2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Higher Cell Probe Lower Bounds for Evaluating Polynomials

Kasper Green Larsen
MADALGO, Department of Computer Science
Aarhus University
Aarhus, Denmark
Email: larsen@cs.au.dk

***Abstract*—In this paper, we study the cell probe complexity of evaluating an n-degree polynomial P over a finite field $\mathbf{F}$ of size at least $n^{1+\Omega(1)}$. More specifically, we show that any static data structure for evaluating $P(x)$, where $x \in \mathbf{F}$, must use $\Omega(\lg|\mathbf{F}|/\lg(Sw/n\lg|\mathbf{F}|))$ cell probes to answer a query, where S denotes the space of the data structure in number of cells and w the cell size in bits. This bound holds in expectation for randomized data structures with any constant error probability $\delta < 1/2$. Our lower bound not only improves over the $\Omega(\lg|\mathbf{F}|/\lg S)$ lower bound of Miltersen [TCS'95], but is in fact the highest static cell probe lower bound to date: For linear space (i.e. $S = O(n\lg|\mathbf{F}|/w)$), our query time lower bound simplifies to $\Omega(\lg|\mathbf{F}|)$, whereas the highest previous lower bound for any static data structure problem having d different queries is $\Omega(\lg d/\lg\lg d)$, which was first achieved by Pătraşcu and Thorup [SICOMP'10].**

We also use the recent technique of Larsen [STOC'12] to show a lower bound of $t_q = \Omega(\lg|\mathbf{F}|\lg n/\lg(wt_u/\lg|\mathbf{F}|)\lg(wt_u))$ for dynamic data structures for polynomial evaluation over a finite field $\mathbf{F}$ of size $\Omega(n^2)$. Here t_q denotes the expected query time and t_u the worst case update time. This lower bound holds for randomized data structures with any constant error probability $\delta < 1/2$. This is only the second time a lower bound beyond $\max\{t_u, t_q\} = \Omega(\max\{\lg n, \lg d/\lg\lg d\})$ has been achieved for dynamic data structures, where d denotes the number of different queries and updates to the problem. Furthermore, it is the first such lower bound that holds for randomized data structures with a constant probability of error.

***Keywords*-cell probe model, lower bounds, data structures, polynomials**

I. Introduction

The field of data structure lower bounds has received much attention over the last decades. As a consequence, numerous models of computation have emerged over the years, in one way or another trying to capture the inherent difficulties of various problems. These models include for instance the semi-group model, pointer machine model and the cell probe model. While the community has been extremely successful in understanding the complexity of data structure problems in the semi-group model and pointer machine model, progress has been considerably slower in the more general cell probe model.

In this paper, we study the cell probe complexity of evaluating polynomials. More specifically, we study the *polynomial evaluation problem*. In this data structure problem, the input consists of an n-degree polynomial $P(x) = a_n x^n + \cdots + a_1 x + a_0$ with coefficients $a_n, \ldots, a_0$ drawn from a finite field $\mathbf{F}$. The goal is to preprocess P into a data structure, such that given an element $x_0 \in \mathbf{F}$, one can efficiently compute $P(x_0)$ over the field $\mathbf{F}$. The cell probe lower bound we obtain for polynomial evaluation not only improves over the previous highest lower bound for the problem, but is in fact the highest lower bound to date for any static data structure problem. In addition, our lower bound holds also for randomized data structures with a constant error probability $\delta < 1/2$.

We also prove a cell probe lower bound for a dynamic version of the polynomial evaluation problem. Here we think of P as being represented by its n roots $r_n, \ldots, r_1$, i.e. $P(x) = (x - r_n)(x - r_{n-1})\cdots(x - r_1)$[1]. The goal is to support updating the roots, i.e. given an element $y_0 \in \mathbf{F}$ and an index $i \in \{1, \ldots, n\}$, we must support updating r_i to y_0. Initially, we have $r_n = \cdots = r_1 = 0$. A query to this problem still asks to evaluate $P(x_0)$ for a query element $x_0 \in \mathbf{F}$. The lower bound we prove for this problem holds for randomized data structures with a constant probability of error $\delta < 1/2$, and is the highest cell probe lower bound to date for dynamic data structures that are allowed to err.

A. The Cell Probe Model

In the cell probe model, a static data structure consists of a memory of cells, each containing w bits that may represent arbitrary information about the input. The memory cells all have an integer address amongst $[2^w] = \{0, \ldots, 2^w - 1\}$ and we say that the data structure uses S cells of space if only cells of addresses $\{0, \ldots, S-1\}$ are used. A common assumption that we also make is that a cell has enough bits to address the input, i.e. we assume $w = \Omega(\lg n)$, where n is the input size.

Kasper Green Larsen is supported in part by a Google Europe Fellowship in Search and Information Retrieval, and in part by MADALGO – Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation.

[1]Note that if the field is not algebraically closed, it is not possible to represent all polynomials over the field in this form. Since we are considering lower bounds, this is not an issue.

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.21

When presented with a query, a static data structure reads (probes) a number of cells from the memory, and at the end must announce the answer to the query. The cell probed at each step may be any function of the query and the contents of the previously probed cells, thus all computations on read data are free of charge. The *worst case query cost* is defined as the maximum over all pairs of an input I and a query q, of the number of cells probed when answering q on input I. Clearly lower bounds in this model also apply to data structures developed in one of the most popular and least restrictive upper bound models, the word-RAM.

A dynamic data structure in the cell probe model must also support updates. When presented with an update, a dynamic data structure may both read and write to memory cells. We refer to reading or writing a cell jointly as probing the cell. Again, the cells probed and the contents written to cells during an update may be arbitrary functions of the previously probed cells and the update itself. The worst case cost of answering a query is defined as for static data structures, but where the maximum is over pairs of queries and update sequences. The *worst case update cost* is defined as the maximum over all pairs of an update u and a sequence of updates U, of the number of cells probed when performing the update u after processing the updates U.

Randomization: In this paper, we also consider randomized data structures. When answering queries, a randomized data structure is given access to a stream of uniform random bits. The cells probed when answering queries are allowed to depend also on this random stream. The *expected query cost* is defined as the maximum over all pairs of a query q and an input I (update sequence U), of the expected number of cells probed when answering q on input I (after processing the updates U).

Furthermore, we allow randomized data structures to return an incorrect result when answering queries. We define the *error probability* of a randomized data structure as the maximum over all pairs of an input I (update sequence U) and a query q, of the probability of returning an incorrect result when answering q on input I (after processing the updates U).

By a standard reduction, any randomized data structure with a constant probability of error $\delta < 1/2$ and expected query cost t, can be transformed into a randomized data structure with the same error probability and worst case query cost $O(t)$, see Section II. Hence, when stating query cost lower bounds for randomized data structures with a constant error probability $\delta < 1/2$, we omit whether the lower bound is for the expected query cost or the worst case query cost.

B. Previous Results

In the following, we first give a brief overview of the previous techniques and highest lower bounds obtained in the field of static and dynamic cell probe lower bounds. We finally present the previous results on polynomial evaluation.

Static Data Structures: Many outstanding results and techniques were proposed in the years following Yao's introduction of the cell probe model [24]. One of the most notable papers from that time was the paper by Miltersen et al. [15], relating asymmetric communication complexity and static data structures. During the next decade, a large number of results followed from their techniques, mainly related to predecessor search, partial match and other nearest neighbor type problems [1], [13], [3], [4], [12], [2], [22]. Unfortunately these techniques could not distinguish near-linear space data structures from data structures using polynomial space, thus for problems where the number of queries is polynomial in the input size, all these results gave no query cost lower bounds beyond $\Omega(1)$, even for linear space data structures (there are at most $n^{O(1)}$ queries, which is trivially solved in polynomial space and constant query time).

This barrier was not overcome until the milestone papers of Pătraşcu and Thorup [20], [21]. The technique they introduced has since then evolved (see e.g. [18]) into an elegant refinement of Miltersen et al.'s reduction from static data structures to asymmetric communication complexity, and it has triggered a renewed focus on static lower bounds, see e.g. [17], [18], [23], [8], [9]. Their results pushed the highest achieved query lower bound to $\Omega(\lg d/\lg\lg d)$ for data structures using $n\lg^{O(1)} d$ cells of space, where d is the number of different queries to the data structure problem. This lower bound was proved also for randomized data structures with any constant error probability $\delta < 1/2$. Their technique thus provided the first non-trivial lower bounds when $d = n^{O(1)}$, and their results remains until today the highest achieved lower bounds for any explicit problem. We note that their technique cannot be used to prove lower bounds beyond $\Omega(\lg d/\lg\lg d)$, even for linear space deterministic data structures.

Recently, Panigrahy et al. [16] presented another technique for proving static cell probe lower bounds. Their technique is based on sampling cells of the data structure instead of relying on communication complexity. Using this technique, they reproved the bounds of Pătraşcu and Thorup [21] for various nearest neighbor search problems. We note that the idea of sampling cells has appeared before in the world of succinct data structures, see e.g. the papers by Gál and Miltersen [6] and Golynski [7].

Dynamic Data Structures: Lower bounds for dynamic data structures have almost exclusively been proved by appealing to the seminal chronogram technique of Fredman and Saks [5]. The basic idea is to divide a sequence of n updates into *epochs* of exponentially decreasing size. From these epochs, one partitions the cells of a data structure into subsets, one for each epoch i. The subset associated to an epoch i contains the cells that where last updated when processing the updates of epoch i. Lower bounds now

follow by arguing that to answer a query after the n updates, one has to probe $\Omega(1)$ cells associated to each epoch. For technical reasons, the epoch sizes have to decrease by a factor of at least wt_u, where t_u is the worst case update cost of the data structure. Thus one obtains lower bounds no higher than $t_q = \Omega(\lg n/\lg(wt_u))$, where t_q is the expected query cost of the data structure. This bound peaks at $\max\{t_u, t_q\} = \Omega(\lg n/\lg\lg n)$ for any polylogarithmic cell size. We note that by minor modifications of these ideas, the same bound can be achieved when t_u is the *amortized* update cost.

The bounds achieved using the chronogram technique remained unchallenged until the seminal paper of Pătraşcu and Demaine [19]. They presented an extension of the chronogram technique, allowing them to obtain lower bounds of $\max\{t_u, t_q\} = \Omega(\lg n)$, where t_u is the amortized update cost and t_q the expected query cost. They applied their technique to several fundamental problems, including *partial sums* and *dynamic connectivity*.

Very recently, Larsen [11] showed how to combine the cell sampling approach of Panigrahy et al. [16] with the chronogram technique of Fredman and Saks [5]. This combination essentially allows one to argue that when answering a query, one has to probe $\Omega(\lg n/\lg(wt_u))$ cells from each epoch instead of $\Omega(1)$, yielding lower bounds of $t_q = \Omega((\lg n/\lg(wt_u))^2)$, where t_u is the worst case update cost and t_q the expected query cost. Larsen applied his technique to the *weighted range counting* problem in 2-d and noted that the technique is limited to proving lower bounds for problems where the number of bits in the output of a query is more than the number of bits needed to describe a query.

Polynomial Evaluation: The particular problem of polynomial evaluation has seen a rather large amount of attention, in particular from a lower bound perspective. Miltersen [14] was the first to prove cell probe lower bounds for polynomial evaluation over a finite field $\mathbf{F}$. His lower bound states that $t = \Omega(\lg|\mathbf{F}|/\lg S)$ whenever $|\mathbf{F}|$ is at least $n^{1+\varepsilon}$ for an arbitrarily small constant $\varepsilon > 0$. Here t is the worst case query cost. This lower bound unfortunately degenerates to $t = \Omega(1)$ for $|\mathbf{F}| = n^{O(1)}$. In [6], Gál and Miltersen considered *succinct* data structures for polynomial evaluation. Succinct data structures are data structures that use space very close to the information theoretic minimum required for storing the input (in this case $(n+1)\lg|\mathbf{F}|$ bits). In this setting, they showed that any data structure for polynomial evaluation must satisfy $tr = \Omega(n)$ when $|\mathbf{F}| \geq (1+\varepsilon)n$ for any constant $\varepsilon > 0$. Here t is the worst case query cost and r is the *redundancy*, i.e. the *additive* number of extra bits of space used by the data structure compared to the information theoretic minimum. If data structures are allowed *non-determinism* (i.e. they can guess the right cells to probe), then Yin [25] proved a lower bound matching that of Miltersen [14].

On the upper bound side, Kedlaya and Umans [10] showed that there exists a static word-RAM data structure (and hence cell probe data structure) for polynomial evaluation, having space usage $n^{1+\varepsilon}\lg^{1+o(1)}|\mathbf{F}|$ and worst case query cost $\lg^{O(1)} n\lg^{1+o(1)}|\mathbf{F}|$ for any constant $\varepsilon > 0$.

C. Our Results

In this paper, we further investigate the cell sampling technique proposed by Panigrahy et al. [16]. Surprisingly, we show that with a small modification, the technique is more powerful than the communication complexity framework of Pătraşcu and Thorup [21]. More specifically, we apply the technique to the static polynomial evaluation problem and obtain a query cost lower bound of $\Omega(\lg|\mathbf{F}|/\lg(Sw/n\lg|\mathbf{F}|))$ when $|\mathbf{F}|$ is at least $n^{1+\varepsilon}$ for an arbitrarily small constant $\varepsilon > 0$. This lower bound holds for randomized data structure with any constant error probability $\delta < 1/2$. For linear space data structures (i.e. $S = O(n\lg|\mathbf{F}|/w)$), this bound simplifies to $\Omega(\lg|\mathbf{F}|)$. This is the highest static cell probe lower bound to date, and is a $\lg\lg|\mathbf{F}|$ factor larger than what can possibly be achieved using the communication framework. Furthermore, our lower bound gives the first non-trivial bounds for polynomial evaluation when $|\mathbf{F}| = n^{O(1)}$, improving over the result of Miltersen [14].

Secondly, we apply the recent technique of Larsen [11] to obtain a lower bound of $t_q = \Omega(\lg|\mathbf{F}|\lg n/\lg(wt_u/\lg|\mathbf{F}|)\lg(wt_u))$ for the dynamic polynomial evaluation problem over a field of size at least $\Omega(n^2)$. Here t_u is the worst case update time and t_q is the query cost of any randomized data structure with a constant error probability $\delta < 1/2$. This is the first time a lower bound beyond $\max\{t_u, t_q\} = \Omega(\max\{\lg n, \lg d/\lg\lg d\})$ has been proved for dynamic data structures that are allowed to err. Here d denotes the maximum of the number of different queries and different updates to the data structure problem. Furthermore, Larsen's lower bound [11] for weighted range counting peaks at $\max\{t_u, t_q\} = \Omega((\lg n/\lg\lg n)^2)$ whereas this lower bound increases even further if $|\mathbf{F}|$ is super-polynomial in n.

II. Static Polynomial Evaluation

In this section, we prove a cell probe lower bound for static polynomial evaluation. In the proof we consider polynomial evaluation over a finite field $\mathbf{F}$, where we assume $|\mathbf{F}| = n^{1+\Omega(1)}$.

We are aiming at proving a lower bound for randomized data structures with error probability δ and expected query cost t, where δ is an arbitrary constant less than $1/2$. So assume the availability of such a randomized data structure. To ease calculations, we first modify this data structure in the following standard way: When presented with a query, we repeat the query algorithm a sufficiently large constant number of times and then return the majority answer. By this

procedure, we have effectively obtained a randomized data structure with error probability at most $1/19$, while maintaining the same asymptotic expected query cost. Secondly, we modify the data structure by letting it return an arbitrary answer whenever the query algorithm does not terminate within a number of steps that is bounded by a sufficiently large constant times the expected query cost. By Markov's inequality, this yields a randomized data structure with error probability at most $1/18$ and worst case query cost $O(t)$.

Finally, by fixing the random bits, this implies the existence of a deterministic data structure with error probability at most $1/18$ and worst case query cost $O(t)$, where the error probability of the deterministic data structure is defined as the probability that it returns an incorrect result when answering a uniform random query on a uniform random input polynomial.

We show that such a deterministic data structure must have worst case query cost $\Omega(\lg|\mathbf{F}|/\lg(Sw/n\lg|\mathbf{F}|))$, which completes the proof.

Notation: We let $\mathbf{P}$ denote a random variable giving a uniform random n-degree polynomial with coefficients in the field $\mathbf{F}$, i.e. each of the $n+1$ coefficients of $\mathbf{P}$ is a uniform random element from $\mathbf{F}$. Clearly $H(\mathbf{P}) = (n+1)\lg|\mathbf{F}|$, where $H(\cdot)$ denotes binary entropy.

An Encoding Proof: Assume for contradiction that a deterministic data structure solution for polynomial evaluation over $\mathbf{F}$, using S cells of space, with worst case query cost $t = o(\lg|\mathbf{F}|/\lg(Sw/n\lg|\mathbf{F}|))$ and error probability $1/18$ exists. Assume furthermore $|\mathbf{F}| = n^{1+\Omega(1)}$. Under this assumption, we show how to encode $\mathbf{P}$ using less than $H(\mathbf{P}) = (n+1)\lg|\mathbf{F}|$ bits in expectation, a contradiction. Following Panigrahy et al. [16], the high level idea of the encoding procedure is to implement the claimed data structure on $\mathbf{P}$. Letting $D(\mathbf{P})$ denote the set of cells stored by the data structure on input $\mathbf{P}$, we then find a subset of cells that *resolves* a large number of queries which do not err, and hence the cell set reveals much information about $\mathbf{P}$. Here we say that a set of cells $C \subseteq D(\mathbf{P})$ resolves a query q, if the query algorithm probes only cells in C when answering q on input $\mathbf{P}$. If the found set of cells can be described in fewer bits than the resolved queries reveal about $\mathbf{P}$, this gives the contradiction. The following is our main technical result:

Lemma 1. *With probability at least $1/2$ over the choice of $\boldsymbol{P}$, there exists a set of cells $\boldsymbol{C} \subseteq D(\boldsymbol{P})$ and an integer $\boldsymbol{t}^*$, where $1 \leq \boldsymbol{t}^* \leq t$, which satisfy all of the following properties:*

1) $|\boldsymbol{C}| = n\lg|\boldsymbol{F}|/5w$.
2) *Let $G^{\boldsymbol{C}}_{\boldsymbol{t}^*}(\boldsymbol{P})$ be the set of queries that succeed on input $\boldsymbol{P}$ (*good *queries), where furthermore each query q in $G^{\boldsymbol{C}}_{\boldsymbol{t}^*}(\boldsymbol{P})$ is resolved by $\boldsymbol{C}$ on input $\boldsymbol{P}$ and the query algorithm probes exactly $\boldsymbol{t}^*$ cells when answering q on input $\boldsymbol{P}$. Then $|G^{\boldsymbol{C}}_{\boldsymbol{t}^*}(\boldsymbol{P})| = |\boldsymbol{F}|^{1-o(1)} = n^{1+\Omega(1)}$.*
3) *Similarly, let $B^{\boldsymbol{C}}_{\boldsymbol{t}^*}(\boldsymbol{P})$ be the set of queries that err on input $\boldsymbol{P}$ (*bad *queries), where furthermore each query q in $B^{\boldsymbol{C}}_{\boldsymbol{t}^*}(\boldsymbol{P})$ is resolved by $\boldsymbol{C}$ and the query algorithm probes exactly $\boldsymbol{t}^*$ cells when answering q on input $\boldsymbol{P}$. Then $|B^{\boldsymbol{C}}_{\boldsymbol{t}^*}(\boldsymbol{P})| \leq |G^{\boldsymbol{C}}_{\boldsymbol{t}^*}(\boldsymbol{P})|/2$.*

Before giving the proof of Lemma 1, we show how we use it in the encoding and decoding procedures. We first give a high-level interpretation of Lemma 1: Examining the lemma, we see that for half of all possible input polynomials, the claimed (too fast) data structure must contain a set of cells $\mathbf{C}$, such that $\mathbf{C}$ can be described in less than $H(\mathbf{P})$ bits, and at the same time, many queries can be answered solely from the contents of the cells in $\mathbf{C}$. Now observe that knowing the answer to an evaluation query provides a point on the polynomial $\mathbf{P}$. Hence from $\mathbf{C}$, we can recover $|G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})| = n^{1+\Omega(1)}$ points on the polynomial $\mathbf{P}$. Since an n-degree polynomial over a field is uniquely determined from any $n+1$ distinct points on the polynomial, it follows that (ignoring the queries that err for now) $\mathbf{P}$ is uniquely determined from $\mathbf{C}$, which gives a contradiction since $\mathbf{C}$ can be described in fewer than $H(\mathbf{P})$ bits. The remaining parts of the lemma ensure that we can recover $\mathbf{P}$ even when facing a number of queries that err. The details of this will become apparent in the encoding and decoding procedures below.

We note that to prove Lemma 1, we have to extend on the ideas in Panigrahy et al. [16] since their cell sampling technique would leave us with a set $\mathbf{C}$ of size a factor t larger than what we obtain. Readjusting parameters, this would loose a factor $\lg\lg|\mathbf{F}|$ in the lower bound and bring us back to what can be achieved using the communication approach. We discuss this further when we give the proof of Lemma 1.

Encoding Algorithm: The algorithm for encoding $\mathbf{P}$ does the following:

1) First we construct the claimed data structure on $\mathbf{P}$ and obtain the set of cells $D(\mathbf{P})$. If for every integer $1 \leq \mathbf{t}^* \leq t$, $D(\mathbf{P})$ does not contain a set of cells $\mathbf{C}$ satisfying the properties of Lemma 1, we simply encode $\mathbf{P}$ as a 0-bit followed by a naive encoding of $\mathbf{P}$, taking a total of $1 + \lceil(n+1)\lg|\mathbf{F}|\rceil \leq 2 + H(\mathbf{P})$ bits.
2) If the integer $1 \leq \mathbf{t}^* \leq t$ and the cell set $\mathbf{C}$ does exist, we first write a 1-bit. We then encode both $\mathbf{t}^*$ and $\mathbf{C}$, including addresses and contents of the cells in $\mathbf{C}$, for a total of $1+\lg t+|\mathbf{C}|(w+\lg S) \leq 3|\mathbf{C}|w \leq 3/5 \cdot H(\mathbf{P})$ bits.

This completes the encoding procedure. Next we show how to recover $\mathbf{P}$ from the encoding:

Decoding Algorithm: To recover the polynomial $\mathbf{P}$ from the above encoding, we do the following: We start by examining the first bit. If this is a 0, we immediately recover $\mathbf{P}$ from the remaining part of the encoding. If the first bit is 1, we obtain the set of cells $\mathbf{C}$ and the integer $\mathbf{t}^*$ from the encoding. We now simulate the query algorithm

for each of the $|\mathbf{F}|$ possible queries. For each such query q, if the query algorithm requests a cell outside $\mathbf{C}$ we simply discard q. Otherwise we recover the contents of the requested cell from the encoding and continue the simulation until we either discard the query or obtain the answer to it (possibly incorrect answer). Once this procedure is done we are left with all the queries that are resolved by $\mathbf{C}$. We now prune this set by deleting all queries where the query algorithm did not probe exactly $\mathbf{t}^*$ cells. We are then left with the set $G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P}) \cup B^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})$, including the correct answer to each query in $G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})$ and an incorrect answer to each query in $B^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})$ (but we do not know which queries belong to each set). We finally iterate through all possible input polynomials and return as our candidate for $\mathbf{P}$, the polynomial which agrees with the most of the answers we have obtained for queries in $G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P}) \cup B^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})$.

To see that the returned polynomial in fact is $\mathbf{P}$, first recall the standard fact that any n-degree polynomial over a field is uniquely determined from $n+1$ distinct points on the polynomial. This implies that any two distinct polynomials over the input field $\mathbf{F}$ can agree on the answer to at most n evaluation queries. Thus any polynomial different from $\mathbf{P}$ can agree with at most n of the answers obtained for queries in $G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})$ and possibly all answers obtained for queries in $B^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})$. By Lemma 1, this is bounded by $n+|B^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})| \leq n+|G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})| - |G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})|/2 = |G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})| + n - n^{1+\Omega(1)} < |G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})|$ query answers. But $\mathbf{P}$ agrees on all answers in $G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})$ and hence it follows that the returned polynomial indeed is $\mathbf{P}$.

Analysis: Invoking Lemma 1, we get that the encoding uses at most

$$1/2 \cdot (2 + H(\mathbf{P})) + 1/2 \cdot 3/5 \cdot H(\mathbf{P}) < 9/10 \cdot H(\mathbf{P})$$

bits in expectation, i.e. a contradiction.

Proof of Lemma 1: In this paragraph, we prove the main technical result, namely Lemma 1. As noted, our approach differs slightly from that of Panigrahy et al. [16]. In their paper, they find a set of cells resolving many queries by picking t random samples of cells, one for each cell probe by the data structure. Our key idea is to pick one sample for all t probes simultaneously. This small difference is crucial to obtain the improved lower bounds when the space is less than a factor t from linear.

First, by Markov's inequality, we get that with probability at least $1/2$, there are at most $2 \cdot |\mathbf{F}|/18 = |\mathbf{F}|/9$ queries that err on input $\mathbf{P}$. When this happens, we show that there exists $\mathbf{t}^*$ and $\mathbf{C}$ satisfying all the properties of Lemma 1. This boils down to counting arguments:

We first choose $\mathbf{t}^*$. For this, initialize a candidate set $T = \{1, \ldots, t\}$ of values for $\mathbf{t}^*$. Now define $G(\mathbf{P})$ as the set of queries that succeed on input $\mathbf{P}$ and similarly define $B(\mathbf{P})$ as the set of queries that err on input $\mathbf{P}$. By assumption we have $|B(\mathbf{P})| \leq |\mathbf{F}|/9$ and hence $|G(\mathbf{P})| \geq 8/9 \cdot |\mathbf{F}| \geq 8|B(\mathbf{P})|$. Examine each $i \in T$ in turn and collect for each choice the set $G_i(\mathbf{P})$, consisting of all queries in $G(\mathbf{P})$ that probe exactly i cells on input $\mathbf{P}$. For each i where $|G_i(\mathbf{P})| < |G(\mathbf{P})|/2t$, we remove i from T, i.e. we set $T \leftarrow T \setminus \{i\}$. After this step, we have $\sum_{i \in T} |G_i(\mathbf{P})| \geq |G(\mathbf{P})|/2$ and $|G_i(\mathbf{P})| \geq |G(\mathbf{P})|/2t$ for each $i \in T$.

Next, we examine each remaining $i \in T$ and remove all such i where $|B_i(\mathbf{P})| > |G_i(\mathbf{P})|/4$. Here $B_i(\mathbf{P})$ is the set of queries in $B(\mathbf{P})$ that probe exactly i cells on input $\mathbf{P}$. Since

$$\sum_{i \in T} |G_i(\mathbf{P})| \geq |G(\mathbf{P})|/2 \geq 4|B(\mathbf{P})| \geq 4 \sum_{i \in T} |B_i(\mathbf{P})|,$$

it follows that T is non-empty after this pruning step. We let $\mathbf{t}^*$ equal an arbitrary remaining value in T, thus we have $|G_{\mathbf{t}^*}(\mathbf{P})| \geq 4|B_{\mathbf{t}^*}(\mathbf{P})|$ and $|G_{\mathbf{t}^*}(\mathbf{P})| \geq |G(\mathbf{P})|/2t$.

We find $\mathbf{C}$ in a similar fashion. For ease of notation, define $\Delta = n \lg |\mathbf{F}|/5w$. First initialize a candidate set Y, containing all Δ-sized subsets of cells in $D(\mathbf{P})$. We thus have $|Y| = \binom{S}{\Delta}$. For a set $C' \in Y$, we define $G^{C'}_{\mathbf{t}^*}(\mathbf{P})$ ($B^{C'}_{\mathbf{t}^*}(\mathbf{P})$) as the subset of queries in $G_{\mathbf{t}^*}(\mathbf{P})$ ($B_{\mathbf{t}^*}(\mathbf{P})$) that are resolved by C', i.e. they probe only cells in C' on input $\mathbf{P}$. Observe that each query in $G_{\mathbf{t}^*}(\mathbf{P})$ and $B_{\mathbf{t}^*}(\mathbf{P})$ is resolved by precisely $\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}$ sets in Y, hence $\sum_{C' \in Y} |G^{C'}_{\mathbf{t}^*}(\mathbf{P})| = |G_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}$ and $\sum_{C' \in Y} |B^{C'}_{\mathbf{t}^*}(\mathbf{P})| = |B_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}$.

We now prune Y by deleting all sets $C' \in Y$ for which $|G^{C'}_{\mathbf{t}^*}(\mathbf{P})| < |G_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}/2\binom{S}{\Delta}$. We then have $\sum_{C' \in Y} |G^{C'}_{\mathbf{t}^*}(\mathbf{P})| \geq |G_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}/2$ and $|G^{C'}_{\mathbf{t}^*}(\mathbf{P})| \geq |G_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}/2\binom{S}{\Delta}$ for all remaining $C' \in Y$.

As the last step, we remove all $C' \in Y$ for which $|B^{C'}_{\mathbf{t}^*}(\mathbf{P})| > |G^{C'}_{\mathbf{t}^*}(\mathbf{P})|/2$. Again, since

$$\begin{aligned}
\sum_{C' \in Y} |G^{C'}_{\mathbf{t}^*}(\mathbf{P})| &\geq |G_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}/2 \\
&\geq 2|B_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*} \\
&\geq 2\sum_{C' \in Y} |B^{C'}_{\mathbf{t}^*}(\mathbf{P})|,
\end{aligned}$$

we conclude that Y is non-empty after this step, and we let $\mathbf{C}$ equal an arbitrary remaining set. We have thus obtained $\mathbf{t}^*$ and $\mathbf{C}$ satisfying

1)

$$\begin{aligned}
|G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})| &\geq |G_{\mathbf{t}^*}(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}/2\binom{S}{\Delta} \\
&\geq |G(\mathbf{P})|\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}/4t\binom{S}{\Delta}.
\end{aligned}$$

2)

$$|B^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})| \leq |G^{\mathbf{C}}_{\mathbf{t}^*}(\mathbf{P})|/2.$$

Lemma 1 now follows since

$$\begin{aligned}|G(\mathbf{P})|\frac{\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}}{4t\binom{S}{\Delta}} &\geq 2|\mathbf{F}|\frac{\binom{S-\mathbf{t}^*}{\Delta-\mathbf{t}^*}}{9t\binom{S}{\Delta}}\\ &= |\mathbf{F}|\frac{2(S-\mathbf{t}^*)!\Delta!}{9tS!(\Delta-\mathbf{t}^*)!}\\ &\geq |\mathbf{F}|\frac{2(\Delta-\mathbf{t}^*)^{\mathbf{t}^*}}{9tS^{\mathbf{t}^*}}\\ &\geq |\mathbf{F}|\left(\frac{n\lg|\mathbf{F}|}{6Sw}\right)^t = |\mathbf{F}|^{1-o(1)},\end{aligned}$$

where the last step followed from the contradictory assumption that $t = o(\lg|\mathbf{F}|/\lg(Sw/n\lg|\mathbf{F}|))$. We finally conclude:

Theorem 1. *Any static cell probe data structure for evaluating an n-degree polynomial over a finite field $\boldsymbol{F}$ must have query cost $t = \Omega(\lg|\boldsymbol{F}|/\lg(Sw/n\lg|\boldsymbol{F}|))$ if $\boldsymbol{F}$ has size at least $n^{1+\Omega(1)}$. Here S is the space usage in number of cells and w is the cell size. This lower bound holds for randomized data structures with any constant error probability $\delta < 1/2$. For linear space (i.e. $S = O(n\lg|\boldsymbol{F}|/w)$), this lower bound simplifies to $t = \Omega(\lg|\boldsymbol{F}|)$.*

III. Dynamic Polynomial Evaluation

In this section, we prove a lower bound for dynamic polynomial evaluation over a finite field $\mathbf{F}$, where we assume $|\mathbf{F}| = \Omega(n^2)$. We furthermore assume there is some arbitrary, but fixed ordering on the elements of $\mathbf{F}$.

We obtain the lower bound by giving a hard distribution over sequences of updates and then bound the expected query cost of any deterministic data structure with error probability at most $1/18$ over the distribution. To be precise, the expectation in the query cost is measured over a uniform random choice of query and a sequence of updates drawn from the hard distribution. Similarly, the error probability is defined as the probability that the data structure returns an incorrect answer on a uniform random query and a sequence of updates drawn from the hard distribution. By arguments similar to those in Section II, this translates into an equivalent lower bound on the expected query cost for randomized data structures with any constant error probability $\delta < 1/2$.

The first step of the proof is thus to design a hard distribution over updates, followed by a uniform random query.

Hard Distribution: The hard distribution is simple: We first execute n updates, where the ith update sets the ith root of the maintained polynomial to a uniform random element from $\mathbf{F}$. Following the n updates, we execute an evaluation query at a uniform random element in $\mathbf{F}$. This concludes the hard distribution.

We use $\mathbf{x}_i$ to denote the random variable giving the uniform random element from $\mathbf{F}$ that is used in the ith update operation. We let $\mathbf{X} = \mathbf{x}_1 \cdots \mathbf{x}_n$ be the random variable giving all the n updates. Finally we let $\mathbf{q}$ denote the uniform random element in $\mathbf{F}$ that is used in the query.

High-Level Proof: For the remainder of this section, we assume the availability of a deterministic data structure for dynamic polynomial evaluation, having worst case update time t_u and error probability $1/18$ over the hard distribution. Our goal is to lower bound the expected query cost of this data structure.

For this, conceptually divide the updates $\mathbf{X} = \mathbf{x}_1 \cdots \mathbf{x}_n$ into $\lg_\beta n$ *epochs* of size β^i for $i = 0, \ldots, \lg_\beta n - 1$, where $\beta = (wt_u)^2$. Epoch 0 consists of the last update $\mathbf{x}_n$, and generally epoch i consists of updates $\mathbf{x}_{n_i+1}, \ldots, \mathbf{x}_{n_i+\beta^i}$ where $n_i = n - \sum_{j=0}^{i}\beta^j$.

We let $\mathbf{X}_i = \mathbf{x}_{n_i+1} \cdots \mathbf{x}_{n_i+\beta^i}$ denote the random variable giving the updates of epoch i. For a sequence of updates $\mathbf{X}$, we define $D(\mathbf{X})$ as the set of cells stored by the available data structure after the sequence of updates $\mathbf{X}$. We additionally partition $D(\mathbf{X})$ into sets $D_i(\mathbf{X})$ for $i = 0, \ldots, \lg_\beta n - 1$, where $D_i(\mathbf{X})$ consists of the subset of cells in $D(\mathbf{X})$ that was updated in epoch i, but not in epochs $j < i$, i.e. $D_i(\mathbf{X})$ is the set of cells last updated in epoch i. Finally, we define $t_i(\mathbf{X}, \mathbf{q})$ as the number of cells in $D_i(\mathbf{X})$ that is probed by the query algorithm when answering $\mathbf{q}$ after the updates $\mathbf{X}$. With this notation, our goal is to show

Lemma 2. *If $\beta = (wt_u)^2$, then $\mathbb{E}[t_i(\boldsymbol{X}, \boldsymbol{q})] = \Omega(\lg|\boldsymbol{F}|/\lg(wt_u/\lg|\boldsymbol{F}|))$ for all epochs $i \geq 1$.*

Before giving the proof of Lemma 2, we show that it immediately gives the claimed lower bound. Since the cells sets $D_i(\mathbf{X})$ are disjoint, we get that the number of cells probed when answering $\mathbf{q}$ is at least $\sum_i t_i(\mathbf{X}, \mathbf{q})$ (due to rounding, there are some updates happening before epoch $\lg_\beta n - 1$, therefore at least and not exactly). It now follows from linearity of expectation that the expected number of cells probed when answering $\mathbf{q}$ is $\Omega(\lg|\mathbf{F}|/\lg(wt_u/\lg|\mathbf{F}|) \cdot \lg_\beta n) = \Omega(\lg|\mathbf{F}|/\lg(wt_u/\lg|\mathbf{F}|) \cdot \lg n/\lg(wt_u))$.

What remains is thus to prove Lemma 2, which will be the focus of Section III-A.

A. Bounding the Probes to Epoch i (Proof of Lemma 2)

To prove Lemma 2 we assume for contradiction that the available data structure satisfies $\mathbb{E}[t_{i^*}(\mathbf{X}, \mathbf{q})] = o(\lg|\mathbf{F}|/\lg(wt_u/\lg|\mathbf{F}|))$ for some epoch $i^* \geq 1$. Now observe that $H(\mathbf{X} \mid \mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}}) = H(\mathbf{X}) - n_{i^*}\lg|\mathbf{F}| = (n - n_{i^*})\lg|\mathbf{F}| \geq \beta^{i^*}\lg|\mathbf{F}|$. We finish the proof by showing that, conditioned on $\mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}}$, we can use the claimed data structure to encode $\mathbf{X}$ in less than $H(\mathbf{X} \mid \mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}})$ bits in expectation, i.e. a contradiction. As a technical detail, note that in contrast to Section II, we are encoding a sequence of updates and not just a polynomial, thus it is important that the encoding not only recovers the polynomial corresponding to the updates $\mathbf{X}$, but it must also recover the *ordering* of the updates.

Before giving the encoding and decoding procedures, we present the main technical lemma. This lemma shows exactly

what happens if the data structure probes too few cells from epoch i^*.

Lemma 3. *Let $i^* \geq 1$ be an epoch for which $\mathbb{E}[t_{i^*}(\boldsymbol{X}, \boldsymbol{q})] = o(\lg|\boldsymbol{F}|/\lg(wt_u/\lg|\boldsymbol{F}|))$. Partition the field $\boldsymbol{F}$ into $|\boldsymbol{F}|^{1/4}$ consecutive groups of $|\boldsymbol{F}|^{3/4}$ elements, denoted $\boldsymbol{F}_1, \ldots, \boldsymbol{F}_{|\boldsymbol{F}|^{1/4}}$ (based on the ordering on $\boldsymbol{F}$). Then there exists a choice of index $1 \leq j^* \leq |\boldsymbol{F}|^{1/4}$ such that with probability at least $1/2$ over the choice of $\boldsymbol{X}$, there exists a subset of cells $\boldsymbol{C}_{i^*} \subseteq D_{i^*}(\boldsymbol{X})$ such that the following holds*

- *All updates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are unique, i.e. $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ for $i \neq j$.*
- *$|\boldsymbol{C}_{i^*}| \leq 1/25 \cdot \beta^{i^*} \lg|\boldsymbol{F}|/w$.*
- *Let $G_{j^*}^{\boldsymbol{C}_{i^*}}(\boldsymbol{X})$ denote the subset of queries from $\boldsymbol{F}_{j^*}$ that do not err on input $\boldsymbol{X}$, and where furthermore the query algorithm probes no cells in $D_{i^*}(\boldsymbol{X}) \setminus \boldsymbol{C}_{i^*}$ when answering a query in $G_{j^*}^{\boldsymbol{C}_{i^*}}(\boldsymbol{X})$ after the updates $\boldsymbol{X}$. Then $|G_{j^*}^{\boldsymbol{C}_{i^*}}(\boldsymbol{X})| = |\boldsymbol{F}|^{74/100-o(1)} \geq n+1$.*

As with Lemma 1, this lemma shows that if a data structure is too efficient, then there must exist a small subset of cells that solves many queries. However, in the dynamic setting, we need several additional properties to obtain a contradiction. As demonstrated by Larsen [11], we can reach a contradiction by encoding a subset of queries for which to simulate the query algorithm and obtain the corresponding answers. Since the answer to an evaluation query reveals at most $\lg|\mathbf{F}|$ bits, encoding these queries must take less than $\lg|\mathbf{F}|$ bits per query to obtain a contradiction. This is the reason why we focus on one particular subset of queries $\mathbf{F}_{j^*}$: Since $|\mathbf{F}_{j^*}| = |\mathbf{F}|^{3/4}$, encoding queries from $\mathbf{F}_{j^*}$ can be done in $\lg|\mathbf{F}|^{3/4}$ bits. This buys us $\lg|\mathbf{F}| - \lg|\mathbf{F}|^{3/4} = 1/4\lg|\mathbf{F}|$ bits per query, which is enough to reach the contradiction.

We defer the proof of Lemma 3 to the end of the section, and instead move on to show how we use Lemma 3 in the encoding and decoding procedures.

Encoding: Given the sequence of updates $\mathbf{X}$, we first construct the claimed data structure on $\mathbf{X}$ and obtain the cell set $D(\mathbf{X})$ as well as the partitions into subsets $D_{\lg_\beta n-1}(\mathbf{X}), \ldots, D_0(\mathbf{X})$. We now encode $\mathbf{X}$ using the following simple procedure:

1) We first examine $D(\mathbf{X})$ to determine whether a cell set $\mathbf{C}_{i^*} \subseteq D_{i^*}(\mathbf{X})$ and a query set $G_{j^*}^{\mathbf{C}_{i^*}}(\mathbf{X})$ exists, satisfying all the properties of Lemma 3. This is done simply by trying all choices for $\mathbf{C}_{i^*}$ and $G_{j^*}^{\mathbf{C}_{i^*}}(\mathbf{X})$ (note that j^* is the same for all choices of $\mathbf{X}$, thus it is assumed known both in the encoding and decoding procedure). If such sets do not exist, or if $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are not all distinct, we first write a 0-bit, followed by a naive encoding of updates $\mathbf{x}_{n_{i^*}+1}, \ldots, \mathbf{x}_n$, taking a total of $1 + \lceil (n - n_{i^*}) \lg|\mathbf{F}| \rceil \leq 2 + H(\mathbf{X} \mid \mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}})$ bits.
2) If the claimed sets $\mathbf{C}_{i^*}$ and $G_{j^*}^{\mathbf{C}_{i^*}}(\mathbf{X})$ exists and $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are all distinct, we instead start by writing a 1-bit. We then examine $G_{j^*}^{\mathbf{C}_{i^*}}(\mathbf{X})$ and find a subset, $\mathbf{Q}_{i^*} \subseteq G_{j^*}^{\mathbf{C}_{i^*}}(\mathbf{X})$, of $\beta^{i^*}+1$ queries, such that none of the queries in $\mathbf{Q}_{i^*}$ evaluate the polynomial at one of the roots that was set during the updates of epochs $j \neq i^*$, i.e. no query in $\mathbf{Q}_{i^*}$ is amongst $\mathbf{x}_1, \ldots, \mathbf{x}_{n_{i^*}}, \mathbf{x}_{n_{i^*-1}+1}, \ldots, \mathbf{x}_n$. Since $G_{j^*}^{\mathbf{C}_{i^*}}(\mathbf{X})$ contains at least $n+1$ elements, we can always find such a set of queries. We now write down a description of $\mathbf{C}_{i^*}$ and $\mathbf{Q}_{i^*}$, including addresses and contents of the cells in $\mathbf{C}_{i^*}$. Accounting for also writing down $|\mathbf{C}_{i^*}|$ this takes a total of $1 + w + |\mathbf{C}_{i^*}|2w + \lg\binom{|\mathbf{F}|^{3/4}}{|\mathbf{Q}_{i^*}|} \leq 3/25 \cdot \beta^{i^*} \lg|\mathbf{F}| + \lg\binom{|\mathbf{F}|^{3/4}}{\beta^{i^*}+1}$ bits (here we exploit that $G_{j^*}^{\mathbf{C}_{i^*}}(\mathbf{X}) \subseteq \mathbf{F}_{j^*}$ to get the exponent $3/4$).
3) Next we write down all updates following epoch i^*, $\mathbf{x}_{n_{i^*-1}+1}, \ldots, \mathbf{x}_n$, and all cell sets $D_{i^*-1}(\mathbf{X}), \ldots, D_0(\mathbf{X})$. This takes another $\sum_{j=0}^{i^*-1} O(|D_j(\mathbf{X})|w + \beta^j \lg|\mathbf{F}|) = O(\beta^{i^*-1}(\lg|\mathbf{F}| + wt_u))$ bits.
4) In the last step, we consider the sorted sequence of the updates in epoch i^*, i.e. sorted by the ordering in $\mathbf{F}$ of the values they assign the roots, and not by the time of executing the updates. From this sequence, we write down the permutation that brings the sequence back into sorted order of execution time, taking a total of $\lceil \lg(\beta^{i^*}!) \rceil$ bits.

Decoding: In the following we show how to recover $\mathbf{X}$ from the encoding. Recall that we are recovering $\mathbf{X}$ conditioned on the updates preceding epoch i^*, i.e. we are given access to $\mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}}$ when recovering $\mathbf{X}$. The decoding procedure does the following:

1) We start by examining the first bit of the encoding. If this is a 0-bit, we immediately recover $\mathbf{X}$ from the remainder of the encoding and the given updates $\mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}}$.
2) If the first bit is a 1, we start by executing updates $\mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}}$ on the claimed data structure. From this, we obtain cell sets $D^*_{\lg_\beta n-1}(\mathbf{X}), \ldots, D^*_{i^*+1}(\mathbf{X})$, where $D^*_i(\mathbf{X})$ denotes the set of cells that were updated in epoch i but not during epochs $i-1, \ldots, i^*+1$. Note that $D_i(\mathbf{X}) \subseteq D^*_i(\mathbf{X})$ for $i = \lg_\beta n - 1, \ldots, i^*+1$. From the encoding, we furthermore recover $D_{i^*-1}(\mathbf{X}), \ldots, D_0(\mathbf{X})$ as well as updates $\mathbf{x}_{n_{i^*-1}+1}, \ldots, \mathbf{x}_n$. Finally, we recover $\mathbf{C}_{i^*}$ and $\mathbf{Q}_{i^*}$ from the encoding.
3) The next step is to recover the answer to each query in $\mathbf{Q}_{i^*}$ as if the query was executed after all updates $\mathbf{X}$. For this, we examine each query in $\mathbf{Q}_{i^*}$ in turn. For a query $q \in \mathbf{Q}_{i^*}$, we execute the query algorithm of the claimed data structure. For each cell c that is requested by the query algorithm, we first examine cell sets $D_{i^*-1}(\mathbf{X}), \ldots, D_0(\mathbf{X})$ and if c is contained in any of them, we have immediately recovered the contents of

c as it is in $D(\mathbf{X})$. If c is not in $D_{i^*-1}(\mathbf{X}), \ldots, D_0(\mathbf{X})$ we continue by examining $\mathbf{C}_{i^*}$. If c is contained therein, we have again recovered the contents of c in $D(\mathbf{X})$ and we continue executing the query algorithm. If c is also not in $\mathbf{C}_{i^*}$, then since $\mathbf{Q}_{i^*} \subseteq G_{i^*}^{\mathbf{C}_{i^*}}(\mathbf{X})$ we know by Lemma 3 that c is not in $D_{i^*}(\mathbf{X}) \backslash \mathbf{C}_{i^*}$. Therefore, the contents of c has not been updated during epochs $i^*, \ldots, 0$ and therefore we recover the contents of c in $D(\mathbf{X})$ from $D^*_{\lg_\beta n-1}(\mathbf{X}), \ldots, D^*_{i^*+1}(\mathbf{X})$. Thus regardless of which cell the query algorithm requests, we can recover the contents as it is in $D(\mathbf{X})$. It follows that the query algorithm terminates with the answer to q after updates $\mathbf{X}$. Finally, since no queries in $\mathbf{Q}_{i^*}$ err on input $\mathbf{X}$, we conclude that the recovered answers are also correct.

4) Recovering $\mathbf{X}$ is now straightforward. From updates $\mathbf{x}_1, \ldots, \mathbf{x}_{n_{i^*}}$ and $\mathbf{x}_{n_{i^*-1}+1}, \ldots, \mathbf{x}_n$ we know $n - \beta^{i^*}$ points on the polynomial $P(\mathbf{X})$ corresponding to $\mathbf{X}$ since all $\mathbf{x}_i$'s are unique. Since the queries in $\mathbf{Q}_{i^*}$ do not evaluate $P(\mathbf{X})$ at $\mathbf{x}_1, \ldots, \mathbf{x}_{n_{i^*}}, \mathbf{x}_{n_{i^*-1}+1}, \ldots, \mathbf{x}_n$ and $|\mathbf{Q}_{i^*}| = \beta^{i^*} + 1$, the answers to queries in $\mathbf{Q}_{i^*}$ give us another $\beta^{i^*} + 1$ points on $P(\mathbf{X})$. Hence $P(\mathbf{X})$ is uniquely determined from the encoding. From $P(\mathbf{X})$, we find the β^{i^*} roots that are not amongst $\mathbf{x}_1, \ldots, \mathbf{x}_{n_{i^*}}, \mathbf{x}_{n_{i^*-1}+1}, \ldots, \mathbf{x}_n$, i.e. we find the β^{i^*} unique elements of $\mathbf{F}$ corresponding to $\mathbf{x}_{n_{i^*}+1}, \ldots, \mathbf{x}_{n_{i^*-1}}$, but we do not know how they are permuted in $\mathbf{X}$. We finally recover $\mathbf{X}$ from the encoding of how to permute these elements.

Analysis: In the following, we bound the expected size of the encoding and finally reach a contradiction. We start by analysing the size of the encoding when the conditions of Lemma 3 are satisfied. In this case, we write down a total of

$$\begin{aligned} &3/25 \cdot \beta^{i^*} \lg |\mathbf{F}| + \lg \binom{|\mathbf{F}|^{3/4}}{\beta^{i^*}+1} \\ &+O(\beta^{i^*-1}(\lg |\mathbf{F}| + wt_u)) + \lg(\beta^{i^*}!) \end{aligned}$$

bits. Since $\beta = (wt_u)^2$, this is bounded by

$$\begin{aligned} &3/25 \cdot \beta^{i^*} \lg |\mathbf{F}| + (\beta^{i^*}+1)\lg(|\mathbf{F}|^{3/4}/\beta^{i^*}) \\ &+O(\beta^{i^*} \lg |\mathbf{F}|/wt_u) + \beta^{i^*} \lg(\beta^{i^*}) + O(1). \end{aligned}$$

This is again upper bounded by

$$\beta^{i^*}(\lg |\mathbf{F}|^{3/4} + 3/25 \lg |\mathbf{F}| + o(\lg |\mathbf{F}|))$$

bits, which finally gives

$$(87/100 + o(1))\beta^{i^*} \lg |\mathbf{F}| \leq 9/10 H(\mathbf{X} \mid \mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}})$$

bits. From Lemma 3, we get that this is the amount of bits spend with probability at least $1/2$, hence the expected size of the encoding is at most

$$\begin{aligned} &1/2 \cdot (2 + H(\mathbf{X} \mid \mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}})) \\ &+1/2 \cdot (9/10 H(\mathbf{X} \mid \mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}})) \\ &< H(\mathbf{X} \mid \mathbf{x}_1 \cdots \mathbf{x}_{n_{i^*}}), \end{aligned}$$

i.e. a contradiction.

Proof of Lemma 3: To prove Lemma 3, let $i^* \geq 1$ be an epoch in which $\mathbb{E}[t_{i^*}(\mathbf{X}, \mathbf{q})] = o(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$. We first partition $\mathbf{F}$ into the $|\mathbf{F}|^{1/4}$ consecutive groups $\mathbf{F}_1, \ldots, \mathbf{F}_{|\mathbf{F}|^{1/4}}$ of $|\mathbf{F}|^{3/4}$ elements each (i.e. based on the ordering on $\mathbf{F}$). We choose j^* in the following way:

Let δ_j denote the error probability of the data structure when restricted to the queries in $\mathbf{F}_j$, i.e. δ_j is the probability of returning an incorrect result when answering a query drawn uniformly from $\mathbf{F}_j$ after a sequence of updates drawn from the hard distribution. Clearly $\sum_j \delta_j/|\mathbf{F}|^{1/4} \leq 1/18$ since the error probability over all queries is at most $1/18$. Also let $t^j_{i^*}(\mathbf{X})$ denote the average number of cells probed by the queries in $\mathbf{F}_j$ on input $\mathbf{X}$, i.e. $t^j_{i^*}(\mathbf{X}) = \sum_{q \in \mathbf{F}_j} t_{i^*}(\mathbf{X}, q)/|\mathbf{F}_j|$. We similarly have $\sum_j \mathbb{E}[t^j_{i^*}(\mathbf{X})]/|\mathbf{F}|^{1/4} = \mathbb{E}[t_{i^*}(\mathbf{X}, \mathbf{q})] = o(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$. It follows immediately from Markov's inequality and a union bound that there must exist a choice of j^* such that both $\delta_{j^*} \leq 4/18$ and $\mathbb{E}[t^{j^*}_{i^*}(\mathbf{X})] \leq 4\mathbb{E}[t_{i^*}(\mathbf{X}, \mathbf{q})] = o(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$. We let j^* equal an arbitrary such choice of index.

The last step is to show that the cell set $\mathbf{C}_{i^*}$ exists with probability at least $1/2$ over the choice of $\mathbf{X}$. For this, let $G_{j^*}(\mathbf{X})$ denote the subset of queries in $\mathbf{F}_{j^*}$ that succeed on input $\mathbf{X}$. Now using a union bound and Markov's inequality, we get that with probability at least $1/2$, there are both at most $|\mathbf{F}_{j^*}|17/18$ queries in $\mathbf{F}_{j^*}$ that err on input $\mathbf{X}$, i.e. $|G_{j^*}(\mathbf{X})| \geq 1/18|\mathbf{F}_{j^*}|$, and at the same time, we have $t^{j^*}_{i^*}(\mathbf{X}) \leq 100\mathbb{E}[t^{j^*}_{i^*}(\mathbf{X})] = o(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$ and finally all $\mathbf{x}_i$'s are unique (we have $|\mathbf{F}| = \Omega(n^2)$, thus all queries are unique with a very good constant probability when the constant in $\Omega(n^2)$ is large enough). When this happens, we show that the cell set $\mathbf{C}_{i^*}$ exists. First observe that since $|G_{j^*}(\mathbf{X})| = \Omega(|\mathbf{F}_{j^*}|)$ it follows that the average number of cells from $D_{i^*}(\mathbf{X})$ probed when answering a query in $G_{j^*}(\mathbf{X})$ is $O(t^{j^*}_{i^*}(\mathbf{X})) = o(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$. Hence there are $\Omega(|G_{j^*}(\mathbf{X})|) = \Omega(|\mathbf{F}_{j^*}|)$ queries in $G_{j^*}(\mathbf{X})$ that probe $o(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$ cells from $D_{i^*}(\mathbf{X})$. We thus prune $G_{j^*}(\mathbf{X})$ by deleting all queries that probe at least $1/100(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$ cells from $D_{i^*}(\mathbf{X})$ and we still have $|G_{j^*}(\mathbf{X})| = \Omega(|\mathbf{F}_{j^*}|) = \Omega(|\mathbf{F}|^{3/4})$.

Now consider all subsets of $\Delta = 1/25 \cdot \beta^{i^*} \lg |\mathbf{F}|/w$ cells in $D_{i^*}(\mathbf{X})$. Since any remaining query in $G_{j^*}(\mathbf{X})$ probes at most $\mu = 1/100(\lg |\mathbf{F}|/\lg(wt_u/\lg |\mathbf{F}|))$ cells from $D_{i^*}(\mathbf{X})$, we get that there must exist a subset $\mathbf{C}' \subseteq D_{i^*}(\mathbf{X})$ of Δ cells, for which at least $|G_{j^*}(\mathbf{X})|\binom{|D_{i^*}(\mathbf{X})|-\mu}{\Delta-\mu}/\binom{|D_{i^*}(\mathbf{X})|}{\Delta}$ queries in

$G_{j^*}(\mathbf{X})$ probes no cells in $D_{i^*}(\mathbf{X}) \setminus \mathbf{C}'$. Since

$$\begin{aligned}
|G_{j^*}(\mathbf{X})|\frac{\binom{|D_{i^*}(\mathbf{X})|-\mu}{\Delta-\mu}}{\binom{|D_{i^*}(\mathbf{X})|}{\Delta}} &= \Omega\left(|\mathbf{F}|^{3/4}\frac{(|D_{i^*}(\mathbf{X})|-\mu)!\Delta!}{|D_{i^*}(\mathbf{X})|!(\Delta-\mu)!}\right)\\
&= \Omega\left(|\mathbf{F}|^{3/4}\frac{(\Delta-\mu)^{\mu}}{|D_{i^*}(\mathbf{X})|^{\mu}}\right)\\
&= |\mathbf{F}|^{3/4}\left(\Omega\left(\frac{\lg|\mathbf{F}|}{wt_u}\right)\right)^{\mu}\\
&= |\mathbf{F}|^{74/100-o(1)},
\end{aligned}$$

we conclude that the claimed set $\mathbf{C}_{i^*}$ exists. We finally get

Theorem 2. *Any dynamic cell probe data structure for evaluating an n-degree polynomial over a finite field $\boldsymbol{F}$ must have query cost $\Omega(\lg|\boldsymbol{F}|\lg n/\lg(wt_u/\lg|\boldsymbol{F}|)\lg(wt_u))$ if $\boldsymbol{F}$ has size $\Omega(n^2)$. Here t_u is the worst case update time and w is the cell size. This lower bound holds for randomized data structures with any constant error probability $\delta < 1/2$.*

REFERENCES

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

[2] A. Andoni, P. Indyk, and M. Pătraşcu. On the optimality of the dimensionality reduction method. In *Proc. 47th IEEE Symposium on Foundations of Computer Science*, pages 449–458, 2006.

[3] O. Barkol and Y. Rabani. Tighter bounds for nearest neighbor search and related problems in the cell probe model. In *Proc. 32nd ACM Symposium on Theory of Computation*, pages 388–396, 2000.

[4] P. Beame and F. E. Fich. Optimal bounds for the predecessor problem and related problems. *Journal of Computer and System Sciences*, 65:38–72, August 2002.

[5] M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *Proc 21st ACM Symposium on Theory of Computation*, pages 345–354, 1989.

[6] A. Gál and P. B. Miltersen. The cell probe complexity of succinct data structures. *Theoretical Computer Science*, 379:405–417, July 2007.

[7] A. Golynski. Cell probe lower bounds for succinct data structures. In *Proc. 20th ACM/SIAM Symposium on Discrete Algorithms*, pages 625–634, 2009.

[8] M. Greve, A. G. Jørgensen, K. D. Larsen, and J. Truelsen. Cell probe lower bounds and approximations for range mode. In *Proc. 37th International Colloquium on Automata, Languages, and Programming*, pages 605–616, 2010.

[9] A. G. Jørgensen and K. G. Larsen. Range selection and median: Tight cell probe lower bounds and adaptive data structures. In *Proc. 22nd ACM/SIAM Symposium on Discrete Algorithms*, pages 805–813, 2011.

[10] K. S. Kedlaya and C. Umans. Fast modular composition in any characteristic. In *Proc. 49th IEEE Symposium on Foundations of Computer Science*, pages 146–155, 2008.

[11] K. G. Larsen. The cell probe complexity of dynamic range counting. In *Proc. 44th ACM Symposium on Theory of Computation*, pages 85–94, 2012.

[12] D. Liu. A strong lower bound for approximate nearest neighbor searching. *Information Processing Letters*, 92:23–29, October 2004.

[13] P. B. Miltersen. Lower bounds for union-split-find related problems on random access machines. In *Proc. 26th ACM Symposium on Theory of Computation*, pages 625–634, 1994.

[14] P. B. Miltersen. On the cell probe complexity of polynomial evaluation. *Theoretical Computer Science*, 143:167–174, May 1995.

[15] 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.

[16] R. Panigrahy, K. Talwar, and U. Wieder. Lower bounds on near neighbor search via metric expansion. In *Proc. 51st IEEE Symposium on Foundations of Computer Science*, pages 805–814, 2010.

[17] M. Pătraşcu. Lower bounds for 2-dimensional range counting. In *Proc. 39th ACM Symposium on Theory of Computation*, pages 40–46, 2007.

[18] M. Pătraşcu. Unifying the landscape of cell-probe lower bounds. In *Proc. 49th IEEE Symposium on Foundations of Computer Science*, pages 434–443, 2008.

[19] M. Pătraşcu and E. D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35:932–963, April 2006.

[20] M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *Proc. 38th ACM Symposium on Theory of Computation*, pages 232–240, 2006.

[21] M. Pătraşcu and M. Thorup. Higher lower bounds for near-neighbor and further rich problems. *SIAM Journal on Computing*, 39(2):730–741, 2010.

[22] P. Sen and S. Venkatesh. Lower bounds for predecessor searching in the cell probe model. *Journal of Computer and System Sciences*, 74:364–385, May 2008.

[23] C. Sommer, E. Verbin, and W. Yu. Distance oracles for sparse graphs. In *Proc. 50th IEEE Symposium on Foundations of Computer Science*, pages 703–712, 2009.

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

[25] Y. Yin. Cell-probe proofs. *ACM Transactions on Computation Theory*, 2:1:1–1:17, November 2010.