Cairo University
Faculty of Computers and Artificial Intelligence
Computer Science Department

Advanced Data Structures **Bloom Filters** Dr. Amin Allam

[For more details, refer to Wikipedia]

1 Bloom Filters

A *Bloom filter* is a data structure used to test whether an *element* is a member of a *set*. It is *probabilistic* since it allows *false positives*; It may decide that an *element* belongs to the *set* while it does not. However, it does not allow *false negatives*; it never decides that an *element* does not belong to the *set* while it does. Thus it returns either *"possibly* $\in$ *set"* or *"definitely* $\notin$ *set"*.

The motivation of this structure is to provide an alternative to *hash tables* which consumes much less space than *hash tables*, but with the drawback of introducing *false positives*. It is usually accompanied with a *hash table* or another data structure which is effectively used to retrieve the *element* only if it passed the fast *Bloom filter* test. It improves query times when most queried *elements* do not belong to the *set*, and when the set *elements* are stored in secondary storage.

An empty *Bloom filter* is an array of m bits, all set to zero. A *hash function* is a function whose parameter is an element to be inserted, and it returns a value in the range $\{0 \dots m-1\}$. To insert an *element* in the *Bloom filter*, k different *hash functions* are applied to the inserted *element* to return k positions in the range $\{0 \dots m-1\}$. Then, all bits at all these positions are set to 1. It is not possible to remove an *element* from a *Bloom filter*.

To estimate the effectiveness of *Bloom filters*, we need to calculate the probability of its *false positives* which depends on the values of k and m. Assume that a *hash function* returns a value in the range $\{0 \dots m-1\}$ with equal probability of $\frac{1}{m}$ for each possible value.
Let $p = 1 - q$ the probability that a certain bit is set to 1.
Let $q = 1 - p$ the probability that a certain bit is not set to 1.
After one *hash function* call: $q = 1 - \frac{1}{m}$.
After inserting one *element* (that is, after k *hash function* calls): $q = \left(1 - \frac{1}{m}\right)^k$.
After inserting n *elements*: $q = \left(1 - \frac{1}{m}\right)^{kn}$ and $p = 1 - \left(1 - \frac{1}{m}\right)^{kn}$.

Now, after inserting n *elements*, suppose that we need to test the membership of an *element* which does not belong to the set (does not equal to any of the n inserted *elements*). The probability fp of declaring a *false positive* equals to the probability that the associated k *hash positions* of that *element* happens to be set to 1 accidentally by some of the *hash positions* of the existing n *elements*.
So: $fp = p^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$ using the approximation of: $1 - \frac{1}{m} \approx e^{-\frac{1}{m}}$.

Given m and n, the optimal k that minimizes fp is $k = \frac{m}{n} \ln 2$ where $fp = \left(\left(\frac{1}{2}\right)^{\ln 2}\right)^{\frac{m}{n}}$ (*).
Given n and the target fp, the required m is $\frac{-n \ln fp}{\ln^2 2}$ (*). Thus:
Given the target fp, the optimal number of bits per element $\frac{m}{n} = \frac{-1}{\ln 2} \log_2 fp$ where $k = -\log_2 fp$.

$*$ Given m and n, the optimal value of k that minimizes fp is $\frac{m}{n}\ln 2$.

Proof: We need to minimize $fp = \left(1 - e^{-\frac{kn}{m}}\right)^k$ with respect to k.
Note that since m and n are given, they are treated as constants.
We will minimize the easier function $\ln(fp)$ which is equivalent to minimizing fp.
$\ln(fp) = \ln\left(1 - e^{-\frac{kn}{m}}\right)^k = k\ln\left(1 - e^{-\frac{kn}{m}}\right)$
Let $r = e^{-\frac{kn}{m}}$, then $k = \frac{-m}{n}\ln(r)$.
Thus $\ln(fp) = \frac{-m}{n}\ln(r)\ln(1-r)$. We are now minimizing with respect to r.
Taking the derivative and equating it to zero (ignoring the $\frac{-m}{n}$ constant):
$\frac{-\ln(r)}{1-r} + \frac{\ln(1-r)}{r} = 0$. So $r\ln(r) = (1-r)\ln(1-r)$.
Clearly, $r = \frac{1}{2}$ satisfies the above equation, so $k = \frac{-m}{n}\ln(\frac{1}{2}) = \frac{m}{n}\ln(\frac{1}{2})^{-1} = \frac{m}{n}\ln 2$.
By substituting the optimal $r = e^{-\frac{kn}{m}} = \frac{1}{2}$ and $k = \frac{m}{n}\ln 2$ in the fp formula:
$fp = (1 - \frac{1}{2})^{\frac{m}{n}\ln 2} = (\frac{1}{2})^{\frac{m}{n}\ln 2} = \left((\frac{1}{2})^{\ln 2}\right)^{\frac{m}{n}}$.

$*$ Given n and the target fp, the required m is $\frac{-n\ln fp}{\ln^2 2}$.

Proof: By taking natural logarithm of both sides of the formula: $fp = \left(\frac{1}{2}\right)^{\frac{m}{n}\ln 2}$.
$\ln(fp) = \frac{m}{n}(\ln 2)\ln\left(\frac{1}{2}\right) = \frac{m}{n}(\ln 2)(-(\ln 2)) = \frac{-m}{n}(\ln 2)^2$. Thus $m = \frac{-n\ln fp}{\ln^2 2}$.