

Algorithm Design 21/22

Hands On 5 - Bloom Filters

Federico Ramacciotti

1 Problem

1. Consider the Bloom filters where a single random universal hash random function $h : U \rightarrow [m]$ is employed for a set $S \subseteq U$ of keys, where U is the universe of keys.

Consider its binary array B of m bits. Suppose that $m \geq c|S|$, for some constant $c > 1$, and that both c and $|S|$ are unknown to us.

Estimate the expected number of 1s in B under a uniform choice at random of $h \in \mathcal{H}$. Is this related to $|S|$? Can we use it to estimate $|S|$?

2. Consider B and its rank function: show how to use extra $O(m)$ bits to store a space-efficient data structure that returns, for any given i , the following answer in constant time:

$$\text{rank}(i) = \#1s \in B[1..i]$$

Hint: easy to solve in extra $O(m \log m)$ bits. To get $O(m)$ bits, use prefix sums on B , and sample them. Use a lookup table for pieces of B between any two consecutive samples.

2 Solution

2.1 Question 1

Define the indicator variable

$$X_i = \begin{cases} 1 & \text{if } B[i] = 1 \\ 0 & \text{otherwise} \end{cases}$$

The probability for it to be 1 is $\Pr[X_i = 1] = \Pr[B[i] = 1] = 1 - p' = 1 - (1 - \frac{1}{m})^{|S|}$.

Define also the sum $Y = \sum_{i=0}^{m-1} X_i$ and find its expected value:

$$\begin{aligned} E[Y] &= E \left[\sum_{i=0}^{m-1} X_i \right] = \sum_{i=0}^{m-1} \Pr[X_i = 1] \\ &= \sum_{i=0}^{m-1} 1 - \left(1 - \frac{1}{m}\right)^n = m - \sum_{i=0}^{m-1} \left(1 - \frac{1}{m}\right)^n \\ &= m - m \left(1 - \frac{1}{m}\right)^n \cong m - me^{-\frac{n}{m}} \\ &= m \left(1 - e^{-\frac{n}{m}}\right) \end{aligned}$$

So we get that the expected number of 1s in B is $m(1 - e^{-\frac{n}{m}})$ and, as we can clearly see, it's strongly related to $|S| = n$. In order to estimate $|S|$, given the number of ones $n_1 = E[Y] = m(1 - e^{-\frac{n}{m}})$, we get $1 - \frac{n_1}{m} = e^{-\frac{n}{m}}$ and, changing the sign and applying the logarithm, we obtain $n = -m \log(1 - \frac{n_1}{m})$.

2.2 Question 2

2.2.1 Baseline solution

The baseline solution uses prefix sums in $O(m)$ time on the array B to solve any query in linear time. The space used is $O(\log m)$ bits for each integer of the array and, with an array of length m , the total space is $O(m \log m)$.

2.2.2 Better solution: sampling

A better solution uses sampling on the prefix sums array created in the baseline solution. Since the goal is to use linear space $O(m)$ with samples of size $O(\log m)$, we need just

$$\# \text{ samples} * \log m = m \implies \# \text{ samples} = \frac{m}{\log m}$$

samples, for a total size of $\log m * \frac{m}{\log m} = m$ bits. This solution takes $O(\log m)$ time per query, since we scan every interval between two samples.

In order to have a cost per query of $O(1)$ time, we pre-compute a lookup table T that stores the prefix sums for any possible integer of $\frac{\log m}{2}$ bits, written in binary. Thus, to get $\text{rank}(i)$, we sum the first smaller sample with the right entry in the lookup table T .

Since a string of $\frac{\log m}{2}$ bits can represent up to $2^{\frac{\log m}{2}} = \sqrt{m}$ distinct integers, the lookup table T takes $\log(\frac{\log m}{2}) * \sqrt{m}$ bits. Summing up with the previous space values, we get that this solution takes $m + 2m + \log(\frac{\log m}{2})\sqrt{m} = O(m)$ bits, with $O(1)$ time per query.