Algorithmic Foundations of Data Science: Assignment #5

Khaled Al Hosani (kah579), Myunggun Seo (ms9144)

New York University Abu Dhabi

March 4, 2018

Problem 1

$$\begin{aligned} f_x &= actual\ frequency\ of\ x \\ m &= stream\ length \\ \widehat{m} &= sum\ of\ k-1\ counters \\ \frac{m}{k} &= ideal\ threshold \end{aligned}$$

$m - \widehat{m}$ is exactly the number of elements in the stream that did not increment a counter but instead decremented others. Let's say this is the number of "destroyers".
The number of destructions, however, is different from the number of destroyers. The number of destroyed counts are: 1 from each of the k-1 counters that already existed, along with the 1 destroyer. This sums to k counts that are destroyed for every destroyer. Thus $\frac{m-\widehat{m}}{k}$ represents the number of destructions per each counter.
However, the key x that finally holds a counter may or may not have dealt all $\frac{m-\widehat{m}}{k}$ destructions. This is because if it was the only key that ever held that counter, it would have dealt all of the destructions; otherwise it would have dealt less than that. If it occurred only at the end of the stream, the key would have dealt 0 destructions.

Thus we can conclude that with maximum $\frac{m-\widehat{m}}{k}$ destructions and minimum 0 destructions, the resulting counter for key x is

$$f_x - \frac{m-\widehat{m}}{k} \leq \widehat{f_x} \leq f_x - 0$$

Problem 2

When the size of the stream is known we use the equation:

$$P(\widehat{f_x} - f_x > \frac{2m}{b}) \geq 2^{-l}$$

The probability of overcounting by more than $\frac{2m}{b}$ is greater than or equal to 2^{-l}.

We use it in order to calculate the values for b and l, which are the number of buckets per hash function and number of hash functions respectively.

First, we set $\frac{2m}{b}$ to $\frac{m}{2k}$ because that is the

$$\frac{2m}{b} = \frac{m}{2k}$$
$$b = 4k$$

We then set 2^{-l} to the probability we want 0.1%

$$2^{-l} = 0.001$$
$$l = \frac{3ln10}{ln2}$$

We then pass the stream through a count-min sketch and remember every element once it reaches an estimated frequency of at least $\frac{m}{k}$.

When we do not have the size m of the stream, we need to store the size m_i so far and keep a sorted array that holds elements and sorted by their count. Whenever we encounter an element, we pass it through the count-min sketch, and increase the estimated frequency, then look at its minimum count. If the count is $\geq \frac{m_i}{k}$, we add or update the element and its count into the sorted array.

Additionally, for each new element we remove the elements whose count has become $< \frac{m_i}{k}$. This is because as m_i increases, $\frac{m_i}{k}$ decreases, and some elements inserted into the array no longer meet the requirement. We can find these elements in $O(1)$ because whenever we update or add to the array, we remove all elements before the newly inserted element because they will all have counts $< \frac{m_i}{k}$.

Problem 3

This will still work and it will actually work better. In the good case that there were no overlap/conflicts at all, we increment all counters all the time and thus we would get the same results using the original and the modified algorithm.

In the worst case that there are lot of conflicts, the value of $\widehat{f_x} = min\{c_0, ..., c_l\}$ (where l is the number of counts and hash functions) can decrease with the modified algorithm but never decrease to a number less than the actual frequency f_x. The modified algorithm never increases $\widehat{f_x}$ either, since we only ever increment counters that would have been incremented in the original algorithm.

Thus the modified algorithm will produce $\widehat{f_x}$ such that the error $\widehat{f_x} - f_x$ is always less than or equal to that of the $\widehat{f_x}$ produced by the original algorithm.

Problem 4

In order to find the expected value of $X_k = min\{x_1, ..., x_k\}$, we define the cumulative distribution function:

$$\begin{aligned} F_{X_k} &= P(x > X_k) \\ &= 1 - P(X_k \geq x) \\ &= 1 - P(min\ of\ all\ x_i \geq x) \\ &= 1 - P(x_i \geq x)^k \\ &= 1 - (1-x)^k \end{aligned}$$

We take the derivative to find the probability density function.

$$\begin{aligned} f_{X_k} &= \frac{d}{dx} F_{X_k} \\ &= k(1-x)^{k-1} \end{aligned}$$

Now use the PDF to find the expectation.

$$\begin{aligned} E(X_k) &= \int_{-\infty}^{\infty} x f_{X_k} dx \\ &= k\Big[0 + \int_0^1 x(1-x)^{k-1} dx + 0\Big] \\ &= k\int_0^1 x(1-x)^{k-1} dx \end{aligned}$$

Substitute $1 - x = t$, $x = 1 - t$, and $dx = -dt$.

$$\begin{aligned} E(X_k) &= k\int_0^1 x(1-x)^{k-1} dx \\ &= k\int_0^1 (t^{k-1} - t^k) dt \\ &= \frac{1}{k+1} \end{aligned}$$

This relation is proven experimentally in the file '`min.py`'. We plotted $E(X_k)$ and $\frac{1}{1+k}$ for k in 1..100.