CSE 312

Foundations of Computing II

Lecture 18: Continuity correction & Distinct elements

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Anna R. Karlin

Slide Credit: Based on Stefano Tessaro's slides for 312 19au
incorporating ideas from Alex Tsun, Rachel Lin, Hunter Schafer & myself ☺

Reminders

- Quiz out tonight at 6pm.
  - Questions about quiz? Send them privately to staff on edstem or hop into OHs
  - We will have someone monitoring edstem closely or else having OHs tonight from 6-10pm, tomorrow from 5-9pm.
  - No concept check today.
- Moving my office hours from tonight to tomorrow at 8-9pm. Same zoom link.
- Pset 6 out Wednesday as usual.
- No section this week. (Veteran's Day)

The CLT – Recap

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

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

Also stated as:

- $S_n = X_1 + \cdots + X_n \approx \mathcal{N}(n\mu, n\sigma^2)$. If n large, this is a decent approximation.
- $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i \to \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$ where $\mu = E[X_i]$ and $\sigma^2 = Var(X_i)$

One main application:

Use Normal Distribution to Approximate Y_n or S_n or $\frac{1}{n}\sum_{i=1}^{n} X_i$

Polling recap

- We wanted to estimate p, the fraction of the population voting in favor.

Formalizing Polls

Population size N, true fraction of voting in favor p, sample size n.

Problem: We don't know p

Polling Procedure (Plan)

```
for i = 1 … $n$ :
    1. Pick uniformly random person to call (prob: $1/N$)
    2. Ask them how they will vote
```

$$X_i = \begin{cases} 1, & \text{voting in favor} \\ 0, & \text{otherwise} \end{cases}$$

Report our estimate of p: $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$

Polling recap

- We wanted to estimate p, the fraction of the population voting in favor.
- We discussed the notion of a **confidence interval**. We came up with a value of n such that when we do this procedure, we'd be 98% confident that true $p \in [\,\hat{p} \pm 0.05\,]$.

Polling recap

- We wanted to estimate p, the fraction of the population voting in favor.
- We discussed the notion of a **confidence interval**. We came up with a value of n such that when we do this procedure, we'd be 98% confident that true $p \in [\,\hat{p} \pm 0.05]\,.$
- We used CLT and $p(1-p) \leq 0.5$ to find a good n.

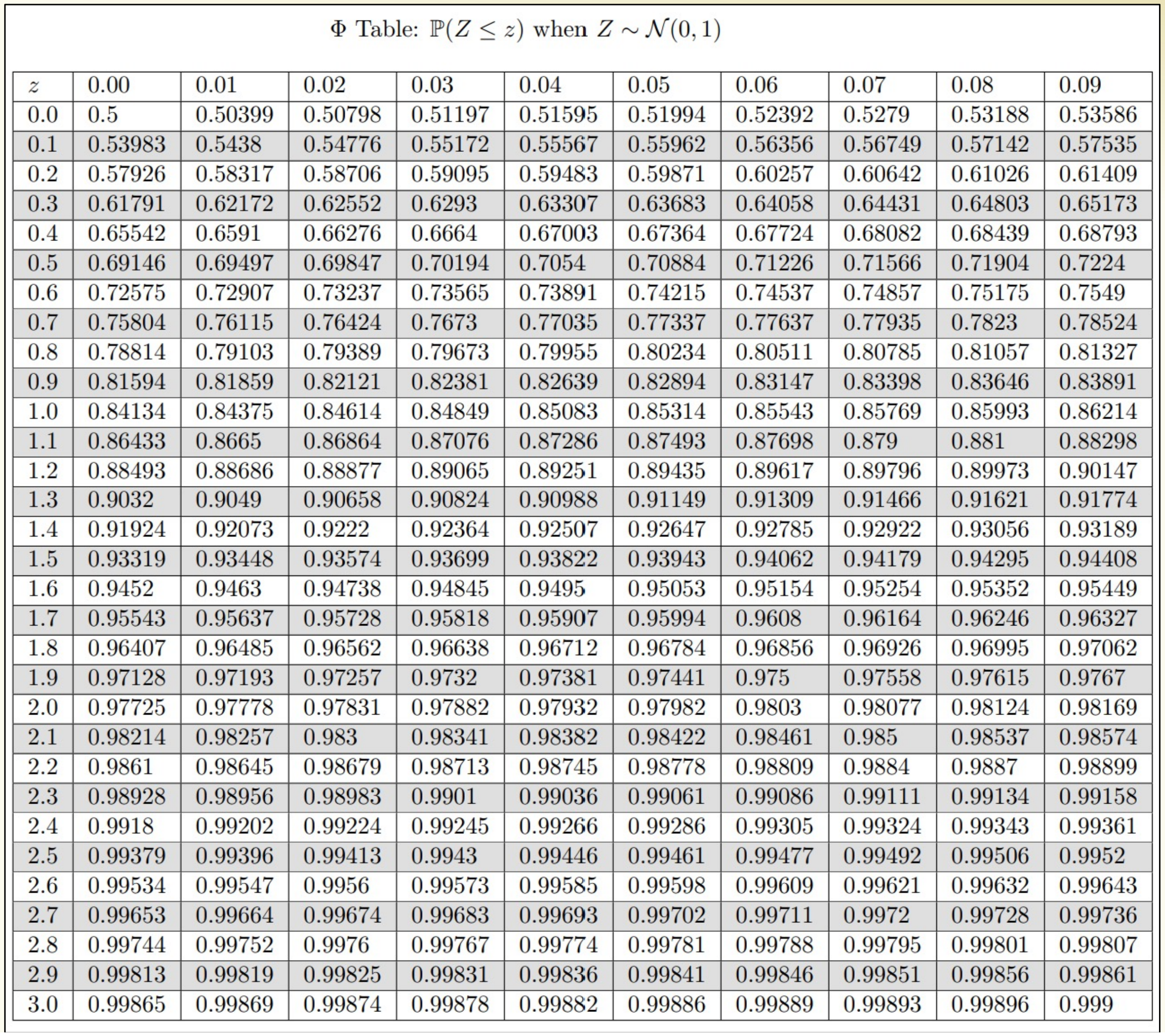
- First foray into statistics!!
- Statistical inference: using samples to infer something about the distribution.

Agenda

- One more CLT example
- Continuity correction
- Application: Distinct Elements

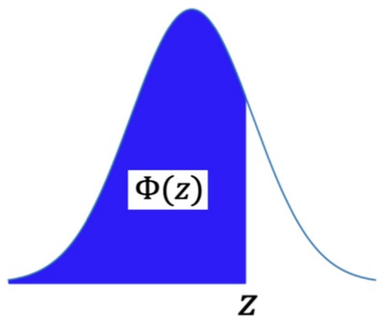
You're trying to figure out your finances over the upcoming year. Based on your recent spending patterns, you know that you spend $1200 a month on average, with a standard deviation of $400, and each month's spending is independent and identically distributed. In addition, because these are hard times, you don't have any income. How much money should you have in your bank account right now if you don't want to go broke in the next 12 months, with probability at least .95? Assume that your spending over the next 12 months is approximately normally distributed (by the CLT).

Table of $\Phi(z)$ CDF of Standard Normal Distn



Φ Table: $\mathbb{P}(Z \leq z)$ when $Z \sim \mathcal{N}(0,1)$

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.5279	0.53188	0.53586
0.1	0.53983	0.5438	0.54776	0.55172	0.55567	0.55962	0.56356	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.6293	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.6591	0.66276	0.6664	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.7054	0.70884	0.71226	0.71566	0.71904	0.7224
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.7549
0.7	0.75804	0.76115	0.76424	0.7673	0.77035	0.77337	0.77637	0.77935	0.7823	0.78524
0.8	0.78814	0.79103	0.79389	0.79673	0.79955	0.80234	0.80511	0.80785	0.81057	0.81327
0.9	0.81594	0.81859	0.82121	0.82381	0.82639	0.82894	0.83147	0.83398	0.83646	0.83891
1.0	0.84134	0.84375	0.84614	0.84849	0.85083	0.85314	0.85543	0.85769	0.85993	0.86214
1.1	0.86433	0.8665	0.86864	0.87076	0.87286	0.87493	0.87698	0.879	0.881	0.88298
1.2	0.88493	0.88686	0.88877	0.89065	0.89251	0.89435	0.89617	0.89796	0.89973	0.90147
1.3	0.9032	0.9049	0.90658	0.90824	0.90988	0.91149	0.91309	0.91466	0.91621	0.91774
1.4	0.91924	0.92073	0.9222	0.92364	0.92507	0.92647	0.92785	0.92922	0.93056	0.93189
1.5	0.93319	0.93448	0.93574	0.93699	0.93822	0.93943	0.94062	0.94179	0.94295	0.94408
1.6	0.9452	0.9463	0.94738	0.94845	0.9495	0.95053	0.95154	0.95254	0.95352	0.95449
1.7	0.95543	0.95637	0.95728	0.95818	0.95907	0.95994	0.9608	0.96164	0.96246	0.96327
1.8	0.96407	0.96485	0.96562	0.96638	0.96712	0.96784	0.96856	0.96926	0.96995	0.97062
1.9	0.97128	0.97193	0.97257	0.9732	0.97381	0.97441	0.975	0.97558	0.97615	0.9767
2.0	0.97725	0.97778	0.97831	0.97882	0.97932	0.97982	0.9803	0.98077	0.98124	0.98169
2.1	0.98214	0.98257	0.983	0.98341	0.98382	0.98422	0.98461	0.985	0.98537	0.98574
2.2	0.9861	0.98645	0.98679	0.98713	0.98745	0.98778	0.98809	0.9884	0.9887	0.98899
2.3	0.98928	0.98956	0.98983	0.9901	0.99036	0.99061	0.99086	0.99111	0.99134	0.99158
2.4	0.9918	0.99202	0.99224	0.99245	0.99266	0.99286	0.99305	0.99324	0.99343	0.99361
2.5	0.99379	0.99396	0.99413	0.9943	0.99446	0.99461	0.99477	0.99492	0.99506	0.9952
2.6	0.99534	0.99547	0.9956	0.99573	0.99585	0.99598	0.99609	0.99621	0.99632	0.99643
2.7	0.99653	0.99664	0.99674	0.99683	0.99693	0.99702	0.99711	0.9972	0.99728	0.99736
2.8	0.99744	0.99752	0.9976	0.99767	0.99774	0.99781	0.99788	0.99795	0.99801	0.99807
2.9	0.99813	0.99819	0.99825	0.99831	0.99836	0.99841	0.99846	0.99851	0.99856	0.99861
3.0	0.99865	0.99869	0.99874	0.99878	0.99882	0.99886	0.99889	0.99893	0.99896	0.999



Agenda

- One more CLT example
- Continuity correction
- Application: Distinct Elements

Example – Naive Approximation of Binomial

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

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

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

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

Example – Even Worse Approximation

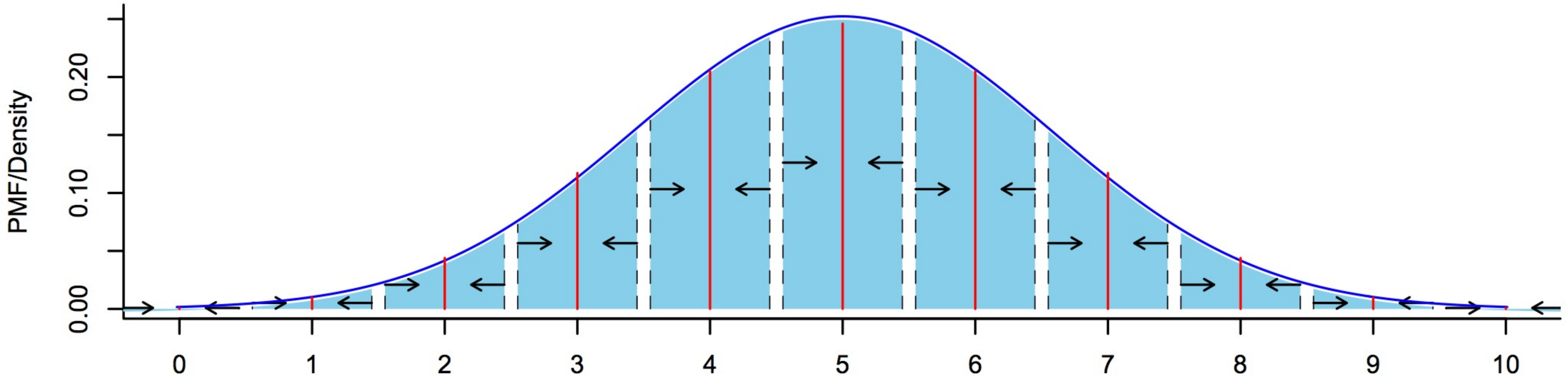
Fair coin flipped (independently) **40** times. Probability of **20** heads?

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

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 $\quad \mu = \mathbb{E}(X) = 0.5n = 20 \quad \sigma^2 = \text{Var}(X) = 0.25n = 10$

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

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

👍

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

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

- One more CLT example
- Continuity correction
- Application: Distinct Elements

Data mining – Stream Model

- In many data mining situations, the data is not known ahead of time.
  Examples: Google queries, Twitter or Facebook status updates
  Youtube video views
- In some ways, best to think of the data as an infinite stream that is non-stationary (distribution changes over time)
- 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?

Problem Setup

- Input: sequence 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

- 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.
How?

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 O(m) , where m is the number of distinct IDs*

- *Consider the number of users of youtube, and the number of videos on youtube. This is not feasible.*

Counting distinct elements

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

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

- *How to do this without storing all the elements?*

Yet another super cool application of probability

Counting distinct elements

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

$\boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{y_3}, \boldsymbol{y_1}, y_4, \boldsymbol{y_2}, \boldsymbol{y_1}, \boldsymbol{y_4}, \boldsymbol{y_1}, y_2, y_5$

Hash function $h: U \rightarrow [0,1]$
Assumption: For distinct values in U, the function maps to iid (independent and identically distributed) Unif(0,1) random numbers.

Important: if you were to feed in two equivalent elements, the function returns the **same** number.

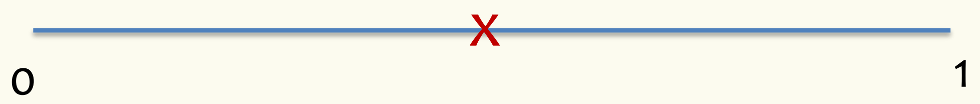
- So m distinct elements → m iid uniform y_i's

Min of IID Uniforms

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

In general, $\mathrm{E}[\min(Y_1, \cdots, Y_m)] = \ ?$

$m = 1$



$\mathrm{E}[\min(Y_1)] =$

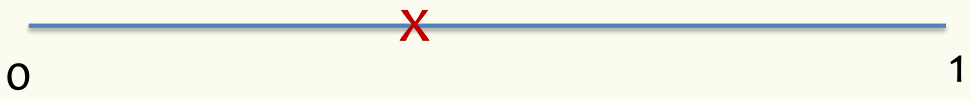
Min of IID Uniforms

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

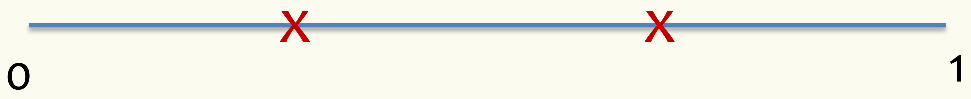
In general, $\mathrm{E}[\min(Y_1, \cdots, Y_m)] = ?$

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

$m = 1$



$m = 2$



$\mathrm{E}[\min(Y_1, Y_2)] = ?$

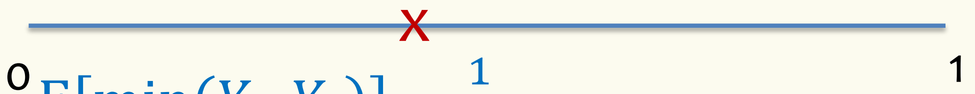
Min of IID Uniforms

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

In general, $\mathrm{E}[\min(Y_1, \cdots, Y_m)] = ?$

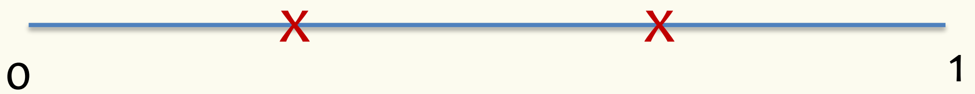
$$\mathrm{E}[\min(Y_1)] = \frac{1}{2}$$

$m = 1$

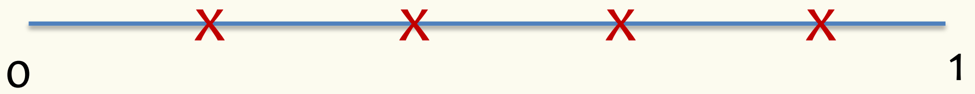


$$\mathrm{E}[\min(Y_1, Y_2)] = \frac{1}{3}$$

$m = 2$



$m = 4$



$$\mathrm{E}[\min(Y_1, \cdots, Y_4)] =$$

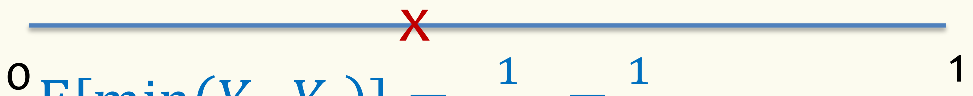
Min of IID Uniforms

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

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

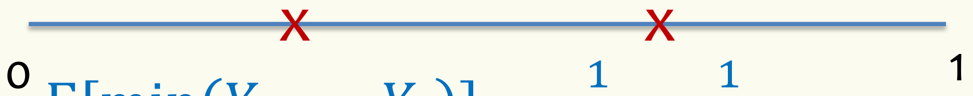
$$\mathrm{E}[\min(Y_1)] = \frac{1}{1+1} = \frac{1}{2}$$

$m = 1$



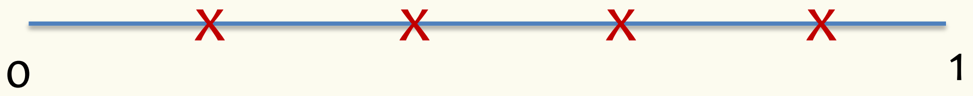
$$\mathrm{E}[\min(Y_1, Y_2)] = \frac{1}{2+1} = \frac{1}{3}$$

$m = 2$



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

$m = 4$



If $Y_1, \cdots, Y_m$ are iid Unif(0,1), then $\mathrm{E}[\min(Y_1, \cdots, Y_m)] = \frac{1}{m+1}$

Counting distinct elements

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

$y_1, \ y_2, \ y_3, \ y_1, \ y_4, y_2, \ y_1, \ y_4, \ y_1, \ y_2, \ y_5$

Hash function $h: U \to [0,1]$ (hashes to a uniform value).

- So m distinct elements → m iid uniform values.

$$val = \min\left(h(X_1), \cdots, h(X_N)\right) = \min(Y_1, \cdots, Y_m)$$

A super duper clever idea!!!!!

If $Y_1, \cdots, Y_n$ are iid Unif(0,1), where do we expect the points to end up?

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

Idea: $\mathrm{m} = \frac{1}{\mathrm{E}[\min(Y_1,\cdots,Y_m)]} - 1$

A super duper clever idea!!!!!

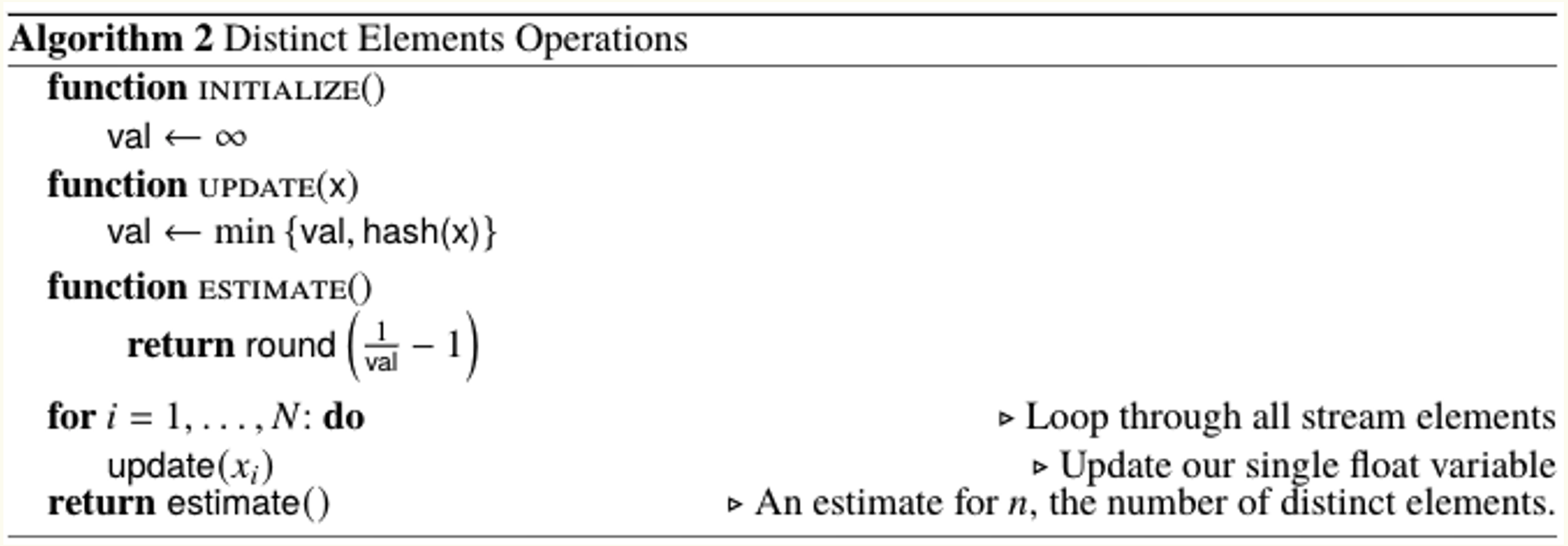
If $Y_1, \cdots, Y_n$ are iid Unif(0,1), where do we expect the points to end up?

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

Idea: $\mathrm{m} = \frac{1}{\mathrm{E}[\min(Y_1,\cdots,Y_m)]} - 1$

Let's keep track of the value val of min of hash values, and estimate m as Round $\left(\frac{1}{val} - 1\right)$

The Distinct Elements Algorithm



Algorithm 2 Distinct Elements Operations

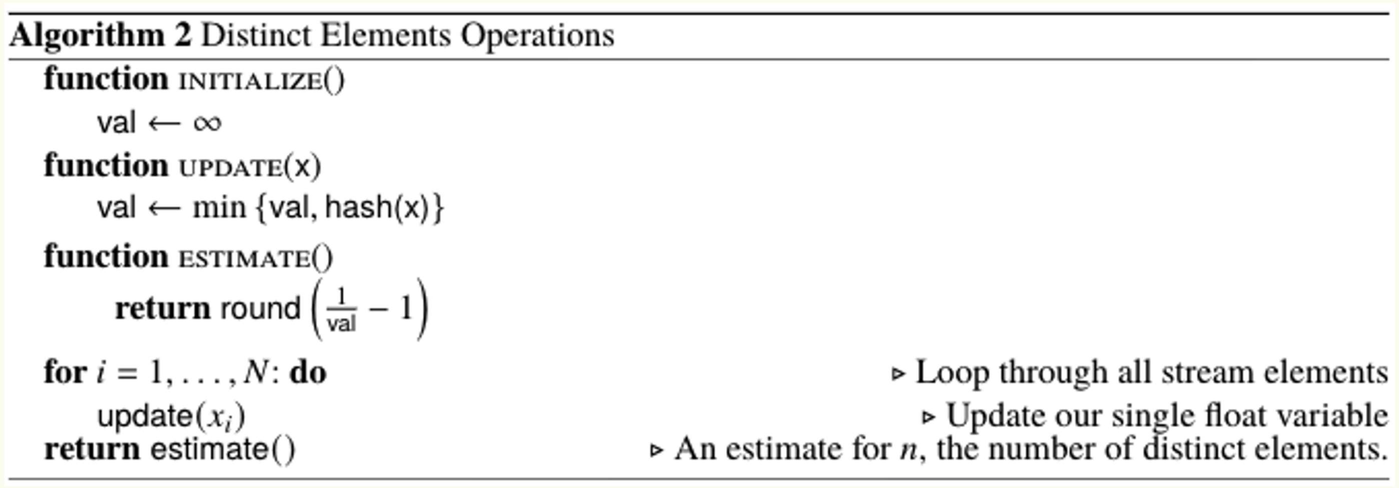
```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: do                         ▷ Loop through all stream elements
    update($x_i$)                                  ▷ Update our single float variable
return estimate()         ▷ An estimate for $n$, the number of distinct elements.
```

val=

Distinct Elements Example

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

Hashes:



Algorithm 2 Distinct Elements Operations

```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \dots, N$: do                    ▹ Loop through all stream elements
    update($x_i$)                            ▹ Update our single float variable
return estimate()      ▹ An estimate for $n$, the number of distinct elements.
```

Suppose that

$\boldsymbol{h(13) = 0.51}$

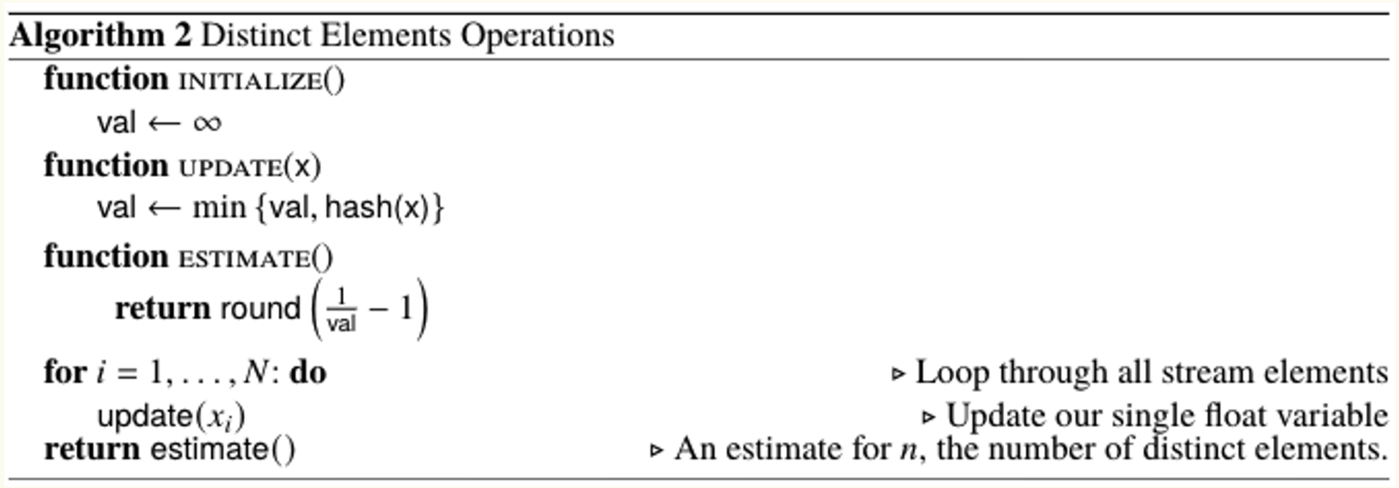
$\boldsymbol{h(25) = 0.26}$

$\boldsymbol{h(19) = 0.79}$

Distinct Elements Example

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

Hashes:



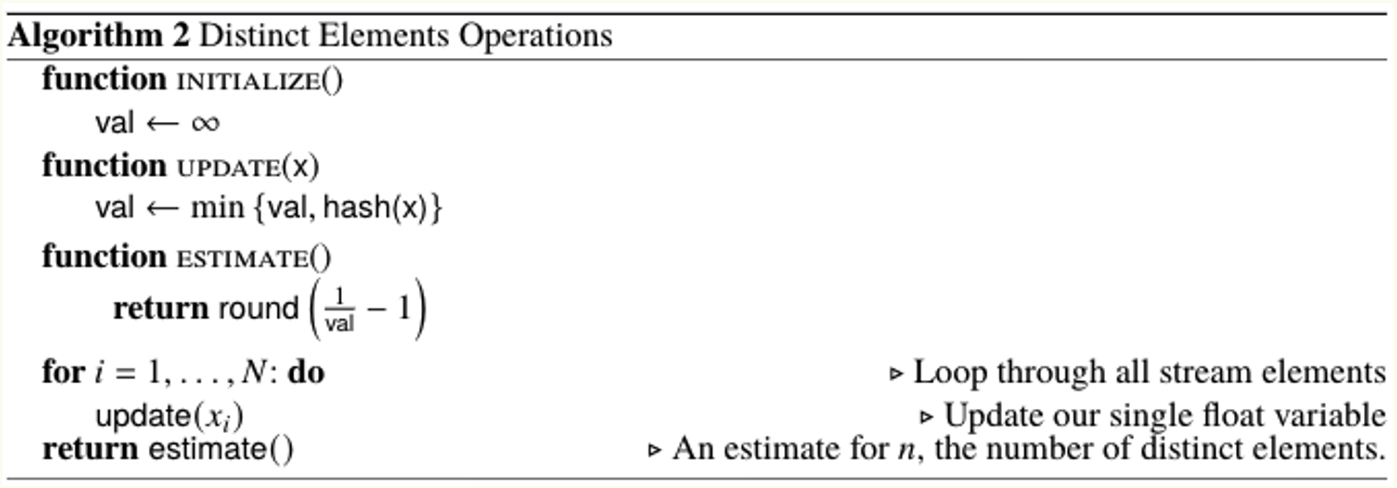
```
Algorithm 2 Distinct Elements Operations
  function INITIALIZE()
      val ← $\infty$
  function UPDATE(x)
      val ← min {val, hash(x)}
  function ESTIMATE()
      return round $\left(\frac{1}{\text{val}} - 1\right)$
  for $i = 1, \ldots, N$: do                      ▷ Loop through all stream elements
      update($x_i$)                                ▷ Update our single float variable
  return estimate()         ▷ An estimate for $n$, the number of distinct elements.
```

val = infty

Distinct Elements Example

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

Hashes: 0.51,



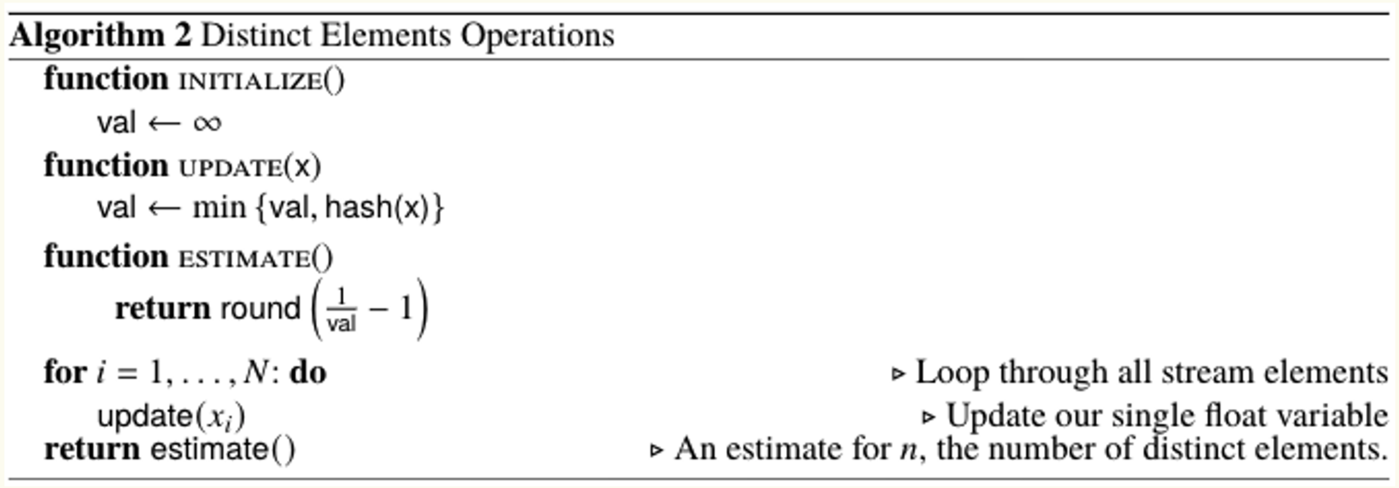
```
Algorithm 2 Distinct Elements Operations
  function INITIALIZE()
      val ← $\infty$
  function UPDATE(x)
      val ← min {val, hash(x)}
  function ESTIMATE()
      return round $\left(\frac{1}{\text{val}} - 1\right)$
  for $i = 1, \ldots, N$: do                 ▹ Loop through all stream elements
      update($x_i$)                          ▹ Update our single float variable
  return estimate()    ▹ An estimate for $n$, the number of distinct elements.
```

val = infty

Distinct Elements Example

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

Hashes: 0.51,



Algorithm 2 Distinct Elements Operations

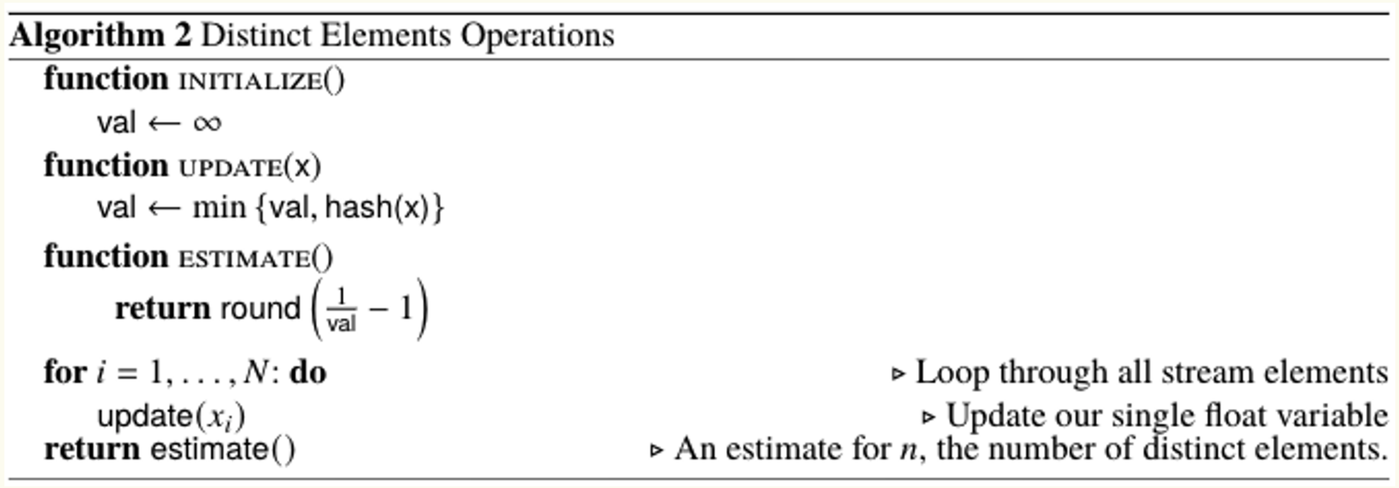
```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: do                 ▹ Loop through all stream elements
    update($x_i$)                          ▹ Update our single float variable
return estimate()                          ▹ An estimate for $n$, the number of distinct elements.
```

val = 0.51

Distinct Elements Example

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

Hashes: 0.51, 0.26,



Algorithm 2 Distinct Elements Operations

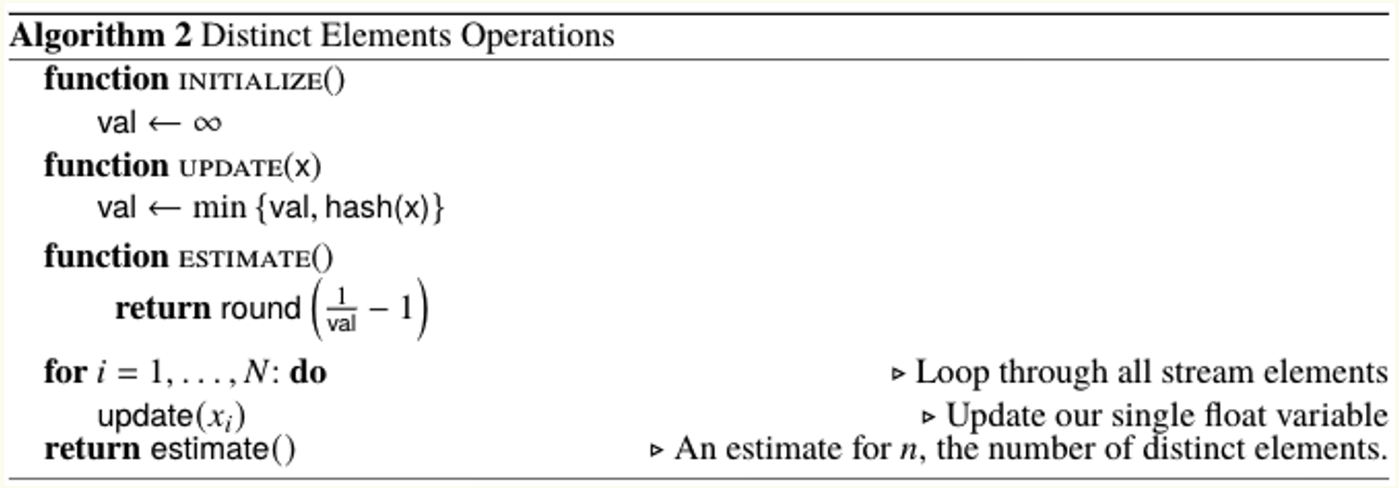
```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: do                    ▹ Loop through all stream elements
    update($x_i$)                             ▹ Update our single float variable
return estimate()                ▹ An estimate for $n$, the number of distinct elements.
```

val = 0.26

Distinct Elements Example

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

Hashes: 0.51, 0.26, 0.79,



Algorithm 2 Distinct Elements Operations

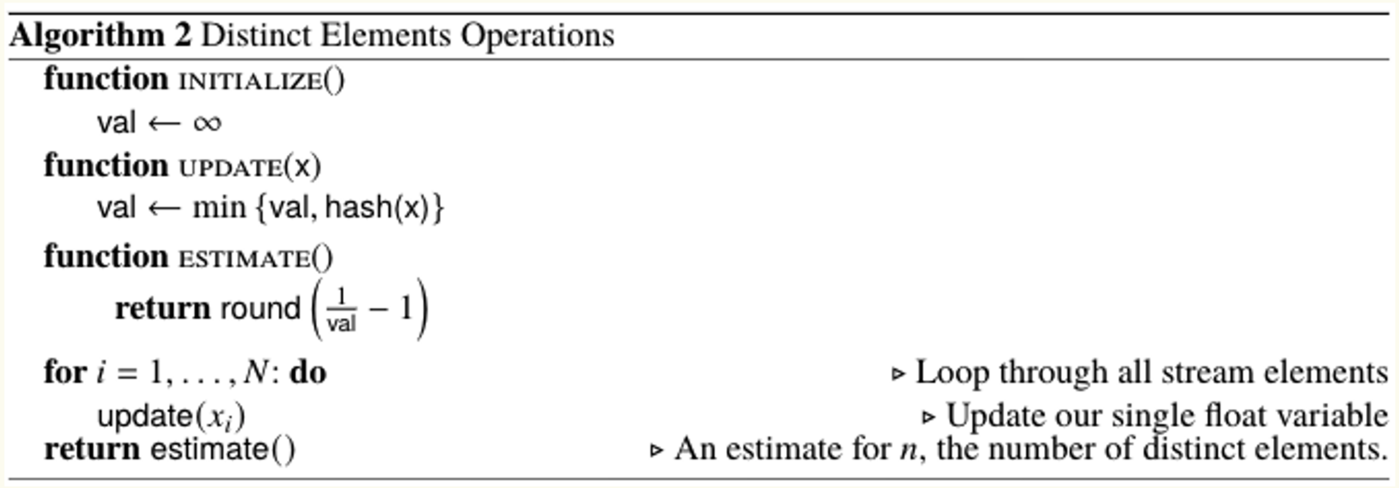
```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \dots, N$: do                ▹ Loop through all stream elements
    update($x_i$)                        ▹ Update our single float variable
return estimate()                        ▹ An estimate for $n$, the number of distinct elements.
```

val = 0.26

Distinct Elements Example

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

Hashes: 0.51, 0.26, 0.79, 0.26,



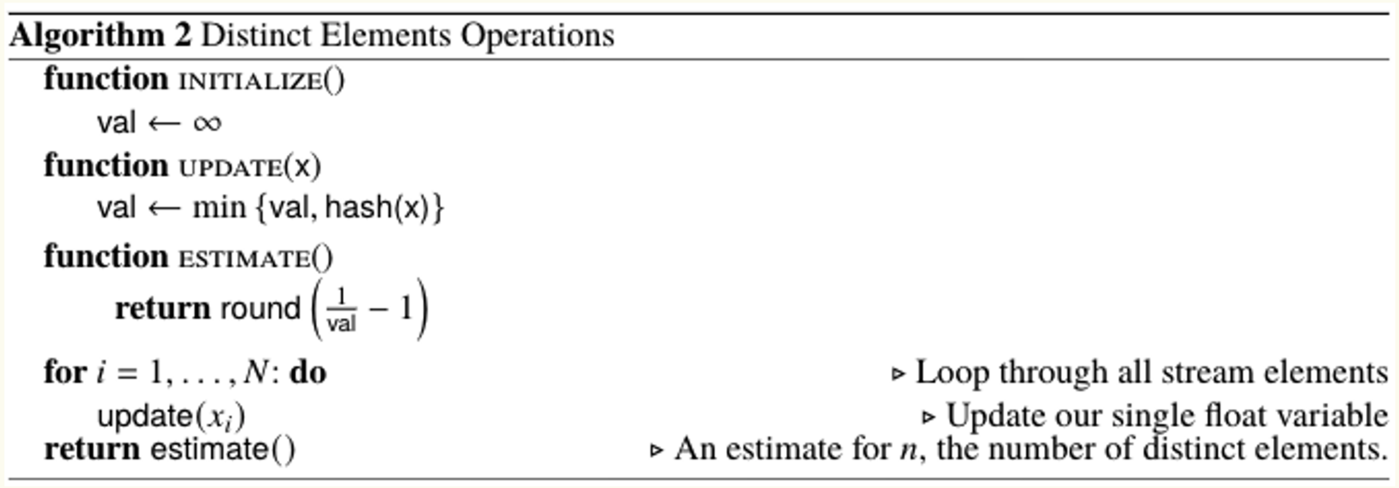
```
Algorithm 2 Distinct Elements Operations
  function INITIALIZE()
      val ← ∞
  function UPDATE(x)
      val ← min {val, hash(x)}
  function ESTIMATE()
      return round $\left(\frac{1}{\text{val}} - 1\right)$
  for $i = 1, \ldots, N$: do                     ▹ Loop through all stream elements
      update($x_i$)                              ▹ Update our single float variable
  return estimate()        ▹ An estimate for $n$, the number of distinct elements.
```

val = 0.26

Distinct Elements Example

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

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



Algorithm 2 Distinct Elements Operations

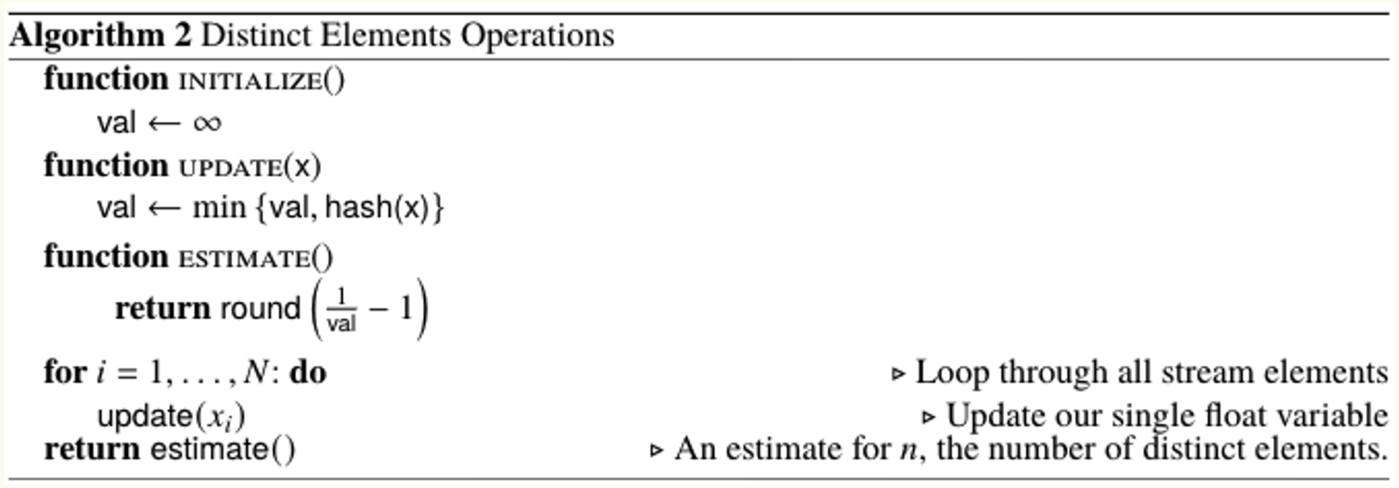
```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: do                      ▹ Loop through all stream elements
    update($x_i$)                                ▹ Update our single float variable
return estimate()          ▹ An estimate for $n$, the number of distinct elements.
```

val = 0.26

Distinct Elements Example

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

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



Algorithm 2 Distinct Elements Operations

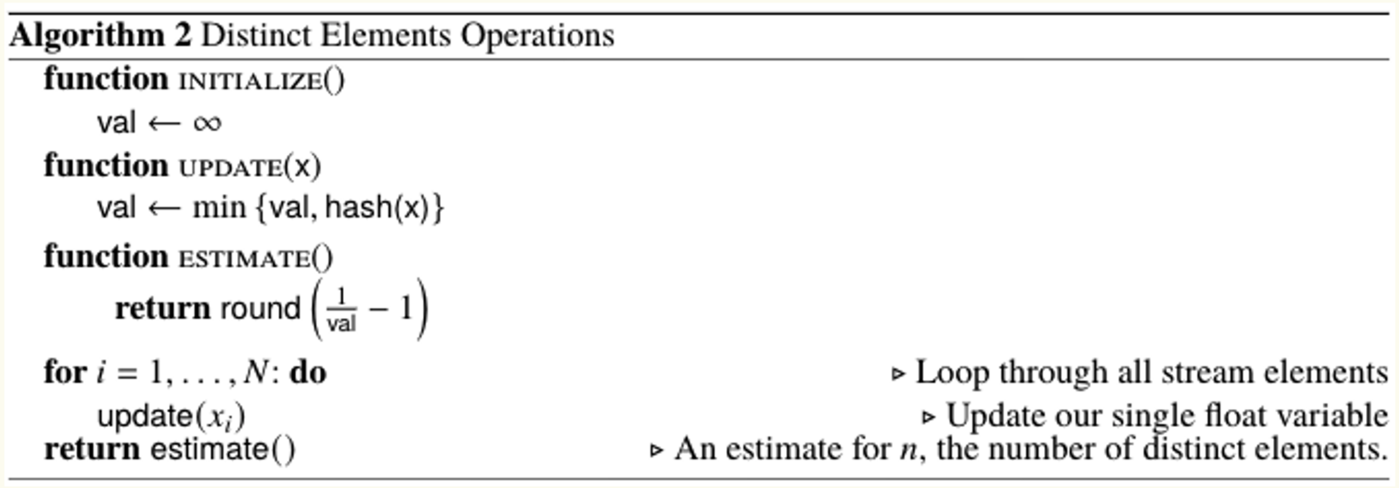
```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: do                    ▹ Loop through all stream elements
    update($x_i$)                             ▹ Update our single float variable
return estimate()      ▹ An estimate for $n$, the number of distinct elements.
```

val = 0.26

Distinct Elements Example

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

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



Algorithm 2 Distinct Elements Operations

```
function INITIALIZE()
    val ← ∞
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: do                    ▹ Loop through all stream elements
    update($x_i$)                             ▹ Update our single float variable
return estimate()   ▹ An estimate for $n$, the number of distinct elements.
```

val = 0.26

Return

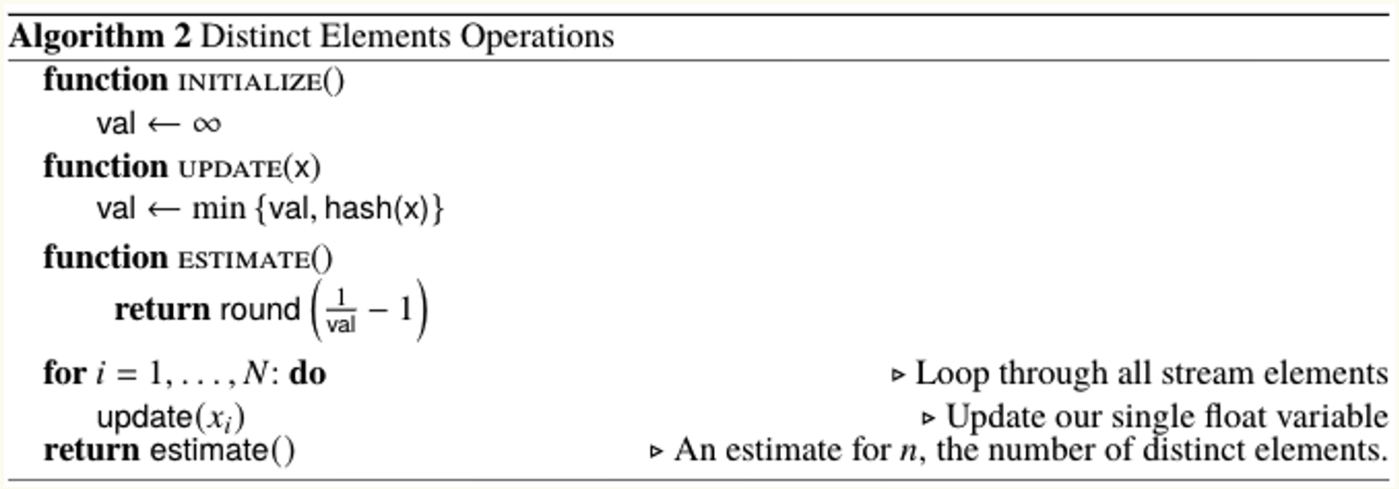
round(1/0.26 - 1) =

round(2.846) = 3

Diy: Distinct Elements Example II

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

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



Algorithm 2 Distinct Elements Operations

```
function INITIALIZE()
    val ← $\infty$
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: do                    ▷ Loop through all stream elements
    update($x_i$)                              ▷ Update our single float variable
return estimate()        ▷ An estimate for $n$, the number of distinct elements.
```

val = 0.1

Return= 9

Problem

val $= \min(Y_1, \cdots, Y_m)$ $\qquad$ $\mathrm{E}[val] = \frac{1}{m+1}$

Algorithm:

Track $val = \min\big(h(X_1), \cdots, h(X_N)\big) = \min(Y_1, \cdots, Y_m)$

estimate m = 1/ val -1

But, val is not E$[val]$! How far is val from E$[val]$?

Problem

$\text{val} = \min(Y_1, \cdots, Y_m)$ $\quad\quad$ $\mathrm{E}[val] = \frac{1}{m+1}$

Algorithm:

Track $val = \min\big(h(X_1), \cdots, h(X_N)\big) = \min(Y_1, \cdots, Y_m)$

estimate m = 1/ val -1

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

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

What can we do to fix this?

How can we reduce the variance?

Idea: Repetition to reduce variance!

How can we reduce the variance?

Idea: Repetition to reduce variance!

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

Keep track of k independent min hash values

$$val^1 = \min\left(h^1(x_1), \cdots, h^1(x_N)\right) = \min(\mathrm{Y}_1^1, \cdots, Y_m^1)$$
$$val^2 = \min\left(h^2(x_1), \cdots, h^2(x_N)\right) = \min(\mathrm{Y}_1^2, \cdots, Y_m^2)$$
$$\cdots \; \cdots$$
$$val^k = \min\left(h^k(x_1), \cdots, h^k(x_N)\right) = \min(\mathrm{Y}_1^\mathrm{k}, \cdots, Y_m^k)$$

$val = \frac{1}{k}\Sigma_i val_i,$ Estimate $m = \frac{1}{val} - 1$

If $Y_1, \cdots, Y_m$ are iid Unif(0,1), then $\mathrm{E}[\min(Y_1, \cdots, Y_m)] = \frac{1}{m+1}$