CSE 312

Foundations of Computing II

Lecture 18: Continuity Correction & Distinct Elements

Review CLT

Theorem. (Central Limit Theorem) $X_1, \ldots, X_n$ i.i.d. with mean μ and variance σ^2. Let $Y_n = \frac{X_1 + \cdots + X_n - n\mu}{\sigma\sqrt{n}}$. Then,

$$\lim_{n\to\infty} Y_n \to \mathcal{N}(0,1)$$

One main application:

Use Normal Distribution to Approximate Y_n

No need to understand Y_n !!

Agenda

- Continuity correction
- Application: Counting distinct elements

Example – Y_n is binomial

We understand binomial, so we can see how well approximation works

We flip n independent coins, heads with probability $p = 0.75$.

X = # heads $\quad \mu = \mathbb{E}(X) = 0.75n \quad \sigma^2 = \text{Var}(X) = p(1-p)n = 0.1875n$

$\mathbb{P}(X \leq 0.7n)$

wide range

n	exact	$\mathcal{N}(\mu, \sigma^2)$ approx
10	0.4744072	0.357500327
20	0.38282735	0.302788308
50	0.25191886	0.207108089
100	0.14954105	0.124106539
200	0.06247223	0.051235217
1000	0.00019359	0.000130365

$\mathcal{N}(\mu = 0.75n, \sigma^2 = 0.1875n)$

Example – Naive Approximation

Fair coin flipped (independently) **40** times. Probability of **20** or **21** heads?

Exact. $\mathbb{P}(X \in \{20,21\}) = \left[\binom{40}{20} + \binom{40}{21}\right]\left(\frac{1}{2}\right)^{40} \approx \boxed{0.2448}$

Approx. $X = \#$ heads $\quad \mu = \mathbb{E}(X) = 0.5n = 20 \quad \sigma^2 = \mathrm{Var}(X) = 0.25n = 10$

$$\mathbb{P}(20 \leq X \leq 21) = \Phi\left(\frac{20-20}{\sqrt{10}} \leq \frac{X-20}{\sqrt{10}} \leq \frac{21-20}{\sqrt{10}}\right)$$

$$\approx \Phi\left(0 \leq \frac{X-20}{\sqrt{10}} \leq 0.32\right)$$

$$= \Phi(0.32) - \Phi(0) \approx \boxed{0.1241}$$

😢

Example – Even Worse Approximation

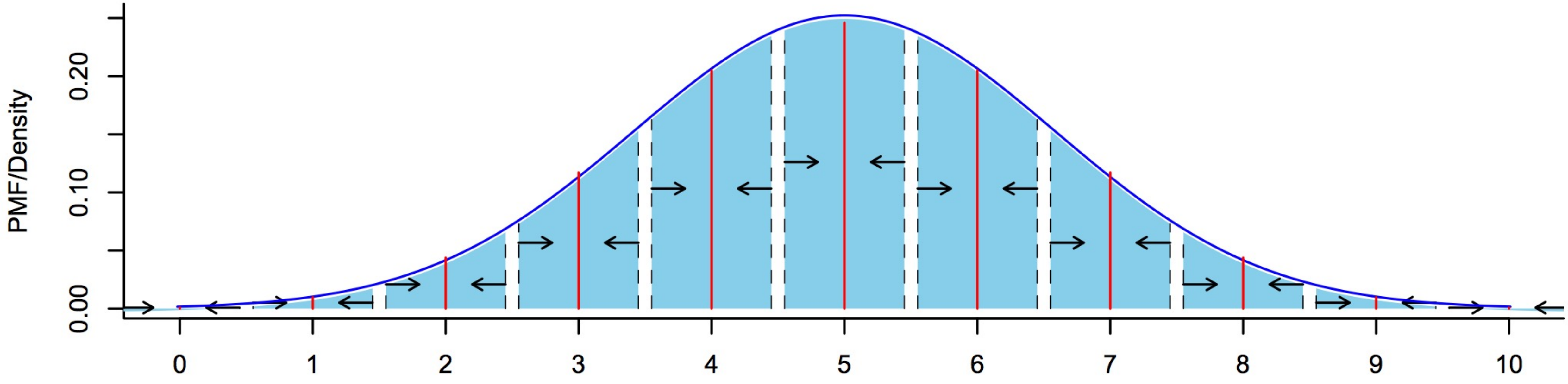
Fair coin flipped (independently) **40** times. Probability of **20** heads?

Exact. $\mathbb{P}(X = 20) = \binom{40}{20}\left(\frac{1}{2}\right)^{40} \approx \boxed{0.1254}$

Approx. $\mathbb{P}(20 \leq X \leq 20) = 0$ 😢

Solution – Continuity Correction

Round to next integer!



To estimate probability that discrete RV lands in (integer) interval $\{a, \ldots, b\}$, compute probability continuous approximation lands in interval $[a - \frac{1}{2}, b + \frac{1}{2}]$

Example – Continuity Correction

Fair coin flipped (independently) **40** times. Probability of **20** or **21** heads?

Exact. $\mathbb{P}(X \in \{20,21\}) = \left[\binom{40}{20} + \binom{40}{21}\right]\left(\frac{1}{2}\right)^{40} \approx \boxed{0.2448}$

Approx. X = # heads $\quad \mu = \mathbb{E}(X) = 0.5n = 20 \quad \sigma^2 = \mathrm{Var}(X) = 0.25n = 10$

$$\mathbb{P}(19.5 \leq X \leq 21.5) = \Phi\left(\frac{19.5 - 20}{\sqrt{10}} \leq \frac{X - 20}{\sqrt{10}} \leq \frac{21.5 - 20}{\sqrt{10}}\right)$$

$$\approx \Phi\left(-0.16 \leq \frac{X - 20}{\sqrt{10}} \leq 0.47\right)$$

$$= \Phi(-0.16) - \Phi(0.47) \approx \boxed{0.2452}$$

Example – Continuity Correction

Fair coin flipped (independently) **40** times. Probability of **20** heads?

Exact. $\mathbb{P}(X = 20) = \binom{40}{20}\left(\frac{1}{2}\right)^{40} \approx \boxed{0.1254}$

Approx. $\mathbb{P}(19.5 \leq X \leq 20.5) = \Phi\left(\frac{19.5 - 20}{\sqrt{10}} \leq \frac{X - 20}{\sqrt{10}} \leq \frac{20.5 - 20}{\sqrt{10}}\right)$

$$\approx \Phi\left(-0.16 \leq \frac{X - 20}{\sqrt{10}} \leq 0.16\right)$$

$$= \Phi(-0.16) - \Phi(0.16) \approx \boxed{0.1272}$$

Agenda

- Continuity correction
- Application: Counting distinct elements

Data mining – Stream Model

- In many data mining situations, data often not known ahead of time.
  - Examples: Google queries, Twitter or Facebook status updates, YouTube video views
- Think of the data as an <u>infinite stream</u>
- Input elements (e.g. Google queries) enter/arrive one at a time.
  - We cannot possibly store the stream.

Question: How do we make critical calculations about the data stream using a limited amount of memory?

Stream Model – Problem Setup

Input: sequence (aka. "stream") of N elements $x_1, x_2, \ldots, x_N$ from a known universe U (e.g., 8-byte integers).

Goal: perform a computation on the input, in a single left to right pass, where:

- Elements processed in real time
- Can't store the full data ⇒ use minimal amount of storage while maintaining working "summary"

What can we compute?

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

val ← ∞ → val = 32 → 12

Some functions are easy:

- Min
- Max
- Sum
- Average

Today: Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

Application

You are the content manager at YouTube, and you are trying to figure out the **distinct** view count for a video. How do we do that?

Note: A person can view their favorite videos several times, but they only count as 1 **distinct** view!

Other applications

- IP packet streams: How many distinct IP addresses or IP flows (source+destination IP, port, protocol)
  - Anomaly detection, traffic monitoring
- Search: How many distinct search queries on Google on a certain topic yesterday
- Web services: how many distinct users (cookies) searched/browsed a certain term/item
  - Advertising, marketing trends, etc.

Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

N = # of IDs in the stream = 11, m = # of distinct IDs in the stream = 5

Want to compute number of **distinct** IDs in the stream.

- *Naïve solution: As the data stream comes in, store all distinct IDs in a hash table.*
- *Space requirement:* $\Omega(m)$

YouTube Scenario: m is huge!

Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

N $=$ # of IDs in the stream = 11, m $=$ # of distinct IDs in the stream = 5

Want to compute number of **distinct** IDs in the stream.

How to do this without storing all the elements?

Detour – I.I.D. Uniforms

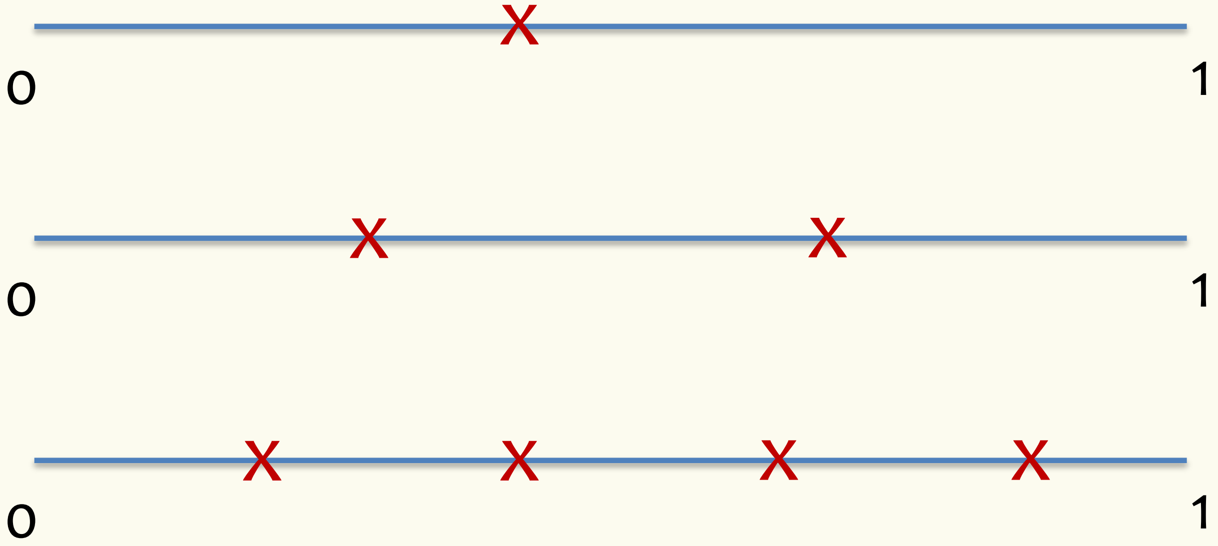
If $Y_1, \cdots, Y_m \sim \mathrm{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

"Evenly spread out"

$m = 1$

$m = 2$

$m = 4$

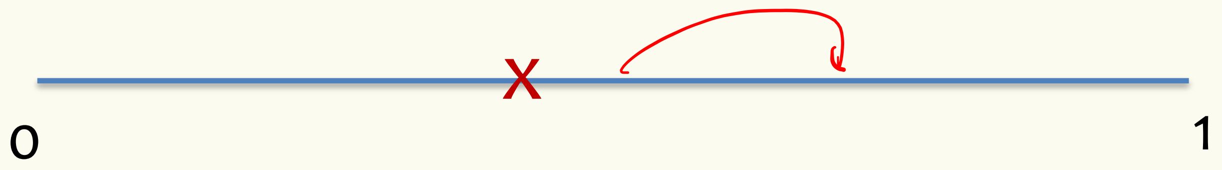


What is some intuition for this?

Detour – I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

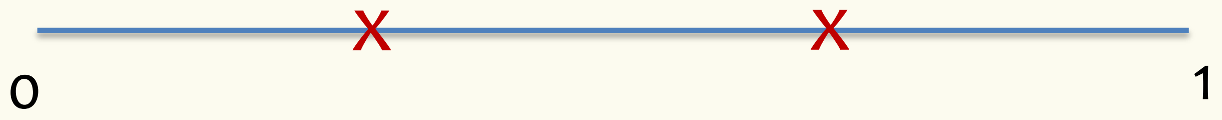
$m = 1$



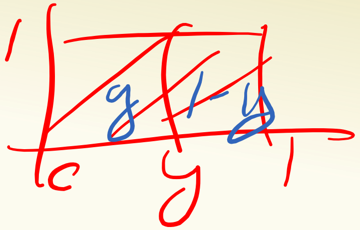
Y_1 has expected value $1/2$
… but probably isn't very close to the middle

… and Y_2 is more likely to be in the bigger gap

$m = 2$



Detour – Min of I.I.D. Uniforms



If $Y_1, \cdots, Y_m \sim \mathrm{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

e.g., what is $\mathbb{E}[\min\{Y_1, \cdots, Y_m\}]$?

CDF: Observe that $\min\{Y_1, \cdots, Y_m\} \geq y$ if and only if $Y_1 \geq y, \ldots, Y_m \geq y$

(Similar to Section 6)

$$P(\min\{Y_1, \cdots, Y_m\} \geq y) = P(Y_1 \geq y, \ldots, Y_m \geq y)$$

$y \in [0,1]$

$$= P(Y_1 \geq y) \cdots P(Y_m \geq y) \qquad \text{(Independence)}$$

$$= (1-y)^m$$

CDF.

$$\Rightarrow P(\min\{Y_1, \cdots, Y_m\} \leq y) = 1 - (1-y)^m$$

Detour – Min of I.I.D. Uniforms

Useful fact. For any random variable Y taking non-negative values

$$\mathbb{E}[Y] = \int_0^{\infty} P(Y \geq y)\,\mathrm{d}y$$

$$\mathbb{E}[Y] = \int_{-\infty}^{\infty} y \cdot f_Y(y)\,dy$$

Proof (Not covered)

$$\mathbb{E}[Y] = \int_0^{\infty} x \cdot f_Y(x)\,\mathrm{d}x = \int_0^{\infty} \left(\int_0^{x} 1\,\mathrm{d}y \right) \cdot f_Y(x)\,\mathrm{d}x = \int_0^{\infty} \int_0^{x} f_Y(x)\,\mathrm{d}y\,\mathrm{d}x$$

$$= \iint_{0 \leq y \leq x \leq \infty} f_Y(x) = \int_0^{\infty} \int_y^{\infty} f_Y(x)\,\mathrm{d}x\,\mathrm{d}y = \int_0^{\infty} P(Y \geq y)\,\mathrm{d}y$$

Detour – Min of I.I.D. Uniforms

$Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.)

$Y = \min\{Y_1, \cdots, Y_m\}$

Useful fact. For any random variable Y taking non-negative values

$$\mathbb{E}[Y] = \int_0^\infty P(Y \geq y)\mathrm{d}y$$

$$\mathbb{E}[Y] = \int_0^\infty P(Y \geq y)\mathrm{d}y = \int_0^1 (1-y)^m \mathrm{d}y$$

$$= -\frac{1}{m+1}(1-y)^{m+1}\Big|_0^1 = 0 - \left(-\frac{1}{m+1}\right) = \frac{1}{m+1}$$

Detour – Min of I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (iid) where do we expect the points to end up?

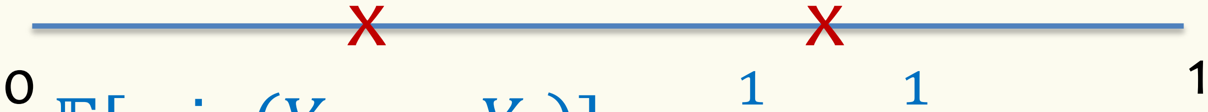
In general, $\mathbb{E}[\min(Y_1, \cdots, Y_m)] = \frac{1}{m+1}$

$$\mathbb{E}[\min(Y_1)] = \frac{1}{1+1} = \frac{1}{2}$$

$m = 1$

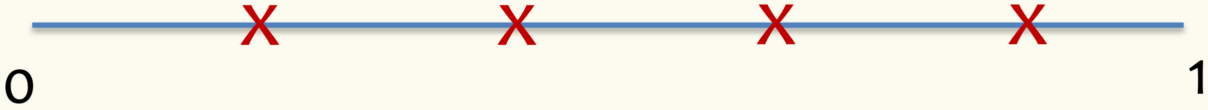
$$\mathbb{E}[\min(Y_1, Y_2)] = \frac{1}{2+1} = \frac{1}{3}$$

$m = 2$



$$\mathbb{E}[\min(Y_1, \cdots, Y_4)] = \frac{1}{4+1} = \frac{1}{5}$$

$m = 4$



Distinct Elements – Hashing into $[0, 1]$

Hash function $h: U \to [0,1]$
Assumption: For all $x \in U$, $h(x) \sim \text{Unif}(0,1)$ and mutually independent

$x_1 = 5$	$x_2 = 2$	$x_3 = 27$	$x_4 = 35$	$x_5 = 4$
$h(5)$	$h(2)$	$h(27)$	$h(35)$	$h(4)$

5 distinct elements

$\rightarrow$ 5 i.i.d. RVs $h(x_1), \ldots, h(x_5) \sim \text{Unif}(0,1)$

$$\rightarrow \mathbb{E}[\min\{h(x_1), \ldots, h(x_5)\}] = \frac{1}{5+1} = \frac{1}{6}$$

Distinct Elements – Hashing into $[0, 1]$

Hash function $h: U \rightarrow [0,1]$
Assumption: For all $x \in U$, $h(x) \sim \text{Unif}(0,1)$ and mutually independent

$x_1 = 5$	$x_2 = 2$	$x_3 = 27$	$x_4 = 5$	$x_5 = 4$
$h(5)$	$h(2)$	$h(27)$	$h(5)$	$h(4)$

4 distinct elements

$\Rightarrow$ 4 i.i.d. RVs $h(x_1), h(x_2), h(x_3), h(x_5) \sim \text{Unif}(0,1)$ and $h(x_1) = h(x_4)$

$\Rightarrow \mathbb{E}[\min\{h(x_1), \dots, h(x_5)\}] = \mathbb{E}[\min\{h(x_1), h(x_2), h(x_3), h(x_5)\}] = \frac{1}{4+1} = \frac{1}{5}$

Distinct Elements – Hashing into $[0, 1]$

Hash function $h: U \to [0,1]$
Assumption: For all $x \in U$, $h(x) \sim \mathrm{Unif}(0,1)$ and mutually independent

$x_1, x_2, \dots, x_N$ contains m distinct elements

$h(x_1), h(x_2), \dots, h(x_N)$ contains m i.i.d. rvs $\sim \mathrm{Unif}(0,1)$

and $N - m$ repeats

$$\mathbb{E}[\min\{h(x_1), \dots, h(x_N)\}] = \frac{1}{m+1} \iff m = \frac{1}{\mathbb{E}[\min\{h(x_1), \dots, h(x_N)\}]} - 1$$

The MinHash Algorithm – Idea

$$m = \frac{1}{\mathbb{E}[\min\{h(x_1), \ldots, h(x_N)\}]} - 1$$

1. Compute $\text{val} = \min\{h(x_1), \ldots, h(x_N)\}$
2. Assume that $\text{val} \approx \mathbb{E}[\min\{h(x_1), \ldots, h(x_N)\}]$
3. Output $\text{round}\left(\frac{1}{\text{val}} - 1\right)$

The MinHash Algorithm – Implementation

Algorithm **`MinHash`**$(x_1, x_2, \dots, x_N)$

$\text{val} \leftarrow \infty$

for $i = 1$ **to** N **do**

  $\text{val} \leftarrow \min\{\text{val}, h(x_i)\}$

return $\text{round}\left(\frac{1}{\text{val}} - 1\right)$

Memory cost = just remember val (with sufficient precision)

MinHash Example

Stream: 13, 25, 19, 25, 19, 19

Hashes: 0.51, 0.26, 0.79, 0.26, 0.79, 0.79

What does MinHash return?

Poll: pollev.com/stefanotessaro617

a. 1
b. 3
c. 5
d. No idea

MinHash Example II

Stream: 11, 34, 89, 11, 89, 23

Hashes: 0.5, 0.21, 0.94, 0.5, 0.94, 0.1

Output is $\frac{1}{0.1} - 1 = 9$

Clearly, not a very good answer!

Not unlikely: $P(h(x) < 0.1) = 0.1$

The MinHash Algorithm – Problem

Algorithm **MinHash**$(x_1, x_2, \dots, x_N)$

$\mathrm{val} \leftarrow \infty$

for $i = 1$ **to** N **do**

$\quad \mathrm{val} \leftarrow \min\{\mathrm{val}, h(x_i)\}$

return $\mathrm{round}\left(\frac{1}{\mathrm{val}} - 1\right)$

But, val is not $\mathbb{E}[\mathrm{val}]$!
How far is val from $\mathbb{E}[\mathrm{val}]$?

$$\mathrm{Var}(\mathrm{val}) \approx \frac{1}{(m+1)^2}$$

$\mathrm{val} = \min\{h(x_1), \dots, h(x_N)\}$ $\quad \mathbb{E}[\mathrm{val}] = \frac{1}{m+1}$

How can we reduce the variance?

Idea: Repetition to reduce variance!

Use k **independent** hash functions $h^1, h^2, \cdots h^k$

Algorithm **`MinHash`**$(x_1, x_2, \ldots, x_N)$

$\mathrm{val}_1, \ldots, \mathrm{val}_\mathrm{k} \leftarrow \infty$

for $i = 1$ **to** N **do**

$\quad \mathrm{val}_1 \leftarrow \min\{\mathrm{val}_1, h^1(x_i)\}, \ldots, \mathrm{val}_\mathrm{k} \leftarrow \min\{\mathrm{val}_k, h^k(x_i)\}$

$\mathrm{val} \leftarrow \frac{1}{k} \sum_{i=1}^{k} \mathrm{val}_\mathrm{i}$

return $\mathrm{round}\left(\frac{1}{\mathrm{val}} - 1\right)$

$$\mathrm{Var}(\mathrm{val}) = \frac{1}{k} \frac{1}{(m+1)^2}$$

MinHash and Estimating # of Distinct Elements in Practice

- MinHash in practice:
  - One also stores the element that has the minimum hash value for each of the k hash functions

    $\frac{|A \cap B|}{|A \cup B|}$

    - Then, just given separate MinHashes for sets A and B, can also estimate
      - what fraction of $A \cup B$ is in $A \cap B$; i.e., how similar A and B are

- Another randomized data structure for distinct elements in practice:
  - HyperLoglog - even more space efficient but doesn't have the set combination properties of MinHash