Cell-Probe Lower Bounds from Online Communication Complexity

Josh Alman*
MIT CSAIL and EECS
Cambridge, Massachusetts, USA
jalman@mit.edu

Joshua R. Wang†
Stanford University
Stanford, California, USA
joshua.wang@cs.stanford.edu

Huacheng Yu‡
Harvard University
Cambridge, Massachusetts, USA
yuhch123@gmail.com

ABSTRACT

In this work, we introduce an online model for communication complexity. Analogous to how online algorithms receive their input piece-by-piece, our model presents one of the players, Bob, his input piece-by-piece, and has the players Alice and Bob cooperate to compute a result each time before the next piece is revealed to Bob. This model has a closer and more natural correspondence to dynamic data structures than classic communication models do, and hence presents a new perspective on data structures.

We first present a tight lower bound for the *online set intersection* problem in the online communication model, demonstrating a general approach for proving online communication lower bounds. The online communication model prevents a batching trick that classic communication complexity allows, and yields a stronger lower bound. We then apply the online communication model to prove data structure lower bounds for two dynamic data structure problems: the Group Range problem and the Dynamic Connectivity problem for forests. Both of the problems admit a worst case $O(\log n)$-time data structure. Using online communication complexity, we prove a tight cell-probe lower bound for each: spending $o(\log n)$ (even amortized) time per operation results in at best an $\exp(-\delta^2 n)$ probability of correctly answering a $(1/2+\delta)$-fraction of the n queries.

CCS CONCEPTS

• **Theory of computation → Communication complexity**; **Cell probe models and lower bounds**; *Random order and robust communication complexity*;

KEYWORDS

Online communication complexity, Data structure lower bounds, Cell-probe model

*Supported by an NSF Graduate Research Fellowship, and by NSF CAREER awards 1651838 and 1552651. Work initiated while at Stanford University.
†Supported by NSF CCF-1524062 and a Stanford Graduate Fellowship.
‡Supported in part 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/10.1145/3188745.3188862

ACM Reference Format:
Josh Alman, Joshua R. Wang, and Huacheng Yu. 2018. Cell-Probe Lower Bounds from Online Communication Complexity. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC'18).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3188745.3188862

1 INTRODUCTION

One major hallmark of complexity theory is Yao's cell-probe model [24], a powerful model of computation that manages to capture the inherent complexity found in a variety of data structure problems. The titular feature of this model is that the data structure is only charged for *the number of memory cells that it accesses (or probes)*, and not for any computation it needs to perform on the contents of those cells. Since this model is so strong – data structures are given the power of 'free computation' – proving lower bounds here yields lower bounds for most other models of data structure computation. Many lower bounds in the cell-probe model are derived via connections to communication complexity, wherein two players try to jointly compute a function but are only charged for the bits that they communicate to each other and again, not for any computation [1, 2, 11, 12, 16, 22, 25].

Unfortunately, the sheer power granted to the data structure by the cell-probe model can often make it difficult to prove strong lower bounds. In fact, in many cases a matching lower bound in the cell-probe model may be impossible; counting just cell probes in lieu of actual computation time might indeed make several problems easier [10]. Partly as a result of this difficulty, only a few techniques are known for proving cell-probe lower bounds. In this paper, we propose a new technique to add to our growing toolbox. We propose a new model of communication complexity which we call *online communication*. We give tools for proving lower bounds in this new model, and then use these tools to show how the model results in new robust lower bounds for two fundamental data structure problems.

1.1 Online Communication Model

Inspired by the fact that a data structure must answer one query before it sees the next, we propose a novel model of communication: the online communication model. The salient feature of our model is that one of the players, Bob, does not receive his entire input at once. Whereas Alice receives her entire input X, Bob receives a small piece Y_1 and the two must jointly compute a function $f_1(X, Y_1)$ before Bob receives the next piece Y_2, and so on. As usual, we care about the total amount of communication that Alice and Bob use. Intuitively, this model is designed to rule out batching techniques; in usual offline models of communication, it may be cheaper for

Bob to discuss all the pieces of his input together, but in the online model, this is impossible since he only receives one piece at a time.

It stands to reason that we should be able to prove better lower bounds now that communication protocols have one fewer trick to work with. In this paper, we develop techniques that relate this model to more familiar entire-input-at-once models. In order to demonstrate how things are different in the online model, we turn to perhaps one of the most important communication problems: set disjointness.

1.2 Online Set Disjointness

The testbed for our new model is the quintessential problem, set disjointness. In the basic version of this problem, Alice and Bob are each given subsets over $[n]$ and want to compute whether their subsets are disjoint. This problem has long been a favored source of hardness, and along with its many variations, it has been thoroughly studied by theorists; see e.g. the surveys [4, 18].

In the context of our model, this problem manifests as the *online set intersection problem*. Alice is given an entire subset $X \subseteq [n]$ of size k while Bob is only given *single* elements y_i of another subset $Y \subseteq [n]$ of size k one at a time. The players need to decide whether $y_i \notin X$ before Bob receives the next element. We show that:

Theorem 1.1 (informal). *When $n \geq k^2$, the online set intersection problem requires $\Omega(k \log\log k)$ bits of total communication.*

In fact, our proof shows that deciding whether X and Y are disjoint requires $\Omega(k \log\log k)$ bits of communication in the online model; it cannot be done more effiicently even if Alice and Bob may stop after finding an intersecting element. We also give a fairly straightforward protocol which solves the problem in $O(k \log\log k)$ bits of communication, showing that this bound is tight. This stands in contrast to known bounds for the classical communication model, in which the set disjointness problem can be solved with just $O(k)$ bits by using a batching trick to test all elements of Y simultaneously [8].

1.3 Group Range Problem

The first data structure problem we consider is a generalization of the Partial Sums problem (from e.g. [14]), and which has connections to a family of problems studied by Frandsen, Miltersen, and Skyum [6]. In the *Group Range Problem*, we have a group G along with a *binary encoding* of the group elements (any injective function) $e : G \to \{0,1\}^s$. We would like a data structure which stores a sequence of n group elements $a_0, \ldots, a_{n-1}$ while supporting the following operations:

- Update(i, a) sets entry $a_i \leftarrow a$.
- Query(ℓ, r, i) returns the i^{th} bit of the binary encoding of the group product $a_\ell a_{\ell+1} \cdots a_{r-1} a_r$.

We focus on the case where the cell-size is $w = \Theta(\log n)$ and the group is polynomially-sized: $\log |G| = O(w)$.

Regarding upper-bounds, there is a folklore data structure which solves the problem with $O(\log n)$ time per operation. This is a worst-case (not just amortized) guarantee, and the data structure is deterministic. There is a matching $\Omega(\log n)$ cell-probe lower-bound by Pătraşcu and Demaine for the Partial Sums problem, wherein queries need to return the entire product rather than a single bit [14]. This lower bound holds for Las Vegas randomized data structures (the number of cell probes is considered in expectation) and amortized operation cost.

However, this lower bound leaves open several plausible ways to improve the running time. What if we really only care about a single bit of each query? What if we are willing to tolerate errors? Our main result shows that even if we permit these concessions, the $\Omega(\log n)$ barrier still stands:

Theorem 1.2. *There exists a distribution over n updates and queries for the Group Range Problem with binary encoding of the group elements $e : G \to \{0,1\}^s$, such that for any randomized cell-probe data structure D with word size $w = \Theta(\log n)$, which with probability p answers at least a $(\frac{1}{2} + \delta)$ fraction of queries correctly and spends $\epsilon n \log n$ total running time, we must have $p \leq \exp(-\delta^2 n)$, as long as $s \leq (1+\epsilon)\log|G|$, $\delta^2 \gg \epsilon \geq \Omega(1/\log n)$, and n is sufficiently large.*[1]

Put another way, Theorem 1.2 settles the trade-off between running time and accuracy of the output for the Group Range Problem. There are two possible regimes. If we are willing to pay $\Theta(\log n)$ time per operation, then there exists a deterministic worst-case data structure. Otherwise, if we require the data structure to spend $o(\log n)$ time per operation, then Theorem 1.2 shows that we cannot hope to do much better than outputting a random bit for each query, up to a constant factor improvement in δ. To the best of our knowledge, this bound and our other lower bound we describe shortly are the first tight data structure lower bounds in such a high error regime, where a data structure may answer barely more than half of the queries correctly, and do so even with a small success probability.

1.4 Dynamic Connectivity

Next, we consider a fundamental problem in graph data structures: Dynamic Connectivity. In this problem, we would like a data structure which stores an undirected graph $G = (V, E)$ on n vertices, while supporting the following operations:

- `insert`(u, v) adds edge (u, v) to E.
- `delete`(u, v) removes edge (u, v) from E.
- `query`(u, v) returns whether or not there currently exists a path between nodes u and v.

Like before, we look at this problem in the cell-probe model with cell size $w = O(\log n)$. The link/cut tree data structure [19] and Euler tour tree data structure [9] for the problem take $O(\log n)$ time per update or query. A matching $\Omega(\log n)$ lower bound was given by Pătraşcu and Demaine [15]. However, their lower bound holds for Las Vegas or Monte Carlo data structures with amortization, where they assume that the error probability for each query is n^{-c} for some large constant c.

Pătraşcu and Demaine still leave open the question of what can be done if we insist on $o(\log n)$ time per operation. Their lower bound asserts that we cannot answer each query correctly with better than $1 - n^{-c}$ probability. However, for one example, it could still be possible to design a data structure which answers *all* queries correctly simultaneously with probability, say $1 - 1/\log n$, and such that each individual query is correct with probability lower than $1 - n^{-c}$. Such a data structure would not violate the existing lower

[1] In this paper, we use $\exp(f(n))$ to mean $2^{\Theta(f(n))}$.

bounds, and its success probability would be good enough in many applications, as it only incurs an additive $1/\log n$ overall error probability.

Again, our new technique yields a robust lower bound, ruling out such data structures:

Theorem 1.3. *There exists a distribution over $O(n)$ updates and queries for the dynamic connectivity problem, such that for any randomized cell-probe data structure D with word-size $w = \Theta(\log n)$, which with probability p answers at least a $(\frac{1}{2} + \delta)$-fraction of the queries correctly and spends $\epsilon n \log n$ total running time, we must have $p \leq \exp(-\delta^2 n)$ as long as $\delta^2 \gg 1/\log(1/\epsilon)$ and $\epsilon \geq \Omega(1/\log n)$ and n is sufficiently large. Moreover, the graph is always a forest throughout the sequence of updates.*

Similar to before, this essentially settles the complexity of the Dynamic Connectivity problem in forests (where the graph is always a forest throughout the sequence of updates). If one wants $o(\log n)$ per update and query, then one cannot do better than outputting the flip of a random coin to answer each query, again up to a constant factor in δ.

Our lower bound almost matches the best known upper bound for Dynamic Connectivity in general graphs of $O(\log n \log^3 \log n)$ by Thorup [21]. Dynamic Connectivity with higher error than Pătraşcu and Demaine allowed for, although still lower error than we consider, was studied by Fredman and Saks [7], but for worst-case update time instead of amortized, and for the problem where edge deletions are not allowed; the only updates allowed are edge insertions. They showed that any 1/32-error data structure for Dynamic Connectivity without deletions with expected query time t_q and *worst-case* update time t_u must have $t_u \geq \Omega(\log(n)/\log(t_u \log(n)))$. Ramamoorthy and Rao [17] recently gave a simplified proof of Fredman and Saks' result as well.

1.5 Further Results

We also prove some complementary results to our two data structure lower bounds.

1.5.1 Group Range Problem. The Group Range Problem is stated very broadly about general groups G. Although it may help the reader to imagine a more common group like $\mathbb{Z}_m$ or a permutation group while reading the proof, there are other important cases. For example, Theorem 1.2 holds when G is the direct product of many smaller groups. In this case, the problem can be viewed as many disjoint copies of the Group Range problem on the smaller component groups with simultaneous updates.

The case where G is the general linear group of invertible matrices also has many applications; see the full paper for a discussion of applications to physics and to other dynamic data structure problems. For this case, we show how the matrix structure can be exploited to prove even stronger results. For example, as a variant of the original problem, consider the *Matrix Product Problem*, where queries can only ask for a bit about the bottom-right entry of the product of the *entire* range of matrices, rather than any bit about the product of any subrange. In the full version of the paper we show that the lower bound still applies:

Corollary 1.4. *Theorem 1.2 holds for the Matrix Product Problem.*

We show a similar result for upper-triangular matrices as well in the full version.

It would be interesting to extend Theorem 1.2 to hold for an even wider class of algebraic structures. For instance, some past work (e.g. [14]) considers the partial sums problem where G is any *semi-group*. However, we show that such an extension is impossible, not only to semi-groups, but even just to monoids (a type of algebraic structure between groups and semi-groups, which satisfies all the group axioms except the existence of inverses). Indeed, we demonstrate in the full version that the $\Omega(\log n)$ lower bound can be beaten for the *Monoid Range Problem* (the same as the Group Range Problem except that G can be any monoid), so no general lower bound applies:

Theorem 1.5. *There exists a family of monoids $(G_n)_n$ such that the Monoid Range Problem can be solved in $O\left(\frac{\log n}{\log \log n}\right)$ time per operation worst-case deterministically in the cell-probe model.*

1.5.2 Dynamic Connectivity. Dynamic Graph Connectivity is one of the most basic and versatile dynamic graph problems. As such, we can extend Theorem 1.3 to hold for a number of other graph problems. Some examples include:

- Dynamic Entire Graph Connectivity: Maintain a dynamic undirected graph, where queries ask whether the entire graph is connected.
- Dynamic Minimum Spanning Forest: Maintain a dynamic undirected graph, where queries ask for the size of a minimum spanning forest.
- Dynamic Planarity Testing: Maintain a dynamic undirected graph, such that edge insertions are guaranteed to maintain that the graph is planar, and where queries ask whether inserting a specific new edge would result in a non-planar graph.

Corollary 1.6. *Theorem 1.3 holds for Dynamic Entire Graph Connectivity, Dynamic Minimum Spanning Forest, and Dynamic Planarity Testing.*

Corollary 1.6 follows from some straightforward reductions given in [15, Section 9].

1.6 Our Technique and Related Work

Next, we discuss our plan of attack for using the online communication model along with other ideas to prove our data structure lower bounds, Theorems 1.2 and 1.3, and we compare it with the approaches of past work. A more detailed overview of our proofs is given later in Section 3.

Our high-level strategy is similar to previous techniques based on communication complexity for proving cell-probe lower bounds [16, 22, 25]. We first "decompose" the computation being done into several communication games, and show that an efficient data structure would induce efficient protocols for these games. We then prove communication lower bounds for these games, ruling out these supposed efficient protocols. The communication games we wind up with consist of a random sequence of interleaved updates and queries divided into two consecutive blocks of operations. Roughly speaking, in each communication game, the first block is only revealed to Alice while the second block is only revealed to Bob. All

other operations are revealed to both players. The goal of the game is for Alice and Bob to cooperatively answer all the queries in Bob's interval.

The choice of what communication model to use in this strategy is crucial. The first step, transforming a fast data structure into an efficient protocol, can be done more efficiently in a stronger model (e.g. randomized over deterministic). On the other hand, the second step, proving communication lower bounds, is more difficult in a stronger model. Designing the right communication model to balance these two proof phases is a crucial ingredients in these types of proofs.

In this paper, we analyze the communication games in our online communication model. Compared to other models used in previous work, our model has a more natural correspondence to the task that data structures face: answering queries in sequential order. Studying these communication games in our online communication model yields a more fine-grained view of the situation. See Section 3 and the full version for more details on this connection between online communication complexity and dynamic data structures.

1.6.1 Group Range Problem. To illustrate this point, consider the communication games induced by the Group Range Problem. When one analyzes these games in the classical communication models considered by past work, where both players receive their inputs at once, there is a protocol which is too efficient to prove a tight lower bound[2]. In other words, it is provably impossible to use any of the previous communication models at this point in the proof; the communication game is simply not "hard" in any of them. We will see that these communication games are hard enough to prove strong lower bounds in our online communication model.

As stated before, Pătraşcu and Demaine [14] proved a $\Omega(\log n)$ lower bound for the Partial Sums problem (queries want entire product rather than a single bit), when no error is allowed in answering queries. Their *information-transfer technique* does not apply directly to our task, since it relies on the fact that each query outputs many bits and hence reveals a lot of information, and that the data structure has no errors. Their technique was later generalized [15] to prove lower bounds for problems with single-bit output, but their argument mostly focuses on the query which the data structure spends the least amount of time on. It is hard to apply this generalization directly when both overall running time and overall accuracy need to be taken into account. However, it is worth noting that their argument does apply to our Group Range Problem if only zero-error data structures are considered.

1.6.2 Dynamic Connectivity. The high-level structure of our Dynamic Connectivity lower bound proof is close to that of Pătraşcu and Demaine's proof [15]. To prove an $\Omega(n \log n)$ lower bound on the total running time on $O(n)$ operations, both proofs reduce the task to proving that given an initial graph, k updates and k queries, if we perform the updates on the initial graph and then ask the k queries, then there must be a big, $\Omega(k)$-size intersection between the set of cells probed and written to during the insertions, and the set of cells probed during the queries. Intuitively, we need to show that the data structure must learn enough information about the updates in order to answer the queries.

The two proofs then diverge from this point onwards. Pătraşcu and Demaine first set up a hard distribution on updates and queries such that when the answers to all k queries are `Yes`, one is able to reconstruct the k updates exactly based on these queries. Then they use an encoding argument to show that if the data structure only probes $o(k)$ such cells, then the k updates can be encoded very efficiently, contradicting an information theoretical lower bound that they prove using the distribution itself. Roughly speaking, they encode the k updates so that one is able to "simulate" a data structure on any sequence of k queries after the updates based only on the encoding. Then one can iterate over all possible queries, simulate the data structure on all of them, and find the one with k `Yes` answers, which can be used to reconstruct the updates.

Since the information about updates is only hidden in the all-`Yes` queries, and one needs to simulate on a large number of queries before k `Yes` queries are found, Pătraşcu and Demaine's argument fails if the data structure is allowed high two-sided error. In fact, their proof only applies to the case where the error probability of each query is $1/n^c$ for some large constant c. It is not hard to prove that under their input distribution, one will not be able to learn much from the simulations if the error probability of each query is higher than about $1/\sqrt{n}$.

In order to resolve this issue in our Dynamic Connectivity lower bound, we first construct a different hard distribution such that not only the all-`Yes` queries, but even a random set of queries reveals a sufficient amount of information about the updates with high probability. To prove our lower bound, we then use a very different encoding argument, based on the transcript of an online communication protocol. We prove that if an efficient data structure exists, then there is an efficient online communication protocol for the problem where Alice receives the k updates, Bob receives the k queries one at a time, and the goal is to answer all queries. Our encoding argument is more similar to those used in [5] and [22]. See Section 3.2 for a more detailed overview of our approach.

Fredman and Saks [7] and Ramamoorthy and Rao [17] proved a lower bound for the insert-only version of Dynamic Connectivity, where deletion updates are not allowed. They proved that for data structures with worst-case update time and constant error probability, the insert-only version of the problem has to take $\Omega(\log n/\log\log n)$ time per operation. However, the insert-only regime is very different from our fully dynamic regime. For worst-case data structures, the $\log n/\log\log n$ bound is tight [3, 20]. If we allow amortization, the standard union-find solution solves the problem in $O(\alpha(n))$ time per operation. Thus, it is difficult to apply their technique to the fully dynamic regime.

1.7 Organization

We first formally define the online communication model in Section 2, and then in Section 3 we give an overview of all three of our lower bound proofs. We then prove the online set intersection lower bound in Appendix A. We prove our remaining results, including the cell-probe lower bounds for the Group Range Problem and Dynamic Connectivity, in the full version.

[2] For some G, Bob has a succinct encoding of his queries and can send the compressed input to Alice in order to solve the problem more efficiently than the trivial protocol would.

2 THE ONLINE COMMUNICATION MODEL

In this section, we define the online communication model, and then throughout the rest of the paper we present some approaches for proving lower bounds in it. We intentionally try to keep the model quite general. In Appendix A, we showcase our approach by proving a *tight* lower bound for the natural variation of set-intersection for this setting, and thereafter we use the model to prove cell-probe data structure lower bounds.

In the online communication model, there are two players, Alice and Bob. Alice is given her entire input $X \in \mathcal{X}$ at the beginning. Bob will be given his input $Y_1, Y_2, \ldots, Y_k \in \mathcal{Y}$ gradually. The two of them want to compute $f_1(X, Y_1), f_2(X, Y_2), \ldots, f_k(X, Y_k)$ under the following circumstances:

(1) The game consists of k stages. The players remember the transcript from previous stages.
(2) At the beginning of Stage i for $i \in [k]$, Y_i is revealed to *Bob*.
(3) Next, the players communicate as if they were in the classical communication model. After that, Bob must output $f_i(X, Y_i)$.
(4) At the end of Stage i, Y_i is revealed to *Alice*, and the players proceed to the next stage.

Note that the number of stages, k, is fixed and known up-front when designing a protocol. In a deterministic (resp. randomized) online communication protocol, the players communicate as if they were in the deterministic (resp. randomized) communication model in the second step of each stage.

We desire protocols that use the minimum amount of *total* communication. A protocol is free to perform a different amount of communication in each stage. However, there is a natural tension on the proper time to communicate: in earlier stages the players have less information, but they still need to solve their current task at hand before they can proceed. Later on, we will see that the total communication will correspond nicely with the amortized cost of data structure operations.

3 PROOF OVERVIEWS

3.1 Online Set Intersection Lower Bound

In this section we give a high-level overview of how we prove our communication lower bound for online set intersection (OSI). Although the lower bound for OSI is not explicitly used in our data structure lower bounds later, the data structure lower bounds do use online communication lower bounds for other problems which we prove using some common techniques. Our OSI lower bound is, in a sense, a warm-up for the more complex proofs to come.

The main idea behind our OSI lower bound is a very general reduction showing how online communication lower bounds can be proved using techniques from offline communication lower bounds. Consider an offline communication problem called the Index problem, where Alice is given a set $X \subseteq \{1, 2, \ldots, n\}$ of size $|X| = k$, and Bob is only given a single element $y \in \{1, 2, \ldots, n\}$, and their task is to determine whether $y \in X$. One can view the OSI problem as k iterations of Alice and Bob solving the offline Index problem.

That said, it is insufficient to simply prove a lower bound for the Index problem. Since Alice has the same set X in all k iterations, Bob can learn information about it throughout the rounds of the protocol, and so it is plausible that later rounds can be completed with less communication than earlier rounds. In order to circumvent this issue, we prove:

Lemma 3.1. *(informal) There is a protocol for OSI which in total uses $g(n, k)$ bits of communication in expectation* if and only if *there is a protocol for Index where*

(1) Alice first sends $O(g(n, k))$ bits in expectation, then
(2) Alice and Bob speak an additional $O(g(n, k)/k)$ bits in expectation.

The high-level idea for proving the 'only if' direction of Lemma 3.1 is that Alice can begin by telling Bob a cleverly-crafted message containing the information that Bob would learn about X during the OSI protocol on a random sequence of inputs. Thereafter, Alice and Bob can pretend they are in the 'easiest' round of their OSI protocol, which only takes $O(g(n, k)/k)$ bits in expectation to solve. Once we prove Lemma 3.1, it remains to prove a lower bound for the Index problem in the usual offline communication model, which can be done using standard counting techniques.

We actually prove a more general version of Lemma 3.1 for any online communication problem in which Alice and Bob are computing the same function $f = f_i$ in each round (in the case of OSI, f is the Index problem). Unfortunately, for our data structure lower bound proofs, the communication games do not have this property, and more care will be needed.

3.2 Data Structure Lower Bounds

In this section, we give a streamlined overview of our data structure lower bound proofs. The proofs of our lower bounds for Group Range and Dynamic Connectivity both have a similar high-level structure. In both proofs, the first step is to design a hard input distribution. The distribution is supported on sequences of operations consisting of $O(n)$ mixed updates and queries. Then by Yao's minimax principle [23], it suffices to prove a lower bound against *deterministic* data structures dealing with inputs drawn from this distribution.

Next, to prove a lower bound of $\Omega(n \log n)$ for answering the random sequence, we use an idea from [13], which reduces proving a lower bound on total running time to proving lower bounds for many subproblems. Each subproblem is defined by two adjacent intervals of operations of equal length from this random sequence, which are denoted by I_A and I_B, e.g., I_A is the interval consisting of the 17th to the 32nd operation in the sequence, and I_B is the interval consisting of the 33rd to the 48th operation. In each subproblem, instead of the running time (i.e., the number of cell-probes), we are interested in the number of cells that are probed in *both* intervals I_A and I_B. A counting argument from [13] shows that

- *if* for every k and adjacent interval pair (I_A, I_B) of length k, at least $\Omega(k)$ cells are probed in both I_A and I_B,
- *then* the total running time is at least $\Omega(n \log n)$.

In order to prove a lower bound when the data structure's goal is only to maximize the probability of answering $(1/2 + \delta)$-fraction of the queries correctly, we generalize the argument, and show that

- *if* for every δ', k and adjacent interval pair (I_A, I_B) of length k, the probability that $o(k)$ cells are probed in both I_A and I_B and $(1/2 + \delta')$-fraction of the queries in I_B is correct is $\exp(-\delta'^2 k)$,

- *then* the probability that total running time is $o(n \log n)$ and $(1/2 + \delta)$-fraction of the queries is correct in total is $\exp(-\delta^2 n)$.

Thus, the tasks boil down to proving such lower bounds for all subproblems.

3.2.1 Online Communication Simulation. We now focus on a single subproblem (I_A, I_B). We would like to show that if a data structure answers a $(1/2 + \delta)$-fraction of the queries in I_B correctly, then it must probe many cells in I_B which were also probed and written to in I_A. Intuitively, if a data structure probes very few cells in I_B that are probed in I_A, then it learns very little information about the updates in I_A. Thus, if the answer to a random query would reveal one bit of information about the updates in I_A, but the data structure has learned a negligible amount of information about I_A, then the data structure cannot hope to answer the query with a nonnegligible advantage above $1/2$. To formulate the above intuition, we model this process by an online communication game.

Communication Game. We define one communication game for each interval pair (I_A, I_B). Fix two intervals I_A and I_B consisting of k updates and queries each, all the operations O prior to these intervals, all the queries Q_A in I_A and all the updates U_B in I_B. That is, the only undetermined operations up to the end of I_B are the updates in I_A and the queries in I_B; everything else is common knowledge to Alice and Bob. We embed these undetermined operations into a communication game. In the associated online communication game $G = G(v, O, Q_A, U_B)$, X consists of the updates in I_A, and Y_i is the i^{th} query in I_B. The goal of Stage i is to compute the i^{th} query in I_B.

Now we present (an informal version of) our main lemma, which connects the data structures to online communication.

Lemma 3.2 (informal). *For any data structure D, there is a protocol $\mathcal{P}_D$ for the communication game $G(v, O, Q_A, U_B)$ such that*

(1) Bob sends no message;
(2) For every $\beta \in (0, 1)$, the probability that
- *Alice sends $o(k \log n)$ bits, and*
- *$\mathcal{P}_D$ answers a $(\beta - o(1))$-fraction of the $f_i(X, Y_i)$'s correctly* is at least the probability conditioned on O, Q_A, U_B *that*
- *$o(k)$ cells are probed in both I_A and I_B by D, and*
- *D answers a β-fraction of queries in I_B correctly.*

For any data structure D, we construct the protocol P_D as follows.

(1) (Preprocessing) Recall that Alice knows all the operations up to the end of I_A and the updates in I_B, and Bob knows all the operations prior to I_A and all the operations in I_B. First, Alice simulates D up to the end of I_A, and Bob simulates D up to the beginning of I_A and *skips* I_A. Denote the memory state that Alice has *at this moment* by M_A. Next, the players are going to simulate operations in I_B.
(2) (Stage i - Alice's simulation) Since the $(i-1)$-th query is revealed to Alice at the end of the last stage, Alice first continues her simulation of D up to right before the i-th query. Alice then sends Bob the cells (their addresses and contents in M_A) that are
- probed during this part of the simulation, and
- probed during I_A, and
- not probed in the previous stages.

(3) (Stage i - Bob's simulation) Bob first updates his memory state according to Alice's message: For each cell in the message, Bob replaces its content with the actual content in M_A. This is the first time D probes these cells, since otherwise Alice would have sent them earlier, and so their contents remain the same as in M_A. Bob then continues his simulation of D up to the beginning of i-th query.
(4) (Stage i - query answering) Bob now simulates D on query Y_i. During the simulation, Bob *pretends* that he has the right memory state of D for the query, even though he skipped I_A, and has only received partial information from Alice about it. He then outputs whatever answer D gives him. Finally, Bob rolls back his copy of D to the version right before this query (after the simulation described in the previous step). Even though he was assuming his copy of D is correct, it may have actually made a mistake, and at the beginning of the next step, Alice will tell Bob what cells he should have queried and changed.

The key observation to make about the above protocol is that Bob might only give a different answer to query Y_i than the real data structure D would have if D would probe a cell that was written to during I_A while answering Y_i. Moreover, at the beginning of the next stage, Alice would then tell Bob the true value which that cell should have had. Hence, each cell which D would write to in I_A and probe in I_B can cause Bob to make at most one mistake. As such, if D would only probe a negligible number, $o(k)$ of cells in both I_A and I_B, then Bob similarly gives the same answer as a correct D would to all but a negligible number of his queries.

3.2.2 Online Communication Lower Bounds. The tasks now reduce to proving online communication lower bounds. We prove the communication lower bounds for Group Range and Dynamic Connectivity using different approaches.

Communication lower bounds for Group Range. We design the hard distribution such that the k updates in I_A have entropy about $k \log n$. Hence, if Alice sends only $o(k \log n)$ bits to Bob, then Bob knows very little about those updates. In particular, we carefully design the queries such that there is a $\Theta(k \log n)$-bit encoding of the k updates, and each query is essentially asking for one random bit of this encoding. Then on average, every bit is still close to unbiased even after Bob sees Alice's message. That is, Bob will not be able to predict the answer with much better probability than $1/2$.

Furthermore, we prove the above *conditioned on* whether Bob answered the previous queries correctly. Therefore, the sequence of numbers consisting of, for each $1 \le i \le k$, the number of correct answers in the first i queries *minus* its expected value, forms a supermartingale. Applying the Azuma-Hoeffding inequality shows that the probability that at least a $(1/2 + \delta)$-fraction of the queries is correct is at most $\exp(-\delta^2 k)$.

Communication lower bounds for Dynamic Connectivity. The lower bound for the Dynamic Connectivity problem is proved in a different way. To prove the communication lower bound, we first show that it suffices to prove that the probability that all k queries are correct is at most $2^{-(1-o(1))k}$. This would in particular imply that the probability that all k queries are wrong is also at most

$2^{-(1-o(1))k}$. In fact, for any fixed sequence of choices of whether each query is correct or not, we show that this sequence happens with probability no more than $2^{-(1-o(1))k}$. This is, in particular, at most a $2^{o(k)}$ factor more than the probability of achieving the fixed sequence by outputting uniformly independent bits. This implies that the probability that a $(1/2+\delta)$-fraction of the queries is correct is at most $2^{o(k)}$ times the probability of the same event when all the bits are independent, which is $\exp(-\delta^2 k)$.

Next, we prove that when the inputs to the communication problem are independent and Bob does not speak, we may assume without loss of generality that Alice only speaks before the first stage,[3] which we call the Monologue lemma:

Lemma 3.3 (Monologue Lemma (informal)). *Suppose that Alice's input X and Bob's inputs $Y_1, \ldots, Y_k$ are independent, and there is a protocol P such that:*

(1) Only Alice talks.
(2) At most C bits are sent.
(3) All k queries are answered correctly with probability p

Then there is another protocol P' with the following properties:

(1) Only Alice talks, and she only does so in the first stage.
(2) $C + O(\log 1/p)$ bits are sent in expectation.
(3) All k queries are answered correctly with probability at least p.

Using this lemma, we will be able to prove the communication lower bound. Assume for the sake of contradiction that Alice sends $o(k \log n)$ bits and Bob answers k queries correctly with probability at least $2^{-0.99k}$. The high-level idea is to let Alice simulate the protocol and send a message about her input, which takes $o(k \log n)$ bits. Since Bob is able to complete the protocol with no further communication, we know that a random sequence of k queries can be answered correctly based solely on this message with probability $2^{-0.99k}$. The players then treat the public random string as repeated samples of queries. On average, there is one entirely-correct sample of queries in every $2^{-0.99k}$ samples from the public randomness. Thus, it only takes about $0.99k$ bits for Alice to specify each sample that would be answered entirely correctly by Bob. Ideally, each of these samples of k queries reveals k bits of information about Alice's input. That is, in the ideal situation, Alice will be able to save about $0.01k$ bits each time at the cost of sending $o(k \log n)$ extra bits in the beginning. If Alice managed to repeat this much more than $0.001 \log n$ times, and each time revealed about k extra bits of information, she would have revealed $0.001k \log n$ bits of information in total using only $(0.00099 + o(1))k \log n$ bits, which yields a contradiction.

A ONLINE SET-INTERSECTION LOWER BOUND

Online Set-Intersection. In the online set-intersection problem (OSI), Alice is given one set X of size k over the universe $[n]$. In each stage, Bob is given an input $Y_i \in [n]$, which is an element in the same universe. The goal of this stage is to verify whether $Y_i \in X$. Equivalently, the inputs are two (multi-)sets $X, Y \subseteq [n]$ of size k each. Each element of the set Y is revealed one by one. The goal is to compute their intersection.

Theorem A.1. *For $n \geq k^2$, any zero-error OSI protocol using public randomness must have expected total communication cost at least $\Omega(k \log \log k)$.*

It is not hard to see that $\Omega(|X \cap Y| \log n)$ is also a lower bound, since Alice and Bob need to confirm that their elements in common are actually equal; in other words, our combined lower bound is $\Omega(k \log \log k + |X \cap Y| \log n)$. Before we prove Theorem A.1, we give a protocol which shows that this bound is tight.

Lemma A.2. *There is a zero-error OSI protocol using public randomness with expected communication cost $O(k \log \log k + |X \cap Y| \log n)$.*

Proof. The protocol is as follows:

(1) The players use public randomness to sample two uniformly random hash functions $h_1 : [n] \to [k^2]$ and $h_2 : [k^2] \to [k \log k]$, and define $h : [n] \to [k \log k]$ by $h = h_2 \circ h_1$.
(2) Alice sends Bob the set $h(X)$ in $O(\log \binom{k \log k}{k}) = O(k \log \log k)$ bits[4].
(3) For each Y_i:
   (a) If $h(Y_i)$ is not in $h(X)$, Bob returns "NO" immediately.
   (b) Otherwise, Bob sends Alice $h_1(Y_i)$, and Alice tells Bob whether it is in $h_1(X)$. If not, Bob returns "NO" immediately.
   (c) Otherwise, for each $X_j \in X$ such that $h_1(X_j) = h_1(Y_i)$, Alice and Bob determine whether $X_j = Y_i$. They do this with the zero-error protocol for equality which uses $O(\log n)$ bits of communication if $X_j = Y_i$ and $O(1)$ bits of communication in expectation if $X_j \neq Y_i$. If $X_j = Y_i$ they return "YES", and if $X_j \neq Y_i$ for each such $X_j \in X$, they return "NO".

For each $Y_i \notin X$, the probability that $h(Y_i) \in h(X)$ is at most $1/\log k$. Since it takes $O(\log k)$ bits for Bob to send $h_1(Y_i)$ to Alice, the total expected communication cost for stage 3b over all i with $Y_i \notin X$ is $O(k)$. Similarly, for each $Y_i \notin X$, the expected number of $X_j \in X$ such that $h_1(X_j) = h_1(Y_i)$ is $\leq k \cdot \frac{1}{k^2} = 1/k$, and so the total expected communication cost for stage 3c over all i with $Y_i \notin X$ is $O(1)$. Thus, the above protocol has the claimed total communication cost. □

In the following, we prove the communication lower bound. First by Yao's Minimax Principle [23], we may fix an input distribution and assume the protocol is deterministic. Now let us consider the following hard distribution.

Hard distribution. We take the first k^2 elements from the universe, and divide them into k blocks of size k each. X will contain one uniformly random element from each block independently. Each Y_i will be a uniformly random element from the first k^2 elements. Different Y_i's are chosen independently.

The high-level idea of the proof is to first reduce from OSI to a classic (non-online) communication complexity problem. In particular, we consider the problem solved in each stage of the OSI problem: Alice is given a set of k elements from a universe of size

[3] Note that even if Bob does not speak during an online communication protocol, Alice still learns what Bob's inputs are each time Bob finishes answering a query.

[4] Recall that for any integers $n \geq m > 0$ we have $\binom{n}{m} \leq (\frac{n \cdot e}{m})^m$. Hence, $\binom{k \log k}{k} \leq O(\log k)^k$.

n and Bob is given a single element from the same universe, and their goal is to determine if Bob's element is in Alice's set. This is precisely the *index* problem. Then we focus on the stage that costs the least amount of communication, and show an index lower bound with respect to this stage. The hard distribution for OSI induces the following hard distribution for index.

Hard distribution for index. Divide the first k^2 elements of the universe into k blocks of size k each. Alice's set X consists of one uniformly random elements from each block independently. Bob's element y is chosen from the first k^2 elements uniformly at random.

We now prove a general lemma which relates protocols for "symmetric" online communication problems (in which each round is essentially the same problem) with protocols for classical communication problems. Note that when applied to OSI, the associated single-round problem is index. In other words, a protocol for OSI can be transformed into a very rigid protocol for index, which will be easier for us to bound. Additionally, since we prove an iff statement, we know that this transformation is lossless (up to constants).

Lemma A.3. *Suppose we have a problem in our online communication model and associated input distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}^k$ with the following properties:*

(1) The function that Alice and Bob want to compute in each round, $f_i(X, Y_i)$, does not depend on the round number i.
(2) Conditioned on Alice's input $X \in \mathcal{X}$, Bob's inputs $Y_1, \ldots, Y_k \in \mathcal{Y}$ are identically (but not necessarily independently) distributed.

The associated single-round classical problem and associated input distribution are as follows. Alice is given an input $X \in \mathcal{X}$ and Bob is given an input $Y \in \mathcal{Y}$, and they want to compute $f_1(X, Y)$. Their inputs are obtained by drawing an input $(X, Y_1, \ldots, Y_k)$ from $\mathcal{D}$, giving Alice X, and giving Bob $Y = Y_1$.

There is a protocol for the online problem which uses $O(g(n,k))$ bits in expectation if and only if there is a protocol for the associated single-round problem where Alice first sends a message of expected length $O(g(n,k))$ bits and then Alice and Bob only speak an additional $O(g(n,k)/k)$ bits in expectation.

Proof. We first prove the more nuanced forward direction. Suppose we have such a protocol P for the online problem; we want a protocol P' for the associated single-round problem with the above properties.

The key idea is to focus on the stage where the players send the least bits in expectation. Choose $i \in [k]$ such that the players only speak $O(g(n,k)/k)$ bits in expectation in stage i. To solve the associated single-round problem on (X, Y), we use the following protocol P':

- The players pretend that they were given an online input where Alice received X and Bob received $Y_i = Y$. They use public randomness to sample $Y_1, \ldots, Y_{i-1}$ according to $\mathcal{D}$.
- Alice has all the information for the first $i-1$ stages, so she simulates those stages of P *for both players*. Note this is possible because P is an online protocol, and hence this simulation does not depend on any of $Y_i, \ldots, Y_k$. Alice then sends Bob the entire transcript.
- Alice and Bob then communicate to simulate stage i of P, continuing from the transcript that Alice sent in the previous step.
- Bob outputs P's decision about $f_i(X, Y_i)$.

In this protocol P', the first message is sent by Alice in step (2). It has expected length no more than the transcript of P, which is $O(g(n,k))$. The players then simulate stage i in step (3). Since the imaginary input follows distribution $\mathcal{D}$, the expected communication in this step is $O(g(n,k)/k)$. Since the goal of stage i in the online problem is to compute $f_i(X, Y_i)$, which is precisely $f_1(X, Y)$ by our assumption about f and choosing $Y_i = Y$. Hence the output of P' is correct is P is correct.

We finish with the easier reverse direction. Suppose we have such a protocol P' for the associated single-round problem; we want a protocol P for the online problem with the above properties.

By construction, when following P', Alice first sends a message with $O(g(n,k))$ bits in expectation. This message can only depend on her input. Our protocol P also begins with Alice sending this message before Bob begins speaking. Now, in each stage, Bob is given an input Y_i. Alice and Bob can simulate P' on (X, Y_i), but skipping the initial message from Alice since it has already been sent.

In our protocol P, Alice sends $O(g(n,k))$ bits in expectation in her first message. Then in each stage, only $O(g(n,k)/k)$ bits in expectation are transmitted between the players. Note that we just used the assumption that Y_i and Y_1 are identically distributed conditioned on X; this is why P' has the usual expected communication cost when run on (X, Y_i). Thus the total communication cost is $O(g(n,k))$ bits in expectation. □

Let P' be a zero-error protocol for index such that Alice first sends c_0 bits in expectation, and then Alice and Bob communicate for c_A and c_B bits respectively (in expectation). The following lemma lower bounds c_0, c_A, c_B.

Lemma A.4. *For sufficiently large k, any such P' must have either*

- $c_0 \geq \frac{1}{7} k \log k$, *or*
- $c_A \geq c_0 \cdot 2^{-13 \max\{c_B, 1\} \cdot 2^{6c_0/k}}$.

The main idea of the proof is to let Alice simulate Bob. For simplicity, let us first assume the protocol has three rounds: Alice sends c_0 bits, then Bob sends c_B bits, finally Alice sends c_A bits. To simulate Bob, Alice goes over all possible messages that Bob could send, then for each message, sends Bob what she would say if she received that message. If Bob sends at most c_B bits *in worst case*, Alice will be able to complete the above simulate in $c_0 + c_A \cdot 2^{c_B}$ bits of communication. Then Bob will output whether his input Y_i is in Alice's set X. In particular, Alice's message depends only her input X, and Bob can do so for any Y_i. That is, Bob will be able to recover the set X based only on this message, which yields a lower bound on c_0, c_A, c_B.

Proof of Lemma A.4. Without loss of generality, we may first assume $c_B \geq 1$. By Markov's inequality and a union bound, for any $C \geq 2$, with probability at least $1 - 2/C$, Alice sends no more than $C \cdot c_A$ bits and Bob sends no more than $C \cdot c_B$ bits after Alice's first message. The next step is to let Alice *simulate* the entire protocol, and turn it into *one-way communication*.

More specifically, the transcript π of a conversation between Alice and Bob is a binary string, in which each bit represents the

message sent in the chronological order. Given π and a fixed protocol, there shall be no ambiguity in which bits are sent by which player. That is, for any π, we can always decompose it into (π_A, π_B), where π_A is a binary string obtained by concatenating the bits sent by Alice in the chronological order, and similar for π_B. On the other hand, given (π_A, π_B), there is a *unique* way to combine them into a single transcript π, since a prefix of the transcript uniquely determines the player who speaks the next. We know that with probability at least $1 - 2/C$, $|\pi_A| \le C \cdot c_A$ and $|\pi_B| \le C \cdot c_B$. In the new protocol, after Alice sends the first $c_0 \cdot k$ bits, she goes over all $2^{C \cdot c_B}$ strings s of length at most $C \cdot c_B$. For each s (in alphabetical order), she sends the first $C \cdot c_A$ bits of π_A based on her input *assuming* $\pi_B = s$. That is, Alice tells Bob that "if s was your first $C \cdot c_B$ bits of the conversation, then here is what I would say for my first $C \cdot c_A$ bits." In total, she sends another $C \cdot c_A \cdot 2^{C \cdot c_B}$ bits. Thus, Bob can figure out the answer based only on the above messages, with probability $1 - 2/C$ (over the random input pairs). To balance the lengths of two messages, we set $C = \frac{1}{2c_B} \log \frac{c_0}{c_A}$. If $C < 2$, then we have $\log \frac{c_0}{c_A} < 4c_B$, and thus

$$c_A > c_0 \cdot 2^{-4c_B},$$

which implies the second inequality in the statement (note $2^{6c_0/k} \ge 1$, so this is much stronger). Otherwise, the above argument holds, and we have

$$\begin{aligned} C \cdot c_A \cdot 2^{C \cdot c_B} &= C \cdot c_A \cdot \sqrt{\frac{c_0}{c_A}} \\ &= \frac{c_A}{2c_B} \cdot \left(\sqrt{\frac{c_0}{c_A}} \log \frac{c_0}{c_A} \right) \\ &\le c_A \cdot \left(\sqrt{\frac{c_0}{c_A}} \log \sqrt{\frac{c_0}{c_A}} \right) \\ &\le c_A \cdot \frac{c_0}{c_A} = c_0. \end{aligned}$$

Thus, Alice sends at most $2c_0$ bits in expectation in total. This message only depends on her input X. By Markov's inequality, for at least $2/3$ of the X's, Alice sends no more than $6c_0$ bits. By Markov's inequality again, for at least $2/3$ of the X's, the probability (over a random y) that Bob can figure out if $b \in A$ based only on Alice's first message is at least $6/C$. Since there are k^k different possible X's, at least $k^k/3$ different X's have both conditions hold. Thus, there must be $k^k/3 \cdot 2^{-6c_0}$ such X's that Alice sends the same message M. Denote this set of X's by $\mathscr{X}$. Moreover, when M is the message Bob receives, there are at least $(1 - 6/C)k^2$ different y's such that Bob can figure out the answer based only on the value of y and M. Denote this set of y's by $\mathscr{Y}$. In the combinatorial rectangle $\mathscr{R} = \mathscr{X} \times \mathscr{Y}$, for every $y \in \mathscr{Y}$, either $y \in X$ for every $X \in \mathscr{X}$, or $y \notin X$ for every $X \in \mathscr{X}$. That is, $\mathscr{R}$ is a *column-monochromatic rectangle*[5] of size $(k^k/3 \cdot 2^{-6c_0}) \times (1 - 6/C)k^2$.

On the other hand, in any column-monochromatic rectangle $\mathscr{R} = \mathscr{X} \times \mathscr{Y}$ for the index problem, the answer is "YES" in no more than k columns of $\mathscr{Y}$ (the element is in the set). This is because each set $X \in \mathscr{X}$ has size k. In order to upper bound the number of $y \in \mathscr{Y}$ that is not in any X, let r_i for $1 \le i \le k$ be the size of the intersection of $\mathscr{Y}$ and the i-th block of the universe. Thus, the number of X's that avoids all $y \in \mathscr{Y}$ is at most

$$(k - r_1)(k - r_2) \cdots (k - r_k) \le \left(k - \frac{1}{k}(r_1 + \cdots + r_k) \right)^k$$

by the AM-GM inequality. That is, at most $k^2 - k|\mathscr{X}|^{1/k}$ y are not in any X. Overall, we have $|Y| \le k + k^2 - k|\mathscr{X}|^{1/k}$. Combining this with the parameters from the last paragraph, we get

$$(1 - 6/C)k^2 \le k + k^2 - k \left((k^k/3 \cdot 2^{-6c_0}) \right)^{1/k}.$$

Simplifying the inequality yields

$$6/C \ge 2^{-6c_0/k} \cdot 3^{-1/k} - 1/k.$$

When $c_0 < \frac{1}{7} k \log k$, we have $2^{-6c_0/k} \cdot 3^{-1/k} - 1/k > \frac{12}{13} \cdot 2^{-6c_0/k}$ for sufficiently large k. Pluging-in the value of $C (= \frac{1}{2c_B} \log \frac{c_0}{c_A})$ and simplifying, we obtain

$$c_A \ge c_0 \cdot 2^{-13 c_B / 2^{-6c_0/k}}.$$

This proves the lemma. □

Proof of Theorem A.1. For any OSI protocol with total communication cost c, by Lemma A.3 and Lemma A.4, we have either

- $c \ge \frac{1}{7} k \log k$, or
- $c/k \ge c \cdot 2^{-13 \max\{c/k, 1\} \cdot 2^{6c/k}}$.

The second inequality simplifies to $\max\{c/k, 1\} \cdot 2^{\Theta(c/k)} \ge \Omega(\log k)$. Thus, we must have $c \ge \Omega(k \log \log k)$. □

[5] A rectangle with the same function value in every column.

REFERENCES

[1] Miklós Ajtai. 1988. A lower bound for finding predecessors in Yao's cell probe model. *Combinatorica* 8, 3 (1988), 235–247.
[2] Paul Beame and Faith E. Fich. 2002. Optimal Bounds for the Predecessor Problem and Related Problems. *J. Comput. Syst. Sci.* 65, 1 (2002), 38–72.
[3] Norbert Blum. 1985. On the Single-Operation Worst-Case Time Complexity on the Disjoint Set Union Problem. In *STACS 85, 2nd Symposium of Theoretical Aspects of Computer Science, Saarbrücken, Germany, January 3-5, 1985, Proceedings*. 32–38.
[4] Arkadev Chattopadhyay and Toniann Pitassi. 2010. The story of set disjointness. *ACM SIGACT News* 41, 3 (2010), 59–85.
[5] Raphaël Clifford, Allan Grønlund, and Kasper Green Larsen. 2015. New Unconditional Hardness Results for Dynamic and Online Problems. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. 1089–1107.
[6] Gudmund Skovbjerg Frandsen, Peter Bro Miltersen, and Sven Skyum. 1997. Dynamic word problems. *Journal of the ACM (JACM)* 44, 2 (1997), 257–271.
[7] Michael L. Fredman and Michael E. Saks. 1989. The Cell Probe Complexity of Dynamic Data Structures. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*. 345–354.
[8] Johan Håstad and Avi Wigderson. 2007. The Randomized Communication Complexity of Set Disjointness. *Theory of Computing* 3, 1 (2007), 211–219.
[9] Monika R Henzinger and Valerie King. 1999. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM (JACM)* 46, 4 (1999), 502–516.
[10] Kasper Green Larsen and R. Ryan Williams. 2017. Faster Online Matrix-Vector Multiplication. In *SODA*. 2182–2189.
[11] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. 1995. On data structures and asymmetric communication complexity. In *STOC*. 103–111.
[12] Mihai Pătraşcu. 2007. Lower bounds for 2-dimensional range counting. In *STOC*. 40–46.
[13] Mihai Pătraşcu and Erik D Demaine. 2004. Lower bounds for dynamic connectivity. In *STOC*. 546–553.
[14] Mihai Pătraşcu and Erik D Demaine. 2004. Tight bounds for the partial-sums problem. In *SODA*. 20–29.
[15] Mihai Pătraşcu and Erik D Demaine. 2006. Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput.* 35, 4 (2006), 932–963.
[16] Mihai Pătraşcu and Mikkel Thorup. 2011. Don't rush into a union: take time to find your roots. In *STOC*. 559–568.

[17] Sivaramakrishnan Natarajan Ramamoorthy and Anup Rao. 2016. Simplified Data Structure Lower Bounds for Dynamic Graph Connectivity. *Electronic Colloquium on Computational Complexity (ECCC)* 23 (2016), 167. http://eccc.hpi-web.de/report/2016/167

[18] Alexander A Sherstov. 2014. Communication complexity theory: Thirty-five years of set disjointness. In *International Symposium on Mathematical Foundations of Computer Science*. Springer, 24–43.

[19] Daniel D Sleator and Robert Endre Tarjan. 1981. A data structure for dynamic trees. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*. ACM, 114–122.

[20] Michiel H. M. Smid. 1990. A Data Structure for the Union-Find Problem Having Good Single-Operation Complexity. *ALCOM: Algorithms Review, Newsletter of the ESPRIT II Basic Research Actions Program* (1990).

[21] Mikkel Thorup. 2000. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. ACM, 343–350.

[22] Omri Weinstein and Huacheng Yu. 2016. Amortized Dynamic Cell-Probe Lower Bounds from Four-Party Communication. In *FOCS*. 305–314.

[23] Andrew Chi-Chin Yao. 1977. Probabilistic computations: Toward a unified measure of complexity. In *FOCS*. 222–227.

[24] Andrew Chi-Chih Yao. 1981. Should tables be sorted? *Journal of the ACM (JACM)* 28, 3 (1981), 615–628.

[25] Huacheng Yu. 2016. Cell-probe lower bounds for dynamic problems via a new communication model. In *STOC*. 362–374.