Bloom Filter Observations

As I began this assignment, the first hurdle I ran into was deciding how large the bit array should be and how many hash functions should be used. All I knew was that I had around 600000 words in the dictionary file so I started there. I arbitrarily decided that a “low” false-positive rate would be 20%, as this seemed more realistic than something like 1%.

I found a useful tool (<https://hur.st/bloomfilter/>) that helped me calculate the correct number of functions to use. It determined that only 3 functions would be needed if I had a bit array that was 2000000 bits long.

Text

Description automatically generated with medium confidence

As I developed the bloom filter, I toyed around both with my program and this calculator to learn more about the ideal size of the bit array vs the number of hash functions.

When it comes to run time of a program, it’s pretty important to limit the number of hash functions used. However, increasing the number of functions will result in a stronger bloom filter. My program took over a minute to run with just three functions being used. Also, while more functions will result in a stronger filter, it will also increase the false-positivity rate. A chart provided by the site listed above shows the positivity rate (p) as the function number (k) grows.

**Chart

Description automatically generated**

One way to reduce the false-positivity rate without unduly increasing the number of functions is to instead increase the size of the bit array. Since the bit array does not take up much memory, it can get quite large without causing a problem. My bit array was 2000000 bits, which was 244.14KiB in size. This graph shows how the false-positivity rate decreases as the array size (m) gets larger.

Chart, line chart

Description automatically generated

If I were to change my bloom filter, I would focus on increasing the size of the array to achieve a smaller false-positive rate. Here’s the calculation if I tripled my array but left all other parameters the same.

Text

Description automatically generated

It’s hard to say that there is one “ideal” ratio between these variables, but the biggest conclusion I got from this is to choose a small but relevant set of hash functions to test against and then create a very large array to store the results. The bit array does not take up much memory at all, and it is worth it to have a fast program, especially if the bloom filter is meant to be used in an application by a user choosing their own password for an account (users want response time to be as fast as possible).