**CS 7800/4810: Advanced Algorithms** Spring 2026

Lecture 14 — Mar 9, 2026

*Prof. Prashant Pandey* *Scribe: Anh Nguyen*

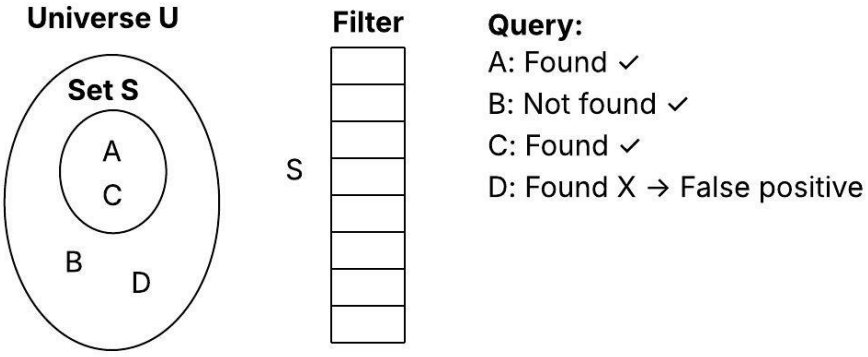
# 1 Filters

Filters represent a set approximately. It is a trading of accuracy for space efficiency. There are different types of filters (dynamic or static). 'Dynamic' meaning data can be inserted at runtime. Some dynamic filter examples:

- Bloom filter [1] - doesn't support deletion
- Quotient filter - support deletion
- Cuckoo filter [2] - support deletion

Static filters:

- XOR filter
- Ribbon filter




A filter guarantees **no false negatives**, meaning if it reports an element is not in the set it is always correct, but it may produce **false positives**, where it reports an element is in the set even though it was never inserted.

A filter guarantees a false positive rate $\epsilon$ (where $0 \leq \epsilon \leq 1$).
If $q \in$ set $S$, returns True with probability Pr $= 1$.
If $q \notin S$, returns False with probability Pr $> 1 - \epsilon$, True with probability Pr $\leq \epsilon$.

### 1.1 Space Usage

Space usage $\geq n \lg \frac{1}{\epsilon}$ bits. (lowerbound)

- $n$ = number of elements in the set
- $\epsilon$ = allowed false positive probability

Reducing the false positive rate requires more space.

$$\epsilon \downarrow \quad \Rightarrow \quad \lg\left(\frac{1}{\epsilon}\right) \uparrow \quad \Rightarrow \quad \text{more bits}$$

For most practical settings, $\epsilon \approx 1\%$, which corresponds to roughly **8 bits per stored item**.

#### 1.1.1 Comparison: Dictionary (Hash Table)

To store n items in U universe, a hash table's space usage is $\Omega(n \lg |U|)$ bits.

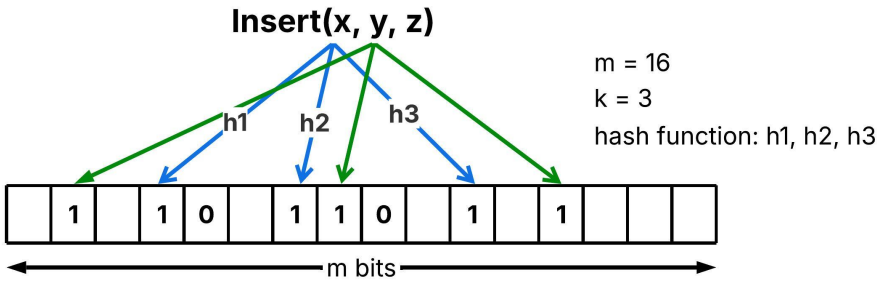
In many applications, $|U| = 2^{64}$ or $2^{128}$, while $|S| = 2^{32}, 2^{34}, 2^{40}$, so $|U - S| \approx |U|$, meaning the universe is much larger than the stored set.

Using Universal Hashing, a random element collides with probability $Pr = \frac{1}{m}$, where $m$ is the number of hash table slots.

Bloom filters exploit this gap to achieve much smaller space usage than storing exact keys.

## 2 Bloom Filter

A Bloom Filter consists 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 usage**: $\sim 1.44 \cdot n \cdot \lg \frac{1}{\epsilon}$ bits

Standard Bloom filters do **not support deletion**. Clearing a bit during deletion could remove a bit that was set by another element, which would cause a **false negative**.

*Counting Bloom Filter*: To support deletion, each slot stores a small counter instead of a single bit.

- Insert: increment the $k$ counters
- Delete: decrement the $k$ counters
- Query: check if all $k$ counters are $> 0$

## 3 False Positive Analysis

Two parameters: $m$ (no of bits in the filter), $k$ (no of hash functions), $n$ (no of items).

$$\mathbb{P}(\text{a certain/specific bit is } \textbf{not} \text{ set to 1 by a } \textbf{certain} \text{ hash function}) = 1 - \frac{1}{m}$$

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

Each insertion uses $k$ hashes, so there are $kn$ total hash operations.

$$\mathbb{P}(\text{a certain bit is } \textbf{not} \text{ set to 1 } \textbf{after n insertions}) = (1 - \frac{1}{m})^{k \cdot n}$$

Rewrite:

$$= \left(1 - \frac{1}{m}\right)^{m \cdot \frac{kn}{m}}$$

Using the limit

$$\lim_{m \to \infty} \left(1 - \frac{1}{m}\right)^m = \frac{1}{e}$$

we approximate

$$= \left(\frac{1}{e}\right)^{\frac{kn}{m}} + e^{-\frac{kn}{m}}$$

$$\mathbb{P}(\text{a certain bit is set to 1 after n insertions}) = 1 - e^{-\frac{kn}{m}}$$

A **false positive** occurs when all $k$ queried bits are 1.

$$\mathbb{P}(\text{all k bits are set to 1 after n insertions}) \ (\textbf{False positive rate}) = \left(1 - e^{-\frac{kn}{m}}\right)^k$$

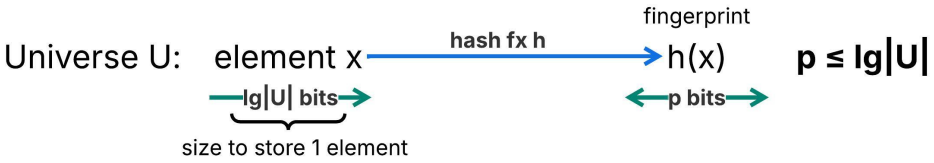
For a given value of m, n, there is a sweet spot for k.

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

At this optimal value, $\epsilon = 2^{-k}$.

# 4 Fingerprint Filter (Single Hashing)

- Use a single hash function to hash items to a $p$-bit fingerprint.
- Store fingerprints compactly in a hash table.
- Now, instead of using $\lg |U|$ bits to store one element, we store only a $p$-bit fingerprint.
- Collisions may occur when many keys from a large universe are mapped into a small $p$-bit space.




Two distinct items $x$ and $y$, they collide if $h(x) = h(y)$.

If $x \in S$ and $y \notin S$, and $y$ is queried, the filter will return `True`, causing a false positive.

The false positive probability is

$$\epsilon = \frac{1}{2^p}$$

Thus, when mapping elements from set $S$ to $p$-bit fingerprints, the probability that an element **not** in the set collides with an **existing** fingerprint is

$$\mathbb{P}(\text{collision}) = \frac{1}{2^p}$$

## 4.1 Space usage

Each stored item keeps a $p$-bit fingerprint, so the total space is

$$\Omega(n \cdot \lg 2^p) \text{ bits}$$

Since

$$\epsilon = \frac{1}{2^p}, \qquad p = \lg\left(\frac{1}{\epsilon}\right),$$

the space becomes

$$\Omega\left(n \cdot \lg\left(\frac{1}{\epsilon}\right)\right) \text{ bits.}$$

In an actual implementation, the total space may be written as

$$n \cdot \lg\left(\frac{1}{\epsilon}\right) + O(n) \text{ bits,}$$

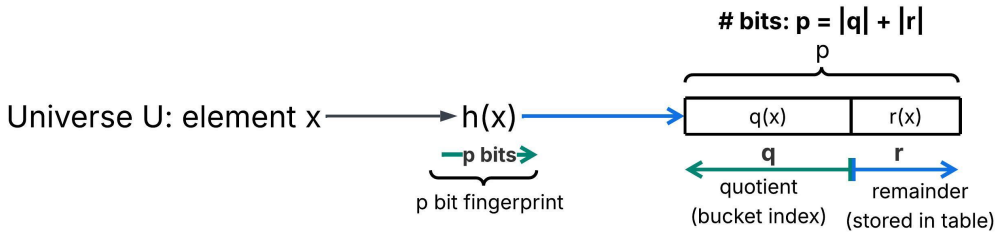
where the $O(n)$ term accounts for additional linear overhead.

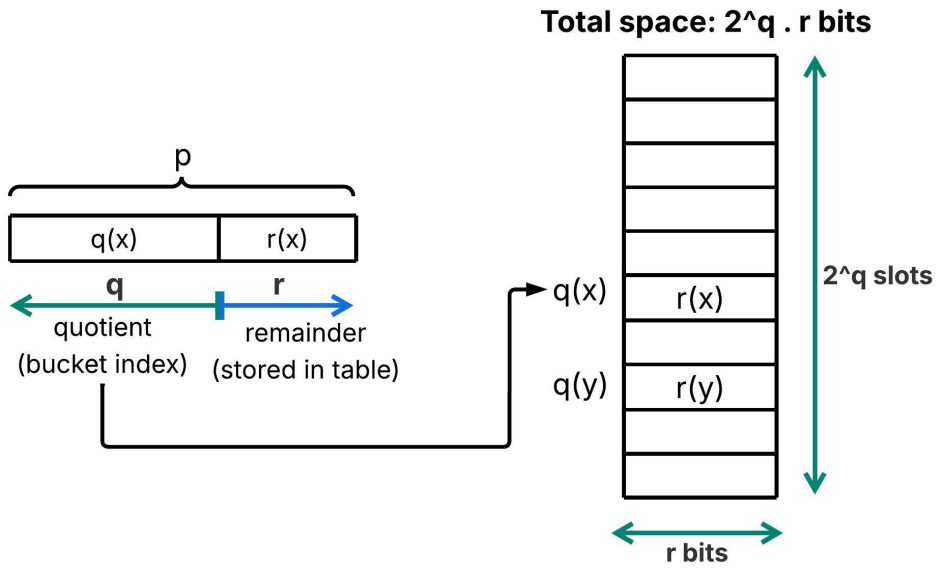
# 5 Quotient Filter

- A **quotient filter** is a probabilistic data structure for approximate membership queries (similar to Bloom filters).
- Instead of storing full elements, it stores a short **fingerprint** derived from a hash function.
- The hash value is split into two parts:

$$h(x) = q(x) \mid\mid r(x)$$

  - **quotient** $q(x)$: determines the bucket index in the table
  - **remainder** $r(x)$: stored as the fingerprint




Two elements may map to the same bucket even if their hashes are different:

$$h(x) \neq h(y)$$
$$q(x) = q(y)$$
$$r(x) \neq r(y)$$

This means both elements map to the same bucket index determined by the same quotient $q(x)$. To resolve this collision, the quotient filter stores elements using **linear probing**.

#### 5.0.1 How to decide q & r

Suppose we store $n$ items. The quotient determines the number of buckets in the table.

$$n \text{ items} \rightarrow 2^q \approx n$$

so we choose

$$q = \lg n$$

so the remainder size is

$$p = q + r$$

$$\rightarrow r = p - q$$

The false positive probability depends on the fingerprint size:

$$\epsilon = \frac{n}{2^p}$$

Since $n \approx 2^q$,

$$\epsilon = \frac{n}{2^p} = \frac{2^q}{2^p} = \frac{2^q}{2^{q+r}} = \frac{1}{2^r}$$

So, the false positive rate decreases exponentially with the remainder size $r$.

## References


[1] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7): 422-426, 1970.

[2] Bin Fan, Dave G. Andersen, Michael Kaminsky, and Michael Mitzenmacher. Cuckoo Filter: Practically Better Than Bloom. *Proceedings of the 10th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 75–88, 2014.