Daniel Mendes
4/24/22
CS 370

Assignment 1 Writeup

1) What hash functions did you choose and why (Hint: Cryptographic or noncryptographic)?:

I used only the cryptographic hash function MD5. When salted, cryptographic hash functions become independent. So, when using passwords to create the indices for my bit array, I added numbers incrementing from 0 to the ends of the passwords. The 3-hash filter just used 3 salted variants of the password, while the 5-hash filter used 5 salted variants.

2) What is the output range of the hash functions and how does it relate to the size of the bloom filter?:

The output of the MD5 hash function is a 128-bit digest or a string of 32 hex numbers. Having an array of 2^128 bits is impossible. So, I just cut down the size of the outputted hash to just 6 hex numbers. This gives an output range of (2^4)^6 bits, or around 16 kilobits that can be set in the filter.

3) What is the size of the Bloom filter in each case? Why did you pick this size?:

According to the functions provided in class, the chance of a false positive when choosing a word not trained in the filter is estimated by the equation. Note that k is the number of hash functions, B is the size of the dictionary, and N is the size of the bit array.

$$(1 - e^{-kB/N})^k$$

When plugging in B = 623,518 and k = [3 or 5], we can calculate the estimated probabilities of false positives given differently sized bloom filters. If I wanted a false positive rate of less than 0.001 (i.e. 0.1%), then these were the calculated values for N:

For 3 hash functions: $N \geq 1.78 * 10^7$

For 5 hash functions: $N \geq 1.07 * 10^7$

The 3-hash function case was really similar to the space occupied by 6 hex numbers, so I just used that value instead. For 5 hash functions, I just used a modulo operation on the hash number computed so that it would fit within the given range.

4) How long does it take for your Bloom Filter to check 1 password in each case (3 hashes vs 5 hashes)? Does one Bloom Filter (3 hashed vs 5 hashes) perform better than the other? Why or why not?:

According to my tests, the 3-hash bit array takes around 8-9 seconds to fully train, while the 5-hash bit array takes 13-14 seconds to train. This makes sense pretty intuitively. In both cases, the MD5 hash function is used to repeatedly calculate the hashes, taking up the bulk of computational power. Since 5 hashes is ~166% the computational power of 3 hashes, then this lines up with the estimated values ($8.0s * 1.66 \approx 13.3s$).

Of course there is some extra overhead, including the time to read/write files and to perform modulo operations on every hash for the 5-hash bit array. However, I'm analyzing the runtime under the assumption that the cryptographic hash functions are taking up most of the computational budget.

5) What is the probability of False Positives in your Bloom Filter for each case?:

Experimentally, the probability of false positives turned out to be 0 in both test cases. Every password that was in the list was reported, while every password that was out of the list was said to be good.

Mathematically, following the function specified above, the probability for the 3-hash filter it turns out to be:

$$(1 - e^{(-3*623518)/16777216})^3 \approx 0.0012$$

For the 5-hash bit function it turns out to be:

$$(1 - e^{(-5*623518)/10700000})^5 \approx 0.00103$$

This means that the 5-hash filter, even with the smaller range, still has a lower rate of false positives.

6) How can you reduce the rate of False Positives?:

Assuming that the number of hash functions and inputted passwords is constant, the only way to reduce the false positives is to increase the range of the bloom filter. Doing so also requires the range of numbers covered by the hashes to be used, so potentially 7 or 8 hex numbers could be used in this improved bloom filter (at the cost of space/time budget). The effect of this will be seen in the following question, where the range of the 5-hash bloom filter is increased.

If these aren't limited, then there are other ways of decreasing the false positive rate, though they are a bit more finicky. Increasing the number of hash functions used can decrease the false positive rate, but only up to a certain point. After implementing too many hash functions, this would actually increase the rate of false positives presumably due to the increasing number of collisions. This method is also computationally expensive. Decreasing the number of passwords the bit array was trained with also lowers the rate of false positives as there are simply less set bits. However, this has the effect of making the filter less effective at finding poor passwords as well.

7) What will happen to the False Positive Rate of your 5-hash bloom filter if you set the size of the filter to be the same as that used for 3-hash filter?:

When placed into the equation listed above, the result is this:

$$(1 - e^{(-5*623518)/16777216})^5 \approx 0.00014$$

This result has a false positive nearly an order of magnitude smaller than the 3-hash bloom filter.

8) What is the probability of False Negative in your Bloom Filters?:

It's impossible for a Bloom filter, by design, to produce any false negatives. When the inputted passwords are hashed, the ensuing indices for the bit arrays are deterministic. Thus, if a bad password (found in the password list) was inputted, it would be impossible to determine different indices than when the password was initially placed in the bit arrays.