Special Topics in Theoretical Computer Science Spring 2019

Lecture 2 — 1/10/2019

Prof. Huy L. Nguyen *Scribe: Lydia Zakynthinou*

In the first lecture we reviewed some basic tail bounds and we discussed Morris' algorithm and its extensions for the counting problem.

In this lecture we will define the streaming model more formally and present algorithms for the problem of counting *distinct* elements in a stream; we will start from an idealized solution and conclude with the non-idealized.

1 Streaming Model

The Streaming Model of Computation is described as follows:

- Input vector $\mathbf{x} \in \mathbb{R}^d$.
- $\mathbf{x}$ is initialized to $\vec{0}$.
- Get a stream of updates $\mathbf{x}_i \leftarrow \mathbf{x}_i + \alpha$, each update characterized by the tuple (i, α).
- At the end, compute $f(\mathbf{x})$.

The primary resource we care about is space. The task would be trivial if we could store all of $\mathbf{x}$. To use as little memory as possible, a lot of times we settle for a $(1 \pm \varepsilon)$ approximation of $f(\mathbf{x})$ and allow for randomized algorithms that fail to satisfy this guarantee with probability at most δ.

Any computational problem can be modeled this way, although a more natural representation is usually desired. Examples of settings where these problems arise are the following:

- *Router*: e.g. the input is $\mathbf{x} = (x_1, \ldots, x_m)$, where x_j represents the number of packets sent my machine j. Often we need to calculate statistics such as the number of distinct machines sending packets, the number of machines using a large fraction of the bandwidth, etc.
- *Log analysis*: e.g. the input is $\mathbf{x} = (x_1, \ldots, x_m)$, where x_j represents the number of times that a Google query j appears in the log. We might need to compare the current distribution of queries with yesterday's distribution.

In many cases, such as these, $\alpha = 1$. But α could also be negative, e.g. if we count the number of connections a machine has then $\alpha \in \{-1, +1\}$.

2 Distinct Elements Problem

We focus now on the problem of counting distinct elements in a stream. More specifically, we have a stream of indices $i_1, i_2, \ldots, i_m$ and want to calculate the number of distinct values in the stream.

Let the indices be in the range $\{1, \ldots, n\}$ and let t denote the number of distinct values in the stream.

A trivial algorithm would be to store all distinct elements in a table. This algorithm would return the exact number but it would need $\min(n, m) \log n$ memory.

2.1 Flajolet-Martin algorithm: Idealized Version

In 1985, P. Flajolet and G. N. Martin [1] gave the following algorithm for the problem, denoted *FM*:

- Choose a random hash function $h : \{1, \ldots, n\} \to [0, 1]$.
- Store the minimum hash value $X = \min_{i \in \text{stream}} h(i)$.
- At the end, return $\hat{t} = \frac{1}{X} - 1$.

This is an idealized version because it uses a hash function in real numbers and we would need infinite memory to store real numbers in a computer. We will analyze this version first, assuming real numbers can be stored, and explain how to deal with this assumption later.

To gain some intuition about why the expected returned value is the correct one, we examine the cases of $t = 1$ and $t = 2$.

- *For $t = 1$:* There is only one value j, and since the hash function is uniform, in expectation $h(j) = 0.5$ and $X = 0.5$ as well. Therefore, in expectation $\hat{t} = \frac{1}{0.5} - 1 = 1$.
- *For $t = 2$:* For two values j_1, j_2, it holds that $h(j_1) = 1/3$ and $h(j_2) = 2/3$ so $X = 1/3$ in expectation. Therefore, the expected returned value is $\hat{t} = \frac{1}{1/3} - 1 = 2$.

As in the last lecture, we will try to understand the distribution of $\hat{t}$ through the expectation and the variance.

Lemma 2.1.1.

$$\mathbb{E}[X] = \frac{1}{t+1}$$

Proof. By the definition of expectation for continuous distributions,

$$\begin{aligned}
\mathbb{E}[X] &= \int_0^\infty \Pr[X \geq \lambda] d\lambda \\
&= \int_0^1 (1-\lambda)^t d\lambda && (t \text{ independent values, each} \geq \lambda) \\
&= -\frac{(1-\lambda)^{t+1}}{t+1}\Big|_0^1 \\
&= \frac{1}{t+1}
\end{aligned}$$

□

Lemma 2.1.2.

$$\mathbb{E}[X^2] = \frac{2}{(t+1)(t+2)}$$

Proof.

$$\begin{aligned}
\mathbb{E}[X^2] &= \int_0^\infty \Pr[X^2 \geq \lambda] d\lambda \\
&= \int_0^1 \Pr[X \geq \sqrt{\lambda}] d\lambda \\
&= \int_0^1 (1-\sqrt{\lambda})^t d\lambda \\
&= \int_0^1 2(1-u)^t u du && (u = \sqrt{\lambda},\ du = d\lambda/2\sqrt{\lambda}) \\
&= 2\left(\int_0^1 (1-u)^t du - \int_0^1 (1-u)^{t+1} du\right) && (u = 1-(1-u)) \\
&= 2\left(\frac{1}{t+1} - \frac{1}{t+2}\right) \\
&= \frac{2}{(t+1)(t+2)}
\end{aligned}$$

□

It follows that $\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = \frac{2}{(t+1)(t+2)} - \frac{1}{(t+1)^2} = \frac{t}{(t+1)^2(t+2)} < (\mathbb{E}[X])^2$. This variance is too high to obtain even a 2-approximation estimate with high probability.

We use the same trick to reduce the variance, as in the last lecture: we keep track of q independent variables $X_1, \ldots, X_q$ and return

$$\hat{t} = \frac{1}{\frac{1}{q}\sum_{i=1}^q X_i} - 1.$$

We call this algorithm FM+. It holds that $\mathrm{Var}(\frac{1}{q}\sum_{i=1}^q X_i) = \frac{1}{q}\mathrm{Var}(X_1)$. Thus, by Chebychev's inequality, if $q = \frac{10}{\varepsilon^2}$, we get

$$\Pr\left[\left|\frac{1}{q}\sum_{i=1}^q X_i - \frac{1}{t+1}\right| > \frac{\varepsilon}{t+1}\right] \leq \frac{\mathrm{Var}(\frac{1}{q}\sum_{i=1}^q X_i)}{\frac{\varepsilon^2}{(t+1)^2}} \leq \frac{1}{q\varepsilon^2} = \frac{1}{10}.$$

Therefore, with probability at least 90%, $\frac{1}{q}\sum_{i=1}^q X_i \in \left[\frac{1-\varepsilon}{t+1}, \frac{1+\varepsilon}{t+1}\right]$.

We still need to reduce the failure probability of the algorithm. To achieve failure probability δ, we use $p = O(\log(1/\delta))$ copies of FM+ and output the median of the outputs. We call this algorithm FM++. As in the last lecture, by Chernoff bound, the probability that at least half of the copies fail is at most $\exp(\Omega(p)) = \delta$. Therefore, FM++ returns a $(1 \pm \varepsilon)$-approximation of t with probability at least $1 - \delta$.

The total space this algorithm uses is $O(p \cdot q) = O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ for the values it maintains (and the space to store a random hash function, which is linear in n and can't be avoided).

2.2 Flajolet-Martin algorithm: Non-Idealized Version

To replace the ideal random hash function, we use *$k-$wise independent hash functions.*

Definition 1. *A family $\mathcal{H}$ of hash functions $h : [a] \to [b]$ is k-wise independent iff for all distinct $i_1, \ldots, i_k \in [a]$ and for all $j_1, \ldots, j_k \in [b]$*

$$\Pr_{h \in \mathcal{H}}[h(i_1) = j_1 \wedge h(i_2) = j_2 \wedge \ldots \wedge h(i_k) = j_k] = \frac{1}{b^k}.$$

Intuitively, it simulates a random hash function if we take only k values. One such family is the set of all degree-$(k-1)$ polynomials in $\mathbb{F}_{q=a=b}$. We can store a hash function from this family in $O(\log(q^k)) = O(k \log(q))$ space.

For our problem, we will only need a 2-wise independent hash function $h : [n] \to [n]$ which can be stored in $O(\log(n))$ space. We assume for simplicity that n is a power of 2. We consider the following algorithm.

- Store the value $X = \max_{i \in \text{stream}} \text{lsb}(h(i))$.
- At the end, return $\hat{t} = 2^X$.

The function $\text{lsb}(a)$ returns the position of the least significant 1 in the binary representation of a, e.g. $\text{lsb}(100110_2) = 1$, $\text{lsb}(101000_2) = 3$.

Let Z_j be the number of distinct values i with $\text{lsb}(h(i)) = j$. It is easy to see that half of the values of the stream have $\text{lsb}(h(i)) = 0$. Therefore, $\mathbb{E}[Z_0] = \frac{t}{2}$. Similarly, it holds that $\mathbb{E}[Z_1] = \frac{t}{4}$ and

$$\mathbb{E}[Z_j] = \frac{t}{2^{j+1}}. \tag{1}$$

Let $Z_{>j} = Z_{j+1} + Z_{j+2} + \ldots$ be the number of distinct values i with $\text{lsb}(h(i)) > j$. We want to have $X \approx \log_2 t$. If we prove that with high probability

1. $Z_{>\log_2 t+5} = 0$ and
2. $Z_{\log_2 t-5} \geq 1$

then it holds that with high probability $\log_2 t - 5 \leq X \leq \log_2 t + 5$ and we have a 32-approximation of t.

Proof. We will first prove that $Z_{>\log_2 t+5} = 0$ holds w.h.p. and then that $Z_{\log_2 t-5} \geq 1$ holds w.h.p.

1. By linearity of expectation,

$$\begin{aligned}
\mathbb{E}[Z_{>\log_2 t+5}] &= \frac{t}{2^{\log_2 t+7}} + \frac{t}{2^{\log_2 t+8}} + \ldots && \text{(Equation (1))} \\
&= \frac{1}{2^7} + \frac{1}{2^8} + \ldots \\
&< \frac{1}{64}.
\end{aligned}$$

By Markov's inequality,

$$\Pr[Z_{>\log_2 t+5} \geq 1] \leq \frac{1/64}{1} = \frac{1}{64}.$$

Therefore with probability at least $1 - \frac{1}{64}$, $Z_{>\log_2 t+5} = 0$.

2. Recall that from equation (1),

$$\mathbb{E}[Z_{\log_2 t-5}] = \frac{t}{2^{\log_2 t-5}} = 16.$$

We need to calculate the variance of $Z_{\log_2 t-5}$. Let us fix j. We define $Y_i = \begin{cases} 1, & \text{if } \operatorname{lsb}(h(i)) = j \\ 0, & \text{otherwise} \end{cases}$.

It holds that $Z_j = \sum_{i \in \text{stream}} Y_i$ and $\Pr[Y_i = 1] = \frac{1}{2^{j+1}}$.

Since the Y_i's are pairwise independent (guaranteed by the choice of the 2-wise independent hash function), by linearity of the variance, we have

$$\begin{aligned} \operatorname{Var}(Z_j) &= \sum_{i \in \text{stream}} \operatorname{Var}(Y_i) \\ &= t \cdot \frac{1}{2^{j+1}} \cdot \left(1 - \frac{1}{2^{j+1}}\right) && (Y_i\text{'s are Bernoulli}(\tfrac{1}{2^{j+1}})) \\ &< \frac{t}{2^{j+1}} \end{aligned}$$

Therefore, $\operatorname{Var}(Z_{\log_2 t-5}) \leq 16$.

By Chebyshev's inequality,

$$\Pr[Z_{\log_2 t-5} = 0] \leq \frac{\operatorname{Var}(Z_{\log_2 t-5})}{(\mathbb{E}[Z_{\log_2 t-5}])^2} \leq \frac{16}{16^2} = \frac{1}{16}.$$

So with probability $1 - \frac{1}{16}$, $Z_{\log_2 t-5} \geq 1$.

Therefore, with probability $1 - \frac{1}{16} - \frac{1}{32}$, $Z_{>\log_2 t+5} = 0$ and $Z_{\log_2 t-5} \geq 1$ hold simultaneously. □

We conclude that with probability at least 90%, $\hat{t} \in [t/32, 32t]$. Since we have a constant approximation with constant probability, we can use the same techniques to improve the accuracy and the failure probability of our solution.

References

[1] Philippe Flajolet, G. Nigel Martin. Probabilistic Counting Algorithms for Data Base Applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.