CS 7280: DSA for Scalable Computing Spring 2025

Lecture : Range & Adaptive Filters — February 19, 2025

Prof. Prashant Pandey *Scribe: Griffin Ravo*

1 Overview

Recap

Given a filter F on a dataset S, we want to determine whether an element x belongs to S.

- If $x \in S$, then the filter always returns "Yes" with probability:
$$P(\text{Yes} \mid x \in S) = 1$$
- If $x \notin S$, the filter may return "Yes" with some false positive probability ϵ:
$$P(\text{Yes} \mid x \notin S) = \epsilon$$
- If the filter returns "No", then x is definitely not in S, with probability:
$$P(\text{No} \mid x \notin S) = 1 - \epsilon$$

To guarantee a maximum error rate of ϵ when queries are selected uniformly at random, the space required by the filter is:

$$O(n \log(1/\epsilon)) \text{ bits}$$

where n is the number of elements in the dataset S.

Recall that filters may produce false positives, but never false negatives, ensuring that all existing elements are always correctly identified. That false positive rate is denoted as ϵ (as seen above).

Today

In this lecture we will cover Range Filters and Adaptive Filters.

2 Range Filters

2.1 Definition

Given a range filter F on a dataset S, we want to determine whether any element exists within a given query range $[q_l, q_r]$.

- If there exists at least one element in S within the range $[q_l, q_r]$, then the filter always returns "Yes" with probability:
$$P(\text{Yes} \mid [q_l, q_r] \cap S \neq \emptyset) = 1$$
- If there are no elements in S within the range $[q_l, q_r]$, the filter may still return "Yes" with some false positive probability ϵ:
$$P(\text{Yes} \mid [q_l, q_r] \cap S = \emptyset) = \epsilon$$
- If the filter returns "No", then there are definitely no elements in S within $[q_l, q_r]$, with probability:
$$P(\text{No} \mid [q_l, q_r] \cap S = \emptyset) = 1 - \epsilon$$

To guarantee a maximum false positive rate of ϵ for range queries, the space required by the filter is:

$$O(n \log(R/\epsilon)) \text{ bits}$$

where n is the number of elements in the dataset S and R is the size of the queried range.

> Any data structure for answering approximate range emptiness queries on intervals of length up to L with false positive probability ϵ, must use space
>
> $$\Omega(n \log(L/\epsilon)) - O(n) \text{ bits.}$$
>
> — Goswami et al., 2014 [1]

The worst-case query performance is:

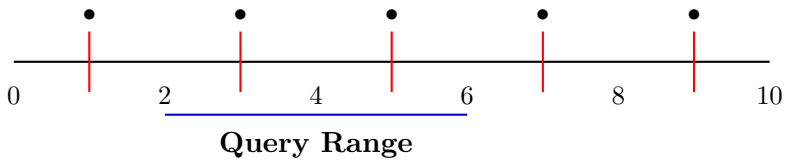
$$O(R)$$

Let's see how this can be improved.

2.2 Prefix Filters

A naive implementation of a range filter.

Below is a visualization of how a prefix filter operates.



The vertical red lines represent precomputed presence indicators at every interval of size s. The blue segment represents a query range. If any marker falls within the range, the filter returns "Yes."

For a prefix filter with step size s, the worst-case query performance is:

$$O(R/s)$$

where R is the query range size.

The probability of a false negative, given that n elements are inserted and the universe size is U, is:

$$1 - \left(1 - \frac{U-s}{U}\right)^n$$

where s is the step size.

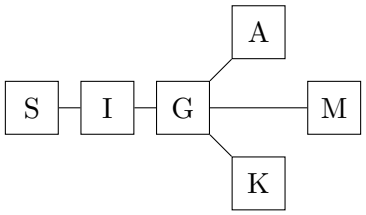
2.3 SURF (Succinct Range Filter)

Proposed by Zhang et al. in 2018 [2] as a succinct prefix trie-based approach for approximate range membership queries.

SURF builds a succinct trie over the dataset's prefixes. For example, given the words:

SIGARCH, SIGKOD, SIGMOD

The prefix trie looks like:



The trie captures common prefixes, avoiding redundant storage of full words. False positives occur due to the querying of prefix presence.

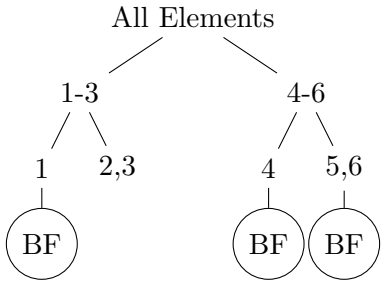
SURF has no theoretical upper bounds. It is only evaluated empirically. It does not support inserts.

2.4 Rosetta

Proposed by Lou et al. in 2020 [3], Rosetta is a hierarchical range filter that extends traditional range filters by constructing a tree structure similar to a binary trie over the dataset. Each level of the tree contains a range filter to refine queries, with the bottom level using a Bloom Filter to store elements.

Given a set $S = \{1, 4, 5\}$:

- The root covers the entire universe.
- Each subsequent level partitions the space further.
- The bottom level is the Bloom Filter used to keep track of elements.



To ensure a false positive rate of ϵ:

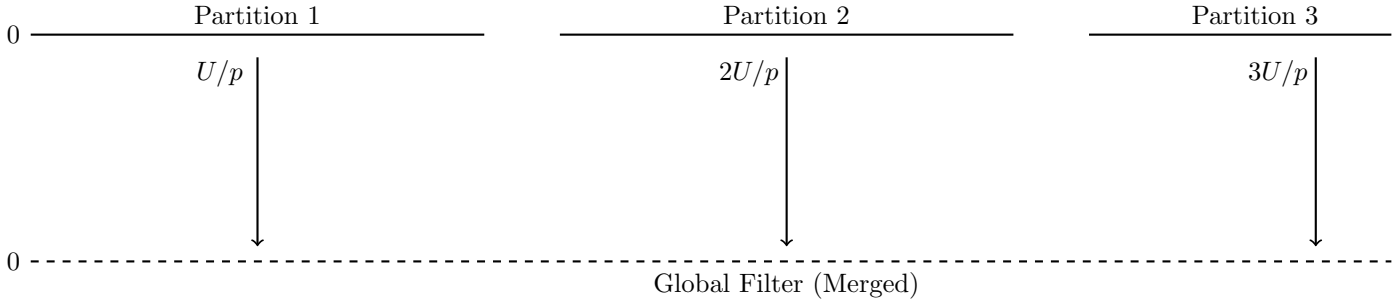
$$\epsilon - > 1.44 \log_2(R/\epsilon) \text{ bits per element}$$

where:
R is the query range size,
ϵ is the desired false positive probability.

2.5 Memento

Proposed by Eslami et al. in 2025 [4], Memento partitions the universe into p partitions and then combines them into a single filter for efficient range queries.

Below is a visualization of how partitions are combined:



The false positive rate of Memento is bounded by:

$$\text{FPR} \leq bEr$$

where:
- b is a constant related to filter compression.
- E is the expected load factor per partition.
- r is the query range factor.

Since the universe U is divided into p partitions, each partition has an expected element load of:

$$\frac{U}{p \cdot nb}$$

where:
- n is the number of elements.
- b is the bits per element.
- p is the number of partitions.

Space complexity:

$$n \left(\log_2(R/\epsilon) + O(1) \right) \text{ bits.}$$

3 Adaptive Filters

Proposed by Bender et al. at FOCS [5], Adaptive Filters aim to "learn from their mistakes." Unlike traditional filters, which assume a uniformly random query workload, an adaptive filter dynamically improves its accuracy based on past queries.

A strongly adaptive filter achieves an expected space complexity of:

$$O(n\epsilon)$$

irrespective of the query workload.

The theoretical lower bound for an adaptive filter is:

$$O(n) \text{ space (words)}$$

Why does this still qualify as a filter?

Adaptive Filters don't need to store all elements in memory. Instead, they use a combination of in-memory storage and external disk storage:

- $O(n \log(1/\epsilon))$ space in memory
- $O(n)$ space on disk

References

[1] Goswami et al. *Approximate Range Emptiness in Constant Time and Optimal Space.* In Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (*SODA*), 2014.

[2] Zhang et al. *SuRF: Practical Range Query Filtering with Fast Succinct Tries.* In Proc. of the ACM SIGMOD International Conference on Management of Data (*SIGMOD*), 2018.

[3] Lou et al. *Rosetta: A Robust Space-Time Optimized Range Filer for Key-Value Stores.* In Proc. of the ACM SIGMOD International Conference on Management of Data (*SIGMOD*), 2020.

[4] N. Eslami, N. Dayan. *Memento Filter: A Fast, Dynamic, and Robust Range Filter* In Proc. of the ACM SIGMOD International Conference on Management of Data (*SIGMOD*), 2025.

[5] M. Bender et al. *Adaptive Quotient Filters* In Proc. of the IEEE Symposium on Foundations of Computer Science (*FOCS*), 2020.