Describe one approximate algorithm for lossy counting &one for sketch

1. Lossy count algorithm.

The lossy count algorithm is an algorithm to identify elements in a data stream whose frequency exceeds a user-given threshold. The algorithm works by dividing the data stream into buckets for frequent items, but fill as many buckets as possible in main memory one time. The frequency computed by this algorithm is not always accurate, but has an error threshold that can be specified by the user. The run time and space required by the algorithm is inversely proportional to the specified error threshold; hence the larger the error, the smaller the footprint.

The algorithm was created by computer scientists Rajeev Motwani and Gurmeet Singh Manku. It finds applications in computations where data takes the form of a continuous data stream instead of a finite data set, such as network traffic measurements, web server logs, and clickstreams.

Algorithm:

Step 1: Divide the incoming data stream into buckets of width $w = \frac{1}{\epsilon}$, where ϵ is mentioned by user as the error bound (along with the minimum support threshold = σ)

Step 2: Increment the frequency count of each item according to the new bucket values. After each bucket, decrement all counters by 1.

Step 3: Repeat – Update counters and after each bucket, decrement all counters by 1.

2. Sketching Algorithm

A sketching algorithm is a data compression technique that generates a smaller surrogate dataset to speed up numerical operations on large datasets. It uses random projections to compress the original dataset, making it amenable to statistical analysis. Sketching algorithms have been largely developed in the computer science community.

Sketching is a probabilistic data compression technique that has been largely developed in the computer science community. Numerical operations on big datasets can be intolerably slow; sketching algorithms address this issue by generating a smaller surrogate dataset. Typically, inference proceeds on the compressed dataset. Sketching algorithms generally use random projections to compress the original dataset and this stochastic generation process makes

them amenable to statistical analysis. We argue that the sketched data can be modelled as a random sample, thus placing this family of data compression methods firmly within an inferential framework. In particular, we focus on the Gaussian, Hadamard and Clarkson-Woodruff sketches, and their use in single pass sketching algorithms for linear regression with huge n. We explore the statistical properties of sketched regression algorithms and derive new distributional results for a large class of sketched estimators. A key result is a conditional central limit theorem for data oblivious sketches. An important finding is that the best choice of sketching algorithm in terms of mean square error is related to the signal to noise ratio in the source dataset. Finally, we demonstrate the theory and the limits of its applicability on two real datasets.

Example: Count-Min Sketch for Data Compression:

Step 1: Initialization:

- Choose the number of hash functions `k` and the number of counters per hash function `w`.
- Initialize a 2D array (matrix) `counters` of size `k x w` with all counters set to zero.

Step 2: Processing Data Stream:

- For each element in the data stream, hash it using each of the `k` hash functions to obtain `k` hash values.
- Increment the corresponding counters in the `counters` matrix at the positions determined by the hash values.

Step 3. Estimating Frequency:

- To estimate the frequency of an element:
  - Hash the element using each of the `k` hash functions.
  - Retrieve the minimum value among the counters at the corresponding positions in the `counters` matrix.
- The minimum value is an approximation of the element's frequency.

Step 4. Compression:

- Instead of storing the actual frequencies of elements, store the parameters of the Count-Min Sketch (hash functions, `k`, `w`, and the `counters` matrix).
- During decompression, reconstruct the Count-Min Sketch and use it to estimate frequencies.