Time-Space Trade-Offs for Predecessor Search

(Extended Abstract) *

Mihai Pătraşcu
mip@mit.edu

Mikkel Thorup
mthorup@research.att.com

ABSTRACT

We develop a new technique for proving cell-probe lower bounds for static data structures. Previous lower bounds used a reduction to communication games, which was known not to be tight by counting arguments. We give the first lower bound for an explicit problem which breaks this communication complexity barrier. In addition, our bounds give the first separation between polynomial and near linear space. Such a separation is inherently impossible by communication complexity.

Using our lower bound technique and new upper bound constructions, we obtain tight bounds for searching predecessors among a static set of integers. Given a set Y of n integers of ℓ bits each, the goal is to efficiently find $\texttt{PREDECESSOR}(x) = \max\ \{y \in Y \mid y \le x\}$. For this purpose, we represent Y on a RAM with word length b using $S \ge n\ell$ bits of space. Defining $a = \lg \frac{S}{n}$, we show that the optimal search time is, up to constant factors:

$$\min \begin{cases} \log_b n \\ \lg \frac{\ell - \lg n}{a} \\ \dfrac{\lg \frac{\ell}{a}}{\lg\left(\frac{a}{\lg n} \cdot \lg \frac{\ell}{a}\right)} \\ \dfrac{\lg \frac{\ell}{a}}{\lg\left(\lg \frac{\ell}{a} \Big/ \lg \frac{\lg n}{a}\right)} \end{cases}$$

In external memory ($b > \ell$), it follows that the optimal strategy is to use either standard B-trees, or a RAM algorithm ignoring the larger block size. In the important case of $b = \ell = \gamma \lg n$, for $\gamma > 1$ (i.e. polynomial universes), and near linear space (such as $S = n \cdot \lg^{O(1)} n$), the optimal search time is $\Theta(\lg \ell)$. Thus, our lower bound implies the surprising conclusion that van Emde Boas' classic data structure from [FOCS'75] is optimal in this case. Note that for space $n^{1+\varepsilon}$, a running time of $O(\lg \ell / \lg\lg \ell)$ was given by Beame and Fich [STOC'99].

Categories and Subject Descriptors

F.2.3 [**Tradeoffs between Complexity Measures**]; E.2 [**Data Storage Representations**]

General Terms

Algorithms, Performance, Theory

Keywords

predecessor search, cell-probe complexity, lower bounds

1. INTRODUCTION

In this paper we provide tight trade-offs between query time and space of representation for static predecessor search. This is one of the most basic data structures, and the trade-off gives the first separation between linear and polynomial space for *any* data structure problem.

1.1 The Complexity-Theoretic View

Yao's cell-probe model [18] is typically the model of choice for proving lower bounds on data structures. The model assumes the memory is organized in b-bit cells (alternatively called words). In the case of static data structures, one first constructs a representation of the input in a table with a bounded number of cells S (the space complexity). Then, a query can be answered by probing certain cells. The time complexity T is defined to be the number of cell probes. The model allows free nonuniform computation for both constructing the input representation, and for the query algorithm. Thus, the model is stronger than the word RAM or its variants, which are used for upper bounds, implementable in a programming language like C. In keeping with the standard assumptions on the upper bound side, we only consider $b = \Omega(\lg n)$.

Typically, lower bounds in this model are proved by considering a two-party communication game. Assume Bob holds the data structure's input, while Alice holds the query. By simulating the cell-probe solution, one can obtain a protocol with T rounds, in which Alice sends $\lg S$ bits and Bob replies with b bits per round. Thus, a lower bound on the number of rounds translates into a cell-probe lower bound.

Intuitively, we do not expect this relation between cell-probe and communication complexity to be tight. In the communication model, Bob can remember past communication, and answer new queries based on this. Needless to say, if Bob is just a table of cells, he cannot remember anything, and his responses must be a function of Alice's last

*A full version of is available as `arXiv:cs.CC/0603043`.

STOC'06, May 21–23, 2006, Seattle, Washington, USA.

message (i.e. the address of the cell probe). By counting arguments, it can be shown [10] that the cell-probe complexity can be much higher than the communication complexity, for natural ranges of parameters. However, a separation for an explicit problem has only been obtained in a very restricted setting. Gál and Miltersen [9] showed such a separation when the space complexity is very close to minimum: given an input of n cells, the space used by the data structure is $n + o(n)$.

Besides the reduction to communication complexity, and the approach of [9] for very small space, there are no known techniques applicable to static cell-probe complexity with cells of $\Omega(\lg n)$ bits. In particular, we note that the large body of work initiated by Fredman and Saks [7] only applies to *dynamic* problems, such as maintaining partial sums or connectivity. In the case of static complexity, there are a few other approaches developed specifically for the bit-probe model ($b = 1$); see [12].

In conclusion, known lower bound techniques for cell-probe complexity cannot surpass the communication barrier. However, one could still hope that communication bounds are interesting enough for natural data structure problems. Unfortunately, this is often not the case. Notice that polynomial differences in S only translate into constant factors in Alice's message size. In the communication game model, this can only change constant factors in the number of rounds, since Alice can break a longer message into a few separate messages. Unfortunately, this means that communication complexity cannot be used to separate, say, polynomial and linear space. For many natural data-structure problems, the most interesting behavior occurs close to linear space, so it is not surprising that our understanding of static data-structure problems is rather limited.

In this work, we develop a new lower-bound technique, the *cell-probe elimination lemma*, targeted specifically at the cell-probe model. Using this lemma, we obtain a separation between space $n^{1+o(1)}$ and space $n^{1+\varepsilon}$ for any $\varepsilon > 0$. This also represents a separation between communication complexity and cell-probe complexity with space $n^{1+o(1)}$. Our lower bounds hold for predecessor search, one of the most natural and well-studied problems.

Our lower bound result has a strong direct sum flavor, which is interesting in its own right. Essentially, we show that for problems with a certain structure, a data structure solving k independent subproblems with space $k \cdot \sigma$ cannot do better than k data structures solving each problem with space σ.

1.2 The Data-Structural View

Using our lower bound technique and new upper bound constructions, we obtain tight bounds for predecessor search. The problem is to represent an ordered set Y, such that for any query x we can find efficiently $\texttt{PREDECESSOR}(x) = \max\ \{y \in Y \mid y \leq x\}$. This is one of the most fundamental and well-studied problems in data structures. For a comprehensive list of references, we refer to [3]; here, we only describe briefly the best known bounds.

1.2.1 The Upper-Bound Story

We focus on the static case, where Y is given in advance for preprocessing. For example, we can sort Y, and later find the predecessor of x by binary search using $O(\lg n)$ comparisons, where $n = |Y|$.

On computers, we are particularly interested in integer keys. Thereby we also handle, say, floating point numbers whose ordering is preserved if they are cast as integers. We can then use all the instructions on integers available in a standard programming language such as C, and we are no longer limited by the $\Omega(\lg n)$ comparison based lower bound for searching. A strong motivation for considering integer keys is that integer predecessor search is asymptotically equivalent to the IP look-up problem for forwarding packets on the Internet [6]. This problem is extremely relevant from a practical perspective. The fastest deployed software solutions use non-comparison-based RAM tricks [5].

More formally, we will represent Y on a unit-cost word RAM with a given word length b. We assume each integers in Y has ℓ bits, and that $\lg n \leq \ell \leq b$. On the RAM, the most natural assumption is $\ell = b$. The case $b > \ell$ models the external memory model with $B = \lfloor \frac{b}{\ell} \lfloor$ keys per page. In this case, the well-known (comparison-based) B-trees achieve a search time of $O(\log_B n)$. For the rest of the discussion, assume $b = \ell$.

Using the classic data structure of van Emde Boas [16] from 1975, we can represent our integers so that predecessors can be searched in $O(\lg \ell)$ time. The space is linear if we use hashing [17].

In the 1990, Fredman and Willard [8] introduced fusion trees, which requires linear space and can answer queries in $O(\log_\ell n)$ time. Combining with van Emde Boas' data structure, they got a search time of $O(\min\{\frac{\lg n}{\lg \ell}, \lg \ell\})$, which is always $O(\sqrt{\lg n})$.

In 1999, Beame and Fich [3] found an improvement to van Emde Boas' data structure bringing the search time down to $O(\frac{\lg \ell}{\lg \lg \ell})$. Combined with fusion trees, this gave them a bound of $O(\min\{\frac{\lg n}{\lg \ell}, \frac{\lg \ell}{\lg \lg \ell}\})$, which is always $O(\sqrt{\frac{\lg n}{\lg \lg n}})$. However, the new data structure of Beame and Fich uses quadratic space, and they asked if the space could be improved to linear or near-linear.

As a partially affirmative answer to this question, we show that their $O(\frac{\lg \ell}{\lg \lg \ell})$ search time can be obtained with space $n^{1+1/\exp(\lg^{1-\varepsilon} \ell)}$ for any $\varepsilon > 0$. However, we also show, as our main result, that with closer to linear space, such as $n \lg^{O(1)} n$, one cannot in general improve the old van Emde Boas bound of $O(\lg \ell)$.

1.2.2 The Lower-Bound Story

Ajtai [1] was the first to prove a superconstant lower bound for our problem. His results, with a correction by Miltersen [11], can be interpreted as saying that there exists n as a function of ℓ such that the time complexity for polynomial space is $\Omega(\sqrt{\lg \ell})$, and likewise there exists ℓ a function of n making the time complexity $\Omega(\sqrt[3]{\lg n})$.

Miltersen [11] revisited Ajtai's work, showing that the lower bound holds in the communication game model, and for a simpler colored predecessor problem. In this problem, the elements of Y have an associated color (say, red or blue), and the query asks only for the color of the predecessor in Y. This distinction is important, as one can reduce other problems to this simpler problem, such as existential range queries in two dimensions [13] or prefix problems in a certain class of monoids [11]. Like previous lower bound proofs, ours also holds for the colored problem, making the lower bounds applicable to these problems.

Miltersen, Nisan, Safra and Wigderson [13] once again revisited Ajtai's proof, extending it to randomized algorithms. More importantly, they captured the essence of the proof in an independent *round elimination lemma*, which forms a general tool for proving communication lower bounds. Our cell-probe elimination lemma is inspired, at a high level, by this result.

Beame and Fich [3] improved the lower bounds to $\Omega(\frac{\lg \ell}{\lg \lg \ell})$ and $\Omega(\sqrt{\frac{\lg n}{\lg \lg n}})$ respectively. Sen and Venkatesh [14] later gave an improved round elimination lemma, which can reprove the lower bounds of Beame and Fich, but also for randomized algorithms. Analyzing the time-space trade-offs obtained by these proofs, one obtains $\Omega(\frac{\lg n}{\lg b}, \frac{\lg \ell}{\lg \lg S})$, where S is the space bound, and possibly $b > \ell$.

1.3 The Optimal Trade-Offs

Define $\lg x = \lceil \log_2(x+2) \rceil$, so that $\lg x \geq 1$ even if $x \in [0,1]$. Assuming S bits of space are available, and defining $a = \lg \frac{S}{n}$, we show that the optimal search time is, up to constant factors:

$$\min \begin{cases} \log_b n \\ \lg \frac{\ell - \lg n}{a} \\ \dfrac{\lg \frac{\ell}{a}}{\lg\left(\frac{a}{\lg n} \cdot \lg \frac{\ell}{a}\right)} \\ \dfrac{\lg \frac{\ell}{a}}{\lg\left(\lg \frac{\ell}{a} \Big/ \lg \frac{\lg n}{a}\right)} \end{cases} \tag{1}$$

The upper bounds are achieved by a deterministic query algorithm on a RAM. The data structure can be constructed in expected time $O(S)$ by a randomized algorithm, starting from a sorted list of integers. The lower bounds hold for deterministic query algorithms answering the colored predecessor problem in the cell-probe model. When $S \geq n^{1+\varepsilon}$ for some constant $\varepsilon > 0$, the lower bounds also hold in the stronger communication game model, even allowing randomization with two-sided error.

1.3.1 External Memory and Branch One

To understand the first branch of the trade-off, first consider the typical case on a RAM, when a word fits exactly one integer, i.e. $b = \ell$. In this case, the bound is $\log_\ell n$, which describes the performance of fusion trees [8].

To understand the case $b > \ell$, consider the external memory model with B words per page. This model has as a nonuniform counterpart the cell-probe model with cells of size $b = B\ell$. Observe that only the first branch of our trade-off depends on b. This branch is $\log_b n = \frac{\lg n}{\lg B + \lg \ell} = \Theta(\min\{\log_\ell n, \log_B n\})$. The first term describes the performance of fusion trees on a RAM with ℓ-bit words, as noted above. The second term matches the performance of the B-tree, the fundamental data structure in external memory.

Thus, we show that it is always optimal to either use a standard B-tree, or the best RAM algorithm which completely ignores the benefits of external memory. The RAM algorithm uses ℓ-bit words, and ignores the grouping of words into pages; this algorithm is the best of fusion trees and the algorithms from branches 2–4 of the trade-off. Thus, the standard comparison-based B-tree is the optimal use of external memory, even in a strong model of computation.

1.3.2 Polynomial Universes: Branch Two

For the rest of the discussion, assume the first branch (B-trees and fusion trees) does not give the minimum. Some of the most interesting consequences of our results can be seen in the very important special case when integers come from a polynomial universe, i.e. $\ell = O(\lg n)$. In this case, the optimal complexity is $\Theta(\lg \frac{\ell - \lg n}{a})$, as given by the second branch of the trade-off.

On the upper bound side, this is achieved by a simple elaboration of van Emde Boas' data structure. This data structure gives a way to reduce the key length from ℓ to $\frac{\ell}{2}$ in constant time, which immediately implies an upper bound of $O(\lg \ell)$. To improve that, first note that when $\ell \leq a$, we can stop the recursion and use complete tabulation to find the result. This means only $O(\lg \frac{\ell}{a})$ steps are needed. Another trivial idea, useful for near-linear universes, is to start with a table lookup based on the first $\lg n$ bits of the key, which requires linear space. Then, continue to apply van Emde Boas for keys of $\ell - \lg n$ bits inside each subproblem, giving a complexity of $O(\lg \frac{\ell - \lg n}{a})$.

Quite surprisingly, our lower bound shows that van Emde Boas' classic data structure, with these trivial tweaks, is optimal. In particular, when the space is not too far from linear (at most $n \cdot 2^{\lg^{1-\varepsilon} n}$) and $\ell \geq (1+\varepsilon)\lg n$, the standard van Emde Boas bound of $\Theta(\lg \ell)$ is optimal. It was often conjectured that this bound could be improved.

Note that with space $n^{1+\varepsilon}$, the optimal complexity for polynomial universes is constant. However, with space $n^{1+o(1)}$, the bound is $\omega(1)$, showing the claimed complexity-theoretic separations.

1.3.3 The Last Two Branches

The last two branches are relevant for superpolynomial universes, i.e. $\ell = \omega(\lg n)$. Comparing the two branches, we see the third one is better than the last one (up to constants) when $a = \Omega(\lg n)$. On the other hand, the last branch can be asymptotically better when $a = o(\lg n)$. This bound has the advantage that in the logarithm in the denominator, the factor $\frac{a}{\lg n}$, which is subconstant for $a = o(\lg n)$, is replaced by $1/\lg \frac{a}{\lg n}$.

The third branch is obtained by a careful application of the techniques of Beame and Fich [3], which can improve over van Emde Boas, but need large space. The last branch is also based on these techniques, combined with novel approaches tailored for small space.

1.3.4 Dynamic Updates

Lower bounds for near-linear space easily translate into interesting lower bounds for dynamic problems. If inserting an element takes time t_u, we can obtain a static data structure using space $O(n \cdot t_u)$ by simply simulating n inserts and storing the modified cells in a hash table. This transformation works even if updates are randomized, but, as before, we require that queries be deterministic. This model of randomized updates and deterministic queries is standard for hashing-based data structures. By the discussion above, as long as updates are reasonably fast, one cannot in general improve on the $O(\lg \ell)$ query time. It should be noted that van Emde Boas data structure can handle updates in the same time as queries, so this classic data structure is also optimal in the typical dynamic case, when one is concerned with the slowest operation.

1.4 Contributions

We now discuss our contributions in establishing the tight results of (1). Our main result is proving the tight lower bounds for $a = o(\lg n)$ (in particular, branches two and four of the trade-off). As mentioned already, previous techniques were helpless, since none could even differentiate $a = 2$ from $a = \lg n$.

Interestingly, we also show improved lower bounds for the case $a = \Omega(\lg n)$, in the classic communication framework. These improvements are relevant to the third branch of the trade-off. Assuming for simplicity that $a \leq \ell^{1-\varepsilon}$, our bound is $\min\{\frac{\lg n}{\lg \ell}, \frac{\lg \ell}{\lg\lg \ell + \lg(a/\lg n)}\}$, whereas the best previous lower bound was $\min\{\frac{\lg n}{\lg \ell}, \frac{\lg \ell}{\lg a}\}$. Our improved bound is based on a simple, yet interesting twist: instead of using the round elimination lemma alone, we show how to combine it with the *message compression lemma* of Chakrabarti and Regev [4]. Message compression is a refinement of round elimination, introduced by [4] to prove a lower bound for the approximate nearest neighbor problem. Sen and Venkatesh [14] asked whether message compression is really needed, or one could just use standard round elimination. Our result sheds an interesting light on this issue, as it shows message compression is even useful for classic predecessor lower bounds.

On the upper bound side, we only need to show the last two branches of the trade-off. As mentioned already, we use techniques of Beame and Fich [3]. The third bound was anticipated[1] by the second author in the concluding remarks of [15]. The last branch of (1), tailored specifically for small space, is based on novel ideas.

Organization. Due to space limitations, this extended abstract only contains the proof of our cell-probe lower bound in the simplest case $\ell = \gamma \lg n$, for constant $\gamma \geq 3$. We begin with a statement of our main technical result, the cell-probe elimination lemma, in Section 2.1. The rest of Section 2 uses this result to prove the predecessor lower bound. Section 3 then gives a proof of the cell-probe elimination lemma. The full version of this paper also contains the cell-probe tradeoffs for general ℓ, the improved communication-complexity lower bounds, and our tight upper bounds.

1.5 Direct-Sum Interpretations

A very strong consequence of our proofs is the idea that sharing between subproblems does not help for predecessor search. Formally, the best cell-probe complexity achievable by a data structure representing k independent subproblems (with the same parameters) in space $k \cdot \sigma$ is asymptotically equal to the best complexity achievable by a data structure for one subproblem, which uses space σ. The simplicity and strength of this statement make it interesting from both the data-structural and complexity-theoretic perspectives.

At a high level, it is precisely this sort of direct-sum property that enables us to beat communication complexity. Say we have k independent subproblems, and total space S. While in the communication game Alice sends $\lg S$ bits per round, our results intuitively state that $\lg \frac{S}{k}$ bits are sufficient. Then, by carefully controlling the increase in k and the decrease in key length (the query size), we can prevent Alice from communicating her entire input over a superconstant number of rounds.

A nice illustration of the strength of our result are the tight bounds for near linear universes, i.e. $\ell = \lg n + \delta$, with $\delta = o(\lg n)$. On the upper bound side, the algorithm can just start by a table lookup based on the first $\lg n$ bits of the key, which requires linear space. Then, it continues to apply van Emde Boas for δ-bit keys inside each subproblem, which gives a complexity of $O(\lg \frac{\delta}{a})$. Obtaining a lower bound is just as easy, given our techniques. We first consider $n/2^\delta$ independent subproblems, where each has 2^δ integers of 2δ bits each. Then, we prefix the integers in each subproblem by the number of the subproblem (taking $\lg n - \delta$ bits), and prefix the query with a random subproblem number. Because the universe of each subproblem ($2^{2\delta}$) is quadratically bigger than the number of keys, we can apply the usual proof showing the optimality of van Emde Boas' bound for polynomial universes. Thus, the complexity is $\Omega(\lg \frac{\delta}{a})$.

2. LOWER BOUNDS FOR SMALL SPACE

2.1 The Cell-Probe Elimination Lemma

An abstract decision data structure problem is defined by a function $f : D \times Q \to \{0,1\}$. An input from D is given at preprocessing time, and the data structure must store a representation of it in some bounded space. An input from Q is given at query time, and the function of the two inputs must be computed through cell probes. We restrict the preprocessing and query algorithms to be deterministic. In general, we consider a problem in conjunction with a distribution $\mathcal{D}$ over $D \times Q$. Note that the distribution need not (and, in our case, will not) be a product distribution. We care about the probability the query algorithm is successful under the distribution $\mathcal{D}$, for a notion of success to be defined shortly.

As mentioned before, we work in the cell-probe model, and let b be the number of bits in a cell. We assume the query's input consists of at most b bits, and that the space bound is at most 2^b. For the sake of an inductive argument, we extend the cell-probe model by allowing the data structure to publish some bits at preprocessing time. These are bits depending on the data structure's input, which the query algorithm can inspect at no charge. Closely related to this concept is our model for a query being accepted. We allow the query algorithm not to return the correct answer, but only in the following very limited way. After inspecting the query and the published bits, the algorithm can declare that it cannot answer the query (we say it *rejects* the query). Otherwise, the query is *accepted*: the algorithm can make cell probes, and at the end it must answer the query correctly. Thus, it is not possible to reject later. In contrast to more common models of error, it actually makes sense to talk about tiny (close to zero) probabilities of accept, even for problems with boolean output.

For an arbitrary problem f and an integer $k \leq 2^b$, we define a direct-sum problem $\bigoplus^k f : D^k \times ([k] \times Q) \to \{0,1\}$ as follows. The data structure receives a vector of inputs $(d^1, \dots, d^k)$. The representation depends arbitrarily on all of these inputs. The query is the index of a subproblem $i \in [k]$, and an element $q \in Q$. The output of $\bigoplus^k f$ is $f(q, d^i)$. We also define a distribution $\bigoplus^k \mathcal{D}$ for $\bigoplus^k f$, given a distribution $\mathcal{D}$ for f. Each d^i is chosen independently at random from the marginal distribution on D induced by $\mathcal{D}$.

[1] As a remark in [15, Section 7.5], it is stated that "it *appears* that we can get the following results. . . ", followed by bounds equivalent to the third branch of (1).

The subproblem i is chosen uniformly from $[k]$, and q is chosen from the distribution on Q conditioned on d^i.

Given an arbitrary problem f and an integer $h \leq b$, we can define another problem $f^{(h)}$ as follows. The query is a vector $(q_1, \ldots, q_h)$. The data structure receives a regular input $d \in D$, and integer $r \in [h]$ and the prefix of the query $q_1, \ldots, q_{r-1}$. The output of $f^{(h)}$ is $f(d, q_r)$. Note that we have shared information between the data structure and the querier (i.e. the prefix of the query), so $f^{(h)}$ is a partial function on the domain $D \times \bigcup_{i=0}^{t-1} Q^i \times Q$. Now we define an input distribution $\mathcal{D}^{(h)}$ for $f^{(h)}$, given an input distribution $\mathcal{D}$ for f. The value r is chosen uniformly at random. Each query coordinate q_i is chosen independently at random from the marginal distribution on Q induced by $\mathcal{D}$. Now d is chosen from the distribution on D, conditioned on q_r.

We give the $f^{(h)}$ operator precedence over the direct sum operator, i.e. $\bigoplus^k f^{(h)}$ means $\bigoplus^k \left[f^{(h)}\right]$. Using this notation, we are ready to state our central cell-probe elimination lemma:

Lemma 1. *There exists a universal constant C, such that for any problem f, distribution $\mathcal{D}$, and positive integers h and k, the following holds. Assume there exists a solution to $\bigoplus^k f^{(h)}$ with accept probability α over $\bigoplus^k \mathcal{D}^{(h)}$, which uses at most $k\sigma$ words of space, $\frac{1}{C}(\frac{\alpha}{h})^3 k$ published bits and T cell probes. Then, there exists a solution to $\bigoplus^k f$ with accept probability $\frac{\alpha}{4h}$ over $\bigoplus^k \mathcal{D}$, which uses the same space, $k \sqrt[h]{\sigma} \cdot Cb^2$ published bits and $T-1$ cell probes.*

2.2 Setup for the Predecessor Problem

Let $P(n, \ell)$ be the colored predecessor problem on n integers of ℓ bits each. Remember that this is the decision version of predecessor search, where elements are colored red or blue, and a query just returns the color of the predecessor. We first show how to identify the structure of $P(n, \ell)^{(h)}$ inside $P(n, h\ell)$, making it possible to apply our cell-probe elimination lemma.

Lemma 2. *For any integers $n, \ell, h \geq 1$ and distribution $\mathcal{D}$ for $P(n, \ell)$, there exists a distribution $\mathcal{D}^{*(h)}$ for $P(n, h\ell)$ such that the following holds. Given a solution to $\bigoplus^k P(n, h\ell)$ with accept probability α over $\bigoplus^k \mathcal{D}^{*(h)}$, one can obtain a solution to $\bigoplus^k P(n, \ell)^{(h)}$ with accept probability α over $\bigoplus^k \mathcal{D}^{(h)}$, which has the same complexity in terms of space, published bits, and cell probes.*

Proof. We give a reduction from $P(n, \ell)^{(h)}$ to $P(n, h\ell)$, which naturally defines the distribution $\mathcal{D}^{*(h)}$ in terms of $\mathcal{D}^{(h)}$. A query for $P(n, \ell)^{(h)}$ consists of $x_1, \ldots, x_h \in \{0, 1\}^\ell$. Concatenating these, we obtain a query for $P(n, h\ell)$. In the case of $P(n, \ell)^{(h)}$, the data structure receives $i \in [h]$, the query prefix $x_1, \ldots, x_{i-1}$ and a set Y of ℓ-bit integers. We prepend the query prefix to all integers in Y, and append zeros up to $h\ell$ bits. Then, finding the predecessor of x_i in Y is equivalent to finding the predecessor of the concatenation of $x_1, \ldots, x_h$ in this new set. □

Observe that to apply the cell-probe elimination lemma, the number of published bits must be just a fraction of k, but applying the lemma increases the published bits significantly. We want to repeatedly eliminate cell probes, so we need to amplify the number of subproblems each time, making the new number of published bits insignificant compared to the new k.

Lemma 3. *For any integers $t, \ell, n \geq 1$ and distribution $\mathcal{D}$ for $P(n, \ell)$, there exists a distribution $\mathcal{D}^{*t}$ for $P(n \cdot t, \ell + \lg t)$ such that the following holds. Starting from a solution to $\bigoplus^k P(n \cdot t, \ell + \lg t)$ with accept probability α over $\bigoplus^k \mathcal{D}^{*t}$, one can construct a solution to $\bigoplus^{kt} P(n, \ell)$ with accept probability α over $\bigoplus^{kt} \mathcal{D}$, which has the same complexity in terms of space, published bits, and cell probes.*

Proof. We first describe the distribution $\mathcal{D}^{*t}$. We draw $Y_1, \ldots, Y_t$ independently from $\mathcal{D}$, where Y_i is a set of integers, representing the data structures input. Prefix all numbers in Y_j by j using $\lg t$ bits, and take the union of all these sets to form the data structure's input for $P(nt, \ell + \lg t)$. To obtain the query, pick $j \in \{0, \ldots, t-1\}$ uniformly at random, pick the query from $\mathcal{D}$ conditioned on Y_j, and prefix this query by j. Now note that $\bigoplus^{kt} \mathcal{D}$ and $\bigoplus^k \mathcal{D}^{*t}$ are really the same distribution, except that the lower $\lg t$ bits of the problems index for $\bigoplus^{kt} \mathcal{D}$ are interpreted as a prefix in $\bigoplus^k \mathcal{D}^{*t}$. Thus, obtaining the new solution is simply a syntactic transformation. □

Our goal is to eliminate all cell probes, and then reach a contradiction. For this, we need the following:

Lemma 4. *For any $n \geq 1$ and $\ell \geq \log_2(n+1)$, there exists a distribution $\mathcal{D}$ for $P(n, \ell)$ such that the following holds. For all $(\forall) 0 < \alpha \leq 1$ and $k \geq 1$, there does not exist a solution to $\bigoplus^k P(n, \ell)$ with accept probability α over $\bigoplus^k \mathcal{D}$, which uses no cell probes and less than αk published bits.*

Proof. The distribution $\mathcal{D}$ is quite simple: the integers in the set are always 0 up to $n-1$, and the query is n. All that matters is the color of $n-1$, which is chosen uniformly at random among red and blue. Note that for $\bigoplus^k P(n, \ell)$ there are only k possible queries, i.e. only the index of the subproblem matters.

Let $\mathbf{p}$ be the random variable denoting the published bits. Since there are no cell probes, the answers to the queries are a function of $\mathbf{p}$ alone. Let $\alpha(p)$ be the fraction of subproblems that the query algorithm doesn't reject when seeing the published bits p. In our model, the answer must be correct for all these subproblems. Then, $\Pr[\mathbf{p} = p] \leq 2^{-\alpha(p)k}$, as only inputs which agree with the $\alpha(p)k$ answers of the algorithm can lead to these published bits. Now observe that $\alpha = \mathbf{E}_p[\alpha(p)] \leq \mathbf{E}_p\left[\frac{1}{k}\log_2 \frac{1}{\Pr[\mathbf{p}=p]}\right] = \frac{1}{k}H(\mathbf{p})$, where $H(\cdot)$ denotes binary entropy. Since the entropy of the published bits is bounded by their number (less than αk), we have a contradiction. □

2.3 Deriving the Trade-Offs

Because we will only be dealing with $\ell = b = O(\lg n)$, the bounds do not change if the space is S words instead of S bits. To simplify calculations, the exposition in this section assume the space is S words.

Our proof starts assuming that we for any possible distribution have a solution to $P(n, \ell)$ which uses $n \cdot 2^a$ space, no published bits, and successfully answers all queries in T probes, where T is small. We will then try to apply T *rounds* of the cell-probe elimination from Lemma 1 and 2 followed by the problem amplification from Lemma 3. After T rounds, we will be left with a non-trivial problem but no cell probes, and then we will reach a contradiction with Lemma 4. Below, we first run this strategy ignoring details about the distribution, but analyzing the parameters for each round. Later in Lemma 5, we will present a formal inductive proof using these parameters in reverse order, deriving difficult distributions for more and more cell probes.

We denote the problem parameters after i rounds by a subscript i. We have the key length ℓ_i and the number of subproblems k_i. The total number of keys remains n, so the have n/k_i keys in each subproblem. Thus, the problem we deal with in round $i+1$ is $\bigoplus^{k_i} P(\frac{n}{k_i}, \ell_i)$, and we will have some target accept probability α_i. The number of cells per subproblem is $\sigma_i = \frac{n}{k_i} 2^a$. We start the first round with $\ell_0 = \ell, \alpha_0 = 1, k_0 = 1$ and $\sigma_0 = n \cdot 2^a$.

For the cell probe elimination in Lemma 1 and 2, our proof will use the same value of $h \geq 2$ in all rounds. Then $\alpha_{i+1} \geq \frac{\alpha_i}{4h}$, so $\alpha_i \geq (4h)^{-i}$. To analyze the evolution of ℓ_i and k_i, we let t_i be the factor by which we increase the number of subproblems in round i when applying the problem amplification from Lemma 3. We now have $k_{i+1} = t_i \cdot k_i$ and $\ell_{i+1} = \frac{\ell_i}{h} - \lg t_i$.

When we start the first round, we have no published bits, but when we apply Lemma 1 in round $i+1$, it leaves us with up to $k_i \sqrt[h]{\sigma_i} \cdot Cb^2$ published bits for round $i+2$. We have to choose t_i large enough to guarantee that this number of published bits is small enough compared to the number of subproblems in round $i+2$. To apply Lemma 1 in round $i+2$, the number of published bits must be at most $\frac{1}{C}(\frac{\alpha_{i+1}}{h})^3 k_{i+1} = \frac{\alpha_i^3 t_i}{64Ch^6} k_i$. Hence we must set $t_i \geq \sqrt[h]{\sigma_i} \cdot 64C^2 b^2 h^6 (\frac{1}{\alpha_i})^3$. Assume for now that $T = O(\lg \ell)$. Using $h \leq \ell$, and $\alpha_i \geq (4h)^{-T} \geq 2^{O(\lg^2 \ell)}$, we conclude it is enough to set:

$$(\forall) i: \qquad t_i \geq \sqrt[h]{\frac{n}{k_i}} \cdot 2^{a/h} \cdot b^2 \cdot 2^{\Theta(\lg^2 \ell)} \tag{2}$$

Now we discuss the conclusion reached at the end of the T rounds. We intend to apply Lemma 4 to deduce that the algorithm after T stages cannot make zero cell probes, implying that the original algorithm had to make more than T probes. Above we made sure that we after T rounds had $\frac{1}{C}(\frac{\alpha_T}{h})^3 k_T < \alpha_T k_T$ published bits, which are few enough compared to the number k_T of subproblems. The remaining conditions of Lemma 4 are:

$$\ell_T \geq 1 \qquad \text{and} \qquad \frac{n}{k_T} \geq 1 \tag{3}$$

Since $\ell_{i+1} \leq \frac{\ell_i}{2}$, this condition entails $T = O(\lg \ell)$, as assumed earlier.

Lemma 5. *With the above parameters satisfying (2) and (3), for $i = 0, \ldots, T$, there is a distribution $\mathcal{D}_i$ for $P(\frac{n}{k_i}, \ell_i)$ so that no solution for $\bigoplus^{k_i} P(\frac{n}{k_i}, \ell_i)$ can have accept probability α_i over $\bigoplus^{k_i} \mathcal{D}_i$ using $n \cdot 2^a$ space, $\frac{1}{C}(\frac{\alpha_i}{h})^3 k_i$ published bits, and $T - i$ cell probes.*

Proof. The proof is by induction over $T - i$. A distribution that defies a good solution as in the lemma is called difficult. In the base case $i = T$, the space doesn't matter, and we get the difficult distribution directly from (3) and Lemma 4. Inductively, we use a difficult distribution $\mathcal{D}_i$ to construct a difficult distribution $\mathcal{D}_{i-1}$.

Recall that $k_i = k_{i-1} t_{i-1}$. Given our difficult distribution $\mathcal{D}_i$, we use the problem amplification in Lemma 3, to construct a distribution $\mathcal{D}_i^{*t_{i-1}}$ for $P(\frac{n}{k_i} \cdot t_{i-1}, \ell_i + \lg t_{i-1}) = P(\frac{n}{k_{i-1}}, \ell_i + \lg t_{i-1})$, which guarantees that no solution for $\bigoplus^{k_{i-1}} P(\frac{n}{k_{i-1}}, \ell_i + \lg t_{i-1})$ can have accept probability α_i over $\bigoplus^{k_{i-1}} \mathcal{D}_i^{*t_{i-1}}$ using $n \cdot 2^a$ space, $\frac{1}{C}(\frac{\alpha_i}{h})^3 k_i$ published bits, and $T - i$ cell probes.

Recall that (2) implies $k_{i-1} \sqrt[h]{\sigma_{i-1}} \cdot Cb^2 \leq \frac{1}{C}(\frac{\alpha_i}{h})^3 k_i$, hence that $k_{i-1} \sqrt[h]{\sigma_{i-1}}$ is less than the number of bits allowed published for our difficult distribution $\mathcal{D}_i^{*t_{i-1}}$. Also, recall that $\sigma_j k_j = n \cdot 2^a$ for all j. We can therefore use the cell probe elimination in Lemma 1, to construct a distribution $\left(\mathcal{D}_i^{*t_{i-1}}\right)^{(h)}$ for $P(\frac{n}{k_{i-1}}, \ell_i + \lg t_{i-1})^{(h)}$ so that no solution for $\bigoplus^{k_{i-1}} P(\frac{n}{k_{i-1}}, \ell_i + \lg t_{i-1})^{(h)}$ can have accept probability $\alpha_{i-1} \geq h\alpha_i$ over $\bigoplus^{k_{i-1}} \left(\mathcal{D}_i^{*t_{i-1}}\right)^{(h)}$ using $n \cdot 2^a$ space, $\frac{1}{C}(\frac{\alpha_{i-1}}{h})^3 k_{i-1}$ published bits, and $T - i + 1$ cell probes. Finally, using Lemma 2, we use $\left(\mathcal{D}_i^{*t_{i-1}}\right)^{(h)}$ to construct the desired difficult distribution $\mathcal{D}_{i-1}$ for $P(\frac{n}{k_{i-1}}, h(\ell_i + \lg t_{i-1})) = P(\frac{n}{k_{i-1}}, \ell_{i-1})$. □

We now show how to choose h and t_i in order to maximize the lower bound T, under the conditions of (2) and (3). In this extended abstract, we only consider the case $\ell = b = \gamma \lg n$, for constant $\gamma \geq 3$. In this case, it is enough to set $h = 2$ and $t_i = (\frac{n}{k_i})^{3/4}$. Then, $\frac{n}{k_{i+1}} = (\frac{n}{k_i})^{1/4}$, so $\lg \frac{n}{k_i} = 4^{-i} \lg n$ and $\lg t_i = \frac{3}{4} 4^{-i} \lg n$. By our recursion for ℓ_i, we have $\ell_{i+1} = \frac{\ell_i}{2} - \frac{3}{4} 4^{-i} \lg n$. Given $\ell_0 \geq 3 \lg n$, it can be seen by induction that $\ell_i \geq 3 \cdot 4^{-i} \lg n$. Indeed, $\ell_{i+1} \geq 3 \cdot 4^{-i} \cdot \frac{1}{2} \lg n - \frac{3}{4} 4^{-i} \lg n \geq 3 \cdot 4^{-(i+1)} \lg n$. By the above, (3) is satisfied for $T \leq \Theta(\lg \lg n)$. Finally, note that condition (2) is equivalent to:

$$\begin{aligned} & \lg t_i \;\geq\; \frac{1}{h} \lg \frac{n}{k_i} + \frac{a}{h} + \Theta(\lg b + \lg^2 \ell) \\ \Leftrightarrow\;\; & \frac{3}{4} 4^{-i} \lg n \;\geq\; \frac{1}{2} 4^{-i} \lg n + \frac{a}{2} + \Theta(\lg^2 \lg n) \\ \Leftrightarrow\;\; & T \;\leq\; \Theta\left(\lg \;\min\left\{\frac{\lg n}{a}, \frac{\lg n}{\lg^2 \lg n}\right\}\right) \;=\; \Theta\left(\lg \frac{\lg n}{a}\right) \end{aligned}$$

Since (2) and (3) are satisfied, we can apply Lemma 5 with $i = 0$ and the initial parameters $\ell_0 = b, \alpha_0 = 1, k_0 = 1$. We conclude that there is a difficult distribution $\mathcal{D}_0$ for $P(n, \ell)$ with no solution getting accept probability 1 using $n \cdot 2^a$ space, 0 published bits, and T cell probes. Thus we have proved:

Theorem 6. *In any solution to the static colored predecessor problem on n ℓ-bit keys, if $\ell = \gamma \lg n$ for constant $\gamma \geq 3$, and we are allowed $n \cdot 2^a$ space, then there are data instances for which some queries take $\Omega\left(\lg \frac{\lg n}{a}\right)$ cell probes.*

3. PROOF OF CELL-PROBE ELIMINATION

We assume a solution to $\bigoplus^k f^{(h)}$, and use it to construct a solution to $\bigoplus^k f$. The new solution uses the query algorithm of the old solution, but skips the first cell probe made by this algorithm. A central component of our construction is a structural property about any query algorithm for $\bigoplus^k f^{(h)}$ with the input distribution $\bigoplus^k \mathcal{D}^{(h)}$. We now define and claim this property. Section 3.1 uses it to construct a solution for $\bigoplus^k f$, while Section 3.2 gives the proof.

We first introduce some convenient notation. Remember that the data structure's input for $\bigoplus^k f^{(h)}$ consists of a vector $(d^1, \dots, d^k) \in D^k$, a vector selecting the interesting segments $(r^1, \dots, r^k) \in [h]^k$ and the query prefixes Q^i_j for all $j \in [r^i - 1]$. Denote by $\mathbf{d}, \mathbf{r}$ and $\mathbf{Q}$ the random variables giving these three components of the input. Also let $\mathbf{p}$ be the random variable representing the bits published by the data structure. Note that $\mathbf{p}$ can also be understood as a function $\mathbf{p}(\mathbf{d}, \mathbf{r}, \mathbf{Q})$. The query consists of an index i selecting the interesting subproblem, and a vector $(q_1, \dots, q_h)$ with a query to that subproblem. Denote by $\mathbf{i}$ and $\mathbf{q}$ these random variables. Note that in our probability space $\bigoplus^k f^{(h)}$, we have $\mathbf{q}_j = \mathbf{Q}^{\mathbf{i}}_j, (\forall) j < \mathbf{r}^{\mathbf{i}}$.

Fix some instance p of the published bits and a subproblem index $i \in [k]$. Consider a prefix $(q_1, \dots, q_j)$ for a query to this subproblem. Depending on $q_{j+1}, \dots, q_h$, the query algorithm might begin by probing different cells, or might reject the query. Let $\Gamma^i(p; q_1, \dots, q_j)$ be the set of cells that could be inspected by the first cell probe. Note that this set could be $\varnothing$, if all queries are rejected.

Now define:

$$\delta^i(p) = \begin{cases} 0, & \text{iff } \Gamma^i(p; \mathbf{Q}^i) = \varnothing \\ \Pr\left[|\Gamma^i(p; \mathbf{q}_1, \dots, \mathbf{q}_{\mathbf{r}^i})| \geq \frac{\min\{\sigma, |\Gamma^i(p; \mathbf{Q}^i)|\}}{\sqrt[h]{\sigma}} \mid \mathbf{i} = i \right] \end{cases} \tag{4}$$

The probability space is that defined by $\bigoplus^k \mathcal{D}^{(h)}$ when the query is to subproblem i. In particular, such a query will satisfy $\mathbf{q}_j = \mathbf{Q}^i_j, (\forall) j < \mathbf{r}^i$, because the prefix is known to the data structure. Note that this definition completely ignores the suffix $\mathbf{q}_{\mathbf{r}^i + 1}, \dots, \mathbf{q}_h$ of the query. The intuition behind this is that for any choice of the suffix, the correct answer to the query is the same, so this suffix can be "manufactured" at will. Indeed, an arbitrary choice of the suffix is buried in the definition of Γ^i.

With these observations, it is easier to understand (4). If the data structure knows that no query to subproblem i will be accepted, $\delta_i = 0$. Otherwise, we compare two sets of cells. The first contains the cells that the querier might probe given what the data structure knows: $\Gamma^i(p, \mathbf{Q}^i)$ contains all cells that could be probed for various $\mathbf{q}^i_{\mathbf{r}^i}$ and various suffixes. The second contains the cells that the querier could choose to probe considering its given input $\mathbf{q}^i_{\mathbf{r}^i}$ (the querier is only free to choose the suffix). Obviously, the second set is a subset of the first. The good case, whose probability is measured by δ_i, is when it is a rather large subset, or at least large compared to σ.

For convenience, we define $\delta^*(p) = \mathbf{E}_{\mathbf{i}}[\delta^{\mathbf{i}}(p)] = \frac{1}{k} \sum_i \delta^i(p)$. Using standard notation from probability theory, we write $\delta^i(p \mid E)$, when we condition on some event E in the probability of (4). We also write $\delta^i(p \mid X)$ when we condition on some random variable X, i.e. $\delta^i(p \mid X)$ is a function $x \mapsto \delta^i(p \mid X = x)$. We are now ready to state our claim, to be proven in Section 3.2.

Lemma 7. *There exist $\mathfrak{r}$ and $\mathfrak{Q}$, such that:*

$$\mathbf{E}_{\mathbf{d}}[\delta^*(\mathbf{p}(\mathfrak{r}, \mathfrak{Q}, \mathbf{d}) \mid \mathbf{r} = \mathfrak{r}, \mathbf{Q} = \mathfrak{Q}, \mathbf{d})] \geq \frac{\alpha}{2h}$$

3.1 The Solution for $\bigoplus^k f$

As mentioned before, we use the solution for $\bigoplus^k f^{(h)}$, and try to skip the first cell probe. To use this strategy, we need to extend an instance of $\bigoplus^k f$ to an instance of $\bigoplus^k f^{(h)}$. This is done using the $\mathfrak{r}$ and $\mathfrak{Q}$ values whose existence is guaranteed by Lemma 7. The extended data structure's input consists of the vector $(d^1, \dots, d^k)$ given to $\bigoplus^k f$, and the vectors $\mathfrak{r}$ and $\mathfrak{Q}$. A query's input for $\bigoplus^k f$ is a problem index $i \in [k]$ and a $q \in Q$. We extend this to $(q_1, \dots, q_h)$ by letting $q_j = \mathfrak{Q}^i_j, (\forall) j < \mathfrak{r}^i$, and $q_{\mathfrak{r}^i} = q$, and manufacturing a suffix $q_{\mathfrak{r}^i+1}, \dots, q_h$ as described below.

First note that extending an input of $\bigoplus^k f$ to an input of $\bigoplus^k f^{(h)}$ by this strategy preserves the desired answer to a query (in particular, the suffix is irrelevant to the answer). Also, this transformation is well defined because $\mathfrak{r}$ and $\mathfrak{Q}$ are "constants", defined by the input distribution $\bigoplus^k \mathcal{D}^{(h)}$. Since our model is nonuniform, we only care about the existence of $\mathfrak{r}$ and $\mathfrak{Q}$, and not about computational aspects.

To fully describe a solution to $\bigoplus^k f$, we must specify how to obtain the data structure's representation and the published bits, and how the query algorithm works. The data structure's representation is identical to the representation for $\bigoplus^k f^{(h)}$, given the extended input. The published bits for $\bigoplus^k f$ consist of the published bits for $\bigoplus^k f^{(h)}$, plus a number of published cells from the data structure's representation. Which cells are published will be detailed below. We publish the cell address together with its contents, so that the query algorithm can tell whether a particular cell is available.

The query algorithm is now simple to describe. Remember that $q_1, \dots, q_{\mathfrak{r}^i - 1}$ are prescribed by $\mathfrak{Q}$, and $q_{\mathfrak{r}^i} = q$ is the original input of $\bigoplus^k f$. We now iterate through all possible query suffixes. For each possibility, we simulate the extended query using the algorithm for $\bigoplus^k f^{(h)}$. If this algorithm rejects the query, or the first probed cell is not among the published cells, we continue trying suffixes. Otherwise, we stop, obtain the value for the first cell probe from the published cells and continue to simulate this query using actual cell probes. If we don't find any good suffix, we reject the query. It is essential that we can recognize accepts in the old algorithm by looking just at published bits. Then, searching for a suffix that would not be rejected is free, as it does not involve any cell probes.

3.1.1 Publishing cells

It remains to describe which cells the data structure chooses to publish, in order to make the query algorithm accept with the desired probability. Let p be the bits published by the $\bigoplus^k f^{(h)}$ solution. Note that in order for query (i, q) to be accepted, we must publish one cell from $\Gamma^i(p; \mathfrak{Q}^i, q)$. Here, we slightly abuse notation by letting $\mathfrak{Q}^i, q$ denote the $\mathfrak{r}^i$ entries of the prefix $\mathfrak{Q}^i$, followed by q. We will be able to achieve this for all (i, q) satisfying:

$$\Gamma^i(p; \mathfrak{Q}^i) \neq \varnothing, \quad |\Gamma^i(p; \mathfrak{Q}^i, q)| \geq \frac{\min\{\sigma, |\Gamma^i(p; \mathbf{Q}^i)|\}}{\sqrt[h]{\sigma}} \tag{5}$$

Comparing to (4), this means the accept probability is at least $\delta^*(p \mid \mathbf{r} = \mathfrak{r}, \mathbf{Q} = \mathfrak{Q}, \mathbf{d} = (d_1, \dots, d_k))$. Then on average over possible inputs $(d_1, \dots, d_k)$ to $\bigoplus^k f$, the accept probability will be at least $\frac{\alpha}{2h}$, as guaranteed by Lemma 7.

We will need the following standard result:

Lemma 8. *Consider a universe $U \neq \varnothing$ and a family of sets $\mathcal{F}$ such that $(\forall) S \in \mathcal{F}$ we have $S \subset U$ and $|S| \geq \frac{|U|}{B}$. Then there exists a set $T \subset U, |T| \leq B \ln |\mathcal{F}|$ such that $(\forall) S \in \mathcal{F}, S \cap T \neq \varnothing$.*

We distinguish three types of subproblems, parallel to (5). If $\Gamma^i(p;\mathfrak{Q}^i)=\varnothing$, we make no claim (the accept probability can be zero). Otherwise, if $|\Gamma^i(p;\mathfrak{Q}^i)|<\sigma$, we handle subproblem i using a local strategy. Consider all q such that $|\Gamma^i(p;\mathfrak{Q}^i,q)|\geq\frac{|\Gamma^i(p;\mathfrak{Q}^i)|}{\sqrt[h]{\sigma}}$. We now apply Lemma 8 with the universe $\Gamma^i(p;\mathfrak{Q}^i)$ and the family $\Gamma^i(p;\mathfrak{Q}^i,q)$, for all interesting q's. There are at most 2^b choices of q, bounding the size of the family. Then, the lemma guarantees that the data structure can publish a set of $O(\sqrt[h]{\sigma}\cdot b)$ cells which contains at least one cell from each interesting set. This means that each interesting q can be handled by the algorithm.

We handle the third type of subproblems, namely those with $|\Gamma^i(p;\mathfrak{Q}^i)|\geq\sigma$, in a global fashion. Consider all "interesting" pairs (i,q) with $|\Gamma^i(p;\mathfrak{Q}^i,q)|\geq\sigma^{1-1/h}$. We now apply Lemma 8 with the universe consisting of all $k\sigma$ cells, and the family being $\Gamma^i(p;\mathfrak{Q}^i,q)$, for interesting (i,q). The cardinality of the family is at most 2^b, since i and q form a query, which takes at most one word. Then by Lemma 8, the data structure can publish a set of $O(k\sqrt[h]{\sigma}\cdot b)$ cells, which contains at least one cell from each interesting set. With these cells, the algorithm can handle all interesting (i,q) queries.

The total number of cells that we publish is $O(k\sqrt[h]{\sigma}\cdot b)$. Thus, we publish $O(k\sqrt[h]{\sigma}\cdot b^2)$ new bits, plus $O(k)$ bits from the assumed solution to $\bigoplus^k f^{(h)}$. For big enough C, this is at most $k\sqrt[h]{\sigma}\cdot Cb^2$.

3.2 An Analysis of $\bigoplus^k f^{(h)}$: Proof of Lemma 7

Our analysis has two parts. First, we ignore the help given by the published bits, by assuming they are constantly set to some value p. As $\mathbf{r}^i$ and $\mathbf{Q}^i$ are chosen randomly, we show that the conditions of (4) are met with probability at least $\frac{1}{h}$ times the accept probability for subproblem i. This is essentially a lower bound on δ^i, and hence on δ^*.

Secondly, we show that the published bits do not really affect this lower bound on δ^*. The intuition is that there are two few published bits (much fewer than k) so for most subproblems they are providing no information at all. That is, the behavior for that subproblem is statistically close to when the published bits would not be used. Formally, this takes no more than a (subtle) application of Chernoff bounds. The gist of the idea is to consider some setting p for the published bits, and all possible inputs (not just those leading to p being published). In this probability space, δ^i are independent for different i, so the average is close to δ^* with overwhelmingly high probability. Now pessimistically assume all inputs where the average of δ^i is not close to δ^* are possible inputs, i.e. input for which p would be the real published bits. However, the probability of this event is so small, that even after a union bound for all p, it is still negligible.

We now proceed to the first part of the analysis. Let $\alpha^i(p)$ be the probability that the query algorithm accepts when receiving a random query for subproblem i. Formally, $\alpha^i(p)=\Pr[\Gamma^i(p;\mathbf{q})\neq\varnothing\mid\mathbf{i}=i]$. We define $\alpha^i(p\mid E),\alpha^i(p\mid X)$ and $\alpha^*(\cdot)$ similar to the functions associated to δ^i. Observe that the probability of correctness guaranteed by assumption is $\alpha=\mathbf{E}_{\mathbf{r},\mathbf{Q},\mathbf{d}}[\alpha^*(\mathbf{p}(\mathbf{r},\mathbf{Q},\mathbf{d})\mid\mathbf{r},\mathbf{Q},\mathbf{d})]$.

LEMMA 9. *For any i and p, we have $\delta^i(p)\geq\frac{\alpha^i(p)}{h}$.*

PROOF. Let us first recall the random experiment defining $\delta^i(p)$. We select a uniformly random $r\in[h]$ and random $q_1,\dots,q_{r-1}$. First we ask whether $\Gamma^i(p;q_1,\dots,q_{r-1})=\varnothing$. If not, we ask about the probability that a random q_r is good, in the sense of (4). Now let us rephrase the probability space as follows: first select $q_1,\dots,q_h$ at random; then select $r\in[h]$ and use just $q_1,\dots,q_r$ as above. The probability that the query $(q_1,\dots,q_h)$ is accepted is precisely $\alpha^i(p)$. Let's assume it doesn't. Then, for any r, $\Gamma^i(p;q_1,\dots,q_{r-1})\neq\varnothing$ because there is at least one suffix which is accepted. We will now show that there is at least one choice of r such that q_r is good when the prefix is $q_1,\dots,q_{r-1}$. When averaged over $q_1,\dots,q_{r-1}$, this gives a probability of at least $\frac{\alpha^i(p)}{h}$

To show one good r, let $\phi_r=\min\{|\Gamma^i(p;q_1,\dots,q_{r-1})|,\sigma\}$. Now observe that $\frac{\phi_1}{\phi_2}\cdot\frac{\phi_2}{\phi_3}\cdot\dots\cdot\frac{\phi_{h-1}}{\phi_h}=\frac{\phi_1}{\phi_h}\leq\phi_1\leq\sigma$. By the pigeonhole principle, $(\exists)r:\frac{\phi_r}{\phi_{r+1}}\leq\sigma^{1/h}$. This implies $|\Gamma^i(p;q_1,\dots,q_r)|\geq\frac{\min\{\sigma,|\Gamma^i(p;q_1,\dots,q_{r-1})|}{\sqrt[h]{\sigma}}$, as desired. □

Note that if the algorithm uses zero published bits, we are done. Thus, for the rest of the analysis we may assume $\frac{1}{C}(\frac{\alpha}{h})^3k\geq1$. We now proceed to the second part of the analysis, showing that δ^* is close to the lower bound of the previous lemma, even after a union bound over all possible published bits.

LEMMA 10. *With probability at least $1-\frac{\alpha}{8h}$ over random $\mathbf{r},\mathbf{Q}$ and $\mathbf{d}$, we have $(\forall)p:\ \delta^*(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})\geq\frac{\alpha^*(p)}{h}-\frac{\alpha}{4h}$*

PROOF. Fix p arbitrarily. By definition, $\delta^*(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})=\frac{1}{k}\sum_i\delta^i(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})$. By Lemma 9, $\mathbf{E}[\delta^i(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})]=\delta^i(p)\geq\frac{\alpha^i(p)}{h}$, which implies $\delta^*(p)\geq\frac{\alpha^*(p)}{h}$. Thus, our condition can be rephrased as:

$$\frac{1}{k}\sum_i\delta^i(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})\geq\mathbf{E}\left[\frac{1}{k}\sum_i\delta^i(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})\right]-\frac{\alpha}{4h}$$

Now note that $\delta^i(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})$ only depends on $\mathbf{r}^i,\mathbf{Q}^i$ and $\mathbf{d}^i$, since we are looking at the behavior of a query to subproblem i for a fixed value of the published bits; see the definition of δ^i in (4). Since $(\mathbf{r}^i,\mathbf{Q}^i,\mathbf{d}^i)$ are independent for different i, it follows that $\delta^i(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})$ are also independent. Then we can apply a Chernoff bound to analyze the mean $\delta^*(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})$ of these independent random variables. We use an additive Chernoff bound [2]:

$$\Pr_{\mathbf{r},\mathbf{Q},\mathbf{d}}\left[\delta^*(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})<\delta^*(p)-\frac{\alpha}{4h}\right]<e^{-\Omega(k(\frac{\alpha}{h})^2)}$$

Now we take a union bound over all possible choices p for the published bits. The probability of the bad event becomes $2^{\frac{1}{C}(\frac{\alpha}{h})^3k}e^{-\Omega((\frac{\alpha}{h})^2k)}$. For large enough C, this is $\exp(-\Omega((\frac{\alpha}{h})^2k))$, for any α and h. Now we use that $\frac{1}{C}(\frac{\alpha}{h})^3k\geq1$, from the condition that there is at lest one published bit, so this probability is at most $e^{-\Omega(Ch/\alpha)}$. Given that $\frac{h}{\alpha}\geq1$, this is at most $\frac{\alpha}{8h}$ for large enough C. □

Unfortunately, this lemma is not exactly what we would want, since it provides a lower bound in terms of $\alpha^*(p)$. This accept probability is measured in the original probability space. As we condition on $\mathbf{r},\mathbf{Q}$ and $\mathbf{d}$, the probability space can be quite different. However, we show next that in fact α^* cannot change too much. As before, the intuition is that there are too few published bits, so for most subproblems they are not changing the query distribution significantly.

LEMMA 11. *With probability at least $1-\frac{\alpha}{8}$ over random $\mathbf{r},\mathbf{Q}$ and $\mathbf{d}$, we have: $(\forall)p:\ \alpha^*(p\mid\mathbf{r},\mathbf{Q},\mathbf{d})\leq\alpha^*(p)+\frac{\alpha}{4}$*

Proof. The proof is very similar to that of Lemma 10. Fix p arbitrarily. By definition, $\alpha^*(p \mid \mathbf{r}, \mathbf{Q}, \mathbf{d})$ is the average of $\alpha^i(p \mid \mathbf{r}, \mathbf{Q}, \mathbf{d})$. Note that for fixed p, α^i depends only on $\mathbf{r}^i, \mathbf{Q}^i$ and $\mathbf{d}^i$. Hence, the α^i values are independent for different i, and we can apply a Chernoff bound to say the mean is close to its expectation. The rest of the calculation is parallel to that of Lemma 10. □

We combine Lemmas 10 and 11 by a union bound. We conclude that with probability at least $1 - \frac{\alpha}{4}$ over random $\mathbf{r}, \mathbf{Q}$ and $\mathbf{d}$, we have that $(\forall)p$:

$$\left.\begin{array}{l}\delta^*(p \mid \mathbf{r}, \mathbf{Q}, \mathbf{d}) \ge \frac{\alpha^*(p)}{h} - \frac{\alpha}{4h}\\ \alpha^*(p \mid \mathbf{r}, \mathbf{Q}, \mathbf{d}) \le \alpha^*(p) + \frac{\alpha}{4}\end{array}\right\} \Rightarrow$$

$$\Rightarrow \quad \delta^*(p \mid \mathbf{r}, \mathbf{Q}, \mathbf{d}) - \frac{\alpha^*(p \mid \mathbf{r}, \mathbf{Q}, \mathbf{d})}{h} \ge -\frac{\alpha}{2h} \qquad (6)$$

Since this holds for all p, it also holds for $p = \mathbf{p}$, i.e. the actual bits $\mathbf{p}(\mathbf{r}, \mathbf{Q}, \mathbf{d})$ published by the data structure given its input. Now we want to take the expectation over $\mathbf{r}, \mathbf{Q}$ and $\mathbf{d}$. Because $\delta^*(\cdot), \alpha^*(\cdot) \in [0, 1]$, we have $\delta^*(\cdot) - \frac{1}{h}\alpha^*(\cdot) \ge -\frac{1}{h}$. We use this as a pessimistic estimate for the cases of $\mathbf{r}, \mathbf{Q}$ and $\mathbf{d}$ where (6) does not hold. We obtain:

$$\begin{array}{rcl}\mathbf{E}\Big[\delta^*(\mathbf{p} \mid \mathbf{r}, \mathbf{Q}, \mathbf{d}) & - & \dfrac{\alpha^*(\mathbf{p} \mid \mathbf{r}, \mathbf{Q}, \mathbf{d})}{h}\Big] \ge -\dfrac{\alpha}{2h} - \dfrac{\alpha}{4} \cdot \dfrac{1}{h}\\ \Rightarrow \quad \mathbf{E}\big[\delta^*(\mathbf{p} \mid \mathbf{r}, \mathbf{Q}, \mathbf{d})\big] & \ge & \dfrac{1}{h}\mathbf{E}\big[\alpha^*(\mathbf{p} \mid \mathbf{r}, \mathbf{Q}, \mathbf{d})\big] - \dfrac{3\alpha}{4h}\\ & = & \dfrac{1}{h}\alpha - \dfrac{3\alpha}{4h} = \dfrac{\alpha}{4h}\end{array}$$

4. CONCLUSIONS

We have presented the cell-probe elimination lemma, a new technical result useful for showing cell-probe lower bounds. Using it, we have shown an $\Omega(\lg \frac{\lg n}{a})$ lower bound for predecessor search, when $b = \ell = \gamma \lg n$ for $\gamma > 3$. In the full version of this paper, we also prove the cell-probe trade-offs for general ℓ and b, the improved communication-complexity lower bounds, and our tight upper bounds.

Our cell-probe elimination lemma and the lower bounds derived by it hold for deterministic query algorithms. In recent work, we were able to extend our lower bound technique to randomized algorithms with two-sided error, showing that the trade-offs presented here are also tight in the randomized case.

An interesting question that remains unresolved is whether one can reduce the update time for dynamic data structures below the optimal search time, without a large blow-up in the search complexity. We conjecture this is impossible, but proving it appears beyond the scope of current techniques.

5. REFERENCES

[1] M. Ajtai. A lower bound for finding predecessors in Yao's cell probe model. *Combinatorica*, 8(3):235–247, 1988.

[2] N. Alon and J. Spencer. *The Probabilistic Method.* John Wiley, 2nd edition, 2000.

[3] P. Beame and F. E. Fich. Optimal bounds for the predecessor problem and related problems. *Journal of Computer and System Sciences*, 65(1):38–72, 2002. See also STOC'99.

[4] A. Chakrabarti and O. Regev. An optimal randomised cell probe lower bound for approximate nearest neighbour searching. In *Proc. 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 473–482, 2004.

[5] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink. Small forwarding tables for fast routing lookups. In *Proc. ACM SIGCOMM*, pages 3–14, 1997.

[6] A. Feldmann and S. Muthukrishnan. Tradeoffs for packet classification. In *Proc. IEEE INFOCOM*, pages 1193–1202, 2000.

[7] M. L. Fredman and M. E. Saks. The cell probe complexity of dynamic data structures. In *Proc. 21st ACM Symposium on Theory of Computing (STOC)*, pages 345–354, 1989.

[8] M. L. Fredman and D. E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993. See also STOC'90.

[9] A. Gál and P. B. Miltersen. The cell probe complexity of succinct data structures. In *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 332–344, 2003.

[10] P. B. Miltersen. The bit probe complexity measure revisited. In *10th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 662–671, 1993.

[11] P. B. Miltersen. Lower bounds for Union-Split-Find related problems on random access machines. In *26th ACM Symposium on Theory of Computing (STOC)*, pages 625–634, 1994.

[12] P. B. Miltersen. Cell probe complexity - a survey. In *19th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 1999. Advances in Data Structures Workshop.

[13] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998. See also STOC'95.

[14] P. Sen and S. Venkatesh. Lower bounds for predecessor searching in the cell probe model. *arXiv:cs.CC/0309033*. See also ICALP'01, CCC'03, 2003.

[15] M. Thorup. Space efficient dynamic stabbing with fast queries. In *Proc. 35th ACM Symposium on Theory of Computing (STOC)*, pages 649–658, 2003.

[16] P. van Emde Boas, R. Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. *Mathematical Systems Theory*, 10:99–127, 1977. Announced by van Emde Boas alone at FOCS'75.

[17] D. E. Willard. Log-logarithmic worst-case range queries are possible in space $\Theta(N)$. *Information Processing Letters*, 17(2):81–84, 1983.

[18] A. C.-C. Yao. Should tables be sorted? *Journal of the ACM*, 28(3):615–628, 1981. See also FOCS'78.