Counting different items in a stream

QT Curs 2020-2021

Counting the number of distinct elements

Counting the number of distinct elements

- Distinct elements problem: Given a stream s, output $|\{j \mid f_j > 0\}|$.
  where f_j is the frequency of the j in the stream s

Counting the number of distinct elements

- Distinct elements problem: Given a stream s, output $|\{j \mid f_j > 0\}|$.
  where f_j is the frequency of the j in the stream s
- In order to solve the problem using sublinear space, we need to use probabilistic algorithms/data structure and some adequate notion of approximation.

An (ϵ, δ)-approximation

An (ϵ, δ)-approximation

- Let $\mathcal{A}(s)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input s; note that this is a random variable.
- Let $\Phi(s)$ be the function that $\mathcal{A}$ is supposed to compute.

An (ϵ, δ)-approximation

- Let $\mathcal{A}(s)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input s; note that this is a random variable.
- Let $\Phi(s)$ be the function that $\mathcal{A}$ is supposed to compute.
- $\mathcal{A}$ is a (ϵ, δ)-approximation to Φ if we have

$$Pr\left[\left|\frac{\mathcal{A}(s)}{\Phi(s)} - 1\right| > \epsilon\right] \leq \delta.$$

An (ϵ, δ)-approximation

- Let $\mathcal{A}(s)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input s; note that this is a random variable.
- Let $\Phi(s)$ be the function that $\mathcal{A}$ is supposed to compute.
- $\mathcal{A}$ is a (ϵ, δ)-approximation to Φ if we have

$$Pr\left[\left|\frac{\mathcal{A}(s)}{\Phi(s)} - 1\right| > \epsilon\right] \leq \delta.$$

- $\mathcal{A}$ is a (ϵ, δ)-additive approximation to Φ if we have

$$Pr\left[|\mathcal{A}(s) - \Phi(s)| > \epsilon\right] \leq \delta.$$

An (ϵ, δ)-approximation

- Let $\mathcal{A}(s)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input s; note that this is a random variable.
- Let $\Phi(s)$ be the function that $\mathcal{A}$ is supposed to compute.
- $\mathcal{A}$ is a (ϵ, δ)-approximation to Φ if we have

$$Pr\left[\left|\frac{\mathcal{A}(s)}{\Phi(s)} - 1\right| > \epsilon\right] \leq \delta.$$

- $\mathcal{A}$ is a (ϵ, δ)-additive approximation to Φ if we have

$$Pr\left[|\mathcal{A}(s) - \Phi(s)| > \epsilon\right] \leq \delta.$$

- When $\delta = 0$, $\mathcal{A}$ must be deterministic.
  When $\epsilon = 0$, $\mathcal{A}$ must be an exact algorithm.

Counting the number of distinct elements

Counting the number of distinct elements

- For an integer $p > 0$, let zeros(p) be the number of zeros at the end of the binary representation of p.

Counting the number of distinct elements

- For an integer $p > 0$, let $\text{zeros}(p)$ be the number of zeros at the end of the binary representation of p.

$$\text{zeros}(p) = \max\{i \mid 2^i \text{ divides } p\}.$$

Counting the number of distinct elements

- For an integer $p > 0$, let zeros(p) be the number of zeros at the end of the binary representation of p.

$$\text{zeros}(p) = \max\{i \mid 2^i \text{ divides } p\}.$$

- Algorithm:

procedure COUNT-DIF(stream s)
    Choose a random hash function $h : [n] \to [n]$ form a universal family
    int $z = 0$
    while not $s.end()$ **do**
        $j = s.read()$
        if zeros$(h(j)) > z$ **then**
            $z = \text{zeros}(h(j))$
    Return $2^{z+\frac{1}{2}}$

Counting the number of distinct elements

- For an integer $p > 0$, let zeros(p) be the number of zeros at the end of the binary representation of p.

$$\text{zeros}(p) = \max\{i \mid 2^i \text{ divides } p\}.$$

- Algorithm:

```
procedure COUNT-DIF(stream s)
    Choose a random hash function h : [n] → [n] form a universal family
    int z = 0
    while not s.end() do
        j = s.read()
        if zeros(h(j)) > z then
            z = zeros(h(j))
    Return 2^(z+1/2)
```

- Assuming that there are d distinct elements, the algorithm computes $\max \text{zeros}(h(j))$ as a good approximation of $\log d$.

Counting the number of distinct elements: Quality

- 1 pass, $O(\log n + \log\log n)$ memory and $O(1)$ time per item.

Counting the number of distinct elements: Quality

- 1 pass, $O(\log n + \log \log n)$ memory and $O(1)$ time per item.
- For $j \in [n]$ and $r \geq 0$, let $X_{r,j}$ be the indicator r.v. for
  $$\text{zeros}(h(j)) \geq r.$$
- Let $Y_r = \sum_{j|f_j>0} X_{r,j}$.
- Let t denote the final value of z.

Counting the number of distinct elements: Quality

- 1 pass, $O(\log n + \log\log n)$ memory and $O(1)$ time per item.
- For $j \in [n]$ and $r \geq 0$, let $X_{r,j}$ be the indicator r.v. for
  $$\text{zeros}(h(j)) \geq r.$$
- Let $Y_r = \sum_{j|f_j>0} X_{r,j}$.
- Let t denote the final value of z.
- $Y_r > 0$ iff $t \geq r$, or equivalently $Y_r = 0$ iff $t \leq r - 1$.

Counting the number of distinct elements: Quality

- 1 pass, $O(\log n + \log\log n)$ memory and $O(1)$ time per item.
- For $j \in [n]$ and $r \geq 0$, let $X_{r,j}$ be the indicator r.v. for $\text{zeros}(h(j)) \geq r$.
- Let $Y_r = \sum_{j|f_j>0} X_{r,j}$.
- Let t denote the final value of z.
- $Y_r > 0$ iff $t \geq r$, or equivalently $Y_r = 0$ iff $t \leq r-1$.
- Since $h(j)$ is uniformly distributed over the $\log n$-bit strings,

$$E[X_{r,j}] = Pr[\text{zeros}(h(j)) \geq r] = Pr[2^r \text{ divides } h(j)] = \frac{1}{2^r}$$

Counting the number of distinct elements: Quality

$$E[X_{r,j}] = Pr[\text{zeros}(h(j)) \geq r] = Pr[2^r \text{ divides } h(j)] = \frac{1}{2^r}.$$

Counting the number of distinct elements: Quality

$$E[X_{r,j}] = Pr[\text{zeros}(h(j)) \geq r] = Pr[2^r \text{ divides } h(j)] = \frac{1}{2^r}.$$

$$E[Y_r] = \sum_{j|f_j>0} E[X_{r,j}] = \frac{d}{2^r}$$

Counting the number of distinct elements: Quality

$$E[X_{r,j}] = Pr[\text{zeros}(h(j)) \geq r] = Pr[2^r \text{ divides } h(j)] = \frac{1}{2^r}.$$

$$E[Y_r] = \sum_{j|f_j>0} E[X_{r,j}] = \frac{d}{2^r}$$

- Random variables Y_r are pairwise independent, as they come from a universal hash family.

$$Var[Y_r] = \sum_{j|f_j>0} Var[X_{r,j}] \leq \sum_{j|f_j>0} E[X_{r,j}^2] = \sum_{j|f_j>0} E[X_{r,j}] = \frac{d}{2^r}$$

Counting the number of distinct elements: Quality

- $E[Y_r] = Var[Y_r] = d/2^r$
- Using Markov's and Chebyshev's inequalities,

$$Pr[Y_r > 0] = Pr[Y_r \geq 1] \leq \frac{E[Y_r]}{1} = \frac{d}{2^r}.$$

$$Pr[Y_r = 0] = Pr[|Y_r - E[Y_r]| \geq \frac{d}{2^r}] \leq \frac{Var[Y_r]}{(d/2^r)^2} \leq \frac{2^r}{d}.$$

Counting the number of distinct elements: Quality

- $Pr[Y_r > 0] \leq \frac{d}{2^r}$ and $Pr[Y_r = 0] \leq \frac{2^r}{d}$.

Counting the number of distinct elements: Quality

- $Pr[Y_r > 0] \leq \frac{d}{2^r}$ and $Pr[Y_r = 0] \leq \frac{2^r}{d}$.
- Let $\hat{d}$ be the estimate of d, $\hat{d} = 2^{t+\frac{1}{2}}$.

Counting the number of distinct elements: Quality

- $Pr[Y_r > 0] \leq \frac{d}{2^r}$ and $Pr[Y_r = 0] \leq \frac{2^r}{d}$.
- Let $\hat{d}$ be the estimate of d, $\hat{d} = 2^{t+\frac{1}{2}}$.
- Let a be the smallest integer so that $2^{a+\frac{1}{2}} \geq 3d$,

$$Pr[\hat{d} \geq 3d] = Pr[t \geq a] = Pr[Y_a = 0] \leq \frac{d}{2^a} \leq \frac{\sqrt{2}}{3}.$$

Counting the number of distinct elements: Quality

- $Pr[Y_r > 0] \leq \frac{d}{2^r}$ and $Pr[Y_r = 0] \leq \frac{2^r}{d}$.
- Let $\hat{d}$ be the estimate of d, $\hat{d} = 2^{t+\frac{1}{2}}$.
- Let a be the smallest integer so that $2^{a+\frac{1}{2}} \geq 3d$,

$$Pr[\hat{d} \geq 3d] = Pr[t \geq a] = Pr[Y_a = 0] \leq \frac{d}{2^a} \leq \frac{\sqrt{2}}{3}.$$

- Let b be the largest integer so that $2^{b+\frac{1}{2}} \leq 3d$,

$$Pr[\hat{d} \leq 3d] = Pr[t \leq b] = Pr[Y_{b+1} = 0] \leq \frac{2^{b+1}}{d} \leq \frac{\sqrt{2}}{3}.$$

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.
- How to improve the quality of the approximation?

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.
- How to improve the quality of the approximation?
- Usual technique: run k several independent copies of the algorithm and take the best information from them, in this case,

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.
- How to improve the quality of the approximation?
- Usual technique: run k several independent copies of the algorithm and take the best information from them, in this case, the median of the k answers.

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.
- How to improve the quality of the approximation?
- Usual technique: run k several independent copies of the algorithm and take the best information from them, in this case, the median of the k answers.
  If the median exceed $3d$ at least $k/2$ of the runs do.

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.
- How to improve the quality of the approximation?
- Usual technique: run k several independent copies of the algorithm and take the best information from them, in this case, the median of the k answers.
  If the median exceed $3d$ at least $k/2$ of the runs do.
- By standard Chernoff bounds, the median exceed $3d$ with probability $2^{-\Omega(k)}$ and the median is below $3d$ with probability $2^{-\Omega(k)}$.

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.
- How to improve the quality of the approximation?
- Usual technique: run k several independent copies of the algorithm and take the best information from them, in this case, the median of the k answers.
  If the median exceed $3d$ at least $k/2$ of the runs do.
- By standard Chernoff bounds, the median exceed $3d$ with probability $2^{-\Omega(k)}$ and the median is below $3d$ with probability $2^{-\Omega(k)}$.
- Choosing $k = \Theta(\log(1/\delta))$, we can make the sum to be at most δ. So we get a $(2, \delta)$-approximation. However, the used memory is now $O(log(1/\delta) \log n)$.