Karl Bringmann and Vasileios Nakos **Summer 2020**

Sublinear Algorithms, Exercise Sheet 1

www.mpi-inf.mpg.de/departments/algorithms-complexity/teaching/summer20/sublinear-algorithms/

Total Points: **40** Due: Friday, **May 22**, 2020

*You are allowed to collaborate on the exercise sheets, but you have to write down a solution on your own, **using your own words**. Please indicate the names of your collaborators for each exercise you solve. Further, cite all external sources that you use (books, websites, research papers, etc.).*

You need to collect at least 50% of all points on exercise sheets to be admitted to the exam.

Exercise 1 — 10 points

Consider the following multiplicative version of the Chernoff bound:

Lemma (Chernoff Bound). *Let $X_1, \dots, X_n$ be independent random variables taking values in $\{0,1\}$, and let $X = \sum_i X_i$. Then for all $0 \le \varepsilon \le 1$ we have*

$$\mathbb{P}\left[|X - \mathbb{E}[X]| > \varepsilon\, \mathbb{E}[X]\right] < 2 \exp\left(-\frac{\varepsilon^2\, \mathbb{E}[X]}{3}\right).$$

Use this lemma to design an algorithm for the following problem: Given an array A of length n filled with zeros and ones, and given an integer k, the task is to check whether the number of ones in A approximately exceeds k. More precisely: If there are $< (1-\varepsilon)k$ ones you should output *"less"*, and if there are $> (1+\varepsilon)k$ ones you should output *"more"*; if neither is the case, we do not care about the output. Your algorithm is supposed to be correct with probability $1-\delta$ and you should access A in at most $O(n/k \cdot \varepsilon^{-2} \log \delta^{-1})$ positions.

Exercise 2 — 5 + 5 points

Let $X_1, \dots, X_n$ be 4-wise independent random variables with expectation $\mathbb{E}[X_i] = 0$ for all i. Prove the following statements:

1. $\mathbb{E}\left[\left(\sum_i X_i\right)^4\right] = \sum_i \mathbb{E}[X_i^4] + 6 \sum_{i<j} \mathbb{E}[X_i^2]\, \mathbb{E}[X_j^2]$.
2. $\mathbb{P}\left[\left|\sum_i X_i\right| \ge t\right] \le \frac{\mathbb{E}\left[(\sum_i X_i)^4\right]}{t^4}$.

Exercise 3 — 10 points

Recall the streaming problem of computing a median element from the previous exercise sheet. The following algorithm solves this problem exactly in the *three-pass* streaming model – that is, we are allowed to scan through the stream three times in the same order. We assume that the stream consists of m distinct elements over the universe $[n]$.

First pass: Sample and store every incoming element with probability $1/\sqrt{m}$. Let r denote the number of samples and write $a_1, \dots, a_r$ for the sampled elements in sorted order. We write $A_i = \{a_i, \dots, a_{i+1} - 1\}$ for $1 \le i \le r-1$ and set $A_0 = \{1, \dots, a_1 - 1\}$ and $A_r = \{a_r, \dots, n\}$.

Second pass: We maintain counters $c_0, \ldots, c_r$ to keep track of the number of stream elements falling into the intervals $A_0, \ldots, A_r$, respectively. After this pass, we compute the (unique) index k with $\sum_{i=0}^{k} c_i \le \lceil m/2 \rceil < \sum_{i=0}^{k+1} c_i$.

Third pass: Store all stream elements lying in the interval A_k. After this pass, sort these elements and report the $(\lceil m/2 \rceil - \sum_{i=0}^{k} c_i)$-th smallest such element.

Convince yourself that the algorithm indeed computes a median element. Then prove that with probability 0.99, the algorithm uses at most $O(\sqrt{m} \log n)$ bits of space.

Exercise 4 — 10 points

In the lecture, Morris' algorithm was presented as a solution for the approximate counting problem: It returns a $(1+\varepsilon)$-approximation with probability $\frac{2}{3}$ and uses $O(\varepsilon^{-2} \log\log(\varepsilon^{-1} n))$ bits of space. Modify Morris' algorithm and improve the space usage to $O(\log \varepsilon^{-1} + \log\log n)$ bits (which is best-possible).

Hint: As opposed to incrementing the counter X with probability 2^{-X}, increment with probability $(1+\gamma)^{-X}$ instead, for some reasonable $\gamma < 1$.