

## Bonusaufgabe

### Bloom-Filter

---

Ziel der Aufgabe ist es, einen Bloom-Filter zu programmieren und die Funktionsweise darzustellen.

Folgende Schritte werden erwartet:

1. Sie verschaffen sich einen Überblick über die Funktionsweise [https://en.wikipedia.org/wiki/Bloom\\_filter](https://en.wikipedia.org/wiki/Bloom_filter).
2. Sie erstellen ein Java-Programm, was folgendes leistet:
  - (a) Bei einer gegebenen Anzahl  $n$  an zu erwartenden Elementen, die in der Datenstruktur gespeichert werden und einer Fehlerwahrscheinlichkeit  $p$  wird eine geeignete Filtergrösse  $m$  und die optimale Anzahl  $k$  an Hashfunktionen berechnet.
  - (b) Die Datenstruktur wird implementiert, mit den Methoden zum Einfügen von Strings und dem Test, ob ein String enthalten ist. Als Hash-Funktionen soll `murmur3_128` (<http://docs.guava-libraries.googlecode.com/git-history/v11.0/javadoc/com/google/common/hash/Hashing.html>) verwendet werden mit jeweils einem anderen Seed. Insbesondere müssen Sie sich überlegen, wie ein Hashwert zu einem Indexwert des Arrays wird.
  - (c) Lassen Sie das Programm die Wörter aus `words.txt` einlesen und in eine Datenstruktur der entsprechenden Grösse bei gegebenen Fehlerwahrscheinlichkeit  $p$  einfügen. Überprüfen Sie die Fehlerwahrscheinlichkeit, in dem Sie für eine grosse Anzahl an nicht enthaltenden Strings testen, ob sie enthalten sind.
3. Sie erstellen eine Zusammenfassung von ca. einer DinA4-Seite, welche folgendes enthält:
  - (a) Idee des Bloom-Filters, mit Vor- und Nachteilen
  - (b) ein Beispiel, wo der Bloom-Filter verwendet wird mit kurzer Beschreibung dieses Programms
  - (c) Eine Beschreibung, wie Sie einen Hashwert in einen Indexwert transformieren.
  - (d) Eine Beschreibung, wie Sie die Fehlerwahrscheinlichkeit ihrer Datenstruktur getestet haben und welche Resultate dabei erzielt worden sind.

Allgemeine Hinweise:

1. Sie können in Gruppen bis zu drei Personen arbeiten.
2. Bei vollständiger Lösung wird auf die Note des kommenden Tests 0.3 drauf addiert. (Aus systemtechnischen Gründen liegt die Erfahrungsnote zwischen 1.0 und 6.0.)
3. Es ist nicht nötig, das Programm hinsichtlich Effizienz zu optimieren.
4. Das Programm sollte verständlich kommentiert sein.

5. Eigentlich gehe ich davon aus, dass Sie aus Fairnessgründen nicht versuchen, zu betrügen. Dennoch werde ich dies (auch mit Hilfe von Tools) kontrollieren. Falls dabei ein Täuschungsversuch festgestellt wird (also: (verschleierte) Kopien von Teilen existierender Programme (Internet oder Kollegen)), wird die Note des nächsten Tests auf 1.0 gesetzt.

**Abgabe:** 02/03.12.2015