\left[\left(s_i[q] \sum_{q' \in A_i[q]} n_q's_i[q']\right)^2\right]
\]
\[
= s_i[q]^2 \sum_{q',q'' \in A_i[q]} \mathbb{E}[n_q's_i[q']n_q''s_i[q'']]
\]
\[
= \sum_{q' \in A_i[q]} \mathbb{E}\left[n_q'^2s_i[q']^2\right]_{=1}
\]
\[
+ \sum_{q' \neq q'' \in A_i[q]} \mathbb{E}[n_q'n_{q''}s_i[q']][\mathbb{E}[s_i[q'']]]_{=0}
\]
\[
= \sum_{q' \in A_i[q]} n_{q'}^2 = v_i[q]
\]
Summary

Lemma

\[ \mathbb{E}[h_i[q]s_i[q]] = n_q \]

Lemma

\[ \text{Var}(h_i[q]s_i[q]) = v_i[q] := \sum_{q' \in A_i[q]} n_{q'}^2 \]
Analysis - Deviation Bound
## Deviation Bound

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. SmallDeviation</td>
<td>$</td>
</tr>
<tr>
<td>ii. NoCollisions</td>
<td>$A_i[q] = A_i^{&gt;k}[q]$</td>
</tr>
<tr>
<td>iii. SmallVariance</td>
<td>$v_i^{&gt;k}[q] \leq 8 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2$</td>
</tr>
</tbody>
</table>

### Lemma

If all events are fulfilled, it holds

$$|h_i[q]s_i[q] - n_q| \leq 8\gamma$$

where

$$\gamma := \sqrt{\frac{1}{b} \sum_{j=k+1}^{m} n_j^2}$$
Deviation Bound

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. SmallDeviation</td>
<td>$</td>
</tr>
<tr>
<td>ii. NoCollisions</td>
<td>$A_i[q] = A_i^{\geq k}[q]$</td>
</tr>
<tr>
<td>iii. SmallVariance</td>
<td>$v_i^{\geq k}[q] \leq \frac{8}{b} \sum_{j=k+1}^{m} n_j^2$</td>
</tr>
</tbody>
</table>

Lemma

If all events are fulfilled, it holds

$$|h_i[q]s_i[q] - n_q| \leq 8\gamma$$

where

$$\gamma := \sqrt{\frac{1}{b} \sum_{j=k+1}^{m} n_j^2}$$
Deviation Bound

|   | SmallDeviation | $|h_i[q]s_i[q] - n_q|^2 \leq 8 \text{Var}[h_i[q]s_i[q]]$ |
|---|----------------|--------------------------------------------------|
| i. | NoCollisions   | $A_i[q] = A_i^{>k}[q]$                           |
| ii.|                |                                                  |
| iii.| SmallVariance | $v_i^{>k}[q] \leq 8 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2$ |

Proof.

\[
|h_i[q]s_i[q] - n_q|^2 \quad \overset{i.}{\leq} \quad 8 \text{Var}[h_i[q]s_i[q]] \\
= 8v_i[q] \\
\overset{ii.}{=} 8v_i^{>k}[q] \\
\overset{iii.}{=} 8^2 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2 = (8\gamma)^2 \\
= \gamma^2 \]


Deviation Bound

| i. | SmallDeviation | \(|h_i[q]s_i[q] - n_q|^2 \leq 8 \text{Var}[h_i[q]s_i[q]]|
| ii. | NoCollisions | \(A_i[q] = A_i^{\geq k}[q]|
| iii. | SmallVariance | \(v_i^{\geq k}[q] \leq 8 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2|

Proof.

\[
|h_i[q]s_i[q] - n_q|^2 \leq 8 \text{Var}[h_i[q]s_i[q]] = 8v_i[q] = 8v_i^{\geq k}[q] = 8^2 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2 = (8\gamma)^2 = \gamma^2
\]
## Deviation Bound

|   | SmallDeviation | \(|h_i[q]s_i[q] - n_q|^2 \leq 8 \text{Var}[h_i[q]s_i[q]]\) |
|---|----------------|----------------------------------------------------------|
| i. |                |                                                          |
| ii. | NoCollisions   | \(A_i[q] = A_i^\geq k[q]\)                             |
| iii. | SmallVariance | \(\nu_i^\geq k[q] \leq 8 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2\) |

### Proof.

\[
|h_i[q]s_i[q] - n_q|^2 \leq 8 \text{Var}[h_i[q]s_i[q]]
= 8\nu_i[q]
\]

\[
\leq 8\nu_i^\geq k[q]
\]

\[
= 8^2 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2
= (8\gamma)^2
\]

\[
= \gamma^2
\]
## Deviation Bound

| i.  | SmallDeviation | $|h_i[q]s_i[q] - n_q|^2 \leq 8 \text{Var}[h_i[q]s_i[q]]$ |
|-----|----------------|--------------------------------------------------------------------------------|
| ii. | NoCollisions   | $A_i[q] = A_i^{>k}[q]$                                                          |
| iii. | SmallVariance | $\nu_i^{>k}[q] \leq 8^{1/b} \sum_{j=k+1}^{m} n_j^2$                             |

## Proof.

\[
|h_i[q]s_i[q] - n_q|^2 \leq 8 \text{Var}[h_i[q]s_i[q]] = 8\nu_i[q] = 8\nu_i^{>k}[q] \\
\equiv 8^2 \frac{1}{b} \sum_{j=k+1}^{m} n_j^2 = (8\gamma)^2 = \gamma^2
\]


**Event 1**

**Lemma**

\[ \mathbb{P}[\text{SmallDeviation}] \geq 1 - \frac{1}{8} \]

**Proof.**

\[
\begin{align*}
\mathbb{P}[\neg \text{SmallDeviation}] &= \mathbb{P}[|h_i[q]s_i[q] - n_q|^2 > 8 \text{Var}[h_i[q]s_i[q]]] \\
&\leq \mathbb{E}[|h_i[q]s_i[q] - n_q|^2] / 8 \text{Var}[h_i[q]s_i[q]] \\
&= \frac{1}{8}
\end{align*}
\]
Event 1

Lemma

\[ \mathbb{P}[\text{SmallDeviation}] \geq 1 - \frac{1}{8} \]

Proof.

\[ \mathbb{P}[\neg \text{SmallDeviation}] = \mathbb{P}[|h_1[q]s_1[q] - n_q|^2 > 8 \text{Var}[h_1[q]s_1[q]]] \]

\[ \leq \frac{\mathbb{E}[|h_1[q]s_1[q] - n_q|^2]}{8 \text{Var}[h_1[q]s_1[q]]} = \frac{1}{8} \]
Event 1

Lemma

\[ \mathbb{P}[\text{SmallDeviation}] \geq 1 - \frac{1}{8} \]

Proof.

\[ \mathbb{P}[\neg \text{SmallDeviation}] = \mathbb{P}[|h_i[q]s_i[q] - n_q|^2 > 8 \text{Var}[h_i[q]s_i[q]]] \]

Markov

\[ \leq \frac{\mathbb{E}[|h_i[q]s_i[q] - n_q|^2]}{8 \text{Var}[h_i[q]s_i[q]]} \]

\[ = \frac{1}{8} \]
Event 1

Lemma

\[ P[\text{SmallDeviation}] \geq 1 - \frac{1}{8} \]

Proof.

\[
P[\neg \text{SmallDeviation}] = P[|h_i[q]s_i[q] - n_q|^2 > 8 \text{Var}[h_i[q]s_i[q]]] \\
\leq \frac{E[|h_i[q]s_i[q] - n_q|^2]}{8 \text{Var}[h_i[q]s_i[q]]} \\
= \frac{1}{8}
\]
**Event 2**

**Lemma**

*Choose* $b \geq 8k$. *Then*

$$P[\text{NoCollisions}] \geq 1 - \frac{1}{8}$$

**Proof.**

For suitable hashing functions $h_i$, it holds

$$P[o_i \not\in A_i[q]] = 1 - \frac{1}{b}.$$ 

Therefore, by the union bound

$$P[\text{NoCollisions}] \geq 1 - k\frac{1}{b} \geq 1 - \frac{1}{8}.$$
Lemma

Choose $b \geq 8k$. Then

$$\mathbb{P}[\text{NoCollisions}] \geq 1 - \frac{1}{8}$$

Proof.

For suitable hashing functions $h_i$, it holds

$$\mathbb{P}[o_i \not\in A_i[q]] = 1 - \frac{1}{b}.$$ 

Therefore, by the union bound

$$\mathbb{P}[\text{NoCollisions}] \geq 1 - k \cdot \frac{1}{b} \geq 1 - \frac{1}{8}.$$
**Lemma**

Choose \( b \geq 8k \). Then

\[
P[\text{NoCollisions}] \geq 1 - \frac{1}{8}
\]

**Proof.**

For suitable hashing functions \( h_i \), it holds

\[
P[o_i \not\in A_i[q]] = 1 - \frac{1}{b}.
\]

Therefore, by the union bound

\[
P[\text{NoCollisions}] \geq 1 - k \frac{1}{b} \geq 1 - \frac{1}{8}.
\]
**Lemma**

\[ \mathbb{P}[\text{SmallVariance}] \geq 1 - \frac{1}{8} \]

**Proof.**

\[
\begin{align*}
\mathbb{P}[\neg \text{SmallVariance}] &= \mathbb{P}[v_i^k[q] > 8\gamma^2] \\
&\leq \mathbb{E}[v_i^k[q]] \\
&\leq \frac{1}{8 \gamma^2} \\
&= \frac{1}{8}
\end{align*}
\]
Event 3

Lemma

\[ P[\text{SmallVariance}] \geq 1 - \frac{1}{8} \]

Proof.

\[ P[\neg \text{SmallVariance}] = P[v_i > k[q] > 8\gamma^2] \]

\[ \leq E[v_i > k[q]] \]

\[ = \frac{1}{8} \]

Markov
Event 3

Lemma

\[ P[\text{SmallVariance}] \geq 1 - \frac{1}{8} \]

Proof.

\[ P[\neg \text{SmallVariance}] = P[v_i^{>k}[q] > 8\gamma^2] \]

Markov

\[ \leq \frac{E[v_i^{>k}[q]]}{8\gamma^2} = \frac{1}{8} \]
Event 3

Lemma

$$\mathbb{P}[\text{SmallVariance}] \geq 1 - \frac{1}{8}$$

Proof.

$$\mathbb{P}[\neg \text{SmallVariance}] = \mathbb{P}[v_i^{>k}[q] > 8\gamma^2]$$

Markov

$$\leq \frac{\mathbb{E}[v_i^{>k}[q]]}{8\gamma^2}$$

$$= \frac{1}{8}$$
Deviation Bound

Choose $b \geq 8k$. All events occur with probability $\geq 1 - \frac{1}{8}$

$\Rightarrow \mathbb{P}[\text{SmallDeviation} \land \text{NoCollisions} \land \text{SmallVariance}] \geq \frac{5}{8}$

$\Rightarrow \mathbb{P}[|h_i[q]s_i[q] - n_q| \leq 8\gamma] \geq \frac{5}{8}$

$\Rightarrow$ Bound is expected to hold in $\frac{5}{8} t$ rows
Deviation Bound

Choose $b \geq 8k$. All events occur with probability $\geq 1 - \frac{1}{8}$

$\Rightarrow \quad P[\text{SmallDeviation} \land \text{NoCollisions} \land \text{SmallVariance}] \geq \frac{5}{8}$

$\Rightarrow \quad P[|h_i[q]s_i[q] - n_q| \leq 8\gamma] \geq \frac{5}{8}$

$\Rightarrow \quad \text{Bound is expected to hold in } \frac{5}{8}t \text{ rows}$
Deviation Bound

Choose $b \geq 8k$. All events occur with probability $\geq 1 - \frac{1}{8}$

$$\Rightarrow \quad P[\text{SmallDeviation} \land \text{NoCollisions} \land \text{SmallVariance}] \geq \frac{5}{8}$$

$$\Rightarrow \quad P[|h_i[q]s_i[q] - n_q| \leq 8\gamma] \geq \frac{5}{8}$$

$$\Rightarrow \quad \text{Bound is expected to hold in } \frac{5}{8}t \text{ rows}$$
Deviation Bound

Choose $b \geq 8k$. All events occur with probability $\geq 1 - \frac{1}{8}$

$\Rightarrow \mathbb{P}[\text{SmallDeviation} \land \text{NoCollisions} \land \text{SmallVariance}] \geq \frac{5}{8}$

$\Rightarrow \mathbb{P}[|h[q]s[q] - n_q| \leq 8\gamma] \geq \frac{5}{8}$

$\Rightarrow$ Bound is expected to hold in $\frac{5}{8}t$ rows
Bound is expected to hold in $\frac{5}{8}t$ rows

Choose $t = \log \left( \frac{n}{\delta} \right)$

\[ \text{Chernoff} \quad \Rightarrow \quad \text{Number of these is at least } \frac{t}{2} \text{ with probability } 1 - \frac{\delta}{n} \]

\[ \Rightarrow \quad \text{Bound holds for median with that probability} \]
Deviation Bound

Bound is expected to hold in $\frac{5}{8} t$ rows

Choose $t = \log \left( \frac{n}{\delta} \right)$

$\Rightarrow$ Number of these is at least $\frac{t}{2}$ with probability $1 - \frac{\delta}{n}$

$\Rightarrow$ Bound holds for median with that probability
Deviation Bound

Bound is expected to hold in $\frac{5}{8} t$ rows

Choose $t = \log \left( \frac{n}{\delta} \right)$

\[ \text{Chernoff} \Rightarrow \text{Number of these is at least } \frac{t}{2} \text{ with probability } 1 - \frac{\delta}{n} \]

$\Rightarrow$ Bound holds for median with that probability
Deviation Bound

Bound is expected to hold in $\frac{5}{8} t$ rows

Choose $t = \log \left( \frac{n}{\delta} \right)$

Chernoff $\Rightarrow$ Number of these is at least $t / 2$ with probability $1 - \frac{\delta}{n}$

$\Rightarrow$ Bound holds for median with that probability
Choose $t = \log \left( \frac{n}{\delta} \right)$, $b \geq 8k$

$| \text{median}_i \{ h_i[q]s_i[q] \} - n_q | \leq 8\gamma$ with probability $\geq 1 - \frac{\delta}{n}$
Extended Applications
Parallelisation

- CountSketch additive $\rightarrow$ Parallel counting
  - MapReduce scheme possible
  - Only little data transfer necessary
Parallelisation

- CountSketch additive → Parallel counting
- MapReduce scheme possible
- Only little data transfer necessary
Parallelisation

- CountSketch additive $\rightarrow$ Parallel counting
- MapReduce scheme possible
- Only little data transfer necessary
Exact Results

- FindApproxTop output: 
  $k$ elements with frequencies

  \[ n_i > (1 - \varepsilon)n_k \]

- Increase heap size to certainly include $k$ most frequent elements (distribution dependent!)
- Exact count for heap elements in second pass
Exact Results

- FindApproxTop output: $k$ elements with frequencies

$$n_i > (1 - \varepsilon)n_k$$

- Increase heap size to certainly include $k$ most frequent elements ($\rightarrow$ distribution dependent!)
- Exact count for heap elements in second pass
Exact Results

- FindApproxTop output:
  - $k$ elements with frequencies

  $$n_i > (1 - \varepsilon)n_k$$

- Increase heap size to certainly include $k$ most frequent elements (distribution dependent!)
- Exact count for heap elements in second pass
Frequency Change Detection

- **Data streams** $S_1, S_2$, only one *CountSketch*
- **First pass (Estimating):**
  - For $q \in S_1$:
    $$ h_i[q] += s_i[q] $$
  - For $q \in S_2$:
    $$ h_i[q] -= s_i[q] $$
- **Second pass (Counting candidates):**
  - Track elements with largest absolute estimates
  - Maintain exact counts
Frequency Change Detection

- Data streams $S_1, S_2$, only one CountSketch
- First pass (Estimating):
  - For $q \in S_1$:
    \[ h_i[q] += s_i[q] \]
  - For $q \in S_2$:
    \[ h_i[q] -= s_i[q] \]
- Second pass (Counting candidates):
  - Track elements with largest absolute estimates
  - Maintain exact counts
Frequency Change Detection

- Data streams $S_1, S_2$, only one CountSketch
- First pass (Estimating):
  - For $q \in S_1$:
    $$h_i[q] += s_i[q]$$
  - For $q \in S_2$:
    $$h_i[q] -= s_i[q]$$
- Second pass (Counting candidates):
  - Track elements with largest absolute estimates
  - Maintain exact counts
Conclusion
Conclusion

- Fast algorithm with logarithmic space bound
- Only one pass required
- Using probabilistic estimates
- Easy to implement and parallelise
- Further applications based on CountSketch possible
Conclusion

- Fast algorithm with logarithmic space bound
- Only one pass required
  - Using probabilistic estimates
  - Easy to implement and parallelise
- Further applications based on CountSketch possible
Conclusion

- Fast algorithm with logarithmic space bound
- Only one pass required
- Using probabilistic estimates
- Easy to implement and parallelise
- Further applications based on CountSketch possible
Conclusion

- Fast algorithm with logarithmic space bound
- Only one pass required
- Using probabilistic estimates
- Easy to implement and parallelise
- Further applications based on CountSketch possible
Conclusion

- Fast algorithm with logarithmic space bound
- Only one pass required
- Using probabilistic estimates
- Easy to implement and parallelise
- Further applications based on CountSketch possible
∃{?_1, ?_2, \ldots}?