Computer aided simulations and performance evaluation

Lab 11 - Bloom Filters

Gabriele Cuni 277957

11.1 Input Parameters

- **seed**: It is used to set the random library seed, which is used to shuffle two times the list of all the words.
- **words**: It is a python list of 370101 english words coming from the words_dictionary.json [1]
- **b**: Number of bits, experiments are done with b from 19 to 23.

11.2 Output Parameters

- **pFPSim**: it is the probability of false positives computed by simulation.
- **pFPTheo**: it is the theoretical probability of false positives.
- **bfSimByte**: It is the Bloom Filter memory occupancy computed by simulation.
- **bfTheoByte**: it is the theoretical Bloom Filter memory occupancy
- **bsTheoByte**: it is the theoretical Bit String Array memory occupancy.
- **ftTheoByte**: it is the theoretical memory occupancy of fingerprint in a table
- **kOpt**: It is the optimal k which is the number of hash functions.
- **distEl**: is is the theoretical number of distinct elements stored in a Bloom Filter

11.3 Main Data Structure

- **bitArray**: It is an array of bits made by the library bitarray [3]. It has range [1,n] where $n = 2^b$
- **all_bits_to_update**: It is a python list which contains all the k hash function results.

11.4 Formulas

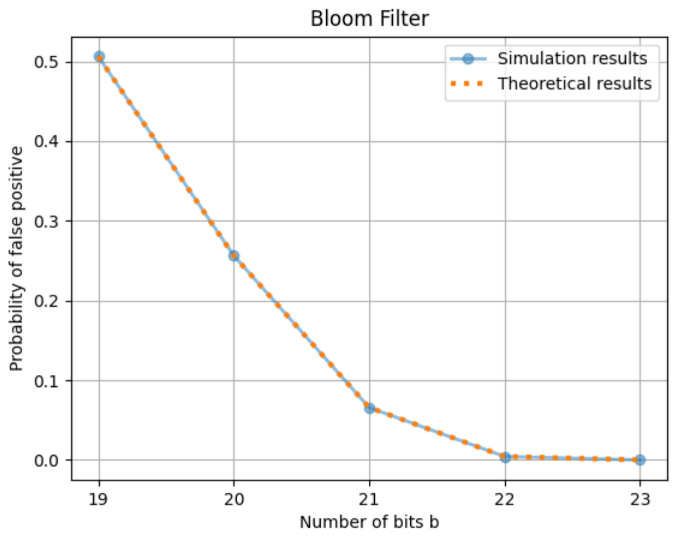
m is the number of words, k is the number of hash function and $n = 2^b$

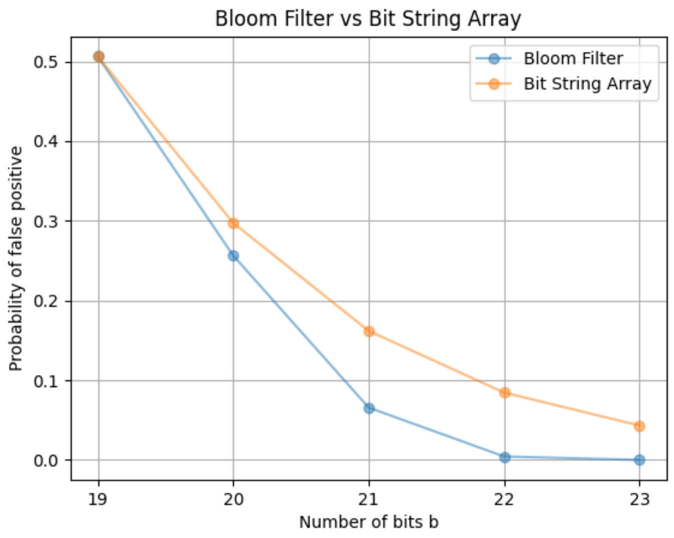
- $pFPSim = (\frac{number\ of\ ones\ in\ the\ bit\ array}{n})^k$
- $pFPTheo = (1 - e^{-\frac{k \cdot m}{n}})^k$
- $bfTheoByte = m \cdot 1.44 \cdot \log_2(\frac{1}{\epsilon})$ where $\epsilon = (1 - e^{-\frac{k \cdot m}{n}})^k$ the final result is divided by 8 to be in bytes
- $bsTheoByte = \frac{m}{\epsilon}$ where $\epsilon = 1 - (1 - \frac{1}{n})^m$ and the final result is divided by 8 to be in bytes
- $ftTheoByte = m \cdot \log_2(\frac{m}{\epsilon})$ where $\epsilon = 1 - (1 - \frac{1}{n})^m$ and the final result is divided by 8 to be in bytes
- $distEl = -\frac{n}{k}\log(1 - \frac{N}{n})$ where N is the actual number of bits equal to 1 in the Bloom Filter
- $kOpt = ceil(\frac{n}{m} \cdot \log(2))$

11.5 Python Functions

- **pympler.asizeof.asizeof():** It is used to assess the occupancy in bytes of **bitArray**. [2]
- **random.shuffle():** It is used two times to shuffle the words inside the list **words,** in order to make them not in the alphabetic order.

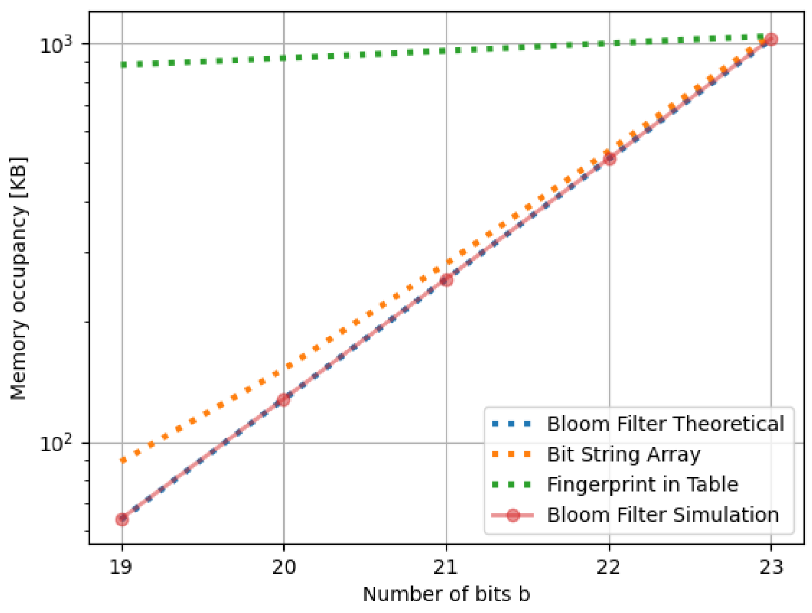
11.6 By simulation, evaluate the probability of false positive and compare the simulation results with the results obtained by just analytical formulas.





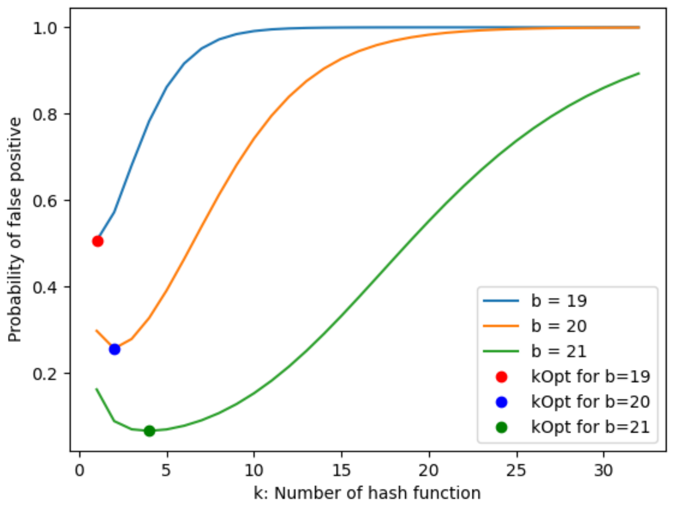
The first chart shows how well the simulation follows the theoretical formula for the probability of false positives. As you can see for low bit numbers the probability is so high that it does not make sense to use a Bloom Filter, but it is usable for a higher number of bits. The probability converges to zero by incrementing the number of bits and keeping steady the number of words.
The second chart shows that the Bloom filter converges faster than the Bit String Array, therefore choosing the first one approach is better than the second one in terms of probability of false positives. In order to make the second chart the script plot11.py takes as input also the result file of the workshop 10.

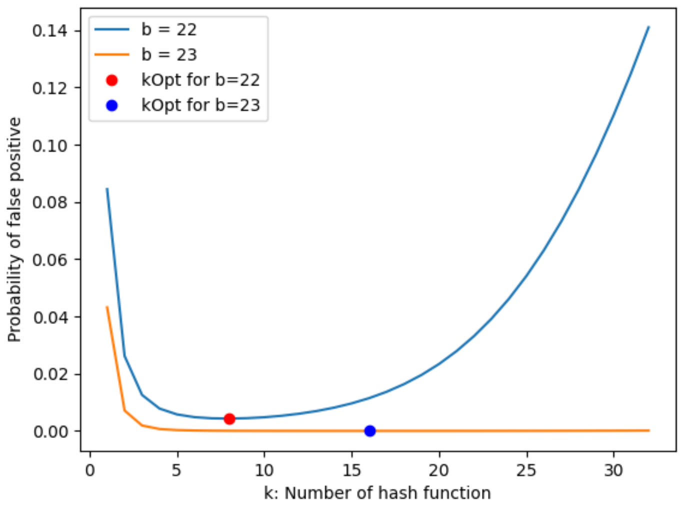
11.7 Compare the results with the scenario in which the fingerprint are actually stored as a integer (as in LAB 8 and 9) and as a bit string array (as in LAB 10)



The chart shows how well the simulation follows the theoretical formula for the Bloom Filter in terms of memory occupancy. Instead for the Bit string array and fingerprint only the theoretical formulas are taken in consideration. It can be seen that the bloom filter occupancy is slightly smaller than the bit string array and both are smaller than plain fingerprints in a table up to 23 bits.
From the previous chart we know that for 22 bit the probability of false positives is very small, therefore it makes sense to use a Bloom Filter with 22 bits given the probability of false positives and the memory occupancy.
Finally, from 23 bits onwards the fingerprint method should be prefered given the memory occupancy.

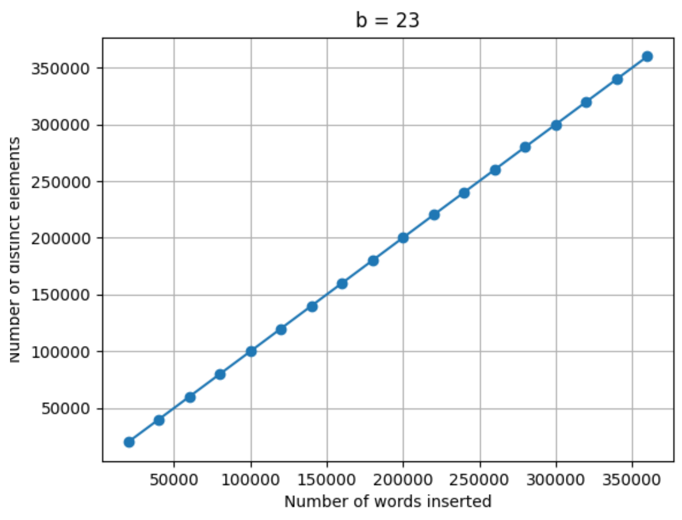
11.8 Show in a graph the probability of false positive in function of the number of hash functions





The two charts above show that the optimal number of hash functions **kOpt** computed by the theoretical formula is always the best choice for the analyzed number of bits, given that in all the cases the **kOpt** selected is the one with the smallest probability of false positive.

11.9 Show how accurate is the formula when inserting word-by-word into the dictionary, in function of the inserted words



The chart shows that the theoretical formula of the number of distinct elements stored in a bloom filter follows very well the real number of distinct elements that in this case is the number of words inserted, given that the words in the dataset are unique.

11.10 References

[1] https://github.com/dwyl/english-words
[2] https://pympler.readthedocs.io/en/latest/
[3] https://pypi.org/project/bitarray/