Lectures on Streaming Algorithms

Qin Zhang

May 13, 2011

1 Models and Motivations

1.1 Models

- *Standard stream model*: m elements from universe of size n, come one by one.

  Goal: compute a function of stream.

  Constraints: (1) Limited space (working memory), sublinear in n and m. (2) Access data sequentially. (3) Process each element quickly.

  Two variants: (1) Cash register (2) Turnstile.

- *Graph stream*: elements of the stream are edges. Let n be the number of vertices. We have $m = O(n^2)$. To be able to handle interesting functions, we typically allow space $O(n \log^{O(1)} n)$.

- *Sliding windows*: (1) Sequence-based sliding window: only interested in the last w elements. (2) Time-based sliding window: only interested in elements that arrive in the last w time steps.

1.2 Motivations

- *Practical Appeal*: Network monitoring (#distinct(sourse, destination) pairs, top-k heaviest flows), query planning, I/O efficiency for massive data, sensor networks aggregation ....

- *Theoretical Appeal*: Easy to state problems but hard to solve. Links to communication complexity, compressed sensing, embeddings, pseudo-random generators, approximation ....

2 Samplings

2.1 Motivation

Sampling is a general technique for tackling massive amounts of data. For example, to compute the median packet size of some IP packets, we could just sample some and use the median of the sample as an estimate for the true median. Statistical arguments relate the size of the sample to the accuracy of the estimate.

2.2 Standard Sampling

- Q: Sampling from a stream of unknown length
- A: Reservoir Sampling
  1. Initially $s = x_1$
  2. On seeing the t-th element, set s to be x_t with probability $1/t$.
- Q: Sampling from a sliding window (last w items).
- A: Algorithm:
  1. For each x_i we pick a random value $v_i \in (0, 1)$.
  2. In a window $\langle x_{j-w+1}; \ldots; x_j \rangle$ return value x_i with smallest v_i.

     How? Maintain a set of all elements in the sliding window whose value is minimal among subsequent values.

  Cost: $O(\log n \log w)$ space expected.
- Q: Can we make space optimal ($O(\log n)$) and worst case?
- A: Yes. See *Optimal sampling from sliding windows* by Braverman et. al. PODS 2009.

2.3 AMS Sampling

- Q: Want to compute some function $\sum_{i \in [n]} g(f_i)$ where $f_i = |\{j : x_j = i\}|$ is the frequency of i and g is an arbitrary function with $g(0) = 0$.
- A: Algorithm:
  1. Pick J uniformly at random from $[m]$.
  2. Compute $R = |\{j \geq J : a_j = a_J\}|$.
  3. Let $X = m(g(R) - g(R - 1))$

  Cost: $O(\log m + \log n)$ space (1-run) guarantees $E[X] = \sum_{i \in [n]} g(f_i)$.

2.3.1 Analysis

$$\begin{aligned}
\mathbf{E}[X] &= \sum_i \mathbf{E}[X \mid a_J = i]\mathbf{Pr}[a_J = i] \\
&= \sum_i \mathbf{E}[g(R) - g(R-1) \mid a_J = i] f_i \\
&= \sum_i \left(\frac{g(1) - g(0)}{f_i} + \ldots + \frac{g(f_i) - g(f_i - 1)}{f_i} \right) f_i \\
&= \sum_i g(f_i)
\end{aligned}$$

3 Heavy Hitters

- Q: Let $f_i = |\{j : a_j = i\}|$. How to maintain for each $i \in [n]$ an estimate $\tilde{f}_i$ s.t. $f_i \leq \tilde{f}_i \leq f_i + \epsilon m$?
- A: Algorithm [CR-Precis]: Let $p_1, p_2, \ldots, p_t$ be the first t prime numbers and define hash functions $h_j(x) = x \bmod p_j$. Next, maintain $p_t t = O(t^2 \log t)$ counters $c_{i,j}$ which are initially 0. When a new item e comes, increment each of $c_{j,h_j(e)}$ for $j \in [t]$.

  At the end, we set $\tilde{f}_i = \min_j c_{j,h_j(i)}$.

  Cost: set $t = O(\epsilon^{-1} \log n)$ and the space cost is $O(\epsilon^{-2} \log^2 n (\log 1/\epsilon + \log \log n))$.
- Q: Can we do better?
- A: Sure. Algorithm[Space-saving, by Metwally et. al. An integrated efficient solution for computing frequent and top-k elements in data streams. TODS 2006]: We construct an array with $1/\epsilon$ cells. Each cell is of the form $(e, f(e))$, where $f(e)$ denotes the estimated frequency of the item e.

  When a new item e comes, we have two cases.

  1. If e is not in the array, we create a new tuple $(e, \texttt{MIN} + 1)$ where $\texttt{MIN} = \min\{f(e_i) : e_i \text{ is in the array}\}$. We always keep the array sorted according to $f(e_i)$, and then MIN is just the estimated frequency of the last item. If the length array is larger than $1/\epsilon$, we delete the last tuple.
  2. If e is already in the array. We just increment $f(e)$ by 1 and reinsert the $(e, f(e))$ into the array.

  Cost: $O(1/\epsilon)$.

- A: So what is the merit of the first (cumbersome) algorithm?
- Q: It is useful since it is an *linear mapping*; it can handle *deletions*.

Linear mapping is one technique to perform *sketching*. The basic idea of sketching is to apply a linear projection "on the fly" that takes high-dimensional data to a smaller dimensional space. Post-process lower dimensional image to estimate the quantities of interest. Since it is a linear mapping, it can support "deletions".

3.1 Analysis of CR-Precis

For any $i' \neq i$, there are at most $\log n$ functions h_j under which i, i' collide, by the Chinese Remainder Theorem. Hence, we deduce that,

$$\sum_j c_{j,h_j(i)} \leq t f_i + \left(\sum_{i' \neq i} f_{i'} \right) \log n.$$

Thus $\tilde{f}_i \leq f_i + (m \log n)/t$.

4 Distinct Elements

- Q: Compute $d = |\{j : f_j > 0\}|$. The number of distinct elements.
- A: Algorithms [FM, Flajoet and Martin, FOCS 1983]: let $\text{zeros}(p) = \max\{i : 2^i \text{ divides } p\}$.

  1. Choose a random hash function $h : [n] \to [n]$ from a 2-universal family. Set $z = 0$.
  2. For each new coming item e, if $\text{zeros}(h(e)) > z$, then set $z = \text{zeros}(h(e))$;
  3. Output $2^{z+0.5}$.

  Cost: we should repeat it $\log(1/\delta)$ times (i.e. maintain $\log(1/\delta)$ hash functions in parallel) and then take the median to get a successful probability of $1 - \delta$. Thus need $O(\log(1/\delta) \log n)$ space.

- Q: Can we do better in terms of approximation ratio?
- A: Yes. Algorithm [BJKST]

  1. Choose a random hash function $h : [n] \to [n]$ from a 2-universal family. Set $z = 0, B = \emptyset$. Choose a secondary 2-universal hash function $g : [n] \to [O(\log n/\epsilon^2)]$.
  2. For each new coming item e, if $\text{zeros}(h(e)) \geq z$, then

(a) set $B = B \cup \{g(e)\}$;
(b) if $|B| > c/\epsilon^2$ then set $z = z + 1$ and rehash B.

3. Output $|B|\, 2^z$.

Cost: $O(\log n + 1/\epsilon^2 \cdot (\log 1/\epsilon) + \log\log n)$.

- Q: Can we do better?
- A: Yes. See the optimal algorithm in *An Optimal Algorithm for the Distinct Elements Problem* by Kane et. al. PODS 2010.

4.1 Analysis

[FM] Basic intuition is that we expect 1 out of the d distinct elements to hit $\text{zeros}(h(j)) > \log d$, and we don't expect any elements to hit $\text{zeros}(h(j)) \gg \log d$. For details, see [LN], page 9.

The median trick. Suppose we know that for a random variable X, with probability no more than 1/3 (could be replaced by any constant $< 1/2$), $X \leq a$; and with probability no more than 1/3, $X \geq b$. Now if we run k copies of the algorithm in parallel, thus get k X_is, and then take the median of X_is, say, $\bar{X}$. We have that $\bar{X} \in [a, b]$ with probability at least $1 - 2^{-\Omega(k)}$.

[BJKST] The algorithm is in the same spirit of [FM], except that at the high end of the "FM hit-array", they introduced a "buffer" to get a finer estimation. For details, see [LN], pages 17-18.

5 AMS Estimator for F_k

- Q: How to compute $F_k = \sum_j |f_j|^k$ where f_j is the frequency of item j in the stream?
- A: Use AMS sampling introduced before (Homework).
- Q: Better bounds?
- A: Yes. See the optimal algorithm in *Optimal Approximations of the Frequency Moments of Data Streams* by Indyk and Woodruff. STOC 2005.
- Q: Can we do even faster for F_2?
- A: Yes. Algorithm: let $f_i = |\{j : a_j = i\}|$.

1. For $1 \leq i \leq s_1 = \frac{16}{\lambda^2}$ and $1 \leq j \leq s_2 = 2\log(1/\epsilon)$, generate $X_{i,j}$ as follows:

(a) Choose a random $\{-1,+1\}$ vector $v = \{\epsilon_1, \ldots, \epsilon_n\}$ from a 4-wise $\{-1,+1\}$ vectors family, compute $Z = \sum_i \epsilon_i f_i$.

(b) Let $X = Z^2$.

2. Let $Y_j = \sum_i X_{i,j}/s_1$
3. Output the median of Y_js.

- Q: Why this works? Any general techniques?
- A: Yes. The p-stable distributions. A distribution is p-stable if $X_1, X_2, \ldots, X_n \sim \mu$ i.i.d. and $\vec{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$, then $a_1X_1 + a_2X_2 + \ldots + a_nX_n \sim \|\vec{a}\|_p X$ where $X \sim \mu$.

  For example, let $X_1, X_2, \ldots, X_n \sim N(0,1)$. Then we have $a_1X_1 + a_2X_2 + \ldots + a_nX_n \sim \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}X$ where $X \sim N(0,1)$

  Cauchy distribution is 1-stable where the Cauchy pdf is $\frac{2}{\pi}\frac{1}{1+x^2}$.

  For details, see P. Indyk *Stable Distributions, Pseudo-random Generators, Embeddings and Data Stream Computation*. STOC 2006.

- Q: Then why $\{+1,-1\}$ works?
- A: Because if $Y_1, \ldots, Y_n \in_R \{+1,-1\}$, then $\vec{Y} = \{Y_1, \ldots, Y_n\}$ is "close" to $\vec{Z} = \{Z_1, \ldots, Z_n\}$ where $Z_i \in N(0,1)$ for $n \to \infty$.

5.1 Analysis of AMS Estimator for F_2

Note that $\mathbf{E}[\epsilon_i] = 0$ for all i.

$$\begin{aligned}\mathbf{E}[X] &= \mathbf{E}[(\sum_i \epsilon_i f_i)^2] = \sum_i f_i^2 \mathbf{E}[\epsilon_i^2] + 2\sum_{i,j} f_i f_j \mathbf{E}[\epsilon_i]\mathbf{E}[\epsilon_j] = \sum_i m_i^2 = F_2.\\ \mathbf{Var}[X] &= E[X^2] - (E[X])^2 = 4\sum_{i,j} m_i^2, m_j^2 \le 2F_2^2.\end{aligned}$$

Now plug in average + median.

$$\mathbf{Pr}[|Y_i - F_2| > \lambda F_2] \le \frac{\mathbf{Var}(Y_i)}{\lambda^2 F_2^2} \le 1/8.$$

6 Graph Algorithms

6.1 Connectivity

- Q: Decide whether a graph is connected in small space.
- A: Algorithm:

1. Maintain label $\ell(u)$ for each node u where labels are initially distinct
2. On seeing edge $(u; v)$ with $\ell(u) \neq \ell(v)$, set $\ell(w)$ for all $\ell(u)$ for all w with $\ell(w) = \ell(v)$.
3. The graph is connected iff every node ends up with the same label.

6.2 Test Bipartite Graphs

- Q: Decide whether a graph is connected or not in $O(n \log n)$ space.
- A: The algorithm is similar as *connectivity*, except that when a cycle is detected, if it is odd, then we output "No".

6.3 Spanners

- Q: An α-spanner of a graph $G = (V; E)$ is a subgraph $H = (V; E')$ such that for all u, v, $d_G(u, v) \leq d_H(u, v) \leq d_G(u, v)$. Can we compute a graph spanner is small space?
- A: Algorithm:
  1. Let E' be initially empty.
  2. On seeing (u, v), set $E' = E' \cup (u; v)$ if $d_H(u, v) > 2t - 1$.
- Q: Can we get faster running time?
- A: Yes. Will talk about it next time.

7 K-center

Problem: Cluster a set of points $X = (x_1, x_2, \ldots, x_m)$ to clusters $c_1, c_2, \ldots, c_k$ with representatives $r_1 \in c_1, r_2 \in c_2, \ldots, r_k \in c_k$ to minimize $\max_i \min_j d(x_i, r_j)$.

Solution: We illustrate the algorithm in four steps.

1. First, a doubling algorithm (not necessary in the streaming setting):
   (a) When we see $(k + 1)$-st point in the input stream, suppose that the minimum pairwise distance is 1, by rescaling the distances.
   (b) Then at this point, we know that OPT ≥ 1, where OPT = cost of optimum clustering
   (c) The algorithm now proceeds in phases. At the start of the ith phase, we know that OPT $\geq d_i/2$ ($d_1 = 1$).
   (d) Phase i does the following:

i. Choose each point as the new representative if its distance from all other representatives $> 2d_i$.

ii. If we are about to choose the $(k+1)$-st representative, stop this phase, and let $d_{i+1} = 2d_i$.

iii. If the algorithm ends with $w < k$ clusters, we are good. and we can just add in some arbitrary cluster. This additional cluster could be empty or not.

2. The concept of *summary*. A summary of a stream $\sigma = \langle x_1, x_2, ... \rangle$ with $x_i \in U$ is any set $S \subseteq U$. The cost of a summary is

$$\Delta(\sigma, S) = \max_{x \in \sigma} \min_{y \in S} d(x, y)$$

.

We say Δ is a metric cost if for any streams $\sigma, \sigma \circ \pi$ and their respective summaries, S, T, we have:

$$\Delta(S \circ \pi, T) - \Delta(\sigma, S) \leq \Delta(\sigma \circ \pi, T) \leq \Delta(S \circ \pi, T) + \Delta(\sigma, S)$$

(The more precise notation for $\Delta(S \circ \pi, T)$ should be $\Delta(\sigma(S) \circ \pi, T)$ where $\sigma(S)$ is the input where every input $x \in \sigma$ is replaced by the corresponding $]hatx$ generated from S which best represents x).

3. The concept of threshold algorithm. For a summarization cost function Δ, Δ has an α-approximate threshold algorithm $\mathcal{A}$, such that $\mathcal{A}$ takes as input an estimate t and a stream σ, and

   (a) Either produces a summary S (where $|S| = k$) s.t. $\Delta(\sigma, S) \leq \alpha t$

   (b) Or fail, proving that $\forall T$ (where $|T| = k$), $\Delta(\sigma, T) > t$

   Note that the doubling algorithm is a constant-approximate threshold algorithm. Let α be this constant.

4. Guha's (2009) k-center algorithm:

   (a) Keep p (such that $(1+\epsilon)^p = \alpha/\epsilon$) instances of the $\mathcal{A}$ algorithm with contiguous geometrically increasing thresholds of the form $(1+\epsilon)^i$ running in parallel.

   (b) Whenever $q \leq p$ of instances fail, start up next q instances of $\mathcal{A}$ using the summaries from the failed instances the "replay" the stream. When an instance fails, kill all the other instances with a lower threshold. (Note that the threshold goes up from $(1+\epsilon)^i \to (1+\epsilon)^{i+p}$.

   (c) At the end of input, output the summary from the lowest threshold alive instances of $\mathcal{A}$.

7.1 Analysis

Idea: we run multiple copies (but controlled in number) of the algorithm corresponding to different estimates of the final cost and we use a "stream-strapping" procedure to use partially completed summarization for a certain estimate to create summarization for a different estimate of cost.

Suppose the final threshold, say, t, has been raised by j times. Let σ_i and S_i be the i-th portion of the stream and the summary of σ_i in phase i, respectively. Let T be the final summary. The cost of the final output =

$$\begin{aligned}
\Delta(\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_j \circ \pi, T) &\leq \Delta(S_j \circ \pi, T) + \Delta(\sigma_1 \circ \sigma_2 \circ \ldots, S_j) \\
&\leq \alpha t + \Delta(\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_j, S_j) \\
&\leq \alpha t + \Delta(S_{j-1} \circ \sigma_j, S_j) + \Delta(\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{j-1}, S_{j-1}) \\
&\leq \alpha t + \epsilon t + \epsilon^2 t + \ldots \\
&\leq \alpha t + (\epsilon/(1-\epsilon))t.
\end{aligned}$$

On the other hand, OPT $> t/(1+\epsilon)$.

8 Edit Distances

8.1 Edit Distance to Monotonicity

Problem: Compute the minimum #elements to delete such that the rest is a monotonically increasing sequence.

First talk about the method used in [GJKK07], without analysis. And then talk about the improved method used in [EJ08], with analysis.

Intuition: for each element e, if there are quite a lot elements in a certain range before it are inverse of it, then probably e should be deleted. To further improve this, we should not not count such "reverse" caused by a (big) element that is actually been deleted (use an example in the [GJKK07] paper). That is why in [EJ08] whether $e \in R$ also depends on previous elements in R.

8.1.1 The first algorithm

Define a set R_δ containing all indices i that are the right endpoints of an interval where more than a δ-fraction of the elements lie in $inv(i)$. Let $R = R_{1/2}$.

Thm: $|R|$ is a good estimate of ED, that is, for all $\delta \leq 1/2$,

$$ed(\sigma)/2 \leq |R| \leq ed(\sigma)/\delta.$$

Show two examples showing that this Thm is tight in terms of the estimator R we choose.

8.1.2 The second algorithm

The algorithm. We define a new estimator R:

1. $i-1 \in inv(i)$, or,
2. there is an interval $I=[j,i-1]$ such that $|inv(i)\cap I| > |R\cap I|$.

We say R is *total* if $\forall i \notin R$, i does not have a witness. We have the following theorem
Thm: $|R| \le ed(\sigma)$. Moreover if R is total then $|R| \ge \frac{1}{2}ed(\sigma)$.

Proof idea: For the left side, show that if we link each item in R to its witness, the resulting graph will be a set of decreasing paths which can lower bound the ED.

For the right side, we can show that we can delete $2|R|$ elements to get a monotonic increasing sequence by deleting $[j+1,i-1]$ where $i\in R, i-1\notin R$ and j is the largest index such that $j<i$ and $j\notin R\cup inv(i)$.

The analysis. Direct comparing $|inv(i)\cap I|$ and $|R\cap I|$ is very difficult. We need a novel method to test majority: to compute the approximate median! Let $A(S,k\phi)$ be an ϵ-approximation for the ϕ-quantile of the k-most recent elements in stream S. The algorithm is as follows:

RedTest(j,i)

1. Let $a=A(\sigma,i-j,\frac{1}{2}-\epsilon)$.
2. If $a\le\sigma(i)$ return FALSE.
3. Let $a'=A(R',i-j,\frac{1}{2}+\epsilon)$.
4. If $a'=0$ then return TRUE otherwise return FALSE.

Note that we do not actually find a total R, but close to.

Finally, we don't have to test for all intervals $[j,i]$ but only those with length $1,2,\ldots,(1+\epsilon'),(1+\epsilon')^2,\ldots,n$.

8.2 Embedding Ulam distance

Idea: turns "count", "there exists" and "majority" to ℓ_1,ℓ_∞ and $(\ell_2)^2$. Our host space: $\bigoplus^d_{(l_p)^p}\bigoplus^{O(\log d)}_{l_\infty} l_1^{2d}$ where $p=1+\epsilon$.

Definition.

1. P_{ak}: the k symbols that appear in the k positions before a.

2. φ_{ak}: the 0/1 incidence vector of P_{ak} scaled by $1/2k$.
3. $\varphi_a = \bigoplus_{k\in K} \varphi_{ak}(P)$ where $K = \{(1+\gamma)^i\}$ $(0 \le i \le \log_{1+\gamma} d)$.
4. $\varphi = \bigoplus_{a\in[d]} \varphi_a(P)$

$$d_{p^p,\infty,1}(\varphi(P),\varphi(Q)) = \sum_{a\in[d]} (d_{\infty,1}(\varphi_a(P),\varphi_a(Q)))^p = \sum_{a\in[d]} \left(\max_{k\in K} \|\varphi_{ak}(P),\varphi_{ak}(Q)\|_1 \right)^p$$

Definition: For $P, Q \in Ulam_d$, and $0 < \delta \le 1/2$. Let T_δ be the set of all $a \in \Sigma$ for which there exists $k \in [d]$ such that the symmetric difference $|P_{ak}\Delta Q_{ak}| > 2\delta k$.

Lem:

$$\frac{1}{2} ed(P,Q) \le |T_\delta| \le \frac{4}{\delta} ed(P,Q)$$

.

Thm: φ is an embedding with constant distortion.

Proof:

1. For each $a \in T_{1/2}$, there exists $k \in [d]$ such that $P_{ak}\Delta Q_{ak} > 2 \cdot 1/2 \cdot k$. Thus there exist k' such that $\|\varphi_{ak'}(P),\varphi_{ak'}(Q)\|_1 = \frac{1}{2k'} |P_{ak'}\Delta Q_{ak'}| = \Omega(1)$.
2. On the other hand,

$$\begin{aligned}
d_{p^p,\infty,1}(\varphi(P),\varphi(Q)) &= \sum_{a\in[d]} \left(\max_{k\in K} \|\varphi_{ak}(P),\varphi_{ak}(Q)\|_1 \right)^p \\
&\le \sum_{a\in[d]} \max_{k\in[d]} \|\varphi_{ak'}(P),\varphi_{ak'}(Q)\|_1^{1+\epsilon} \\
&\quad \text{(relax } k \in K \text{ to } k \in [d]) \\
&= \sum_{a\in[d]} \max_{k\in[d]} \left[\frac{1}{2k} |P_{ak}\Delta Q_{ak}| \right]^{1+\epsilon} \\
&\le 1 + \sum_{j=1}^{\log d} (2^{-j+1})^{1+\epsilon} \cdot |T_{2^{-j}}| \\
&\quad \text{(break the contribution of } a \in \Sigma \text{ into buckets of the form } [2^{-j}, 2^{-j+1}]) \\
&\le 1 + \sum_{j=1}^{\log d} (2^{-j+1})^{1+\epsilon} \cdot 2^{j+2} ed(P,Q) \\
&\le 1/\epsilon \cdot ed(P,Q)
\end{aligned}$$

9 Lower Bounds

9.1 ST-connectivity

- Q: Given an input graph stream, are two specific vertices s and t connected? What is the minimum space needed?
- A: $\Omega(n)$. By reduction from DISJ. Let (x, y) be an instance of DISJ. Construct an instance of CONN on a graph with vertex set $(s, t, v_1, v_2, \ldots, v_{n-2})$, and edges $A = \{(s, v) : v \in x\}$ and $B = \{(v, t) : v \in x\}$. It is easy to see that $\text{DISJ}(x, y) = 1$ iff s, t are connected in the graph.

9.2 Perfect Matching

- Q: Given an input graph stream, does it have a perfect matching of size $n/2$?
- A: $\Omega(n^2)$. By reduction from INDEX. Let (x, k) $(x \in \{0, 1\}^{n^2}, k \in [n^2])$ be an instance of INDEX. Construct an instance of PM on a graph with vertex set $\cup_{i=1}^{n}(a_i, b_i, u_i, v_i)$, and edges $A = \{(u_i, v_j)$ for each $x_{f(i,j)} = 1\}$ where $f(i, j) = (n-1)i + j$ and $B = \{(a_l, u_l)$ for each $l \neq i^*\} \cup \{(b_l, v_l)$ for each $l \neq j^*\} \cup \{a_{i^*}, b_{j^*}\}$ where $f(i^*, j^*) = k$. It is easy to see that $\text{INDEX}(x, k) = 1$ iff the graph has a perfect matching.

9.3 F_k

- Q: What is the space lower bounds to estimate F_k?
- A: $\Omega(n^{1-(2+\gamma)/k})$ for arbitrarily small γ. By reduction from miltiparty DISJ (the promised version).

  Let $\mathcal{A}$ be an s-space data stream algorithm that approximates F_k within $1 \pm \epsilon$ multiplicative error with confidence $1 - \delta$ $(0 < \delta < 1/4)$. We use $\mathcal{A}$ to construct a δ-error one-way protocol for $\text{DISJ}_{n,t}$, where $t = ((1 + 3\epsilon)n)^{1/k}$ by partitioning the stream to t parts and feeding each player one part. If there exists an intersection, then $F_k \geq t^k = (1 + 3\epsilon)n$. Otherwise $F_k \leq n$. So a $(1 + \epsilon)$ estimation of F_k can solve $\text{DISJ}_{n,t}$. Therefore we get a protocol that solves $\text{DISJ}_{n,t}$ with communication $s(t - 1) + 1$. On the other hand we know that any protocol that solves $\text{DISJ}_{n,t}$ has communication at least $n/t^{1+\gamma}$. Thus we have that $s = \Omega(n^{1-(2+\gamma)/k})$.

10 Some Other Topics

1. Histograms: The input is $\langle x_1, x_2, \ldots, x_m \rangle \in [n]^m$. Goal: Determine B bucket histogram $H : [m] \to \mathbb{R}$ minimizing $\sum_{i \in [m]}(x_i - H(i))^2$.

Result: $(1+\epsilon)$ estimation in $\tilde{O}(B^2\epsilon^{-1})$ space. See the paper *Fast, Small-Space Algorithms for Approximate Histogram Maintenance* by Gilbert et. al.

2. Transpositions: The input is $\langle x_1, x_2, \ldots, x_m \rangle \in [n]^m$. Goal is to estimate number of transpositions $|\{i < j : x_i > x_j\}|$.

   Result: $(1+\epsilon)$ estimation in $\tilde{O}(\epsilon^{-1})$ space. See the paper *Counting inversions in lists* by A. Gupta and F. Zane. And an improvement by G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Space- and time-efficient deterministic algorithms for biased quantiles over data streams. PODS 2006.