Bloom Filter Requirements Specification

1. Scope

1.1 Overview

A Bloom filter is a probabilistic data structure that can be used to test for set membership in constant space and constant time. It may return false positives, but never false negatives. Descriptions of the algorithm can be found at http://en.wikipedia.org/wiki/Bloom_filter, http://www.cs.wisc.edu/~cao/papers/summary-cache/node8.html, and http://portal.acm.org/citation.cfm?id=362692&dl=ACM&coll=portal.

1.2 Logic Requirements

1.2.1 Initialization

It must be possible to create a Bloom filter either by specifying a capacity (maximum number of items to be inserted) and a maximum error rate, or by specifying a size for the bit vector and a number of hashing functions. When a Bloom filter is created by specifying a capacity and an error rate, the size of the bit vector and the number of hashing functions should be computed to minimize the size of the bit vector while preserving the maximum error rate at the given capacity. Note that more items than the capacity may be inserted, but beyond the capacity, the guaranteed error rate no longer holds.

1.2.2 Hash Functions

The Bloom filter must use a family of hashing functions, such that when the algorithm requires *k* distinct hashing functions for some positive integer *k*, those functions can be generated. Using one algorithm and salting it or seeding it with *k* different integers to generate *k* different hash functions is an appropriate technique.

1.2.3 Insertion and Membership Testing

It must be possible to insert an item into a Bloom filter, and check if an item is contained in a Bloom filter.

1.2.4 Intersection

It must be possible to compute the intersection of two identically initialized Bloom filters (same bit vector size and number of hash functions). Note that this is simply the bitwise-and of the two bit vectors.

1.2.5 Union

It must be possible to compute the union of two identically initialized Bloom filters (same bit vector size and number of hash functions). Note that this is simply the bitwise-or of the two bit vectors.

1.2.6 Serialization

It must be possible to convert a Bloom filter to and from a string representation that would be suitable for use in a plain text or HTML document.

1.3 Required Algorithms

Bloom filters

1.4 Example of the Software Usage

Bloom filters could be used to make a large, complex web site searchable. For each page, a Bloom filter is created and every word on that page is inserted into it. To search the web site for a keyword, that word is checked for membership in the Bloom filter for each page, and the matching pages are returned. Since the Bloom filter is probabilistic, some false positives will be returned, but no matching pages will ever be missed.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

None

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure

com.topcoder.bloom

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None

3.2.2 TopCoder Software Component Dependencies:

None

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of Poseidon.