Lecture 3: CountSketch, Graph sketching, ℓ_0 Samplers

Michael Kapralov

EPFL

May 26, 2017

- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- Designing ℓ_0 samplers

- **CountSketch (recap and proofs)**
- Graph streaming
- Connectivity via sketching
- Designing ℓ_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!

1 2 3 4 5 6 7 8 9 10

Heavy hitters problem

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

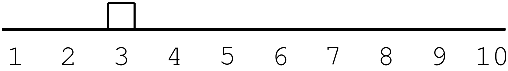
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3

Heavy hitters problem

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

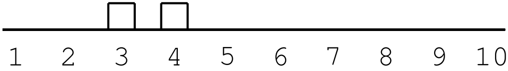
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4

Heavy hitters problem

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

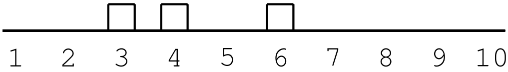
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6

Heavy hitters problem

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

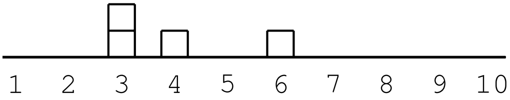
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3

Heavy hitters problem

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

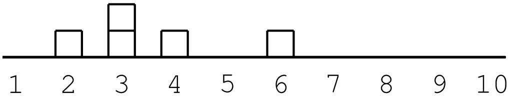
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3 2

Heavy hitters problem

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

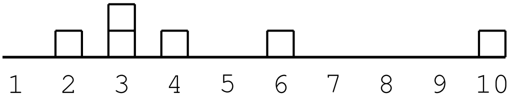
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3 2 10

Heavy hitters problem

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

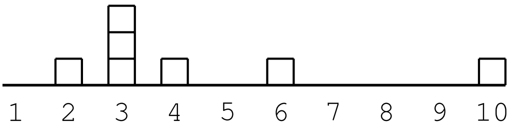
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3 2 10 3

Heavy hitters problem

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

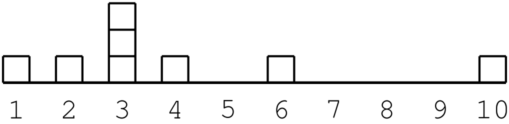
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3 2 10 3 1

Heavy hitters problem

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

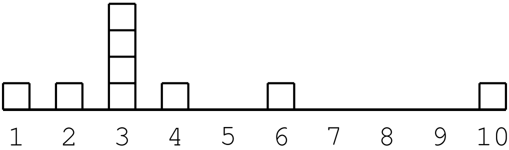
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3 2 10 3 1 3

Heavy hitters problem

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

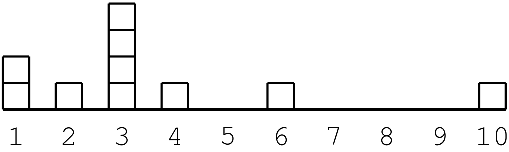
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1

Heavy hitters problem

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

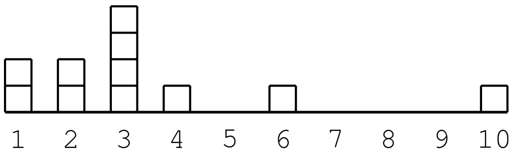
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



3 4 6 3 2 10 3 1 3 1 2

Heavy hitters problem

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

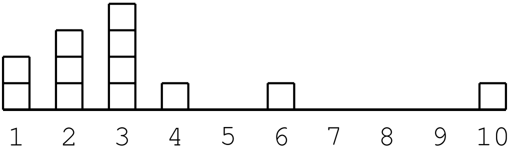
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

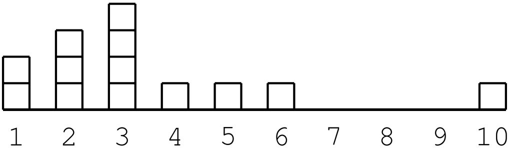
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

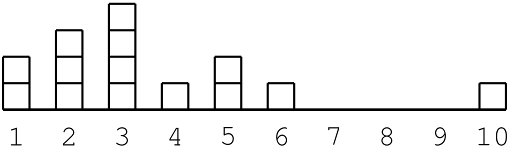
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

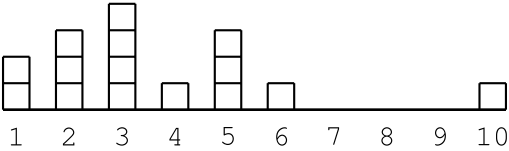
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

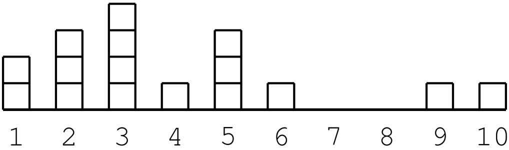
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

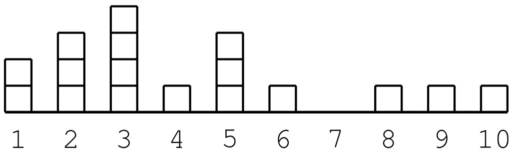
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

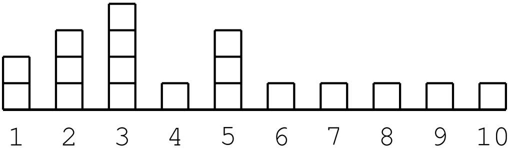
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

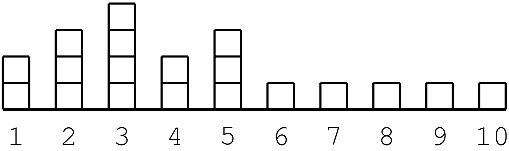
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

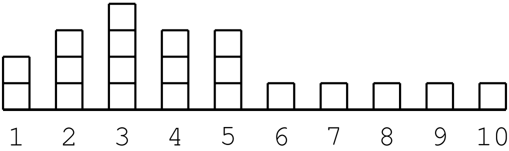
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

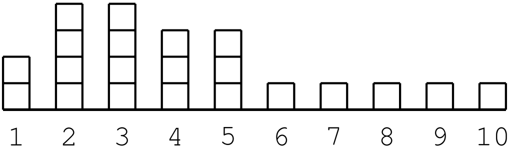
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

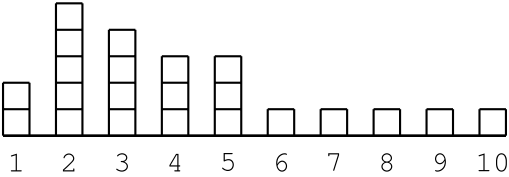
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

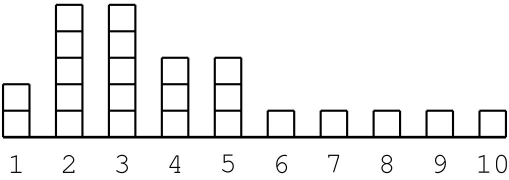
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

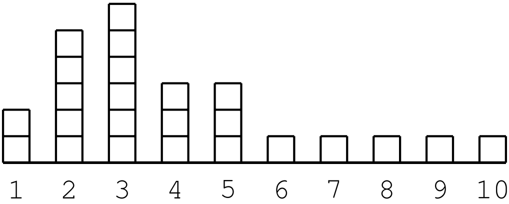
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

Heavy hitters problem

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

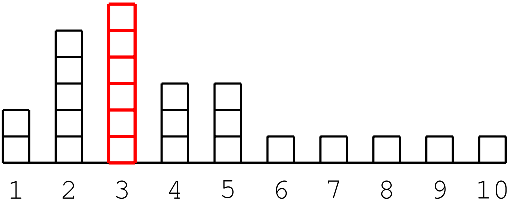
  Assume N is known

- Output k most frequent items

  (Heavy hitters)

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

  Much better than storing all items!



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

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 $\widehat{f_i}$ of f_i

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 $\widehat{f}_i$ of f_i

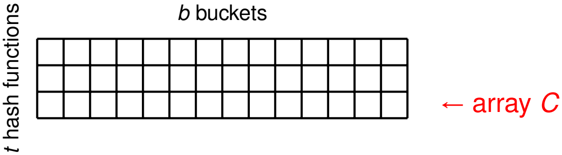
To be specified:

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

ApproxPointQuery

Choose

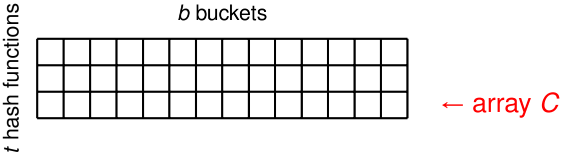
- t random hash functions $h_1, h_2, \dots, h_t$ from items $[m]$ to $b \approx k$ buckets $\{1, 2, \dots, b\}$
- t random hash functions $s_1, s_2, \dots, s_t$ from items $[m]$ to $\{-1, +1\}$



ApproxPointQuery

Choose

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



The algorithm runs t independent copies of basic estimate:

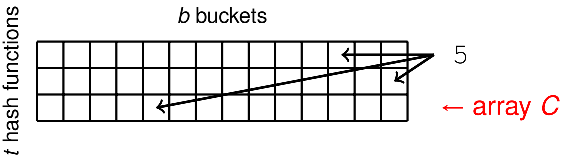
UPDATE(C, i)
for $r \in [1:t]$
  $C[r, h_r(i)] \leftarrow C[r, h_r(i)] + s_r(i)$
end for

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

ApproxPointQuery

Choose

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



The algorithm runs t independent copies of basic estimate:

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

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

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

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

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

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

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

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

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

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

and

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

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

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

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

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

and

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

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

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

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

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

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

and

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

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

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

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

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

and

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

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

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

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

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

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

Lemma

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

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

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

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

Lemma

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

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

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 $O(b \log N)$

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

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

Lemma

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

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

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

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

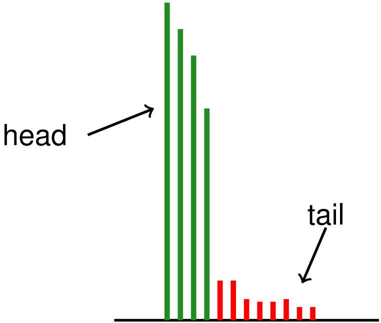
Space complexity is $O(b \log N)$

How large is b?

Space complexity

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

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



In practice, choose b subject to space constraints, detect elements with counts above $O\left(\varepsilon\sqrt{\frac{1}{k}\sum_{j\in TAIL} f_j^2}\right)$

Space complexity

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

Space complexity

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

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

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

Space complexity

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

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

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

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

Space complexity

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

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

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

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

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

Space complexity

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

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

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

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

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

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

Final algorithm: CountSketch

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

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

ApproxPointQuery(S, i, ε): returns $\widehat{f_i} \in [f_i - \varepsilon f_k, f_i + \varepsilon f_k]$

Find head items if they contribute the bulk of the stream in ℓ_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 $\text{median}_r \{C[r, h_r(i)] \cdot s_r(i)\}$

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

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

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 $r \in [1:t]$

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

and

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

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

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

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

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

and

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

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

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

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

and

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

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

Consider contribution of head and tail items separately:

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

Consider contribution of head and tail items separately:

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

Consider contribution of head and tail items separately:

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

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

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

Consider contribution of head and tail items separately:

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

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

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

Consider contribution of head and tail items separately:

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

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

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

Consider contribution of head and tail items separately:

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

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

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

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

(No) collisions with head items

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

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

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

(No) collisions with head items

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

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

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

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

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

(No) collisions with head items

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

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

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

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

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

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

$$\begin{aligned}\mathbf{Pr}[\textsc{No-Collisions}_r(i)] &\geq 1 - k/b \\ &\geq 1 - 1/8\end{aligned}$$

Consider contribution of head and tail items separately:

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

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

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

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

Consider contribution of head and tail items separately:

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

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

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

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

Small variance from tail elements

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

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

Small variance from tail elements

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

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

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

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

Small variance from tail elements

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

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

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

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

So by linearity of expectation

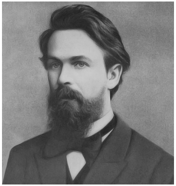
$$\begin{aligned}\mathbf{E}\left[\sum_{\substack{j\in TAIL, j\neq i\\ h_r(j)=h_r(i)}} f_j^2\right] &= \sum_{j\in TAIL, j\neq i} f_j^2 \cdot \mathbf{Pr}_{h_r}[h_r(i)=h_r(j)]\\ &\leq \frac{1}{b}\sum_{j\in TAIL} f_j^2\end{aligned}$$

Markov's inequality

Theorem

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

$$\mathbf{Pr}[X \geq k \cdot \mu] \leq 1/k$$



We proved that

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

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

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

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

Consider contribution of head and tail items separately:

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

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

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

Consider contribution of head and tail items separately:

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

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

- first term is zero

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

Consider contribution of head and tail items separately:

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

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

- first term is zero
- second term is at most

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

Small deviation event

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

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

Small deviation event

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

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

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

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

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

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

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

So by the union bound

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

Let

$$\gamma := \sqrt{\frac{1}{b}\sum_{j \in TAIL} f_j^2}$$

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

Lemma

If $b \geq 8k$, then for every i, every $r \in [1:t]$,

$$\mathbf{Pr}[|C[r, h_r(i)] \cdot s_r(i) - f_i| \leq 8\gamma] \geq 5/8$$

Let

$$\gamma := \sqrt{\frac{1}{b} \sum_{j \in TAIL} f_j^2}$$

For every $p \in [1:N]$ let $f_i(p)$:=frequency of i 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 $\geq 1 - 1/N^4$

$$\left| median_r \{C[r, h_r(i)] \cdot s_r(i)\} - f_i \right| \leq 8\gamma$$

at the end of the stream.

Proof.

Chernoff bounds. □

Let

$$\gamma := \sqrt{\frac{1}{b}\sum_{j \in TAIL} f_j^2}$$

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

$$\left| median_r \{C[r, h_r(i)] \cdot s_r(i)\} - f_i(p) \right| \leq 8\gamma$$

at the end of the stream.

Proof.

Chernoff bounds. $\square$

Let

$$\gamma := \sqrt{\frac{1}{b}\sum_{j\in TAIL} f_i^2}$$

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

Lemma

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

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

at the end of the stream.

Let

$$\gamma := \sqrt{\frac{1}{b}\sum_{j\in TAIL} f_i^2}$$

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

Lemma

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

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

at the end of the stream.

Proof.

Substitute value of b into definition of γ:

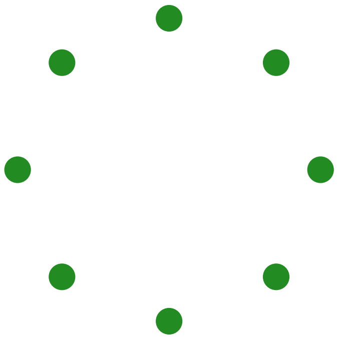
$$\gamma = \sqrt{\frac{1}{b}\sum_{j\in TAIL} f_i^2} \leq \varepsilon f_k/8$$

- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- Designing ℓ_0 samplers

- CountSketch (recap and proofs)
- **Graph streaming**
- Connectivity via sketching
- Designing ℓ_0 samplers

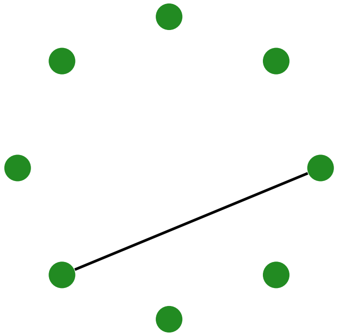
Streaming model

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



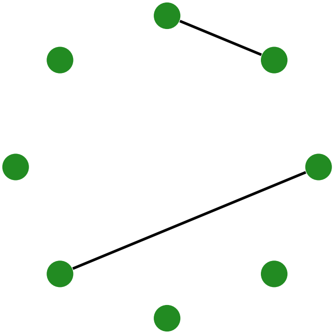
Streaming model

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



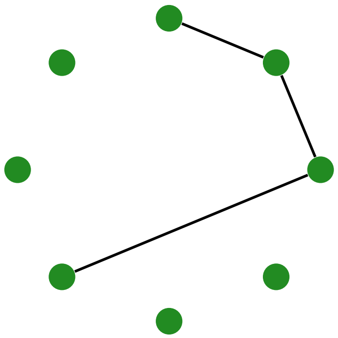
Streaming model

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



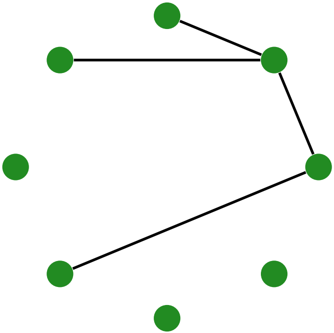
Streaming model

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



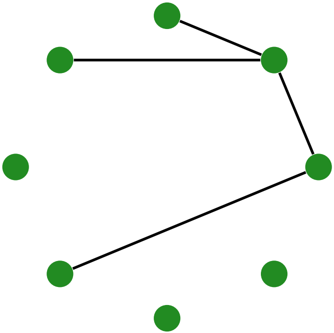
Streaming model

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



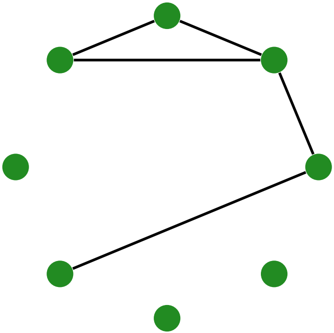
Streaming model

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



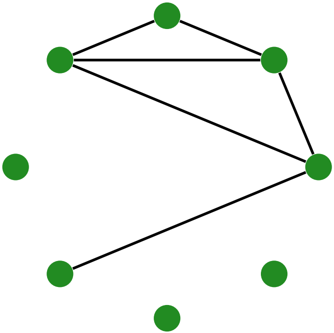
Streaming model

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



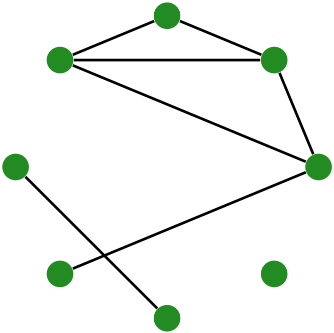
Streaming model

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



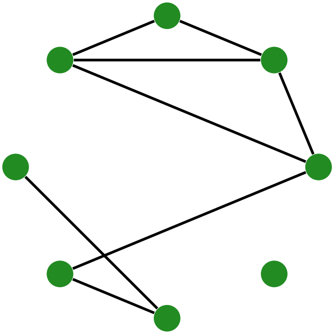
Streaming model

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



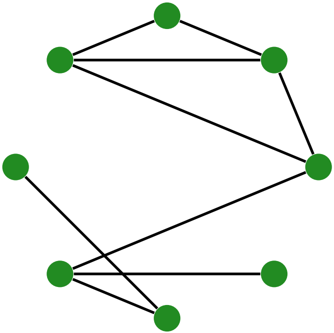
Streaming model

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



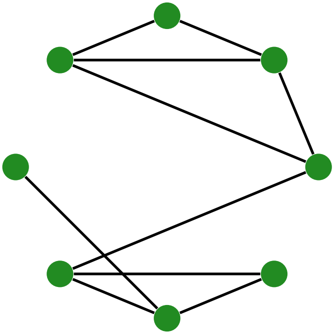
Streaming model

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



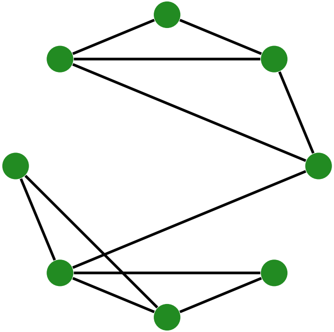
Streaming model

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



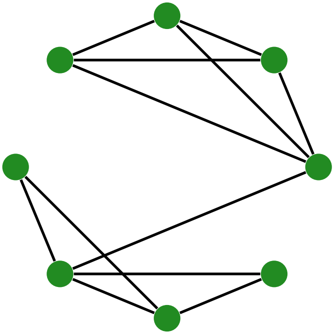
Streaming model

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



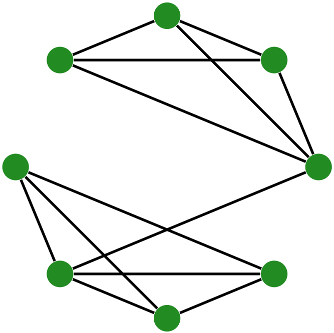
Streaming model

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



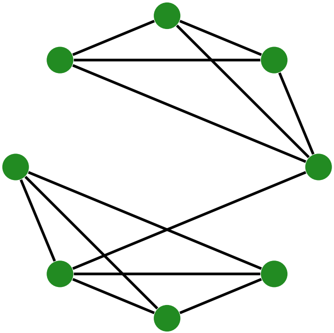
Streaming model

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



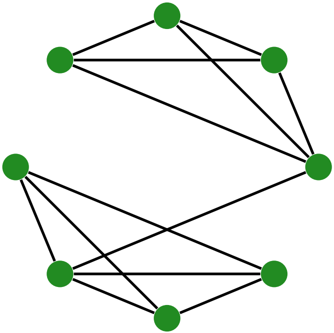
Streaming model

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



Insertion-only stream

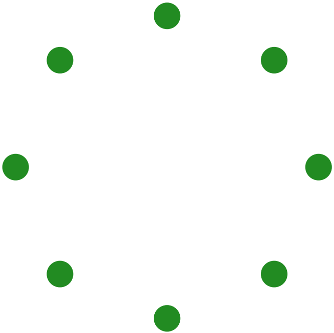
Construct a spanning tree of the graph G in a single pass?

Construct a spanning tree of the graph G in a single pass?

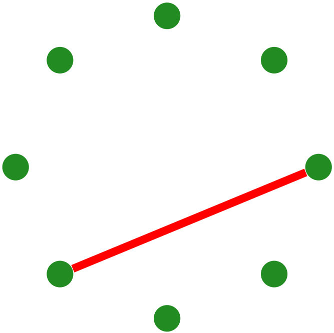
Very easy in insertion only streams:

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

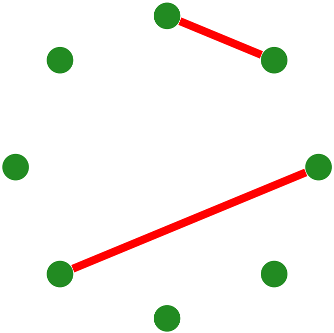
Spanning trees in insertion-only streams



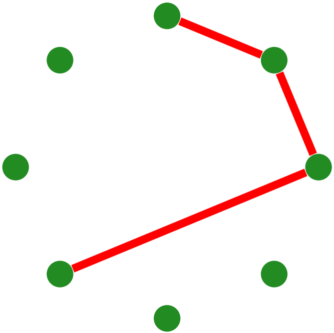
Spanning trees in insertion-only streams



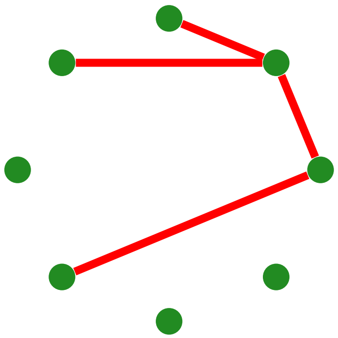
Spanning trees in insertion-only streams



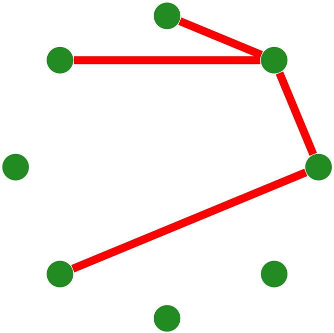
Spanning trees in insertion-only streams



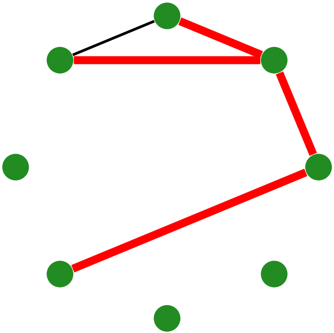
Spanning trees in insertion-only streams



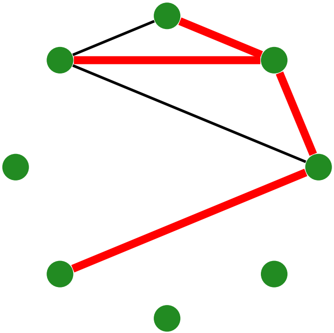
Spanning trees in insertion-only streams



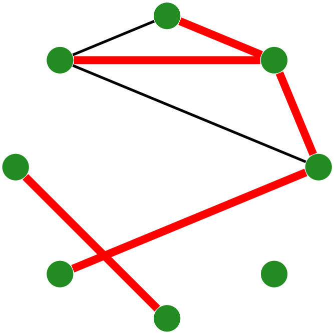
Spanning trees in insertion-only streams



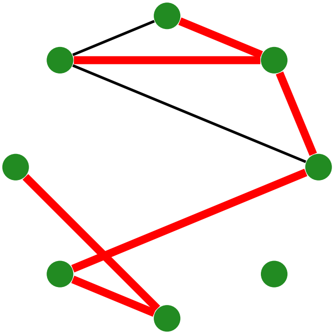
Spanning trees in insertion-only streams



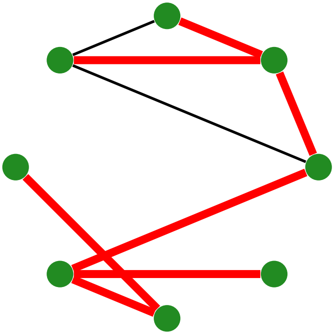
Spanning trees in insertion-only streams



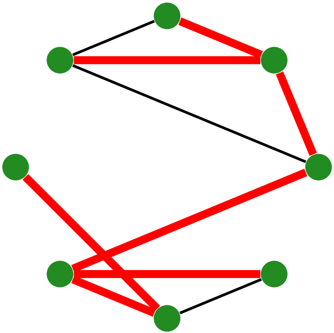
Spanning trees in insertion-only streams



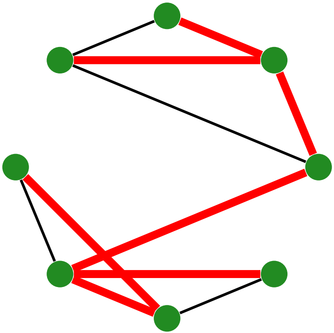
Spanning trees in insertion-only streams



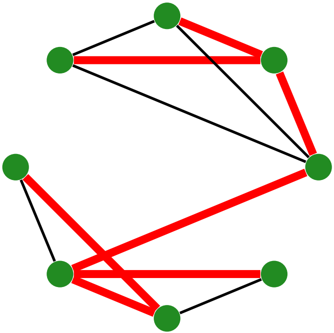
Spanning trees in insertion-only streams



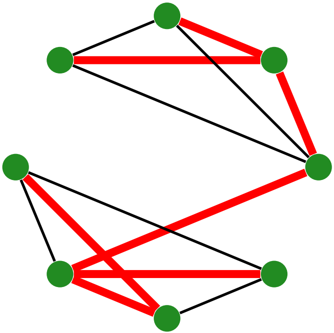
Spanning trees in insertion-only streams



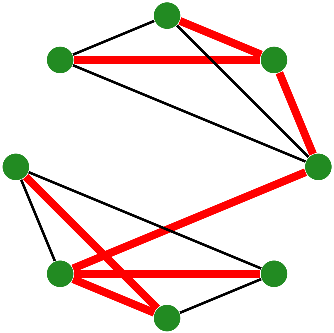
Spanning trees in insertion-only streams



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?

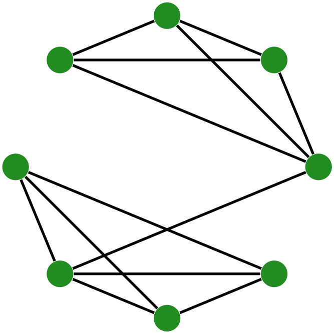
Very easy in insertion only streams:

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

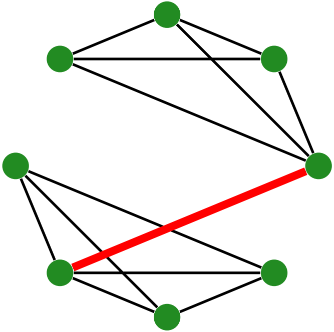
Many modern networks evolve over time, edges both inserted and deleted

Construct spanning trees in dynamic streams in small space?

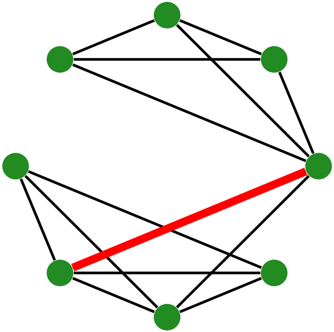
What if we have deletions?



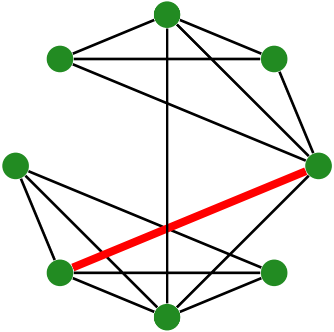
What if we have deletions?



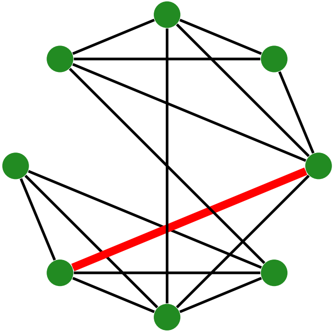
What if we have deletions?



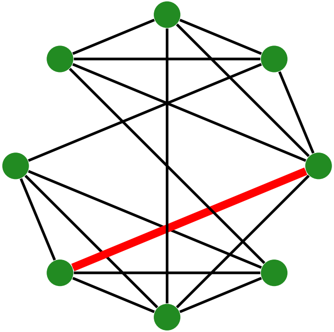
What if we have deletions?



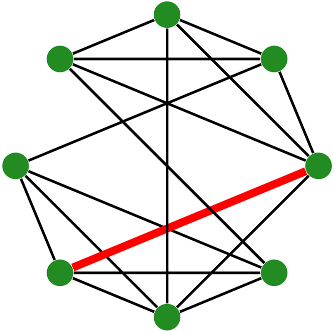
What if we have deletions?



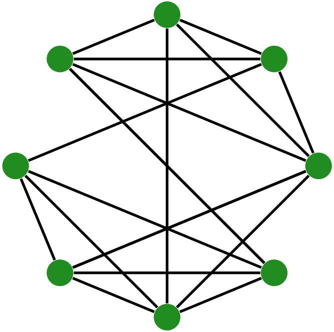
What if we have deletions?



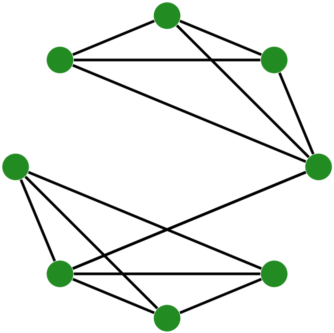
What if we have deletions?



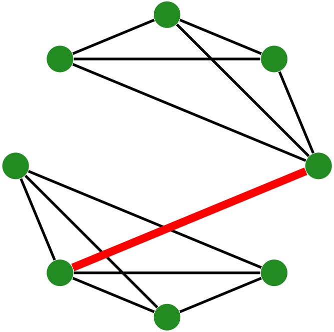
What if we have deletions?



What if we have deletions?



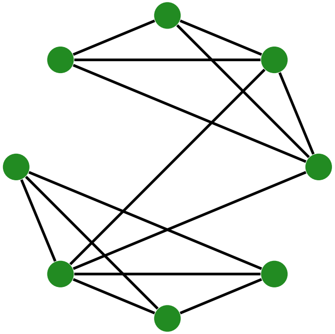
What if we have deletions?



Very different algorithms are needed...

Graph sketching

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



$$B = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & \vdots \end{bmatrix}$$

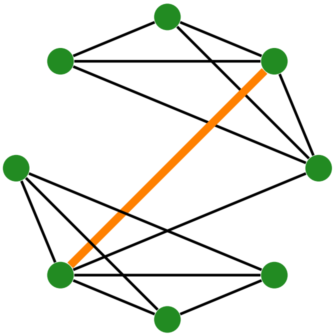
$\binom{n}{2} \times n$

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



$$B = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & \vdots \end{bmatrix}$$

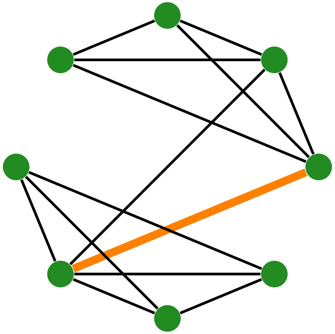
$\binom{n}{2} \times n$

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



$$B = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{-1} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & \vdots \end{bmatrix}$$

$\binom{n}{2} \times n$

Each row of B=potential edge in G.

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

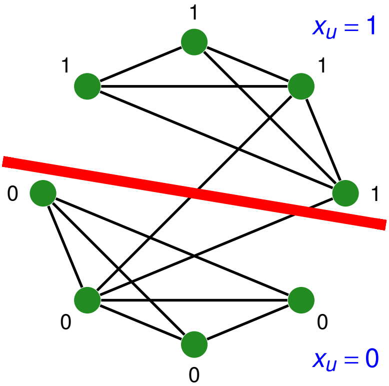
Graph sketching

For every $S \subseteq V$ let

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

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

$$\boxed{Bx \text{ is the (signed) indicator of } \delta(S)}$$



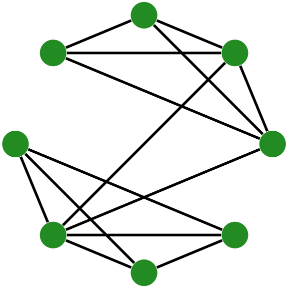
$$B = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & \vdots \end{bmatrix}$$

$\binom{n}{2} \times n$

- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- Designing ℓ_0 samplers

- CountSketch (recap and proofs)
- Graph streaming
- **Connectivity via sketching**
- Designing ℓ_0 samplers

A simple algorithm for connectivity



$F^0 \leftarrow \emptyset$

$C^0 \leftarrow V$ $\quad \triangleright$ current connected components

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

  For each $u \in C^t$

    Choose an edge in $\delta(u)$ (nonzero of $B \cdot \mathbf{1}_u$)

  End For

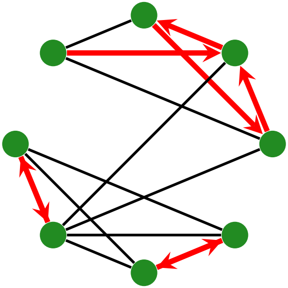
  $F^{t+1} \leftarrow F^t \cup \{$spanning forest on selected edges$\}$

  $C^{t+1} \leftarrow \{$new connected components$\}$

End

return F^{T+1}

A simple algorithm for connectivity



$F^0 \leftarrow \emptyset$

$\color{red}{C^0} \leftarrow V$ $\quad \triangleright$ current connected components

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

  For each $u \in \color{red}{C^t}$

    Choose an edge in $\delta(u)$ (nonzero of $B \cdot \mathbf{1}_u$)

  End For

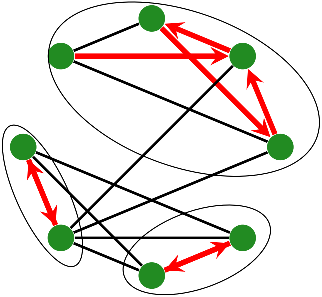
  $F^{t+1} \leftarrow F^t \cup$ {spanning forest on selected edges}

  $\color{red}{C^{t+1}} \leftarrow$ {new connected components}

End

return F^{T+1}

A simple algorithm for connectivity



$F^0 \leftarrow \emptyset$

$C^0 \leftarrow V$ $\quad \triangleright$ current connected components

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

For each $u \in C^t$

Choose an edge in $\delta(u)$ (nonzero of $B \cdot \mathbf{1}_u$)

End For

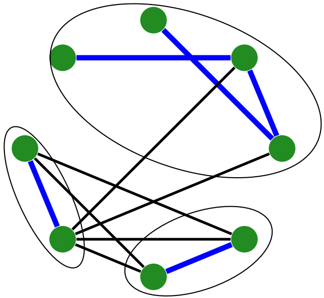
$F^{t+1} \leftarrow F^t \cup$ {spanning forest on selected edges}

$C^{t+1} \leftarrow$ {new connected components}

End

return F^{T+1}

A simple algorithm for connectivity



$F^0 \leftarrow \emptyset$

$C^0 \leftarrow V$ $\quad \triangleright$ current connected components

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

  For each $u \in C^t$

    Choose an edge in $\delta(u)$ $\quad$ (nonzero of $B \cdot \mathbf{1}_u$)

  End For

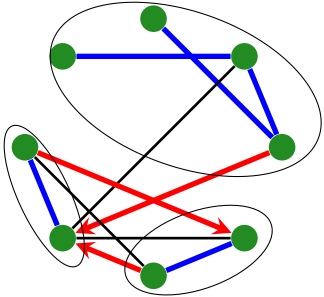
  $F^{t+1} \leftarrow F^t \cup \{\text{spanning forest on selected edges}\}$

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

End

return F^{T+1}

A simple algorithm for connectivity



$F^0 \leftarrow \emptyset$

$C^0 \leftarrow V$ $\quad \triangleright$ current connected components

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

  For each $u \in C^t$

    Choose an edge in $\delta(u)$ (Sample nnz of $B \cdot \mathbf{1}_u$?)

  End For

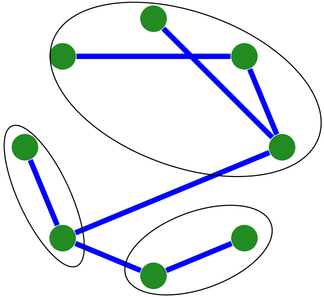
  $F^{t+1} \leftarrow F^t \cup$ {spanning forest on selected edges}

  $C^{t+1} \leftarrow$ {new connected components}

End

return F^{T+1}

A simple algorithm for connectivity



$F^0 \leftarrow \emptyset$

$C^0 \leftarrow V$ $\quad \triangleright$ current connected components

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

  For each $u \in C^t$

    Choose an edge in $\delta(u)$ (Sample nnz of $B \cdot \mathbf{1}_u$?)

  End For

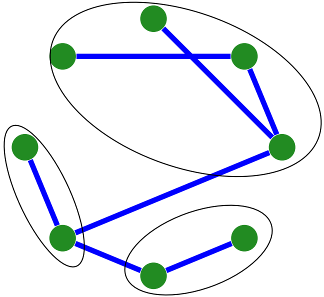
  $F^{t+1} \leftarrow F^t \cup$ {spanning forest on selected edges}

  $C^{t+1} \leftarrow$ {new connected components}

End

return F^{T+1}

A simple algorithm for connectivity



$F^0 \leftarrow \emptyset$

$C^0 \leftarrow V$ $\quad \triangleright$ current connected components

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

  For each $u \in C^t$

    Choose an edge in $\delta(u)$

  End For

  $F^{t+1} \leftarrow F^t \cup$ {spanning forest on selected edges}

  $C^{t+1} \leftarrow$ {new connected components}

End

return F^{T+1}

(Sample nnz of $B \cdot \mathbf{1}_u$?)

ℓ_0-samplers

Definition

A δ-error ℓ_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

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

ℓ_0-samplers

Definition

A δ-error ℓ_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

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

Informally: sample a uniformly random element, output FAIL or just garbage with probability at most δ

ℓ_0-samplers

Definition

A δ-error ℓ_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

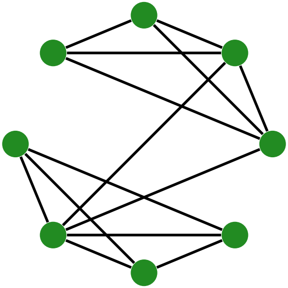
$$||J - UNIF_{\text{supp}(x)}||_{TVD} \leq \delta.$$

Informally: sample a uniformly random element, output FAIL or just garbage with probability at most δ

Recent constructions of ℓ_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

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



$F^0 \leftarrow \emptyset$

$C^0 \leftarrow V$ $\quad\triangleright$ current connected components

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

  For each $u \in C^t$

    Choose an edge in $\delta(u)$

  End For

 $F^{t+1} \leftarrow F^t \cup$ {spanning forest on selected edges}

 $C^{t+1} \leftarrow$ {new connected components}

End

return F^{T+1}

$S^1, \ldots, S^T \leftarrow \ell_0$-samplers

Maintain $S^1 B, \ldots, S^T B$

Run $Dec(S^t B \cdot \mathbf{1}_u)$

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 ℓ_0 samplers

- CountSketch (recap and proofs)
- Graph streaming
- Connectivity via sketching
- **Designing ℓ_0 samplers**

ℓ_0-samplers

Definition

A δ-error ℓ_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

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

ℓ_0-samplers

Definition

A δ-error ℓ_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

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

Informally: sample a uniformly random element, output FAIL or just garbage with probability at most δ

ℓ_0-samplers

Definition

A δ-error ℓ_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

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

Informally: sample a uniformly random element, output FAIL or just garbage with probability at most δ

Recent constructions of ℓ_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

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=number of occurrences of i)

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

1 2 3 4 5 6 7 8 9 10

ℓ_p sampling problem

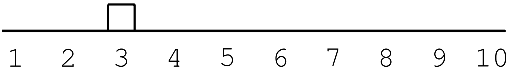
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3

ℓ_p sampling problem

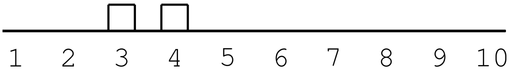
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4

ℓ_p sampling problem

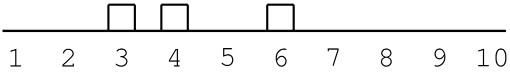
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6

ℓ_p sampling problem

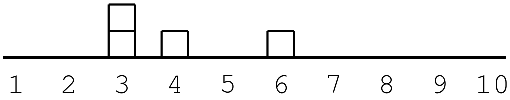
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3

ℓ_p sampling problem

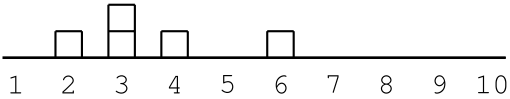
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3 2

ℓ_p sampling problem

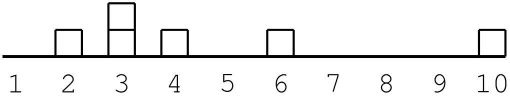
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3 2 10

ℓ_p sampling problem

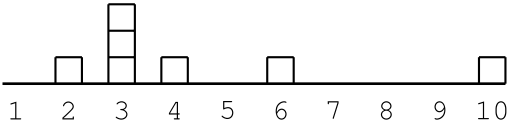
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3 2 10 3

ℓ_p sampling problem

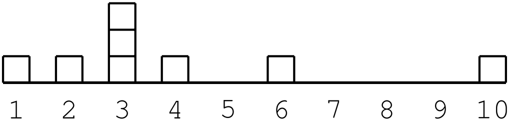
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3 2 10 3 1

ℓ_p sampling problem

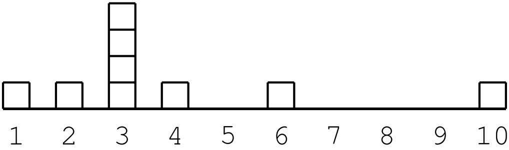
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3 2 10 3 1 3

ℓ_p sampling problem

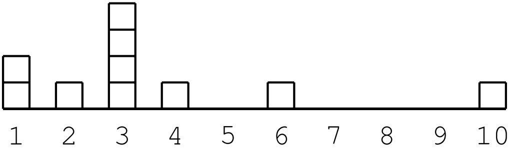
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3 2 10 3 1 3 1

ℓ_p sampling problem

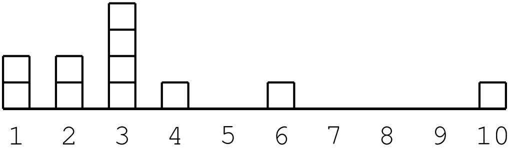
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



3 4 6 3 2 10 3 1 3 1 2

ℓ_p sampling problem

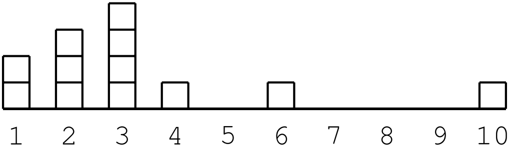
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

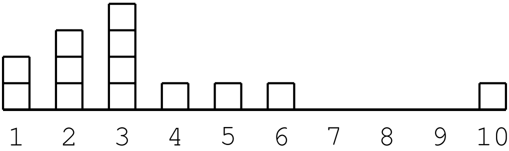
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

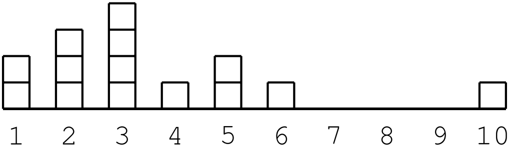
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

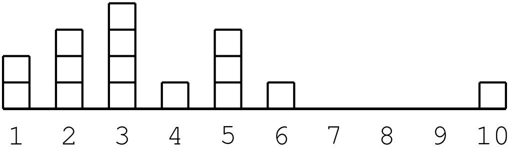
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

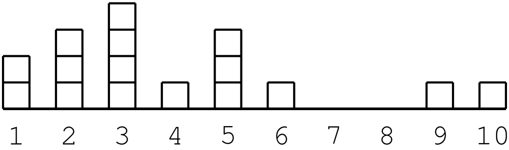
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

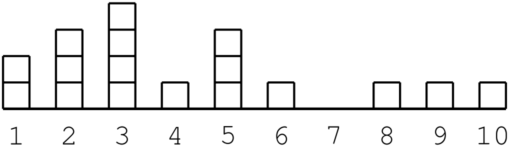
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

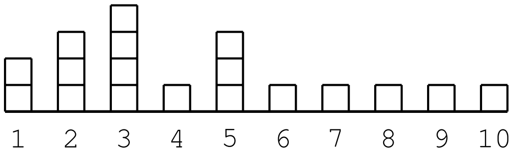
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

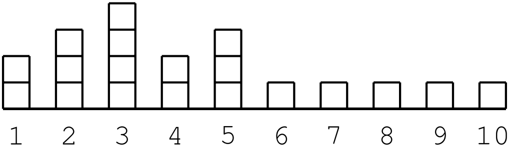
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

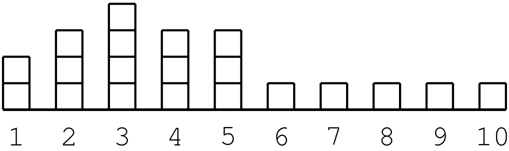
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

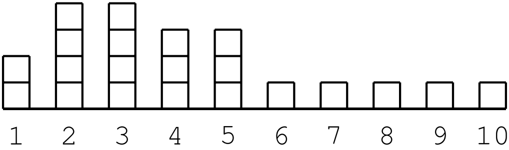
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

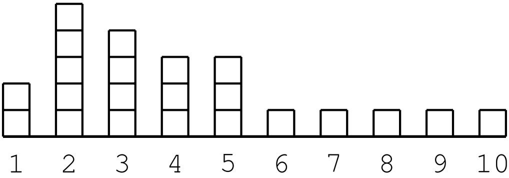
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

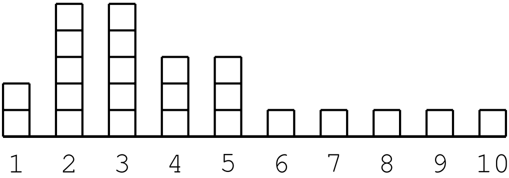
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

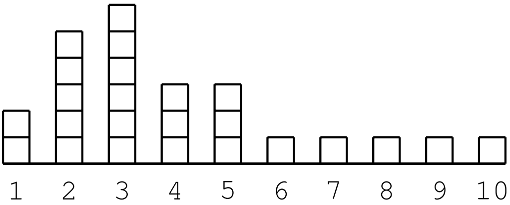
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_p sampling problem

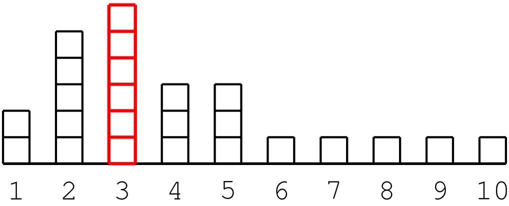
- Single pass over the data: $i_1, i_2, \ldots, i_N$

  Assume N is known

- Output item i with probability $\sim f_i^p$

  (f_i=number of occurrences of i)

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



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

ℓ_0 sampler construction (sketch)

Main idea:

- if x is 1-sparse (has a single nonzero), can recover x using few rows

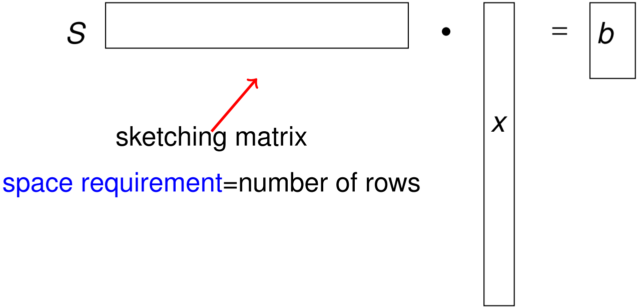
ℓ_0 sampler construction (sketch)

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 $\mathrm{supp}(x) = 1$ can be recovered with probability 1?



Recovering 1-sparse signals

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

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

sketching matrix

space requirement=number of rows

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

Compute

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

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

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 $||x||_0$.

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 $i \in [n]$

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

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 $i \in [n]$

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

Then

$$\mathbf{Pr}\left[|\mathrm{supp}(x) \cap S| = 1\right] = \Omega(1)$$

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 $i \in [n]$

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

Then

$$\mathbf{Pr}\left[|\mathrm{supp}(x) \cap S| = 1\right] = \Omega(1)$$

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 ℓ_0-samplers

Definition

A δ-error ℓ_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 δ-error ℓ_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)