Bloom Filter Calculator

Below, m denotes the number of bits in the Bloom filter, n denotes the number of elements inserted into the Bloom filter, k represents the number of hash functions used, and p denotes the false positive rate.

Enter values for any combination of 2 or 3 of the parameters below and you will get back the "optimal" values for the remaining parameters and an informative message. The values for m, n, and k have to be positive integers. The value of p has to be greater than 0 and less than 1. Certain abbreviations are allowed for (just) the value of p, e.g., instead of ".0000000001", you can write "1E-10".

m:

n:

k:

p:

When ready, click on the "Submit" button. Submit or Erase all

What is a Bloom filter?

Bloom filters are used to probabilistically and compactly represent subsets of some universe U. A Bloom filter is implemented as an array of m bits, uses k hash functions mapping elements in U to $[0..m)$, and supports two basic operations: *add* and *query*. Initially, all bits in the Bloom filter are set to 0. To add u (an element of U) to a Bloom filter, the hash functions are used to generate k indices into the array and the corresponding bits are set to 1. A query is positive iff all k referenced bits are 1. A negative query clearly indicates that the element is not in the Bloom filter, but a positive query may be due to a *false positive*, the case in which the queried element was not added to the Bloom filter, but all k queried bits are 1 (due to other additions).

The probability of false positives, p, is an important metric because minimizing it is the key to making effective use of Bloom filters. The analysis proceeds as follows. If q is the probability that a random bit of the Bloom filter is 1, then the probability of a false positive, p is q^k, the probability that all k hash functions map to a 1. If we let n be the number of elements that have been added to the Bloom filter, then $q = 1 - (1-(1/m))^{nk}$, as nk bits were randomly selected, with probability $1/m$, in the process of adding n elements. Broder and Mitzenmacher show that the probability of false positives is minimized when k is approximately $m/n \log_e 2$.

Bloom Filter References

There are many. We recommend the following.

1. This paper provides a good overview. Network Applications of Bloom Filters: A Survey. A. Broder and M. Mitzenmacher. Proc. of the 40th Annual Allerton Conference on Communication, Control, and Computing, pages 636-646, 2002.
2. This paper presents recent work by myself and Peter Dillinger that shows how one can obtain Bloom filters that are simultaneously fast, accurate, and memory-efficient. Bloom Filters in Probabilistic Verification. *FMCAD 2004, Formal Methods in Computer-Aided Design*, 2004.
3. This is a companion paper that describes an implementation of the ideas based on the model checker SPIN. Fast and Accurate Bitstate Verification for SPIN. Peter C. Dillinger and Panagiotis Manolios. In SPIN 2004, pages 57-75, 2004.

Last modified: Sun Aug 29 16:13:58 EDT 2004
manolios@cc.gatech.edu

Back to Pete's home page
College of Computing