Lab 17

Ali Ghasemi - S289223

INTRODUCTION

This simulation lab consists of different sections, each section will be analyzed and explained in order. The algorithms that are used will be explained and the outputs will be shown here as well.

PRELIMINARY STUDY AND SET OF SENTENCES

In this part of the simulation, the words, distinct words, verses, and the set of verses will be created. To make the verses, a sliding window method has been obtained. The inputs of the code are the Commedia text file and the size of the verses which will be the size of the sliding window. The output of this part will be the number of words, the number of distinct words and the number of verses and distinct verses. The size of the set of sentences is also available in the code (in bytes) and also in the final table at the end of the report. Also, the number of collisions are checked using the fingerprints for the set of sentences.

In this part of the simulation, first the sentences are created based on the sliding windows method with length of 6 and then for each word, a fingerprint is created using the hash function provided in the previous labs (md5). Then after creating the list of fingerprints, Python's keyword, set() is used. The lengths of these two items (the list and the set) are subtracted from each other. If this value is more than zero, it means that a collision has happened and if not, it means that there's no collision.

The same method is also used to find the Bexp in the next part.

FINGERPRINT SET

In the first question, we go through the different values for b (which is the minimum number of bits) and we find the smallest value for b that does not lead to a collision, and we call it Bexp.

In my simulation, this value was equal to 34.

In the second part we find the theoretical value for b and we call it Bteo. We use the formula below to find this value:

$$m = 1.17 * \sqrt{n}$$

$$and\ since\ n = 2^b\ we\ can\ say\ that\ b = Log_2^{m/1.17^2}$$

The obtained value is equal to 33. In the third question of this part, we must find the value of the false positive based on the value of Bexp which in my case is equal to 34.
Several formulas could be used, the below formula is one of them: $\varepsilon = m/n = m/2\hat{}b$
The answer to the fourth question is, yes, Bteo is a good approximation of Bexp since the they are equal to 33 and 34 (respectively).

BIT STRING ARRAY

In the first question, we create bit arrays with the different lengths. The lengths are based on the values that are given in the question for the memory considering the fact that $n = 2^b$. We create the hash values for each verse and we the set the value of the corresponding bit equal to one and at the end, we count the number of ones and divide it by the number inputs which will be the number of verses . You can see the output plot below in figure 1.

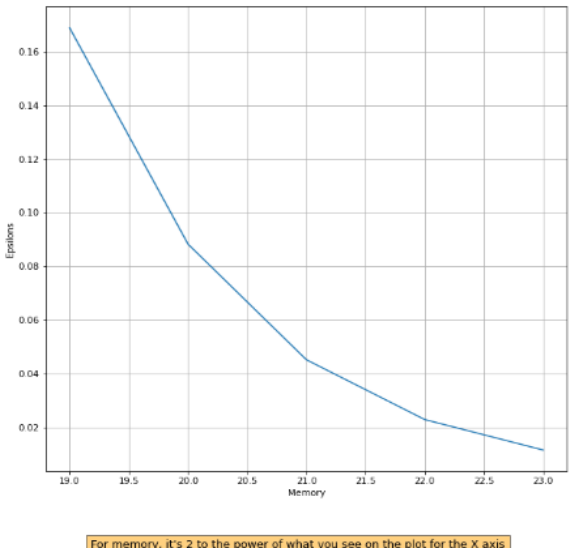

Figure 1

In the second part, we find the value for the probability of false positive using the formula $\varepsilon = m/n = m/2\hat{}b$.

You can see the output in figure 2.

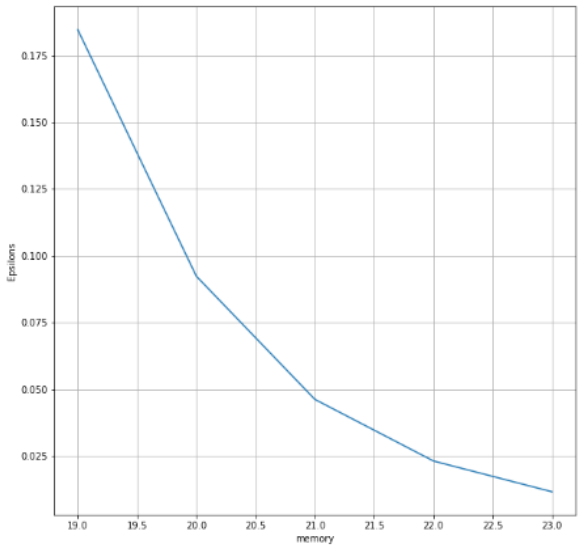

Figure 2

For the third question, I plotted the theoretical values for memory and the probability of false positive and the values obtained from simulation in a graph to compare them and as you can see in figure 3, they are close to each other, so we can assume that the theoretical values are good approximations of the values obtained from simulation.

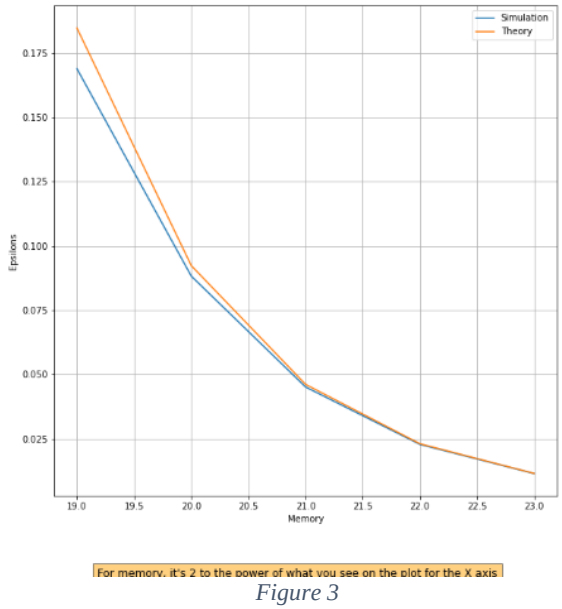

Figure 3

BLOOM FILTERS

In the first question, we need to find the right values for the number of hashes for different values for the memory. For that matter, we first need to find the optimal number of values for the number of hashes using the formula below (The output of this formula is available as a plot in the code).

$$\frac{n}{m} * \log(2)$$

This formula gives us real numbers, but we want integers, for that matter need to plot the performance graph (probability of false positive and the number of hashes) and find the right number of integer values for the number of hashes(k). You can see the performance graph in figure 4.

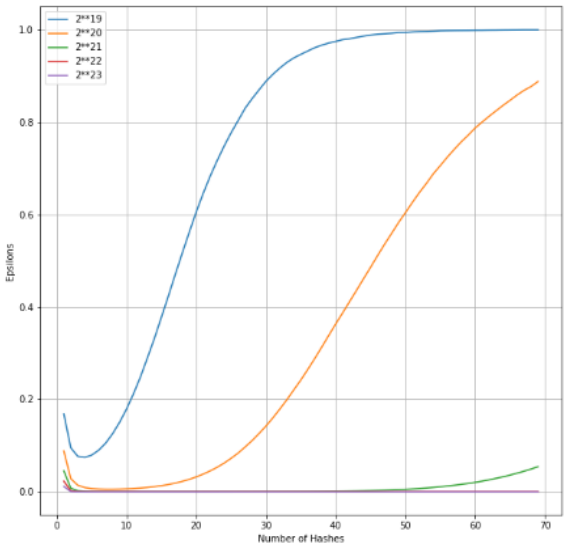

Figure 4

Based on the values obtained from the graph and the values for the probability of false positive, we can find the right values for the number of hashes. The values are as shown in the table below:

Number of hashes	Memory
4	2 ** 19
8	2 ** 20
15	2 ** 21
30	2 ** 22
60	2 ** 23

Table 1

In The second part of the questions, we calculate the values for epsilon based on the formula below:

$$(1 - e^{(-k*m/n)})^k$$

The answer to the third question is that each time we add a number to the string before using the hashing algorithm to have different hash values.

In the fourth question we go through all the values for memory and calculate the values for false positive based on the optimal values we've found for the optimal number of hashes (k). you can see the output of this part in figure 5.

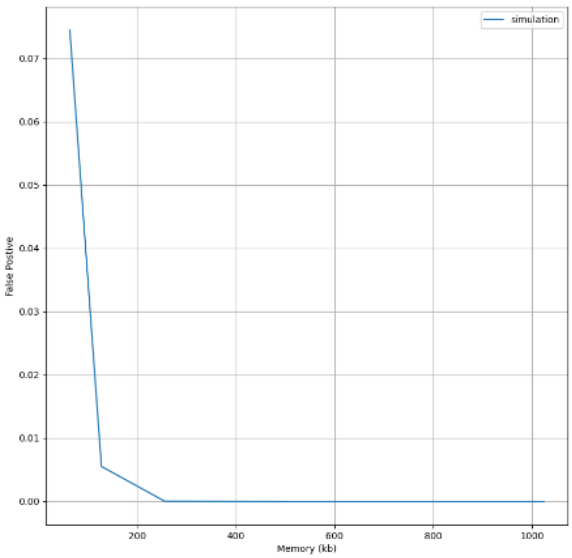

Figure 5

In the fifth part we compare the outputs of the simulation and theory, and we can see that these values overlap. You can see the output in figure 6

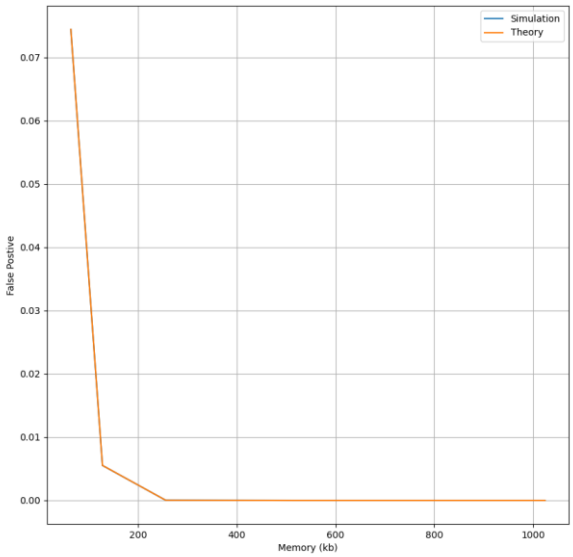

Figure 6

For the optional part, some explanations have been given in the code as comments.

You can see the final table below(for bit string array and bloom filters, the values for memory and P(FP) relate to each other respectively)

Dasta structure	Memory (KB)	Prob. False positive
Set of sentences	12684.7109375	5.636131390929222e-06
Fingerprint set (X=Bexp)	3808.0859375	5.636131390929222e-06
Bit string array (X varying)	64.078125, 128.078125, 256.078125, 512.078125, 1024.078125	0.16888427734375, 0.08826446533203125, 0.04514932632446289, 0.022835254669189453, 0.011480927467346191
Bloom filter (X varying)	64.078125, 128.078125, 256.078125, 512.078125, 1024.078125	0.07441333677135287, 0.005522559661331668, 2.9930506277165827e-05, 9.131377740901003e-10, 8.263579823178387e-19