

Hashing and Sketching

Part One

Outline for Today

- ***Hash Functions***
 - Understanding our basic building blocks.
- ***Frequency Estimation***
 - Estimating how many times we've seen something.
- ***Probabilistic Techniques***
 - Standard but powerful tools for reasoning about randomized data structures.

Preliminaries: *Hash Functions*

Hashing in Practice

- Hash functions are used extensively in programming and software engineering:
 - They make hash tables possible: think C++ `std::hash`, Python's `__hash__`, or Java's `Object.hashCode()`.
 - They're used in cryptography: SHA-256, HMAC, etc.
- **Question:** When we're in Theoryland, what do we mean when we say "hash function?"

Hashing in Theoryland

- In Theoryland, a hash function is a function from some domain called the ***universe*** (typically denoted \mathcal{U}) to some codomain.
- The codomain is usually a set of the form
 $[m] = \{0, 1, 2, 3, \dots, m - 1\}$

$$h : \mathcal{U} \rightarrow [m]$$

Hashing in Theoryland

- ***Intuition:*** No matter how clever you are with designing a specific hash function, that hash function isn't random, and so there will be pathological inputs.
 - You can formalize this with the pigeonhole principle.
- ***Idea:*** Rather than finding the One True Hash Function, we'll assume we have a collection of hash functions to pick from, and we'll choose which one to use randomly.

Families of Hash Functions

- A **family** of hash functions is a set \mathcal{H} of hash functions with the same domain and codomain.
- We can then introduce randomness into our data structures by sampling a random hash function from \mathcal{H} .
- **Key Point:** The randomness in our data structures almost always derives from the random choice of hash functions, not from the data.

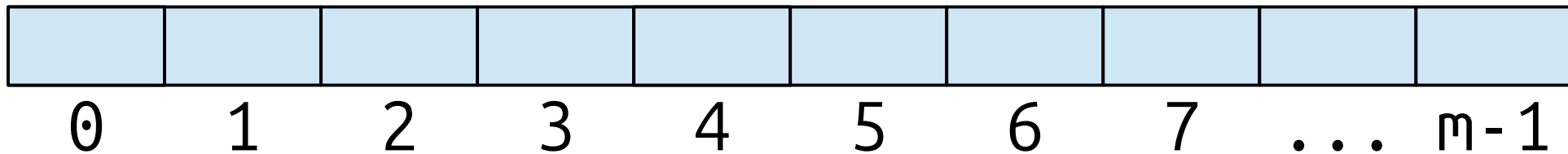
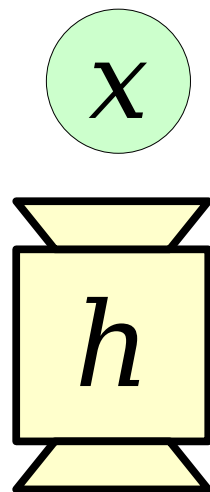


Data are adversarial.

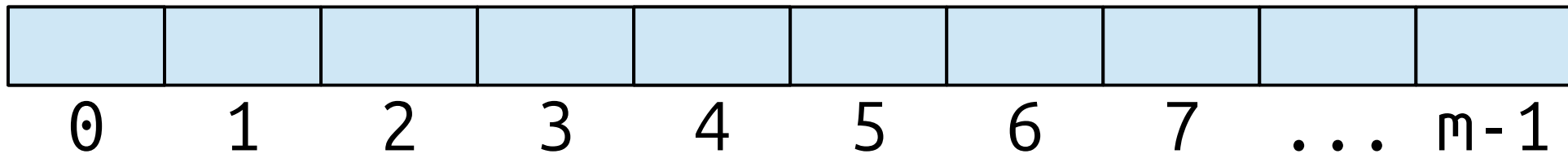
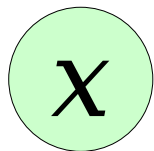
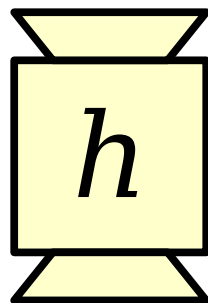
Hash function selection is random.

- **Question:** What makes a family of hash functions \mathcal{H} a “good” family of hash functions?

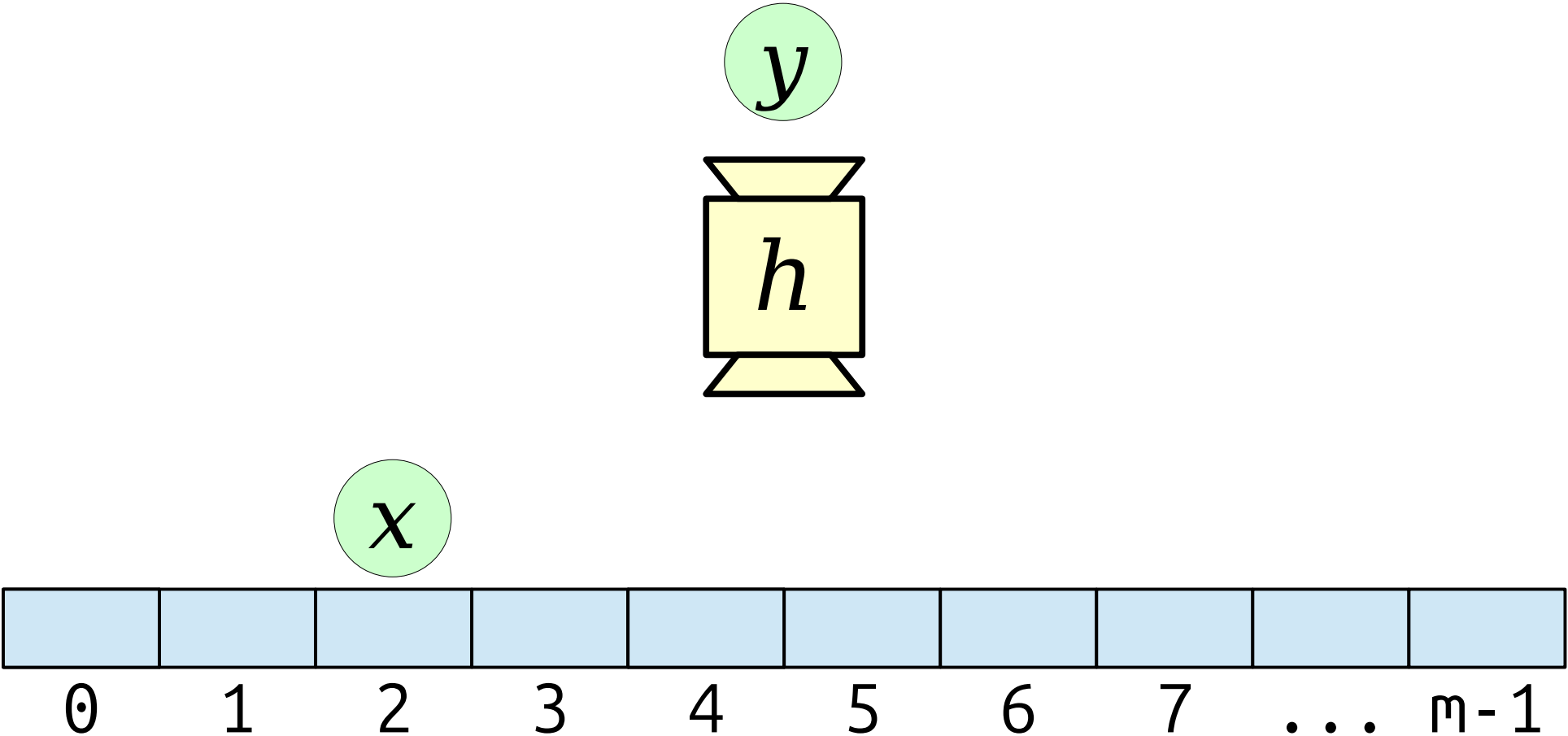
Goal: If we pick $h \in \mathcal{H}$ uniformly at random, then h should distribute elements uniformly randomly.



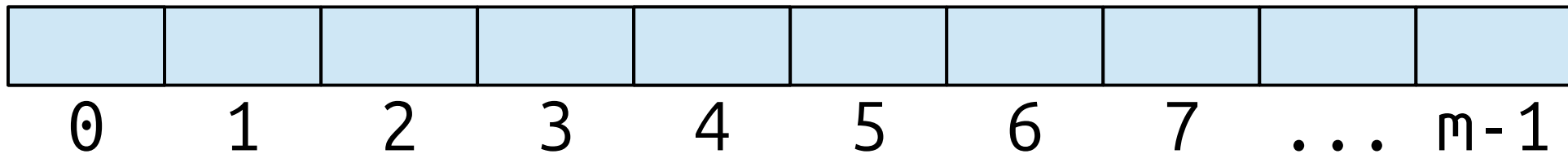
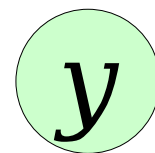
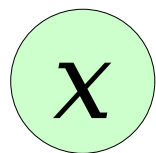
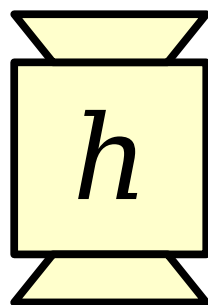
Goal: If we pick $h \in \mathcal{H}$ uniformly at random, then h should distribute elements uniformly randomly.



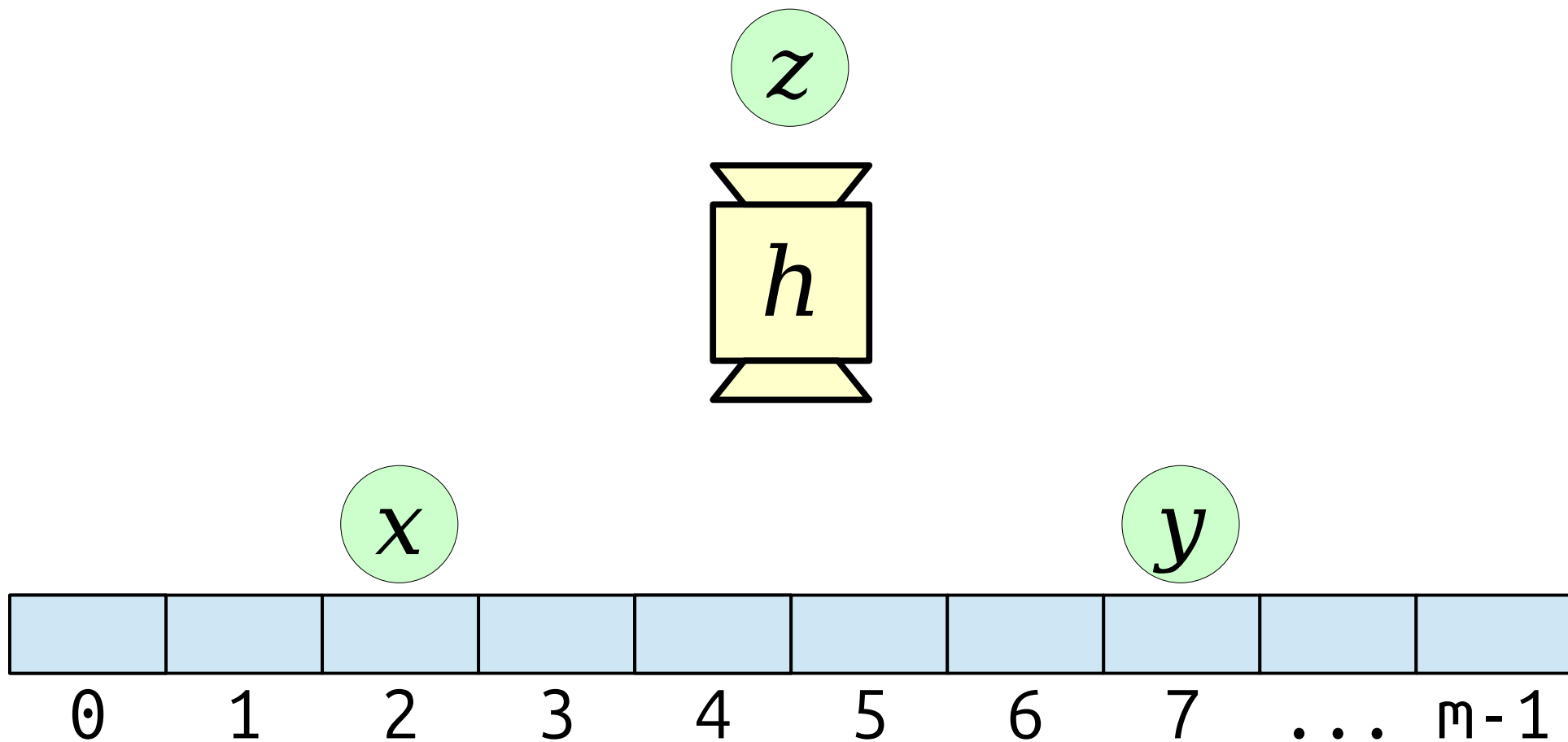
Goal: If we pick $h \in \mathcal{H}$ uniformly at random, then h should distribute elements uniformly randomly.



Goal: If we pick $h \in \mathcal{H}$ uniformly at random, then h should distribute elements uniformly randomly.



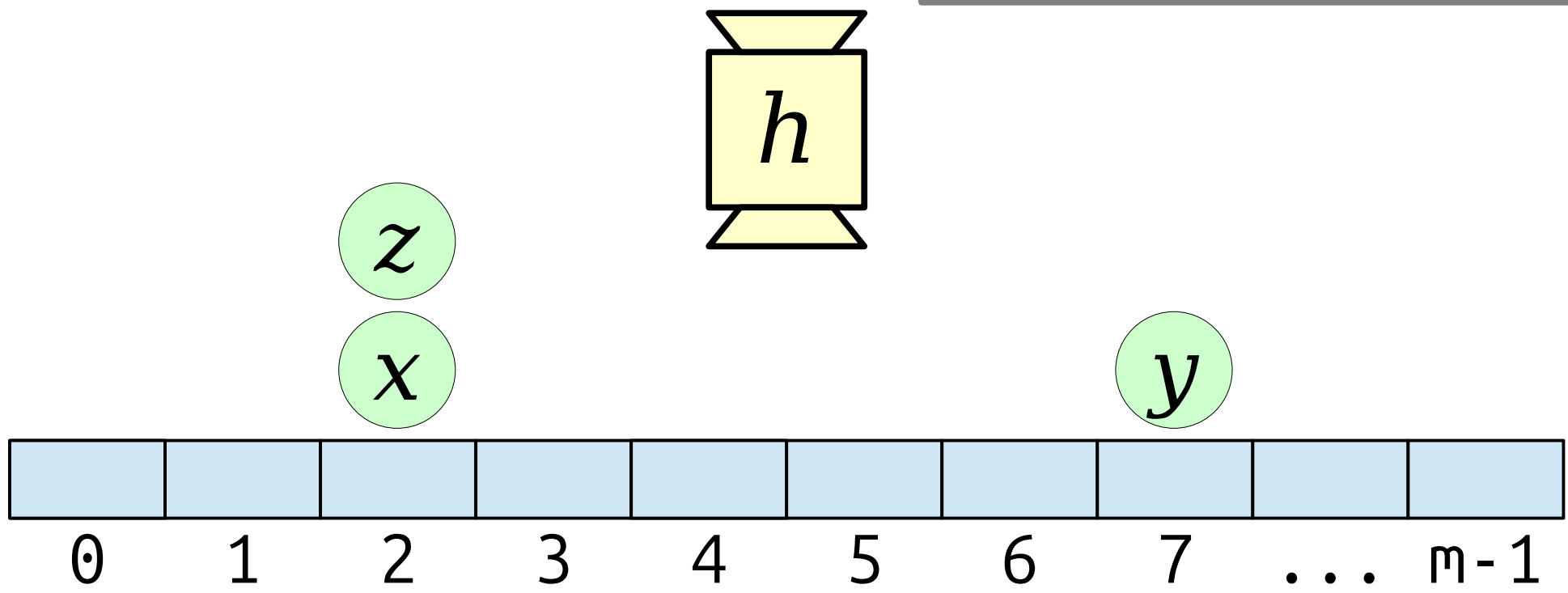
Goal: If we pick $h \in \mathcal{H}$ uniformly at random, then h should distribute elements uniformly randomly.



Goal: If we pick $h \in \mathcal{H}$ uniformly at random, then h should distribute elements uniformly randomly.

Problem: A hash function that distributes n elements uniformly at random over $[m]$ requires $\Omega(n \log m)$ space in the worst case.

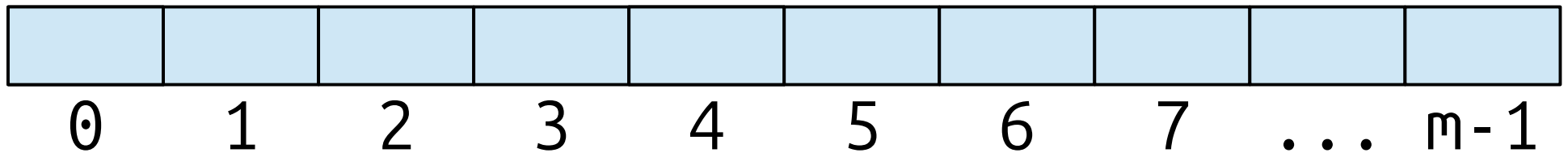
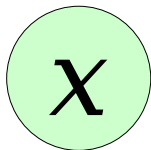
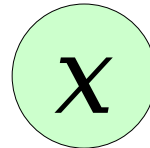
Question: Do we actually need true randomness? Or can we get away with something weaker?



Distribution Property:

Each element should have an equal probability of being placed in each slot.

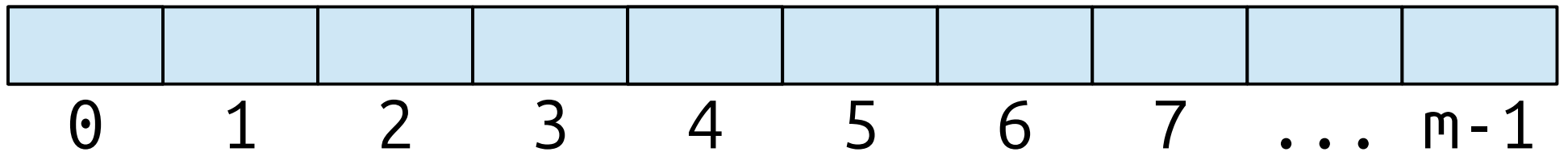
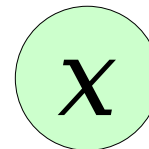
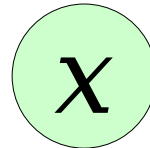
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.



Distribution Property:

Each element should have an equal probability of being placed in each slot.

For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.



Distribution Property:

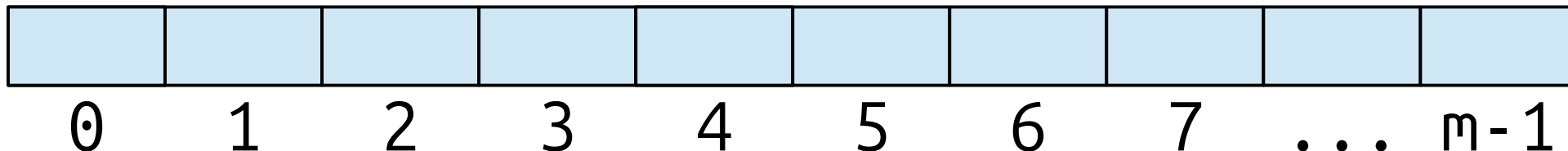
Each element should have an equal probability of being placed in each slot.

For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

Find an “obviously bad” family of hash functions that satisfies the distribution property.

Answer at

<https://cs166.stanford.edu/pollev>

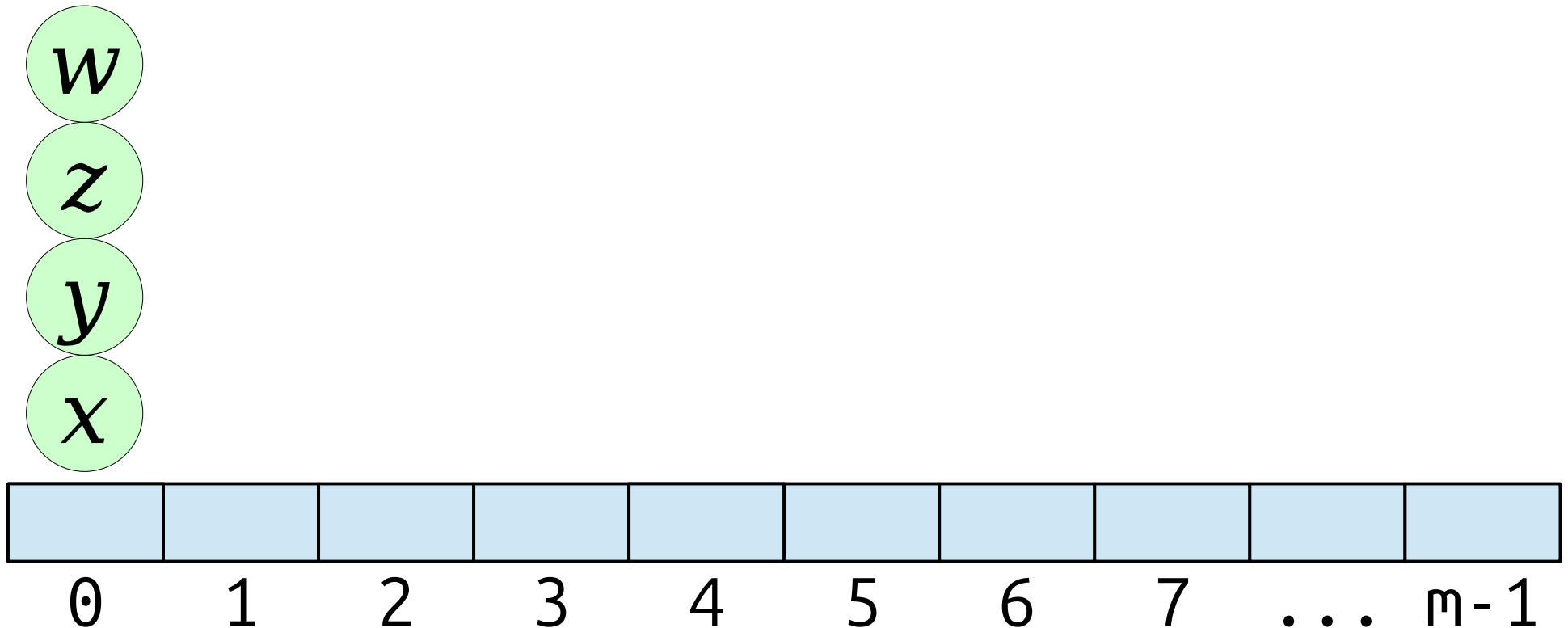


Distribution Property:

Each element should have an equal probability of being placed in each slot.

For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

Problem: This rule doesn't guarantee that elements are spread out.

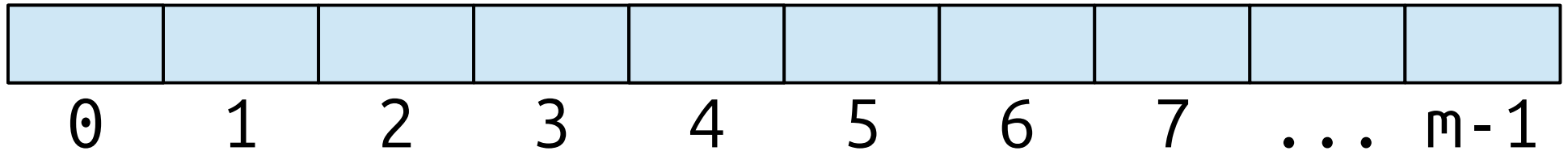
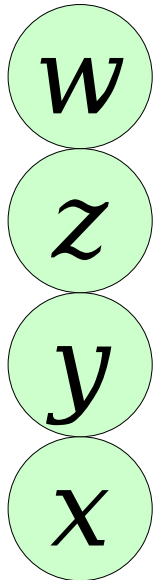


Distribution Property:

Each element should have an equal probability of being placed in each slot.

For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

Problem: This rule doesn't guarantee that elements are spread out.

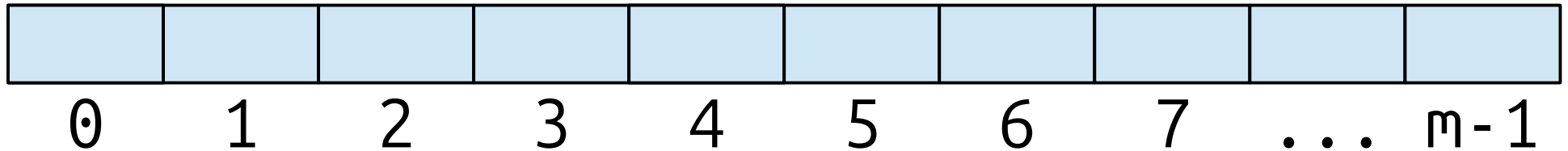
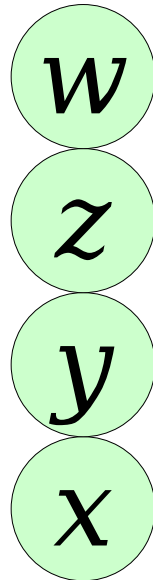


Distribution Property:

Each element should have an equal probability of being placed in each slot.

For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

Problem: This rule doesn't guarantee that elements are spread out.



Distribution Property:

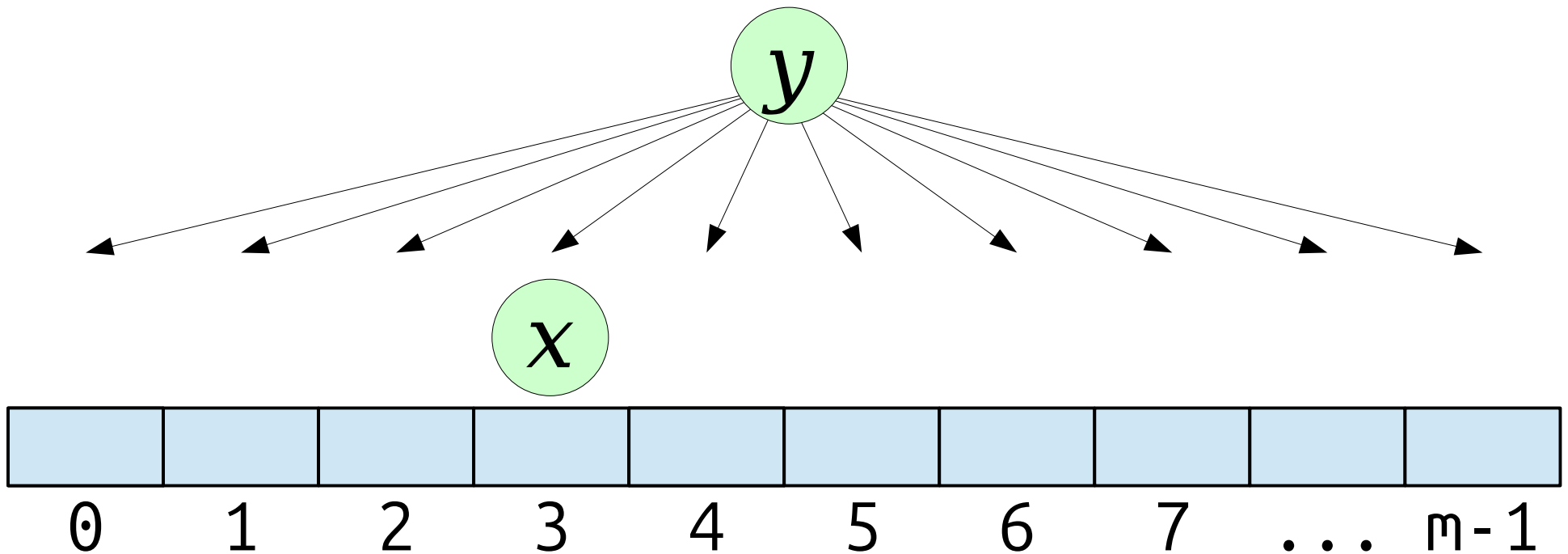
Each element should have an equal probability of being placed in each slot.

For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

Independence Property:

Where one element is placed shouldn't impact where a second goes.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.



Distribution Property:

Each element should have an equal probability of being placed in each slot.

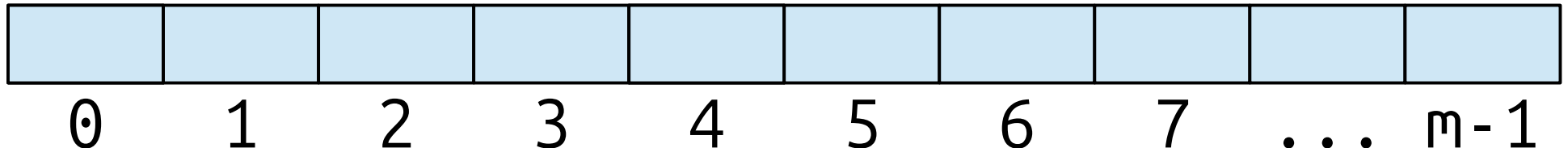
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

Independence Property:

Where one element is placed shouldn't impact where a second goes.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

A family of hash functions \mathcal{H} is called ***2-independent*** (or ***pairwise independent***) if it satisfies the distribution and independence properties.

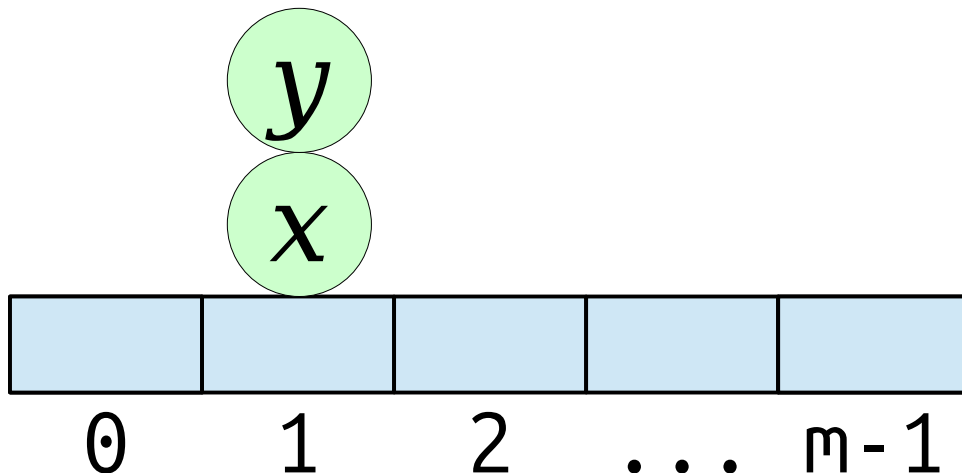


For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.



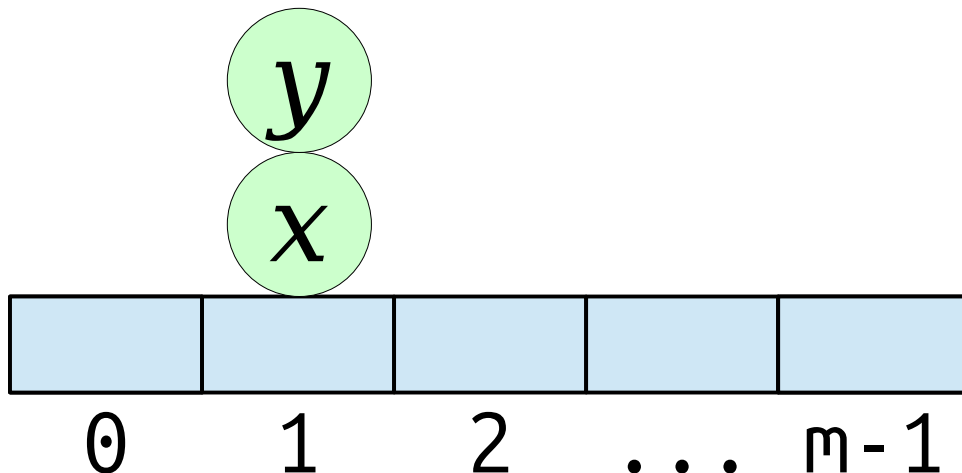
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\Pr[h(x) = h(y)]$$



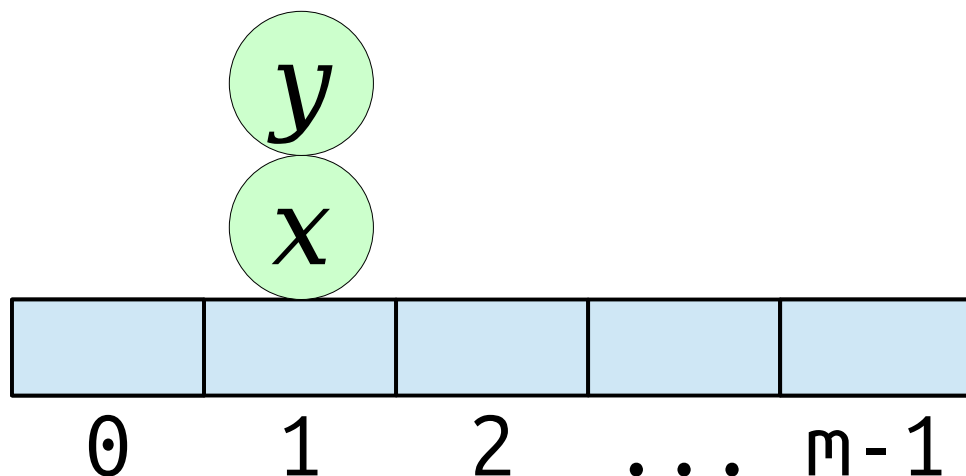
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\Pr[h(x) = h(y)]$$



Question: Where did these elements collide with one another?

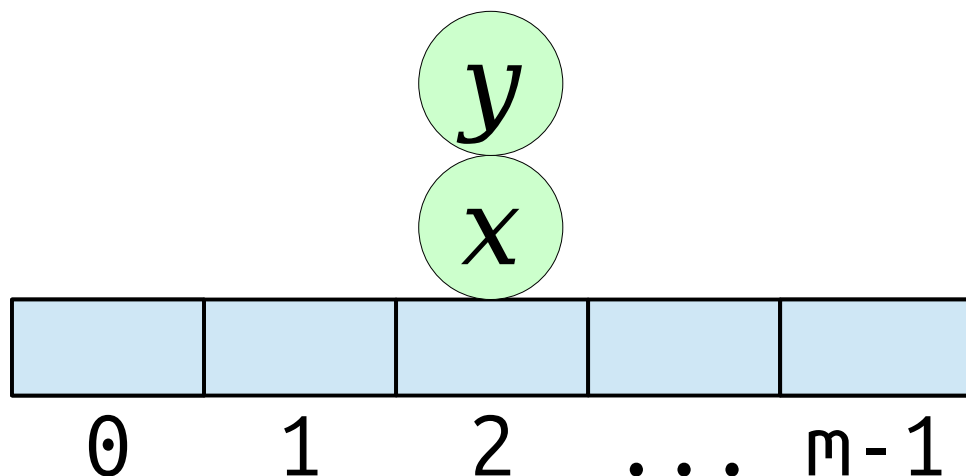
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\Pr[h(x) = h(y)]$$



Question: Where did these elements collide with one another?

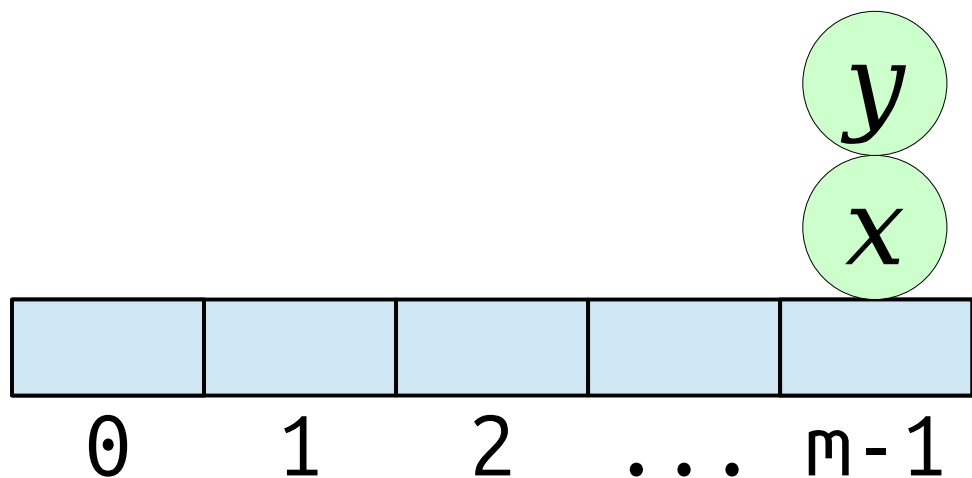
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\Pr[h(x) = h(y)]$$



Question: Where did these elements collide with one another?

For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

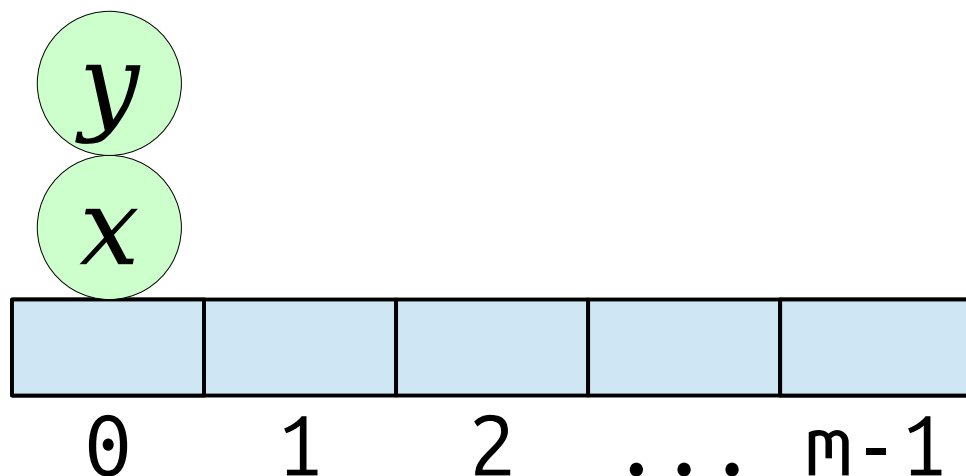
For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\Pr[h(x) = h(y)]$$

Question: Where did these elements collide with one another?



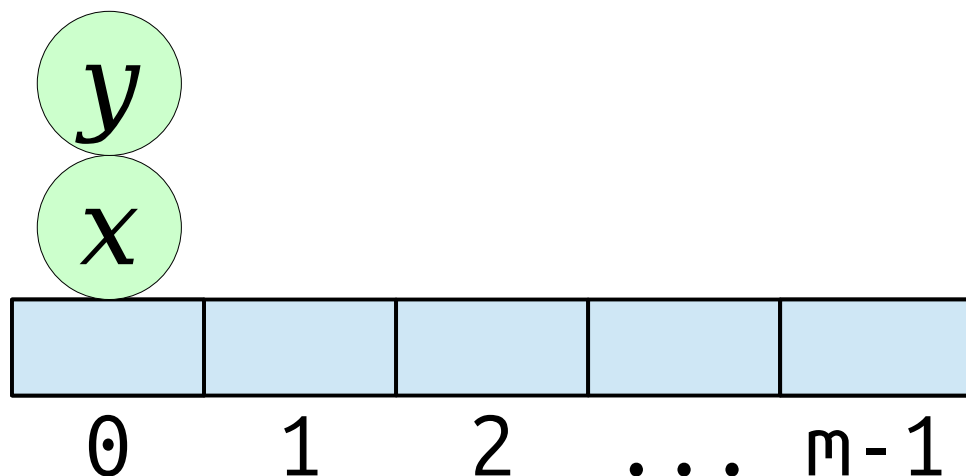
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \end{aligned}$$



Question: Where did these elements collide with one another?

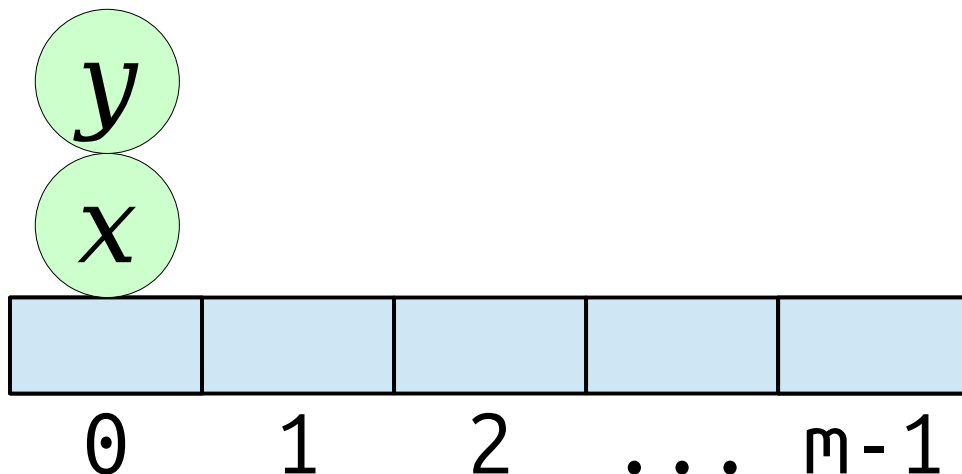
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \end{aligned}$$



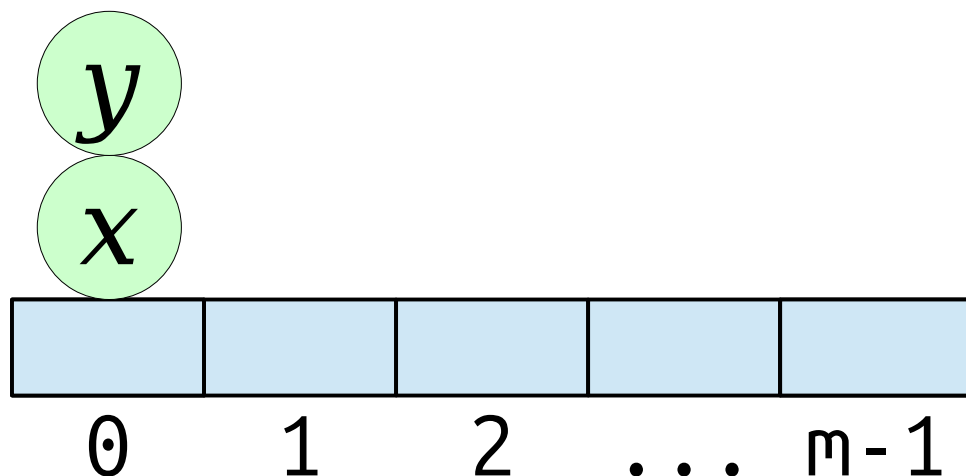
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \end{aligned}$$



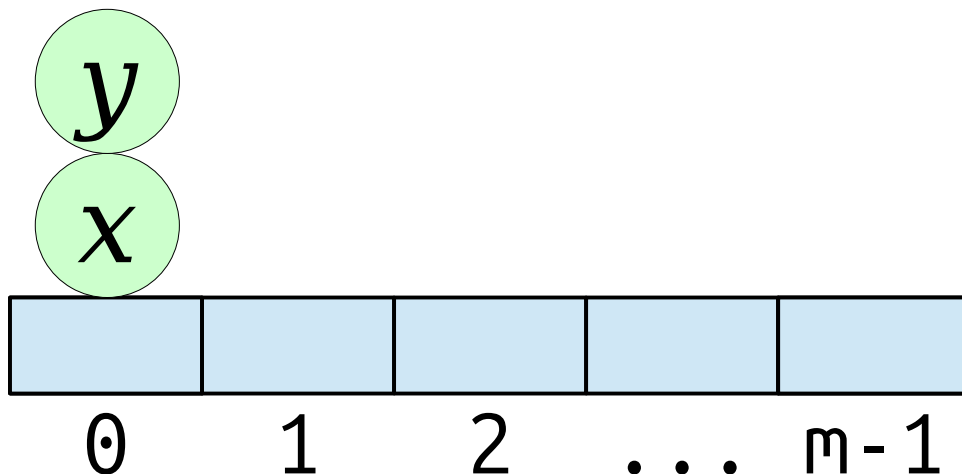
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i] \cdot \Pr[h(y) = i] \end{aligned}$$



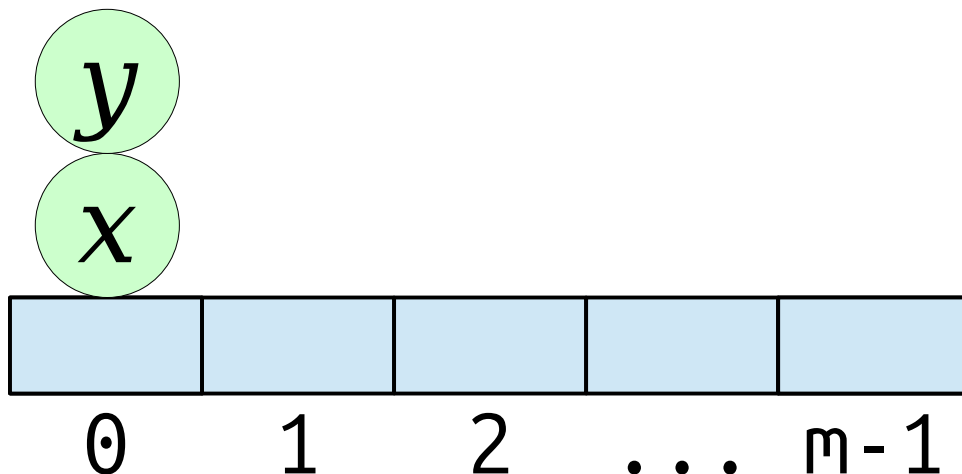
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i] \cdot \Pr[h(y) = i] \end{aligned}$$



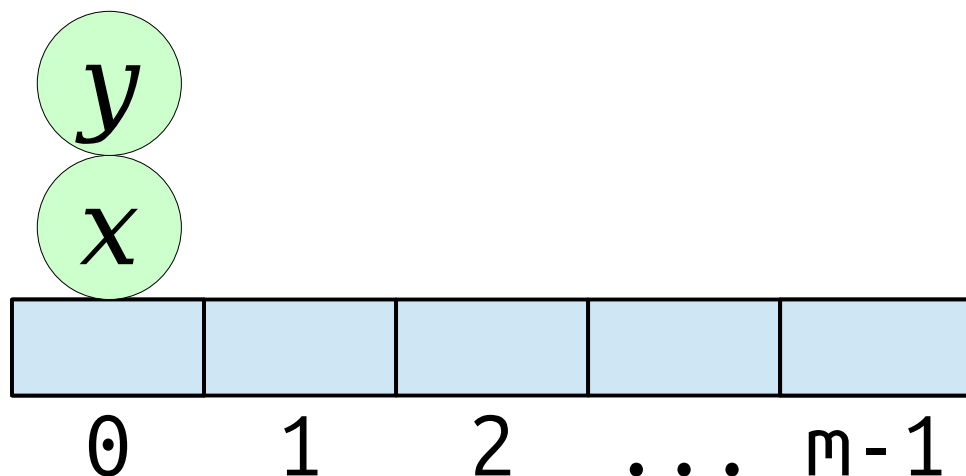
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i] \cdot \Pr[h(y) = i] \end{aligned}$$



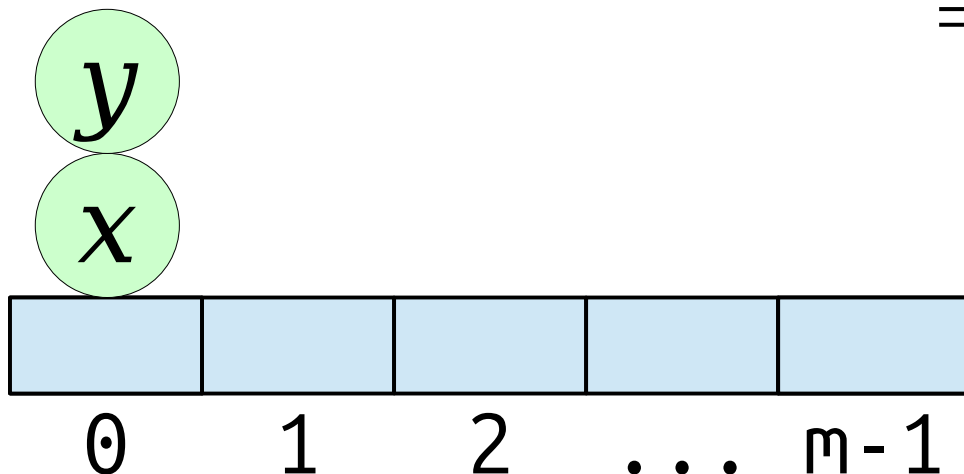
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i] \cdot \Pr[h(y) = i] \\ &= \sum_{i=0}^{m-1} \frac{1}{m^2} \end{aligned}$$



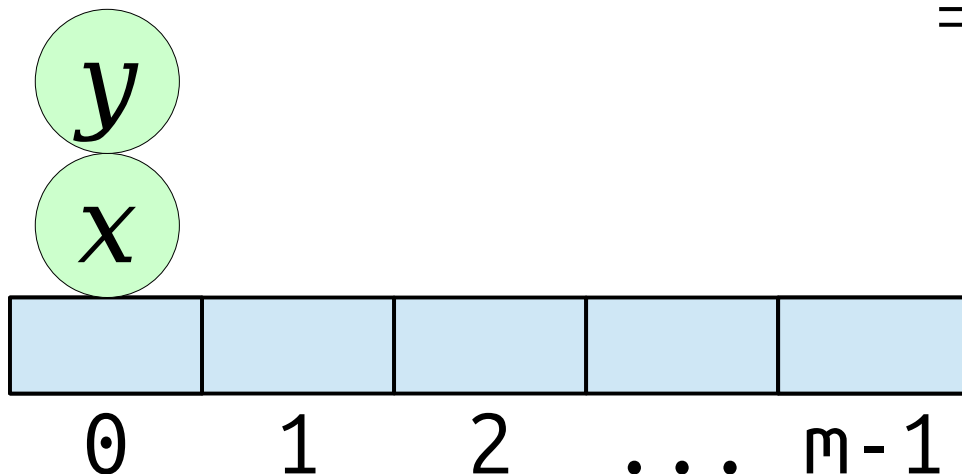
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i] \cdot \Pr[h(y) = i] \\ &= \sum_{i=0}^{m-1} \frac{1}{m^2} \end{aligned}$$



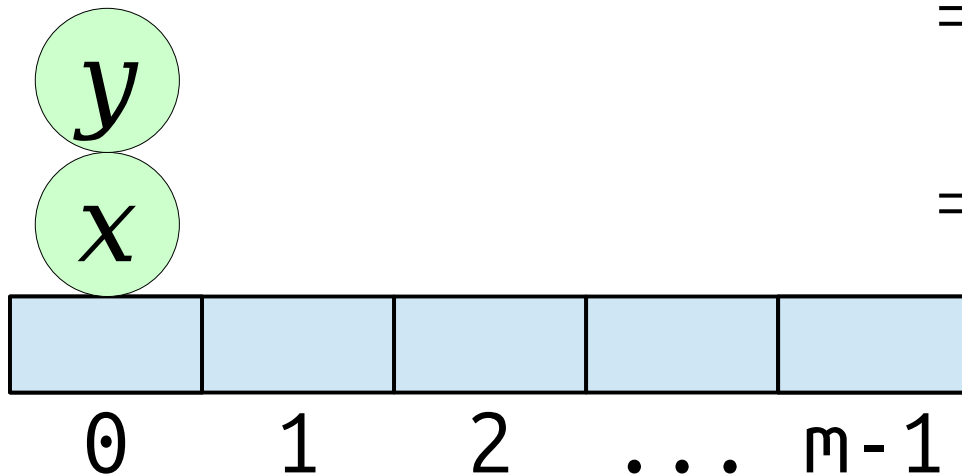
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i] \cdot \Pr[h(y) = i] \\ &= \sum_{i=0}^{m-1} \frac{1}{m^2} \\ &= \frac{1}{m} \end{aligned}$$



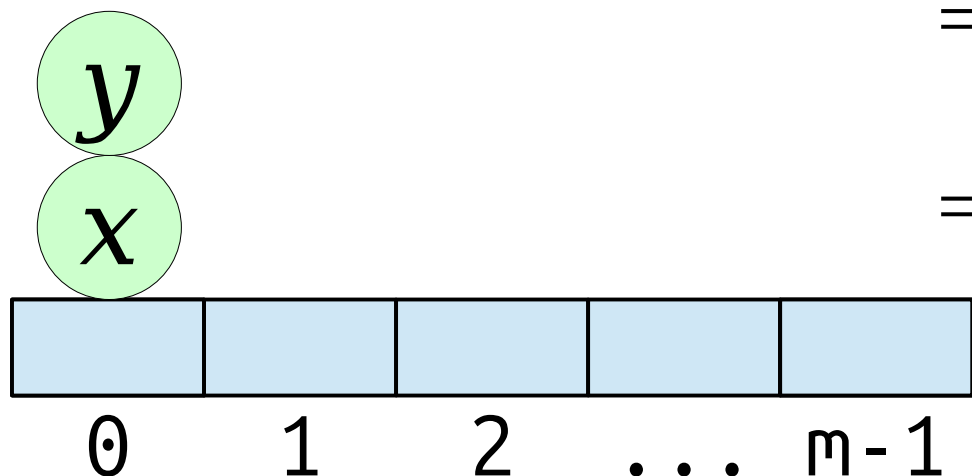
For any $x \in \mathcal{U}$ and random $h \in \mathcal{H}$, the value of $h(x)$ is uniform over its codomain.

For any distinct $x, y \in \mathcal{U}$ and random $h \in \mathcal{H}$, $h(x)$ and $h(y)$ are independent random variables.

Intuition:

2-independence means any pair of elements is unlikely to collide.

$$\begin{aligned} & \Pr[h(x) = h(y)] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=0}^{m-1} \Pr[h(x) = i] \cdot \Pr[h(y) = i] \\ &= \sum_{i=0}^{m-1} \frac{1}{m^2} \\ &= \frac{1}{m} \end{aligned}$$



This is the same as if h were a truly random function.

For more on hashing outside of Theoryland,
check out *[this Stack Exchange post](#)*.

Time-Out for Announcements!

Problem Sets

- Problem Set 0 has been graded.
 - Assignment regrades are handled on Gradescope. They open 48 hours after grades are returned and close one week after they're returned.
- Problem Set 1 was due at 1PM today.
 - Feel free to use one or two late days to extend the deadline by 24 or 48 hours.
- Problem Set 2 goes out today. It's due next Thursday at 1:00PM.
 - Play around with succinct data structures, and design your own!

ASSU Elections

- ASSU Elections run tomorrow through Thursday.
- ***Please take these seriously.***
 - The university looks to the Undergraduate Senate and Graduate Student Council when they want to speak to student representatives.
 - ASSU has the power to nominate students to important university committees.
 - ASSU has a direct say in some university governance policies.

Back to CS166!

Frequency Estimation

Frequency Estimators

- A ***frequency estimator*** is a data structure supporting the following operations:
 - ***increment***(x), which increments the number of times that x has been seen, and
 - ***estimate***(x), which returns an estimate of the frequency of x .
- This is easy to solve exactly using BSTs or hash tables, except that we need $\Omega(n)$ space simply to write down everything we've ***incremented***.
- ***Question:*** Can we solve this problem without using $\Omega(n)$ bits of space?

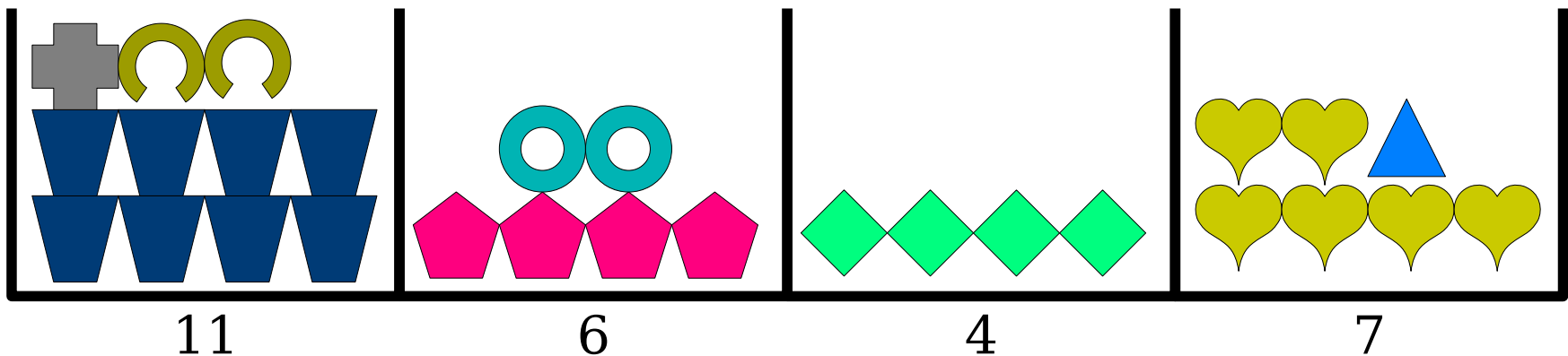
The Count-Min Sketch

How to Build an Estimator

	<i>Count-Min Sketch</i>
Step One: Build a Simple Estimator	

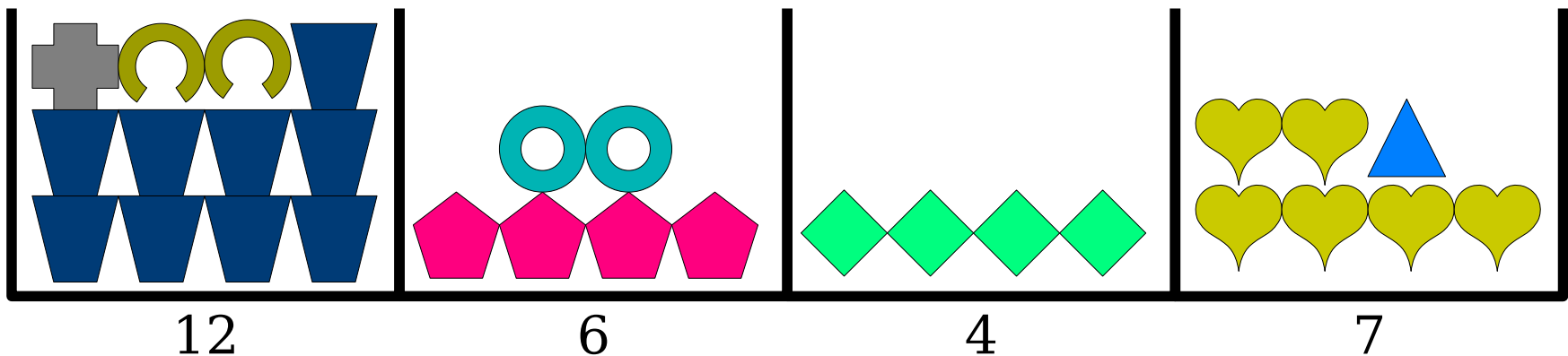
Revisiting the Exact Solution

- In the exact solution to the frequency estimation problem, we maintained a single counter for each distinct element. This is too space-inefficient.
- **Idea:** Store a fixed number of counters and assign a counter to each $x \in \mathcal{U}$. Multiple objects might be assigned to the same counter.
- To **increment**(x), increment the counter for x .
- To **estimate**(x), read the value of the counter for x .



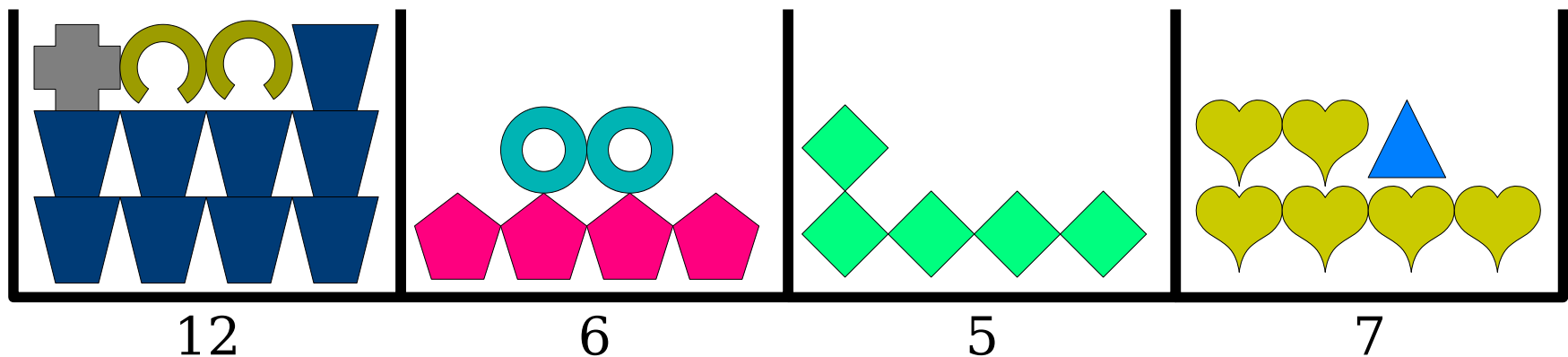
Revisiting the Exact Solution

- In the exact solution to the frequency estimation problem, we maintained a single counter for each distinct element. This is too space-inefficient.
- **Idea:** Store a fixed number of counters and assign a counter to each $x \in \mathcal{U}$. Multiple objects might be assigned to the same counter.
- To **increment**(x), increment the counter for x .
- To **estimate**(x), read the value of the counter for x .



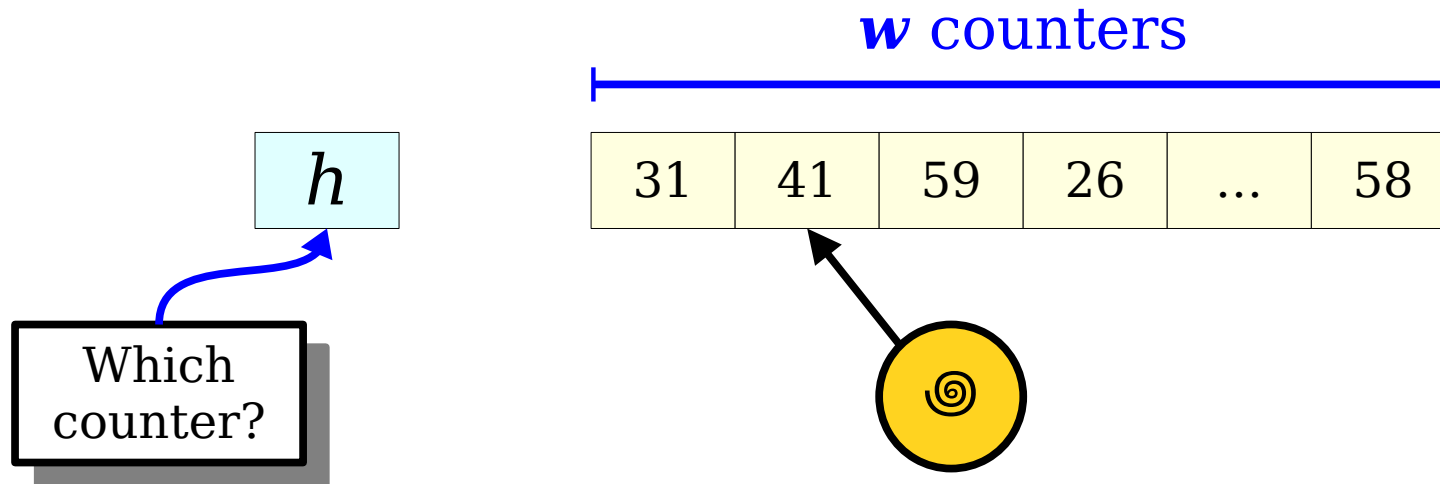
Revisiting the Exact Solution

- In the exact solution to the frequency estimation problem, we maintained a single counter for each distinct element. This is too space-inefficient.
- **Idea:** Store a fixed number of counters and assign a counter to each $x \in \mathcal{U}$. Multiple objects might be assigned to the same counter.
- To **increment**(x), increment the counter for x .
- To **estimate**(x), read the value of the counter for x .



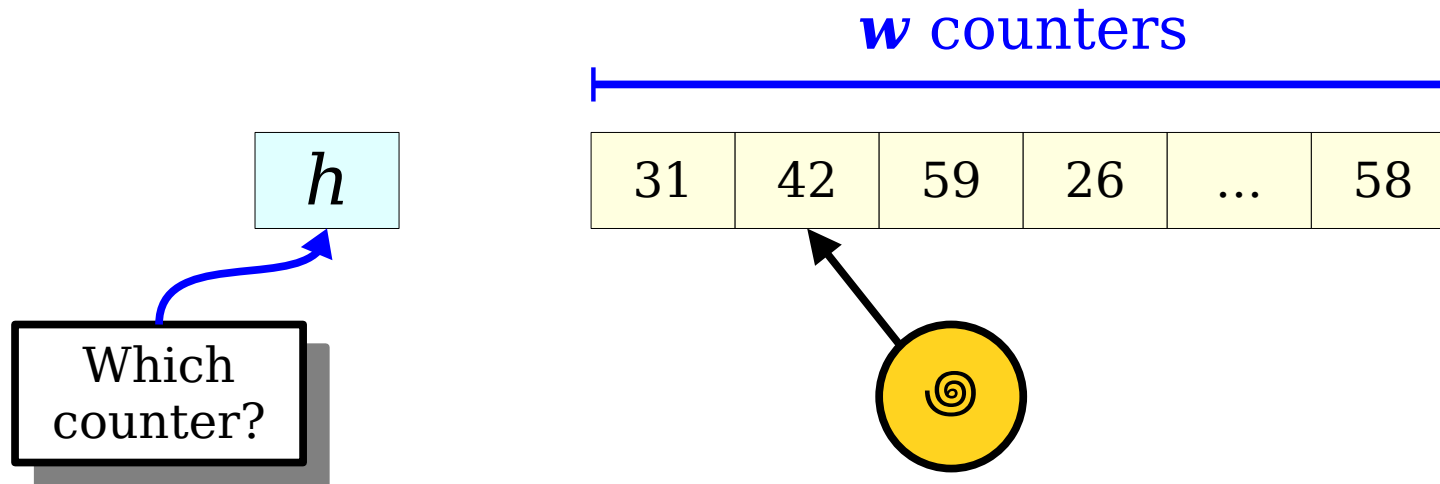
Our Initial Structure

- Create an array of counters, all initially 0, called **count**. It will have w elements for some w we choose later.
- Choose, from a family of 2-independent hash functions \mathcal{H} , a uniformly-random hash function $h : \mathcal{U} \rightarrow [w]$.
- To **increment**(x), increment **count**[$h(x)$].
- To **estimate**(x), return **count**[$h(x)$].



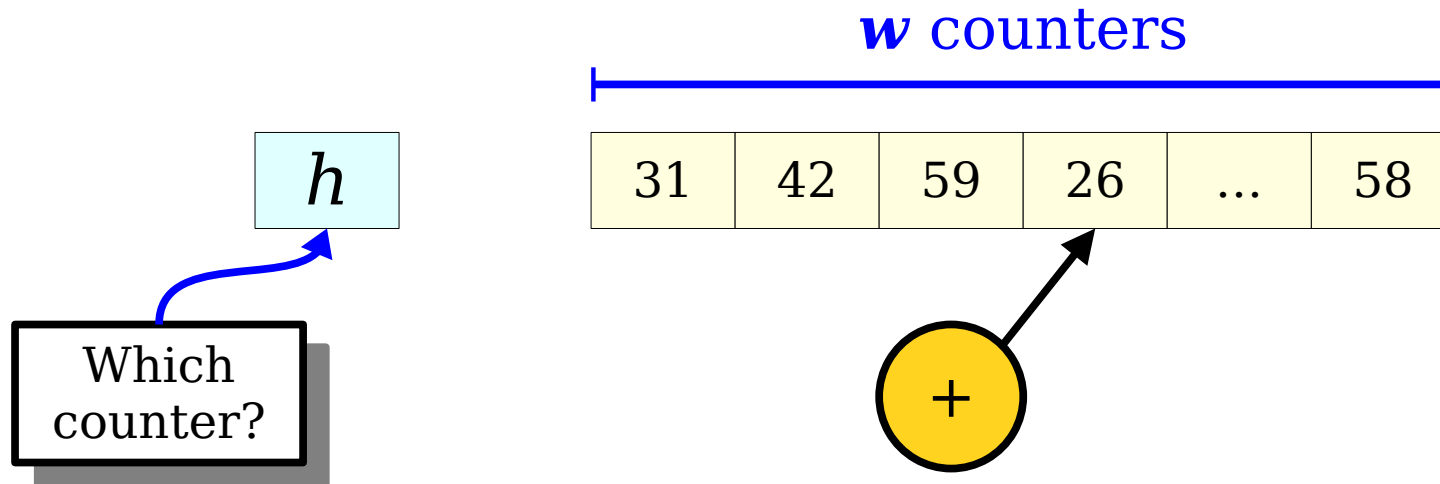
Our Initial Structure

- Create an array of counters, all initially 0, called **count**. It will have w elements for some w we choose later.
- Choose, from a family of 2-independent hash functions \mathcal{H} , a uniformly-random hash function $h : \mathcal{U} \rightarrow [w]$.
- To **increment**(x), increment **count**[$h(x)$].
- To **estimate**(x), return **count**[$h(x)$].



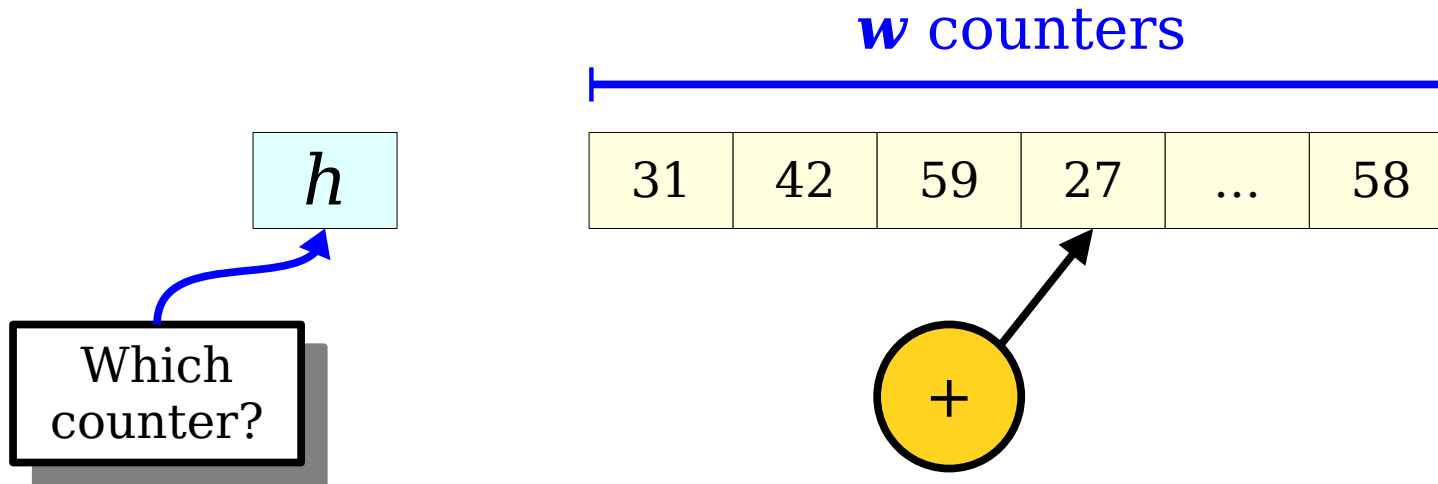
Our Initial Structure

- Create an array of counters, all initially 0, called **count**. It will have w elements for some w we choose later.
- Choose, from a family of 2-independent hash functions \mathcal{H} , a uniformly-random hash function $h : \mathcal{U} \rightarrow [w]$.
- To **increment**(x), increment **count**[$h(x)$].
- To **estimate**(x), return **count**[$h(x)$].



Our Initial Structure

- Create an array of counters, all initially 0, called **count**. It will have w elements for some w we choose later.
- Choose, from a family of 2-independent hash functions \mathcal{H} , a uniformly-random hash function $h : \mathcal{U} \rightarrow [w]$.
- To **increment**(x), increment **count**[$h(x)$].
- To **estimate**(x), return **count**[$h(x)$].



How to Build an Estimator

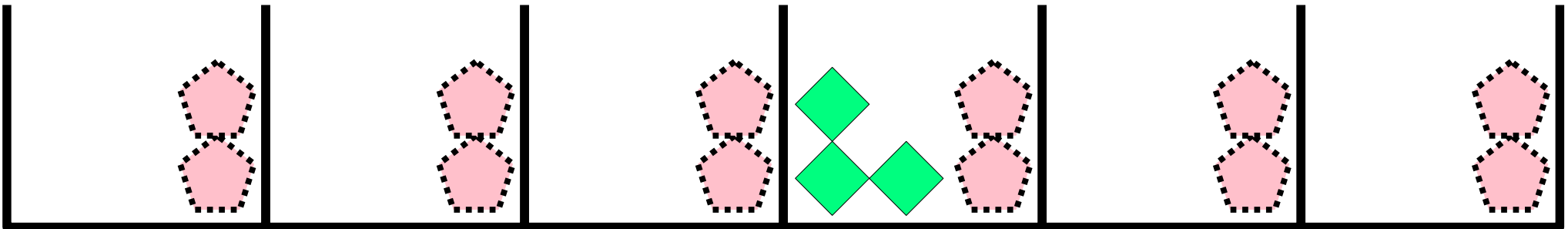
	<i>Count-Min Sketch</i>
<i>Step One:</i> Build a Simple Estimator	Hash items to counters; add +1 when item seen.
<i>Step Two:</i> Compute Expected Value of Estimator	

Some Notation

- Let x_1, x_2, x_3, \dots denote the list of distinct items whose frequencies are being stored.
- Let $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \dots$ denote the frequencies of those items.
 - e.g. \mathbf{a}_i is the true number of times x_i is seen.
- Let $\hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \hat{\mathbf{a}}_3, \dots$ denote the estimate our data structure gives for the frequency of each item.
 - e.g. $\hat{\mathbf{a}}_i$ is our estimate for how many times x_i has been seen.
 - **Important detail:** the \mathbf{a}_i values are not random variables (data are chosen adversarially), while the $\hat{\mathbf{a}}_i$ values are random variables (they depend on a randomly-sampled hash function).
- In what follows, imagine we're querying the frequency of some specific element x_i . We want to analyze $\hat{\mathbf{a}}_i$.

Analyzing our Estimator

- We're interested in learning more about $\hat{\mathbf{a}}_i$. A good first step is to work out $E[\hat{\mathbf{a}}_i]$.
- $\hat{\mathbf{a}}_i$ will be equal to \mathbf{a}_i , plus some “noise” terms from colliding elements.
- Each of those elements is very unlikely to collide with us, though. (There's a $1/w$ chance of a collision for any one other element.)
- **Reasonable guess:** $E[\hat{\mathbf{a}}_i] = \mathbf{a}_i + \sum_{j \neq i} \frac{\mathbf{a}_j}{w}$

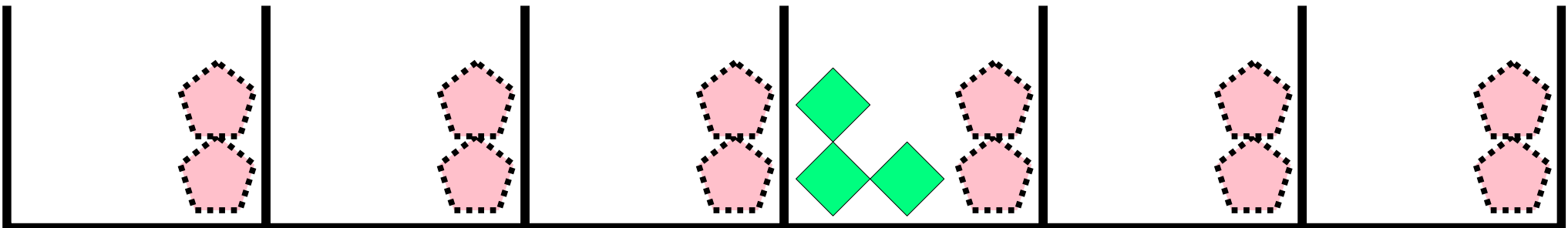


Analyzing our Estimator

- We're interested in learning more about $\hat{\mathbf{a}}_i$. A good first step is to work out $E[\hat{\mathbf{a}}_i]$.
- $\hat{\mathbf{a}}_i$ will be equal to \mathbf{a}_i , plus some “noise” terms from colliding elements.
- Each of those elements is very unlikely to collide with us, though. (There's a $1/w$ chance of a collision for any one other element.)

- **Reasonable guess:** $E[\hat{\mathbf{a}}_i] = \mathbf{a}_i + \sum_{j \neq i} \frac{\mathbf{a}_j}{w}$

Frequency of each other item, scaled to account for chance of a collision.



Making Things Formal

- Let's make this more rigorous.
- For each element x_j :
 - If $h(x_i) = h(x_j)$, then x_j contributes a_j to **count** $[h(x_i)]$.
 - If $h(x_i) \neq h(x_j)$, then x_j contributes 0 to **count** $[h(x_i)]$.

Making Things Formal

- Let's make this more rigorous.
- For each element x_j :
 - If $h(x_i) = h(x_j)$, then x_j contributes a_j to **count** $[h(x_i)]$.
 - If $h(x_i) \neq h(x_j)$, then x_j contributes 0 to **count** $[h(x_i)]$.
- To pin this down precisely, let's introduce some random variables to track collisions:

$$\mathbb{1}_{h(x_i)=h(x_j)} = \begin{cases} 1 & \text{if } h(x_i) = h(x_j) \\ 0 & \text{if } h(x_i) \neq h(x_j) \end{cases}$$

Making Things Formal

- Let's make this more rigorous.
- For each element x_j :
 - If $h(x_i) = h(x_j)$, then x_j contributes a_j to **count** $[h(x_i)]$.
 - If $h(x_i) \neq h(x_j)$, then x_j contributes 0 to **count** $[h(x_i)]$.
- To pin this down precisely, let's introduce some random variables to track collisions:

$$\mathbb{1}_{h(x_i)=h(x_j)} = \begin{cases} 1 & \text{if } h(x_i) = h(x_j) \\ 0 & \text{if } h(x_i) \neq h(x_j) \end{cases}$$

Each of these variables is called an **indicator random variable**, since it “indicates” whether some event occurs.

Making Things Formal

- Let's make this more rigorous.
- For each element x_j :
 - If $h(x_i) = h(x_j)$, then x_j contributes \mathbf{a}_j to **count** $[h(x_i)]$.
 - If $h(x_i) \neq h(x_j)$, then x_j contributes 0 to **count** $[h(x_i)]$.
- To pin this down precisely, let's introduce some random variables to track collisions:

$$\mathbb{1}_{h(x_i)=h(x_j)} = \begin{cases} 1 & \text{if } h(x_i) = h(x_j) \\ 0 & \text{if } h(x_i) \neq h(x_j) \end{cases}$$

- The value of $\hat{\mathbf{a}}_i - \mathbf{a}_i$ is then given by

$$\hat{\mathbf{a}}_i - \mathbf{a}_i = \sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(x_i)=h(x_j)}$$

$$\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] = \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]$$

$$\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] = \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]$$

$$\begin{aligned} \mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \end{aligned}$$

This follows from ***linearity of expectation***:

$$\mathbb{E}[\sum X_i] = \sum \mathbb{E}[X_i]$$

This is an extremely useful property of expected values. We'll use it extensively.

$$\begin{aligned}\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\begin{aligned} \mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \end{aligned}$$

$$\begin{aligned}\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbb{E}\left[\mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]\end{aligned}$$

The \mathbf{a}_j 's are not random variables. Randomness comes from the choice of hash function.

$$\begin{aligned}\mathbf{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbf{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\begin{aligned}\mathbf{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbf{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right]$$

$$\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)} = \begin{cases} 1 & \text{if } h(\mathbf{x}_i) = h(\mathbf{x}_j) \\ 0 & \text{if } h(\mathbf{x}_i) \neq h(\mathbf{x}_j) \end{cases}$$

$$\begin{aligned}\mathbf{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbf{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] = 1 \cdot \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] + 0 \cdot \Pr[h(\mathbf{x}_i) \neq h(\mathbf{x}_j)]$$

$$\begin{aligned}\mathbf{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbf{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\begin{aligned}\mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] &= 1 \cdot \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] + 0 \cdot \Pr[h(\mathbf{x}_i) \neq h(\mathbf{x}_j)] \\ &= \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)]\end{aligned}$$

For an event \mathcal{E} , we have $\mathbf{E}[\mathbb{1}_{\mathcal{E}}] = \Pr[\mathcal{E}]$. This is useful with linearity of expectation.

$$\begin{aligned}\mathbf{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbf{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\begin{aligned}\mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] &= 1 \cdot \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] + 0 \cdot \Pr[h(\mathbf{x}_i) \neq h(\mathbf{x}_j)] \\ &= \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)]\end{aligned}$$

$$\begin{aligned}\mathbf{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbf{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\begin{aligned}\mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] &= 1 \cdot \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] + 0 \cdot \Pr[h(\mathbf{x}_i) \neq h(\mathbf{x}_j)] \\ &= \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] \\ &= \frac{1}{w}\end{aligned}$$

$$\begin{aligned}\mathbf{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbf{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right]\end{aligned}$$

$$\begin{aligned}\mathbf{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] &= 1 \cdot \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] + 0 \cdot \Pr[h(\mathbf{x}_i) \neq h(\mathbf{x}_j)] \\ &= \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] \\ &= \frac{1}{w}\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbb{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \frac{\mathbf{a}_j}{w}\end{aligned}$$

$$\begin{aligned}\mathbb{E}\left[\mathbb{1}_{h(\mathbf{x}_i)=h(\mathbf{x}_j)}\right] &= 1 \cdot \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] + 0 \cdot \Pr[h(\mathbf{x}_i) \neq h(\mathbf{x}_j)] \\ &= \Pr[h(\mathbf{x}_i)=h(\mathbf{x}_j)] \\ &= \frac{1}{w}\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbb{E}\left[\mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \frac{\mathbf{a}_j}{w}\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\
&= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\
&= \sum_{j \neq i} \mathbf{a}_j \mathbb{E}\left[\mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\
&= \sum_{j \neq i} \frac{\mathbf{a}_j}{w}
\end{aligned}$$

Idea: Think of our element frequencies $\mathbf{a}_1, \mathbf{a}_2, \dots$ as a vector $\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \dots]$.

The total number of objects is the sum of the vector entries.

This is called the **L_1 norm** of \mathbf{a} , and is denoted $\|\mathbf{a}\|_1$:

$$\|\mathbf{a}\|_1 = \sum_i |\mathbf{a}_i|$$

$$\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] = \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]$$

$$= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]$$

$$= \sum_{j \neq i} \mathbf{a}_j \mathbb{E}\left[\mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right]$$

$$= \sum_{j \neq i} \frac{\mathbf{a}_j}{w}$$

$$\leq \frac{\|\mathbf{a}\|_1}{w}$$

Idea: Think of our element frequencies $\mathbf{a}_1, \mathbf{a}_2, \dots$ as a vector

$$\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \dots].$$

The total number of objects is the sum of the vector entries.

This is called the **L_1 norm** of \mathbf{a} , and is denoted $\|\mathbf{a}\|_1$:

$$\|\mathbf{a}\|_1 = \sum_i |\mathbf{a}_i|$$

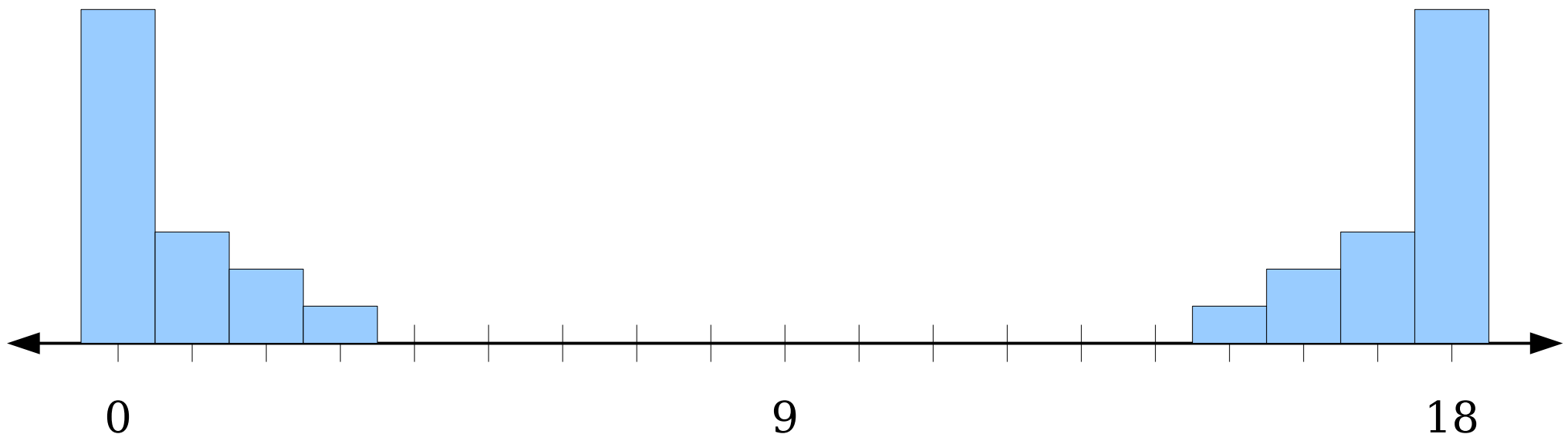
$$\begin{aligned}\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] &= \mathbb{E}\left[\sum_{j \neq i} \mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbb{E}\left[\mathbf{a}_j \mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \mathbf{a}_j \mathbb{E}\left[\mathbb{1}_{h(\mathbf{x}_i) = h(\mathbf{x}_j)}\right] \\ &= \sum_{j \neq i} \frac{\mathbf{a}_j}{w} \\ &\leq \frac{\|\mathbf{a}\|_1}{w}\end{aligned}$$

How to Build an Estimator

	<i>Count-Min Sketch</i>
<i>Step One:</i> Build a Simple Estimator	Hash items to counters; add +1 when item seen.
<i>Step Two:</i> Compute Expected Value of Estimator	Sum of indicators; 2-independent hashes have low collision rate.
<i>Step Three:</i> Apply Concentration Inequality	

On Expected Values

- We know that $E[\hat{\mathbf{a}}_i - \mathbf{a}_i] \leq \|\mathbf{a}\|_1 / w$. This means that the expected overestimate is low.
- **Claim:** This fact, in isolation, is not very useful.
- Below is a probability distribution for a random variable whose expected value is 9 that never takes values near 9.
- If this is the sort of distribution we get for $\hat{\mathbf{a}}_i$, then our estimator is not very useful!



On Expected Values

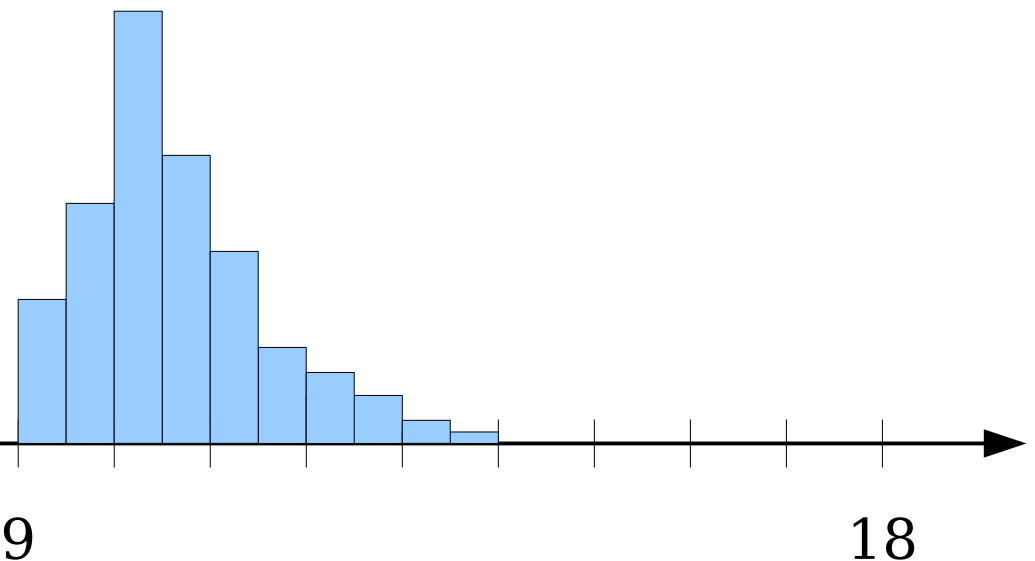
- We're looking for a way to say something like the following:

“Not only is our estimate's expected value pretty close to the real value, our estimate has a high probability of being close to the real value.”

- In other words, if the true frequency is 9, we want the distribution of our estimate to kinda sorta look like this:

If the true frequency is 9, why isn't there any probability mass below 9?

Answer at
<https://cs166.stanford.edu/pollev>

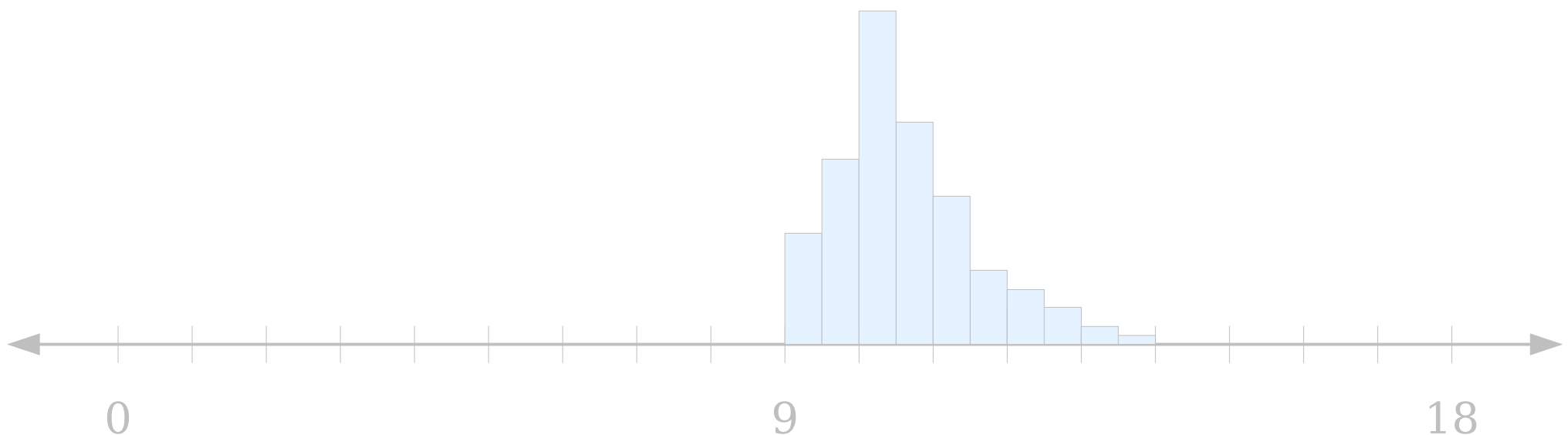


On Expected Values

- We're looking for a way to say something like the following:

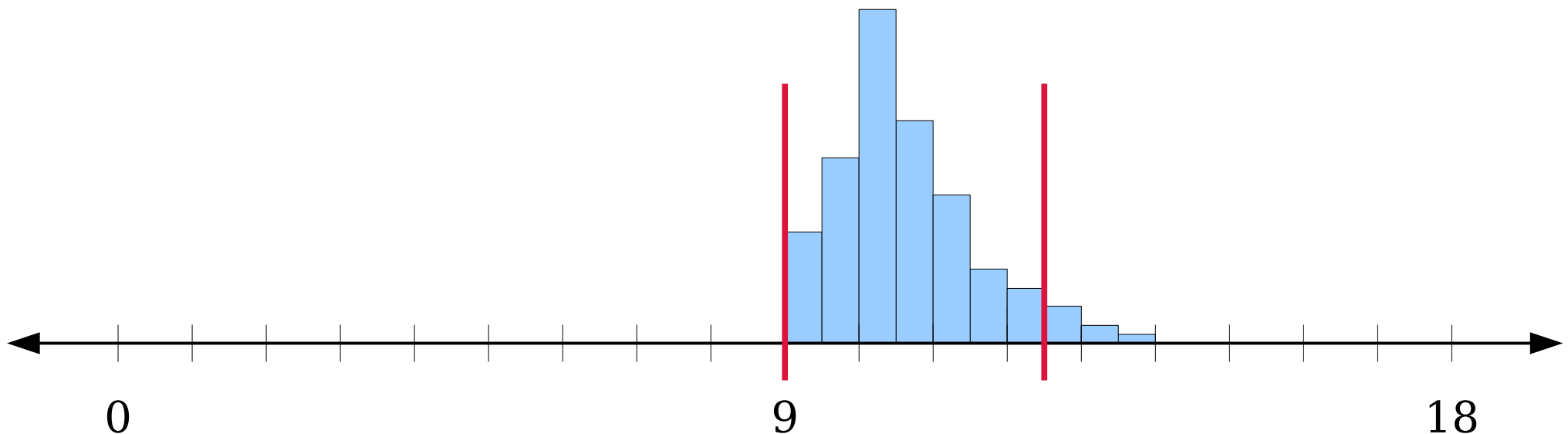
*“Not only is our estimate's expected value pretty close to the real value, our estimate has a high probability of being **close to the real value.**”*

- In other words, if the true frequency is 9, we want the distribution of our estimate to kinda sorta look like this:



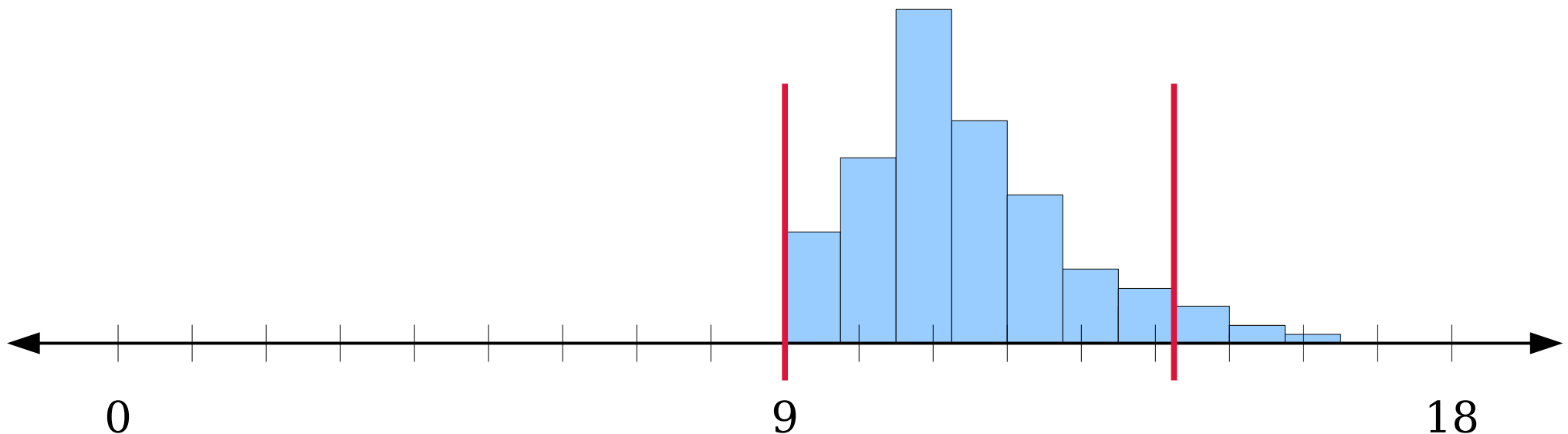
How Close is Close?

- In some applications, we might be okay overshooting by a larger amount (e.g. roughly estimating which restaurants people are visiting).
- In others, it's really bad if we overestimate by too much (e.g. polling for an election).
- **Idea:** Allow the client of the estimator to pick some value ϵ between 0 and 1 indicating how close they want to be to the true value. The closer ϵ is to 0, the better the approximation we want.



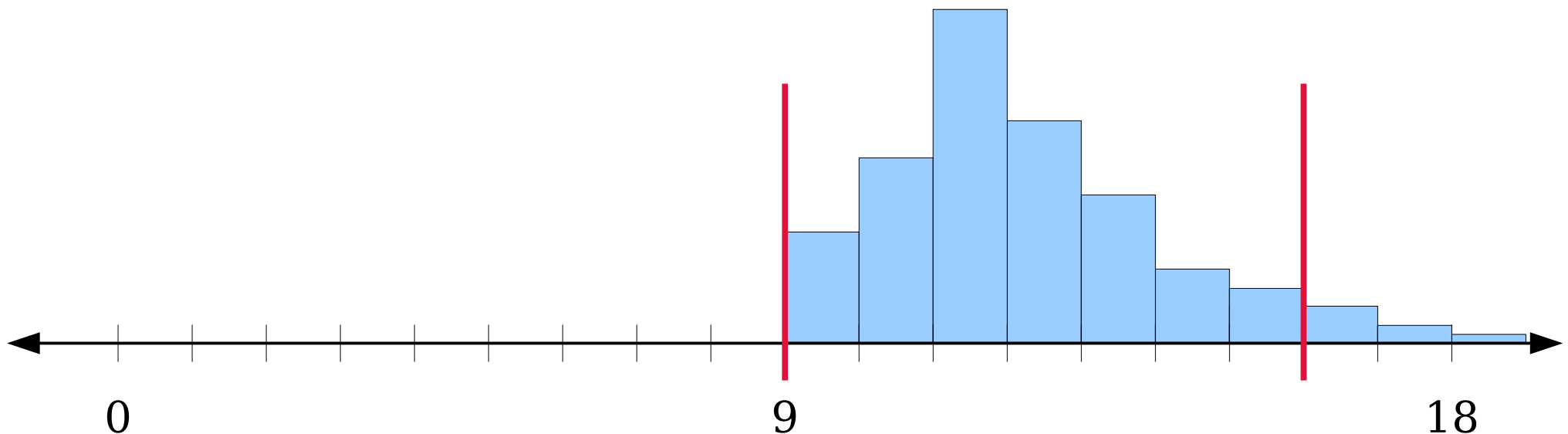
How Close is Close?

- In some applications, we might be okay overshooting by a larger amount (e.g. roughly estimating which restaurants people are visiting).
- In others, it's really bad if we overestimate by too much (e.g. polling for an election).
- **Idea:** Allow the client of the estimator to pick some value ϵ between 0 and 1 indicating how close they want to be to the true value. The closer ϵ is to 0, the better the approximation we want.



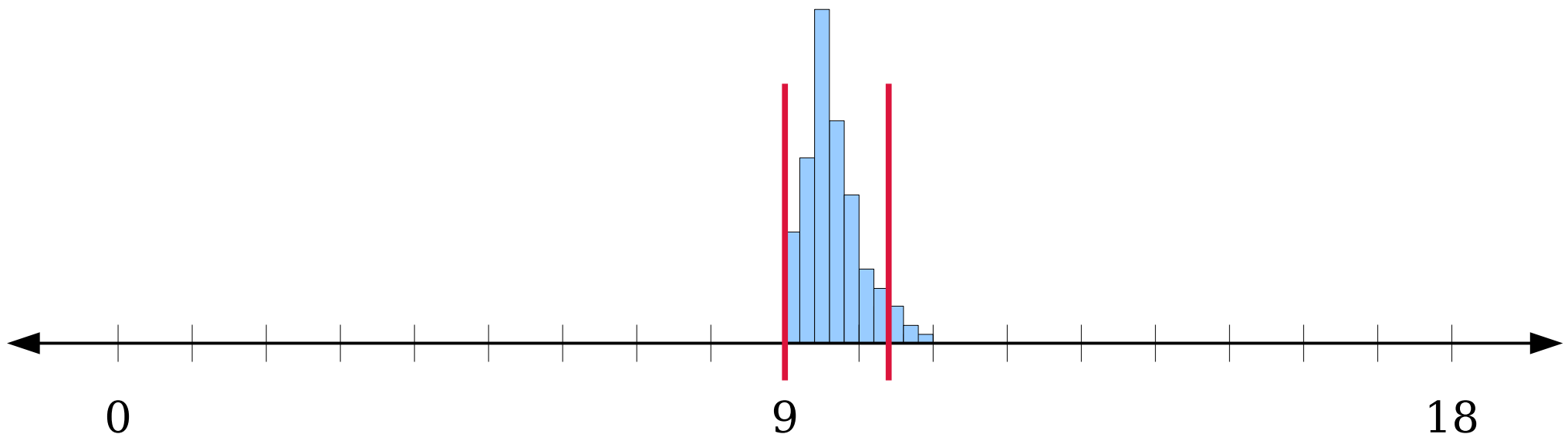
How Close is Close?

- In some applications, we might be okay overshooting by a larger amount (e.g. roughly estimating which restaurants people are visiting).
- In others, it's really bad if we overestimate by too much (e.g. polling for an election).
- **Idea:** Allow the client of the estimator to pick some value ϵ between 0 and 1 indicating how close they want to be to the true value. The closer ϵ is to 0, the better the approximation we want.



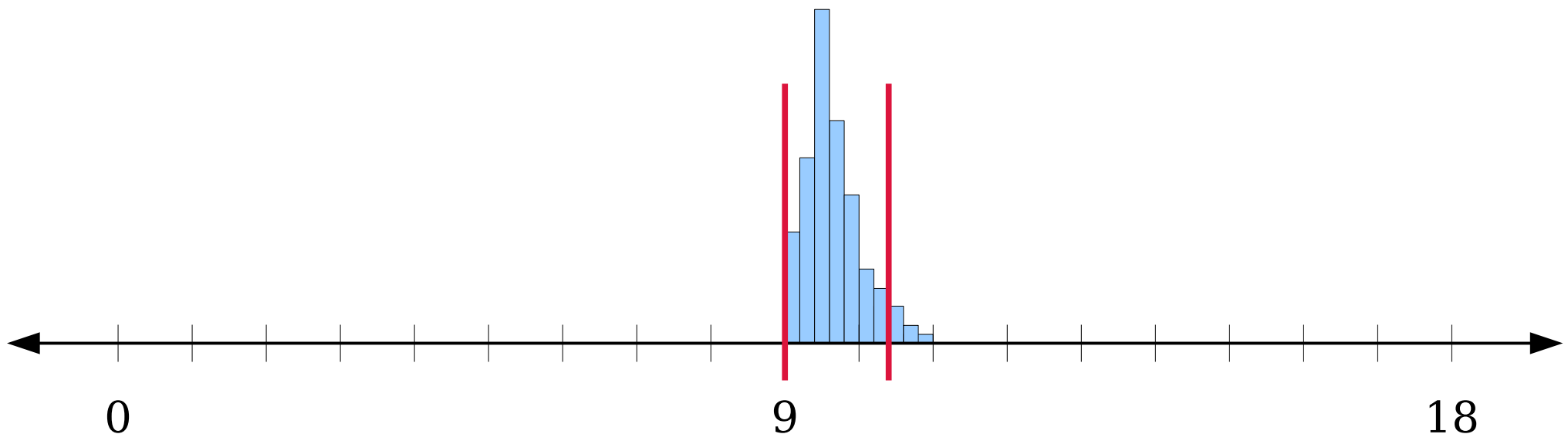
How Close is Close?

- In some applications, we might be okay overshooting by a larger amount (e.g. roughly estimating which restaurants people are visiting).
- In others, it's really bad if we overestimate by too much (e.g. polling for an election).
- **Idea:** Allow the client of the estimator to pick some value ϵ between 0 and 1 indicating how close they want to be to the true value. The closer ϵ is to 0, the better the approximation we want.



How Close is Close?

- Our overestimate is related to $\|\mathbf{a}\|_1$.
- We'll formalize how ε works as follows: we'll say we're okay with any estimate that's within $\varepsilon\|\mathbf{a}\|_1$ of the true value.
- This is okay for high-frequency elements, but not so great for low-frequency elements. (*Why?*)
- But that's okay. In practice, we are most interested in finding the high-frequency items.



Making Things Formal

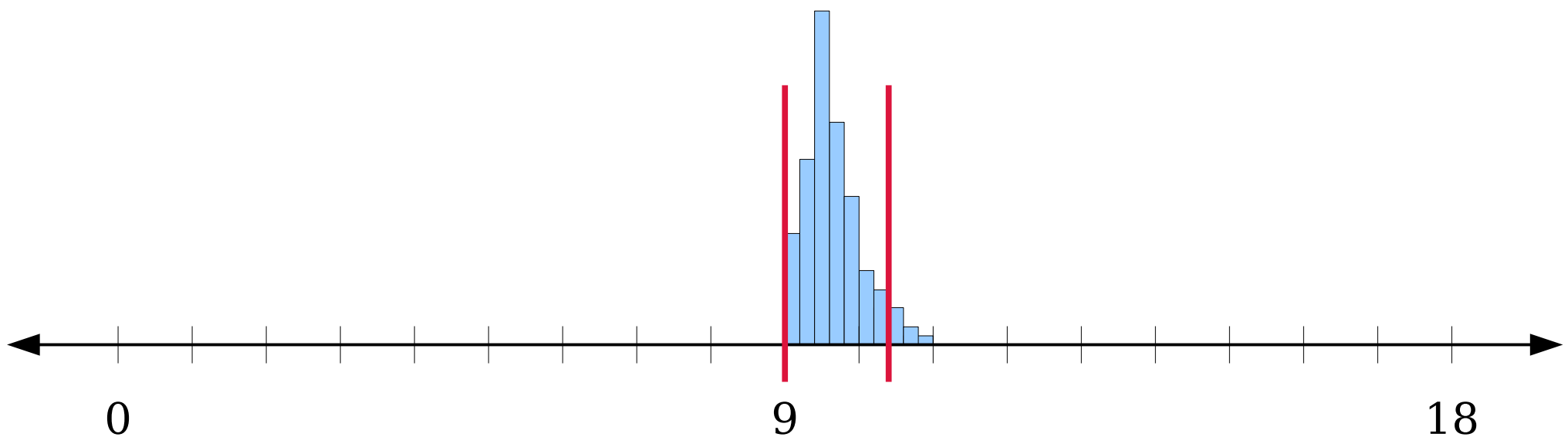
- We know that

$$\mathbb{E}[\hat{\mathbf{a}}_i - \mathbf{a}_i] \leq \frac{\|\mathbf{a}\|_1}{w}$$

- We want to bound this quantity:

$$\Pr[\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

- Let's run the numbers!



$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

We don't know the exact distribution of this random variable.

However, we have a **one-sided error**: our estimate can never be lower than the true value. This means that $\hat{\mathbf{a}}_i - \mathbf{a}_i \geq 0$.

Markov's inequality says that if X is a nonnegative random variable, then

$$\Pr[X \geq c] \leq \frac{\mathbb{E}[X]}{c}.$$

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq \frac{\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i]}{\varepsilon \|\mathbf{a}\|_1}$$

We don't know the exact distribution of this random variable.

However, we have a **one-sided error**: our estimate can never be lower than the true value. This means that $\hat{\mathbf{a}}_i - \mathbf{a}_i \geq 0$.

Markov's inequality says that if X is a nonnegative random variable, then

$$\Pr[X \geq c] \leq \frac{\mathbb{E}[X]}{c}.$$

$$\begin{aligned} & \Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\ \leq & \frac{\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i]}{\varepsilon \|\mathbf{a}\|_1} \end{aligned}$$

$$\begin{aligned} & \Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\ \leq & \frac{\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i]}{\varepsilon \|\mathbf{a}\|_1} \end{aligned}$$

$$\begin{aligned} & \Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\ & \leq \frac{\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i]}{\varepsilon \|\mathbf{a}\|_1} \end{aligned}$$

$$\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i] \leq \frac{\|\mathbf{a}\|_1}{w}$$

$$\begin{aligned} & \Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\ & \leq \frac{\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i]}{\varepsilon \|\mathbf{a}\|_1} \\ & \leq \frac{\|\mathbf{a}\|_1}{w} \cdot \frac{1}{\varepsilon \|\mathbf{a}\|_1} \end{aligned}$$

$$\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i] \leq \frac{\|\mathbf{a}\|_1}{w}$$

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

$$\leq \frac{\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i]}{\varepsilon \|\mathbf{a}\|_1}$$

$$\leq \frac{\|\mathbf{a}\|_1}{w} \cdot \frac{1}{\varepsilon \|\mathbf{a}\|_1}$$

$$\begin{aligned} & \Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\ & \leq \frac{\mathbb{E} [\hat{\mathbf{a}}_i - \mathbf{a}_i]}{\varepsilon \|\mathbf{a}\|_1} \\ & \leq \frac{\|\mathbf{a}\|_1}{w} \cdot \frac{1}{\varepsilon \|\mathbf{a}\|_1} \\ & = \frac{1}{\varepsilon w} \end{aligned}$$

Interpreting this Result

- Here's what we just proved:

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq \frac{1}{\varepsilon w}$$

- What does this tell us?
 - Increasing w decreases the chance of an overestimate. Decreasing w increases the chance of an overestimate.
 - As the user decreases ε , we have to proportionally increase w for this bound to tell us anything useful.
- **Idea:** Choose $w = e \cdot \varepsilon^{-1}$.
 - The choice of e is “somewhat” arbitrary in that any constant will work – but I peeked ahead and there’s a good reason to choose e here. 😊

Interpreting this Result

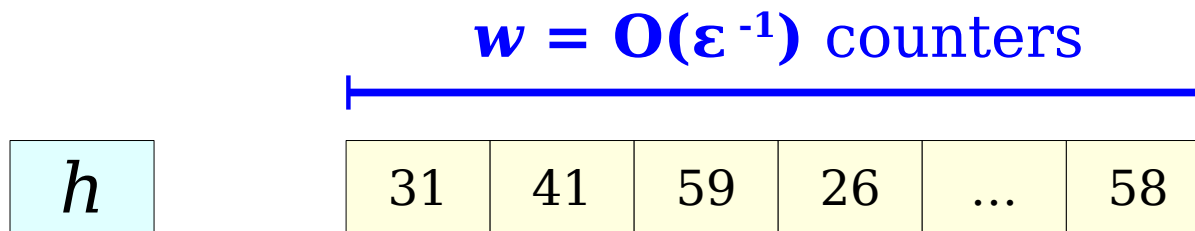
- Here's what we just proved:

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq e^{-1}$$

- What does this tell us?
 - Increasing w decreases the chance of an overestimate. Decreasing w increases the chance of an overestimate.
 - As the user decreases ε , we have to proportionally increase w for this bound to tell us anything useful.
- **Idea:** Choose $w = e \cdot \varepsilon^{-1}$.
 - The choice of e is “somewhat” arbitrary in that any constant will work – but I peeked ahead and there's a good reason to choose e here. 😊

The Story So Far

- The user chooses a value $\varepsilon \in (0, 1)$. We pick $w = e \cdot \varepsilon^{-1}$.
- Create an array **count** of w counters, each initially zero.
- Choose, from a family of 2-independent hash functions \mathcal{H} , a uniformly-random hash function $h : \mathcal{U} \rightarrow [w]$.
- To **increment**(x), increment **count**[$h(x)$].
- To **estimate**(x), return **count**[$h(x)$].
- With probability at least $1 - 1/e$, the estimate for the frequency of item x_i is within $\varepsilon \cdot \|\mathbf{a}\|_1$ of the true frequency.



How to Build an Estimator

	<i>Count-Min Sketch</i>
<i>Step One:</i> Build a Simple Estimator	Hash items to counters; add +1 when item seen.
<i>Step Two:</i> Compute Expected Value of Estimator	Sum of indicators; 2-independent hashes have low collision rate.
<i>Step Three:</i> Apply Concentration Inequality	One-sided error; use expected value and Markov's inequality.
<i>Step Four:</i> Boost Confidence	

The Story So Far

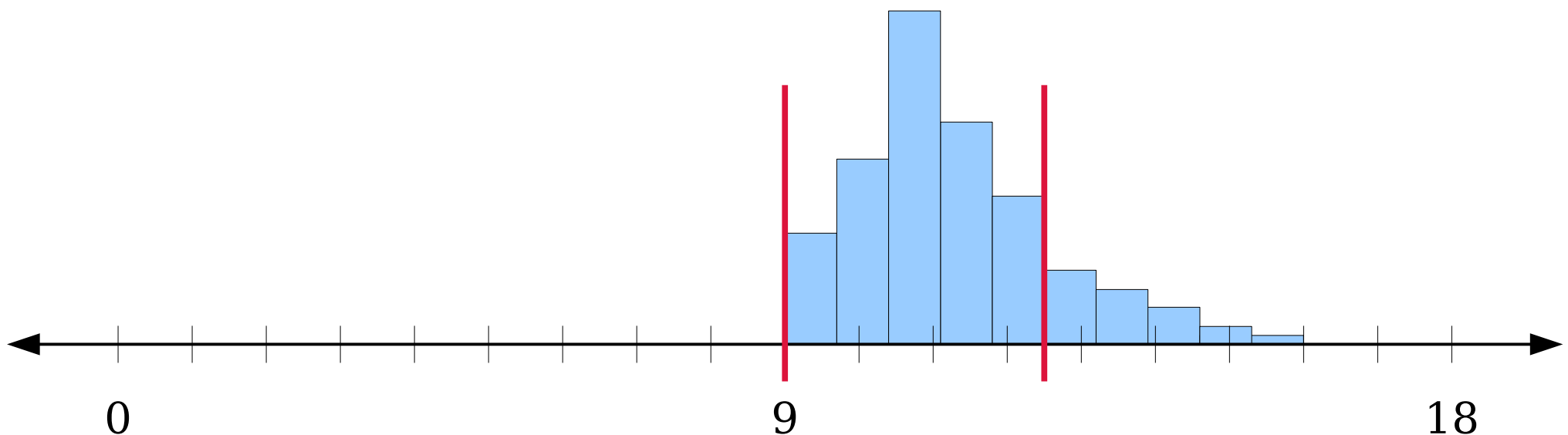
- We now have a simple estimator where

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq e^{-1}$$

- This means we have a decent chance of getting an estimate we're happy with.
- **Problem:** We want to be more confident than this.
 - In some applications, maybe it's okay to have a 63% success rate.
 - In others (say, election polling) we'll need to be a lot more confident than this.
- **Question:** How do you define "confident enough"?

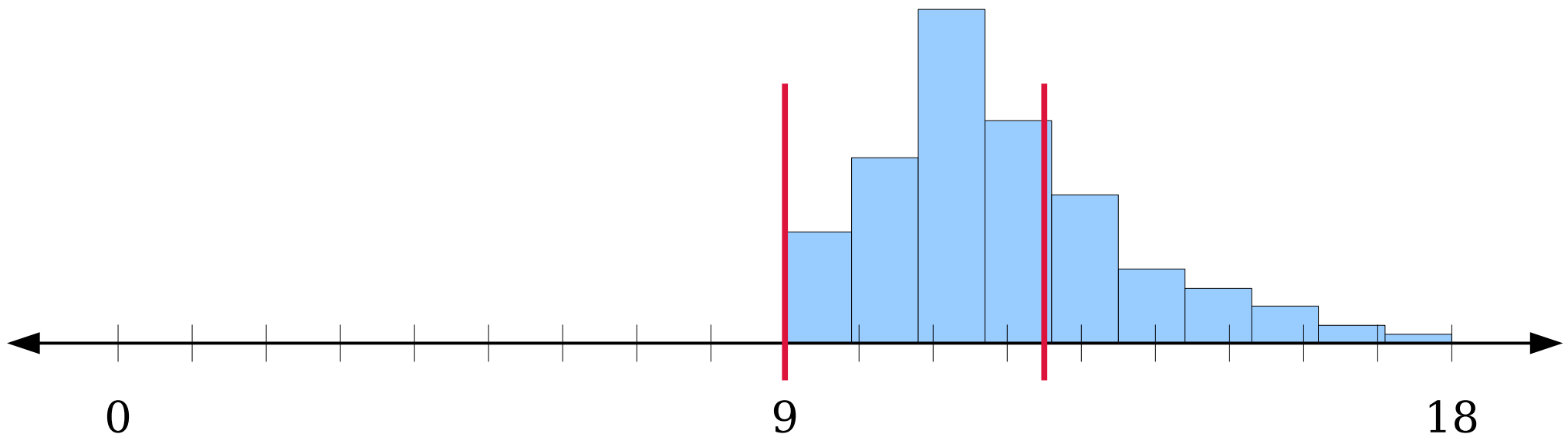
The Parameter δ

- The user already can select a parameter ε tuning the **accuracy** of the estimator: how close we want to be to the true value.
- Let's have them also select a parameter δ tuning the **confidence** of the estimator: how likely it is that we achieve this goal.
- δ ranges from 0 to 1. Lower δ means a higher chance of getting a good estimate.



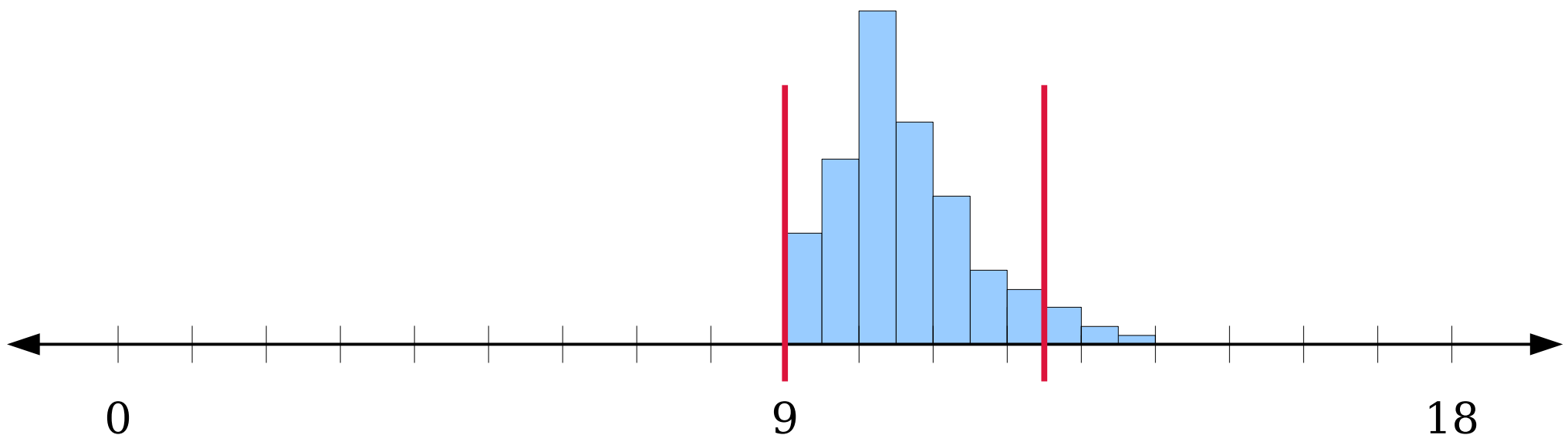
The Parameter δ

- The user already can select a parameter ε tuning the **accuracy** of the estimator: how close we want to be to the true value.
- Let's have them also select a parameter δ tuning the **confidence** of the estimator: how likely it is that we achieve this goal.
- δ ranges from 0 to 1. Lower δ means a higher chance of getting a good estimate.



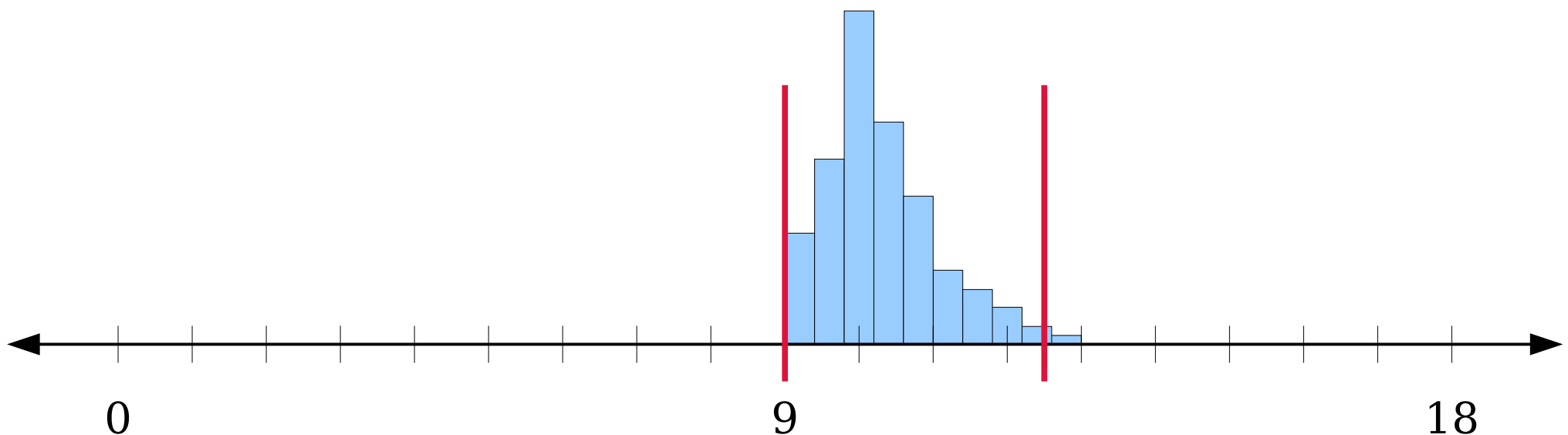
The Parameter δ

- The user already can select a parameter ε tuning the **accuracy** of the estimator: how close we want to be to the true value.
- Let's have them also select a parameter δ tuning the **confidence** of the estimator: how likely it is that we achieve this goal.
- δ ranges from 0 to 1. Lower δ means a higher chance of getting a good estimate.



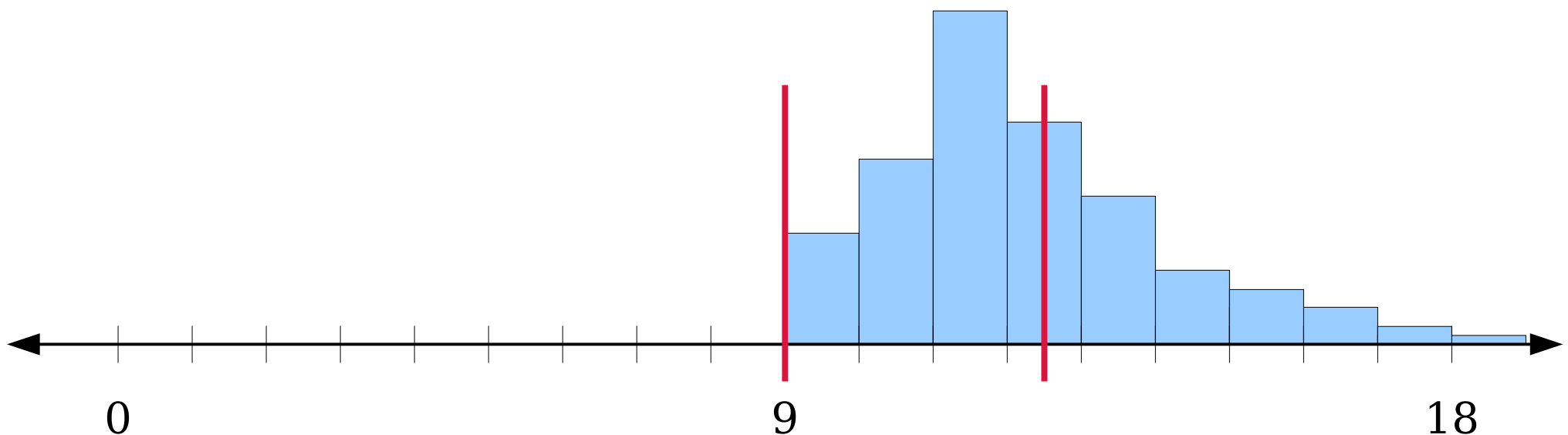
The Parameter δ

- The user already can select a parameter ε tuning the **accuracy** of the estimator: how close we want to be to the true value.
- Let's have them also select a parameter δ tuning the **confidence** of the estimator: how likely it is that we achieve this goal.
- δ ranges from 0 to 1. Lower δ means a higher chance of getting a good estimate.



The Parameter δ

- The user already can select a parameter ε tuning the **accuracy** of the estimator: how close we want to be to the true value.
- Let's have them also select a parameter δ tuning the **confidence** of the estimator: how likely it is that we achieve this goal.
- δ ranges from 0 to 1. Lower δ means a higher chance of getting a good estimate.



Our Goal

- Right now, we have this statement:

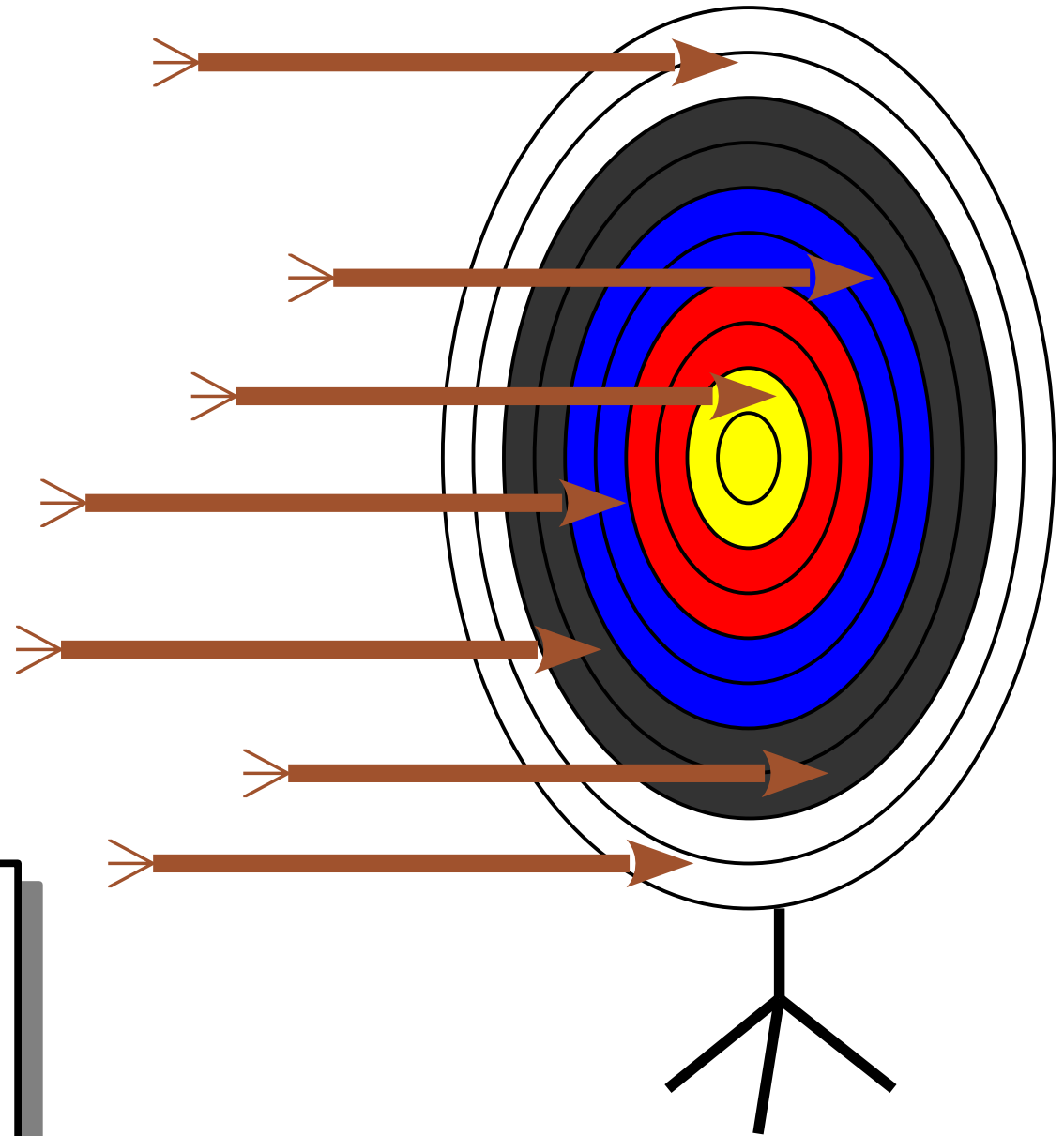
$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq e^{-1}$$

- We want to get to this one:

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq \delta$$

- How might we achieve this?

A Key Technique



It's *super unlikely* that every shot will miss the center of the target!

Running in Parallel

- Let's run d independent copies of our data structure in parallel with one another.
- Each row has its hash function sampled uniformly and independently from our hash family.
- Each time we *increment* an item, we perform the corresponding *increment* operation on each row.

$w = \lceil e \cdot \epsilon^{-1} \rceil$

h_1	31	41	59	26	53	...	58
h_2	27	18	28	18	28	...	45
h_3	16	18	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	59

$d = ??$

Running in Parallel

- Let's run d independent copies of our data structure in parallel with one another.
- Each row has its hash function sampled uniformly and independently from our hash family.
- Each time we *increment* an item, we perform the corresponding *increment* operation on each row.

$w = \lceil e \cdot \epsilon^{-1} \rceil$

h_1	31	41	59	26	53	...	58
h_2	27	18	28	18	28	...	45
h_3	16	18	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	59

$d = ??$

Running in Parallel

- Let's run d independent copies of our data structure in parallel with one another.
- Each row has its hash function sampled uniformly and independently from our hash family.
- Each time we *increment* an item, we perform the corresponding *increment* operation on each row.

$w = \lceil e \cdot \epsilon^{-1} \rceil$

h_1	32	41	59	26	53	...	58
h_2	27	18	29	18	28	...	45
h_3	16	18	3	40	88	...	75
...	...						
h_d	69	31	47	18	5	...	60

$d = ??$

Running in Parallel

- Let's run d independent copies of our data structure in parallel with one another.
- Each row has its hash function sampled uniformly and independently from our hash family.
- Each time we *increment* an item, we perform the corresponding *increment* operation on each row.

$w = \lceil e \cdot \epsilon^{-1} \rceil$

h_1	32	41	59	26	53	...	58
h_2	27	18	29	18	28	...	45
h_3	16	18	3	40	88	...	75
...	...						
h_d	69	31	47	18	5	...	60

$d = ??$

Running in Parallel

- Imagine we call *estimate*(x) on each of our estimators and get back these estimates.
- We need to give back a single number.
- **Question:** How should we aggregate these numbers into a single estimate?

Answer at
<https://cs166.stanford.edu/pollev>

Estimator 1:
137

Estimator 2:
271

Estimator 3:
166

Estimator 4:
103

Estimator 5:
261

Running in Parallel

- Imagine we call *estimate*(x) on each of our estimators and get back these estimates.
- We need to give back a single number.
- **Question:** How should we aggregate these numbers into a single estimate?

Estimator 1:
137

Estimator 2:
271

Estimator 3:
166

Estimator 4:
103

Estimator 5:
261

Running in Parallel

- Imagine we call *estimate*(x) on each of our estimators and get back these estimates.
- We need to give back a single number.
- **Question:** How should we aggregate these numbers into a single estimate?

Estimator 1:
137

Estimator 2:
271

Estimator 3:
166

Estimator 4:
103

Estimator 5:
261

Running in Parallel

- Imagine we call *estimate*(x) on each of our estimators and get back these estimates.
- We need to give back a single number.
- **Question:** How should we aggregate these estimates to give back a single estimate?

Intuition: The smallest estimate returned has the least “noise,” and that’s the best guess for the frequency.

Estimator 1:
137

Estimator 2:
271

Estimator 3:
166

Estimator 4:
103

Estimator 5:
261

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{\hat{\mathbf{a}}_{ij}\}$

$$\Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$

The only way the minimum estimate is inaccurate is if *every* estimate is inaccurate.

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$
$$= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right]$$

The only way the minimum estimate is inaccurate is if *every* estimate is inaccurate.

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$
$$= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right]$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1]$$
$$= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right]$$

Each copy of the data structure is independent of the others.

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\begin{aligned} & \Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\ &= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right] \\ &= \prod_{j=1}^d \Pr [\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \end{aligned}$$

Each copy of the data structure is independent of the others.

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\begin{aligned} & \Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\ &= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right] \\ &= \prod_{j=1}^d \Pr \left[\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right] \end{aligned}$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\begin{aligned}
& \Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\
&= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right] \\
&= \prod_{j=1}^d \Pr \left[\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right]
\end{aligned}$$

$$\Pr[\hat{\mathbf{a}}_i - \mathbf{a}_i \geq \varepsilon \|\mathbf{a}\|_1] \leq e^{-1}$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\begin{aligned}
& \Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\
&= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right] \\
&= \prod_{j=1}^d \Pr \left[\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right] \\
&\leq \prod_{j=1}^d e^{-1}
\end{aligned}$$

$$\Pr[\hat{\mathbf{a}}_i - \mathbf{a}_i \geq \varepsilon \|\mathbf{a}\|_1] \leq e^{-1}$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\begin{aligned}
& \Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\
&= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right] \\
&= \prod_{j=1}^d \Pr \left[\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right] \\
&\leq \prod_{j=1}^d e^{-1}
\end{aligned}$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

$$\begin{aligned}
& \Pr [\min \{ \hat{\mathbf{a}}_{ij} \} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\
&= \Pr \left[\bigwedge_{j=1}^d \left(\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1 \right) \right] \\
&= \prod_{j=1}^d \Pr [\hat{\mathbf{a}}_{ij} - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \\
&\leq \prod_{j=1}^d e^{-1} \\
&= e^{-d}
\end{aligned}$$

Let $\hat{\mathbf{a}}_{ij}$ be the estimate from the j th copy of the data structure.

Our final estimate is $\min \{ \hat{\mathbf{a}}_{ij} \}$

Finishing Touches

- We now see that

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq e^{-d}$$

- We want to reach this goal:

$$\Pr [\hat{\mathbf{a}}_i - \mathbf{a}_i > \varepsilon \|\mathbf{a}\|_1] \leq \delta$$

- So set $d = \ln \delta^{-1}$.

The Count-Min Sketch

$$w = \lceil e \cdot \epsilon^{-1} \rceil$$

h_1	31	41	59	26	53	...	58
h_2	27	18	28	18	28	...	45
h_3	16	18	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	59

Width: $w = \lceil e \cdot \epsilon^{-1} \rceil$

Height: $d = \lceil \ln \frac{1}{\delta} \rceil$

Sampled uniformly and independently from a 2-independent family of hash functions

The Count-Min Sketch

h_1	31	41	59	26	53	...	58
h_2	27	18	28	18	28	...	45
h_3	16	18	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	59

```
increment(x):  
  for i = 1 ... d:  
    count[i][hi(x)]++
```

The Count-Min Sketch

h_1	31	41	59	26	53	...	58
h_2	27	18	28	18	28	...	45
h_3	16	18	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	59

```
increment(x):  
  for  $i = 1 \dots d$ :  
     $\text{count}[i][h_i(x)]++$ 
```

The Count-Min Sketch

h_1	32	41	59	26	53	...	58
h_2	27	18	28	19	28	...	45
h_3	16	19	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	60

```
increment(x):  
  for  $i = 1 \dots d$ :  
     $\text{count}[i][h_i(x)]++$ 
```

The Count-Min Sketch

h_1	32	41	59	26	53	...	58
h_2	27	18	28	19	28	...	45
h_3	16	19	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	60

```
increment(x):  
  for i = 1 ... d:  
    count[i][hi(x)]++
```

The Count-Min Sketch

h_1	32	41	59	26	53	...	58
h_2	27	18	28	19	28	...	45
h_3	16	19	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	60

```
increment(x):  
  for i = 1 ... d:  
    count[i][hi(x)]++
```

```
estimate(x):  
  result = ∞  
  for i = 1 ... d:  
    result = min(result, count[i][hi(x)])  
  return result
```

The Count-Min Sketch

h_1	32	41	59	26	53	...	58
h_2	27	18	28	19	28	...	45
h_3	16	19	3	39	88	...	75
...	...						
h_d	69	31	47	18	5	...	60

```
increment(x):  
  for i = 1 ... d:  
    count[i][hi(x)]++
```

```
estimate(x):  
  result = ∞  
  for i = 1 ... d:  
    result = min(result, count[i][hi(x)])  
  return result
```

The Count-Min Sketch

- Update and query times are $\Theta(\log \delta^{-1})$.
 - That's the number of replicated copies, and we do $O(1)$ work at each.
- Space usage: $\Theta(\varepsilon^{-1} \cdot \log \delta^{-1})$ counters.
 - Each individual estimator has $\Theta(\varepsilon^{-1})$ counters, and we run $\Theta(\log \delta^{-1})$ copies in parallel.
 - How many bits do you use per counter? Depends on the particulars of your problem.
- Provides an estimate to within $\varepsilon \|\mathbf{a}\|_1$ with probability at least $1 - \delta$.
- This can be *significantly* better than just storing a raw frequency count – especially if your goal is to find items that appear very frequently.

How to Build an Estimator

	<i>Count-Min Sketch</i>
<i>Step One:</i> Build a Simple Estimator	Hash items to counters; add +1 when item seen.
<i>Step Two:</i> Compute Expected Value of Estimator	Sum of indicators; 2-independent hashes have low collision rate.
<i>Step Three:</i> Apply Concentration Inequality	One-sided error; use expected value and Markov's inequality.
<i>Step Four:</i> Replicate to Boost Confidence	Take min; only fails if all estimates are bad.

Major Ideas From Today

- ***2-independent hash families*** are useful when we want to keep collisions low.
- A “good” approximation of some quantity should have tunable ***confidence*** and ***accuracy*** parameters.
- ***Sums of indicator variables*** are useful for deriving expected values of estimators.
- ***Concentration inequalities*** like ***Markov's inequality*** are useful for showing estimators don't stay too much from their expected values.
- Good estimators can be built from ***multiple parallel copies*** of weaker estimators.

Next Time

- ***Count Sketches***
 - An alternative frequency estimator with different time/space bounds.
- ***Cardinality Estimation***
 - Estimating how many different items you've seen in a data stream.