Lecture 10 — Feb 12, 2025

Prof. Prashant Pandey *Scribe: Yibo Zhao*

1 Overview

In the last lecture, we went over perfect hashing, cuckoo hashing, two choice hashing, and iceberg hashing.

In this lecture, we will talk about different types of filters.

2 Filters

Filters represent a set approximately. It is a trade off of accuracy for space efficiency. There some different types of filters, some are dynamic, which means data can inserted at runtime, these are

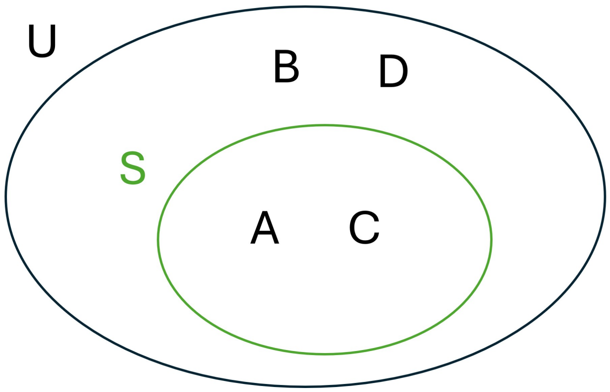
- Bloom Filter
- Quotient Filter
- Cuckoo Filter

Bloom filter does not support deletion of inserted keys, while the other two filters can.

There are also static filters, which means they can only be constructed on a static set of known data, and does not support insertion or deletion. These are,

- Xor Filter
- Ribbon

These five types of filter and their variants covers most of the hundreds of filters that have been developed over the years.



A, C: always True

B, D: True or False

Has False positive, but no False negative

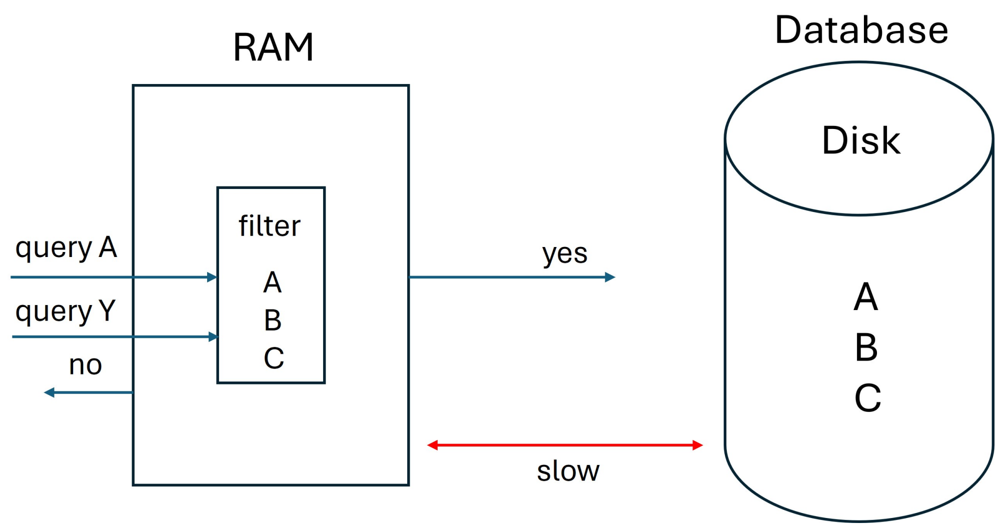
Formally, a filter guarantees a false positive rate $\epsilon (0 < \epsilon < 1)$. If a key $q \in S$, a filter return True with probability 1. If a key $q \notin S$, a filter returns False with probability $> 1 - \epsilon$, and returns True with probility $\leq \epsilon$

2.1 Space Usage

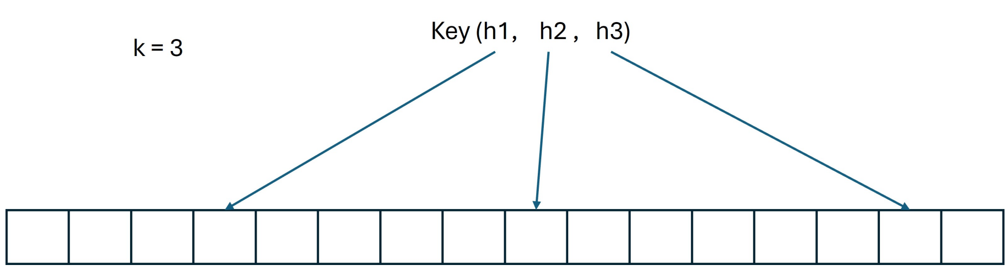
- Hash Table: For a universe U, use $\Omega(n \log(|U|))$ bits. For $U = 64$ bits, and number of keys $n = 2^{30}$, it uses 2^{36} bits.
- Filter: Use $\geq n \log \frac{1}{\epsilon}$ bits. For $\epsilon \approx 2\%$, and number of keys $n = 2^{30}$, it uses $2^{30} \cdot 6$ bits. This is much less then hash table's space requirement.

2.2 Filter Use Case

Database, Storage System, Network System, etc.



3 Bloom Filter



A bloom filter consist of a bit vector of size m and k hash functions.

- Query: If all k bits are 1, returns True. If any of the k bits is 0, returns false.

- Space $\approx 1.44n \log \frac{1}{\epsilon}$

3.1 False Positive Analysis

Let **n** be the number of items, **m** be the number of bits, and **k** be the number of hash functions.

$$\mathbb{P}[\text{a certain bit is not set to 1 by a certain hash function}] = 1 - \frac{1}{m} \tag{1}$$

$$\mathbb{P}[\text{a certain bit is not set to 1 by any hash functions}] = (1 - \frac{1}{m})^k \tag{2}$$

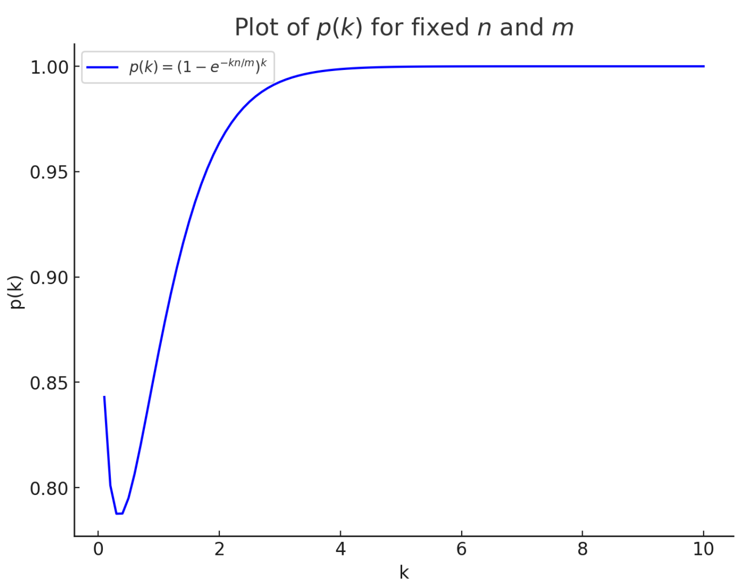
Using the equation, $\boxed{\lim_{m \to \infty} (1 - \frac{1}{m})^m} = \frac{1}{e}$

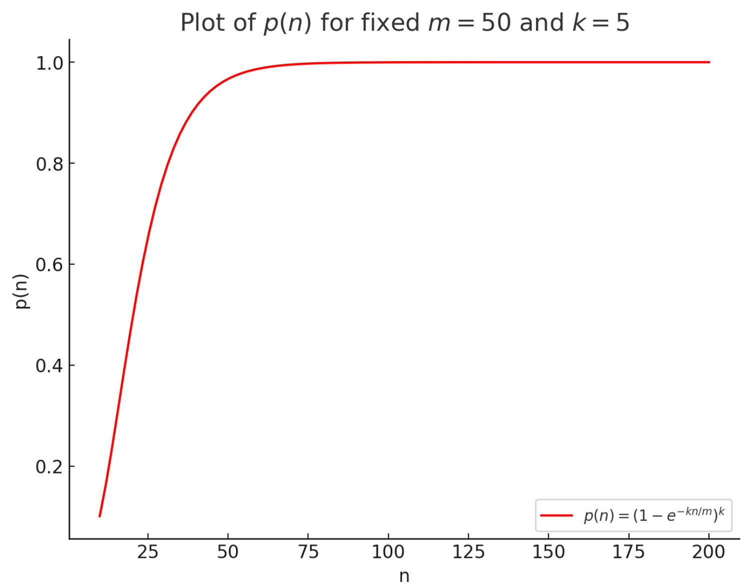
$$\mathbb{P}[\text{a certain bit is not set to 1 after n insertion}] = (1 - \frac{1}{m})^{kn} = \exp(-\frac{kn}{m}) \tag{3}$$

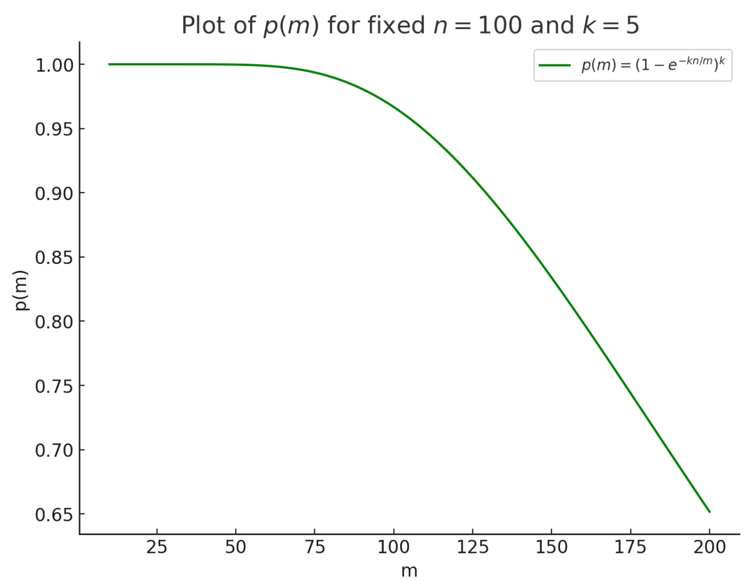
$$\mathbb{P}[\text{a certain bit is set to 1 after n insertion}] = 1 - \exp(-\frac{kn}{m}) \tag{4}$$

$$\mathbb{P}[\text{all k bits are set to 1 after n insertion}] = (1 - \exp(-\frac{kn}{m}))^k \tag{5}$$

The following three plots illustrate how m,n and k affects the probability of all k bits are set to 1 after n insertion.







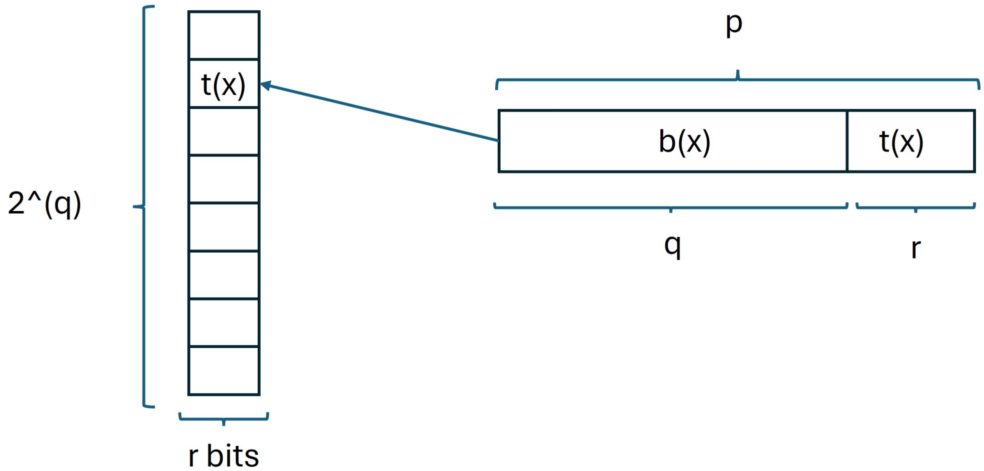
We always want to use as few hash functions as possible for both compute and space efficiency. We can solve for a local minima of k as a function of m and n, $\boxed{k = \frac{m}{n} \ln 2}$. In practice, using $k = 7$ is generally a good starting point.

4 Single Hash Filter

Use a single p-bit hash function, h. For a U size universe, each $\log |U|$ bits key x is hashed into a p bits **finger print**. Only the finger print is stored in a hash table. This is space efficient because $p < \log |U|$.

The only source of false positive: Two distinct key x and y, where h(x) = h(y). If x is part of the set and y is not, query for y will be a false positive with $\mathbb{P}[\text{x and y collide}] = \frac{1}{2^p}$

4.1 Quotient



$\log |U| < p < r$

$$\begin{aligned}
\mathbb{P}[\text{False positive after n insertion}] &= \frac{n}{2^p} && (6)\\
&= \frac{n}{2^q + 2^r} && (7)\\
&= \frac{2^q}{2^q + 2^r} && (8)\\
&= \frac{1}{2^r} && (9)
\end{aligned}$$

References

[1] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970. `https://doi.org/10.1145/362686.362692`.