LECTURE 7

Last time

- Tolerant testing and distance estimation
- Online erasure-resilient testing
- Other models of computation
- Streaming

Today

- Project discussion
- Counting the number of distinct elements in a stream

Sign up for project meetings (next week), scribing, grading

Streaming Puzzle

 A stream contains $n-1$ **distinct** elements from $[n]$ in arbitrary order.

Problem: Find the missing element, using $O(\log n)$ space.

Counting Distinct Elements

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

Goal: Output the number of distinct elements in the stream.

Exact solutions:

- Store the stream: $O(m \log n)$ bits.
- Store n bits, indicating whether each domain element has appeared.

Known lower bounds:

- Every deterministic algorithm requires $\Omega(m)$ bits (even for a constant-factor approximation).
- Every exact algorithm (even randomized) requires $\Omega(n)$ bits.

Need to use both randomization and approximation to get $\mathrm{polylog}(m, n)$ space

Counting Distinct Elements

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

Goal: Estimate the number of distinct elements in the stream up to a multiplicative factor $(1+\varepsilon)$ with probability $\geq 2/3$

- Studied by [Flajolet Martin 83, Alon Matias Szegedy 96,...]

- Today: $O(\varepsilon^{-2} \log n)$ space algorithm
  [Bar–Yossef Jayram Kumar Sivakuar Trevisan 02]

- Optimal: $O(\varepsilon^{-2} + \log n)$ space algorithm [Kane Nelson Woodruff 10]

Counting Distinct Elements

Input: a stream $\langle a_1, a_2, \ldots, a_m \rangle \in [n]^m$

Goal: Estimate the number of distinct elements in the stream up to a multiplicative factor $(1+\varepsilon)$ with probability $\geq 2/3$

Algorithm

1. Apply a random hash function $h : [n] \to [n]$ to each element.
2. Compute X, the t-th smallest value of the hash seen where $t = 10\ /\ \varepsilon^2$.
3. Return $\tilde{r} = t \cdot n/X$ as estimate for r, the number of distinct elements.

Analysis:

- Algorithm uses $O(\varepsilon^{-2} \log n)$ bits of space (not accounting for storing h).
- We'll show: estimate $\tilde{r}$ has good accuracy with reasonable probability.

Claim. $\Pr[|\tilde{r} - r| \leq \varepsilon r] \geq 2/3$

Counting Distinct Elements: Analysis

Claim. $\Pr[|\tilde{r} - r| \le \varepsilon r] \ge 2/3$

X: t-th smallest hashed value

$t = 10 / \varepsilon^2$

$\tilde{r} = t \cdot n / X$

Proof: Suppose the distinct elements are $e_1, \ldots, e_r$

- Overestimation:

$$\Pr[\tilde{r} \ge (1+\varepsilon)r] = \Pr\left[\frac{t \cdot n}{X} \ge (1+\varepsilon)r\right] = \Pr\left[X \le \frac{t \cdot n}{r(1+\varepsilon)}\right]$$

- Let $Y_i = \mathbb{1}\left[h(e_i) \le \frac{t \cdot n}{r(1+\varepsilon)}\right]$ and $Y = \sum_{i=1}^{r} Y_i$

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

$\mathrm{Var}[Y] \le \mathbb{E}[Y]$

$$\mathbb{E}[Y] = r \cdot \mathbb{E}[Y_1] = r \cdot \frac{t}{r(1+\varepsilon)} = \frac{t}{1+\varepsilon}$$

$$\mathrm{Var}[Y] = \mathrm{Var}\left[\sum_{i=1}^{r} Y_i\right] = \sum_{i=1}^{r} \mathrm{Var}[Y_i]$$

$$\le \sum_{i=1}^{r} \mathbb{E}[Y_i^2] = \sum_{i=1}^{r} \mathbb{E}[Y_i] = \mathbb{E}[Y]$$

Counting Distinct Elements: Analysis

Claim. $\Pr[|\tilde{r} - r| \le \varepsilon r] \ge 2/3$

X: t-th smallest hashed value

$t = 10 / \varepsilon^2$

$\tilde{r} = t \cdot n / X$

Proof: Suppose the distinct elements are $e_1, \dots, e_r$

- Overestimation:

$$\Pr[\tilde{r} \ge (1+\varepsilon)r] = \Pr\left[\frac{t \cdot n}{X} \ge (1+\varepsilon)r\right] = \Pr\left[X \le \frac{t \cdot n}{r(1+\varepsilon)}\right]$$

- Let $Y_i = \mathbb{1}\left[h(e_i) \le \frac{t \cdot n}{r(1+\varepsilon)}\right]$ and $Y = \sum_{i=1}^{r} Y_i$

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

$\mathrm{Var}[Y] \le \mathbb{E}[Y]$

$$\Pr\left[X \le \frac{t \cdot n}{r(1+\varepsilon)}\right] = \Pr[Y \ge t] = \Pr\big[Y \ge (1+\varepsilon)\mathbb{E}[Y]\big]$$

- By the Chebyshev's inequality, for $\varepsilon \le 2/3$,

$$\Pr\big[Y \ge (1+\varepsilon)\mathbb{E}[Y]\big] \le \frac{\mathrm{Var}[Y]}{(\varepsilon \cdot \mathbb{E}[Y])^2} \le \frac{1}{\varepsilon^2 \mathbb{E}[Y]} = \frac{1+\varepsilon}{\varepsilon^2 \cdot t} = \frac{1+\varepsilon}{10} \le \frac{1}{6}$$

- Underestimation: A similar analysis shows $\Pr[\tilde{r} \le (1-\varepsilon)r] \le \frac{1}{6}$

Removing the Random Hashing Assumption

Idea: Use limited independence

- A family $\mathcal{H} = \{h: [a] \to [b]\}$ of hash functions is k-wise independent if for all distinct $x_1, \ldots, x_k \in [a]$ and all $y_1, \ldots, y_k \in [b]$,

$$\Pr_{h \in \mathcal{H}}[h(x_1) = y_1, \ldots, h(x_k) = y_k] = \frac{1}{b^k}$$

  Note: a uniformly random family is k-wise independent for all k

- Observations: For $x_1, \ldots, x_k$ as above,
  1. $h(x_1)$ is uniform over $[b]$;
  2. $h(x_1), \ldots, h(x_k)$ are mutually independent.

Construction of k-wise Independent Family

Idea: Use limited independence

- A family $\mathcal{H} = \{h: [a] \to [b]\}$ of hash functions is k-wise independent if for all distinct $x_1, \ldots, x_k \in [a]$ and all $y_1, \ldots, y_k \in [b]$,

$$\Pr_{h \in \mathcal{H}}[h(x_1) = y_1, \ldots, h(x_k) = y_k] = \frac{1}{b^k}$$

Construction of k-wise Independent Family of Hash Functions

1. Let p be a prime.
2. Consider the set of polynomials of degree $k-1$ over $\mathbb{F}_p$
$$\mathcal{H} = \{h: \{0, \ldots, p-1\} \to \{0, \ldots, p-1\}$$
$$h(x) = c_{k-1}x^{k-1} + \cdots + c_1 x + c_0, \text{with } c_0, \ldots, c_{k-1} \in \mathbb{F}_p\}$$
3. To sample $h \in \mathcal{H}$, sample $c_0, \ldots, c_{k-1} \in \mathbb{F}_p$ u.i.r.

- Space to store h is $O(k \log p)$
- For arbitrary a, b, need $O\big(k \cdot (\log a + \log b)\big)$ space.

Counting Distinct Elements: Final Algorithm

Input: a stream $\langle a_1, a_2, \ldots, a_m \rangle \in [n]^m$

Goal: Estimate the number of distinct elements in the stream up to a multiplicative factor $(1 + \varepsilon)$ with probability $\geq 2/3$

Algorithm

1. Sample a hash function $h : [n] \to [n]$ from a 2-wise independent family and apply h to each element
2. Compute X, the t-th smallest value of the hash seen where $t = 10 / \varepsilon^2$
3. Return $\tilde{r} = t \cdot n/X$ as estimate for r, the number of distinct elements.

Analysis:

- Algorithm uses $O(\varepsilon^{-2} \log n)$ bits of space
- Our correctness analysis applies.

Frequency Moments Estimation

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

- The frequency vector of the stream is $f = (f_1, \dots, f_n)$, where f_i is the number of times i appears in the stream
- The p-th frequency moment is $F_p = \left|\left|f\right|\right|_p^p = \sum_{i=1}^n f_i^p$

  F_0 is the number of nonzero entries of f (# of distinct elements)

  $F_1 = m$ (# of elements in the stream)

  $F_2 = \left|\left|f\right|\right|_2^2$ is a measure of non-uniformity

  used e.g. for anomaly detection in network analysis

  $F_\infty = \max_i f_i$ is the most frequent element

Goal: Estimate F_p up to a multiplicative factor $(1 \pm \varepsilon)$ with probability $\geq 2/3$

Approximate Counting: Estimating F_1

Input: a stream $\langle a_1, a_2, \ldots, a_m \rangle \in [n]^m$

Warm-up: Compute m. How much space do you need?

Goal: Estimate m up to a multiplicative factor $(1 \pm \varepsilon)$ with probability $\geq \frac{2}{3}$

Today: $O(\varepsilon^{-2} \log \log m)$ space algorithm [Morris 78]

Morris Algorithm (initial version)

1. Initialize $X \leftarrow 0$
2. For each element, increment X by 1 w. p. 2^{-X}
3. Return $\tilde{m} = 2^X - 1$.

- Intuitively, X is keeping track of $\log(m+1)$
- Intuitively, expected increment to 2^X at each step is $2^X \cdot 2^{-X} = 1$.

Based on Jelani Nelson's lecture notes: https://www.sketchingbigdata.org/fall20/lec/notes.pdf

Morris Algorithm: Analysis

Morris Algorithm (initial version)

1. Initialize $X \leftarrow 0$
2. For each element, increment X by 1 w. p. 2^{-X}
3. Return $\tilde{m} = 2^X - 1$.

- Let X_i represent X after i elements.
- $2^{X_0} = 1$
- $\mathbb{E}[2^{X_i}] = \mathbb{E}\big[\mathbb{E}[\,2^{X_i} \mid X_{i-1}\,]\big]$

By the compact form of the Law of Total Expectation

$$= \mathbb{E}[2^{X_{i-1}+1} \cdot 2^{-X_{i-1}} + 2^{X_{i-1}} \cdot (1 - 2^{-X_{i-1}})]$$

$$= \mathbb{E}[2 + 2^{X_{i-1}} - 1] = \mathbb{E}[2^{X_{i-1}}] + 1 = i + 1$$

$\mathbb{E}[2^X] = m + 1$

Claim. $\mathrm{Var}[2^X] \leq m^2/2$

Variance Calculation

Claim. $\mathrm{Var}[2^X] \leq m^2/2$

Proof: $\mathrm{Var}[2^{X_i}] = \mathbb{E}[(2^{X_i})^2] - \mathbb{E}[2^{X_i}]^2$ — by definition of variance

$= \mathbb{E}[2^{2X_i}] - (i+1)^2$ — by our calculation of expectation

$\mathbb{E}[2^{2X_i}] = \mathbb{E}[\mathbb{E}[2^{2X_i} \mid X_{i-1}]]$ — by compact form of Law of Total Expectation

$= \mathbb{E}[2^{2(X_{i-1}+1)} \cdot 2^{-X_{i-1}} + 2^{2X_{i-1}}(1-2^{-X_{i-1}})]$

$= \mathbb{E}[2^{X_{i-1}+2} + 2^{2X_{i-1}} - 2^{X_{i-1}}]$

$= \mathbb{E}[3 \cdot 2^{X_{i-1}} + 2^{2X_{i-1}}] = 3i + \mathbb{E}[2^{2X_{i-1}}]$

$= 1 + 3(1+2+\ldots+i) = 1 + 3\frac{i(i+1)}{2}$ — by induction

$\mathrm{Var}[2^{X_i}] = \mathbb{E}[2^{2X_i}] - (i+1)^2$

$= 1 + \frac{3}{2}i^2 + \frac{3}{2}i - i^2 - 2i - 1$

$= \frac{i^2}{2} - \frac{i}{2} \leq \frac{i^2}{2}$

Recall that $X = X_m$.

Morris Algorithm: Analysis

Morris Algorithm (initial version)

1. Initialize $X \leftarrow 0$
2. For each element, increment X by 1 w. p. 2^{-X}
3. Return $\tilde{m} = 2^X - 1$.

- Let X_i represent X after i elements.
- $2^{X_0} = 1$
- $\mathbb{E}[2^{X_i}] = \mathbb{E}\big[\mathbb{E}[\,2^{X_i} \mid X_{i-1}\,]\big]$

By the compact form of the Law of Total Expectation

$$= \mathbb{E}[2^{X_i} \cdot 2^{-X_{i-1}} + 2^{X_{i-1}} \cdot (1 - 2^{-X_{i-1}})]$$

$$= \mathbb{E}[2 + 2^{X_{i-1}} - 1] = \mathbb{E}[2^{X_{i-1}}] + 1 = i + 1$$

$\mathbb{E}[2^X] = m + 1$

Claim. $\mathrm{Var}[2^X] \le m^2/2$

- By Chebyshev, $\Pr[|\tilde{m} - m| \ge \varepsilon m] \le \frac{\mathrm{Var}[\tilde{m}]}{(\varepsilon \cdot m)^2} \le \frac{1}{2\varepsilon^2}$
- Idea: to reduce variance, keep t independent counters and average their estimates.

Morris Algorithm: Improvement

Morris Algorithm

1. Initialize t independent counters $X \leftarrow 0$
2. For each element, increment each X by 1 w. p. 2^{-X}
3. Return $\widetilde{m} =$ the average of $2^X - 1$ over all counters

- Then $E[\widetilde{m}]$ remains m
- But Var$[\widetilde{m}]$ is $\frac{1}{t} \cdot \mathrm{Var}[2^X]$

$\mathbb{E}[2^X] = m + 1$

Claim. $\mathrm{Var}[2^X] \leq m^2/2$

- By Chebyshev, $\Pr[|\widetilde{m} - m| \geq \varepsilon m] \leq \frac{\mathrm{Var}[\widetilde{m}]}{(\varepsilon \cdot m)^2} \leq \frac{1}{2t\varepsilon^2}$
- It is sufficient to set $t = O\left(\frac{1}{\varepsilon^2}\right)$

Summary

Streaming Model

- Reservoir sampling
- Distinct Elements (approximating F_0)
- k-wise independent hashing
- Morris counter