What’s going to be different about our last two papers from what we’ve seen so far

input:

output:

runtime:

worst-case

Problem	Input	Output	Runtime
Max flow	$G = (V, E), s, t \in V, C: E \to \mathbb{R}$	Maximum flow $F: E \to \mathbb{R}$	$O(E^{1+o(1)})$ $O(VE^2)$
Flow decomposition	$G = (V, E), s, t \in V, F: E \to \mathbb{R}$	P, w decomposing flow	$O(E^2)$
Minimum flow decomposition	$G = (V, E), s, t \in V, F: E \to \mathbb{R}$	P, w decomposing flow $\|P\|$ minimized	NP-Hard
Linear programming	$\max c^t x$ $Ax \leq b$ $x \geq 0$ $x \in \mathbb{R}^d$	Feasible x^* maximizing objective (or infeasible/unbounded)	Matrix multiplication time
Integer linear programming	$\max c^t x$ $Ax \leq b$ $x \geq 0$ $x \in \mathbb{Z}^d$	Feasible x^* maximizing objective (or infeasible/unbounded)	NP-Hard

one single answer

Data structures vs. algorithms

specific input	specific input
thing that can give many outputs (query)	one output
space / memory	runtime
queries must be fast	okay (?) w/ slower runtimes
e.g. Google	e.g. oil + gas

Goals for today

data structures: hash table
bloom filter

randomness

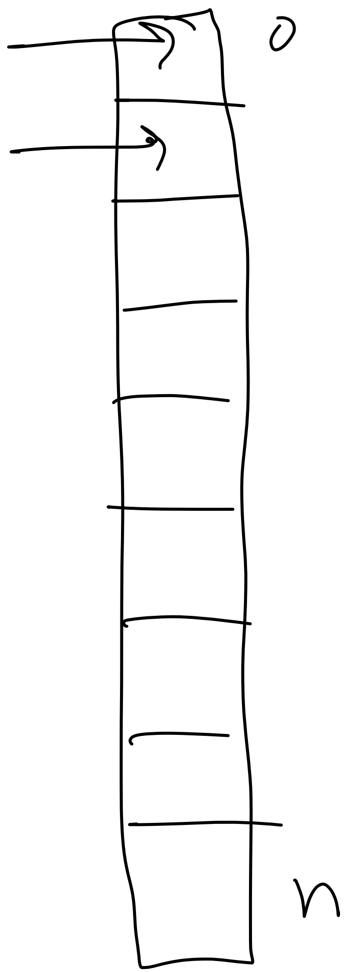
estimate vs. exact

What do you already know about hash functions and hash tables?

- data structure
- spread out inputs
- hash function maps inputs to indices

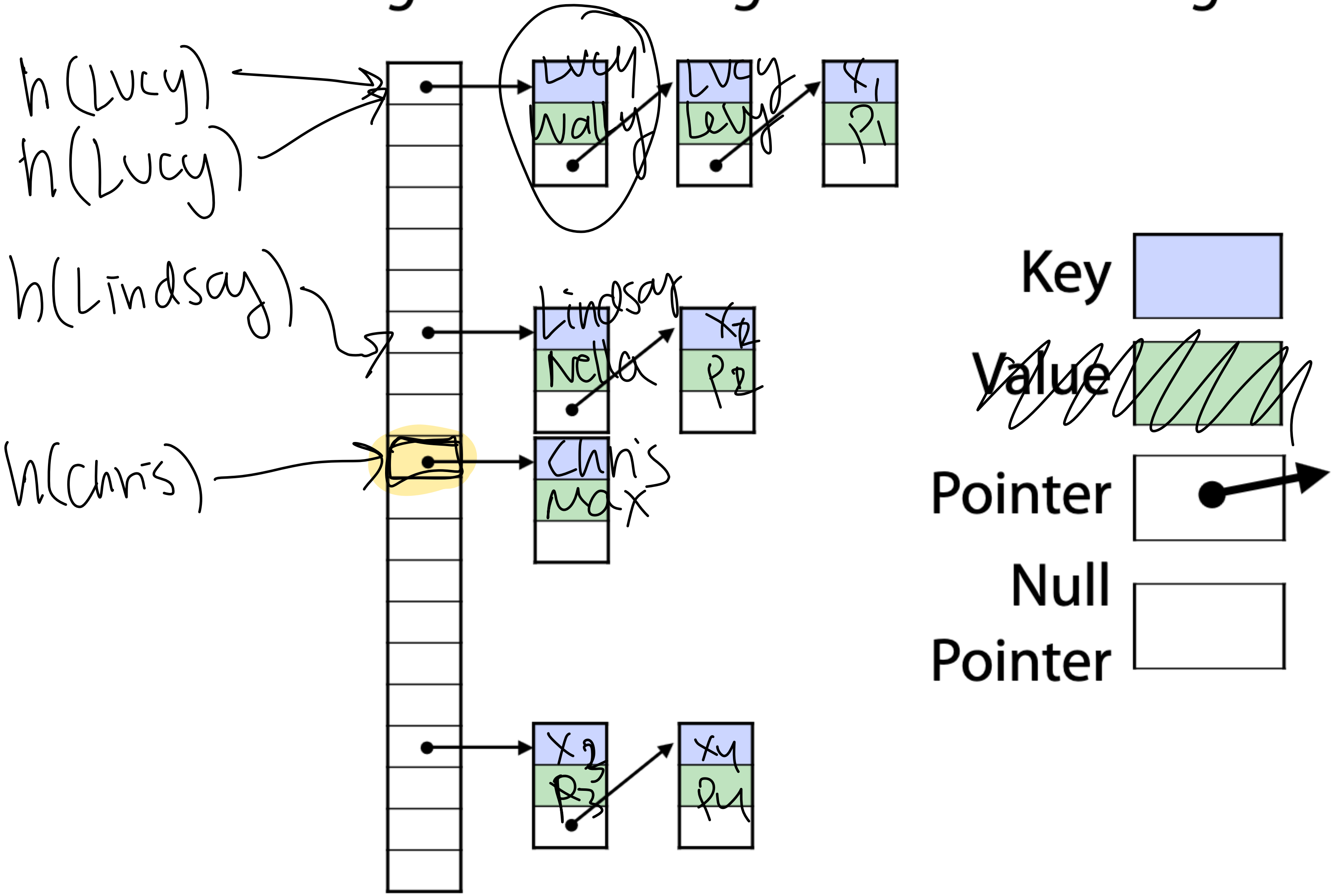
modular arithmetic
prime

- insert: $O(1)$
- query: $O(1)$



Hash Table

"Hashing with chaining" or "chain hashing"



dictionary
key : value

pet owner : pet

insert (Lucy, Wally)
insert (Lucy, Levy)

query(Chris)

h(Chris)

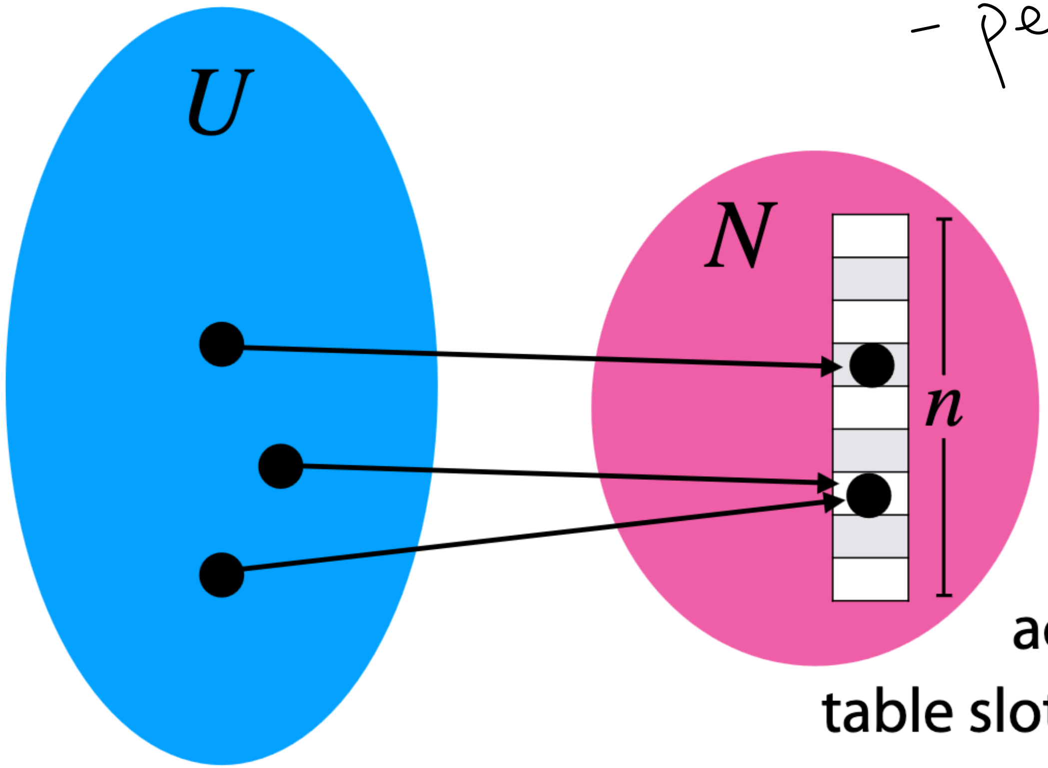
set
is x in set

(owner, pet) dictionary

What is U?

- pet owners

Hash Function

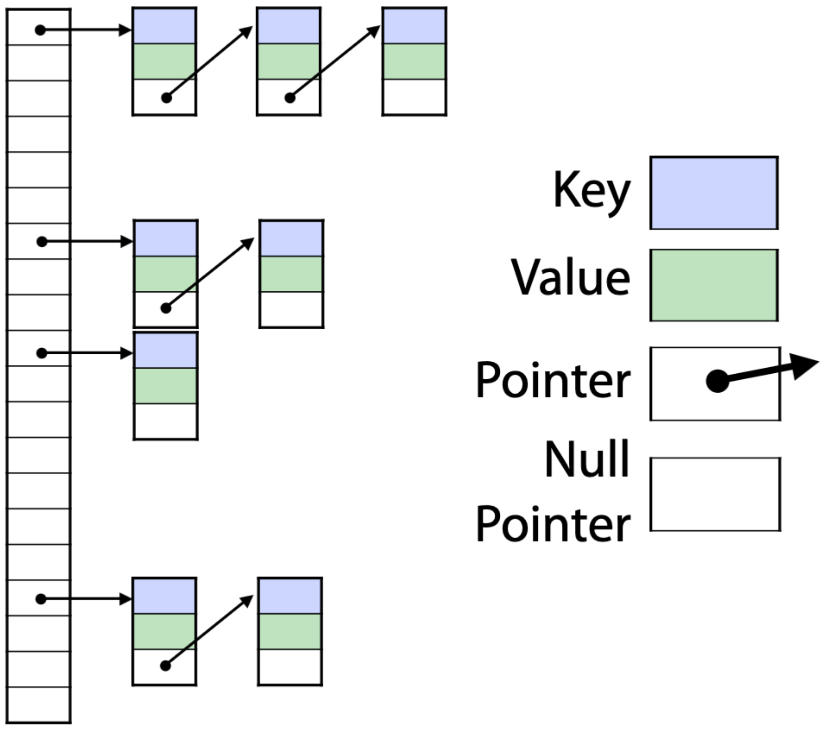


Assume accessing table slot is $O(1)$

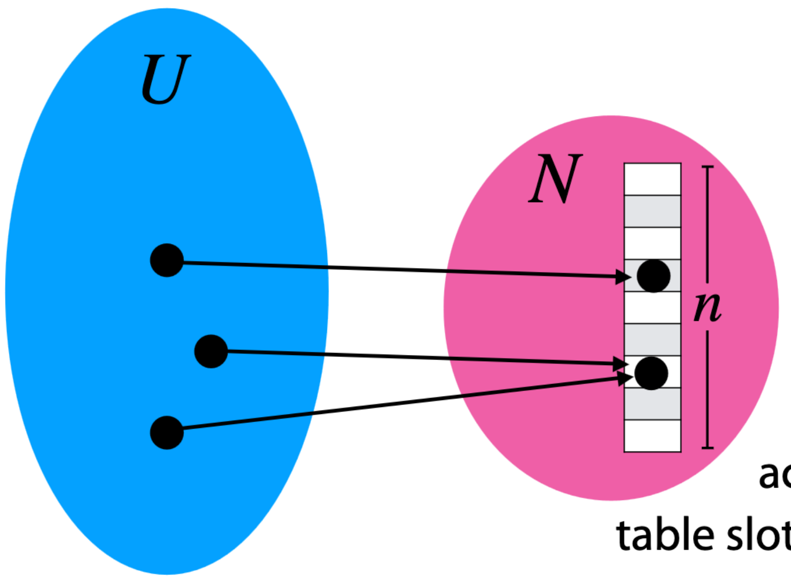
Assume hash function operates on any item from U (integers, strings, etc) and is $O(1)$ time

Hash Table

"Hashing with chaining" or *"chain hashing"*



Hash Function



Assume accessing table slot is $O(1)$

Assume hash function operates on any item from U (integers, strings, etc) and is $O(1)$ time

Hash Table

I add m items to an n-bucket hash table

Without probability, what can I say?

Question	Assumption	Statement	Comment
Does any bucket have more the one item?	$m > n$	yes	by pigeonhole principle
Is any bucket empty?	$m < n$	yes	reverse PHP
What is the average bucket occupancy?		m/n	

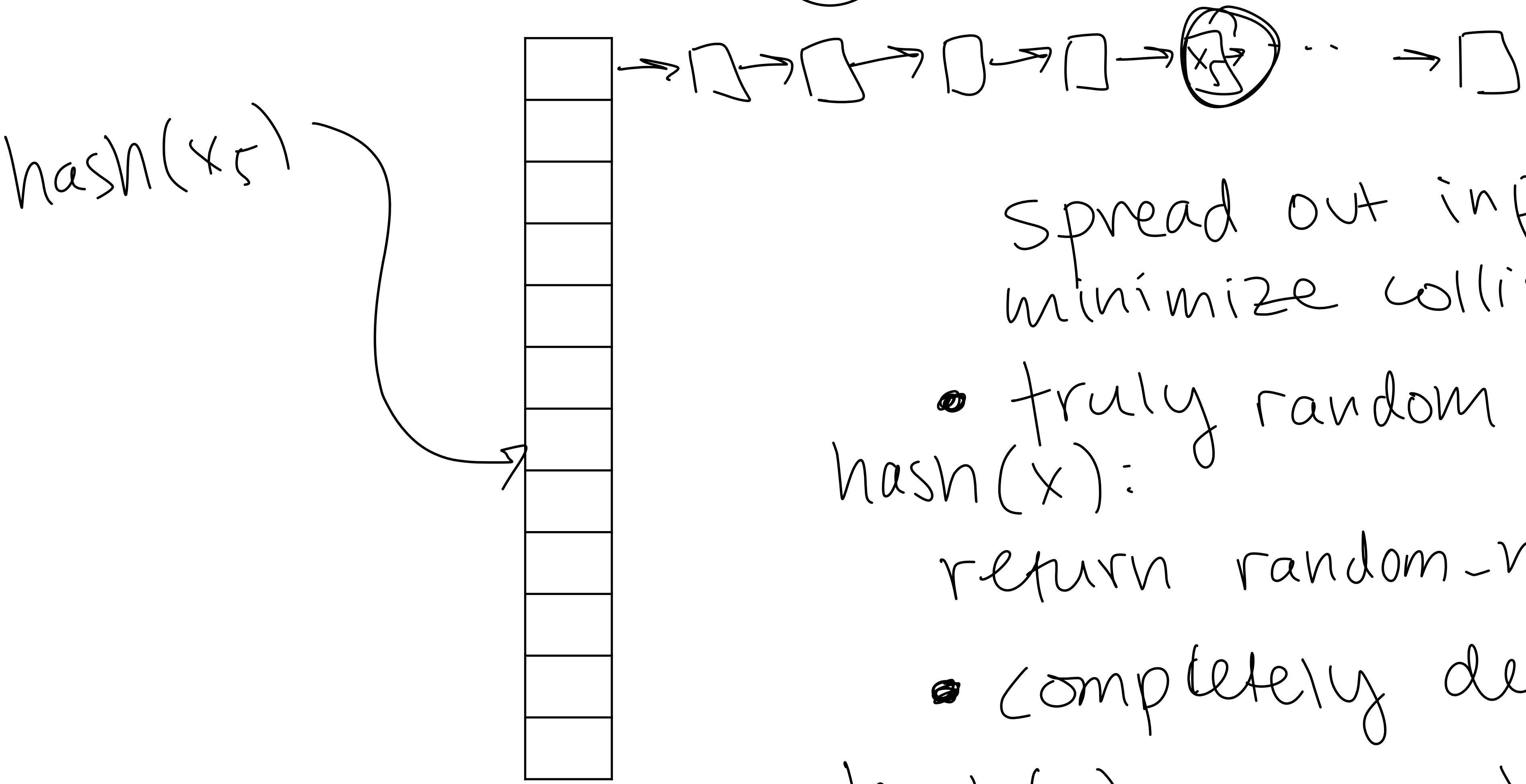
Hash Table

I add m items to an n-bucket hash table

Without probability, what can I say?

Question	Assumption	Statement	Comment
Does any bucket have more the one item?	$m > n$	Yes	Pigeonhole principle
Is any bucket empty?	$m < n$	Yes	"Empty pigeonhole" principle
What is the average bucket occupancy?	-	m/n	-

How do we get O(1) expected query time?



spread out inputs
minimize collisions

- truly random hash function

```
hash(x):
    return random_num()
```

X

- completely deterministic

```
hash(x):
    x mod n
```

X

Hash Function

salt

```
int hash(int x) {
    int a = 349534879; // randomly chosen
    int b = 23479238;  // randomly chosen

    ...

    // return some function of x, a and b
}
```

E.g. The family $h_{a,b}(x) = (ax + b) \mod p$ where p is prime & a, b are uniform, independent draws from $\{0, 1, \ldots, p-1\}$

When did we choose a and b?

Algorithm phases

family $\mathcal{H}$

choose $h \in \mathcal{H}$

Phase 1

Choose algorithm

Determines *where* randomness is needed & *how much*

Phase 2

Random interlude

Make random draws.

Choose hash functions.

Phase 3

Data arrives; Execute!

Use hash functions chosen in Phase 2.

Algorithm phases

Random variables used in analysis are random over the ***choice of hash functions***

Not over the input data

We make ***no distributional assumptions*** about the input.

Phase 1

Choose algorithm

Phase 2

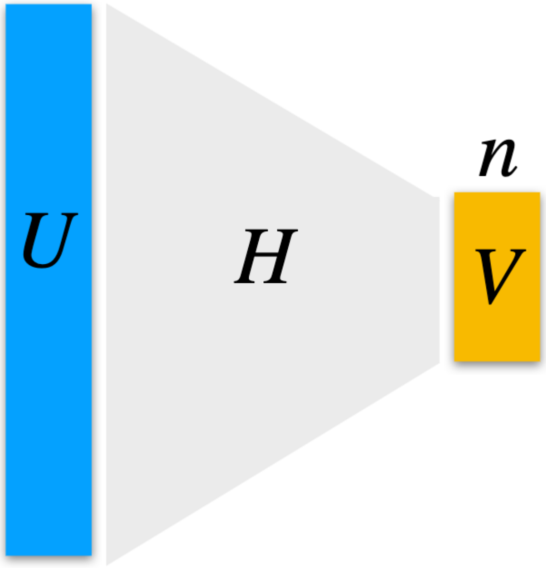
Random interlude

Phase 3

Data arrives; Execute!

Universal hashing

A family of hash functions H from universe U with $|U| \geq n$ to range $\{0, 1, \ldots, n-1\}$

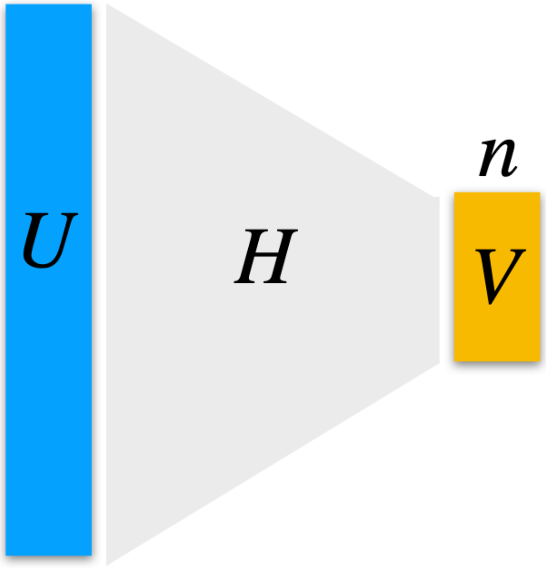


Universal hashing

A family of hash functions H from universe U with $|U| \geq n$ to range $\{0, 1, \ldots, n-1\}$ is **2-universal** if

for distinct elements x_1, x_2 and for function h drawn uniformly from H:

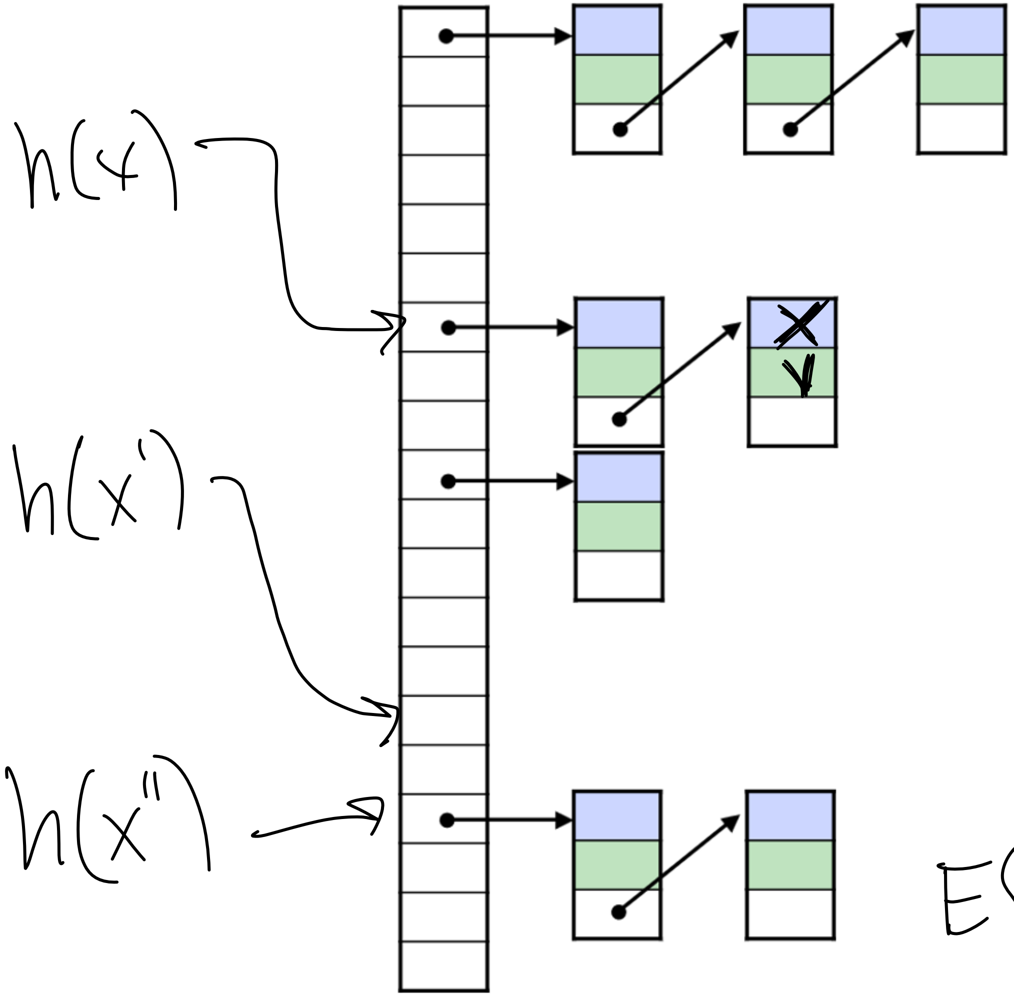
$$\Pr\left(h(x_1) = h(x_2)\right) \leq \frac{1}{n}$$



$\Pr(h(x_1) = h(x_2)) \le \frac{1}{n}$

How do we get O(1) expected query time?

I add m items to an n-bucket hash table



query: what is x's value?

2 options:

x is at $h(x)$

x is not at $h(x)$

- empty
- not in list

C: # of items at $h(x)$

$$E[C] = \begin{cases} \text{if } x \text{ at } h(x) \le 1 + \frac{(m-1)}{n} \\ \text{if } x \text{ not at } h(x) \le \frac{m}{n} \end{cases}$$

Set

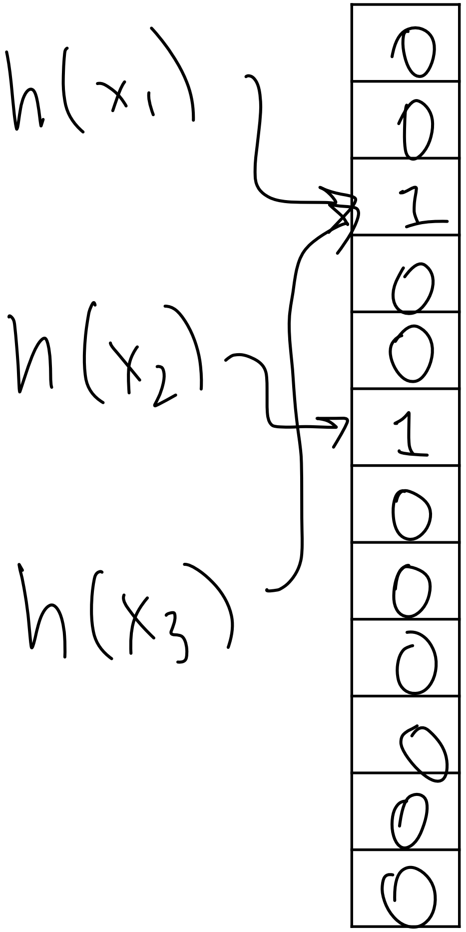
A simpler problem: have we seen x?

But suppose my data is so big that I can't have n close to m

estimate vs. exact

A simpler problem: have we seen x?

But suppose my data is so big that I can't have n close to m



query: have I seen x?

two options:

$h(x) = 1$: have seen x

have seen x' w/ $h(x') = h(x)$

$h(x) = 0$: know I haven't seen x

Bloom Filter