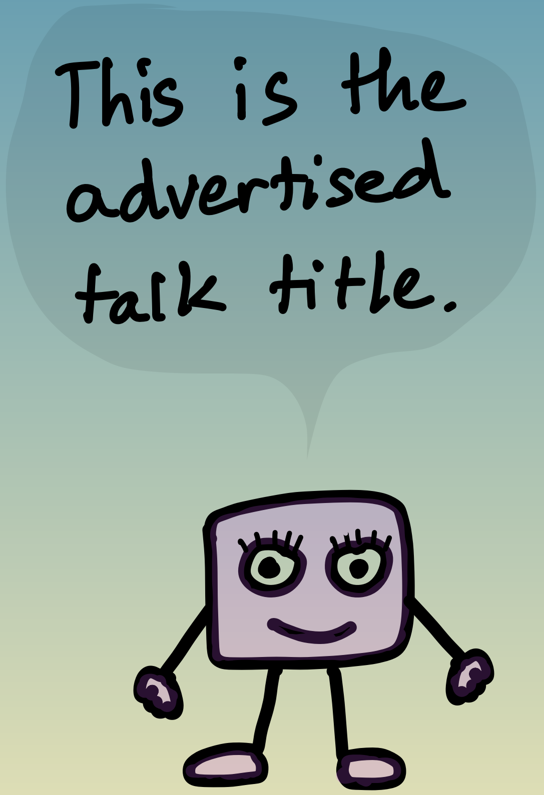
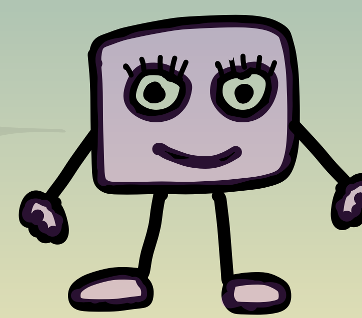
Filters



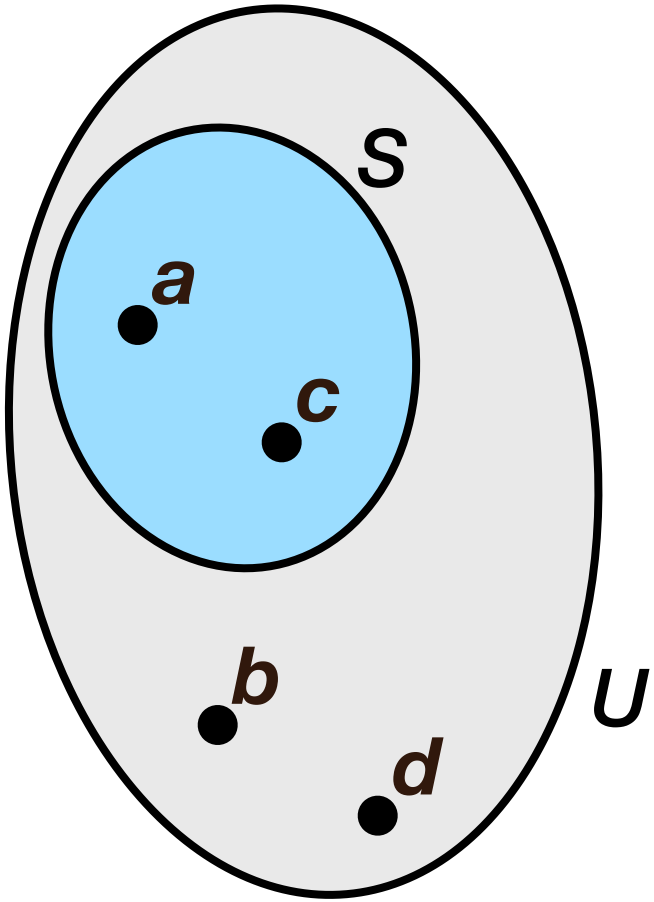
Michael A. Bender

Time To Change Your Filter

Michael A. Bender



A dictionary maintains a set *S* from universe *U*.



member(*a*): ✔

member(*b*): ×

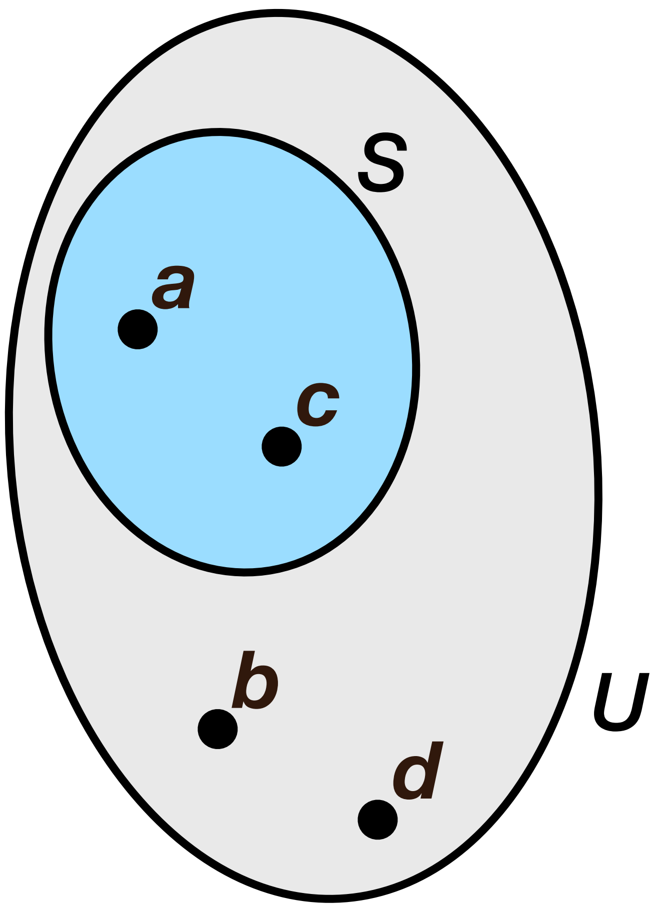
member(*c*): ✔

member(*d*): ×

A dictionary supports membership queries on S.

Filter Data Structure

A filter is an ***approximate*** dictionary.



member(*a*): ✔

member(*b*): ✕

member(*c*): ✔

member(*d*): ✔ false positive

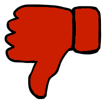
A filter supports <u>approximate</u> membership queries on S.

A Filter Guarantees a False-Positive Rate ε

if $q \in \mathbf{S}$, return ✔ with probability 1 — true positive

if $q \notin \mathbf{S}$, return $\begin{cases} \times & \text{with probability } > 1 - \varepsilon \\ ✔ & \text{with probability } \leq \varepsilon \end{cases}$

true negative



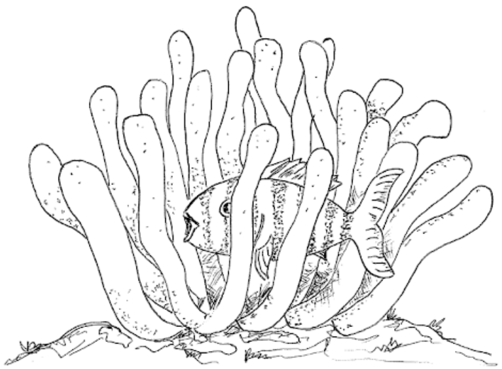
false positive

one-sided errors

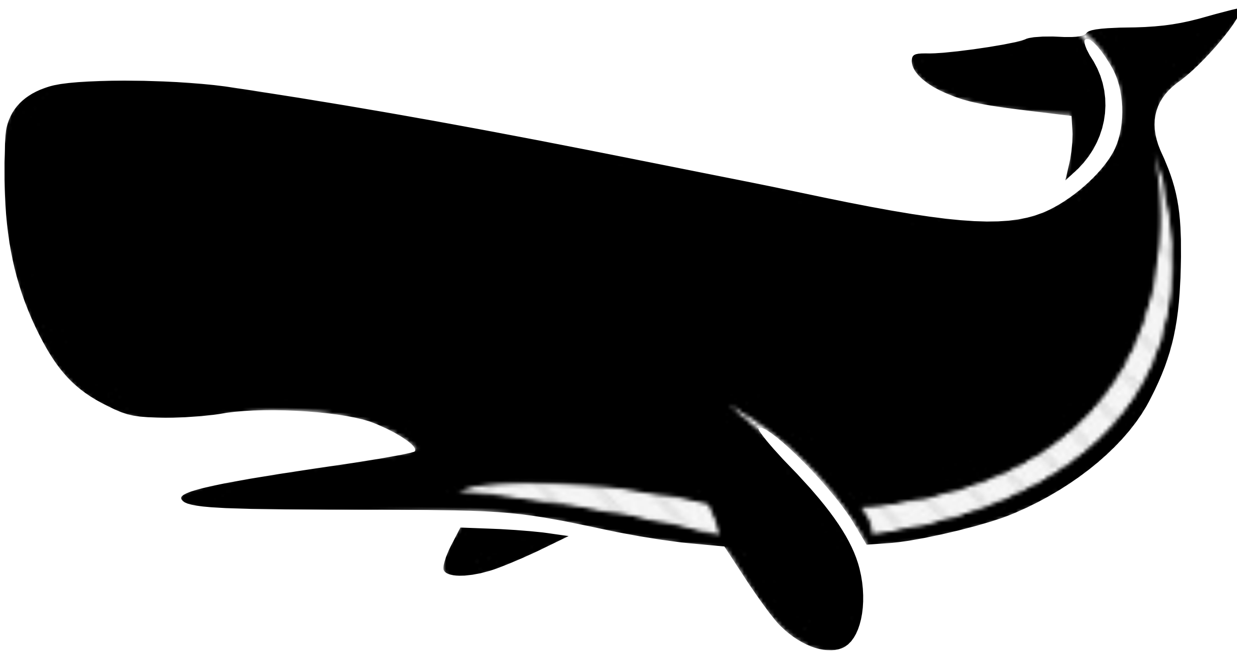
False-positive rate enables filters to be compact

space $\geq n \log \overbrace{(1/\varepsilon)}^{\text{small}}$

space $= \Omega(n \log \overbrace{|U|}^{\text{large}})$



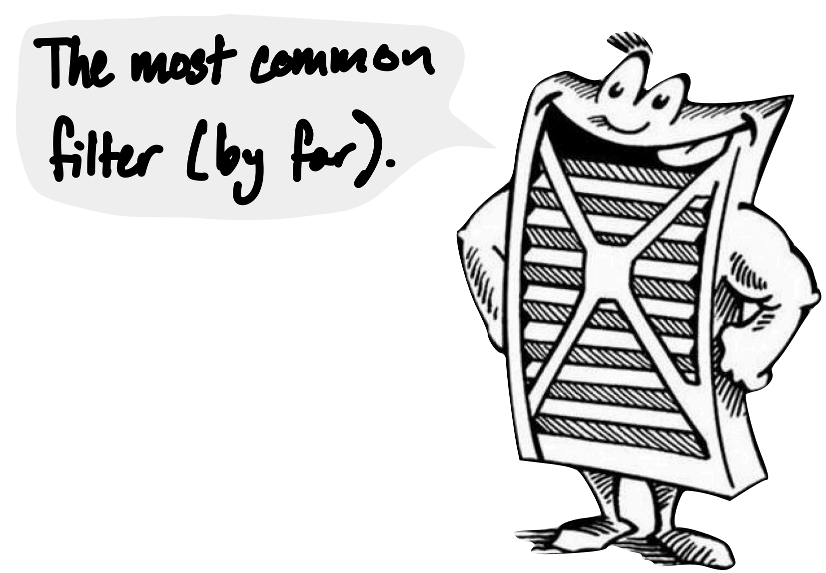
Filter



Dictionary

Talk So Far

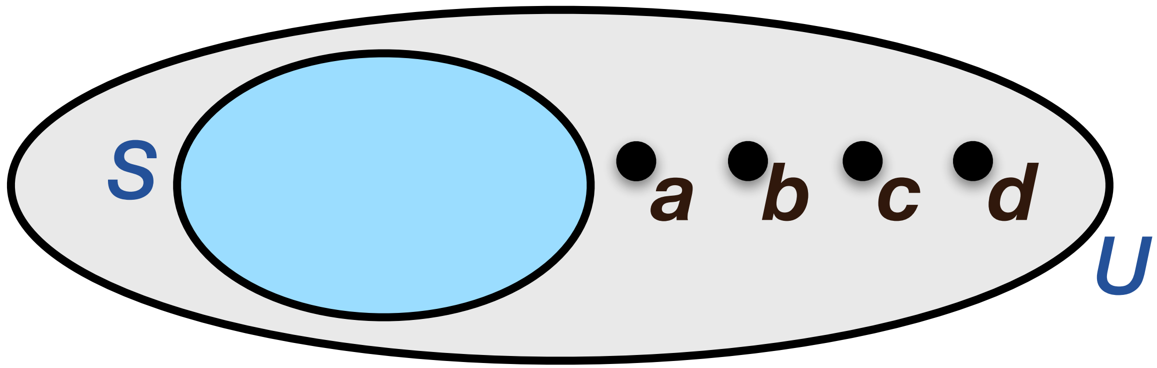
- Filter data structure
- Next: the **Bloom filter** [Bloom '70]



Classic Filter: The Bloom Filter [Bloom '70]

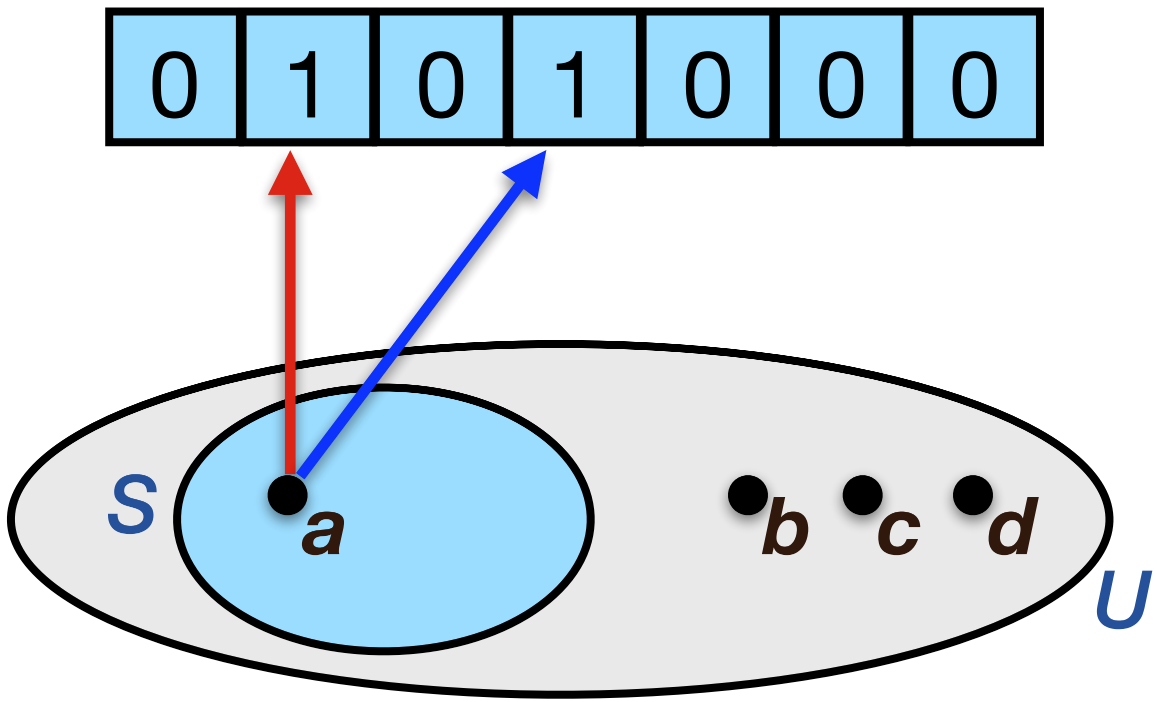
Bloom filter: a bit array + *k* hash functions. (Here k=2.)

0	0	0	0	0	0	0



Classic Filter: The Bloom Filter [Bloom '70]

Bloom filter: a bit array + k hash functions. (Here k=2.)

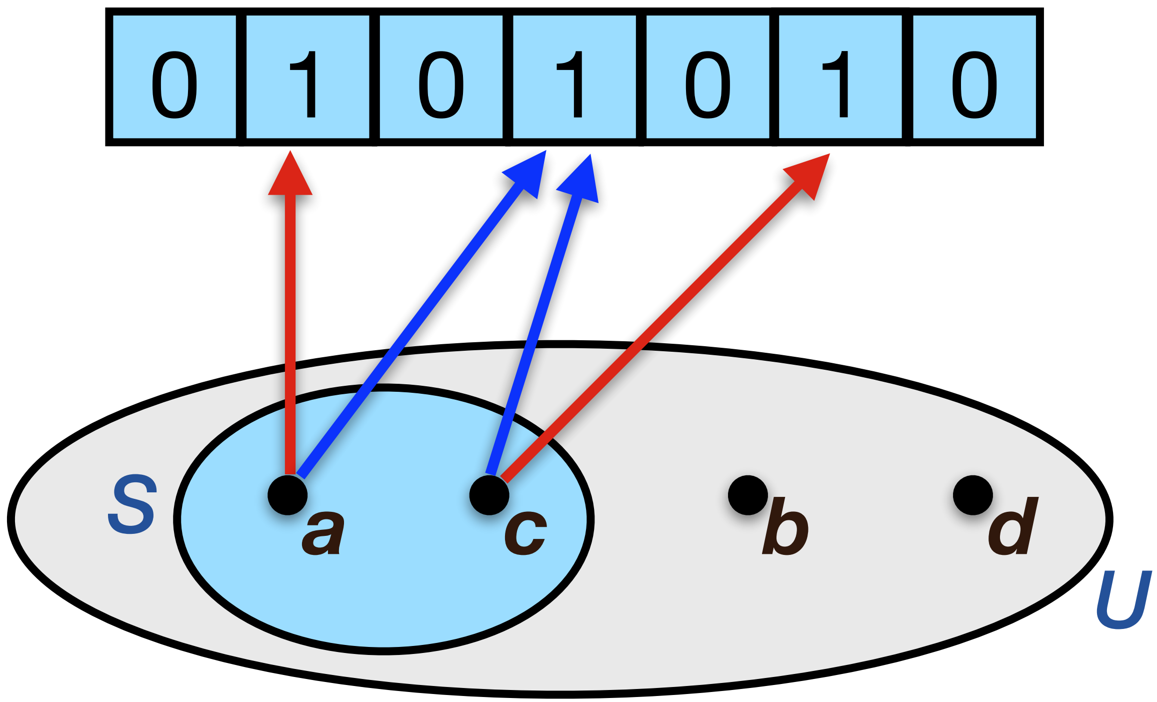


$h_1(a)=1$

$h_2(a)=3$

Classic Filter: The Bloom Filter [Bloom '70]

Bloom filter: a bit array + *k* hash functions. (Here k=2.)



$h_1(a)=1$

$h_2(a)=3$

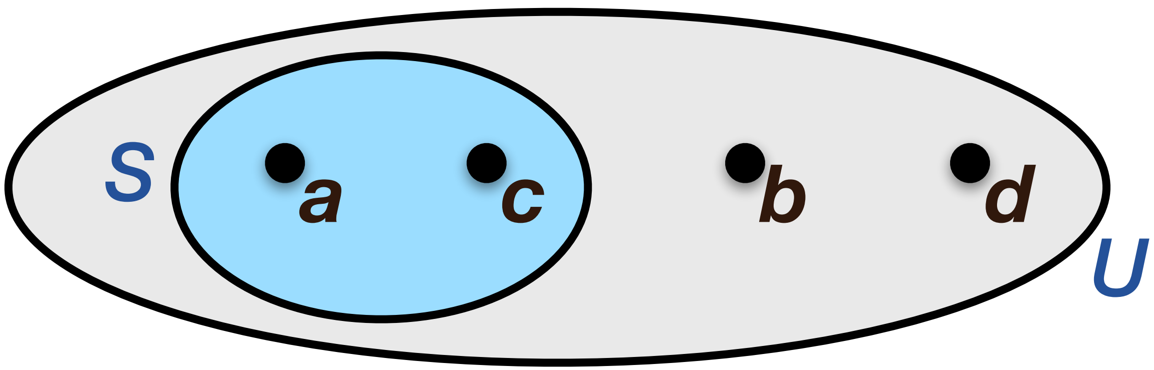
$h_1(c)=5$

$h_2(c)=3$

Classic Filter: The Bloom Filter [Bloom '70]

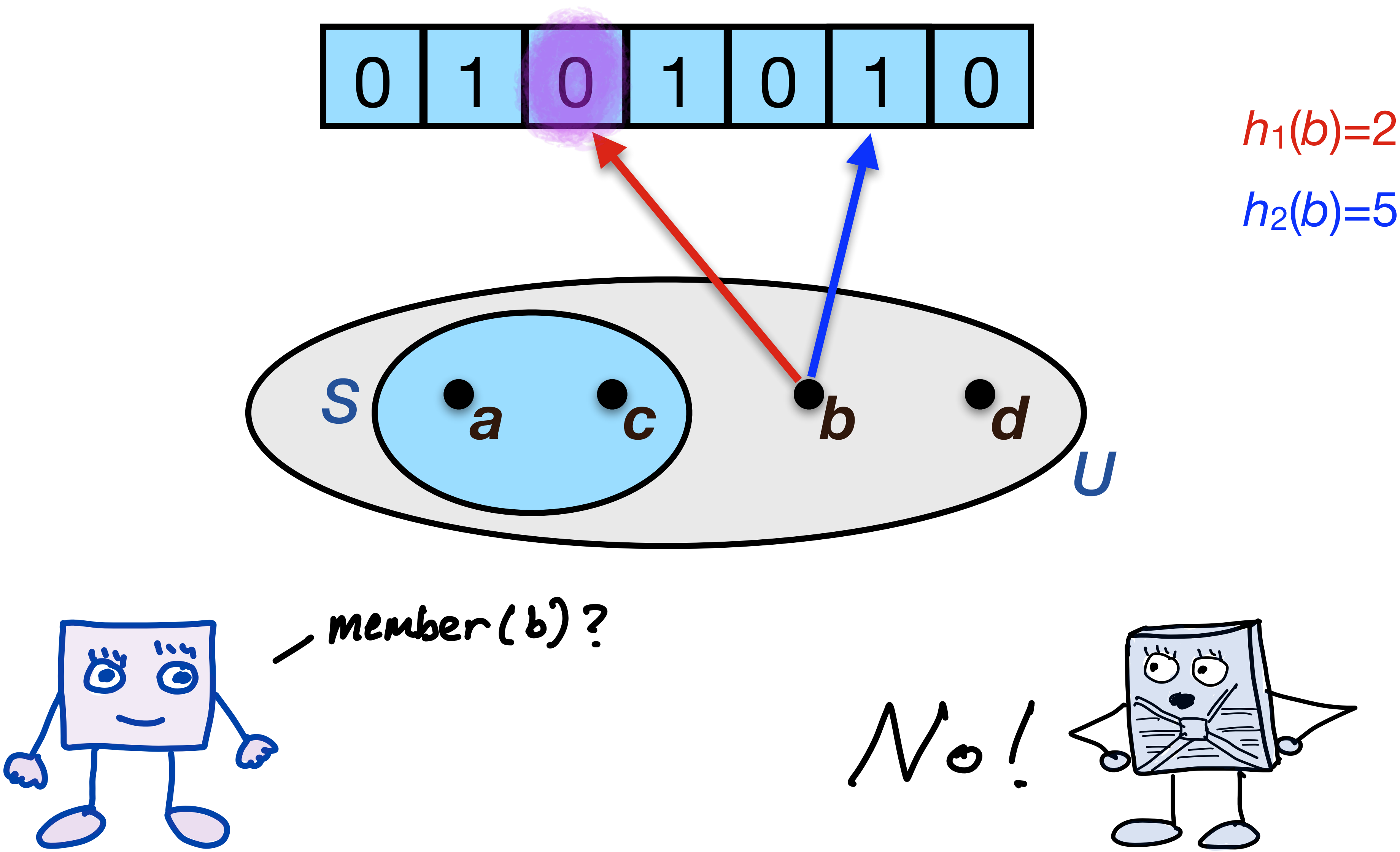
Bloom filter: a bit array + *k* hash functions. (Here k=2.)

0	1	0	1	0	1	0



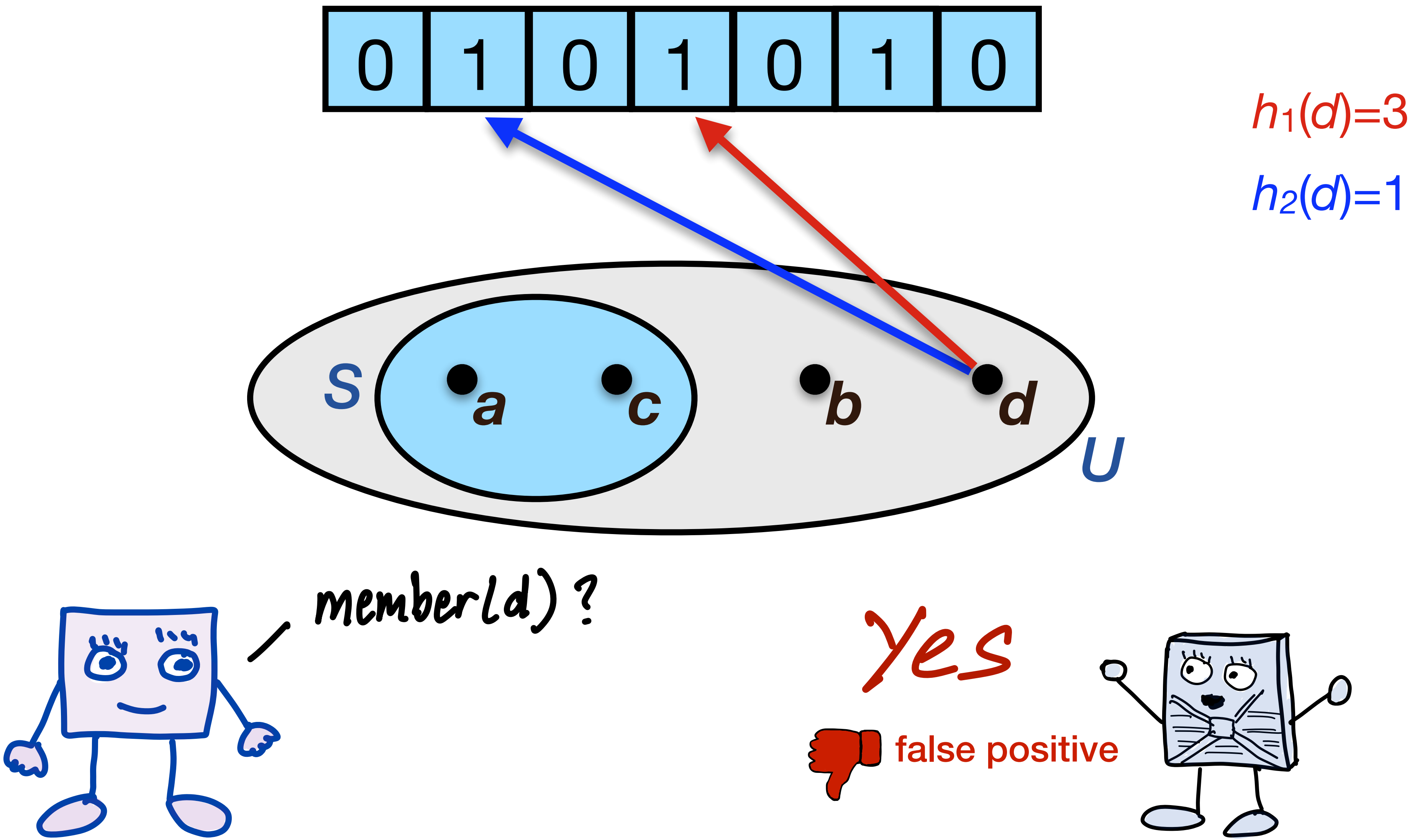
Classic Filter: The Bloom Filter [Bloom ’70]

Bloom filter: a bit array + k hash functions. (Here k=2.)



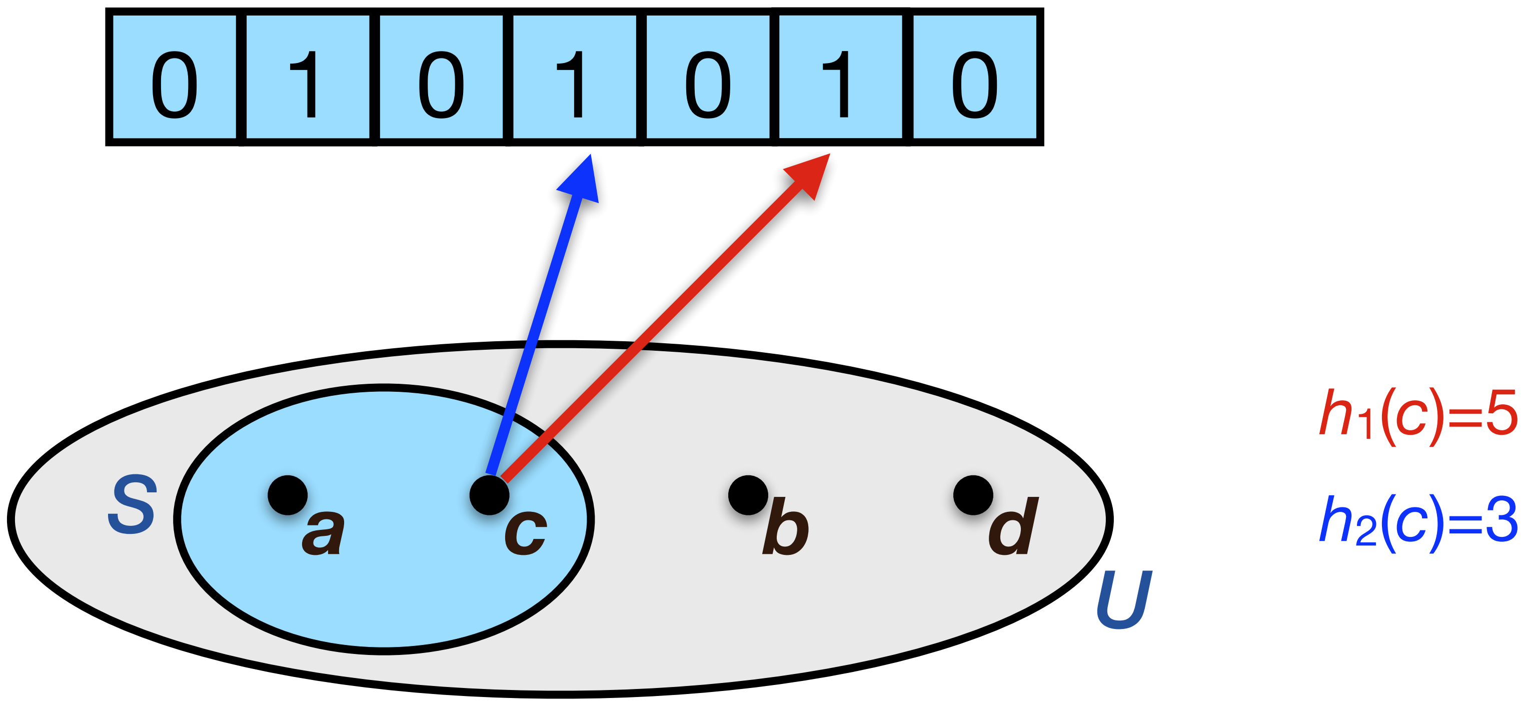
Classic Filter: The Bloom Filter [Bloom '70]

Bloom filter: a bit array + k hash functions. (Here k=2.)



Classic Filter: The Bloom Filter [Bloom '70]

Bloom filter: a bit array + k hash functions. (Here k=2.)



Bloom filters don't support delete.

Issue: on a delete, which 1s get decremented?

Bloom Filter Space Usage [Bloom '70]

Bloom filter space with false-positive rate ε:

≈1.44 lg (1/ε) bits/element.

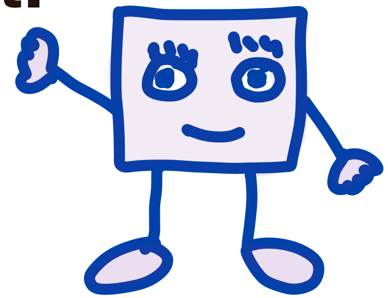
Example:

For ε = 2%,

bits/element ≈ 8.

Common rule of thumb:

Bloom filters take about 1 byte/element.



Bloom filters are ubiquitous

≥4300 citations

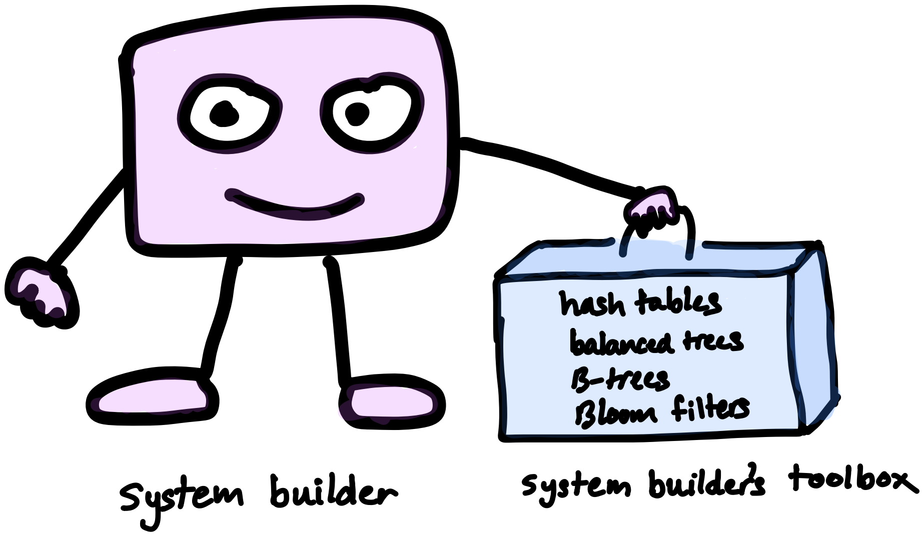
Computational biology

Databases

Networking

Storage systems

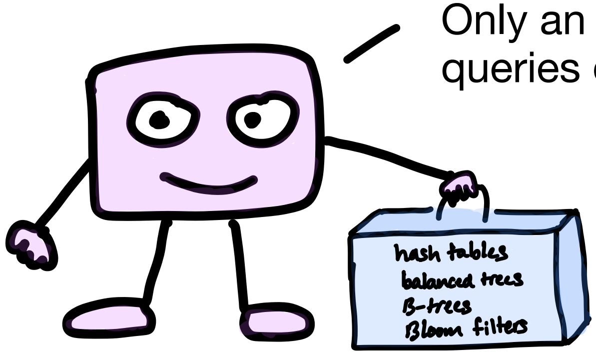
Streaming applications



Talk So Far

- Filter data structure
- The Bloom filter [Bloom '70]
- **How filters are used**

Filter out queries to a large remote dictionary.



Only an ε-fraction of negative queries don't get filtered out.

Filter

local, e.g., in RAM

Dictionary

remote, e.g., on disk

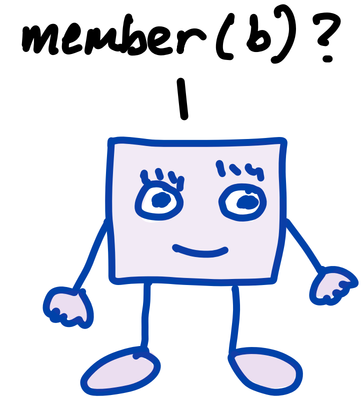
Speedup from Filter Use

Workload has *P* positive and *N* negative queries.

Dictionaries w/o Filters	Dictionaries w/ Filters
$P+N$	$P+\varepsilon N$

Remote Accesses of Dictionary

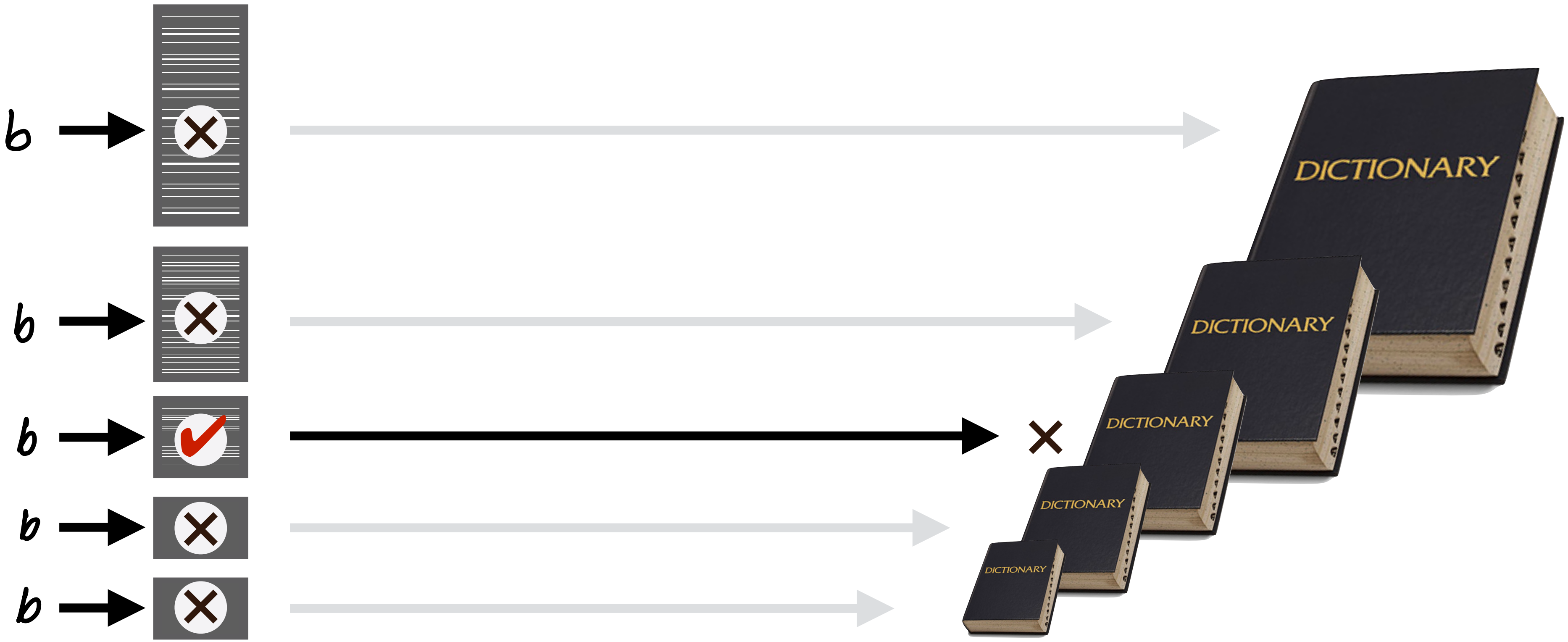
Example: Filters Help Queries in LSM Trees



Log-structured merge tree (LSM)

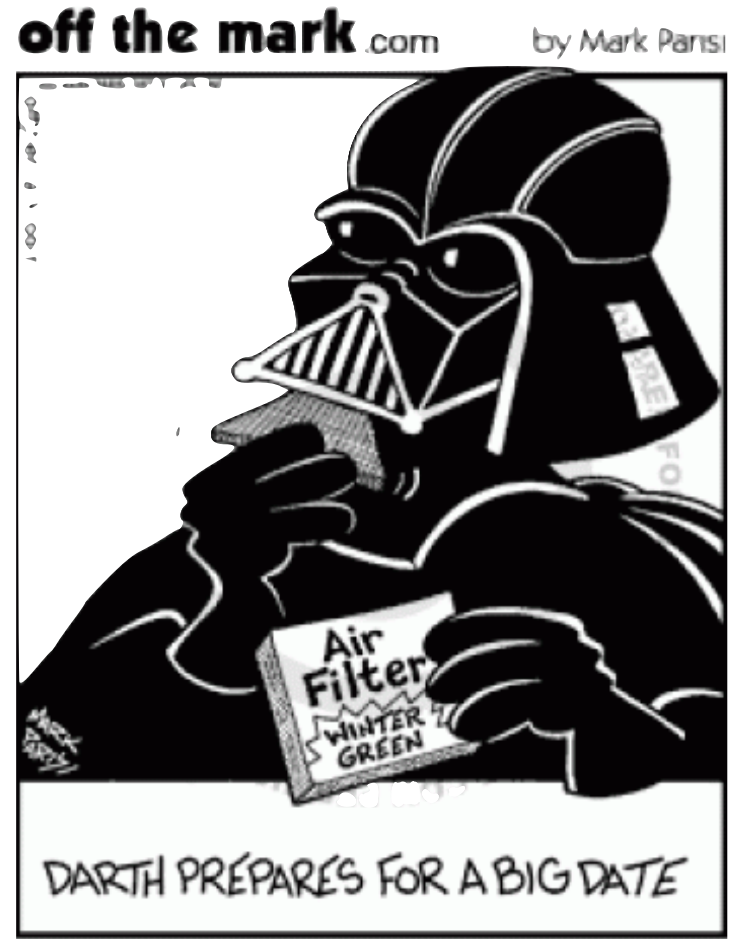
- An LSM tree supports fast inserts by partitioning into independent dictionaries.
  [O'Neil, Cheng, Gawlick, O'Neil '96]

Po*int queries* are slow without filters.



Talk So Far

- Filter data structure
- The Bloom filter [Bloom '70]
- How filters are used
- **Time to change your filter**
  (the talk title)



Application must work around limited Bloom filter capabilities

Limitations	**Work-arounds**
No deletes	Rebuild
No resizes	Guess *N*, and rebuild if wrong
No filter merging nor enumeration of elements	???
No values associated with keys	Combine with other data structure

Bloom filter limitations increase system complexity, waste space, and slow down application performance.

Bloom filters also have suboptimal guarantees

	Bloom filter	Optimal
Space	$\approx 1.44\, n \lg(1/\varepsilon)$	$\approx n \lg(1/\varepsilon) + O(1)$
CPU cost	$\Omega(\lg(1/\varepsilon))$	$O(1)$
Data locality	$\Omega(\lg(1/\varepsilon))$ probes	$O(1)$ probes

Bloom filter limitations increase system complexity, waste space, and slow down application performance.

Tons of Research on Extending/Improving/Replacing Bloom Filters

Deletes + counting

[Bonomi, Mitzenmacher, Panigrahy, Singh, Varghese 06],
[Yuan, Miao, Jia, Wang 08],
[Pandey, Bender, Patro, Johnson SIGMOD 17],

Keys

[Chazelle, Kilian, Rubinfeld, Tal 04]

Filters on SSD

[Canim, Mihaila, Bhattacharhee, Lang, Ross 10],
[Debnath, Sengupta, Lilja, Du 11], [Lu, Debnath, Du. 11], [Bender, Farach-Colton, Johnson,
Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok 12],
[Pandey, Singh, Bender, Berry, Farach-Colton, Johnson, Kroeger, Phillips 20]

Optimizing asymptotics

[Pagh, Pagh, Rao 05],
[Arbitman, Naor, Segev 10],
[Lovett & Porat 10],
[Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18]

Engineering

[Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok 12],
[Fan, Andersen, Kaminsky, Mitzenmacher 14],
[Pandey, Bender, Patro, Johnson SIGMOD 17],
[Pandey, Bender, Johnson, Patro 17],
[Breslow, Jayasena 18],
[Pandey, Conway, Durie, Bender, Martin, Johnson 21]

Adaptivity

[Mitzenmacher, Pontarelli, Reviriego 18]
[Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18]
[Bender, Das, Farach-Colton, Mo, Wang 21]

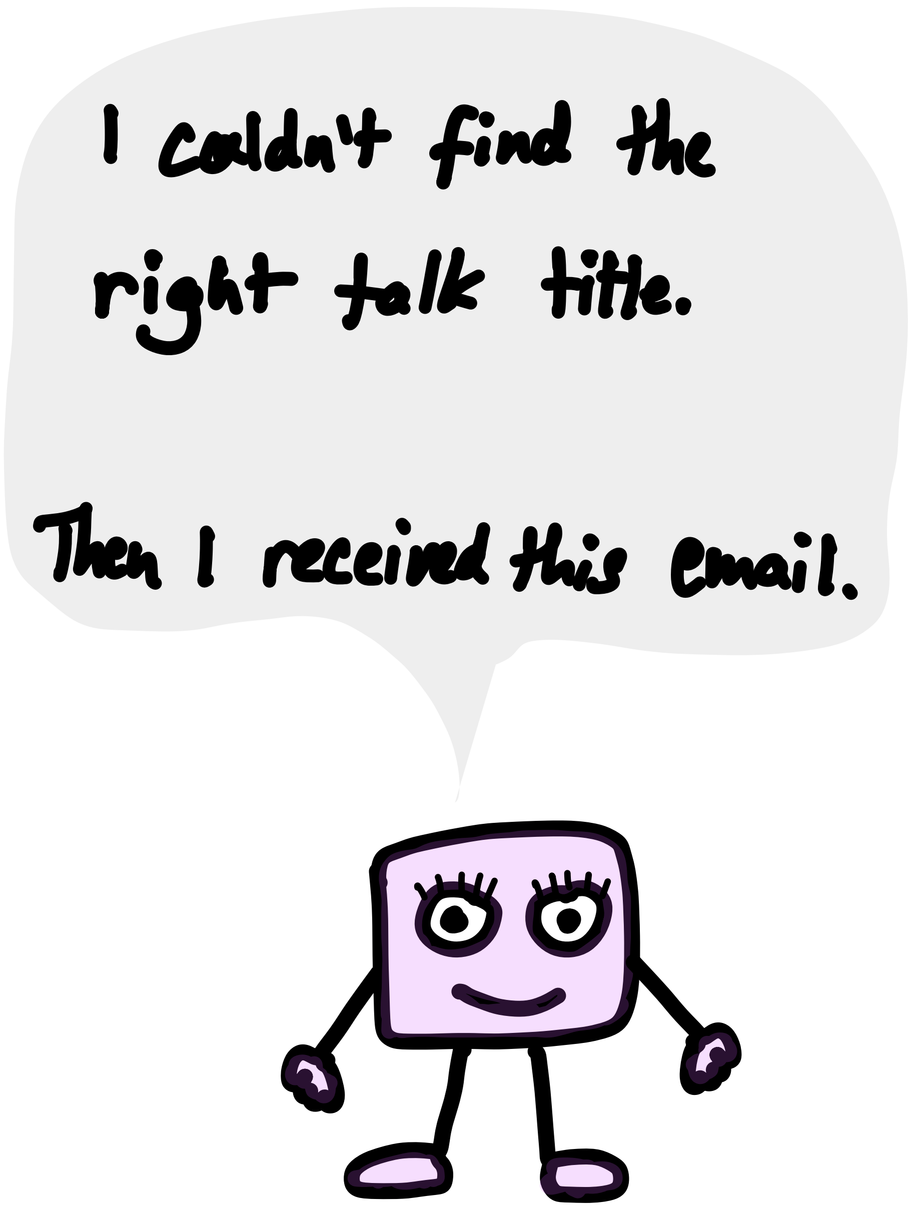
When too much stuff is growing on your filter, it's time to change the filter.

Talk So Far

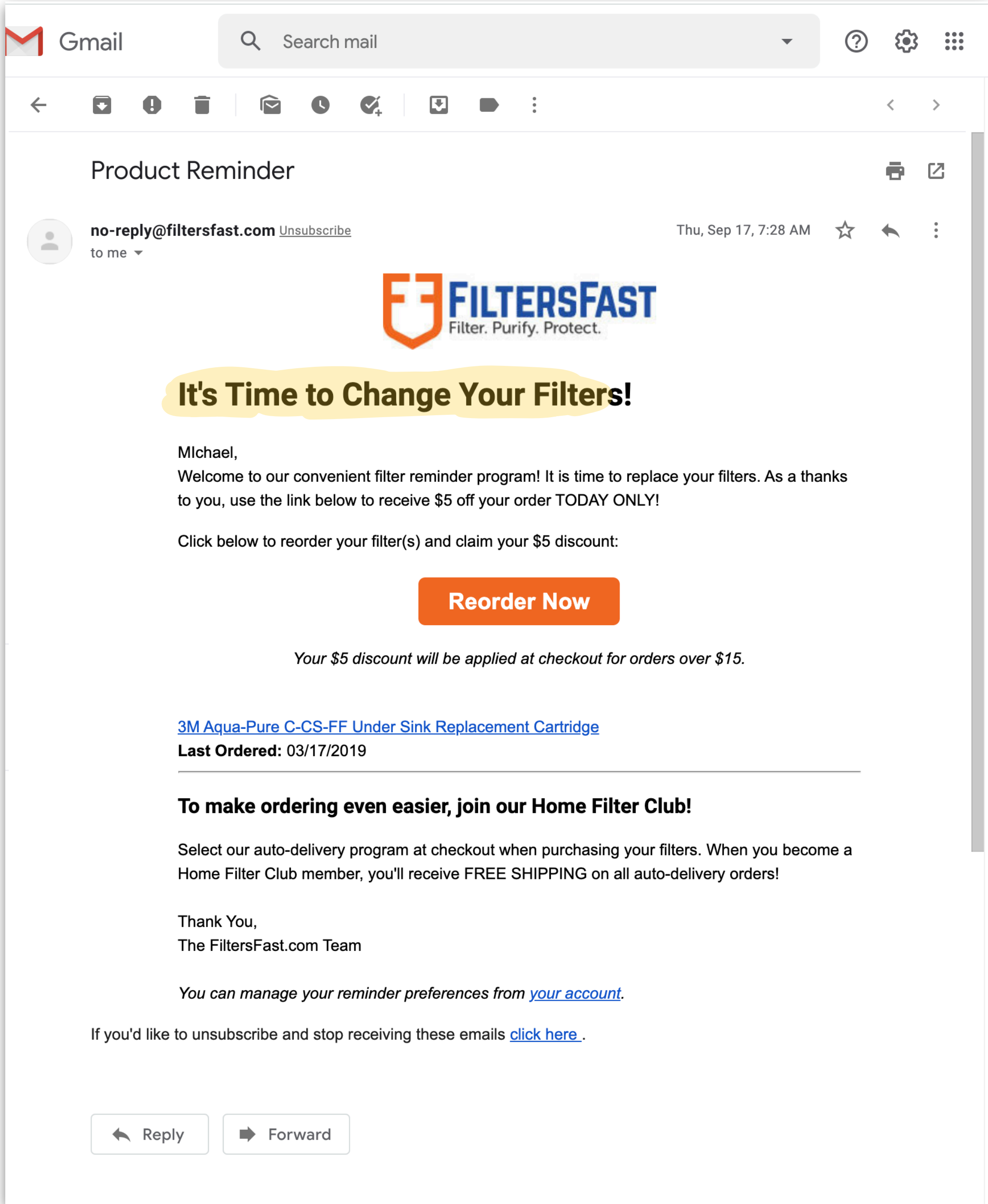
- Filter data structure
- The Bloom filter [Bloom ’70]
- How filters are used
- **Time to change your filter**
  (the talk title)

“I see that you change the filter about as often as you change your mind.”

Message of Talk



Message of Talk


Gmail

Search mail

Product Reminder

no-reply@filtersfast.com Unsubscribe

to me

Thu, Sep 17, 7:28 AM

FILTERSFAST
Filter. Purify. Protect.

It's Time to Change Your Filters!

MIchael,
Welcome to our convenient filter reminder program! It is time to replace your filters. As a thanks to you, use the link below to receive $5 off your order TODAY ONLY!

Click below to reorder your filter(s) and claim your $5 discount:

Reorder Now

Your $5 discount will be applied at checkout for orders over $15.

3M Aqua-Pure C-CS-FF Under Sink Replacement Cartridge
Last Ordered: 03/17/2019

To make ordering even easier, join our Home Filter Club!

Select our auto-delivery program at checkout when purchasing your filters. When you become a Home Filter Club member, you'll receive FREE SHIPPING on all auto-delivery orders!

Thank You,
The FiltersFast.com Team

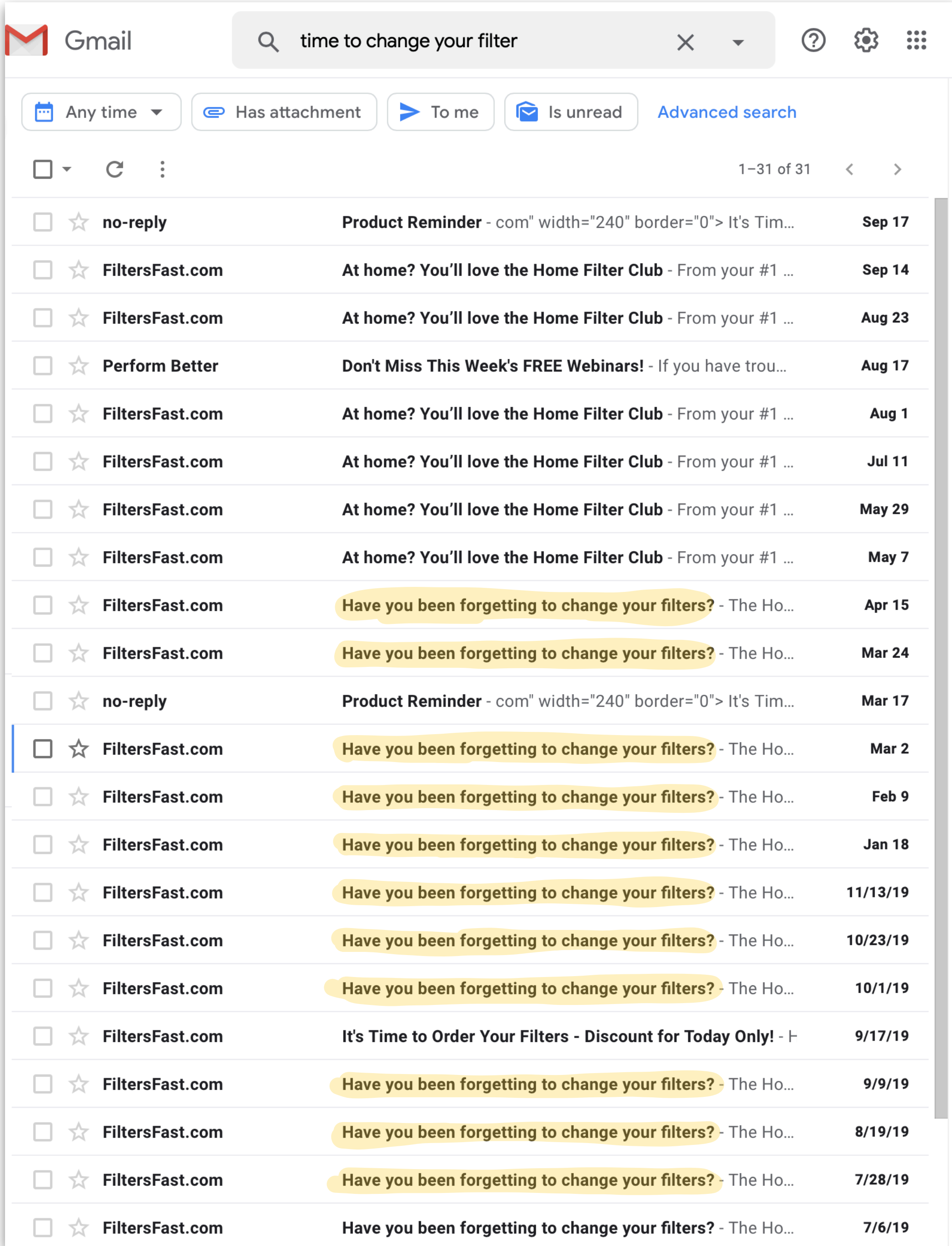
You can manage your reminder preferences from your account.

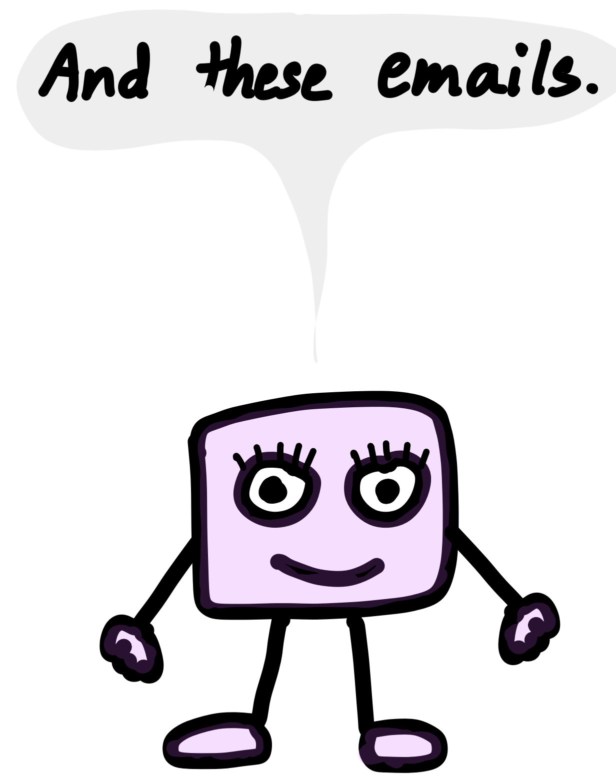
If you'd like to unsubscribe and stop receiving these emails click here .

Reply

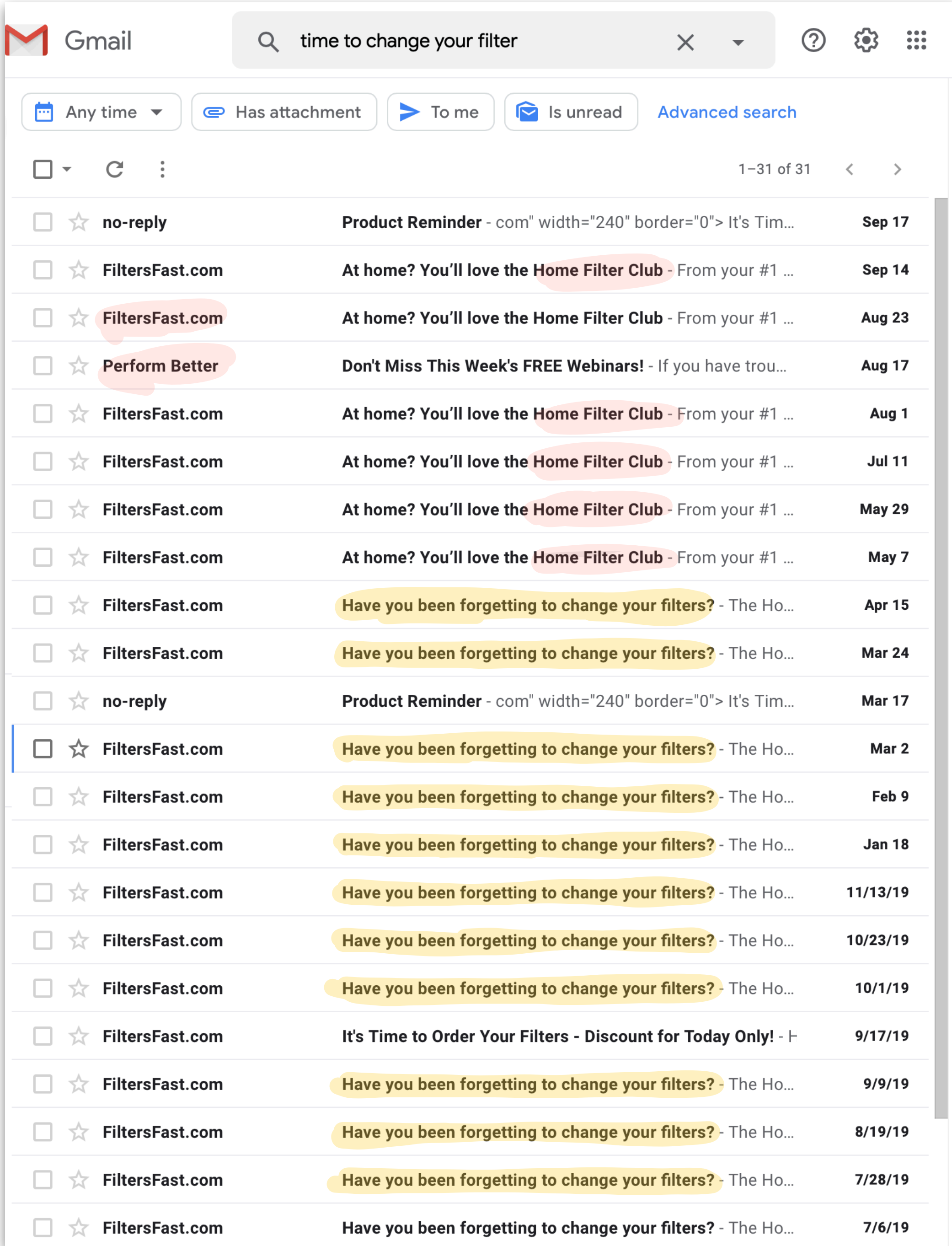
Forward

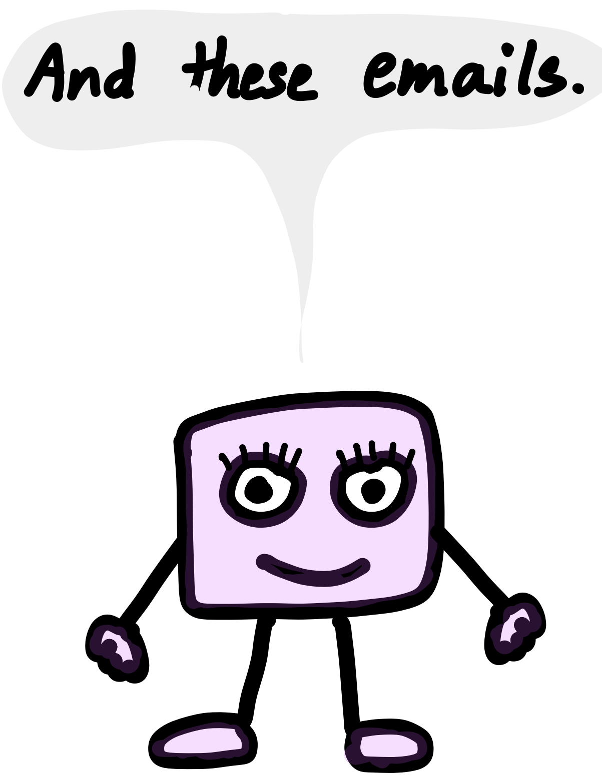
Message of Talk





Message of Talk





This Talk: It’s Time to Change Your Filter

- Tutorial-like introduction to filters.
- General techniques for solving filter problems.

fingerprinting

quotienting

collision resolution

This Talk: It’s Time to Change Your Filter

- Tutorial-like introduction to filters.
- General techniques for solving filter problems.

fingerprinting

quotienting

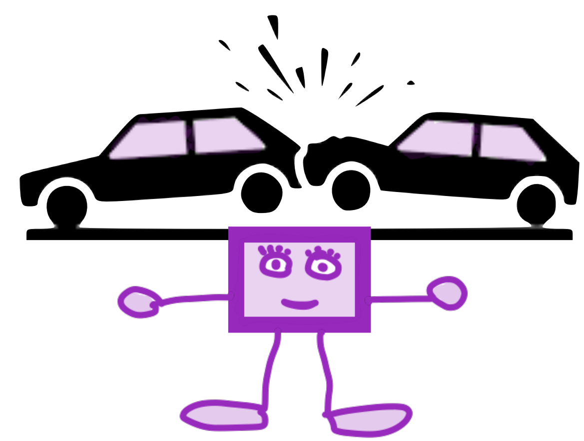
collusion
~~collision~~ resolution

This Talk

- Tutorial-like introduction to filters.
- General techniques for solving filter problems.

fingerprinting

quotienting

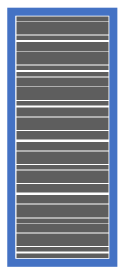


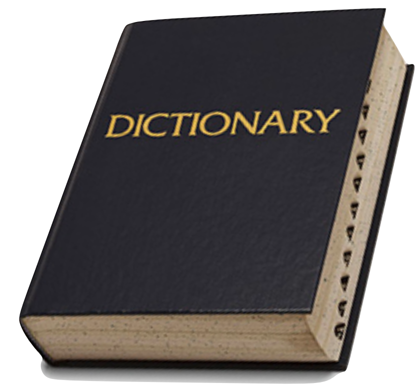
collision resolution

Fingerprinting

[Cleary 84] [Pagh, Pagh, Rao 05], [Arbitman, Naor, Segev 10], [Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok 12], [Fan, Andersen, Kaminsky, Mitzenmacher 14], [Pandey, Bender, Johnson, Patro 17], [Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18]

Filter F of set S : $\{h(x) \mid x \in S\}$





$$\boldsymbol{F} = \{\, h(x_1), h(x_2), \ldots, h(x_n) \}$$

$$\boldsymbol{S} = \{x_1, x_2, \ldots, x_n\}$$

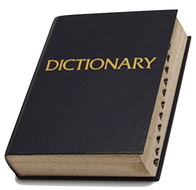
F can be stored more compactly than a Bloom filter.

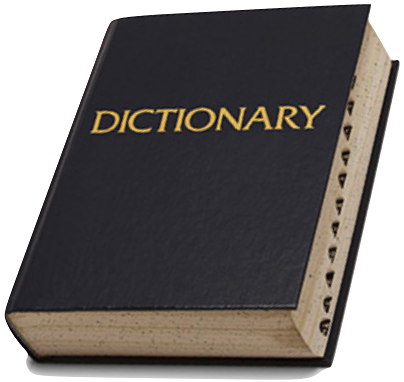
fingerprinting

Fingerprinting

[Cleary 84] [Pagh, Pagh, Rao 05], [Arbitman, Naor, Segev 10], [Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok 12], [Fan, Andersen, Kaminsky, Mitzenmacher 14], [Pandey, Bender, Johnson, Patro 17], [Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18]

Filter F of set S : $\{h(x) \mid x \in S\}$





$$F = \{\, h(x_1), h(x_2), \ldots, h(x_n) \}$$

$$S = \{x_1, x_2, \ldots, x_n\}$$

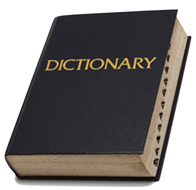
F can be stored more compactly than a Bloom filter.

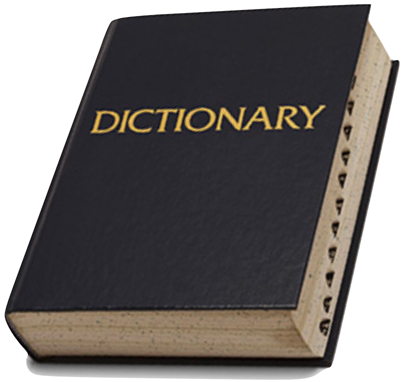
F is also just dictionary—just a more compact one.

Fingerprinting

[Cleary 84] [Pagh, Pagh, Rao 05], [Arbitman, Naor, Segev 10], [Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok 12], [Fan, Andersen, Kaminsky, Mitzenmacher 14], [Pandey, Bender, Johnson, Patro 17], [Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18]

Filter F of set S : $\{h(x) \mid x \in S\}$





$$F = \{\, h(x_1), h(x_2), \ldots, h(x_n) \,\}$$

$$S = \{x_1, x_2, \ldots, x_n\}$$

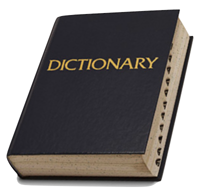
$$\textbf{member}(x) = \begin{cases} \checkmark & \text{if } \ h(x) \in F \\ \times & \text{if } \ h(x) \notin F \end{cases}$$

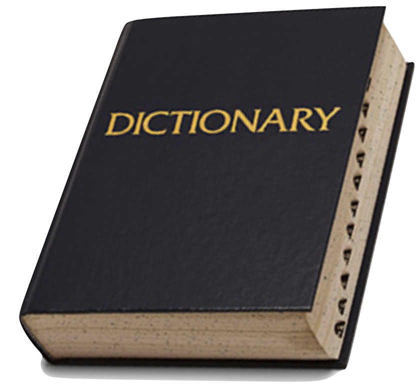
fingerprinting

Fingerprinting

[Cleary 84] [Pagh, Pagh, Rao 05], [Arbitman, Naor, Segev 10], [Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok 12], [Fan, Andersen, Kaminsky, Mitzenmacher 14], [Pandey, Bender, Johnson, Patro 17], [Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18]

Filter F of set S : $\{h(x) \mid x \in S\}$





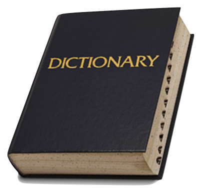
$$F = \{\, h(x_1), h(x_2), \ldots, h(x_n) \,\}$$

$$S = \{x_1, x_2, \ldots, x_n\}$$

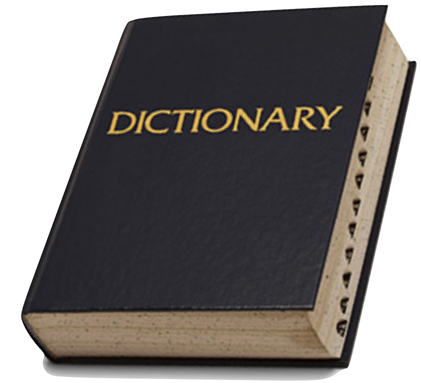
$$\mathbf{member}(x) = \begin{cases} \checkmark & \text{if } h(x) \in F \\ \times & \text{if } h(x) \notin F \end{cases}$$

F can be stored more compactly than a Bloom filter.

False-Positive Analysis for Fingerprinting



$$F = \{\, h(x_1), h(x_2), \ldots, h(x_n) \,\}$$



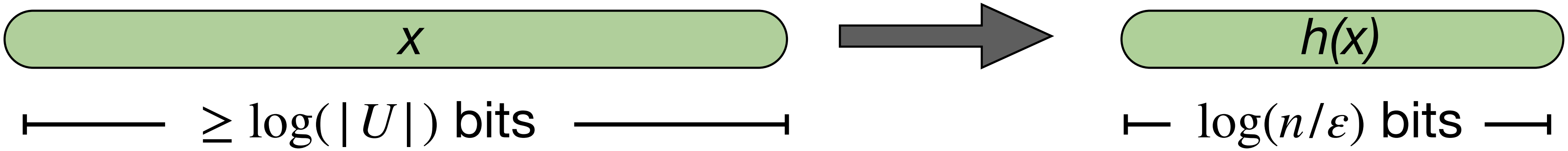
$$S = \{x_1, x_2, \ldots, x_n\}$$

member(y)=✔
if $h(y) \in F$.

y **is a false positive**
if $\exists\, x_i\; h(y)=h(x_i)$, but $y \notin S$.

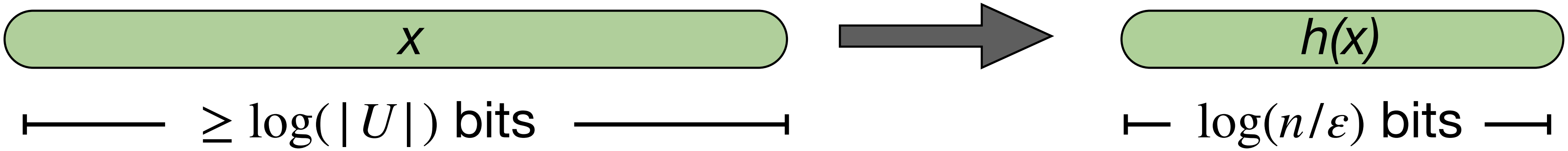
Fingerprint collisions: *only* source of false positives.

False-Positive Analysis for Fingerprinting



$$\Pr[x \text{ and } y \text{ collide}] = \frac{1}{2^{\log(n/\varepsilon)}} = \frac{\varepsilon}{n}$$

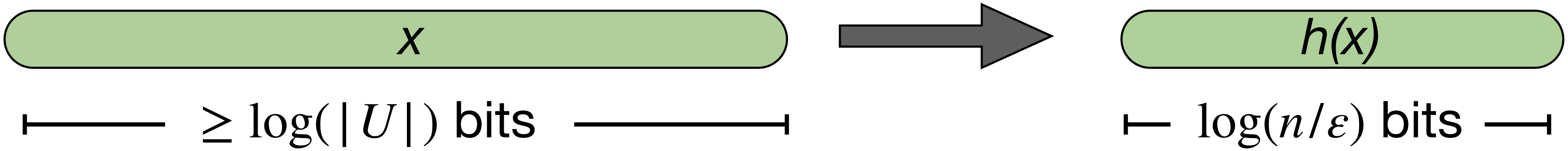
False-Positive Analysis for Fingerprinting



$$\Pr[x \text{ and } y \text{ collide}] = \frac{1}{2^{\log(n/\varepsilon)}} = \frac{\varepsilon}{n}$$

$$\Pr[y \notin S \text{ is a false positive}] \leq \varepsilon$$

False-Positive Analysis for Fingerprinting



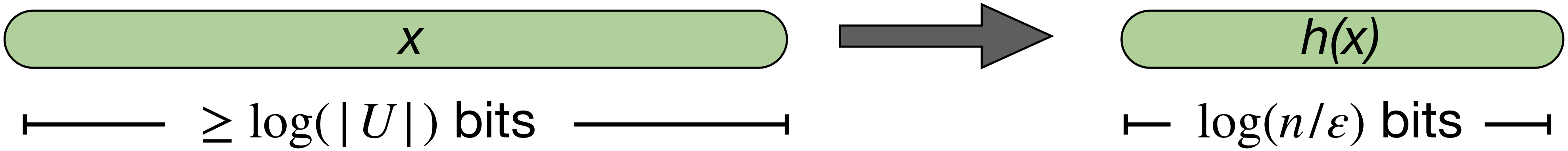
$$\Pr[x \text{ and } y \text{ collide}] = \frac{1}{2^{\log(n/\varepsilon)}} = \frac{\varepsilon}{n}$$

$$\Pr[y \notin S \text{ is a false positive}] \leq \varepsilon$$

***n* fingerprints can be stored compactly:**

$$\log(1/\varepsilon) + O(1) \text{ bits/element}$$

False-Positive Analysis for Fingerprinting



$$\Pr[x \text{ and } y \text{ collide}] = \frac{1}{2^{\log(n/\varepsilon)}} = \frac{\varepsilon}{n}$$

$$\Pr[y \notin S \text{ is a false positive}] \leq \varepsilon$$

***n* fingerprints can be stored compactly:**

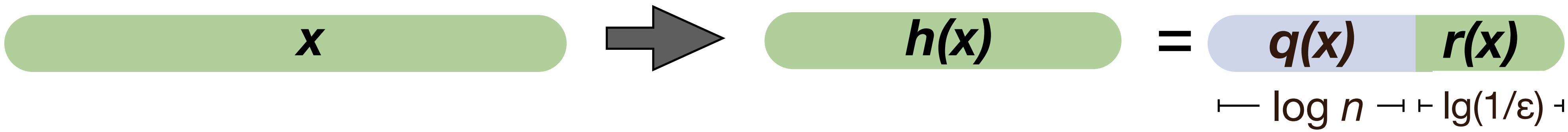
$$\log(1/\varepsilon) + O(1) \text{ bits/element}$$

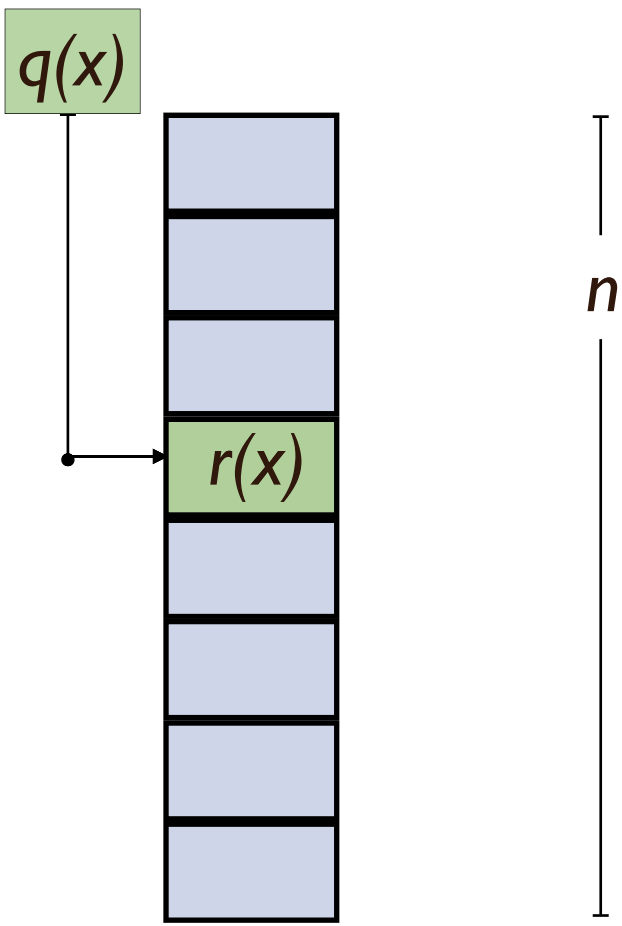

fingerprinting


quotienting

Compact Storage Using Quotienting [Knuth]

Space: O(lg (1/ε)) bits per element



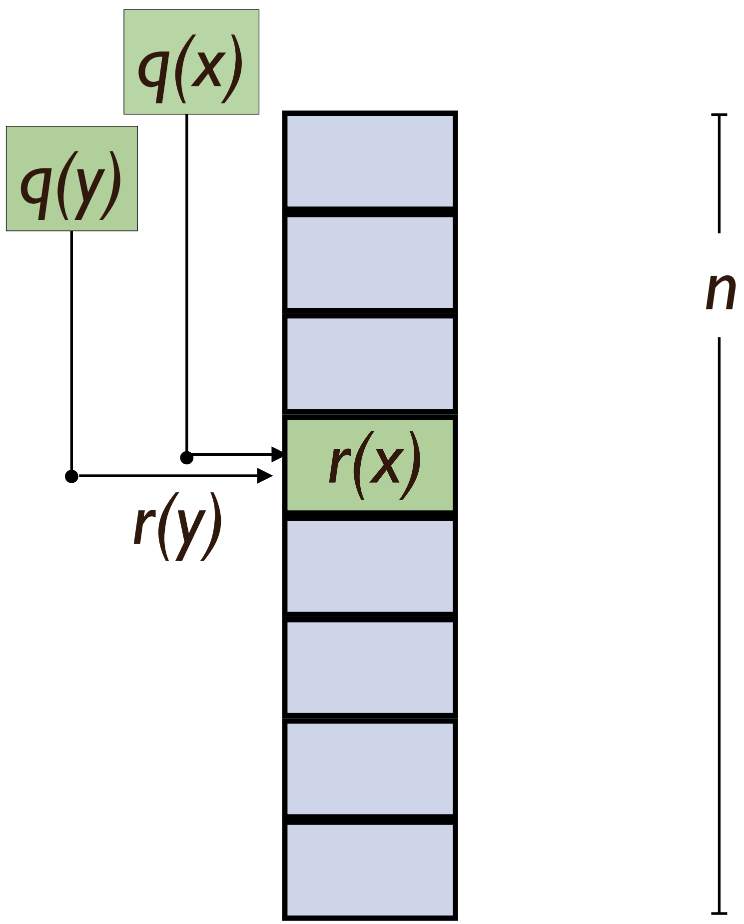


q(x)=location in hash table

r(*x*)=data stored in hash table

Compact Storage Using Quotienting [Knuth]

Space: O(lg (1/ε)) bits per element



q(*x*)=location in hash table

r(*x*)=data stored in hash table

How to deal with collisions in the hash table?

Isn't this a solved problem?

What's wrong with out-of-the-box linear probing?

Hash Collisions in Quotienting

Ex: 6 bit hash. 3 bits for address, 3 for data.

000	001
001	011
010	
011	010
100	111
101	
110	100
111	

Does this element represent 001011 or 000011?

Does this element represent 100111 or 011111?

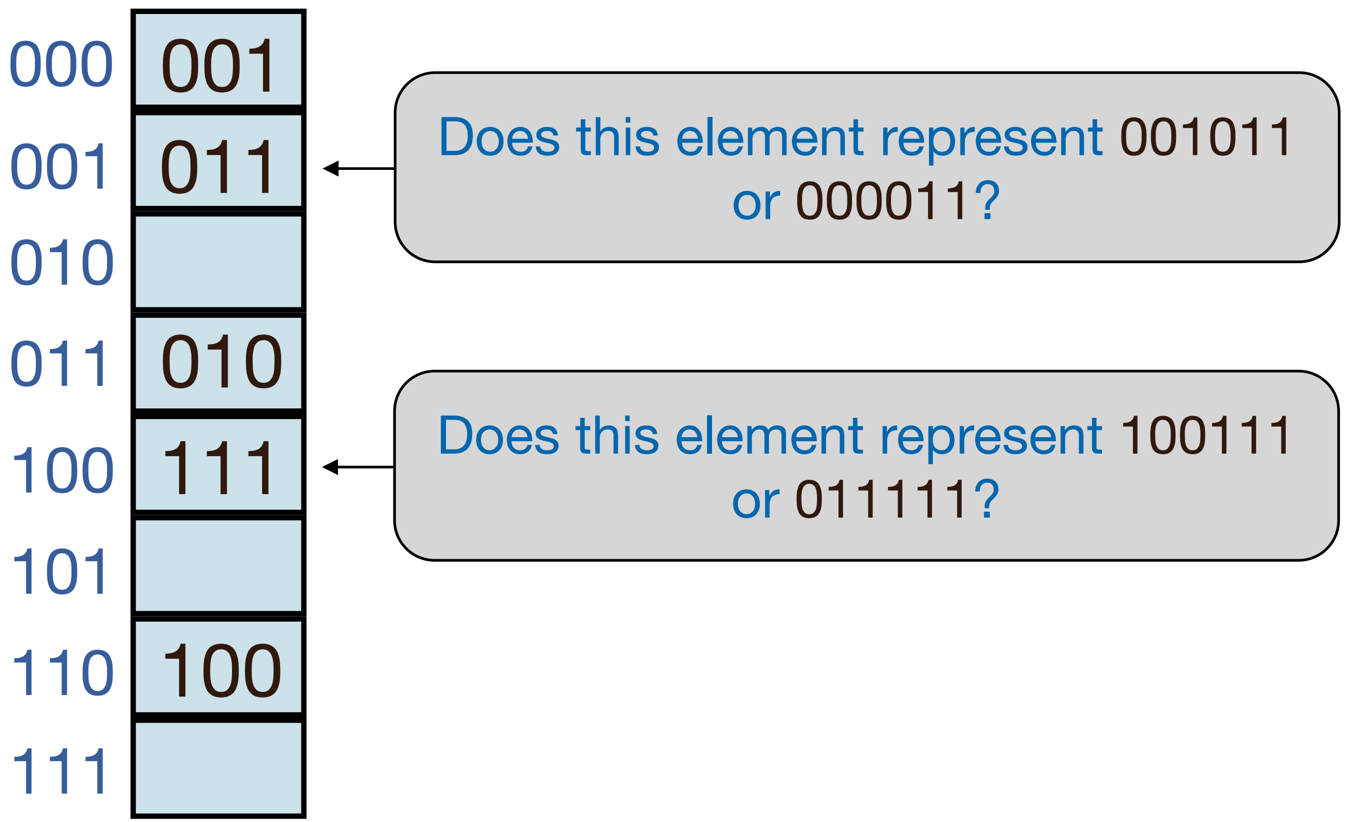
The hash is stored implicitly based on location. So how can we change its location?


fingerprinting

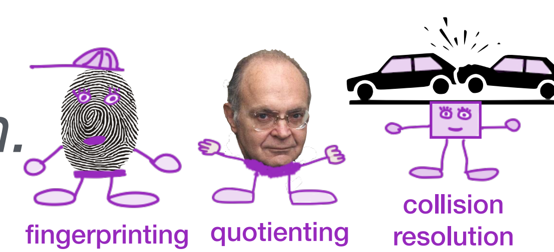

quotienting

Hash Collisions in Quotienting

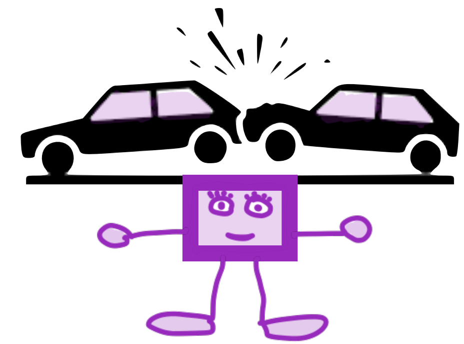
Ex: 6 bit hash. 3 bits for address, 3 for data.



The hash is stored implicitly based on location.
So how can we change its location?



Talk Structure

- Filters + how filters are used + Bloom limitations
-  fingerprinting  quotienting  collision resolution
-

Quotient filters variants	Cuckoo filter variants	Miscellaneous
[Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok 12], [Pandey, Bender, Johnson, Patro 17], [Pandey, Bender, Conway, Farach-Colton, Johnson 21]	[Fan, Andersen, Kaminsky, Mitzenmacher 14], [Breslow, Jayasena 18]	[Pagh, Pagh, Rao 05], [Arbitman, Naor, Segev 10], [Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18]
uses linear proving and Robinhood hashing [Celis, Larson, Munro 85]	Uses Cuckoo hashing [Pagh, Rodler 01]	Balls and bins + more advanced hashing

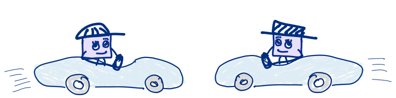
[Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok VLDB 12]
[Pandey, Bender, Patro, Johnson SIGMOD 17]

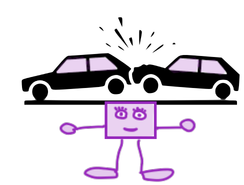
2 metadata bits per slot let us recover original location.

000	0		0
001	1	001	0
010	1	010	1
011	0	111	1
100	0	010	0
101	0		0
110	1	001	0
111	0	011	1

1= “something is hashed here”

1= “I’m hashed to the same place as the element before me”





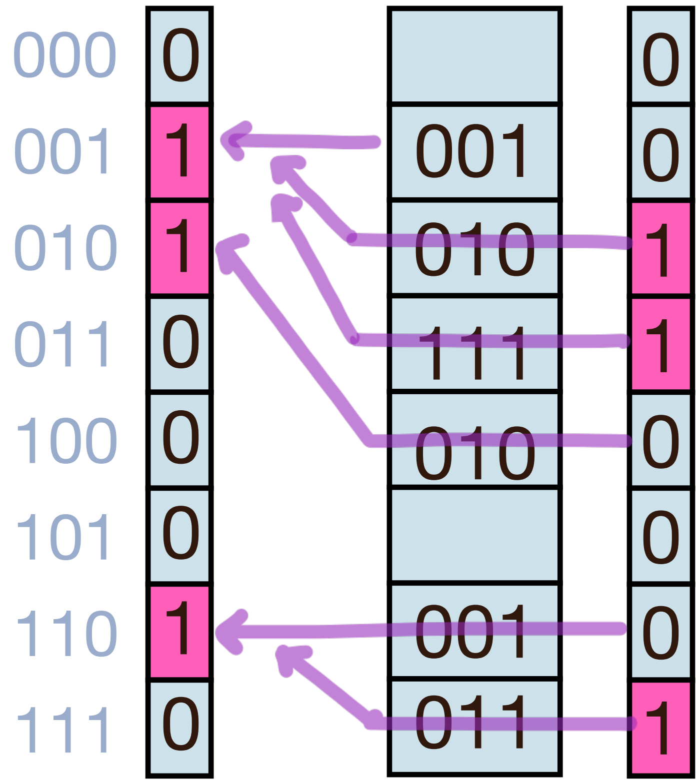
collision resolution

Quotient Filters

[Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok VLDB 12]
[Pandey, Bender, Patro, Johnson SIGMOD 17]

2 metadata bits per slot let us recover original location.

Recall: no element is stored before its target position.



This is an empty slot.

Remainder is in correct slot.

Remainder is mapped to same location as previous remainder.

Remainder is mapped to same location as previous remainder.

Remainder is mapped to next array position.

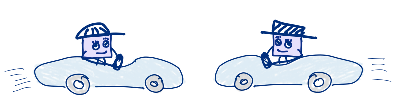
Remainder would map to the next position. But there's none, so it's empty.

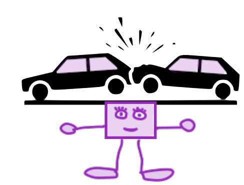
Remainder is in correct slot.

Remainder is mapped to same location as previous remainder.

1= "something is hashed here"

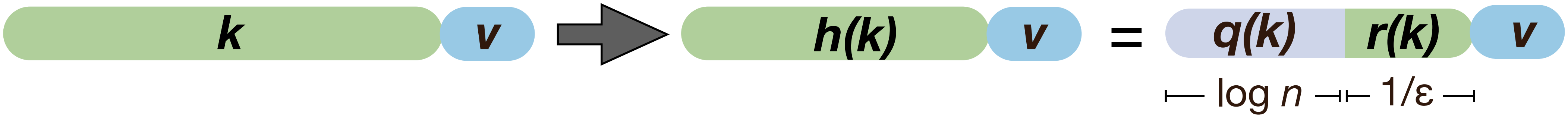
1= "I'm hashed to the same place as the element before me"

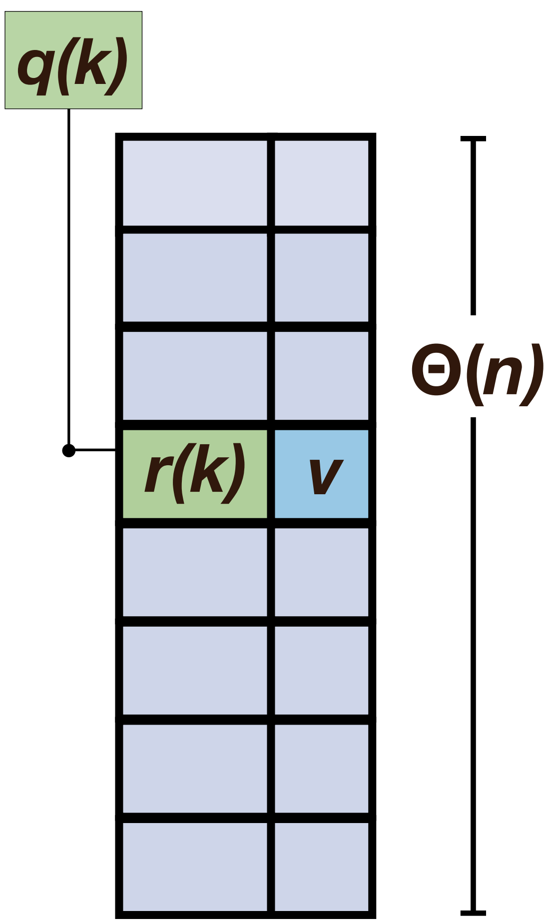




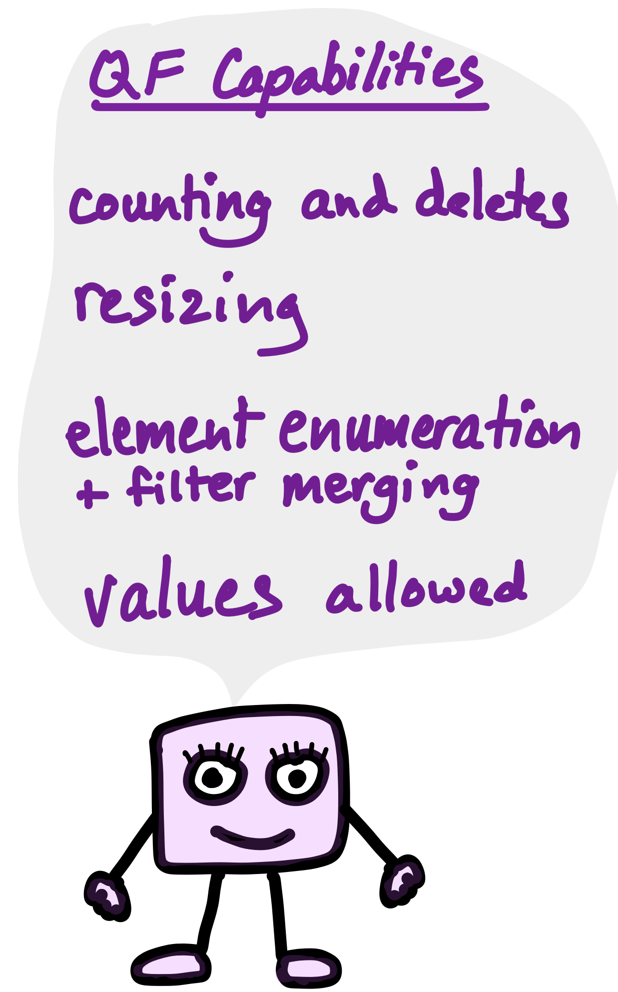
Quotient Filter Capabilities

[Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok VLDB 12]
[Pandey, Bender, Patro, Johnson SIGMOD 17]



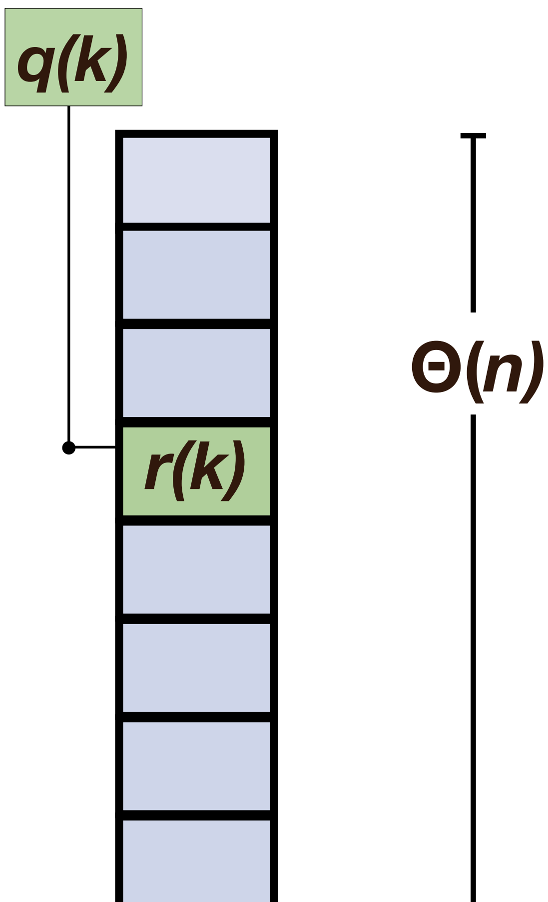


Bloom limitations
No deletes or counting
No resizes
No element enumeration or merging of filters
Keys have no values



Quotient Filter Capabilities

[Bender, Farach-Colton, Johnson, Kraner, Kuszmaul, Medjedovic, Montes, Shetty, Spillane, Zadok VLDB 12]
[Pandey, Bender, Patro, Johnson SIGMOD 17]



	Bloom performance
Space	$\approx 1.44\, n \lg(1/\varepsilon)$
CPU cost	$\Omega(\lg(1/\varepsilon))$
Data locality	$\Omega(\lg(1/\varepsilon))$ probes

QF performance

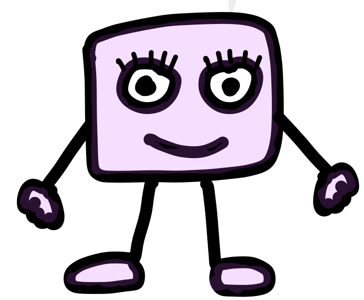
$(1+\text{tiny})n$

$\approx \cancel{1.44\, n}(\lg(1/\varepsilon) + 2)$

$O(1)$ expected

1 probe + scan

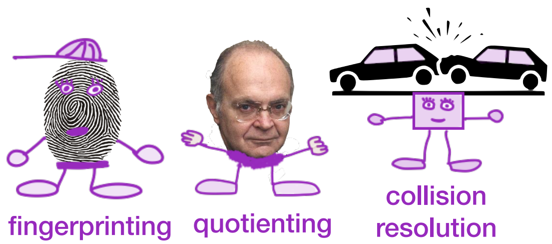
QF has practical + theoretical performance advantages.



Reminder of General Approach

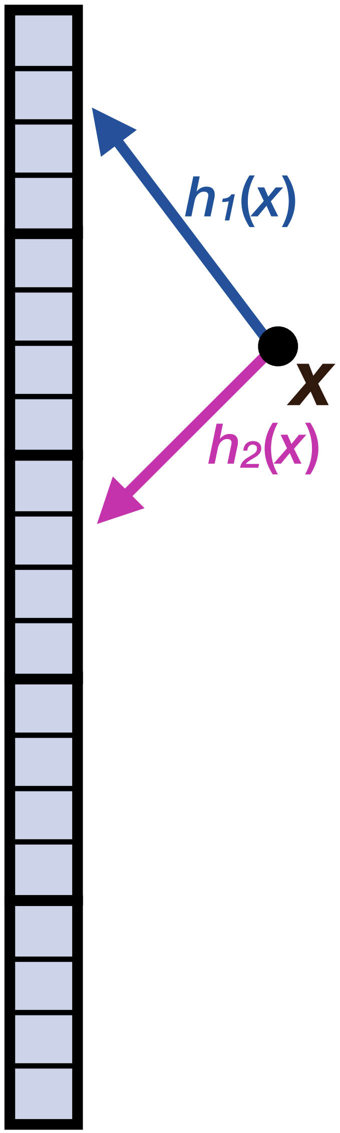
- **Fingerprinting**: key $k \in S \rightarrow h(k) \in F$.
  - False-positives only come from fingerprint collisions.
- **Quotienting**: Store fingerprints in a hash table implicitly.
  - $\lg n$ bits of fingerprint depend on hash location.
- **Collision resolution:**
  - E.g., using linear probing/Robinhood hashing.
  - Use metadata bits to recover each $h(k)$.

- **Designs alternatives:** cuckoo, Morton, broom
  - Use a different hash table but the same general approach



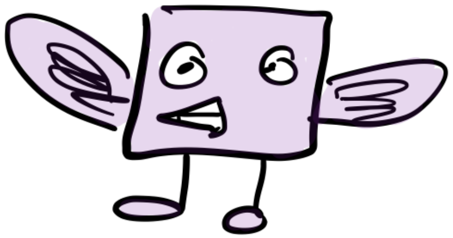
Cuckoo Hashing → Cuckoo Filters

[Pagh, Rodler 01] [Fan, Andersen, Kaminsky, Mitzenmacher 14] [Breslow, Jayasena 18]



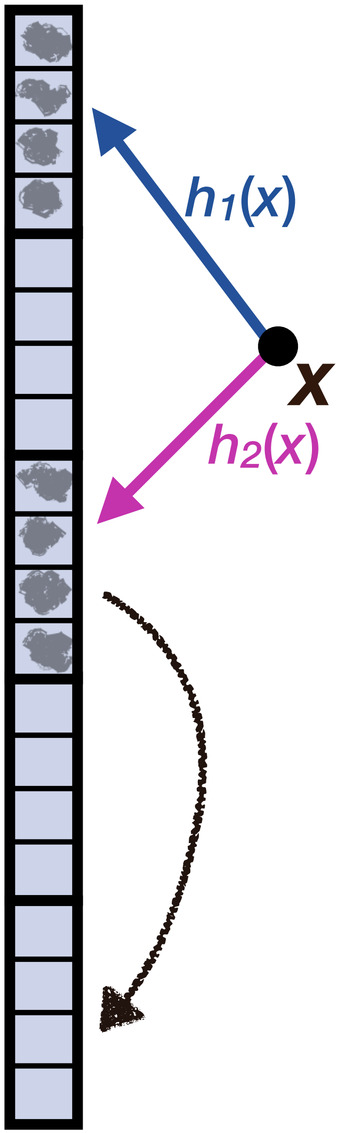
Cuckoo hash table has 2 **hash functions** h_1 and h_2.

Each hash bucket has **4 slots.**



Cuckoo Hashing → Cuckoo Filters

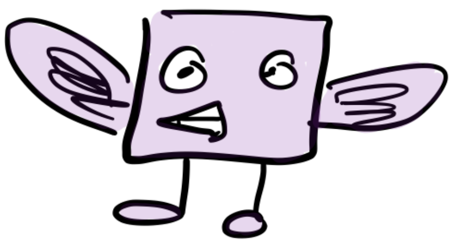
[Pagh, Rodler 01] [Fan, Andersen, Kaminsky, Mitzenmacher 14] [Breslow, Jayasena 18]



Cuckoo hash table has 2 **hash functions** h_1 and h_2.

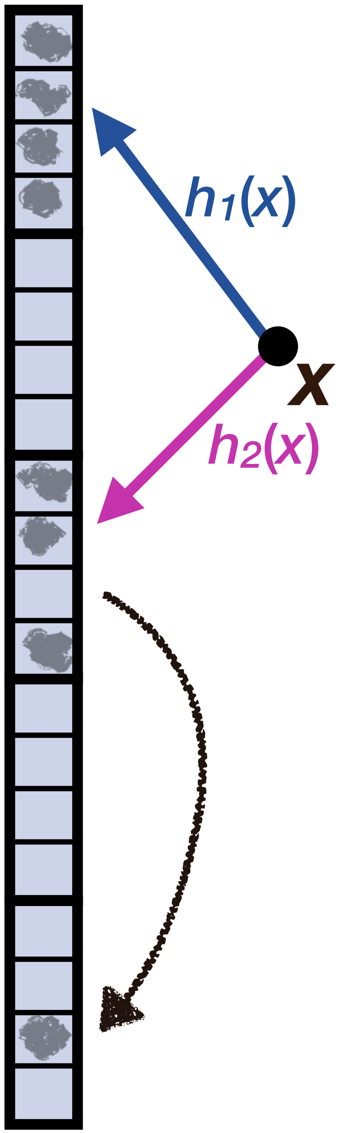
Each hash bucket has **4 slots.**

If there's no space in any of the 8 slots:
kick out an element, and
move it to the alternative location
(which may cause other kicks).



Cuckoo Hashing → Cuckoo Filters

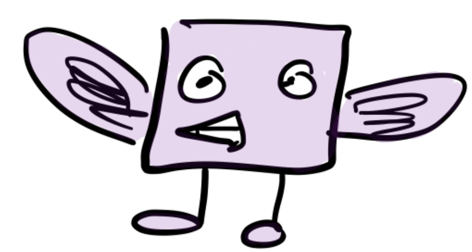
[Pagh, Rodler 01] [Fan, Andersen, Kaminsky, Mitzenmacher 14] [Breslow, Jayasena 18]



Cuckoo hash table has 2 **hash functions** h_1 and h_2.

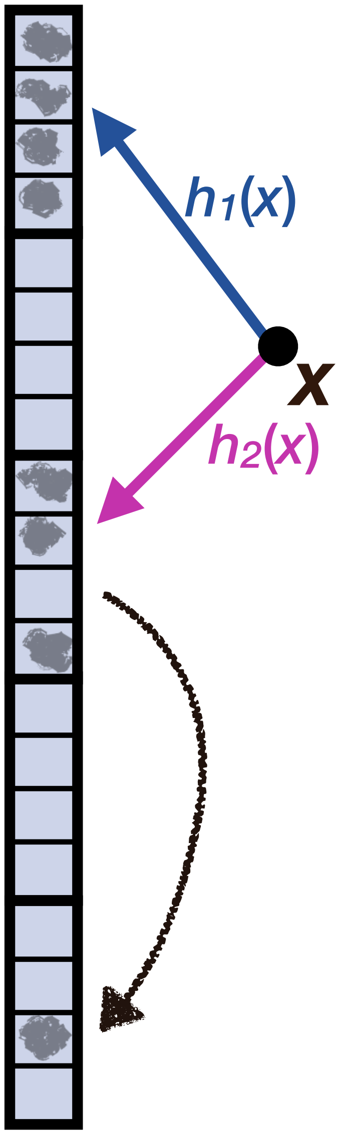
Each hash bucket has **4 slots.**

If there's no space in any of the 8 slots:
kick out an element, and
move it to the alternative location
(which may cause other kicks).



Obstacles for Cuckoo Filters

[Pagh, Rodler 01] [Fan, Andersen, Kaminsky, Mitzenmacher 14] [Breslow, Jayasena 18]



Q: If $f(x)$ is kicked, how to find an alternative location when we don't store x?

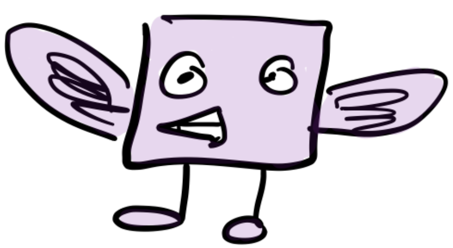
A: We give up on independent hash functions. The alternate location only depends on $r(x)$.

We also give up on asymptotic correctness.

Amazingly, it works for (practical) n not too large.

Cuckoo hashing seemingly doesn't have metadata bits.

But because there are 4 slots per cell,
2 more fingerprint bits are stored explicitly.



Quotient Filter and Cuckoo Filter Comparison

Quotient filter	Cuckoo filter
good space	good space
very good locality	ok locality
some degradation at high load factors	degradation at high load factors
good searches	very good searches
fast inserts	fast inserts
supports counting, multisets, values, deletes	
complicated code	code is easier
good for all n.	pre-asymptotic guarantees. fails w.h.p. large enough n

Optimal Theoretical Guarantees

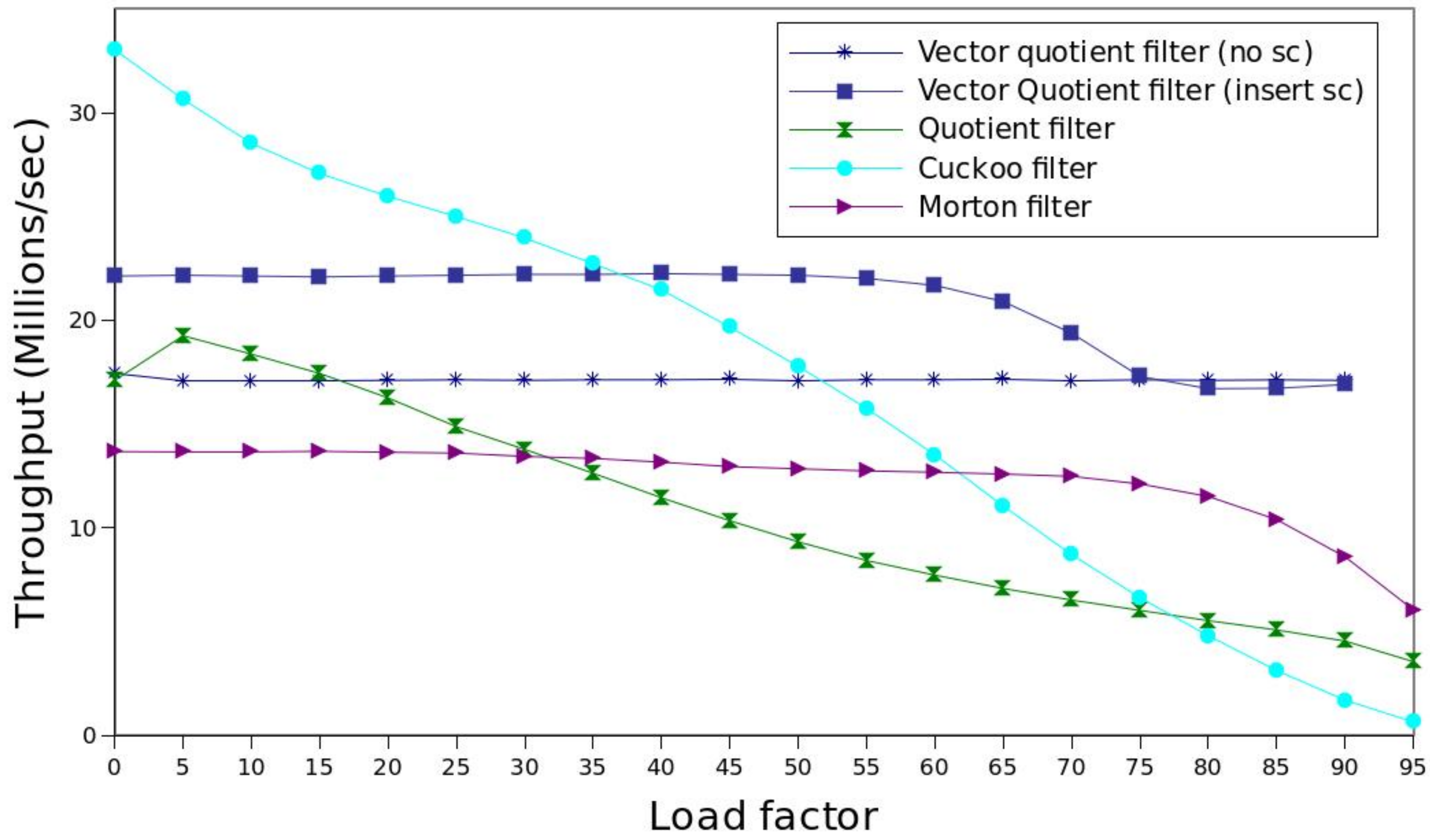
Theorem: There is an optimal filter with

- **Space:** $(1 + o(1))\, n \log(1/\varepsilon) + O(n)$
- **Error rate:** $\leq \varepsilon$
- **Operations: O(1) insert, delete, query.**

- [Pagh, Pagh, Rao 05]
- [Arbitman, Naor, and Segev 10]
- [Lovett & Porat 10]
- [Bender, Farach-Colton, Goswami, Johnson, McCauley, Singh 18] adaptive & worst case

Empirical Performance of a Vector Quotient Filter

[Pandey, Conway, Durie, Bender, Farach-Colton, Johnson 21

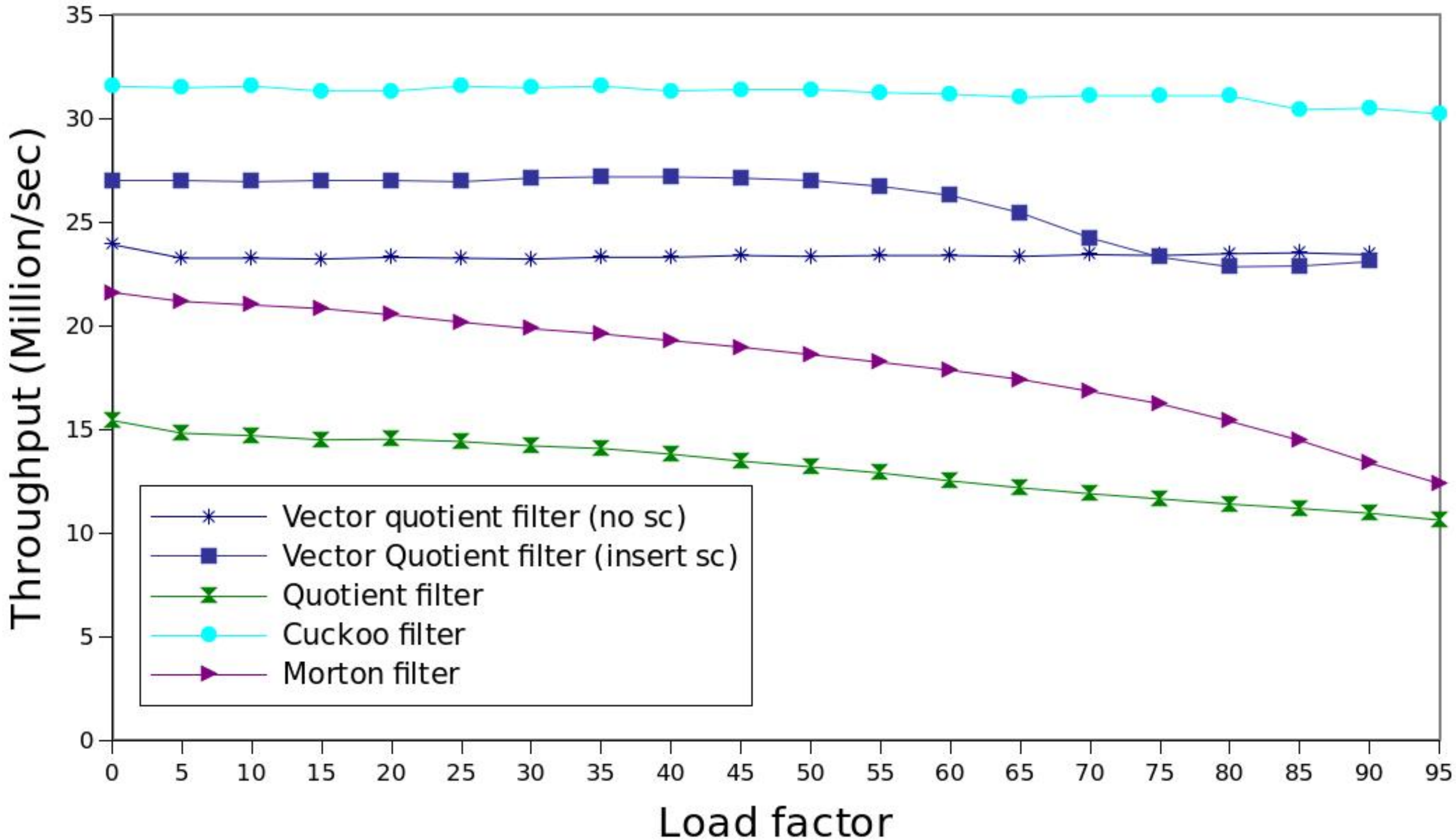


(a) Insertion (Higher is better.)

Quotient filter plus minimum of 2 choice retains good performance even when almost full.

Empirical Performance of a Vector Quotient Filter

[Pandey, Conway, Durie, Bender, Farach-Colton, Johnson 21



(c) Successful lookup (Higher is better.)

Cuckoo still kicks butt on queries.

Summery Slide

Summery Slide