CSE 632: Analysis of Algorithms II **Fall 2017**

Lecture 24 (11/15/2017): Sketching and Streaming (I)

Lecturer: Shi Li *Scribe: Xiangyu Guo*

24.1 Introduction

Data Stream: A massive input sequence of elements $\langle i_1, \ldots, i_n \rangle$ (appears one by one and irreversible), where the elements are drawn from a universe $[m]$

Goal: Compute some function on $\langle i_1, \ldots, i_n \rangle$ with limited storage (usually $\text{poly}\log(n, m)$ or $\min\{n, m\}^\alpha$ for some $\alpha < 1$). The required function usually has the following characters:

- Functions are usually "trivial" to compute in traditional non-streaming model.
- Approximation is needed: α-approximation.
- Randomness is needed: with success probability at least $1 - \delta$.

24.2 Counting Distinct Elements

Problem description: Given data stream $\langle i_1, \ldots, i_n \rangle$, count the number of distinct elements, i.e., $f = |\{i_1, \ldots, i_n\}|$, where $i_t \in [m]$ for each i_t, and assume n is known.

Analysis: if we allow $\Theta(m)$ storage then the problem is trivial: just use an array of size m to count the number of the occurence for each $x \in [m]$, and output the number of nonzero slots at the end. Therefore, we assume m is huge, and our goal is to compute f (approximately) with storage $\text{poly}(1/\epsilon, 1/\delta, \log n, \log m)$. Here ϵ represents the degree of approximation and δ denotes the success probability: we want to get a $(1 + \epsilon)$-approximation of f with probability at least $1 - \delta$.

The first step of our solution is trying to solve a "desicion" version of f: given a threshold T, we want to distinguish between the following two cases

1. "Yes" case: if $f \geq (1 + \epsilon)T$.
2. "No" case: if $f \leq T$.

with success probability at least $1 - \delta'$. If we have such an algorithm $\mathcal{A}$ that can distinguish between the two cases above efficiently, then we can run multiple parallel copies of $\mathcal{A}'$ with $T = 1, 1 + \epsilon, (1 + \epsilon), \ldots, (1 + \epsilon)^{\log_{1+\epsilon} n}$, and choose the smallest T that $\mathcal{A}'$ returns "No". Apparently, $T/(1 + \epsilon) \leq f \leq T$, i.e., we get a $(1 + \epsilon)$-approximation to f.

So, how should we design $\mathcal{A}'$? If we have a subroutine that returns correct answer with a constant probability, then by repeatedly running this subroutine independently for many times, we are able to boost the success probability to arbitrarily high. We call this subroutine as a single "experiment": Suppose there're indeed l distinct elements, then each single experi will return "Yes"

Algorithm 1 A Single experiment

1: **for** every $i \in [m]$ **do**
2: Include i in sample S with probability $1/T$.
3: **for** every $t = 1, 2, \ldots, n$ **do**
4: **if** $i_t \in S$ **then return** "Yes";

with probability $1 - (1 - 1/T)^l \approx 1 - e^{-l/T}$. So we have:

$$\Pr[\text{A single experiment report "Yes"}] \geq 1 - e^{-(1+\epsilon)} \triangleq p_1 \quad (\text{if } l \geq (1+\epsilon)T)$$

$$\Pr[\text{A single experiment report "No"}] \leq 1 - e^{-1} \triangleq p_0 \quad (\text{if } l \leq T)$$

To boost the success probability, we run N copies of experiments parallelly. And if the number of "Yes" answers is at least $\frac{p_0+p_1}{2} \cdot N$, return "Yes", otherwise return "No". By a standard application of Chernoff's bound, we can show that $N = \Theta\left(\frac{1}{\epsilon^3} \log \frac{1}{\delta}\right)$.

There's only one problem left: in Algorithm 1, the sample S has a expected size of m/T, which is of order m for small T's. But we really don't need to store the whole S; instead, we can use a *pseudo-random generator* of size $\operatorname{poly}\log(n)$ to replace this sample S. The key idea here is that you cannot distinguish between a S uniform randomly sampled from $[m]$ and one that is generated by the pseudo-random generator.

Some final words: Recall that each element i_t in the stream ranges in $[m]$, so we can use a m-dimension vector $\mathcal{X} = (x_1, x_2, \ldots, x_m)$ to store the information of the stream: here x_i denote the number of times that i appears in the stream. Now $f = |\{i : x_i > 0\}|$ is simply the ℓ_0-norm of $\mathcal{X}$. This particular f is also denoted as F_0. We can further consider some different types of f, for example, $F_1 = \|\mathcal{X}\|_1$; but this is trivial since $\|\mathcal{X}\|_1 \equiv n$. A more interesting f will be $F_2 = \|\mathcal{X}\|_2^2$, and we'll discuss more about it in the next lecture.