Approximate Median in the vanilla streaming model via sampling

Let \( O = \{ a_1, a_2, \ldots, a_m \} \) and define \( \text{rank}(y) = \left\{ \{ a_i : a_i \leq y \} \right\} \).

For simplicity, assume all \( a_i \)'s distinct.

Problem: Find an \( \epsilon \)-approximate median of \( O \), i.e., \( y \) such that

\[
\frac{m}{2} - \epsilon m \leq \text{rank}(y) \leq \frac{m}{2} + \epsilon m
\]

Algorithm: Sample \( t \) values from \( O \) with replacement and return median of the sampled values. (Easy: \( \frac{m}{3} \leq \text{streamlength} \) in \( \epsilon \)-median.)

Lemma: If \( t = \frac{7}{2} \log(2e) \epsilon^{-1} \) then the algorithm returns an \( \epsilon \)-approximate median with probability \( 1 - 5 \).

Later in Seminar:

Quadratically better dependence on \( \epsilon \).

Proof: Partition the \( a_i \)'s into 3 groups:

\( S_L = \{ a_i : \text{rank}(a_i) \leq \frac{m}{2} - \epsilon m \} \)

\( S_M = \{ a_i : \frac{m}{2} - \epsilon m \leq \text{rank}(a_i) \leq \frac{m}{2} + \epsilon m \} \)

\( S_U = \{ a_i : \text{rank}(a_i) \geq \frac{m}{2} + \epsilon m \} \)

If fewer than \( \frac{t}{2} \) elements of both \( S_L \) and \( S_U \) are in sample, then median of the sample is in \( S_M \). (Seeing this is more subtle than it appears at first glance.)
Let $X_j = 1$ if $j$th sample is in $S_L$ and $X_j = 0$ otherwise.

Let $X = \sum_j X_j$, so by Chernoff bound, $\Pr(\sum_j X_j > \ln(n)) \leq \frac{1}{2}\exp\left(-\frac{n \cdot (\epsilon)^2}{3}\right)$.

Similarly, there are at most $\frac{\epsilon}{2}$ elements from $S_U$ with probability $\leq \frac{\epsilon}{2}$.

By a union bound, there are at most $\frac{\epsilon}{2}$ elements from $\text{Batch of } S_L$ and $S_U$ with probability $1-\frac{\epsilon}{2}$. 
How to compute a random sample (with replacement) of size $t$ from a stream when you don't know the stream length in advance?

Consider $s_1, \ldots, s_t$ (for general $t$, run the $s_i$ samples in $O(n)$ time independently in parallel).

Algorithm. Initially $s = x_1$.

On seeing $x_i$, set $s \leftarrow x_i$ with probability $\frac{1}{i}$. 

Analysis. What is the probability the $s = x_i$ at some point $j > i$?

$\Pr[s = x_i] = \frac{1}{i} \times (1 - \frac{1}{i+1}) \times (1 - \frac{1}{i+2}) \times \ldots \times (1 - \frac{1}{j} )$

\[ = \frac{1}{\prod_{i}^{j} (1 - \frac{1}{i+1})} \]

Obvious when $j = i$. Assume it's true for $j$, let's show it is true for $j+1$. By induction,

\[ \frac{1}{\prod_{i}^{j+1} (1 - \frac{1}{i+1})} = \frac{1}{\prod_{i}^{j} (1 - \frac{1}{i+1})} \times (1 - \frac{1}{j+2}) \]

There is a variant for sampling items without replacement (much trickier article), that says has the benefit of running our update time rather than $O(t)$. 

\[ \frac{1}{\prod_{i}^{j+1} (1 - \frac{1}{i+1})} = \frac{1}{j+1} - \frac{1}{j+1} \]
An overview of uniform sampling algorithms for various streaming problems.

- We just saw uniform random sampling gives an \( O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)\)-space streaming algorithm for outputting an \( \varepsilon \)-approximate median with probability \( \geq 1 - \delta \). *(Insertion-only)*

  - Suboptimal dependence on \( \frac{1}{\varepsilon} \) we'll see a better approximate median algorithm later on the course.

- Uniform random sampling can also give an algorithm using space \( O\left(\frac{\log n \cdot \log(1/\delta)}{\varepsilon^2}\right) \) for answering \( \varepsilon \)-approximate rank queries, i.e., outputting a summary of the above size such that, with probability \( \geq 1 - \delta \), for any \( i \in [n] \), an estimate \( \hat{f}_i \) of \( f_i \) can be derived from the summary, satisfying

\[
|f_i - \hat{f}_i| \leq \varepsilon \cdot \Delta
\]

Algorithm: Sample \( O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right) \) stream updates, and output the estimate \( \hat{f}_i := \frac{\text{# of samples equal to } i}{m} \cdot \frac{m}{\varepsilon} \).

Analysis: Easy to see \( \mathbb{E}[\hat{f}_i] = f_i \). Bound the probability of \( |f_i - \hat{f}_i| > \varepsilon \cdot \Delta \) using additive Chernoff bounds (exercise).

- Suboptimal dependence on \( \frac{1}{\varepsilon} \). We saw a different algorithm in Lecture 2 using \( O\left(\frac{\log n}{\varepsilon^2}\right) \) space.
Are there any streaming products for which random sampling is optimal?

Answer: Yes. Itemset frequency estimates.

Consider a database of grocery purchases. Each row is a receipt, each column is a product, $D_{ij} = 1$ if person i purchased item j, and $D_{ij} = 0$ otherwise.

<table>
<thead>
<tr>
<th>Alice's receipt</th>
<th>Bob's receipt</th>
<th>John's receipt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 0 1 0</td>
<td>1 0 1 1 0 0 0</td>
<td>0 1 1 0 1 1 1</td>
</tr>
<tr>
<td>0 1 1 0 1 1 1</td>
<td>0 0 0 0 1 1 0</td>
<td>0 0 1 0 1 1 0</td>
</tr>
<tr>
<td>0 1 0 0 0 0 1</td>
<td>1 1 0 0 0 0 0</td>
<td>1 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

An itemset of size $K$ is a set of $K$ columns.

The frequency of an itemset $S$ is $f_S = \#$ of rows with a 1 in all columns in $S$.

E.g., $f_{\{mops, gloves\}} = \#$ of people who bought both mops and gloves.

Mining frequent itemsets is very well-studied in the data mining community.

Goal: Output a summary of the database capable of returning for any $K$-itemset $S$, an estimate $f_S$ satisfying $\epsilon \leq \frac{|S|}{f_S} \leq \frac{1}{\epsilon}$.
The simple summary: Sample $t$ rows (they each take $d$ bits to write down), for $t \geq O\left(\frac{1}{\epsilon^3} \log(\frac{k}{\delta})\right)$.

Output for each $k$-item set $S$ the estimate
\[
\frac{m}{t} \cdot \left(\text{\# of sampled rows containing all $k$}\right)
\]

Additive Chernoff bounds imply this is a good summary with probability $\geq 1 - \delta$.

- [MTV16]: This space cost is optimal (even among summaries not computed by streaming algorithms).

Intuitively, a key difference between point queries and itemset frequency queries is that for point queries, there can be at most $\frac{1}{\epsilon}$ items $i$ with frequency $\frac{f_i}{\epsilon}$ in $S$ (and for all other items, it is okay to output the estimate 0). This is not the case for itemsets since a single row can contribute to the frequency of many items.