Randomised Algorithms

Lecture 13: Streaming Algorithms

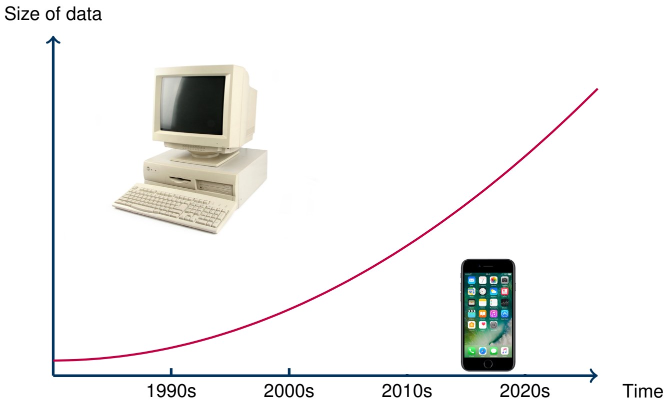
Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2022

Outline

Background of Streaming Algorithms

- The amount of data has been increased exponentially over the last years
- For many applications computational devices' memories are limited
- We need to find good (approximate) solutions without storing the entire input!



Motivation: Analysing Search Engine Queries

- What is the total number queries?
- What is the total number of different IP addresses?
- Extension 1: only consider queries within a certain interval (sliding window)
- Extension 2: also allow the cancellation/removal of a query (turnstile model)
- Extension 3: What if we have different data centers? (distributed streaming)

⋮

- memory is much smaller than needed to store entire data stream

⇒ We can only read each data item once and in sequential order

IP:
54.73.1
Time:
Text:

Other Applications:

- Monitoring Financial Transactions
- Analysing Buying Histories of Users

.....

Streaming algorithms

- The input of a streaming algorithm is given as a data stream, which is a sequence of data
$$\mathcal{S} = s_1, s_2, \ldots, s_i, \ldots$$
and every s_i belongs to the universe U.

- Constraints for streaming algorithms: the space complexity should be sublinear in $|U|$ and $|\mathcal{S}|$.

- Quality of the output: The algorithm needs to give a good approximate value with high probability.

(ε, δ)-approximation

For confidence parameter δ and approximation parameter ϵ, the algorithm's output Output and the exact answer Exact satisfies

$$\mathbf{P}\left[\,\text{Output} \in (1-\varepsilon, 1+\varepsilon)\cdot \text{Exact}\,\right] \geq 1-\delta.$$

Outline

Introduction

Approximate Counting

Distinct Elements and Frequency Moments

Extra Material (non-examinable): An Algorithm for F_0 in the Turnstile Model

Approximate Counting and Morris Algorithm

This could be also described as a data structure maintaining an integer n and supporting two operations:

- `update()`: increment n by 1
- `query()`: output n

Approximate Counting

An approximate counting algorithm must monitor a sequence of events. At any given time, the algorithm must output an **estimate** of the number of events.

Trivial (and exact) solution uses $\log_2 n$ space. Can we do better?

MORRIS ALGORITHM

1: $X \leftarrow 0$
2: **While** `update()`
3: With probability 2^{-X} set $X \leftarrow X + 1$
4: **Return** $2^X - 1$

Intuition: X will be an approximation of $\log_2 n$ (that is, we try to approximate the number of bits of n in binary)

Analysis (1/3)

Lemma (Expectation Analysis)

Let X_n denote the value of X after n updates. For every $n \geq 0$,

$$\mathbf{E}\left[2^{X_n}\right] = n + 1.$$

Hence $\Theta_n := 2^{X_n} - 1$ is an unbiased estimator of n.

Proof:

- Base case: For $n = 0$, we have $X_n = X_0 = 0$ ✓
- Induction step: $n \to n+1$: By conditioning on X_n,

$$\begin{aligned}
\mathbf{E}\left[2^{X_{n+1}}\right] &= \sum_{j=0}^{\infty} \mathbf{P}\left[X_n = j\right] \cdot \mathbf{E}\left[2^{X_{n+1}} \mid X_n = j\right] \\
&= \sum_{j=0}^{\infty} \mathbf{P}\left[X_n = j\right] \cdot \left(2^j \cdot \left(1 - \frac{1}{2^j}\right) + 2^{j+1} \cdot \frac{1}{2^j}\right) \\
&= \sum_{j=0}^{\infty} \mathbf{P}\left[X_n = j\right] \cdot 2^j + \sum_{j=0}^{\infty} \mathbf{P}\left[X_n = j\right] \\
&= \mathbf{E}\left[2^{X_n}\right] + 1 \\
&= (n+1) + 1.
\end{aligned}$$

By Induction Hypothesis (last step)

Analysis (2/3)

Lemma (Second Moment Analysis)

Let X_n denote the value of X after n updates. For every $n \geq 0$,

$$\mathbf{E}\left[\left(2^{X_n}\right)^2\right] = \mathbf{E}\left[2^{2 \cdot X_n}\right] = \frac{3}{2}n^2 + \frac{3}{2}n + 1.$$

This is shown similarly to that of the previous Lemma (see supervision sheet)

- Recall $\Theta_n = 2^{X_n} - 1$.
- Since $\mathbf{V}[Z] = \mathbf{E}[Z^2] - \mathbf{E}[Z]^2$,

$$\begin{aligned}\mathbf{V}[\Theta_n] = \mathbf{V}\left[2^{X_n}\right] &= \mathbf{E}\left[2^{2 \cdot X_n}\right] - \left(\mathbf{E}\left[2^{X_n}\right]\right)^2 \\ &= \frac{3}{2}n^2 + \frac{3}{2}n + 1 - (n+1)^2 = \frac{n^2 - n}{2}\end{aligned}$$

- Using Chebysheff's inequality,

This failure probability (estimate) is at least $\frac{1}{2}$ ☹

$$\mathbf{P}\left[|\Theta_n - n| \geq \epsilon \cdot n\right] \leq \frac{\mathbf{V}[\Theta_n]}{\epsilon^2 \cdot n^2} \leq \frac{\frac{n^2}{2}}{\epsilon^2 \cdot n^2} = \frac{1}{2\epsilon^2}.$$

Analysis (3/3)

Idea: Reduce Variance by Running Independent Instances and Taking Average.

IMPROVED MORRIS ALGORITHM(G)
1: Let $\Theta^1, \Theta^2, \ldots, \Theta^k$ be k independent instances of MORRIS
2: **Return** $\overline{\Theta} := \frac{1}{k} \sum_{i=1}^{k} \Theta^i$

- Clearly, $\mathbf{E}\left[\overline{\Theta}\right] = n$. For the variance,

$$\mathbf{V}\left[\overline{\Theta}\right] = \frac{1}{k^2} \cdot \mathbf{V}\left[\sum_{i=1}^{k} \Theta^i\right] = \frac{1}{k} \cdot \mathbf{V}\left[\Theta^1\right] \leq \frac{1}{k} \cdot \frac{n^2}{2}$$

- Hence using Chebyshev,

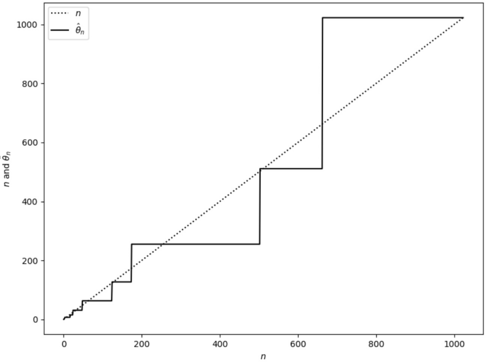
$$\mathbf{P}\left[\left|\overline{\Theta} - n\right| \geq \epsilon \cdot n\right] \leq \frac{1}{2k\epsilon^2}.$$

Conclusion

For any $\varepsilon, \delta < 1$, the IMPROVED MORRIS ALG. with $k \geq \frac{1}{2\epsilon^2\delta}$ satisfies:

$$\mathbf{P}\left[\left|\overline{\Theta} - n\right| \leq \epsilon \cdot n\right] \geq 1 - \delta.$$

Simulation



A run of Morris's algorithm on $n = 1024$ data points

(source: `http://gregorygundersen.com/blog/2019/11/11/morris-algorithm/`)

Outline

Norm Estimation: the Alon-Matias-Szegedy algorithm

F_p-norm (Frequency Moments)

Let U with $|U| = n$. For $i \in U$, let f_i be the number of occurrences of $i \in U$ in the stream $\mathcal{S}$. Then for any $p > 0$, the F_p-norm is defined by

$$F_p := \sum_{i \in U} f_i^p.$$

- F_1 = total number of items in stream $\mathcal{S}$.
- F_0 = total number of distinct items in stream $\mathcal{S}$.

Alon, Matias, and Szegedy (1996) presented a systematical study for approximating frequency moments.

- F_0, F_1, F_2 can be approximated in space logarithmic in n and $|\mathcal{S}|$.
- Approximating F_p for $p \geq 6$ requires $n^{\Omega(1)}$ space.
- The paper won 2005 Gödel Award for "their foundational contribution to streaming algorithms".

Important Tool: Pairwise independent Hash Functions

We will focus on the simpler case of F_0, the number of distinct elements.

Pairwise Independence

A family of functions $H = \{h \mid h : U \mapsto [n]\}$ is pairwise independent if, for any h chosen uniformly at random from H, the following holds:

1. $h(x)$ is uniformly distributed in $[n] = \{1, 2, \ldots, n\}$ for any $x \in U$;
2. For any $x_1 \neq x_2 \in U$, $h(x_1)$ and $h(x_2)$ are independent.

Theorem (Fact)

Let n be a prime number, and let $h_{a,b}(x) = (ax + b) \bmod n$. Define

$$H = \{h_{a,b} \mid 0 \leq a, b \leq n - 1\}.$$

Then H is a family of pairwise independent hash functions.

Intuition behind the AMS algorithm

Assume that we have a random hash function h. Define

$$\rho(x) := \max_{i \geq 0} \left\{ i : x \bmod 2^i = 0 \right\},$$

which is the number of consecutive 0's among the lowest bits of x.

Example: $\rho(2) = 1, \rho(3) = 0, \rho(4) = 2, \rho(8) = 3, \rho(16) = 4, \rho(17) = 0$.

Observation. Since $h(x)$ is uniformly distributed over $[n]$, the following holds:

- with probability $1/2$, we have $\rho(h(x)) \geq 1$
- with probability $1/4$, we have $\rho(h(x)) \geq 2$
- with probability $1/8$, we have $\rho(h(x)) \geq 3$

$\vdots$

- with probability $1/2^r$, we have $\rho(h(x)) \geq r$

Since n is not a power of 2, this probability is in fact equal to $\frac{\lfloor n/2^r \rfloor}{n} \approx 1/2^r - o(1)$.

The AMS Algorithm

AMS ALGORITHM

```
Choose a random hash function h : [n] → [n]
Z ← 0
while item x from stream S arrives
    if ρ(h(x)) > Z then Z ← ρ(h(x))
return 2^{Z+1/2}
```

$Z \leftarrow \max\{Z, \rho(h(x))\}$

Analysis of AMS Algorithm

With constant probability > 0, the algorithm's output satisfies

$$2^{Z+1/2} \in [F_0/3, 3 \cdot F_0].$$

We get an $(O(1), \delta)$-approximation of F_0 by running $\Theta(\log(1/\delta))$ independent copies of the algorithm and returning the median.

Recall (ε, δ)-approximation:
$\mathbf{P}[\,\text{Output} \in (1-\varepsilon, 1+\varepsilon) \cdot \text{Exact}\,] \geq 1 - \delta$

Example of the AMS Algorithm

- Assume $n = 101$ (which is prime)
- The hash function is $h(x) = (ax + b) \bmod n$ with $a = 28$, $b = 16$
- The data stream is:

$$\mathcal{S} = (25, 76, 14, 51, 25, 14, 76, 76, 3, 51, 96, 14, 67, 3, 15, 25, 2, 76, 14, 71)$$

- $F_0 = 10$, as the following numbers appeared: $\{2, 3, 14, 15, 25, 51, 67, 71, 76, 96\}$

x	$h(x)$	Binary Representation							$\rho(h(x))$
2	72	1	0	0	1	0	0	0	3
3	100	1	1	0	0	1	0	0	2
14	4	0	0	0	0	1	0	0	2
15	32	0	1	0	0	0	0	0	5
25	9	0	0	0	1	0	0	1	0
51	30	0	0	1	1	1	1	0	1
67	74	1	0	0	1	0	1	0	1
71	85	1	0	1	0	1	0	1	0
76	23	0	0	1	0	1	1	1	0
96	78	1	0	0	1	1	1	0	1

returned estimate: $2^{5+1/2} \approx 45.25$

Analysis (1/2)

Let $X_{r,j}$ be a 0/1 indicator random variable such that

$$X_{r,j} = 1 \Leftrightarrow \rho(h(j)) \geq r.$$

We say item j reaches level r if $X_{r,j} = 1$.

Let $Y_r = \sum_{j \in \mathcal{S}} X_{r,j}$ be the number of items j reaching level r.

Using that $h(j)$ is uniformly distributed, we conclude

$$\mathbf{E}\left[X_{r,j}\right] = \mathbf{P}\left[\rho(h(j)) \geq r\right] = \mathbf{P}\left[h(j) \bmod 2^r = 0\right] = 2^{-r}.$$

definition of function ρ

By linearity of expectation, we have

$$\mathbf{E}\left[Y_r\right] = \sum_{j \in \mathcal{S}} \mathbf{E}\left[X_{r,j}\right] = \frac{F_0}{2^r},$$

$$\mathbf{V}\left[Y_r\right] = \sum_{j \in \mathcal{S}} \mathbf{V}\left[X_{r,j}\right] \leq \sum_{j \in \mathcal{S}} \mathbf{E}\left[X_{r,j}^2\right] = \sum_{j \in \mathcal{S}} \mathbf{E}\left[X_{r,j}\right] = \frac{F_0}{2^r}$$

using pairwise independence of h!

Analysis (2/2)

We have proved $\mathbf{E}[\,Y_r\,] = \frac{F_0}{2^r}$ and $\mathbf{V}[\,Y_r\,] \leq \frac{F_0}{2^r}$.

By Markov's inequality, we have

$$\mathbf{P}[\,Y_r > 0\,] = \mathbf{P}[\,Y_r \geq 1\,] \leq \frac{\mathbf{E}[\,Y_r\,]}{1} = \frac{F_0}{2^r}.$$

By Chebyshev's inequality, we have

$$\mathbf{P}[\,Y_r = 0\,] \leq \mathbf{P}\left[\,|Y_r - \mathbf{E}[\,Y_r\,]\,| \geq F_0/2^r\,\right] \leq \frac{\mathbf{V}[\,Y_r\,]}{(F_0/2^r)^2} \leq \frac{2^r}{F_0}.$$

Let Z be the final integer the algo. keeps. So the algo. returns $2^{Z+1/2}$.

Let p be the smallest integer such that $2^{p+1/2} \geq 3F_0$:

$$\mathbf{P}\left[2^{Z+1/2} \geq 3F_0\right] = \mathbf{P}[\,Z \geq p\,] = \mathbf{P}[\,Y_p > 0\,] \leq \frac{F_0}{2^p} \leq \frac{\sqrt{2}}{3}.$$

Let q be the largest integer such that $2^{q+1/2} \leq F_0/3$:

Union Bound: Error $\leq 2 \cdot \frac{\sqrt{2}}{3} < 1$

$$\mathbf{P}\left[2^{Z+1/2} \leq F_0/3\right] = \mathbf{P}[\,Z \leq q\,] \leq \mathbf{P}[\,Y_{q+1} = 0\,] \leq \frac{2^{q+1}}{F_0} \leq \frac{\sqrt{2}}{3}. \qquad \square$$

Final Remarks

- Durand and Flajolet (2003) proposed the LogLog algorithm for estimating F_0
- Their algorithm condenses the whole of Shakespeare's works to a table of 256 “small bytes” of 4 bits each
- The estimate of the number of distinct words is $\widetilde{F_0} = 30897$, while the true answer is $F_0 = 28239$, which represents a relative error $+9.4\%$.

```
ghfffghfghgghggggghghheehfhfhhgghghghhfgffffhhhiigfhhffgfiihfhhh
igigighfgihfffghigihghigfhhgeegeghgghhhgghhfhidiigihighihehhhfgg
hfgighigffghdieghhhggghhfghhfiiheffghghihifgggffihgihfggighgiiif
fjgfgjhhjiifhjgehgghfhhfhjhigggghghihigghhihihgiighgfhlgjfgjjjmfl
```

Outline

The AMS algorithm (cash register model)

Common approach for designing algorithms in the cash register model:

1. Sample the data items based on hashed values;
2. Store the statistical information of the sampled items, or store the sampled items directly.

Downside of this framework:

- Sampling probability for the current item usually depends on the whole data stream that algorithm has seen so far.
- Deleting an item appeared before could potentially make the current statistical information useless! :(

Sampling techniques are usually non-applicable in the turnstile model.

Algorithm to approximate F_2 in the turnstile model

Algorithm to approximate F_2 (simplified description)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
2: $y = 0$
3: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives
4: **if** x is inserted **then** $y \leftarrow y + h(x)$
5: **else** $y \leftarrow y - h(x)$
6: **return** $Z := y^2$

The algorithm runs in the turnstile model!

Key Lemma

It holds that $\mathbf{E}[Z] = F_2$ and $\mathbf{V}[Z] \leq 2 \cdot \left(\sum_{i \in \mathcal{S}} m_i^2\right)^2 = 2F_2^2$.

Hence, we can (ε, δ)-approximate F_2, by **running multiple copies of the algorithm in parallel and return the average value.**

Algorithm to approximate F_2 in the turnstile model

Algorithm to approximate F_2 (details)

$t = \lceil 6/\varepsilon^2 \rceil$
Choose t **4-wise independent hash function** $h_1, \ldots, h_t$, where

$$h_i : [n] \to \{-1, 1\}$$

$y_i = 0$ for each $i = 1, \ldots, t$
while item $(x, \pm)$ from stream $\mathcal{S}$ arrives
  if x is inserted **then** $y_i = y_i + h_i(x)$ for every $1 \leq i \leq t$
    else $y_i = y_i - h_i(x)$ for every $1 \leq i \leq t$
return $\frac{1}{t} \cdot \sum_{i=1}^{t} Z_i$, where $Z_i = y_i^2$

Analysis

With constant probability, the returned value of the algorithm lies in $(1 - \varepsilon, 1 + \varepsilon) \cdot F_2$. Moreover, the space complexity is $O\left((1/\varepsilon^2) \log n\right)$ bits.