Advanced Algorithms (2IMA10) – Solutions to Homework Exercises on Streaming Algorithms

Mark de Berg

This document gives solutions to some of the Homework Exercises for Advanced Algorithms (2IMA10). Sometimes not only the answer itself is given, but also the idea behind it. In your exam, you should only write the solution itself.

Exercise 8.2

Idea: Let's first try to solve a related but easier problem, namely the variant where only one item is missing. For that variant it is sufficient to maintain the sum of all items in the stream. Then you get the following equality for the missing item, j:

$$\text{(sum of all items in stream)} + j = \text{(sum of all items in universe)}$$

and so you can compute j. If there are two missing items, j and k, then you have two unknowns and so you need to somehow get two equations to determine j and k.

Solution: This problem can be solved exactly using only $O(\log n + \log m)$ bits, using an algorithm that computes two numbers: $s_1 = \sum_{i=0}^{m-1} a_i$ and $s_2 = \sum_{i=0}^{m-1} a_i^2$ (where $m = n-2$). Note that $s_1 < mn < n^2$ and $s_2 < mn^2 < n^3$. Hence, each of these variables needs $O(\log n)$ bits. It remains to show that we can uniquely recover the two missing items from s_1 and s_2.

Define $S_1 := \sum_{i=0}^{n-1} i$ (thus $S_1 = (n-1)n/2$) and $S_2 = \sum_{i=0}^{n-1} i^2$ (thus $S_2 = (n-1)n(2n-1)/6$). If j and k denote the missing items, then we get the set of equations

$$\begin{cases} j + k = S_1 - s_1 \\ j^2 + k^2 = S_2 - s_2 \end{cases}$$

This set of equations has a unique solution (up to exchanging the values of j and k), namely $\{j,k\} = \{\frac{1}{2}(A_1 + \sqrt{2A_2 - A_1}, \frac{1}{2}(A_1 - \sqrt{2A_2 - A_1}\}$, where $A_1 := S_1 - s_1$ and $A_2 := S_2 - s_2$.

Exercise 8.4

Define $\sigma := \langle 1,1,2,2,\ldots,j^*,j^*,0,0,\ldots,0\rangle$, where j^* is the largest integer smaller than $1/\varepsilon$ and where item 0 occurs $m - 2j^*$ times. If $m > 4j^*$, then item 0 is a majority item in σ. When item 0 occurs for the first time, all items $j \in \{1,2,\ldots,j^*\}$ are in I with $c(j) = 2$. When 0 arrives, the new algorithm inserts 0 into I with $c(0) = 1$, discover that $I = j^* + 1 > 1/\varepsilon$, and then decrease $c(0)$ (and subsequently remove item 0 from I) and keep $c(j) = 2$ for the other items $j \in I$. So after processing 0, the situation is exactly the same as before. Hence, all other arrivals of 0 are handled in the same way, so at the end of the stream $0 \notin I$, which is incorrect.

Exercise 8.5

We run two instances of Algorithm 8.1 in parallel, one generating a set I_1 with counters $c_1(\cdot)$, and one generating a set I_2 with counters $c_2(\cdot)$. Both instances use parameter $\varepsilon/2$, instead of ε, that is, the size of the sets I_1 and I_2 will be smaller than $2/\varepsilon$. Finally, we start the run for I_1 from scratch at items $a_{2W}, a_{4W}, \ldots$, and we start the run for I_2 from scratch at items $a_W, a_{3W}, \ldots$. Hence, at every point in time—except during the first W arrivals—one of the runs was restarted more than W and at most $2W$ items ago. The set of that run will be reported. A description of this algorithm in pseudo-code is as follows. (*Remark:* In the exam you would not need to give pseudo-code (unless this is explicitly asked for); the textual description above would be sufficient.)

Input:
A stream $\langle a_0, \ldots, a_{m-1} \rangle$ in the vanilla model, a window size W, a value $\varepsilon > 0$.

Initialize:
$I_1 \leftarrow \emptyset$; $I_2 \leftarrow \emptyset$.

Process(a_i)**:**

if $i \mod 2W = 0$ **then** set $I_1 \leftarrow \emptyset$ and set all counters $c_1(\cdot)$ to zero.
else
    if $i \mod W = 0$ **then** set $I_2 \leftarrow \emptyset$ and set all counters $c_2(\cdot)$ to zero.
    end if
end if
if $a_i \in I_1$ **then** $c_1(a_i) \leftarrow c_1(a_i) + 1$
else
    Insert a_i into I_1 with counter $c_1(a_i) = 1$
    if $|I_1| \geqslant 2/\varepsilon$ **then**
        for all items $j \in I_1$ **do**
            $c_1(j) \leftarrow c_1(j) - 1$; delete j from I_1 when $c_1(j) = 0$
        end for
    end if
end if
if $a_i \in I_2$ **then** $c_2(a_i) \leftarrow c_2(a_i) + 1$
else
    Insert a_i into I_2 with counter $c_2(a_i) = 1$
    if $|I_2| \geqslant 2/\varepsilon$ **then**
        for all items $j \in I_2$ **do**
            $c_2(j) \leftarrow c_2(j) - 1$; delete j from I_2 when $c_2(j) = 0$
        end for
    end if
end if

Output:
If $i \mod 2W \geqslant W$ then $I \leftarrow I_1$ else $I \leftarrow I_2$. Report I

Correctness: If $i < W$ then I_2 has not yet been restarted, so it has the properties stated in Theorem 8.3 for the current window. In particular, since we run the algorithm with parameter $\varepsilon/2$, the set I_2 contains all $(\varepsilon/2)$-frequent items in the window. Hence, I_2 (which is the set we report when $i < W$), also contains all ε-frequent items in the window.

Now consider the case $i \geqslant W$. We report $I = I_1$ if $i \mod 2W \geqslant W$ and $I = I_2$ otherwise. In either case, the reported set I was generated by a run that was restarted more than W items and at most $2W$ items earlier. Hence, the set I contains all $(\varepsilon/2)$-frequent items from the stream $\sigma^* = \langle a_{i-L+1}, a_{i-L+2}, \ldots, a_i \rangle$ of length L where $W < L \leqslant 2W$. Now let j be an item that is ε-frequent in the current window, that is, an item that occurs more than εW times among the last W items. Since this window is a part of σ', item j also occurs more than εW times in σ'. Hence, its frequency in σ' is more than $\varepsilon W \geqslant (\varepsilon/2)L$. This means j is present in I, and so I contains all ε-frequent items in the window.

Amount of storage: Each of the two runs uses $O((2/\varepsilon)\log(n+m)) = O((1/\varepsilon)\log(n+m))$ bits by Theorem 8.3, so the algorithm uses $((1/\varepsilon)\log(n+m))$ bits in total.

Exercise 8.6

Consider the set X of streams defined as follows. Let $\mathcal{S}$ be the set of all subsets of $m-1$ distinct elements from the set $\{3i : 0 \leqslant i < n/3\}$. (We can choose $m-1$ distinct items in this way since $m < n/6 < n/3$.) For each subset $S \in \mathcal{S}$ put a stream σ into X whose elements are exactly the items in S. Note that

$$|X| = |S| = \binom{n/3}{m-1} \geqslant \left(\frac{n/3}{m-1}\right)^{m-1} = 2^{(m-1)\log(n/3(m-1))}. \tag{1}$$

Now suppose a deterministic streaming algorithm Alg for Similar Items uses at most s bits of storage, so it can be in at most 2^s states. When $2^s < |X|$ there must be two different streams $\sigma_1 \in X$ and $\sigma_1' \in X$ such that Alg is in exactly the same state after processing σ_1 as it would be after processing σ_1'. Since σ_1 and σ_1' are different streams from X, each having distinct items, there must be an item $j \in \{3i : 0 \leqslant i < n/3\}$ such that $j \in \sigma_1$ and $j \notin \sigma_1'$. Now let $\sigma_2 := \langle j+1 \rangle$ if $j < n-1$ and $\sigma_2 := \langle j-1 \rangle$ if $j = n-1$. Note that $j \in [n]$, and $j \notin \sigma_1$ and $j \notin \sigma_1'$, so $\sigma_1 \circ \sigma_2$ and $\sigma_1' \circ \sigma_2$ are valid input streams.

The stream $\sigma_1 \circ \sigma_2$ has a pair of items a_i, a_j such that $|a_i - a_j| = 1$, namely the pair $j, j+1$ (or $j, j-1$ if $j = n-1$). The stream $\sigma_1' \circ \sigma_2$ cannot have such a pair: since σ_1' only contains multiples of 3, the only possible such pairs must involve σ_2, that is, the only possible such pairs are $j, j+1$ and $j+1, j+2$ (or $j-1, j$ and $j-2, j-1$ if $j = n-1$). But $j \notin \sigma_1'$ by definition of j, and $(j+1) \notin \sigma_1'$ and $(j-2) \notin \sigma_1'$ because $j+1$ and $j-2$ are not a multiples of 3. Hence, the correct answer for $\sigma_1 \circ \sigma_2$ is yes, while the correct answer for $\sigma_1' \circ \sigma_2$ is no. Since Alg is deterministic and the state of Alg after processing σ_1 is the same as it would be after processing σ_1', Alg will report the same answer for the stream $\sigma_1 \circ \sigma_2$ as it would for $\sigma_1' \circ \sigma_2$, so one of these answers is incorrect.

The conclusion is that if Alg uses s bits and $2^s < |X|$ it cannot be correct on all inputs. Hence, any deterministic streaming algorithm that solves Similar Items exactly must use at least $\log|X| = \Omega(m\log(n/m))$ bits of storage. (The fact that $\log|X| = \Omega(m\log(n/m))$ uses that $m < n/6$.)

Exercise 9.2

Idea: We need to adapt the computations in the proof of Lemma 9.6, to express the probability that the reported answer is a (1/10)-approximate median as a function of k. Then we need to pick k such that the probability is at least 0.95.

Solution: We first bound $\Pr[\text{reported answer is too large}]$. To this end we define for each $i \in \{1, \ldots, k\}$ a random variable X_i:

$$X_i := \begin{cases} 1 & \text{if } \mathit{rank}(a_{r_i}) > \lceil 6(m+1)/10 \rceil \\ 0 & \text{otherwise} \end{cases}$$

Note that $X := \sum_{i=1}^{k} X_i$ is the total number of answers that are too large. The median answer, which is the one we report, is too large when $X \geqslant k/2$. Since $\mathrm{E}[X_i] = 4/10$ we get

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} \mathrm{E}[X_i] = \sum_{i=1}^{k} \frac{4}{10} = 2k/5.$$

The Chernoff bound for Poisson trials thus gives

$$\Pr[\text{reported answer is too large}] = \Pr[X \geqslant k/2] = \Pr[X \geqslant (5/4) \cdot \mathrm{E}[X]] \leqslant \left(\frac{e^{1/4}}{(5/4)^{5/4}}\right)^{2k/5}$$

Observe that the probability that a single answer is too small is exactly the same as the probability that a single answer is too large, and that we report the median answer. Hence,

$$\Pr[\text{reported answer is too small}] = \Pr[\text{reported answer is large small}] \leqslant \left(\frac{e^{1/4}}{(5/4)^{5/4}}\right)^{2k/5},$$

and so

$$\Pr[\text{reported answer is good}] \geqslant 1 - 2\left(\frac{e^{1/4}}{(5/4)^{5/4}}\right)^{2k/5}.$$

To get a success probability of at least 0.95 we need $2\left(\frac{e^{1/4}}{(5/4)^{5/4}}\right)^{2k/5} \geqslant 0.05$. The smallest k for which this is satisfied is $k = \left\lceil (5/2) \cdot \frac{\log 0.05}{\log(e^{1/4}/(5/4)^{5/4})} \right\rceil = 319$.

Exercise 9.4

The "basic algorithm". which is run k times in parallel in Algorithm 9.1, reports an item from the stream uniformly at random. In other words, it reports each a_j with probability $1/m$. If we don't know m, we need to make sure that after processing a_i, we have selected an item, lets call it sample_i, uniformly at random from $a_1, \ldots, a_i$. This is done by the following algorithm:

Process(a_i)**:**

```
1: if RANDOM(1, i) = i then
2:     sample_i ← a_i
3: else
4:     sample_i ← sample_{i−1}
5: end if
```

Correctness: We will prove by induction on i that $\Pr[\mathit{sample}_i = a_j] = 1/i$ for all $1 \leqslant j \leqslant i$. For $i = 1$ this is obviously true since $\mathit{sample}_1 = a_1$. Now consider the case $i > 1$. We have $\Pr[\mathit{sample}_i = a_i] = 1/i$ since $\Pr[\textsc{Random}(1, i) = i] = 1/i$. For $1 \leqslant j < i$ we have

$$\Pr[\mathit{sample}_i = a_j] = \Pr[\textsc{Random}(1, i) \neq i] \cdot \Pr[\mathit{sample}_{i-1} = a_j] = \left(1 - \frac{1}{i}\right) \cdot \frac{1}{i-1} = \frac{1}{i}.$$

Hence, the basic algorithm selects an element from the current stream uniformly at random, as desired. The overall algorithm runs this basic algorithm k times in parallel. Since the basic algorithm does the same thing as the basic algorithm from the course notes (but without knowing m), Theorem 9.7 still holds.

Exercise 9.6

This is a nice exercise to do yourself, so I won't give away the solution :) Just one hint: Suppose that you first make a pass in which you select an item a_i from the stream, and then a second pass where you count the number of items smaller than a_i. After that you know whether the median is an item smaller than a_i, or a_i itself, or an item larger than a_i.

Exercise 9.9

(i) Note that

$$\Pr[\text{median is too large}] = 1 - \Pr[\text{median is good or too small}].$$

Let X_i be an indicator random variable where

$$X_i := \begin{cases} 1 & \text{if } \textsc{Alg}_i(\sigma) \leqslant 2D(\sigma) \\ 0 & \text{otherwise} \end{cases}$$

and let $X = \sum_{i=1}^{k} X_i$. Thus X is the number of answers that are good or too small. If the median of the k answers is good or too small then $X \geqslant k/2$. Since

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} \mathrm{E}[X_i] = \sum_{i=1}^{k} 3/10 = 3k/10,$$

we can use the Chernoff bound for Poisson trial to obtain

$$\Pr[X \geqslant k/2] = \Pr[X \geqslant (5/3) \cdot \mathrm{E}[X]] \leqslant \left(\frac{e^{2/3}}{(5/3)^{5/3}}\right)^{3k/10} < (0.84)^{3k/10}.$$

Hence,

$$\Pr[\text{median is too large}] > 1 - (0.84)^{3k/10},$$

so the larger k the larger the probability that the reported answer is too large, if we report the median answer.

(ii) *Idea:* Take the answer that is expected to be "in the middle of the good range". We expect $k/5$ answer to be too small and $k/10$ answers to be good, so we take the $(k/5 + (1/2) \cdot (k/10))$-smallest answer.

Solution: Take the t-th smallest answer, where $t = k/5 + (1/2) \cdot (k/10) = k/4$.

(iii) This can be computed in the same way as in the course notes; see also Exercise 9.2. You should go through the computations yourself! (You should be able to do this without looking at the Course Notes or the solution to Exercise 9.2.) The final answer would be:

$$\Pr[\text{reported answer is good}] \geqslant 1 - \left(\frac{e^{1/4}}{(5/4)^{5/4}}\right)^{k/5} - \left(\frac{e^{1/14}}{(15/14)^{15/14}}\right)^{7k/10}.$$

Remark. It is not clear if the choice $t = k/4$ is best possible. To analyze this, you can express the success probability as a function of t, and then try to find the t maximizing the probability.

(iv) Since the probability to give an answer that is too small is 0, we report the smallest of all k answers.

(iv) The smallest answer, which is the one we report, is too large when all k answers are too large. This happens with probability $(9/10)^k$. The probability that we report an answer that is too small is 0. Hence,

$$\Pr[\text{reported answer is good}] = 1 - (9/10)^k.$$

Exercise 10.1

This can be proved using the proof technique used for Theorem 8.1. See the solution to Exercise 8.6 for another example of how to use this technique.

Exercise 10.3

Suppose the sketch for σ_1 results in a table $C_1[0..t_1-1][0..k_1-1]$ and the sketch for σ_2 results in a table $C_2[0..t_2-1][0..k_2-1]$. To be able to combine these two sketches we need:

- the sketches must have been produced with the same parameter settings, that is, with the same values for ε and δ, so that $k_1 = k_2$ and $t_1 = t_2$;
- the sketches should have used the same hash functions, that is, for each s with $0 \leqslant s < t_1$, the two sketches should have used the same function h_s.

When these conditions are satisfied, the table $C[0..t_1-1][0..k_1-1]$ defined by $C[s,\ell] = C_1[s,\ell] + C_2[s,\ell]$ for all $0 \leqslant s < t_1$ and $0 \leqslant \ell < k_1 - 1$ is a valid Count-Min Sketch for $\sigma_1 \circ \sigma_2$.

Exercise 10.5

Consider a stream σ of length m, where $m < 2n$ and m is even, defined as

$$\sigma = \langle 1, \ldots, m/2-1, 0, 0, \ldots, 0\rangle.$$

Since $t = 1$ we use just a single hash function h_1. Hence, for any item $j \in \{1, \ldots, m/2-1\}$ with $h_1(j) = h_1(0)$, we have $\tilde{F}_\sigma[j] - F_\sigma[j] \geqslant F_\sigma[0] > m/2$. There is such an item j unless $h_1(j) \neq h_1(0)$ for all $j \in \{1, \ldots, m/2-1\}$. Since $\Pr[h_1(j) \neq h_1(0)] = 0.9$, we have

$$\Pr[h_1(j) \neq h_1(0) \text{ for all } j \in \{1, \ldots, m/2-1\}] = 0.9^{m/2-1}.$$

Since $\log 0.01 / \log 0.9 \approx 43.7$, this probability is less than 0.01 for $m \geqslant 90$.