Algorithms for Big Data (I)

Chihao Zhang

Shanghai Jiao Tong University

Sept. 20, 2019

Course Information

- Instructor: Chihao Zhang

Course Information

- Instructor: Chihao Zhang
- Course Homepage: `http://chihaozhang.com/teaching/BDA2019fall`

Course Information

- Instructor: Chihao Zhang
- Course Homepage: `http://chihaozhang.com/teaching/BDA2019fall`
- Time: Every Friday, 12:55 - 15:40

Course Information

- Instructor: Chihao Zhang
- Course Homepage: `http://chihaozhang.com/teaching/BDA2019fall`
- Time: Every Friday, 12:55 - 15:40
- Office hour: Every Monday, 18:00 - 20:00

Course Information

- Instructor: Chihao Zhang
- Course Homepage: `http://chihaozhang.com/teaching/BDA2019fall`
- Time: Every Friday, 12:55 - 15:40
- Office hour: Every Monday, 18:00 - 20:00
- Grading: Homework 60%, Final Exam 40%

Course Information

- Instructor: Chihao Zhang
- Course Homepage: `http://chihaozhang.com/teaching/BDA2019fall`
- Time: Every Friday, 12:55 - 15:40
- Office hour: Every Monday, 18:00 - 20:00
- Grading: Homework 60%, Final Exam 40%
- Pre-request: Algorithms, Basic Probability Theory

Algorithms

Algorithms

We learnt many efficient algorithms before...

Algorithms

We learnt many efficient algorithms before...

- Dijkstra algorithm, Floyd algorithm, Blossom algorithm...
- These algorithms costs polynomial-time.

Algorithms

We learnt many efficient algorithms before...

- Dijkstra algorithm, Floyd algorithm, Blossom algorithm...
- These algorithms costs polynomial-time.

What if the input is too large to store?

Algorithms

We learnt many efficient algorithms before...

- Dijkstra algorithm, Floyd algorithm, Blossom algorithm...
- These algorithms costs polynomial-time.

What if the input is too large to store?

Throw some of them away! sublinear space algorithms.

A programmer for routers

A programmer for routers

A router has limited memory, but needs to process large data...

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

$$23$$

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

38

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

45

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

28

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

11

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

$$10$$

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

36

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

17

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

$$0$$

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

2

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

23

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

40

A programmer for routers

A router has limited memory, but needs to process large data…

The router can monitor the id of devices connecting to it.

$$23$$

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

18

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

24

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

31

A programmer for routers

A router has limited memory, but needs to process large data…

The router can monitor the id of devices connecting to it.

$$3$$

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

48

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

25

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

$$43$$

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

14

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

21

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

17

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

46

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

46

- How many numbers?

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

46

- How many numbers?
- How many distinct numbers?

A programmer for routers

A router has limited memory, but needs to process large data...

The router can monitor the id of devices connecting to it.

46

- How many numbers?
- How many distinct numbers?
- What is the most frequent number?

Streaming Model

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

One can process the input stream using at most s *bits* of memory

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

One can process the input stream using at most s *bits* of memory

We say the algorithm is sublinear if $s = o(\min\{m, n\})$.

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

One can process the input stream using at most s *bits* of memory

We say the algorithm is sublinear if $s = o(\min\{m, n\})$.

We can ask

- How many numbers (what is m?)

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

One can process the input stream using at most s *bits* of memory

We say the algorithm is sublinear if $s = o(\min\{m, n\})$.

We can ask

- How many numbers (what is m?)
- How many distinct numbers?

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

One can process the input stream using at most s *bits* of memory

We say the algorithm is sublinear if $s = o(\min\{m, n\})$.

We can ask

- How many numbers (what is m?)
- How many distinct numbers?
- What is the median of σ?

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

One can process the input stream using at most s *bits* of memory

We say the algorithm is sublinear if $s = o(\min\{m, n\})$.

We can ask

- How many numbers (what is m?)
- How many distinct numbers?
- What is the median of σ?
- What is the most frequent number?

Streaming Model

The input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_i \in [n]$

One can process the input stream using at most s *bits* of memory

We say the algorithm is sublinear if $s = o(\min\{m, n\})$.

We can ask

- How many numbers (what is m?)
- How many distinct numbers?
- What is the median of σ?
- What is the most frequent number?
- ...

How many numbers?

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many bits of memory needed?

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many bits of memory needed? $\log_2 m$.

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many bits of memory needed? $\log_2 m$.

Can be improved to $o(\log m)$?

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many bits of memory needed? $\log_2 m$.

Can be improved to $o(\log m)$?

Impossible

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many bits of memory needed? $\log_2 m$.

Can be improved to $o(\log m)$?

Impossible (Why?)

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many bits of memory needed? $\log_2 m$.

Can be improved to $o(\log m)$?

Impossible (Why?)

Possible if allow approximation:

How many numbers?

We can maintain a counter k. Whenever one reads a number a_i, let $k = k + 1$.

How many bits of memory needed? $\log_2 m$.

Can be improved to $o(\log m)$?

Impossible (Why?)

Possible if allow approximation:

For every $\varepsilon > 0$, compute a number $\widehat{m}$ such that

$$1 - \varepsilon \leq \frac{\widehat{m}}{m} \leq 1 + \varepsilon.$$

Morris' algorithm

Morris' algorithm

Algorithm Morris' Algorithm for Counting Elements

Init:

A variable $X \leftarrow 0$.

On Input y:

increase X with probability 2^{-X}.

Output:

Output $\widehat{m} = 2^X - 1$.

- This is a randomized algorithm.

Morris' algorithm

Algorithm Morris' Algorithm for Counting Elements

Init:

A variable $X \leftarrow 0$.

On Input y**:**

increase X with probability 2^{-X}.

Output:

Output $\widehat{m} = 2^X - 1$.

- This is a randomized algorithm.
- Therefore we look at the expectation of its output.

Analysis

Analysis

The output $\widehat{m}$ is a random variable, we prove that its expectation $\mathbf{E}\left[\widehat{m}\right] = m$ by induction on m.

Analysis

The output $\widehat{m}$ is a random variable, we prove that its expectation $\mathbf{E}\left[\widehat{m}\right] = m$ by induction on m.

Since $X = 1$ when $m = 1$, we have $\mathbf{E}\left[\widehat{m}\right] = 1$.

Analysis

The output $\widehat{m}$ is a random variable, we prove that its expectation $\mathbf{E}[\widehat{m}] = m$ by induction on m.

Since $X = 1$ when $m = 1$, we have $\mathbf{E}[\widehat{m}] = 1$.

Assume it is true for smaller m, let X_i denote the value of X after processing ith input.

Analysis (cont'd)

Analysis (cont'd)

$$
\begin{aligned}
\mathbf{E}\left[\widehat{m}\right] &= \mathbf{E}\left[2^{X_m}\right] - 1 \\
&= \sum_{i=0}^{m} \mathbf{Pr}\left[X_m = i\right] \cdot 2^i - 1 \\
&= \sum_{i=0}^{m} \left(\mathbf{Pr}\left[X_{m-1} = i\right]\left(1 - 2^{-i}\right) + \mathbf{Pr}\left[X_{m-1} = i-1\right] \cdot 2^{1-i}\right) \cdot 2^i - 1 \\
&= \sum_{i=0}^{m-1} \mathbf{Pr}\left[X_{m-1} = i\right]\left(2^i + 1\right) - 1 \\
&= \mathbf{E}\left[2^{X_{m-1}}\right] \\
&= m \quad \text{(induction hypothesis)}
\end{aligned}
$$

It is now clear that Morris' algorithm is an unbiased estimator for m.

It is now clear that Morris' algorithm is an unbiased estimator for m.

It uses approximately $O(\log \log m)$ bits of memory.

It is now clear that Morris' algorithm is an unbiased estimator for m.

It uses approximately $O(\log \log m)$ bits of memory.

However, for a practical randomized algorithm, we further require its output to concentrate on the expectation.

It is now clear that Morris' algorithm is an unbiased estimator for m.

It uses approximately $O(\log \log m)$ bits of memory.

However, for a practical randomized algorithm, we further require its output to concentrate on the expectation.

That is, we want to establish concentration inequality of the form

$$\mathbf{Pr}\left[|\widehat{m} - m| > \varepsilon\right] \le \delta,$$

for $\varepsilon, \delta > 0$.

It is now clear that Morris' algorithm is an unbiased estimator for m.

It uses approximately $O(\log\log m)$ bits of memory.

However, for a practical randomized algorithm, we further require its output to concentrate on the expectation.

That is, we want to establish concentration inequality of the form

$$\mathbf{Pr}\left[|\widehat{m} - m| > \varepsilon\right] \leq \delta,$$

for $\varepsilon, \delta > 0$.

For fixed ε, the smaller δ is, the better the algorithm will be.

Concentration

Concentration

We need some probabilistic tools to establish the concentration inequality.

Concentration

We need some probabilistic tools to establish the concentration inequality.

Markov's inequality

For every nonnegative random variable X and every $a \geq 0$, it holds that

$$\mathbf{Pr}\left[X \geq a\right] \leq \frac{\mathbf{E}\left[X\right]}{a}.$$

Concentration

We need some probabilistic tools to establish the concentration inequality.

Markov's inequality

For every nonnegative random variable X and every $a \geq 0$, it holds that

$$\mathbf{Pr}\left[X \geq a\right] \leq \frac{\mathbf{E}\left[X\right]}{a}.$$

Chebyshev's inequality

For every random variable X and every $a \geq 0$, it holds that

$$\mathbf{Pr}\left[\left|X - \mathbf{E}\left[X\right]\right| \geq a\right] \leq \frac{\mathbf{Var}\left[X\right]}{a^2}.$$

Concentration (cont'd)

Concentration (cont'd)

In order to apply Chebyshev's inequality, we have to compute the variance of $\widehat{m}$.

Lemma

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

Concentration (cont'd)

In order to apply Chebyshev's inequality, we have to compute the variance of $\widehat{m}$.

Lemma

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

We can prove the claim using an induction argument similar to our proof for the expectation.

Concentration (cont'd)

In order to apply Chebyshev's inequality, we have to compute the variance of $\widehat{m}$.

Lemma

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

We can prove the claim using an induction argument similar to our proof for the expectation.

Therefore,

$$\mathbf{Var}\left[\widehat{m}\right] = \mathbf{E}\left[\widehat{m}^2\right] - \mathbf{E}\left[\widehat{m}\right]^2 = \mathbf{E}\left[\left(2^{X_m} - 1\right)^2\right] - m^2 \le \frac{m^2}{2}$$

Applying Chebyshev's inequality, we obtain for every $\varepsilon > 0$,

$$\mathbf{Pr}\left[\left|\widehat{m} - m\right| \geq \varepsilon m\right] \leq \frac{1}{2\varepsilon^2}.$$

Applying Chebyshev's inequality, we obtain for every $\varepsilon > 0$,

$$\mathbf{Pr}\left[|\widehat{m} - m| \geq \varepsilon m\right] \leq \frac{1}{2\varepsilon^2}.$$

Can we improve the concentration?

Applying Chebyshev's inequality, we obtain for every $\varepsilon > 0$,

$$\mathbf{Pr}\left[|\widehat{m} - m| \geq \varepsilon m\right] \leq \frac{1}{2\varepsilon^2}.$$

Can we improve the concentration?

Two common tricks work here.

Averaging trick

Averaging trick

The Chebyshev's inequality tells us that we can improve the concentration by reducing the variance.

Averaging trick

The Chebyshev's inequality tells us that we can improve the concentration by reducing the variance.

Note that variance satisfies

- $\mathbf{Var}\left[a \cdot X\right] = a^2 \cdot \mathbf{Var}\left[X\right]$;
- $\mathbf{Var}\left[X + Y\right] = \mathbf{Var}\left[X\right] + \mathbf{Var}\left[Y\right]$ for independent X and Y.

Averaging trick

The Chebyshev's inequality tells us that we can improve the concentration by reducing the variance.

Note that variance satisfies

- $\mathbf{Var}\left[a \cdot X\right] = a^2 \cdot \mathbf{Var}\left[X\right]$;
- $\mathbf{Var}\left[X + Y\right] = \mathbf{Var}\left[X\right] + \mathbf{Var}\left[Y\right]$ for independent X and Y.

We can independently run Morris algorithm t time in parallel, and let the outputs be $\widehat{m}_1, \dots, \widehat{m}_t$.

Averaging trick

The Chebyshev's inequality tells us that we can improve the concentration by reducing the variance.

Note that variance satisfies

- $\mathbf{Var}[a \cdot X] = a^2 \cdot \mathbf{Var}[X]$;
- $\mathbf{Var}[X + Y] = \mathbf{Var}[X] + \mathbf{Var}[Y]$ for independent X and Y.

We can independently run Morris algorithm t time in parallel, and let the outputs be $\widehat{m}_1, \dots, \widehat{m}_t$.

The final output is $\widehat{m}^* := \frac{\sum_{i=1}^{t} \widehat{m}_i}{t}$.

Averaging trick

The Chebyshev's inequality tells us that we can improve the concentration by reducing the variance.

Note that variance satisfies

- $\mathbf{Var}\left[a \cdot X\right] = a^2 \cdot \mathbf{Var}\left[X\right]$;
- $\mathbf{Var}\left[X + Y\right] = \mathbf{Var}\left[X\right] + \mathbf{Var}\left[Y\right]$ for independent X and Y.

We can independently run Morris algorithm t time in parallel, and let the outputs be $\widehat{m}_1, \dots, \widehat{m}_t$.

The final output is $\widehat{m}^* := \frac{\sum_{i=1}^{t} \widehat{m}_i}{t}$.

Apply Chebyshev's inequality to $\widehat{m}^*$:

$$\mathbf{Pr}\left[\left|\widehat{m}^* - m\right| \geq \varepsilon m\right] \leq \frac{1}{t \cdot 2\varepsilon^2}.$$

For $t \geq \frac{1}{2\varepsilon^2\delta}$, we have

$$\mathbf{Pr}\left[\left|\widehat{m}^* - m\right| \geq \varepsilon m\right] \leq \delta.$$

For $t \geq \frac{1}{2\varepsilon^2\delta}$, we have

$$\mathbf{Pr}\left[\left|\widehat{m}^* - m\right| \geq \varepsilon m\right] \leq \delta.$$

Our algorithm uses $O\left(\frac{\log\log n}{\varepsilon^2\delta}\right)$ bits of memory.

For $t \geq \frac{1}{2\varepsilon^2\delta}$, we have

$$\mathbf{Pr}\left[\left|\widehat{m}^* - m\right| \geq \varepsilon m\right] \leq \delta.$$

Our algorithm uses $O\left(\frac{\log\log n}{\varepsilon^2\delta}\right)$ bits of memory.

A trade-off between the quality of the randomized algorithm and the consumption of memory space.

The Median trick

The Median trick

We choose $t = \frac{3}{2\varepsilon^2}$ in the previous algorithm.

The Median trick

We choose $t = \frac{3}{2\varepsilon^2}$ in the previous algorithm.

Independently run the algorithm s times in parallel, and let the outputs be $\widehat{m}_1^*, \widehat{m}_2^*, \ldots, \widehat{m}_s^*$.

The Median trick

We choose $t = \frac{3}{2\varepsilon^2}$ in the previous algorithm.

Independently run the algorithm s times in parallel, and let the outputs be $\widehat{m}_1^*, \widehat{m}_2^*, \ldots, \widehat{m}_s^*$.

It holds that for every $i = 1, \ldots, s$,

$$\mathbf{Pr}\left[\left|\widehat{m}_i^* - m\right| \geq \varepsilon m\right] \leq \frac{1}{3}.$$

Output the median of $\widehat{m}_1^*, \ldots, \widehat{m}_s^* \quad (=: \widehat{m}^{**})$.

Chernoff bound

Chernoff bound

Chernoff bound

Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in [0, 1]$ for every $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{n} X_i$. Then for every $0 < \varepsilon < 1$, it holds that

$$\mathbf{Pr}\left[|X - \mathbf{E}\left[X\right]| > \varepsilon \cdot \mathbf{E}\left[X\right]\right] \leq 2\exp\left(-\frac{\varepsilon^2 \mathbf{E}\left[X\right]}{3}\right).$$

Analysis of the median trick

Analysis of the median trick

For every $i = 1, \ldots, s$, we let Y_i be the indicator of the (good) event

$$\left|\widehat{m}_i^* - m\right| < \varepsilon \cdot m.$$

Analysis of the median trick

For every $i = 1, \ldots, s$, we let Y_i be the indicator of the (good) event

$$\left|\widehat{m}_i^* - m\right| < \varepsilon \cdot m.$$

Then $Y := \sum_{i=1}^{s} Y_i$ satisfies $\mathbf{E}\left[Y\right] \geq \frac{2}{3}s$.

Analysis of the median trick

For every $i = 1, \ldots, s$, we let Y_i be the indicator of the (good) event

$$\left|\widehat{m}_i^* - m\right| < \varepsilon \cdot m.$$

Then $Y := \sum_{i=1}^{s} Y_i$ satisfies $\mathbf{E}\left[Y\right] \geq \frac{2}{3}s$.

If the median $\widehat{m}^{**}$ is bad (namely $|\widehat{m}^{**} - m| \geq \varepsilon \cdot m$), then at least half of $\widehat{m}_i^*$'s are bad.

Analysis of the median trick

For every $i = 1, \ldots, s$, we let Y_i be the indicator of the (good) event

$$\left|\widehat{m}_i^* - m\right| < \varepsilon \cdot m.$$

Then $Y := \sum_{i=1}^{s} Y_i$ satisfies $\mathbf{E}\left[Y\right] \geq \frac{2}{3}s$.

If the median $\widehat{m}^{**}$ is bad (namely $|\widehat{m}^{**} - m| \geq \varepsilon \cdot m$), then at least half of $\widehat{m}_i^*$'s are bad.

Equivalently, $Y \leq \frac{1}{2}s$.

Analysis of the median trick

For every $i = 1, \ldots, s$, we let Y_i be the indicator of the (good) event

$$\left|\widehat{m}_i^* - m\right| < \varepsilon \cdot m.$$

Then $Y := \sum_{i=1}^{s} Y_i$ satisfies $\mathbf{E}\left[Y\right] \geq \frac{2}{3}s$.

If the median $\widehat{m}^{**}$ is bad (namely $|\widehat{m}^{**} - m| \geq \varepsilon \cdot m$), then at least half of $\widehat{m}_i^*$'s are bad.

Equivalently, $Y \leq \frac{1}{2}s$.

By Chernoff bound,

$$\mathbf{Pr}\left[\left|Y - \mathbf{E}\left[Y\right]\right| \geq \frac{1}{6}s\right] \leq 2\exp\left(-\frac{s}{108}\right).$$

Therefore, for $t = O\left(\frac{1}{\varepsilon^2}\right)$ and $s = O\left(\log \frac{1}{\delta}\right)$, we have

$$\mathbf{Pr}\left[|\widehat{m}^{**} - m| \geq \varepsilon m\right] < \delta.$$

Therefore, for $t = O\left(\frac{1}{\varepsilon^2}\right)$ and $s = O\left(\log\frac{1}{\delta}\right)$, we have

$$\mathbf{Pr}\left[\left|\widehat{m}^{**} - m\right| \geq \varepsilon m\right] < \delta.$$

We use $O\left(\frac{1}{\varepsilon^2} \cdot \log\frac{1}{\delta} \cdot \log\log n\right)$ bits of memory.