

Data Structures and Algorithms

Bloom Filters

CS 225

November 18, 2024

Brad Solomon



UNIVERSITY OF
ILLINOIS
URBANA - CHAMPAIGN

Department of Computer Science

Announcements

MP_mosaic survey EC reached

MP_traversal survey EC **not reached** (Have until 11/20 to submit!)

MP_puzzle released, due after break. **Break doesn't count as a week**

Learning Objectives

Review when you would prefer different data structures

Build a conceptual understanding of a bloom filter

Review probabilistic data structures and one-sided error

Formalize the math behind the bloom filter

Which collision resolution strategy is better?

- Big Records:
- Structure Speed:

What structure do hash tables implement?

What constraint exists on hashing that doesn't exist with BSTs?

Why talk about BSTs at all?

Running Times

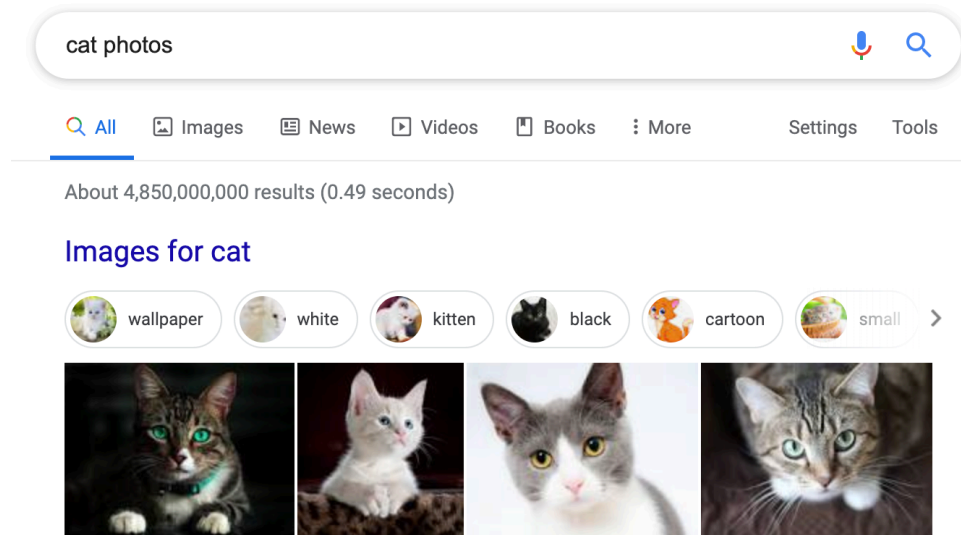


	Hash Table	AVL	Linked List
Find	Expectation*: $O(1)^{***}$ Worst Case: $O(n)$	$O(\log n)$	$O(n)$
Insert	Expectation*: $O(1)^{***}$ Worst Case: $O(n)$	$O(\log n)$	$O(1)$
Storage Space	$O(n)$	$O(n)$	$O(n)$

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)



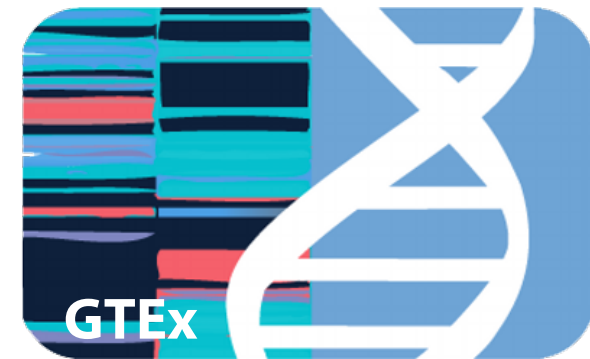
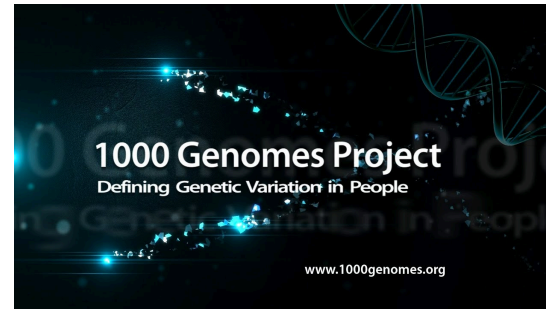
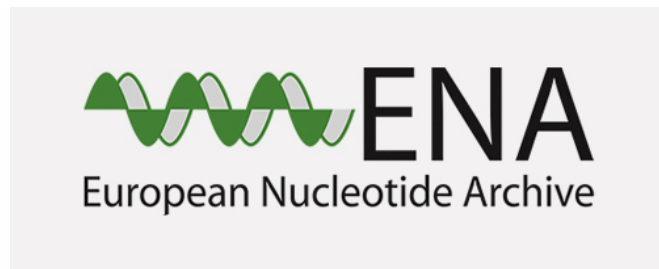
Google Index Estimate: >60 billion webpages

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)



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)

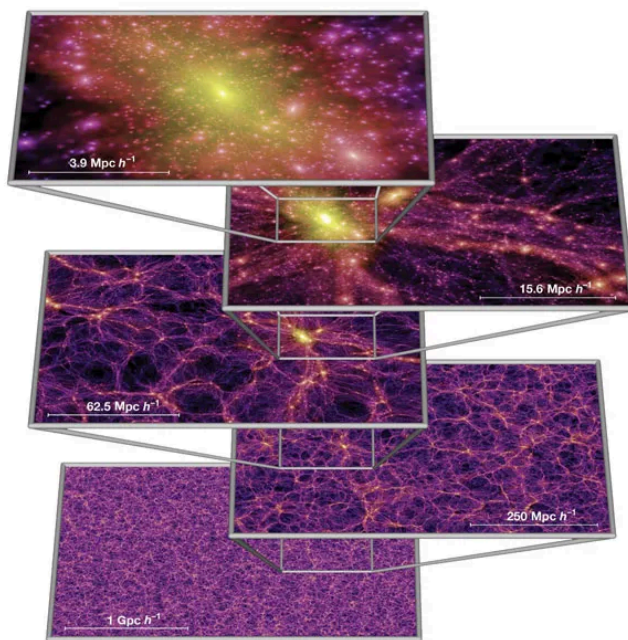


Image: <https://doi.org/10.1038/nature03597>

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

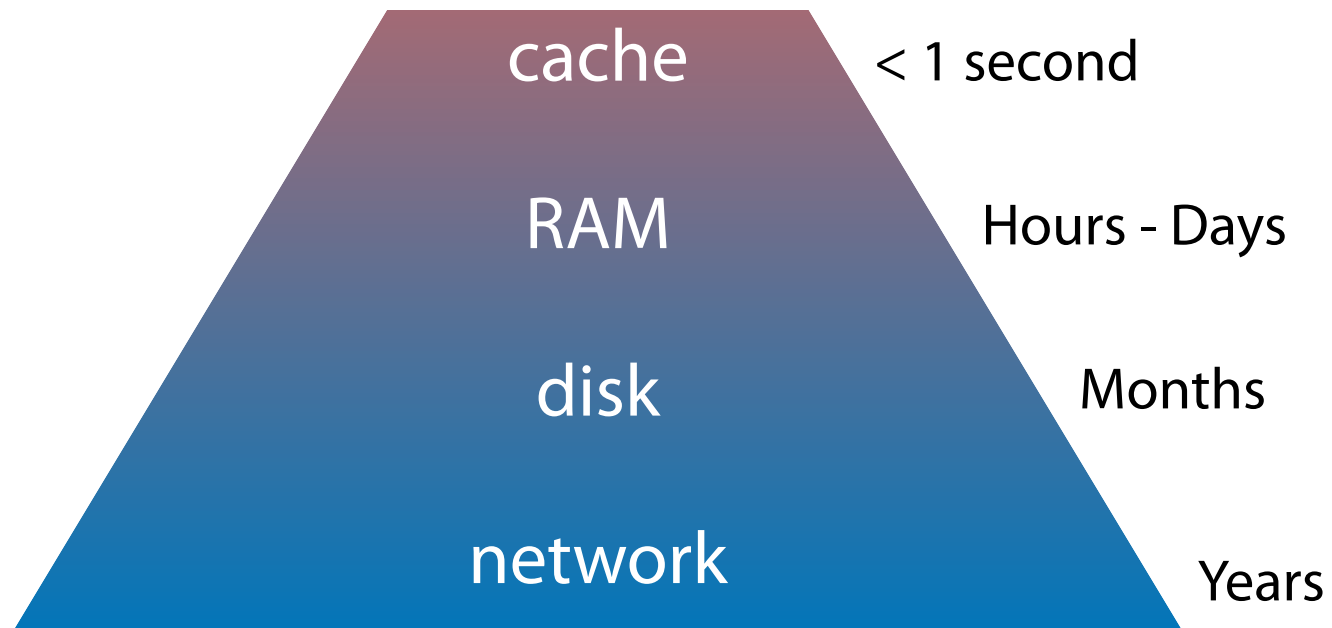
Table: <http://doi.org/10.5334/dsj-2015-011>

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*?

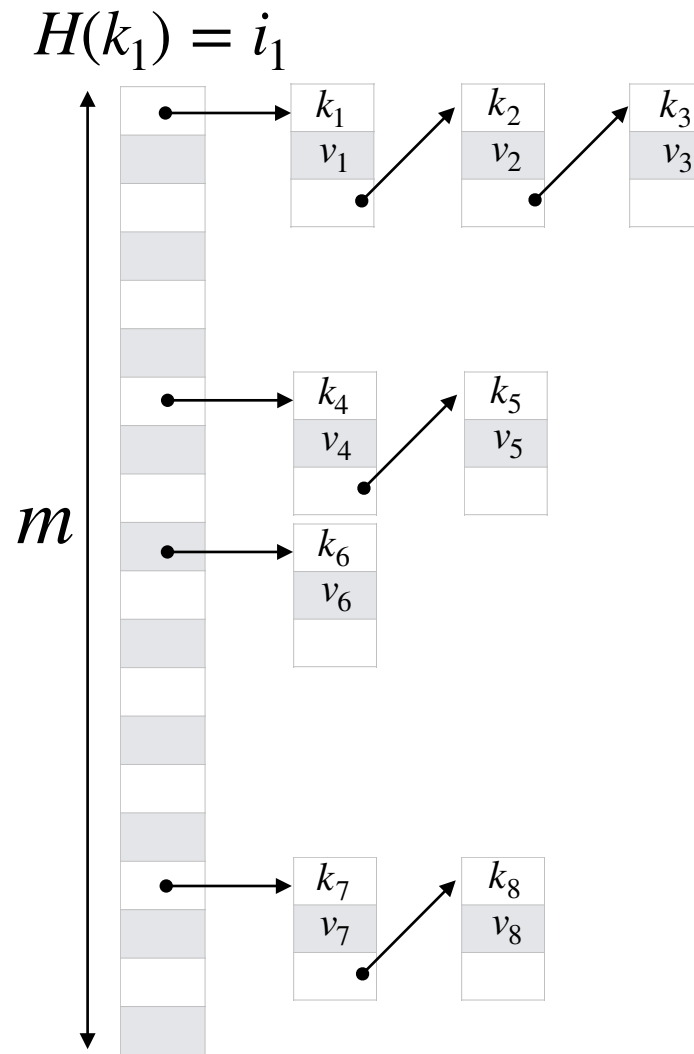
Reducing storage costs

1) Throw out information that isn't needed

2) Compress the dataset

Reducing a hash table

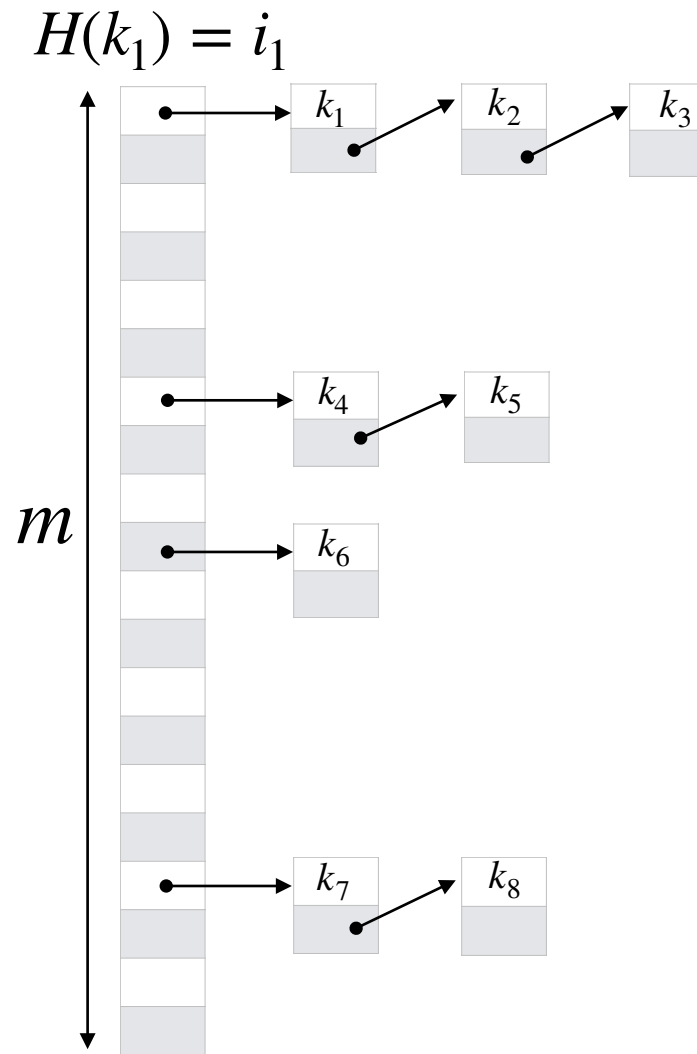
What can we remove from a hash table?



Reducing a hash table

What can we remove from a hash table?

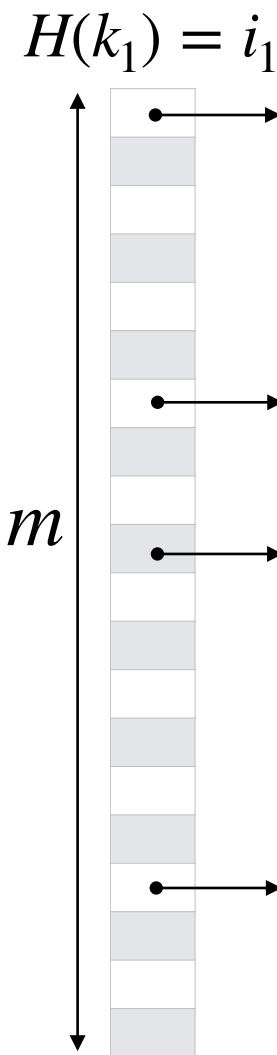
Take away values



Reducing a hash table

What can we remove from a hash table?

Take away values and keys



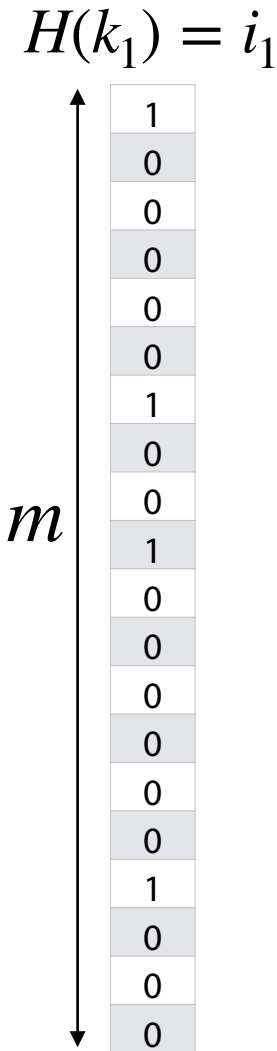


Reducing a hash table

What can we remove from a hash table?

Take away values and keys

This is a ***bloom filter***





Bloom Filter ADT

Constructor

Insert

Find

Bloom Filter: Insertion

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

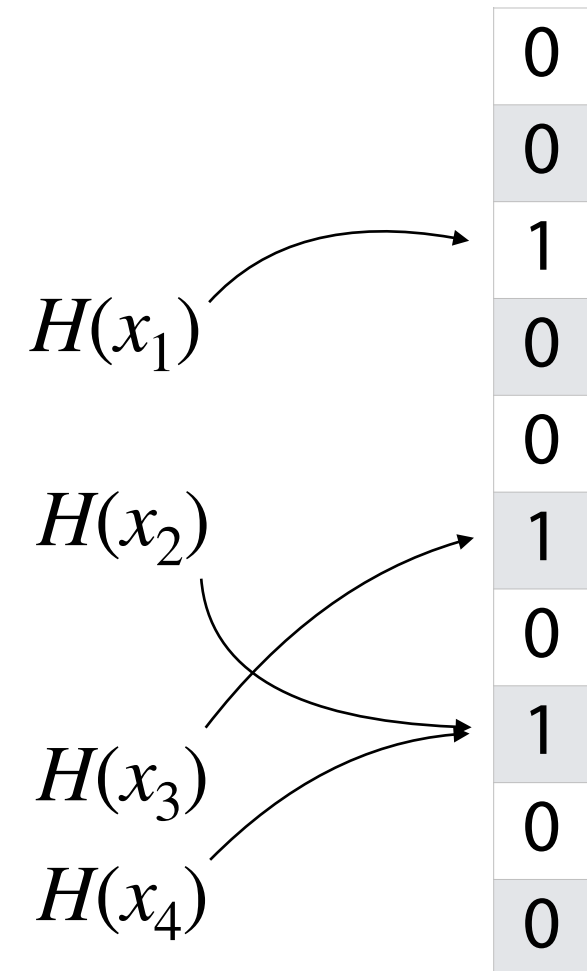
$h(k) = k \% 7$

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

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

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

$h(k) = k \% 7$

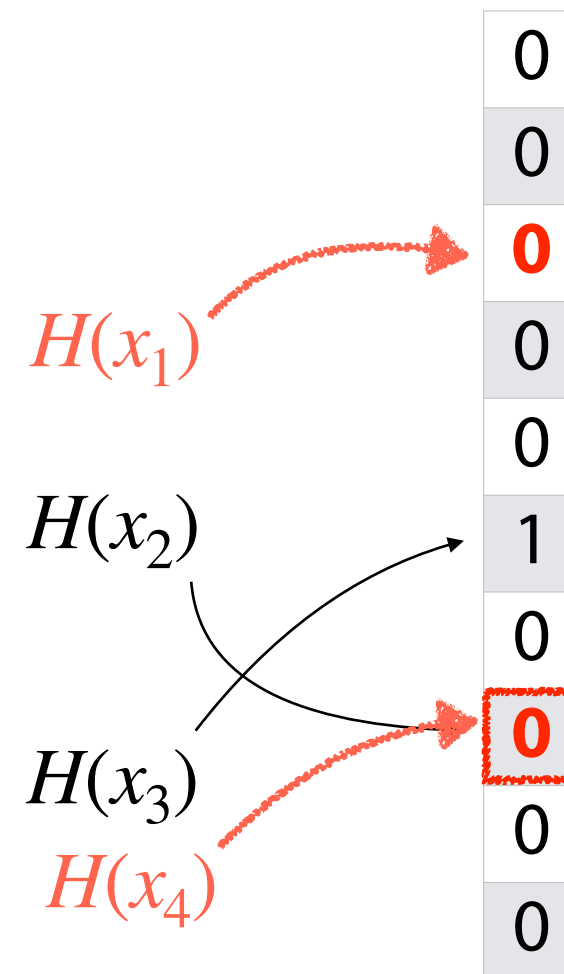
0	0
1	1
2	1
3	0
4	1
5	0
6	1

`_delete(13)`

`_delete(29)`

Bloom Filter: Deletion

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



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)`

`_find(20)`

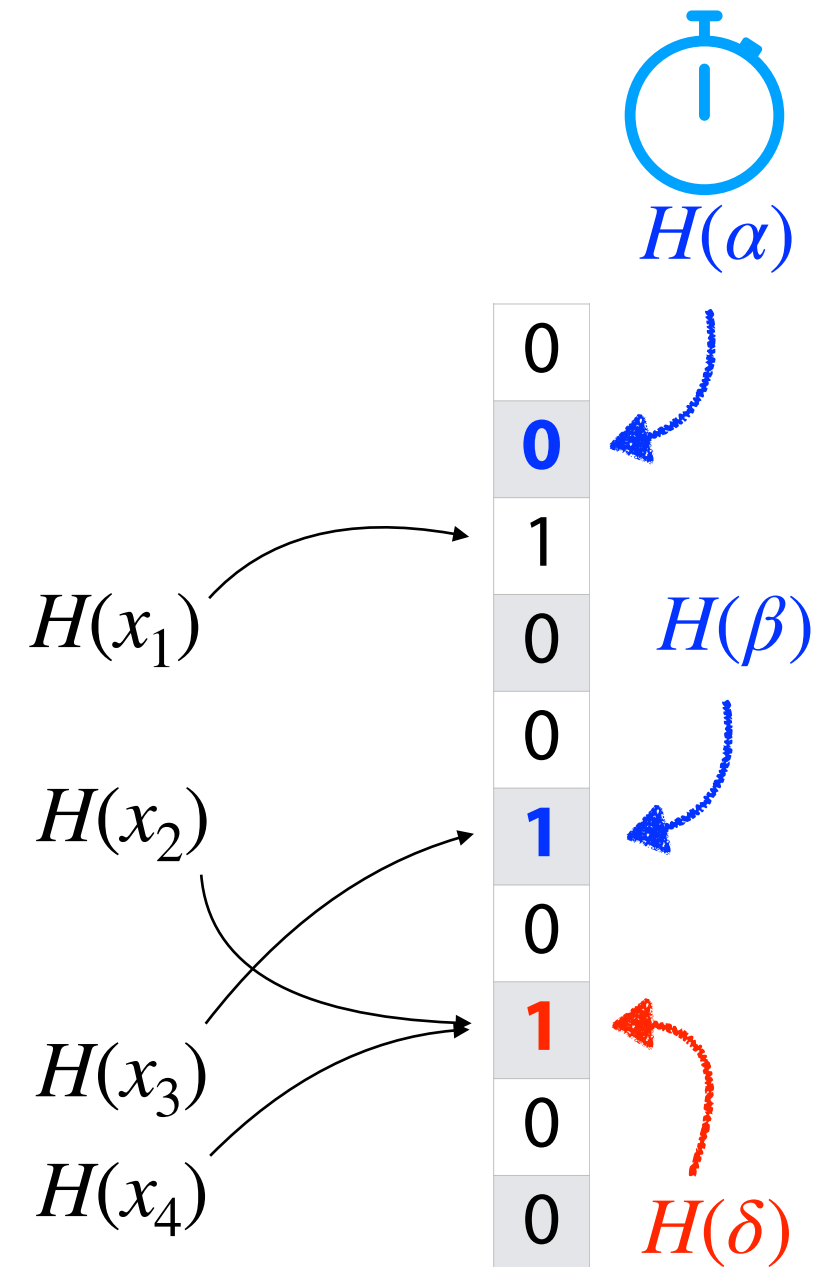
`_find(3)`

Bloom Filter: Search

The bloom filter is a *probabilistic* data structure!

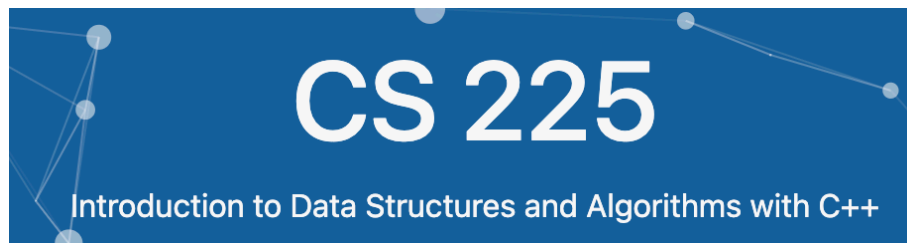
If the value in the BF is 0:

If the value in the BF is 1:

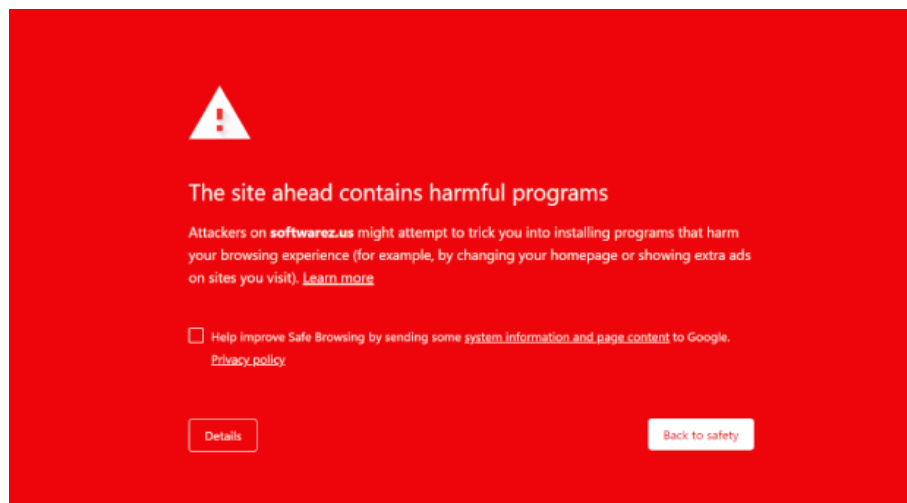


Probabilistic Accuracy: Malicious Websites

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



"Not malicious"



"Malicious"

Probabilistic Accuracy: Malicious Websites

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

True Positive:

False Positive:

False Negative:

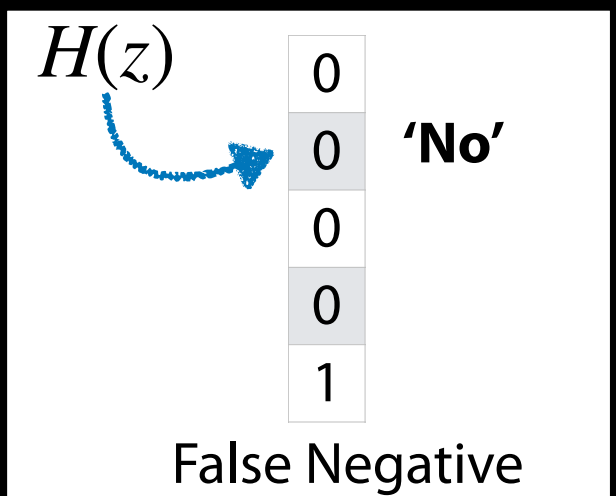
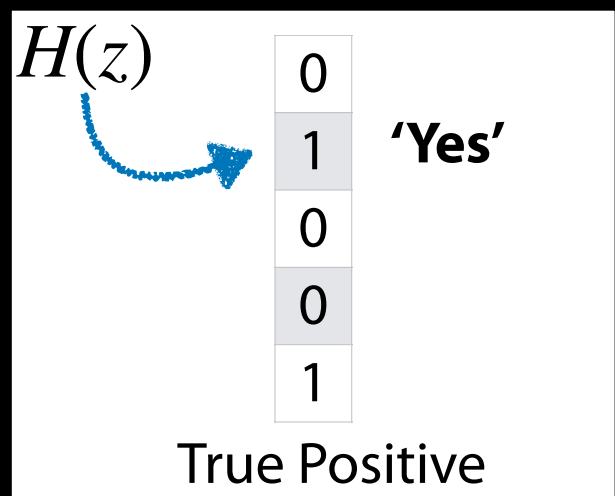
True Negative:

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

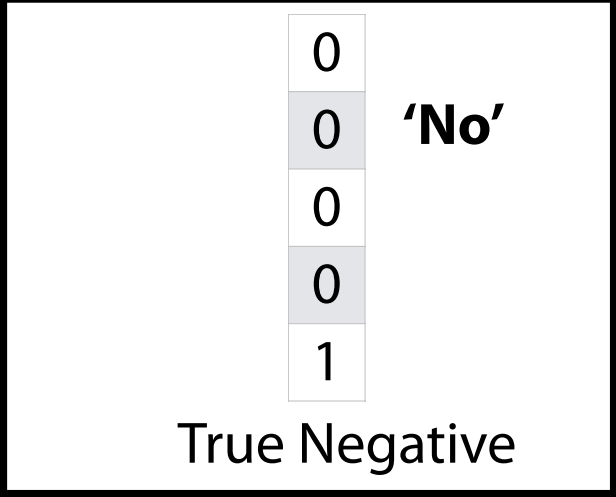
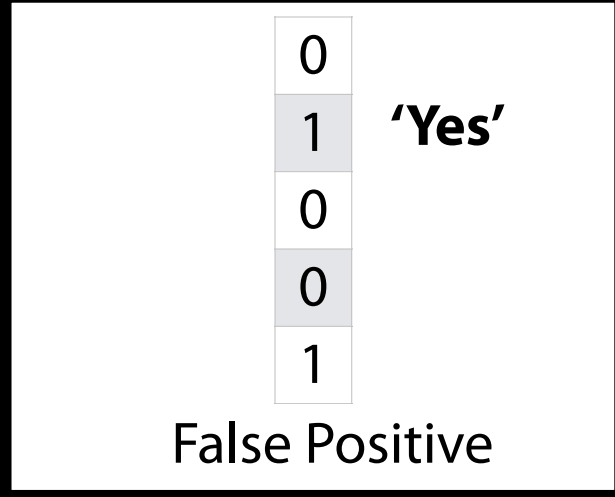
Bit Value = 1

Bit Value = 0

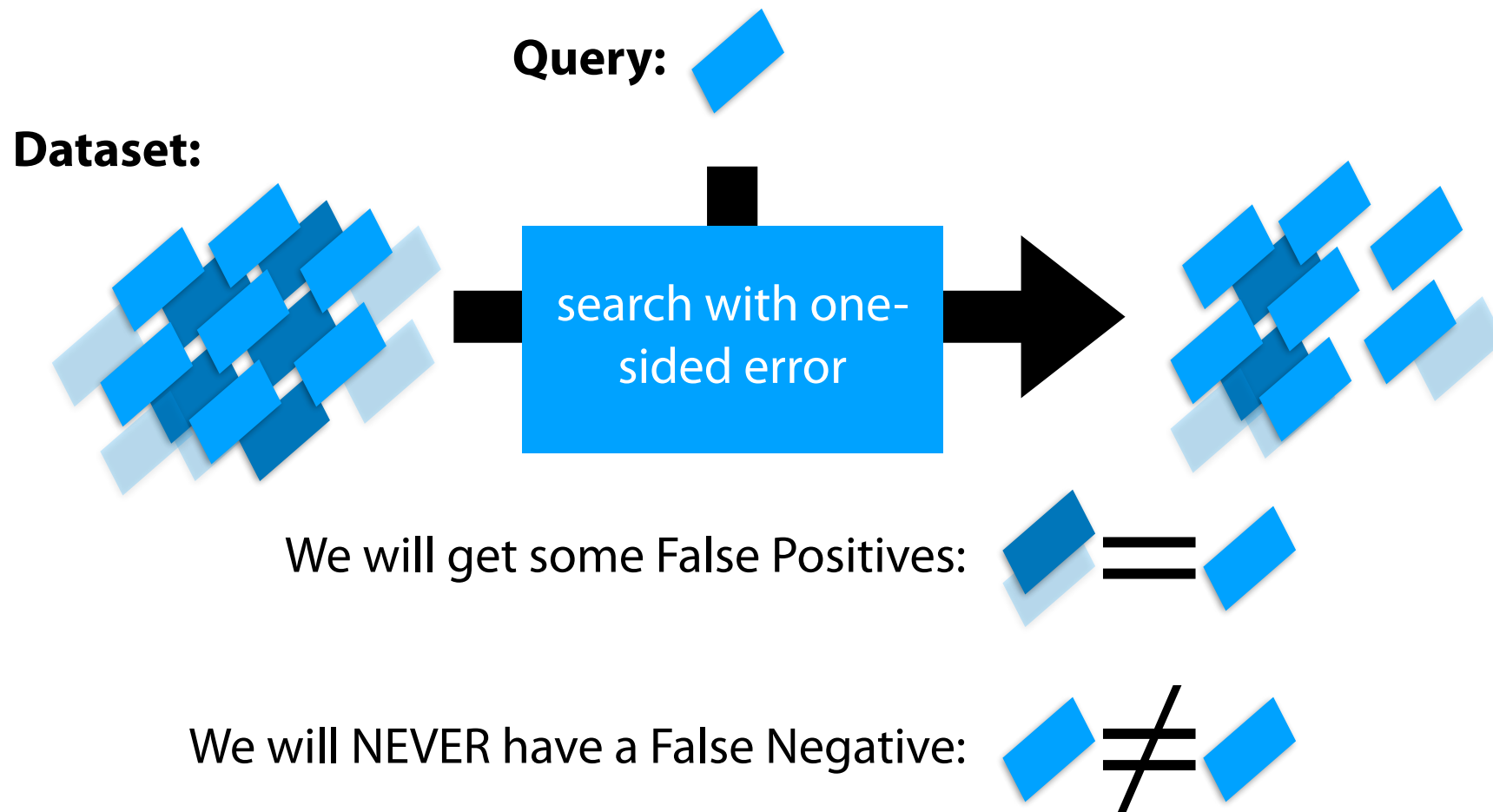
Item Inserted



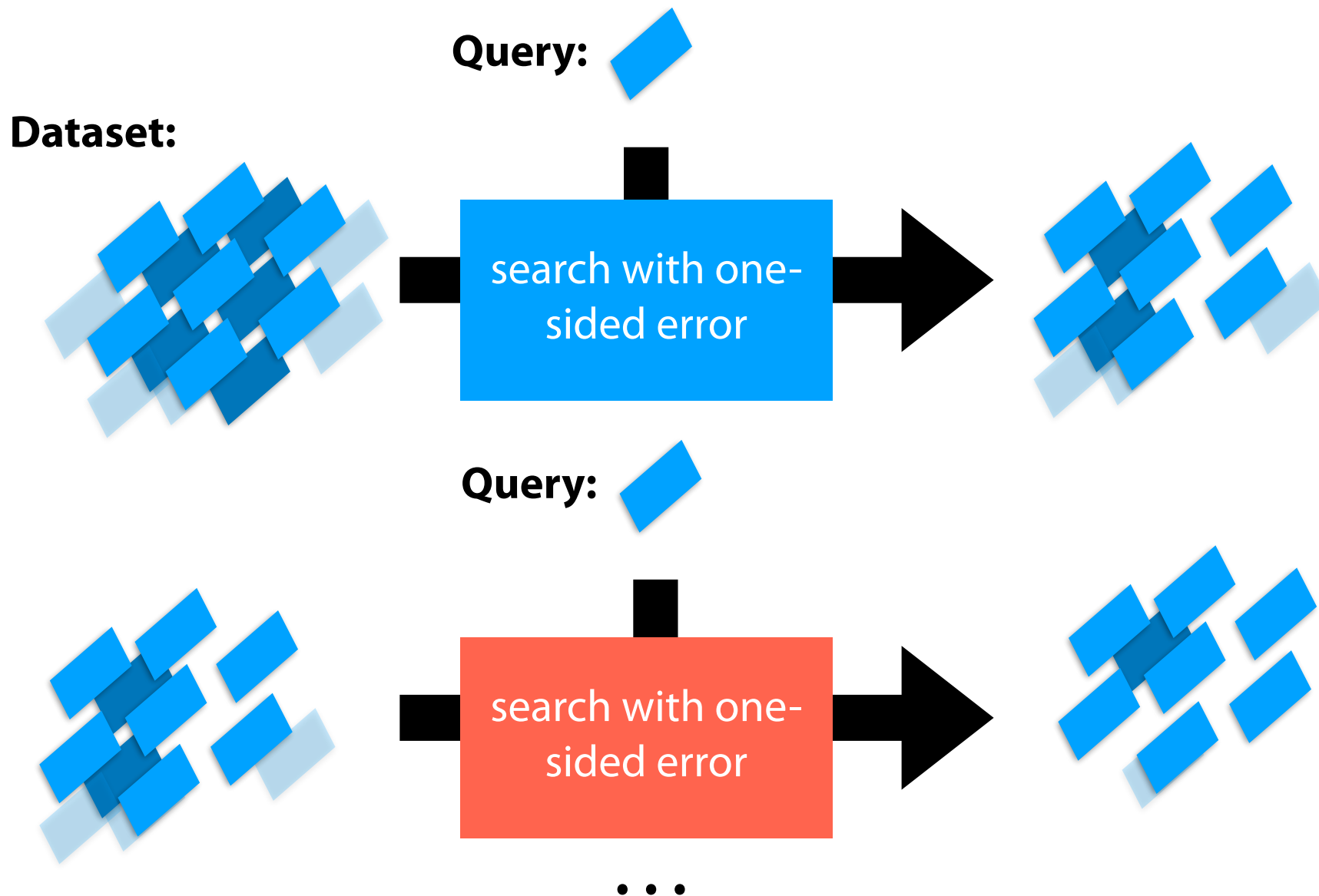
Item NOT inserted



Probabilistic Accuracy: One-sided error

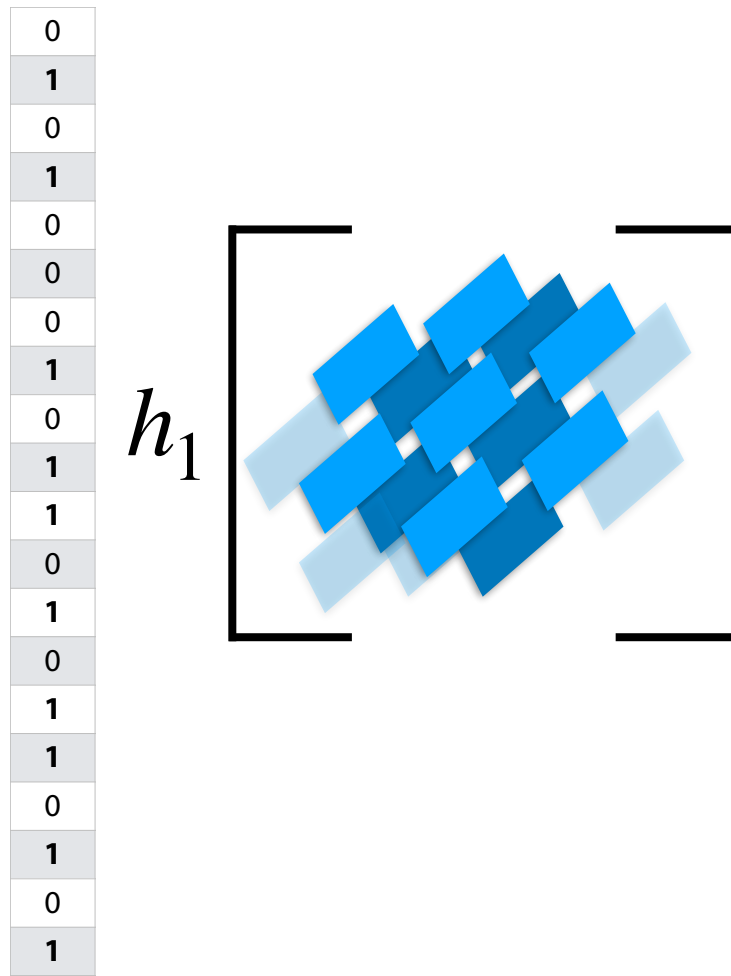


Probabilistic Accuracy: One-sided error



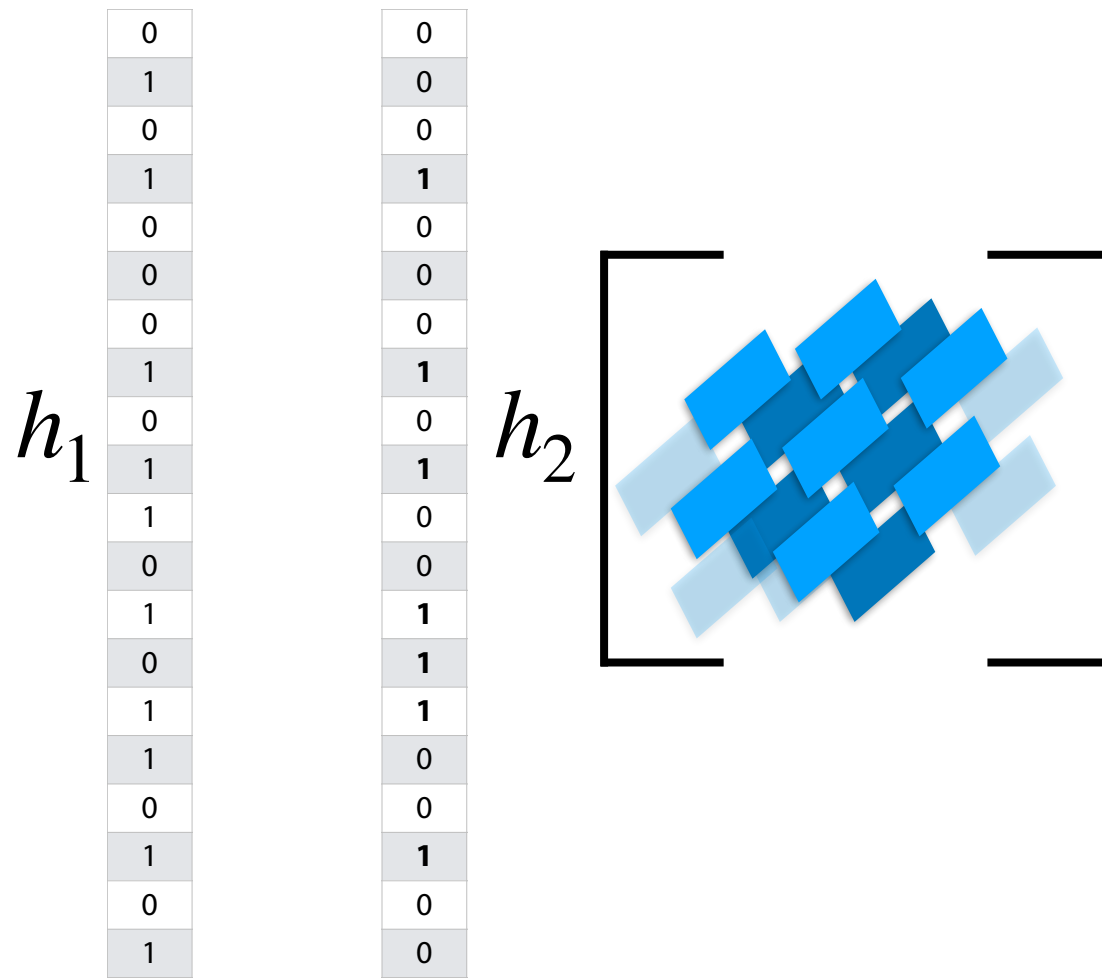
Bloom Filter: Repeated Trials

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



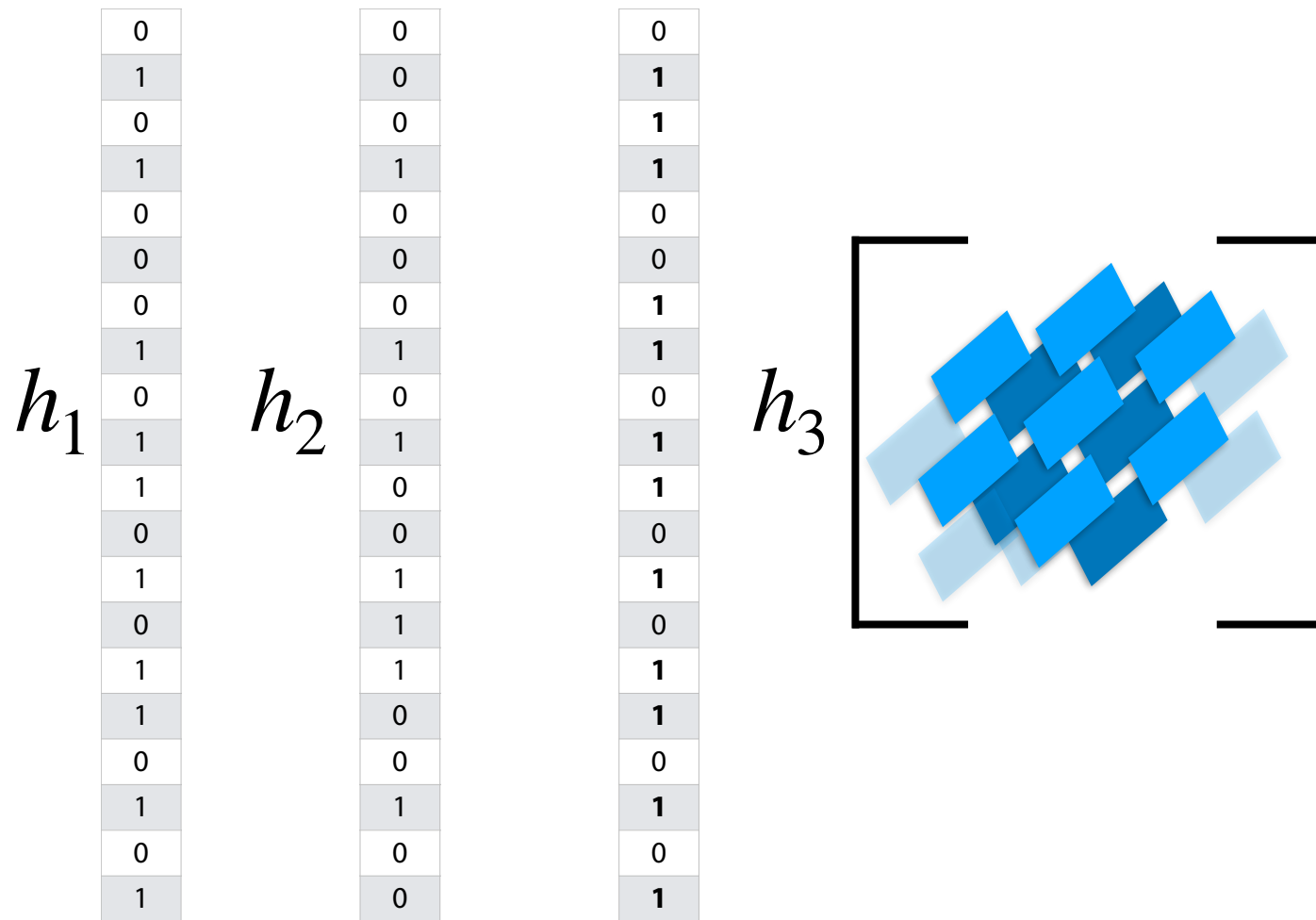
Bloom Filter: Repeated Trials

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



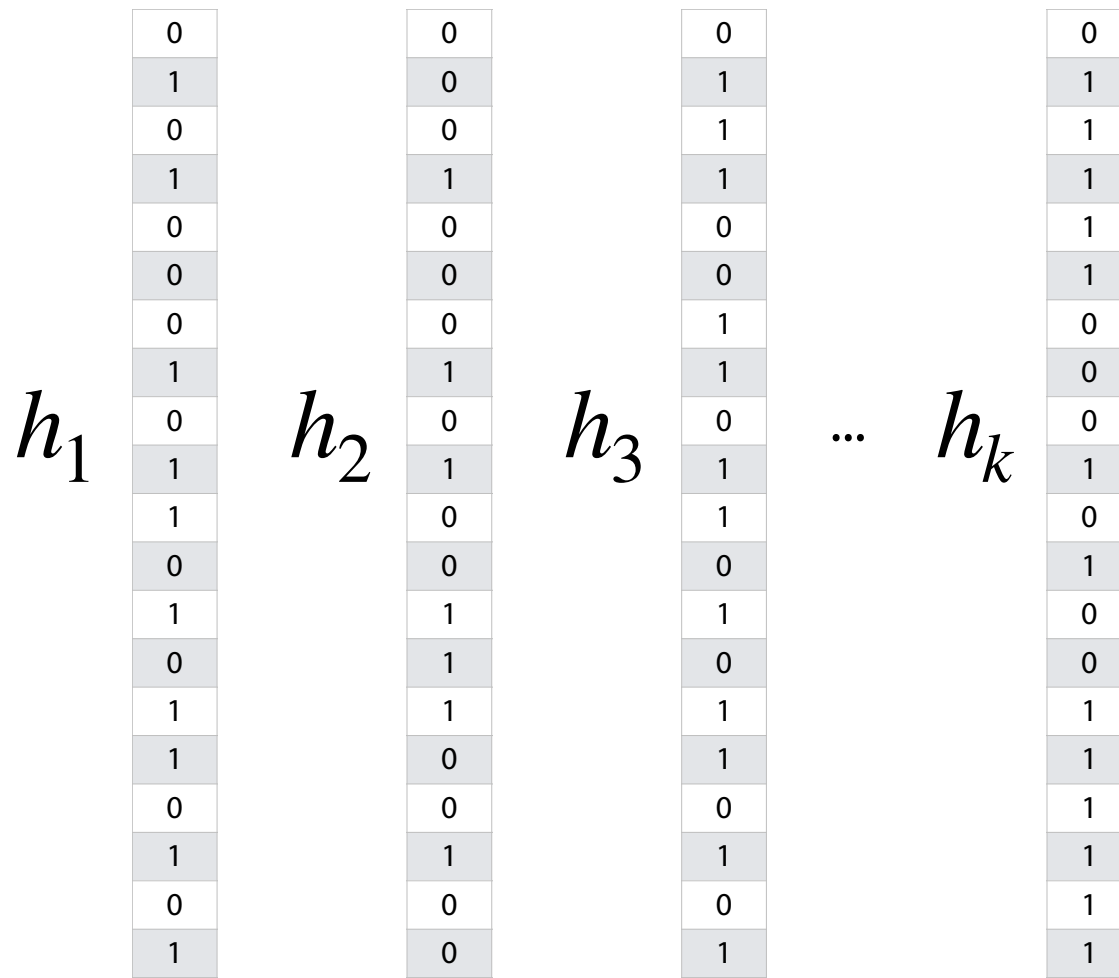
Bloom Filter: Repeated Trials

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

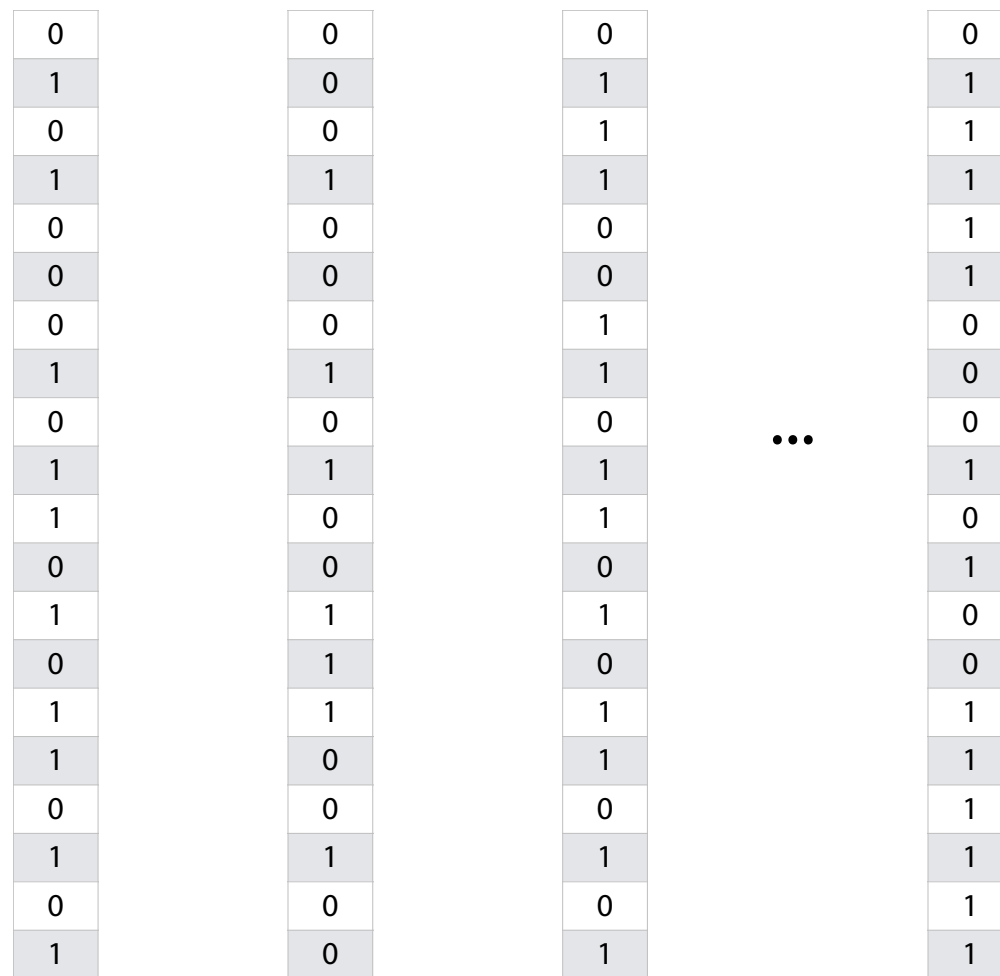


Bloom Filter: Repeated Trials

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

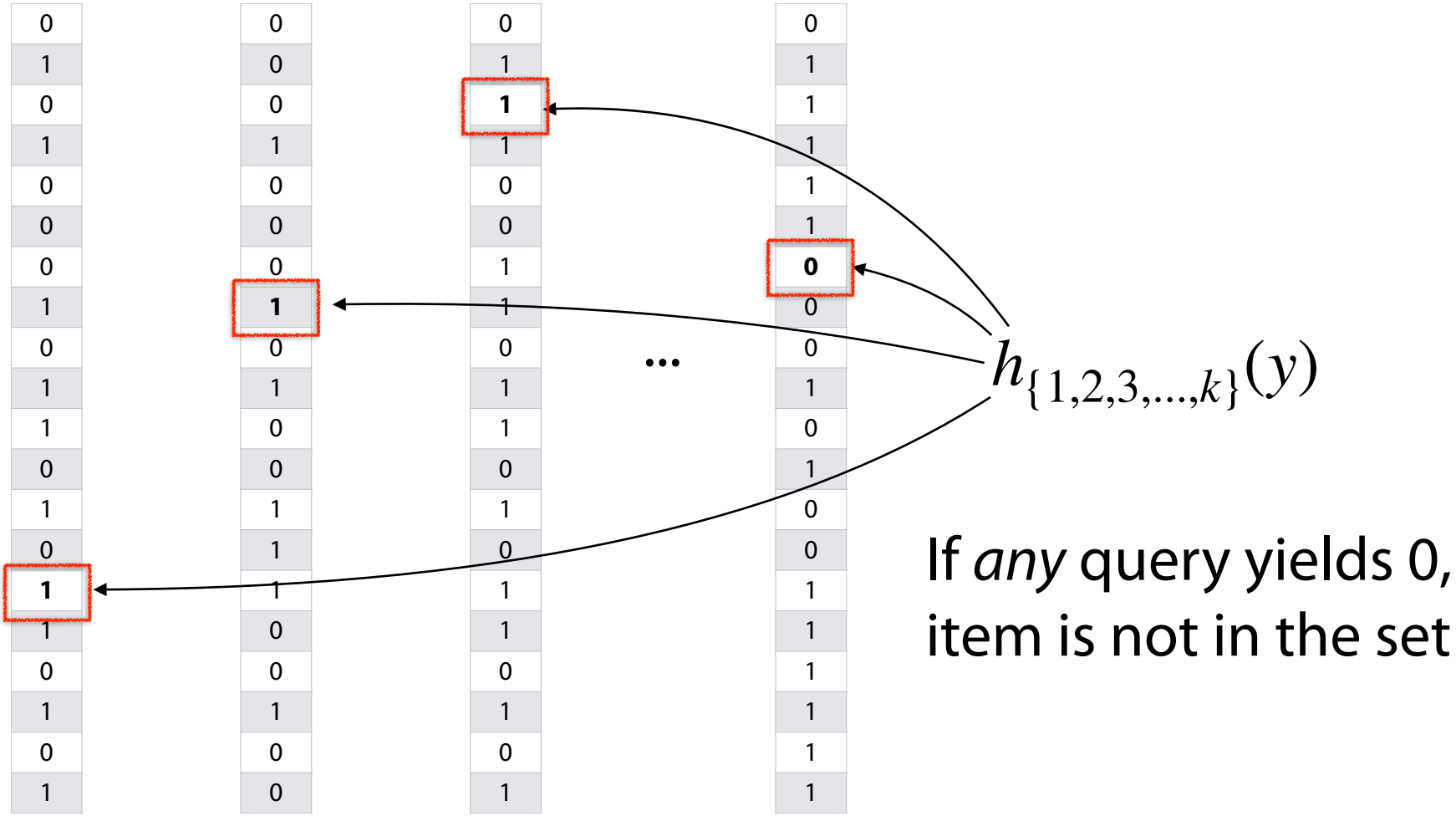


Bloom Filter: Repeated Trials

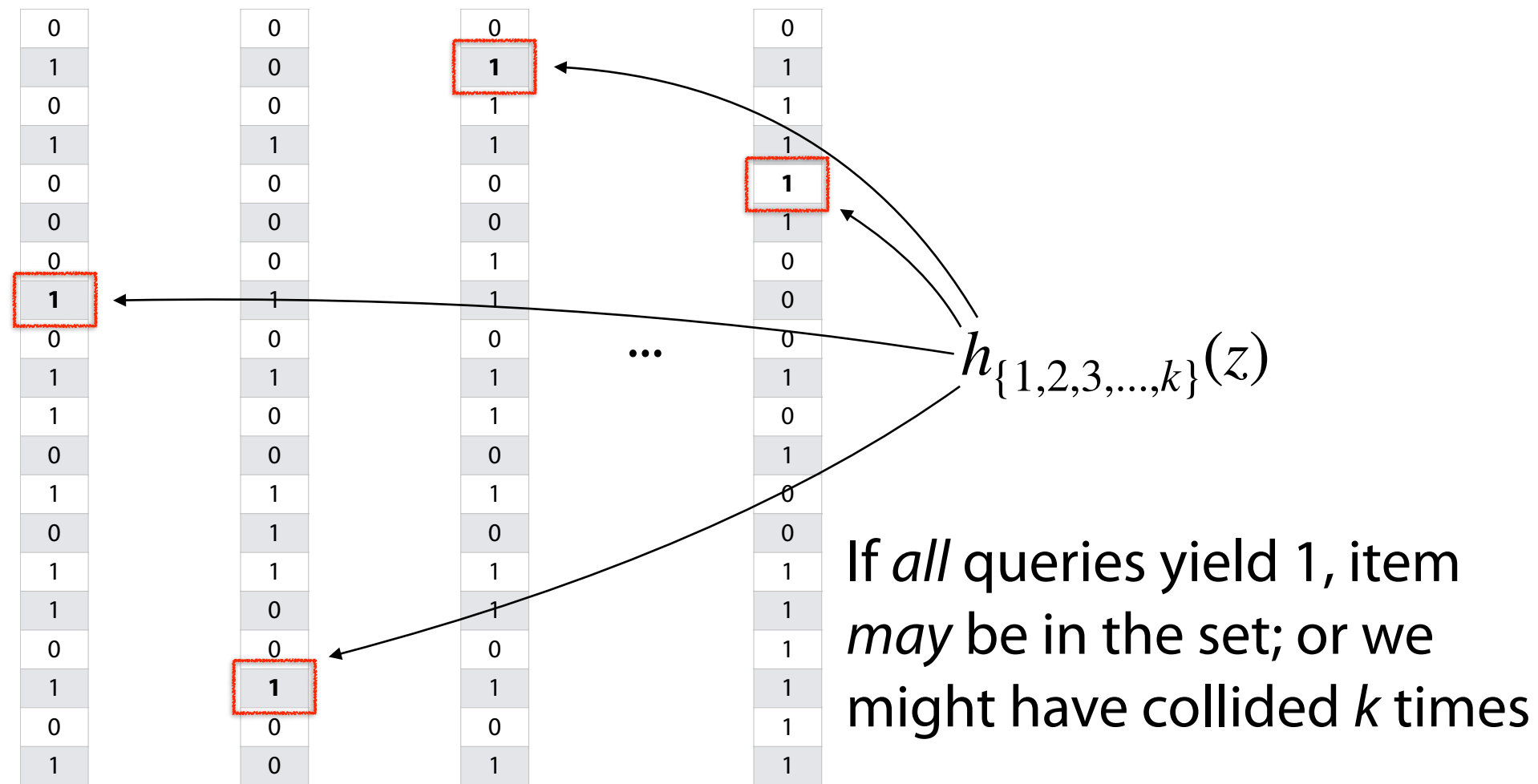


$$h_{\{1,2,3,\dots,k\}}(y)$$

Bloom Filter: Repeated Trials



Bloom Filter: Repeated Trials



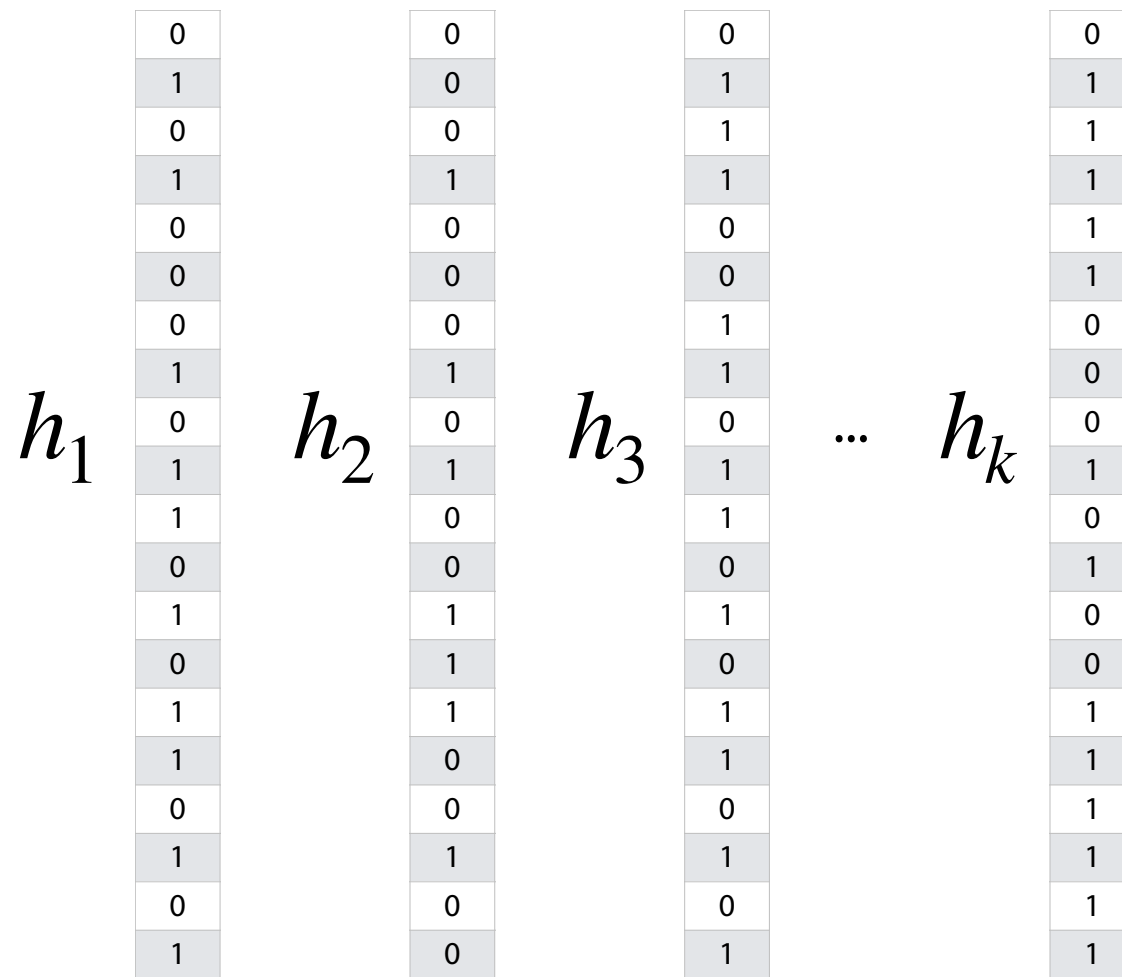
Bloom Filter: Repeated Trials

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	$h_1(x) = x \% 10$	$h_2(x) = 2x \% 10$	$h_3(x) = (5+3x) \% 10$
1	0			
2	1	<code><u>find</u>(1)</code>		
3	1			
4	1			
5	0			
6	1	<code><u>find</u>(16)</code>		
7	1			
8	1			
9	1			

Bloom Filter



A probabilistic data structure storing a set of values

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

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