Lower Bounds for 2-Dimensional Range Counting

Mihai Pătraşcu
MIT
mip@mit.edu

ABSTRACT

Proving lower bounds for range queries has been an active topic of research since the late 70s, but so far nearly all results have been limited to the (rather restrictive) semigroup model. We consider one of the most basic range problem, orthogonal range counting in two dimensions, and show almost optimal bounds in the group model and the (holy grail) cell-probe model.

Specifically, we show the following bounds, which were known in the semigroup model, but are major improvements in the more general models:

- In the group and cell-probe models, a static data structure of size $n \lg^{O(1)} n$ requires $\Omega(\lg n / \lg\lg n)$ time per query. This is an exponential improvement over previous bounds, and matches known upper bounds.

- In the group model, a dynamic data structure takes time $\Omega\big((\frac{\lg n}{\lg\lg n})^2\big)$ per operation. This is close to the $O(\lg^2 n)$ upper bound, whereas the previous lower bound was $\Omega(\lg n)$.

Proving such (static and dynamic) bounds in the group model has been regarded as an important challenge at least since [Fredman, JACM 1982] and [Chazelle, FOCS 1986].

Categories and Subject Descriptors

E.1 [**Data Structures**]

General Terms

Algorithms, Performance, Theory

Keywords

orthogonal range queries, lower bounds,
cell-probe complexity

1. INTRODUCTION

Range-query problems include some of the most natural and fundamental problems in computational geometry and data structures. The goal is to represent a set of n points in d dimensions, such that *queries* about points in a given *range* can be answered efficiently. In this line of research, the dimension d is a small constant. Range-query problems also have dynamic versions, which support insertions and deletions of points.

Usual choices for the query include *counting* the number of points in the range, *reporting* all points in the range, and testing *emptiness* (testing whether the range contains any point). By far, the most common choice for the range is axis-parallel rectangles $[a_1, b_1] \times \cdots \times [a_d, b_d]$. The term *orthogonal range queries* is typically used for this, though the choice is so common that talking about range queries without further qualification usually means orthogonal range queries. An important special case is *dominance queries*, where query rectangles have the form $[0, b_1] \times \cdots \times [0, b_d]$. Outside orthogonal queries, other choices for the range are half-spaces, simplices, balls etc.

Range queries have been studied intensively, and an overview is well beyond the scope of this work. We instead refer the reader to a survey by Agarwal [1]. It should also be noted that orthogonal range queries, which are simply conjunctions of comparisons, are also very important outside the realm of computational geometry. In fact, they are one of the most natural examples of what computers might be queried for. The introductory lecture of any database course is virtually certain to contain an example like "find employees with a salary under 75000 hired before 2001".

1.1 Models and Lower Bounds

Given these problems have been studied actively for four decades, it is not surprising that a large variety of models has been considered. The strongest accepted model for upper bounds is the word RAM, but there is also interest in weaker models such as pointer machines or functional implementations.

In addition, the counting problem (or, rather, its weighted version) is considered in algebraic models. In the *group* and *semigroup models*, each point has an associated weight from an arbitrary commutative (semi)group and the "counting" query asks for the sum of the weights of the points in the range. The data structure can only manipulate weights through the black-box addition (and, in the group model, subtraction), and must work for any choice of the (semi)group. The running time of a query is the number of algebraic operations performed. Any other computation, i.e. planning algebraic operations based on coordinates, is free.

On the lower-bound side, proving bounds in the *cell-probe model* is considered the holy grail. In this model, memory cells have $w = \Omega(\lg n)$ bits, and store arbitrary information about the input. The cost of the query is the number of memory cells that are read, and any computation on the read data is free. One can see the model is stronger than the RAM, the pointer machine, and likely any imaginable model of computers available today. As a technicality, note that the cell-probe model is incomparable to the algebraic models, because these allow unbounded computation on all data except the (semi)group elements.

The semigroup model has seen a lot of progress on the lower-bound side. Many problems have been considered, and remarkably precise bounds have been shown; see the survey by Agarwal [1] for a comprehensive list of examples. In general, semigroup lower bounds hide arguments of a very geometric flavor. To see why, note than when a value is included in a sum, it can never be taken out again (no subtraction is available). In particular, if a the sum stored in one cell includes an input point, this immediately places a geometric restriction on the set of ranges that could use the cell (the range must include the input point). Thus, semigroup lower bounds are essentially bounds on a certain kind of "decomposability" of geometric shapes.

On the other hand, bounds in the group or cell-probe model require a different, information theoretic understanding of the problem. In the group model, the sum stored in a cell does not tell us anything, as terms could easily be subtracted out. The lower bound must now find certain bottlenecks in the manipulation of information that make the problem hard. On the bright side, when such bottlenecks were found, it was generally possible to use them both for group-model lower bounds (arguing about dimensionality in vector spaces), and for cell-probe lower bounds (arguing about entropy).

Philosophically speaking, the difference in the type of reasoning behind semigroup lower bounds and group/cell-probe lower bounds is parallel to the difference between "understanding geometry" and "understanding computation". Since we have been vastly more successful at the former, it should not come as a surprise that progress outside the semigroup model has been extremely slow.

1.2 Orthogonal Range Counting

We summarize the known results for orthogonal range queries in Table 1, along with our improvements. Before diving into formal details, let us quickly glance at the table.

Orthogonal range counting provides an excellent illustration of the major rift between the semigroup model and the stronger models, which was the topic of the discussion in the previous section. In the semigroup model, we have tight bounds in all cases, at least up to $\lg \lg n$ factors. For the stronger models, however, known bounds are nowhere close to that. Not only do known lower bounds fail to grow appropriately with the dimension, but they are even far from optimal for $d = 2$. For example, in the static case, the lower bound for $d = 2$ is exponentially weaker than the upper bound.

As one might expect, proving good bounds outside the restrictive semigroup model has been recognized as an important challenge for a long time. As early as 1982, Fredman [5] asked for better bounds in the group model for dynamic range counting. In FOCS'86, Chazelle [2] echoed this, and also asked about the static case.

In this paper, we address these challenges for the case $d = 2$, where we obtain an almost perfect understanding. Unfortunately, our techniques cannot obtain better bounds in higher dimensions. Depending on mood, the reader may view this as a breakthrough (e.g. providing the first convincingly superconstant bounds for the static case), or as a lucky discovery that moves the borderline of the big open problem from $d = 2$ to $d = 3$. We believe there is truth in both views.

Formal details. Having delighted ourselves with analyzing the entries of Table 1, let us now see what they mean. For static data structures, the complexity is described by a space/query tradeoff. To simplify exposition, we list the best query time achievable by a data structure of size $n \lg^{O(1)} n$. For dynamic data structures, one has an update/query tradeoff, but we concentrate on the case when query and update times are equal (i.e. one wants a bound on the slowest operation).

The upper bounds have been stated in a variety of sources; see e.g. [2]. Interestingly, the known upper bounds (for the static and the dynamic cases) are the same in all three models. The only exception is the dynamic problem in the cell-probe model. However, we will obstinately avoid a discussion of this problem until the Conclusion.

Dynamic semigroup problems come in two flavors, depending on whether a DELETE operation is allowed. This distinction exists only for semigroups, since otherwise we can delete a value Δ by inserting a value $-\Delta$.

In the cell-probe model, there is also a question about the representation of coordinates. Using a predecessor structure for each of the d coordinates, and running two queries in each, it is possible to reduce all coordinates to *rank space*: $[n] = \{0, \ldots, n-1\}$. On the other hand, it is known [8] that the complexity of the (colored) predecessor query is a lower bound even for dominance range queries in 2 dimensions. Thus, the optimal query time is equal to the optimal running time in rank space, plus the predecessor bound. Since the predecessor bound is well understood [12], we can assume all coordinates are in $[n]$, and concentrate on the pure range counting hardness. In almost all cases, the predecessor bound is a asymptotically smaller, so the additional term is inconsequential anyway.

Our results. The following theorems give formal statements for our lower bounds. We note that the tradeoffs we obtain are very similar to the ones known in the semigroup model. The space/time tradeoffs are known to be tight for space $\Omega(n \lg^{1+\varepsilon} n)$. The query/time tradeoff is tight for update time $t_u = \Omega(\lg^{2+\varepsilon} n)$.

Lower bounds are shown for a fixed set of input points in $[n]^2$, and *dominance* queries chosen uniformly at random in $[n]^2$. When we mention expected running times, we are taking expectation over the choice of the query. By fixing coin flips, this means bounds also hold for Las Vegas randomization.

THEOREM 1. *In the group model, a static data structure of size $n \cdot \sigma$ must take $\Omega(\frac{\lg n}{\lg \sigma + \lg \lg n})$ expected time for dominance counting queries.*

Problem	Model	Lower Bounds	Dimension
static	semigroup	$\Omega\big((\lg n/\lg\lg n)^{d-1}\big)$ [2]	
$O\big((\frac{\lg n}{\lg\lg n})^{d-1}\big)$	group	$\Omega(\lg\lg n)$ $\star$ [3]	$d=2$
		$\mathbf{\Omega(\lg n/\lg\lg n)}$ **new**	$d=2$
	cell-probe	$\Omega(\lg\lg n)$ [12]	$d=2$
		$\mathbf{\Omega(\lg n/\lg\lg n)}$ **new**	$d=2$
dynamic	semigroup, with DELETE	$\Omega(\lg^d n)$ [4]	
$O(\lg^d n)$	semigroup, no DELETE	$\Omega(\lg n/\lg\lg n)$ [13]	$d=1$
		$\Omega\big((\lg n/\lg\lg n)^d\big)$ [2]	
		$\Omega(\lg n)$ [7]	$d=1$
	group	$\Omega(\lg n/\lg\lg n)$ [6]	$d=1$
		$\Omega(\lg n)$ [9]	$d=1$
		$\mathbf{\Omega\big((\lg n/\lg\lg n)^2\big)}$ **new**	$d=2$

Table 1: Old and new results for orthogonal range counting. ($\star$) The bound of [3], starred, says that for n input points and n queries, the offline problem takes $\Omega(n\lg\lg n)$ time.

This is the central contribution of our paper, and is shown in Section 2.

THEOREM 2. *In the cell-probe model with w-bit cells, a deterministic static data structure of size $n\cdot\sigma$ must take $\Omega(\frac{\lg n}{\lg\sigma+\lg w})$ time for dominance counting queries.*

The proof is quite similar to that for the group model. To avoid particularly tedious complications, Section 3 only argues this bound for deterministic data structures. The randomized lower bound is deferred to the final version of this paper.

THEOREM 3. *In the group model, a dynamic data structure which supports updates in expected time t_u requires $t_q = \Omega\big((\frac{\lg n}{\lg t_u+\lg\lg n})^2\big)$ expected time for dominance counting queries.*

This bound is shown in Section 4. Note that the space is irrelevant for the dynamic lower bound. We can also obtain a similar bound in the cell-probe model, though not as general as we might want it to be. We describe this result in the concluding remarks (Section 5).

1.3 Technical Contributions

The group model. A key idea, and starting point is to consider roughly $n/\lg n$ queries at the same time, and bound the number of cells that they read *together.* The analysis is a somewhat elaborate exercise in juggling between this idea and the ancient folk wisdom that hardness of orthogonal range problems comes from bit-reversal permutations.

It is interesting to note that the previous lower bound in the group model, due to Chazelle [3], was shown in the offline version of the problem: given n points and n rectangles (offline), compute all answers using a minimum number of operations. Though proving a superlinear bound for such an offline problem is a significant achievement, the data-structural implications are weak. We obtain exponentially better bounds for data structures, by considering just $n/\lg n$ queries (which, in addition, seems to require a radically different analysis). Unfortunately, our approach cannot prove a superlinear bound for the offline problem. It would be very interesting if one could get the best of both worlds, proving an $\Omega(n\frac{\lg n}{\lg\lg n})$ bound for the offline problem.

Static cell-probe bounds. Switching to the cell-probe model, we note that our bound is part of a richer context. Until recently, the best known technique for proving cell-probe lower bounds on static data structures was a simple reduction to asymmetric communication complexity [8]. Among the most important limitations of this approach is that it is insensitive to polynomial changes in the space. This makes is useless for orthogonal range queries (in rank space), where the problem can be solved trivially in constant time with $O(n^4)$ space.

Together with Mikkel Thorup, we recently [11] showed the first separation between linear- and polynomial-size data structures. This original separation was only doubly logarithmic, but even more recently [10] we were able to show a better query bound of $\Omega(\frac{w}{\lg\sigma+\lg w})$ for space $n\cdot\sigma$, which is currently the highest cell-probe lower bound known for *any* problem. In the usual case $w=\Theta(\lg n)$, this is the same bound that we are proving for range counting. Thus, we are exhibiting a second (and, as argued below, much more natural) example achieving the highest bound currently known.

The fact that we achieve the same bound as [10] is no coincidence. The only known idea for breaking the communication complexity barrier, used by [11], [10] and the current paper, is to consider many queries *at the same time*, and reduce them *en masse* to communication complexity. The bound that we achieve is the highest possible by this technique.

It is quite instructive to compare the *differences* between the current paper and [11, 10]. When considering k queries simultaneously, the previous papers took an additional step of breaking the input into k independent subproblems. Each subproblem was paired up with one query, a standard scenario for communication complexity, which could be analyzed by available tools. Then, it only remained to show a direct sum property, implying that the hardness of k independent instances is $\Omega(k)$ times higher.

This turns out to be the right idea for the predecessor problem, where we obtained [11] matching upper and lower bounds (but remember that the bound was only doubly-logarithmic). However, in the case of the better bound from [10], the idea is less natural. That bound was shown for some very hard problems, such as exact nearest neighbor searching, where we expect the correct bound to be exponentially higher. Given that the problems were so hard, it is not too surprising that we could force some particular structure onto them.

Unfortunately, the direct sum structure is hopeless for easier problems like range counting. To see why, start by observing that the querier must send $\Theta(\lg n)$ bits about the query (unless the data structure sends a prohibitive amount). Now imagine that we are in a subproblem with $\frac{n}{k}$ points. After a predecessor search of negligible complexity, we can reduce to rank space inside the subproblem. But now the querier is down to sending $\Theta(\lg \frac{n}{k})$ bits, so the direct sum property is not true. In fact, it seems very natural that the querier's communication complexity should get smaller as the subproblem gets smaller. The cases where this is not true are pathologically hard problems (like exact nearest neighbor), and the technique seems limited to such applications.

The proofs of the current paper manage to show hardness of multiple queries from basic principles, without direct sum arguments. At some level of abstraction, one can view our proofs as *implicitly* breaking the input into subproblems, but in an adaptive way, only after seeing what the queries do.

The main point that we wish to emphasize is that general tools (direct-sum results) turn out to be too weak for our problem. Thus, we keep just the most basic idea of [11, 10], namely considering multiple queries at the same time. The main contribution of this paper is problem specific, namely understanding range counting in the information-theoretic context of simultaneous multiple queries. It seems this is a lesson that might also be useful for other problems of logarithmic complexity: direct sum doesn't seem to work, and the bounds need to understand the structure of the problem in the presence of multiple queries.

Dynamic lower bounds. The $\Omega((\frac{\lg n}{\lg \lg n})^2)$ lower bound for dynamic range counting in the group model is obtained by combining our static lower bound with a classic tool in dynamic lower bounds: the chronogram technique of Fredman and Saks [6]. In the chronogram technique, one constructs a nearly logarithmic number of epochs, and, traditionally, argues that the query must read at least one cell from each epoch with constant probability. Armed with our static lower bound, however, we can argue that a query must in fact read almost $\lg n$ cells per epoch, and thus obtain an almost $\lg^2 n$ lower bound.

This is obviously not such a surprising idea, but this is the first time it has been applied successfully. We find this argument an interesting proof of concept, which should become mainstream as static lower bounds become better understood.

2. THE GROUP MODEL

2.1 The Hard Instance

Let B and h be parameters to be fixed, where B is a power of two. We will construct a set S with $n = B^h$ points. Specifically, S is fixed according to the bit-reversal permutation: $S = \{(i, \text{REV}(i)) \mid i \in [n]\}$, where $\text{REV}(x)$ reverses the $\lg n$ bits of x.

We will typically think of numbers in base B. Let s be a string of digits in base B (note that initial zeros are relevant, e.g. the string 01 is different from the string 1). We write λ for the empty string. If d is a digit in $[B]$, we write sd when appending the digit d to the string s. For some N and H satisfying $N = B^H$, we define $[N]_{s\star} = \{sd_{|s|+1} \cdots d_H \mid d_i \in [B], (\forall)|s| < i \leq H\}$. That is, $[N]_{s\star}$ represents all H-digit numbers (numbers in $[N]$), which start with prefix s.

For a prefix of digits p, define $S_p = \{y \mid (x, y) \in S,\ x \in [n]_{p\star}\}$. Note that $S_p = \cup_{d \in [B]} S_{pd}$. If p has k digits, $|S_p| = B^{h-k}$, because there is one point at every vertical $x \in [n]_{p\star}$. Furthermore, consecutive elements in S_p are at distance B^k, because the first $k \lg B$ bits of x are fixed, meaning the last $k \lg B$ bits of y are fixed. The remaining bits take all possible values.

We will prove the lower bound by considering a set Q of $\frac{n}{B^2}$ dominance queries, $Q = \{(x_i, y_i) \mid i \in [\frac{n}{B^2}]\}$. Remember that a query (x, y) asks for the sum of the values associated with points in the rectangle $[0, x] \times [0, y]$.

The point (x_i, y_i) is chosen uniformly at random from $B^2 i \leq x_i < B^2(i+1), 0 \leq y_i < n$. As for input points, for a prefix p of digits, we define $Q_p = \{y \mid (x, y) \in Q, x \in [n]_{p\star}\}$. From the restriction on (x_i, y_i) it follows that for all p of $h-2$ digits, $|Q_p| = 1$ (and it contains an independent random value from $[n]$).

For two sets T_L (left) and T_R (right), we define the interleave pattern $\chi(T_L, T_R)$ to be a string in $\{L, R\}^{|T_L|+|T_R|}$ as follows. Sort the elements of T_L and T_R together, placing elements of T_L first in case of ties. The i-th position of the string indicates whether the i-th element in sorted order came from the left or from the right.

We will take special interest in interleave patterns of the form $\chi(S_p, Q_p)$. These are particularly nice to analyze when values in Q_p are spread out. Consider an element $(x, y) \in Q$, where the digits of x are $d_0 \cdots d_{h-1}$. We say (x, y) is *well separated* at level k, if in $\chi(S_{d_0 \cdots d_k}, Q_{d_0 \cdots d_k})$, the R value corresponding to y is flanked by at least $2B$ consecutive L's on each side. For uniformity of notation, this definition considers interleave pattern as a circular string, where the last position is followed by the first.

We define $Q^\star$ to be the elements of Q which are well-separated at all levels. Let $Q^\star_p = Q_p \cap Q^\star$. For the rest of the proof, we essentially only look at elements of $Q^\star$. This is made possible by:

OBSERVATION 4. *Assume $B \geq 20000h$. Then with probability at least $\frac{99}{100}$, $|Q^\star| \geq \frac{99}{100}|Q|$.*

PROOF. Let p have i digits. A query from Q_p is not well separated on level i if its y-coordinate lands close to another value from Q_p, in terms of rank in S_p. As observed above, S_p has B^{h-i} uniformly spaced elements, so the rank of a random query is uniformly distributed. There are only B^{h-k-2} queries in Q_p, so our fixed query will conflict with another one with probability $< \frac{B^{h-i-2} \cdot 2B}{B^{h-i}} = \frac{2}{B}$. By a union bound over h levels, an element is not included in $Q^\star$ with probability at most $\frac{2h}{B} \leq \frac{1}{10000}$. The conclusion follows by linearity of expectation and the Markov bound. □

To aggregate information about the interleave patterns, define Υ_i to be the collection of the patterns $\chi(S_p, Q^\star_{pd})$, for all prefixes p of i digits, and all $d \in [B]$.

OBSERVATION 5. *The set $Q^\star$ is determined completely from $\Upsilon_0, \dots, \Upsilon_{h-1}$.*

PROOF. From Υ_{h-1}, we know the x coordinates of all points in Q (each set in this collections is either a singleton or empty). We now extract the y coordinates by looking at the Υ_i's in reverse. Let us concentrate on some point with some x coordinate given by digits $d_0 \cdots d_{h-1}$. We show by induction that after seeing $\Upsilon_i, \dots, \Upsilon_{h-1}$, we know its y coordinate relative to $S_{d_0 \cdots d_i}$. Knowing the y coordinate relative to S_λ is full information, as $S_\lambda = [n]$.

Assume the property for $i+1$ by induction. We know where the y coordinate fits in $S_{d_0 \cdots d_{i+1}}$ (say, between y_1 and y_2). Furthermore, since the point is well separated, it is the only y coordinate from $Q_{d_0 \cdots d_{i+1}}$ fitting between y_1 and y_2. Then, we see Υ_i, which tells us the interleave of $S_{d_0 \cdots d_i}$ and $Q_{d_0 \cdots d_{i+1}}$. Since we know y is unique is $Q_{d_0 \cdots d_{i+1}} \cap [y_1, y_2)$, the interleave pattern tells us where it lies among $S_{d_0 \cdots d_i}$. $\square$

2.2 Information as Entropy and Rank

Since the data structure must work for any group, we can choose the group $(\Re, +)$. This choice makes it easy to interpret everything in vector spaces, and make dimensionality arguments.

Consider a set of queries $Q \subset [n]^2$. A query (x, y) asks for a linear combination of point weights (in particular, the sum of all weights for points in $[0, x] \times [0, y]$). The collection of answers to all queries in Q represent a linear map from the input weights to $|Q|$ values. It makes sense to talk about the *rank* of Q, that is the rank of the linear map defined by Q. The following observation justifies our interest in the rank:

Observation 6. *Assume that an algorithm can answer any query in Q by seeing only a set M of memory cells. Then, $|M| \geq \operatorname{rank}(Q)$.*

Proof. In the group model, the answer to any query is a linear map of the cells probed. Then, the rank of the computed answers is at most $|M|$. $\square$

Let E be an event, and assume for simplicity that it fixes $|Q^\star|$ to some value $\geq \frac{9}{10}|Q|$. Conditioned on E, certain choices for the set Q are still possible, while others are not. We let $Q[E]$ be the union of all possible choices for Q, i.e. all queries that could still be asked given E. The crux of the argument is that if E is not too revealing (from an entropy perspective), $\operatorname{rank}(Q[E])$ must be large.

It turns out that the right entropy to measure is $H(\Upsilon_i \mid E, \Upsilon_{i+1}, \ldots, \Upsilon_{h-1})$.

Observation 7. $H(\Upsilon_i \mid E, \Upsilon_{i+1}, \ldots, \Upsilon_{h-1}) \leq |Q| \lg B$.

Proof. Note that Υ_{i+1} describes the interleaving patterns for every prefix of $i+1$ digits. When switching to prefixes of i digits, $B-1$ points get inserted between every two points that were consecutive. Thus, it suffices to describe where each query fits among B choices. $\square$

We now show that if the entropy is high for at least one i, we get a lower bound on $\operatorname{rank}(Q[E])$.

Lemma 8. *If there exists an $i < h-2$ such that*

$$H(\Upsilon_i \mid E, \Upsilon_{i+1}, \ldots, \Upsilon_{h-1}) \geq \frac{9}{10}|Q| \lg B,$$

then $\operatorname{rank}(Q[E]) \geq \frac{1}{40}|Q| \cdot B^{1/3}$.

Proof. By inspecting $\Upsilon_{h-1}, \Upsilon_{h-2}, \ldots, \Upsilon_{i+1}$ and following the reasoning of Observation 5, we learn for any query in $Q^\star$ where its y coordinate fits relative to the corresponding $S_{d_0 \cdots d_{i+1}}$. In this analysis, we discovered some queries outside $Q^\star$ (those which fail to be well-separated for some level above i). We eliminate such queries from discussion.

Furthermore, we preventively eliminate queries which are "close" on level i, according to the following criterion. Say after looking at $\Upsilon_{h-1}, \Upsilon_{h-2}, \ldots, \Upsilon_{i+1}$, we concluded that some query fits in $[y_1, y_2)$ and some other in $[y_1', y_2')$. If $[y_1, y_2) \cap [y_1', y_2') \neq \emptyset$, we remove both queries from consideration. Note that between y_1 and y_2, which are consecutive elements with regard to Υ_{i+1}, exactly $B-1$ elements get inserted. Similarly for y_1', y_2'. Then, if the intervals intersect, the two queries have at most $2(B-1)$ consecutive L's between them at level i. Thus, they were not well-separated at level i, implying our rule never removes points from $Q^\star$.

After all this pruning, we are left with all queries in $Q^\star$ and possibly more. We now perform one final step, possibly eliminating queries from $Q^\star$: we remove queries with $d_{i+1} < \frac{B}{16}$. Originally, there were exactly $|Q|/16$ such queries. We are now working with at least $|Q^\star| \geq \frac{9}{10}|Q|$ queries, so even after eliminating those with bad d_{i+1}, we still have more than $\frac{4}{5}|Q|$ queries left.

Fix some prefix p of i digits. Since for every value in Q_p (that we haven't eliminated) we know where it fits in some S_{pd}, there are B choices for where it can fit in S_p. By the second step of pruning, which eliminated close queries, these choices are distinct for all remaining queries. We now want to lower bound the number of choices, based on the high entropy.

For every query that was eliminated, we can lose at most $\lg B$ bits of entropy. Thus, the entropy remaining at the end is at least $\frac{7}{10}|Q| \lg B$. As we still have $\frac{4}{5}|Q|$ queries remaining, it follows by subadditivity that the entropy of at least $\frac{2}{5}|Q|$ queries is at least $\frac{3.5}{10} \lg B$. For these queries, the support is at least $B^{1/3}$.

Thus, $|Q[E]| \geq \frac{2}{5}|Q|B^{1/3}$, but this does not imply $Q[E]$ also has this rank. For that, we need a few more observations. First, let us consider prefixes in order. Queries in Q_p include weights from S_q when $q < p$ (compared as numbers), but queries in Q_q do not include weights from S_p. (In other word, the matrix describing the linear map associated with Q is "subdiagonal" by blocks.) Thus, it suffices to lower bound the rank of Q_p when projected onto the subspace spanned by S_p. These ranks are then additive.

We can lower bound the rank by projecting just on the subspace spanned by elements of S_p with $d_{i+1} < \frac{B}{16}$. Since all remaining queries Q_p have $d_{i+1} \geq \frac{B}{16}$, whether a point is included in some query is now a one-dimensional problem depending on the interleaving of y's. Since the support intervals of all query are linearly separated with respect to the y-axis (by step 2 of pruning), the rank lower bound is the number of interleaves between the support of Q_p, and S_p restricted to $d_{i+1} < \frac{B}{16}$. Due to the structure of the bit reversal permutation, two values of S_p which are consecutive among points with $d_{i+1} < \frac{B}{16}$ are separated by exactly 15 values from all S_p. Thus, the number of interleaves is at most 16 times lower than the support for Q_p. $\square$

2.3 Proof of the Lower Bound

We are finally ready to prove our lower bound. By Observation 5, the sequence of Υ_i's reveals $Q^\star$, so it has entropy at least $|Q^\star| \lg n$. By Observation 4, $|Q^\star| \geq \frac{99}{100}|Q|$ with probability $\frac{99}{100}$, so $H(\Upsilon_0, \ldots, \Upsilon_{h-1}) \geq \frac{98}{100}|Q| \lg n$.

Let t be an upper bound on the expected running time of a query. The $|Q|$ queries, taken together, read a set M of cells from the memory. Since query i is uniformly random in the range $[B^2 i, B^2(i+1) - 1] \times [n]$, a random query from Q will actually be uniform in $[n]^2$. Thus, by linearity of expectation, $\mathbf{E}[|M|] \leq |Q|t$. Note that M is a function of Q since Q contains all random choices ever made. By Markov,

$|M| \le 3|Q|t$ with probability 2/3.

Since there are S cells in total, we have $H(M \mid |M| \le 3|Q|t) = O(\lg \binom{n\sigma}{3|Q|t}) = O(|Q|t \lg \frac{n\sigma}{|Q|t})$. By averaging, there exists a choice M_0, with $|M_0| \le 3|Q|t$ such that the event $E = \left[M = M_0 \wedge |Q^\star| \ge \frac{9}{10}|Q|\right]$ reveals $O(|Q|t \lg \frac{n\sigma}{|Q|t})$ bits of information with respect to the universe (i.e. conditioning on E reduces entropy by at most this much). We can now write:

$$\begin{aligned} H(\Upsilon_0, \dots, \Upsilon_{h-1} \mid E) &= \sum_i H(\Upsilon_i \mid E, \Upsilon_0, \dots, \Upsilon_{i-1}) \\ &\ge \frac{98}{100}|Q| \lg n - O\left(|Q|t \lg \frac{n\sigma}{|Q|t}\right) \end{aligned}$$

Assume for contradiction that $t \le \varepsilon \lg n / \lg \frac{n\sigma}{|Q|t}$, for a small enough constant ε. With an appropriate choice of ε, the lower bound from above is at most $\frac{97}{100}|Q| \lg n$.

Remember that $\lg n = h \lg B$. Furthermore, $H(\Upsilon_{h-1})$ and $H(\Upsilon_{h-2})$ are each at most $|Q| \lg B$, so they are at most $\frac{1}{100}|Q| \lg n$, for h bigger than a constant. Then, by averaging among the terms for $i < h-2$, we can find an i such that $H(\Upsilon_i \mid E, \Upsilon_0, \dots, \Upsilon_{i-1}) \ge \frac{96}{100}|Q| \lg B$.

Now Lemma 8 implies that $\mathrm{rank}(Q[E]) \ge \frac{1}{40}|Q| \cdot B^{1/3}$. By Observation 6, $\mathrm{rank}(Q[E]) \le |M_0| \le 3|Q|t$. Thus, $3t \ge \frac{1}{40}B^{1/3}$. But B is still an unspecified parameter, so setting $B = \max\{10^9 t, 20000 \cdot h\}$ gives a contradiction. What this contradicts is our earlier assumption that $t \le \varepsilon \lg n / \lg \frac{n\sigma}{|Q|t} = \varepsilon \lg n / \lg \frac{n\sigma}{(n/B^2)t}$. Thus, we have shown a lower bound on t. Since we have $h, t, h = O(\lg n)$, we always have $B = \lg^{O(1)} n$, so the lower bound is $t = \Omega(\frac{\lg n}{\lg \sigma + \lg\lg n})$.

3. THE CELL-PROBE MODEL

The hard set of points is fixed in the same way, but the weights are now uniform random bits. Equivalently, this is the counting problem without weights: the data structure is given a random set $S' \subset S$, where each element of S is included in S' with probability 1/2. We ask the query to report only the parity of the points inside the rectangle. This is sufficient for the lower bound, and has nice consequences, like turning range counting into a decision problem.

For the lower bound, we again consider a set of queries Q in parallel. We reduce this to a communication game where Alice has the set Q, Bob has the set S', and they must compute the answer vector in $\{0,1\}^{|Q|}$. The game will have t rounds, where t is the cell-probe complexity. In round i, Alice sends the set of probes made by all queries in step i. Sending this set requires on the order of $\lg \binom{n\sigma}{|Q|} = \Theta(|Q| \lg \frac{n\sigma}{|Q|})$ bits. Then, Bob replies with the contents of the cells, using $|Q| \cdot w$ bits, and Alice can simulate step i for her queries.

We now apply a standard rectangle argument in communication complexity, fixing each message to its most likely value. At the end, we obtain a rectangle (a cartesian product of a set of inputs for Alice and a set of inputs for Bob). Because there are no more messages, Alice must be able to output the answer to the queries without additional information.

Let the sides of the rectangle be $\mathfrak{S}$ for Bob and $\mathfrak{Q}$ for Alice. Before the rectangle argument, we can restrict our attention to Q such that $|Q^\star| \ge \frac{99}{100}|Q|$, ensuring this condition holds for $Q \in \mathfrak{Q}$.

For the analysis, assume Q and S are uniformly distributed as before (note that randomization is only for the sake of the analysis). Observe that $H(Q) - H(Q \mid Q \in \mathfrak{Q}) = O(t \cdot |Q| \lg \frac{n\sigma}{|Q|})$, because in each round Alice communicated $O(|Q| \lg \frac{n\sigma}{|Q|})$ bits, and we fixed the most likely message. The event E used by the group lower bound can now be defined by $Q \in \mathfrak{Q}$.

Assume for contradiction $t \lg \frac{n\sigma}{|Q|} < \varepsilon \lg n$, for small enough ε. The information revealed by E is as small as before, so we can apply the same analysis culminating in Lemma 8, which shows $\mathrm{rank}(Q[E]) \ge |Q| B^{\Omega(1)}$.

Before, this implied a lower bound immediately, because the number of read cells bounded the rank. Now, we convert this into an equally simple entropy argument. Remember that Alice can now output answers without additional communication. The set of queries which can be asked for various inputs in the rectangle are precisely $Q[E]$. Thus, the output to all these queries must be fixed, *regardless* of the Bob's input $S' \in \mathfrak{S}$. This follows from the fact that we are in a rectangle.

Fixing the output on $Q[E]$ imposes $\mathrm{rank}(Q[E])$ linearly independent constraints over Bob's input (viewed as a vector in $\mathbb{Z}_2^n$). This is only possible if $H(S') - H(S' \mid S' \in \mathfrak{S}) = \Omega(\mathrm{rank}(Q[E]))$. On the other hand, $H(S') - H(S' \mid S' \in \mathfrak{S}) = O(t \cdot |Q| w)$, by the bound on Bob's messages. Hence, we have a contradiction if $|Q| B^{\Omega(1)} = \Omega(t|Q|w)$, i.e. for $B = (tw)^{O(1)} = w^{O(1)}$.

4. DYNAMIC LOWER BOUNDS

Let β be a parameter to be determined. For $i = 1, \dots, \log_\beta n$, we define epoch i to contain β^i queries. Epochs occur in time from biggest to smallest. At the end of epoch 1, we run a uniformly random query in $[n]^2$.

To construct the updates of epoch i, consider the hard instance from Section 2 on the grid $[\beta^i]^2$. We then blow up this grid by $(x, y) \mapsto (x \cdot \frac{n}{\beta^i}, y \cdot \frac{n}{\beta^i})$. Note that we still have just β^i points, but arranged more sparsely. This transformation means a uniform query in $[n]^2$ is hard for these points, because we might as well invert the transformation, and bring everything back to $[\beta^i]^2$. (The query will not have integer coordinates, but rounding does not change the answer.)

At the time of the query we associate each cell with the last epoch that updated it. Since the update sequence is fixed, the query knows which cells were written in each epoch. We now wish to argue that on average over the choice of the query, for any i, the query needs to read $\Omega\big(\lg(\beta^i)/\lg(t_u \lg \beta^i)\big)$ cells from epoch i. This is done by transforming epoch i into a data structure. Consider the following types of cells:

- cells written during epoch i. There are $\beta^i t_u$ such cells, and we collect them into a data structure of size $\beta^i \sigma = \beta^i t_u$.
- cells written before epoch i can be assumed to be zero (i.e. any probe to them is ignored). Indeed, such cells cannot reflect updates from epoch i (they occurred back in time), so any terms they contain must cancel eventually during the query's computations.
- cells written after epoch i, which may of course contain useful sums of weights from epoch i. There are $\beta^{i-1} t_u$ such cells. We assume the query reads them *all*.

The last step seems deadly, seems it increases the query complexity significantly. However, remember that our lower bound for n points is shown by considering $n/\lg^c n$ queries (for some constant c), and bounding the cells that they must read together. The cells written after epoch i are shared by all these queries, so they only contribute additively. Thus, if we set β such that $\beta^{i-1}t_u = \beta^i/\lg^c(\beta^i)$, this additive increase is a lower order term, and we can immediately use the lower bound for the static problem that we have shown in Section 2 (for $n = \beta^i$, $\sigma = t_u$). Our choice implies $\beta = t_u \lg^{O(1)} n$.

We can now sum the lower bounds over all epochs, by linearity of expectation, because the same query distribution works for all epochs. Note that for half of the epochs $\beta^i \geq \sqrt{n}$, so we obtain $\Omega\big((\frac{\lg n}{\lg t_u + \lg\lg n})^2\big)$.

5. CONCLUSIONS

So far, we have not talked about the dynamic problem in the cell-probe model. There, it is not as easy to apply the chronogram technique to relate the static and dynamic problems. The trouble (which is standard in dynamic cell-probe complexity [9]) is that the data structure can be adaptive, so the query has no idea which cells where written in which epochs.

We mention that it is possible to use the set-separator machinery of [9] (a.k.a. Bloomier filters), and obtain an $\Omega\big((\frac{\lg n}{\lg\lg n})^2\big)$ lower bound, but only under two assumptions. First, we change from pure counting to the weighted version of the problem, and assume weights have w bits, matching the word size. Second, we assume $w = \lg^{2+\varepsilon} n$. Thus, the weights (and consequently, the words), need to be somewhat big.

Note that these assumptions are not all that bad, and this may be regarded as an interesting lower bound. Nonetheless, we feel it is a very important open problem to circumvent these limitations, and prove a dynamic lower bound for the regular counting problem without weights.

6. REFERENCES

[1] P. K. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry (2nd edition)*. Chapman & Hall/CRC, 2004.

[2] B. Chazelle. Lower bounds for orthogonal range searching II. The arithmetic model. *Journal of the ACM*, 37(3):439–463, 1990. See also FOCS'86.

[3] B. Chazelle. Lower bounds for off-line range searching. *Discrete & Computational Geometry*, 17(1):53–65, 1997. See also STOC'95.

[4] M. L. Fredman. A lower bound on the complexity of orthogonal range queries. *Journal of the ACM*, 28:696–705, 1981.

[5] M. L. Fredman. The complexity of maintaining an array and computing its partial sums. *Journal of the ACM*, 29(1):250–260, 1982.

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

[7] H. Hampapuram and M. L. Fredman. Optimal biweighted binary trees and the complexity of maintaining partial sums. *SIAM Journal on Computing*, 28(1):1–9, 1998. See also FOCS'93.

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

[9] M. Pătraşcu and E. D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35(4):932–963, 2006. See also SODA'04 and STOC'04.

[10] M. Pătraşcu and M. Thorup. Higher lower bounds for near-neighbor and further rich problems. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 646–654, 2006.

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

[12] M. Pătraşcu and M. Thorup. Randomization does not help searching predecessors. In *Proc. 18th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 555–564, 2007.

[13] A. C.-C. Yao. On the complexity of maintaining partial sums. *SIAM Journal on Computing*, 14:277–288, 1985.