Lecture 3: CountSketch, Graph sketching, $\ell_0$ Samplers

Michael Kapralov

EPFL

May 26, 2017
CountSketch (recap and proofs)

Graph streaming

Connectivity via sketching

Designing \( \ell_0 \) samplers
- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- Designing $\ell_0$ samplers
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!

```
1 2 3 4 5 6 7 8 9 10
3 4
```
Heavy hitters problem

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

  Assume $N$ is known

- Output $k$ most frequent items

  (Heavy hitters)

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

  Much better than storing all items!
Heavy hitters problem

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

  Assume $N$ is known

- Output $k$ most frequent items

  (Heavy hitters)

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

  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!

```
3 4 6 3 2
```
Heavy hitters problem

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

  Assume \( N \) is known

- Output \( k \) most frequent items
  
  (Heavy hitters)

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

  Much better than storing all items!

3 4 6 3 2 10
Heavy hitters problem

- Single pass over the data: \(i_1, i_2, \ldots, i_N\)
  
  Assume \(N\) is known

- Output \(k\) most frequent items
  
  (Heavy hitters)

- Small storage: will get \(O(k \log N)\)
  
  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!
Heavy hitters problem

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

Assume \( N \) is known

- Output \( k \) most frequent items

(Heavy hitters)

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

Much better than storing all items!
Heavy hitters problem

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

  Assume $N$ is known

- Output $k$ most frequent items

  (Heavy hitters)

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

  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  - Assume $N$ is known

- Output $k$ most frequent items
  - (Heavy hitters)

- Small storage: will get $O(k \log N)$
  - Much better than storing all items!

```
3 4 6 3 2 10 3 1 3 1 2 4 5 5 5 9 8 7 4 4 2 2 3 3
```
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!
Heavy hitters problem

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

  Assume \( N \) is known

- Output \( k \) most frequent items

  (Heavy hitters)

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

  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

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

  Much better than storing all items!

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items

  (Heavy hitters)

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

  Much better than storing all items!

```
3 4 6 3 2 10 3 1 3 1 2 2 5 5 5
```
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!
Heavy hitters problem

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

  Assume $N$ is known

- Output $k$ most frequent items

  (Heavy hitters)

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

  Much better than storing all items!

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

```
1 2 3 4 5 6 7 8 9 10
```
Heavy hitters problem

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

  Assume $N$ is known

- Output $k$ most frequent items

  (Heavy hitters)

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

  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!
Heavy hitters problem

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

  Assume $N$ is known

- Output $k$ most frequent items

  (Heavy hitters)

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

  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

- Small storage: will get $O(k \log N)$
  
  Much better than storing all items!

```
head
  3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2
tail
  2 3 1
```
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  - Assume $N$ is known

- Output $k$ most frequent items
  - (Heavy hitters)

- Small storage: will get $O(k \log N)$
  - Much better than storing all items!

```
3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2 2
```
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  - Assume $N$ is known
- Output $k$ most frequent items
  - (Heavy hitters)
- Small storage: will get $O(k \log N)$
  - Much better than storing all items!
Heavy hitters problem

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

  Assume \( N \) is known

- Output \( k \) most frequent items

  (Heavy hitters)

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

  Much better than storing all items!
Heavy hitters problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output $k$ most frequent items
  
  (Heavy hitters)

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

  Much better than storing all items!
Main primitive: APPROXPOINTQUERY in small space

Observe a stream of updates, maintain small space data structure

**Task:** after observing the stream, given \( i \in \{1, 2, \ldots, m\} \), compute estimate \( \hat{f}_i \) of \( f_i \)
Main primitive: APPR OX POINT QUERY in small space

Observe a stream of updates, maintain small space data structure

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

To be specified:

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

Choose

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

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

Choose

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

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

The algorithm runs $t$ independent copies of basic estimate:

\[
\text{UPDATE}(C, i)
\]

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

\[
\text{ESTIMATE}(C, i)
\]

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

Choose

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

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

The algorithm runs $t$ independent copies of basic estimate:

$\text{UPDATE}(C, i)$

\begin{itemize}
  \item \textbf{for} $r \in [1 : t]$
  \item $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
\end{itemize}

$\text{ESTIMATE}(C, i)$

\begin{itemize}
  \item return median$_r \{C[r, h_r(i)] \cdot s_r(i)\}$
\end{itemize}

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

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

By basic estimate analysis for every $r \in [1 : t]$

\[ \mathbb{E}[C[r, h_r(i)] \cdot s_r(i)] = f_i \]
\textbf{UPDATE}(C, i) \\
\textbf{for} \ r \in [1 : t] \\
\hspace{1em} C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i) \\
\textbf{end for} \\

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

By basic estimate analysis for every \( r \in [1 : t] \)

\[ E[C[r, h_r(i)] \cdot s_r(i)] = f_i \]

and

\[ E_s[(C[r, h_r(i)] \cdot s_r(i) - f_i)^2] = \sum_{j \neq i: h_r(j) = h_r(i)} f_j^2 \]
UPDATE($C, i$)
for $r \in [1 : t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

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

By basic estimate analysis for every $r \in [1 : t]$

$E[C[r, h_r(i)] \cdot s_r(i)] = f_i$

and

$E_s[(C[r, h_r(i)] \cdot s_r(i) - f_i)^2] = \sum_{j \neq i : h_r(j) = h_r(i)} f_j^2$

How large can the variance be? Can be be be reduced by making number of buckets $b$ large?
UPDATE(C, i)
for r ∈ [1 : t]
  C[r, h_r(i)] ← C[r, h_r(i)] + s_r(i)
end for

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

By basic estimate analysis for every r ∈ [1 : t]

\[ \mathbb{E}[C[r, h_r(i)] \cdot s_r(i)] = f_i \]

and

\[ \mathbb{E}_s[(C[r, h_r(i)] \cdot s_r(i) - f_i)^2] = \sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 \]

How large can the variance be? Can be be reduced by making number of buckets \( b \) large?

**YES:** hashing into a sufficiently large number of buckets reduces estimation error to below \( \varepsilon \cdot f_k \)
**UPDATE(C, i)**

for $r \in [1 : t]$

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

end for

**ESTIMATE(C, i)**

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

By basic estimate analysis for every $r \in [1 : t]$

\[ \mathbb{E}[C[r, h_r(i)] \cdot s_r(i)] = f_i \]

and

\[ \mathbb{E}_s[(C[r, h_r(i)] \cdot s_r(i) - f_i)^2] = \sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 \]

How large can the variance be? Can be be reduced by making number of buckets $b$ large?

**YES:** hashing into a sufficiently large number of buckets reduces estimation error to below $\varepsilon \cdot f_k$

$O(\log N)$ repetitions ensure estimates are correct for all $i$ with high probability.
\textbf{UPDATE}(C, i) \\
for \( r \in [1 : t] \) \\
\quad \quad C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i) \\
end for \\
\textbf{ESTIMATE}(C, i) \\
\textbf{return} \quad \text{median}_r \{ C[r, h_r(i)] \cdot s_r(i) \}
UPDATE(C, i)
for $r \in [1 : t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

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

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

$$|\text{median}_r \{ C[r, h_r(i)] \cdot s_r(i) \} - f_i(p)| \leq \varepsilon f_k$$

at every point $p \in [1 : N]$ in the stream.

($f_i(p)$ is the frequency of $i$ up to position $p$)
UPDATE(C, i)
for r ∈ [1 : t]
    C[r, h_r(i)] ← C[r, h_r(i)] + s_r(i)
end for

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

Lemma
If \( b \geq 8 \max \left\{ k, \frac{32 \sum_{j \in TAIL} f_j^2}{(\varepsilon f_k)^2} \right\} \) and \( t = O(\log N) \), then for every \( i \in [m] \)
\[ |\text{median}_r \{ C[r, h_r(i)] \cdot s_r(i) \} - f_i(p)| \leq \varepsilon f_k \]

due to every point \( p \in [1 : N] \) in the stream.

\((f_i(p) \text{ is the frequency of } i \text{ up to position } p)\)

Space complexity is \( O(b \log N) \)
UPDATE(C, i)
for r ∈ [1 : t]
    C[r, h_r(i)] ← C[r, h_r(i)] + s_r(i)
end for

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

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

\[ |\text{median}_r \{ C[r, h_r(i)] \cdot s_r(i) \} - f_i(p)| \leq \varepsilon f_k \]

at every point \( p \in [1 : N] \) in the stream.

\( (f_i(p) \) is the frequency of \( i \) up to position \( p) \)

Space complexity is \( O(b \log N) \)

**How large is \( b \)?**
Space complexity

Set \( b = 8 \max \left\{ k, \frac{32 \sum_{j \in \text{T A I L}} f_j^2}{(\varepsilon f_k)^2} \right\} \)

Note that \( b = O\left( \frac{k}{\varepsilon^2} \right) \) if \( \frac{1}{k} \sum_{j \in \text{T A I L}} f_j^2 = O\left( f_k^2 \right) \)

In practice, choose \( b \) subject to space constraints, detect elements with counts above \( O\left( \varepsilon \sqrt{\frac{1}{k} \sum_{j \in \text{T A I L}} f_j^2} \right) \)
Space complexity

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(In fact also every \( i \) with \( f_i \geq (1 - \varepsilon)f_k \) is reported)

\[ \text{APPROXPOINTQUERY}(S, i, \varepsilon) : \text{returns } \hat{f}_i \in [f_i - \varepsilon f_k, f_i + \varepsilon f_k] \]

Find head items if they contribute the bulk of the stream in \( \ell_2 \) sense
CountSketch: proof details
UPDATE(C, i)
for $r \in [1 : t]$
    $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

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

By basic estimate analysis for every $r \in [1 : t]$

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

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

By basic estimate analysis for every $r \in [1 : t]$

\[
E[C[r, h_r(i)] \cdot s_r(i)] = f_i
\]

and

\[
E_s[(C[r, h_r(i)] \cdot s_r(i) - f_i)^2] = \sum_{j \neq i : h_r(j) = h_r(i)} f_j^2
\]
UPDATE(C, i)
for \( r \in [1 : t] \)
    \( C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i) \)
end for

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

By basic estimate analysis for every \( r \in [1 : t] \)

\[
E[C[r, h_r(i)] \cdot s_r(i)] = f_i
\]

and

\[
E_s[(C[r, h_r(i)] \cdot s_r(i) - f_i)^2] = \sum_{j \neq i : h_r(j) = h_r(i)} f_j^2
\]

How large can the variance be? Does it reduce by about a factor of \( b \)?
UPDATE(C, i)
for $r \in [1 : t]$
\[ C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i) \]
end for

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

By basic estimate analysis for every $r \in [1 : t]$
\[ \mathbb{E}[C[r, h_r(i)] \cdot s_r(i)] = f_i \]

and
\[ \mathbb{E}_s[(C[r, h_r(i)] \cdot s_r(i) - f_i)^2] = \sum_{j \neq i: h_r(j) = h_r(i)} f_j^2 \]

How large can the variance be? Does it reduce by about a factor of $b$?

Consider contribution of head and tail items separately:
\[ \sum_{j \neq i: h_r(j) = h_r(i)} f_j^2 = \sum_{j \in \text{HEAD}, j \neq i: h_r(j) = h_r(i)} f_j^2 + \sum_{j \in \text{TAIL}, j \neq i: h_r(j) = h_r(i)} f_j^2 \]
Consider contribution of head and tail items separately:

$$\sum_{j \neq i: h_r(j) = h_r(i)} f_j^2 = \sum_{j \in \text{HEAD}, j \neq i} f_j^2 + \sum_{j \in \text{TAIL}, j \neq i} f_j^2$$
Consider contribution of head and tail items separately:

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

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

- **NO-COLLISIONS}_r(i) \) – \( i \) does not collide with any of the head items under hashing \( r \)
Consider contribution of head and tail items separately:

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

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

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

- **SMALL-VARIANCE\(_r\)(i)** – \( i \) does not collide with too many of tail items under hashing \( r \)
Consider contribution of head and tail items separately:

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

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

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

- **SMALL-VARIANCE\(_r\)(i)** – \( i \) does not collide with too many of tail items under hashing \( r \)

- **SMALL-DEVIATION\(_r\)(i)** – success event from basic analysis
Consider contribution of head and tail items separately:

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

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

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

- **SMALL-VARIANCE}_r(i) – \( i \) does not collide with too many of tail items under hashing \( r \)

- **SMALL-DEVIATION}_r(i) – success event from basic analysis

Show that all three events hold simultaneously with probability strictly bigger than \( 1/2 \) – so median gives good estimate
(No) collisions with head items

\[
\text{NO-COLLISIONS}_r(i) := \text{event that } \\
\{j \in HEAD \setminus i : h_r(j) = h_r(i)\} = \emptyset,
\]
i.e. that \(i\) collides with none of top \(k\) elements under \(h_r\).
(No) collisions with head items

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

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

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

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

\[ \Pr[h_r(i) = h_r(j)] \leq 1/b \]
(No) collisions with head items

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

\[ \{ j \in \text{HEAD} \setminus i : h_r(j) = h_r(i) \} = \emptyset, \]

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

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

\[ \Pr[h_r(i) = h_r(j)] \leq 1/b \]

Suppose that \( b \geq 8k \). Then by the union bound

\[ \Pr[\text{NO-COLLISIONS}_r(i)] \geq 1 - k/b \geq 1 - 1/8 \]
Consider contribution of head and tail items separately:

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

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

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

Show that all three events hold simultaneously with probability strictly bigger than \( 1/2 \) – so median gives good estimate
Consider contribution of head and tail items separately:

\[
\sum_{j \neq i : h_r(j) = h_r(i)} f_j^2 = \sum_{j \in \text{HEAD}, j \neq i \atop h_r(j) = h_r(i)} f_j^2 + \sum_{j \in \text{TAIL}, j \neq i \atop h_r(j) = h_r(i)} f_j^2
\]

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

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

Show that all three events hold simultaneously with probability strictly bigger than \( 1/2 \) – so median gives good estimate
Small variance from tail elements

\[ \text{SMALL-VARIANCE}_r(i) := \text{event that} \]

\[ \sum_{j \in TAIL, j \neq i} f_j^2 \leq \frac{8}{b} \sum_{j \in TAIL} f_j^2 \]

For every \( i, j \in \{1, \ldots, m\} \), \( i \neq j \) and \( r \in \{1, \ldots, t\} \)

\[ \Pr[h_r(i) = h_r(j)] = \frac{1}{b} \]

(Since \( b \) is the number of buckets)
Small variance from tail elements

\[ \text{SMALL-VARIANCE}_r(i) := \text{event that} \]
\[ \sum_{j \in TAIL, j \neq i, h_r(j) = h_r(i)} f_j^2 \leq \frac{8}{b} \sum_{j \in TAIL} f_j^2 \]

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

\[ \Pr_{h_r}[h_r(i) = h_r(j)] = 1/b \quad (b \text{ is the number of buckets}) \]
Small variance from tail elements

**SMALL-VARIANCE}_r(i):=event that**

\[
\sum_{j \in \text{TAIL}, j \neq i, h_r(j) = h_r(i)} f_j^2 \leq \frac{8}{b} \sum_{j \in \text{TAIL}} f_j^2
\]

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

\[
\text{Pr}_{h_r}[h_r(i) = h_r(j)] = \frac{1}{b} \quad (b \text{ is the number of buckets})
\]

So by linearity of expectation

\[
E \left[ \sum_{j \in \text{TAIL}, j \neq i, h_r(j) = h_r(i)} f_j^2 \right] = \sum_{j \in \text{TAIL}, j \neq i} f_j^2 \cdot \text{Pr}_{h_r}[h_r(i) = h_r(j)] \\
\leq \frac{1}{b} \sum_{j \in \text{TAIL}} f_j^2
\]
Theorem

For every non-negative random variable $X$ with mean $\mu \geq 0$, and every $k \geq 1$ one has

$$\Pr[X \geq k \cdot \mu] \leq 1/k$$
We proved that

\[ \mathbb{E} \left[ \sum_{j \in \text{TAIL}, j \neq i \atop h_r(j) = h_r(i)} f_j^2 \right] \leq \frac{1}{b} \sum_{j \in \text{TAIL}} f_j^2 \]

By Markov’s inequality one has, for every \( i \) and every \( r \),

\[ \Pr[\text{SMALL-VARIANCE}_r(i)] \geq 1 - 1/8 \]
NO-COLLISIONS$_r(i)$ and SMALL-VARIANCE$_r(i)$: recap

Consider contribution of head and tail items separately:

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

Conditioned on NO-COLLISIONS$_r(i)$ and SMALL-VARIANCE$_r(i)$
NO-COLLISIONS$_r(i)$ and SMALL-VARIANCE$_r(i)$: recap

Consider contribution of head and tail items separately:

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

Conditioned on NO-COLLISIONS$_r(i)$ and SMALL-VARIANCE$_r(i)$

- first term is zero
**NO-COLLISIONS_r(i) and SMALL-VARIANCE_r(i): recap**

Consider contribution of head and tail items separately:

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

Conditioned on **NO-COLLISIONS_r(i)** and **SMALL-VARIANCE_r(i)**

- first term is zero
- second term is at most

\[
\frac{8}{b} \sum_{j \in \text{TAIL}} f_j^2
\]
Small deviation event

\[ \text{SMALL-DEVIA}\text{TION}_r(i) = \text{event that} \]

\[ (C[r, h_r(i)] \cdot s_r(i) - f_i)^2 \leq 8\text{Var}(C[r, h_r(i)] \cdot s_r(i)). \]
Small deviation event

\( \text{SMALL-DEVIATION}_r(i) \) = event that

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

By Chebyshev’s inequality one has, for every \( i \) and every \( r \),

\[ \Pr[\text{SMALL-DEVIATION}_r(i)] \geq 1 - 1/8 \]
\[
\Pr[\text{SMALL-VARIANCE}_{r(i)}] \geq 1 - 1/8
\]

\[
\Pr[\text{NO-COLLISIONS}_{r(i)}] \geq 1 - 1/8
\]

\[
\Pr[\text{SMALL-DEVIATION}_{r(i)}] \geq 1 - 1/8
\]

So by the union bound

\[
\Pr[\text{SMALL-VARIANCE}_{r(i)} \text{ and NO-COLLISIONS}_{r(i)} \text{ and SMALL-DEVIATION}_{r(i)}] \geq 5/8.
\]
Let
\[ \gamma := \sqrt{\frac{1}{b} \sum_{j \in TAIL} f_j^2} \]

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

**Lemma**
*If \( b \geq 8k \), then for every \( i \), every \( r \in [1 : t] \),*

\[
\text{Pr}[|C[r, h_r(i)] \cdot s_r(i) - f_i| \leq 8\gamma] \geq 5/8
\]
Let
\[ \gamma := \sqrt{\frac{1}{b} \sum_{j \in \text{TAIL}} f_j^2} \]

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

Lemma
If \( b \geq 8k \) and \( t \geq A \log N \) for an absolute constant \( A > 0 \), then for every \( i \), with probability \( \geq 1 - 1/N^4 \)

\[ |\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\} - f_i| \leq 8\gamma \]

at the end of the stream.

Proof.
Chernoff bounds. \( \square \)
Let
\[ \gamma := \sqrt{\frac{1}{b} \sum_{j \in TAIL} f_j^2} \]

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

**Lemma**

*If \( b \geq 8k \) and \( t \geq A \log N \) for an absolute constant \( A > 0 \), then with probability \( \geq 1 - 1/N^3 \) for every \( i \in [m] \)\
\[ \left| \text{median}_r \{ C[r, h_r(i)] \cdot s_r(i) \} - f_i(p) \right| \leq 8\gamma \]

*at the end of the stream.*

**Proof.**

Chernoff bounds. \( \square \)
Let
\[ \gamma := \sqrt{\frac{1}{b} \sum_{j \in TAIL} f_j^2} \]

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

**Lemma**

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

\[
\left| \text{median}_r \{ C[r, h_r(i)] \cdot s_r(i) \} - f_i(p) \right| \leq \varepsilon f_k
\]

at the end of the stream.
Let
$$\gamma := \sqrt{\frac{1}{b} \sum_{j \in \text{T  A I L}} f_i^2}$$

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

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

$$\left| \text{median}_r \{ C[r, h_r(i)] \cdot s_r(i) \} - f_i(p) \right| \leq \varepsilon f_k$$

at the end of the stream.

**Proof.**
Substitute value of $b$ into definition of $\gamma$:

$$\gamma = \sqrt{\frac{1}{b} \sum_{j \in \text{T  A I L}} f_i^2} \leq \varepsilon f_k / 8$$
CountSketch (recap and proofs)

Graph streaming

Connectivity via sketching

Designing $\ell_0$ samplers
CountSketch (recap and proofs)

Graph streaming

Connectivity via sketching

Designing $\ell_0$ samplers
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- streaming model: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
**Streaming model**

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\mathcal{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream
Streaming model

- **streaming model**: edges of $G$ arrive in an arbitrary order in a stream;
- algorithm can only use $\tilde{O}(n)$ space
- several passes over the stream *(ideally one pass)*
Construct a spanning tree of the graph $G$ in a single pass?
Construct a spanning tree of the graph $G$ in a single pass?

Very easy in insertion only streams:

- maintain a spanning forest
- add incoming edge if it connects two components, discard otherwise
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Spanning trees in insertion-only streams
Construct a spanning tree of the graph $G$ in a single pass?

Very easy in insertion only streams:

- maintain a spanning forest
- add incoming edge if it connects two components, discard otherwise
Construct a spanning tree of the graph $G$ in a single pass?

Very easy in insertion only streams:

- maintain a spanning forest
- add incoming edge if it connects two components, discard otherwise

Many modern networks evolve over time, edges both inserted and deleted

Construct spanning trees in dynamic streams in small space?
What if we have deletions?
What if we have deletions?
What if we have deletions?
What if we have deletions?
What if we have deletions?
What if we have deletions?

![Graph with deletions highlighted](image-url)
What if we have deletions?
What if we have deletions?
What if we have deletions?
What if we have deletions?

Very different algorithms are needed...
Graph sketching

Main idea: apply classical sketching techniques on the edge incidence matrix of a graph

Each row of $B$ = potential edge in $G$.

If $e = (u, v) \in E$, then $b_e = \chi_u - \chi_v$, otherwise $b_e = 0$
Graph sketching

Main idea: apply classical sketching techniques on the edge incidence matrix of a graph

Each row of $B$=potential edge in $G$.

If $e = (u, v) \in E$, then $b_e = \chi_u - \chi_v$, otherwise $b_e = 0$
Graph sketching

Main idea: apply classical sketching techniques on the edge incidence matrix of a graph

Each row of $B$ = potential edge in $G$.

If $e = (u, v) \in E$, then $b_e = \chi_u - \chi_v$, otherwise $b_e = 0$
Graph sketching

For every $S \subseteq V$ let

$$\delta(S) := E \cap (S \times (V \setminus S))$$

denote the edges crossing the cut. Let $x = 1_S$ (indicator of $S$).

$Bx$ is the (signed) indicator of $\delta(S)$
- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- Designing $\ell_0$ samplers
- CountSketch (recap and proofs)
- Graph streaming
- **Connectivity via sketching**
- Designing $\ell_0$ samplers
Return $F_{T+1}$
A simple algorithm for connectivity

\[
F^0 \leftarrow \emptyset \\
C^0 \leftarrow V \quad \triangleright \text{current connected components} \\
\text{For } t = 0 \text{ to } T \quad \triangleright T = O(\log n) \\
\quad \text{For each } u \in C^t \\
\quad \quad \text{Choose an edge in } \delta(u) \\
\quad \text{End For} \\
F^{t+1} \leftarrow F^t \cup \{\text{spanning forest on selected edges}\} \\
C^{t+1} \leftarrow \{\text{new connected components}\} \\
\text{End} \\
\text{return } F^{T+1}
\]
A simple algorithm for connectivity

\[ F^0 \leftarrow \emptyset \]
\[ C^0 \leftarrow V \quad \triangleright \text{current connected components} \]
\[ \text{For } t = 0 \text{ to } T \quad \triangleright T = O(\log n) \]
\[ \quad \text{For each } u \in C^t \]
\[ \quad \quad \text{Choose an edge in } \delta(u) \]
\[ \quad \text{End For} \]
\[ F^{t+1} \leftarrow F^t \cup \{\text{spanning forest on selected edges}\} \]
\[ C^{t+1} \leftarrow \{\text{new connected components}\} \]
\[ \text{End} \]
\[ \text{return } F^{T+1} \]

(nonzero of \( B \cdot 1_u \))
A simple algorithm for connectivity

\[ F^0 \leftarrow \emptyset \]
\[ C^0 \leftarrow V \quad \triangleright \text{ current connected components} \]
\[ \text{For } t = 0 \text{ to } T \quad \triangleright T = O(\log n) \]
\[ \quad \text{For each } u \in C^t \]
\[ \quad \quad \text{Choose an edge in } \delta(u) \]
\[ \quad \text{End For} \]
\[ F^{t+1} \leftarrow F^t \cup \{ \text{spanning forest on selected edges} \} \]
\[ C^{t+1} \leftarrow \{ \text{new connected components} \} \]
\[ \text{End} \]
\[ \text{return } F^{T+1} \]
A simple algorithm for connectivity

\[ F^0 \leftarrow \emptyset \]
\[ C^0 \leftarrow V \quad \triangleright \quad \text{current connected components} \]

\textbf{For} \ t = 0 \ \textbf{to} \ T \quad \triangleright \quad T = O(\log n) \]

\textbf{For} each \ u \in C^t

Choose an edge in \( \delta(u) \)

\textbf{End For}

\[ F^{t+1} \leftarrow F^t \cup \{ \text{spanning forest on selected edges} \} \]

\[ C^{t+1} \leftarrow \{ \text{new connected components} \} \]

\textbf{End}

\textbf{return} \ F^{T+1}

(Sample nnz of \( B \cdot 1_u \)?)
A simple algorithm for connectivity

\[ F^0 \leftarrow \emptyset \]
\[ C^0 \leftarrow V \quad \triangleright \text{current connected components} \]
\[ \text{For } t = 0 \text{ to } T \quad \triangleright T = O(\log n) \]
\[ \text{For each } u \in C^t \]
\[ \quad \text{Choose an edge in } \delta(u) \]
\[ \text{End For} \]
\[ F^{t+1} \leftarrow F^t \cup \{\text{spanning forest on selected edges}\} \]
\[ C^{t+1} \leftarrow \{\text{new connected components}\} \]
\[ \text{End} \]
\[ \text{return } F^{T+1} \]
A simple algorithm for connectivity

\[ F^0 \leftarrow \emptyset \]
\[ C^0 \leftarrow V \quad \triangleright \quad \text{current connected components} \]
\[ \text{For } t = 0 \text{ to } T \quad \triangleright \quad T = O(\log n) \]
\[ \quad \text{For each } u \in C^t \]
\[ \quad \quad \text{Choose an edge in } \delta(u) \]
\[ \quad \text{End For} \]
\[ F^{t+1} \leftarrow F^t \cup \{ \text{spanning forest on selected edges} \} \]
\[ C^{t+1} \leftarrow \{ \text{new connected components} \} \]
\[ \text{End} \]
\[ \text{return } F^{T+1} \]
**Definitions**

A $\delta$-error $\ell_0$ sampler is

- a linear sketch $S \in \mathbb{R}^{m \times n}$
- a decoding primitive $\text{Dec} : \mathbb{R}^m \rightarrow [n]$

such that for every $x \in \mathbb{R}^n$ with integer entries $J \leftarrow \text{Dec}(Sx)$ satisfies

\[ \|J - \text{UNIF}_{\text{supp}(x)}\|_{TVD} \leq \delta. \]
\textbf{\(\ell_0\)-samplers}

\textbf{Definition}

A \(\delta\)-error \(\ell_0\) sampler is

\begin{itemize}
  \item a linear sketch \(S \in \mathbb{R}^{m \times n}\)
  \item a decoding primitive \(Dec : \mathbb{R}^m \to [n]\)
\end{itemize}

such that for every \(x \in \mathbb{R}^n\) with integer entries \(J \leftarrow Dec(Sx)\) satisfies

\[\|J - UNIF_{\text{supp}(x)}\|_{TVD} \leq \delta.\]

Informally: sample a uniformly random element, output \texttt{FAIL} or just garbage with probability at most \(\delta\)
\( \ell_0 \)-samplers

**Definition**

A \( \delta \)-error \( \ell_0 \) sampler is

- a linear sketch \( S \in \mathbb{R}^{m \times n} \)
- a decoding primitive \( \text{Dec} : \mathbb{R}^m \rightarrow [n] \)

such that for every \( x \in \mathbb{R}^n \) with integer entries \( J \leftarrow \text{Dec}(Sx) \) satisfies

\[ \| J - \text{UNIF}_{\text{supp}(x)} \|_{TVD} \leq \delta. \]

**Informally:** sample a uniformly random element, output FAIL or just garbage with probability at most \( \delta \)

**Recent constructions of \( \ell_p \) samplers due to**

Connectivity via sketching (Ahn-Guha-McGregor’12)

\[ F^0 \leftarrow \emptyset \]
\[ C^0 \leftarrow V \quad \triangleright \text{current connected components} \]

For \( t = 0 \) to \( T \quad \triangleright \quad T = O(\log n) \)

For each \( u \in C^t \)

Choose an edge in \( \delta(u) \)

End For

\[ F^{t+1} \leftarrow F^t \cup \{ \text{spanning forest on selected edges} \} \]
\[ C^{t+1} \leftarrow \{ \text{new connected components} \} \]

End

return \( F^{T+1} \)

\[ S^1, \ldots, S^T \leftarrow \ell_0\text{-samplers} \]
Maintain \( S^1 B, \ldots, S^T B \)

Run \( Dec(S^t B \cdot 1_u) \)
Some remarks

Why did we need $T$ sketches $S^1, \ldots, S^T$?

Very surprising: decoding is adaptive ($T = O(\log n)$ rounds), but sketch is not

Which other graph problems admit sketching solutions?
- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- Designing $\ell_0$ samplers
- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- Designing $\ell_0$ samplers
\textbf{\(\ell_0\)-samplers}

\textbf{Definition}
A \(\delta\)-error \(\ell_0\) sampler is

- a linear sketch \(S \in \mathbb{R}^{m \times n}\)
- a decoding primitive \(Dec : \mathbb{R}^m \rightarrow [n]\)

such that for every \(x \in \mathbb{R}^n\) with integer entries \(J \leftarrow Dec(Sx)\) satisfies

\[\|J - \text{UNIF}_{\text{supp}(x)}\|_{TVD} \leq \delta.\]
$\ell_0$-samplers

**Definition**
A $\delta$-error $\ell_0$ sampler is

- a linear sketch $S \in \mathbb{R}^{m \times n}$
- a decoding primitive $Dec : \mathbb{R}^m \rightarrow [n]$

such that for every $x \in \mathbb{R}^n$ with integer entries $J \leftarrow Dec(Sx)$ satisfies

$$\|J - UNIF_{supp(x)}\|_{TVD} \leq \delta.$$ 

*Informally:* sample a uniformly random element, output FAIL or just garbage with probability at most $\delta$. 

Recent constructions of $\ell_p$-samplers due to Frahling-Indyk-Sohler'08, Andoni-Krauthgamer-Onak'11, Jowhari-Saglam-Tardos'11, Nelson-Pachocki-Wang'17, Kapralov-Nelson-Pachocki-Wang-Woodruff-Yahyazadeh'17
\textbf{Definition}

A $\delta$-error $\ell_0$ sampler is

\begin{itemize}
\item a linear sketch $S \in \mathbb{R}^{m \times n}$
\item a decoding primitive $Dec : \mathbb{R}^m \rightarrow [n]$
\end{itemize}

such that for every $x \in \mathbb{R}^n$ with integer entries $J \leftarrow Dec(Sx)$ satisfies

$$\|J - \text{UNIF}_{\text{supp}(x)}\|_{TVD} \leq \delta.$$ 

**Informally:** sample a uniformly random element, output FAIL or just garbage with probability at most $\delta$

Recent constructions of $\ell_p$ samplers due to

\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
- Assume \( N \) is known
- Output item \( i \) with probability \( \sim f^p_i \)
  
  \((f_i=\text{number of occurrences of } i)\)
- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

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

  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)

  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( l_p \) sampling problem

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

  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)

  \((f_i=\text{number of occurrences of } i)\)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

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

Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)

\( (f_i = \text{number of occurrences of } i) \)

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

\[ 3 4 6 3 2 10 3 1 3 3 2 2 5 5 5 9 8 7 4 4 2 2 3 1 \]
$\ell_p$ sampling problem

- Single pass over the data: $i_1, i_2, ..., i_N$
  
  Assume $N$ is known

- Output item $i$ with probability $\sim f_i^p$
  
  ($f_i=$number of occurrences of $i$)

- Small storage: will get $\log^{O(1)} N$
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
$\ell_p$ sampling problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output item $i$ with probability $\sim f_i^p$
  
  ($f_i =$ number of occurrences of $i$)

- Small storage: will get $\log^{O(1)} N$
ℓ_p sampling problem

- Single pass over the data: \(i_1, i_2, \ldots, i_N\)
  
  Assume \(N\) is known

- Output item \(i\) with probability \(\sim f_i^p\)

  \((f_i=\text{number of occurrences of } i)\)

- Small storage: will get \(\log^{O(1)} N\)
$\ell_p$ sampling problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output item $i$ with probability $\sim f_i^p$
  
  ($f_i=$number of occurrences of $i$)

- Small storage: will get $\log^{O(1)} N$
$\ell_p$ sampling problem

- Single pass over the data: $i_1, i_2, \ldots, i_N$
  
  Assume $N$ is known

- Output item $i$ with probability $\sim f^p_i$
  
  ($f_i=$number of occurrences of $i$)

- Small storage: will get $\log^{O(1)} N$
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
ℓ_p sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \((f_i=\text{number of occurrences of } i)\)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i = \text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  - Assume \( N \) is known
- Output item \( i \) with probability \( \sim f_i^p \)
  - \( f_i = \) number of occurrences of \( i \)
- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)

  \((f_i=\text{number of occurrences of } i)\)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( f_i = \text{number of occurrences of } i \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \( (f_i=\text{number of occurrences of } i) \)

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

```plaintext
3 4 6 3 2 10 3 1 3 1 2 2 5 5 5 9 8 7 4 4 2 2 3
```
\( \ell_p \) sampling problem

- Single pass over the data: \( i_1, i_2, \ldots, i_N \)
  
  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)
  
  \((f_i=\text{number of occurrences of } i)\)

- Small storage: will get \( \log^{O(1)} N \)
\( l_p \) sampling problem

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

  Assume \( N \) is known

- Output item \( i \) with probability \( \sim f_i^p \)

  \( (f_i=\text{number of occurrences of } i) \)

- Small storage: will get \( \log^{O(1)} N \)
Main idea:

- if $x$ is 1-sparse (has a single nonzero), can recover $x$ using few rows
$\ell_0$ sampler construction (sketch)

Main idea:

- if $x$ is 1-sparse (has a single nonzero), can recover $x$ using few rows
- if $x$ is $t$-sparse, a subsampling of $x$ at rate $\approx 1/t$ is likely 1-sparse
Recovering 1-sparse signals

Design a sketch $S$ from which any $x \in \mathbb{R}^n$ with $\text{supp}(x) = 1$ can be recovered with probability 1?

$S \cdot x = b$

sketching matrix

space requirement = number of rows
Recovering 1-sparse signals

Design a sketch $S$ from which any $x \in \mathbb{R}^n$ with $\text{supp}(x) = 1$ can be recovered with probability 1?

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & \cdots & n-1 & n \end{bmatrix} \cdot x = b$$

sketching matrix

space requirement = number of rows

Suppose $x$ has one nonzero: $x = \alpha \cdot e_{i^*}$

Compute

$$A = \sum_{i=1}^{n} x_i = \langle u, x \rangle = \alpha$$

$$B = \sum_{i=1}^{n} i \cdot x_i = \langle v, x \rangle = \alpha \cdot i^*$$

So

$$\alpha = \langle x, u \rangle$$

and

$$i^* = \frac{\langle x, v \rangle}{\langle x, u \rangle}$$
What if $x$ is not sparse?

Let $2^j$ be the closest power of 2 to $\|x\|_0$. 

Try $O(\log n)$ powers of 2, run 1-sparse recovery on $x^S$! Also need to verify that recovery was successful (can be done).
What if $x$ is not sparse?

Let $2^j$ be the closest power of 2 to $\|x\|_0$.

Choose a subset $S \subseteq [n]$ such that for every $i \in [n]$

$$\Pr[i \in S] = 2^{-j}$$
What if $x$ is not sparse?

Let $2^j$ be the closest power of 2 to $\|x\|_0$.

Choose a subset $S \subseteq [n]$ such that for every $i \in [n]$

$$\Pr[i \in S] = 2^{-j}$$

Then

$$\Pr[|\text{supp}(x) \cap S| = 1] = \Omega(1)$$
What if $x$ is not sparse?

Let $2^j$ be the closest power of 2 to $\|x\|_0$.

Choose a subset $S \subseteq [n]$ such that for every $i \in [n]$

$$\Pr[i \in S] = 2^{-j}$$

Then

$$\Pr[|\text{supp}(x) \cap S| = 1] = \Omega(1)$$

Try $O(\log n)$ powers of 2, run 1-sparse recovery on $x_S$!

Also need to verify that recovery was successful (can be done)
Optimal bounds for $\ell_0$-samplers

**Definition**
A $\delta$-error $\ell_0$ sampler is

- a linear sketch $S \in \mathbb{R}^{m \times n}$
- a decoding primitive $Dec : \mathbb{R}^m \to [n]$

such that for every $x \in \mathbb{R}^n$ with integer entries $J \leftarrow Dec(Sx)$ satisfies

$$\|J - UNIF_{\text{supp}(x)}\|_{TVD} \leq \delta.$$ 

**Jowhari-Saglam-Tardos'11**: there exist $\delta$-error $\ell_0$-samplers with $m = O(\log n \log(1/\delta))$ rows.

**Kapralov-Nelson-Pachocki-Wang-Woodruff-Yahyazadeh'17**: this space bound is optimal for $\delta > 2^{-n^{0.99}}$ (and more results)