(CS602) Applied Algorithms 13 Jan, 2014

Lecture 3: Computing the number of distinct elements

Lecturer: Nutan Limaye *Scribe: Nutan Limaye*

In the last class we gave a randomized (ε, δ) algorithm[1] for approximating the length of the input. In this class we will build towards coming up with an algorithm for approximating the number of distinct elements in the input stream.

Given an input stream $x_1, x_2, \ldots, x_n$, where each $x_i \in [m]$ and $\varepsilon, \delta > 0$ constants, our goal is to design a randomized (ε, δ) algorithm for approximating the number of distinct elements using space $O(\log m)$. We will give two approaches for this, which will introduce you to two ideas. Unfortunately, none of them will manage to achieve the correct space bounds. Finally, the exact space bounds will be achieved in the next class.

3.1 Approach 1

```
Pick $h$ uniformly randomly from $\mathcal{H} = \{h : [m] \to [0,1]\}$;
$Z \leftarrow 1$;
while there exists $x$, an input element do
    if $Z > h(x)$ then
        $Z \leftarrow h(x)$;
    end
end
```

The algorithm assigns minimum over all x, $h(x)$ to Z. For a fixed input string, suppose $D \subseteq [m]$ is the set of distinct elements in the string. Let $d = |D|$. The above algorithm takes the set D and maps it to the real interval $[0, 1]$. As this happens uniformly in the interval, intuitively the elements in D will be equispaced inside the interval $[0, 1]$. Therefore, suppose $min_{x \in D} h(x)$ is equal to 0.25 then it is very likely that there were $3 = (1/0.25 - 1)$ elements in D: one which got mapped to 0.25, another to 0.5 and the last to 0.75. It seems that by simply randomly hashing into the $[0, 1]$ interval, and computing the min of the hash values, we can get a good idea about the number of distinct elements in the input.

We will now make this intuition concrete in the next few steps of calculations. We will compute the expectation and variance of Z. Note that the probability is over the random choice of $h \in \mathcal{H}$.

[1]Defined in last lecture note.

$$\begin{aligned}
\mathbb{E}(Z) &= \int_0^1 \Pr\left[Z > \lambda\right] d\lambda \\
&= \int_0^1 \Pr\left[\prod_{x\in D} h(x) > \lambda\right] d\lambda && (Z \text{ is the min among all } h(x)) \\
&= \int_0^1 (1-\lambda)^d d\lambda && (\text{As } h(x)\text{s are uniformaly distributed over } [0,1] \text{ and } |D| = d) \\
&= \left.\frac{u^{d+1}}{d+1}\right|_0^1 && (\text{By change of variables}) \\
&= \frac{1}{d+1}
\end{aligned}$$

To compute $\mathbb{V}ar(Z)$, we first compute $\mathbb{E}(Z^2)$ similarly.

$$\begin{aligned}
\mathbb{E}(Z^2) &= \int_0^1 \Pr\left[Z^2 > \lambda\right] d\lambda \\
&= \int_0^1 \Pr\left[Z > \sqrt{\lambda}\right] d\lambda \\
&= \int_0^1 \Pr\left[\prod_{x\in D} \left(h(x) > \sqrt{\lambda}\right)\right] d\lambda && (Z \text{ is the min among all } h(x)) \\
&= \int_0^1 (1-\sqrt{\lambda})^d d\lambda && (\text{As } h(x)\text{s are uniformaly distributed over } [0,1] \text{ and } |D| = d) \\
&= 2\left(\frac{u^{d+1}}{d+1} - \frac{u^{d+2}}{d+2}\right)\Bigg|_0^1 && (\text{By change of variables}) \\
&= \frac{2}{(d+1)(d+2)}
\end{aligned}$$

From this we get that $\mathbb{V}ar(Z) = O(1/d^2)$.

To bring down the variance by a factor of q, we run q parallel copies of the algorithm and take average of their outputs and declare that as our new output. This is called the averaging trick and is decsribed in Lecture 2, Section 2.2. Let $\tilde{Z}$ denote the averaged output. Then it is easy to see that $\mathbb{E}(\tilde{Z}) = \mathbb{E}(Z)$ and $\mathbb{V}ar(\tilde{Z}) = \frac{\mathbb{V}ar(Z)}{q}$. By Chebyshev's inequality we will get the following:

$$\Pr_{h\in\mathcal{H}}\left[\left|\tilde{Z} - \frac{1}{d+1}\right| \geq \frac{\varepsilon}{d+1}\right] \leq O\left(\frac{1}{q\varepsilon^2}\right)$$

Now choosing q appropriately, we get the above probability is strictly smaller than 1/3.

This finishes the description of this approach. There are a couple of problems with this approach:

- As the range of the functions is $[0,1]$, the family $\mathcal{H}$ is uncountable. Therefore, we will not be able to implement the step of picking a random function from $\mathcal{H}$.

- Again, as Z takes value in $[0,1]$ infinite precision may be required to store Z.

Our next approach partially addresses these issues.

3.2 Approach 2

We first change the family of functions. Let $\mathcal{H}' = \{h : [m] \to \{0,1\}^K\}$, where K is such that $2^K > m$. Note that by changing the range of the functions, we manage to address the problems related to Approach 1. However, if we implement the exact same algorithm as in Section 3.1 with respect to this family then the algorithm will use $\log(2^{mK}) = O(m^2)$ bits to pick a random function.[2] This is better than Approach 1, however not good enough — as we already know a $O(m)$ algorithm for exactly computing the number of distinct elements in any input stream. So this change of family of functions is not quite enough to achieve good space bounds.

In any case, we will use this family $\mathcal{H}'$ to design a different randomized algorithm, say $\mathcal{A}_2$, to approximate the number of distinct elements. We will then observe some properties of the algorithm. These properties will indicate that to achieve the same performce guarantee — that is the same approximation factor and the same amount of error — we need not pick a random function from the family $\mathcal{H}'$ but from a sub-family $\mathcal{H}'' \subset \mathcal{H}'$. In the next class we will construct a sub-family $\mathcal{H}''$ and an algorithm $\mathcal{A}'_2$ such that $|\mathcal{H}''| = O(m^2)$ and $\mathcal{A}'_2$ is the same as $\mathcal{A}_2$, except that it chooses a random function from $\mathcal{H}''$. Note that picking a random function from $\mathcal{H}''$ will need only $O(\log m)$ bits.

We now describe the algorithm.

```
Pick $h$ uniformly randomly from $\mathcal{H}' = \{h : [m] \to \{0,1\}^K\}$;
$Z \leftarrow 0$;
while there exists $x$, an input element do
    if $Z <$ #traling zeroes in $h(x)$ then
        $Z \leftarrow h(x)$;
    end
end
Output $2^Z$;
```

Let D denote the distinct elements in the streams and $|D| = d$. The algorithm spreads the elements of D uniformaly in the range $\{0,1\}^K$. It then stores the maximum over $x \in D$ the number of trailing 0s observed in $h(x)$. The rationale behind counting the trailing zeroes can be explained as follows: consider the random experiment in which you pick a random subset from $\{1, 2, \ldots, n\}$. In this case, about half the elements will be divisible by 2, about 1/4th will be divisible by 4 and so on. Turning it around, if d elements are mapped uniformly and randomly to $\{1, 2, \ldots, n\}$ then at least 1 of them will be divisible by $2^{\log d}$.

We now work out the calculations. For the ease of calculations, let us introduce some notation. Let $Y_{x,\ell}$ be a 0-1 random variable, which is defined to be 1 iff #traling zeroes in $h(x) > \ell$. And let $\tilde{Y}_\ell$ denote $\sum_{x \in D} Y_{x,\ell}$.

[2]This is because $|\mathcal{H}'| = (2^K)^m$.

Observation 3.2.1. *$\tilde{Y}_\ell = 0$ if and only if $Z < \ell$.*

The proof of the above observation is left as an easy exercise. To bound the error probability of the algorithm, we would like to show that the following probability is high: $\Pr\left[\frac{d}{c} \le 2^Z \le cd\right]$. We will do this by bounding the probabilities of the following two quantities:

$$\Pr\left[2^Z > cd\right] \le \Pr\left[Z > \lceil \log(cd) \rceil\right] \qquad (A)$$

$$\Pr\left[2^Z < d/c\right] \le \Pr\left[Z < \lfloor \log(d/c) \rfloor\right] \qquad (B)$$

Let U denote $\lceil \log(cd) \rceil$ and L denote $\lfloor \log(d/c) \rfloor$ in $(A), (B)$, respectively. Using Observation 3.2.1 we know that $\Pr[Z < L] = \Pr\left[\tilde{Y}_L = 0\right]$. Similarly, we also know that $\Pr[Z > U] = \Pr\left[\tilde{Y}_U > 0\right]$. Therefore, now we are left to upper bound the probability of these two events. We bound them as follows:

$$\begin{aligned} \Pr\left[\tilde{Y}_U > 0\right] &= \Pr\left[\tilde{Y}_U \ge 1\right] \\ &\le \mathbb{E}(\tilde{Y}_U)/1 && \text{(By Markov's inequality)} \end{aligned}$$

$$\begin{aligned} \Pr\left[\tilde{Y}_L = 0\right] &= \Pr\left[\left|\tilde{Y}_L - \mathbb{E}(\tilde{Y}_L)\right| \ge \mathbb{E}(\tilde{Y}_L)\right] \\ &\le \mathbb{V}ar(\tilde{Y}_L)/(\mathbb{E}(\tilde{Y}_L))^2 && \text{(By Chebyshev's inequality)} \end{aligned}$$

Lemma 3.2.2. *For every $1 \le \ell \le K$, $\mathbb{E}(\tilde{Y}_\ell) = \frac{d}{2^l}$ and $\mathbb{V}ar(\tilde{Y}_\ell) < \frac{d}{2^l}$*

The proof was worked out in detail in class. It is now left as an exercise. Now putting it all together we get that $\Pr\left[\tilde{Y}_U > 0\right] \le \frac{d}{2^U} \le 1/c$. The first inequality is due to Lemma 3.2.2 and the second inequality is by using the value of U. Similarly, we get $\Pr\left[\tilde{Y}_L = 0\right] < \frac{2^L}{d} \le 1/c$. Again, the first inequality is due to Lemma 3.2.2 and the second inequality is by using the value of L.

3.3 Exercise

Exercise 3.1.1. *Suppose the algorithm for Approach 1 is changed as follows:*

```
Pick $h$ uniformly randomly from $\mathcal{H} = \{h : [m] \to [0,1]\}$;
$Z \leftarrow 0$;
while there exists $x$, an input element do
    if $Z < h(x)$ then
        $Z \leftarrow h(x)$;
    end
end
```

Compute $\mathbb{E}(Z)$ *and* $\mathbb{V}ar(Z)$*.*

Exercise 3.2.2. *Prove Observation 3.2.1.*

Exercise 3.3.3. *Prove Lemma 3.2.2.*