

# CMPSCI 711: “Really Advanced Algorithms”

## Lecture 19 – Summary of Research Papers

Andrew McGregor

# Basic Data Stream Model

## Definition

Input is a length  $m$  list  $\langle a_1, \dots, a_m \rangle$  where  $a_i \in [n]$ . You can only access the elements sequentially and have memory that is sub-linear in  $m$  and  $n$ .

# Basic Data Stream Model

## Definition

Input is a length  $m$  list  $\langle a_1, \dots, a_m \rangle$  where  $a_i \in [n]$ . You can only access the elements sequentially and have memory that is sub-linear in  $m$  and  $n$ .

## Problem

Let  $f_i = |\{j : a_j = i\}|$ .

1. Compute approximations to all  $f_i$ .
2. Estimate  $F_k = \sum_{i \in [n]} f_i^k$ .
3. Estimate  $F_0 = \sum_{i \in [n]} f_i^0$ , i.e., the number of distinct items.

# Outline

Estimating Point Queries

Estimating  $F_k$

Estimating  $F_2$

Estimating  $F_0$

$F_0$  lower bound

$\ell_p$  and Stable Distributions

# Estimating Point Queries

## Problem

*Estimate all  $f_i$  upto  $\pm\epsilon m$  error.*

# Estimating Point Queries

## Problem

Estimate all  $f_i$  upto  $\pm \epsilon m$  error.

## Algorithm

1. Pick  $d$  2-wise hash functions  $h_1, \dots, h_d : [n] \rightarrow [w]$ .
2. Compute  $T_{i,j} = \sum_{\ell: h_i(\ell)=j} f_\ell$
3. Let  $\hat{f}_\ell = \min_{i: h_i(\ell)=j} T_{i,j}$

# Estimating Point Queries

## Problem

Estimate all  $f_i$  upto  $\pm \epsilon m$  error.

## Algorithm

1. Pick  $d$  2-wise hash functions  $h_1, \dots, h_d : [n] \rightarrow [w]$ .
2. Compute  $T_{i,j} = \sum_{\ell: h_i(\ell)=j} f_\ell$
3. Let  $\hat{f}_\ell = \min_{i: h_i(\ell)=j} T_{i,j}$

## Theorem

There exists a  $O(\epsilon^{-1} \log(n/\delta))$  space algorithm that returns  $(\hat{f}_1, \dots, \hat{f}_n)$  where  $f_i \leq \hat{f}_i \leq f_i + \epsilon m$  with probability  $1 - \delta$ .

# Estimating Point Queries

## Problem

Estimate all  $f_i$  upto  $\pm \epsilon m$  error.

## Algorithm

1. Pick  $d$  2-wise hash functions  $h_1, \dots, h_d : [n] \rightarrow [w]$ .
2. Compute  $T_{i,j} = \sum_{\ell: h_i(\ell)=j} f_\ell$
3. Let  $\hat{f}_\ell = \min_{i: h_i(\ell)=j} T_{i,j}$

## Theorem

There exists a  $O(\epsilon^{-1} \log(n/\delta))$  space algorithm that returns  $(\hat{f}_1, \dots, \hat{f}_n)$  where  $f_i \leq \hat{f}_i \leq f_i + \epsilon m$  with probability  $1 - \delta$ .

## Proof.

1. Set  $w = 2/\epsilon$ :  $\mathbb{E}[T_{i,j} | h_i(\ell) = j] \leq f_\ell + \epsilon m/2$ .

# Estimating Point Queries

## Problem

Estimate all  $f_i$  upto  $\pm \epsilon m$  error.

## Algorithm

1. Pick  $d$  2-wise hash functions  $h_1, \dots, h_d : [n] \rightarrow [w]$ .
2. Compute  $T_{i,j} = \sum_{\ell: h_i(\ell)=j} f_\ell$
3. Let  $\hat{f}_\ell = \min_{i: h_i(\ell)=j} T_{i,j}$

## Theorem

There exists a  $O(\epsilon^{-1} \log(n/\delta))$  space algorithm that returns  $(\hat{f}_1, \dots, \hat{f}_n)$  where  $f_i \leq \hat{f}_i \leq f_i + \epsilon m$  with probability  $1 - \delta$ .

## Proof.

1. Set  $w = 2/\epsilon$ :  $\mathbb{E}[T_{i,j} | h_i(\ell) = j] \leq f_\ell + \epsilon m/2$ .
2. Set  $d = \log(n/\delta)$ :  $\mathbb{P}[\min_{i: h_i(\ell)=j} T_{i,j} > f_\ell + \epsilon m] \leq \delta/n$ .



# Outline

Estimating Point Queries

Estimating  $F_k$

Estimating  $F_2$

Estimating  $F_0$

$F_0$  lower bound

$\ell_p$  and Stable Distributions

# Estimating $F_k$

Problem

*Estimate*  $F_k = \sum_i f_i^k$ .

# Estimating $F_k$

## Problem

Estimate  $F_k = \sum_i f_i^k$ .

## Algorithm

1. Pick  $j$  uniformly from  $[m]$
2. Compute  $r = |\{j \leq j' \leq m : a_{j'} = a_j\}|$
3. Output  $X = mr^k - m(r-1)^k$

# Estimating $F_k$

## Problem

Estimate  $F_k = \sum_i f_i^k$ .

## Algorithm

1. Pick  $j$  uniformly from  $[m]$
2. Compute  $r = |\{j \leq j' \leq m : a_{j'} = a_j\}|$
3. Output  $X = mr^k - m(r - 1)^k$

## Theorem

There exists a  $O(kn^{1-1/k}\epsilon^{-2} \log(1/\delta))$  space algorithm that returns a  $(1 + \epsilon)$  factor approximation to  $F_k$  with probability  $1 - \delta$ .

# Estimating $F_k$

## Problem

Estimate  $F_k = \sum_i f_i^k$ .

## Algorithm

1. Pick  $j$  uniformly from  $[m]$
2. Compute  $r = |\{j \leq j' \leq m : a_{j'} = a_j\}|$
3. Output  $X = mr^k - m(r-1)^k$

## Theorem

There exists a  $O(kn^{1-1/k}\epsilon^{-2} \log(1/\delta))$  space algorithm that returns a  $(1 + \epsilon)$  factor approximation to  $F_k$  with probability  $1 - \delta$ .

## Proof.

$\mathbb{E}[X] = F_k$  and  $\mathbb{V}[X] \leq kn^{1-1/k}F_k^2$ . Run copies of algorithm. Average results appropriately and apply Chernoff/Chebyshev.  $\square$

# Outline

Estimating Point Queries

Estimating  $F_k$

Estimating  $F_2$

Estimating  $F_0$

$F_0$  lower bound

$\ell_p$  and Stable Distributions

## Estimating $F_2$

Problem

*Estimate  $F_2 = \sum_i f_i^2$ .*

# Estimating $F_2$

## Problem

Estimate  $F_2 = \sum_i f_i^2$ .

## Algorithm

1. Pick  $(x_1, \dots, x_n) \in \{-1, 1\}^n$  with  $x_i$  and  $x_j$  independent
2. Compute  $\sum_{i \in [n]} x_i f_i$
3. Return  $X = (\sum_{i \in [n]} x_i f_i)^2$

# Estimating $F_2$

## Problem

Estimate  $F_2 = \sum_i f_i^2$ .

## Algorithm

1. Pick  $(x_1, \dots, x_n) \in \{-1, 1\}^n$  with  $x_i$  and  $x_j$  independent
2. Compute  $\sum_{i \in [n]} x_i f_i$
3. Return  $X = (\sum_{i \in [n]} x_i f_i)^2$

## Theorem

There exists a  $O(\epsilon^{-2} \log(1/\delta)(\log m + \log n))$  space algorithm that returns a  $(1 + \epsilon)$  factor approximation to  $F_2$  with probability  $1 - \delta$ .

# Estimating $F_2$

## Problem

Estimate  $F_2 = \sum_i f_i^2$ .

## Algorithm

1. Pick  $(x_1, \dots, x_n) \in \{-1, 1\}^n$  with  $x_i$  and  $x_j$  independent
2. Compute  $\sum_{i \in [n]} x_i f_i$
3. Return  $X = (\sum_{i \in [n]} x_i f_i)^2$

## Theorem

There exists a  $O(\epsilon^{-2} \log(1/\delta)(\log m + \log n))$  space algorithm that returns a  $(1 + \epsilon)$  factor approximation to  $F_2$  with probability  $1 - \delta$ .

## Proof.

$\mathbb{E}[X] = F_2$  and  $\mathbb{V}[X] \leq 2F_2^2$ . Run  $O(\epsilon^{-2} \log(1/\delta))$  copies. Average results appropriately and apply Chernoff/Chebyshev.  $\square$

# Relationship to Johnson Lindenstrauss

## Algorithm

1. Let  $Y = \frac{1}{\|X\|_2}(x_1, \dots, x_n)$  be such that each  $x_i$  are independent  $N(0, 1)$  variables where  $\|X\|_2 = (x_1^2 + \dots + x_n^2)^{1/2}$
2. Compute  $h(x) = \sum_{i \in [n]} x_i f_i$
3. Repeat  $d$  times to get  $(h(x)_1, \dots, h(x)_d)$

# Relationship to Johnson Lindenstrauss

## Algorithm

1. Let  $Y = \frac{1}{\|X\|_2}(x_1, \dots, x_n)$  be such that each  $x_i$  are independent  $N(0, 1)$  variables where  $\|X\|_2 = (x_1^2 + \dots + x_n^2)^{1/2}$
2. Compute  $h(x) = \sum_{i \in [n]} x_i f_i$
3. Repeat  $d$  times to get  $(h(x)_1, \dots, h(x)_d)$

## Theorem

For any  $p$  vectors in Euclidean space  $v_1, \dots, v_p \in \mathbb{R}^n$ , there exists a map  $h : \mathbb{R}^n \rightarrow \mathbb{R}^d$  such that

$$(1 - \epsilon) \|h(v_i) - h(v_j)\|_2 \leq \|v_i - v_j\|_2 \leq (1 + \epsilon) \|h(v_i) - h(v_j)\|_2$$

where  $d = O(\epsilon^{-2} \log n)$ .

# Outline

Estimating Point Queries

Estimating  $F_k$

Estimating  $F_2$

Estimating  $F_0$

$F_0$  lower bound

$\ell_p$  and Stable Distributions

# Problem and First Attempt

## Problem

*Estimate  $F_0 = \sum_i f_i^0$ , i.e., number of distinct values in the stream.*

# Problem and First Attempt

## Problem

Estimate  $F_0 = \sum_i f_i^0$ , i.e., number of distinct values in the stream.

## Algorithm

1. Pick a random hash function  $h : \{1, \dots, n\} \rightarrow [0, 1]$
2. As the stream arrives, compute  $h(a_i)$ .
3. Keep track of minimum value  $v$  seen so far.
4. Output  $1/v - 1$  as estimate for  $F_0$

# Problem and First Attempt

## Problem

Estimate  $F_0 = \sum_i f_i^0$ , i.e., number of distinct values in the stream.

## Algorithm

1. Pick a random hash function  $h : \{1, \dots, n\} \rightarrow [0, 1]$
2. As the stream arrives, compute  $h(a_i)$ .
3. Keep track of minimum value  $v$  seen so far.
4. Output  $1/v - 1$  as estimate for  $F_0$

## Issues:

1. If hash function is totally random it may be too big to store:  
Assume  $h$  is pairwise independent.
2. Precision issues: Hash into  $[n^3]$  rather than  $[0, 1]$
3. The above estimator varies too much: Track  $t$ -th smallest

# The Algorithm

## Algorithm

1. Pick a random 2-wise hash function  $h : [n] \rightarrow [n^3]$
2. Compute  $t$ -th smallest hash values  $v_1, \dots, v_t$ .
3. If  $F_0 \leq t$ , output  $\hat{F}_0 = F_0$ .
4. If  $F_0 > t$ , output  $\hat{F}_0 = tn^3/v_t$

# The Algorithm

## Algorithm

1. Pick a random 2-wise hash function  $h : [n] \rightarrow [n^3]$
2. Compute  $t$ -th smallest hash values  $v_1, \dots, v_t$ .
3. If  $F_0 \leq t$ , output  $\hat{F}_0 = F_0$ .
4. If  $F_0 > t$ , output  $\hat{F}_0 = tn^3/v_t$

## Theorem

If  $t = 20/\epsilon^2$  then  $\mathbb{P} \left[ |\hat{F}_0 - F_0| \geq \epsilon F_0 \right] \leq 2/5$ .

# The Algorithm

## Algorithm

1. Pick a random 2-wise hash function  $h : [n] \rightarrow [n^3]$
2. Compute  $t$ -th smallest hash values  $v_1, \dots, v_t$ .
3. If  $F_0 \leq t$ , output  $\hat{F}_0 = F_0$ .
4. If  $F_0 > t$ , output  $\hat{F}_0 = tn^3/v_t$

## Theorem

If  $t = 20/\epsilon^2$  then  $\mathbb{P} \left[ |\hat{F}_0 - F_0| \geq \epsilon F_0 \right] \leq 2/5$ .

## Corollary

We can  $(1 + \epsilon)$  approximate  $F_0$  with probability  $1 - \delta$  in only  $O(\epsilon^{-2} \log(1/\delta) \log n)$  space.

## Proof of Theorem

1. Assuming  $F_0 > t$ , we get good estimate if

$$(1 - \epsilon)F_0 \leq tn^3/v_t \leq (1 + \epsilon)F_0$$

or equivalently  $\frac{tn^3}{(1+\epsilon)F_0} \leq v_t \leq \frac{tn^3}{(1-\epsilon)F_0}$

## Proof of Theorem

1. Assuming  $F_0 > t$ , we get good estimate if

$$(1 - \epsilon)F_0 \leq tn^3/v_t \leq (1 + \epsilon)F_0$$

or equivalently  $\frac{tn^3}{(1+\epsilon)F_0} \leq v_t \leq \frac{tn^3}{(1-\epsilon)F_0}$

2. Consider  $\mathbb{P} \left[ \frac{tn^3}{(1+\epsilon)F_0} \leq v_t \right] = \mathbb{P} [Y < t]$  where  $Y$  is the number of items hashing to under  $\frac{tn^3}{(1+\epsilon)F_0}$ . Other case is similar.

## Proof of Theorem

1. Assuming  $F_0 > t$ , we get good estimate if

$$(1 - \epsilon)F_0 \leq tn^3/v_t \leq (1 + \epsilon)F_0$$

or equivalently  $\frac{tn^3}{(1+\epsilon)F_0} \leq v_t \leq \frac{tn^3}{(1-\epsilon)F_0}$

2. Consider  $\mathbb{P} \left[ \frac{tn^3}{(1+\epsilon)F_0} \leq v_t \right] = \mathbb{P} [Y < t]$  where  $Y$  is the number of items hashing to under  $\frac{tn^3}{(1+\epsilon)F_0}$ . Other case is similar.
3. For each  $a_i$ ,

$$\mathbb{P} \left[ h(a_i) \leq \frac{tn^3}{(1 + \epsilon)F_0} \right] \leq \frac{t}{(1 + \epsilon)F_0}$$

## Proof of Theorem

1. Assuming  $F_0 > t$ , we get good estimate if

$$(1 - \epsilon)F_0 \leq tn^3/v_t \leq (1 + \epsilon)F_0$$

or equivalently  $\frac{tn^3}{(1+\epsilon)F_0} \leq v_t \leq \frac{tn^3}{(1-\epsilon)F_0}$

2. Consider  $\mathbb{P} \left[ \frac{tn^3}{(1+\epsilon)F_0} \leq v_t \right] = \mathbb{P} [Y < t]$  where  $Y$  is the number of items hashing to under  $\frac{tn^3}{(1+\epsilon)F_0}$ . Other case is similar.
3. For each  $a_i$ ,

$$\mathbb{P} \left[ h(a_i) \leq \frac{tn^3}{(1 + \epsilon)F_0} \right] \leq \frac{t}{(1 + \epsilon)F_0}$$

4. By linearity of expectation,  $\mathbb{E}[Y] = t/(1 + \epsilon)$

## Proof of Theorem

1. Assuming  $F_0 > t$ , we get good estimate if

$$(1 - \epsilon)F_0 \leq tn^3/v_t \leq (1 + \epsilon)F_0$$

or equivalently  $\frac{tn^3}{(1+\epsilon)F_0} \leq v_t \leq \frac{tn^3}{(1-\epsilon)F_0}$

2. Consider  $\mathbb{P} \left[ \frac{tn^3}{(1+\epsilon)F_0} \leq v_t \right] = \mathbb{P} [Y < t]$  where  $Y$  is the number of items hashing to under  $\frac{tn^3}{(1+\epsilon)F_0}$ . Other case is similar.
3. For each  $a_i$ ,

$$\mathbb{P} \left[ h(a_i) \leq \frac{tn^3}{(1 + \epsilon)F_0} \right] \leq \frac{t}{(1 + \epsilon)F_0}$$

4. By linearity of expectation,  $\mathbb{E}[Y] = t/(1 + \epsilon)$
5. Since  $h$  is 2-wise independent  $\mathbb{V}[Y] \leq t/(1 + \epsilon)$

## Proof of Theorem

1. Assuming  $F_0 > t$ , we get good estimate if

$$(1 - \epsilon)F_0 \leq tn^3/v_t \leq (1 + \epsilon)F_0$$

or equivalently  $\frac{tn^3}{(1+\epsilon)F_0} \leq v_t \leq \frac{tn^3}{(1-\epsilon)F_0}$

2. Consider  $\mathbb{P}\left[\frac{tn^3}{(1+\epsilon)F_0} \leq v_t\right] = \mathbb{P}[Y < t]$  where  $Y$  is the number of items hashing to under  $\frac{tn^3}{(1+\epsilon)F_0}$ . Other case is similar.
3. For each  $a_i$ ,

$$\mathbb{P}\left[h(a_i) \leq \frac{tn^3}{(1+\epsilon)F_0}\right] \leq \frac{t}{(1+\epsilon)F_0}$$

4. By linearity of expectation,  $\mathbb{E}[Y] = t/(1 + \epsilon)$
5. Since  $h$  is 2-wise independent  $\mathbb{V}[Y] \leq t/(1 + \epsilon)$
6. By Chebyshev:  $\mathbb{P}[Y \geq t] \leq 1/5$  if  $t \geq 20\epsilon^{-2}$

# Outline

Estimating Point Queries

Estimating  $F_k$

Estimating  $F_2$

Estimating  $F_0$

$F_0$  lower bound

$\ell_p$  and Stable Distributions

## $F_0$ lower bound via communication complexity

### Theorem

*Any data stream algorithm that approximates  $F_0$  up to a  $(1 + \epsilon)$  factor with probability  $9/10$  needs  $\Omega(\epsilon^{-2})$  space.*

## $F_0$ lower bound via communication complexity

### Theorem

*Any data stream algorithm that approximates  $F_0$  up to a  $(1 + \epsilon)$  factor with probability  $9/10$  needs  $\Omega(\epsilon^{-2})$  space.*

### Lemma

*Alice has  $x \in \{0, 1\}^n$  and Bob has  $y \in \{0, 1\}^n$ . Alice needs to send Bob  $\Omega(n)$  bits if he is to estimate the hamming distance  $\Delta(x, y)$  up to  $\pm o(\sqrt{n})$  with probability  $9/10$ .*

## $F_0$ lower bound via communication complexity

### Theorem

*Any data stream algorithm that approximates  $F_0$  up to a  $(1 + \epsilon)$  factor with probability  $9/10$  needs  $\Omega(\epsilon^{-2})$  space.*

### Lemma

*Alice has  $x \in \{0, 1\}^n$  and Bob has  $y \in \{0, 1\}^n$ . Alice needs to send Bob  $\Omega(n)$  bits if he is to estimate the hamming distance  $\Delta(x, y)$  up to  $\pm o(\sqrt{n})$  with probability  $9/10$ .*

- ▶ Define a multi-set by  $S = \{i : x_i = 1\} \cup \{i : y_i = 1\}$

$$F_0(S) = |x|/2 + |y|_1/2 + \Delta(x, y)/2$$

- ▶  $o(\sqrt{n})$  estimate for  $F_0(S)$  gives  $o(\sqrt{n})$  estimate for  $\Delta(x, y)$
- ▶ If  $n = o(\epsilon^{-2})$ :  $(1 + \epsilon)$  factor approx of  $F_0(S)$  is  $o(\sqrt{n})$  approx.

## One-Pass Lower Bound (1/2)

- ▶ Reduction from INDEX problem: Alice knows  $z \in \{0, 1\}^t$  and Bob knows  $j \in [t]$ . Let's assume  $|z| = t/2$  and this is odd.

## One-Pass Lower Bound (1/2)

- ▶ Reduction from INDEX problem: Alice knows  $z \in \{0, 1\}^t$  and Bob knows  $j \in [t]$ . Let's assume  $|z| = t/2$  and this is odd.
- ▶ Alice and Bob pick  $r \in_R \{-1, 1\}^n$  using public random bits.

## One-Pass Lower Bound (1/2)

- ▶ Reduction from INDEX problem: Alice knows  $z \in \{0, 1\}^t$  and Bob knows  $j \in [t]$ . Let's assume  $|z| = t/2$  and this is odd.
- ▶ Alice and Bob pick  $r \in_R \{-1, 1\}^n$  using public random bits.
- ▶ Alice computes  $\text{sign}(r \cdot z)$  and Bob computes  $\text{sign}(r_j)$

## One-Pass Lower Bound (1/2)

- ▶ Reduction from INDEX problem: Alice knows  $z \in \{0, 1\}^t$  and Bob knows  $j \in [t]$ . Let's assume  $|z| = t/2$  and this is odd.
- ▶ Alice and Bob pick  $r \in_R \{-1, 1\}^n$  using public random bits.
- ▶ Alice computes  $\text{sign}(r \cdot z)$  and Bob computes  $\text{sign}(r_j)$
- ▶ **Claim:** For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

## One-Pass Lower Bound (1/2)

- ▶ Reduction from INDEX problem: Alice knows  $z \in \{0, 1\}^t$  and Bob knows  $j \in [t]$ . Let's assume  $|z| = t/2$  and this is odd.
- ▶ Alice and Bob pick  $r \in_R \{-1, 1\}^n$  using public random bits.
- ▶ Alice computes  $\text{sign}(r \cdot z)$  and Bob computes  $\text{sign}(r_j)$
- ▶ **Claim:** For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

- ▶ Repeat  $n = O(t)$  times to construct

$$x_i = I[\text{sign}(r \cdot z) = +] \quad \text{and} \quad y_i = I[\text{sign}(r_j) = +]$$

## One-Pass Lower Bound (1/2)

- ▶ Reduction from INDEX problem: Alice knows  $z \in \{0, 1\}^t$  and Bob knows  $j \in [t]$ . Let's assume  $|z| = t/2$  and this is odd.
- ▶ Alice and Bob pick  $r \in_R \{-1, 1\}^n$  using public random bits.
- ▶ Alice computes  $\text{sign}(r \cdot z)$  and Bob computes  $\text{sign}(r_j)$
- ▶ **Claim:** For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

- ▶ Repeat  $n = O(t)$  times to construct

$$x_i = I[\text{sign}(r \cdot z) = +] \quad \text{and} \quad y_i = I[\text{sign}(r_j) = +]$$

- ▶ With probability  $9/10$ , for some constants  $c_1 < c_2$ ,

$$z_j = 0 \Rightarrow \Delta(x, y) \geq n/2 - c_1\sqrt{n}$$

$$z_j = 1 \Rightarrow \Delta(x, y) \leq n/2 - c_2\sqrt{n}$$

## One-Pass Lower Bound (2/2)

### Claim

For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

## One-Pass Lower Bound (2/2)

### Claim

For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

- ▶ If  $z_j = 0$  then  $\text{sign}(r \cdot z)$  and  $\text{sign}(r_j)$  are independent.

## One-Pass Lower Bound (2/2)

### Claim

For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

- ▶ If  $z_j = 0$  then  $\text{sign}(r \cdot z)$  and  $\text{sign}(r_j)$  are independent.
- ▶ If  $z_j = 1$ , let  $s = r \cdot z - r_j$ ,  $A = \{\text{sign}(r \cdot z) = \text{sign}(r_j)\}$ :

## One-Pass Lower Bound (2/2)

### Claim

For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

- ▶ If  $z_j = 0$  then  $\text{sign}(r \cdot z)$  and  $\text{sign}(r_j)$  are independent.
- ▶ If  $z_j = 1$ , let  $s = r \cdot z - r_j$ ,  $A = \{\text{sign}(r \cdot z) = \text{sign}(r_j)\}$ :

$$\mathbb{P}[A] = \mathbb{P}[A|s = 0] \mathbb{P}[s = 0] + \mathbb{P}[A|s \neq 0] \mathbb{P}[s \neq 0]$$

## One-Pass Lower Bound (2/2)

### Claim

For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

- ▶ If  $z_j = 0$  then  $\text{sign}(r \cdot z)$  and  $\text{sign}(r_j)$  are independent.
- ▶ If  $z_j = 1$ , let  $s = r \cdot z - r_j$ ,  $A = \{\text{sign}(r \cdot z) = \text{sign}(r_j)\}$ :

$$\mathbb{P}[A] = \mathbb{P}[A|s = 0] \mathbb{P}[s = 0] + \mathbb{P}[A|s \neq 0] \mathbb{P}[s \neq 0]$$

$$\mathbb{P}[A|s = 0] = 1 \text{ and } \mathbb{P}[A|s \neq 0] = 1/2$$

## One-Pass Lower Bound (2/2)

### Claim

For some constant  $c > 0$ ,

$$\mathbb{P}[\text{sign}(r \cdot z) = \text{sign}(r_j)] = \begin{cases} 1/2 & \text{if } z_j = 0 \\ 1/2 + c/\sqrt{t} & \text{if } z_j = 1 \end{cases}$$

- ▶ If  $z_j = 0$  then  $\text{sign}(r \cdot z)$  and  $\text{sign}(r_j)$  are independent.
- ▶ If  $z_j = 1$ , let  $s = r \cdot z - r_j$ ,  $A = \{\text{sign}(r \cdot z) = \text{sign}(r_j)\}$ :

$$\mathbb{P}[A] = \mathbb{P}[A|s = 0] \mathbb{P}[s = 0] + \mathbb{P}[A|s \neq 0] \mathbb{P}[s \neq 0]$$

$$\mathbb{P}[A|s = 0] = 1 \text{ and } \mathbb{P}[A|s \neq 0] = 1/2$$

$$\mathbb{P}[s = 0] = 2c/\sqrt{n} \text{ for some constant } c > 0$$

# Outline

Estimating Point Queries

Estimating  $F_k$

Estimating  $F_2$

Estimating  $F_0$

$F_0$  lower bound

$\ell_p$  and Stable Distributions

## Problem Definition

### Definition

Input is a length  $m$  list  $S = \langle a_1, \dots, a_m \rangle$  where

$$a_i = (j_i, \Delta_i) \in [n] \cup \{-M, \dots, +M\}$$

You can only access the elements sequentially and have memory that is sub-linear in  $m$  and  $n$ .

# Problem Definition

## Definition

Input is a length  $m$  list  $S = \langle a_1, \dots, a_m \rangle$  where

$$a_i = (j_i, \Delta_i) \in [n] \cup \{-M, \dots, +M\}$$

You can only access the elements sequentially and have memory that is sub-linear in  $m$  and  $n$ .

## Problem

Let  $v_j = \sum_{i:j_i=j} \Delta_i$ . Note that  $v_j$  can be negative. Estimate

$$\ell_p(v) = \|v\|_p = \left( \sum_j v_j^p \right)^{1/p}$$

## Theorem

Using only  $\tilde{O}(\epsilon^{-2} \log \delta^{-1})$  space we can find a  $(1 + \epsilon)$  error approx. with probability  $1 - \delta$

## $p$ -stable distribution

### Definition

A distribution  $D$  over  $\mathbb{R}$  is called  $p$ -stable, if for any  $v \in \mathbb{R}^n$  and i.i.d. variables  $X_1, \dots, X_n$  variables with distribution  $D$ , the random variable

$$\sum_i v_i X_i$$

has the same distribution as  $\|v\|_p X$  where  $X$  is a random variable with distribution  $D$ .

## $p$ -stable distribution

### Definition

A distribution  $D$  over  $\mathbb{R}$  is called  $p$ -stable, if for any  $v \in \mathbb{R}^n$  and i.i.d. variables  $X_1, \dots, X_n$  variables with distribution  $D$ , the random variable

$$\sum_i v_i X_i$$

has the same distribution as  $\|v\|_p X$  where  $X$  is a random variable with distribution  $D$ .

### Example

The Normal(0,1) distribution is 2-stable:

$$\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Cauchy is 1-stable:

$$\frac{1}{\pi} \frac{1}{1+x^2}$$

# Ideal Algorithm for $p = 1$

## Algorithm

- ▶ Let  $A$  be a  $k \times n$  matrix where each  $A_{ij}$  is an independent Cauchy distribution. Let  $A^j$  be the  $j$  column of  $A$ .
- ▶ Let  $c$  be length- $k$  zero vector. For each stream element  $(j, \Delta)$ :

$$c \leftarrow c + \Delta A^j$$

- ▶ Return  $\text{median}(|c_1|, \dots, |c_k|)$

# Ideal Algorithm for $p = 1$

## Algorithm

- ▶ Let  $A$  be a  $k \times n$  matrix where each  $A_{i;j}$  is an independent Cauchy distribution. Let  $A^j$  be the  $j$  column of  $A$ .
- ▶ Let  $c$  be length- $k$  zero vector. For each stream element  $(j, \Delta)$ :

$$c \leftarrow c + \Delta A^j$$

- ▶ Return  $\text{median}(|c_1|, \dots, |c_k|)$

## Lemma

- ▶  $c = Av$  but algorithm only needs to maintain length  $k$  vector.
- ▶ Each  $|c_i|$  independently distributed as  $\|v\|_p |X|$  where  $X$  is a Cauchy random variable.

## Details

### Lemma

If  $X$  is Cauchy distributed,  $\text{median}(|X|) = 1$ .

### Proof.

Follows since  $\tan(\pi/4) = 1$  and

$$F_X(z) := \mathbb{P}[X \leq z] = \int_0^z \frac{2}{\pi} \frac{1}{1+x^2} = \frac{2}{\pi} \arctan(z)$$



## Details

### Lemma

If  $X$  is Cauchy distributed,  $\text{median}(|X|) = 1$ .

### Proof.

Follows since  $\tan(\pi/4) = 1$  and

$$F_X(z) := \mathbb{P}[X \leq z] = \int_0^z \frac{2}{\pi} \frac{1}{1+x^2} = \frac{2}{\pi} \arctan(z)$$

□

### Lemma

Let  $Y_1, \dots, Y_k$  i.i.d. and  $X = \text{median}(Y_1, \dots, Y_k)$ .

$$\mathbb{P}[F_X(z) \in [1/2 - \epsilon, 1/2 + \epsilon]] \geq 1 - \delta \quad \text{if } k = 9\epsilon^{-2} \log \delta^{-1}.$$

## Details

### Lemma

If  $X$  is Cauchy distributed,  $\text{median}(|X|) = 1$ .

### Proof.

Follows since  $\tan(\pi/4) = 1$  and

$$F_X(z) := \mathbb{P}[X \leq z] = \int_0^z \frac{2}{\pi} \frac{1}{1+x^2} = \frac{2}{\pi} \arctan(z)$$

□

### Lemma

Let  $Y_1, \dots, Y_k$  i.i.d. and  $X = \text{median}(Y_1, \dots, Y_k)$ .

$$\mathbb{P}[F_X(z) \in [1/2 - \epsilon, 1/2 + \epsilon]] \geq 1 - \delta \quad \text{if } k = 9\epsilon^{-2} \log \delta^{-1}.$$

### Lemma

For small  $\epsilon$  sufficiently small. Let  $X$  be Cauchy distributed and  $z$  satisfy  $F_{|X|}(z) \in [1/2 - \epsilon, 1/2 + \epsilon]$ . Then,  $z \in [1 - 4\epsilon, 1 + 4\epsilon]$ .

Thanks!