Data Structures and Algorithms
Bloom Filters 3 & Cardinality Intro

CS 225
G Carl Evans

April 28, 2025

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

From slides by Brad Solomon

Bloom Filter: Error Rate

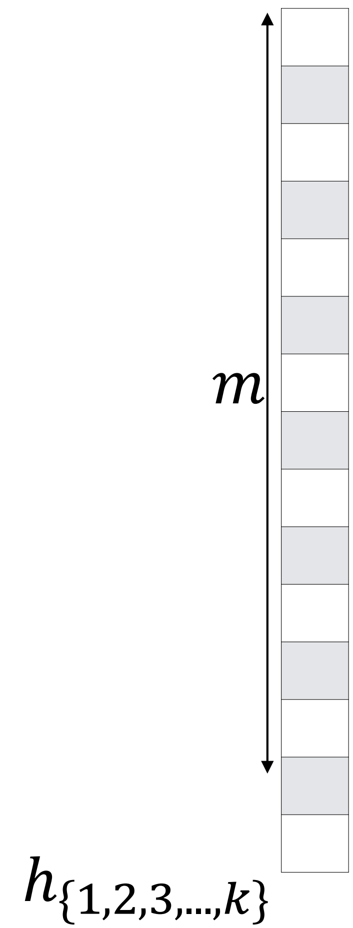
Given bit vector of size m and k SUHA hash function

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



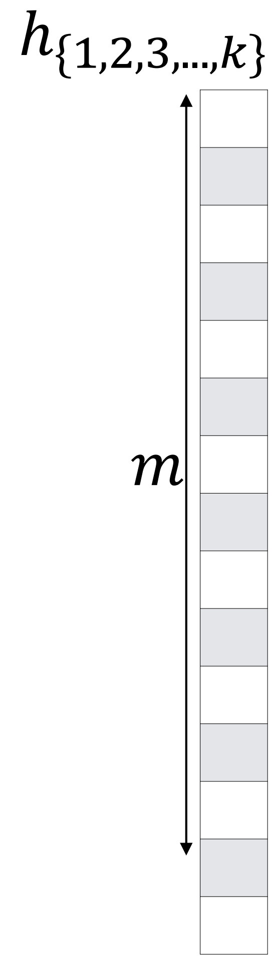
Bloom Filter: Error Rate

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

To build the optimal hash function, fix **m** and **n**!

Claim: The optimal hash function is when $k* = ln2 \cdot \frac{m}{n}$

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

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

Bloom Filter: Optimal Error Rate

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

$$\left(1-\frac{1}{m}\right)^{nk} = e^{ln\left[\left(1-\frac{1}{m}\right)^{nk}\right]}$$

Bloom Filter: Optimal Error Rate

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

$$\left(1-\frac{1}{m}\right)^{nk} = e^{ln\left[\left(1-\frac{1}{m}\right)^{nk}\right]}$$

$$= e^{ln\left[\left(1-\frac{1}{m}\right)\right]nk}$$

Bloom Filter: Optimal Error Rate

Taylors expansion of $ln(1+x)$: $x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$

"Mercator Series"

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

Bloom Filter: Optimal Error Rate

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

$$\left(1-\frac{1}{m}\right)^{nk} = e^{ln\left[\left(1-\frac{1}{m}\right)^{nk}\right]}$$

$$= e^{ln\left[\left(1-\frac{1}{m}\right)\right]nk}$$

$$\approx e^{\frac{-nk}{m}}$$

Bloom Filter: Optimal Error Rate

Claim 2: $\frac{d}{dk}\left(1-e^{\frac{-nk}{m}}\right)^k \approx \frac{d}{dk}\left(kln(1-e^{\frac{-nk}{m}})\right)$

Fact: $\frac{d}{dx}\ln f(x) = \frac{1}{f(x)}\frac{df(x)}{dx}$

TL;DR: $min[f(x)] = min[\ln f(x)]$

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

Bloom Filter: Error Rate

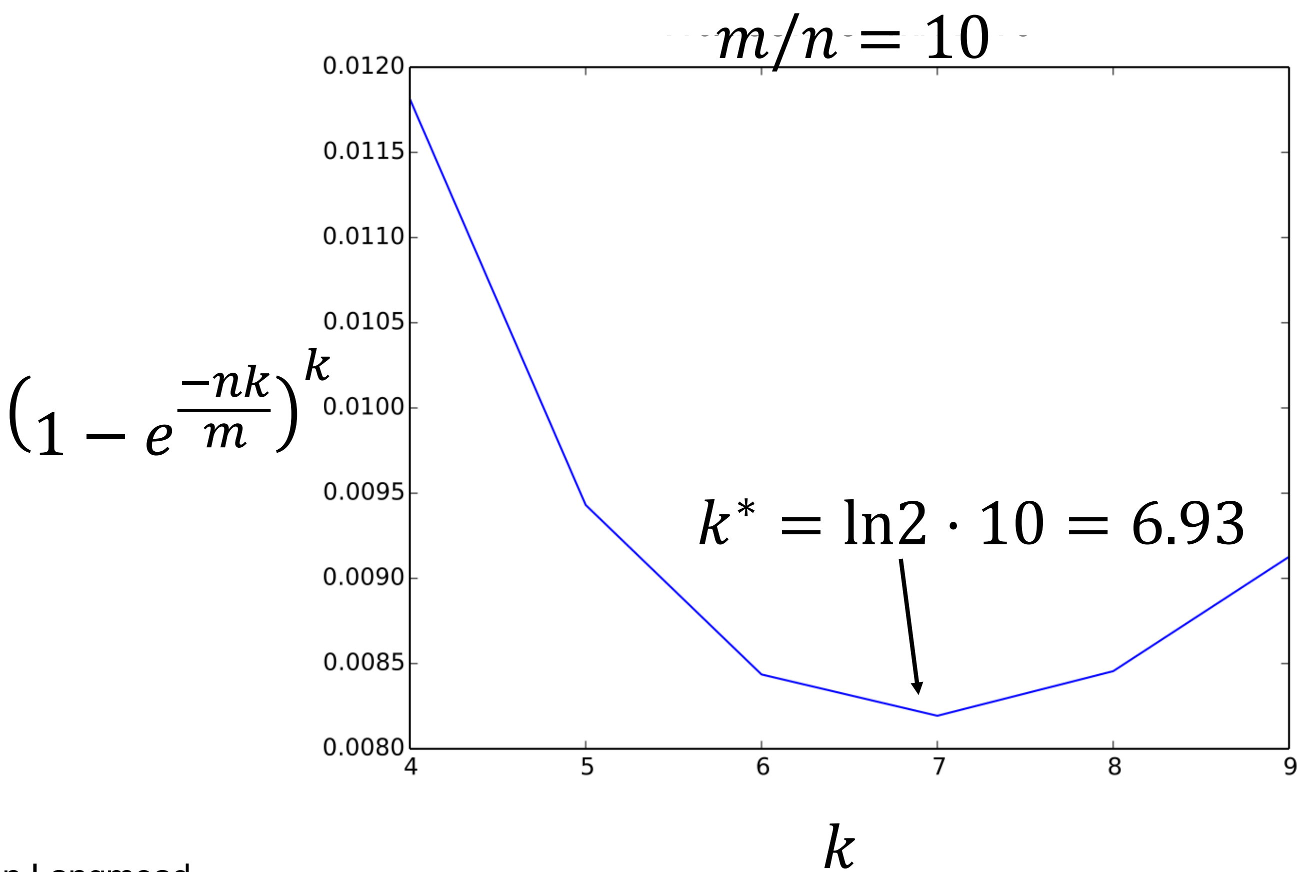


Figure by Ben Langmead

Bloom Filter: Optimal Parameters

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

Given any two values, we can optimize the third

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

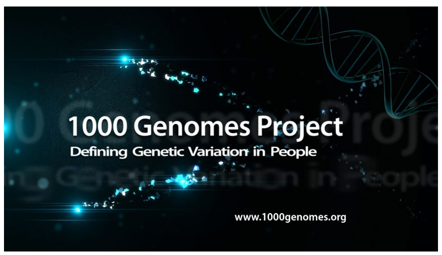
$m = 100$ bits $n = 20$ items $k =$

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

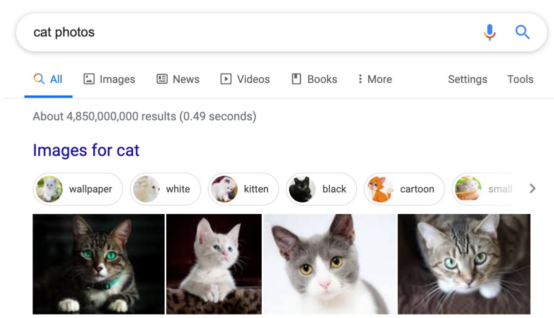
Bloom Filter: Optimal Parameters

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

Optimal hash function is still O(n)



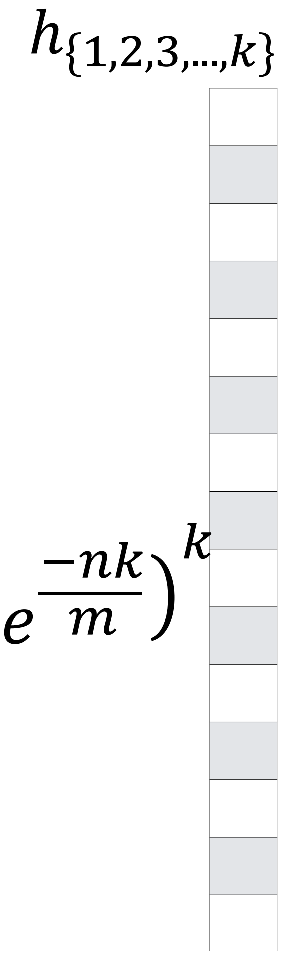
n = 250,000 files vs ~10^{15} nucleotides vs 260 TB



n = 60 billion — 130 trillion

Bloom Filters

A probabilistic data structure storing a set of values



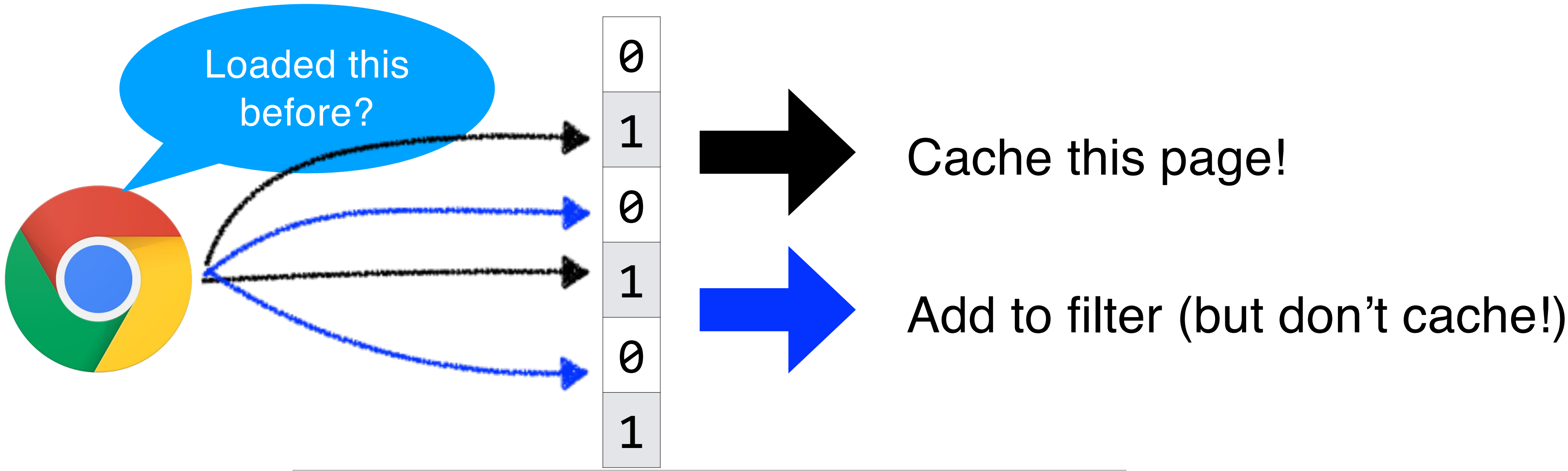
Has three key properties:

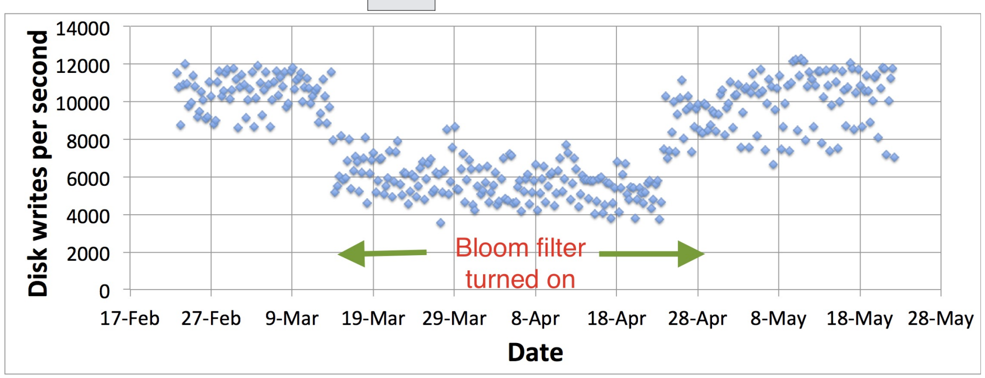
- k, number of hash functions
- n, expected number of insertions
- m, filter size in bits

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

Optimal accuracy when: $$k^{*} = \ln 2 \cdot \frac{m}{n}$$

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.

Bitwise Operators in C++

Let **A = 10110** Let **B = 01110**

```
 ~B:

A & B:

A | B:

A >> 2:

B << 2:
```

Bit Vectors: Unioning

Bit Vectors can be trivially merged using bit-wise union.

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

U

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

=

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Bit Vectors: Intersection

Bit Vectors can be trivially merged using bit-wise intersection.

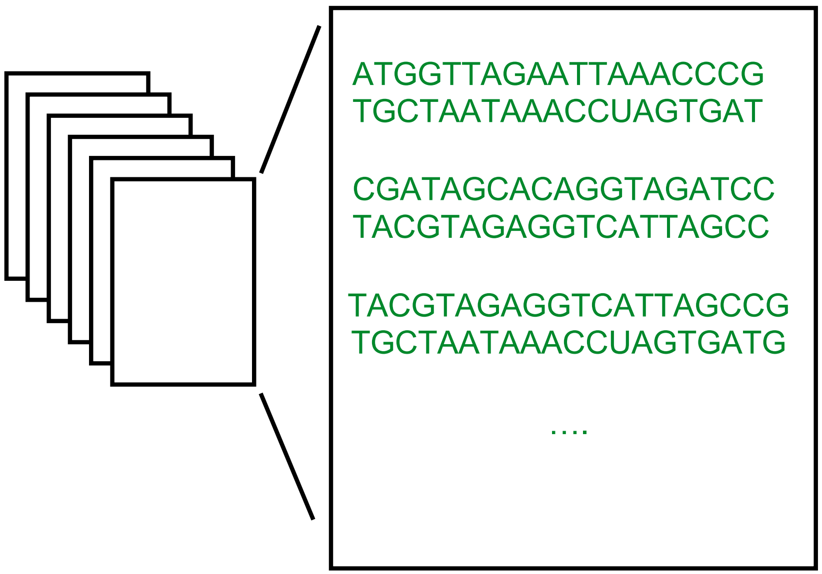
0	1		0	0		0	
1	0		1	1		1	
2	1		2	1		2	
3	1		3	0		3	
4	0	U	4	0	=	4	
5	0		5	0		5	
6	1		6	1		6	
7	0		7	1		7	
8	0		8	1		8	
9	1		9	1		9	

Bit Vector Merging

What is the conceptual meaning behind **union** and **intersection**?

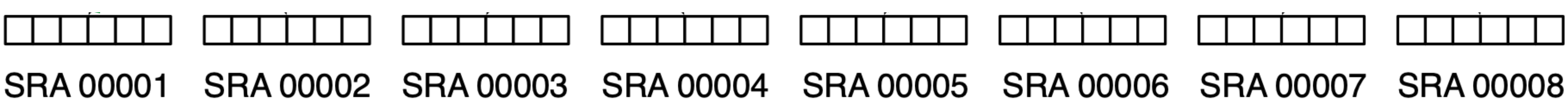
Sequence Bloom Trees

Imagine we have a large collection of text…

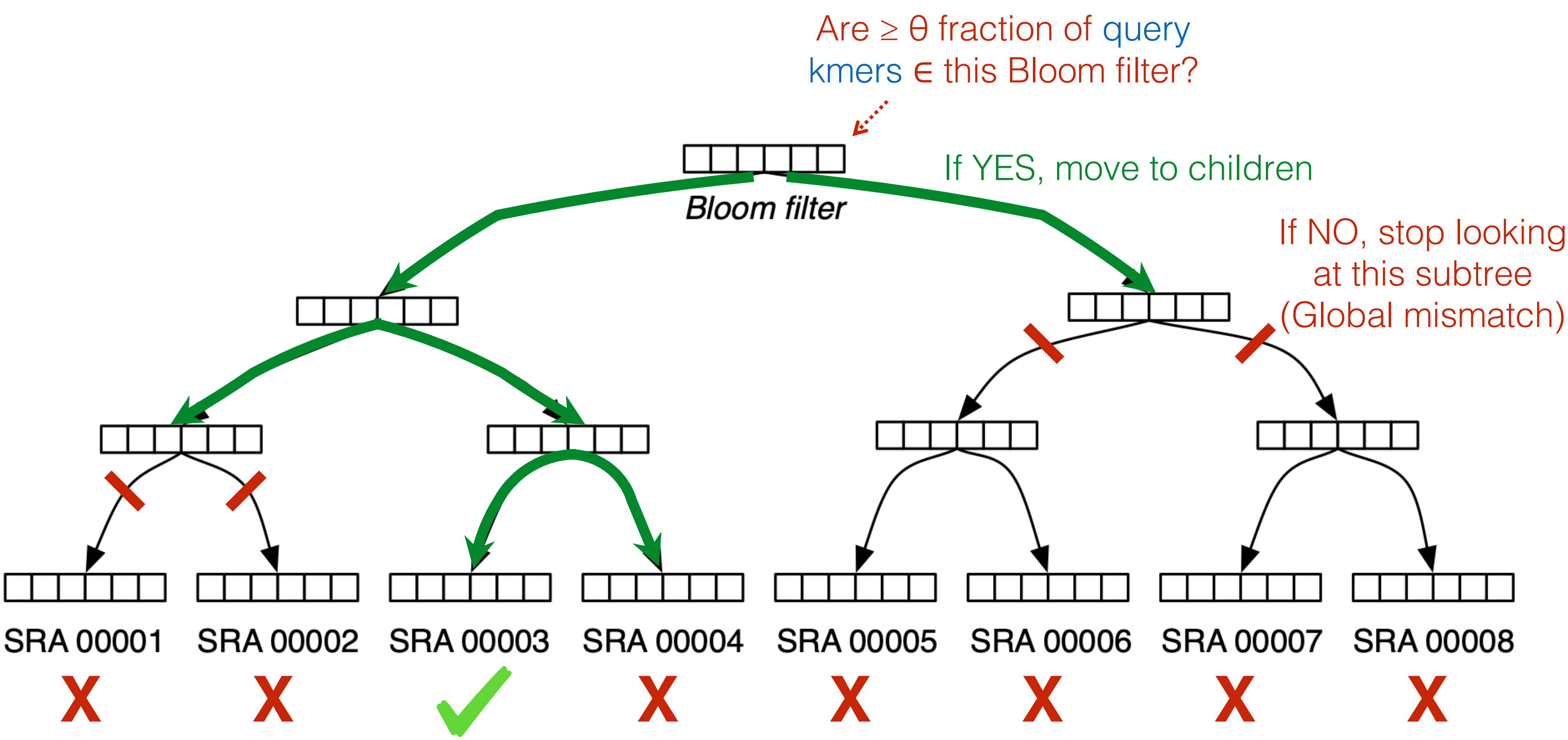


And our goal is to search these files for a query of interest…

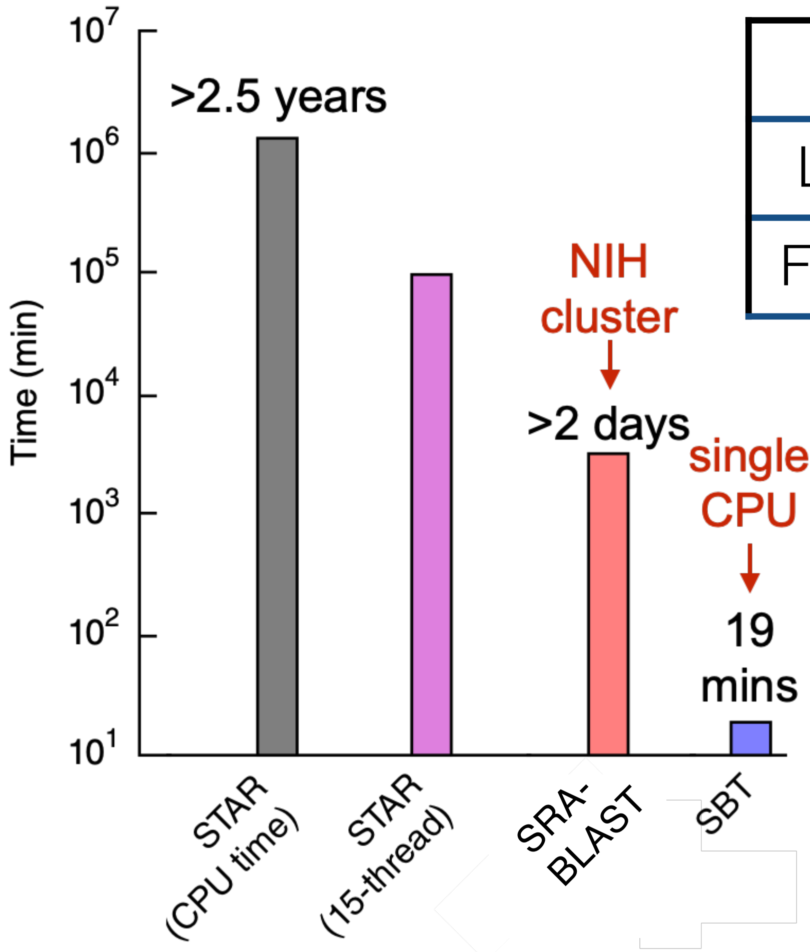
Sequence Bloom Trees



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 co*

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.

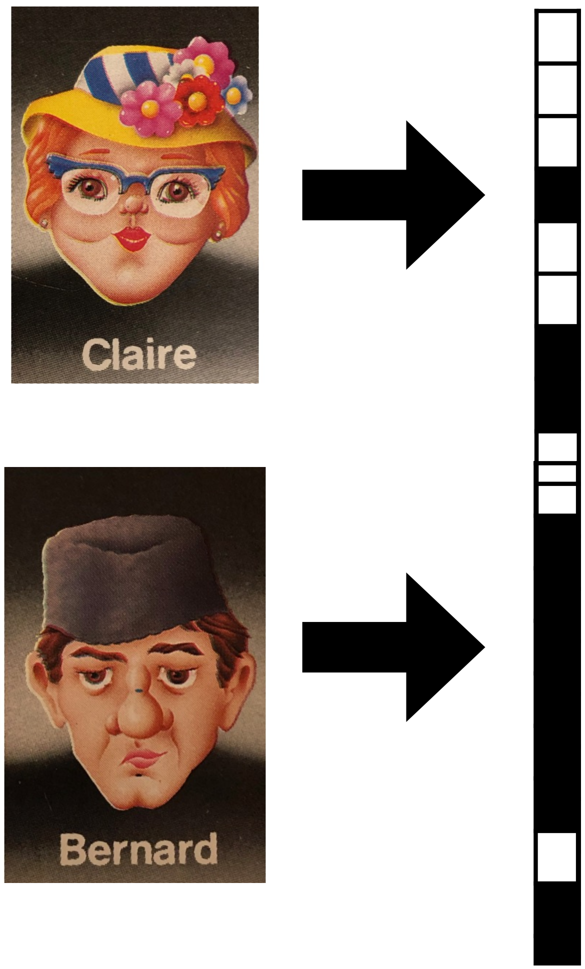
Mitzenmacher, Michael. "Compressed bloom filters." *IEEE/ACM transactions on networking* 10.5 (2002): 604-612

Crainiceanu, Adina, and Daniel Lemire. "Bloofi: Multidimensional bloom filters." *Information Systems* 54 (2015): 3

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…

The hidden problem with (most) sketches…



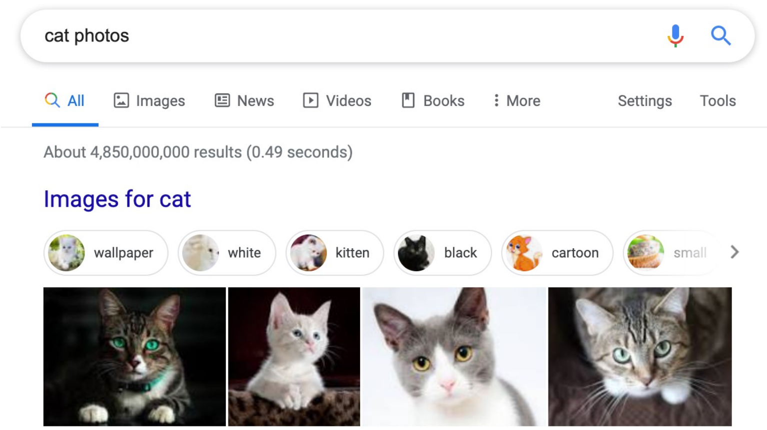
Cardinality

Cardinality is a measure of how many unique items are in a set

2
4
9
3
7
9
7
8
5
6

Cardinality

Sometimes its not possible or realistic to count all objects!



Estimate: 60 billion — 130 trillion

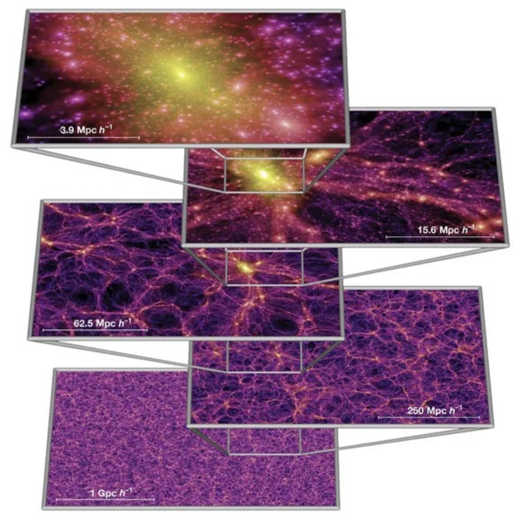


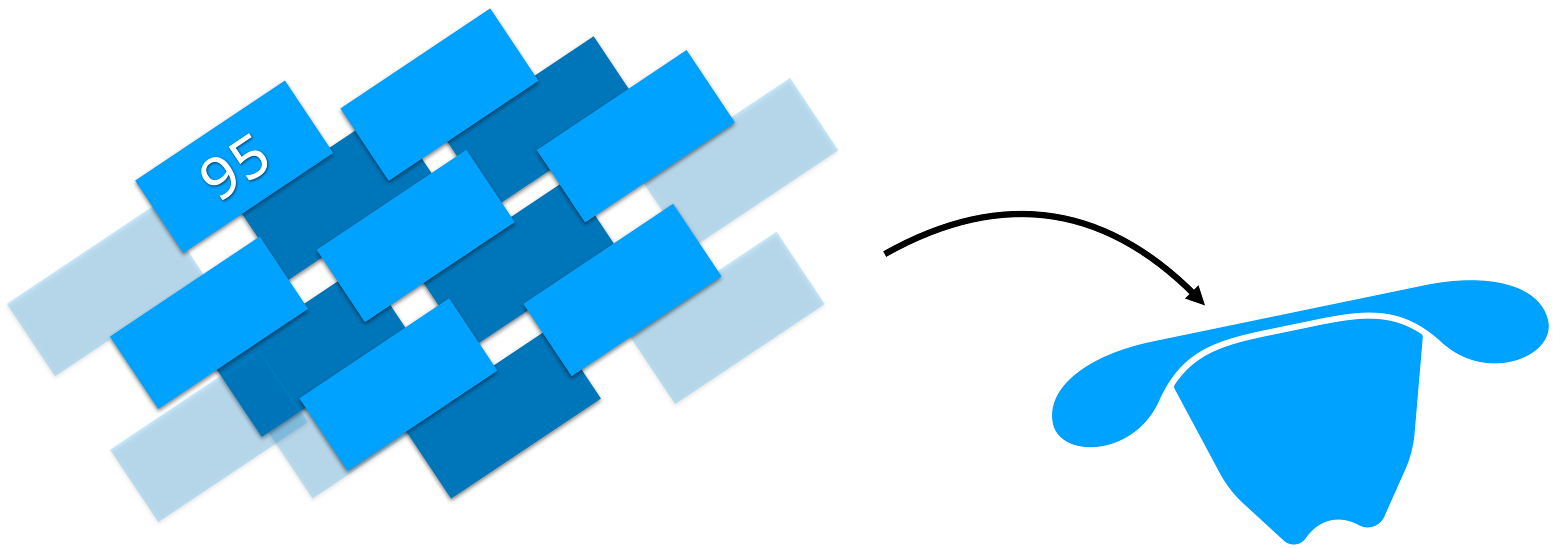
Image: https://doi.org/10.1038/nature03597

946
5581
8945
6145
8126
3887
8925
1246
8324
4549
9100
5598
8499
8970
3921
8575
4859
4960
42
6901
4336

Cardinality Estimation

Imagine I fill a hat with numbered cards and draw one card out at randor

If I told you the value of the card was 95, what have we learned?

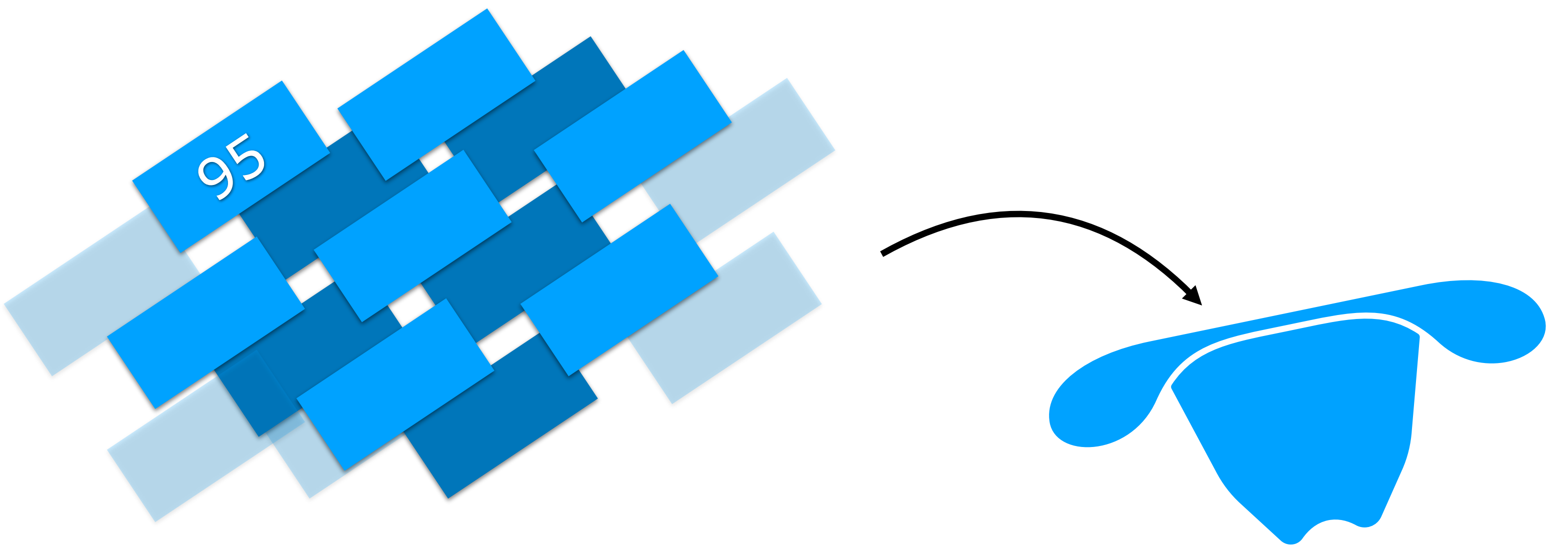


Analogy from Ben Langmead

Cardinality Estimation

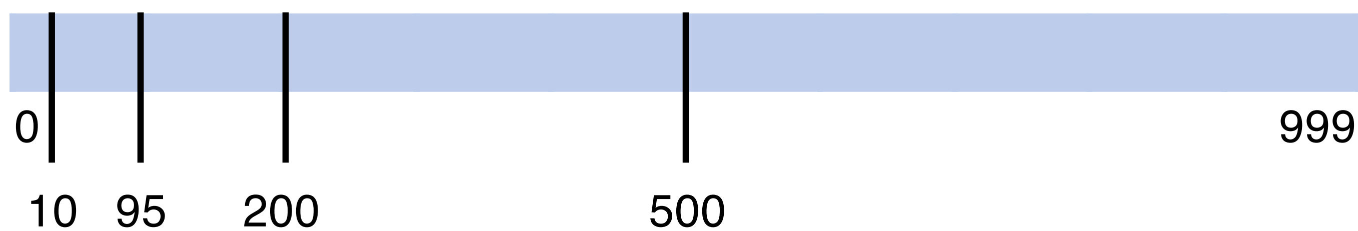
Imagine I fill a hat with **a random subset** of numbered cards **from 0 to 9**

If I told you that the **minimum** value was 95, what have we learned?



Cardinality Estimation

Imagine we have multiple uniform random sets with different minima.



Cardinality Estimation

Let min = 95. Can we estimate N, the cardinality of the set?

Cardinality Estimation

Let min = 95. Can we estimate N, the cardinality of the set?

Claim: $95 \approx \frac{1000}{(N+1)}$

Cardinality Estimation

Let min = 95. Can we estimate N, the cardinality of the set?

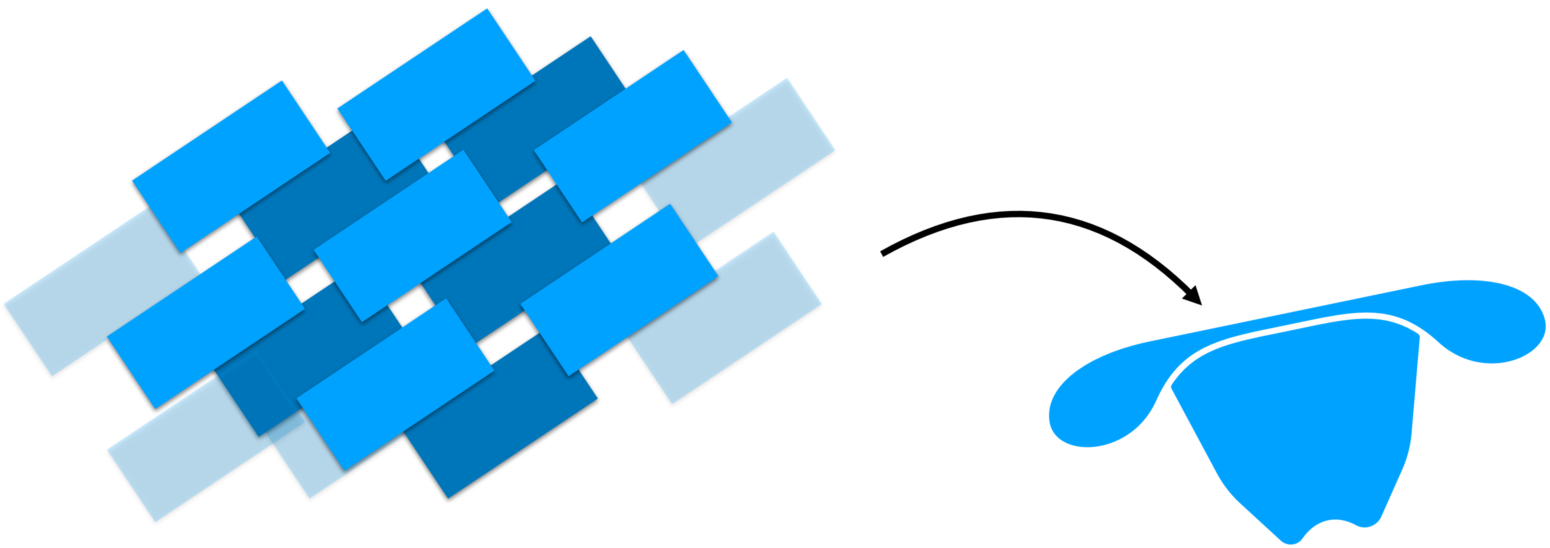
Conceptually: If we scatter N points randomly across the interval, we end up with N + 1 partitions, each about $1000/(N+1)$ long

Assuming our first 'partition' is about average:

$$\begin{aligned} 95 &\approx 1000/(N+1) \\ N+1 &\approx 10.5 \\ N &\approx 9.5 \end{aligned}$$

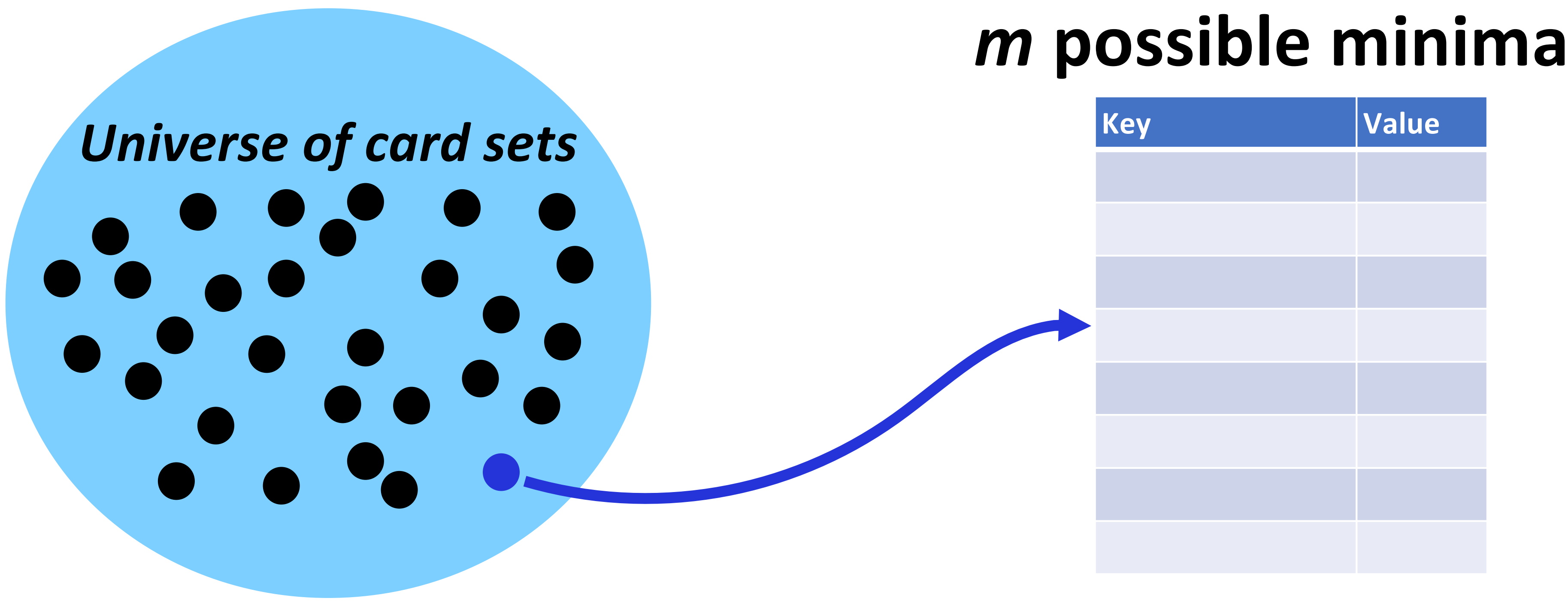
Cardinality Estimation

Why do we care about “the hat problem”?



Cardinality Estimation

Why do we care about “the hat problem”?

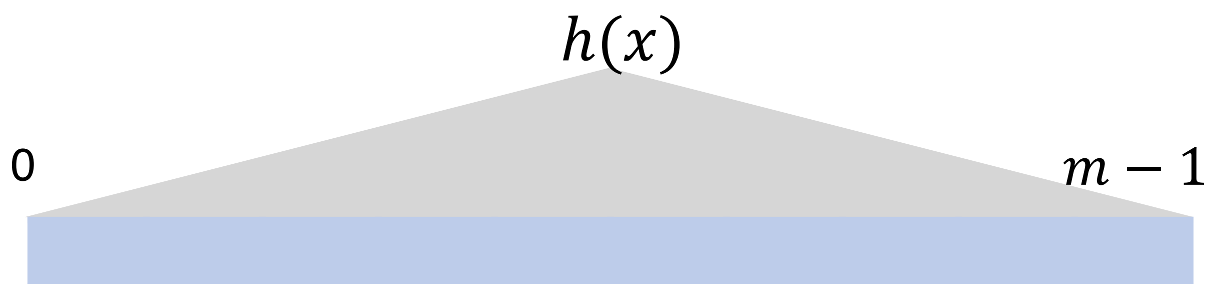


Cardinality Estimation

Imagine we have a SUHA hash h over a range m.

Inserting a new key is equivalent to adding a card to our hat!

Tracking only the minimum value is a **sketch** that estimates the cardinal



Cardinality Estimation

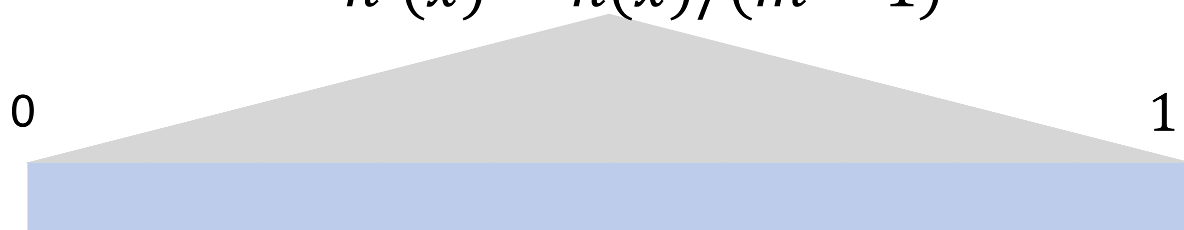
Imagine we have a SUHA hash h over a range m.

Inserting a new key is equivalent to adding a card to our hat!

Tracking only the minimum value is a **sketch** that estimates the cardinal

To make the math work out, lets normalize our hash…

$$h'(x) = h(x)/(m-1)$$



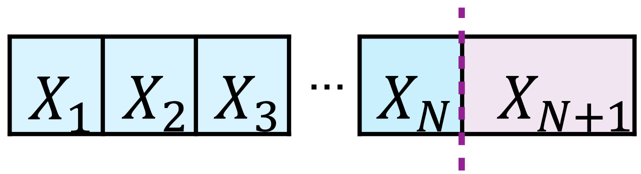
Cardinality Sketch

Let $M = min(X_1, X_2, ..., X_N)$ where each $X_i \in [0,1]$ is an uniform independent random variable

Claim: $\mathbf{E}[M] = \frac{1}{N+1}$

Cardinality Sketch

Consider an $N + 1$ draw:

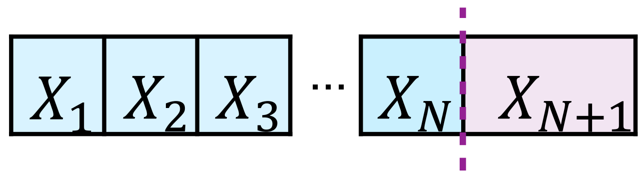


$$M = \min_{1 \leq i \leq N} X_i$$

X_{N+1} can end up in one of two ranges:

Cardinality Sketch

Consider an $N+1$ draw:



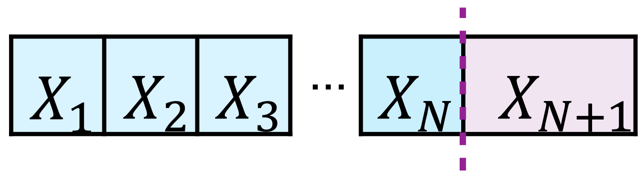
$$M = \min_{1 \le i \le N} X_i$$

X_{N+1} can end up in one of two ranges:

X_{N+1} will be the new minimum with probability M

Cardinality Sketch

Consider an $N+1$ draw:



$$M = \min_{1 \le i \le N} X_i$$

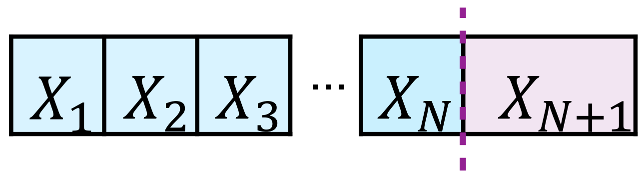
X_{N+1} can end up in one of two ranges:

X_{N+1} will be the new minimum with probability M

X_{N+1} will not change minimum with probability $1 - M$

Cardinality Sketch

Consider an $N + 1$ draw:



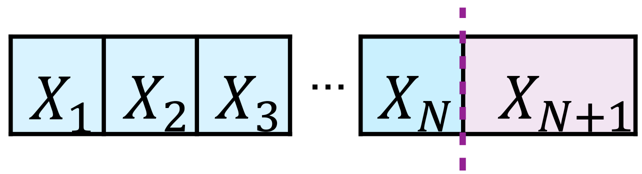
$$M = \min_{1 \le i \le N} X_i$$

X_{N+1} **will be the new minimum with probability** M

By definition of SUHA, X_{N+1} has a $\frac{1}{N+1}$ chance of being smallest item

Cardinality Sketch

Consider an $N+1$ draw:



$$M = \min_{1 \le i \le N} X_i$$

X_{N+1} will be the new minimum with probability M

By definition of SUHA, X_{N+1} has a $\frac{1}{N+1}$ chance of being smallest item

Thus, $\mathbf{E}[M] = \frac{1}{N+1}$

Cardinality Sketch

Claim: $\mathbf{E}[M] = \frac{1}{N+1}$

$$N \approx \frac{1}{M} - 1$$

Attempt 1	0.962	0.328	0.771	0.952	0.923
Attempt 2	0.253	0.839	0.327	0.655	0.491
Attempt 3	0.134	0.580	0.364	0.743	0.931

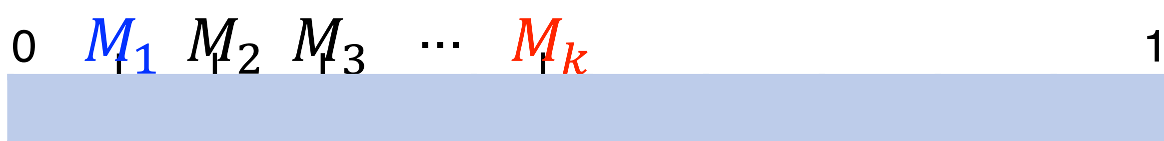
Cardinality Sketch

The minimum hash is a valid sketch of a dataset but can we do better?

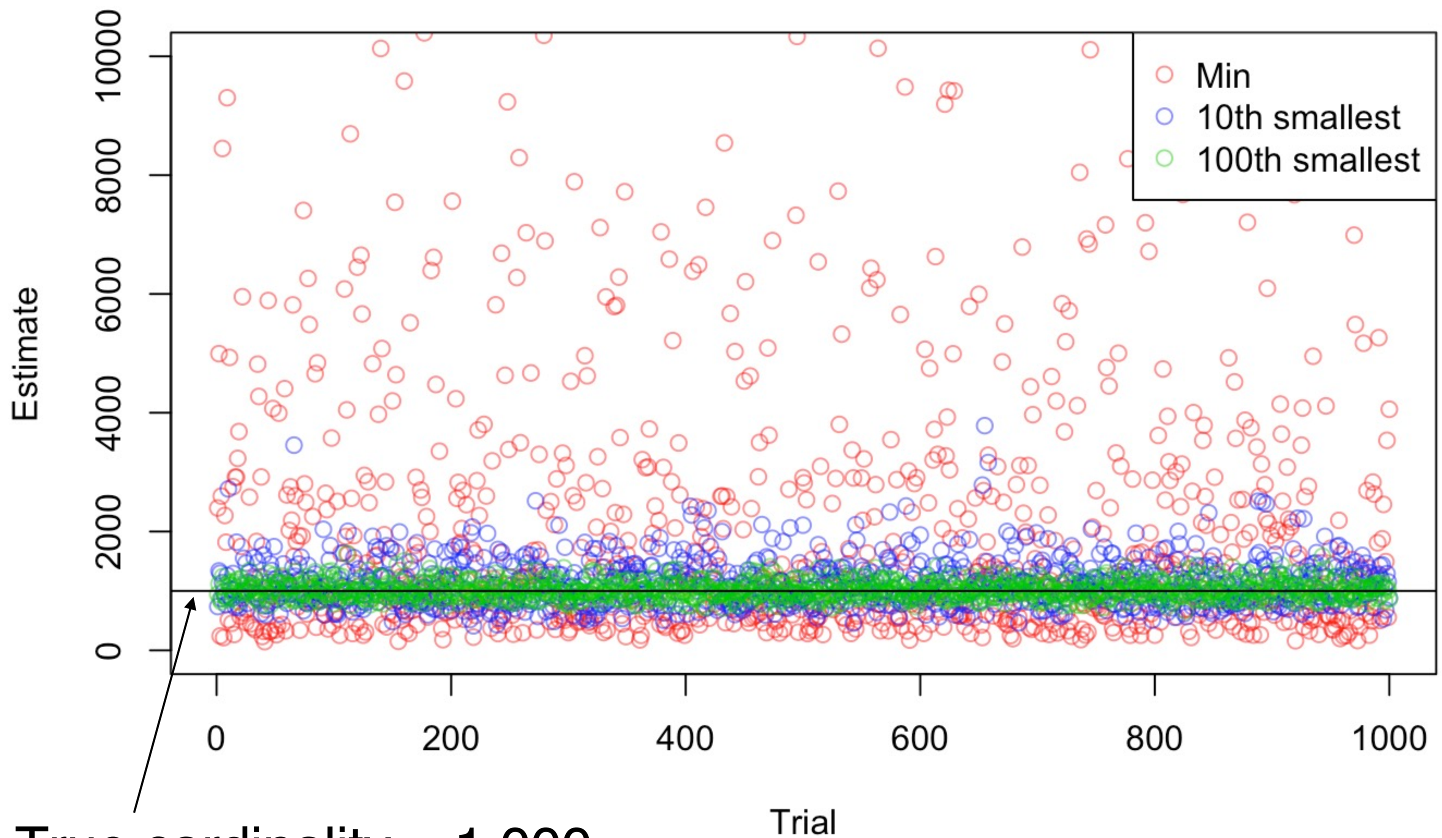
Cardinality Sketch

Claim: Taking the k^{th}-smallest hash value is a better sketch!

Claim: $\mathbf{E}[M_k] = \frac{k}{N+1}$



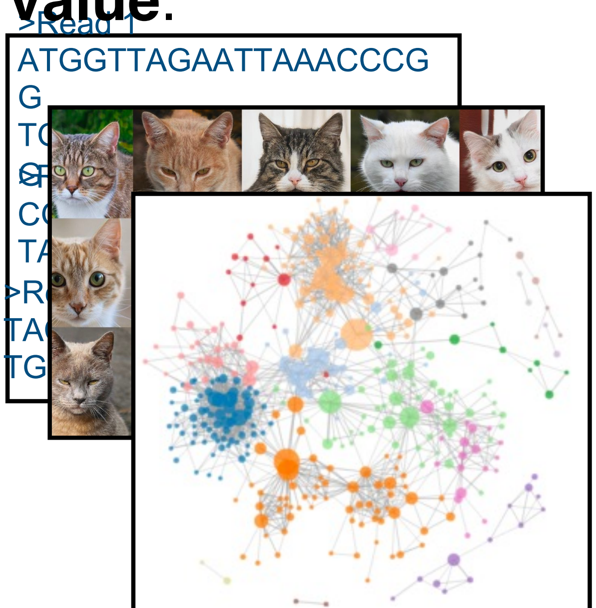
Cardinality Sketch



True cardinality = 1,000

Cardinality Sketch

Given any dataset and a SUHA hash function, we can **estimate the number of unique items** by tracking the **k-th minimum hash value**.



Hash

0.253	0.839	0.327	0.655	0.491

To use the k-th min, we have to track k minima. **Can we use ALL minima?**

Applied Cardinalities

Cardinalities

$$|A|$$

$$|B|$$

$$|A \cup B|$$

$$|A \cap B|$$

Set similarities

$$O = \frac{|A \cap B|}{min(|A|, |B|)}$$

$$J = \frac{|A \cap B|}{|A \cup B|}$$

Real-world Meaning

```
AGGCCACAGTGTATTATGACTG
||||||||||| |||||||||
AGGCCACAGTGAGTTATGACTG

AAAAAAAAAAAGATGT-AAGTA
|||||||||||||||| |||||
AAAAAAAAAAAGATGTAAAGTA

GAGG--TCAGATTCACAGCCAC
||||  ||||||||||||||||
GAGGGGTCAGATTCACAGCCAC
```