Sublinear Time and Space Algorithms 2018B – Lecture 3
\[ \ell_2 \] Frequency Moment and Point Queries*

Robert Krauthgamer

1 \[ \ell_1 \] Point Query via CountMin (continued from last time)

Algorithm CountMin+:

1. Run \( t = \log n \) independent copies of algorithm CountMin, keeping in memory the vectors \( S^1, \ldots, S^t \) (and functions \( h^1, \ldots, h^t \))

2. Output: the minimum of all estimates \( \hat{x}_i = \min_{l \in [t]} S_{h(l)}^l \)

Analysis (correctness): As before, \( \hat{x}_i \geq x_i \) and

\[
\Pr[\hat{x}_i > x_i + \alpha \|x\|_1] \leq (1/4)^t = 1/n^2.
\]

By a union bound, with probability at least \( 1 - 1/n \), for all \( i \in [n] \) we will have \( x_i \leq \hat{x}_i \leq x_i + \alpha \|x\|_1 \).

Space requirement: \( O(\alpha^{-1} \log n) \) words (for success probability \( 1 - 1/n^2 \)), without counting memory used to represent/store the hash functions.

Space requirement: \( O(\alpha^{-1} \log n) \) words (for success probability \( 1 - 1/n^2 \)), without counting memory used to represent/store the hash functions.

General \( x \) (allowing negative entries):

We saw in class that Algorithm CountMin actually extends to general \( x \) that might be negative, and achieves the guarantee

\[
\Pr[\hat{x}_i \in x_i \pm \alpha \|x\|_1] \leq 1/4.
\]

Next class we will see how to amplify the success probability, using median (instead of minimum) of \( O(\log n) \) independent repetitions.

Exer: Let \( x \in \mathbb{R}^n \) be the frequency vector of a stream of \( m \) items (insertions only). Show how to use the CountMin+ sketch seen in class (for \( \ell_1 \) point queries) to estimate the median of \( x \), which means to report an index \( j \in [n] \) that with high probability satisfies \( \sum_{i=1}^j x_j \in (\frac{1}{2} \pm \varepsilon)m \).

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.
2 Frequency Moments and the AMS algorithm

\( \ell_p \)-norm problem: Let \( x \in \mathbb{R}^n \) be the frequency vector of the input stream, and fix a parameter \( p > 0 \).

Goal: estimate its \( \ell_p \)-norm \( \|x\|_p = \left( \sum |x_i|^p \right)^{1/p} \). We focus on \( p = 2 \).

**Theorem 1 [Alon, Matthias, and Szegedy, 1996]:** One can estimate the \( \ell_2 \) norm within factor \( 1 + \varepsilon \) [with high constant probability] using a linear sketch of size (dimension) \( s = O(\varepsilon^{-2}) \). It implies, in particular, a streaming algorithm.

**Algorithm AMS (also known as Tug-of-War):**

1. Init: choose \( r_1, \ldots, r_n \) independently at random from \([-1, +1]\)
2. Update: maintain \( Z = \sum_i r_i x_i \)
3. Output: to estimate \( \|x\|_2 \) report \( Z^2 \)

The sketch \( Z \) is linear, hence can be updated easily.

Storage requirement: \( O(\log(nm)) \) bits, not including randomness; we will discuss implementation issues a bit later.

**Analysis:** We saw in class that \( \mathbb{E}[Z^2] = \sum_i x_i^2 = \|x\|_2^2 \), and \( \text{Var}(Z^2) \leq 2(\mathbb{E}[Z^2])^2 \).

**Algorithm AMS+:**

1. Run \( t = O(1/\varepsilon^2) \) independent copies of Algorithm AMS, denoting their \( Z \) values by \( Y_1, \ldots, Y_t \), and output their mean \( \tilde{Y} = \frac{1}{t} \sum_j Y_j^2 \).

Observe that the sketch \( (Y_1, \ldots, Y_t) \) is still linear.

Storage requirement: \( O(t) = O(1/\varepsilon^2) \) words (for constant success probability), not including randomness.

**Analysis:** We saw in class that

\[
\Pr[|\tilde{Y} - \mathbb{E}[\tilde{Y}]| \geq \varepsilon \mathbb{E}[\tilde{Y}]) \leq \frac{\text{Var}(\tilde{Y})}{\varepsilon^2(\mathbb{E}[\tilde{Y}])^2} \leq \frac{2}{t\varepsilon^2}.
\]

Choosing appropriate \( t = O(1/\varepsilon^2) \) makes the probability of error an arbitrarily small constant.

Notice it is actually a \((1 \pm \varepsilon)\)-approximation to \( \|x\|_2^2 \), but it immediately yields a \((1 \pm \varepsilon)\)-approximation to \( \|x\|_2 \).

**Exer:** What would happen in the accuracy analysis if the \( r_i \)'s were chosen as standard gaussians \( \mathcal{N}(0, 1) \)?

3 \( \ell_2 \) Point Query via CountSketch

The idea is to hash coordinates to buckets (similar to algorithm CountMin), but furthermore use tug-of-war inside each bucket (as in algorithm AMS). The analysis will show it is a good estimate
for each $x_i^2$ (instead of $x_i$).

**Theorem 2 [Charikar, Chen and Farach-Colton, 2003]:** One can estimate $\ell_2$ point queries within error $\alpha$ with constant high probability, using a linear sketch of dimension $O(\alpha^{-2})$. It implies, in particular, a streaming algorithm.

It achieves better accuracy than CountMin ($\ell_2$ instead of $\ell_1$), but requires more storage ($1/\alpha^2$ instead of $1/\alpha$).

**Algorithm CountSketch:**

1. **Init:** Set $w = 4/\alpha^2$ and choose a pairwise independent hash function $h : [n] \rightarrow [w]$
2. Choose pairwise independent signs $r_1, \ldots, r_n \in \{-1, +1\}$
3. **Update:** Maintain vector $S = [S_1, \ldots, S_w]$ where $S_j = \sum_{i:h(i)=j} r_i x_i$.
4. **Output:** To estimate $x_i$ return $\tilde{x}_i = r_i \cdot S_{h(i)}$.

Storage requirement: $O(w)$ words, i.e., $O(\alpha^{-2} \log(nm))$ bits. The hash functions can be stored using $O(\log n)$ bits.

**Correctness:** We saw in class that $\Pr[|\tilde{x}_i - x_i|^2 \geq \alpha^2 \|x\|_2^2] \leq 1/4$, i.e., with high (constant) probability, $\tilde{x}_i \in x_i \pm \alpha \|x\|_2$.

Next class we will see how to amplify the success probability to $1 - 1/n^2$ using the median of $O(\log n)$ independent copies.