Data Structures and Algorithms

Bloom and Counting Sketches

CS 225

November 30, 2022

Brad Solomon

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Reminder: Important Final Project Deadline!

Last day to have a mid-project checkin

Last day to (as a team) drop the project in favor of the final

PURE Symposium this Friday (Dec 2nd)

Location: ECEB Atrium (5-7 PM)

Learning Objectives

Review the math behind the bloom filter

Identify the optimal parameters for a bloom filter

Introduce extensions to the bloom filter

Discuss counting alternatives to bloom filters

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)

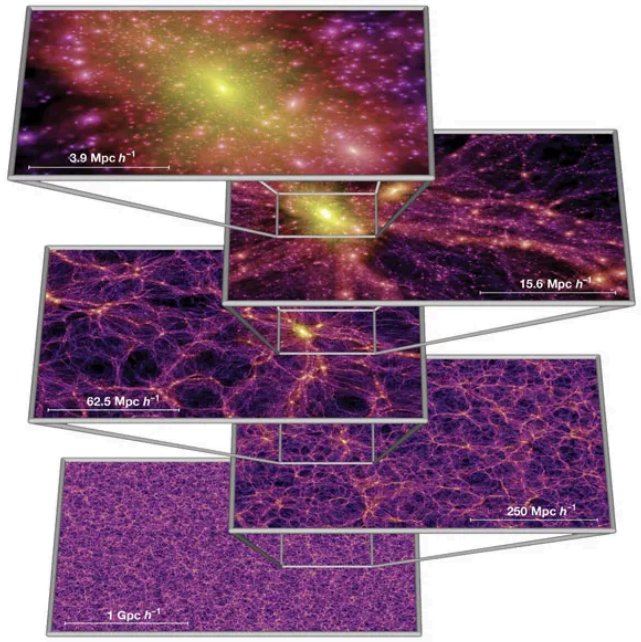


Image: https://doi.org/10.1038/nature03597

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

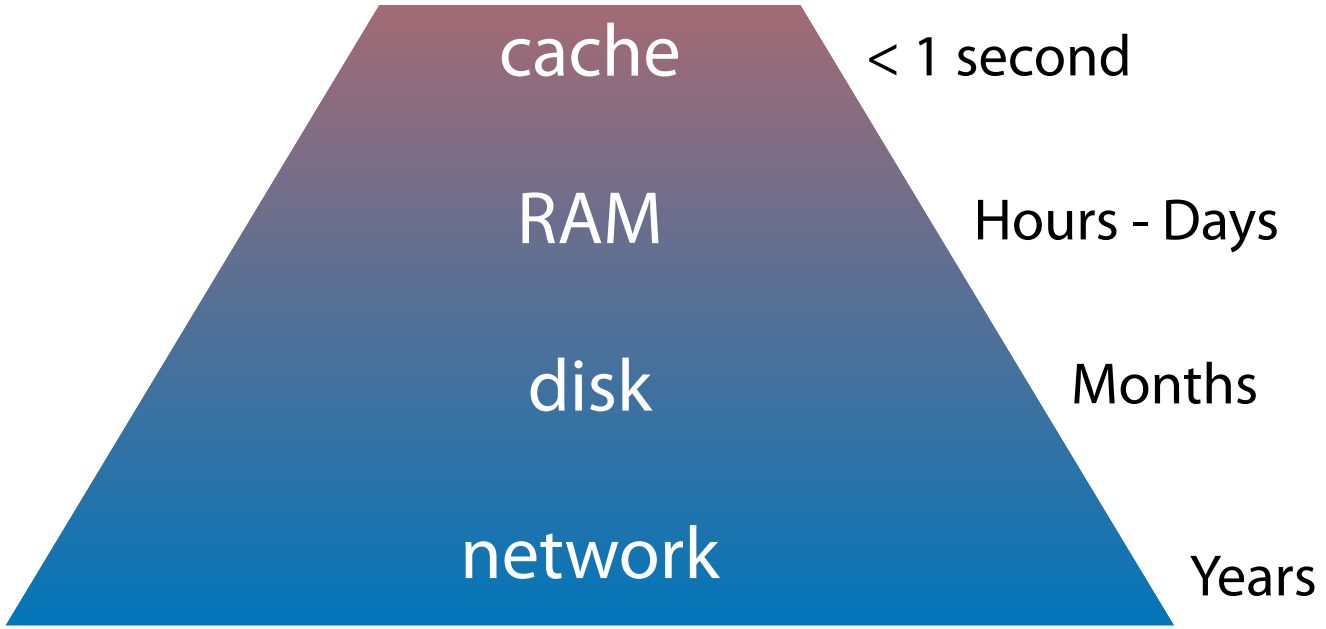
Table: http://doi.org/10.5334/dsj-2015-011

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)

Bloom Filters

A probabilistic data structure storing a set of values

$h_{\{1,2,3,\ldots,k\}}$

Has three key properties:

- k, number of hash functions
- n, expected number of insertions
- m, filter size in bits

Expected false positive rate: $\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k}$

Bloom Filter: Error Rate

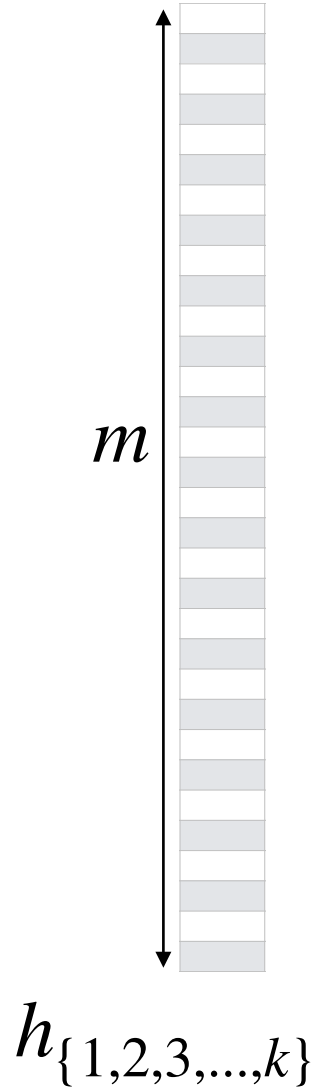
Given bit vector of size m and k SUHA hash function

What is our expected FPR after n objects are inserted?

The probability my bit is 1 after n objects inserted

$$\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k}$$

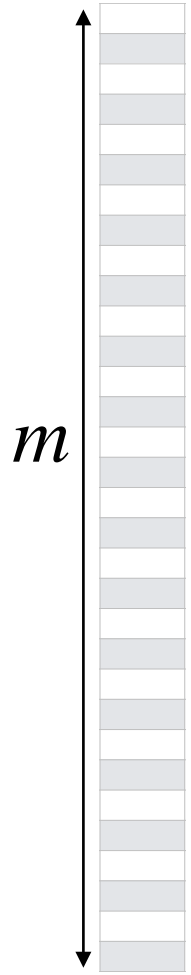
The number of [assumed independent] trials



Bloom Filter: Error Rate

$h_{\{1,2,3,\ldots,k\}}$

Vector of size m, k SUHA hash function, and n objects



To minimize the FPR, do we prefer…

(A) large k **(B) small** k

$$\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k}$$

Bloom Filter: Optimal Error Rate

So how can we find the minimum error rate?

Bloom Filter: Optimal Error Rate

$$\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k} \approx \left(1 - e^{\frac{-nk}{m}}\right)^{k}$$

Bloom Filter: Optimal Error Rate

Claim: The optimal hash function is when $k* = ln\ 2 \cdot \frac{m}{n}$

Bloom Filter: Optimal Error Rate

Claim: The optimal hash function is when $k* = ln\ 2 \cdot \frac{m}{n}$

$$\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k} \approx \left(1 - e^{\frac{-nk}{m}}\right)^{k}$$

$$\frac{d}{dk}\left(1 - e^{\frac{-nk}{m}}\right)^{k} \approx \frac{d}{dk}\left(k \ \ ln(1 - e^{\frac{-nk}{m}})\right)$$

Derivative is zero when $k* = \ln 2 \cdot \frac{m}{n}$

Bloom Filter: Error Rate

$$\left(1 - e^{\frac{-nk}{m}}\right)^k$$

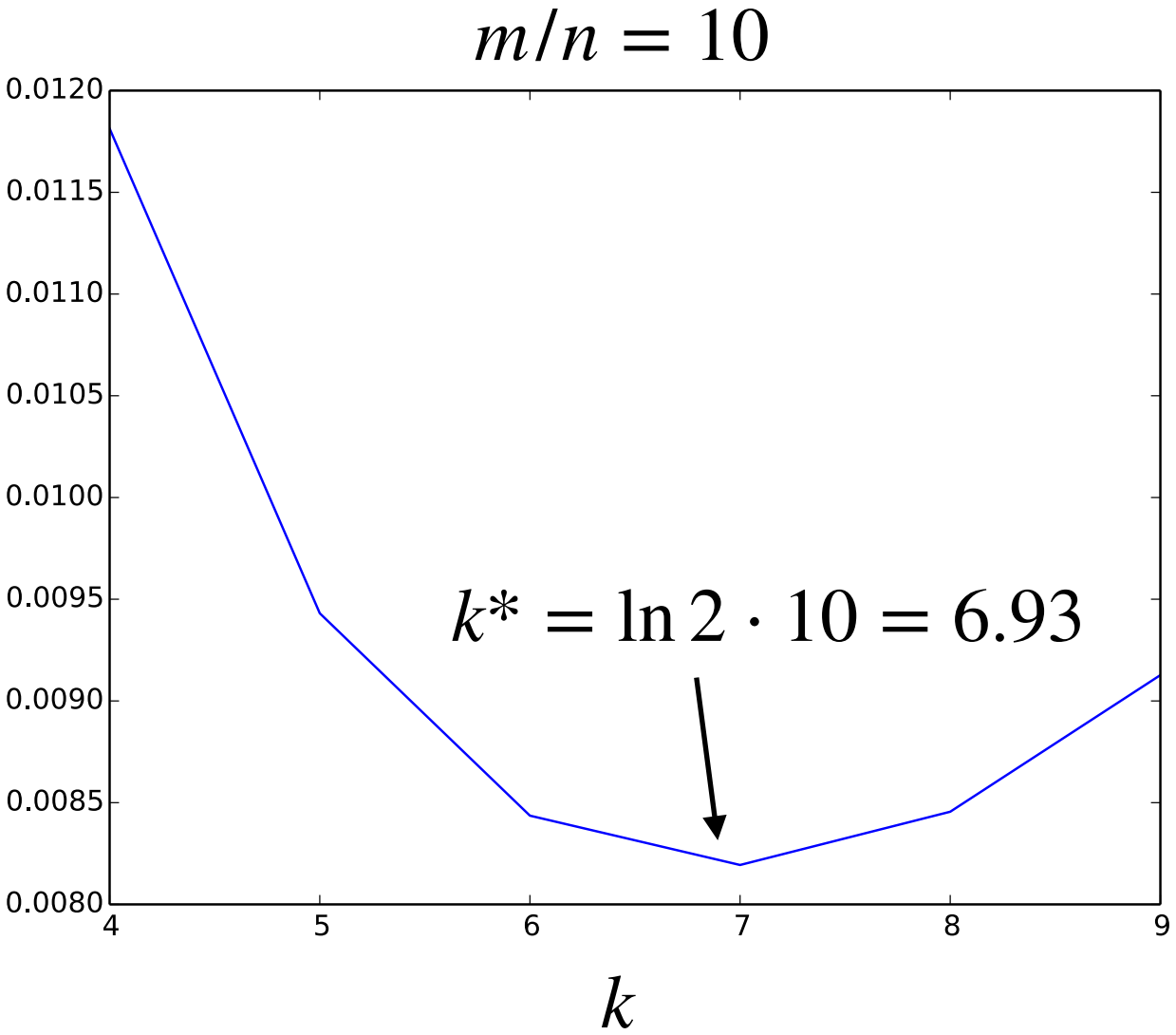


Figure by Ben Langmead

Bloom Filter: Optimal Parameters

$$k^* = \ln 2 \cdot \frac{m}{n}$$

Given any two values, we can optimize the third

$n = 100$ items $\quad k = 3$ hashes $\quad m =$

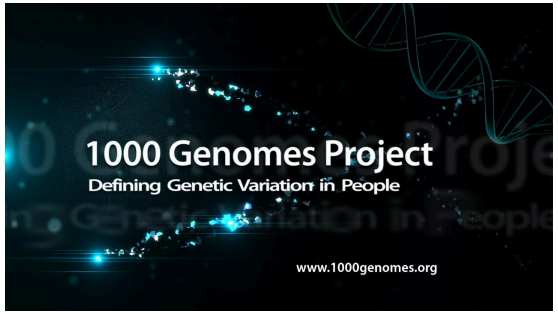
$m = 100$ bits $\quad n = 20$ items $\quad k =$

$m = 100$ bits $\quad k = 2$ items $\quad n =$

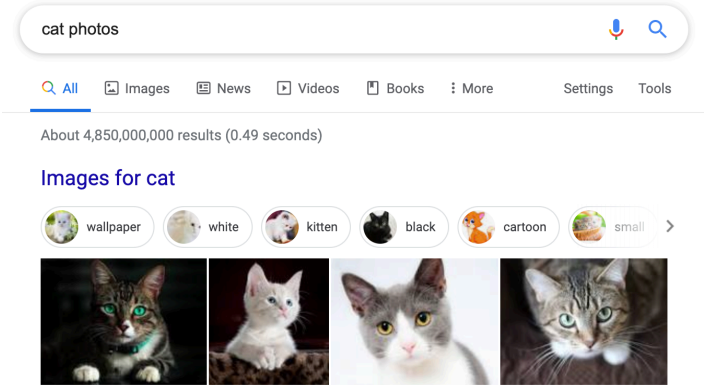
Bloom Filter: Optimal Parameters

$$m = \frac{nk}{\ln 2} \approx 1.44 \cdot nk$$

Optimal hash function is still O(m)!

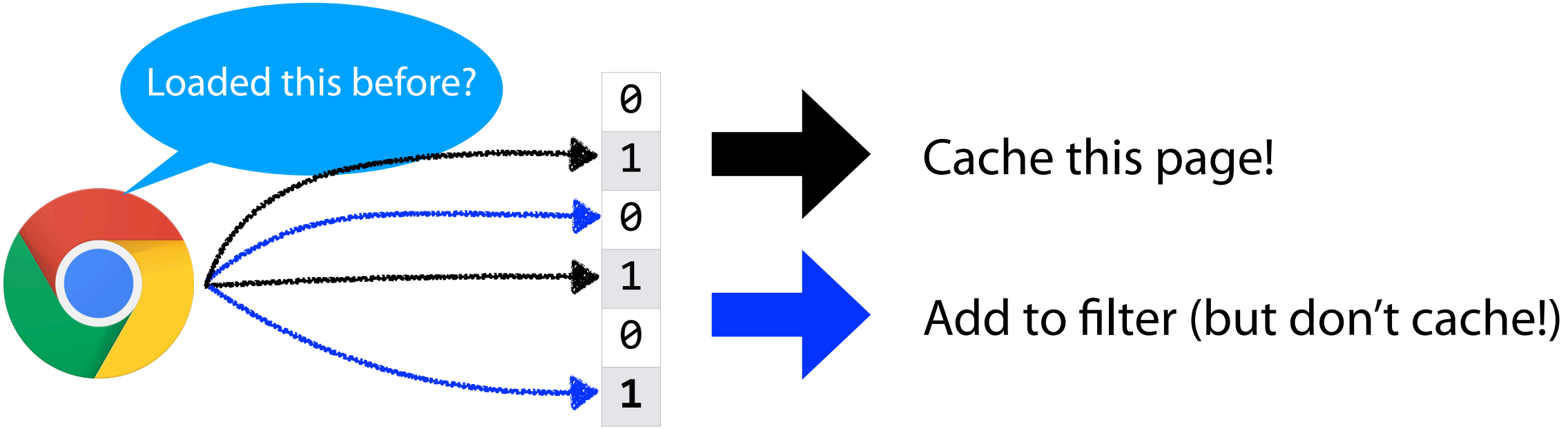


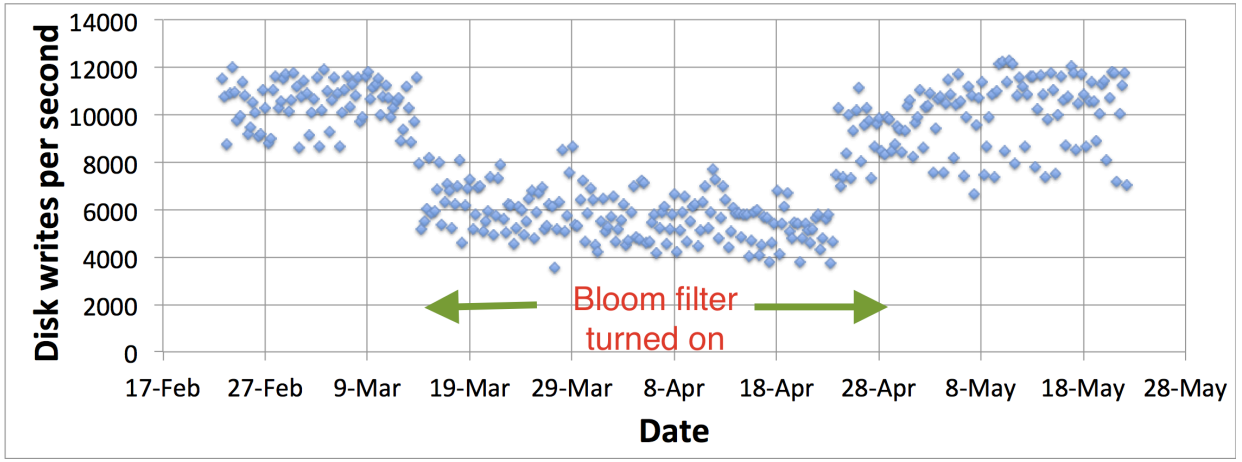
n = 250,000 files vs ~10^{15} nucleotides vs 260 TB



n = 60 billion — 130 trillion

Bloom Filter: Website Caching

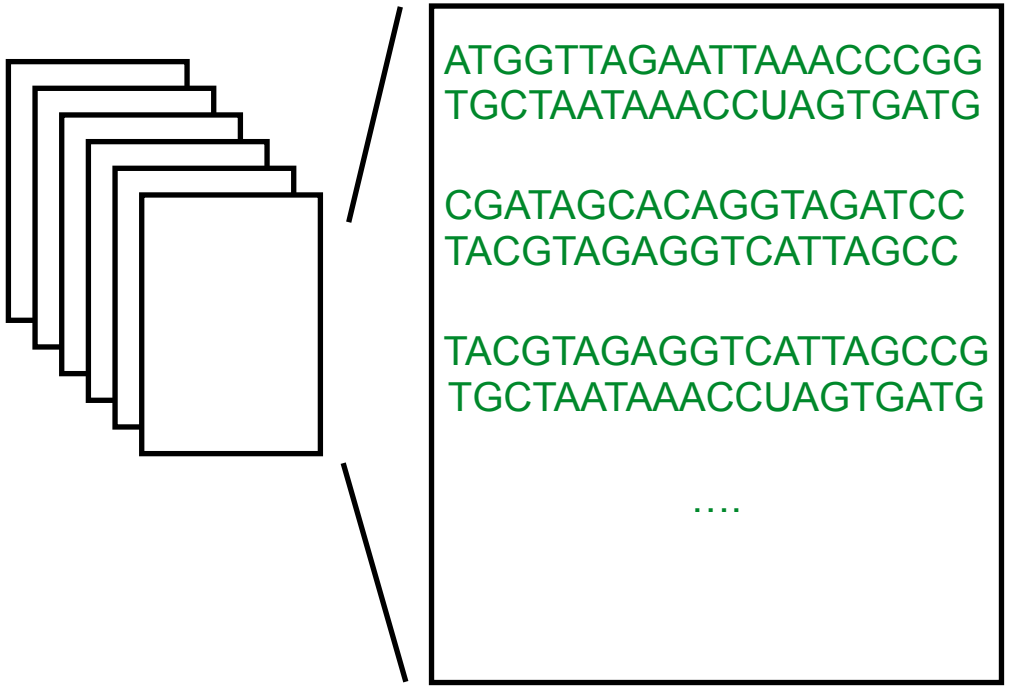




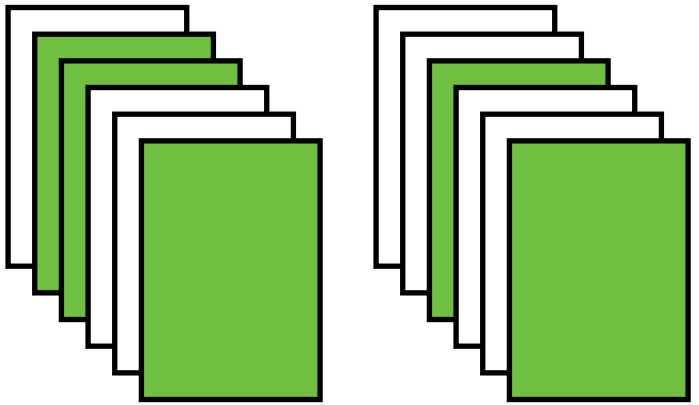
Maggs, Bruce M., and Ramesh K. Sitaraman. **Algorithmic nuggets in content delivery.** *ACM SIGCOMM Computer Communication Review* 45.3 (2015): 52-66.

Sequence Bloom Trees

Imagine we have a large collection of text…

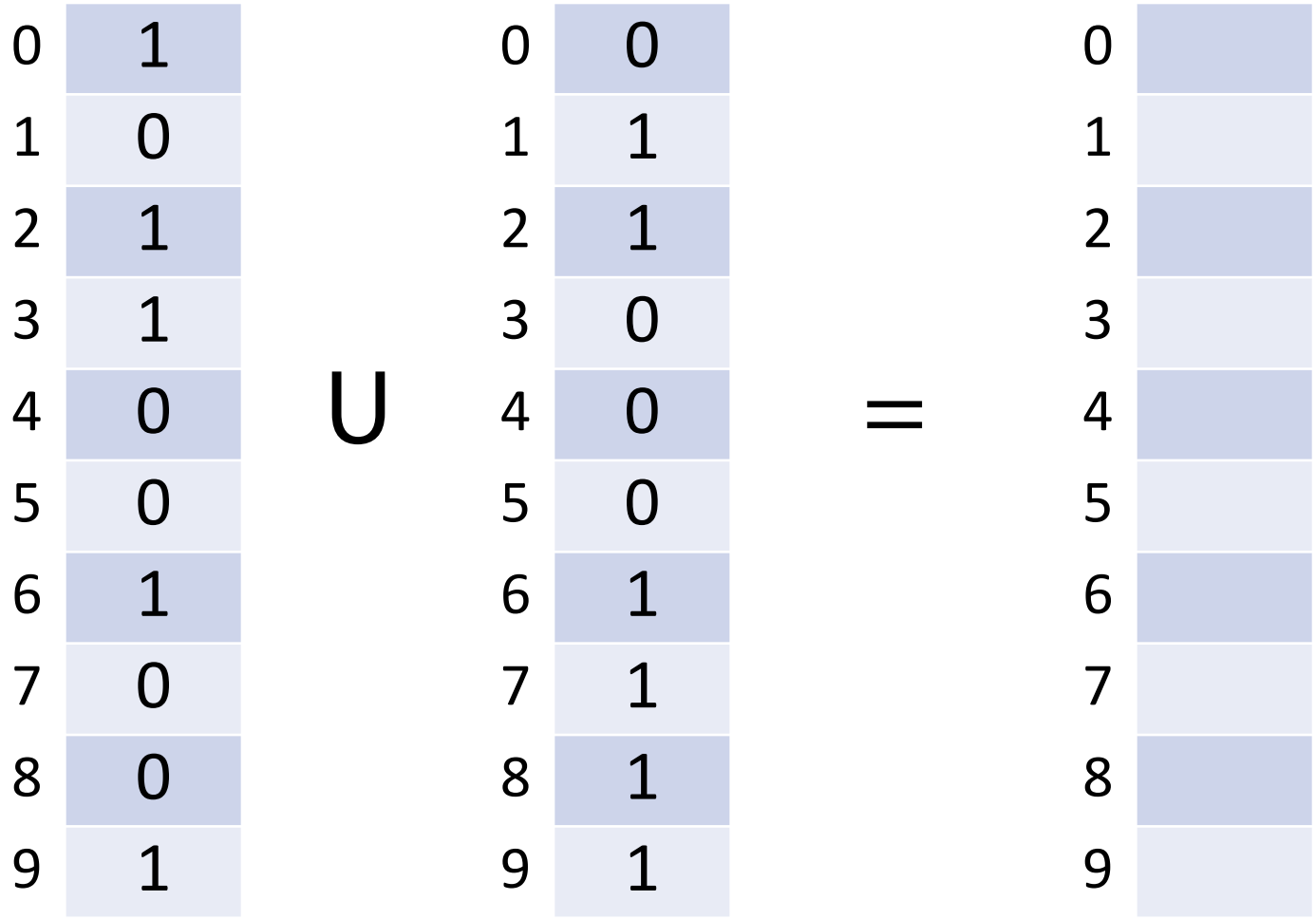


And our goal is to search these files for a query of interest…



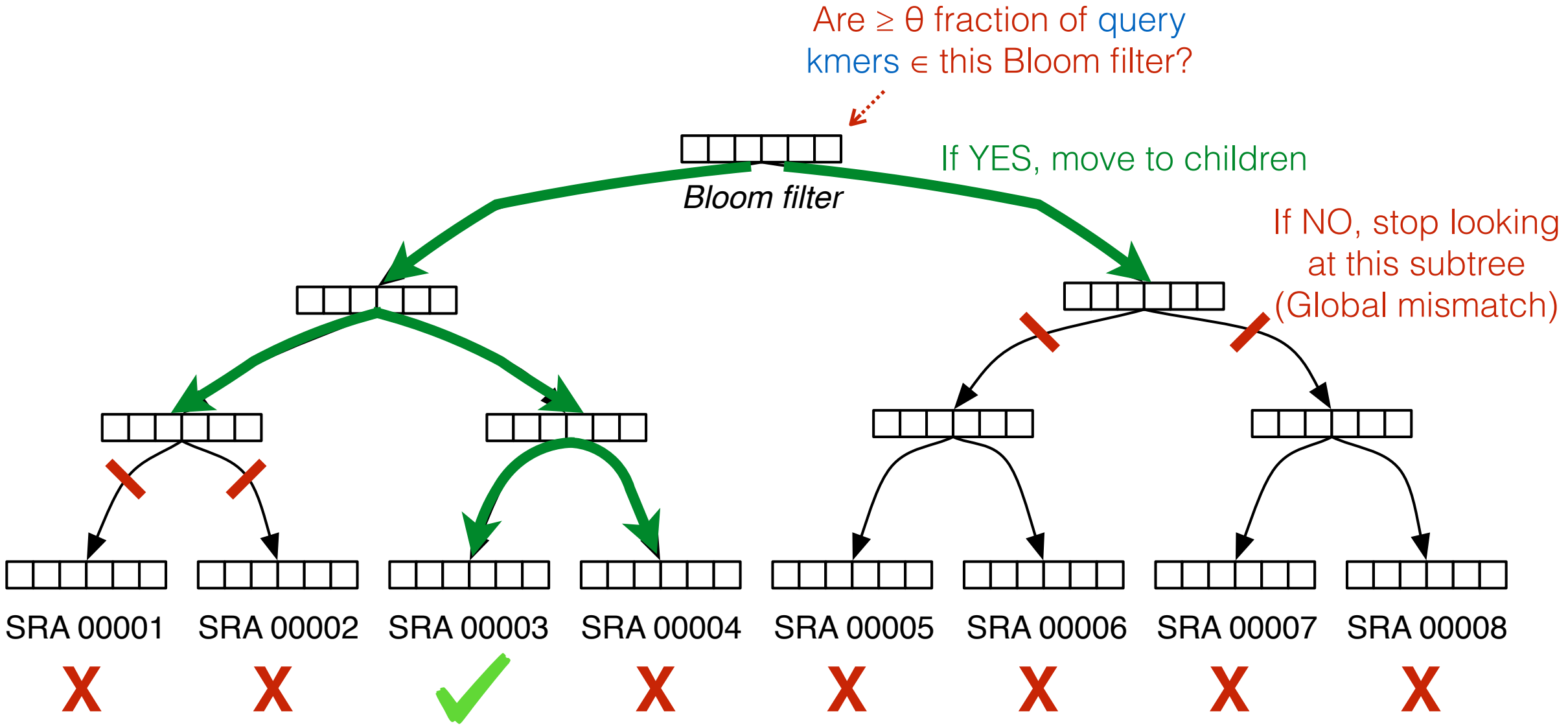
Bloom Filters: Unioning

Bloom filters can be trivially merged using bit-wise union.

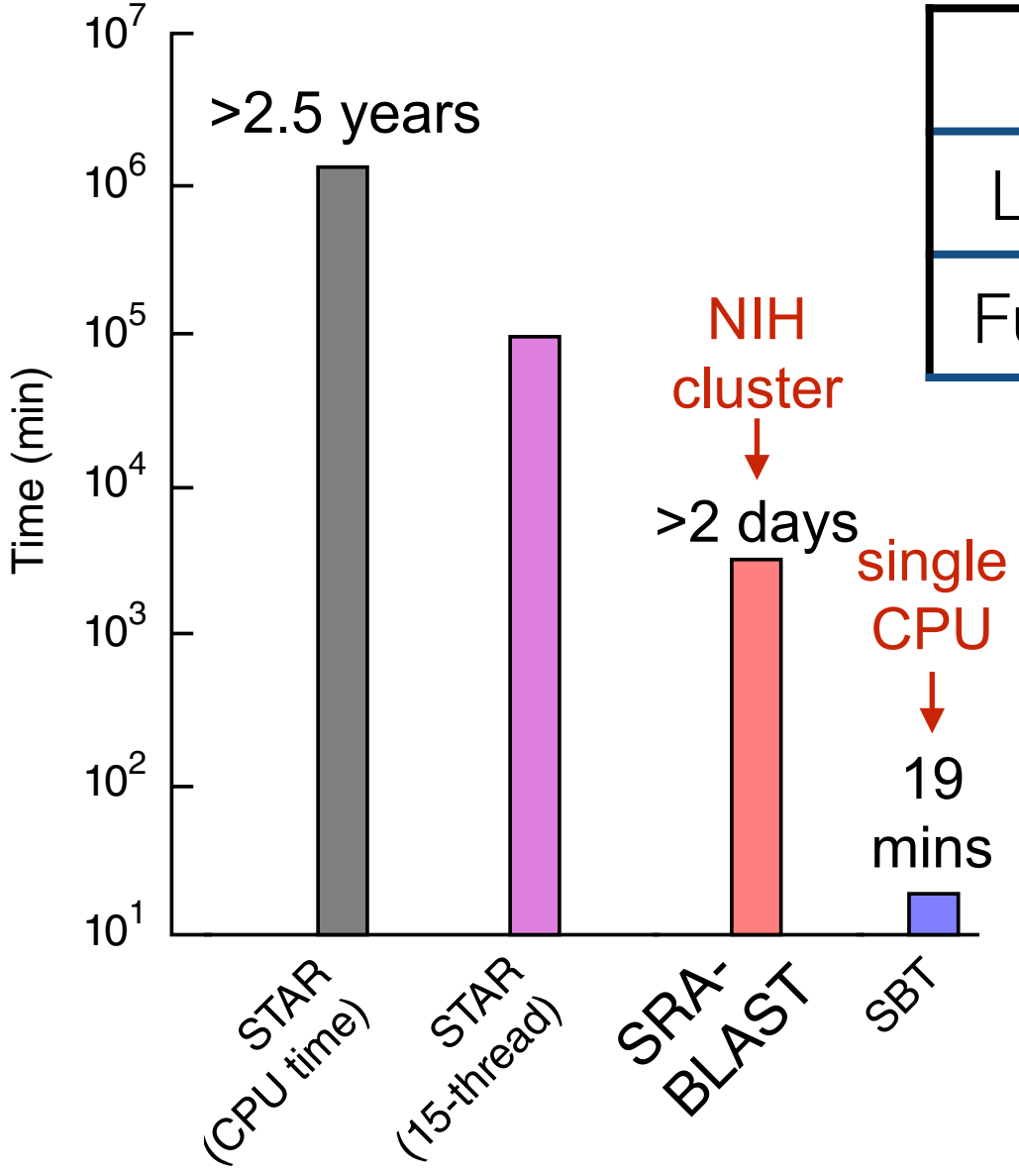


Sequence Bloom Trees

Sequence Bloom Trees



Sequence Bloom Trees



	SRA	FASTA.gz	SBT
Leaves	4966 GB	2692 GB	63 GB
Full Tree	-	-	200 GB

Solomon, Brad, and Carl Kingsford. "Fast search of thousands of short-read sequencing experiments." *Nature biotechnology* 34.3 (2016): 300-302.

Solomon, Brad, and Carl Kingsford. "Improved search of large transcriptomic sequencing databases using split sequence bloom trees." *International Conference on Research in Computational Molecular Biology*. Springer, Cham, 2017.

Sun, Chen, et al. "Allsome sequence bloom trees." *International Conference on Research in Computational Molecular Biology*. Springer, Cham, 2017.

Harris, Robert S., and Paul Medvedev. "Improved representation of sequence bloom trees." *Bioinformatics* 36.3 (2020): 721-727.

Bloom Filters: Tip of the Iceberg

Cohen, Saar, and Yossi Matias. "Spectral bloom filters." *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. 2003.

Fan, Bin, et al. "Cuckoo filter: Practically better than bloom." *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. 2014.

Nayak, Sabuzima, and Ripon Patgiri. "countBF: A General-purpose High Accuracy and Space Efficient Counting Bloom Filter." *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE, 2021.

Mitzenmacher, Michael. "Compressed bloom filters." *IEEE/ACM transactions on networking* 10.5 (2002): 604-612.

Crainiceanu, Adina, and Daniel Lemire. "Bloofi: Multidimensional bloom filters." *Information Systems* 54 (2015): 311-324.

Chazelle, Bernard, et al. "The bloomier filter: an efficient data structure for static support lookup tables." *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*. 2004.

There are many more than shown here…

Counting Sketches

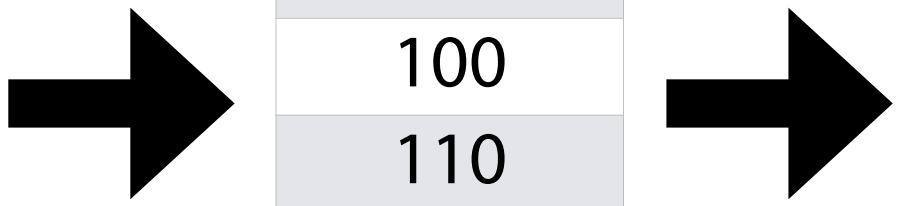
A **sketch** is a compact (reduced) representation of a dataset that acts as a replacement for calculations.

Sometimes we need more information than ‘presence/absence’…

3201
946
5581
8945
6145
8126
3887
8925
1246
8324
4549
9100
3887
8499
8970
3921
8925
4859

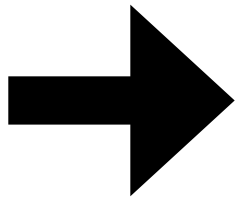
Count Min Sketch

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

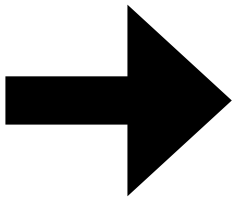


Count Min Sketch

0
1
0
1
0
0
0
1
0
1



000
110
010
001
100
110
000
000
100
111



0	6	2	1	4	6	0	0	4	7
5	0	5	2	0	5	0	3	5	5
3	3	3	3	3	3	3	3	3	3
4	2	4	2	4	2	4	2	4	2
30	0	0	0	0	0	0	0	0	0

Count Min Sketch

A **count-min sketch** has k rows, each with its own hash function.

Each bucket is b-bits (allowing 2^b count values)

$h_1(x)$	0	0	0	0	0	0	0	0	0	0
$h_2(x)$	0	0	0	0	0	0	0	0	0	0
$h_3(x)$	0	0	0	0	0	0	0	0	0	0
$h_4(x)$	0	0	0	0	0	0	0	0	0	0
$h_5(x)$	0	0	0	0	0	0	0	0	0	0

Count Min Sketch Insertion

S = { 1, 3, 8, 16}

$h_1(k) = k \% 7$ $h_2(k) = k+3(k\%2) \% 7$ $h_3(k) = |k - 4| \% 7$

h_1	0	0	0	0	0	0	0
h_2	0	0	0	0	0	0	0
h_3	0	0	0	0	0	0	0

Count Min Sketch Find

$S = \{1, 3, 8, 16\}$

$h_1(k) = k \% 7 \quad h_2(k) = k+3(k\%2) \% 7 \quad h_3(k) = |k - 4| \% 7$

```
_find(16)
```

```
_find(1)
```

h_1	0	2	1	1	0	0	0
h_2	0	1	1	0	1	0	1
h_3	0	1	0	1	1	1	0

Count Min Sketch Find

What is our estimated count of x?

How many **known** collisions?

$h_1(x)$	10	5	13	17	8	1	6
$h_2(x)$	3	7	20	12	2	9	7
$h_3(x)$	12	6	7	5	9	18	3
$h_4(x)$	4	19	26	1	4	5	1
$h_5(x)$	6	6	8	11	6	7	16

Improving Count Min Sketch

Given what we know about collisions here, can we improve our insert strategy?

$h_1(x)$	10	5	13	17	8	1	6
$h_2(x)$	3	7	20	12	2	9	7
$h_3(x)$	12	6	7	5	9	18	3
$h_4(x)$	4	19	26	1	4	5	1
$h_5(x)$	6	6	8	11	6	7	16

Count Min Sketch Improved Insertion

S = { … , 1, 3, 8, 16, …}

$h_1(k) = k \% 7$ $h_2(k) = k+3(k\%2) \% 7$ $h_3(k) = |k - 4| \% 7$

h_1	7	9	10	1	5	7	6
h_2	4	8	7	5	4	10	7
h_3	15	12	2	5	8	2	1

Count Min Sketch Insertion

Minimal Increase: When inserting, only update the minimum count.

Default

h_1	7	**11**	**11**	**2**	5	7	6
h_2	4	**9**	**8**	5	**5**	10	**8**
h_3	15	**13**	2	**6**	**9**	**3**	1

Minimal Increase

h_1	7	9	10	**2**	5	7	6
h_2	4	**9**	7	5	**5**	10	7
h_3	15	12	2	5	**9**	**3**	1

Count-Min Sketch

A probabilistic data structure storing a set of values

Has **four** key properties:

- k, number of hash functions
- n, expected number of insertions
- m, filter size in ***registers***
- b**, number of bits per register**

$h_{\{1,2,3,\ldots,k\}}$

0	3	1	0
0	0	2	3
2	0	1	0
3	1	0	1
0	0	2	2

Minimal increase reduces overcounting by identifying collisions.

Count value returned here [underestimates / overestimates / matches] the true count of the query.

Trivia Point 1: Deletion is Possible!

$h_1(x)$	0	1	2	2	3	1	1
$h_2(x)$	3	2	0	0	2	2	1
$h_3(x)$	0	1	1	2	4	2	0
$h_4(x)$	4	1	3	0	2	0	0
$h_5(x)$	1	1	5	1	0	1	1

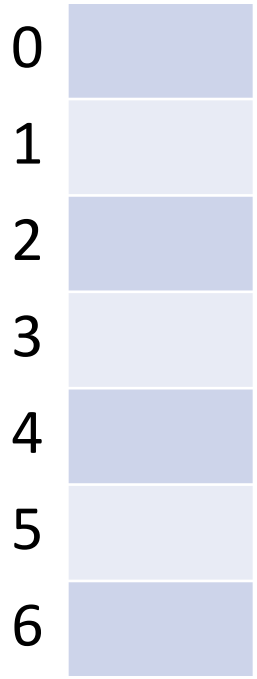
Trivia Point 2: Counting Bloom Filters

$h_1(x)$	10	5	13	17	8	1	6
$h_2(x)$	3	7	20	12	2	9	7
$h_3(x)$	12	6	7	5	9	18	3
$h_4(x)$	4	19	26	1	4	5	1
$h_5(x)$	6	6	8	11	6	7	16

Trivia Point 2: Counting Bloom Filters

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

$h_1(k) = k \% 7$ **$h_2(k) = 2k+1 \% 7$**



Trivia Point 2: Counting Bloom Filters

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

$h_1(k) = k \% 7$ **$h_2(k) = 2k+1 \% 7$**

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

Next Time: Cardinality

Cardinality: how many *distinct* values in a data stream?