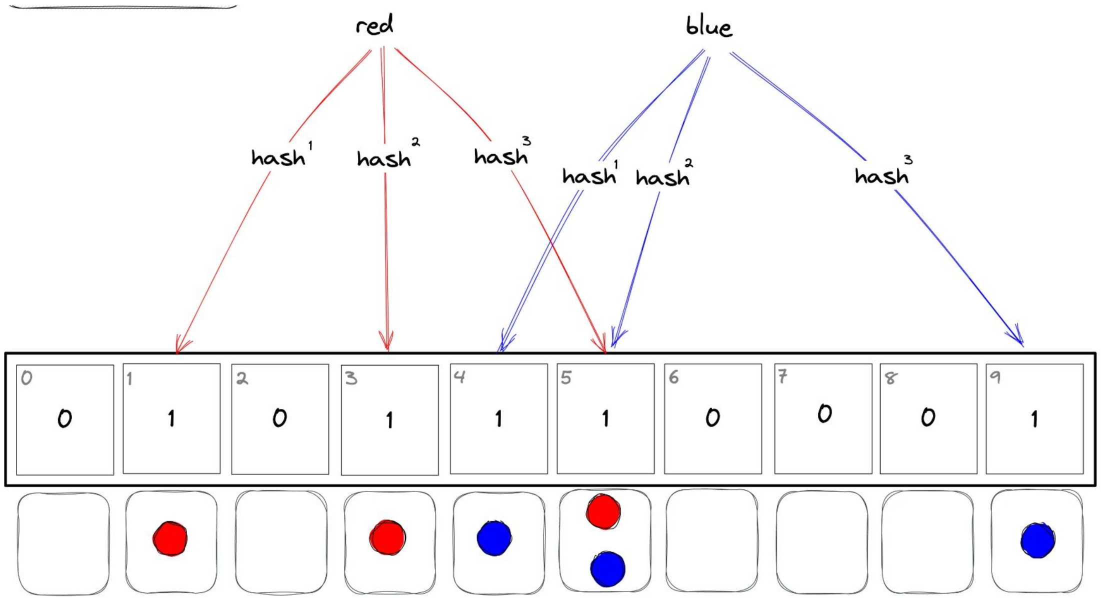
Information theory in data structure

6/4/2025

Carol Hsu

Bloom filter

- bit array
- space-efficient probabilistic data structure
- lossy compression

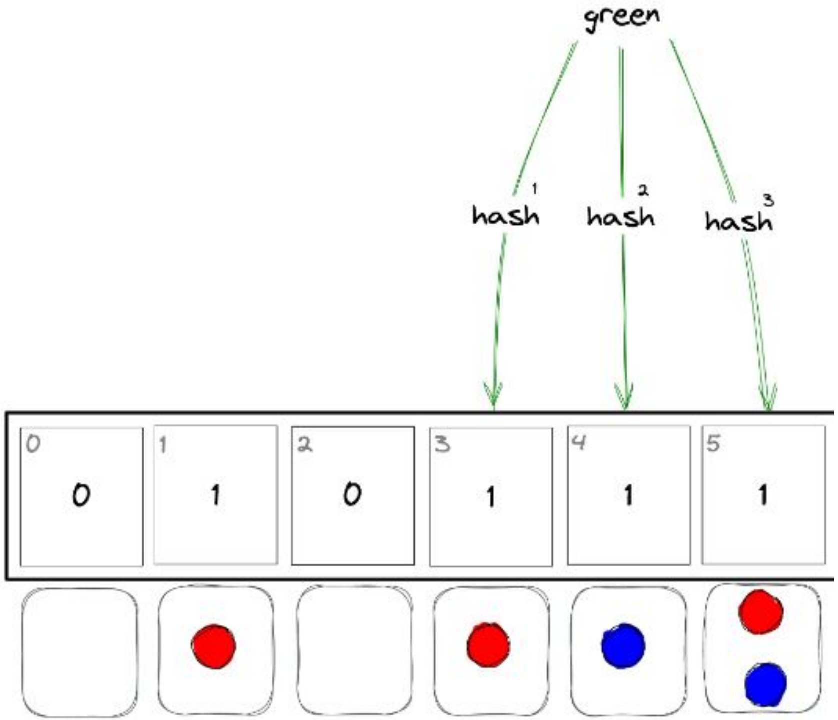


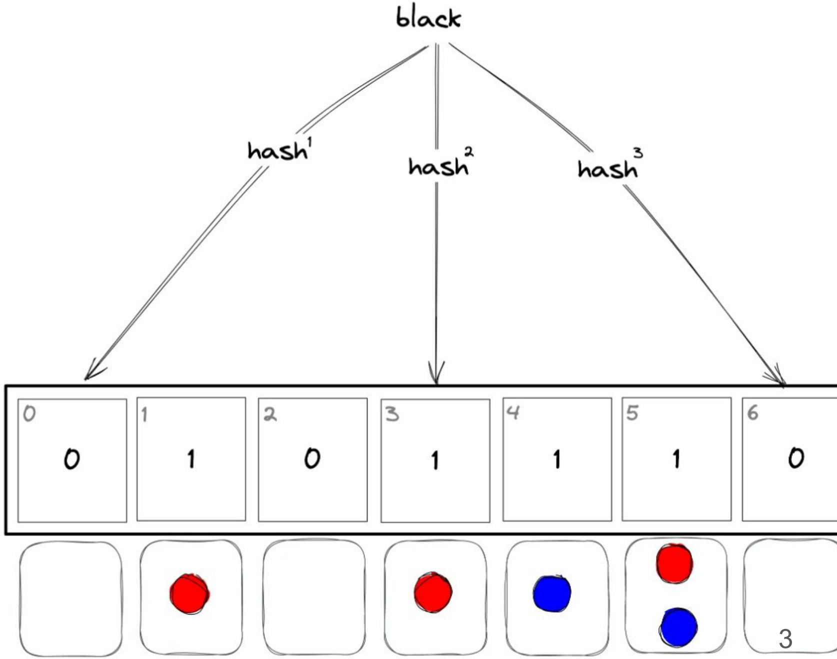
Application:

Web crawler
Blocklist…etc

https://en.wikipedia.org/wiki/Bloom_filter
https://systemdesign.one/bloom-filters-explained/

Bloom filter - check element





What is the minimum number of bits required per element? (Lower bound)

False positive rate = p

the bit array has size **m**,
n elements are inserted

= $$\log_2\left(\frac{1}{p}\right)$$

$$H(x) = -\sum P(x) \cdot \log_2 P(x)$$

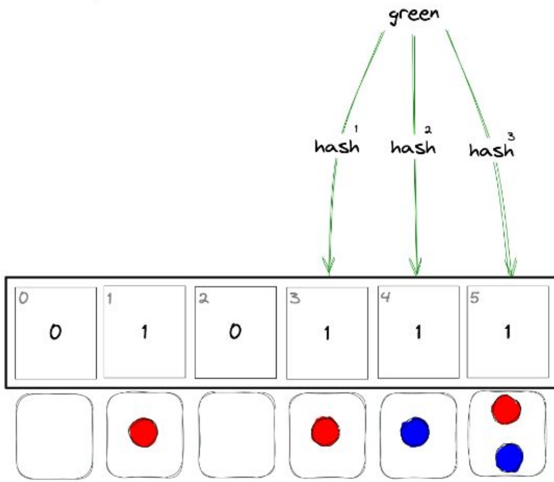
This H(x) is the information-theoretic lower bound of the data:

No lossless compression method can, on average, compress the data to a size smaller than H(x).

At least we need m >= n * log2 (1/p)

False positive rate = p

the bit array has size **m**,
n elements are inserted
k hash functions are used



By Probability, false positive rate $\approx 1 - e^{-kn/m}$.

How to maximize the correctness of set membership representation (minimize false positives) under a fixed space constraint (m bits).

$$k = \frac{m}{n} \ln 2$$

https://en.wikipedia.org/wiki/Bloom_filter#Probability_of_false_positives

the bit array has size **m**,
n elements are inserted
k hash functions are used

Actually bit need per element

$$p \approx \left(1 - e^{-kn/m}\right)^k$$

$$k = \frac{m}{n} \ln 2 \Rightarrow k \cdot n/m = \ln 2$$

$$p \approx \left(1 - e^{-\ln 2}\right)^{\frac{m}{n} \ln 2}$$

$$p = 2^{-\frac{m}{n} \ln 2}$$

$$\log_2 p = -\frac{m}{n} \ln 2$$

$$\Rightarrow \frac{m}{n} = -\frac{\log_2 p}{\ln 2} = \frac{\log_2(1/p)}{\ln 2}$$

$$\boxed{\frac{m}{n} \approx \frac{\log_2(1/p)}{0.693} \approx 1.44 \cdot \log_2\left(\frac{1}{p}\right)}$$

Example:

Number of elements n=1,000,000

Target false positive rate p=0.01 (1%)

$$\frac{m}{n} \approx 1.44 \cdot 6.644 \approx 9.56 \text{ bits per element}$$

$$m = 9.56 \cdot 1,000,000 = 9,560,000 \text{ bits} = \frac{9,560,000}{8 \times 1024^2} \approx 1.14\,\mathrm{MB}$$

$$k = \frac{m}{n} \cdot \ln 2 = 9.56 \cdot 0.693 \approx 6.63$$

Bloom Filters use only **~44% more space** than the **information-theoretic lower bound**, making them highly space-efficient.

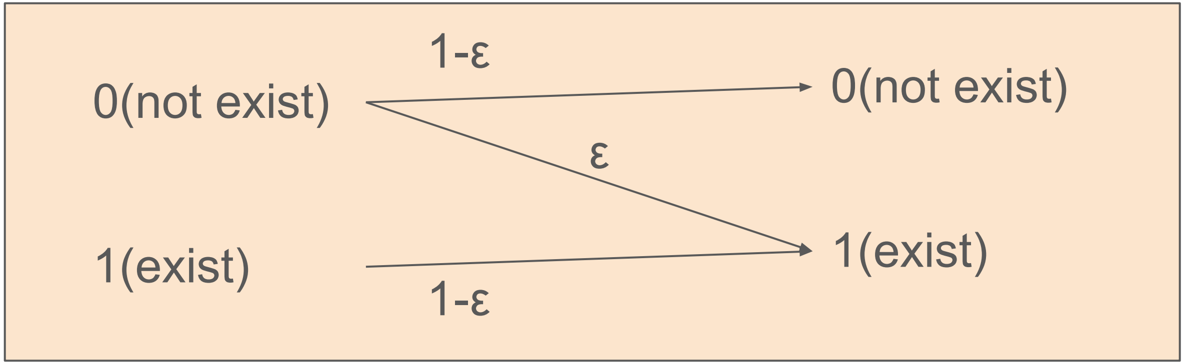
Check element

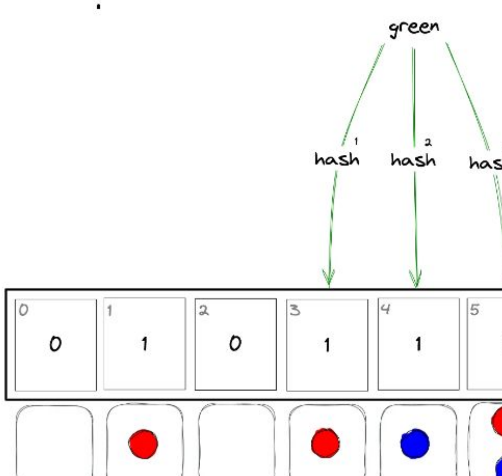
False positive rate = ε
$\Pr(X=1)=p, \ \Pr(X=0)=1-p$

(only false positive, not BSC)

Real element X

Bloom filter Output Y





False positive rate = ε

Check element - channel capacity

how much 'element membership information' can be reliably transmitted through this channel?

$$C = \max_{p} I(X;Y)$$

- $I(X;Y) \to H(X)$ when $\varepsilon \to 0$: No false positives → Perfect information.
- $I(X;Y) \to 0$ when $\varepsilon \to 1$: All queries say "possibly present" → Useless results.

Check element - channel capacity

how much 'element membership information' can be reliably transmitted through this channel?

$$C(\varepsilon) = \max_{0 \le p \le 1} H_b(p \cdot (1 - \varepsilon)) - p \cdot H_b(\varepsilon)$$

Z channel https://en.wikipedia.org/wiki/Z-channel_(information_theory)

Where:

- $p = \Pr(X = 1)$ is the probability the element is in the set
- $H_b(x) = -x \log_2 x - (1 - x) \log_2 (1 - x)$ is the binary entropy function

False positive rate = ε

the bit array has size **m**,

n elements are inserted

Back to design bloom filter

We at least need $n \cdot \log_2 \left(\frac{1}{\varepsilon} \right)$ bits

Each bit in the Bloom filter carries C(ε) bits of useful information (from the channel capacity), then the minimum number of bits mmm required is:

$$m \geq \frac{n \cdot \log_2 \left(\frac{1}{\varepsilon} \right)}{C(\varepsilon)}$$

False positive rate = ε

the bit array has size **m**,

n elements are inserted

Example

ε=0.01

Recall C(ε)= max[H(p(1−ε))−p · H(ε)],

C(0.01) = 0.92

We get m

$$m \geq \frac{n \cdot 6.64}{0.92} \approx 7.2n \text{ bits}$$

Which matches

$$m = \frac{n \cdot \ln(1/\varepsilon)}{(\ln 2)^2} \approx 1.44n \cdot \log_2\left(\frac{1}{\varepsilon}\right)$$

Practical way

https://hur.st/bloomfilter/

Bloom Filter Calculator

Bloom filters are space-efficient probablistic data structures used to test whether an element is a member of a set.

They're surprisingly simple: take an array of **m** bits, and for up to **n** different elements, either test or set **k** bits using positions chosen using hash functions. If all bits are set, the element *probably* already exists, with a false positive rate of **p**; if any of the bits are not set, the element *certainly* does not exist.

Bloom filters find a wide range of uses, including tracking which articles you've read, speeding up Bitcoin clients, detecting malicious web sites, and improving the performance of caches.

This page will help you choose an optimal size for your filter, or explore how the different parameters interact.

---

n

Number of items in the filter (optionally with SI units: k, M, G, T, P, E, Z, Y)

4000

p

Probability of false positives, fraction between 0 and 1 or a number indicating 1-in-p

1.0E-7

m

Number of bits in the filter (or a size with KB, KiB, MB, Mb, GiB, etc)

k

Number of hash functions

Submit

n = 4,000
p = 0.0000001 (1 in 9,994,297)
m = 134,191 (16.38KiB)
k = 23

Reference

https://en.wikipedia.org/wiki/Bloom_filter#Probability_of_false_positives

https://www.eecs.harvard.edu/~michaelm/postscripts/im2005b.pdf

On the computation of Shannon Entropy from Counting Bloom Filters

https://randall.math.gatech.edu/AlgsF09/bloomfilters.pdf

https://commons.apache.org/proper/commons-collections/bloomFilters/intro.html#fn5

Optimizing Bloom Filter: Challenges, Solutions, and Comparisons