

**Brett Sullivan**

**7-21-24**

**CS-370**

## **Project 2 Bloom Filters**

- **Report on explanations to observations that are interesting or surprising. Below are some specific areas to address**

**How long does it take to run your code?**

With my bloom filter size set to 137,491,748 and my hash functions count at 4, it takes my program 39.46 seconds to run (varies a little bit from run to run). I noticed the count of hash functions made a significant difference in how long the code would run, and 4 seemed like a good tradeoff with time and accuracy.

**How many bits are in your bit array and why was this appropriate for your implementation?**

I initially had too few bits in my array. I was not paying enough attention to the massive size of rockyou.txt and the smaller yet still large size of dictionary.txt. Once I figured out that the 1 million size, I had was far too small, I searched how to calculate the approximate size, then made a python file to do so for me. It turned out being 137,491,

**How many hashing algorithms did you use and why was this appropriate for your implementation?**

I ended up using a single hashing algorithm in your Bloom filter implementation, specifically hashlib.sha256(). I set my hash functions count to 4 as this was a good amount to not have the program run for too long, but still provide enough hashing.

**Describe the meanings of true positive, true negative, false positive, and false negative in relation to a Bloom Filter.**

**True Positive:** An item is identified as present in the Bloom filter, and it actually exists in the set, in this case a bit array.

**True Negative:** An item is identified as absent in the Bloom filter, and it does not exist in the set. After confirming that it is in fact not in the array or set.

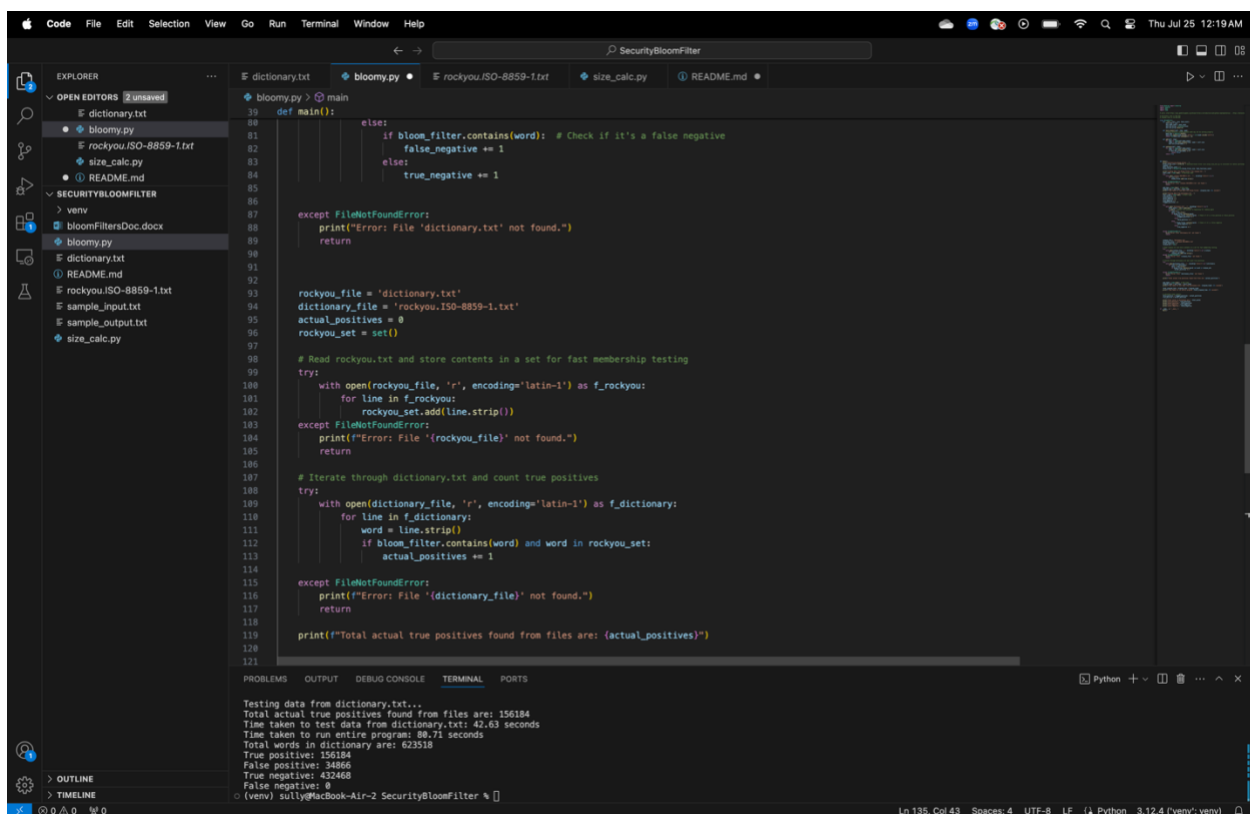
**False Positive:** An item is identified as present in the Bloom filter, but it does not actually exist in the set. Somewhat common in bloom filters.

**False Negative:** An item is identified as absent in the Bloom filter, but it actually exists in the set. This should never happen in a bloom filter.

**Provide statistics on true positive, true negative, false positive, and false negative for the dictionary.txt based on the rockyou.txt (i.e. preload rockyou.txt, test entries in dictionary.txt).**

I had many runs with my hash count and bloom filter size varying at times. Once I settled on these, and considering the dictionary size is 623,518, my rates were as follows: my true positives were 156,184, my false positives were 34,866. My true negatives were 432,468 each time, and my false negatives were 0.

**Include screenshot of output (show full screen / maximized terminal window - no clipping).**



The screenshot shows a VS Code editor with a Python script named `bloomy.py` open. The script is a Bloom filter implementation that tests a dictionary against a set of words from a rockyou file. The terminal output at the bottom shows the results of the test:

```
Testing data from dictionary.txt...
Total actual true positives found from files are: 156184
Time taken to test data from dictionary.txt: 42.63 seconds
Time taken to run entire program: 88.71 seconds
Total words in dictionary are: 623518
True positive: 156184
False positive: 34866
True negative: 432468
False negative: 0
```

Based on the design parameters of the Bloom filter, what was the projected false positive rate (see calculator link above) and how does that compare to your actual results?

According to the calculator, the false positive rate should have been around 1 in 74 or 0.13 roughly. Mine was substantially higher than that, somewhere around .05, and I am not sure why. IT could have to do with either my hash functions not being enough at 4, or my bloom filter size being slightly off.