Counting the number of distinct elements

Question 1 This is a network data streaming question, but it is designed in such a way that there is really no need (and of little extra help) to be familiar with the literature.

- (a) Elaborate on one example network application (preferably from your own research area) in which we need to know the total number of active TCP or UDP flows during a certain time window. The following paper provides some background information concerning the problem of estimating this quantity approximately:

  Cristian Estan, George Varghese, Mike Fisk: Bitmap algorithms for counting active flows on high speed links. Internet Measurement Conference 2003: 153-166

  Since counting it exactly using a hash table can be very expensive (in terms of space requirement) when the number of flows is large, we often have to settle with approximate counting using a network data streaming algorithm. In such an algorithm, each incoming packet will be looked at upon its arrival, and its "contribution" to the statistics we would like to estimate is credited to a synopsis data structure. Then we will never be able to see this packet again later on . In other words, such an algorithm typically takes only "one pass" on the data stream.

  Subproblems (b) and (c) will guide you through designing a network data streaming algorithm for estimating the number of active flows during a time window.

- (b) Let X_1, X_2, ..., X_n be iid random variables uniformly distributed in $[0, 1]$. Let Y be defined as $min\{X_1, X_2, ..., X_n\}$. Prove that its probability density function is $f_Y(y) = n(1-y)^{n-1}$ for $y \in [0, 1]$. Please also derive the expectation and variance of Y.

- (c) Based on the result of (b), design a data streaming algorithm that approximately estimates the number of active flows using only $o(N)$ space, where N is the number of active flows during a certain time window. Note also that the aforementioned naive hash table approach would require $\Omega(N)$ space. Your algorithm should produce reasonably accurate estimations, but you are NOT required to prove any performance guarantees. (Hint: think about "method of moments" in the statistical estimation theory and you may assume that multiple statistically independent hash functions exist, each of which hashes distinct flow labels into IID random values uniformly distributed in $[0, 1]$).

- (d) Association rule mining is the most fundamental problem in data mining. It can be described in the context of the following network application. Let us organize a set of web documents across the Internet as a 2D table. Each column corresponds to an English word (say "evil") and each row corresponds to a web document. The cell at the intersection of the i_{th} row and j_{th} column takes value 1 if the i_{th} document has the j_{th} word in it and takes value 0 otherwise. We would like to find a pair of words (for example "evil" and "sun") that occur frequently together in these documents, which is referred to as an association in the data mining literature. More precisely, if we view each column as a bit vector, we would like to find a pair of column vectors that if bitwise-ANDed together, will have significantly larger number of 1's than their bitwise-AND result should statistically have. For example, if the word "sun" is included in 1/2 of the documents and the word "evil" is included in 1/3 of the documents, then only approximately 1/6 of the documents should have both "sun" and "evil", if there is indeed no association between them. You will be asked to design an algorithm that will be used as a subroutine for finding **all** such associations in the 2D table T.

  Now building on your algorithm developed for (c), design an algorithm for (approximately) estimating the number of 1's in the bitwise-AND of any two column vectors. Your estimation algorithm should be much more efficient (in terms of execution time) than the naive algorithm (bitwise-AND two bit vectors together and count the number of 1's in it). Let us assume T

has $m = 5,000$ columns (i.e., 5000 different English words) and $n = 10$ billion rows (i.e., 10 billion different documents) when you are comparing the efficiency of two algorithms. Your algorithm should produce reasonably accurate estimations, but you are NOT required to prove any performance guarantees.