**Project 2**

**Venkata Sai Karthik Metlapalli**

**UFID: 65764476**

**Description**

* This project involves java-based implementation of Bloom filter, counting Bloom filter, and coded Bloom filter.
* The zipped folder includes the source codes and the output text files generated for each of the hashing algorithms

1. **Bloom Filter:**

* This Bloom Filter is implemented in JAVA and the class file name is BloomFilter.java.
* To run this file, go to the IDS-2/src directory, then type the following commands in command prompt/ terminal
  + - javac BloomFilter.java
    - java BloomFilter <no of elements to be encoded> <no of bits in the filter> <no of hashes>

This program accepts input parameters through command line arguments. For the demo input used for output file generation, the values of number of elements to be encoded, number of bits in the filter, and number of hashes are 1000, 10000 and 7.

In case of wrong input parameters, the program throws valid exception messages to the user suggesting to correct the input.

* After passing the inputs, the parametrized constructor initializes the values passed as class level variables.
* The array s[number of hashes for each flow] is initialized with unique random numbers in the constructor using random() function in java.
* Next in the fillBloomFilter() function, flowIDs in sets A and B are randomly generated using random() function in java.
* In the generateRandomElements() function, we calculate the XOR of each flowID in sets A and B with all the random numbers in array s[no Of hashes] and store in filterMap where key = flowID and value = array of all the XORed hash values.
* Then in the encode () method, all the values of each flow in set A are stored in the Bloom Filter and the corresponding value in Bloom filter/ bit array is set to 1.
* Once all the values of Set A are encoded, lookup function is called to check the bitmap. If all the hashes of a particular element in Set A are 1, then the particular element is present in the bloom Filter else it is not.
* Similarly, lookup is also performed for the elements of Set B.
* Output file “NewOutputBloomFilter.txt” is generated which contains the elements found in bloom filter when looked up for set A and the second line consists of the elements looked up for set B.
* For the set A, we could find all the elements in it in the bloom filter. Hence output of 1000 for A and for B it gives around 5-15 false positive values. (Though not present, bloom filter outputs present)

1. **Counting Bloom Filter:**

* This Counting Bloom Filter is implemented in JAVA and the class file name is CountingBloomFliter.java.
* To run this file, go to the IDS-2/src directory, then type the following commands in command prompt/ terminal
  + - javac CountingBloomFliter.java
    - java CountingBloomFliter < number of Elements to be encoded > < number of elements to be removed> < number of elements to be added>< number of bits in the filter > < number of hashes >

This program accepts input parameters through command line arguments. For the demo input used for output file generation, the values of no of elements to be encoded, no of elements to be removed, no of elements to be added, no of bits in the filter, and no of hashes are 1,000, 500, 500, 10,000 and 7, respectively.

In case of wrong input parameters, the program throws valid exception messages to the user suggesting to correct the input.

* After passing the inputs, the parametrized constructor initializes the values passed as class level variables.
* The array s[number of hashes for each flow] is initialized with unique random numbers in the constructor using random() function in java.
* Next in the fillBloomFilter() function, flowIDs in set A are randomly generated using random() function in java. These are stored in the originalElement Set.We also calculate the XOR of each flowID in set A with all the random numbers in array s[no Of hashes] and store in filterMap where key = flowID and value = array of all the XORed hash values.
* Then in the encode () method, all the values of each flow in set A are stored in the Bloom Filter and the corresponding value in Bloom filter/ bit array is set to 1.
* Then the removeElements method is called, which removes the values from Bloom Filter for the specified input of no of elements to be removed and decrements the corresponding counter of these hash values.
* After this, the addElements method generates a set of random numbers, XORs them with array s and hashes its values in the bloom filter i.e., encodes in the counting bloom filter by increasing the counter.
* Once all the above steps are done, lookup function is called to check the bitmap for all the elements of originalElementSet. Here we check for false positives.
* Output file “NewOutputCountingBloomFilter.txt” is generated which contains the elements found in the counting bloom filter when looked up for the original elements of set A.This value is around 500-520 entries.

1. **Coded Bloom Filter:**

* This Coded Bloom Filter is implemented in JAVA and the class file name is CodedBloomFliter.java.
* To run this file, go to the IDS-2/src directory, then type the following commands in command prompt/ terminal
  + - javac CodedBloomFliter.java
    - java CodedBloomFliter <number of Sets> < number of Elements to be encoded > < number of filters> < number of bits in the filter > < number of hashes >

This program accepts input parameters through command line arguments. For the demo input used for output file generation, the values of no of sets, no of elements to be encoded, no of filters, no of bits in the filter, and no of hashes are 7, 1000, 3, 30000 and 7, respectively.

In case of wrong input parameters, the program throws valid exception messages to the user suggesting to correct the input.

* After passing the inputs, the parametrized constructor initializes the values passed as class level variables.
* The array s[number of hashes for each flow] is initialized with unique random numbers in the constructor using random() function in java.
* The constructor also makes a call to assign method which creates a map of size log (g +1) for given g sets. This map has the bloom filters to be filled with.
* In the fillBloomFilter method, we generate g sets of random numbers. We also calculate the XOR of each flowID in sets with all the random numbers in array s[no Of hashes.
* Then we assign each set a code of length log(g+1). Encoding of a set in specific jth bloom filter happens if character at jth index for binaryCode of the given set is 1.
* Finally, lookup function is called which checks for all the values generated in sets and generates a new code. For each bloom filter, it checks if all the hashes could be found. If yes, it puts the 1 in that location of string else puts 0. Then count of lookup is done by matching the binaryCode generated vs the actual code.
* Output file “NewOutputCodedBloomFilter.txt is generated which contains the elements found in bloom filters when looked up for the elements of set A. This value is greater than 6700+ entries for the given input.