

# bloom-filter egg

---

Provides a simple Bloom Filter  
Extension for Chicken Scheme  
Version 1.1

**Kon Lovett**

---

# Table of Contents

<b>1</b>	<b>About this egg</b> .....	<b>1</b>
1.1	Version history .....	1
1.2	Requirements .....	1
1.3	Usage .....	1
<b>2</b>	<b>Documentation</b> .....	<b>2</b>
2.1	Bloom Filter Object .....	2
2.2	Auxillary Procedures .....	3
2.3	Hash Primitives Configuration File .....	3
<b>3</b>	<b>References</b> .....	<b>4</b>
<b>4</b>	<b>License</b> .....	<b>5</b>
	<b>Index</b> .....	<b>6</b>

# 1 About this egg

## 1.1 Version history

- 1.1 Support for "optimal K"
- 1.0 Exports
- 0.2 Add hash primitives configuration file
- 0.1 Initial release

## 1.2 Requirements

This egg requires the following extensions:

`iset, hashes, md5, sha1, sha2, tiger-hash, ripemd, message-digest, lookup-table, mathh, misc-extn`

## 1.3 Usage

Load this egg like so:

```
(require-extension bloom-filter)
```

## 2 Documentation

### 2.1 Bloom Filter Object

`make-bloom-filter` [procedure]  
 (`make-bloom-filter` M MESSAGE-DIGEST-PRIMITIVE-LIST [K])

Returns a bloom-filter object with M bits of discrimination and a set of hash functions built from the supplied MESSAGE-DIGEST-PRIMITIVE-LIST. The elements of the list of primitives may be an actual primitive object or a symbol naming the desired message-digest.

The number of hash functions, k, is not necessarily the same as the number of message-digests. A hash function is defined as returning an unsigned 32 bit integer. Most message-digests return more 32 bits of hash. The actual length of the hash is divided into 32 bit blocks to get the individual hash functions.

The argument K will restrict the actual number of hash functions to the "first" k, no matter how many more the supplied message-digests create. First in the order of MESSAGE-DIGEST-PRIMITIVE-LIST.

Selecting the optimal set of message-digests is beyond the scope of `make-bloom-filter`.

`bloom-filter-n` [procedure]  
 (`bloom-filter-n` BLOOM-FILTER)

The current population - the number of objects added to the filter.

`bloom-filter-m` [procedure]  
 (`bloom-filter-m` BLOOM-FILTER)

The number of bits of discrimination.

`bloom-filter-k` [procedure]  
 (`bloom-filter-k` BLOOM-FILTER)

The number of hash functions. (See above.)

`bloom-filter-p-false-positive` [procedure]  
 (`bloom-filter-p-false-positive` BLOOM-FILTER [N])

The probability of false positives for the given population size. The current population is assumed.

`bloom-filter-set!` [procedure]  
 (`bloom-filter-set!` BLOOM-FILTER OBJECT)

Add the specified OBJECT to the indicated BLOOM-FILTER.

`bloom-filter-exists?` [procedure]  
 (`bloom-filter-exists?` BLOOM-FILTER OBJECT)

Is the specified OBJECT in the indicated BLOOM-FILTER.

## 2.2 Auxillary Procedures

`bloom-filter:optimum-k` [procedure]

(`bloom-filter:optimum-k` N M)

Optimal count of hash functions for the given population size N and M bits of discrimination.

`bloom-filter:optimum-m` [procedure]

(`bloom-filter:optimum-m` K N)

Optimal count of bits of discrimination for the given population size N and K number of hash functions.

`bloom-filter:p-false-positive` [procedure]

(`bloom-filter:p-false-positive` K N M)

What is the probability of false positives for the population size N assuming K hash functions and M bits of discrimination.

`bloom-filter:desired-m` [procedure]

(`bloom-filter:desired-m` P N [K])

Calculates a near-optimal number of bits of discrimination to meet the desired probability of false positives P, with the given population size N and number of hash functions K. When the k parameter is missing the `bloom-filter:optimum-k` procedure is used to calculate a value.

A multi-valued return of the calculated M, K, and P values. The calculated probability may be lower than the desired.

`bloom-filter:actual-k` [procedure]

(`bloom-filter:actual-k` MESSAGE-DIGEST-PRIMITIVE-LIST)

Calculates the actual number of hash functions for the MESSAGE-DIGEST-PRIMITIVE-LIST. The elements of the list of primitives may be an actual primitive object or a symbol naming the desired message-digest.

`bloom-filter:p-random-one-bit` [procedure]

(`bloom-filter:p-random-one-bit` K N M)

Guess.

## 2.3 Hash Primitives Configuration File

A file, "hash-primitives-info", is located in the Chicken Repository. The file contains the information needed by bloom-filter to load hash primitives at runtime. The file is self-documenting.

## 3 References

- [Nice exposition of Bloom Filter False Positive Probability.](#)
- [A web interface for a better version of bloom-filter:desired-m.](#)

## 4 License

Copyright (c) 2006, Kon Lovett. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the Software), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED ASIS, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## Index

### B

bloom-filter-exists? ..... 2  
bloom-filter-k ..... 2  
bloom-filter-m ..... 2  
bloom-filter-n ..... 2  
bloom-filter-p-false-positive ..... 2  
bloom-filter-set! ..... 2  
bloom-filter:actual-k ..... 3

bloom-filter:desired-m ..... 3  
bloom-filter:optimum-k ..... 3  
bloom-filter:optimum-m ..... 3  
bloom-filter:p-false-positive ..... 3  
bloom-filter:p-random-one-bit ..... 3

### M

make-bloom-filter ..... 2