Are bitvectors optimal?

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

ABSTRACT

We study the *static membership problem*: Given a set S of at most n keys drawn from a universe of size m, store it so that queries of the form "Is x in S?" can be answered quickly. 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 there is a randomized scheme with error ϵ that stores $O(\frac{n}{\epsilon^2}\log m)$ bits and answers queries using a *single* bitprobe. It is based on a family of sets with small intersections. If the error is required to be restricted to queries not in S, then we have a scheme that stores $O((\frac{n}{\epsilon})^2\log m)$ bits, answers queries with one bitprobe and works with probability of error less than ϵ. We also show that better schemes with one-sided error can be obtained if more probes are allowed.

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 upper and lower bounds.

1. INTRODUCTION

In this paper, we study the *static membership* problem: Given a subset S of at most n keys, store it so that queries of the form "Is x in S?" can be answered quickly. This is a fundamental data structure problem with a long history. Yao [27] 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 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 x. 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.

Background and related research. Fredman, Komlós and Szemerédi [14] 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 $\lceil\log\sum_{i\leq n}\binom{m}{i}\rceil$ on the number of bits used. For $n \leq m^{1-\Omega(1)}$, 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. Furthermore, if we do care about constant factors, and want to handle the case $n = m^{1-o(1)}$ as well, Fiat *et al.* [10] use non-oblivious hashing to produce an $O(1)$ worst case scheme that requires only logarithmic additional memory (improving the linear space upper bound of [14]), while Brodnik and Munro [3] and Pagh [22] have shown how to use space (in bits) that is within an additive low order term of $\lceil\log\sum_{i\leq n}\binom{m}{i}\rceil$.

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.* They were interested in the *average*-case upper bounds for this

---

[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, venkat}@tcs.tifr.res.in.

[4] part of this work was done while visiting BRICS at the University of Aarhus.

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 [7; 11; 12; 13; 19; 26], 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 all 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 [20; 17]. 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 x is in the set S); (b) schemes that are allowed to make two-sided errors (that is, errors are allowed for both 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 one cannot do better than deterministic schemes (see below).

1.1.1 Randomized schemes with two-sided errors

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 $\epsilon > 0$, 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 $x \in S$?" can be answered with error probability less than ϵ by a randomized algorithm which probes the memory at just one location determined by its coin tosses and the query element x.*

In contrast, deterministic schemes that answer queries using a single bitprobe need m bits of storage. 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 less than ϵ for *all* sets and *all* queries.

Until the present paper, every non-trivial solution to the membership problem we are aware of has been based on hashing [14; 27; 26]. 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 constructed by Erdős, Frankl and Füredi [8] 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 paper of Nisan [23], and Nisan and Wigderson [24], and have come to be known as Nisan-Wigderson designs; we refer to them as NW-designs. The properties of NW-designs are, unfortunately, not strong enough for our proof. No previously constructed set system we know has these properties, 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.

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 subconstant error probability and still use optimal space. We show that this limitation is unavoidable: if ϵ is made subconstant, then we must use more than $n \log m$ space.

THEOREM 2. *For any $\epsilon < 1$ and $\frac{n}{\epsilon} < m^{1/3}$, 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 [8].

1.1.2 Randomized schemes with one-sided errors

As stated above, we do not use NW-designs directly in our proof of Theorem 1, but instead use a related set system. If we use NW-designs directly, 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 $\epsilon > 0$, 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 $x \in S$?" can be answered with error probability less than ϵ by a randomized algorithm which makes a single bitprobe to the data structure. Furthermore, if $x \in S$, 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. *For any $\epsilon < 1$ and $\frac{n}{\epsilon} \leq m^{1/3}$, 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.

---

[1] The only exception we are aware of is a remark by Yao and Yao [26] 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.

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 and answers queries using one bitprobe. This result uses the explicit NW-designs (details in the full version).
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 that randomness does not help at all: Any one probe scheme must use m bits of storage (details in the full version).

Thus, no one-sided schemes make just one probe and use optimal space. But the space requirement can be reduced if we allow more probes.

THEOREM 5. *Fix any constant $\delta > 0$. There exists a constant t such that the following holds: there is an one-sided $\frac{1}{3}$-error randomized scheme that uses space $O(n^{1+\delta} \log m)$ and answers membership queries with at most t bitprobes.*

The proof of this result, in fact, composes two-sided error schemes similar to the one claimed in Theorem 1.

Connection with communication complexity. Theorems 1–4 can also be viewed in the communication complexity setting: Alice gets $x \in \{1, \dots, m\}$, Bob gets $S \subseteq \{1, \dots, m\}$ of size at most n, and Alice sends a single message to Bob after which Bob announces whether $x \in S$. Indeed, as was pointed out in [20], 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 $\phi(x)$ of any m-bit string x with at most n ones so that the length of the encoding is close to the *first order entropy* of x and so that any bit of x can be retrieved by looking at a few bits of $\phi(x)$. Thus, we are trying to construct a *locally decodable source code,* analogous to the locally decodable *channel* codes of [2].

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, $\binom{m}{n} \le 2^{nt}\binom{s}{nt}$.*

COROLLARY 1. *Let $\epsilon > 0, c \ge 1$ be any constants. There is a constant $\delta > 0$ so that the following holds. Let $n \le m^{1-\epsilon}$ 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 which answers queries using t bitprobes must use space $s = \Omega(m^{1/t})$ in the worst case. We show the existence of schemes which almost match the lower bound. There is a scheme that answers membership queries using t bitprobes and uses $O(s^4)$ bits where s is the lower bound implied by Theorem 6.

THEOREM 7.
1. *There is an non-adaptive scheme that stores sets of size at most n from a universe of size m using $O(m^{3/t} n \log m)$ bits and answers queries using t 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 finally 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, $(n \ge 2)$ of a universe of size m such that membership queries can be answered efficiently by two non-adaptive bitprobes uses space $s \ge m$ bits.*
2. *Let $m \ge 2$ and let $n = 2$. Then there is a scheme with two adaptive bitprobes and space $s = O(m^{3/4})$.*

The proof of Theorem 8 is omitted in this extended abstract. Details can be found in the full version.

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. FORMAL DEFINITIONS

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)-scheme is a map ϕ from the subsets of size at most n of $\{1, \ldots, m\}$ to $\{0,1\}^s$.

A deterministic (m, s, t)-*query scheme* is a family of m Boolean decision trees, $\{T_1, T_2, \ldots, T_m\}$, 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. All the edges are labeled by "0" or "1" indicating the bit stored in the parent node. The leaf nodes of every tree are marked "Yes" or "No". Each tree T_i induces a map from $\{0,1\}^s \to \{Yes, No\}$; this map will also be referred to as T_i.

An (n, m, s)-storing scheme ϕ and an (m, s, t)-query scheme $\{T_i\}$ together forms an (n, m, s, t)-*scheme* which solves the (n, m)-membership problem if $\forall S, x : T_x(\phi(S)) = Yes$ if and only if $x \in S$.

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

A randomized (m, s, t)-storing scheme is a probability distribution over deterministic (m, s, t)-query schemes. We execute the scheme on a query by picking a deterministic scheme according to the probability distribution and proceeding according to that. We'll say that a randomized scheme has *positive one-sided error* ϵ if, for $x \notin S$, the error probability on queries "Is x in S?" is 0, i.e., the answer "0" is always returned, while if $x \in S$, the error probability on the query "Is x in S?" is less than ϵ. Similarly a randomized scheme has *negative one-sided error* ϵ, if, for $x \in S$, the error probability on queries "Is x in S?" is 0, i.e. the answer "1" is always returned, while if $x \notin S$, the error probability on the query "Is x in S?" is less than ϵ.

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

A family of schemes is explicit if the associated storing and query schemes are explicit.

3. UPPER BOUNDS FOR RANDOMIZED SCHEMES

For randomized set membership schemes, it will be easier to describe the randomized scheme with one-sided error first; the scheme with two-sided error can then be seen as a generalization.

Every one-probe randomized scheme can be modeled as 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. Each edge of the bipartite graph is labeled by 0 or 1. An edge (u, v) is labeled 0[1] if on query "Is u in S?", we probe location v and say 'Yes' if 0[1] is present.

3.1 Randomized scheme with one-sided error

Proof of Theorem 3. We now describe the main ideas in construction of our randomized scheme. On query "Is u in S?", the query algorithm of our randomized scheme probes a random location in $\Gamma(u)$, the neighborhood of u uniformly and says 'Yes' if and only if the location probed contains 1. 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 in the memory, 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 less than ϵ, for all $u' \notin S$, less than $\epsilon|\Gamma(u')|$ locations in $\Gamma(u')$ can be colored 1. Thus, we get the following condition on the bipartite graph:

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

We are thus required to pick neighborhoods for vertices $u \in U$ so that the resulting bipartite graph satisfies (1). A NW-design has precisely this property. If we use the bounds from [8; 23; 24], then we can manage with a V of size at $O((\frac{n}{\epsilon})^2 \log m)$.

We will make use of the following result of Nisan and Wigderson [24] (alternatively, we could also use [8, Theorem 3.1].)

Lemma 2. *For any m, a and l, there exists a family of sets $F = \{S_1, \ldots, S_m\}$ such that:*

- $S_i \subseteq [t]$ *where* $t = O(e^{\log m/a} l^2/a)$.
- $|S_i| = l$.
- $|S_i \cap S_j| \leq a$.

The family F is called a (m, l, a)-design.

Choose $a = \log m$ and $l = \frac{n \log m}{\epsilon}$. Then we have a $(m, \frac{n \log m}{\epsilon}, \log m)$-design with $t = O(\frac{n^2 \log m}{\epsilon^2})$. We now use such a design to describe a randomized scheme.

Storing scheme. Given a set S of size at most n, do the following:

- Step 1: With each $i \in [m]$, associate the set S_i.
- Step 2: Suppose the set $S = \{i_1, \ldots, i_k\}, k \leq n$. Then construct a bitstring T of size t with 1's in all locations in $M = S_{i_1} \cup S_{i_2} \cup \ldots \cup S_{i_k}$ and 0 elsewhere.

Query scheme. Given a query $q \in [m]$, do the following:

- Step 1: Pick a random location l uniformly from S_q and probe the location l of the bitstring T.
- Step 2: If the location l has a 1, say *yes.* Otherwise say *no.*

If $q \in S$, then every location l in S_q has a 1 and hence we say *yes.* On the other hand, if $q \notin S$, then $|M \cap S_q| \leq n \log m$. Hence, the probability that we will say *yes* when we choose a random location in S_q is bounded by ϵ. This proves the theorem. ■

3.2 Randomized scheme with two-sided error

We now show a randomized scheme that that uses space within a constant factor of a sorted table or a hash table (for constant ϵ) and answers queries using a single bitprobe to the data structure.

Proof of Theorem 1. We again use the bipartite graph formulation described earlier. On query "Is u in S?", the query algorithm of our scheme probes a random location in $\Gamma(u)$, the neighborhood of u uniformly 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 coloring $\chi_S : V \to \{0,1\}$, such that for all $u \in S$, more than $(1-\epsilon)|\Gamma(u)|$ locations of $\Gamma(u)$ are colored 1, and for all $u' \notin S$, more than $(1-\epsilon)|\Gamma(u')|$ 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 $S \subseteq U$ ($|S| \leq n$). We first identify a sufficient condition on the set system for it to admit such coloring, and relate this sufficient condition to the following *expansion property* of the underlying bipartite graph $G(U,V,E)$:

$$\forall S \subseteq U \, (|S| \leq 2n) : |\Gamma(S)| > (1 - \frac{\epsilon}{2})| \bigcup_{u \in S} \Gamma(u)|.$$

Using a probabilistic argument, we show that such a bipartite graph exists. The counting argument we use is different from what is usually used for showing the existence of expander graphs.[2]

The proof goes through the following steps: We first show that a certain expanders exist. Using these expanders, we get a set system satisfying the ϵ-intersection- property. It is then shown that ϵ-intersection-property implies ϵ-two-colorability. Finally, we get a randomized scheme from an ϵ-two-colorable set system.

DEFINITION 3. *A (m,s,n,d,ϵ)-expander is a bipartite graph $A \times B$, $|A| = m, |B| = s$, in which the degree of each vertex v in A is d and the following condition holds: for any subset S of A of size at most $2n$,*

$$N(S) \geq (1 - \frac{\epsilon}{2})|S|d.$$

LEMMA 4. *A $(m, \lceil\frac{100n \log m}{\epsilon^2}\rceil, n, \lceil\frac{\log m}{\epsilon}\rceil, \epsilon)$-expander exists.*

PROOF. We show the existence of such expanders by a probabilistic argument. Let $A = \{u_1, \ldots, u_m\}$ and $B = \{1, \ldots, s\}$ where $s = \lceil\frac{100n \ln m}{\epsilon^2}\rceil$. For any vertex u_i in A, let us denote its set of neighbors in B by $N(u_i)$. For any subset S of A, $N(S) = \bigcup_{u_i \in S} N(u_i)$.

For each vertex u_i in A, we choose its set of neighbors in B, $N(u_i)$, by the following procedure: we first pick $d = \lceil\frac{\log m}{\epsilon}\rceil$ elements from B at random by sampling without replacement to form the set $N(u_1)$. We, then, independently pick $\lceil\frac{\log m}{\epsilon}\rceil$ elements from B at random by sampling without replacement to form the set $N(u_2)$ and so on.

We now compute the probability that there exists a subset S of A, $|S| = t$, $t \leq 2n$ such that $N(S) < (1 - \frac{\epsilon}{2})td$. Let us denote this event by E_t.

CLAIM 5. $\Pr[E_t] \leq 1/m^t$.

PROOF. We prove the claim under a stronger assumption that the sets of neighbors $\{N(u_i)\}_{i=1}^m$ are chosen by sampling with replacement.

Fix a set S, say, $S = \{u_1, u_2, \ldots, u_t\}$. With the choice of elements for $N(u_1), \ldots, N(u_t)$ where $N(u_j) = \{e_{(j-1)d+1}, \ldots, e_{jd}\}$, we associate td random variables $X_1, \ldots, X_{td}$ such that:

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

Then, it follows that

$$\Pr[E_t] \leq \binom{m}{t} \Pr[\sum_{i=1}^{td} X_i \geq \frac{\epsilon}{2} td].$$

Now we define td independent random variables $Y_1, \ldots, Y_{td}$ such that:

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

Then using the fact that for any i, $\Pr[X_i = 1] \leq \frac{\epsilon}{100}$, we get that:

$$\Pr[\sum_{i=1}^{td} X_i \geq \frac{\epsilon}{2} td] \leq \Pr[\sum_{i=1}^{td} Y_i \geq \frac{\epsilon}{2} td].$$

Using Chvátal's bound for the tails of hypergeometric distribution [4], we get:

$$\begin{aligned} \Pr[\sum_{i=1}^{td} Y_i \geq \frac{\epsilon}{2} td] &\leq (\frac{1}{10})^{\epsilon td} \\ &= (\frac{1}{10})^{t \log m} \\ &\leq \frac{1}{m^{2t}} \end{aligned}$$

So, we conclude that

$$\Pr[E_t] \leq \frac{1}{m^t}.$$

and the claim follows. □

Therefore,

$$\Pr[\bigwedge_{t=1}^{2n} \neg E_t] \geq 1 - \frac{1}{m}.$$

Hence the lemma follows.

Let Z be any set and let $F \subseteq 2^Z$ be a family of subsets of Z.

---

[2] 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 [1, p. 331] bounds the probability of this event by

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

For our choice of parameters, $d = \log m/\epsilon$ and $|V| = 100n \log m/\epsilon^2$, this quantity is not less than 1 (for example, consider the term with $i = n$), although in Lemma 4, we show that G has the required expansion property with high probability.

DEFINITION 6. *We say that the event* $E(F, i, \epsilon)$ *holds if for all collection* F' *of* i *sets,* $F' \subseteq F$,

$$|\{f \in F - F' \mid |f \cap (\bigcup_{f' \in F'} f')| \geq \epsilon|f|\}| < i.$$

CLAIM 7. *For any* $(m, \lceil \frac{100n \log m}{\epsilon^2} \rceil, n, \lceil \frac{\log m}{\epsilon} \rceil, \epsilon)$*-expander,* $\bigwedge_{i=1}^{n} E(\{N(u_i)\}_{u_i \in A}, i, \epsilon)$ *holds.*

PROOF. Suppose not. Then there exists a collection F' of i sets with the following property:

$$|F_1 = \{f \in F - F' \mid |f \cap (\bigcup_{f' \in F'} f')| \geq \epsilon|f|\}| \geq i.$$

Consider any $F_2 \subseteq F_1$, $|F_2| = i$. Let $S' = \{u_i \mid N(u_i) \in F' \cup F_2\}, |S'| = 2i$. Then S' does not satisfy E_{2i} which is a contradiction. $\square$

Suppose $S, T \subseteq P(Z)$ are disjoint and $\bigcup_{f \in S \cup T} f = Z$.

DEFINITION 8. (S, T) *has* ϵ*-intersection property if:*

- $\forall i \leq n, \forall S' \subseteq S, |S'| = i, |\{f \in T \mid |f \cap (\bigcup_{f' \in S'} f')| \geq \epsilon|f|\}| < i.$
- $\forall i \leq n, \forall T' \subseteq T, |T'| = i, |\{f \in S \mid |f \cap (\bigcup_{f' \in T'} f')| \geq \epsilon|f|\}| < i.$

DEFINITION 9. S, T *is* ϵ*-two colorable if there exists* $\chi : Z \to \{0, 1\}$ *such that:*

1. $\chi(z) = 1 \Rightarrow z \in \bigcup_{f \in S} f.$
2. $\chi(z) = 0 \Rightarrow z \in \bigcup_{f \in T} f.$
3. $\forall f \in S, |\chi^{-1}(0) \cap f| \leq \epsilon|f|.$
4. $\forall f \in T, |\chi^{-1}(1) \cap f| \leq \epsilon|f|.$

LEMMA 10. *If* $|S| \leq n$ *and* (S, T) *has* ϵ*-intersection property, then* (S, T) *is* ϵ*-two colorable.*

PROOF. We may assume that $|S| \leq |T|$. Otherwise, interchange the roles of S, T and $0, 1$. We prove the lemma by induction on $|S| + |T|$.

$$T' = \{f \in T \mid |f \cap (\bigcup_{f' \in S} f')| \geq \epsilon|f|\}$$

.
By the ϵ-intersection property, we have $|T'| < |S| \leq |T|$. By induction hypothesis, (S, T') is ϵ-two colorable. This implies that for $Z' = \bigcup_{f \in S \cup T'} f$, there exists $\chi' : Z' \to \{0, 1\}$ such that:

1. $\chi'(z) = 1 \Rightarrow z \in \bigcup_{f \in S} f.$
2. $\chi'(z) = 0 \Rightarrow z \in \bigcup_{f \in T'} f.$
3. $\forall f \in S, |{\chi'}^{-1}(0) \cap f| \leq \epsilon|f|.$
4. $\forall f \in T', |{\chi'}^{-1}(1) \cap f| \leq \epsilon|f|.$

Now define $\chi : Z \to \{0, 1\}$ by $\chi(z) = \chi'(z), z \in Z'$ and 0 otherwise.

CLAIM 11. χ *is an* ϵ*-two coloring for* (S, T).

PROOF. We just need to check that all the four conditions are satisfied.

- (1) holds by the property of χ'.
- Using the induction hypothesis and the fact that χ assigns 0 to $Z - Z'$, (2) holds.
- (3) follows directly from the induction hypothesis.
- If $f \in T'$, then $|\chi^{-1}(1) \cap f| \leq \epsilon|f|$ by property of χ'. If $f \in T - T'$, then by definition of T', $|\chi^{-1}(1) \cap f| \leq |f \cap (\bigcup_{f' \in S} f')| \leq \epsilon|f|$. Hence (4) holds.

Hence χ is an ϵ-two coloring for (S, T). $\square$

So, we have proved that if (S, T) has ϵ-intersection property, then it is ϵ-two colorable.

We now describe a randomized scheme for the membership problem.

The storing scheme. Given a set S of size at most n, we do the following:

Step 1: With $i \in [m]$, associate the set $N(u_i)$.

Step 2: Suppose without loss of generality, $S = \{1, 2, \ldots, k\}, k \leq n$. Obtain an ϵ-two coloring of $(\{N(u_i)\}_{i=1}^{k}, \{N(u_i)\}_{i=k+1}^{m})$. This is possible due to Lemma 10. Store this in a table $\overline{T}$ of size $s = \lceil \frac{100n \ln m}{\epsilon^2} \rceil$.

The query scheme. Given a query $q \in [m]$, we do the following:

Step 1: Pick a random location l from $N(u_q)$ and probe the location l of the table $\overline{T}$.

Step 2: If the location l has a 1, say *yes*. Otherwise, say *no*.

This randomized scheme satisfies the requirements of Theorem 1. ∎

3.3 Multiprobe randomized scheme with one-sided error

We now show that better schemes with one-sided error can be obtained if more probes are allowed.

Proof of Theorem 5. Just as one-probe algorithms were modeled using a bipartite graph, we can model algorithms that make t probes using query trees of depth t. For each element in U, we have a tree, Γ_u, of depth t whose nodes are labeled by elements of V (the set of locations in memory). To answer the query "Is u in S?," the algorithm picks a random leaf in the tree (compare this to picking a random element in $\Gamma(u)$ in the case of one-probe algorithms). If any of the locations whose label appears on the path from the root to this leaf contains a 1, then the algorithm says 'Yes', otherwise it says 'No'. Thus, our task now is to choose labeled query trees with the following property: for all $S \subseteq U$ ($|S| \leq n$), there is a coloring χ_S of V such that

1. For $u \in S$, for all leaves of Γ_u, there is at least one label on the path from the root to the leaf that is colored 1 by χ_S;

2. for $u' \notin S$, for most leaves of $\Gamma_{u'}$, all labels appearing on the path from the root to the leaf are colored 0 by χ_S.

The existence of such query trees with appropriate colorings is obtained by composing many query schemes of Theorem 1. We prove the following statement: For any fixed t, there is a randomized scheme with negative one-sided error $\epsilon = 1/3$ that uses $O(n^{1+\frac{5}{t-1}} \log m)$ bits of storage and answers queries using at most t bitprobes. To show this, we first define *query schemes* and show that query schemes with certain properties exist. This will imply the randomized scheme stated above.
We will need the following parameters in the proof.

- $\alpha = n^{-\frac{1}{t-1}}$.
- $d_i = \frac{i+1}{\alpha}$ for $0 \le i \le t-1$.
- $S = \frac{t!tn \log m}{\alpha^5}$.
- $\delta_i = \frac{\alpha}{i!}$ for $1 \le i \le t$.

We will also need the following definitions.

Definition 12. *A $(d_0, d_1, \dots, d_{t-1})$-tree is a tree of depth t such that:*

- *All leaves are at distance t from the root.*
- *The root has degree $d_0 \log m$.*
- *All the internal nodes (except the root) at depth i have degree d_i.*

Definition 13. *For any $(d_0, \dots, d_{t-1})$-tree T and $i \in [0, \dots, t]$, let $L_i[T]$ be the set of nodes at depth i of T, that is, the set of nodes at distance i from the root. For a node v at depth i, let $\Gamma(v)$ be the set of all its children at level $i+1$. For any subset S of nodes of T, $\Gamma(S) = \bigcup_{v \in S} \Gamma(v)$.*

Definition 14. *A query tree is a pair (T, f) where T is a $(d_0, \dots, d_{t-1})$- tree and $f : V(T) \to [S/t]$ such that f is one-to-one for $L_i[T]$ for $i \in [0, 1, \dots, t]$.*

Definition 15. *A query scheme is a collection of m query trees (T_q, f_q), one for each q in $[m]$.*

We will now prove the following claim.

Claim 16. *There exists a query scheme $\{(T_q, f_q)\}_{q \in [m]}$ such that: $\forall Q \in \binom{[m]}{n}$, $\forall i$ and $\forall \{S_q\}_{q \in Q}$ such that $S_q \subseteq \binom{L_i(T_q)}{\alpha^i |L_i(T_q)|}$,*

$$(\{f_q(\Gamma(S_q))\}_{q \in Q}, \{f_q(L_{i+1}(T_q))\}_{q \notin Q}) \textit{is } \alpha\textit{-two-colorable.}$$

Proof. We prove the existence of such query schemes by a probabilistic argument. We fix a level i and show that we can define f_q on $L_i(T_q)$ for every query $q \in [m]$ such that: $\forall Q \in \binom{[m]}{n}$ and $\forall \{S_q\}_{q \in Q}$ such that $S_q \subseteq \binom{L_{i-1}(T_q)}{\alpha^{i-1} |L_{i-1}(T_q)|}$,

$$(\{f_q(\Gamma(S_q))\}_{q \in Q}, \{f_q(L_i(T_q))\}_{q \notin Q}) \text{is } \alpha\text{-two-colorable.}$$

Since this is done independently for each depth i, the claim follows.

For depth i. For any $q \in [m]$, $f_q(v), v \in L_i(T_q)$ is chosen uniformly at random by sampling without replacement from $[\frac{S}{t}]$. This is done independently for each $q \in [m]$.
Now we note that:

- For any $q \in Q$, $|f_q(\Gamma(S_q))| = \frac{\log m}{\delta_i}$. we will refer to these sets as *good sets*. We note that the number of different possible good sets for each $q \in Q$ is $(\frac{1}{\alpha})^{\frac{(i-1)\alpha \log m}{\delta_{i-1}}}$.
- For any $q \notin Q$, $|f_q(L_i(T_q))| = \frac{\log m}{\alpha^{i-1}\delta_i}$. We will refer to these sets as *bad sets*.

Definition 17. *Given sets $S_1, S_2, \dots, S_l$ and another set T, we say that $S_1, S_2, \dots, S_l$* attacks *T if:*

$$|\bigcup_{i=1}^{l} S_i \cap T| \ge \alpha |T|.$$

Definition 18. *We say that $E(l, \alpha)$ holds if there exists l good sets $S_1, S_2, \dots, S_l$ and $l' (= \alpha^{i-1} l)$ bad sets $T_1, \dots, T_{l'}$ such that $S_1, S_2, \dots, S_l$* attacks *T_j for $j = 1, 2, \dots, l'$. We say that $E'(k, \alpha)$ holds if there exists k bad sets $T_1, T_2, \dots, T_k$ and $\frac{k}{2}$ good sets $S_1, \dots, S_{\frac{k}{2}}$ such that $T_1, T_2, \dots, T_k$* attacks *S_i for $i = 1, 2, \dots, \frac{k}{2}$.*

Given $l \le n$ good sets $S_i, i = 1, \dots, l$ and a fixed bad set T, by Chvátal's bound [4], we have:

$$\Pr[S_1, \dots, S_l \text{ attacks } T] \le (\alpha^3)^{\frac{\log m}{\alpha^{i-2}\delta_i}}.$$

Therefore,

$$\begin{aligned} \Pr[E(l, \alpha)] &\le \binom{m}{\alpha^{i-1} l} \binom{m}{l} (\frac{1}{\alpha})^{\frac{(i-1)\alpha \log m}{\delta_{i-1}} l} (\alpha^3)^{\frac{\log m}{\alpha^{i-2}\delta_i} \alpha^{i-1} l} \\ &\le \frac{1}{m^i}. \end{aligned}$$

Therefore,

$$\Pr[\bigwedge_{l=1}^{n} \neg E(l, \alpha)] > \frac{1}{2}.$$

Similarly, Given $k \le \alpha^{i-1} n$ bad sets $S_1, \dots, S_k$ and a fixed good set T, by Chvátal's bound [4], we have:

$$\Pr[S_1, \dots, S_k \text{ attacks } T] \le (\alpha^3)^{\frac{\alpha \log m}{\delta_i}}.$$

Therefore,

$$\begin{aligned} \Pr[E'(k, \alpha)] &\le \binom{m}{\frac{k}{2}} \binom{m}{k} (\frac{1}{\alpha})^{\frac{(i-1)\alpha \log m}{\delta_{i-1}} k} (\alpha^3)^{\frac{\alpha \log m}{\delta_i} \frac{k}{2}} \\ &\le \frac{1}{m^i}. \end{aligned}$$

Therefore,

$$\Pr[\bigwedge_{k=1}^{\alpha^{i-1} n} \neg E'(k, \alpha)] > \frac{1}{2}.$$

Now take a choice of a query scheme $\{(T_q, f_q)\}_{q \in [m]}$ s.t. $\bigwedge_{l=1}^{n} \neg E(l, \alpha)$ and $\bigwedge_{k=1}^{\alpha^{i-1} n} \neg E'(k, \alpha)$ holds. This choice has the α-intersection property and hence is α-two-colorable by Lemma 10. Thus the claim follows for depth i.

We now make an important observation that since $\alpha^{t-1}n < 1$ at depth t, the good sets do not attack any bad set. Hence the *coloring at depth t is a coloring with all the good sets filled with ones.* $\square$

Now we describe the storing scheme and the query scheme using the claim above.

Storing scheme. The scheme stores t tables $N_1, \ldots, N_t$ of size $\frac{S}{t}$ each, one corresponding to each depth i of the query trees. The total space used is S. We now describe the $\{0,1\}$-coloring stored in any table N_i. We start with table N_1 and then describe the storage scheme for table N_i for any i.
Given a set Q, the scheme does the following:

Table 1. Consider the set system $\{L_1(T_q)\}_{q\in[m]}$. Using the claim above, the storing scheme will α- two-color this set system and store it in table N_1.

$\vdots$

Table i. Suppose we have described the scheme up to table N_{i-1}. We now explain what the scheme does for table N_i: let $S_q = \{v \in L_{i-1}(T_q) \mid N_{i-1}[f_q(v)] = 0\}$ for $q \in Q$. We consider $(\{f_q(\Gamma(S_q))\}_{q\in Q}, \{f_q(L_i(T_q))\}_{q\notin Q})$. This is α-two-colorable by the claim above. We store this coloring in table N_i.

Query scheme. Given a query q, we pick a random leaf l of T_q and consider all the nodes of T_q appearing on the path from the root to l. When we reach the node v_i on this path at depth i, we probe the location $f_q(v_i)$ in table N_i and check if it is a 1. If there exists some j such that the Table $N(j)$ has a 1 in location $f_q(v_j)$, we say *yes*. Otherwise, we say *no*. ■

4. LOWER BOUNDS FOR RANDOMIZED SCHEMES

We now prove lower bounds for randomized schemes that answers queries using one bitprobe. These lower bounds come close to the upper bounds we proved in the previous section. We have not been able to prove lower bounds for randomized schemes with one-sided error that use more than one bitprobe.

4.1 Randomized schemes with one-sided error

Proof of Theorem 4. We again recall the bipartite graph formulation described earlier. To prove the lower bound, we make the following observation: we can assume, without loss of generality, that the query algorithm on query "Is u in S?" chooses a random location in $\Gamma(u)$ and says 'Yes' if the location chosen contains a 1. Then, it follows that no $\Gamma(u)$ is included in the union of the neighborhoods of $\lfloor\frac{n}{\epsilon}\rfloor$ other vertices in U. That is, the family $\{\Gamma(u)\}_{u\in U}$ is an r-cover-free family with $r = \lfloor\frac{n}{\epsilon}\rfloor$. We are, thus, reduced to the problem of determining the minimum size for V, so that it can accommodate an r-cover-free family. The lower bound then follows immediately from the result of Dyachkov and Rykov.

Suppose there exists a randomized scheme with negative one-sided error ϵ which uses s bits of storage and answers queries using one bitprobe. We will show that $s \geq \Omega(\frac{n^2\log m}{\epsilon^2\log(\frac{n}{\epsilon})})$.

Definition 19. *A family of sets $F \subseteq 2^{[m]}$ is r-cover-free if $f_0 \not\subseteq f_1 \cup \ldots \cup f_r$ for all distinct $f_0, \ldots, f_r \in F$.*

> **Theorem (Dyachkov and Rykov [6])** If F is an r-cover-free family of sets and $r \leq m^{1/3}$, then
>
> $$|\bigcup_{f\in F} f| \geq \frac{r^2}{2\log r + O(1)}\log|F|.$$
>
> [Several proofs of similar bounds are known [6; 25; 15]; the one due to Füredi [15] is the most accessible.]

Consider any randomized scheme with negative one-sided error ϵ which uses s bits of storage and answers queries with one bitprobe. The query scheme assigns to each possible query q in U, a set $S_q \subseteq [s] \times [0,1] \times \{0,1\}$, $S_q = \{(l_1,p_1,b_1),(l_2,p_2,b_2),\ldots,(l_r,p_r,b_r) \mid \sum_1^r p_i = 1\}$ such that on query q, the location l_i is picked with probability p_i and the scheme says *yes* if bit b_i is found in location l_i and says *no* if $\overline{b_i}$ is in location l_i.
The next claim proves that, without loss of generality, we can assume that a 1 indicates presence and a 0 indicates absence.

Claim 20. *We can assume, without loss of generality, that for every query q, $b_i = 1$ for all i in S_q.*

Proof. We show that from any scheme A with space s, we can obtain a new scheme A' with space requirement $s' = 2s$ that satisfies the above property. The scheme A' encodes a bit i of scheme A by two bits $2i$ and $2i+1$; the bit $i = 0$ by $2i = 0$ and $2i+1 = 1$ and the bit $i = 1$ by $2i = 1$ and $2i+1 = 0$.
For any query q, we form a set S'_q by the following rule:

$$(l_i,p_i,b_i) \rightarrow (2l_i,p_i,1)\ if\ b_i = 1.$$

$$(l_i,p_i,b_i) \rightarrow (2l_i+1,p_i,1)\ if\ b_i = 0.$$

This proves the claim. $\square$

Now, consider the set $L_q = \{l_i \mid (l_i,p_i,1) \in S_q\}$ for every q in U.

Claim 21. *The sets $L_q, q \in U$, form a $\lfloor\frac{n}{\epsilon}\rfloor$- cover-free family.*

Proof. Suppose not. Without loss of generality, suppose

$$L_0 \subseteq L_1 \cup L_1 \cup \ldots \cup L_{\lfloor\frac{n}{\epsilon}\rfloor}.$$

$$\begin{aligned} Let\ G_1 &= L_1 \cup L_2 \cup \ldots \cup L_n \\ G_2 &= L_{n+1} \cup L_{n+2} \cup \ldots \cup L_{2n} \\ &\vdots \\ G_i &= L_{(i-1)n+1} \cup L_{in+2} \cup \ldots \cup L_{in} \\ &\vdots \end{aligned}$$

Let $L_{0k} = L_0 \cap G_k$ for all k and $P(L_{0k}) = \{\sum p_i \mid l_i \in L_{0k}\}$. Then there exists j such that $P(L_{0j}) \geq \epsilon$.
Now if we store $S = G_j$ and ask the query "Is 0 in S?", we say *yes* with probability more than ϵ. This is a contradiction. □

Hence we have a $\lfloor \frac{n}{\epsilon} \rfloor$-cover-free family of cardinality m. Using the result of Dyachkov and Rykov, we get:

$$\exp(O(s \ln(\lfloor \frac{n}{\epsilon} \rfloor)/(\lfloor \frac{n}{\epsilon} \rfloor)^2)) \geq m.$$

This implies that $s \geq \Omega(\frac{n^2 \log m}{\epsilon^2 \log(\frac{n}{\epsilon})})$. ■

4.2 Randomized schemes with two-sided error

For the lower bound, we again use the theorem of Dyachkov and Rykov, cited above. Since the error probability is less than ϵ, we conclude (by restricting ourselves to the case $|S| = 1$) that $|\Gamma(u) \cap \Gamma(u')| < 2\epsilon d$, for all $u, u' \in U$ $(u \neq u')$. Thus, the system $\{\Gamma(u)\}_{u \in U}$ is $(\frac{1}{2\epsilon})$-cover-free, and we get a lower bound of $\Omega((\frac{1}{\epsilon^2 \log(1/\epsilon)}) \log m)$ on the size of $|V|$. Although this gives the required dependence on ϵ, the dependence on n is missing: every scheme needs size $\Omega(n \log m)$ bits of storage. To get the lower bound which depends on ϵ and n simultaneously, we need to use *upper bounds* on r-cover-free families together with the lower bound cited above.

Proof of Theorem 2. Consider any randomized scheme with two-sided error that answers queries with one bitprobe. Suppose it uses a table T with s bits of storage.
We start by constructing a $\lfloor \frac{1}{\epsilon} \rfloor$-cover free family F of $[m]$ using subsets of size n. Using the result by Erdős et al [8, Theorem 3.1], we get:

$$|F| \geq \frac{\binom{m}{\epsilon n}}{\binom{n}{\epsilon n}^2}.$$

It is easy to observe that we can always construct a $\lfloor \frac{1}{\epsilon} \rfloor$-cover free family of size $m - n + 1$ (the ith set $S_i = \{1, 2, \ldots, n-1\} \cup \{n - 1 + i\}$ for $1 \leq i \leq m - n + 1$). Hence, we can assume that:

$$|F| \geq \max\{\frac{\binom{m}{\epsilon n}}{\binom{n}{\epsilon n}^2}, m - n + 1\}.$$

Now corresponding to each $S \in F$, we define T_S to be the locations of the table T that are set to 1 when S is required to be stored. Let us define Q_s to be the set of locations we are likely to probe for a 1 if s is the query string.

Claim 22. *$\{T_S \mid S \in F\}$ is an $\lfloor \frac{1}{10\epsilon} \rfloor$-cover free family.*

Proof. Suppose not. Then there exists $S_0, S_1, \ldots, S_{\lfloor \frac{1}{10\epsilon} \rfloor} \in F$ such that

$$T_{S_0} \subseteq T_{S_1} \cup T_{S_2} \cup \ldots \cup T_{S_{\lfloor \frac{1}{10\epsilon} \rfloor}}.$$

Since $S_0 \not\subseteq S_1 \cup \ldots \cup S_{\lfloor \frac{1}{10\epsilon} \rfloor}$, there exists $s \in S_0$ such that $s \notin S_1 \cup \ldots \cup S_{\lfloor \frac{1}{10\epsilon} \rfloor}$. Consider Q_s. Clearly $(1-\epsilon)$-fraction (by probability) of Q_s must be contained in T_{S_0}. Hence, we get that $(1-\epsilon)$-fraction (by probability) of Q_s is covered by $T_{S_1} \cup T_{S_2} \cup \ldots \cup T_{S_{\lfloor \frac{1}{10\epsilon} \rfloor}}$. Therefore, there exists S_i such that T_{S_i} covers $\frac{1-\epsilon}{\frac{1}{10\epsilon}}$ fraction of Q_s. This fraction is more than ϵ for $\epsilon < 1/2$.
Now suppose S_i is stored and we ask the query "Is $s \in S_i$?" we make an error of more than ϵ which is a contradiction. This proves the claim. □

Now since $\{T_S \mid S \in F\}$ is a $\lfloor \frac{1}{10\epsilon} \rfloor$-cover free family, using the result by Dyachkov and Rykov, we have:

$$\begin{aligned} s &\geq \frac{1}{(10\epsilon)^2 \ln(\frac{1}{10\epsilon})} \ln\{\max(\frac{\binom{m}{\epsilon n}}{\binom{n}{\epsilon n}^2}, m - n + 1)\}. \\ &= \Omega\left(\frac{n \log m}{\epsilon \log(\frac{1}{\epsilon})}\right). \end{aligned}$$

■

5. DETERMINISTIC SCHEMES

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

Proof of Theorem 6. Take a set S of size n. Store it as the s-bit string $\phi(S)$. For each element $x \in S$, Let $T_{S,x} = \{(l_1, b_1), ..., (l_t, b_t)\}$ where l_i is the i'th location probed in $\phi(S)$ and b_i is the answer on query "Is $x \in S$?. Note that by storing it as a set, we want to forget the actual order in which these locations were probed.
Now, let $T_S = \bigcup_{x \in S} T_{S,x}$. Clearly $|T_S| \leq nt$.
We now observe that T_S has to be different for different S. So, $\binom{m}{n} \leq 2^{nt}\binom{s}{nt}$. ■

As a corollary of the tradeoff result, we get that the bitvector scheme is optimal for one bitprobe schemes and the FKS scheme is optimal for schemes that use $O(n \log m)$ space. We now study deterministic schemes in general.

Proof of Theorem 7, Part (1). We can show that a $(m, \frac{100kn \log m}{\epsilon^2}, n, \frac{\log m}{\epsilon \log k}, \epsilon)$-expander exists using a calculation similar to the one in Lemma 4. We now choose $\epsilon = 1/3$ and $k = m^{3/t}$. From the proof of Theorem 1, we can infer the following: instead of probing a random element of the set associated with query x, if we probe all the locations in the set and say *yes* if 2/3 fraction of the locations contain a 1 and say *no* otherwise, we get a non-adaptive scheme that uses $O(m^{3/t} n \log m)$ bits of storage and answers queries using t bitprobes.

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 thus 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 [14]. 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 a 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 x, do the following.

- Step 1: Read the prime p. This requires $O(\log n + \log\log m)$ bitprobes.
- Step 2: Find x mod p. Now, check if there is an element y in T such that x mod $p = y$ mod 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 x into t blocks and check if $x = y$ block by block. This requires t bitprobes.

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

6. REFERENCES

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

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

[3] A. Brodnik and J.I. Munro, Membership in constant time and minimum space, *Lecture Notes In Computer Science*, 855:72–81, 1994.

[4] V. Chvátal, The tail of hypergeometric distribution, *Discrete Mathematics*, 25:285–287, 1979.

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

[6] A.G. Dyachkov and V.V. Rykov, Bounds on the length of disjunctive codes, *Problemy Peredachi Informatsii,*18, No.3(1982), 7–13 [Russian].

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

[8] 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*, 51:79–89, 1985.

[9] A. Fiat and M. Noar, Implicit $O(1)$ probe search, *SIAM Journal of Computing,* 22:1-10, 1993.

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

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

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

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

[14] 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):538–544, 1984.

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

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

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

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

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

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

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

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

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

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

[25] M. Ruszinkó, On the upper bound of the size of r-cover-free families, *Journal of Combinatorial Theory, Series A,*, 66:302–310, 1994.

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

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