Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 379 (2007) 405–417
Theoretical Computer Science
www.elsevier.com/locate/tcs

The cell probe complexity of succinct data structures☆

Anna Gál[a], Peter Bro Miltersen[b,*]

[a] *Department of Computer Science, University of Texas at Austin, Austin, TX 78712-1188, USA*
[b] *Department of Computer Science, University of Aarhus, Denmark*

Abstract

We consider time-space tradeoffs for static data structure problems in the cell probe model with word size 1 (the bit probe model). In this model, the goal is to represent n-bit data with $s = n + r$ bits such that queries (of a certain type) about the data can be answered by reading at most t bits of the representation. Ideally, we would like to keep both s and t small, but there are tradeoffs between the values of s and t that limit the possibilities of keeping both parameters small.

In this paper, we consider the case of *succinct* representations, where $s = n + r$ for some *redundancy* $r \ll n$. For a Boolean version of the problem of polynomial evaluation with preprocessing of coefficients, we show a lower bound on the redundancy–query time tradeoff of the form

$$(r + 1)t \geq \Omega(n/\log n).$$

In particular, for very small redundancies r, we get an almost optimal lower bound stating that the query algorithm has to inspect almost the entire data structure (up to a logarithmic factor). We show similar lower bounds for problems satisfying a certain combinatorial properties of a coding theoretic flavor, and obtain $(r + 1)t \geq \Omega(n)$ for certain problems. Previously, no $\omega(m)$ lower bounds were known on t in the general model for explicit Boolean problems, even for very small redundancies.

By restricting our attention to *systematic* or *index* structures ϕ satisfying $\phi(x) = x \cdot \phi^*(x)$ for some map ϕ^* (where $\cdot$ denotes concatenation), we show similar lower bounds on the redundancy–query time tradeoff for the natural data structuring problems of Prefix Sum and Substring Search.

Keywords: Succinct data structures; Cell probe complexity; Polynomial evaluation with preprocessing

1. Introduction

In the cell probe model (e.g. [1,4–6,8,9,11,23–26,31]), a static data structure problem is given by a map

$$f : \{0, 1\}^n \times \{0, 1\}^m \to \{0, 1\}^q,$$

☆ An early version of this paper [A. Gál, P.B. Miltersen, The cell probe complexity of succinct data structures, in: Proc. of 30th International Colloquium on Automata, Languages and Programming, ICALP'03, pp. 332–344] appeared in the Proceedings of ICALP (2003).

* Corresponding author. Tel.: +45 8942 5600; fax: +45 8942 5601.
E-mail addresses: panni@cs.utexas.edu (A. Gál), bromille@daimi.au.dk (P.B. Miltersen).

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2007.02.047

where $\{0,1\}^n$ is a set of possible data to be stored, $\{0,1\}^m$ is a set of possible queries and $f(x,y)$ is the answer to question y about data x. A case of particular interest to us is a *Boolean* problem where $q = 1$.

For natural problems, we have $m \ll n$: the question we pose to the database is much shorter than the database itself. Examples of natural static data structure problems include:

- **Substring Search**: Given a string x in $\{0,1\}^n$, we want to store it in a data structure so that given a query string y of length m, we can tell whether y is a substring of x by inspecting the data structure. This problem is modeled by the function f defined by $f(x,y) = 1$ iff y is a substring of x.
- **Prefix Sum**: Given a bit vector $x \in \{0,1\}^n$, store it in a data structure so that queries "What is $(\sum_{i=1}^{k} x_i) \bmod 2$?" can be answered. This problem is modeled by the function f defined by $f(x,y) = (\sum_{i=1}^{v_y} x_i) \bmod 2$ where y is the binary representation of the integer v_y.

For Substring Search, the data to be stored, both the queries and the answers are bit strings, as our framework requires. The only reason for this requirement of our framework is to make our discussion about current lower bound techniques and their limitations clear. In particular, we want the parameter n to always refer to the number of bits of the data to be stored and the parameter m to always refer to the number of bits of a query, and the output of the query to be a single bit. In general, we don't necessarily expect the data we want to store to be bit strings, but an arbitrary encoding as bit strings may take care of this, as in the following example.

- **Membership**: Given a set S of k binary strings, each of length m, store S as a data structure so that given a query $y \in \{0,1\}^m$, we can tell whether $y \in S$. To make this problem fit into the framework above, the function f would be defined by letting $n = \lceil \log_2 \binom{2^m}{k} \rceil$ and fixing, in some arbitrary way, a compact encoding of k-sets as n-bit strings and letting $f(S,y) = 1$ iff $y \in S$.

A point which is often overlooked but which we want to emphasize is that the framework captures not only the classical "storage and retrieval" static data structure problems but also arbitrary problems of dealing with preprocessed information, such as the classical algebraic problem of polynomial evaluation with preprocessing of coefficients ([20, pp. 470–479], see also [24]):

- **Polynomial Evaluation**: Let g be a univariate polynomial over $\mathbf{F}$, where $|\mathbf{F}| = 2^k$ and g is of degree $\leq d$. Store g as a memory image so that queries "What is $g(x)$?" can be answered for any $x \in \mathbf{F}$.

Having specified the problems we want to consider, we now turn to their solutions.

Definition 1 (*Elias and Flower*). In the *bit probe model* (the cell probe model with word size 1), a *solution* with *space* s and *time* t to a problem f is given by

1. a storage scheme $\phi : \{0,1\}^n \to \{0,1\}^s$, and
2. a query algorithm q so that $q(\phi(x), y) = f(x,y)$. The time t of the query algorithm is its bit probe complexity, i.e., the worst case number of bits it reads in $\phi(x)$.

The quantity $r = s - n$ is called the *redundancy* of the storage scheme.

When discussing what upper and lower bounds can be obtained, we want to restrict our attention to Boolean problems. All problems above are Boolean, except for Polynomial Evaluation. We can make a "Boolean version" of Polynomial Evaluation by letting $n = (d+1)k$, $m = k + \lceil \log k \rceil$, fixing an arbitrary compact encoding of polynomials and field elements as bit strings and letting $f(g, x \cdot y) = v_y$th bit of $g(x)$, where y is the binary representation of v_y and $\cdot$ denotes concatenation. A similar transformation can be made for any other non-Boolean problem. Note that when we model a non-Boolean problem with q bits in the query output as a Boolean problem above, the upper and lower bounds on the query times of the original problem and the Boolean version only agree up to a factor of q. That is, if the Boolean problem can be solved with redundancy r and query time t, the non-Boolean can be solved with redundancy r and query time qt. For the case of polynomial evaluation, $q = \log_2 n$. Our focus on Boolean problems in the current discussion is due to the fact that when understanding the power of various lower bound techniques (see below), it seems useful to first ensure that the lower bounds we consider cannot rely on the trivial information theoretic lower bound: if a query must answer q bits and all 2^q bit strings are possible such answers, one needs to read q bits in the data structure in the worst case to get a correct answer.

Considering first upper bounds, every Boolean problem possesses two trivial solutions:

1. The solution of explicitly storing the answer to every query. This solution has space $s = 2^m$ and time $t = 1$.
2. The solution of storing the data verbatim and reading the entire data when answering queries. This solution has space $s = n$ and time $t = n$ (as we only charge for reading bits in $\phi(x)$, not for computation).

The study of cell probe complexity concerns itself with the tradeoff between s and t that may be obtained by solutions somewhere between the two extremes defined by the trivial solutions. Such solutions may be quite non-trivial and depend strongly on the problem considered. A *polynomial* solution satisfies $s = n^{O(1)}$ and $t = m^{O(1)}$. For instance, perfect hashing schemes form solutions to Membership with $s = O(n)$ and $t = O(m)$ and even $s = n + o(n)$ and $t = O(m)$ [13,7,30]. Substring Search also admits an $s = O(n)$, $t = O(m)$ solution [16] and recently a solution with $s = n + o(n)$ and $t = m^{O(1)}$ was constructed [17], but no solution with $s = n + o(n)$ and $t = O(m)$ is known. For a problem such as the Boolean version of Polynomial Evaluation (and many natural data structure problems, such as partial match type problems [4,6,9]), we know of no solution with $s = n^{O(1)}$, $t = m^{O(1)}$. *Thus, a main concern is to prove that no such solution exists.* For $s = O(n)$, lower bounds of the form $t = \Omega(m)$ may be obtained for explicit and natural problems by simple counting arguments [8]. For $s = n^{O(1)}$, we can do almost equally well: Lower bounds of the form $t = \Omega(m/\log n)$ can be obtained using communication complexity [25]. But no very good (i.e., $\omega(m)$) lower bounds are known on t for *any* explicit Boolean problem f for the case of $s = O(n)$ or $s = n^{O(1)}$, even though counting arguments prove the existence of (non-explicit) problems f with lower bounds of the form $t = \Omega(n)$, even for $m \approx (\log n)^2$ [23]. Thus, it is consistent with our current knowledge that solutions with $s = O(n)$ and $t = O(m)$ exist for *all* explicit (e.g., all exponential time computable) Boolean problems, though it is certainly a generally believed conjecture that this is not the case! One should note that Demaine and Lopez-Ortiz [10] prove $\omega(m)$ lower bounds on t for a non-Boolean problem (in their problem, $q = \Theta(m)$) in the systematic model (see the definition below). Since their bound holds for $r = O(n)$, it also implies $\omega(m)$ lower bounds on t in the general (non-systematic) model. However, their technique does not seem to imply $\omega(m)$ lower bounds for Boolean versions of the problem.

Given our lack of tools strong enough to show statements such as $s = O(n) \Rightarrow t = \omega(m)$ for explicit Boolean problems, it seems appropriate to lower our ambitions slightly and try to show such lower bounds for t for *any* non-trivial value of s. Achieving such goals is well in line with the current trend in the theoretical as well as practical studies of data structures (e.g., [22,7,30,17,32]) of focusing on *succinct* data structures where $s = n + r$ for some redundancy $r \ll n$, i.e., on structures whose space requirement is close to the information theoretic minimum. Restricting our attention to such succinct structures by no means trivializes obtaining the lower bounds we want to show. For instance, it is open (and remains open after this work) whether a solution with $r = 0$ and $t = O(m)$ exists for the Membership problem. However, in this paper, we show that for certain explicit (polynomial time computable) Boolean problems it *is* possible to show lower bounds of the form $t = \omega(m)$ and even $t = \Omega(n)$ for structures with a sufficiently strong upper bound on r.

We consider the polynomial evaluation problem described above, over the field $\mathbf{F} = \mathrm{GF}(2^k)$ (the unique finite field with 2^k elements).

Theorem 2. *Let k, d be positive integers so that $d < 2^k/3$. Let $\mathbf{F} = \mathrm{GF}(2^k)$ and let $n = (d+1)k$. Let a storage scheme $\phi : \{f \mid f$ a univariate polynomial over $\mathbf{F}$, $\mathrm{degree}(f) \leq d\} \to \{0,1\}^{n+r}$ and associated query scheme for "What is $f(x)$?", $x \in \mathbf{F}$ with bit probe complexity t be given. Then,*

$$(r+1)t \geq n/3.$$

In particular, for very small redundancies, we get an almost optimal lower bound stating that the query algorithm has to inspect almost the entire data structure. The theorem concerns the (more natural) non-Boolean version of the polynomial evaluation problem. A lower bound of $(r+1)t \geq n/3k$ for the Boolean version of polynomial evaluation we defined previously immediately follows. The bound stated in the abstract follows by letting $d = 2^{\Theta(k)}$.

The proof of Theorem 2 (presented in Section 2.1) is based on the fact that the problem of polynomial evaluation hides an error-correcting code: The strings of query answers for each possible set of data (i.e., each polynomial) form the Reed–Solomon code. We can generalize Theorem 2 to any problem hiding an error-correcting code in a similar way (see Theorems 5 and 6 in Section 2.1). However, not many natural data structuring problems contain an error-correcting code in this way. In Section 2.2, we introduce a parameter of data structuring problems called *balance* and, using the sunflower lemma of Erdős and Rado, show that for problems having constant balance, we get a lower bound of the form $t(r+1)^2 \geq \Omega(n)$ (Theorem 10). A problem hiding a good error-correcting code in the way described

above has constant balance, but the converse statement is not necessarily true. Hence Theorem 10 has the potential to prove lower bounds on a wider range of problems than Theorems 5 and 6, though we do not have any natural data structuring problems as examples of this at the moment.

The results above are based on combinatorial properties of a coding theoretic flavor of the problems f to be solved. We don't know how to prove similar lower bounds for natural storage and retrieval problems such as Substring Search. However, we get a natural restriction of the cell probe model by looking at the case of *systematic* or *index* structures. These are storage schemes ϕ satisfying $\phi(x) = x \cdot \phi^*(x)$ for some map ϕ^* (where $\cdot$ denotes concatenation), i.e, we require that the original data is kept "verbatim" in the data structure. We refer to $\phi^*(x)$ as the *index part* of $\phi(x)$. We emphasize that such a restriction only makes sense relative to a particular fixed encoding of input data as a Boolean string. However, for many problems there is a canonical such encoding. For instance, for Prefix Sum and Substring Search, the data to be stored is explicitly given as a bit string in the first place. The restriction is practically motivated: the original data may reside in read-only memory or belong to someone else, or it may be necessary to keep it around for reasons unrelated to answering the queries defined by f. For more discussion, see, e.g. Manber and Wu [22]. In the systematic model, we prove a tight lower bound for Prefix Sum (in fact, we show that the lower bound is implicit in work of Nisan, Rudich and Saks [29]), and we prove a lower bound for Substring Search (no attempt was made to optimize the constants in the statement concerning Substring Search).

Theorem 3. *Consider Prefix Sum with a parameter n. For a systematic scheme with redundancy r, $t = \Theta(n/(r+1))$ bit probes are necessary and sufficient.*

Theorem 4. *Consider Substring Search with parameters n, m so that $2\log_2 n + 5 \leq m \leq 5\log_2 n$. For any systematic scheme solving it with redundancy r and bit probe complexity t, we have $(r+1)t \geq \frac{1}{800} n/\log_2 n$.*

Both proofs are presented in Section 3. We are aware of one paper previous to this one where lower bounds of the form $t = \omega(m)$ were established for succinct, systematic data structures: Demaine and Lopez-Ortiz [10] show such a lower bound for a variation of the Substring Search problem. In their variation, a query does not just return a Boolean value, but an index of an occurrence of the substring if it does indeed occur in the string, i.e., for their variation $q \approx \log n$. For this variation, they prove [10, Corollary 3.1] the following lower bound for a value of m which is $\Theta(\log n)$ as in our bound:

$$t = o(m^2/\log m) \Rightarrow (r+1)t = \Omega(n \log n).$$

Thus, they give a lower bound on the query time even with *linear* redundancy, which our method cannot. On the other hand, their method cannot give lower bounds on the query time better than $\Omega(m^2/\log m)$, even for very small redundancies, which our method can. Furthermore, as discussed above, our lower bound applies to the Boolean version of the problem.

2. Lower bounds for non-systematic structures

2.1. Polynomial evaluation and error-correcting codes

The general randomized construction described in the following will be useful in all proofs in this section. Let a storage scheme ϕ with space s and an associated query scheme with bit probe complexity t be given. Let $1 \leq \ell < s$.

We make a randomized construction of another storage scheme ϕ' by randomly removing ℓ bits of the data structures $\phi(x)$ of the storage scheme ϕ. That is, we pick $S \subset \{1, .., s\}$ of size ℓ at random and let $\phi'(x) = \phi(x)$ with bits in positions $i \in S$ removed. Thus, $\phi'(x) \in \{0,1\}^{s-\ell}$. We make an associated query scheme for ϕ' by simulating the query scheme for ϕ, but whenever a bit has to be read that is no longer there, we immediately answer "Don't know". Clearly, if we use our new storage scheme ϕ' and the associated query scheme, we will, on every query, either get the right answer or the answer "Don't know".

Note that the query algorithm for ϕ' depends on S, and S does not have to be stored.

Theorem 2. *Let k, d be positive integers so that $d < 2^k/3$. Let $\mathbf{F} = \mathrm{GF}(2^k)$ and let $n = (d+1)k$. Let a storage scheme $\phi : \{f \mid f$ a univariate polynomial over $\mathbf{F}$, $\mathrm{degree}(f) \leq d\} \to \{0,1\}^{n+r}$ and associated query scheme for "What is $f(x)$?", $x \in \mathbf{F}$ with bit probe complexity t be given. Then,*

$$(r+1)t \geq n/3.$$

Proof. Let a storage scheme ϕ with redundancy r and an associated query scheme with bit probe complexity t be given. Let $s = n + r$.

Assume to the contrary that the scheme satisfies $(r + 1)t < n/3$. As $r \geq 0$ in any valid scheme, we have $t < n/3$. Consider the randomized construction of a scheme ϕ' described in the beginning of this section with $\ell = r + 1$.

Now fix a polynomial f and a query x and let us look at the probability that the randomized construction gives us the answer "Don't know" on this particular data/query-pair. The probability is equal to the probability that the random set S intersects the fixed set T of bits that are inspected on query x in structure $\phi(f)$ according to the old scheme. As $|S| = r + 1$ and $|T| \leq t$, the probability of no intersection can be bounded as

$$\begin{aligned}\Pr[S \cap T = \emptyset] &= \left(\frac{s-t}{s}\right)\left(\frac{s-1-t}{s-1}\right)\cdots\left(\frac{s-(r+1)+1-t}{s-(r+1)+1}\right)\\ &\geq \left(1-\frac{t}{n}\right)^{r+1}\\ &\geq 1-\frac{(r+1)t}{n}\\ &> 2/3.\end{aligned}$$

This means that if we fix f and count the number of answers that are not "Don't know" among all answers to "What is $f(x)$?", $x \in \mathbf{F}$, the expected number of such valid answers is greater than $2|\mathbf{F}|/3$, and the expected number of "Don't know" answers is smaller than $|\mathbf{F}|/3$. Thus, for fixed f, the probability that the number of valid answers for this f is smaller than $|\mathbf{F}|/3$ is smaller than $1/2$.

Define f to be "good" for a particular choice of S if the number of valid answers for f is at least $|\mathbf{F}|/3$. Thus, for random S, the probability that a particular fixed f is good is greater than $1/2$, by the above calculation; so if we count among all 2^n possible f's the number of good f's, the expectation of this number is greater than $2^n/2$. Thus, we can fix a value of S so that the number of good f's is greater than $2^n/2$. Let the set of good f's relative to this choice of S be called G.

We now argue that the map $\phi' : G \to \{0, 1\}^{n-1}$ is a 1-1 map: Given the value $\phi'(f)$ for a particular $f \in G$, we can run the query algorithm for $f(x)$ for all $x \in \mathbf{F}$ and retrieve a valid answer in at least $|\mathbf{F}|/3$ cases — in the other cases we get the answer "Don't know". Since the degree of f is less than $|\mathbf{F}|/3$, the information we retrieve is sufficient to reconstruct f.

Thus, we have constructed a 1-1 map from G with $|G| > 2^n/2$ to the set $\{0, 1\}^{n-1}$ which has size $2^n/2$. This violates the pigeonhole principle, and we conclude that our assumption $(r + 1)t < n/3$ was in fact wrong. □

Theorem 2 can be generalized to any problem based on error-correcting codes. Consider an arbitrary Boolean static data structure problem, given by a map $f : \{0, 1\}^n \times \{0, 1\}^m \to \{0, 1\}$. Let $N = 2^n$, and $M = 2^m$. Then the problem can be represented by an $N \times M$ Boolean matrix A_f, where the entry at the row indexed by x and the column indexed by y is equal to $f(x, y)$.

Theorem 5. *Let A_f be the N by M ($N = 2^n$, $M = 2^m$) matrix of a data structure problem such that the rows of A_f have pairwise distance at least δM. If the problem can be solved with redundancy r and query time t, then $t(r + 1) \geq \delta n/2$.*

Let $\alpha_{i_1}, \ldots, \alpha_{i_{M-d}}$ be an arbitrary 0/1 assignment to the $M - d$ coordinates $i_1, \ldots, i_{M-d}$. The set $S \subseteq \{0, 1\}^M$ of size $|S| = 2^d$ formed by all possible vectors from $\{0, 1\}^M$ agreeing with $\alpha_{i_1}, \ldots, \alpha_{i_{M-d}}$ and arbitrary in the remaining coordinates is called a *subcube of dimension d*.

The argument can also be extended to problems where the minimum distance may not be large, but instead we require that within any ball of radius ρM, there are at most L codewords (i.e., codes with certain *list decoding* properties). In fact, the even weaker property of having only few codewords in every *subcube* of dimension ρM is sufficient for our purposes. Note that this property corresponds to the problem of list decoding from erasures, rather than from errors.

Theorem 5 is a direct consequence of the next theorem.

Theorem 6. *Let A_f be the N by M ($N = 2^n$, $M = 2^m$) matrix of a data structure problems such that within any subcube of dimension ρM, there are at most L row vectors from A_f. If the problem can be solved with redundancy r and query time t, then $t(r + 1 + \lceil \log_2 L \rceil) \geq \rho(n - \lceil \log_2 L \rceil)/2$.*

Proof. Let a storage scheme ϕ with redundancy r and an associated query scheme with bit probe complexity t be given. Let $s = n + r$. Assume to the contrary that the scheme satisfies $(r + 1 + \lceil \log_2 L \rceil)t < \rho(n - \lceil \log_2 L \rceil)/2$. As $r \geq 0$ in any valid scheme, we have $t < \rho(n - \lceil \log_2 L \rceil)/2$.

Consider the randomized construction of a scheme ϕ' described in the beginning of this section with $\ell = r + 1 + \lceil \log_2 L \rceil$.

Now fix a codeword x and a query y (asking for a given bit of the word) and let us look at the probability that the randomized construction gives us the answer "Don't know" on this particular data/query-pair. The probability is equal to the probability that the random set S intersects the fixed set T of bits that are inspected on query y in the structure $\phi(x)$ according to the old scheme. Recall that $|S| = r + 1 + \lceil \log_2 L \rceil$ and $|T| \leq t$.

$$\begin{aligned}
\Pr[S \cap T = \emptyset] &= \left(\frac{s-t}{s}\right)\left(\frac{s-1-t}{s-1}\right)\cdots\left(\frac{s-(r+1+\lceil \log_2 L \rceil)+1-t}{s-(r+1+\lceil \log_2 L \rceil)+1}\right) \\
&\geq \left(1 - \frac{t}{n - \lceil \log_2 L \rceil}\right)^{r+1+\lceil \log_2 L \rceil} \\
&\geq 1 - \frac{t(r+1+\lceil \log_2 L \rceil)}{n - \lceil \log_2 L \rceil} \\
&> 1 - \rho/2,
\end{aligned}$$

if we assume that $t(r + 1 + \lceil \log_2 L \rceil) < \rho(n - \lceil \log_2 L \rceil)/2$.

Then for a fixed x, the expected number of valid answers is $>(1 - \rho/2)M$. For fixed x, the probability that the number of valid answers is $<(1 - \rho)M$ is less than $1/2$.

Define x to be "good" for a particular choice of S if the number of valid answers for x is at least $(1 - \rho)M$, and fix a value of S so that the number of good x's is $>2^n/2$. Let the set of good x's relative to this choice of S be called G.

We now consider the map $\phi' : G \to \{0, 1\}^{n-1-\lceil \log_2 L \rceil}$. Given a string in the image of ϕ', we run the query algorithm for y for each question y, and get a valid answer in at least $(1 - \rho)M$ cases — in the other cases we get the answer "Don't know". Thus, we retrieve at least $(1 - \rho)M$ of the M bits of the codeword x. This means that for any string z that is equal to $\phi'(x)$ for some $x \in G$, there are at most L words in G that could possibly be mapped by ϕ' to the same string z. Thus, $|G| \leq L2^{n-1-\lceil \log_2 L \rceil} = 2^{n-1}$ must hold, which leads to a contradiction. □

Known constructions of error-correcting codes yield examples of problems where we obtain bounds of the form $t(r+1) \geq \Omega(n)$ for $n = \Omega(2^m)$. For example, using the codes of [2] we get $t(r+1) \geq (1/2-\epsilon)\frac{n}{2}$ with $n = \Omega(\epsilon^3 2^m)$, and using the erasure list-decodable code constructions of [18] we get $t(r + 1 + \log_2(1/\epsilon)) \geq (1 - \epsilon)\frac{n - log_2(1/\epsilon)}{2}$ with $n = \Omega(\frac{\epsilon^2}{\log_2(1/\epsilon)} 2^m)$.

2.2. A general lower bound based on balance

We give a general lower bound on the product of the redundancy r and query time t for any problem whose matrix satisfies certain conditions. Informally, we require that the submatrix formed by any small subset of rows contains a balanced column.

Definition 7. Let A be a matrix with 0/1 entries. We say that A has balance at least λ for parameter k, if for any k rows of the matrix A there exists a column that contains at least λk 0-s and at least λk 1-s among the entries of the given k rows.

The property of having large balance for every $k > 1$ follows from the property of being a good error-correcting code.

Lemma 8. *Given a code with N words in $\{0, 1\}^l$, let A be the N by l matrix formed by the words as rows. If the minimum distance of the code is δl, then A has balance at least $\delta/2$ for every $1 < k \leq N$.*

Proof. Look at the k by l table formed by k rows of A. The sum of the distances between all pairs of rows of this table is at least $\binom{k}{2}\delta l$. Assume that the largest balance achieved in this table is λ. Then each column contributes at most $\lambda k(k-1)$ to the sum of the distances between pairs of rows, and we get $\lambda \geq \delta/2$. □

The proof of Lemma 8 extends to codes where the minimum distance may not be large, but where instead it holds that within any ball of certain radius there are not too many words, i.e., to problems satisfying the condition of Theorem 6. More precisely, we get the following statement:

Lemma 9. *Let A be the N by l matrix formed by the words of a code such that in any ball of radius $\delta/2$ there are at most L words. Then A has balance at least $(1-\frac{L-1}{k-1})\delta/2$ for every $L < k \leq N$.*

Proof. In this case, the sum of the distances between pairs of any $k > L$ rows is at least $k(k-L)/2\delta l$. As before, the contribution of each column to the sum of the distances between pairs of rows is at most $\lambda k(k-1)$, and we get $\lambda \geq (1-\frac{L-1}{k-1})\delta/2$. □

Lemma 8 allows us to obtain constructions of matrices with large balance, based on known constructions of error-correcting codes. This shows that there are matrices that have balance close to 1/4 for every k, and at the same time have a large number of rows. (We can use for example the codes of [2] that achieve a relative minimum distance $1/2-\epsilon$ and rate $\Omega(\epsilon^3)$.)

We can, however, construct matrices that satisfy the property of having large balance for every k, *without* the property of having few codewords in every Hamming ball of a given radius, and even without the weaker property of having few codewords in every subcube of a given dimension.

Consider the following example of such a construction. Start with any N by M matrix A that has balance at least λ for every $2 \leq k \leq N$, for some $0 < \lambda \leq 1/2$. Add L rows to the matrix A as follows. Take a code of L words on $c\log_2 L$ coordinates for some constant c, with relative minimum distance 1/4. (Use for example the codes of [2] that achieve relative minimum distance $1/2-\epsilon$ and rate $\Omega(\epsilon^3)$, with $\epsilon = 1/2$.) Let the first $c\log_2 L$ coordinates of the extra L words be words from this code of size L, and let the L words be identical in the remaining $M - c\log_2 L$ coordinates. Let $0 < \rho < 1$ be any constant such that $c\log_2 L < \rho M$. Then we have L words in a subcube of dimension ρM. On the other hand, the corresponding matrix has balance at least $\lambda/5$ for any $2 \leq k \leq N + L$. To see this, let $\alpha = \frac{1}{8\lambda+1}$. Since $\lambda \leq 1/2$ we have $\alpha \geq 1/5$. Take any k rows of the matrix. Among these, either at least αk rows are from A, or at least $(1-\alpha)k$ rows are from the extra L rows. Since A has balance λ for every k, in the first case there is a column with at least $\lambda\alpha k \geq \lambda/5k$ 0-s and 1-s among the given k rows. By Lemma 8, the submatrix consisting of the extra L rows has balance at least 1/8, thus in the second case there is a column with at least $1/8(1-\alpha)k \geq \lambda/5k$ 0-s and 1-s among the given k rows.

Thus, the following theorem has potential of giving lower bounds for a wider range of problems than the theorems of Section 2.1.

Theorem 10. *Consider an arbitrary Boolean static data structure problem given by a map $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$. Let A_f be the N by M ($N = 2^n$, $M = 2^m$) matrix of f. If A_f has balance at least λ for every $1 < k \leq \log N$, and the problem defined by f can be solved with redundancy r and query time t, then $t(r+1)^2 \geq \lambda n$.*

We begin with some statements that we use in the proof.

A family of k sets $S_1, \ldots, S_k$ is called a *sunflower* with k petals and core T, if $S_i \cap S_j = T$ for all $i \neq j$. We also require that the sets $S_i \setminus T$ be nonempty.

Lemma 11 (*Erdős and Rado, [12]*). *Let $\mathcal{F}$ be a family of sets each with cardinality w. If $|\mathcal{F}| > w!(k-1)^w$, then $\mathcal{F}$ contains a sunflower with k petals.*

We also use the following standard observation.

Observation 12. *Given a set $\mathcal{C}$ of $N = 2^{s-r}$ vectors in $\{0,1\}^s$, for every $0 \leq w \leq s$ there is a vector $v \in \{0,1\}^s$ such that there are at least $\binom{s}{w}/2^r$ vectors in $\mathcal{C}$ at distance w from v.*

Proof. Let $\chi(u,v) = 1$ if u and v differ in w coordinates, and $\chi(u,v) = 0$ otherwise. We have

$$\sum_{u\in\mathcal{C}} \sum_{v\in\{0,1\}^s} \chi(u,v) = |\mathcal{C}|\binom{s}{w}.$$

On the other hand,

$$\sum_{v\in\{0,1\}^s}\sum_{u\in\mathcal{C}}\chi(u,v)\le 2^s\max_{v\in\{0,1\}^s}|\mathcal{C}_{v,w}|,$$

where $\mathcal{C}_{v,w}=\{z\in\mathcal{C}|\ z$ and v differ in w coordinates$\}$. □

The following lemma holds for arbitrary data structure problems.

Lemma 13. *Let $\phi:\{0,1\}^n\to\{0,1\}^s$ be the storage scheme of a solution with redundancy r for an arbitrary data structure problem. Let B be a matrix of size $2^n\times s$, such that the row of B indexed by $x\in\{0,1\}^n$ is the vector $\phi(x)$. Then there exist some vector $z\in\{0,1\}^s$ and $k=\lceil s/(r+1)^2\rceil$ rows $u_1,\ldots,u_k$ of the matrix B, such that the sets of coordinates where z differs from u_i for $i=1,\ldots,k$ are pairwise disjoint.*

Proof. Let $w=r+1$ (note that $r+1\ge 1$), and let $v\in\{0,1\}^s$, guaranteed to exist by the observation, be such that there are at least $\binom{s}{r+1}/2^r$ rows of B at distance $r+1$ from v. Let B_v be the matrix obtained from B by adding v to each row of B (taking bitwise XOR).

With each vector $u\in\{0,1\}^s$, we associate a set $U\subseteq[s]$ such that $i\in[s]$ belongs to U if and only if the i-th entry of u is 1. Then the matrix B_v specifies a family $\mathcal{B}$ of N sets such that at least $\binom{s}{r+1}/2^r$ members of $\mathcal{B}$ have cardinality $r+1$.

Next we show that $\binom{s}{r+1}/2^r\ge(r+1)!(s/(r+1)^2)^{r+1}$. We use $\binom{s}{r+1}\ge(s/(r+1))^{(r+1)}$. Then, $\binom{s}{r+1}/2^r\ge 2(\frac{r+1}{2})^{(r+1)}(s/(r+1)^2)^{r+1}$. Using $2((r+1)/2)^{(r+1)}\ge(r+1)!$, we get the desired bound. Thus, Lemma 11 implies that $\mathcal{B}$ contains a sunflower with $k=\lceil s/(r+1)^2\rceil$ petals. Let $S_1,\ldots,S_k$ be the sets of the sunflower, and let T be its core. Then, the sets $S_i\triangle T$ are pairwise disjoint. ($S_i\triangle T=S_i\setminus T\cup T\setminus S_i$ denotes the symmetric difference of the sets S_i and T.) Let z and $u_1,\ldots,u_k$ be the vectors obtained by adding the vector v to the characteristic vectors of the set T and $S_1,\ldots,S_k$, respectively. Then the vectors $u_1,\ldots,u_k$ are rows of the matrix B, and they have the property that the vectors $z\oplus u_1,\ldots,z\oplus u_k$ have no common 1's, since the set $S_i\triangle T$ is exactly the set of coordinates where the vectors z and u_i differ from each other. □

Proof (*of Theorem* 10). A solution to the data structure problem is given by a representation $\phi:\{0,1\}^n\to\{0,1\}^s$ and a query algorithm. We consider a matrix B of size $N\times s$ such that the row of B indexed by x is the vector $\phi(x)$.

Let $k=\lceil s/(r+1)^2\rceil$ and let $z\in\{0,1\}^s$ be the vector and $u_1,\ldots,u_k$ be the rows of the matrix B guaranteed by Lemma 13.

Let $x_1,\ldots,x_k$ be the data such that $u_i=\phi(x_i), i=1,\ldots,k$. Consider now the k rows of A_f indexed by $x_1,\ldots,x_k$. Since $k\le n=\log_2 N$, by our assumption on A_f, there is a question y, such that at least λk of the answers $f(x_i,y)$ are 0, and at least λk of the answers $f(x_i,y)$ are 1.

We think of the query algorithm as a decision tree, and show that it has large depth. In particular, we show that the path consistent with the vector z has to be at least λk long.

Note that the vector z may not be a row of the matrix B. Nevertheless, for every vector z, the decision tree has exactly one path that is consistent with z. If none of the vectors that are consistent with the path reaching a given leaf appears as a row of the matrix B, then the query algorithm will never actually reach this leaf, and the label of the leaf will not affect the correctness of the algorithm. We can assume that the decision tree has been trimmed, so that there are no paths that can be cut off without affecting the correctness of the algorithm. Then, proving that the depth of the tree is at least λk implies that there is at least one path corresponding to a vector $\phi(x)$ that the algorithm may actually have to follow, and it is at least λk long.

Assume that the query algorithm reads at most $t<\lambda k$ bits on any input when trying to answer the question y, and assume that the bits read are consistent with the vector z. Since the sets of coordinates where z differs from u_i for $i=1,\ldots,k$ are pairwise disjoint, after asking at most t questions, the algorithm can rule out at most t of the data $x_1,\ldots,x_k$, and the remaining $k-t$ are still possible. If $t<\lambda k$, then among the data that are still not ruled out, both the answer 0 and the answer 1 is possible, and the algorithm cannot determine the answer to the given question y. This completes the proof of Theorem 10. □

It is not hard to find examples of matrices with large balance for $k\le\log N$, provided we are not worried about the number of rows N being large enough compared to the number of columns M. We should mention that there are well known constructions (e.g. [3,19,27,28,33]) for the much stronger property requiring that all possible 2^k patterns

appear in the submatrix formed by arbitrary k rows. However, in such examples, $N \leq M$ or $2^k \leq M$ must trivially hold. Error-correcting codes provide examples where N can be very large compared to M.

Let $n(k, \lambda, M)$ denote the largest possible number n such that 2^n by M 0/1 matrices exist with balance at least λ for k. Note that balance $1/k$ for k can always be achieved as long as there are no repeated rows; thus $n(k, 1/k, M) = 2^M$. Estimates on the largest achievable rate of error-correcting codes or list decodable codes provide lower bounds on $n(k, \lambda, M)$. For example, the Gilbert–Varshamov bound (see e.g. [21]) together with Lemma 8 implies $n(k, \lambda, M) \geq (1 - H(2\lambda))M$, for every $k > 1$, where $H(\alpha) = \alpha \log_2 \frac{1}{\alpha} + (1-\alpha) \log_2 \frac{1}{1-\alpha}$.

Note that while error-correcting codes give large balance for every $k > 1$, for our purposes matrices that have large balance for only certain values of k may already be useful. In particular, any construction with nontrivial $\lambda > 1/k$ balance for a given k implies nontrivial lower bounds on t when $(r+1)^2 \leq s/k$. In the statement of Theorem 10, we require balance λ for *every* $1 < k \leq \log_2 N$ to obtain a tradeoff between t and r.

It would be interesting to know if $n(k, \lambda, M)$ can be significantly larger (for certain values of k) than what is achievable by error-correcting or list decodable codes.

3. Lower bounds for systematic structures

We first show the bounds for Prefix Sum. In fact, the lower bound is already implicit in Nisan, Rudich and Saks [29], as is seen by the proof. Note that in the non-systematic model, Prefix Sum is trivial, as the number of queries equals the number of bits in the input.

Theorem 3. *Consider Prefix Sum with parameter n. For answering queries about it in a systematic scheme with redundancy r, $t = \Theta(n/(r+1))$ bit probes are necessary and sufficient.*

Proof. *Upper bound:* For $r = 0$, the upper bound is obvious. For $r \geq 1$, divide the input vector into r equal sized blocks and let y_i be the parity of the i'th block. Now store for each $j = 1, ...r$, the parity of $y_1, y_2, \ldots, y_j$. Given a prefix sum query, it can be answered by reading a non-systematic bit, that gives the parity of a collection of blocks and XORing it with a number of individual input bits, all found in a single block of size n/r. The bit probe complexity is $O(n/r)$.

Lower bound: Let a scheme of redundancy r be given, and suppose the queries can be answered with t bit probes, i.e., we can find $x_1 \oplus \cdots \oplus x_j$ using a decision tree of depth t over the input bits and the index bits. Split the input into $r+1$ blocks of about equal length, each block containing at least $\lfloor \frac{n}{r+1} \rfloor$ bits. It is possible to determine the parity of one of the blocks by a decision tree of depth $2t$ over the input bits and the index bits.

We now apply a theorem of Nisan, Rudich and Saks [29]: Given $l+1$ instances of computing parity of k bits, with l *help bits* (which can be arbitrary functions of the $(l+1)k$ input bits), given for free. At least one of the $l+1$ parity functions has decision tree complexity $\geq k$. We immediately get the desired bound. □

Theorem 4. *Consider Substring Search with parameters n, m so that $2\log_2 n + 5 \leq m \leq 5 \log_2 n$. For any systematic scheme solving it with redundancy r and bit probe complexity t, we have $(r+1)t \geq \frac{1}{800} n / \log n$.*

Proof. Since we must have $r \geq 0$ and $t \geq 1$ in a valid scheme, we can assume that $1 \leq t \leq \frac{n}{800 \log n}$ otherwise there is nothing to prove.

We need to prove a claim about a certain two-player game. Let $b \geq a \geq 40$ be integers and assume b is even. The game is played with b boxes labeled $0, \ldots, b-1$ and a slips of papers, labeled $0, \ldots, a-1$. Player I colors each slip of paper either red or blue and puts each slip of paper in a box (with no two slips going into one box) without Player II watching. Now Player II can open at most $b/2$ boxes using any adaptive strategy, and based on this, must make a guess about the color of every slip of paper. Player II wins the game if he correctly announces the color of every slip of paper.

Claim. *Suppose Player I adopts the strategy of coloring each slip of paper uniformly and independently at random, and putting them at random into a boxes chosen uniformly at random.*[1] *Then no matter which strategy Player II adopts, the probability that Player II wins the game is at most $2^{-a/20}$.*

[1] This is actually easily seen to be the optimal mixed strategy of Player I, but we shall not use this fact.

Proof of Claim. When Player I is playing uniformly at random in the way described, then by symmetry the adaptiveness of Player II is useless and the optimal strategy for Player II is to open boxes 1, 2, ..., $b/2$, announce the colors of the slips of papers found, and then make an arbitrary guess for the rest. The probability that he finds more than $\frac{9}{10}a$ slips of papers is

$$\sum_{j>\frac{9}{10}a}^{a} \frac{\binom{b/2}{j}\binom{b/2}{a-j}}{\binom{b}{a}} = \sum_{i=0}^{\lfloor \frac{1}{10}a \rfloor} \frac{\binom{b/2}{i}\binom{b/2}{a-i}}{\binom{b}{a}}.$$

Since $a \leq b$, for $i \leq \frac{1}{10}a$, we have $\frac{b}{2(b-i)} \leq 5/9$. Then,

$$\frac{\binom{b/2}{i}\binom{b/2}{a-i}}{\binom{b}{a}} \leq \binom{a}{i}(1/2)^i \left(\frac{b}{2(b-i)}\right)^{a-i} \leq \binom{a}{i}(5/9)^a.$$

$$\begin{aligned}\sum_{i=0}^{\lfloor \frac{1}{10}a \rfloor} \frac{\binom{b/2}{i}\binom{b/2}{a-i}}{\binom{b}{a}} &\leq (5/9)^a \sum_{i=0}^{\lfloor \frac{1}{10}a \rfloor} \binom{a}{i} \\ &\leq (5/9)^a 2^{H(1/10)a} \\ &\leq 2^{(H(1/10)-\log_2(3/2))a} \\ &\leq 2^{-0.115a}.\end{aligned}$$

The probability that he guesses the colors of all remaining slips correctly, given that at least $a/10$ were not found, is at most $2^{-a/10}$. Thus, the probability that Player II correctly guesses the color of every slip of paper is bounded by

$$2^{-0.115a} + 2^{-a/10} \leq 2^{-a/20},$$

as $a \geq 40$. This completes the proof of the claim. □

We show that a good scheme for Substring Search leads to a good strategy for Player II in the game. So given a scheme with parameters n, m, r, t, we let $a = \lfloor \frac{n}{4tm} \rfloor$ and $b = 4ta$. Since $t \leq n/(800 \log n)$ and $m \leq 5 \log n$, we have $a \geq 40$.

We consider a string of length n as consisting of b concatenated chunks of length m, padded with 0's to make the total length n (note that $bm = 4tam \leq n$). We can now let such a string encode a move of Player I (i.e. a coloring of slips of paper and a distribution of them into boxes) as follows: the content of Box i is encoded in chunk number i. If the box is empty, we make the chunk 000000...000. If the box contains paper slip number j, colored blue, we make the chunk $001j_1 1 j_2 1 j_3 1 ... 1 j_k 0$, padded with zeros to make the total length m, where $j_1 ... j_k$ is the binary representation of j with $\lceil \log a \rceil$ binary digits (note that $3 + 2\lceil \log a \rceil \leq 2 \log n + 5 \leq m$). Similarly, if the box contains paper slip number j, colored red, we make the chunk $001j_1 1 j_2 1 j_3 1 ... 1 j_k 1$, padded with zeros in the same way.

Now consider the set X of strings encoding all legal moves of player I. Each element x of X has some systematic data structure $\phi(x) = x \cdot \phi^*(x)$, where $\phi^*(x) \in \{0,1\}^r$. Pick the most likely setting z of $\phi^*(x)$ of these among elements of X; i.e., if we take a random element x of X, the probability that $\phi^*(x) = z$ is at least 2^{-r}. We now make a strategy for Player II in the game. Player II will pretend to have access to a Substring Search data structure which he will hope encodes the move of Player I. The index part of this data structure will be the string z which is fixed and independent of the move of Player I, and hence can be hardwired into the protocol of Player II.

Player II shall simulate certain query operations on the pretend data structure. However, he has only access to the index part of the structure (i.e., z). Thus, whenever he needs to read a bit of the non-index bits, he shall open the box corresponding to the chunk of the bit from which he can deduce the bit (assuming that the entire data structure really does encode the move of Player I).

In this way, Player II simulates performing the query operations "Is $001j_1 1 j_2 1 j_3 1 ... 1 j_k 0$ a substring?" and "Is $001j_1 1 j_2 1 j_3 1 ... 1 j_k 1$ a substring?" with $j = j_1 j_2 \ldots j_k$ being the binary representations of all $y \in \{0, \ldots, a-1\}$, i.e., $2a$ query operations. From the answers to the queries, he gets a coloring of the slips of papers. Of course, the coloring obtained may be unrelated to the correct coloring. But suppose the randomly chosen z happens to be equal to $\phi^*(x)$, where x is an encoding of the move of Player I. Then, the coloring obtained is the correct coloring by the correctness

of the Substring Search scheme. Since z is a randomly chosen bitstring of length r, it is equal to $\phi^*(x)$ with probability 2^{-r}, so this is a lower bound on the probability of the strategy for Player II obtaining the correct coloring. Thus, since the total number of boxes opened is at most $t2a \leq b/2$, we have by the claim that $r \geq a/20$, i.e., $20r \geq \lfloor n/4tm \rfloor$. Since $m \leq 5 \log n$ we have $(r+1)t \geq \frac{1}{400} n/\log n$. □

We could potentially get a better lower bound by considering a more complicated game that takes into account the fact that the different query operations do not communicate. Again we have b boxes labeled $0, \ldots, b-1$ and a slips of paper, labeled $0, \ldots, a-1$. The modified game is played between Player I and a *team* consisting of Players $\mathrm{II}_0, \mathrm{II}_1, \ldots, \mathrm{II}_{a-1}$. Again, Player I colors each slip of paper either red or blue and puts each slip of paper in a box without Players $\mathrm{II}_0, \mathrm{II}_1, \ldots, \mathrm{II}_{a-1}$ watching. Now Player II_i can look in at most $b/2$ boxes using any adaptive strategy, and based on this must make a guess about the color of the slip labeled i. This is done by each player on the team individually, without communication or observation between them. The team wins if *every* player in the team correctly announces the color of "his" slip.

About this game, we stated the following hypothesis in the preliminary version [14] of this paper.

Hypothesis. Let $b \geq 2a$. Suppose Player I adopts the strategy of coloring each slip of paper uniformly at random and independently putting them at random into a boxes chosen uniformly at random. Then no matter which strategy the team adopts, the probability that they win is at most $2^{-\Omega(a)}$.

Since then, Goyal and Saks [15] have shown that our hypothesis is not true exactly as stated: they proved that there is a strategy for the team that succeeds with probability at least $2^{-c\sqrt{a}\log a}$, for some constant c (independent of a and b). However, it is still possible that a weaker version of the hypothesis holds. Is the probability that the team wins always at most $2^{-a^{\Omega(1)}}$? If this is the case, it would still imply stronger lower bounds for systematic structures.

The intuition for the validity of the weaker hypothesis (i.e. with revised parameters) is the fact that the players of the team are unable to communicate, and each will find his own slip of paper with probability $\leq \frac{1}{2}$. However, Sven Skyum (personal communication) has pointed out that if this hypothesis is true, the parameters under which it is true are somewhat fragile: if $b = a$, the team can win the game with probability bounded from below by a constant (roughly 0.31) for arbitrary large values of a. The catch is that even though each player will find his own slip of paper with probability only $\frac{1}{2}$, one can make these events highly dependent (despite the fact that the players do not communicate). Though the existence of this protocol is quite striking and has surprised many people since the publication of the preliminary version of this paper, it is also very simple: If we let $l(i)$ denote the number on the label of the slip in Box i, the strategy prescribes for Player j to open first Box j, then Box $l(j)$, then Box $l(l(j))$, etc. Then, each member of the team will find his own slip and hence be able to announce its color correctly *if* the permutation of slips into boxes does not have a cycle of length bigger than $n/2$. Asymptotically, as $n \to \infty$, this is the case with probability $1 - \ln 2 \approx 0.31$.

4. Open problems

It is interesting that all our best bounds, both in the non-systematic and in the systematic case, are of the form "$(r+1)t$ must be linear or almost linear in n." We don't see any inherent reason for this and in general do not expect the lower bounds obtained to be tight. Thus, it would be nice to to prove a lower bound of, say, the form, $t < n/\mathrm{polylog}\, n \Rightarrow r > n/\mathrm{polylog}\, n$ for Polynomial Evaluation in the non-systematic case, or Substring Search in the systematic case. To get a superlinear lower bound on $(r+1)t$ at least for some non-trivial values of r, it would be sufficient to verify the revised hypothesis about the "slips in boxes" game defined above. It is also interesting to note that our lower bound for Substring Search and the lower bound of Demaine and Lopez-Ortiz are incomparable. Can the two techniques perhaps be combined to yield a better lower bound?

We have only been able to prove lower bounds in the non-systematic case for problems satisfying certain coding theoretic properties. It would be very nice to extend the non-systematic lower bounds to more natural search and retrieval problems, such as Substring Search.

A prime example of a problem for which we would like better bounds is Membership, as defined in the introduction. As the data to be stored have no canonical representation as a bitstring, it only makes sense to consider this problem in the non-systematic model. The lower bound $r = O(n) \Rightarrow t = \Omega(m)$ was shown by Buhrman et al [8]. On the other hand, a variety of low-redundancy dictionaries with $r = o(n)$ and $t = O(m)$ have been constructed

[7,30]. We conjecture that any solution for membership with $t = O(m)$ must have *some* redundancy, i.e., that $t = O(m) \Rightarrow r \geq 1$. It would be very nice to establish this.

A question that may be interesting in its own right is that of whether $n(k, \lambda, M)$ (defined at the end of Section 2.2) can be significantly larger than what is implied by coding theory bounds on error-correcting or list decodable codes. (See Section 2.2 for more details.)

In our view, the main open problem of cell probe complexity remains: show, for some explicit *Boolean* problem, a tradeoff of the form $r = O(n) \Rightarrow t = \omega(m)$. Clearly, for such tradeoffs the distinction between systematic and non-systematic structures is inconsequential.

Acknowledgements

We thank the anonymous referees for helpful comments and suggestions. The first author was supported in part by NSF CAREER Award CCR-9874862, NSF Grant CCF-0430695 and an Alfred P. Sloan Research Fellowship. The second author was supported by BRICS, Basic Research in Computer Science, a centre of the Danish National Research Foundation.

References

[1] M. Ajtai, A lower bound for finding predecessors in Yao's cell probe model, Combinatorica 8 (1988) 235–247.
[2] N. Alon, J. Bruck, J. Naor, M. Naor, R. Roth, Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs, IEEE Transactions on Information Theory 38 (1992) 509–516.
[3] N. Alon, O. Goldreich, J. Håstad, R. Peralta, Simple constructions of almost k-wise independent random variables, Random Structures and Algorithms 3 (1992) 289–304.
[4] O. Barkol, Y. Rabani, Tighter bounds for nearest neighbor search and related problems in the cell probe model, Journal of Computer and System Sciences 64 (2002) 873–896.
[5] P. Beame, F.E. Fich, Optimal bounds for the predecessor problem, Journal of Computer and System Sciences (2002) 38–72.
[6] A. Borodin, R. Ostrovsky, Y. Rabani, Lower bounds for high dimensional nearest neighbor search and related problems, in: Discrete and Computational Geometry — The Goodman–Pollack Festschrift, in: Algorithms and Combinatorics Series, vol. 3143, 2003, pp. 255–276.
[7] A. Brodnik, J.I. Munro, Membership in constant time and almost-minimum space, SIAM Journal on Computing 28 (1999) 1627–1640.
[8] H. Buhrman, P.B. Miltersen, J. Radhakrishnan, S. Venkatesh, Are bitvectors optimal? in: Proc. 32th Annual ACM Symposium on Theory of Computing, STOC'00, pp. 449–458.
[9] A. Chakrabarti, B. Chazelle, B. Gum, A. Lvov, A lower bound on the complexity of approximate nearest-neighbor searching on the Hamming Cube, in: Proc. 31th Annual ACM Symposium on Theory of Computing, STOC'99, pp. 305–311.
[10] E.D. Demaine, A. Lopez-Ortiz, A linear lower bound on index size for text retrieval, Journal of Algorithms 48 (2003) 2–15.
[11] P. Elias, R.A. Flower, The complexity of some simple retrieval problems, Journal of the Association for Computing Machinery 22 (1975) 367–379.
[12] P. Erdős, R. Rado, Intersection theorems for systems of sets, Journal of London Mathematical Society 35 (1960) 85–90.
[13] M.L. Fredman, J. Komlós, E. Szemerédi, Storing a sparse table with $O(1)$ worst case access time, Journal of the Association for Computing Machinery 31 (1984) 538–544.
[14] A. Gál, P.B. Miltersen, The cell probe complexity of succinct data structures, in: Proc. of 30th International Colloquium on Automata, Languages and Programming, ICALP'03, pp. 332–344.
[15] N. Goyal, M. Saks, A Parallel Search Game, Random Structures and Algorithms 27 (2) (2005) 227–234.
[16] R. Grossi, J.S. Vitter, Compressed suffix arrays and suffix trees with applications to text indexing and string matching, in: Proc. 32th Annual ACM Symposium on Theory of Computing, STOC'00, pp. 397–406.
[17] R. Grossi, A. Gupta, J.S. Vitter, High-order entropy-compressed text indexes, in: Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'03, pp. 841–850.
[18] V. Guruswami, List decoding from erasure: Bounds and code constructions, IEEE Transactions on Information Theory 49 (2003) 2826–2833.
[19] D.J. Kleitman, J. Spencer, Families of k-independent sets, Discrete Mathematics 6 (1973) 255–262.
[20] D.E. Knuth, Seminumerical Algorithms, 2nd ed., in: The Art of Computer Programming, Vol. II, Addison-Wesley, Reading, MA, 1980.
[21] F.J. MacWilliams, N.J.A. Sloane, The theory of Error-correcting Codes, Elsevier/North-Holland, Amsterdam, 1981.
[22] U. Manber, S. Wu, GLIMPSE — A Tool to Search Through Entire Filesystems. White Paper. Available at: http://glimpse.cs.arizona.edu/.
[23] P.B. Miltersen, The bitprobe complexity measure revisited, in: 10th Annual Symposium on Theoretical Aspects of Computer Science, STACS'93, pp. 662–671.
[24] P.B. Miltersen, On the cell probe complexity of polynomial evaluation, Theoretical Computer Science 143 (1995) 167–174.
[25] P.B. Miltersen, N. Nisan, S. Safra, A. Wigderson, On data structures and asymmetric communication complexity, Journal of Computer and System Sciences 57 (1998) 37–49.
[26] M. Minsky, S. Papert, Perceptrons, MIT Press, Cambridge, Mass, 1969.
[27] J. Naor, M. Naor, Small-bias probability spaces: efficient constructions and applications, SIAM Journal on Computing 22 (4) (1993) 838–856.

[28] M. Naor, L. Schulman, A. Srinivasan, Splitters and near optimal derandomization, in: Proc. of 36th Annual IEEE Symposium on Foundations of Computer Science, FOCS'95, pp. 182–191.
[29] N. Nisan, S. Rudich, M. Saks, Products and Help Bits in Decision Trees, SIAM Journal on Computing 28 (1999) 1035–1050.
[30] R. Pagh, Low redundancy in static dictionaries with $O(1)$ lookup time, SIAM Journal on Computing 31 (2001) 353–363.
[31] J. Radhakrishnan, V. Raman, S.S. Rao, Explicit deterministic constructions for membership in the bitprobe model, in: Proc. of 9th Annual European Symposium on Algorithms, ESA'01, pp. 290–299.
[32] R. Raman, V. Raman, S.S. Rao, Succinct dynamic data structures, in: Proc. of 7th International Workshop on Algorithms and Data Structures, WADS'01, pp. 426–437.
[33] G. Seroussi, N. Bshouty, Vector sets for exhaustive testing of logic circuits, IEEE Transactions on Information Theory 34 (1988) 513–522.