Bloom-Filter: Implementierung und Auswertung

Daniel Barber Tamira Leber

4. Dezember 2025

1. Idee und Funktionsweise

Ein Bloom-Filter ist eine probabilistische Datenstruktur, die mit sehr wenig Speicher auskommt. Statt Elemente direkt zu speichern, verwendet er ein Bitfeld der Länge m und k Hashfunktionen. Beim Einfügen eines Strings werden k Bit-Positionen gesetzt. Bei einer Abfrage werden dieselben Positionen geprüft:

- Wenn mindestens ein Bit 0 ist, ist das Element *sicher nicht* enthalten.
- Wenn alle Bits 1 sind, ist das Element *möglicherweise* enthalten (false positive möglich).

Vorteile:

- Sehr geringer Speicherbedarf
- Schnelle Einfüge- und Abfrageoperationen
- Einfache Implementierung

Nachteile:

- False positives sind möglich.
- Löschen ist nicht ohne Weiteres möglich (nur mit Counting-Bloom-Filtern)

2. Beispiel aus der Praxis

Bloom-Filter werden in vielen Anwendungen eingesetzt, in denen schnelle Vorabprüfungen erforderlich sind. Ein bekanntes Beispiel ist **Google Safe Browsing**. Dort wird ein Bloom-Filter verwendet, um effizient zu prüfen, ob eine URL möglicherweise auf einer Liste schädlicher Webseiten steht. So lassen sich unnötige Netzwerkabfragen reduzieren.

3. Test der Fehlerrate

Für die Aufgabe haben wir die Wortliste `words.txt` verwendet. Die erwartete Elementanzahl n wurde automatisch aus der Datei bestimmt. Das Programm erhält einen gewünschten Fehlerparameter p als Eingabe und berechnet daraus die nötige Filtergrösse m und die optimale Anzahl Hashfunktionen k. Die Berechnung erfolgt mit den Standardformeln:

$$m = -\frac{n \ln p}{(\ln 2)^2} \qquad k = \frac{m}{n} \ln 2$$

Als Hashverfahren kam `murmur3_128` zum Einsatz, wobei für jede der k Hashfunktionen ein anderer Seed verwendet wurde.

Zur Bestimmung der tatsächlichen Fehlerrate wurden viele zufällige Nicht-Wörter erzeugt und im Bloom-Filter abgefragt. Die Anzahl der false positives geteilt durch die Gesamtanzahl der Tests ergibt die experimentelle Fehlerrate.

```
danielbarber@mac dist-hs25-bloom-filter % ./gradlew run


> Task :run
Loaded 58109 words from data/words.txt

Creating Bloom filter with:
  n = 58109 elements
  p = 0.01 target false positive probability
  m = 556979 bits
  k = 7 hash functions
All words inserted into the Bloom filter.

Manual check:
  word from list: aardvark -> true
  made-up word: thiswordisnotinthedictionary123 -> false

Running false-positive experiment with 100000 random strings

===== RESULT =====
Target false positive probability p: 0.01
Experimental false positive rate: 0.01009
Parameters: n = 58109, m = 556979, k = 7
==================

BUILD SUCCESSFUL in 481ms
2 actionable tasks: 2 executed
```