Lecture 8: Streaming and Sketching I

"In low-space, nobody can remember your stream."

We will follow for this chapter the (excellent) lecture notes by Amit Chakrabarti [AC], available at `https://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf`.

The Basic Setup

We will specifically focus on one-pass algorithms, unless specified otherwise. m denotes the length of the stream

$$\sigma = \langle a_1, \dots, a_m \rangle$$

where each a_i belongs to the universe $\mathcal{X}$ of size n. We do not impose any bound on the time complexity of our algorithms, but we will enforce that they use very little memory (space), with space complexity denoted by s. We will aim for

Note that this notation is swapped with respect to the previous lectures, in other to match the lecture notes.

$$s = o(\min(m, n))$$

and would love to use much less, ideally

$$s = O(\log m + \log n)$$

or, if not, $s = \mathrm{poly}(\log m, \log n)$. To do so, we will allow for randomised algorithms *and* approximation algorithms, where the quality of the approximation will be controlled by a parameter $\varepsilon > 0$, usually thought of as an (arbitrarily) small fixed constant.

The Majority Problem

Chapter 1 of [AC]

Theorem 39. *The* Misra-Gries *algorithm is a* deterministic *one-pass algorithm which, for any given parameter $\varepsilon \in (0, 1]$, provides $\hat{f}_1, \dots, \hat{f}_n$ of all element frequencies such that*

$$f_j - \varepsilon m \leq \hat{f}_j \leq f_j, \qquad j \in [n]$$

with space complexity $s = O(\log(mn)/\varepsilon)$. (In particular, it can be used to solve the Majority *problem in two passes.)*

The Approximate Counting Problem

Chapter 4 of [AC]

We will describe and analyse the Morris Counter algorithm, due to, well, Morris, which provides a constant-factor estimate of the number of elements of the stream: that is, an F_1 estimator.

Theorem 40. *The medians-of-means version of the* MORRIS COUNTER *is a* randomised *one-pass algorithm which, for any given parameters* $\varepsilon, \delta \in (0,1]$, *provides an estimate* $\widehat{d}$ *of the number* d *of non-zero elements of the stream such that*

$$\Pr\Big[\,(1-\varepsilon)d \leq \widehat{d} \leq (1+\varepsilon)d\,\Big] \geq 1-\delta$$

with space complexity

$$s = O\left(\frac{\log\log m}{\varepsilon^2} \cdot \log\frac{1}{\delta}\right)$$

that is, doubly logarithmic *in* m.

But we can do better!

Theorem 41. *The "careful" version of* MORRIS COUNTER *is a* randomised *one-pass algorithm which, for any given parameters* $\varepsilon, \delta \in (0,1]$, *provides an estimate* $\widehat{d}$ *of the number* d *of non-zero elements of the stream such that*

$$\Pr\Big[\,(1-\varepsilon)d \leq \widehat{d} \leq (1+\varepsilon)d\,\Big] \geq 1-\delta$$

with space complexity

$$s = O\left(\log\log m + \log\frac{1}{\varepsilon} + \log\frac{1}{\delta}\right)$$

that is, doubly logarithmic in m and logarithmic in $1/\varepsilon$.

The Distinct Elements Problem

Chapters 2 and 3 of [AC]

We start this section with the TIDEMARK algorithm, due to Alon, Matias and Szegedy (AMS), which provides a constant-factor estimate of the number of distinct elements of the stream: that is, an F_0 estimator.

Algorithm 15: The TIDEMARK algorithm

```
Pick h: [n] → [n] from a strongly universal hashing family
z ← 0
for all 1 ≤ i ≤ m do
    Get item a_i ∈ [n]
    if zeros(h(a_i)) ≥ z then
        z ← zeros(h(a_i))
return √2 · 2^z
```

Theorem 42. *The (median trick version of the)* TIDEMARK *(AMS) algorithm is a* randomised *one-pass algorithm which, for any given* *parameter* $\delta \in (0,1]$, *provides an estimate* $\widehat{d}$ *of the number* d *of distinct elements of the stream such that, for some absolute constant* $C > 0$,

$$\Pr\left[\frac{1}{C} \cdot d \leq \widehat{d} \leq C \cdot d \right] \geq 1 - \delta$$

with space complexity

$$s = O\left(\log n \cdot \log \frac{1}{\delta} \right).$$

This is not bad, but can we achieve estimation factor arbitrarily close to one, say, $1 + \varepsilon$? The answer is yes: the following algorithm, due to Bar-Yossef, Jayram, Kumar, Sivakumar and Trevisan (BJKST), does exactly that.

Algorithm 16: The BJKST algorithm

```
Input: Parameter ε ∈ (0,1]
 1: Set k ← O(log² n/ε⁴), T ← Θ(1/ε²)
 2: Pick h: [n] → [n] from a strongly universal hashing family
 3: Pick g: [n] → [k] from a strongly universal hashing family

 4: z ← 0, B ← ∅
 5: for all 1 ≤ i ≤ m do
 6:     Get item a_i ∈ [n]
 7:     if zeros(h(a_i)) ≥ z then
 8:         B ← B ∪ {(g(a_i), zeros(h(a_i)))}
 9:         while |B| ≥ T do
10:             z ← z + 1
11:             Remove every (a, b) with b < z from B
12: return |B| · 2^z
```

Theorem 43. *The (median trick version of the)* BJKST *algorithm is a* randomised *one-pass algorithm which, for any given parameters* $\varepsilon, \delta \in (0,1]$, *provides an estimate* $\widehat{d}$ *of the number* d *of distinct elements of the stream such that, for some absolute constant* $C > 0$,

$$\Pr\left[(1-\varepsilon) \cdot d \leq \widehat{d} \leq (1+\varepsilon) d \right] \geq 1 - \delta$$

with space complexity

$$s = O\left(\left(\log n + \frac{\log(1/\varepsilon) + \log \log n}{\varepsilon^2} \right) \cdot \log \frac{1}{\delta} \right).$$

This is pretty good, but. . . *Is it optimal?*