Are bitvectors optimal?

H. Buhrman[1] P. B. Miltersen[2] J. Radhakrishnan[3] S. Venkatesh[4]

Abstract

We study the *static membership problem*: Given a set S of at most n keys drawn from a universe U of size m, store it so that queries of the form "Is u in S?" can be answered by making few accesses to the memory. We study schemes for this problem that use space close to the information theoretic lower bound of $\Omega(n \log(\frac{m}{n}))$ bits and yet answer queries by reading a small number of bits of the memory.

We show that for $\epsilon > 0$, there is a scheme that stores $O(\frac{n}{\epsilon^2} \log m)$ bits and answers membership queries using a randomized algorithm that reads just one bit of memory and errs with probability at most ϵ. We consider schemes that make no error for queries in S, but are allowed to err with probability at most ϵ for queries not in S. We show that there exist such schemes that store $O((\frac{n}{\epsilon})^2 \log m)$ bits and answer queries using just one bitprobe. If multiple probes are allowed, then the number of bits stored can be reduced to $O(n^{1+\delta} \log m)$ for any $\delta > 0$. The schemes mentioned above are based on probabilistic constructions of set systems with small intersections.

We show lower bounds that come close to our upper bounds (for a large range of n and ϵ): Schemes that answer queries with just one bitprobe and error probability ϵ must use $\Omega(\frac{n}{\epsilon \log(1/\epsilon)} \log m)$ bits of storage; if the error is restricted to queries not in S, then the scheme must use $\Omega(\frac{n^2}{\epsilon^2 \log(n/\epsilon)} \log m)$ bits of storage.

We also consider deterministic schemes for the static membership problem and show tradeoffs between space and the number of probes.

1 Introduction

In this paper, we study the *static membership* problem: Given a subset S of at most n keys from a universe $U = \{1, 2, \ldots, m\}$, store it so that queries of the form "Is u in S?" can be answered by making few accesses to the memory. This is a fundamental data structure problem with a long history. Yao [36] showed that if the data structure consists of a table with n cells where the keys are stored explicitly and the universe from which the set S is chosen is large enough, then the sorted table with binary search is optimal. In order to study data structures where elements of the set S are not stored explicitly, Yao (in the same paper) proposed the *cell*

[1]CWI, PO Box 94079, 1090 GB Amsterdam, The Netherlands, email: buhrman@cwi.nl.

[2]BRICS and Department of Computer Science, University of Aarhus, 8000 Aarhus C, Denmark, email: bromille@brics.dk.

[3]School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai 400005, India, email: jaikumar@tcs.tifr.res.in.

[4]Institute for Advanced Study, Olden Lane, Princeton 08540, USA, email:venkat@math.ias.edu. Most of this work was done while the author was at the Tata Institute of Fundamental Research, Mumbai 400005, India; part of this work was done while visiting BRICS at the University of Aarhus.

probe model. In this model, the set S is stored as a table of cells, each capable of holding one element of the universe; that is, if the universe has size m, where m is a power of two, then each cell holds $\log m$ bits. Queries are to be answered by probing the table adaptively; that is, each probe can depend on the results of earlier probes and the query element u. The goal is to process membership queries with as few probes as possible, and at the same time keep the size of the table small.

Fredman, Komlós and Szemerédi [15] gave a solution for the static membership problem in the cell probe model that used a constant number of probes and a table of size $O(n)$. We shall refer to this scheme as the FKS scheme. Note that if one is required to store sets of size at most n, then there is an information theoretic lower bound of $\lceil \log \sum_{i \le n} \binom{m}{i} \rceil$ on the number of bits used. For $n \le m^{1-\Omega(1)}$, this implies that the data structure must store $\Omega(n \log m)$ bits (and must, therefore, use $\Omega(n)$ cells). Thus, up to constant factors, the FKS scheme uses optimal space and number of cell probes. In fact, Fiat *et al.* [11], Brodnik and Munro [5] and Pagh [24] obtain schemes that use space (in bits) that is within a small additive term of $\lceil \log \sum_{i \le n} \binom{m}{i} \rceil$ and yet answer queries by reading at most a constant number of cells.

An important variation of the cell probe model is the *bitprobe* model, where each cell holds just a single bit, rather than an element of the universe. Thus, in this model, the query algorithm is given bitwise access to the data structure. Arguably, the bitprobe complexity of a data structure problem is a fundamental measure; this, in particular, applies to decision problems such as the membership problem, where the final answer to a query *is* a single bit. The bitprobe model is older than the cell probe model. The membership problem was studied in the bitprobe model already by Minsky and Papert in their 1969 book *Perceptrons* [23]. They were interested in *average*-case upper bounds for this problem, and did not study worst case bounds. Although the bitprobe complexity of several other static and dynamic data structure problems has been studied since then [8, 12, 13, 14, 21, 35], the bitprobe complexity of the static membership problem has received very little attention since the work of Minsky and Papert.[1] In this paper, we study *worst*-case bounds for the membership problem. Thus, our goal is to *answer queries using the minimum number of bitprobes*, and at the same time *keep the number of bits stored in the table small.*

1.1 Randomized schemes

We investigate the complexity of the static membership problem when the query processing algorithm tosses coins to decide which bits of the memory to read and is allowed to answer incorrectly with a certain small probability. Though using Las Vegas-style randomization to *construct* data structures is a well known and established technique (used, for instance, in many hashing based data structures, such as the FKS-scheme), Monte Carlo style randomization in the *query algorithm* has been used in the field of data structures only very recently [22, 18]. We consider two kinds of randomized schemes: (a) those that make one-sided errors, where the errors are restricted to negative instances alone (that is, these schemes never say 'No' when the query element u is in the set S); (b) schemes that are allowed to make two-sided errors (that

[1] The only exception we are aware of is a remark by Yao and Yao [35] stating without proof that if one ignores constant factors, the FKS scheme is optimal in the bitprobe model as well: that is, every scheme that uses $O(n \log m)$ bits of storage must use $\Omega(\log m)$ bitprobes (assuming $n \ll m$). For justification and generalization of this remark, see Theorem 6 of this paper.

is, errors are allowed for positive as well as negative instances). It is also possible to consider schemes that make errors on positive instances alone, but for the important case of one-probe schemes, we show that such schemes cannot do better than deterministic schemes.

1.1.1 Randomized schemes with two-sided error

Our main result says that there are randomized schemes that use *just one bitprobe*, and yet use space close to the information theoretic lower bound of $\Omega(n \log m)$ bits.

Theorem 1 *For any $0 < \epsilon \leq \frac{1}{4}$, there is a scheme for storing subsets S of size at most n of a universe of size m using $O(\frac{n}{\epsilon^2} \log m)$ bits so that any membership query "Is $u \in S$?" can be answered with error probability at most ϵ by a randomized algorithm which probes the memory at just one location determined by its coin tosses and the query element u.*

In contrast, deterministic schemes that answer queries using a single bitprobe need m bits of storage (see Theorem 11 of Section 4.1). By allowing randomization, we can reduce this bound (for constant ϵ) to $O(n \log m)$ bits. This is within a *constant factor* of the space used by a sorted table or a hash table; for $n \leq m^{1-\Omega(1)}$, it is within a constant factor of the *information theoretic* minimum number of bits needed to store the data. *Yet membership queries can be answered with small error probability by looking at a single bit of the data structure.* Note that we allow randomization only in the query algorithm; it is still the case that for each set S, there is exactly one associated data structure $\phi(S)$. Also, the probability of error is at most ϵ for *all* sets and *all* queries.

Many of the previous results for the membership problem have been based on hashing [15, 36, 35]. We depart from this tradition. The proof of Theorem 1 is based on two-colorings of set systems. The set systems we use are related to those considered by Erdős, Frankl and Füredi [9] in their study of r-cover-free family of sets. Similar set systems have been used to great advantage in the construction of pseudorandom generators and extractors, starting with the papers of Nisan [25] and Nisan and Wigderson [26] (for recent applications, see [34, 30, 29, 2]); we refer to them as NW-designs.

The properties of NW-designs are, unfortunately, not strong enough for our proof. So, we construct an appropriate set system ourselves. In Section 3, we describe this set system in more detail and relate it to the existence of a certain kind of strong *expander graphs.* Although we believe that such schemes can have practical uses, our proof relies on an existential argument, which we have not been able to make constructive. Subsequently, Ta-Shma [31], using recent developments in pseudorandomess [32, 33], has obtained an explicit one-probe randomized scheme with two-sided error ϵ for a certain range of n and ϵ. This scheme uses less space than the scheme in Theorem 9 (of Section 3.1).

Is the result of Theorem 1 the best possible? As remarked above, $\Omega(n \log m)$ bits of space is necessary if $n \leq m^{1-\Omega(1)}$. So, let us concentrate on the dependence of the size on the error probability ϵ. Unfortunately, our construction does not permit us to have sub-constant error probability and still use optimal space. We show that this limitation is unavoidable: if ϵ is made sub-constant, then we must use more than $n \log m$ space.

Theorem 2 *Suppose $\frac{n}{m^{1/3}} \leq \epsilon \leq \frac{1}{4}$. Then, any two-sided ϵ-error randomized scheme which answers queries using one bitprobe must use space $\Omega(\frac{n}{\epsilon \log(1/\epsilon)} \log m)$.*

Interestingly, the proof uses upper bounds as well as lower bounds for r-cover-free families [9].

1.1.2 Randomized schemes with one-sided error

As stated above, we do not use NW-designs in our proof of Theorem 1, but instead use a related set system. If we use NW-designs, then we don't get the same savings in space. However, we can now ensure that the error made while processing the query is one-sided.

Theorem 3 *For any $0 < \epsilon \leq \frac{1}{4}$, there is a scheme for storing subsets S of size at most n of a universe of size m using $O((\frac{n}{\epsilon})^2 \log m)$ bits so that any membership query "Is $u \in S$?" can be answered with error probability at most ϵ by a randomized algorithm which makes a single bitprobe to the data structure. Furthermore, if $u \in S$, the probability of error is 0.*

Note that the dependence on n is now quadratic, unlike in the two-sided scheme where it was linear. Though this scheme does not operate with optimal space, it still uses significantly less space than a bitvector. We also show that the scheme we have is essentially optimal: there is necessarily a quadratic dependence on $\frac{n}{\epsilon}$ for any scheme with one-sided error.

Theorem 4 *Suppose $\frac{n}{m^{1/3}} \leq \epsilon \leq \frac{1}{4}$. Consider the static membership problem for sets S of size at most n from a universe of size m. Then, any scheme with one-sided error ϵ that answers queries using at most one bitprobe must use $\Omega(\frac{n^2}{\epsilon^2 \log(n/\epsilon)} \log m)$ bits of storage.*

To prove this theorem, we again use the lower bounds for r-cover-free families.

Remarks

1. The proof of Theorem 3 is non-constructive. We also show that there is an explicit one-sided error randomized scheme that uses $O((\frac{n \log m}{\epsilon})^2)$ bits of storage and answers queries using one bitprobe. This result uses the explicit NW-designs (see Theorem 9 of Section 3.1).

2. One might also consider one-probe one-sided error schemes where no error is made for query elements *not* in the set S. In this case, we show (Theorem 11 of Section 4.1) that randomness does not help at all: such a scheme must use m bits of storage.

Thus, there is no one-sided error, one-probe scheme that uses optimal space. But the space requirement can be reduced if we allow more probes.

Theorem 5 *Suppose $0 < \delta < 1$. There is a randomized scheme with one-sided error $n^{-\delta}$ that solves the static membership problem using $O(n^{1+\delta} \log m)$ bits of storage and $O(\frac{1}{\delta})$ bitprobes.*

To prove this, we combine a two-sided scheme obtained from Theorem 1 with ideas for one-sided schemes from Theorem 3.

Connection with communication complexity. Theorems 1–4 can also be viewed in the communication complexity setting: Alice gets $u \in \{1, \ldots, m\}$, Bob gets $S \subseteq \{1, \ldots, m\}$ of size at most n, and Alice sends a single message to Bob after which Bob announces whether $u \in S$. Indeed, as was pointed out in [22], this communication game characterizes the data structure problem in the following way: If s is the optimal number of bits in the data structure that can be queried with one bitprobe and a particular bound on the error on positive and negative instances, then $\log s \pm o(\log s)$ is the number of bits sent from Alice to Bob in an optimal protocol for the communication problem with the same error bounds. Thus, Theorems 1–4 give bounds for the communication problem which are optimal within a low order term.

Connection with coding theory. The membership problem in the bitprobe model has an interesting coding theoretic interpretation: We are trying to give an encoding $\phi(u)$ of any m-bit string u with at most n 1's so that the length of the encoding is close to the *first order entropy* of u and so that any bit of u can be retrieved by looking at a few bits of $\phi(u)$. Thus, we are trying to construct a *locally decodable source code,* analogous to the locally decodable *channel* codes of [4, 20].

1.2 Deterministic schemes

As noted previously, the FKS hashing scheme is a data structure for storing sets of size at most n from a universe of size m using $O(n \log m)$ bits, so that membership queries can be answered using $O(\log m)$ bitprobes. We show that the FKS scheme makes an optimal number of bitprobes, within a constant factor, for this amount of space. This fact follows from the following general time-space tradeoff.

Theorem 6 *Suppose a deterministic scheme stores subsets of size n from a universe of size m using s bits of storage and answers membership queries with t bitprobes to memory. Then, $\binom{m}{n} \leq \max_{i \leq nt} \binom{2s}{i}$.*

Corollary 1.1 *Let $\epsilon > 0, c \geq 1$ be any constants. There is a constant $\delta > 0$ so that the following holds. Let $n \leq m^{1-\epsilon}$ and let a scheme for storing sets of size at most n of a universe of size m as data structures of at most $cn \log m$ bits be given. Then, any deterministic algorithm answering membership queries using this structure must make at least $\delta \log m$ bitprobes in the worst case.*

While the FKS scheme makes an optimal number of probes, the probes made are *adaptive.* In fact, adaptiveness seems to be quite inherent in hashing-based schemes. Somewhat surprisingly, as a corollary to the proof of Theorem 1, we can prove that there is a scheme that uses $O(n \log m)$ bits and answers membership queries with $O(\log m)$ *non-adaptive* bitprobes. Thus, adaptive probes do not help much when we consider deterministic schemes that use $O(n \log m)$ space.

More generally, from Theorem 6, it follows that any deterministic scheme that answers queries using t bitprobes must use space at least $ntm^{\Omega(1/t)}$ in the worst case. We show the existence of schemes which almost match the lower bound.

Theorem 7 1. *There is a non-adaptive scheme that stores sets of size at most n from a universe of size m using $O(ntm^{\frac{2}{t+1}})$ bits and answers queries using $2t+1$ bitprobes. This scheme is non-explicit.*

2. *There is an explicit adaptive scheme that stores sets of size at most n from a universe of size m using $O(m^{1/t} n \log m)$ bits and answers queries using $O(\log n + \log\log m) + t$ bitprobes.*

Thus, somewhat surprisingly, if we only care about space up to a polynomial, adaptive schemes are not more powerful than non-adaptive ones.

Finally, we turn our attention to deterministic *two probe* schemes and ask if they can do better than one probe schemes, where bitvectors are optimal. We have not been able to answer this question in general. We can show that this is the case, if the two bitprobes made are non-adaptive. Thus, the second bitprobe is useless for non-adaptive schemes. However, we show that there is a scheme with two adaptive bitprobes that does better than any scheme with two non-adaptive bitprobes for $n = 2$. We do not know whether a second adaptive probe helps for values of n greater than 2.

Theorem 8 *1. Any scheme for storing subsets S of size at most n ($n \geq 2$) of a universe of size m such that membership queries can be answered by two non-adaptive bitprobes uses space $s \geq m$ bits.*

2. Let $m \geq 2$ and let $n = 2$. Then there is a scheme with two adaptive bitprobes and space $s = O(m^{3/4})$.

1.3 Organization of the paper

We start with the formal definitions in the next section. In Section 3, randomized schemes with one-sided error and two-sided error are presented. In Section 4, we prove lower bound results for randomized schemes. We end with results on deterministic schemes in Section 5.

2 Notation and definitions

Notation: Unless mentioned explicitly, all logarithms in this paper are to the base 2. We use $[m]$ to denote the set $\{1, 2, \ldots, m\}$. For a set A, $\binom{A}{n}$ denotes the set of all subsets of A of size n, and $\binom{A}{\leq n}$ denotes the set of all its subsets of size at most n.

Definition 2.1 (Storing schemes) *An* (n, m, s)-storing scheme *is a method for representing any subset of size at most n of a universe of size m as an s-bit string. Formally, an (n, m, s)-storing scheme is a map ϕ from $\binom{[m]}{\leq n}$ to $\{0, 1\}^s$.*

Definition 2.2 (Deterministic query schemes) *A deterministic* (m, s, t)-query scheme *is a family $\{T_u\}_{u \in [m]}$ of m Boolean decision trees of depth at most t. Each internal node in a decision tree is marked with an index between 1 and s, indicating the address of a bit in an s-bit data structure. For each internal node, there is one outgoing edge labeled "0" and one labeled "1". The leaf nodes of every tree are marked 'Yes' or 'No'. Such a tree T_u induces a map from $\{0, 1\}^s$ to {Yes, No}; this map will also be referred to as T_u.*

Definition 2.3 (Deterministic schemes) *An (n, m, s)-storing scheme ϕ and an (m, s, t)-query scheme $\{T_u\}_{u \in [m]}$ together forms an* (n, m, s, t)-scheme *if $\forall S \in \binom{[m]}{\leq n}, \forall u \in [m] : T_u(\phi(S)) =$ Yes if and only if $u \in S$.*

In a non-adaptive scheme, the next probe to be made depends only on the input query q. It does not depend on the results of the previous probes.

Definition 2.4 (Non-adaptive query schemes) *A non-adaptive query scheme is a deterministic scheme where in each decision tree, all nodes on a particular level are marked with the same index between 1 and s (but nodes on the same level in different trees may be marked differently).*

In a randomized scheme, the storing scheme is deterministic as before. However, the query algorithm is allowed to make random coin tosses to decide the next location to be probed.

Definition 2.5 (Randomized schemes) *A randomized (m, s, t)-query scheme is a family $\{\pi_u\}_{u\in[m]}$ of probability distributions on the set of all Boolean decision trees of depth at most t. We answer the query "Is u in S?" by picking a decision tree according to the distribution π_u, and return the answer it gives. An (n, m, s)-storing scheme and a randomized (m, s, t)-query scheme together form an (n, m, s, t)-randomized scheme. We say that a randomized scheme has* positive one-sided error ϵ *if, for $u \notin S$, the error probability on queries "Is u in S?" is 0, i.e., the answer 'No' is always returned, while if $u \in S$, the error probability on the query "Is u in S?" is at most ϵ. Similarly, a randomized scheme has* negative one-sided error ϵ*, if, for $u \in S$, the error probability on queries "Is u in S?" is 0, i.e. the answer 'Yes' is always returned, while if $u \notin S$, the error probability on the query "Is u in S?" is at most ϵ. We say that a randomized scheme has* two-sided error ϵ *if on query "Is u in S?", the scheme returns the wrong answer with probability at most ϵ.*

We will be interested in one-probe randomized schemes where, in particular, π_u will be a probability distribution on Boolean decision trees that make at most one probe.

We say that a scheme is *explicit* if there are efficient algorithms that can simulate the storing scheme and the query scheme.

Definition 2.6 (Explicit storing schemes) *A family of storing schemes, indexed by (n, m, s), is* explicit *if there is a Turing machine, running in time $s^{O(1)}$, which given $S \subseteq \{0,1\}^m$ of size n, outputs the representation $\phi(S)$.*

Definition 2.7 (Explicit query schemes) *A family of (randomized) query schemes, indexed by (m, s, t), is* explicit *if there is a (probabilistic) Turing machine, running in time $(t+\log m)^{O(1)}$ which on input u and with oracle access to $\phi(S)$ executes the correct sequence of probes according to the query scheme and accepts or rejects accordingly.*

Definition 2.8 (Explicit schemes) *A family of schemes is explicit if the associated storing and query schemes are explicit.*

3 Upper bounds for randomized schemes

In this section, we show that there exist randomized one-probe schemes that use small space. We first describe the randomized scheme with one-sided error; the scheme with two-sided error can then be seen as a generalization. Randomized multi-probe schemes with one-sided error will be obtained by combining one-probe schemes with one-sided error and one-probe schemes with two-sided error.

All our schemes will be based on set systems with small intersections. In particular, we will use a set system of the form $\{\Gamma_u\}_{u\in[m]}$, where $\Gamma_u \subseteq [s]$. The query algorithm, on receiving the query "Is u in S?", will probe a location in Γ_u uniformly and answer 'Yes' if and only if it finds a 1 there.

We will describe our schemes using a bipartite graph with vertex sets U and V: U is the universe from which the set S to be stored is drawn and V is the set of locations in the memory.

We connect $u \in U$ to $v \in V$ if on query "Is u in S?" the cell v in the memory is probed by the algorithm (for some outcome of coin tosses). Thus, when the query element is u, the algorithm probes the memory locations in the neighborhood, $\Gamma(u)$, of u with uniform distribution.

3.1 Randomized scheme with one-sided error

Proof of Theorem 3: Our randomized scheme is based on a bipartite graph with vertex sets U and V, where $U = [m]$ and $|V| = O((\frac{n}{\epsilon})^2 \log m)$. Any instance of the data structure corresponds to a coloring of V using colors from the set $\{0, 1\}$. Hence, if the set $S \subseteq U$ is to be stored correctly, then all locations in $\bigcup_{u \in S} \Gamma(u)$ must be colored 1. This is because the algorithm is not allowed to say 'No' when the query element u is in S. Furthermore, since the error probability is at most ϵ, for all $u' \notin S$, at most $\epsilon|\Gamma(u')|$ locations in $\Gamma(u')$ can be colored 1. Thus, we get the following condition on the bipartite graph:

$$\forall S \in \binom{U}{\leq n} \forall u' \in U - S, \, |\Gamma(u') - \bigcup_{u \in S} \Gamma(u)| \geq (1 - \epsilon)|\Gamma(u')|. \tag{1}$$

We are thus required to pick neighborhoods for vertices $u \in U$ so that the resulting bipartite graph satisfies (1). An NW-design allows us to do this.

Definition 3.1 *A family of sets F is an (m, ℓ, a)-design if F has m sets each of size ℓ, and two different sets in F have at most a elements in common.*

Lemma 3.2 (Erdős, Frankl and Füredi [9, Theorem 2.1]) *If $m \leq \binom{s}{a} / \binom{\ell}{a}^2$, then there is an $(m, \ell, a-1)$-design all of whose elements are subsets of $[s]$.*

We will use an (m, ℓ, a)-design with $a = \lceil \log m \rceil$ and $\ell = \lceil na/\epsilon \rceil$. If $s = \lceil 2e^2\ell^2/a \rceil$ (which is $O((\frac{n}{\epsilon})^2 \log m)$, then

$$\frac{\binom{s}{a}}{\binom{\ell}{a}^2} \;\geq\; \frac{(\frac{s}{a})^a}{(\frac{e\ell}{a})^{2a}} \;\geq\; \frac{(\frac{2e^2\ell^2}{a^2})^a}{(\frac{e\ell}{a})^{2a}} \;\geq\; 2^a \;\geq\; m,$$

and by the above lemma, there is an (m, ℓ, a)-design (in fact, an $(m, \ell, a-1)$-design), all of whose elements are subsets of $[s]$. Let $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_m\}$ be such a design.

Storing scheme: Suppose the set to be stored is $\{u_1, \ldots, u_k\}, k \leq n$. Store a bitstring T of size s with 1's in all locations in $M = \Gamma_{u_1} \cup \Gamma_{u_2} \cup \ldots \cup \Gamma_{u_k}$ and 0 elsewhere.

Query scheme: On query "Is u in S?", do the following:

Step 1: Pick a random location i uniformly from Γ_u.

Step 2: If $T(i)$ is 1, say 'Yes', otherwise say 'No'.

Correctness: If $u \in S$, then every location in Γ_u has a 1 and hence we say 'Yes'. On the other hand, if $u \notin S$, then $|M \cap \Gamma_u| \leq na$. Hence, the probability that the algorithm says 'Yes' when we choose a random location in Γ_u is at most $\frac{na}{|\Gamma_u|} = \frac{na}{\ell} \leq \epsilon$.

■

The following theorem gives a slightly weaker bound than the one stated in Theorem 3 but it has the advantage that the scheme is explicit. It is based on the explicit version of Lemma 3.2 (see [9, Example 3.2] or [26]).

Theorem 9 *For any $\epsilon > 0$ and any n, m, there is an (n, m, s)-storing scheme with $s = O((\frac{n \log m}{\epsilon})^2)$ and an associated randomized one probe query scheme with a negative one-sided error at most ϵ. This scheme is explicit.*

Proof Let $\mathbf{F}$ be a finite field of size q, where q is the smallest power of two which is at least $(n \log m)/\epsilon$. If $d = \lceil \log m \rceil - 1$, then the number of univariate polynomials of degree at most d is $q^{d+1} \geq m$. We will associate with the element $u \in [m]$ a unique polynomial

$$p_u(X) \stackrel{\text{def}}{=} \sum_{i=1}^{\lceil \log m \rceil} u_i X^{i-1},$$

where u_i is the i-th bit in the binary representation of u. Now, we store S as a $q \times q$ bitmap, indexed by $\mathbf{F} \times \mathbf{F}$, with bit (x, y) on, if and only if $p_u(x) = y$ for some $u \in S$. The size of the data structure is as claimed. To answer the query "Is u in S?", we pick $x \in \mathbf{F}$ at random, and say 'Yes' if and only if bit $(x, p_u(x))$ of the bitmap is 1. Clearly, if $u \in S$, we always say 'Yes'. If $u \notin S$, then for all $u' \in S$, the graphs of p_u and $p_{u'}$ have at most d points in common. Thus, at most nd locations of the form $(x, p_u(x))$ of the bitmap will contain a 1. Note that $nd/q \leq \epsilon$. ■

Remark: By choosing parameters more carefully in the above proof, we can reduce the space requirement in the above theorem to $O(\frac{n \log m}{\epsilon \log((n \log m)/\epsilon)})^2$.

3.2 Randomized scheme with two-sided error

We now present a scheme that uses space $O(\frac{n}{\epsilon^2} \log m)$ and answers queries using a single bit probe, making an error with probability at most ϵ. The space needed depends linearly on n, and when ϵ is a constant, it is within a constant factor of a sorted table or a hash table.

Proof of Theorem 1: We may assume that m is large, say $m \geq 100$. We first describe the main ideas of our randomized scheme using the the bipartite graph $G = (U, V, E)$, where $U = [m]$ and $V = [s]$. On query "Is u in S?", the query algorithm probes a random location in $\Gamma(u)$, and says 'Yes' if and only if the location probed contains 1. Now, suppose we need to store the set $S \subseteq U$ ($|S| \leq n$). Then, we need a coloring $\chi_S : V \to \{0, 1\}$, such that for all $u \in S$, at least $(1-\epsilon)|\Gamma(u)|$ elements of $\Gamma(u)$ are colored 1, and for all $u' \notin S$, at least $(1-\epsilon)|\Gamma(u')|$ elements of $\Gamma(u')$ are colored 0. Thus, we need to find neighborhoods for the vertices so that the system of sets $\{\Gamma(u)\}_{u \in U}$ admits such a coloring χ_S for all $S \in \binom{U}{\leq n}$. This motivates the following definition.

Definition 3.3 *Let $\mathcal{C}_1, \mathcal{C}_0 \subseteq 2^V$. We say that $(\mathcal{C}_1, \mathcal{C}_0)$ is ϵ-two-colorable if there exists $\chi : V \to \{0, 1\}$ such that:*

1. $\forall T \in \mathcal{C}_1, |\chi^{-1}(0) \cap T| \leq \epsilon |T|$; *and*

2. $\forall T \in \mathcal{C}_0, |\chi^{-1}(1) \cap T| \leq \epsilon |T|$.

We say that $\mathcal{C} \subseteq 2^V$ *is* (n, ϵ)*-two-colorable if* $(\mathcal{C}_1, \mathcal{C}_0 = \mathcal{C} - \mathcal{C}_1)$ *is* ϵ*-two-colorable for all* $\mathcal{C}_1 \in \binom{\mathcal{C}}{\leq n}$.

In this terminology, our goal can be stated as follows: find a bipartite graph $G = (U, V, E)$, with $U = [m]$ and $V = [s]$, such that $\{\Gamma(u)\}_{u \in U}$ is an (n, ϵ)-two-colorable collection of m distinct non-empty sets. We will show that such a graph exists with $|V| = O(\frac{n}{\epsilon^2} \log m)$. For this, we first identify a sufficient condition, which we call *the* (n, ϵ)*-intersection property*, for a collection to be (n, ϵ)-two-colorable. We then observe that if G has a certain *expansion property* then $\{\Gamma(u)\}_{u \in U}$ has the (n, ϵ)-intersection property. Finally, using a probabilistic argument, we show that graphs with the required expansion property exist. To outline the main steps of our proof, we show the following implications, and the existence of an (m, s, n, d, ϵ)-expander with $s = O(\frac{n}{\epsilon^2} \log m)$.

$$\boxed{G = (U, V, E) \text{ is an } (m, s, n, d, \epsilon)\text{-expander}} \ \textit{Def. 3.8}$$

$$\Big\Downarrow \textit{Lemma 3.9}$$

$$\boxed{\{\Gamma(u)\}_{u \in U} \text{ has the } (n, \epsilon)\text{-intersection property}} \ \textit{Def. 3.5}$$

$$\Big\Downarrow \textit{Lemma 3.6}$$

$$\boxed{\{\Gamma(u)\}_{u \in U} \text{ is } (n, \epsilon)\text{-two-colorable}} \ \textit{Def. 3.3}$$

$$\Big\Downarrow \textit{Claim 3.4}$$

$$\boxed{(n, m, s, 1)\text{-randomized scheme with error } \epsilon} \ \textit{Def. 2.5}$$

$\blacksquare$

Colorable families and randomized schemes:

Lemma 3.4 *Suppose the bipartite graph* $G = (U, V, E)$*, with* $U = [m]$ *and* $V = [s]$*, is such that* $\{\Gamma(u)\}_{u \in U}$ *is an* (n, ϵ)*-two-colorable collection of* m *non-empty sets. Then there is a randomized scheme for the membership problem that uses* s *bits to store sets* $S \in \binom{U}{\leq n}$*, and answers membership queries using one bitprobe, and with error probability at most* ϵ.

Proof

The storing scheme: To store the set $S \in \binom{U}{\leq n}$ consider the collections $\mathcal{C}_1 = \{\Gamma(u)\}_{u \in S}$ and $\mathcal{C}_0 = \{\Gamma(u)\}_{u \notin S}$. Since $\{\Gamma(u)\}_{u \in U}$ is an (n, ϵ)-two-colorable collection, $(\mathcal{C}_1, \mathcal{C}_0)$ is ϵ-two-colorable. Let $\chi : V \to \{0, 1\}$ be a coloring satisfying the two conditions of Def. 3.3. We store χ as a table of $s = |V|$ bits.

The query scheme: Given a query "Is u in S?", pick i uniformly at random from $\Gamma(u)$, and return 'Yes' if $\chi(i) = 1$, and return 'No' if $\chi(i) = 0$.

Correctness: If $u \in S$, then $\Gamma(u) \in \mathcal{C}_1$ and $|\chi^{-1}(0) \cap \Gamma(u)| \le \epsilon|\Gamma(u)|$. Thus, $\Pr[\chi(i) = 0] \le \epsilon$. Similarly, if $u \notin S$, $\Gamma(u) \in \mathcal{C}_0$ and $\Pr[\chi(i) = 1] \le \epsilon$.

□

Intersection property and coloring: Suppose we wish to ϵ-two-color $(\mathcal{C}_1, \mathcal{C}_0)$. The first thing to try would be to color all the elements in $\bigcup_{S\in\mathcal{C}_1} S$ with 1 and the rest with 0. Unfortunately, some sets in $\mathcal{C}_0$ might get more than an ϵ fraction of their elements colored 1, and our coloring might not be proper. So, we need to first identify those sets in $\mathcal{C}_0$ that might be badly colored in this method: these are precisely the sets that have large intersection with the union of the sets in $\mathcal{C}_1$. Let $\mathcal{C}'_0$ be the collection of these sets. We will now modify our coloring method to pay special attention to sets in $\mathcal{C}'_0$. For now, ignore the sets in $\mathcal{C}_0 - \mathcal{C}'_0$, for they are at no risk of being badly colored while ensuring that the sets in $\mathcal{C}_1$ are properly colored. So, we are left with the problem of ϵ-two-coloring $(\mathcal{C}_1, \mathcal{C}'_0)$. If $|\mathcal{C}'_0| < |\mathcal{C}_1|$, this is a smaller problem, and we can use induction for this. This motivates the following definition of the ϵ-intersection property. That this definition is sufficient for ϵ-two-colorability is the main observation of this section, which we state formally in Lemma 3.6 below.

Definition 3.5 *Suppose $\mathcal{C}_1$ and $\mathcal{C}_0$ are collections of sets. We say that $(\mathcal{C}_1, \mathcal{C}_0)$ has the ϵ-intersection property if*

$$\forall \mathcal{C}'_1 \subseteq \binom{\mathcal{C}_1}{\le n} (\mathcal{C}'_1 \neq \emptyset), |\{S \in \mathcal{C}_0 : |T \cap (\bigcup_{T' \in \mathcal{C}'_1} T')| > \epsilon|T|\}| \quad < \quad |\mathcal{C}'_1|; \text{ and} \tag{2}$$

$$\forall \mathcal{C}'_0 \subseteq \binom{\mathcal{C}_0}{\le n} (\mathcal{C}'_0 \neq \emptyset), |\{T \in \mathcal{C}_1 : |T \cap (\bigcup_{T' \in \mathcal{C}'_0} T')| > \epsilon|T|\}| \quad < \quad |\mathcal{C}'_0|. \tag{3}$$

We say that a collection of sets $\mathcal{C}$ has the (n, ϵ)-intersection property if $(\mathcal{C}_1, \mathcal{C}_0 = \mathcal{C} - \mathcal{C}_1)$ has the ϵ-intersection property for all $\mathcal{C}_1 \in \binom{\mathcal{C}}{\le n}$.

Lemma 3.6 *Suppose $\mathcal{C}_1, \mathcal{C}_0 \subseteq 2^V$. If $|\mathcal{C}_1| \le n$ and $(\mathcal{C}_1, \mathcal{C}_0)$ has the (n, ϵ)-intersection property, then $(\mathcal{C}_1, \mathcal{C}_0)$ is ϵ-two-colorable.*

Corollary 3.7 *If $\mathcal{C}$ has the (n, ϵ)-intersection property, then $\mathcal{C}$ is (n, ϵ)-two-colorable.*

Proof (of Lemma 3.6) We use induction on $|\mathcal{C}_1| + |\mathcal{C}_0|$. The base case, when either $\mathcal{C}_1$ or $\mathcal{C}_0$ is empty, is obvious. For the induction step, consider a pair $(\mathcal{C}_1, \mathcal{C}_0)$ of non-empty collections, with $|\mathcal{C}_1| \le n$, that has the (n, ϵ)-intersection property. We may assume that $|\mathcal{C}_1| \le |\mathcal{C}_0|$; otherwise, interchange the roles of 0 and 1. Let

$$\mathcal{C}'_0 \stackrel{\text{def}}{=} \{T \in \mathcal{C}_0 : |T \cap (\bigcup_{T' \in \mathcal{C}_1} T')| > \epsilon|T|\}.$$

Since $(\mathcal{C}_1, \mathcal{C}_0)$ has the (n,ϵ)-intersection property and $1 \le |\mathcal{C}_1| \le n$, $|\mathcal{C}_0'| < |\mathcal{C}_1| \le |\mathcal{C}_0|$. By induction, there exists a two-coloring $\chi : V \to \{0,1\}$ such that

$$\begin{aligned} \forall T \in \mathcal{C}_1, |\chi^{-1}(0) \cap T| &\le \epsilon |T|; \text{ and} \qquad (4) \\ \forall T \in \mathcal{C}_0', |\chi^{-1}(1) \cap T| &\le \epsilon |T|. \qquad (5) \end{aligned}$$

We may assume that if $\chi(v) = 1$ then $v \in \bigcup_{T \in \mathcal{C}_1} T$ (otherwise, change $\chi(v)$ to 0—this cannot hurt (4) and can only help (5)), that is,

$$\chi^{-1}(1) \subseteq \bigcup_{T \in \mathcal{C}_1} T. \qquad (6)$$

We claim that χ is also an ϵ-two-coloring of $(\mathcal{C}_1, \mathcal{C}_0)$. For, if $T \in \mathcal{C}_1$, this follows from (4), if $T \in \mathcal{C}_0'$, then it follows from (5), and if $T \in \mathcal{C}_0 - \mathcal{C}_0'$, then

$$|\chi^{-1}(1) \cap T| \le |T \cap \bigcup_{T' \in \mathcal{C}_1} T'| \le \epsilon |T|,$$

where the first inequality follows from (6) and the second from the definition of $\mathcal{C}_0'$. $\square$

Expanders and the intersection property: We now relate the (n,ϵ)-intersection property of $\{\Gamma(u)\}_{u \in U}$ to a certain expansion property of $G = (U, V, E)$.

Definition 3.8 *We say that $G = (U, V, E)$ is an (m, s, n, d, ϵ)-expander if $U = [m]$, $V = [s]$, each vertex in U has degree d and the following condition holds:*

$$\forall S \in \binom{U}{\le 2n}, \ |\Gamma(S)| \ge (1 - \frac{\epsilon}{2})|S|d. \qquad (7)$$

Lemma 3.9 *Suppose $\epsilon < 1$ and $G = (U, V, E)$ is an (m, s, n, d, ϵ)-expander with $d \ge 1$. Then, $\{\Gamma(u)\}_{u \in U}$ is a collection of m non-empty sets, and has the (n, ϵ)-intersection property.*

Proof Since $\epsilon < 1$, we have from (7) that $|\Gamma(\{u, u'\})| > d$ for distinct $u, u' \in U$. Thus, $\Gamma(u) \ne \Gamma(u')$, and there are m non-empty sets in $\{\Gamma(u)\}_{u \in U}$. Suppose $\{\Gamma(u)\}_{u \in U}$ does not have the (n, ϵ)-intersection property, then there exists a set $S = \{u_1, \ldots, u_k, v_1, \ldots, v_k\} \subseteq U$ of $2k$ (for some $k \in [n]$) distinct elements such that for $j = 1, \ldots, k$,

$$|\Gamma(v_j) \cap \bigcup_{i=1}^{k} \Gamma(u_i)| > \epsilon d.$$

But then $|\Gamma(S)| < 2kd - k\epsilon d \le (1 - \frac{\epsilon}{2}) \cdot 2kd = (1 - \frac{\epsilon}{2})|S|d$, violating (7). $\square$

Existence of expanders: Finally, we show that the required expander graph exits.

Lemma 3.10 *For $\epsilon > 0$, $m \ge 8$ and $n \le m/2$, there is an $(m, \lceil \frac{200 n \log m}{\epsilon^2} \rceil, n, \lfloor \frac{\log m}{\epsilon} \rfloor, \epsilon)$-expander.*

Proof We show the existence of the expander graph $G = (U, V, E)$, where $U = [m]$ and $V = [s]$, using a standard probabilistic argument[2]. For each vertex u in U, we independently choose its set of neighbors in V, $\Gamma(u)$, by picking without replacement $d = \lfloor \frac{\log m}{\epsilon} \rfloor$ elements from V at random. We wish to show that with non-zero probability the resulting graph is an expander (satisfying (7)). For, $T \in \binom{U}{\leq 2n}$, let

$$\mathcal{E}(T) \stackrel{\text{def}}{\equiv} |\Gamma(T)| < (1 - \frac{\epsilon}{2})|T|d.$$

We wish to show that with non-zero probability we can (simultaneously) avoid $\mathcal{E}(T)$ for all $T \in \binom{U}{\leq 2n}$.

Claim 3.11 $\Pr[\mathcal{E}(T)] \leq \left(\frac{2}{m}\right)^{2|T|}$.

Proof Let $t \stackrel{\text{def}}{=} |T|$. We prove the claim under the assumption that the neighbors of $u \in U$ are chosen by sampling with replacement. This will imply the claim even for sampling without replacement: to pick a random set of size d, first pick d elements with replacement, resulting in d' distinct elements (say), and then add $d - d'$ new elements randomly. Suppose $T = \{u_1, u_2, \ldots, u_t\}$. Let $N \stackrel{\text{def}}{=} td$. With the choice of elements for $\Gamma(u_1), \ldots, \Gamma(u_t)$, where $\Gamma(u_j) = \{e_{(j-1)d+1}, \ldots, e_{jd}\}$, we associate N random variables $X_1, \ldots, X_N$ such that:

$$X_i \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } e_i \in \{e_1, \ldots, e_{i-1}\} \\ 0 & \text{otherwise} \end{cases}.$$

Thus,

$$\mathcal{E}(T) \equiv \sum_{i=1}^{N} X_i > \frac{\epsilon}{2} N. \tag{8}$$

Now, for all $i \in [N]$ and $\sigma \in \{0,1\}^{i-1}$,

$$\Pr[X_i = 1 \mid X_1 X_2 \cdots X_{i-1} = \sigma] \leq \frac{(i-1)}{s} \leq \frac{2nd - 1}{s} \leq \frac{\epsilon}{100}.$$

Let $p \stackrel{\text{def}}{=} \frac{\epsilon}{100}$, and define N independent random variables $Y_1, \ldots, Y_N$ such that

$$Y_i = \begin{cases} 1 & \text{with probability } \frac{\epsilon}{100} \\ 0 & \text{otherwise} \end{cases}.$$

[2] The argument we use is different from what is usually used for showing the existence of expander graphs. Consider the random graph G obtained by choosing d random neighbors from V (with replacement) for each vertex in U. We wish to avoid the event $\exists S \subseteq U\ (|S| \leq 2n) : |\Gamma(S)| \leq (1 - \frac{\epsilon}{2})d|S|$. One usually [3, p. 331] bounds the probability of this event by

$$\sum_{i=1}^{2n} \binom{m}{i} \binom{|V|}{(1-\frac{\epsilon}{2})di} \left(\frac{(1-\frac{\epsilon}{2})di}{|V|}\right)^{di}.$$

For our choice of parameters, $d = \lfloor \frac{\log m}{\epsilon} \rfloor$ and $|V| = \lceil \frac{200n \log m}{\epsilon^2} \rceil$, this quantity is not less than 1 (for example, consider the term with $i = n$), although in Lemma 3.10, we show that G has the required expansion property with high probability.

Then, for all k,

$$\Pr[\sum_{i=1}^{N} X_i \geq k] \leq \Pr[\sum_{i=1}^{N} Y_i \geq k]. \tag{9}$$

We will use the following form of Chernoff's bound (see, for example, Alon and Spencer [1, Theorem A.12]):

$$\Pr[\sum_{i=1}^{N} Y_i \geq (p+\delta)N] \leq \left(\frac{ep}{p+\delta}\right)^{(p+\delta)N}. \tag{10}$$

Thus,

$$\begin{aligned}\Pr[\sum_{i=1}^{N} Y_i \geq \frac{\epsilon}{2}N] &\leq \left(\frac{e\epsilon/100}{\epsilon/2}\right)^{\epsilon N/2}\\ &\leq (\frac{1}{4})^{\epsilon t d}\\ &\leq \left(\frac{1}{4}\right)^{\epsilon t(\frac{\log m}{\epsilon}-1)}\\ &\leq \left(\frac{2}{m}\right)^{2t}.\end{aligned}$$

Claim 3.11 now follows from (8) and (9). □

Since $\Pr[\mathcal{E}(\emptyset)] = 0$, Claim 3.11 implies that

$$\Pr[\bigvee_{T\in\binom{U}{\leq 2n}} \mathcal{E}(T)] \leq \sum_{t=1}^{2n}\binom{m}{t}\left(\frac{2}{m}\right)^{2t} \leq \sum_{t=1}^{2n}\left(\frac{4}{m}\right)^{t} < 1.$$

where we use our assumption, $m \geq 8$, to justify the last inequality. Thus, with non-zero probability, G is an expander. ■

3.3 Multiprobe randomized scheme with one-sided error

We now show that the space requirement for schemes with one-sided error can be reduced if more bitprobes are allowed.

Proof of Theorem 5: Our scheme will have two tables. The first table, T_1, will come directly from Theorem 1 (with an appropriate choice of ϵ); the second table will be based on ideas used in Theorem 3. Our multiprobe query algorithm will, correspondingly, have two phases. In the first phase, it will probe T_1 several times (to reduce the error). In the second phase, it will probe T_2 just once. The scheme is non-adaptive: the locations to be probed are completely determined by the query element and the random string, but don't depend on the actual values read.

We will view the first half of our scheme as a randomized multiprobe scheme with small two-sided error.

Lemma 3.12 *There is a randomized $(n, m, O(\frac{n}{\epsilon^2}\log m), t)$-scheme that makes an error of at most $(2e\epsilon)^{t/2}$.*

Proof We use the randomized $(n, m, s, 1)$-scheme of Theorem 1, but run the query algorithm t times (with independent coin tosses) and say 'Yes' if and only if at least $t/2$ of the t runs give the answer 'Yes'. Using Chernoff's bound (see (10) above), we conclude that

$$\Pr[\mathsf{Error}] \leq (2e\epsilon)^{t/2}.$$

□

We will use the above lemma with $\epsilon = n^{-\delta/2}/(2e)$ and $t = \lceil 4/\delta \rceil$. This will give us an $(n, m, O(n^{1+\delta} \log m), \lceil 4/\delta \rceil)$-scheme Π with error probability at most n^{-2}. We use Π to construct our multiprobe scheme $\hat{\Pi}$. The first table, T_1, of our scheme is exactly what Π uses; it has $s = O(n^{1+\delta} \log m)$ bits. The second table, T_2, has $s' = \lceil 2n^{1+\delta} \log m \rceil$ bits. To see how the contents of T_2 are decided, let us first describe how the query algorithm uses T_2.

The query algorithm: Suppose the query algorithm of Π uses random strings of length ℓ. In our query algorithm, for each $u \in [m]$, we have a sequence $\sigma_u = \langle r_1, r_2, \ldots, r_{s'} \rangle$, of strings $r_i \in \{0,1\}^\ell$. On receiving the query "Is u in S?", the algorithm first chooses i uniformly at random from $[s']$, and then uses the query algorithm of scheme Π with the ith element of σ_u (that is, r_i above) as the random string and T_1 as its table. If the answer returned by Π is 'Yes', we say 'Yes'. If the answer is 'No', we move on to table T_2, probe the location i there, and say 'Yes' if and only if we read a 1. Thus, to completely specify the query algorithm, we must fix sequences σ_u for each $u \in [m]$. We will show later how suitable sequences σ_u can be found. First, let us determine the contents of table T_2.

The table T_2: Once the query algorithm is specified, there is a natural choice for the contents of T_2. Let $\mathsf{Error}_\Pi(S, u, r)$ denote the event "after storing the set S in its table, the protocol Π gives a wrong answer for the query "Is u in S?" when it uses the random string r." For $S \in \binom{[m]}{\leq n}$, let

$$\begin{aligned} R(S, u) &\stackrel{\text{def}}{=} \{i : \mathsf{Error}_\Pi(S, u, \sigma_u(i))\}; \text{ and} \\ R(S) &\stackrel{\text{def}}{=} \bigcup_{u \in S} R(S, u). \end{aligned}$$

Since we allow no error for query elements $u \in S$, $T_2(i)$ must be 1 for all $i \in R(S)$, when we store the set S in our tables. Let the remaining bits of T_2 be 0. This clearly ensures that $\Pr_i[\mathsf{Error}_{\hat{\Pi}}(S, u, i)] = 0$ for all $S \in \binom{[m]}{\leq n}$ and $u \in S$. It remains only to ensure that $\Pr_i[\mathsf{Error}_{\hat{\Pi}}(S, u, i)] \leq n^{-\delta}$, when $u \notin S$. For this, we need to choose σ_u carefully.

Choosing σ_u: For $u \notin S$, we have $\mathsf{Error}_{\hat{\Pi}}(S, u, i) \equiv \mathsf{Error}_\Pi(S, u, \sigma_u(i)) \vee i \in R(S)$. Thus,

$$\Pr_i[\mathsf{Error}_{\hat{\Pi}}(S, u, i)] \leq \Pr_i[\mathsf{Error}_\Pi(S, u, \sigma_u(i))] + \Pr_i[i \in R(S)]. \tag{11}$$

To bound the first term on the right we show

$$\forall u, S, |R(S, u)| \leq n \log m; \tag{12}$$

to bound the second, we show

$$\forall S, |R(S)| \;\leq\; n \log m. \tag{13}$$

Since $n \log m \leq \frac{s'}{2n^\delta}$, using these bounds in (11), we obtain $\Pr_i[\mathsf{Error}_{\hat{\Pi}}(S,u,i)] \leq n^{-\delta}$.

Thus, we need to choose σ_u such that (12) and (13) hold. Let each σ_u be a sequence of s' randomly (uniformly and independently) chosen strings from $\{0,1\}^\ell$. Now,

$$\Pr_{r \in \{0,1\}^\ell}[\mathsf{Error}_\Pi(S,u,r)] \;\leq\; \frac{1}{n^2}.$$

Since σ_u is obtained by picking $s' = \lceil 2n^{1+\delta} \log m \rceil$ strings from $\{0,1\}^\ell$ independently, we get using Chernoff's bound (see (10) above) that

$$\Pr_{\{\sigma_u\}}[|R(S,u)| > n \log m] \;\leq\; \left(\frac{e/n^2}{1/(4n^\delta)}\right)^{n \log m} \;\leq\; \left(\frac{4e}{n}\right)^{n \log m}.$$

[Note $(n \log m)/s' \geq 1/(4n^\delta)$.] Thus,

$$\Pr_{\{\sigma_u\}}[\exists u \in [m], S \in \binom{[m]}{\leq n} \; |R(S,u)| > n \log m] \leq m^{n+1} \left(\frac{4e}{n}\right)^{n \log m} < \frac{1}{2}.$$

Thus, (12) holds with probability more than $\frac{1}{2}$.

To ensure (13), we first observe that for each $i \in [s']$,

$$\Pr[i \in R(S)] \;\leq\; \sum_{u \in S} \Pr[i \in R(S,u)] \;\leq\; n \times \frac{1}{n^2} = \frac{1}{n}.$$

Furthermore, the events '$i \in R(S)$' are independent for different $i \in [s']$. Thus, using Chernoff's bound (see (10) above), we obtain

$$\Pr_{\{\sigma_u\}}[|R(S)| > n \log m] \;\leq\; \left(\frac{e/n}{1/(4n^\delta)}\right)^{n \log m} \;\leq\; \left(\frac{4e}{n^{1-\delta}}\right)^{n \log m},$$

and

$$\Pr_{\{\sigma_u\}}[\exists S \in \binom{[m]}{\leq n} \; |R(S)| > n \log m] \;\leq\; m^n \left(\frac{4e}{n}\right)^{n \log m} < \frac{1}{2}.$$

Thus, (13) holds with probability more than $\frac{1}{2}$.

Since (12) and (13) both hold with probability more than $\frac{1}{2}$, they hold simultaneously for some choice $\{\sigma_u\}_{u \in [m]}$. We fix one such choice in our query algorithm. ■

4 Lower bounds for randomized schemes

Consider a scheme that uses space s and just one bitprobe. In general, on receiving a query the algorithm does one of three things based on the outcome of its coin tosses and the query element:

1. It decides to answer 'Yes', regardless of what is stored in the table (which it may or may not read);

2. It decides to answer 'No', regardless of what is stored in the table;

3. It computes, based on the coin tosses and the query element, an index $i \in [s]$ and

   (a) answers 'Yes' if and only if the ith bit of the table is 1; or

   (b) answers 'Yes' if and only if the ith bit of the table is 0.

It will be convenient if our query algorithm has the following *standard form*: it always reads some bit of the table and answers 'Yes' if and only if it reads a 1. A scheme with a general query algorithm can be modified easily so that the new query algorithm is in standard form. This modification will roughly double the space required but keep the error probability the same. Suppose the original algorithm used the table $T : [s] \to \{0,1\}$. The new algorithm will use a table $T' : [s+1] \times \{0,1\} \to \{0,1\}$, whose contents are defined by

$$T'(i,b) \stackrel{\text{def}}{=} \begin{cases} T(i) & \text{if } i \in [s] \text{ and } b = 1 \\ \neg T(i) & \text{if } i \in [s] \text{ and } b = 0 \\ b & \text{if } i = s+1 \end{cases} .$$

The query algorithm is then modified as follows: in case (1) above, when the old algorithm always said 'Yes', the new algorithm reads the bit $T'(s+1,1)$; in case (2) the new algorithm reads $T'(s+1,0)$; in case (3a) it reads $T'(i,1)$; and in case (3b) it reads $T'(i,0)$. In all cases, the answer is 'Yes' if and only if the bit read is 1.

Our lower bounds for randomized one-probe schemes are based on bounds for r-cover-free families.

Definition 4.1 *A family of sets $\mathcal{F}$ is r-cover-free if for $T_0, T_1, \ldots, T_r \in \mathcal{F}$ such that $T_0 \notin \{T_1, T_2, \ldots, T_r\}$, $T_0 \not\subseteq T_1 \cup \ldots \cup T_r$.*

Theorem 10 (Füredi [16]) *If $\mathcal{F}$ is an r-cover-free family of sets and $r \leq |\mathcal{F}|^{1/3}$, then*

$$|\bigcup_{T \in \mathcal{F}} T| \geq \frac{r^2}{4 \log r + O(1)} \log |\mathcal{F}|.$$

[Similar bounds have been shown by [7, 28, 6].]

4.1 Randomized schemes with one-sided error

Proof of Theorem 4: Suppose there is a randomized $(n, m, s, 1)$-scheme with negative one-sided error ϵ. As discussed earlier, this implies that there is a randomized $(n, m, 2s+2, 1)$-scheme with negative one-sided error ϵ, whose query algorithm is in standard form. Consider the bipartite graph $G = (U, V, E)$, where $U = [m]$, $V = [2s+2]$, and $(u, v) \in E$ if and only if location v is probed (with non-zero probability) on query "Is u in S?". In particular, on query "Is u in S?", the algorithm picks an element $v \in \Gamma(u)$ at random, according to some distribution D_u, and answers 'Yes' if and only if there is a 1 in location v of the table.

Let $r \stackrel{\text{def}}{=} \lceil \frac{n}{\epsilon} \rceil - 1$; note $r < \frac{n}{\epsilon}$.

Claim 4.2 $\{\Gamma(u)\}_{u\in U}$ *is an r-cover-free family.*

Proof Suppose the claim is false. Then, there exist distinct $u, u_1, u_2, \ldots, u_r \in U$ such that $\Gamma(u) \subseteq \bigcup_{i=1}^{r} \Gamma(u_i)$. Let S be a random subset of $\{u_1, u_2, \ldots, u_r\}$ of size n. Then, for each $i \in \Gamma(u)$, $\Pr[i \in \Gamma(S)] \geq \frac{n}{r}$. For $T \subseteq V$, let $D_u(T) \stackrel{\text{def}}{=} \sum_{i\in T} D_u(i)$. By linearity of expectation

$$\mathrm{E}[D_u(\Gamma(S))] \;=\; \sum_{i\in\Gamma(u)} D_u(i)\Pr[i\in\Gamma(S)] \;\geq \frac{n}{r} \sum_{i\in\Gamma(u)} D_u(i) \;=\; \frac{n}{r} \;>\; \epsilon.$$

Fix a choice for S with $D_u(\Gamma(S)) > \epsilon$. When S is stored, all locations in $\Gamma(S)$ must contain a 1 (because the error is negative one-sided). Then, on query "Is u in S?", the query algorithm answers 'Yes' with probability more than ϵ, but the error allowed is at most ϵ. □

Claim 4.2 and Theorem 10 imply that if $r \leq m^{1/3}$, then

$$|V| \;\geq\; \frac{r^2}{4\log r + O(1)} \log m.$$

Thus, $s = \Omega(\frac{n^2}{\epsilon^2 \log(n/\epsilon)} \log m)$. ■

We next consider positive one-sided error and observe that bitvectors are optimal in this case.

Theorem 11 *Let $\epsilon < 1$ and $m \geq 1$. Any randomized $(1, m, s, 1)$-scheme with positive one-sided error ϵ must have $s \geq m$.*

Proof Since $\epsilon < 1$, for each $u \in [m]$ there must be a coin toss sequence r_u for which the query algorithm says 'Yes' when the set $\{u\}$ is stored and the query "Is u in S?" is posed. In this case, the algorithm must probe some location of the table, for otherwise, it would say 'Yes' with non-zero probability even when the empty set is stored. Let $\ell_u \in [s]$ be the location probed, and let $b_u \in \{0,1\}$ be the bit read. We claim that $\ell_u \neq \ell_{u'}$ for $u \neq u'$. For, suppose $u \neq u'$ and $\ell_u = \ell_{u'}$. We have two cases.

$b_u = b_{u'}$: Store $S = \{u\}$. On query "Is u' in S?" and coin toss sequence $r_{u'}$ the algorithm will answer 'Yes', which is not allowed.

$b_i \neq b_j$: Store the empty set. Either on query "Is u in S?" with coin toss sequence r_u, or on query "Is u' in S?" with coin toss sequence $r_{u'}$, the answer will be 'Yes'. But when the empty set is stored the answer should be 'No' with probability 1 for all queries.

Thus, $\ell_u \neq \ell_{u'}$ when $u \neq u'$, implying $s \geq m$. ■

4.2 Randomized schemes with two-sided error

To prove the lower bound for randomized schemes with two-sided error, we need to use *upper bounds* on r-cover free families together with the lower bound.

Proof of Theorem 2: Fix a randomized $(n, m, s, 1)$-scheme that answers queries with probability of error at most ϵ. We assume that the query algorithm is in standard form, and, as before, model it using the bipartite graph (U, V, E), where $U = [m]$ and $V = [s]$: on query "Is u in S?", the algorithm probes a random location in $[s]$ according to a distribution D_u ($D_u(i) \neq 0$ iff $i \in \Gamma(u)$), and answers 'Yes' if and only if the location contains a 1.

For our lower bound, we will need an r-cover-free family $\mathcal{F} \subseteq \binom{[m]}{\leq n}$, where $r = \lfloor \frac{1}{\epsilon} \rfloor$. We first present the argument assuming such a family; later we will obtain our lower bound by choosing a suitably large $\mathcal{F}$. For $S \in \mathcal{F}$, let $T_S \subseteq [s]$ be the set of locations of the table that contain a 1 when S is stored. Let $\ell = \lfloor \frac{1}{2\epsilon} \rfloor$. Since $\epsilon \leq \frac{1}{4}$, we have $\ell \geq 1$.

Claim 4.3 *$\{T_S : S \in \mathcal{F}\}$ is ℓ-cover-free.*

Proof Suppose $T_{S_0} \subseteq T_{S_1} \cup T_{S_2} \cup \ldots \cup T_{S_\ell}$, for some $S_0, S_1, \ldots, S_\ell \in \mathcal{F}$ such that $S_0 \notin \{S_1, S_2, \ldots, S_\ell\}$. We will derive a contradiction.

Since $\mathcal{F}$ is r-cover-free and $r \geq \ell$, we have $S_0 \not\subseteq \bigcup_{i=1}^{\ell} S_i$; let $u \in S_0 - \sum_{i=1}^{\ell} S_i$. Since $u \in S_0$, $D_u(T_{S_0}) \geq 1 - \epsilon$. Thus, $D_u(\bigcup_{i=1}^{\ell} T_{S_i}) \geq 1 - \epsilon$ and $D_u(T_{S_i}) \geq \frac{1-\epsilon}{\ell}$ for some $i \in [n]$. Fix one such i. Now, when the scheme stores the set S_i and receives the query "Is u in S_i?", it says "Yes" with probability at least $\frac{1-\epsilon}{\ell} \geq 2\epsilon(1 - \epsilon) > \epsilon$. But this is not possible. □

Using the above claim and Theorem 10, we obtain that if $\ell \leq |\mathcal{F}|^{1/3}$, then

$$s \geq \frac{\ell^2}{4 \log \ell + O(1)} \log |\mathcal{F}|. \tag{14}$$

To prove our lower bound, we need to find a r-cover-free family of large size. If $\epsilon \geq \frac{1}{n}$, we use Lemma 3.2 to obtain $\mathcal{F} \subseteq \binom{[m]}{n}$ of size at least

$$\frac{\binom{m}{\lceil \epsilon n \rceil}}{\binom{n}{\lceil \epsilon n \rceil}^2} \geq \left(\frac{m\epsilon}{e^2 n}\right)^{\epsilon n},$$

where the pairwise intersection of sets is of size at most $\lceil \epsilon n \rceil - 1$. Such a family is $\lfloor \frac{1}{\epsilon} \rfloor$-cover-free, for otherwise some pair of sets would intersect on at least $\lceil \frac{n}{\lfloor 1/\epsilon \rfloor} \rceil \geq \lceil \epsilon n \rceil$ elements. Then, (14) gives us (using our assumption $n \leq m^{1/3}$)

$$s \geq \frac{(1/\epsilon)^2}{4 \log(1/\epsilon) + O(1)} \log \left(\frac{m\epsilon}{e^2 n}\right)^{\epsilon n} = \Omega\left(\frac{n}{\epsilon \log(1/\epsilon)} \log m\right).$$

If $\epsilon < \frac{1}{n}$, we use the r-cover-free family $\binom{[m]}{1}$ and use (14) to obtain

$$s \geq \frac{(1/\epsilon)^2}{4 \log(1/\epsilon) + O(1)} \log m = \Omega\left(\frac{n}{\epsilon \log(1/\epsilon)} \log m\right).$$

■

5 Deterministic schemes

We now show a time-space tradeoff result for deterministic schemes.

Proof of Theorem 6: Recall that the bitstring used to store the set $S \in \binom{[m]}{n}$ is called $\phi(S)$. Let

$$T_S \stackrel{\text{def}}{=} \{\langle \ell, \phi(S)(\ell)\rangle : \text{location } \ell \text{ of } \phi(S) \text{ is probed on query "Is } u \text{ in } S\text{?" for some } u \in S\}.$$

We now observe that the sets T_S have to be incomparable for different S. For, if $T_{S_1} \supseteq T_{S_2}$ for $S_1 \neq S_2$, store the set S_1 and ask the query "Is u in S_1?" for an element $u \in S_2 - S_1$. The scheme will err on this query which is a contradiction. Now, each T_S is a subset of size at most nt of the set $[s] \times \{0,1\}$. It follows, from the LYM inequality (see, for example, Alon and Spencer [1, p. 183]) that $\binom{m}{n} \leq \max_{i \leq nt} \binom{2s}{i}$. ■

We now study deterministic schemes, when the number of probes allowed is small.

Proof of Theorem 7, Part (1): We will obtain our deterministic scheme from a randomized $(m, n, O(tnm^{2/(t+1)}), 1)$-scheme with error probability *less* than $\frac{1}{2}$. The randomized scheme we use will be in the standard from: on query "Is u in S?", the query algorithm will pick a location randomly from a set $\Gamma(u)$, and say 'Yes' if and only if a 1 is stored there. The size of $\Gamma(u)$ will be exactly $2t+1$. To obtain the deterministic scheme, we read all locations in $\Gamma(u)$, and say 'Yes' if a majority (at least $t+1$) of them contain 1.

To construct the randomized scheme, we use the method of Theorem 1. Using calculations similar to those in the proof of Lemma 3.10, one can show that there is a graph $G = (U, V, E)$, where $U = [m]$ and $V = [s]$, such that $s = O(tnm^{2/(t+1)})$ and $|\Gamma_u| = 2t+1$ for $u \in U$, such that $\{\Gamma(u)\}_{u \in U}$ has the $(n, \frac{t}{2t+1})$-intersection property. By Lemmas 3.6 and 3.4, it follows that there is an $(m, n, O(tnm^{2/(t+1)}), 1)$-scheme with error probability less than $\frac{1}{2}$. ■

Proof of Theorem 7, Part (2): Our adaptive scheme uses a combination of the FKS scheme and the bitvector scheme.

Storing Scheme: Given a set T, do the following.

Step 1: Find a prime $p < n^2 \log m$ such that if $x \neq y, x, y \in T$, then $x \bmod p \neq y \bmod p$. The fact that such a prime exists has been shown by Fredman, Komlós and Szemerédi [15]. Store p using $O(\log n + \log\log m)$ bits.

Step 2: Now, the set $T \bmod p$ consists of n elements, each less than $n^2 \log m$. Store this set using the FKS data structure. This requires space $O(n(\log n + \log\log m))$.

Step 3: For every $x \in T$, do the following: divide the string x into t blocks $B_1, \ldots, B_t$ each of size $\log m/t$. For each such block B_i, construct a look-up table of size $2^{\log m/t}$ with a 1 in the index given by B_i. Space required is $nt2^{\log m/t}$ bits.

Space used by the storing scheme is $O(n(\log n + \log\log m) + nt2^{\log m/t})$.

Query Scheme: Given a query u, do the following.

Step 1: Read the prime p. This requires $O(\log n + \log\log m)$ bitprobes.

Step 2: Find u mod p. Now, check if there is an element y in T such that $u \bmod p = y \bmod p$ using the FKS structure. This requires $O(\log n + \log\log m)$ bitprobes.

Step 3: If there is no such y, say *no*.

Step 4: If there is such a y, retrieve a pointer to it using $O(\log n)$ bitprobes. Then divide u into t blocks and check if $x = y$ block by block. This requires t bitprobes.

Time used by the query scheme is $O(\log n + \log\log m) + t$. ■

Finally, we consider deterministic schemes that use two bitprobes. We show that two non-adaptive probes do not help even for $n = 2$. We also show that adaptiveness helps for $n = 2$.

Proof of Theorem 8, Part(1): There are 16 different functions mapping $\{0,1\}^2$ to $\{0,1\}$. We will divide them into 3 classes.

1. Degenerate functions. These are the functions depending on at most one variable. There are 6 such functions, namely 0, 1, u, y, $\bar{x}$, $\bar{y}$.

2. Inflexible functions. These are the functions f, so that there there is a value of $f(x, y)$ which determines the value of u as well as the value of y. For instance $x \wedge y = 1 \Rightarrow x = y = 1$. There are 8 such functions, namely $x \wedge y$, $\bar{x} \wedge y$, $x \wedge \bar{y}$, $\bar{x} \wedge \bar{y}$, $x \vee y$, $\bar{x} \vee y$, $x \vee \bar{y}$, $\bar{x} \vee \bar{y}$.

3. Flexible functions. These are functions which are neither degenerate nor inflexible. There are 2 such functions, namely $x \oplus y$, and $\bar{x} \oplus y$.

Suppose the theorem fails. Fix m at the smallest value for which this happens. Note that the theorem is trivially true for $m = 1$. Let U be a universe of size m. The corresponding scheme associates with each $z \in U$ two locations u_z and v_z in $\{1, .., s\}$ and a Boolean function $f_z : \{0,1\}^2 \to \{0,1\}$. Clearly, if for some z, f_z is constant, the scheme is incorrect. Now assume that for some z, f_z is degenerate. Assume that it depends on its first variable. We can get a scheme for a universe of size $m-1$ by fixing the value of u_z, yielding a structure of size $s-1$. This is a contradiction. Thus we can assume that all functions f_z are inflexible or flexible. Now assume, to the contrary, that $s < m$. Let the multi-graph $G = (V, E)$ be given by $V = \{1, \ldots, s\}$ and $E = \{e_z = (u_z, v_z) : z \in U\}$. Let an edge e_z be denoted flexible if the corresponding function f_z is flexible and let it be denoted inflexible otherwise. As $|E| = m$, $|V| = s$, and $s < m$, G contains a cycle C (as G is a multi-graph, it may be the case that C consists of exactly two identical edges).

Now there are two cases:

1. *The cycle C contains only flexible edges.* Let Z be the elements corresponding to the edges in C. Fix any setting τ of the bits in the data structure. Each edge e_z on the cycle corresponds to an element $z \in Z$, and we can see if $z \in S$ by adding, modulo 2, the setting $\tau(u_z)$ of the bit u_z and the setting $\tau(v_z)$ of the bit v_z or the setting $\overline{\tau(u_z)}$ of the bit $\overline{u}_z$ and the setting $\tau(v_z)$ of the bit v_z. This quantity summed over all z in Z uniquely determines the parity of the number of elements of $S \cap Z$. But this sum is zero or one depending on the number of z of the second type. As the sum determines the parity of $|Z \cap S|$, either $\emptyset$ or $\{z\}$, where z is any element of Z, has no valid representation.

2. *The cycle C contains an inflexible edge.* Let this edge be e_z. There is a choice of $z \in S$ or $z \notin S$ which makes only one configuration of the values of the bits u_z,v_z possible. Fix this choice. Let us assume that it is $z \in S$ (the case $z \notin S$ is similar). Now let z_1 be the other edge on C, adjacent to v_z. Having already fixed $v_z = u_{z_1}$, there are two possibilities: either having thus fixed u_{z_1} determines whether $z_1 \in S$ or it doesn't. If it does, either $\{z\}$ or $\{z, z_1\}$ has no valid representation and we are done. If it doesn't, we fix the setting of v_{z_1} in the unique way so that $z_1 \notin S$. Now let z_2 be the other edge on C, adjacent to v_{z_1}. Having already fixed $v_{z_1} = u_{z_2}$, there are two possibilities: fixing u_{z_2} in this manner either determines whether or not $z_2 \in S$, or it doesn't. If it does, either $\{z\}$ or $\{z, z_1\}$ has no valid representation and we are done. If it doesn't, we fix the setting of v_{z_2} in the unique way so that $z_2 \notin S$. Now let z_3 be the other edge on C, adjacent to v_{z_2}, etc... Thus working our way around the cycle, we finally come to an edge e_{z_l} adjacent to u_z. Thus, we have fixed u_{z_l} as well as $v_{z_l} = u_z$ and we conclude that either $\{z\}$ or $\{z, z_l\}$ has no valid representation.

Thus, we have arrived at a contradiction, and we conclude $s \geq m$. ■

It is easy to get a 3 probe non-adaptive scheme for sets of size at most 2 using space $O(m^{1/2})$. Such a scheme can be obtained, for instance, from the explicit one-sided error scheme by setting the parameters appropriately.

Proof of Theorem 8, Part(2): The structure of the proof is as follows. We will first define a certain combinatorial object. We then show that the existence of this object implies the two-probe scheme claimed in the Theorem. Then we show, using the probabilistic method (in particular, the *alteration* technique), that the desired object exists.

Thus, let $U = \{1, 2, \ldots, m\}$. Let an *augmented s-partition* of U be a system consisting of

1. A partition of U into classes $U_1, U_2, \ldots, U_s$
2. a set system $M_1, M_2, \ldots, M_s$ on $\{1, 2, \ldots, s\}$
3. a set system $N_1, N_2, \ldots, N_s$ on $\{1, 2, \ldots, s\}$
4. a family of one-to-one maps $f_i : U_i \to M_i$
5. a family of one-to-one maps $g_i : U_i \to N_i$

with the following property

For all $U_i \neq U_j$, $x \in U_i$ and $y \in U_j$, we have *either* $f_i(x) \notin M_j$ and $f_j(y) \notin M_i$ *or* $g_i(x) \notin N_j$ and $g_j(y) \notin N_i$.

Claim 5.1 *If an augmented s-partition exists, there is an adaptive two probe scheme using $3s$ bits solving the membership problem.*

Proof

Storing scheme: The scheme will make use of three tables T, T_0 and T_1 each of size s. Given a set $S = \{x, y\}$, there are three cases.

1. *u and y are in the same class, U_i.* We let $T[i] = 0$ and $T[j] = 1$ for all $j \neq i$. We let $T_0[f_i(x)] = T_0[f_i(y)] = 1$ and let $T_0[j] = 0$ for all $j \notin \{f_i(x), f_i(y)\}$. We let $T_1[j] = 0$ for all j.

2. *u and y are in different classes, $x \in U_i$ and $y \in U_j$. Furthermore $f_i(x) \notin M_j$ and $f_j(y) \notin M_i$.* We let $T[i] = T[j] = 0$ and $T[k] = 1$ for all $k \notin \{i, j\}$. We let $T_0[f_i(x)] = T_0[f_j(y)] = 1$ and let $T_0[j] = 0$ for all $j \notin \{f_i(x), f_j(y)\}$. We let $T_1[j] = 0$ for all j.

3. *x and y are in different classes, $x \in U_i$ and $y \in U_j$. Furthermore $g_i(x) \notin N_j$ and $g_j(y) \notin N_i$.* We let $T[i] = T[j] = 1$ and $T[k] = 0$ for all $k \notin \{i, j\}$. We let $T_1[g_i(x)] = T_1[g_j(y)] = 1$ and let $T_0[j] = 0$ for all $j \notin \{g_i(x), g_j(y)\}$. We let $T_0[j] = 0$ for all j.

Query scheme: Given a query q in class U_i , do the following:

1. Read $T[i]$.
2. If it is 0, read $T_0[f_i(q)]$. If 1 is seen, say *yes*. Otherwise say *no*.
3. If it is 1, read $T_1[g_i(q)]$. If 1 is seen, say *yes*. Otherwise say *no*.

It is easily seen that the augmented partition property ensures that the scheme is correct. □

The theorem now follows from the following claim. ■

Claim 5.2 *For any m, an augmented $O(m^{3/4})$-partition exists.*

Proof Let $U' = \{1, 2, \ldots, 2m\}$. We shall construct a "blemished" augmented partition of U' and then alter it so that it has the right property.

We assume without loss of generality that $s = 2(2m)^{3/4}$ is an integer and divides $2m$. Thus, we can partition U' evenly into $U_1, U_2, \ldots, U_s$, each of size $\frac{1}{2}(2m)^{1/4}$. Now we choose M_i and N_i independently at random from among all subsets of size $\frac{1}{2}(2m)^{1/4}$ of $\{1, 2, \ldots, s\}$. We choose f_i and g_i independently at random from all one-to-one functions.

Now, consider fixed $x \in U_i, y \in U_j$ for $i \neq j$. We will bound the probability that the property fails for this particular pair. Clearly,

$$\Pr[f_i(x) \in M_j] = \frac{|M_j|}{s} = \frac{\frac{1}{2}(2m)^{1/4}}{2(2m)^{3/4}} = \frac{1}{4(2m)^{1/2}}$$

By the union bound,

$$\Pr[f_i(x) \in M_j \vee f_j(y) \in M_i] \leq \frac{1}{2(2m)^{1/2}}$$

and similarly

$$\Pr[g_i(x) \in N_j \vee g_j(y) \in N_i] \leq \frac{1}{2(2m)^{1/2}}.$$

As the two events are independent, we get

$$\Pr[(f_i(x) \in M_j \vee f_j(y) \in M_i) \wedge (g_i(x) \in N_j \vee g_j(y) \in N_i)] \leq \frac{1}{8m}.$$

Therefore, the expected number of pairs (x, y) for which the desired property does not hold is less than $\binom{2m}{2}\frac{1}{8m} < m$. Hence there is a choice of M_i, N_i, f_i, g_i such that the number of bad pairs (x, y) is at most m. Now from each such bad pair, remove one of the elements. We remove at most m elements, yielding a universe of size at least m, as desired. For the remaining elements, the property holds, and we are done. $\square$

The bound of $O(m^{3/4})$ in the above claim has been improved to $O(m^{2/3})$ by Venkatesh Raman and S. Srinivasa Rao (see [27]).

Acknowledgments

We are grateful to Venkatesh Raman, S. Srinivasa Rao and Amnon Ta-Shma for their comments on this work, and to the referees for their help in improving the paper.

References

[1] N. Alon and J. H. Spencer. The Probabilistic Method. Wiley-Interscience, New York, 1992.

[2] V. Arvind and J. Köbler, On resource-bounded measure and pseudorandomness, *Proceedings of FSTTCS'97, LNCS 1346*, 1997, pages 235–249.

[3] B. Bollobás, Random Graphs, Academic Press, 1985.

[4] L. Babai, L. Fortnow, L.A. Levin, M. Szegedy, Checking computations in polylogarithmic time, *Proceedings of STOC'91*, pages 21–31.

[5] A. Brodnik and J.I. Munro, Membership in constant time and minimum space, *Lecture Notes In Computer Science*, 855, 1994, pages 72–81. Final version: Membership in Constant Time and Almost-Minimum Space, *SIAM Journal on Computing,* 28(5), 1999, pages 1627–1640.

[6] S. Chaudhuri and J. Radhakrishnan, Deterministic restrictions in circuit complexity, *Proceedings of STOC'96*, pages 30–36.

[7] A.G. Dyachkov and V.V. Rykov, Bounds on the length of disjunctive codes, *Problemy Peredachi Informatsii,*18(3), 1982, pages 7–13 [Russian].

[8] P. Elias and R. A. Flower, The complexity of some simple retrieval problems, *Journal of the Association for Computing Machinery*, 22, 1975, pages 367–379.

[9] P. Erdős, P. Frankl, and Z. Füredi, Families of finite sets in which no set is covered by the union of r others, *Israel Journal of Mathematics*, 51, 1985, pages 79–89.

[10] A. Fiat and M. Naor, Implicit $O(1)$ probe search, *SIAM Journal of computing,* 22, 1993, pages 1–10.

[11] A. Fiat, M. Naor, J.P. Schmidt and A. Siegel, Non-oblivious hashing, *Journal of the Association of Computing Machinery*, 31, 1992, pages 764–782.

[12] G.S. Frandsen, P.B. Miltersen, and S. Skyum, Dynamic word problems, *Journal of the Association of Computing Machinery*, 44, 1997, pages 257–271.

[13] M.L. Fredman, Observations on the complexity of generating quasi-Gray codes, *SIAM Journal of Computing*, 7, 1978, pages 134–146.

[14] M.L. Fredman, The Complexity of Maintaining an Array and Computing its Partial Sums, *Journal of the Association of Computing Machinery*, 29, 1982, pages 250–260.

[15] M. L. Fredman, J. Komlós, and E. Szemerédi, Storing a sparse table with $O(1)$ worst case access time, *Journal of the Association for Computing Machinery*, 31(3), 1984, pages 538–544.

[16] Z. Füredi, On r-cover-free families, *Journal of Combinatorial Theory, Series A*, 73, 1996, pages 172–173.

[17] C.T.M. Jacobs and P. van Emde Boas, Two results on tables, *Information Processing Letters*, 22, 1986, pages 43–48.

[18] E. Kushilevitz, R. Ostrovsky, Y. Rabani, Efficient search for approximate nearest neighbor in high-dimensional spaces, *Proceedings of STOC'98*, pages 614–623.

[19] P. Indyk, R. Motwani, Approximate nearest neighbors: Towards removing the curse of dimensionality, *Proceedings of STOC'98*, pages 604–613.

[20] J. Katz and L. Trevisan, On the efficiency of local decoding procedures for error-correcting codes, In *Proceedings of STOC'00*, pages 80–86.

[21] P.B. Miltersen, The bitprobe complexity measure revisited, In *Proceedings of STACS'93*, pages 662–671.

[22] P.B. Miltersen, N. Nisan, S. Safra, and A. Wigderson, On data structures and asymmetric communication complexity, *Journal of Computer and System Sciences*, 57, 1998, pages 37–49.

[23] M. Minsky and S. Papert, *Perceptrons.* MIT Press, Cambridge, Mass., 1969.

[24] R. Pagh, Low redundancy in static dictionaries with $O(1)$ lookup time, *Proceedings of ICALP '99, LNCS 1644*, pages 595–604.

[25] N. Nisan, Pseudorandom generators for space-bounded computation, *Proceedings of STOC'90*, pages 204–212.

[26] N. Nisan and A. Wigderson, Hardness vs randomness, *Journal of Computer and System Sciences*, 49, 1994, pages 149–167.

[27] J. Radhakrishnan, V. Raman and S.S. Rao, Explicit deterministic construction for membership in the bitprobe model, *Proc. of ESA '01, LNCS 2161*, pages 290–299.

[28] M. Ruszinkó, On the upper bound of the size of r-cover-free families, *Journal of Combinatorial Theory, Series A,* 66, 1984, pages 302–310.

[29] R. Raz, O. Reingold and S. Vadhan, Extracting all the randomness and reducing the error in Trevisan's extractors, *Proceedings of STOC'99*, pages 149–158.

[30] M. Sudan, L. Trevisan and S. Vadhan, Pseudorandom generators without the XOR lemma, *Proceedings of STOC'99*, pages 537–546.

[31] A. Ta-Shma, Explicit one-probe storing schemes using universal extractors, Manuscript, 2001.

[32] A. Ta-Shma, C. Umans and D. Zuckerman, Loss-less Condensers, Unbalanced Expanders, and Extractors, *Proceedings of STOC'2001*, pages 143–152.

[33] A. Ta-Shma and D. Zuckerman, Extractor Codes, *Proceedings of STOC'2001*, pages 193–199.

[34] L. Trevisan, Construction of extractors using pseudo-random generators, *Proceedings of STOC'99*, pages 141–148.

[35] A. C. Yao and F. F. Yao, Dictionary look-up with one error, *Journal of Algorithms*, 25(1), 1997, pages 194–202.

[36] A. C. C. Yao, Should tables be sorted?, *Journal of the Association of Computing Machinery*, 28(3), 1981, pages 615–628.