Project 2: Report

I have created a total of 4 classes for this project:

1) "Main" class

2) "BloomFilter" class - contains the implementation of Bloom Filter.

Important methods:

encode: This encodes elements into the bloom filter.

checkSetAElementsPresent: checks if Set A elements are present in the Bloom Filter.
checkSetBElementsPresent: checks if set B elements are present in the Bloom Filter.

3) "CountingBloomFilter" class - contains the implementation of the Counting Bloom Filter.

Important methods:

encode: This method encodes elements in the Counting Bloom Filter.

remove: removes elements from the counting bloom filter.

lookOriginalElements: looks for original elements of set A in the counting bloom filter.

4) "CodedBloomFilter" class - contains the implementation of the Coded Bloom Filter.

Important methods:

encode: encodes the elements in the Coded Bloom Filter.

performLookupForElements: performs lookup on elements in the coded bloom filter.

The "Main" class has the "main" function which is the entry point of the program. This main function gives you a list of options and asks from the user for the input.

The list of options displayed are:

1: Bloom Filter 2: Counting Bloom Filter 3: Coded Bloom Filter 4: exit

Based on the user input, it invokes the implementation methods of Bloom Filter, Counting Bloom Filter and Coded Bloom Filter classes respectively.

Steps to run the program:

1) copy all the .java code files in one folder/directory

2) compile all the .java code files using this command on the command line: **javac *.java**

3) run the command "**java Main**" on the command line to invoke the "main" method

   of the “Main class”. A list of options will be displayed as:

   Enter 1: Bloom Filter 2: Counting Bloom Filter 3: Coded Bloom Filter 4: exit

4) select one of the options (1, 2, or 3) from the above options according to which filter code you want to run by entering a number (1,2 or 3) on the command line and press enter. Select 4 if you want to exit the program.

5) The code will run for the type of filter (bloom filter, counting, coded bloom filter) selected and the output will be printed on the console/terminal and also output files will be created.

6) After the console output is generated, the main menu will appear again with the same options as shown in step 3 above so that we can run the program again and again if required.

Currently, all the three Bloom Filter implementation is called from the "main" java function in the "Main" class using the parameter values given in the project requirements file.

For generating the elements in all the three types of Bloom filter implementations, I am using Math.random() function. My elements lie between 1 and Integer.MAX_VALUE in Java.

Also, for the k hash functions, I have defined an integer array of size k as h[k] and initialized with randomly generated values. I am using this to compute Hi(f). So, Hi(f) is computed using H(f XOR h[i]). This is the same logic to implement multiple hash functions as written in the project document shared on canvas by the professor.

Different Bloom Filter Invocations and Function Parameters:

1) BloomFilter.callBloomFilter(1000, 10000, 7) - calls the Bloom Filter implementation and results are printed on the console. The first parameter in the " callBloomFilter " method call is the number of elements to be encoded(1000 here), the second parameter is the number of bits in the filter (10000 here) and the third parameter is the number of hashes (7 here). If you want to change any of these parameters, change them here in the "main" method of the Main class while calling the “callBloomFilter " method of BloomFilter class.

2) CountingBloomFilter.callCountingBloomFilter(1000, 500, 500, 10000, 7) - calls the Counting Bloom Filter implementation.
The first parameter in the " callCountingBloomFilter " method is the number of elements to be encoded (1000 here), the second parameter is the number of elements to be removed (500 here) ,third parameter is the number of elements to be added(500), the fourth parameter is the number of counters (10000 here) and the last parameter is the number of hashes (7 here). If you want to change any of these parameters, change them here in the "main" method of the Main class in the " callCountingBloomFilter " function call of CountingBloomFilter class.

3) CodedBloomFilter.callCodedBloomFilter(7, 1000, 3, 30000, 7);
- Calls the Coded Bloom Filter implementation and results are printed on the console.

The first parameter in the "callCodedBloomFilter " method is the number of sets (7 here), the second parameter is the number of elements in each set (1000 here) , the third parameter is the number of filters (3 here) , 4th parameter is the number of bits in each filter (30000 here) and the last parameter is the number of hashes (7 here). If you want to change any of these parameters, change them here in the "main" method of the "Main" class in the " callCodedBloomFilter " function call of CodedBloomFilter class.

Output of Program:

For each of the Filter implementations (Bloom Filter, Counting Bloom Filter, Coded Bloom Filter), the outputs are printed on the console.

Also, the output files for different filters will be created in the current directory every time we run the code for different filters by selecting options (1, 2 or 3) from the menu. The file names of different output files are:
1) BloomFilter.txt - output file for bloom filter.
2) CountingBloomFilter.txt - output file for counting bloom filter.
3) CodedBloomFilter.txt - output file for coded bloom filter.

One output file with names as specified above for each filter is also submitted along with the code files.