Heather Koyuk
Dr. Millman
CSCI 540
12/02/2021

# Bloom Filters

The problem of figuring out whether something was or wasn't in a set was present even in the late 60's and early 70's. A few such example problems are checking new usernames for registering for a database, seeing whether a website is already in a user's cache, and expensive disk lookups that have a chance of returning null data. Before 1970, most existing data structures (including linear search, error free hashes, and binary search trees) were either too large, too slow, or both. In the 1960's and through 1970, the computer scientist Burton Howard Bloom wrote three papers, culminating in his paper "Space/Time Trade-offs in Hash Coding with Allowable Errors", published in 1970.

So, what is a Bloom Filter? A Bloom filter is a "space-efficient probabilistic data structure." It is both space-efficient and probabilistic because it relies on hashes. A hash function, if you remember, is "any function that can be used to map data of arbitrary size to fixed-size values." For example, at it most trivial, a function that returns the number 17 for every input is technically a hash (just not a useful one). A slightly less trivial but still simple hash is based on the modular function – in fact, modular hashes are common, and we walked through a very simple example of a modular hash where every character of a string is hashed into a Bloom filter as a case in point (note: in actuality, you would hash entire strings rather than single characters one at a time).

Returning to Howard Bloom's paper and the idea of "Allowable Errors" / false positives, there are three variables that affect the probability of collisions/false positives in a Bloom filter: the size of the data being stored (n), the size of filter (m), and the number of hash functions (k) used. Numerous studies have been done on this false positive rate and there is a general approximate formula commonly used. (Additionally, Goel and Gupta have established a rigorous upper bound.) These formulas can also be used to estimate the approximate or upper bounds of unique counts within a Bloom filter as well as the counts of unions and intersections between sets.

The best hashes to use for a Bloom filter are fast (non-cryptographic), uniformly distributed, and independent, for instance murmur and HashMix.

Although Bloom's paper didn't make a huge splash back in 1970, there has been a resurgence of interest in Bloom filters in the last 10-20 years. All in all there are over 60 different types of Bloom filter variants, including Counting Bloom Filters, Blocked Bloom Filters, Bloomier Filters, Replicating Bloom Filters, Spatial Bloom Filters, and Layered Bloom Filters. Many companies both small and large (including PostgreSQL, Medium, and Google Chrome to name a few) employ various types of Bloom filters to accomplish various tasks such as safe browsing, detecting weak passwords, and enhancing cyber security.

In conclusion, if an estimatable amount of false positives are acceptable and we want to know quickly and in a space-efficient manner whether something is definitely not in a set (or likely in a set), a Bloom filter might be your answer.

Bibliography


1. "Bloom Filter." Wikipedia, Wikimedia Foundation, 26 October 2021, https://en.wikipedia.org/wiki/Bloom_filter.
2. "Burton H. Bloom." dblp, Schloss Dagstuhl,22 April 2021. https://dblp.org/pid/38/994.html.
3. "Hash Function." Wikipedia, Wikimedia Foundation, 13 November 2021, https://en.wikipedia.org/wiki/Hash_function.
4. Singh, Vivek Kumar. "Bloom Filter : A Probabilistic Data Structure." Medium.com, 27 May 2019, https://medium.com/system-design-blog/bloom-filter-a-probabilistic-data-structure-12e4e5cf0638. Accessed 22 November 2021.
5. Martí, Vicent. "Some performance tweaks #19." https://github.com/bitly/dablooms/pull/19. Accessed 22 November 2021. Cornell University. "CS 312 Lecture 21, Hash functions." https://www.cs.cornell.edu/courses/cs312/2008sp/lectures/lec21.html. Accessed 23 November 2021.
6. Sedgewick, Robert and Wayne, Kevin. "Hash Tables." https://algs4.cs.princeton.edu/34hash/. Accessed 24 November 2021.
7. Hassan, Ahmed Shamim. "Probabilistic Data structures: Bloom filter." https://hackernoon.com/probabilistic-data-structures-bloom-filter-5374112a7832. Accessed 24 November 2021