

BloomFilter

A Idee des Bloom-Filters

Ein BloomFilter liefert schnell eine Antwort, ob ein Wert bereits vorgekommen ist oder nicht.

Der Filter liefert also zwei verschiedene Antworten:

- Mit hoher Wahrscheinlichkeit enthalten
- Definitiv nicht enthalten

Gerade das ein Wert nicht bekannt ist, kann genutzt werden, um schnelle Entscheidungen zu treffen.

Vor- und Nachteile	
Vorteil	Nachteil
<ul style="list-style-type: none"> • sehr performant • Treffergenauigkeit über die Grösse des Index konfigurierbar, dabei besteht kein Risiko, dass ein Treffer nicht gefunden wird • kompaktes Speichern der Daten (oft wird nur circa 1/8 der Ausgangs-Datenmenge benötigt) • Implementierungen in allen relevanten Programmiersprachen verfügbar • leicht parallelisierbar (über Aufteilung des Suchindex auf mehrere Server/CPU-Kerne) 	<ul style="list-style-type: none"> • keinerlei Ähnlichkeitssuche, keine Fehler-toleranz • Suchergebnisse werden nicht sortiert • Risiko, sogenannte False Positives zu erhalten, das heisst, es können Datensätze zurückgegeben werden, die den Suchbegriff nicht enthalten

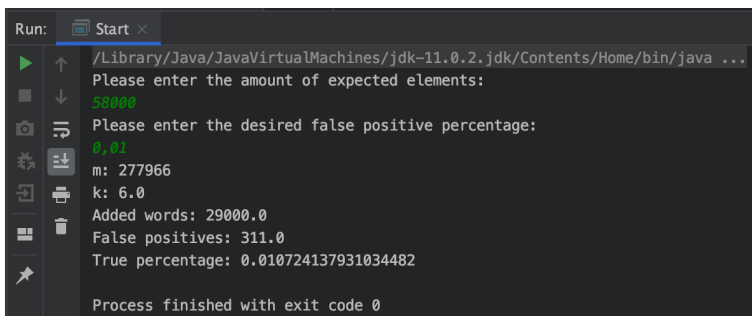
B Praxisbeispiel



Ethereum benutzt einen Bloom Filter, um Logs schnell und effizient in der Ethereum Blockchain finden zu können.

Im Ethereum-System müssen Events, einschliesslich historischer Events, leicht und ohne unnötigen Aufwand gefiltert und gesucht werden können. Gleichzeitig ist der Speicherplatz teuer, dass nicht viele Daten gespeichert werden sollen, wie z. B. die Liste der Transaktionen und die von ihnen erstellten Protokolle. Wenn ein Block generiert oder verifiziert wird, wird die Adresse eines Protokollierungsvertrags einem Bloom Filter hinzugefügt, der im Blockheader enthalten ist. Die eigentlichen Protokolle sind aus Platzgründen nicht in den Blockdaten enthalten. Wenn nun alle Protokolleinträge durchsucht werden, kann der Bloom Filter überprüfen, ob relevante Protokolle vorhanden sind.

C Testen der Fehlerwahrscheinlichkeit



```
Run: Start x
/Library/Java/JavaVirtualMachines/jdk-11.0.2.jdk/Contents/Home/bin/java ...
Please enter the amount of expected elements:
50000
Please enter the desired false positive percentage:
0.01
m: 277966
k: 6.0
Added words: 29000.0
False positives: 311.0
True percentage: 0.010724137931034482

Process finished with exit code 0
```