**Project 3**

**Venkata Sai Karthik Metlapalli**

**UFID: 65764476**

**Description**

* This project involves java-based implementation of Countmin, Active counter, and Counter sketches.
* The zipped folder includes the source codes and the output text files generated for each of the algorithms

1. **Count Min:**

* This Count Min sketch is implemented in JAVA and the class file name is CountMin.java.
* To run this file, go to the IDS-3/src directory, then type the following commands in command prompt/ terminal
  + - javac CountMin.java
    - java CountMin <no of counter arrays/ k> <no of counters in each array>

This program accepts the no of counter arrays and no of counters in each array as input parameters through command line arguments. It also reads the flowIDs and number of packets in each flow from the project3input.txt file. For the demo, the input text file is placed in the resources folder in the main project directory. And the values of command line arguments passed are 3 and 3000.

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

* The CountMin class consists of a 2D counterArray, list of flowIds, and maps to store the actual size of every flow, calculated estimated size after recording and querying and error values. These are stored in the actualSize, estimatedSize, error hashMaps respectively.
* 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 recordAll() function, flowIDs and respective flow sizes that are read from input file in the main method are stored in the actualSize map. Then record method is called for each flow to record the packet in the countMin sketch.
* In the record() function, the XOR of each flowID with all the random numbers in array s[no Of hashes] is calculated and is stored in xorResult array. Then for each element/packet of this flow, counterArray is updated k times, once at calculated index in each row i.e., index to record this flowId = Hash2(Hash1(flowId) XOR s[hashArray\_value]) mod w
* Then in the query() method, the minimum of all the counter values for each flow is stored in estimatedSize map.
* The error in size estimation for each flow is nothing but the difference in actual and estimated sizes. These are stored in the error hashMap where key = flowId and value = error.
* The average error of all the flows is computed using getAverage() method.
* Output file “NewOutputCountMin.txt” is generated which contains the average error of all flows and top 100 flows sorted based on estimated size values. (Output contains flow ID, estimated and true sizes).

1. **Counter Sketch:**

This Counter Sketch is implemented in JAVA and the class file name is CounterSketch.java.

* To run this file, go to the IDS-3/src directory, then type the following commands in command prompt/ terminal
  + - javac CounterSketch.java
    - java CounterSketch <no of counter arrays/ k> <no of counters in each array>

This program accepts the no of counter arrays and no of counters in each array as input parameters through command line arguments. It also reads the flowIDs and number of packets in each flow from the project3input.txt file. For the demo, the input text file is placed in the resources folder in the main project directory. And the values of command line arguments passed are 3 and 3000.

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

* The CounterSketch class consists of a 2D counterArray, list of flowIds, and maps to store the actual size of every flow, calculated estimated size after recording and querying and error values. These are stored in the actualSize, estimatedSize, error hashMaps respectively. The hashCodes map consists of flowId(string) and its corrsponding hash code value.
* After passing the inputs, the parametrized constructor initializes the values passed as class level variables.
* The array s[number of hashes for each flow] and hashCodes are initialized with unique random numbers in the constructor using random() function in java.
* Next in the recordAll() function, flowIDs and respective flow sizes that are read from input file in the main method are stored in the actualSize map. Then record method is called for each flow to record the packet in the CounterSketch.
* In the record() function, the XOR of each flowID’s hashCode with all the random numbers in array s[no Of hashes] is calculated. For each element/packet of this flow, counterArray is updated k times, once at calculated index in each row i.e., index to record this flowId = Hash2(Hash1(flowId) XOR s[hashArray\_value]) mod w.
* If the first bit of hash is 0 then the flowsize of that flowId is subtracted else its value is added at the corresponding index.
* Then in the query() method, the estimated flow Sizes from each of the ‘k’ rows/counters is calculated based on if the 0th bit is set/unset like above and these are stored in estimates array.
* This estimates array is sorted in increasing order and the median value is calculated and stored in the estimatedSize Map for this flowId.
* The error in size estimation for each flow is nothing but the difference in actual and estimated sizes. These are stored in the error hashMap where key = flowId and value = error.
* The average error of all the flows is computed using getAverage() method.
* Output file “NewOutputCounterSketch.txt” is generated which contains the average error of all flows and top 100 flows sorted based on estimated size values. (Output contains flow ID, estimated and true sizes).

1. **Active Counter:**

This Active Counter is implemented in JAVA and the class file name is ActiveCounter.java.

* To run this file, go to the IDS-3/src directory, then type the following commands in command prompt/ terminal
  + - javac ActiveCounter.java
    - java ActiveCounter <numberPartBits> <exponentPartBits>

This program accepts the no of bits in the number Part of the counter array and the no of bits in the exponent part of counter array as input parameters through command line arguments. For the demo, the values of command line arguments passed are 16 and 16.

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

* The ActiveCounter class consists of a counterArray, numberBits and exponentBits to store the values in number and exponent part, max variable which stores the overflow value for numberPart.
* After passing the inputs, the parametrized constructor initializes the values passed as class level variables.
* In the add() method, incrementing of number bits is done with probability of 1/2exponentBits. To implement this a random number between 0 and 1 is generated and if it is less than 1/2exponentBits, then the number part is incremented. Else not incremented.
* If there is an overflow of numerical part, then numerical/ number part is right shifted and the exponent part is incremented by 1.
* Output file “NewOutputActiveCounter.txt is generated which contains the value recorded in the ActiveCounter which is equal to (numberBits \* 2exponentBits).