**IDS Project 2**

For the ease of operation, the input is automatically set for every run in eclipse by following the steps below:

run-> run configurations->arguments-> give the required values.

**Bloom Filter:**

Bloom Filter is a data structure that we use to encode a set of elements into a bit array and then check its membership with a lookup function

The flow of the program is as follows:

The global variables declared are:

* numElements: These are the number of elements that we will be encoding from our flows
* numBits: These are the total number of bits that we have to store the encoded elements
* numHashes: This is also refereed as ‘k’ which stands for the number of times that we will be hashing.
* bitMapArray: This is an array where we update the encoded elements.
* randomHashes: This is an array of random values that are used for hashing.
* uniqueSetElements: We use this set to store all the random elements we generate for sets A and B to avoid repetition
* rand: this is an instance of Random class which we will be using for generating random values.

Here we have the following methods:

* Parameterized constructor -> BloomFilter
  + This takes in the input values that are read from the command line and set them to the global variables that have been declared along with initializing the set and arrays with respective sizes.
  + For every instance of the object created here we initialize an array that is filled with random values for hashes.
* FillRandomHashes
  + This method starts off by creating a set in which we use a random number generator to fill it in with unique random values.
  + The values form this set is then iterated one by one to fill in the array of random hashes that we will be using through the program.
* hashXOR
  + This is the hash function that we will be using to generate hash values.
  + Here we declare a new array and start filling it in with the hashes generated by using XOR of flow ID that is taken in and the random hashes array values that we generated earlier.
  + This generates a new hashed array whose contents are uniquely hashed.
* fillBloomfilter
  + Here we create two sets A and B and then fill them with random elements and their hashValues.
  + We then encode these map A and lookup to see how many of them are encoded.
  + We take another set B that is filled with completely new elements and their hashes, we try to look up these elements in our bitArray and return the result that contains both the count values.
* fillSets:
  + we fill the sets with unique random elements at the same time keep updating the set of elements and the map.
* encode
  + we go through the hashed values and encode the bits as 1 in bitArray.
* lookup
  + we assign the count as total elements and check the bit array for all the hashed indexes, if they are 0 that menas they are not mapped and hence we reduce the count.
* main
  + This method takes in the input and gives the values to the required variable names. We invoke the constructor by creating the object here and then we true to read the output of this from eclipse IDE console to a new file in the system.

**Counting Bloom Filter:**

Here we create a set fill it up encode it and later remove the specified number of elements and add new elements and look up to see if they are encoded or not and then get the count and display it.

The flow of the program is as follows:

The global variables declared are:

* numElements: These are the number of elements that we will be encoding from our flows
* numBits: These are the total number of bits that we have to store the encoded elements
* numHashes: This is also refereed as ‘k’ which stands for the number of times that we will be hashing.
* numElementsToAdd: These are the number of elements to be added after removing from initial set
* numElementsToRemove: These are the number of elements to be removed from initial set.
* bitMapArray: This is an array where we update the encoded elements.
* randomHashes: This is an array of random values that are used for hashing.
* initialElemnts: We use this set to store all the random elements we generate for sets A to keep track of our initial original values.
* rand: this is an instance of Random class which we will be using for generating random values.

Here we have the following methods:

* Parameterized constructor -> CountingBloomFilter
  + This takes in the input values that are read from the command line and set them to the global variables that have been declared along with initializing the set and arrays with respective sizes.
  + For every instance of the object created here we initialize an array that is filled with random values for hashes.
* FillRandomHashes
  + This method starts off by creating a set in which we use a random number generator to fill it in with unique random values.
  + The values form this set is then iterated one by one to fill in the array of random hashes that we will be using through the program.
* hashXOR
  + This is the hash function that we will be using to generate hash values.
  + Here we declare a new array and start filling it in with the hashes generated by using XOR of flow ID that is taken in and the random hashes array values that we generated earlier.
  + This generates a new hashed array whose contents are uniquely hashed.
* fillBloomfilter
  + Here we create set A and then fill with random elements and their hashValues.
  + We then encode these map A
  + Now we remove the specified number of elements and then add new random values to this and lookup to see how many of them are encoded.
* fillSets:
  + we fill the sets with unique random elements at the same time keep updating the set of elements and the map.
* encode
  + we go through the hashed values and encode the bits and increment the value by 1 everytime in bitArray.
* lookup
  + we assign the count as total elements and check the bit array for all the hashed indexes, if they are 0 that menas they are not mapped and hence we reduce the count.
* addElements
  + this will add the required number of elements
* removeElements
  + this will remove the specified number of elements
* main
  + This method takes in the input and gives the values to the required variable names. We invoke the constructor by creating the object here and then we true to read the output of this from eclipse IDE console to a new file in the system.

**Coded Bloom Filter:**

Here we create a set fill it up encode it and later remove the specified number of elements and add new elements and look up to see if they are encoded or not and then get the count and display it.

The flow of the program is as follows:

The global variables declared are:

* numElements: These are the number of elements that we will be encoding from our flows
* numBits: These are the total number of bits that we have to store the encoded elements
* numHashes: This is also refereed as ‘k’ which stands for the number of times that we will be hashing.
* numSets: These are the number of 1000 element sets to be created
* numFilters: These are the number of filters well be using.
* randomHashes: This is an array of random values that are used for hashing.
* rand: this is an instance of Random class which we will be using for generating random values.

Here we have the following methods:

* Parameterized constructor -> CodedBloomFilter
  + This takes in the input values that are read from the command line and set them to the global variables that have been declared along with initializing the set and arrays with respective sizes.
  + For every instance of the object created here we initialize an array that is filled with random values for hashes.
* FillRandomHashes
  + This method starts off by creating a set in which we use a random number generator to fill it in with unique random values.
  + The values form this set is then iterated one by one to fill in the array of random hashes that we will be using through the program.
* hashXOR
  + This is the hash function that we will be using to generate hash values.
  + Here we declare a new array and start filling it in with the hashes generated by using XOR of flow ID that is taken in and the random hashes array values that we generated earlier.
  + This generates a new hashed array whose contents are uniquely hashed.
* fillBloomfilter
  + Here we create sets A and then fill with random elements and their hashValues.
  + We then encode these map A
  + Now we remove the specified number of elements and then add new random values to this and lookup to see how many of them are encoded.
* fillSets:
  + we fill the sets with unique random elements at the same time keep updating the set of elements and the map.
* encode
  + we go through the hashed values and encode the bits set the value by 1 in bitArray.
* setCode:
  + this is the logical operations performed to sett the code
* main
  + This method takes in the input and gives the values to the required variable names. We invoke the constructor by creating the object here and then we true to read the output of this from eclipse IDE console to a new file in the system.