Assignment 3 - Homework Exercises on Streaming Algorithms

Duy Pham - 0980384
Maciej Wikowski - 0927420
Pattarawat Chormai - 0978675

October 23, 2015

Str.I-1

Suppose a deterministic ALG for ELEMENT UNIQUENESS uses at most s bits of storage. Then ALG can only be in 2^s states at any point in time, in particular after processing $m/2$ tokens.

On the other hand, the number of different frequency vector $F[1, ..., n]$ where $F[j] = 1$ if an item j exists in the stream otherwise $F[j] = 0$ for all $j \in [n]$, that can be generated by a stream of $m/2$ tokens from [n] is equal to the number of way we put $m/2$ balls into n bins :

$$\binom{n}{m/2} \geq \left(\frac{n}{m/2}\right)^{m/2} = 2^{m/2\log(2n/m)}$$

Hence, when $s < (m/2)\log(2n/m)$, there must be two sequences $\sigma_1 := \langle a_1, \dots, a_{m/2}\rangle$ and $\sigma'_1 := \langle a'_1, \dots, a'_{m/2}\rangle$ whose frequency vectors are different but ALG is in the same state after processing σ_1 as it would be after processing σ'_1. Now consider $\sigma_2 := \langle a_{m/2+1}, \dots, a_m\rangle$ which contains an item j in the stream. Such that all items in $\sigma_1 \circ \sigma_2$ are unique, while there is a duplicated item in $\sigma'_1 \circ \sigma_2$. Indeed such a stream exists if we take an item $j \in [n]$ for which $F_{\sigma_1}[j] = 0$ and $F_{\sigma'_1}[j] = 1$, and all items in σ_2 are distinct from σ_1 and σ'_1 except only j. Then j is unique in $\sigma_1 \circ \sigma_2$ but not in $\sigma'_1 \circ \sigma_2$.

Since ALG is deterministic and the state of ALG after processing σ_1 is the same as it would be after processing σ'_1, we can conclude that ALG will report the same answer for the $\sigma_1 \circ \sigma_2$ and $\sigma'_1 \circ \sigma_2$. In fact, this is not correct, because j is unique in $\sigma_1 \circ \sigma_2$ but not in $\sigma'_1 \circ \sigma_2$. Hence, any deterministic streaming algorithm that solves ELEMENT UNIQUENESS exactly must use at least $\Omega(m\log(2n/m)$ bits of storage.

Str.I-2

Algorithm 1 Find Two Missing Item

```
Require: A stream $\langle a_1, \dots, a_{n-2}\rangle$ where $a_i = (j, 1)$
  Compute $remainingSum \leftarrow n(n+1)/2$ and $remainingSumOfSquare \leftarrow n(n+1)(2n+1)/6$
  Process( $a_i$ ):
      $remainingSum := remainingSum - j$
      $remainingSumOfSquare := remainingSumOfSquare - j^2$
  Compute 2 missing items from $\frac{remainingSum \pm \sqrt{remainingSum^2 - 2(remainingSum^2 - remainingSumOfSquare)}}{2}$
  return the missing items
```

We will prove the correctness of the algorithm by analysing a quadratic equation degree 2.

$$(x+y)^2 = x^2 + 2xy + y^2$$

Let x, y denote 2 missing items in the stream and Δ_s, Δ_{ss} denote *remainingSum* and *remainingSumOfSquare* after the algorithm processes $n-2$ tokens. Thus

$$\begin{aligned}\Delta_s^2 &= \Delta_{ss} + 2xy \\ 2xy &= \Delta_s^2 - \Delta_{ss} \\ xy &= \frac{\Delta_s^2 - \Delta_{ss}}{2}\end{aligned}$$

Next, we first simplify the equation by denoting A as $(\Delta_s^2 - \Delta_{ss})/2$. We know that

$$\begin{aligned}x + y &= \Delta_s \\ x + A/x &= \Delta_s \\ x^2 + A &= \Delta_s x \\ x^2 - \Delta_s x + A &= 0 \\ x &= \frac{\Delta_s + \sqrt{\Delta_s^2 - 4A}}{2} \\ &= \frac{\Delta_s + \sqrt{\Delta_s^2 - 2(\Delta_s^2 - \Delta_{ss})}}{2}\end{aligned}$$

Thus

$$y = \frac{\Delta_s - \sqrt{\Delta_s^2 - 2(\Delta_s^2 - \Delta_{ss})}}{2}$$

Therefore, the algorithm always returns correct answer. Next, considering bits of storage, we see that the algorithm need to store only 2 values, namely *totalSum* and *totalSumOfSquare* and the maximum value is $O(n^3)$. Thus, the algorithm need $O(\log n)$ bits to store such values.

Str.I-3

Algorithm 2 ϵ-Frequent Items

```
Require: A stream $\langle a_1, \ldots, a_m \rangle$ in Cash Register Model where $a_i = (j, c)$
  Initialize $I \leftarrow \emptyset$
  Process( $a_i$ ):
  if $j \in I$ then
     $c(j) \leftarrow c(j) + c$
  else
     Insert $j$ into $I$ with counter $c(j) = c$
     if $|I| \geq 1/\epsilon$ then
        $MinFreq \leftarrow \min_{j \in I} c(j)$
        for all items $j \in I$ do
           $c(j) \leftarrow c(j) - MinFreq$ ; delete $j$ from $I$ when $c(j) = 0$
        end for
     end if
  end if
     return $I$
```

The algorithm in vanilla model can be adapted to cash-register model by initialise $c(j)$, a counter of j, with c or increase $c(j)$ by c if $j \in I$. Secondly, if there are too many items in I, $|I| \geq 1/\epsilon$, $c(j)$ for all j in I will be decreased by $\min\limits_{j \in I} c(j)$.

Next, let denote a token (j^*, c^*) as a token that is going to be processed next and j_{min} whose $c(j_{min}) = \min\limits_{j \in I} c(j)$ in that time. We will argue that the algorithm in cash-register model correctly computes a superset of the ϵ-frequent items by comparing its behaviours with the original algorithm in vanilla model in 2 conditions, namely when $c^* \geq minFreq$ and $c^* < minFreq$.

For $c^* \geq minFreq$, after the original algorithm processes a token $(j^*, 1)$ $c(j_{min})$ time, j_{min} will be removed from I and when it processes the rest of $(j^*, 1)$, $c(j^*)$ will equal to $c^* - c(j_{min})$. This is exactly the same as the modified algorithm does when (j^*, c^*) arrives, j_{min} will be removed since $c(j_{min})$ becomes zero after subtracted by itself and j^* remains in I with $c(j^*) := c(j^*) - c(j_{min})$.

For $c^* < minFreq$, after the original algorithm processes a token $(j^*, 1)$ c^* time, $c(j)$ for $j \in I$ will be subtracted by c^* and there is no j^* in I which is the same result from the modified algorithm does.

Therefore, we can conclude that the modified algorithm returns correct result.

Str.II-1

(i)

We know that $\mathrm{E}[\widetilde{\Phi}(\sigma)] = \Phi(\sigma)$ and $\mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right] = (1/3) \cdot \Phi(\sigma)$, thus:

$$\Pr[|\widetilde{\Phi}(\sigma) - \Phi(\sigma)| \geq c \cdot \Phi(\sigma)] = \Pr[|\widetilde{\Phi}(\sigma) - \mathrm{E}[\widetilde{\Phi}(\sigma)] \geq \ 3c \cdot \mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right]]$$

According to Chebyshev Inequality, we have:

$$\begin{aligned}
&\Pr[|\widetilde{\Phi}(\sigma) - \mathrm{E}[\widetilde{\Phi}(\sigma)|] \geq 3c \cdot \mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right]] \\
&= \Pr[|\widetilde{\Phi}(\sigma) - \mathrm{E}[\widetilde{\Phi}(\sigma)|] \geq 3c \cdot \sqrt{(}\,\mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right]) \cdot \sqrt{(}\,\mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right])] \\
&\leq \frac{1}{9 \cdot c^2 \cdot \mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right]}
\end{aligned}$$

To let 1/6 be the upper bound of this probability, we have to choose a value of c such that:

$$\begin{aligned}
9 \cdot c^2 \cdot \mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right] &= 6 \\
c &= \sqrt{(}\frac{6}{9\,\mathrm{Var}\left[\widetilde{\Phi}(\sigma)\right]}) \\
&= \sqrt{(}\frac{2}{3\frac{\Phi(\sigma)}{3}})
\end{aligned}$$

Since we know that $\Phi(\sigma) > 3$, then:

$$c = \sqrt{(}\frac{2}{\Phi(\sigma)})) \quad < \sqrt{(}\frac{2}{3})$$

This value of c statisfies the condition that $0 < c < 1$.

(ii)

The idea of the new algorithm is to run Alg k times, then taking the median of the k return values $\widetilde{\Phi}(\sigma)$ (the Median trick). Before modifying the algorithm formally, we make some analysis as follows.

Let X_i be an indicator random variable, which is defined as:

$$X_i = \begin{cases} 1 \text{ if} \Pr[|\widetilde{\Phi}(\sigma) - \Phi(\sigma)| \geq c \cdot \Phi(\sigma)] \\ 0 \text{ otherwise} \end{cases}$$

Let $X = \sum_{i=1}^{k} X_i$ be the random variable representing the final outcome of the new algorithm. We have:

$$\mathrm{E}[X] = \mathrm{E}[\sum_{i=1}^{k} X_i] = \sum_{i=1}^{k} \mathrm{E}[X_i]$$

In this algorithm, we used the specified value of c in part (i), which confirms that:

$$\Pr[|\widetilde{\Phi}(\sigma) - \Phi(\sigma)| \geq c \cdot \Phi(\sigma)] \leq 1/6$$

which implies that

$$\Pr[X_i = 1] \leq 1/6$$

Because X_i is an indicator random variable, then:

$$\mathrm{E}[X_i] = 1/6$$

Thus, we have:

$$\mathrm{E}[X] = \sum_{i=1}^{k} \mathrm{E}[X_i] = k/6$$

When $X_i = 1$, it implies that we do not have the event described in the problem statement. The probability that we do not have it after performing the Median trick is:

$$\begin{aligned} \Pr[X > \frac{k}{2}] &= \Pr[X > 3\,\mathrm{E}[X]] = \Pr[X > (1+2)\,\mathrm{E}[X]] \\ &\leq (\frac{e^2}{3^3})^{\frac{k}{6}} \\ &= (\frac{e^2}{27})^{\frac{k}{6}} \end{aligned}$$

Thus, the probability that we get the event described is $1 - (\frac{e^2}{27})^{\frac{k}{6}}$. To get the desired value $(1-\delta)$, we have to choose k such that:

$$\begin{aligned} \delta &\geq (\frac{e^2}{27})^{\frac{k}{6}} \\ \iff \frac{1}{\delta} &\leq (\frac{27}{e^2})^{\frac{k}{6}} \\ \iff \log\frac{1}{\delta} &\leq \frac{k}{6}\log\frac{27}{e^2} \\ \iff k &\geq \frac{6}{\log\frac{27}{e^2}} \cdot \log\frac{1}{\delta} \end{aligned}$$

We can choose

$$k = \lceil 4\log\frac{1}{\delta} \rceil$$

In conclusion, we modify the algorithm as represented in algorithm 3.

Algorithm 3 Revised Algorithm Taking ALG as Sub-Routine

```
Require: A stream $\sigma =< a_1, \cdots, a_m >, \delta$
Ensure: Statistics $\widehat{\Phi}(\sigma)$
Operation:
  Choose $k := \lceil 4 \log \frac{1}{\delta} \rceil$
  Init $J := \emptyset$
  for i from 1 to k do
    $\widetilde{\Phi}_i(\sigma) :=$ ALG ()
    $J = J \cup \widetilde{\Phi}_i(\sigma)$
  end for
      return The median of set $J$
```

The algorithm needs to store the set J of k elements, and each run of ALG stores $B(n, m)$ bits, so the total number of bits used by the algorithm is $O(kB(n, m))$.

Str.II-2

To approximate the MEDIAN of a stream without knowing m in advance, we modify the algorithm.

Algorithm 4 Estimate MEDIAN of a stream

```
Require: A stream $\langle a_1, \ldots, a_m \rangle$
  Initialize:
  Choose suitable integer $k \geq 1$ to obtain the desired success probability.
  Init $J := \emptyset$
  Process( $a_i$ ):
  for j from 1 to k do
    Generate a random number $r$ from $\{1, \cdots, i\}$
    if $r = 1$ then
      $J := J \cup a_i$
      if $|J| > k$ then
        Remove a random item from $J$
      end if
    end if
  end for
      return the median of set J
      return $I$
```

Similar to the algorithm in the course note, this algorithm needs to store the set J and k random numbers, which use $O(k \log(n + m)$ bits in total.

We define for each $j \in 1, \cdots, k$ the indicator random variables X_j and Y_j as:

$$X_j = \begin{cases} 1 \text{ if } rank(a_r) > \lfloor 3(i+1)/4 \rfloor \\ 0 \text{ otherwise} \end{cases}$$

$$Y_j = \begin{cases} 1 \text{ if } rank(a_r) < \lceil (i+1)/4 \rceil \\ 0 \text{ otherwise} \end{cases}$$

We also define $X = \Sigma_{j=3}^{k} X_j$ and $Y = \Sigma_{j=1}^{k} Y_j$. Now suppose the item we report is not in a (1/4)-approx median. Then its rank is either in the right-most quartile or the left-most quartile. In the former case, $X > k/2$. Similarly, in the latter case, $Y > k/2$.

Because r is chosen as random, then $\mathrm{E}[X_j] = \mathrm{E}[Y_j] = 1/4$. Then:

$$\mathrm{E}[X] = E\left[\Sigma_{j=1}^{k} X_j\right] = \Sigma_{j=1}^{k} \mathrm{E}[X_j] = k/4$$

Using the Chernoff bound, we get:

$$\Pr[X > k/2] = \Pr[X > 2E[X]] \leq \left(\frac{e}{4}\right)^{k/4}$$

Similarly,

$$\Pr[X > k/2] \leq \left(\frac{e}{4}\right)^{k/4}$$

. Hence,

$$\Pr[\text{The algo returns a (1/4)-approx median}] \geq 1 - 2\left(\frac{e}{4}\right)^{k/4}$$

We can get the desired success probability $(1 - \delta)$ by choosing k such that $(e/4)^{k/4} \leq \delta/2$. By solving it, we get $k = \lceil 8 \log(2/\delta) \rceil$.

Str.II-3

By choosing a suitable k, the algorithm performs as follows.

- Pass 1: Sampling items.

  Take k random integers from $1, \cdots, m$

  Collect all tokens at the positions we have just random sampled, into a set J.

  Sort the set J

- Pass 2: Calculate Ranks.

  Init a set C of counters.

  For each token a_i, find suitable a_j in J such that $a_{j-1} < a_i \leq a_j$.

  Increase $C[j]$. After pass 2, C is the set of ranks of the sampled items.
  From J, pick 2 consecutive items a_l and a_r such that $C[l] \leq (m+1)/2 \leq C[r]$.

- Pass 3: Get the median.

  Collect all items between a_l and a_r and sort into array A.

  Init $rank = C[l]$.

  Traverse each item in A, $rank+ = 1$ for each item.
  Stop when $rank = (m+1)/2$. Return $A[rank - C[l]]$.

The idea of the algorithm is to use random sampling to split the stream into bins. Then we pick the bin that contain the exact median (pass 2). Finally, we collect all data in that bin, and find the exact median by calculating ranks. Because in the last pass, we store all related values, so the algorithm always returns the exact median.

The storage used depends on the size of the $\{a_l, \cdots, a_r\}$ array.

Let X be the random variable indicating the distance between a_l and a_r, then we have $\mathrm{E}[X] = m/k$.

Markov inequality says: $\Pr[X \geq t \cdot \mathrm{E}[X]] \leq 1/t$. We want $1/t$ to be 0.05, then $t = 20$. So we choose $k = 20\sqrt{m}$ to have the number of storage of $O(\sqrt{m}\log n)$ with probatility at least $1 - 0.05 = 0.95$.

Str.III-1

In theorem 10.4, we prove the lower bound and the upper bound of the Count-Min sketch. To get the lower bound, we use the property of the Strict Turnstile model that $F_\sigma[j] \geq 0, \forall j \in [n]$, then $C[s, h_s(j)] \geq F_\sigma[j]$. In the general turnstile model, $F_\sigma[j]$ can be negative, so $C[s, h_s(j)]$ is not always greater than or equal to $F_\sigma[j]$. The lower bound is not $F_\sigma[j]$ anymore.

To get the upper bound, we use Markov Inequality. But it holds only if X is a non-negative random variable. In this case, $X = C[s, h_s(j)] - F_\sigma[j]$ can be negative, so the upper bound can also be different.

So in general, the proof of 10.4 does not work in the general turnstile model.

Str.III-2

According to the question we know that $\widetilde{F}_\sigma[j] > m/2 + F_\sigma[j]$. That means $h(j)$ must be equal to $h(j^*)$ where j^* is an item that occurs $m/2 + 1$ time. We also know that the probability that $h(j) = h(j^*)$ is $1/k$. Thus $h(j) \neq h(j^*)$ is $(k-1)/k$.

Then the probability that the hash value of other $m/2 - 1$ items is not $h(j^*)$ is :

$$\begin{aligned}
(\frac{k-1}{k})^{m/2-1} &< (1 - 0.99) \\
(\frac{9}{10})^{m/2-1} &< 0.01 \\
(m/2 - 1)\log(0.9) &< \log(0.01) \\
m &> 2\left(\frac{log(0.01)}{log(0.9)} + 1\right) \\
m &> 89.41 \\
&\geq 90
\end{aligned}$$

Therefore, one possible stream for this particular case is a stream size 90 that has an item j^* occurs 46 time and one of the other 44 tokens is an item j whose $h(j) = h(j^*)$.