Lecture 1

TTIC 41000: Algorithms for Massive Data

Toyota Technological Institute at Chicago

Spring 2021

Instructor: Sepideh Mahabadi

Welcome to Algorithms for Massive Data!

- Lectures: Monday and Wednesday - 4:10-5:30pm
- Office Hours: Monday 6-7pm
- Zoom link (for both): Here

Make sure to check

- Course page
- Canvas page

Grading

- Two PSets (Each 25%)
- Final Project (40%)
- Class Participation (10%)

Problem Sets

- Free to discuss the problems
- Each person should completely understand and write their own solutions
- Write the name of the collaborators

Projects

- Can be done individually or in groups of size two
- Goal: Motivate you to write a paper in this area
- Each project includes a 10-20 minutes presentation in class, and a 5-10 pages report
- Two types of projects:
  - Summary paper
  - Research Project
- Important dates
  - April 30th: proposal
  - May 30th: first draft
  - June 10th: final draft

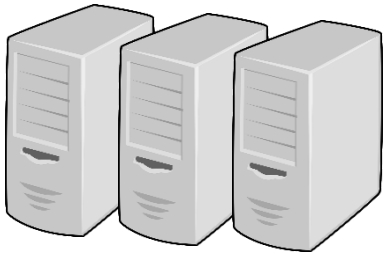
Introduction

Large Data Restrictions and Models

- ❑ Cannot even read the whole data
  - ➢ Sub-linear Time Algorithms

- ❑ Cannot afford storing all the data
  - ➢ Streaming Algorithms

- ❑ Single Processor does not suffice
  - ➢ E.g. Map-Reduce algorithms



- ❑ …

Large Data Tasks

- Searching through the data
  - (Approximate Nearest Neighbor)

- Extracting the most relevant data (efficient summary)
  - (Sketching, Composable Core-sets)

- …

Large Data Techniques

- ❑ Sampling, Importance Sampling
- ❑ Dimensionality Reduction
- ❑ …

This Course

- ❑ Models
- ❑ Tasks
- ❑ Techniques
- ❑ ...

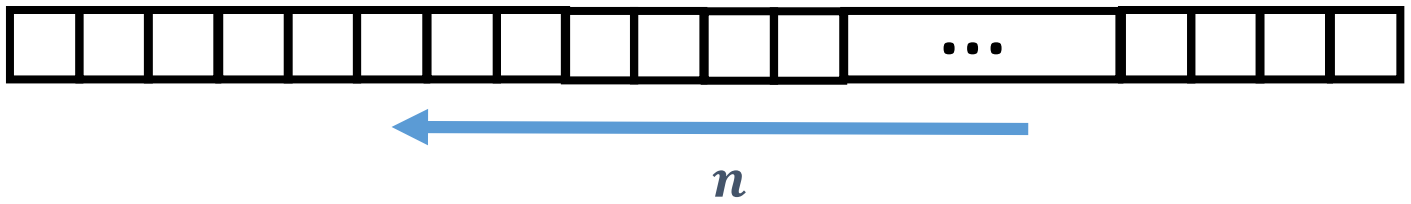
Streaming Model of Computation

Problems:

- Distinct Elements
- Morris Counter

Streaming Model

- Huge data set (does not fit into the main memory)
- Only sequential access to the data
  - One pass
  - Few passes (the data is stored somewhere else)
- Use little memory
  - Sublinear in input parameters
  - Sublinear in the input size
- Solve the problem (approximately)

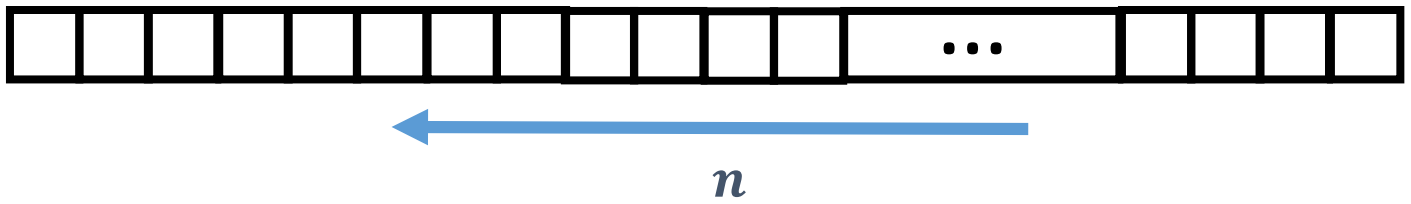


Streaming Model

- Huge data set (does not fit into the main memory)
- Only sequential access to the data
  - One pass
  - Few passes (the data is stored somewhere else)
- Use little memory
  - Sublinear in input parameters
  - Sublinear in the input size
- Solve the problem (approximately)

Parameters of Interest:

1. Memory usage
2. Number of passes
3. Approximation Factor
4. (Sometimes) query/update time



Restrictions

- ❑ Approximation
- ❑ Randomization
  - Analyze expectation
  - Concentration: But also this happens with a good probability

Prob[being approximately correct] is high

Homework: take a look at the inequalities.

- Markov
- Chebychev
- Chernoff/Hoeffding

Probability Recap

Markov's Inequality

- Argue about concentration using the expected value
- Applies to positive random variables

- If X is a non-negative random variable, then

$$\forall \lambda > 0, \qquad \Pr(X > \lambda \cdot \mathbb{E}[X]) \leq \frac{1}{\lambda}$$

The probability of exceeding expectation by more than a factor of λ is at most $\frac{1}{\lambda}$

Chebyshev's Inequality

- Argue about concentration using the variance
- Applies to r.v. with finite expected value and non-zero variance

$$\forall \lambda > 0, \qquad \Pr(|X - \mathbb{E}[X]| > \lambda \cdot \sigma) \leq \frac{1}{\lambda^2}$$

The probability of deviating from expectation by more than $\lambda \cdot \sigma$ *is at most* $\frac{1}{\lambda^2}$

Or,

$$\forall \lambda > 0, \qquad \Pr(|X - \mathbb{E}[X]| > \lambda) \leq \frac{\sigma^2}{\lambda^2}$$

Chebyshev's Inequality

- Argue about concentration using the variance
- Applies to r.v. with finite expected value and non-zero variance

$$\forall \lambda > 0, \qquad \Pr(|X - \mathbb{E}[X]| > \lambda \cdot \sigma) \leq \frac{1}{\lambda^2}$$

The probability of deviating from expectation by more than $\lambda \cdot \sigma$ *is at most* $\frac{1}{\lambda^2}$

Or,

$$\forall \lambda > 0, \qquad \Pr(|X - \mathbb{E}[X]| > \lambda) \leq \frac{\mathbb{E}[(X - \mathbb{E}[X])^2]}{\lambda^2}$$

More generally,

$$\forall p \geq 1, \forall \lambda > 0, \qquad \Pr(|X - \mathbb{E}[X]| > \lambda) \leq \frac{\mathbb{E}[(X - \mathbb{E}[X])^p]}{\lambda^p}$$

Chernoff Bound

- Suppose $X_1, \cdots, X_n$ are independent r.v. with $X_i \in [0,1]$.
  Let $X = \sum_i X_i$ and $\mu = \mathbb{E}[X]$. Then

$$\forall \lambda > 0, \quad \Pr(X > (1+\lambda)\mu\ ) < e^{-\frac{\lambda^2 \mu}{3}} \qquad \textit{(upper tail)}$$

and,

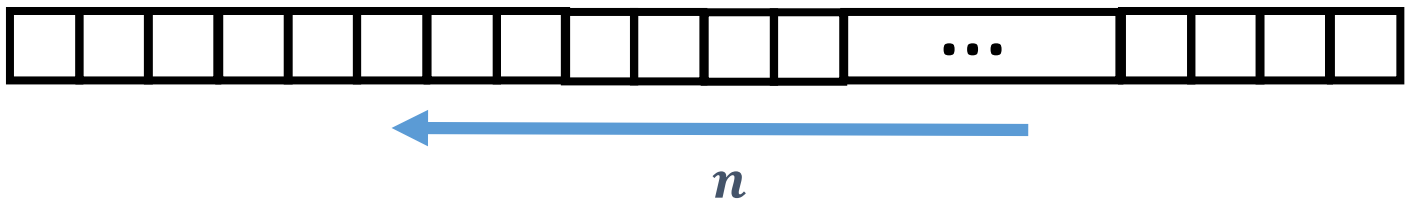
$$\forall \lambda > 0, \quad \Pr(X < (1-\lambda)\mu\ ) < e^{-\frac{\lambda^2 \mu}{2}} \qquad \textit{(lower tail)}$$

- *Strong guarantee but requires the independence of r.v.*

Distinct Elements

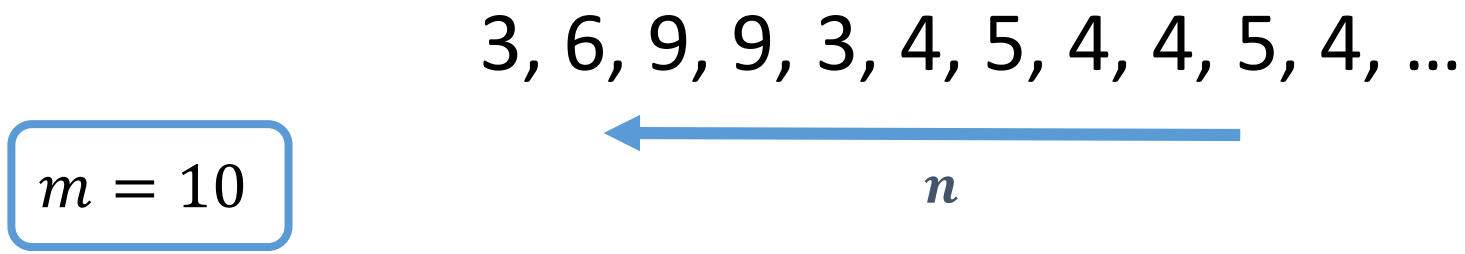
Distinct Element Problem

- Input: a stream of n numbers from $[m]$
- Goal: estimate the number of distinct elements in the stream



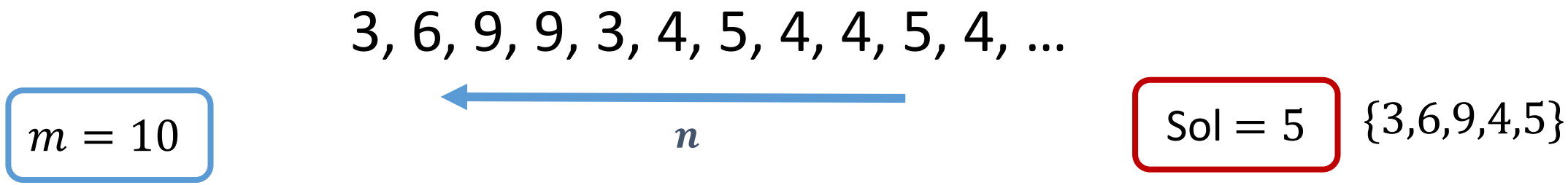
Distinct Element Problem

- Input: a stream of n numbers from $[m]$
- Goal: estimate the number of distinct elements in the stream



Distinct Element Problem

- Input: a stream of n numbers from $[m]$
- Goal: estimate the number of distinct elements in the stream



Distinct Element Problem

- Input: a stream of n numbers from $[m]$
- Goal: estimate the number of distinct elements in the stream

- Trivial solution : $\min(m, n\lceil \log m \rceil)$
  - Keep a counter for each of the m elements, space: m bits.
  - Keep everyone in the stream, space: $n\lceil \log m \rceil$

- We want to do much better

3, 6, 9, 9, 3, 4, 5, 4, 4, 5, 4, …

n

Distinct Element Problem

- Input: a stream of n numbers from $[m]$
- Goal: estimate the number of distinct elements in the stream

- Approximate (up to a factor of $1 + \epsilon$)
- Randomized (with probability $1 - \delta$)

3, 6, 9, 9, 3, 4, 5, 4, 4, 5, 4, …

n

Vector Notation

- Receive updates to coordinates of a vector $x \in \mathbb{R}^m$
  - Coordinate i is denoted as x_i
  - Upon receiving i, we increment x_i by one
- Estimate the number of non-zero coordinates.
- L_0-norm estimation
- L_p-norm estimation: Next Lecture!

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{matrix}$$
$$x = [0, 0, 2, 4, 2, 1, 0, 0, 2, 0]$$

$m = 10$

3, 6, 9, 9, 3, 4, 5, 4, 4, 5, 4, …

n

Distinct Element Problem

- Relax the conditions: allow approximation and randomization:
- Find an estimate $\widetilde{\mathrm{DE}}$ of #distinct elements DE, such that

$$\Pr\left[\left|\widetilde{\mathrm{DE}} - \mathrm{DE}\right| > \epsilon \mathrm{DE}\right] < \delta$$

- Simpler version: Decision problem where given D, identify
  - **YES case:** $\mathrm{DE} \geq D(1+\epsilon)$
  - **NO case:** $\mathrm{DE} < D(1-\epsilon)$
- Solve threshold variant, and run for $D = \{1, 1+\epsilon, \dots (1+\epsilon)^i, \dots, n\}$

➢ Total space will be multiplied by $\log_{1+\epsilon} n \approx \frac{\log n}{\epsilon}$

➢ Probability of failure multiplied by the same factor

Distinct Element Problem

- Relax the conditions: allow approximation and randomization:
- Find an estimate $\widetilde{\mathrm{DE}}$ of #distinct elements DE, such that

$$\Pr\left[\left|\widetilde{\mathrm{DE}} - \mathrm{DE}\right| > \epsilon \mathrm{DE}\right] < \delta$$

- Simpler version: Decision problem where given D, identify
  - **YES case:** $\mathrm{DE} \geq D(1+\epsilon)$
  - **NO case:** $\mathrm{DE} < D(1-\epsilon)$
- Solve threshold variant, and run for $D = \{1, 1+\epsilon, \dots (1+\epsilon)^{\iota}, \dots, n\}$

➢ Total space will be multiplied by $\log_{1+\epsilon} n \approx \frac{\log n}{\epsilon}$

➢ Probability of failure multiplied by the same factor

Basic Algorithm

YES case: $\mathrm{DE} \geq D(1+\epsilon)$

NO case: $\mathrm{DE} < D(1-\epsilon)$

- Sample each of m coordinate w.p. $\frac{1}{D}$

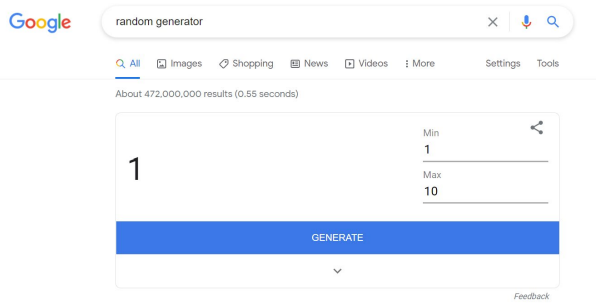
Basic Algorithm

YES case: $\mathrm{DE} \geq D(1+\epsilon)$
NO case: $\mathrm{DE} < D(1-\epsilon)$

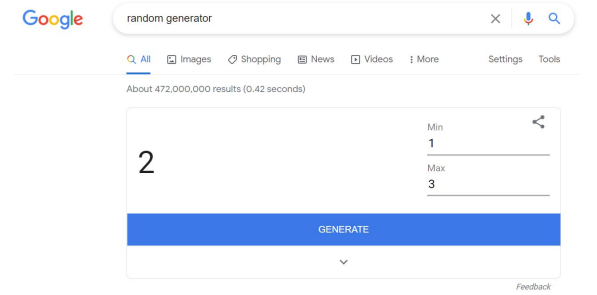
- Sample each of m coordinate w.p. $\frac{1}{D}$
- If all sampled coordinates are 0, return NO
- Otherwise, return YES

Example.

- $D = 10$
  - $S = \{8, 10\}$
  - **zero**



- $D = 3$
  - $S = \{1, 7, 9, 10\}$
  - **non-zero ($x_9 = 2$)**



3, 6, 9, 9, 3, 4, 5, 4, 4, 5, 4, …

n

Basic Algorithm

YES case: $\mathrm{DE} \geq D(1+\epsilon)$
NO case: $\mathrm{DE} < D(1-\epsilon)$

- Sample each of m coordinate w.p. $\frac{1}{D}$
- If all sampled coordinates are 0, return NO
- Otherwise, return YES

S: set of sampled coordinates
Whether $\sum_{i \in S} x_i = 0$?

Basic Algorithm

YES case: $\mathrm{DE} \geq D(1+\epsilon)$
NO case: $\mathrm{DE} < D(1-\epsilon)$

- Sample each of m coordinate w.p. $\frac{1}{D}$
- If all sampled coordinates are 0, return NO
- Otherwise, return YES

- Pr[NO] = $\left(1-\frac{1}{D}\right)^{DE} \approx e^{-\frac{DE}{D}}$
- For small enough ϵ
  - If $\mathrm{DE} > (1+\epsilon)D$, then $e^{-(1+\epsilon)} < \frac{1}{\mathrm{e}} - \frac{\epsilon}{3}$
  - If $\mathrm{DE} < (1-\epsilon)D$, then $e^{-(1-\epsilon)} > \frac{1}{\mathrm{e}} + \frac{\epsilon}{3}$

S: set of sampled coordinates
Whether $\sum_{i \in S} x_i = 0$?

Basic Algorithm

YES case: $\mathrm{DE} \geq D(1+\epsilon)$
NO case: $\mathrm{DE} < D(1-\epsilon)$

- Sample each of m coordinate w.p. $\frac{1}{D}$
- If all sampled coordinates are 0, return NO
- Otherwise, return YES

- Pr[NO] = $\left(1-\frac{1}{D}\right)^{DE} \approx e^{-\frac{DE}{D}}$
- For small enough ϵ
  - If $\mathrm{DE} > (1+\epsilon)D$, then $e^{-(1+\epsilon)} < \frac{1}{\mathrm{e}} - \frac{\epsilon}{3}$
  - If $\mathrm{DE} < (1-\epsilon)D$, then $e^{-(1-\epsilon)} > \frac{1}{\mathrm{e}} + \frac{\epsilon}{3}$

S: set of sampled coordinates
Whether $\sum_{i \in S} x_i = 0$?

Space usage: a single number in [n]

Decision Variant

- Sample each of m coordinate w.p. $\frac{1}{D}$
- If all sampled coordinates are 0, return NO
- Otherwise, return YES

- Pr[NO] = $\left(1-\frac{1}{D}\right)^{DE} \approx e^{-\frac{DE}{D}}$
- For small enough ϵ
  - If $\mathrm{DE} > (1+\epsilon)D$, then $e^{-(1+\epsilon)} < \frac{1}{\mathrm{e}} - \frac{\epsilon}{3}$
  - If $\mathrm{DE} < (1-\epsilon)D$, then $e^{-(1-\epsilon)} > \frac{1}{\mathrm{e}} + \frac{\epsilon}{3}$

- Repeat $k = O\left(\frac{\log\frac{1}{\delta}}{\epsilon^2}\right)$ times
  - $S_1, \cdots, S_k$
- Define $C \triangleq \left|\left\{S_j \middle| \sum_{i\in S_j} x_i = 0\right\}\right|$
- Compare C with k/e

Decision Variant

Boost the success probability using repetition and Chernoff

- Repeat $k = O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ times and let C be #times we get 0
- Using Chernoff bound, with probability at least $(1-\delta)$
  1. If $\mathrm{DE} > (1+\epsilon)D$ then $C < \frac{k}{e}$
  2. If $\mathrm{DE} < (1-\epsilon)D$ then $C > \frac{k}{e}$

Case 1.

$$\forall \lambda > 0, \Pr(X > (1+\lambda)\mu) < e^{-\frac{\lambda^2 \mu}{3}}$$

- $\mathbb{E}[C] < \frac{k}{e}(1 - \frac{\epsilon e}{3})$ and set $\lambda = \frac{\epsilon e}{3}$
- $\Pr\left(C > \frac{k}{e}\right) < \Pr\left(C > \mathbb{E}[C](1+\frac{\epsilon e}{3})\right) < e^{-\frac{(\epsilon e/3)^2 \cdot O((\log\frac{1}{\delta})/\epsilon^2)}{3}} = \delta$

Decision Variant

Boost the success probability using repetition and Chernoff

- Repeat $k = O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ times and let C be #times we get 0
- Using Chernoff bound, with probability at least $(1-\delta)$
  1. If $\mathrm{DE} > (1+\epsilon)D$ then $C < \frac{k}{e}$
  2. If $\mathrm{DE} < (1-\epsilon)D$ then $C > \frac{k}{e}$
- Space usage of the algorithm: $O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ numbers from $[n]$
- Probability of error δ

Distinct Elements

- Space usage of the algorithm: $O\left(\frac{\log n}{\epsilon} \cdot \frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ numbers from $[n]$ (Can handle deletions too)

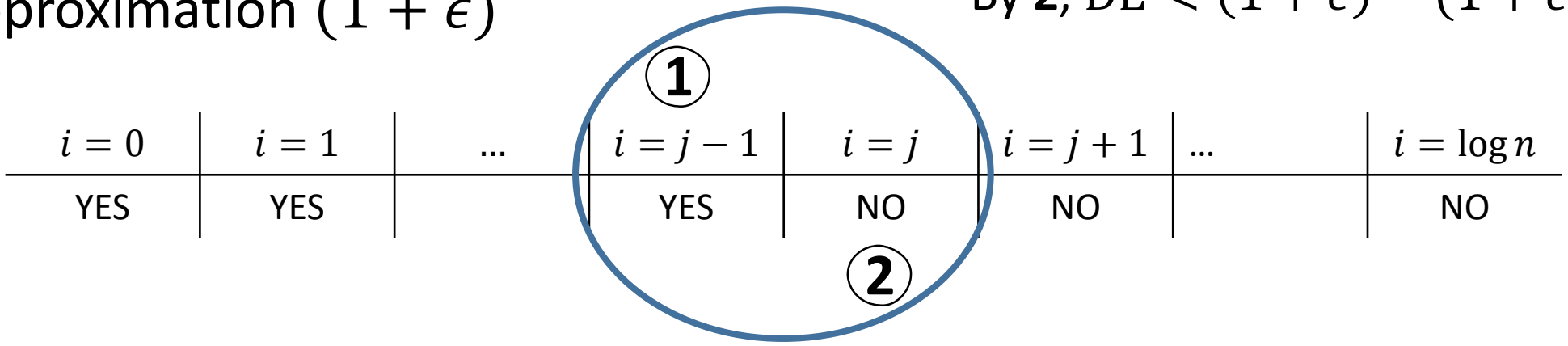
thresholds

sample set (per threshold)

Distinct Elements

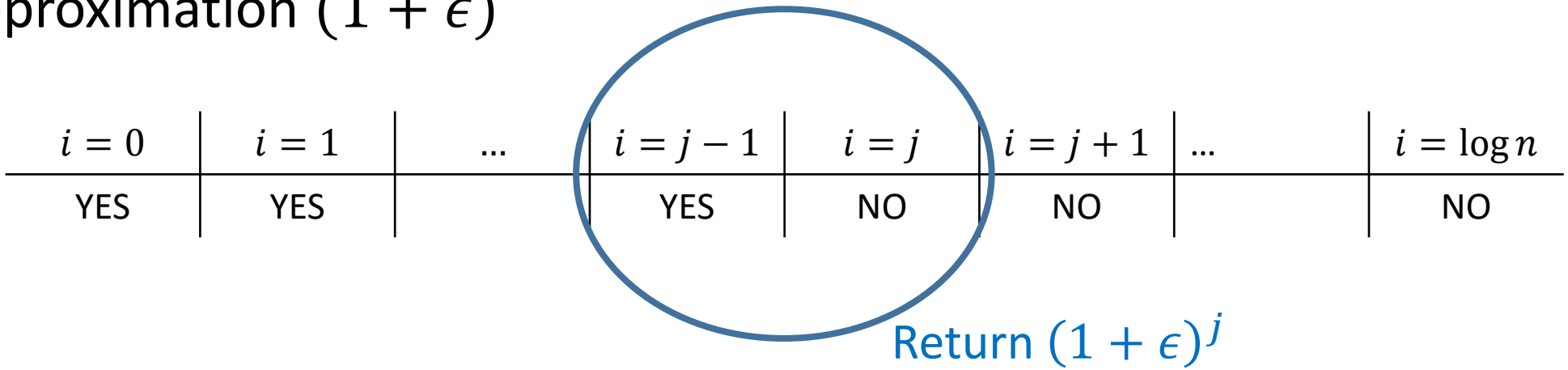
- Space usage of the algorithm: $O\left(\frac{\log n}{\epsilon} \cdot \frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ numbers from $[n]$ (Can handle deletions too)
- Probability of error $\delta \cdot \frac{\log n}{\epsilon}$
- Number of passes: 1
- Approximation $(1+\epsilon)$

By **1**, $\mathrm{DE} > (1+\epsilon)^{j-1} \cdot (1-\epsilon)$
By **2**, $\mathrm{DE} < (1+\epsilon)^{j} \cdot (1+\epsilon)$



Distinct Elements

- Space usage of the algorithm: $O\left(\frac{\log n}{\epsilon} \cdot \frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ numbers from $[n]$ (Can handle deletions too)
- Probability of error $\delta \cdot \frac{\log n}{\epsilon}$
- Number of passes: 1
- Approximation $(1+\epsilon)$



Some Comments

1. Deletions
   - The algorithm can handle deletion as long as values of coordinates remain non-negative
   - Dynamic or turnstile streaming

2. Implementing the sample set S
   - Pick a hash function h: $[m] \to [D]$ and $S = \{i | h(i) = 1\}$
   - h can be implemented using pseudo-random generators

State of the art

Uses space: $\Theta(\epsilon^{-2}\log\left(\frac{1}{\delta}\right)+\log n)$

- Upper bound:
  - [Jaroslaw Blasiok'20]. Optimal streaming and tracking distinct elements with high probability
- Lower bound:
  - [Noga Alon, Yossi Matias, and Mario Szegedy'99]. The space complexity of approximating the frequency moments
  - [T. S. Jayram and David P. Woodruff'13]. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error

Morris Counter

Counting Problem

- Space efficient algorithm that monitors a sequence of events, then at any *given query time* t, returns an *estimate of the number of events* seen so far (i.e. up to time t)

  update(): *increment the number of events by 1*

  query(): *output (an estimate of) the number of events*

- *Trivial Bound?*
  - If #events is n, $\log n$ *bits* suffice to maintain #events exactly
  - Impossible to solve this problem exactly using $o(\log n)$

Approximately Counting

- In most practical cases, it suffices to provide an estimate of #events

- **Goal:** Given error parameters (ϵ, δ), return an estimate $\tilde{n}$ of #events n, s.t.

$$\Pr[|\tilde{n} - n| > \epsilon n] < \delta$$

Morris Algorithm

1. Initialize $X \leftarrow 0$
2. Per each ***update()***, increment X with probability $\frac{1}{2^X}$

❑ For ***query()***, output $\tilde{n} = 2^X - 1$

Intuition: X is attempting to store a value that is roughly $\approx \log n$

The amount of space required to store X is $O(\log \log n)$

Analysis of Morris's Algorithm

Claim 1. Let X_n denote X after n updates. Then, $\mathbb{E}[2^{X_n}] = n + 1$.

Proof by induction on n.

Base case $n = 0, X = 0$ trivially holds. Consider the inductive step.

$$\mathbb{E}[2^{X_{n+1}}] = \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot \mathbb{E}[(2^{X_{n+1}} | X_n = j)]$$

Analysis of Morris's Algorithm

Claim 1. Let X_n denote X after n updates. Then, $\mathbb{E}[2^{X_n}] = n + 1$.

Proof by induction on n.

Base case $n = 0, X = 0$ trivially holds. Consider the inductive step.

$$\mathbb{E}[2^{X_{n+1}}] = \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot \mathbb{E}[(2^{X_{n+1}} | X_n = j)]$$

$$= \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot (2^j \cdot \left(1 - \frac{1}{2^j}\right) + 2^{j+1} \cdot \frac{1}{2^j})$$

Probability of incrementing X

Analysis of Morris's Algorithm

Claim 1. Let X_n denote X after n updates. Then, $\mathbb{E}[2^{X_n}] = n + 1$.

Proof by induction on n.

Base case $n = 0, X = 0$ trivially holds. Consider the inductive step.

$$\mathbb{E}[2^{X_{n+1}}] = \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot \mathbb{E}[(2^{X_{n+1}} | X_n = j)]$$

$$= \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot (2^j \cdot \left(1 - \frac{1}{2^j}\right) + 2^{j+1} \cdot \frac{1}{2^j})$$

$$= \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot 2^j + \sum_{j=0}^{\infty} \Pr(X_n = j)$$

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

$$= n + 1$$

Analysis of Morris's Algorithm (ctd.)

- We just showed that $\tilde{n}$ is an unbiased estimator of n
  - i.e., $\tilde{n} \stackrel{\text{def}}{=} \mathbb{E}[2^X] = n + 1$

Analysis of Morris's Algorithm (ctd.)

- We just showed that $\tilde{n}$ is an unbiased estimator of n
  - i.e., $\tilde{n} \stackrel{\text{def}}{=} \mathbb{E}[2^X] = n + 1$
- By bounding the variance of $\tilde{n}$ and an application of Chebyshev's, we can achieve the desired accuracy guarantee.

Analysis of Morris's Algorithm (ctd.)

- We just showed that $\tilde{n}$ is an unbiased estimator of n
  - i.e., $\tilde{n} \stackrel{\text{def}}{=} \mathbb{E}[2^X] = n + 1$
- By bounding the variance of $\tilde{n}$ and an application of Chebyshev's, we can achieve the desired accuracy guarantee.
  - To recall, goal is to show $\Pr[|\tilde{n} - n| > \epsilon n] < \delta$

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{\epsilon^2 n^2} \cdot \mathbb{E}[(\tilde{n} - n)^2] = \frac{1}{\epsilon^2 n^2} \cdot \mathbb{E}[\left(2^X - 1 - n\right)^2]$$

Claim 2. $\mathbb{E}[2^{2X_n}] = \frac{3}{2}n^2 + \frac{3}{2}n + 1.$

Proof. Similar to Claim 1 (exercise)

Analysis of Morris's Algorithm (ctd.)

- We just showed that $\tilde{n}$ is an unbiased estimator of n
  - i.e., $\tilde{n} \stackrel{\text{def}}{=} \mathbb{E}[2^X] = n + 1$
- By bounding the variance of $\tilde{n}$ and an application of Chebyshev's, we can achieve the desired accuracy guarantee.
  - To recall, goal is to show $\Pr[|\tilde{n} - n| > \epsilon n] < \delta$

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{\epsilon^2 n^2} \cdot \mathbb{E}[(\tilde{n} - n)^2] = \frac{1}{\epsilon^2 n^2} \cdot \mathbb{E}[\left(2^X - 1 - n\right)^2]$$

Claim 2. $\mathbb{E}[2^{2X_n}] = \frac{3}{2}n^2 + \frac{3}{2}n + 1.$

Proof. Similar to Claim 1 (exercise)

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{\epsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\epsilon^2}$$

Analysis of Morris's Algorithm (ctd.)

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{\epsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\epsilon^2}$$

- Not very meaningful! RHS is better than ½ only when $\epsilon > 1$ (for which we can instead always return 0 !)

Analysis of Morris's Algorithm (ctd.)

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{\epsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\epsilon^2}$$

- Not very meaningful! RHS is better than ½ only when $\epsilon > 1$ (for which we can instead always return 0 !)

- How to decrease the failure probability?

Analysis of Morris's Algorithm (ctd.)

- **Morris+**

  Average of s **Morris** estimators. Variance is multiplied by $(\frac{1}{s})$.

  Setting $s = \Theta(\frac{1}{\epsilon^2 \delta})$ suffices to get failure probability δ

Analysis of Morris's Algorithm (ctd.)

- **Morris+**

  Average of s **Morris** estimators. Variance is multiplied by $(\frac{1}{s})$.

  Setting $s = \Theta(\frac{1}{\epsilon^2 \delta})$ suffices to get failure probability δ

- **Morris++**

  Median of t **Morris+** estimators.

  Setting $s = \frac{1}{\epsilon^2}$, each **Morris+** estimator succeeds w.p. at least $\frac{2}{3}$.

  By Chernoff and setting $t = \Theta(\log \frac{1}{\delta})$, the failure probability becomes at most δ

Analysis of Morris's Algorithm (ctd.)

- **Morris+**

  Average of s **Morris** estimators. Variance is multiplied by $(\frac{1}{s})$.

  Setting $s = \Theta(\frac{1}{\epsilon^2 \delta})$ suffices to get failure probability δ

- **Morris++**

  Median of t **Morris+** estimators.

  Setting $s = \frac{1}{\epsilon^2}$, each **Morris+** estimator succeeds w.p. at least $\frac{2}{3}$.

  By Chernoff and setting $t = \Theta(\log \frac{1}{\delta})$, the failure probability becomes at most δ

Total Space of Morris++: $\Theta(\frac{1}{\epsilon^2} \cdot \log \frac{1}{\delta} \cdot \log \log n)$ w.p. at least $1 - \delta$