L3S Research Center
Large Scale Data Mining, SS 2016
Dr. Avishek Anand
Solution to Assignment 2, due: **28 April 2016**

Problem 1. For the situation of our running example in Bloom Filters (8 n bits, 1 m members of the set S), *Derive* the false-positive rate if we use three hash functions in terms of m and n. What if we use four hash functions?
Solution: First we derive the false positive rate when S contains m members and the array consists of n bits when we have k hash functions.
The false positive rate is simply the probability of getting at least one 1 in the array when encountering m members. (Refer to the lecture notes for the complete derivation)
The probability of bit being set to 1 in the array is $\frac{1}{n}$

The probability of none of the bits being set to 1 is $1 - \frac{1}{n}$

Since we have k hash functions, we have to assign km members to the array.

The probability that none of the km items get assigned to a 1 is $(1 - \frac{1}{n})^{km}$

The probability that at least one of the bits is a 1 when we have km items is given by
$1 - (1 - \frac{1}{n})^{km}$

when n is large this can be approximated as: $1 - e^{-\frac{km}{n}}$

we have k independent hash functions and only allow an item to pass through if all k hash functions result in a 1. Thus we get the false positive probability as: $(1 - e^{-\frac{km}{n}})^k$
For 8 n bits and 1 m bits with 3 hash functions we get: $(1 - e^{-\frac{3m}{8n}})^3$
Similarly for 4 hash functions we get $(1 - e^{-\frac{4m}{8n}})^4$

Problem 2.

1. Suppose we have n bits of memory available, and our set S has m members. Instead of using k hash functions, we could divide the n bits into k arrays, and hash once to each array. As a function of n, m, and k, what is the probability of a false positive? How does it compare with using k hash functions into a single array?

2. As a function of n, the number of bits and m the number of members in the set S, what number of hash functions minimizes the false- positive rate?

Solution:

1. If we divide the initial array in k parts, we get k arrays of size n/k. Each array has a single hash function.
   The false positive probability for a single array of size n/k is $1 - e^{-\frac{m}{n/k}}$

   For k arrays we get $(1 - e^{-\frac{km}{n}})^k$ which is the same as having k hash functions for a single array.

2. To find the k that minimizes the false positive rate we must differentiate g with respect to k and set the outcome to 0.
   Let $g = (1 - e^{-\frac{km}{n}})^k$. g is continuous and can be differentiated to find the minima.

$$\frac{dg}{dk} = \frac{d}{dk}\{1 - e^{-\frac{km}{n}}\}^k = 0 \tag{1}$$

Using the chain rule we find that $k = \frac{n}{m}ln(2)$

Problem 3.

1. Suppose our stream consists of the integers 3, 1, 4, 1, 5, 9, 2, 6,5. Our hash functions will all be of the form $h(x) = ax + b$ mod32 for some a and b. You should treat the result as a 5-bit binary integer. Determine the tail length for each stream element and the resulting estimate of the number of distinct elements if the hash function is:
   (a) $h(x) = 2x + 1$ mod32
   (b) $h(x) = 3x + 7$ mod32
   (c) $h(x) = 4x$ mod32
2. Do you see any problems with the choice of hash functions in the previous question? What advice could you give someone who was going to use a hash function of the form $h(x) = ax + b$ mod 2^k?

Solution:

1. The number of distinct elements is given by 2^R where $R = max_a\, r(a)$. $r(a)$ is the number of trailing 0's for an element a in the stream.
   (a) $R = 0$, the estimate is 1.
   (b) $R = 4$, the estimate is 16.
   (c) $R = 4$, the estimate is 16.
2. For this stream the ideal hash function should give us a tail length of 3 in order to get 8 as the estimate. However the first hash function only produces odd numbers which always end in 1. The third hash function produces multiples of 4 which always end in at least two zeros. The second hash function is not accurate because 32 is too large for the given stream.
   An ideal hash function should unifromly at random produce numbers in N, where N is the set of all integers. In this way the number of elements with r trailing 0s is given by 2^{-r}. We must choose a and b such that they do not consistently produce odd or even numbers. We must choose 2^k close to the size of the stream. $h(x) = 3x + 7$ mod 16 produces an estimate of 8.

Problem 4.

1. Compute the surprise number (second moment) for the stream 3, 1, 4, 1, 3, 4, 2, 1, 2. What is the third moment of this stream?

2. Suppose we are given the same stream, to which we apply the Alon-Matias-Szegedy Algorithm to estimate the surprise number. For each possible value of i, if X_i is a variable starting position i, what is the value of X_i.value?

Solution:

1. Second moment $= 2^2 + 3^2 + 2^2 + 2^2 = 21$.
   Third moment $= 2^3 + 3^3 + 2^3 + 2^3 = 51$
2.

i	1	2	3	4	5	6	7	8	9
X_i	3	1	4	1	3	4	2	1	2
X_i.value	1	1	1	2	2	2	1	3	2

Problem 5.
If a stream has n elements, of which m are distinct, what are the minimum and maximum possible surprise number, as a function of m and n?
Solution: The surprise number is maximized when the distribution frequency in a stream of length n is skewed heavily towards one variable in m.
The scenario that maximizes the second moment is a single element occuring $n-(m-1)$ times and the rest of the $m-1$ elements occuring once each.
Max second moment $= (n-m+1)^2 + (m-1).1^2$

The second moment is minimized when all m elements are distributed evenly n/m times.
Min second moment $= (\frac{n}{m})^2 m$