Mining Password Stream Data Project

This phase of the project focuses on using Bloom Filters for mining a stream of password data. Imagine you create a system in which users should have log-in data. To protect the user data, their passwords should be strong and unique. You can keep an archive with the passwords used with the help of a Bloom Filter. New user passwords will arrive to your system in a form of a stream and once a password arrived has been seen before by the filter, you'll need to warn the user that their password is weak.

1 Data

The data can be found in the .zip file you are given: "code/data/passwords.csv". It is a .csv file that contains approximately 670k user password data collected from 000webhost leak that is available online. Each password is in a different line and passwords have numbers, letters, and special characters.

2 Code

You are given a Python code in which the main methods of reading the data, reading the input parameters, and giving the results are implemented.
The methods that are implemented are described as following:

- **read_parameters():** Reads the input parameters from the "default_parameters.ini" parameter file. The input parameters are h and N, and they are described in Table 1. The method stores the input parameters in the "parameters_dictionary" dictionary they can be accessed as
"parameters_dictionary['parameter_name']".

Parameter Name	Type	Default	Description
h	int	3	the number of hash functions
N	int	500000	the size of the Bloom Filter

Table 1: Input Parameters

- **read_data(data_path):** Reads the input data that are in the "data_path" as a data stream. The stream is simulated with a for loop. Within the for loop you'll have to implement your methods to process the data.

You will need to use and extend this code in order to warn your users if their passwords are weak.

2.1 Important notes before you start

- Please use Python 3.8 or newer to develop you solution.
- Make sure that the code given is running on your environment **before** you start your own implementation.
- Make sure that the dataset is in "/data" directory of your project.
- There are some default empty methods in the code where you need to implement your tasks. You can change the methods and add more if you need, but **don't change** the way the code is reading the inputs and giving the outputs.
- For testing your code while you implement it, you can create a subset of the data that you can verify if the implementation is going well. There is a small dataset given in "/data/test" if you want to use.
- If you create your own dataset, or use another existing one, make sure the documents are in a directory and have sequential integer names starting from 1.
- Since you will be dealing with string variables that represent passwords, use case-sensitive comparison if needed.

3 Tasks

Your task is to implement a Bloom Filter in the code given, and experiment with the parameters of the method by following the tasks below:

1. Create h number of random hash functions with the form of

$$H_h(s) = (\sum_{i=1}^{len(s)} s[i] * p^i) mod N$$

   Use h and N as input parameters and p as a random prime number. Make sure that the h hash functions have different p.

2. Implement a Bloom Filter of size N of seen passwords, using the hash functions from the previous task. Use N as an input parameter.
3. Count the number of false positive passwords seen.

4 Report

In your report briefly describe the following:

1. What are Bloom Filters best used for? What are their limitations?

2. What is the theoretical probability of false positivity, given the size of the bit array of the Bloom Filter, the number of the hash functions and the number of elements expected to be inserted in the filter?

3. Using parameter $h = [1, 2, 3, 5, 10]$, and the default values for the rest of the parameters, how does the parameter h affect the quality of the Bloom Filter in terms of false positives? Is having more hash functions reducing the probability of having collisions in the filter, i.e. having less false positives, and why?

4. Using $N = [100000, 200000, 300000, 500000, 1000000]$, and the default values for the rest of the parameters, how does the size of the Bloom Filter affect the number of false positives? Why?

5 Delivery

You should deliver a ZIP file containing:

- the report in PDF format $(50\%points)$
- the source code in .py file(s) $(50\%points)$

Your project will be evaluated taking into consideration the following points:

- if the code runs using the command: "`python3 bloom.py`"
- if the code doesn't include external libraries that solve the tasks (for example, numpy is accepted, bloom_filter is not accepted).
- if the code doesn't have bugs.
- if the code gives the correct results.
- if the code runs fast (we are dealing with stream data).
- if the answers in the report have **justification**, i.e. answering the "why" questions.