**Project 5**

**Venkata Sai Karthik Metlapalli**

**UFID: 65764476**

**Description**

* This project involves java-based implementation of Virtual Bitmap and bSketch( HLL).
* The zipped folder includes the source codes and the output text files generated for each of the algorithms.
* The project5Input.txt is placed in the resources folder and contains the input for the files.

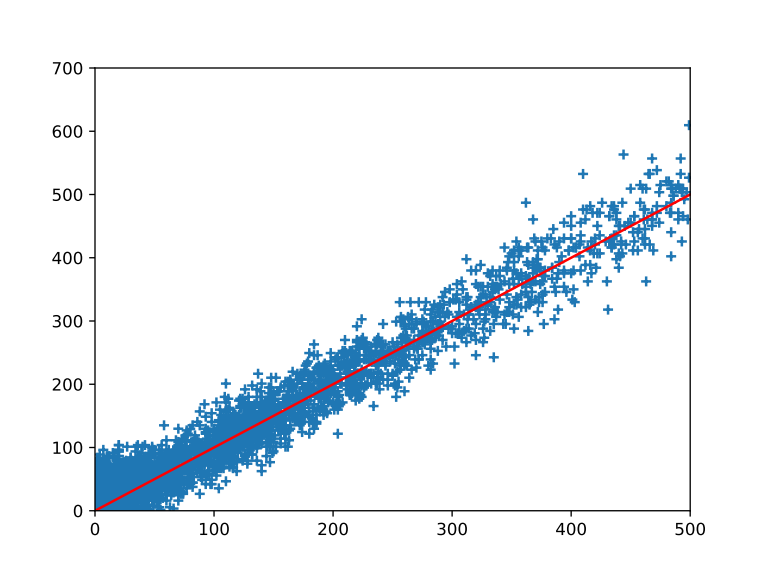
1. **Virtual Bitmap:**

* This Bitmap is implemented in JAVA and the class file name is Bitmap.java.
* To run this file, go to the IDS-5/src directory, then type the following commands in command prompt/ terminal
  + - javac VirtualBitmap.java
    - java VirtualBitmap <no of bits in physical array (m)> < no of bits in virtual bitmap (l)>

This program accepts m and l as input parameters through command line arguments. It also reads the flowIDs and spread of each flow from the project5input.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 500000 and 500.

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

* The VirtualBitmap class consists of a 1D parent bitMap, parent bitmapSize, virtual bitmap size, an array of random integers () and maps to store the flowId and its flow elements, actual spread of every flow, calculated estimated spread after recording and querying. These are stored in the ipValueMap, ipCountMap and estimatedSpread hashMaps respectively.
* After passing the inputs, the parametrized constructor initializes the physical bitmap of the passed size whose initial values are all zeros, and unique random numbers of given spreads are generated and placed in the ipValueMap using fillValuesInIPMap() function which makes use of random() function in java. The constructor also creates a random number array of l size.
* Next in the record () function, we set indices in parent BitMap to 1, for all the unique elements of all flows. B[H(f⊗R[H(e) mod 𝑙]) mod m]=1
* Then in the estimate() method, estimation of flow spreads is done using the l \* ln(Vb) - l \* ln(Vf) formula. If the value turns out to be less than 0, then 0 is inserted in the estimated spread value. Similarly for larger flows, we get infinity (since Vb is 0) and hence we store INT\_MAX.
* Output consists of Result.pdf for which has true spreads on X axis and estimated spreads on Y- axis.
* Output also includes NewOutputVirtualBitmap.txt that has the true and estimated spread values for all the flows in the input file. The estimated spreads that are below 500 are stored as comma separated string in NewOutputVirtualBitmapGrpah.txt that is used in constructing the above graph.
* **NOTE:** For larger spread values, the entire bitmap values are set to 1 and hence the fraction of zeros = 0; which causes ln value to be undefined since ln(0) is undefined and hence the program outputs INTEGER.MAX\_VALUE for such cases.



1. **BSkt(HLL):**

* This bSkt is implemented in JAVA and the class file name is BSkt.java.
* To run this file, go to the IDS-5/src directory, then type the following commands in command prompt/ terminal
  + - javac BSkt.java
    - java BSkt < number Of Estimators > < number Of Registers > < registerSize > < k >

This program accepts input parameters through command line arguments. It also reads the flowIDs and spread of each flow from the project5input.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 4000, 128, 5 and 3 respectively.

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

* The BSkt class consists of registerSize, numberOfEstimators(m), numberOfRegisters(l -> in one estimator), 2 dimensional array A to record the elements and maps to store the flowId and its flow elements, actual spread of every flow, calculated estimated spread after recording and querying. These are stored in the flowValues, actualSpread, estimatedSpread hashMaps respectively.
* After passing the inputs, the parametrized constructor initializes the l registers in m estimators of the given size whose initial values are all zeros, and unique random numbers of given spread sizes are generated and placed in the flowValues using fillValuesInIPMap () function which makes use of random() function in java.
* Next in the recordAndQuery() function, we generate k+1 random numbers for k number of estimators each flow is hashed to i.e., Hi(f) and one for H(e). For each flowID, we xor each unique element of the flow with the above generated random number. Hi(f) for all k hash functions is calculated as flowId.hashCode() ^ randomNumber[i]. We also calculate G’(element) which is a geometric hash by finding the number of leading zeros in the element’s binary representation and adding 1 to it. We store in A[Hi(f)][H(e)], the greater value among G’(e) and A[Hi(f)][H(e)].
* In the query part of each flow, we calculate the estimated spreads using the formula.
* All the flowIds, actualSpread and estimatedSpread values are outputted in the NewBSkt2.txt file for the top 25 largest estimated flow spreads.