# The space complexity of approximating the frequency moments *

Noga Alon †  Yossi Matias ‡  Mario Szegedy §

**Abstract**

The frequency moments of a sequence containing $m_i$ elements of type $i$, for $1 \leq i \leq n$, are the numbers $F_k = \sum_{i=1}^{n} m_i^k$. We consider the space complexity of randomized algorithms that approximate the numbers $F_k$, when the elements of the sequence are given one by one and cannot be stored. Surprisingly, it turns out that the numbers $F_0, F_1$ and $F_2$ can be approximated in logarithmic space, whereas the approximation of $F_k$ for $k \geq 6$ requires $n^{\Omega(1)}$ space. Applications to data bases are mentioned as well.

## 1 Introduction

Let $A = (a_1, a_2, \ldots, a_m)$ be a sequence of elements, where each $a_i$ is a member of $N = \{1, 2, \ldots, n\}$. Let $m_i = |\{j : a_j = i\}|$ denote the number of occurrences of $i$ in the sequence $A$, and define, for each $k \geq 0$

$$F_k = \sum_{i=1}^{n} m_i^k .$$

In particular, $F_0$ is the number of distinct elements appearing in the sequence, $F_1$ $( = m)$ is the length of the sequence, and $F_2$ is the *repeat rate* or *Gini's index of homogeneity* needed in order to compute the *surprise index* of the sequence (see, e.g., [11]). It is also natural to define

$$F_\infty = \max_{1 \leq i \leq n} m_i .$$

The numbers $F_k$ are called the *frequency moments* of $A$ and provide useful statistics on the sequence.

The frequency moments of a data set represent important demographic information about the data, and are important features in the context of database applications. Indeed, Haas *et al* [13] claim that virtually all query optimization methods in relational and object-relational database systems require a means for assessing the number of distinct values of an attribute in a relation, i.e., the function $F_0$ for the sequence consisting of the relation attribute.

The frequency moments $F_k$ for $k \geq 2$ indicate the degree of *skew* of the data, which is of major consideration in many parallel database applications. Thus, for example, the degree of the skew may determine the selection of algorithms for data partitioning, as discussed by DeWitt *et al* [6] (see also references therein). In particular, $F_2$ is used by Ioannidis and Poosala [14] for error estimation in the context of estimating query result sizes and access plan costs. Their method is based on selecting appropriate histograms for a small number of values to approximate the frequency distribution of values in the attributes of relations. The selection involves joining a relation with itself; note that $F_2$ is the output size of such join.

recently Haas *et al* [13] considered sampling based algorithms for estimating $F_0$, and proposed a hybrid approach in which the algorithm is selected based on the degree of skew of the data, measured essentially by the function $F_2$.

Since skew information plays an important role for many applications, it may be beneficial to maintain estimates for frequency moments; and, most notably, for $F_2$. For efficiency purposes, the computation of estimates for frequency moments of a relation should preferably be done and updated as the records of the relation are inserted to the database. The general approach of maintaining views, such as distribution statistics, of the data has been well-studied as the problem of *incremental view maintenance* (cf. [10]).

Note that it is rather straightforward to maintain the

*To be presented in STOC '96.

†AT&T Bell Labs, Murray Hill, NJ 07974, USA and Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Email: noga@math.tau.ac.il. Research supported in part by a USA-Israel BSF grant and by the Fund for Basic Research administered by the Israel Academy of Sciences.

‡AT&T Bell Labs, Murray Hill, NJ 07974, USA. Email: matias@research.att.com.

§AT&T Bell Labs, Murray Hill, NJ 07974, USA. Email: ms@research.att.com.

(exact) frequency moments by maintaining a full histogram on the data, i.e., maintaining a counter $m_i$ for each data value $i \in \{1, 2, \ldots, n\}$, which requires memory of size $\Omega(n)$ (cf. [16]). However, it is important that the memory used for computing and maintaining the estimates be limited. Large memory requirements would require storing the data structures in external memory, which would imply an expensive overhead in access time and update time. The restriction on memory size is further emphasized by the observation that typically incoming data records will belong to different relations that are stored in the database; each relation requires its own separate data structure. Thus, the problem of computing or estimating the frequency moments in one pass under memory constraints arises naturally in the study of databases.

There are several known randomized algorithms that approximate some of the frequency moments $F_k$ using limited memory. For simplicity, let us consider first the problem of approximating these numbers up to some fixed constant factor, say with relative error that does not exceed 0.1, and with success probability of at least, say, 3/4, given that $m \leq n^{O(1)}$. (In the following sections we consider the general case, that is, the space complexity as a function of $n$, $m$, the relative error $\lambda$ and the error-probability $\epsilon$.) Morris [17] (see also [7], [12]) showed how to approximate $F_1$ (that is; how to design an approximate counter) using only $O(\lg \lg m)$ $(= \ O(\lg \lg n)\ )$ bits of memory. Flajolet and Martin [8] designed an algorithm for approximating $F_0$ using $O(\lg n)$ bits of memory. (Their analysis, however, is based on the assumption that explicit families of hash functions with very strong random properties are available.) Whang *et al* [19] considered the problem of approximating $F_0$ in the context of databases.

Here we obtain tight bounds for the minimum possible memory required to approximate the numbers $F_k$. We prove that for every $k > 0$, $F_k$ can be approximated randomly using at most $O(n^{1-1/k} \lg n)$ memory bits. We further show that for $k \geq 6$, any (randomized) approximation algorithm for $F_k$ requires at least $\Omega(n^{1-5/k})$ memory bits and any randomized approximating algorithm for $F_\infty$ requires $\Omega(n)$ space. Surprisingly, $F_2$ can be approximated (randomly) using only $O(\lg n)$ memory bits.

In addition we observe that a version of the Flajolet-Martin algorithm for approximating $F_0$ can be implemented and analyzed using very simple linear hash functions, and that (not surprisingly) the $O(\lg \lg n)$ and the $O(\lg n)$ bounds in the algorithms of [17] and [8] for estimating $F_1$ and $F_0$ respectively are tight.

We also make some comments concerning the space complexity of *deterministic* algorithms that approximate the frequency moments $F_k$ as well as on the space complexity of randomized or deterministic algorithms that compute those precisely.

The rest of this extended abstract is organized as follows. In Section 2 we describe our space-efficient randomized algorithms for approximating the frequency moments. The tools applied here include the known explicit constructions of small sample spaces which support a sequence of four-wise independent uniform binary random variables, and the analysis is based on Chebyshev's Inequality and a simple application of the Chernoff bound. In Section 3 we present our lower bounds which are mostly based on techniques from communication complexity. The final Section 4 contains some concluding remarks and open problems.

## 2 Space efficient randomized approximation algorithms

In this section we describe several space efficient randomized algorithms for approximating the frequency moments $F_k$. Note that each of these moments can be computed precisely and deterministically using $O(n \lg m)$ memory bits, by simply computing each of the numbers $m_i$ precisely. Using the method of [17] the space requirement can be slightly reduced, by approximating (randomly) each of the numbers $m_i$ instead of computing its precise value, thus getting a randomized algorithm that approximates the numbers $F_k$ using $O(n \lg \lg m)$ memory bits. We next show that one can do better.

### 2.1 Estimating $F_k$

The basic idea in our algorithm, as well as in the next randomized algorithm described in this section, is a very natural one. Trying to estimate $F_k$ we define a random variable that can be computed under the given space constraints, whose expected value is $F_k$, and whose variance is relatively small. The desired result can then be deduced from Chebyshev's Inequality.

**Theorem 2.1** *For every $k \geq 1$, every $\lambda > 0$ and every $\epsilon > 0$ there exists a randomized algorithm that computes, given a sequence $A = (a_1, \ldots, a_m)$ of members of $N = \{1, 2, \ldots, n\}$, in one pass and using*

$$O\left(\frac{k \lg (1/\epsilon)}{\lambda^2} n^{1-1/k} (\lg n + \lg m)\right)$$

*memory bits, a number $Y$ so that the probability that $Y$ deviates from $F_k$ by more than $\lambda F_k$ is at most $\epsilon$.*

**Proof.** Without trying to optimize our absolute constants, define $s_1 = \frac{8kn^{1-1/k}}{\lambda^2}$ and $s_2 = 2\lg(1/\epsilon)$. (To simplify the presentation we omit, from now on, all floor and ceiling signs whenever these are not essential). We first assume the length of the sequence $m$ is known in advance, and then comment on the required modifications if this is not the case.

The algorithm computes $s_2$ random variables $Y_1, \ldots, Y_{s_2}$ and outputs their median $Y$. Each $Y_i$ is the average of $s_1$ random variables $X_{ij} : 1 \leq j \leq s_1$, where the $X_{ij}$ are independent, identically distributed random variables. Each of the variables $X = X_{ij}$ is computed from the sequence in the same way, using $O(\lg n + \lg m)$ memory bits, as follows. Choose a random member $a_p$ of the sequence $A$, where the index $p$ is chosen randomly and uniformly among the numbers $1, 2, \ldots, m$. Suppose that $a_p = l$ ( $\in N = \{1, 2, \ldots, n\}$.) Let

$$r = |\{q : q \geq p, a_q = l\}| \ ( \geq 1)$$

be the number of occurrences of $l$ among the members of the sequence $A$ following $a_p$ (inclusive), and define

$$X = m(r^k - (r-1)^k) .$$

Note that in order to compute $X$ we only need to maintain the $\lg n$ bits representing $a_p = l$ and the $\lg m$ bits representing the number of occurrences of $l$.

The expected value $\mathrm{E}(X)$ of $X$ is, by definition,

$$\begin{aligned}
\mathrm{E}(X) &= \\
&\frac{m}{m} \Big[ \big(1^k + (2^k - 1^k) + \ldots + (m_1^k - (m_1 - 1)^k)\big) + \\
&\big(1^k + (2^k - 1^k) + \ldots + (m_2^k - (m_2 - 1)^k)\big) + \cdots + \\
&\big(1^k + (2^k - 1^k) + \ldots + (m_n^k - (m_n - 1)^k)\big) \Big] \\
&= \sum_{i=1}^{n} m_i^k = F_k.
\end{aligned}$$

To estimate the variance $\mathrm{Var}(X) = \mathrm{E}(X^2) - (\mathrm{E}(X))^2$ of $X$ we bound $\mathrm{E}(X^2)$;

$$\begin{aligned}
&E(X^2) = \\
&\frac{m^2}{m} \Big[\big(1^{2k} + (2^k - 1^k)^2 + \ldots + (m_1^k - (m_1 - 1)^k)^2\big) + \\
&\big(1^{2k} + (2^k - 1^k)^2 + \ldots + (m_2^k - (m_2 - 1)^k)^2\big) + \cdots + \\
&\big(1^{2k} + (2^k - 1^k)^2 + \ldots + (m_n^k - (m_n - 1)^k)^2\big)\Big] \\
&\leq m \Big[ \ \big(k1^{2k-1} + k2^{k-1}(2^k - 1^k) + \ldots + \\
&km_1^{k-1}(m_1^k - (m_1 - 1)^k)\big) + \big(k1^{2k-1} + k2^{k-1}(2^k - 1^k) \\
&+ \cdots + km_2^{k-1}(m_2^k - (m_2 - 1)^k)\big) + \ldots + \\
&\big(k1^{2k-1} + k2^{k-1}(2^k - 1^k) + \ldots + \\
&km_n^{k-1}(m_n^k - (m_n - 1)^k)\big)\Big] \\
&\leq m \left[ \ km_1^{2k-1} + km_2^{2k-1} + \ldots + km_n^{2k-1} \ \right] \\
&= kmF_{2k-1} = kF_1 F_{2k-1} ,
\end{aligned}$$

where the first ineq. is obtained from the following inequality which holds for any numbers $a > b > 0$:

$$\begin{aligned}
&a^k - b^k = (a-b)(a^{k-1} + a^{k-2}b + \cdots + ab^{k-2} + b^{k-1}) \\
&\leq (a-b)ka^{k-1} .
\end{aligned}$$

We need the following simple inequality:

**Fact:** For every $n$ positive reals $m_1, m_2 \ldots, m_n$

$$(\sum_{i=1}^{n} m_i)(\sum_{i=1}^{n} m_i^{2k-1}) \leq n^{1-1/k} (\sum_{i=1}^{k} m_i^k)^2 .$$

(Note that the sequence $m_1 = n^{1/k}, m_2 = \ldots = m_n = 1$ shows that this is tight, up to a constant factor.)

**Proof (of fact):** Put $M = max_{1 \leq i \leq n} m_i$, then $M^k \leq \sum_{i=1}^{n} m_i^k$ and hence

$$\begin{aligned}
(\sum_{i=1}^{n} m_i)(\sum_{i=1}^{n} m_i^{2k-1}) &\leq (\sum_{i=1}^{n} m_i)(M^{k-1} \sum_{i=1}^{n} m_i^k) \\
&\leq (\sum_{i=1}^{n} m_i)(\sum_{i=1}^{n} m_i^k)^{(k-1)/k} (\sum_{i=1}^{n} m_i^k) \\
&= (\sum_{i=1}^{n} m_i)(\sum_{i=1}^{n} m_i^k)^{(2k-1)/k} \\
&\leq n^{1-1/k} (\sum_{i=1}^{n} m_i^k)^{1/k} (\sum_{i=1}^{n} m_i^k)^{(2k-1)/k} \\
&= n^{1-1/k} (\sum_{i=1}^{n} m_i^k)^2 ,
\end{aligned}$$

where for the last inequality we use the fact that $\left(\sum_{i=1}^{n} m_i\right)/n \leq \left(\sum_{i=1}^{n} m_i^k/n\right)^{1/k}$. □

By the above fact, the definition of the random variables $Y_i$ and the computation above,

$$\mathrm{Var}(Y_i) = \leq \mathrm{E}(X^2)/s_1 \leq kF_1F_{2k-1}/s_1 \leq kn^{1-1/k}F_k^2/s_1 ,$$

whereas

$$\mathrm{E}(Y_i) = \mathrm{E}(X) = F_k .$$

Therefore, by Chebyshev's Inequality and by the definition of $s_1$, for every fixed $i$,

$$\mathrm{Prob}\left[|Y_i - F_k| > \lambda F_k\right] \leq \frac{\mathrm{Var}(Y_i)}{\lambda^2 F_k^2} \leq \frac{kn^{1-1/k}F_k^2}{s_1 \lambda^2 F_k^2} \leq \frac{1}{8} .$$

It follows that the probability that a single $Y_i$ deviates from $F_k$ by more than $\lambda F_k$ is at most $1/8$, and hence, by the standard estimate of Chernoff (cf., for example, [3] Appendix A), the probability that more than $s_2/2$ of the variables $Y_i$ deviate by more than $\lambda F_k$ from $F_k$ is at most $\epsilon$. In case this does not happen, the median $Y_i$ supplies a good estimate to the required quantity $F_k$, as needed.

It remains to show how the algorithm can be implemented in case $m$ is not known in advance. In this case, we start with $m = 1$ and choose the member $a_l$ of the sequence $A$ used in the computation of $X$ as $a_1$. If indeed $m = 1$, $r = 1$ and the process ends, else we update the value of $m$ to 2, replace $a_l$ by $a_2$ with probability $1/2$, and update the value of $r$ as needed. In general, after processing the first $m - 1$ elements of the sequence we have (for each variable $X_{ij}$) some value for $a_l$ and for $r$. When the next element $a_m$ arrives we replace $a_l$ by that element with probability $1/m$. In case of such a replacement, we update $r$ and define it to be 1. Else, $a_l$ stays as it is and $r$ increases by 1 in case $a_m = a_l$ and otherwise does not change. It is easy to check that for the implementation of the whole process, $O(\lg n + \lg m)$ memory bits for each $X_{ij}$ suffice. This completes the proof of the theorem. □

**Remark.** In case $m$ is much bigger than a polynomial in $n$, one can use the algorithm of [17] and approximate each number $r$ used in the computation of each $X_{ij}$ using only $O(\lg \lg m + \lg(1/\lambda))$ memory bits. Since storing the value of $a_l$ requires $\lg n$ additional bits this changes the space complexity to $O\left(\frac{k \lg(1/\epsilon)}{\lambda^2} n^{1-1/k}(\lg n + \lg \lg m + \lg \frac{1}{\lambda})\right)$.

### 2.2 Improved estimation for $F_2$

The second frequency moment, $F_2$, is of particular interest, since the repeat rate and the surprise index arise in various statistical applications. By the last theorem, $F_2$ can be approximated (for fixed positive $\lambda$ and $\epsilon$) using $O(\sqrt{n}(\lg n + \lg m))$ memory bits. In the following theorem we show that in fact a logarithmic number of bits suffices in this case.

**Theorem 2.2** *For every $\lambda > 0$ and $\epsilon > 0$ there exists a randomized algorithm that computes, given a sequence $A = (a_1, \ldots, a_m)$ of members of $N$, in one pass and using*

$$O\left(\frac{\lg(1/\epsilon)}{\lambda^2}(\lg n + \lg m)\right)$$

*memory bits, a number $Y$ so that the probability that $Y$ deviates from $F_2$ by more than $\lambda F_2$ is at most $\epsilon$. For fixed $\lambda$ and $\epsilon$, the algorithm can be implemented by performing, for each member of the sequence, a constant number of arithmetic and finite field operations on elements of $O(\lg n + \lg n)$ bits.*

**Proof.** Put $s_1 = \frac{16}{\lambda^2}$ and $s_2 = 2\lg(1/\epsilon)$. As in the previous algorithm, the output $Y$ of the present algorithm is the median of $s_2$ random variables $Y_1, Y_2, \ldots, Y_{s_2}$, each being the average of $s_1$ random variables $X_{ij} : 1 \leq j \leq s_1$, where the $X_{ij}$ are independent, identically distributed random variables. Each $X = X_{ij}$ is computed from the sequence in the same way, using $O(\lg n + \lg m)$ memory bits, as follows.

Fix an explicit set $V = \{v_1, \ldots, v_h\}$ of $h = O(n^2)$ vectors of length $n$ with $+1, -1$ entries, which are *four-wise independent*, that is, for every four distinct coordinates $1 \leq i_1 \leq \ldots \leq i_4 \leq n$ and every choice of $\epsilon_1, \ldots, \epsilon_4 \in \{-1, 1\}$ exactly a $(1/16)-$fraction of the vectors have $\epsilon_j$ in their coordinate number $i_j$ for $j = 1, \ldots, 4$. As described in [1] such sets (also known as *orthogonal arrays of strength* 4) can be constructed using the parity check matrices of BCH codes. To implement this construction we need an irreducible polynomial of degree $d$ over $GF(2)$, where $2^d$ is the smallest power of 2 greater than $n$. It is not difficult to find such a polynomial (using $O(\lg n)$ space), and once it is given it is possible to compute each coordinate of each $v_i$ in $O(\lg n)$ space, using a constant number of multiplications in the finite field $GF(2^d)$ and binary inner products of vectors of length $d$. To compute $X$ we choose a random vector $v_p = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n) \in V$, where $p$ is chosen uniformly between 1 and $h$. We then define $Z = \sum_{l=1}^{n} \epsilon_i m_i$. Note that $Z$ is a linear function of the numbers $m_i$, and can thus be computed in one pass from the sequence $A$, where during the process we only have to maintain the current value of the sum and to keep the value $p$ (since the bits of $v_p$ can be generated from $p$ in $O(\lg n)$ space). Therefore, $Z$ can be computed using only $O(\lg n + \lg m)$ bits. When the sequence terminates define $X = Z^2$.

As in the previous proof, we next compute the expectation and variance of $X$. Since the random variables $\epsilon_i$ are pairwise independent and $\mathrm{E}(\epsilon_i) = 0$ for all $i$,

$$\mathrm{E}(X) = \mathrm{E}\left((\sum_{i-1}^{n} \epsilon_i m_i)^2\right) = \sum_{i=1}^{n} m_i^2 \mathrm{E}(\epsilon_i^2) +$$

$$2 \sum_{1 \leq i < j \leq n} m_i m_j \mathrm{E}(\epsilon_i)\mathrm{E}(\epsilon_j) = \sum_{i=1}^{n} m_i^2 = F_2 \ .$$

Similarly, the fact that the variables $\epsilon_i$ are four-wise independent implies that

$$\mathrm{E}(X^2) = \sum_{i=1}^{n} m_i^4 + 6 \sum_{1 \leq i < j \leq j} m_i^2 m_j^2 \ .$$

It follows that

$$\mathrm{Var}(X) = \mathrm{E}(X^2) - (\mathrm{E}(X))^2 = 4 \sum_{1 \leq i < j \leq n} m_i^2 m_j^2 \leq 2F_2^2 \ .$$

Therefore, by Chebyshev's Inequality, for each fixed $i$, $1 \leq i \leq s_2$,

$$\mathrm{Prob}\left[|Y_i - F_2| > \lambda F_2\right] \leq \frac{\mathrm{Var}(Y_i)}{\lambda^2 F_2^2} \leq \frac{2F_2^2}{s_1 \lambda^2 F_2^2} = \frac{1}{8} \ .$$

The standard estimates of Chernoff now imply, as in the previous proof, that the probability that the median $Y$ of the numbers $Y_i$ deviates from $F_2$ by more than $\lambda F_2$ is at most $\epsilon$, completing the proof. □

**Remark.** The space complexity can be reduced for very large $m$ to $O\left(\frac{\lg(1/\epsilon)}{\lambda^2}\right.$ $(\lg n + \lg \lg m + \lg(1/\lambda))$ by applying the method of [17] to maintain the sum $Z$ with a sufficient accuracy. The easiest way to do so is to maintain approximations of the negative and positive parts of this sum using $O(\lg n + \lg \lg m + \lg(1/\lambda))$ bits for each, and use the analysis in [12] and Chebyshev's Inequality to show that this gives, with a sufficiently high probability, the required result. We omit the details.

### 2.3 Comments on the estimation of $F_0$

Flajolet and Martin [8] described a randomized algorithm for estimating $F_0$ using only $O(\lg n)$ memory bits, and analyzed its performance assuming one may use in the algorithm an explicit family of hash functions which exhibits some ideal random properties. Since we are not aware of the existence of such a family of hash functions we briefly describe here a slight modification of the algorithm of [8] and a simple analysis that shows that for this version it suffices to use a linear hash function. For simplicity we only describe here the problem of estimating $F_0$ up to an absolute multiplicative constant factor, with constant success probability. It is possible to improve the accuracy and the success probability of the algorithm by increasing the space it uses.

**Proposition 2.3** *For every $c > 2$ there exists an algorithm that, given a sequence $A$ of members of $N$, computes a number $Y$ using $O(\lg n)$ memory bits, such that the probability that the ratio between $Y$ and $F_0$ is not between $1/c$ and $c$ is at most $2/c$.*

**Proof.** Let $d$ be the smallest integer so that $2^d > n$, and consider the members of $N$ as elements of the finite field $F = GF(2^d)$, which are represented by binary vectors of length $d$. Let $a$ and $b$ be two random members of $F$, chosen uniformly and independently. When a member $a_i$ of the sequence $A$ appears, compute $z_i = a \cdot a_i + b$ , where the product and addition are computed in the field $F$. Thus $z_i$ is represented by a binary vector of length $d$. For any binary vector $z$, let $r(z)$ denote the largest $r$ so that the $r$ rightmost bits of $z$ are all 0 and put $r_i = r(z_i)$. Let $R$ be the maximum value of $r_i$, where the maximum is taken over all elements $a_i$ of the sequence $A$. The output of the algorithm is $Y = 2^R$. Note that in order to implement the algorithm we only have to keep (besides the $d = O(\lg n)$ bits representing an irreducible polynomial needed in order to perform operations in $F$) the $O(\lg n)$ bits representing $a$ and $b$ and maintain the $O(\lg \lg n)$ bits representing the current maximum $r_i$ value.

Suppose, now, that $F_0$ is the correct number of distinct elements that appear in the sequence $A$, and let us estimate the probability that $Y$ deviates considerably from $F_0$. The only two properties of the random mapping $f(x) = ax + b$ that maps each $a_i$ to $z_i$ we need is that for every fixed $a_i$, $z_i$ is uniformly distributed over $F$ (and hence the probability that $r(z_i) \geq r$ is precisely $1/2^r$), and that this mapping is pairwise independent. Thus, for every fixed distinct $a_i$ and $a_j$, the probability that $r(z_i) \geq r$ and $r(z_j) \geq r$ is precisely $1/2^{2r}$.

Fix an $r$. For each element $x \in N$ that appears at least once in the sequence $A$, let $W_x$ be the indicator random variable whose value is 1 iff $r(ax + b) \geq r$. Let $Z = Z_r = \sum W_x$, where $x$ ranges over all the $F_0$ elements $x$ that appear in the sequence $A$. By linearity of expectation and since the expectation of each $W_x$ is $1/2^r$, the expectation $\mathrm{E}(Z)$ of $Z$ is $F_0/2^r$. By pairwise independence, the variance of $Z$ is $F_0 \frac{1}{2^r}(1 - \frac{1}{2^r}) < F_0/2^r$. Therefore, by Markov's Inequality

$$\text{If } \; 2^r > cF_0 \text{ then } \mathrm{Prob}(Z_r > 0) < 1/c\,,$$

since $\mathrm{E}(Z_r) = F_0/2^r < 1/c$. Similarly, by Chebyshev's Inequality

$$\text{If } c2^r < F_0 \text{ then } \mathrm{Prob}(Z_r = 0) < 1/c\,,$$

since $\mathrm{Var}(Z_r) < F_0/2^r = \mathrm{E}(Z_r)$ and hence $\mathrm{Prob}(Z_r = 0) \leq \mathrm{Var}(Z_r)/(\mathrm{E}(Z_r)^2) < 1/\mathrm{E}(Z_r) = 2^r/F_0$. Since our algorithm outputs $Y = 2^R$, where $R$ is the maximum $r$ for which $Z_r > 0$, the two inequalities above show that the probability that the ratio between $Y$ and $F_0$ is not between $1/c$ and $c$ is smaller than $2/c$, as needed. □

## 3 Lower bounds

In this section we present our lower bounds for the space complexity of randomized algorithms that approximate the frequency moments $F_k$ and comment on the space required to compute these moments randomly but precisely or approximate them deterministically. Most of our lower bounds are obtained by reducing the problem to an appropriate communication complexity problem, where we can either use some existing results, or prove the required lower bounds by establishing those for the corresponding communication problem. The easiest result that illustrates the method is the proof that the randomized approximation of $F_\infty$ requires linear memory, presented in the next subsection. Before presenting this simple proof, let us recall some basic definitions and facts concerning the *$\epsilon$-error probabilistic communication complexity* $C_\epsilon(f)$ of a function $f : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$, introduced by Yao [20]. Consider two parties with unlimited computing power, that wish to compute the value of a Boolean function $f(x, y)$, where $x$ and $y$ are binary vectors of length $n$, the first party possesses $x$ and the second possesses $y$. To perform the computation, the parties are allowed to send messages to each other, and each of them can make random decisions as well. At the end of the communication they must output the correct value of $f(x, y)$ with probability at least $1 - \epsilon$ (for the worst possible $x$ and $y$). The complexity $C_\epsilon(f)$ is the expected number of bits communicated in the worst case (under the best protocol).

As shown by Yao [21] and extended by Babai, Frankl and Simon [4], $C_\epsilon(f)$ can be estimated by considering the related notion of the *$\epsilon$-error distributional communication complexity* $D_\epsilon(f|\mu)$ under a probability measure on the possible inputs $(x, y)$. Here the two parties must apply a deterministic protocol, and should output the correct value of $f(x, y)$ on all pairs $(x, y)$ besides a set of inputs whose $\mu$-measure does not exceed $\epsilon$. As shown in [21], [4], $C_\epsilon(f) \geq \frac{1}{2} D_{2\epsilon}(f|\mu)$ for all $f$, $\epsilon$ and $\mu$.

Let $DIS_n : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$ denote the Boolean function (called the *Disjointness function*) where $DIS_n(x, y)$ is 1 iff the subsets of $\{1, 2, \ldots, n\}$ whose characteristic vectors are $x$ and $y$ intersect. Several researchers studied the communication complexity of this function. Improving a result in [4], Kalyanasundaram and Schnitger [15] proved that for any fixed $\epsilon < 1/2$, $C_\epsilon(DIS_n) \geq \Omega(n)$. Razborov [18] exhibited a simple measure $\mu$ on the inputs of this function and showed that for this measure $D_\epsilon(DIS_n|\mu) \geq \Omega(n)$. Our lower bound for the space complexity of estimating $F_\infty$ follows easily from the result of [15]. The lower bound for the approximation of $F_k$ for fixed $k \geq 6$ is more complicated and requires an extension of the result of Razborov in [18].

### 3.1 The space complexity of approximating $F_\infty$

**Proposition 3.1** *Any randomized algorithm that outputs, given a sequence $A$ of at most $2n$ elements of $N = \{1, \ldots, n\}$ a number $Y$ such that the probability that $Y$ deviates from $F_\infty$ by at least $F_\infty/3$ is less than $\epsilon$, for some fixed $\epsilon < 1/2$, must use $\Omega(n)$ memory bits.*

**Proof.** Given an algorithm as above that uses $s$ memory bits, we describe a simple communication protocol for two parties possessing $x$ and $y$ respectively to compute $DIS_n(x, y)$, using only $s$ bits of communication. Let $|x|$ and $|y|$ denote the numbers of 1-entries of $x$ and $y$, respectively. Let $A$ be the sequence of length $|x| + |y|$ consisting of all members of the subset of $N$ whose characteristic vector is $x$ (arranged arbitrarily) followed by all members of the subset of $N$ whose characteristic vector is $y$.

The first party, knowing $x$, runs the approximation algorithm on the first $|x|$ members of $A$. It then sends the content of the memory to the second party which, knowing $y$, continues to run the algorithm for approximating $F_\infty$ on the rest of the sequence $A$. The second party then outputs "disjoint" (or 0) iff the output of the approximation algorithm is smaller than $4/3$; else it outputs 1. It is obvious that this is the correct value with probability at least $1 - \epsilon$, since the precise value of $F_\infty$ is 1 if the sets are disjoint, and otherwise it is 2.

The desired result thus follows from the theorem of [15] mentioned above. □

**Remark.** It is easy to see that the above lower bound holds even when $m$ is bigger than $2n$, since we may consider sequences in which every number in $N$ occurs either 0 or $m/n$ or $2m/n$ times. The method of the next subsection shows that the linear lower bound holds even if we wish to approximate the value of $F_\infty$ up to a factor of 100, say. It is not difficult to see that $\Omega(\lg \lg m)$ is also a lower bound for the space complexity of any randomized approximation algorithm for $F_\infty$ (simply because its final output must attain at least $\Omega(\lg m)$ distinct values with positive probability, as $m$ is not known in advance.) Thus $\Omega(n + \lg \lg m)$ is a lower bound for the space complexity of estimating $F_\infty$ for some fixed positive $\lambda$ and $\epsilon$. On the other hand, as mentioned in the previous section, all frequency moments (including $F_\infty$) can be approximated using $O(n \lg \lg m)$ bits.

Note that in the above lower bound proof we only need a lower bound for the one-way probabilistic communication complexity of the disjointness function, as in the protocol described above there is only one communication, from the first party to the second one. Since the lower bound of [15] holds for arbitrary communication we can deduce a space lower bound for the approximation of $F_\infty$ even if we allow algorithms that observe the whole sequence $A$ in its order a constant number of times.

### 3.2 The space complexity of approximating $F_k$

In this subsection we prove the following.

**Theorem 3.2** *For any fixed $k > 5$ and $\gamma < 1/2$, any randomized algorithm that outputs, given an input sequence $A$ of at most $n$ elements of $N = \{1, 2, \ldots, n\}$, a number $Z_k$ such that $\mathrm{Prob}(|Z_k - F_k| > 0.1 F_k) < \gamma$ uses at least $\Omega(n^{1-5/k})$ memory bits.*

We prove the above theorem by considering an appropriate communication game and by studying its complexity. The analysis of the game is similar to that of Razborov in [18], but requires several modifications and additional ideas. **Proof.** For positive integers $s$ and $t$, let $D(s, t)$ be the following communication game, played by $s$ players $P_1, P_2, \ldots, P_s$. Define $n = (2t - 1)s + 1$ and put $N = \{1, 2, \ldots, n\}$. The input of each player $P_i$ is a subset $A_i$ of cardinality $t$ of $N$ (also called a *$t$-subset* of $N$). Each player knows his own subset, but has no information on those of the others. An input sequence $(A_1, A_2, \ldots, A_s)$ is called *disjoint* if the sets $A_i$ are pairwise disjoint, and it is called *uniquely intersecting* if all the sets $A_i$ share a unique common element $x$ and the sets $A_i - \{x\}$ are pairwise disjoint. The objective of the game is to distinguish between these two types of inputs. To do so, the players can exchange messages according to any predetermined probabilistic protocol. At the end of the protocol the last player outputs a bit. The protocol is called *$\epsilon$-correct* if for any disjoint input sequence the probability that this bit is 0 is at least $1 - \epsilon$ and for any uniquely intersecting input sequence the probability that this bit is 1 is at least $1 - \epsilon$. (The value of the output bit for any other input sequence may be arbitrary). The *length* of the protocol is the maximum, over all possible input sequences $(A_1, \ldots, A_s)$, of the expected number of bits in the communication. In order to prove Theorem 3.2 we prove the following.

**Proposition 3.3** *For any fixed $\epsilon < 1/2$, and any $t \geq s^4$, the length of any randomized $\epsilon$-correct protocol for the communication problem $DIS(s, t)$ is at least $\Omega(t/s^3)$.*

By the simple argument of [21] and [4], in order to prove the last proposition it suffices to exhibit a distribution on the inputs and prove that any deterministic communication protocol between the players in which the total communication is less than $\Omega(t/s^3)$ bits produces an output bit that errs with probability $\Omega(1)$, where the last probability is computed over the input distribution. Define a distribution $\mu$ on the input sequences $(A_1, \ldots, A_s)$ as follows. Let

$P = I_1 \cup I_2 \cup \cdots \cup I_s \cup \{x\}$ be a random partition of $N$ into $s+1$ pairwise disjoint sets, where $|I_j| = 2t-1$ for each $1 \leq j \leq s, x \in N$ and $P$ is chosen uniformly among all partitions of $N$ with these parameters. For each $j$, let $\overline{A}_j$ be a random subset of cardinality $t$ of $I_j$. Finally, with probability $1/2$, define $A_j = \overline{A}_j$ for all $1 \leq j \leq s$, and with probability $1/2$, define $A_j = (I_j - \overline{A}_j) \cup \{x\}$ for all $j$. It is useful to observe that an alternative, equivalent definition is to choose the random partition $P$ as above, and then let each $A_j$ be a random subset of cardinality $t$ of $A_j \cup \{x\}$. If either none of the subsets $A_j$ contain $x$ or all of them contain $x$ we keep them as our input sets, and otherwise we discard them and repeat the random choice.

Note that the probability that the input sequence $(A_1, \ldots, A_s)$ generated under the above distribution is disjoint is precisely $1/2$, whereas the probability that it is uniquely intersecting is also $1/2$. Note also that $\mu$ gives each disjoint input sequence the same probability and each uniquely intersecting input sequence the same probability. Let $(A_1^0, \ldots, A_s^0)$ denote a random disjoint input sequence, and let $(A_1^1, \ldots, A_s^1)$ denote a random uniquely intersecting input sequence.

A *box* is a family $\overline{X}_1 \times \overline{X}_2 \times \cdots \times \overline{X}_s$, where each $\overline{X}_i$ is a set of $t$-subsets $N$. This is clearly a family of $s$-tuples of $t$-subsets of $N$. Standard (and simple) arguments imply that the set of all input sequences $(A_1, A_2, \ldots, A_s)$ corresponding to a fixed communication between the players forms a box. As we shall see later, this shows that the following lemma suffices to establish a lower bound on the average communication complexity of any deterministic $\epsilon$-correct protocol for the above game.

**Lemma 3.4** *There exists an absolute constant $c > 0$ such that for every box $\overline{X}_1 \times \overline{X}_2 \times \cdots \times \overline{X}_s$*

$$\mathrm{Prob}\left[(A_1^1, \ldots, A_s^1) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right] \geq \frac{1}{2e}\mathrm{Prob}\left[(A_1^0, \ldots, A_s^0) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right] - s2^{-ct/s^3}$$

To prove the lemma, fix a box $\overline{X}_1 \times \overline{X}_2 \times \cdots \times \overline{X}_s$. Recall that the distribution $\mu$ on the inputs has been defined by first choosing a random partition $P$. For such a partition $P$, let $\mathrm{Prob}_P[A_j \in \overline{X}_j]$ denote the conditional probability that $A_j$ lies in $\overline{X}_j$, given that the partition used in the random choice of the input sequence $(A_1, \ldots, A_s)$ is $P$. The conditional probabilities $\mathrm{Prob}_P[A_j^0 \in \overline{X}_j]$ and $\mathrm{Prob}_P[A_j^1 \in \overline{X}_j]$ are defined analogously. A partition $P = I_1 \cup I_2 \cup \cdots \cup A_s \cup \{x\}$ is called *$j$-bad*, where $j$ satisfies $1 \leq j \leq s$, if

$$\mathrm{Prob}_P[A_j^1 \in \overline{X}_j] < \left(1 - \frac{1}{s+1}\right)\mathrm{Prob}_P[A_j^0 \in \overline{X}_j] - 2^{-ct/s^3},$$

where $c > 0$ is a (small) absolute constant, to be chosen later. The partition is *bad* if it is $j$-bad for some $j$. If it is not bad, it is *good*.

We need the following two statements about good and bad partitions.

**Lemma 3.5** *There exists a choice for the constant $c > 0$ in the last inequality such that the following holds. For any set of $s-1$ pairwise disjoint $t$-subsets $I'_r \subset N$, $(1 \leq r \leq s, r \neq j)$, the conditional probability that the partition $P = I_1 \cup I_2 \cup \cdots \cup I_s \cup \{x\}$ is $j$-bad, given that $I_r = I'_r$ for all $r \neq j$, is at most $\frac{1}{20s}$.*

**Proof.** Note that since $I_r$ is known for all $r \neq j$, the union $I_j \cup \{x\}$ is known as well, and there are only $2t$ possibilities for the partition $P$. If the number of $t$-subsets of $I_j \cup \{x\}$ that belong to $\overline{X}_j$ is smaller than

$$\frac{1}{2}\binom{2t}{t}2^{-ct/s^3}$$

then for each of the $2t$ possible partitions $P$, $\mathrm{Prob}_P[A_j^0 \in \overline{X}_j] < 2^{-ct/s^3}$, implying that $P$ is not $j$-bad. Therefore, in this case the conditional probability we have to bound is zero and the assertion of the lemma holds. Consider, thus, the case that there are at least that many $t$-subsets of $I_j \cup \{x\}$ in $\overline{X}_j$, let $\mathcal{F}$ denote the family of all these $t$-subsets and put $I_j \cup \{x\} = \{x_1, x_2, \ldots, x_{2t}\}$. Let $p_i$ denote the fraction of members of $\mathcal{F}$ that contain $x_i$, and let $H(p) = -p\lg_2 p - (1-p)\lg_2(1-p)$ be the binary entropy function. By a standard entropy inequality (cf., e.g., [5]),

$$|\mathcal{F}| \leq 2^{\sum_{i=1}^{2t} H(p_i)}.$$

In order to determine the partition $P = I_1 \cup I_2 \cup \cdots \cup I_s \cup \{x\}$ we have to choose one of the elements $x_i$ as $x$. The crucial observation is that if the choice of $x_i$ as $x$ results in a $j$-bad partition $P$, then $p_i < (1 - \frac{1}{s+1})(1-p_i)$, implying that $H(p_i) \leq 1 - c'/s^2$ for some absolute positive constant $c'$. Let $b$ denote the number of elements $x_i$ whose choice as $x$ results in a $j$-bad partition $P$. By the above discussion

$$\frac{1}{2}\binom{2t}{t}2^{-ct/s^3} \leq |\mathcal{F}| \leq 2^{2t - bc'/s^2}.$$

This implies that if $t/s^3$ is much larger than $\lg t$, then $b \leq O(ct/s)$, and by choosing $c$ to be sufficiently small this upper bound for $b$ is smaller than $2t/(20s)$, completing the proof of the lemma. □

**Lemma 3.6** *If $P = I_1 \cup I_2 \cup \cdots \cup I_s \cup \{x\}$ is a good partition then*

$$\mathrm{Prob}_P\left[(A_1^1, A_2^1, \ldots, A_s^1) \in \overline{X}_1 \times \overline{X}_2 \times \cdots \times \overline{X}_s\right] \geq \frac{1}{e}\mathrm{Prob}_P\left[(A_1^0, \ldots, A_s^0) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right] - s2^{-ct/s^3}.$$

**Proof.** By the definition of a good partition

$$\mathrm{Prob}_P[A_j^1 \in \overline{X}_j] \geq (1 - \frac{1}{s+1})\mathrm{Prob}_P[A_j^0 \in \overline{X}_j] - 2^{-ct/s^3}$$

for every $j$, $1 \leq j \leq s$. Multiplying the above inequalities and using the definition of the distribution $\mu$ as well as the fact that $(1 - \frac{1}{s+1})^s > \frac{1}{e}$ the desired result follows. □

Returning to the proof of Lemma 3.4, let $\chi(P)$ be the indicator random variable whose value is 1 iff $P$ is a bad partition. Similarly, let $\chi_j(P)$ be the indicator random variable whose value is 1 iff $P$ is $j$-bad. Note that $\chi(P) \leq \sum_{j=1}^{s} \chi_j(P)$.

By computing the expectation over all partitions $P$

$$\begin{aligned}
&\mathrm{Prob}\left[(A_1^1,\ldots,A_s^1) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right] \\
&= \mathrm{E}\left(\mathrm{Prob}_P\left[(A_1^1,\ldots,A_s^1) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right]\right) \\
&\geq \mathrm{E}\left(\mathrm{Prob}_P\left[(A_1^1,\ldots,A_s^1) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right]\right. \\
&\quad \left.\cdot(1-\chi(P))\right) \geq \frac{1}{e}\mathrm{E}\left(\mathrm{Prob}_P\left[(A_1^0,\ldots,A_s^0) \in \right.\right. \\
&\quad \left.\left.\overline{X}_1 \times \cdots \times \overline{X}_s\right](1-\chi(P))\right) - s2^{-ct/s^3},
\end{aligned}$$

where the last inequality follows from Lemma 3.6.

It follows that in order to prove the assertion of Lemma 3.4 it suffices to show that for every $j$, $1 \leq j \leq s$,

$$\mathrm{E}\left(\mathrm{Prob}_P\left[(A_1^0,\ldots,A_s^0) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right]\chi_j(P)\right) \quad (1)$$

$$\leq \frac{1}{2s}\mathrm{E}\left(\mathrm{Prob}_P\left[(A_1^0,\ldots,A_s^0) \in \overline{X}_1 \times \cdots \times \overline{X}_s\right]\right). \quad (2)$$

Consider a fixed choice for the subsets $I_r$, $r \neq j$ in the definition of the partition $P = I_1 \cup I_2 \cup \cdots \cup I_s \cup \{x\}$. Given this choice, the union $U = I_j \cup \{x\}$ is known, but the actual element $x$ should still be chosen randomly in this union. Given the above information on $P$, the quantity (2) is

$$\frac{1}{2s}\prod_{r=1}^{s}\mathrm{Prob}_P[A_r^0 \in \overline{X}_r],$$

and each of these factors besides the one corresponding to $r = j$ is fixed. The same $s-1$ factors appear also in (1). The last factor in the above product, $\mathrm{Prob}_P[A_j^0 \in \overline{X}_j]$, is also easy to compute as follows. Let $l$ denote the number of $t$-subsets in $\overline{X}_j$ which are contained in $I_j \cup \{x\}$. Then $\mathrm{Prob}_P[A_j^0 \in \overline{X}_j]$ is precisely $l/\binom{2t}{t}$. Note, also, that for any choice of a member of $U$ as $x$, the probability that $A_j^0$ lies in $\overline{X}_j$ cannot exceed $l/\binom{2t-1}{t} = 2l/\binom{2t}{t}$. By Lemma 3.5, the probability that $\chi_j(P) = 1$ given the choice of $I_r$, $r \neq j$, is at most $1/(20s)$ and we thus conclude that

$$\begin{aligned}
&\mathrm{E}\left(\mathrm{Prob}_P\left[(A_1^0,A_2^0,\ldots,A_s^0) \in \overline{X}_1 \times \overline{X}_2 \times \cdots \times \overline{X}_s\right]\right. \\
&\quad \left.\chi_j(P)\right) \leq \frac{1}{10s}\mathrm{E}\left(\mathrm{Prob}_P\left[(A_1^0,\ldots,A_s^0) \in \right.\right. \\
&\quad \left.\left.\overline{X}_1 \times \cdots \times \overline{X}_s\right]\right),
\end{aligned}$$

implying the inequality in (1), (2) and completing the proof of Lemma 3.4. □

**Proof of Proposition 3.3.** Since it is possible to repeat the protocol and amplify the probabilities, it suffices to prove the assertion of the proposition for some fixed $\epsilon < 1/2$, and thus it suffices to show that any deterministic protocol whose length is smaller than $\Omega(t/s^3)$, applied to inputs generated according to the distribution $\mu$, errs with probability $\Omega(1)$. It is easy and well known that any fixed communication pattern corresponds to a box of inputs. Therefore, if the number of communication patterns in the end of which the protocol outputs 0 is smaller than $\frac{\rho}{s}2^{ct/s^3}$ then, by summing the assertion of Lemma 3.4 over all the boxes corresponding to such communication patterns, we conclude that the probability that the protocol outputs 0 on a random input $(A_1^1,\ldots,A_s^1)$ is at least $\frac{1}{2e}$ times the probability it outputs 0 on a random input $(A_1^0,\ldots,A_s^0)$ minus $\rho$. By choosing a sufficiently small absolute constant $\rho > 0$ this shows that in this case the algorithm must err with probability $\Omega(1)$. Thus, the number of communication patterns must be at least $\Omega(\frac{1}{s}2^{ct/s^3})$ and hence the number of bits in the communication must be at least $\Omega(t/s^3)$. □

**Proof of Theorem 3.2.** Fix an integer $k > 5$. Given a randomized algorithm for approximating the frequency moment $F_k$ for any sequence of at most $n$ members of $N = \{1,2,\ldots,n\}$, where $n = (2t-1)s+1$, using $M$ memory bits, we define a simple randomized protocol for the communication game $DIS(s,t)$ for $s = n^{1/k}$, $t = \Theta(n^{1-1/k})$. Let $A_1, A_2, \ldots, A_s$ be the inputs given to the players. The first player runs the algorithm on the $t$ elements of his set and communicates the content of the memory to the second player. The second player then continues to run the algorithm, starting from the memory configuration he received, on the elements of his set, and communicates the resulting content of the memory to the third one, and so on. The last player, player number $s$, obtains the output $Z_k$ of the algorithm. If it is at most $1.1st$ he reports that the input sequence $(A_1,\ldots,A_s)$ is disjoint. Else, he reports it is uniquely intersecting. Note that if the input sequence is disjoint, then the correct value of $F_k$ is $st$, whereas if it is uniquely intersecting the correct value of $F_k$ is $s^k + s(t-1) = n + s(t-1) > (3t-2)s = (\frac{3}{2}+o(1))n$. Therefore, if the algorithm outputs a good approximation to $F_k$ with probability at least $1-\gamma$, the protocol for $DIS(s,t)$ is $\gamma$-correct and its total communication is $(s-1)M < sM$. By Proposition 3.3 this implies that $sM \geq \Omega(t/s^3)$, showing that

$$M \geq \Omega(t/s^4) = \Omega(n/s^5) = \Omega(n^{1-5/k}).$$

This completes the proof. □

**Remark.** Since the lower bound in Proposition 3.3 holds for general protocols, and not only for one-way protocols in which every player communicates only once, the above lower bound for the space complexity of approximating $F_k$ holds even for algorithms that may read the sequence $A$ in its original order a constant number of times.

We show in the remainder of this section that the randomization and approximation are both required in the estimation of $F_k$ when using $o(n)$ memory bits.

### 3.3 Deterministic algorithms

It is obvious that given a sequence $A$, its length $F_1$ can be computed precisely and deterministically in logarithmic space. Here we show that for any nonnegative $k$ besides 1, even an *approximation* of $F_k$ up to, say, a relative error of 0.1 cannot be computed deterministically using less than a linear number of memory bits. This shows that the randomness is crucial in the two approximation algorithms described in Section 2. This is a simple corollary of the known results concerning the deterministic communication complexity of the equality function. Since, however, these known results are not difficult, we present a self contained proof, without any reference to communication complexity.

**Proposition 3.7** *For any nonnegative integer $k \neq 1$, any deterministic algorithm that outputs, given a sequence $A$ of $n/2$ elements of $N = \{1, 2, \ldots, n\}$, a number $Y$ such that $|Y - F_k| \leq 0.1F_k$ must use $\Omega(n)$ memory bits.*

**Proof.** Let $\mathcal{G}$ be a family of $t = 2^{\Omega(n)}$ subsets of $N$, each of cardinality $n/4$ so that any two distinct members of $\mathcal{G}$ have at most $n/8$ elements in common. (The existence of such a $\mathcal{G}$ follows from standard results in coding theory, and can be proved by a simple counting argument). Fix a deterministic algorithm that approximates $F_k$ for some fixed nonnegative $k \neq 1$. For every two members $G_1$ and $G_2$ of $\mathcal{G}$ let $A(G_1, G_2)$ be the sequence of length $n/2$ starting with the $n/4$ members of $G_1$ (in a sorted order) and ending with the set of $n/4$ members of $G_2$ (in a sorted order). When the algorithm runs, given a sequence of the form $A(G_1, G_2)$, the memory configuration after it reads the first $n/4$ elements of the sequence depends only on $G_1$. By the pigeonhole principle, if the memory has less than $\lg t$ bits, then there are two distinct sets $G_1$ and $G_2$ in $\mathcal{G}$, so that the content of the memory after reading the elements of $G_1$ is equal to that content after reading the elements of $G_2$. This means that the algorithm must give the same final output to the two sequences $A(G_1, G_1)$ and $A(G_2, G_1)$. This, however, contradicts the assumption, since for every $k \neq 1$, the values of $F_k$ for the two sequences above differ from each other considerably; for $A(G_1, G_1)$, $F_0 = n/4$ and $F_k = 2^k n/4$ for $k \geq 2$, whereas for $A(G_2, G_1)$, $F_0 \geq 3n/8$ and $F_k \leq n/4 + 2^k n/8$. Therefore, the answer of the algorithm makes a relative error that exceeds 0.1 for at least one of these two sequences. It follows that the space used by the algorithm must be at least $\lg t = \Omega(n)$, completing the proof. $\Box$

### 3.4 Randomized precise computation

As shown above, the randomness is essential in the two algorithms for approximating the frequency moments $F_k$, described in Section 2. In this subsection we observe that the fact that these are approximation algorithms is crucial as well, in the sense that the precise computation of these moments (for all $k$ but $k = 1$) requires linear space, even if we allow randomized algorithms.

**Proposition 3.8** *For any nonnegative integer $k \neq 1$, any randomized algorithm that outputs, given a sequence $A$ of at most $2n$ elements of $N = \{1, 2, \ldots, n\}$ a number $Y$ such that $Y = F_k$ with probability at least $1 - \epsilon$ for some fixed $\epsilon < 1/2$ must use $\Omega(n)$ memory bits.*

**Proof.** The reduction in the proof of Proposition 3.1 easily works here as well and proves the above assertion using the main result of [15]. $\Box$

## 4 Tight lower bounds for the approximation of $F_0, F_1, F_2$

The results in [17], [8] and those in Section 2 here show that logarithmic memory suffices to approximate randomly the frequency moments $F_0, F_1$ and $F_2$ of a sequence $A$ of at most $m$ terms up to a constant factor with some fixed small error probability. More precisely, $O(\lg \lg m)$ bits suffice for approximating $F_1$, $O(\lg n)$ bits suffice for estimating $F_0$ and $O(\lg n + \lg \lg m)$ bits suffice for approximating $F_2$, where the last statement follows from the remark following the proof of Theorem 2.2. It is not difficult to show that all these upper bounds are tight, up to a constant factor, as shown below.

**Proposition 4.1** *Let $A$ be a sequence of at most $m$ elements of $N = \{1, 2, \ldots, n\}$.*
*(i) Any randomized algorithm for approximating $F_0$ up to an additive error of $0.1F_0$ with probability at least $3/4$ must use at least $\Omega(\lg n)$ memory bits.*
*(ii) Any randomized algorithm for approximating $F_1$ up to $0.1F_1$ with probability at least $3/4$ must use at least $\Omega(\lg \lg m)$ memory bits.*
*(ii) Any randomized algorithm for approximating $F_2$ up to $0.1F_1$ with probability at least $3/4$ must use at least $\Omega(\lg n + \lg \lg m)$ memory bits.*

**Proof (sketch).**
(i) The result follows from the construction in the proof of Proposition 3.7, together with the well known fact that the randomized communication complexity of the equality function $f(x, y)$ whose value is 1 iff $x = y$, where $x$ and $y$ are $l$-bit numbers, is $\Theta(\lg l)$.
(ii) Since the length $F_1$ of the sequence can be any number up to $m$, the final content of the memory should admit at least $\Omega(\lg m)$ distinct values with positive probability, giving the desired result.
(iii) The required memory is at least $\Omega(\lg n)$ by the argument mentioned in the proof of part (i) and is at least $\Omega(\lg \lg m)$ by the argument mentioned in the proof of part (ii). $\Box$

## 5 Concluding remarks

We have seen that there are surprisingly space efficient randomized algorithms for approximating the first three frequency moments $F_0, F_1, F_2$, whereas not much space can be gained over the trivial algorithms in the approximation of $F_k$ for $k \geq 6$. We conjecture that an $n^{\Omega(1)}$ space lower bound holds for any $k$ (integer or non-integer), when $k > 2$. It would be interesting to determine or estimate the space complexity of the approximation of $\sum_{i=1}^{n} m_i^k$ for non-integral values of $k$ for $k < 2$, or the space complexity of estimating other functions of the numbers $m_i$. The method described in Section 2.1 can be applied in many cases and give some nontrivial space savings. Thus, for example, it is not too difficult to design a randomized algorithm based on the general scheme in Subsection 2.1, that approximates $\sum_{i=1}^{n} m_i \lg m_i$ up to some fixed small relative error with some small fixed error-probability, using $O(\lg n \lg m)$ memory bits. We omit the detailed description of this algorithm.

In a recent work [2] Alon *et al* presented an experimental study of the estimation algorithms for $F_2$. The experimental results demonstrate the practical utility of these algorithms. The algorithms are also extended to deal with the fully dynamic case, in which set items may be deleted as well. We finally remark that in practice, one may be able to obtain estimation algorithms which for typical data sets would be more efficient than the worst case performance implied by the lower bounds. Gibbons *et al* [9] recently presented an algorithm for maintaining an approximate list of the $k$ most popular items and their approximate counts (and hence also approximating $F_\infty$) using small memory, which works well for frequency distributions of practical interest.

**Acknowledgment**

We thank Colin Mallows for helpful comments regarding the statistics literature, and for pointing out [11].

## References

[1] N. Alon, L. Babai and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. Algorithms 7(1986), 567-583.

[2] N. Alon, P. Gibbons, Y. Matias and M. Szegedy, *Dynamic probabilistic maintenance of self-join sizes in limited storage*, manuscript, Feb. 1996.

[3] N. Alon and J. H. Spencer, *The Probabilistic Method*, John Wiley and Sons Inc., New York, 1992.

[4] L. Babai, P. Frankl and J. Simon, *Complexity classes in communication complexity theory*, Proc. of the $27^{th}$ IEEE FOCS, 1986, 337-347.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[6] D.J. DeWitt, J.F. Naughton, D.A. Schneider, and S. Seshadri, *Practical skew handling in parallel joins*, Proc. $18^{th}$ Int'l. Conf. Very Large Data Bases, 1992. pp. 27.

[7] P. Flajolet, *Approximate counting: a detailed analysis*, BIT 25 (1985), 113-134.

[8] P. Flajolet and G. N. Martin, *Probabilistic counting*, FOCS 1983, 76-82.

[9] P. Gibbons, Y. Matias and A. Witkowski, *Practical maintenance algorithms for high-biased histograms using probabilistic filtering*, Technical Report, AT&T Bell Laboratories, Murray Hill, NJ, Dec. 1995.

[10] A. Gupta and I.S. Mumick, *Maintenance of Materialized View: Problems, Techniques, and Applications*, IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing, 18(2), (1995).

[11] I. J. Good, *Surprise indexes and P-values*, J. Statistical Computation and Simulation 32 (1989), 90-92.

[12] M. Hofri and N. Kechris, *Probabilistic counting of a large number of events*, manuscript, 1995.

[13] P.J. Haas, J.F. Naughton, S. Seshadri, and L. Stokes, *Sampling-Based Estimation of the Number of Distinct Values of an Attribute*, Proc. of the $21^{st}$ VLDB Conf., 1995, 311-322.

[14] Y. E. Ioannidis and V. Poosala, *Balancing Histogram Optimality and Practicality for Query Result Size Estimation*, Proc. ACM-SIGMOD 1995.

[15] B. Kalyanasundaram and G. Schnitger, *The probabilistic communication complexity of set intersection*, $2^{nd}$ Structure in Complexity Theory Conf. (1987), 41-49.

[16] Y. Ling and W. Sun, *A supplement to sampling-based methods for query size estimation in a database system*, SIGMOD RECORD, 21(4) (1992), 12–15.

[17] R. Morris, *Counting large numbers of events in small registers*, CACM 21 (1978), 840-842.

[18] A. A. Razborov, *On the distributional complexity of disjointness*, Proc. of the ICALP (1990), 249-253. (To appear in Theoretical Computer Science.)

[19] K.-Y. Whang, B.T. Vander-Zanden, and H.M. Taylor, *A linear-time probabilistic counting algorithm for database applications*, ACM Transactions on Database Systems, 15(2) (1990), 208-229.

[20] A. C. Yao, *Some complexity questions related to distributed computing*, Proc of the $11^{th}$ ACM STOC, 1979, 209-213.

[21] A. C. Yao, *Lower bounds by probabilistic arguments*, Proc of the $24^{th}$ IEEE FOCS, 1983, 420-428.