Crossing the Logarithmic Barrier for Dynamic Boolean Data Structure Lower Bounds*

Kasper Green Larsen
Aarhus University
Denmark
larsen@cs.au.dk

Omri Weinstein
Columbia University
USA
omri@cs.columbia.edu

Huacheng Yu
Harvard University
USA
yuhch123@gmail.com

ABSTRACT

This paper proves the first super-logarithmic lower bounds on the cell probe complexity of dynamic *boolean* (a.k.a. decision) data structure problems, a long-standing milestone in data structure lower bounds.

We introduce a new approach and use it to prove a $\tilde{\Omega}(\lg^{1.5} n)$ lower bound on the operational time of a wide range of boolean data structure problems, most notably, on the query time of dynamic range counting *over* $\mathbb{F}_2$ (Pătraşcu, 2007). Proving an $\omega(\lg n)$ lower bound for this problem was explicitly posed as one of five important open problems in the late Mihai Pătraşcu's obituary (Thorup, 2013). This result also implies the first $\omega(\lg n)$ lower bound for the classical 2D range counting problem, one of the most fundamental data structure problems in computational geometry and spatial databases. We derive similar lower bounds for boolean versions of dynamic *polynomial evaluation* and *2D rectangle stabbing*, and for the (non-boolean) problems of *range selection* and *range median*.

Our technical centerpiece is a new way of "weakly" simulating dynamic data structures using efficient *one-way* communication protocols with small advantage over random guessing. This simulation involves a surprising excursion to low-degree (Chebyshev) polynomials which may be of independent interest, and offers an entirely new algorithmic angle on the "cell sampling" method of Panigrahy et al. (2010).

CCS CONCEPTS

• **Theory of computation → Cell probe models and lower bounds**; *Computational complexity and cryptography*; *Communication complexity*;

KEYWORDS

Data Structures, Cell Probe Complexity, Lower Bounds, Range Searching, Dynamic Problems

ACM Reference Format:
Kasper Green Larsen, Omri Weinstein, and Huacheng Yu. 2018. Crossing the Logarithmic Barrier for Dynamic Boolean Data Structure Lower Bounds. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC'18).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3188745.3188790

*Kasper Green Larsen is supported by Villum Grant 13163 and an AUFF Starting Grant. Huacheng Yu is supported by NSF CCF-1212372.

STOC'18, June 25–29, 2018, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5559-9/18/06…$15.00
https://doi.org/10.1145/3188745.3188790

1 INTRODUCTION

Proving unconditional lower bounds on the operational time of data structures in the cell probe model [Yao81] is one of the holy grails of complexity theory, primarily because lower bounds in this model are oblivious to implementation considerations, hence they apply essentially to any imaginable data structure (and in particular, to the ubiquitous word-RAM model). Unfortunately, this abstraction makes it notoriously difficult to obtain data structure lower bounds, and progress over the past three decades has been very slow. In the dynamic cell probe model, where a data structure needs to maintain a database under an "online" sequence of n operations (updates and queries) by accessing as few memory cells as possible, a few lower bound techniques have been developed. In [FS89], Fredman and Saks proved $\Omega(\lg n/\lg\lg n)$ lower bounds for a list of dynamic problems. About 15 years later, Pătraşcu and Demaine [PD04, PD06] proved the first $\Omega(\lg n)$ lower bound ever shown for an explicit dynamic problem. The celebrated breakthrough work of Larsen [Lar12a] brought a near quadratic improvement on the lower bound frontier, where he showed an $\Omega((\lg n/\lg\lg n)^2)$ cell probe lower bound for the *2D range sum* problem (a.k.a. weighted orthogonal range counting in 2D). This is the highest cell probe lower bound known to date.

Larsen's result has one substantial caveat, namely, it inherently requires the queries to have large ($\Theta(\lg n)$-bit) output size. Therefore, when measured per output-bit of a query, the highest lower bound remains only $\Omega(\lg n)$ per bit (for dynamic connectivity due to Pătraşcu and Demaine [PD06]).

In light of this, a concrete milestone that was identified en route to proving $\omega(\lg^2 n)$ dynamic cell probe lower bounds, was to *prove an* $\omega(\lg n)$ *cell probe lower bound for* boolean (a.k.a. decision) *data structure problems* (the problem was explicitly posed in [Lar12a, Tho13, Lar13] and the caveat with previous techniques requiring large output has also been discussed in e.g. [Pat07, CGL15]). We stress that this challenge is *provably* a prerequisite for going beyond the $\omega(\lg^2 n)$ barrier for general ($\Theta(\lg n)$-bit output) problems: Indeed, consider a dynamic data structure problem $\mathcal{P}$ maintaining a database with updates $\mathcal{U}$ and queries $\mathcal{Q}$, where each query $q \in \mathcal{Q}$ outputs $\lg n$ bits. If one could prove an $\omega(\lg^2 n)$ lower bound for $\mathcal{P}$, this would directly translate into an $\omega(\lg n)$ lower bound for the following induced dynamic *boolean* problem $\mathcal{P}^{\text{bool}}$: $\mathcal{P}^{\text{bool}}$ has the same set of update operations $\mathcal{U}$, and has queries $\mathcal{Q}' := \mathcal{Q} \times [\lg n]$. Upon a query (q, i), the data structure should output the i-th bit $(\mathcal{P}(q, \mathcal{U}))_i$ of the answer to the original query q w.r.t the database

$\mathcal{U}$. An $\omega(\lg n)$ lower bound then follows, simply because each query of $\mathcal{P}$ can be simulated by $\lg n$ queries of $\mathcal{P}^{\text{bool}}$, and the update time is preserved. Thus, to break the $\lg^2 n$-barrier for cell probe lower bounds, one must first prove a *super-logarithmic* lower bound for some dynamic boolean problem. Of course, many classic data structure problems are naturally boolean (e.g., reachability, membership, etc.), hence studying decision data structure problems is interesting on its own.

Technically speaking, the common reason why all previous techniques hitherto (e.g., [Pat07, Lar12a, WY16]) fail to prove super-logarithmic lower bounds for dynamic problems with small output size, is that they all heavily rely on each query revealing a large amount of information about the database. In contrast, boolean queries can reveal at most one bit of information, and thus any such technique is doomed to fail. We elaborate on this excruciating obstacle and how we overcome it in the following subsection.

We develop a fundamentally new lower bound method and use it to prove the first super-logarithmic lower bounds for dynamic boolean data structure problems. Our results apply to natural boolean versions of several classic data structure problems. Most notably, we study a boolean variant of the dynamic *2D range counting* problem. In 2D range counting, n points are inserted one-by-one into an $[n] \times [n]$ integer grid, and given a query point $q = (x, y) \in [n] \times [n]$, the data structure must return the number of points p *dominated* by q (i.e., $p.x \leq x$ and $p.y \leq y$). This is one of the most fundamental data structure problems in computational geometry and spatial database theory (see e.g., [Aga04] and references therein). It is known that a variant of dynamic "range trees" solve this problem using $O((\lg n/\lg\lg n)^2)$ amortized update time and $O((\lg n/\lg\lg n)^2)$ worst case query time ([BGJS11]). We prove an $\tilde{\Omega}(\lg^{1.5} n)$ lower bound even for a boolean version, called *2D range parity*, where one needs only to return the parity of the number of points dominated by q. This is, in particular, the first $\omega(\lg n)$ lower bound for the (classical) 2D range counting problem. We are also pleased to report that this is the first progress made on the 5 important open problems posed in Mihai Pătraşcu's obituary [Tho13].

In addition to the new results for 2D range parity, we also prove the first $\omega(\lg n)$ lower bounds for the classic (non-boolean) problems of dynamic *range selection* and *range median*, as well as an $\omega(\lg n)$ lower bound for a boolean version of *polynomial evaluation*. We formally state these problems, our new lower bounds, and a discussion of previous state-of-the-art bounds in Section 3.

Organization. Due to space constraints, the following extended abstract contains only an exposition of our main results and technical contributions, while most proofs are deferred to the full version of this paper. We begin with a streamlined overview of our technical approach in light of previous data structure lower bounds (Section 2). Section 3 describes applications of our main result (Theorem 1) to higher dynamic lower bounds for 2D Range Counting and for the rest of the aforementioned data structure problems. In Section 5 we provide the necessary background, definitions and setup for our main result. Section 6 contains the proof of our main technical contribution (Theorem 1 below).

2 TECHNIQUES

2.1 Previous Techniques

To better understand the challenge involved in proving super-logarithmic lower bounds for boolean data structure problems, and how our approach departs from previous techniques that fail to overcome it, we first revisit Larsen's $\tilde{\Omega}(\lg^2 n)$ lower bound technique for problems with $\Theta(\lg n)$-bit output size, which is most relevant for our work. (We caution that a few variations [CGL15, WY16] of Larsen's [Lar12a] approach have been proposed, yet all of them crucially rely on large query output size). The following overview is presented in the context of the *2D range sum* problem for which Larsen originally proved his lower bound. 2D range sum is the variant of 2D range counting where each point is assigned a $\Theta(\lg n)$-bit integer weight, and the goal is to return the sum of weights assigned to points dominated by the query q. Clearly this is a harder problem than 2D range counting (which corresponds to all weights being 1) and 2D range parity (which again has all weights being 1, but now only 1 bit of the output must be returned).

Larsen's Lower Bound [Lar12a]. Larsen's result combines the seminal *chronogram method* of Fredman and Saks [FS89] together with the *cell sampling* technique introduced by Panigrahy et al. [PTW10]. The idea is to show that, after n random updates have been performed,[1] any data structure (with poly $\lg n$ update time) must probe many cells when prompted on a random range query. To this end, the n random updates are partitioned into $\ell := \Theta(\lg n/\lg\lg n)$ *epochs* $\mathbf{U}_\ell, \ldots, \mathbf{U}_i, \ldots, \mathbf{U}_1$, where the i-th epoch $\mathbf{U}_i$ consists of β^i updates for $\beta = \text{poly}\lg n$. The goal is to show that, for each epoch $i \in \{1, \ldots, \ell\}$, a random query must read in expectation $\Omega(\lg n/\lg\lg n)$ memory cells whose *last modification* occurred during the ith epoch $\mathbf{U}_i$. Summing over all epochs then yields a $\tilde{\Omega}(\lg^2 n)$ query lower bound.

To carry out this approach, one restricts the attention to epoch i, assuming all remaining updates in other epochs ($\mathbf{U}_{-i}$) are fixed (i.e., only $\mathbf{U}_i$ is random). For a data structure D, let A_i denote the set of memory cells *associated* with epoch i, i.e., the cells whose *last* update occurred in epoch i. Clearly, any cell that is written *before* epoch i cannot contain any information about $\mathbf{U}_i$, while the construction guarantees there are relatively few cells written *after* epoch i, due to the geometric decay in the lengths of epochs. Thus, "most" of the information D provides on $\mathbf{U}_i$ comes from cell probes to A_i (hence, intuitively, the chronogram method reduces a dynamic problem into $\approx \lg n$ nearly independent *static* problems).

The high-level idea is to now prove that, if a too-good-to-be-true data structure D exists, which probes $o(\lg n/\lg\lg n)$ cells associated with epoch i on an average query, then D can be used to devise a *compression* scheme (i.e., a "one-way" communication protocol) which allows a decoder to reconstruct the random update sequence $\mathbf{U}_i$ from an $o(H(\mathbf{U}_i))$-bit message, an information-theoretic contradiction.

Larsen's encoding scheme has the encoder (Alice) find a subset $C \subseteq A_i$ of a *fixed* size, such that *sufficiently many* range queries $q \in [n] \times [n]$ can be *resolved by* C, meaning that these queries can be answered without probing any cell in $A_i \setminus C$. Indeed, the assumption that the query algorithm of D probes only $o(\lg n/\lg\lg n)$ cells

[1] Each update inserts a random point and assigns it a random $\Theta(\lg n)$-bit weight.

from A_i, implies that a *random* subset of size $|C| = |A_i|/\text{poly}\lg n$ cells resolves at least a $(1/\text{poly}\lg n)^{o(\lg n/\lg\lg n)} = n^{-o(1)}$-fraction of the n^2 possible queries, an observation first made in [PTW10]. This observation in turn implies that by sending the contents and addresses of C, the decoder (Bob) can recover the answers to some *specific subset* $Q^* \subseteq [n] \times [n]$ of at least $n^{2-o(1)}$ queries (recall that updates $\mathbf{U}_{-i}$ are fixed, hence Bob can privately construct the contents of cells associated to epochs $j > i$ and the sets A_j for $j < i$ are so small that Alice can send those to Bob as well). Intuitively, if the queries of the problem are "sufficiently independent", e.g., the answers to all queries are n-wise independent over a random $\mathbf{U}_i$, then answering Q^* or even any subset of Q^* of size n would be sufficient to reconstruct the entire update sequence $\mathbf{U}_i$. Thus, by simulating the query algorithm $\forall q \in Q^*$ and using the set C to "fill in" his missing memory cells associated with $\mathbf{U}_i$, Bob could essentially recover $\mathbf{U}_i$. On the other hand, the update sequence itself contains at least $\Omega(|\mathbf{U}_i|) \gg |C| \cdot w$ bits of entropy, hence it cannot possibly be reconstructed from C, yielding an information-theoretic contradiction. Here, and throughout the paper, w denotes the number of bits in a memory cell. We make the standard assumption that $w = \Omega(\lg n)$, such that a cell has enough bits to store an index into the sequence of updates performed.

It is noteworthy that range queries do not directly possess such "n-wise independence" property per-se, but using (nontrivial) technical manipulations (a-la [Pat07, Lar12a, WY16]) this argument can be made to work, see the full version for details.

Alas, a subtle but crucial issue with the above scheme is that *Bob cannot identify the subset* Q^*, that is, when simulating the query algorithm of D on a given query, he can never know whether an *unsampled* ($\notin C$) cell requested by his simulation of the query algorithm in fact belongs to A_i or not. This issue is also faced by Pătraşcu's approach in [Pat07]. Larsen resolves this excruciating problem by having Alice further send Bob the indices of (a subset of) Q^* that already reveals enough information about $\mathbf{U}_i$ to get a contradiction. In order to achieve the anticipated contradiction, the problem must therefore guarantee that the answer to a query reveals more information than it takes to specify the query itself ($\Theta(\lg n)$ bits for 2D range sum). This is precisely the reason why Larsen's lower bound requires $\Omega(\lg n)$-bit weights assigned to each input point, whereas for the boolean 2D range parity problem, all bets are off.

2.2 Our Techniques

We develop a new lower bound technique which ultimately circumvents the aforementioned obstacle that stems from Bob's inability to identify the subset Q^*. Our high-level strategy is to argue that an efficient dynamic data structure for a boolean problem, induces an efficient one-way protocol from Alice (holding the entire update sequence $\mathcal{U} := \mathbf{U}_\ell, \ldots, \mathbf{U}_1$ as before) to Bob (who now receives a query $q \in \mathcal{Q}$ and $\mathcal{U} \setminus \{\mathbf{U}_i\}$), which enables Bob to answer his boolean query with some tiny yet nontrivial advantage over random guessing. For a dynamic boolean data structure problem $\mathcal{P}$, we denote this induced communication game (corresponding to the ith epoch) by $G^i_{\mathcal{P}}$. The following "weak simulation" theorem, which is the centerpiece of this paper, applies to *any* dynamic boolean data structure problem $\mathcal{P}$:

Theorem 1 (One-Way Weak Simulation Theorem, informal). *Let $\mathcal{P}$ be any dynamic boolean data structure problem, with n random updates grouped into epochs $\mathcal{U} = \{\mathbf{U}_i\}_{i=1}^{\ell}$ followed by a single query $q \in \mathcal{Q}$. If $\mathcal{P}$ admits a dynamic data structure D with word-size w, worst-case update time t_u and average (over $\mathcal{Q}$) expected query time t_q with respect to $\mathcal{U}$, satisfying $t_q, t_u, w \leq n^{0.1}$, then there exists some epoch $i \in [\ell]$ for which there is a* one-way *randomized communication protocol for $G^i_{\mathcal{P}}$ in which Alice sends Bob a message of only $|\mathbf{U}_i|/(wt_u)^{\Theta(1)}$ bits, and after which Bob successfully computes $\mathcal{P}(q, \mathcal{U})$ with probability at least $1/2 + \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right)$.*[2]

The formal statement and proof of the above theorem can be found in Section 6. Before we elaborate on the proof of Theorem 1, let us explain informally why such a seemingly modest guarantee suffices to prove super-logarithmic cell probe lower bounds on boolean problems with a certain "list-decoding" property. If we view query-answering as mapping an update sequence to an answer vector,[3] then answering a random query correctly with probability $1/2 + e^{-r(n)}$ would correspond to mapping an update sequence to an answer vector that is $(1/2 - e^{-r(n)})$-far from the true answer vector defined by the problem. Intuitively, if the correct mapping defined by the problem is *list-decodable* in the sense that in the $(1/2 - e^{-r(n)})$-ball centered at *any* answer vector, there are very few *codewords* (which are the correct answer vectors corresponding to some update sequences), then knowing any vector within distance $(1/2 - e^{-r(n)})$ from the correct answer vector would reveal a lot of information about the update sequence. Standard probabilistic arguments [Vad12] show that when the *code rate* is $n^{-\Theta(1)}$ (i.e., $|\mathcal{Q}| = n^{\Theta(1)}$ as for 2D range parity), a random code is "sufficiently" list-decodable with $r(n) = \Omega(\lg n)$, i.e., for most data structure problems, the protocol in the theorem would reveal too much information if Bob can predict the answer with probability, say $1/2 + e^{-0.01 \lg n}$. Therefore, Theorem 1 would imply that the query time must be at least $t_q = \Omega(\frac{\lg^{1.5} n}{\lg^2(w \cdot t_u)})$. Assuming the data structure has $t_u = \text{poly}\lg n$ worst-case update time and standard word-size $w = \Theta(\lg n)$, the above bound gives $t_q \geq \tilde{\Omega}(\lg^{1.5} n)$. Indeed all our concrete lower bounds are obtained by showing a similar list-decoding property with $r(n) = \Omega(\lg n)$, yielding a lower bound of $\tilde{\Omega}(\lg^{1.5} n)$. See Subsection 3 for more details.

Overview of Theorem 1 and the "Peak-to-Average" Lemma. We now present a streamlined overview of the technical approach and proof of our weak one-way simulation theorem, the main result of this paper. Let $\mathcal{P}$ be any boolean dynamic data structure problem and denote by $n_i := |\mathbf{U}_i| = \beta^i$ the size of each epoch of random updates (where $\beta := (t_u \cdot w)^{\Theta(1)}$ and $\sum_{i=1}^{\ell} n_i = n$). Recall that in $G^i_{\mathcal{P}}$, Alice receives the entire sequence of epochs $\mathcal{U}$, Bob receives $q \in_R \mathcal{Q}$ and $\mathcal{U} \setminus \{\mathbf{U}_i\}$, and our objective is to show that Alice can send Bob a relatively short message ($n_i/(t_u \cdot w)^{\Theta(1)}$ bits) which allows him to compute the answer to q w.r.t $\mathcal{U}$, denoted $\mathcal{P}(q, \mathcal{U}) \in \{0, 1\}$, with advantage $\delta := \exp(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n})$ over $1/2$.

Suppose $\mathcal{P}$ admits a dynamic data structure D with worst-case update time t_u and expected query time t_q with respect to $\mathcal{U}$ and

[2] Throughout the paper, we use $\exp(x)$ to denote $2^{\Theta(x)}$.

[3] An answer vector is a $|\mathcal{Q}|$-dimensional vector containing one coordinate per query, whose value is the answer to this query.

$q \in_R Q$. Following Larsen's cell sampling approach, a natural course of action for Alice is to generate the updated memory state M of D (w.r.t $\mathcal{U}$), and send Bob a relatively small random subset C_0 of the *the cells A_i associated with epoch i*, where each cell is sampled with probability $p = 1/(t_u \cdot w)^{\Theta(1)}$. Since the expected query time of D is t_q and there are $\ell = \Theta(\lg_\beta n)$ epochs, the average (over $i \in [\ell]$) number of cells in A_i probed by a query is t_q/ℓ, hence the probability that Alice's random set C_0 *resolves* Bob's random query $q \in_R Q$ is at least $\epsilon := p^{\Theta(t_q/\ell)}$. Let us henceforth denote this desirable event by $\mathcal{W}_q$. It is easy to see that, if Alice further sends Bob all cells that were written (associated) with *future* epochs $\mathbf{U}_{<i}$ (which can be done using less than $n_i/(w \cdot t_u)^{\Theta(1)}$ bits due to the geometric decay of epochs and the assumption that D probes at most t_u cells on each update operation), then *conditioned on $\mathcal{W}_q$*, Bob would have acquired all the necessary information to perfectly simulate the correct query-path of D on his query q.

Thus, if Bob could *detect* the event $\mathcal{W}_q$, the above argument would have already yielded an advantage of roughly $\Pr[\mathcal{W}_q] \geq \epsilon = p^{\Theta(t_q/\ell)} \geq \exp(-t_q \lg^2(w \cdot t_u)/\lg n) \gg \delta$ (as Bob could simply output a random coin-toss unless $\mathcal{W}_q$ occurs), and this would have finished the proof. Unfortunately, certifying the occurrence of $\mathcal{W}_q$ is prohibitively expensive, precisely for the same reason that identifying the subset Q^* is costly in Larsen's argument. Abandoning the hope for *certifying* the event $\mathcal{W}_q$ (while insisting on low communication) means that we must take a fundamentally different approach to argue that the noticeable occurrence of this event can somehow still be exploited implicitly so as to guarantee a nontrivial advantage. This is the heart of the paper, and the focal point of the rest of this exposition.

The most general strategy Bob has is to output his "maximum likelihood" estimate for the answer $\mathcal{P}(q, \mathcal{U})$ given the information he receives, i.e., the more likely posterior value of $(\mathcal{P}(q, \mathcal{U}) \,|\mathbf{U}_{-i}, C_0) \in \{0, 1\}$ (for simplicity of exposition, we henceforth ignore the conditioning on $\mathbf{U}_{-i}, C_0$ and on the set of updates D makes to future epochs $\mathbf{U}_{<i}$ which Alice sends as well). Assuming without loss of generality that the answer to the query is $\mathcal{P}(q, \mathcal{U}) = 1$, when $\mathcal{W}_q$ occurs, this strategy produces an advantage ("bias") of $1/2$ (since when $\mathcal{W}_q$ occurs, the answer $\mathcal{P}(q, \mathcal{U})$ is completely determined by $\mathbf{U}_{-i}, C_0$ and the updates to $\mathbf{U}_{<i}$), and when it does not occur, the strategy produces a bias of $\Pr[(\mathcal{P}(q, \mathcal{U}) = 1|\overline{\mathcal{W}_q})] - 1/2$. Thus, the overall bias is

$$\Pr[\mathcal{W}_q] \cdot (1/2) + \Pr[\overline{\mathcal{W}_q}] \cdot \left(\Pr[(\mathcal{P}(q, \mathcal{U}) = 1|\overline{\mathcal{W}_q})] - 1/2\right).$$

This quantity could be arbitrarily close to 0, since we have no control over the distribution of the answer conditioned on the complement event $\overline{\mathcal{W}_q}$, which might even cause perfect cancellation of the two terms.

Nevertheless, one could hope that such unfortunate cancellation of our advantage can be avoided if Alice reveals to Bob some little extra "relevant" information. To be more precise, let S_q be the set of memory addresses D would have probed when invoked on the query q *according to Bob's simulation*. That is, Bob simulates D until epoch i, updates the contents for all cells that appear in Alice's message, and simulates the query algorithm for q on this memory state. In particular, *if* the event $\mathcal{W}_q$ occurs, then S_q is the correct set of memory cells the data structure probes. Of course, the set S_q is extremely *unlikely* to be "correct" as $\Pr[\mathcal{W}_q]$ is tiny, so S_q should generally be viewed as an arbitrary subset of memory addresses. Now, the *true contents* of the cells S_q (w.r.t the true memory state M) induce some *posterior distribution* on the *correct* answer $\mathcal{P}(q, \mathcal{U})$ (in particular, when $\mathcal{W}_q$ occurs, the path is correct and its contents induce the true answer).

Imagine that Alice further reveals to Bob the true contents of some small subset $Y \subseteq S_q$, i.e., an assignment $x \in [2^w]^Y$. The posterior distribution of the answer $\mathcal{P}(q, \mathcal{U})$ conditioned on x is simply the convex combination of the posterior distributions conditioned on "$S_q = z$" for all z's that are consistent with x ($z|_Y = x$), weighted by the probability of z ($\Pr[S_q = z]$) up to some normalizer. The contribution of each term in this convex combination (i.e., of each posterior distribution induced by a partial assignment x) to the overall bias, is precisely the average, over all *full* assignments z to cells in S_q which are x-consistent, of the posterior bias induced by the event "$S_q = z$" (i.e., when the entire S_q is revealed). For each full assignment z, we denote its latter contribution by $f(z)$, hence the expected bias contributed by the event "$z|_Y = x$" is nothing but the sum of $f(z)$ over all z's satisfying $z|_Y = x$. Furthermore, we know that there is some assignment z^*, namely the contents of S_q when $\mathcal{W}_q$ occurs, such that $|f(z^*)|$ is "large" (recall the bias is $1/2$ in this event). Thus, the key question we pose and set out to answer, is whether it is possible to translate this ℓ_∞ "peak" of f into a comparable lower bound on the *"average"* bias $\sum_x \left|\sum_{z|_Y = x} f(z)\right|$, by conditioning on the assignments to a small subset of coordinates Y. Indeed, if such Y exists, Alice can sample independently another set of memory cells C_1 and send it to Bob. With probability $p^{|Y|}$, all contents of Y are revealed to Bob, and we will have the desired advantage. In essence, the above question is equivalent to the following information-theoretic problem:

> *Let Z be a k-variate random variable and B a uniform binary random variable in the same probability space, satisfying: (i) $\Pr[Z = z^*] \geq \epsilon$ for some z^*; (ii) $H(B \mid Z = z^*) = 0$. What is the smallest subset of coordinates $Y \subseteq [k]$ such that $H(B \mid (Z|_Y)) \leq 1 - \eta$?*

The crux of our proof is the following lemma, which asserts that conditioning on only $|Y| = O(\sqrt{k \lg(1/\epsilon)})$ coordinates suffices to achieve a non-negligible *average* advantage $\eta = \exp(-\sqrt{k \lg(1/\epsilon)})$.

Lemma 1 (Peak-to-Average Lemma). *Let $f : \Sigma^k \to \mathbb{R}$ be any real function satisfying: (i) $\sum_{z \in \Sigma^k} |f(z)| \leq 1$; and (ii) $\max_{z \in \Sigma^k} |f(z)| \geq \epsilon$. Then there exists a subset Y of indices, $|Y| \leq O\left(\sqrt{k \cdot \lg 1/\epsilon}\right)$, such that $\sum_{y \in \Sigma^Y} |\sum_{z|_Y = y} f(z)| \geq \exp(-\sqrt{k \cdot \lg 1/\epsilon})$.*

An indispensable ingredient of the proof is the usage of low-degree (multivariate) polynomials with "threshold"-like phenomena, commonly known as *(discrete) Chebyshev polynomials*.[4] The lemma can be viewed as an interesting and efficient way of "decomposing" a distribution into a small number of conditional distributions, "boosting" the effect of a single desirable event, hence the Peak-to-Average Lemma may be of independent interest (see the full version for a high-level overview and the formal proof). In

[4] These are real polynomials defined on the k-hypercube, of degree $O(\sqrt{k \lg(1/\gamma)})$ and whose value is uniformly bounded by γ everywhere *on the cube* except the all-0 point which attains the value 1.

the full version, we also show that the lemma is tight, in the sense that there are functions for which conditioning on $o\left(\sqrt{k \cdot \lg 1/\epsilon}\right)$ of their coordinates provides *no advantage at all.*

To complete the proof of the simulation theorem, we apply the Peak-to-Average Lemma with f, $k := t_q$ and $\epsilon := p^{\Theta(t_q/\ell)} = (1/wt_u)^{O(t_q/\ell)}$. The lemma guarantees that Bob can find a small (specific) set of coordinates $Y \subseteq S_q$, such that his maximum likelihood estimate conditioned on the true value y of the coordinates in Y must provide an advantage of at least

$$\exp(-\sqrt{k \cdot \lg 1/\epsilon}) = \exp\left(-t_q \lg(w \cdot t_u)/\sqrt{\lg n}\right).$$

Since $|Y|$ is small, the probability that Y is contained in Alice's second sample C_1 is $p^{|Y|} \geq \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right)$. Overall, Bob's maximum-likelihood strategy provides the desired advantage δ we sought, which completes the proof of Theorem 1.

3 APPLICATIONS: NEW LOWER BOUNDS

We apply our new technique to a number of classic data structure problems, resulting in a range of new lower bounds. This section describes the problems and the lower bounds we derive for them, in context of prior work. As a warm-up, we prove a lower bound for a somewhat artificial version of polynomial evaluation:

Polynomial Evaluation. Consider storing, updating and evaluating a polynomial P over the Galois field $GF(2^d)$. Here we assume that elements of $GF(2^d)$ are represented by bit strings in $\{0,1\}^d$, i.e. there is some bijection between $GF(2^d)$ and $\{0,1\}^d$. Elements are represented by the corresponding bit strings. Any bijection between elements and bit strings suffice for our lower bound to apply.

The *least-bit polynomial evaluation* data structure problem is defined as follows: A degree $n \leq 2^{d/4}$ polynomial $P(x) = \sum_{i=0}^{n} a_i x^i$ over $GF(2^d)$ is initialized with all $n+1$ coefficients a_i being 0. An update is specified by a tuple (i, b) where $i \in [n+1]$ is an index and b is an element in $GF(2^d)$. It changes the coefficient a_i such that $a_i \leftarrow a_i + b$ (where addition is over $GF(2^d)$). A query is specified by an element $y \in GF(2^d)$ and one must return the least significant bit of $P(y)$. Recall that we make no assumptions on the concrete representation of the elements in $GF(2^d)$, only that the elements are in a bijection with $\{0,1\}^d$ so that precisely half of all elements in $GF(2^d)$ have a 0 as the least significant bit.

Using our weak one-way simulation theorem, we prove the following lower bound in the full version of the paper:

Theorem 2. *Any cell probe data structure for* least-bit polynomial evaluation *over $GF(2^d)$, having cell size w, worst case update time t_u and expected average query time t_q must satisfy:*

$$t_q = \Omega\left(\min\left\{\frac{d\sqrt{\lg n}}{\lg^2(t_u w)}, \frac{\sqrt{n}}{(t_u w)^{O(1)}}\right\}\right).$$

Note that this lower bound is not restricted to have $d = O(\lg n)$ (corresponding to having polynomially many queries). It holds for arbitrarily large d and thus demonstrates that our lower bound actually grows as log of the number of queries, times a $\sqrt{\lg n}$. At least up to a certain (unavoidable) barrier (the $\sqrt{n}$ bound in the min is precisely when the query time is large enough that the data structure can read all cells associated to more than half of the epochs). We remark that the majority of previous lower bound techniques could also replace a $\lg n$ in the lower bounds by a d for problems with 2^d queries. Our introduction focuses on the most natural case of polynomially many queries ($d = \Theta(\lg n)$) for ease of exposition.

Polynomial evaluation has been studied quite intensively from a lower bound perspective, partly since it often allows for very clean proofs. The previous work on the problem considered the standard (non-boolean) version in which we are required to output the value $P(x)$, not just its least significant bit. Miltersen [Mil95] first considered the static version where the polynomial is given in advance and we disallow updates. He proved a lower bound of $t_q = \Omega(d/\lg S)$ where S is the space usage of the data structure in number of cells. This was improved by Larsen [Lar12b] to $t_q = \Omega(d/\lg(Sw/(nd)))$, which remains the highest static lower bound proved to date. Note that the lower bound peaks at $t_q = \Omega(d)$ for linear space $S = O(nd/w)$. Larsen [Lar12b] also extended his lower bound to the dynamic case (though for a slightly different type of updates), resulting in a lower bound of $t_q = \Omega(d \lg n/(\lg(wt_u) \cdot \lg(wt_u/d))$. Note that none of these lower bounds are greater than $t_q = \Omega(\lg n/\lg t_u)$ per output bit and in that sense they are much weaker than our new lower bound.

In [GM07], Gál and Miltersen considered *succinct* data structures for polynomial evaluation. Succinct data structures are data structures that use space close to the information theoretic minimum required for storing the input. In this setting, they showed that any data structure for polynomial evaluation must satisfy $t_q r = \Omega(n)$ when $2^d \geq (1+\epsilon)n$ for any constant $\epsilon > 0$. Here 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. Note that even for data structures using just a factor 2 more space than the minimum possible, the time lower bound reduces to the trivial $t_q = \Omega(1)$. For data structures with *non-determinism* (i.e., they can guess the right cells to probe), Yin [Yin10] proved a lower bound matching that of Miltersen.

On the upper bound side, Kedlaya and Umans [KU08] presented a word-RAM data structure for the static version of the problem, having space usage $n^{1+\epsilon} d^{1+o(1)}$ and worst case query time $\lg^{O(1)} n \cdot d^{1+o(1)}$, getting rather close to the lower bounds. While not discussed in their paper, a simple application of the *logarithmic method* makes their data structure dynamic with an amortized update time of $n^{\epsilon} d^{1+o(1)}$ and worst case query time $\lg^{O(1)} n \cdot d^{1+o(1)}$.

Parity Searching in Butterfly Graphs. In a seminal paper [Păt08], Pătraşcu presented an exciting connection between an entire class of data structure problems. Starting from a problem of *reachability oracles in the Butterfly graph*, he gave a series of reductions to classic data structure problems. His reductions resulted in $t_q = \Omega(\lg n/\lg(Sw/n))$ lower bounds for static data structures solving any of these problems.

We modify Pătraşcu's reachability problem such that we can use it in reductions to prove new *dynamic* lower bounds. In our version of the problem, which we term *parity searching in Butterfly graphs*, the data structure must maintain a set of directed acyclic graphs (Butterfly graphs of the same degree B, but different depths) under

updates which assign binary weights to edges, and support queries that ask to compute the parity of weights assigned to edges along a number of paths in these graphs. The formal definition of this version of the problem is given in the full version.

While this new problem might sound quite artificial and incompatible to work with, we show that parity searching in Butterfly graphs in fact reduces to many classic problems, hence proving lower bounds on this problem is the key to many of our results. Indeed, our starting point is the following lower bound:

Theorem 3. *Any dynamic data structure for* parity searching in Butterfly graphs *of degree* $B = (wt_u)^8$*, with a total of* n *edges, having cell size* w*, worst case update time* t_u *and expected average query time* t_q *must satisfy:*

$$t_q = \Omega\left(\frac{\lg^{3/2} n}{\lg^3(t_u w)}\right).$$

In the remainder of this section, we present new lower bounds which we derive via reductions from parity searching in Butterfly graphs . For context, our results are complemented with a discussion of previous work.

2D Range Counting. In 2D range counting, we are given n points P on a $[U] \times [U]$ integer grid, for some $U = n^{O(1)}$. We must preprocess the points such that given a query point $q = (x, y) \in [U] \times [U]$, we can return the number of points $p \in P$ that are *dominated* by q (i.e. $p.x \le q.x$ and $p.y \le q.y$). In the dynamic version of the problem, an update specifies a new point to insert. 2D range counting is a fundamental problem in both computational geometry and spatial databases and many variations of it have been studied over the past many decades.

Via a reduction from reachability oracles in the Butterfly graph, Pătraşcu [Păt08] proved a static lower bound of $t_q = \Omega(\lg n/\lg(Sw/n))$ for this problem, even in the case where one needs only to return the *parity* of the number of points dominated by q. Recall that this is the *2D range parity* problem.

It turns out that a fairly easy adaptation of Pătraşcu's reduction implies the following:

Theorem 4. *Any dynamic cell probe data structure for 2D range parity, having cell size* w*, worst case update time* t_u *and expected query time* t_q*, gives a dynamic cell probe data structure for parity searching in Butterfly graphs (for any degree* B*) with cell size* w*, worst case update time* $O(t_u)$ *and average expected query time* t_q*.*

Combining this with our lower bound for parity searching in Butterfly graphs (Theorem 3), we obtain:

Corollary 1. *Any cell probe data structure for 2D range parity, having cell size* w*, worst case update time* t_u *and expected query time* t_q *must satisfy:*

$$t_q = \Omega\left(\frac{\lg^{3/2} n}{\lg^3(t_u w)}\right).$$

In addition to Pătraşcu's static lower bound, Larsen [Lar12a] studied the aforementioned variant of the range counting problem, called *2D range sum*, in which points are assigned $\Theta(\lg n)$-bit integer weights and the goal is to compute the sum of weights assigned to points dominated by q. As previously discussed, Larsen's lower bound for dynamic 2D range sum was $t_q = \Omega((\lg n/\lg(t_u w))^2)$ and was the first lower bound to break the $\Omega(\lg n)$-barrier, though only for a problem with $\Theta(\lg n)$ bit output. Weinstein and Yu [WY16] later re-proved Larsen's lower bound, this time extending it to the setting of amortized update time and a very high probability of error. Note that these lower bounds remain below the logarithmic barrier when measured per output bit of a query. While 2D range counting (not the parity version) also has $\Theta(\lg n)$-bit outputs, it seems that the techniques of Larsen and Weinstein and Yu are incapable of proving an $\omega(\lg n)$ lower bound for it. Thus the strongest previous lower bound for the dynamic version of 2D range counting is just the static bound of $t_q = \Omega(\lg n/\lg(t_u w))$ (since one cannot build a data structure with space usage higher than $S = t_u n$ in n operations). As a rather technical explanation for why the previous techniques fail, it can be observed that they all argue that a collection of $m = n/\text{poly}(\lg n)$ queries have $\Omega(m \lg n)$ bits of entropy in their output. But for 2D range counting, having $n/\text{poly}(\lg n)$ queries means that on average, each query contains just $\text{poly}(\lg n)$ new points, reducing the total entropy to something closer to $O(m \lg\lg n)$. This turns out to be useless for the lower bound arguments. It is conceivable that a clever argument could show that the entropy remains $\Omega(m \lg n)$, but this has so forth resisted all attempts.

From the upper bound side, JáJá, Mortensen and Shi [JMS04] gave a static 2D range counting data structure using linear space and $O(\lg n/\lg\lg n)$ query time, which is optimal by Pătraşcu's lower bound. For the dynamic case, Brodal et al. [BGJS11] gave a data structure with $t_q = t_u = O((\lg n/\lg\lg n)^2)$. Our new lower bound shrinks the gap between the upper and lower bound on t_q to only a factor $\sqrt{\lg n}\lg\lg n$ for $t_u = \text{poly}(\lg n)$.

2D Rectangle Stabbing. In 2D rectangle stabbing, we must maintain a set of n 2D axis aligned rectangles with integer coordinates, i.e. rectangles are of the form $[x_1, x_2] \times [y_1, y_2]$. We assume coordinates are bounded by a polynomial in n. An update inserts a new rectangle. A query is specified by a point q, and one must return the number of rectangles containing q. This problem is known to be equivalent to 2D range counting via a folklore reduction. Thus all the bounds in the previous section, both upper and lower bounds, also apply to this problem. Furthermore, 2D range parity is also equivalent to 2D rectangle parity, i.e. returning just the parity of the number of rectangles stabbed.

Range Selection and Range Median. In range selection, we are to store an array $A = \{A[0], \ldots, A[n-1]\}$ where each entry stores an integer bounded by a polynomial in n. A query is specified by a triple (i, j, k). The goal is to return the index of the k'th smallest entry in the subarray $\{A[i], \ldots, A[j]\}$. In the dynamic version of the problem, entries are initialized to 0. Updates are specified by an index i and a value a and has the effect of changing the value stored in entry $A[i]$ to a. In case of multiple entries storing the same value, we allow returning an arbitrary index being tied for k'th smallest.

We also give a reduction from parity searching in Butterfly graphs. We remark that no reductions were known between the static versions of the problems and thus our new reductions also simplify previous static lower bounds.

Theorem 5. *Any dynamic cell probe data structure for range selection, having cell size* w*, worst case update time* t_u *and expected*

query time t_q, gives a dynamic cell probe data structure for parity searching in Butterfly graphs (for any degree B) having cell size w, worst case update time $O(t_u \lg^2 n)$ and expected average query time t_q. Furthermore, this holds even if we force $i = 0$ in queries (i, j, k) and require only that we return whether the k'th smallest element in $A[0], \ldots, A[j]$ is stored at an even or odd position.

Since we assume $w = \Omega(\lg n)$, combining this with Theorem 3 immediately proves the following:

Corollary 2. *Any cell probe data structure for range selection, having cell size w, worst case update time t_u and expected query time t_q must satisfy:*

$$t_q = \Omega\left(\frac{\lg^{3/2} n}{\lg^3(t_u w)}\right).$$

Furthermore, this holds even if we force $i = 0$ in queries (i, j, k) and require only that we return whether the k'th smallest element in $A[0], \ldots, A[j]$ is stored at an even or odd position.

While range selection is not a boolean data structure problem, it is still a fundamental problem and for the same reasons as mentioned under 2D range counting, the previous lower bound techniques seem incapable of proving $\omega(\lg n)$ lower bounds for the dynamic version. Thus we find our new lower bound very valuable despite the problem not beeing boolean . Also, we do in fact manage to prove the same lower bound for the boolean version where we need only determine whether the index of the k'th smallest element is even or odd.

For the static version of the problem, Jørgensen and Larsen [JL11] proved a lower bound of $t_q = \Omega(\lg n/\lg(Sw/n))$. Their proof was rather technical and a new contribution of our work is that their static lower bound now follows by reduction also from Pătraşcu's lower bound for reachability oracles in the Butterfly graph. For the dynamic version of the problem, no lower bound stronger than the $t_q = \Omega(\lg n/\lg(t_u w))$ bound following from the static bound was previously known.

On the upper bound side, Brodal et al. [BGJS11] gave a linear space static data structure with query time $t_q = O(\lg n/\lg\lg n)$. This matches the lower bound of Jørgensen and Larsen. They also gave a dynamic data structure with $t_q = t_u = O((\lg n/\lg\lg n)^2)$.

Since we prove our lower bound for the version of range selection where $i = 0$, also known as prefix selection, we can re-execute a reduction of Jørgensen and Larsen [JL11]. This means that we also get a lower bound for the fundamental range median problem. Range median is the natural special case of range selection where $k = \lceil (j - i + 1)/2 \rceil$.

Corollary 3. *Any cell probe data structure for range median, having cell size w, worst case update time t_u and expected query time t_q must satisfy:*

$$t_q = \Omega\left(\frac{\lg^{3/2} n}{\lg^3(t_u w)}\right).$$

Furthermore, this holds even if we are required only to return whether the median amongst $A[i], \ldots, A[j]$ is stored at an even or odd position.

We note that the upper bound of Brodal et al. for range selection is also the best known upper bound for range median.

4 ORGANIZATION OF SUBSEQUENT SECTIONS

In Section 5 we introduce both the dynamic cell probe model and the one-way communication model, which is the main proxy for our results. In Section 6 we state the formal version of Theorem 1 and give its proof. Due to space constraints, the proof of the *Peak-to-Average* lemma, as well as the proofs of all our lower bounds for concrete data structure problems, are deferred to the full version of this paper.

5 PRELIMINARIES

The dynamic cell probe model. A dynamic data structure in the cell probe model consists of an array of memory cells, each of which can store w bits. Each memory cell is identified by a w-bit address, so the set of possible addresses is $[2^w]$. It is natural to assume that each cell has enough space to address (index) all update operations performed on it, hence we assume that $w = \Omega(\lg n)$ when analyzing a sequence of n operations.

Upon an update operation, the data structure can perform read and write operations to its memory so as to reflect the update, by *probing* a subset of memory cells. This subset may be an arbitrary function of the update and the content of the memory cells previously probed during this process. The update time of a data structure, denoted by t_u, is the number of probes made when processing an update (this complexity measure can be measured in worst-case or in an amortized sense). Similarly, upon a query operation, the data structure performs a sequence of probes to read a subset of the memory cells in order to answer the query. Once again, this subset may by an arbitrary (adaptive) function of the query and previous cells probed during the processing of the query. The query time of a data structure, denoted by t_q, is the number of probes made when processing a query.

5.1 One-way Protocols and "Epoch" Communication Games

A useful way to abstract the information-theoretic bottleneck of dynamic data structures is *communication complexity*. Our main results (both upper and lower bounds) are cast in terms of the following two-party communication games, which are induced by dynamic data structure problems:

Definition 1 (Epoch Communication Games $G^i_{\mathcal{P}}$). *Let $\mathcal{P}$ be a dynamic data structure problem, consisting of a sequence of n update operations divided into* epochs $\mathcal{U} = (\mathrm{U}_\ell, \mathrm{U}_{l-1}, \ldots, \mathrm{U}_1)$, *where $|\mathrm{U}_i| = n_i$ (and $\sum_i n_i = n$), followed by a single query $q \in \mathcal{Q}$. For each epoch $i \in [\ell]$, the two-party communication game $G^i_{\mathcal{P}}$ induced by $\mathcal{P}$ is defined as follows:*

- *Alice receives all update operations $\mathcal{U} = (\mathrm{U}_\ell, \mathrm{U}_{l-1}, \ldots, \mathrm{U}_1)$.*
- *Bob receives $\mathrm{U}_{-i} := \mathcal{U} \setminus \{\mathrm{U}_i\}$ (i.e., all updates* except *those in epoch i) and a query $q \in \mathcal{Q}$ for $\mathcal{P}$.*
- *The goal of the players is to output the correct answer to q, that is, to output $\mathcal{P}(q, \mathcal{U})$.*

We shall consider the following restricted model of communication for solving such communication games.

Definition 2 (One-Way Randomized Communication Protocols). *Let $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0,1\}$ be a two-party boolean function. A* one-way *communication protocol π for $f(x,y)$ under input distribution μ proceeds as follows:*

- *Alice and Bob have shared access to a public random string R of their choice.*
- *Alice sends Bob a* single *message, $M_A(x,R)$, which is only a function of her input and the public random string.*
- *Based on Alice's message, Bob must output a value $v_\pi = v_\pi(y,R,M_A) \in \{0,1\}$.*

*We say that π ϵ-*solves *f under μ with cost m, if:*

- *For any input x, Alice never sends more than m bits to Bob, i.e., $|M_A(x,R)| \le m$, for all x, r.*
- $\Pr_{(x,y)\sim\mu,R}[v_\pi = f(x,y)] \ge 1/2 + \epsilon$.

Let us denote by

$$\overrightarrow{\text{adv}}(f,\mu,m) := \sup\{\epsilon \mid \exists \text{ one-way protocol } \pi \text{ that } \epsilon\text{-solves } f \text{ under } \mu \text{ with cost } m\}$$

the largest advantage ϵ achievable for predicting f under μ via an m-bit one-way communication protocol. For example, when applied to the boolean communication problem $G^i_{\mathcal{P}}$, we say that $G^i_{\mathcal{P}}$ has an m-bit one-way communication protocol with advantage ϵ, if $\overrightarrow{\text{adv}}(G^i_{\mathcal{P}},\mu,m) \ge \epsilon$. We remark that we sometimes use the notation $\|\pi\|$ to denote the message-length (i.e., number of bits m) of the communication protocol π.

6 ONE-WAY WEAK SIMULATION OF DYNAMIC DATA STRUCTURES

In this section we prove our main result, Theorem 1. For *any* dynamic decision problem $\mathcal{P}$, we show that if $\mathcal{P}$ admits an efficient data structure D with respect to a random sequence of n updates divided into $\ell := \lg_\beta n$ epochs $\mathcal{U} = (\mathrm{U}_\ell, \mathrm{U}_{\ell-1}, \ldots, \mathrm{U}_1)$, then we can use it to devise an efficient *one-way* communication protocol for the underlying two-party communication problem $G^i_{\mathcal{P}}$ of *some* (large enough) epoch i, with a nontrivial success (advantage over random guessing).

Throughout this section, let us denote the size of epoch i by $n_i := |\mathrm{U}_i| = \beta^i$, where we require $\beta = (w \cdot t_u)^{\Theta(1)}$, and $\sum_{i=1}^{\ell} n_i = n$. We prove the following theorem.

Theorem 1 (restated). *Let $\mathcal{P}$ be a dynamic boolean data structure problem, with n random updates grouped into epochs $\mathcal{U} = \{\mathrm{U}_i\}_{i=1}^{\ell}$, such that $|\mathrm{U}_i| = \beta^i$, followed by a single query $q \in \mathcal{Q}$. If $\mathcal{P}$ admits a dynamic data structure D with worst-case update time t_u and average (over $\mathcal{Q}$) expected query time t_q satisfying $t_q(w \cdot t_u)^{a+1} \le n^{1/2}$, then there exists some epoch $i \in [\ell/2, \ell]$ for which*

$$\overrightarrow{\text{adv}}\left(G^i_{\mathcal{P}},\ \mathcal{U},\ n_i/(w \cdot t_u)^{a-1}\right) \ \ge\ \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right)$$

as long as $\beta = (w \cdot t_u)^{\Theta(1)} \ge (w \cdot t_u)^a$ for a constant $a > 1$.

Proof. Consider the memory state $M = M(\mathcal{U})$ of D after the entire update sequence $\mathcal{U}$, and for each cell $c \in M$, define its *associated epoch* $E(c)$ to be the *last* epoch in $[\ell]$ during which c was probed (note that $E(c)$ is a random variable over the random update sequence $\mathcal{U}$). For each query $q \in \mathcal{Q}$, let T_q be the random variable denoting the number of probes made by D on query q (on the random update sequence). For each query q and epoch i, let T^i_q denote the number of probes on query q to cells *associated with epoch* i (i.e., cells c for which $E(c) = i$).

By definition, we have $\frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \mathbb{E}[T_q] = t_q$ and $T_q = \sum_{i=1}^{\ell} T^i_q$. Then by averaging, there exists one epoch $i \in [\ell/2, \ell]$ such that $\frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \mathbb{E}[T^i_q] \le 2t_q/\ell$. By Markov's inequality and a union bound, there exists a subset $\mathcal{Q}' \subseteq \mathcal{Q}$ of $|\mathcal{Q}|/2$ queries such that both

$$\mathbb{E}[T^i_q] \le 8t_q/\ell \quad \text{and} \quad \mathbb{E}[T_q] \le 8t_q\ , \tag{1}$$

for every query $q \in \mathcal{Q}'$. By Markov's inequality and union bound, for each $q \in \mathcal{Q}'$, we have

$$\Pr_{\mathcal{U}}[T^i_q \le 32t_q/\ell, T_q \le 32t_q] \ge 1/2. \tag{2}$$

Note that, while Bob cannot identify the event "$T^i_q \le 32t_q/\ell, T_q \le 32t_q$" (as it depends on Alice's input as well), he does know whether his query q is in $\mathcal{Q}'$ or not, which is enough to certify (2).

Now, suppose that Alice samples each cell *associated with epoch* i in M independently with probability p, where

$$p := \frac{1}{(w \cdot t_u)^a}$$

(note that, by definition of $G^i_{\mathcal{P}}$, Alice can indeed generate the memory state M and compute the associated epoch for each cell, as her input consists of the entire update sequence). Let C_0 be the resulting set of cells sampled by Alice. Alice sends Bob C_0 (both addresses and contents). For a query $q \in \mathcal{Q}'$, let $\mathcal{W}_q$ denote the event that the set of cells C_0 Bob receives, contains *all* T^i_q cells associated with epoch i and probed by the data structure. By Equation (2), we have that for every $q \in \mathcal{Q}'$

$$\Pr_{C_0,\mathcal{U}}[\mathcal{W}_q, T_q \le 32t_q] \ge p^{32t_q/\ell} \cdot \Pr_{\mathcal{U}}[T^i_q \le 32t_q/\ell, T_q \le 32t_q] \ge p^{32t_q/\ell}/2. \tag{3}$$

If Bob align *detect* the event $\mathcal{W}_q$, we would be done. Indeed, let C_2 denote the set of (addresses and contents of) cells associated with all future epochs $j < i$, i.e., all the cells probed by D succeeding epoch i. Due to the geometrically decreasing sizes of epochs, sending C_2 requires less than $n_i/(w \cdot t_u)^{a-1}$ bits of communication. Since Bob has all the updates preceding epoch i, he can simulate the data structure and generate the correct memory state of D right before epoch i. In particular, Bob knows for every cell, assuming it is not probed since epoch i (thus associated with some epoch $j > i$), what its content will be. Therefore, when he is further given the messages (c_0, c_2), Bob would be able to simulate the data structure perfectly on query q, assuming the event $\mathcal{W}_q$ occurs. If Bob could detect $\mathcal{W}_q$, he could simply output a random bit if it does not occur, and follow the data structure if it does. This strategy would have already produced an advantage of $p^{32t_q/\ell} \ge \exp(-t_q \lg^2(w \cdot t_u)/\lg n)$, which would have finished the proof. As explained in the introduction, Bob has no hope of certifying the occurrence of the event $\mathcal{W}_q$, hence we must take a fundamentally different approach for arguing that condition (3) can nevertheless be (implicitly) used to devise a

strategy for Bob with a nontrivial advantage. This is the heart of the proof.

To this end, note that, given a query $q \in Q'$, a received sample c_0 and all cells c_2 associated with some epoch $j < i$, Bob can simulate D on his partial update sequence (u_{-i}), filling in the memory updates according to c_0 and c_2, and *pretending* that all cells in the query-path of q which are associated with epoch i are actually sampled in c_0 (i.e., pretending that the event $\mathcal{W}_q$ occurs). See Step 5 of Figure 1 for the formal simulation argument. Let $M'(u_{-i}, c_0, c_2)$ denote the resulting memory state obtained by Bob's simulation in the figure, given u_{-i} and his received sets of cells c_0, c_2.

Now, let us consider the (deterministic) sequence of cells S_q that D would probe given query q in the above simulation with respect to Bob's memory state $M'(u_{-i}, c_0, c_2)$. Let us say that the triple (u_{-i}, c_0, c_2) is *good* for a query $q \in Q'$, if $\Pr_{\mathbf{U}_i}[\mathcal{W}_q | \mathbf{U}_{-i} = u_{-i}, C_0 = c_0, C_2 = c_2] \geq p^{32t_q/\ell}/4$ and $|S_q| \leq 32t_q$. That is, (u_{-i}, c_0, c_2) is good for q, if the posterior probability of $\mathcal{W}_q$ is (relatively) high and S_q is not too large. By Equation (3) and Markov's inequality, the probability that the triple (u_{-i}, c_0, c_2) satisfies $\Pr_{\mathcal{U}}[\mathcal{W}_q, T_q \leq 32t_q | u_{-i}, c_0, c_2] \geq p^{32t_q/\ell}/4$, is at least $p^{32t_q/\ell}/4$ (indeed, the expectation in (3) can be rewritten as

$$\mathop{\mathbb{E}}_{\mathbf{U}_{-i}, C_0, C_2} \Pr_{\mathbf{U}_i}[\mathcal{W}_q, T_q \leq 32t_q | \mathbf{U}_{-i}, C_0, C_2],$$

since C_2 is a deterministic function of $\mathcal{U}$). Note that when $\mathcal{W}_q$ occurs, the value of T_q is completely determined given u_{-i}, c_0 and c_2, in which case $|S_q| = T_q$, and thus the probability that (u_{-i}, c_0, c_2) is good is at least $p^{32t_q/\ell}/4$. From now on, let us focus only on the case that (u_{-i}, c_0, c_2) that Alice sends is good, since Bob can identify whether u_{-i}, c_0, c_2 is good based on q and Alice's message, and if it is not, he will output a random bit.

We caution that S_q is simply a set of memory addresses in M, not necessarily the correct one – in particular, while the *addresses* of the cells S_q are determined by the above simulation, the *contents* of these cells (in M) are not – they are a random variable of $\mathbf{U}_i$, as the sample c_0 is very unlikely to contain all the associated cells). For any assignment $z \in [2^w]^{S_q}$ to the *contents* of the cells in S_q, let us denote by

$$\mu_q(z) := \Pr_{\mathbf{U}_i}[S_q \leftarrow z | u_{-i}, c_0, c_2]$$

the probability that the memory content of the sequence of cells S_q is equal to z, conditioned on u_{-i}, c_0, c_2.

Every content assignment $Z = z$ to S_q, generates some posterior distribution on the *correct* query path (i.e., with respect to the *true* memory state M) and therefore on the output $\mathcal{P}(q, \mathcal{U})$ of the query q with respect to $\mathcal{U}$. Hence we may look at the joint probability distribution of the event "$\mathcal{P}(q, \mathcal{U}) = 1$" and the assignment Z which is

$$\eta_q(z) := \Pr_{\mathbf{U}_i}[\mathcal{P}(q, \mathcal{U}) = 1,\ S_q \leftarrow z \mid u_{-i}, c_0, c_2].$$

Now, consider the function

$$f(z) = f^q_{u_{-i}, c_0, c_2}(z) := \eta_q(z) - \frac{1}{2} \cdot \mu_q(z). \tag{4}$$

Equivalently, conditioned on u_{-i}, c_0 and c_2, $f(z)$ is the bias of the random varaible $\mathcal{P}(q, \mathcal{U})$ conditioned on $S_q \leftarrow z$, multiplied by the probability of $S_q \leftarrow z$.

Note that, since $\eta_q(z) \leq \mu_q(z)$ for every assignment z, we have $|f(z)| \leq \mu_q(z)/2$, and since $\mu_q(z)$ is a probability distribution, this fact implies that: (i) $\sum_z |f(z)| \leq \frac{1}{2}$. Furthermore, we shall argue that $\Pr[\mathcal{W}_q \mid u_{-i}, c_0, c_2] \geq p^{32t_q/\ell}/4$ (as we always condition on good u_{-i}, c_0, c_2), in which case the contents of S_q are completely determined by u_{-i}, c_0, c_2 (we postpone the formal argument to the Analysis section below). Denoting by z^* the content assignment to S_q induced by u_{-i}, c_0, c_2, we observe that conditioned on $\mathcal{W}_q$, S_q will be precisely the correct set of cells probed by D on q, in which case $\mathcal{P}(q, \mathcal{U})$ is determined by z^*, q, u_{-i}, c_0, c_2. Formally, this fact means that: (ii) $|f(z^*)| = \frac{1}{2} \cdot \Pr[S_q \leftarrow z^* \mid u_{-i}, c_0, c_2] \geq \Omega(p^{32t_q/\ell})$.

Conditions (i)+(ii) above imply that $f = f^q_{u_{-i}, c_0, c_2}$ satisfies the premise of the Peak-to-Average Lemma (Lemma 1) with $\Sigma := [2^w], k := O(t_q), \epsilon := \Omega(p^{32t_q/\ell}) = \exp(-t_q \lg^2(w \cdot t_u)/\lg n)$. Recall that the lemma guarantees there is a not-too-large subset $Y \subseteq S_q$ of coordinates (= addresses) of S_q, which Bob can *privately compute*,[5] such that if the values of the coordinates in Y are also revealed, then the *conditional* expectation of $f^q_{u_{-i}, c_0, c_2}$, namely

$$\mathop{\mathbb{E}}_{S_q|_Y} \left| \Pr_{\mathbf{U}_i} \left[\mathcal{P}(q, \mathcal{U}) = 1 \mid u_{-i}, c_0, c_2, S_q|_Y \right] - 1/2 \right|,$$

which is the average of Bob's "maximum-likelihood" estimate for $\mathcal{P}(q, \mathcal{U})$, is non-negligible (the formal details are postponed to the Analysis section below).

Given this insight, a natural strategy for the players is for Alice to further send Bob the contents of cells in the subset Y. While Alice does not know the subset Y,[6] she can use *public randomness* to sample yet another random set C_1 of cells from the *entire* memory M, where now *every* cell is sampled with equal probability p, and send the subset of C_1 that is associated with epoch i to Bob. (Note that it is important that this time the players use public randomness to subsample from the entire memory state M, since Alice does not know Y and yet Bob must be absolutely certain that all cells in Y were subsampled. Notwithstanding, to keep communication low, it is crucial that Alice sends Bob only the contents of cells associated with epoch i). Since $|Y|$ is guaranteed to be relatively small (of order $O(\sqrt{k \lg(1/\epsilon)})$), the probability $p^{|Y|}$ that all cells in Y get sampled will be sufficiently noticeable, in which case we shall argue that Bob's maximum-likelihood strategy will output the correct answer $\in \{0, 1\}$ with the desired nontrivial advantage. The formal one-way protocol π that the parties execute is described in Figure 1.

Analysis. We now turn to the formal analysis of the protocol π. We need to show

- (Communication cost) $\|\pi\| \leq O(n_i/(w \cdot t_u)^{a-1})$.
- (Correctness) $\Pr_{G^i_{\mathcal{P}} \sim \mathcal{U}, q \in_R Q} \left[\pi(G^i_{\mathcal{P}}) = \mathcal{P}(q, \mathcal{U}) \right] \geq 1/2 + \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right)$.

Communication. In both Step 2 and Step 3, Alice sends at most $2p|\mathbf{U}_i|t_u \cdot (2w) + 1$ bits. In Step 4, Alice sends at most $|\mathbf{U}_{<i}| \cdot t_u \cdot (2w) = \sum_{j<i} |\mathbf{U}_j| \cdot t_u \cdot (2w)$ bits. Since $|\mathbf{U}_j| = n_j = \beta^j$, the total communication cost is at most

$$O(p \cdot n_i \cdot t_u w) + O(\beta^{i-1} \cdot t_u w) \leq O(n_i/(w \cdot t_u)^{a-1}).$$

[5] Indeed, Y is only a function of q, $f^q_{u_{-i}, c_0, c_2}$, c_0,c_2 and the prior distribution on $\mathcal{U}$, and Bob possesses all this information.
[6] Indeed, Y is a function of q.

One-way protocol π for $G^i_{\mathcal{P}}$

Henceforth, by "sending a cell", we mean sending the address and (up to date) content of the cell in M.

Encoding.

(1) Alice generates the memory state M of D by simulating the data structure on $\mathcal{U}$, and computes the associated epoch for each cell.
(2) Alice samples each cell associated with epoch i independently with probability p. Let c_0 be the set of sampled cells. If $|c_0| > 2p|\mathbf{U}_i| \cdot t_u$, Alice sends a bit 0 and aborts. Otherwise, she sends a bit 1, followed by all cells in c_0.
(3) Alice uses *public randomness* to sample *every cell in M* independently with probability p. Let c_1 be the set of sampled cells. If there are more than $2p|\mathbf{U}_i| \cdot t_u$ cells in c_1 that are associated with epoch i, Alice sends a bit 0 and aborts. Otherwise, she sends a bit 1, followed by all cells in c_1 that are *associated with epoch i*.
(4) Alice sends Bob all cells associated with epoch j for all $j < i$, i.e., all the cells probed by D succeeding epoch i. Denote this set of (address and contents of) cells by c_2.

Decoding.

(5) Given his query $q \in \mathcal{Q}$, Bob simulates the data structure D on $u_{>i}$ and obtains a memory state M_0. He updates the contents of c_0 and c_2 in M_0, obtains a memory state $M' = M'(u_{-i}, c_0, c_2)$, and then simulates the query algorithm of D on query q and memory state M'. Let S_q be the set of (memory addresses of) cells probed by D in this simulation. If any of the following events occur, Bob outputs a random bit and aborts:
   (i) $q \notin \mathcal{Q}'$,
   (ii) Bob receives a bit 0 before c_0 or c_1,
   (iii) (u_{-i}, c_0, c_2) is not good for q.
(6) Let $Y \subset S_q$ be a subset of cells of size $\kappa := |Y| \le O\left(\sqrt{k \cdot \lg 1/\epsilon}\right)$ guaranteed by Lemma 1, when applied with $f := f^q_{u_{-i}, c_0, c_2}$, $\Sigma := [2^w]$, $k := |S_q| \le 32t_q, \epsilon := p^{32t_q/\ell}/4$.
(recall that Bob can privately compute the set Y).
(7) If $Y \not\subseteq c_1$ (i.e., if the sample c_1 sent by Alice does *not* contain all cells in Y), Bob outputs a random bit. Otherwise, let $y \in [2^w]^Y$ denote the content of the cells Y according to c_1. Let $S_q|_Y \leftarrow y$ denote the event that the memory content of Y is assigned the value y. Bob outputs 1 iff

$$\Pr_{\mathbf{U}_i}\left[\mathcal{P}(q, \mathcal{U}) = 1 \mid u_{-i}, c_0, c_2, S_q|_Y \leftarrow y\right] \quad > \quad 1/2.$$

Otherwise, Bob outputs 0.

Figure 1: The one-way weak simulation protocol of data structure D.

Correctness. Let π' be the variant of the protocol π in which, when executing Step 2 and Step 3, Alice ignores the condition of whether the samples C_0 or C_1 exceed the specified size limit, i.e., she always sends a bit 1 followed by all sampled cells. For simplicity of analysis, we will first show that π' has the claimed success probability, and then show that the impact of the above event (i.e., conditioning on C_0 and C_1 being within the size bound) is negligible, as it occurs with extremely high probability.

We first claim that the probability (over $\mathcal{U}$ and an average query $q \in_R Q$) that π' reaches Step 6 is not too small. By (1) and Markov's inequality, and by the discussion below (3), the probability that $q \in \mathcal{Q}'$ and (u_{-i}, c_0, c_2) is "good" for q is at least $\Omega(p^{32t_q/\ell}) \ge \exp(-t_q \lg^2(w \cdot t_u)/\lg n)$. This is precisely the probability that π' reaches Step 6.

We now calculate the success probability of π' conditioned on reaching Step 6. To this end, fix a set $Y \subseteq S_q$ of size κ. Then by Step 7, the success probability of π' conditioned on u_{-i}, c_0, c_2 and the event "$Y \subseteq C_1$" is

$$\frac{1}{2} + \mathop{\mathbb{E}}_{S_q|_Y}\left|\Pr_{\mathbf{U}_i}\left[\mathcal{P}(q, \mathcal{U}) = 1 \mid u_{-i}, c_0, c_2, S_q|_Y\right] - 1/2\right|$$

$$= \frac{1}{2} + \sum_{y \in [2^w]^Y} \Pr_{\mathbf{U}_i}[(S_q|_Y \leftarrow y) \mid u_{-i}, c_0, c_2]$$

$$\cdot \left|\Pr_{\mathbf{U}_i}\left[\mathcal{P}(q, \mathcal{U}) = 1 \mid (S_q|_Y \leftarrow y), u_{-i}, c_0, c_2\right] - 1/2\right|$$

$$= \frac{1}{2} + \sum_{y \in [2^w]^Y} \left|\Pr_{\mathbf{U}_i}\left[\mathcal{P}(q, \mathcal{U}) = 1, (S_q|_Y \leftarrow y) \mid u_{-i}, c_0, c_2\right] - \frac{1}{2} \cdot \Pr_{\mathbf{U}_i}[(S_q|_Y \leftarrow y) \mid u_{-i}, c_0, c_2]\right|$$

$$= \frac{1}{2} + \sum_{y \in [2^w]^Y} \left|\sum_{z \in [2^w]^{S_q} : z|_Y = y} \left(\Pr_{\mathbf{U}_i}\left[\mathcal{P}(q, \mathcal{U}) = 1, (S_q \leftarrow z) \mid u_{-i}, c_0, c_2\right] - \frac{1}{2} \cdot \Pr_{\mathbf{U}_i}[S_q \leftarrow z \mid u_{-i}, c_0, c_2]\right)\right|$$

$$= \frac{1}{2} + \sum_{y \in [2^w]^Y} \left|\sum_{z \in [2^w]^{S_q} \,:\, z|_Y = y} f^q_{u_{-i}, c_0, c_2}(z)\right|, \qquad (5)$$

where the last transition is by the definition of $f^q_{u_{-i}, c_0, c_2}$ in (4). Note that for any z, it holds that $|f(z)| \le \frac{1}{2} \cdot \Pr[S_q \leftarrow z \mid u_{-i}, c_0, c_2]$. Thus, $\sum_{z \in [2^w]^{S_q}} |f(z)| \le \frac{1}{2}$. On the other hand, since we always

condition on good (u_{-i}, c_0, c_2), we have $\Pr[\mathcal{W}_q \mid u_{-i}, c_0, c_2] \geq p^{32t_q/\ell}/4$. That is, with probability at least $p^{32t_q/\ell}/4$ all cells in S_q associated with epoch i are contained in c_0. In this case, the contents of S_q are completely determined by u_{-i}, c_0, c_2. Indeed, the contents of the cells associated with epoch $< i$ are determined by c_2; the cells associated with epoch i are determined by c_0; the remaining cells are determined by $u_{>i}$. Let z^* denote the assignment to S_q, induced by u_{-i} and the contents of c_0, c_2 conditioned on the occurrence of $\mathcal{W}_q$. By the definition of S_q, when $\mathcal{W}_q$ happens, S_q will be exactly the set of cells the data structure probes. Thus, the output of q is also determined. We therefore have $|f(z^*)| = \frac{1}{2} \cdot \Pr[S_q \leftarrow z^* \mid u_{-i}, c_0, c_2] \geq \Omega(p^{32t_q/\ell})$. We conclude that the function $f = f^q_{u_{-i}, c_0, c_2}$ satisfies the premise of the Peak-to-Average lemma (Lemma 1) with

- $\Sigma = [2^w]$;
- $k = |S_q| \leq O(t_q)$;
- $\epsilon = p^{32t_q/\ell}/4 \geq \exp(-t_q \lg^2(w \cdot t_u)/\lg n)$.[7]

Without loss of generality, we may assume $\lg(w \cdot t_u) \ll \sqrt{\lg n}$, and thus $\epsilon \in [2^{-O(k)}, 1]$.[8] Therefore, the lemma guarantees there is a set $Y \subset S_q$ of cells that has size at most

$$|Y| = \kappa \leq O\left(\sqrt{k \lg 1/\epsilon}\right) \leq O\left(t_q \lg(w \cdot t_u)/\sqrt{\lg n}\right),$$

for which

$$\sum_{y \in [2^w]^Y} \left| \sum_{z \in [2^w]^{S_q} \,:\, z|_Y = y} f^q_{u_{-i}, c_0, c_2}(z) \right| \geq \exp\left(-t_q \lg(w \cdot t_u)/\sqrt{\lg n}\right).$$

This justifies Step 6 of the protocol. It follows that, for any $q \in Q'$, the probability that the sample C_1 of cells contains the set Y is at least

$$\Pr_{C_1}[Y \subseteq C_1] = p^{|Y|} = p^{O\left(t_q \lg(w \cdot t_u)/\sqrt{\lg n}\right)} = \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right). \quad (6)$$

Equation (5) therefore implies that, conditioned on the event that $|Y| \subseteq C_1$, the probability that π' outputs a correct answer is

$$1/2 + \exp\left(-t_q \lg(w \cdot t_u)/\sqrt{\lg n}\right),$$

and combining this with (6) and the probability that π' reaches Step 6, we conclude that the overall success probability of π, conditioned on the protocol not aborting when c_0 or c_1 is too large, is

$$1/2 + \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right). \quad (7)$$

To finish the proof, it therefore suffices to argue that the probability that π aborts due to this event is tiny. To this end, let A_i denote the random variable representing the number of associated cells with epoch i. We know that $A_i \leq |\mathrm{U}_i| \cdot t_u = n_i \cdot t_u$ (since the worst-case update time of D is t_u by assumption). Now, let $\mathcal{E}_0$ denote the event that Alice's sample in Step 2 of the protocol is too large, i.e., that "$|C_0| > 2p|\mathrm{U}_i| \cdot t_u$". Similarly, let $\mathcal{E}_1$ denote the event that in Step 3 of the protocol, "$|C_1| > 2p|\mathrm{U}_i| \cdot t_u$". Denote $\mathcal{E} := \mathcal{E}_0 \vee \mathcal{E}_1$ (note that this is the event (ii) in Step 5 of π). Since both sets C_0 and C_1 are *i.i.d* samples where each cell is sampled independently with probability p, a standard Chernoff bound implies that

$$\Pr[\mathcal{E}] \leq 2\Pr[|C_0| \geq 2\mathbb{E}[|C_0|]] \leq \exp(-p(n_i \cdot t_u)) \leq \exp(-n_i/(w \cdot t_u)^a). \quad (8)$$

Finally, since $i \geq \ell/2$ and thus $n_i \geq n^{1/2} \geq t_q(w \cdot t_u)^{a+1}$, by (7), (8) and a union bound, we conclude that

$$\begin{aligned} &\Pr_{\mathcal{U}, q}[\pi(q) \neq \mathcal{P}(q, \mathcal{U})] \\ &\leq 1/2 - \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right) + \Pr[\mathcal{E}] \\ &\leq 1/2 - \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right) + \exp(-t_q \cdot (w \cdot t_u)) \\ &\leq 1/2 - \exp\left(-t_q \lg^2(w \cdot t_u)/\sqrt{\lg n}\right), \end{aligned}$$

which completes the proof of the entire theorem. □

ACKNOWLEDGEMENT

We are grateful to Rocco Servedio and Oded Regev for insightful discussions on the Peak-to-Average Lemma, and in particular, to Alexander Sherstov for observing and sharing with us the proof that our peak-to-average lemma is tight.

REFERENCES

[Aga04] Pankaj K. Agarwal. Range searching. In *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 809–837. 2004.

[BGJS11] Gerth Stølting Brodal, Beat Gfeller, Allan Grønlund Jørgensen, and Peter Sanders. Towards optimal range medians. *Theoretical Computer Science*, 412(24):2588–2601, May 2011.

[CGL15] Raphaël Clifford, Allan Grønlund, and Kasper Green Larsen. New unconditional hardness results for dynamic and online problems. In *Proc. 56th IEEE Symposium on Foundations of Computer Science*, 2015.

[FS89] Michael L. Fredman and Michael E. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 345–354, 1989.

[GM07] Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. *Theoretical Computer Science*, 379:405–417, July 2007.

[JL11] Allan Grønlund Jørgensen and Kasper Green 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.

[JMS04] Joseph JáJá, Christian Worm Mortensen, and Qingmin Shi. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *Proc. 15th International Symposium on Algorithms and Computation*, pages 558–568, 2004.

[KU08] Kiran S. Kedlaya and Christopher Umans. Fast modular composition in any characteristic. In *Proc. 49th IEEE Symposium on Foundations of Computer Science*, pages 146–155, 2008.

[Lar12a] Kasper Green Larsen. The cell probe complexity of dynamic range counting. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 85–94, 2012.

[Lar12b] Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *Proc. 53rd IEEE Symposium on Foundations of Computer Science*, pages 293–301, 2012.

[Lar13] Kasper Green Larsen. *Models and Techniques for Proving Data Structure Lower Bounds.* PhD thesis, Aarhus University, 2013.

[Mil95] Peter Bro Miltersen. On the cell probe complexity of polynomial evaluation. *Theoretical Computer Science*, 143:167–174, May 1995.

[Pat07] Mihai Patrascu. Lower bounds for 2-dimensional range counting. In *Proc. 39th ACM Symposium on Theory of Computation*, pages 40–46, 2007.

[7] We used the fact that $\ell = \Theta(\lg_\beta n)$ and $\beta = (w \cdot t_u)^{\Theta(1)}$.

[8] In fact, if $\lg(w \cdot t_u) \geq \Omega(\sqrt{\lg n})$, the right-hand side of the inequality in the theorem statement is less than p^{t_q}, hence the statement becomes trivial. Indeed, with probability p^{t_q}, Alice samples *all* cells probed by the data structure on query q.

[Păt08] Mihai Pătraşcu. Unifying the landscape of cell-probe lower bounds. In *Proc. 49th IEEE Symposium on Foundations of Computer Science*, pages 434–443, 2008.

[PD04] Mihai Pătraşcu and Erik D. Demaine. Tight bounds for the partial-sums problem. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 20–29, 2004.

[PD06] Mihai Pătraşcu and Erik D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput.*, 35(4):932–963, 2006.

[PTW10] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, pages 805–814, 2010.

[Tho13] Mikkel Thorup. Mihai Pătraşcu: Obituary and open problems. *Bulletin of the EATCS*, 109:7–13, 2013.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[WY16] Omri Weinstein and Huacheng Yu. Amortized dynamic cell-probe lower bounds from four-party communication. In *Proc. 57th IEEE Symposium on Foundations of Computer Science*, pages 305–314, 2016.

[Yao81] Andrew Chi-Chih Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, 1981.

[Yin10] Yitong Yin. Cell-probe proofs. *ACM Transactions on Computation Theory*, 2:1:1–1:17, November 2010.