Data Structures and Algorithms

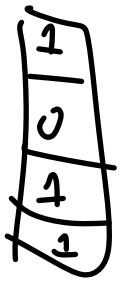
Bloom Filters

CS 225

Brad Solomon

April 27, 2026

The best data structure !!



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

More Logistic Announcements

Labs and MPs are both done! (This semester we had 10 labs)

This week's lab is optional (lab_ML)!

It is NOT an extra credit lab but will act as a direct replacement

(Your final lab grade will take the top 10 of these 11 labs)

Don't forget to take Exam 5 and your optional retake!

Retake is 70-30 averaging between higher and lower grade!

CA Hiring Application is out now (Opportunities Portal)

Learning Objectives

Build a conceptual understanding of a bloom filter

Review probabilistic data structures and one-sided error

Formalize the math behind the bloom filter

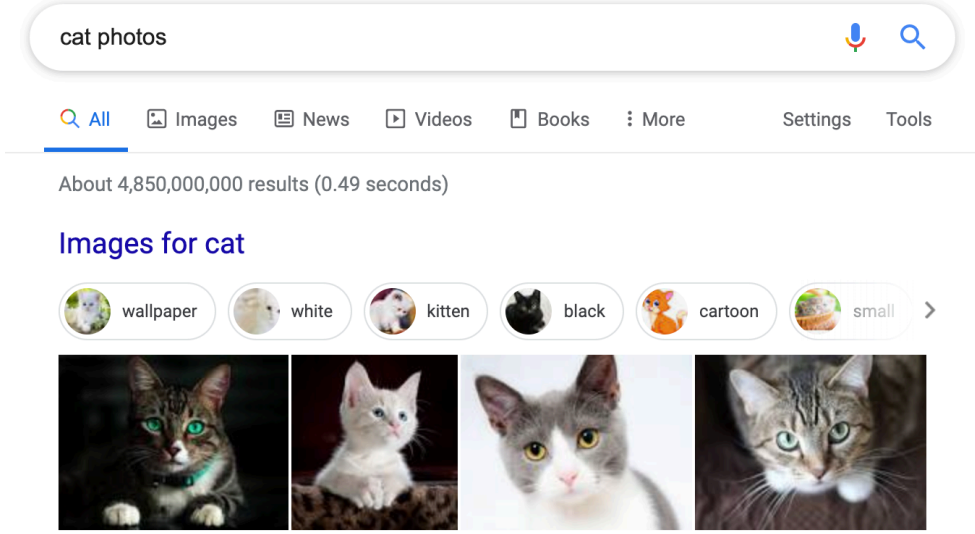
Review previous data structures

	Hash table	AVL Tree	Unsorted Array
Insert	O(n) / O(1) O(1)***	O(log n)	O(1)*
Find	O(n) O(1)***	O(log n)	O(N)
Size	O(n)	O(n)	O(n)

Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects *in a memory-constrained environment*?

Constrained by Big Data (Large N)



Google Index Estimate: >60 billion webpages

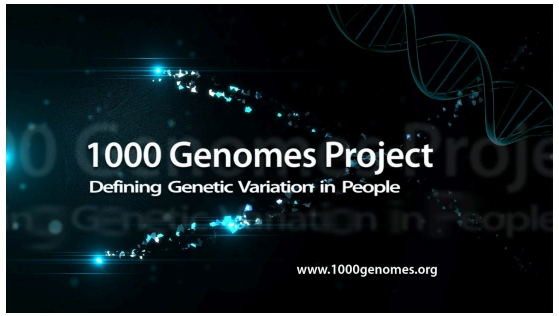
Google Universe Estimate (2013): >130 trillion webpages

Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects *in a memory-constrained environment*?

Constrained by Big Data (Large N)

ENA
European Nucleotide Archive



GTEx

SRA

Sequence Read Archive (SRA) makes biological sequence data available to the research community to enhance reproducibility and allow for new discoveries by comparing data sets. The SRA stores raw sequencing data and alignment information from high-throughput sequencing platforms, including Roche 454 GS System®, Illumina Genome Analyzer®, Applied Biosystems SOLiD System®, Helicos Heliscope®, Complete Genomics®, and Pacific Biosciences SMRT®.

Sequence Read Archive Size: >60 petabases (10^{15})

Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects *in a memory-constrained environment*?

Constrained by Big Data (Large N)

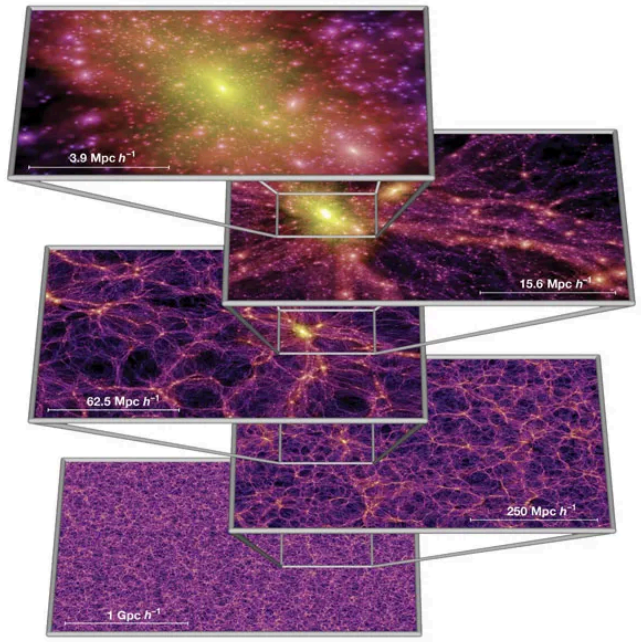


Image: https://doi.org/10.1038/nature03597

Sky Survey Projects	Data Volume
DPOSS (The Palomar Digital Sky Survey)	3 TB
2MASS (The Two Micron All-Sky Survey)	10 TB
GBT (Green Bank Telescope)	20 PB
GALEX (The Galaxy Evolution Explorer)	30 TB
SDSS (The Sloan Digital Sky Survey)	40 TB
SkyMapper Southern Sky Survey	500 TB
PanSTARRS (The Panoramic Survey Telescope and Rapid Response System)	~ 40 PB expected
LSST (The Large Synoptic Survey Telescope)	~ 200 PB expected
SKA (The Square Kilometer Array)	~ 4.6 EB expected

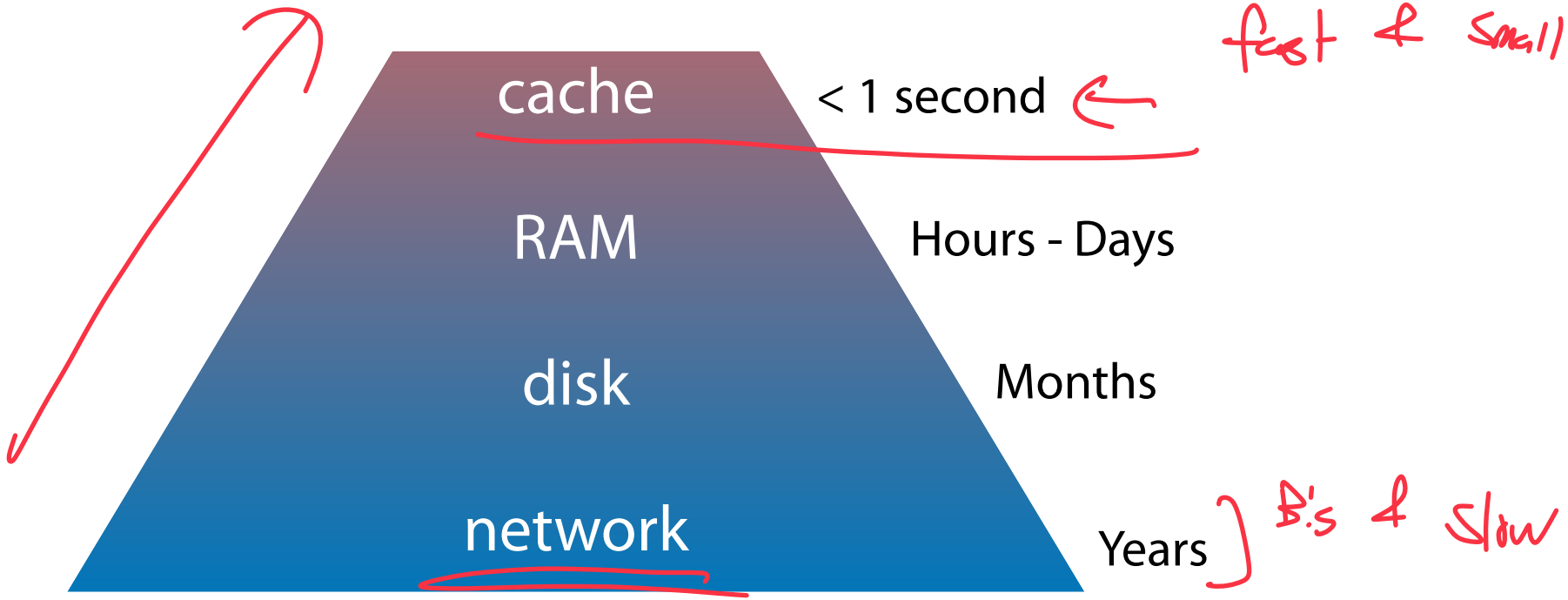
Table: http://doi.org/10.5334/dsj-2015-011

Estimated total volume of one array: 4.6 EB

Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects *in a memory-constrained environment*?

Constrained by resource limitations

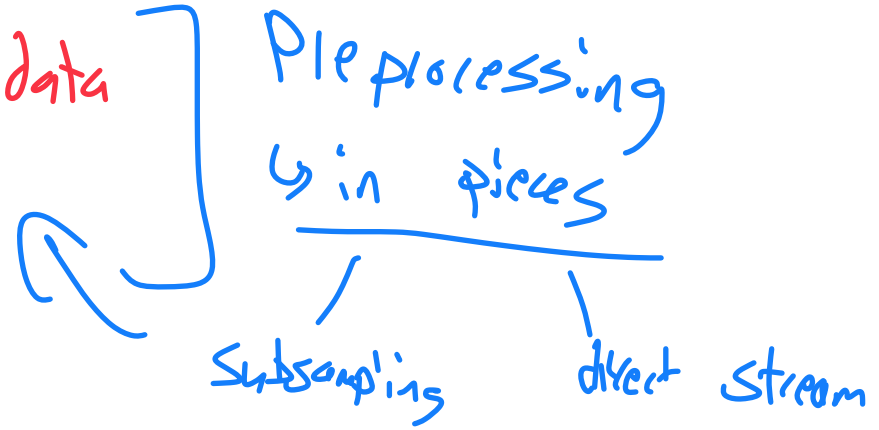


(Estimates are Time x 1 billion courtesy of https://gist.github.com/hellerbarde/2843375)

Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects *in a memory-constrained environment*?

1) Try to find repeated patterns in data



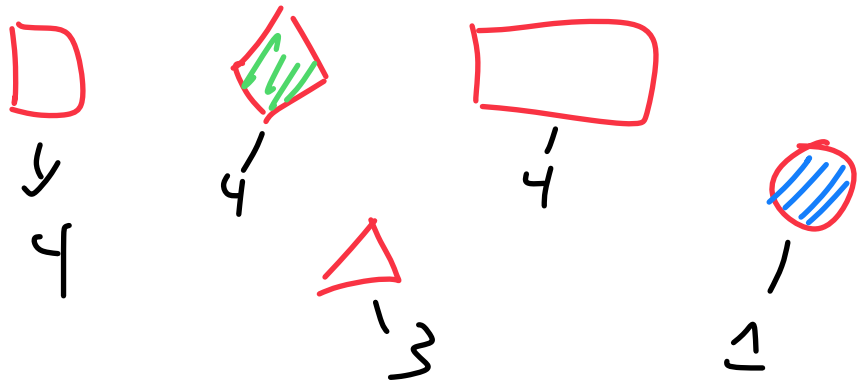
2) Compress data into smaller space

Reducing storage costs

preprocess is a kind of this

1) Throw out information that isn't needed

↳ # of shapes w/ 4 sides



Lossy Data compression

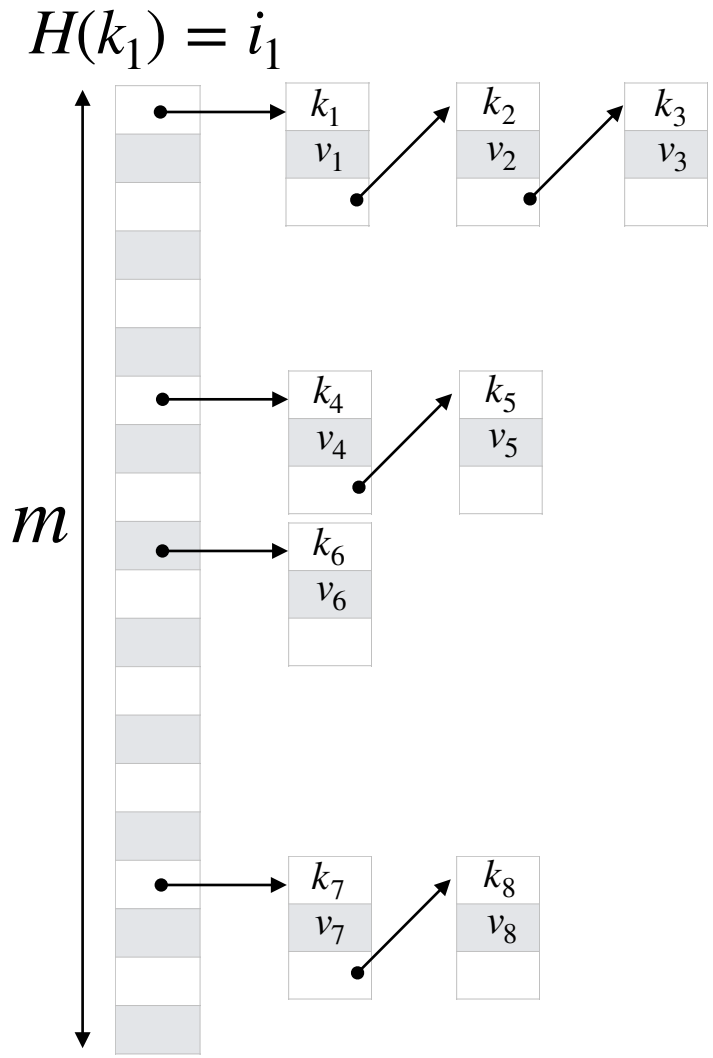
2) Compress the dataset

"AAAABBB" = 4A 3B

Lossless compression

Reducing a hash table

What can we remove from a hash table?

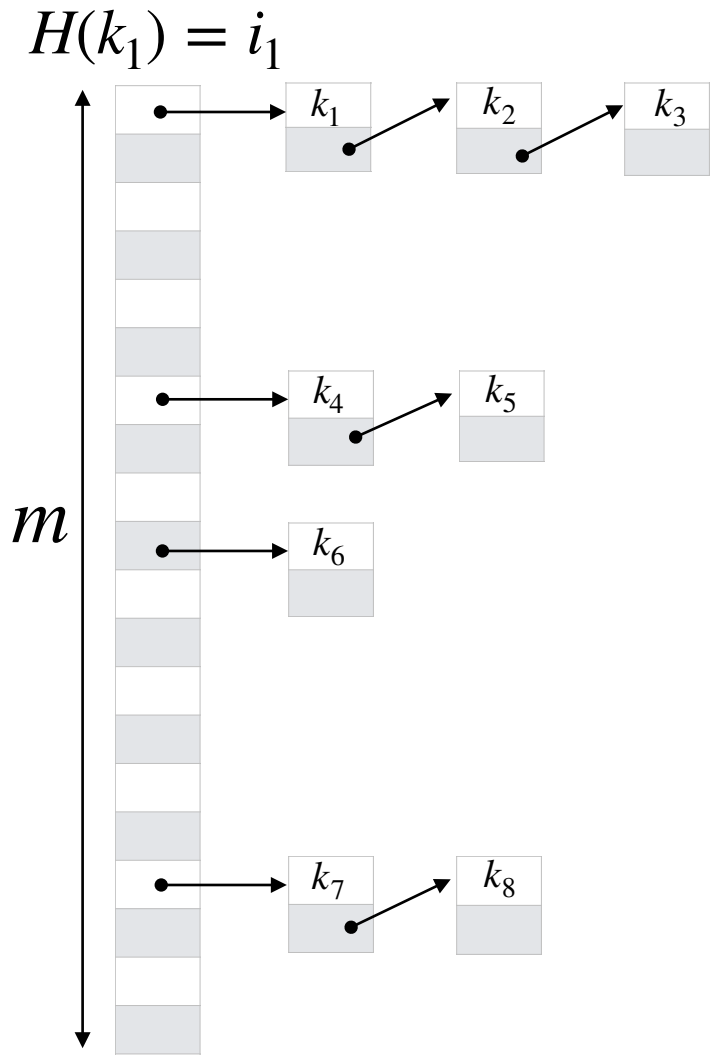


Reducing a hash table

An implementation of a set

What can we remove from a hash table?

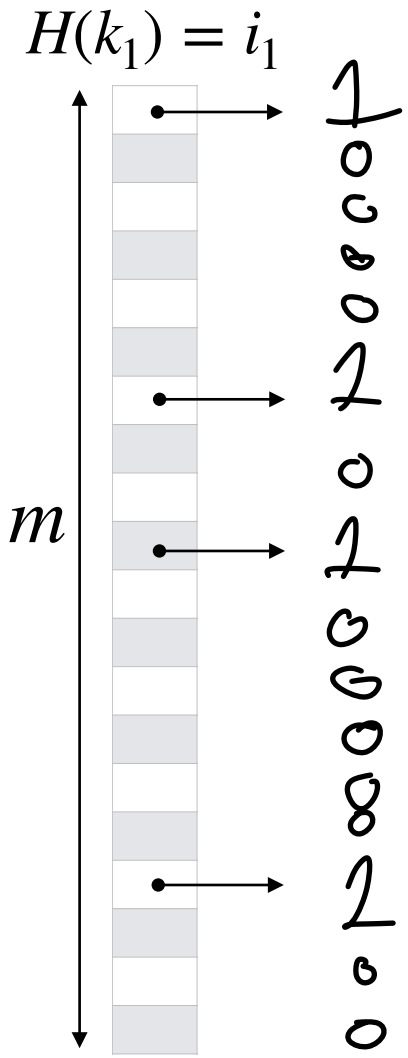
Take away values



Reducing a hash table

What can we remove from a hash table?

Take away values and keys



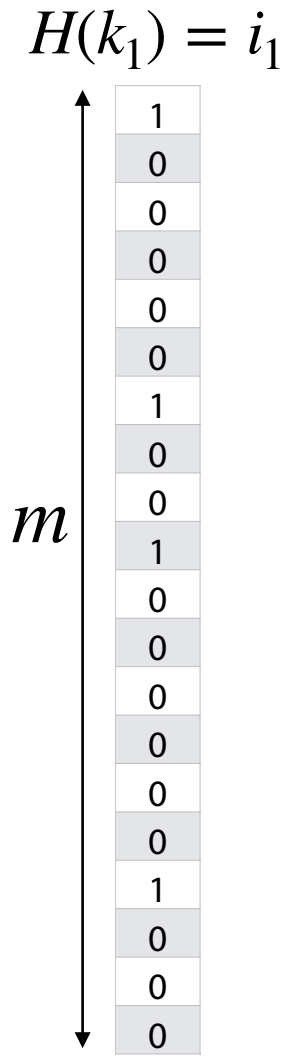
if anything hashed here ever

Reducing a hash table

What can we remove from a hash table?

Take away values and keys

This is a ***bloom filter***



Bloom Filter ADT

Constructor

↳ Give one or more hash functions
↳ Input array size

Insert (Add item)

Find (Does item exist?)

Bloom Filter: Insertion

S = { 16, 8, 4, 13, 29, 11, 22 }

h(k) = k % 7

0	0
1	~~0~~ 1
2	~~0~~ 1
3	0
4	~~0~~ 1
5	0
6	~~0~~ 1

29 % 7 = 1

1) Hash each item

2) Insert by setting bit at hash value to 1

If collision, ignore.
Bit is already 1

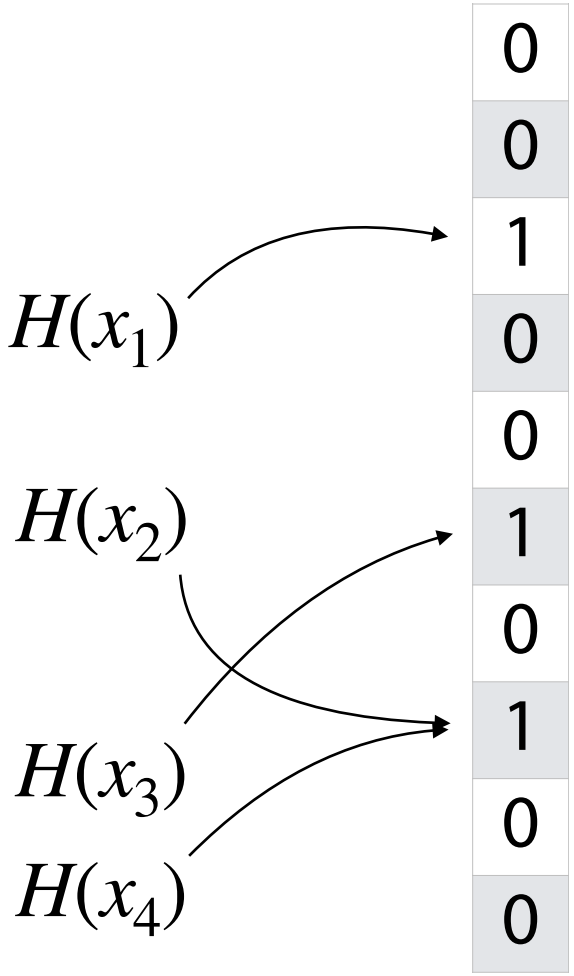
(Any value set to 1 stays 1)

O(1)

Bloom Filter: Insertion

An item is inserted into a bloom filter by hashing and then setting the hash-valued bit to 1

If the bit was already one, it stays 1



Bloom Filter: Deletion

S = { 16, 8, 4, 13, 29, 11, 22 }

h(k) = k % 7

0	0
1	~~1~~ 0
2	1
3	0
4	1
5	0
6	~~1~~ 0

_delete(13)

1) Hash item

2) Remove item at location

→ _delete(29)

- find(8)

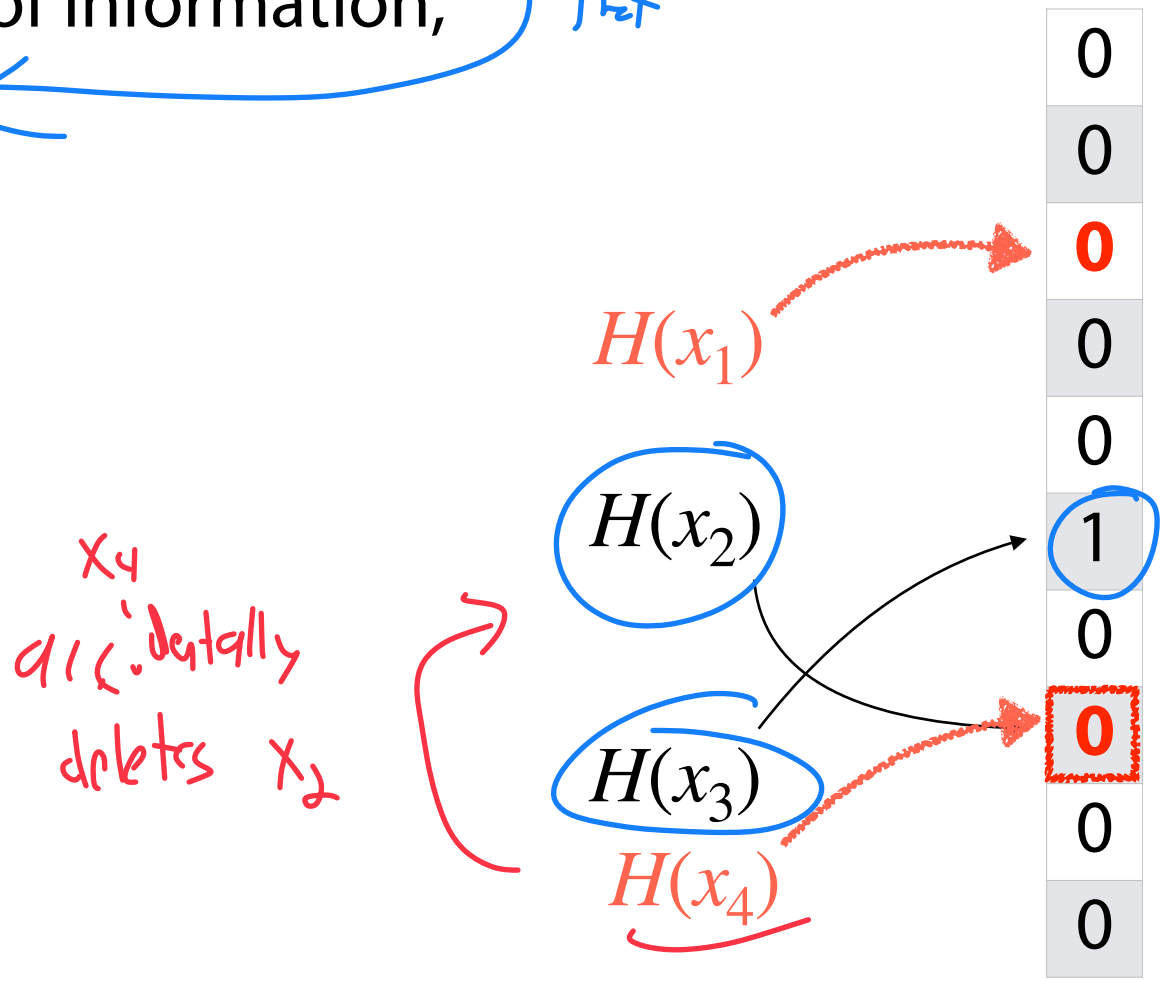
Bloom Filter: Deletion

Tradeoff:

we don't track

Due to hash collisions and lack of information, items cannot be deleted!

b/c of that



x_4 accidentally deletes x_2

Rule: No deletion! Bit set to 1 stays 1 forever

Bloom Filter: Search

S = { 16, 8, 4, 13, 29, 11, 22 }

h(k) = k % 7

0	0
1	1
2	1
3	0
4	1
5	0
6	1

_find(16) Yes!

1) Hash item

2) Lookup value of bit

Does it exist?

1 = Yes

0 = No

20%7 = 6 _find(20) - Yes!

↳ Probabilistic accuracy!

34%7 = 3 _find(3) - No!

Bloom Filter: Search

The bloom filter is a *probabilistic* data structure!

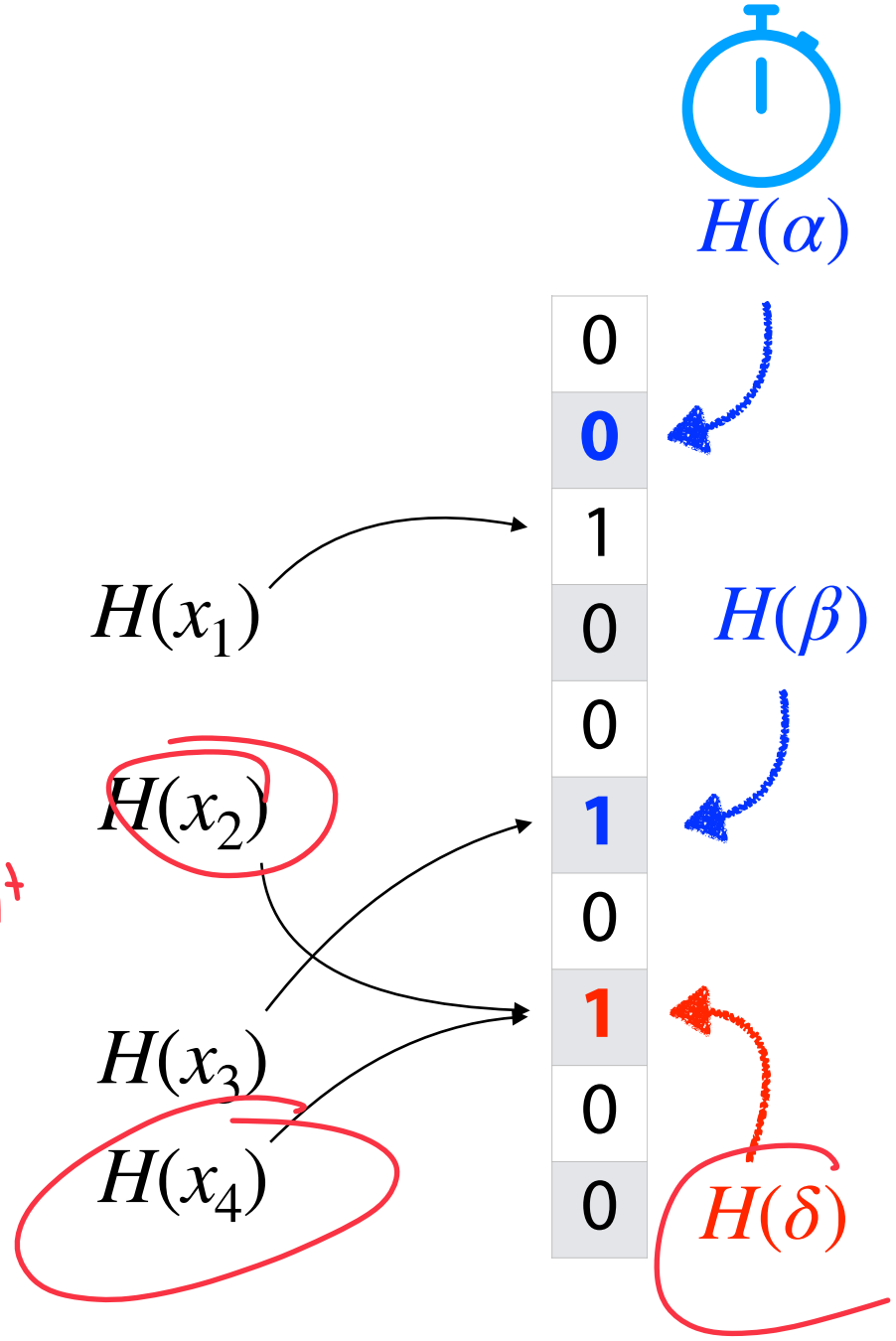
If the value in the BF is 0: 100% chance item not in dataset

If ○ no items in universe hashed here

If the value in the BF is 1: item might be present

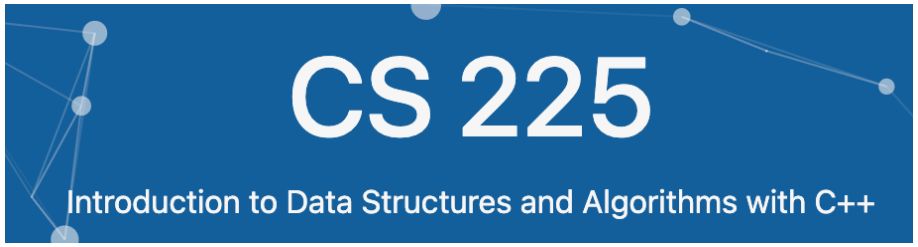
It could be a collision!

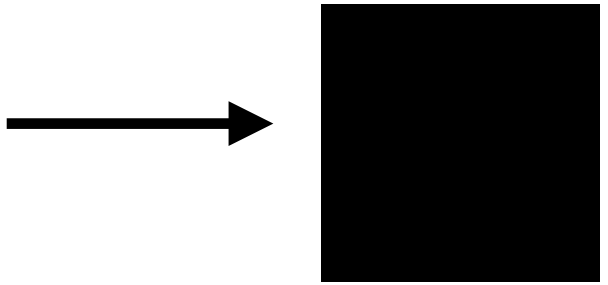
How likely is this error?
Bad is this error?



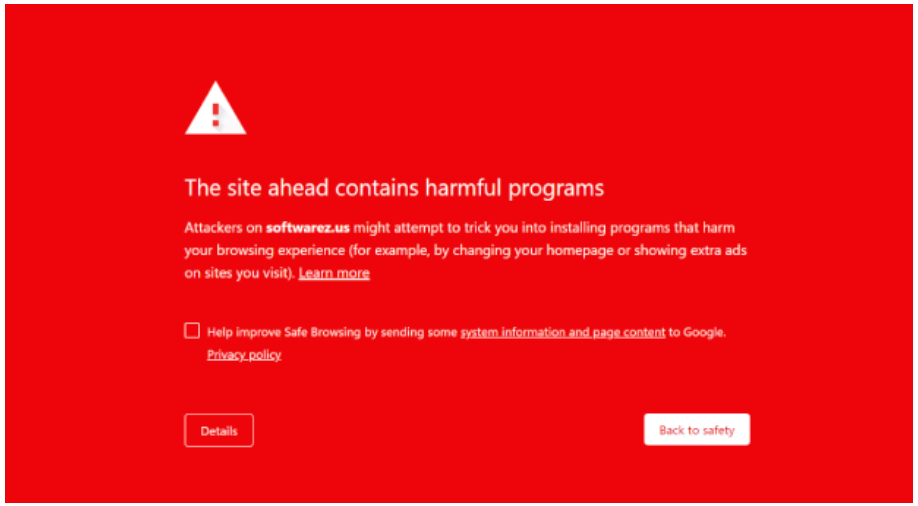
Probabilistic Accuracy: Malicious Websites

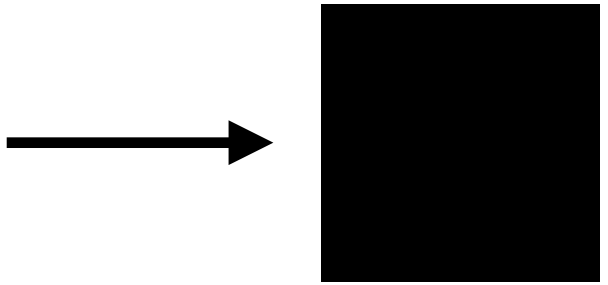
Imagine we have a detection oracle that identifies if a site is malicious





"Not malicious"





"Malicious"

This is safer if unsure!

Probabilistic Accuracy: Malicious Websites

Imagine we have a detection oracle that identifies if a site is malicious

True Positive: Oracle says M, website is M

False Positive: Oracle says M, website not M

↳ At best an annoyance

False Negative: Oracle says not M (safe), website is M

↳ A major issue!

True Negative: Oracle says not M, website is not M

Imagine we have a **bloom filter** that **stores malicious sites…**

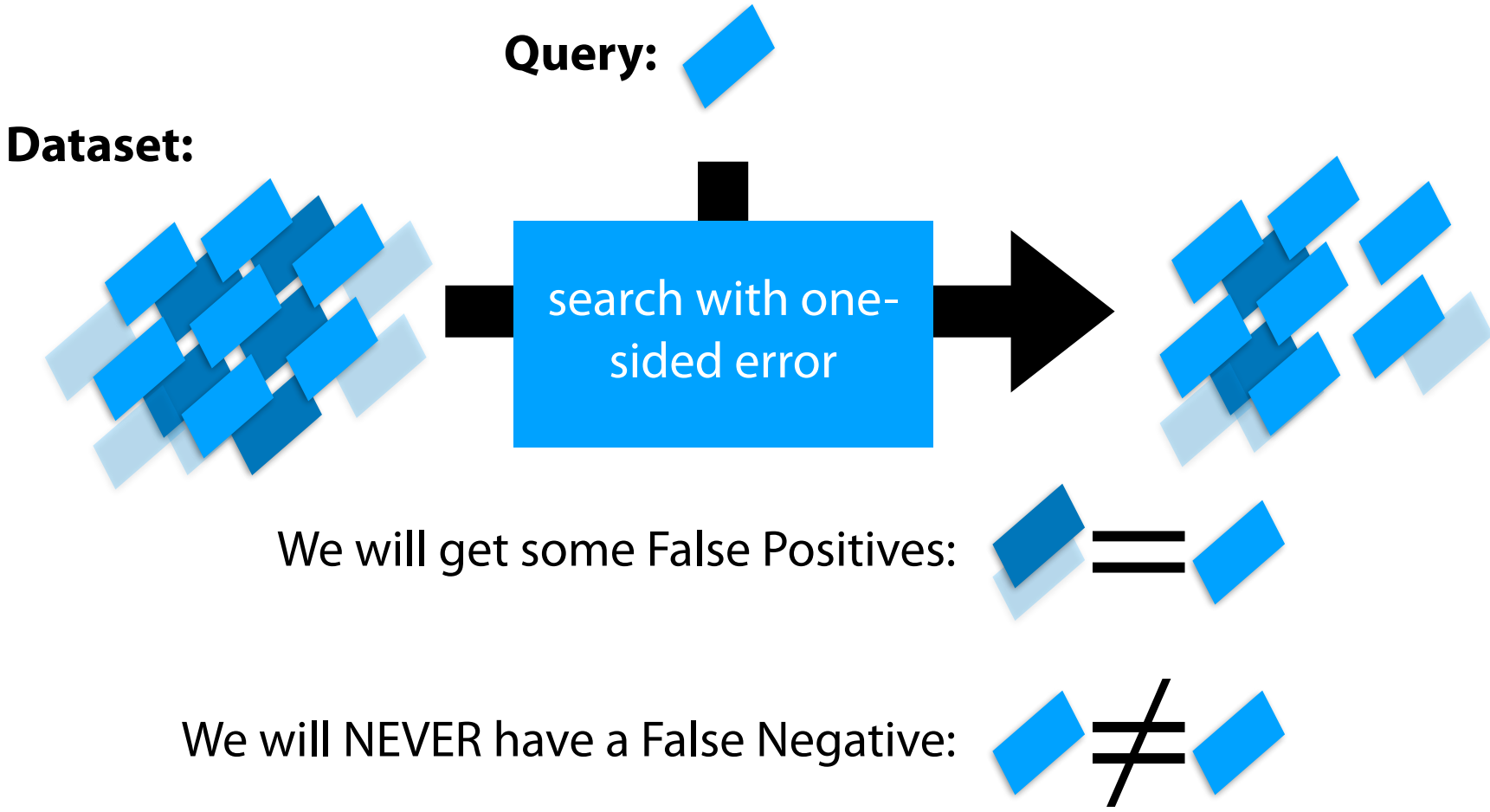
	Bit Value = 1	**Bit Value = 0**
Item Inserted	$H(z)$ → 0, **1**, 0, 0, 1 — **'Yes'** — True Positive	$H(z)$ → 0, **0**, 0, 0, 1 — **'No'** — False Negative (75%)
Item NOT inserted	0, **1**, 0, 0, 1 — **'Yes'** — False Positive (hash collision)	0, **0**, 0, 0, 1 — **'No'** — True Negative (20%)

This cannot happen! 1 stays 1 forever

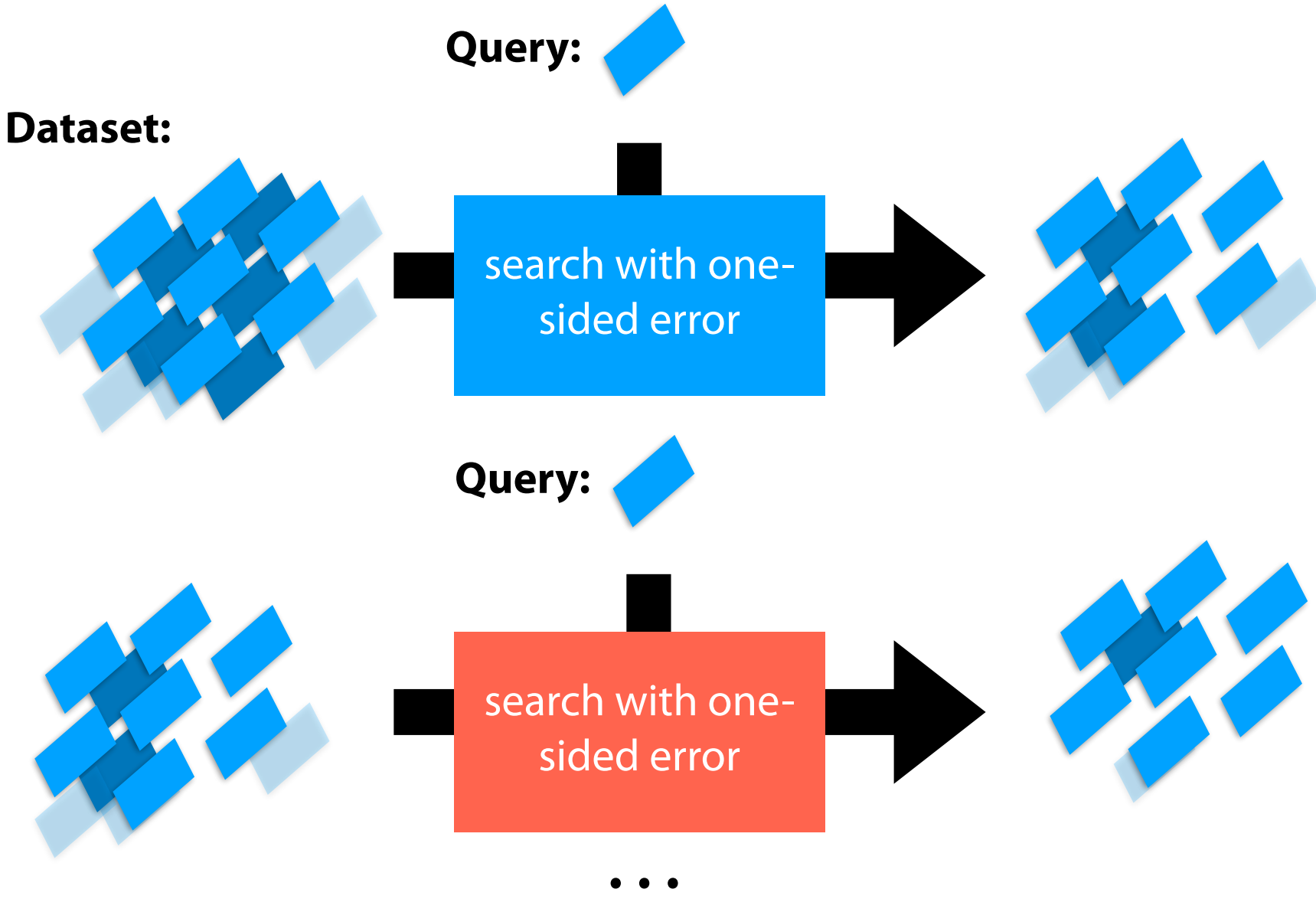
We will finish on Wednesday if sucss!

Tornado interrupt :(

Probabilistic Accuracy: One-sided error



Probabilistic Accuracy: One-sided error

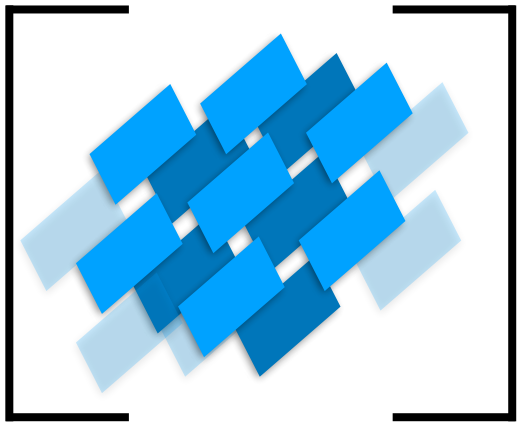


Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter

0
1
0
1
0
0
0
1
0
1
1
0
1
0
1
1
0
1
0
1

h_1



Bloom Filter: Repeated Trials

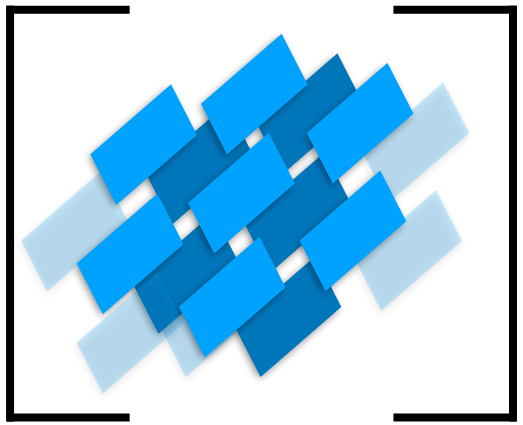
Use many hashes/filters; add each item to each filter

h_1

0
1
0
1
0
0
0
1
0
1
1
0
1
0
1
1
0
1
0
1

0
0
0
1
0
0
0
1
0
1
0
0
1
1
1
0
0
1
0
0

h_2



Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter

h_1

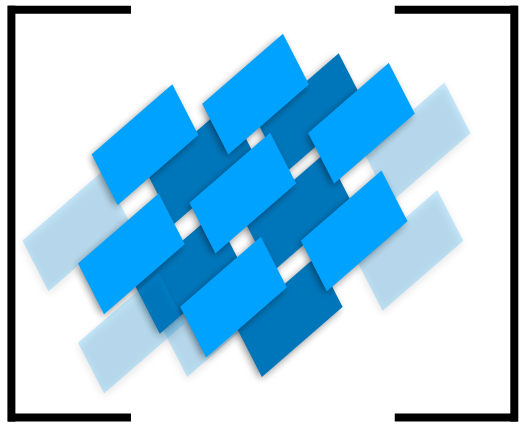
0
1
0
1
0
0
0
1
0
1
1
0
1
0
1
1
0
1
0
1

h_2

0
0
0
1
0
0
0
1
0
1
0
0
1
1
1
0
0
1
0
0

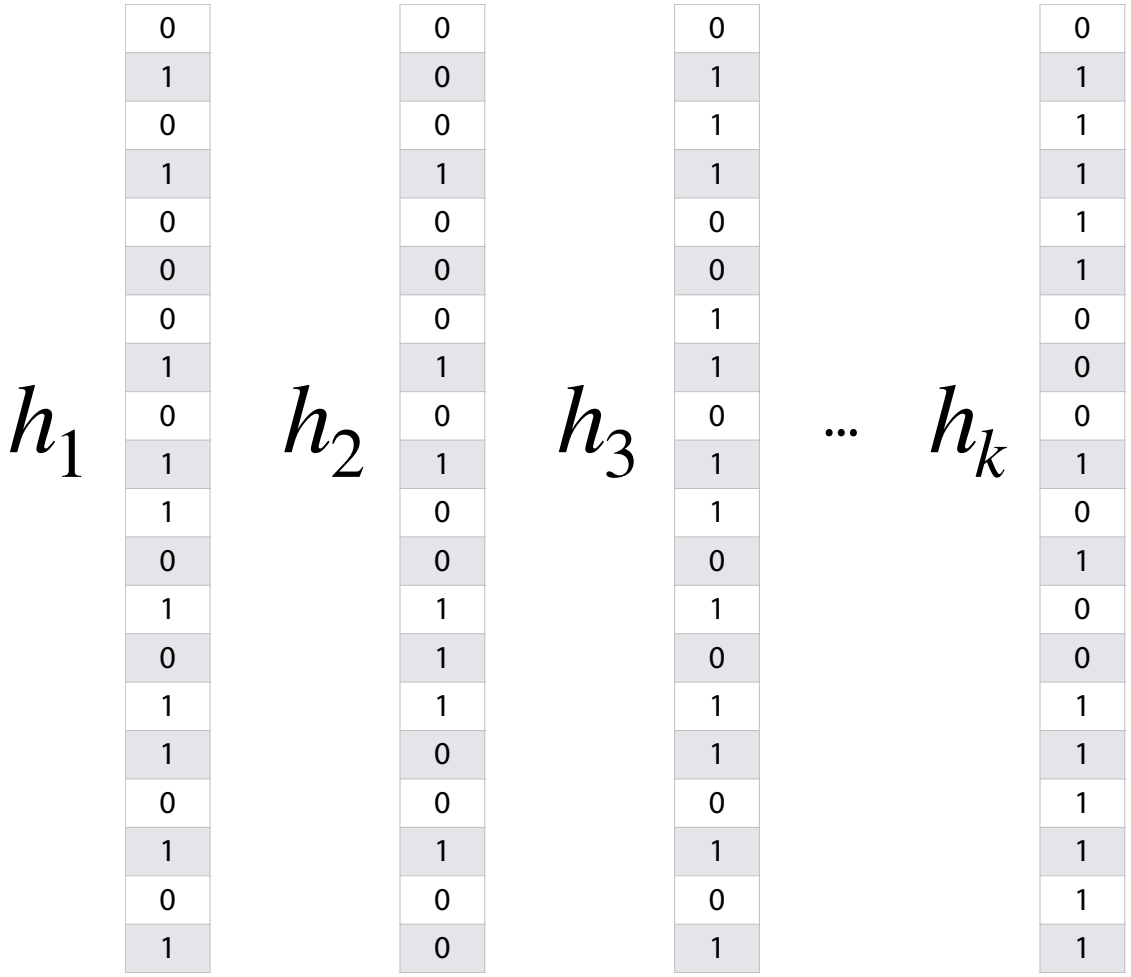
h_3

0
1
1
1
0
0
1
1
0
1
1
0
1
0
1
1
0
1
0
1

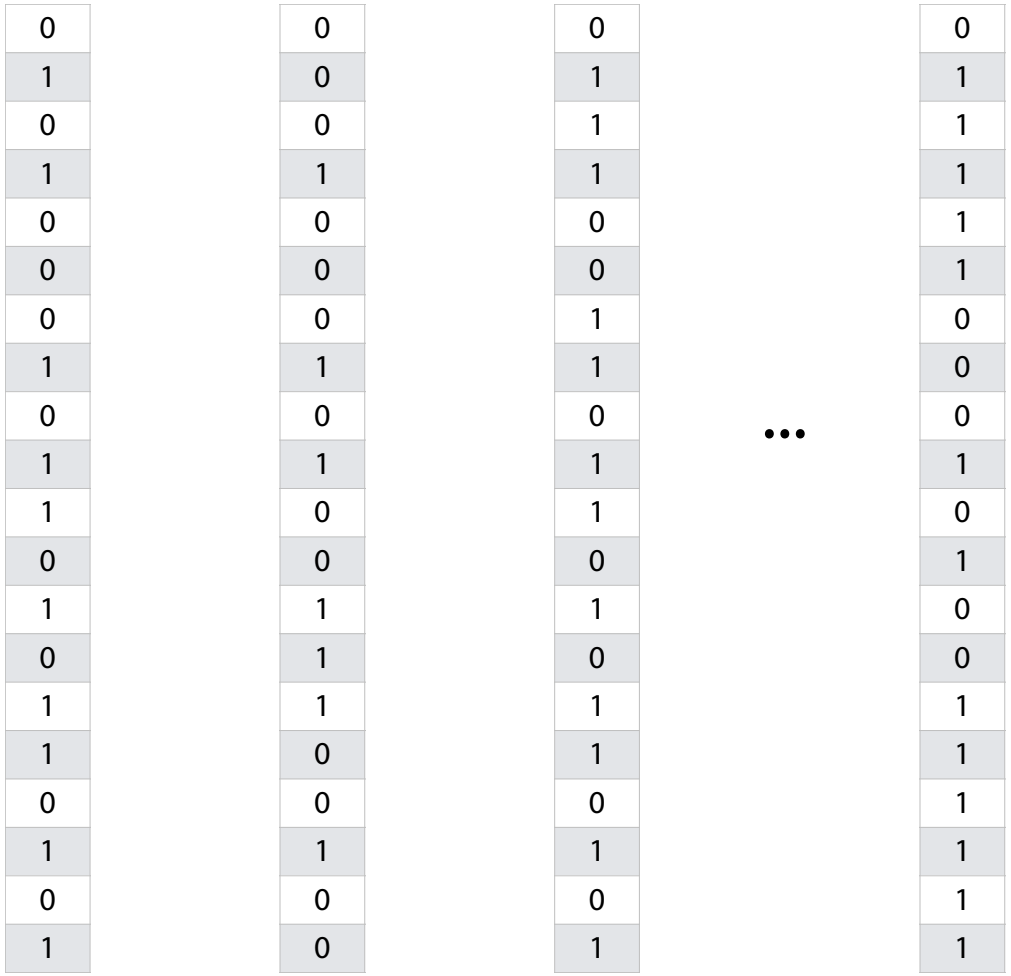


Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter

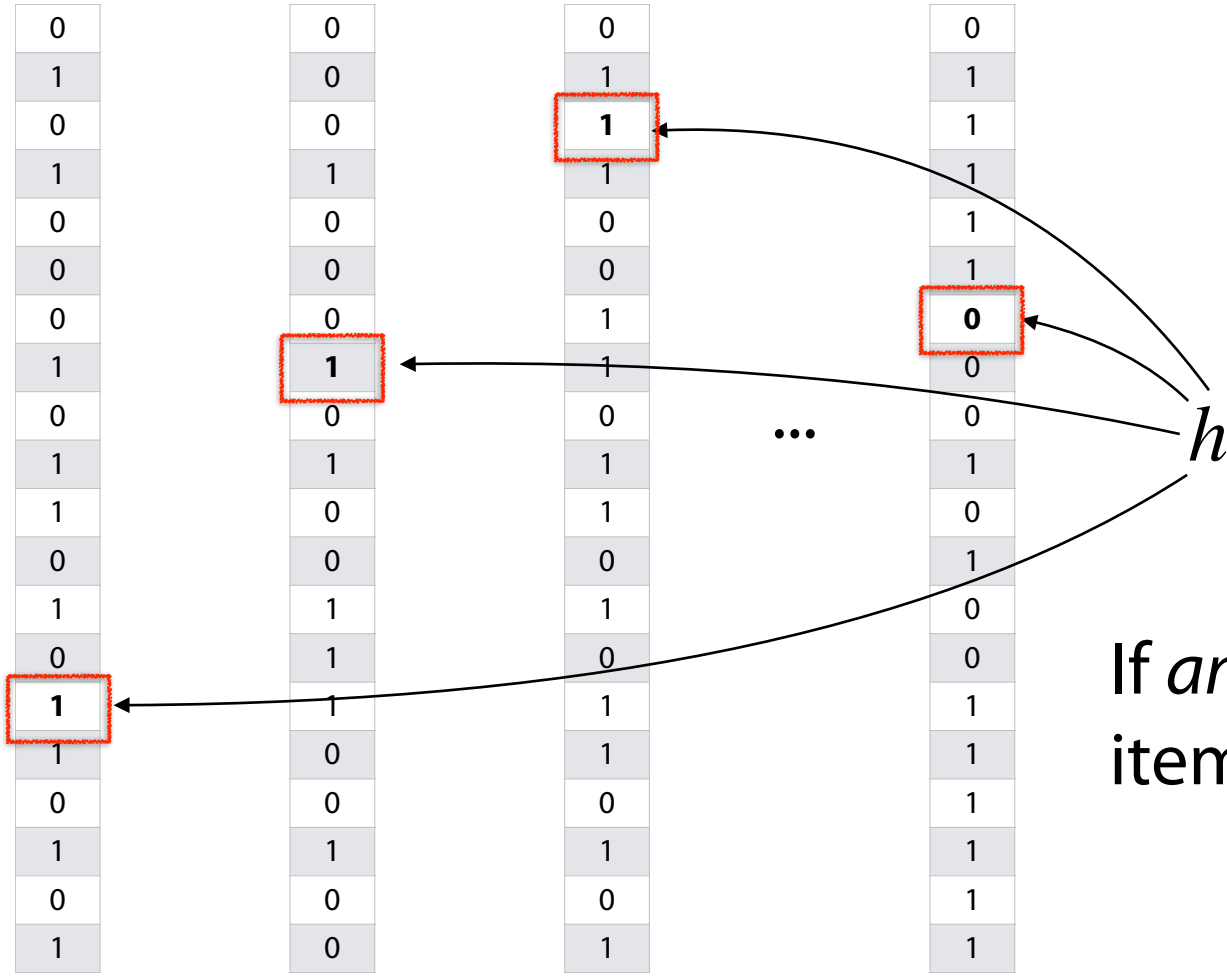


Bloom Filter: Repeated Trials



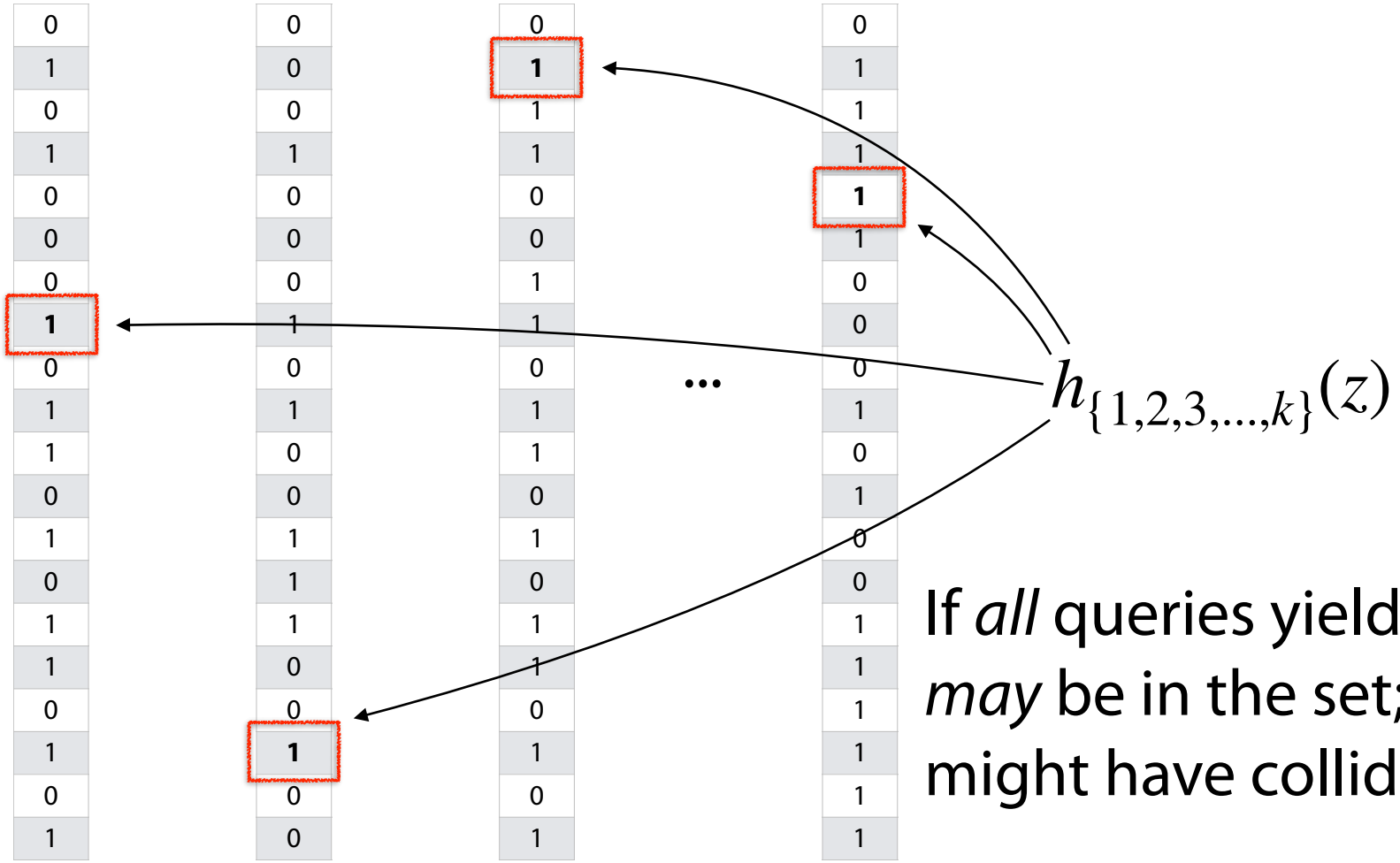
$$h_{\{1,2,3,\ldots,k\}}(y)$$

Bloom Filter: Repeated Trials



If *any* query yields 0, item is not in the set

Bloom Filter: Repeated Trials



If *all* queries yield 1, item *may* be in the set; or we might have collided *k* times

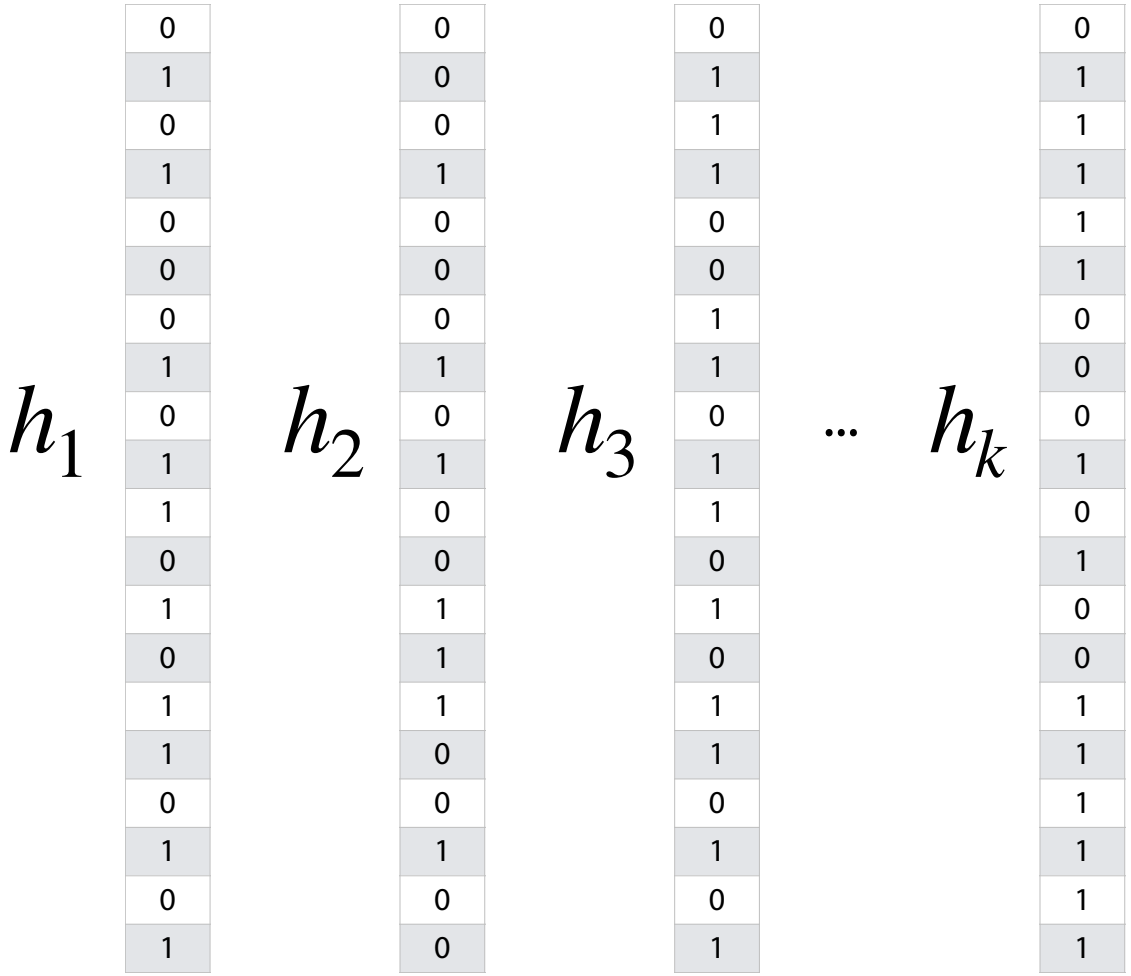
Bloom Filter: Repeated Trials

Using repeated trials, even a very bad filter can still have a very low FPR!

If we have k bloom filter, each with a FPR p, what is the likelihood that ***all*** filters return the value '1' for an item we didn't insert?

Bloom Filter: Repeated Trials

But doesn't this hurt our storage costs by storing k separate filters?



Bloom Filter: Repeated Trials

Rather than use a new filter for each hash, one filter can use k hashes

S = { 6, 8, 4 }

$h_1(x) = x \% 10$ **$h_2(x) = 2x \% 10$** **$h_3(x) = (5+3x) \% 10$**

Bloom Filter: Repeated Trials

Rather than use a new filter for each hash, one filter can use k hashes

0	0
1	0
2	1
3	1
4	1
5	0
6	1
7	1
8	1
9	1

$h_1(x) = x \% 10$ **$h_2(x) = 2x \% 10$** **$h_3(x) = (5+3x) \% 10$**

```
_find(1)
```

```
_find(16)
```

Bloom Filter

$H = \{h_1, h_2, \ldots, h_k\}$

A probabilistic data structure storing a set of values

Built from a bit vector of length m and k hash functions

Insert / Find runs in: _____

Delete is not possible (yet)!

0
0
1
0
0
1
0
1
0
0