Cache-Oblivious Range Reporting With Optimal Queries Requires Superlinear Space

Peyman Afshani*
MADALGO
Dept. of Computer Science
University of Aarhus
IT-Parken, Aabogade 34
DK-8200 Aarhus N
Denmark
peyman@madalgo.au.dk

Chris Hamilton†
Faculty of Computer Science
Dalhousie University
6050 University Avenue
Halifax, NS B3H 1W5
Canada
chamilton@cs.dal.ca

Norbert Zeh‡
Faculty of Computer Science
Dalhousie University
6050 University Avenue
Halifax, NS B3H 1W5
Canada
nzeh@cs.dal.ca

ABSTRACT

We consider a number of range reporting problems in two and three dimensions and prove lower bounds on the amount of space required by any cache-oblivious data structure for these problems that achieves an optimal query bound of $O(\log_B N + K/B)$ block transfers in the worst case, where K is the size of the query output.

The problems we study are three-sided range reporting, 3-d dominance reporting, and 3-d halfspace range reporting. We prove that, in order to achieve the above query bound or even a bound of $O((\log_B N)^c(1 + K/B))$, for any constant $c > 0$, the structure has to use $\Omega(N(\log\log N)^\varepsilon)$ space, where $\varepsilon > 0$ is a constant that depends on c and on the constant hidden in the big-Oh notation of the query bound.

Our result has a number of interesting consequences. The first one is a new type of separation between the I/O model and the cache-oblivious model, as I/O-efficient data structures with the optimal query bound and using linear or $O(N \log^* N)$ space are known for the above problems. The second consequence is the non-existence of a linear-space cache-oblivious persistent B-tree with worst-case optimal 1-d range reporting queries.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms

Algorithms, Design, Theory

Keywords

Memory hierarchies, cache-obliviousness, data structures, lower bounds, range searching

1. INTRODUCTION

Range reporting is a well studied fundamental problem in computational geometry. Given a set S of points in $\mathbb{R}^d$, the problem is to preprocess S so that, for any query range q, all points in $S \cap q$ can be reported efficiently. Typical types of query ranges include axis-aligned boxes, circles, simplices or halfspaces. In the plane, the special case of *three-sided range reporting* considers axis-aligned boxes whose top boundaries are fixed at $+\infty$. *Dominance reporting* is another important special case: given a query point q, the problem is to report all points in S that are dominated by q, that is, whose coordinates are less than q's in all dimensions.

Most previous work on this type of problems has focused on standard models of computation such as the RAM model or the pointer machine model. The distinguishing feature of these models is that the access cost to a data item is independent of the location where the item is stored in memory. These models are useful for studying the fundamental computational difficulty of a problem, but they ignore that in reality the time to access an item can vary by up to a factor of 10^6 depending on its present location (disk, internal memory, CPU cache, etc.).

Two models have emerged that have proven useful for modelling non-uniform memory access costs: the *input-output model* (or *I/O model*) [6] and the *cache-oblivious model* [16]. The I/O model considers two levels of memory: a slow but conceptually unlimited *external memory* and a fast *internal memory* with the capacity to hold M data items. All computation has to happen on data in internal memory. The transfer of data between internal and external memory happens in blocks of B consecutive data

*Part of this work was done while visiting Dalhousie University.

†Supported by a Killam Graduate Scholarship.

‡Supported in part by the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, and the Canada Research Chairs programme.

SCG'09, June 8–10, 2009, Aarhus, Denmark.

Query type	Space	Query bound	References/Notes
2-d three-sided	N	$\log N + K$	internal memory, [21]
	N	$\log_B N + K/B$	I/O model, [10]
	$N \log N$	$\log_B N + K/B$	cache-oblivious, [3, 4, 7, 11]
2-d dominance	N	$\log_B N + K/B$	cache-oblivious, [11]
3-d dominance	N	$\log N + K$	internal memory, [1, 20]
	N	$\log_B N + K/B$	I/O model, [1]
	$N \log N$	$\log_B N + K/B$	cache-oblivious, [3]
3-d halfspace	N	$\log N + K$	internal memory, [2]
	$N \log^* N$	$\log_B N + K/B$	I/O model, [2]
	$N \log N$	$\log_B N + K/B$	cache-oblivious, [3]

Table 1: A summary of related work on three-sided range reporting, 2-d and 3-d dominance reporting, and 3-d halfspace range reporting with the optimal query bound.

items; the complexity of an algorithm is the number of such *block transfers* it performs.

The cache-oblivious model provides a simple framework for building algorithms for multi-level memory hierarchies, while using the simple two-level I/O model for the analysis. In this model, the algorithm is oblivious of the memory hierarchy and, thus, cannot initiate block transfers explicitly. Instead the swapping of data between internal and external memory is the responsibility of a paging algorithm, which is assumed to be *offline optimal*, that is, performs the minimum number of block transfers for the memory access sequence of the algorithm. Since the memory parameters are used only in the analysis, the analysis applies to *any* two consecutive levels of the memory hierarchy. In particular, if the analysis shows that the algorithm is optimal with respect to two levels of memory, then it is simultaneously optimal at all levels of the memory hierarchy. See [16] for more details and for a justification of the optimality assumption of the paging algorithm.

1.1 Related Work

In the I/O model, much work has focused on orthogonal range reporting. A number of linear-space structures were proposed that achieve a query bound of $O(\sqrt{N/B} + K/B)$ block transfers in two dimensions and $O((N/B)^{1-1/d}+K/B)$ block transfers in d dimensions [9,17,18,19,22,24], where K is the number of reported points. The same bounds were obtained in the cache-oblivious model [4,8]. In 2-d, Arge et al. [10] showed that $\Theta(N \log_B N/ \log_B \log_B N)$ space is sufficient and necessary to obtain a query bound of $O(\log_B N + K/B)$ block transfers for orthogonal range reporting in the I/O model. The lower bound, when applied to blocks of size N^ε, implies that achieving the optimal query bound cache-obliviously requires $\Omega(N \log N)$ space. The main tool used to prove the upper bound is an I/O-efficient version of McCreight's priority search tree [21] with a query bound of $O(\log_B N + K/B)$ for three-sided queries and using linear space.

In the cache-oblivious model, three structures were proposed that achieve a query bound of $O(\log_B N + K/B)$ for three-sided queries, but using $O(N \log N)$ space. The first one, by Agarwal et al. [4], requires that $\log \log B$ is an integer. The second and third structures, by Arge et al. [7] and Arge and Zeh [11], remove this restriction. The structure by Arge and Zeh was obtained by combining standard techniques with a linear-space structure for optimal 2-d dominance queries proposed in the same paper.

For 3-d dominance reporting in the I/O model, Vengroff and Vitter [25] presented a data structure with a query bound of $O(\log(N/B) \log \log \log_B N + K/B)$ block transfers and using $O(N \log(N/B))$ space. The query bound can be reduced to $O(\log_B N + K/B)$ using a careful choice of parameters [26]. Afshani [1] showed that an optimal query bound can in fact be obtained using linear space, raising the question whether this result can be achieved in the cache-oblivious model as well. In [3], we show that the optimal query bound can indeed be achieved by a cache-oblivious structure, but our structure uses $O(N \log N)$ space.

Halfspace range reporting in 3-d has a longer history, particularly because it can be used to solve other problems, such as 2-d circular range reporting and 2-d k-nearest neighbour searching. In internal memory, Chan described an $O(N \log N)$-space structure with an expected query time of $O(\log N+K)$ [14]. Building on these ideas, Agarwal et al. [5] obtained an $O(N \log N)$-space structure with an expected query bound of $O(\log_B N+K/B)$ in the I/O model. Further research led to the development of internal-memory structures with the optimal query bound in the worst case and using $O(N \log \log N)$ space [15,23]. The same improvements can be carried over to the I/O model. Recently, Afshani and Chan [2] described a linear-space structure with the optimal query bound in internal memory and an $O(N \log^* N)$-space structure that answers queries using $O(\log_B N+K/B)$ block transfers in the I/O model. In [3], we show how to achieve the optimal query bound in the cache-oblivious model, using $O(N \log N)$ space. Table 1 summarizes these results.

1.2 New Results

As discussed in the previous section, there exist linear- or $O(N \log^* N)$-space structures that achieve the optimal query bound of $O(\log_B N + K/B)$ for three-sided range reporting, 3-d dominance reporting, and 3-d halfspace range reporting in the I/O model. In contrast, the best known structures in the cache-oblivious model use $O(N \log N)$ space. This raises the question of whether linear-space cache-oblivious structures with the optimal query bound exist for these problems. In this paper, we give a negative answer to this question. In particular, we prove that any cache-oblivious data structure for three-sided range reporting, 3-d dominance reporting, or 3-d halfspace range reporting that achieves a query bound of $O((\log_B N)^c(1+K/B))$ block transfers, for any $c > 0$, has

to use $\Omega(N(\log\log N)^{\varepsilon})$ space, where ε depends on c and on the constant factor in the query bound. As a consequence of our lower bound, it follows that there is no linear-space cache-oblivious persistent B-tree that achieves the optimal 1-d range searching bound of $O(\log_B N + K/B)$ block transfers in the worst case, while such a structure is known to exist in the I/O model.

There have been previous results showing that the cache-oblivious model is less powerful than the I/O model. Brodal and Fagerberg [13] established a lower bound on the amount of main memory necessary for optimal cache-oblivious sorting, while Bender et al. [12] proved that searching in the cache-oblivious model has to cost a constant factor more than the search bound achieved in the I/O model using B-trees. In contrast to these results, our result establishes a gap between the resource consumption of cache-oblivious and I/O-efficient structures that grows with the input size.

The key to obtaining our result is the construction of a hard point set, a set of hard queries over this point set, and techniques to explicitly use the multi-level structure of the cache-oblivious model. Previous lower bound proofs for range reporting problems in the I/O model also involved the construction of a hard point set together with a hard query set consisting of many "sufficiently different" queries of the *same* size. Combined with counting arguments, this ensured that the point set cannot be represented in linear space while guaranteeing a certain proximity (on disk) of the points reported by each query. The problems we study in this paper allow linear-space solutions in the I/O model, as well as linear-space cache-oblivious solutions for queries of any fixed output size. This means that previous techniques are ineffective for our purposes. In order to force a given point set to be hard for the problems we study, we construct many queries of *different* sizes. Combined with the multi-level nature of the cache-oblivious model, this allows us to create many incompatible proximity requirements for subsets of the point set and thereby force substantial duplication. It should be noted here that our construction of a hard point set (as well as the proof of Lemma 3) is inspired by a similar construction of a hard point set used by Afshani and Chan to prove a lower bound on the shallow partition theorem [2].

2. A LOWER BOUND FOR THREE-SIDED RANGE REPORTING

In this section, we present the main result of our paper: a lower bound on the space used by any cache-oblivious data structure that supports three-sided range reporting queries using $O((\log_B N)^c(1+K/B))$ block transfers. This result is summarized in the following theorem. The lower bounds for 3-d dominance reporting and 3-d halfspace range reporting are discussed in Section 3 and are obtained using reductions from three-sided range reporting to those problems.

THEOREM 1. *Any cache-oblivious data structure for three-sided range reporting that achieves a query bound of at most $a(\log_B N)^c(1+K/B)$ block transfers on a memory hierarchy with block sizes up to $N^{2\delta}$, for a constant $0 < \delta \leq 1/2$, must use $\Omega(N(\log\log N)^{\varepsilon})$ space, where $\varepsilon = \delta^c/(3a)$.*

Since we can prove this lower bound only by considering block sizes up to $N^{2\delta}$, the lower bound can be circumvented by using a sufficiently strong tall cache assumption that allows the entire point set to fit in memory ($M = \omega(B^{1/(2\delta)})$, for all $0 < \delta \leq 1/2$). It is reasonable, however, to assume that M is polynomial in B, in which case the stated lower bound holds.

Just as the lower bound proof for 4-sided range searching in the I/O model by Arge et al. [10], our proof ignores the cost of locating the points to be reported and proves that the stated space bound is necessary even to ensure only that the points to be reported by a query are stored in at most $a(\log_B N)^c(1+K/B)$ blocks.

The first step towards proving Theorem 1 is to show that it suffices to consider the case $\delta = 1/2$, that is, blocks that may have size up to N.

LEMMA 1. *If Theorem 1 holds for $\delta = 1/2$, it holds for any $0 < \delta \leq 1/2$.*

PROOF. Consider a particular choice of N, a, c, and δ in Theorem 1, and let $N' = N^{2\delta}$ and $a' = a/(2\delta)^c$. Since Theorem 1 holds for $\delta = 1/2$, there exist a point set S' of size N' and a query set Q' over S' such that a structure storing S' and capable of answering the queries in Q' using $a'(\log_B N')^c(1+K/B)$ block transfers, for block sizes up to N', must use $\Omega(N'(\log\log N')^{\varepsilon})$ space for $\varepsilon = 1/(3a' \cdot 2^c)$.

Now we construct a point set S of size N by making $m = N/N'$ copies $S_1, S_2, \ldots, S_m$ of S', which are placed side by side. We also construct a query set Q, which is the union of query sets $Q_1, Q_2, \ldots, Q_m$, where Q_i is a copy of query set Q' over the copy S_i of S'. By the argument in the previous paragraph, if we can answer the queries in Q_i over point set S_i using $a(\log_B N)^c(1+K/B) = a'(\log_B N')^c(1+K/B)$ block transfers, for block sizes up to $N' = N^{2\delta}$, the points in S_i must occupy $\Omega(N'(\log\log N')^{\varepsilon}) = \Omega(N'(\log\log N)^{\varepsilon})$ space in the data structure. Therefore, since we have $m = N/N'$ copies S_i, the space occupied by the entire set S is $\Omega(N(\log\log N)^{\varepsilon})$, for $\varepsilon = 1/(3a' \cdot 2^c) = \delta^c/3a$. □

By Lemma 1, it suffices to prove Theorem 1 for arbitrarily large block sizes. Before we present the details of the proof, we give an intuitive description of the main ideas. We recursively construct a point set S and a set Q of queries over this point set; see Figure 1(a). At the first level of recursion, we divide the plane into a $t \times 2^{t-1}$-grid T, for a parameter t to be chosen later, and place different numbers of points into its cells. The points within each cell are arranged by dividing the cell into subcells and distributing the points over these subcells. This process continues recursively as long as each grid cell contains at least $\sqrt{N}$ points.

The construction of the query set Q follows the recursive construction of S. Each query at the top level comprises a union of grid cells chosen so that each top-level query outputs roughly the same number of points. When recursing on a grid cell, we construct a set of queries over the subgrid in this cell in a similar fashion. The main idea is to prove that, by choosing the queries in Q appropriately, we can force that there exists at least one grid cell at the top level of recursion at least half of whose points are duplicated $\Omega(t^{\varepsilon})$ times. The next level of recursion ensures that again at least one of the subcells in each of the remaining cells has at least half of its points duplicated $\Omega(t^{\varepsilon})$ times, and so on. By making the recursion sufficiently deep, we can ensure that at least a constant fraction of the points are duplicated $\Omega(t^{\varepsilon})$ times. Thus, by choosing $t \approx \log\log N$, we obtain the claimed lower bound.

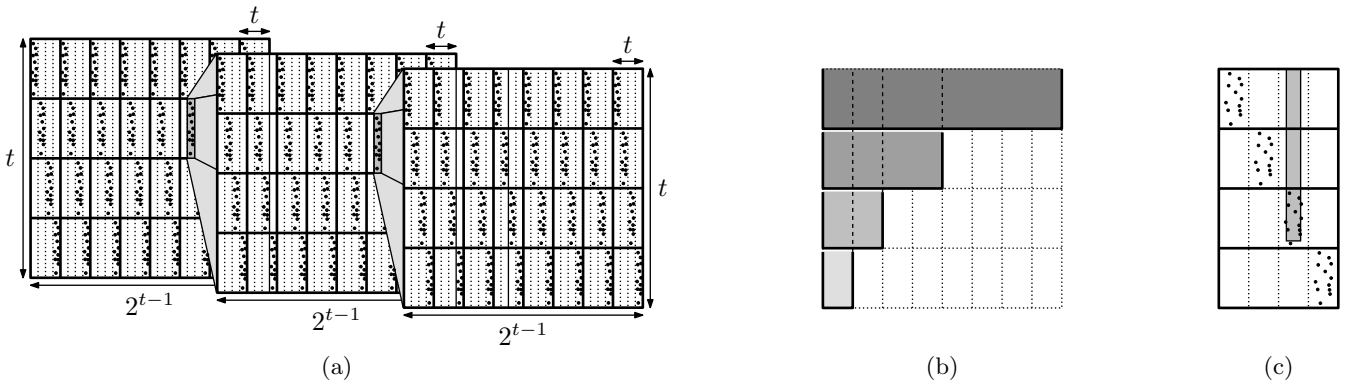

Figure 1: (a) The recursive construction of the point set. Fat solid lines bound grid cells, dotted lines separate subcolumns. (b) The set of queries in Q_T. Only one query is shown for each level of Q_T. (c) Queries at recursive levels output only points from their subgrids. This figure shows only one grid column, stretched horizontally to better show the query.

The key to forcing this type of duplication at each level of recursion is to exploit the assumption of the cache-oblivious model that the data structure has to be able to cope with any block size. This allows us to choose the block size at each level of recursion to be between $K/2$ and K and no less than $\sqrt{N}$. Any data structure capable of coping with this block size must answer the query using $a(\log_B N)^c(1 + K/B) \leq 3a/2^c = \mathrm{O}(1)$ block transfers. This completely eliminates the potential benefit of the dependence of the query bound on the size of the point set and on the output size of the query and allows us to place very stringent constraints on the layout (in memory) of the points contained in each query.

We divide the details of the proof into two parts. In Section 2.1, we discuss precisely the construction of the point set S and query set Q and prove that, if we can force the duplication claimed above in one cell of each of the subgrids at each level of recursion, Theorem 1 follows. In Section 2.2, we discuss how to achieve this duplication at each level of recursion.

2.1 The Point Set and Query Set

To define the point set S, we construct a $t \times 2^{t-1}$-grid T, for a parameter $t := (\log\log N)/4$. This parameter remains fixed throughout the construction. We refer to the grid cell in row i and column j as T_{ij}. Every column of T is divided into t subcolumns, which also splits each cell T_{ij} into t subcells T_{ijk}, for $1 \leq k \leq t$. We now place $2^{i-1}N_1$ points into each cell in row i, where $N_1 = N/(2^{2t-1} - 2^{t-1})$. The points in cell T_{ij} are placed into subcell T_{iji}. This is illustrated in Figure 1(a). Observe that this ensures that each column of the grid receives $(2^t - 1)N_1$ points. Since there are 2^{t-1} columns, the total number of points in the grid is $(2^{2t-1} - 2^{t-1})N_1 = N$. The layout of the points within each cell is now obtained by applying the same procedure recursively to the set of points assigned to each cell. The recursion stops when the smallest cell in the current grid receives at most $\sqrt{N}$ points.

The query set Q is constructed by following the recursive construction of S. For the top-level grid T, we construct a set Q_T of queries consisting of t *levels*. Level i contains 2^{i-1} queries, the kth of which is the union of all grid cells $T_{i'j'}$ satisfying $0 < i' \leq i$ and $(k-1)2^{t-i} < j' \leq k2^{t-i}$; see Figure 1(b). It is easily verified that every query in Q_T has output size $\Theta(2^t N_1) = \Theta(N/2^t)$ (between $N_1 2^{t-1}$ and $N_1(2^t - 1)$ to be precise). Note that, even though we specify these queries as unions of grid cells, that is, effectively, as four-sided queries, we can move their top boundaries to infinity without changing the set of points they report. To complete the construction of the query set Q, we apply the same construction recursively to the subgrids in each cell, adding a query set $Q_{T'}$ to Q, for each such subgrid T'. Again, we can move the top boundary of each query in $Q_{T'}$ to infinity to make it three-sided without changing the set of points it reports. Indeed, this does not change the set of points from T' reported by the query, and the staggered layout of the points in each grid column into x-disjoint subcolumns ensures that there are no points in S that belong to the x-range of T' but are outside its y-range. This is illustrated in Figure 1(c).

The following lemma now provides the framework we use to prove Theorem 1.

LEMMA 2. *Assume that the following is true for any subgrid T' of T containing N' points and its corresponding query set $Q_{T'}$:*

> *If each query in $Q_{T'}$ can be answered using at most $a(\log_B N)^c(1 + K/B)$ block transfers, for a block size $B = \Theta(N'/2^t)$, then there exists a cell C' in T' at least half of whose points are duplicated $\Omega(t^\varepsilon)$ times, for $\varepsilon = 1/(3a \cdot 2^c)$.*

Then any cache-oblivious data structure achieving a query bound of at most $a(\log_B N)^c(1 + K/B)$ block transfers for queries over S requires $\Omega(Nt^\varepsilon) = \Omega(N(\log\log N)^\varepsilon)$ space.

PROOF. Consider a subgrid T'. By setting the block size to a value $B = \Theta(N'/2^t)$ as in the lemma, we know there exists a cell C' at least half of whose points are duplicated $\Omega(t^\varepsilon)$ times. Let $|C'|$ be the number of points in C'. Then $|C'| \geq N'/4^t$. We call the points in C' *accounted for*. By recursing on the other grid cells of T', we can account for a subset of the points in those grid cells (by using a *different* block size corresponding to the numbers of points in those grid cells). We claim that, if there are at least r' levels of

recursion inside each grid cell of T', the number of points in T' that are not accounted for is at most $N'(1-4^{-t})^{r'+1}$. For $r'=0$, the claim follows because the cells of T' other than C' contain at most $N''=(1-4^{-t})N'$ points in total. For $r'>0$, we observe that, by the induction hypothesis, the r' levels of recursion inside those grid cells leave at most $(1-4^{-t})^{r'}N''=(1-4^{-t})^{r'+1}N'$ points unaccounted for.

Now, if there are at least $r:=\sqrt{\log N}$ levels of recursion inside each cell of the top level grid T, at most

$$N(1-4^{-t})^{r+1}\leq Ne^{-(r+1)/4^t}\leq N/e$$

points in T are unaccounted for. Of the $(1-e^{-1})N$ points that are accounted for, at least half are duplicated $\Omega(t^\varepsilon)$ times, which requires them to use $\Omega(Nt^\varepsilon)$ space. It remains to show that there are at least $\sqrt{\log N}$ levels of recursion in each cell of T.

Since the recursion stops when the size of the grid cells falls below $\sqrt{N}$, the number of recursion levels in each cell of T is at least

$$\log_{4^t}\frac{N}{\sqrt{N}}=\frac{(\log N)/2}{2t}=\frac{\log N}{\log\log N}>\sqrt{\log N},$$

for N sufficiently large. This completes the proof. □

2.2 Forcing Duplication in at Least One Cell

By Lemma 2, it suffices to consider the top-level grid T and prove that the queries in Q_T force it to contain at least one cell at least half of whose points are duplicated $\Omega(t^\varepsilon)$ times. The same argument then applies to any subgrid T' in the recursive construction. Note that every grid cell b in the ith row of T is contained in exactly one level-i query in Q_T; we denote this query by q_b. Further, recall that every query in Q_T has an output size between $N_1 2^{t-1}$ and $N_1(2^t-1)$. By choosing a block size of $B=N_1 2^{t-1}\geq\sqrt{N}$, we ensure that every query outputs between B and $2B$ points and, hence, must be answerable using $\alpha=3a\cdot 2^c$ block transfers.

Now consider a layout of the points in T in memory, assign a unique colour to each memory block, and define the colour set $C(p)$ of a point to be the set of colours of all blocks containing p. By the definition of the block size, for every colour γ, there are at most $B=N_1 2^{t-1}$ points $p\in T$ such that $\gamma\in C(p)$. Furthermore, since we assume that any query $q\in Q_T$ can be answered using α block transfers, there exists a set $F(q)$ of at most α colours for each query $q\in Q_T$ such that every point $p\in q$ satisfies $C(p)\cap F(q)\neq\emptyset$. We say that query q is *α-coloured* and call $F(q)$ the *fixed colour set* of query q to indicate that query q is allowed to inspect only blocks in $F(q)$ to report its output. We make no assumption about the structure of $F(q)$, only that the data structure has to decide once and for all which α blocks to query in order to answer query q.

Next our goal is to find a column of T at least one of whose cells has to contain many points that are duplicated $\Omega(t^\varepsilon)$ times. We say that a point p in a cell T_{ij} is *killed* by a cell $T_{i'j}$, $i'<i$, if $F(q_{T_{i'j}})\cap C(p)\neq\emptyset$. For a sequence of cells $T_{i_1,j},T_{i_2,j},\ldots,T_{i_r,j}$ in the same column, with $i_1<i_2<\cdots<i_r$, we say that cell $T_{i_k,j}$ is *alive* if at most half of its points are killed by cells $T_{i_1,j},T_{i_2,j},\ldots,T_{i_{k-1},j}$. Our goal is to find a long sequence of cells in the same column that are all alive. Intuitively, such a sequence of alive cells has the property that the blocks accessed to answer queries including only the top cells contain at most half of the points in the bottom cells. Thus, a query including the bottom cells needs to access new blocks in order to retrieve the contents of those cells. To keep such a query α-colourable, however, it also needs to copy the contents of the top cells into those blocks, thus causing the points in the top cells to be duplicated. First we show that we can always find a sequence of $\Omega(t)$ alive cells. Then we prove that at least half of the points in at least one cell of this sequence have to be duplicated $\Omega(t^{1/\alpha})$ times.

LEMMA 3. *There exists a column in T containing a sequence of $\Omega(t)$ alive cells.*

PROOF. Let h be a constant to be chosen later, and consider the subgrid T_h of T consisting of the rows with numbers $1,h+1,2h+1,\ldots$. First we bound the number of points in row $ih+1$ killed by cells in rows $jh+1$, for $0\leq j<i$. Level $jh+1$ of the query set Q_T contains 2^{jh} queries. Hence, the total number of queries at levels $jh+1$ with $0\leq j<i$ is $\sum_{j=0}^{i-1}2^{jh}<2^{(i-1)h+1}$. The fixed colour set of each such query contains at most α colours, and there are at most $N_1 2^{t-1}$ points with the same colour. Hence, less than $X:=\alpha N_1 2^{t-1}2^{(i-1)h+1}$ points have a colour belonging to at least one of the fixed colour sets of these queries. As each cell in row $ih+1$ contains $N_1 2^{ih}$ points, this means that less than $X/(N_1 2^{ih}/2)=2\alpha 2^{t-h}$ cells are killed in row $ih+1$. There are 2^{t-1} cells in row $ih+1$. Thus, less than a $4\alpha/2^h$ fraction of the cells in row $ih+1$ are killed by rows $jh+1$ with $0\leq j<i$.

Now choose $h=\lceil\log(8\alpha)\rceil$. Then at least half of the cells in each row of T_h are alive. This implies that T_h contains at least one column at least half of whose cells are alive. Since T_h has $\lfloor t/h\rfloor=\Omega(t)$ rows, this column —and, hence, the corresponding column of T—contains a sequence of $\Omega(t)$ alive cells. □

Now observe that at least half of the points in each alive cell in a sequence as in Lemma 3 are alive. Thus, to prove our claim that there exists one cell in T at least half of whose points are duplicated $\Omega(t^{1/\alpha})$ times, it suffices to prove that *all alive points* in at least one of the cells of this sequence are duplicated that often. We do exactly this to prove the following lemma.

LEMMA 4. *Grid T contains at least one cell at least half of whose points are duplicated $\Omega(t^{1/\alpha})$ times.*

PROOF. Consider a column C of T with a sequence of $r=\Omega(t)$ alive cells $b_1,b_2,\ldots,b_r$ in the same column. By Lemma 3, such a sequence exists. For $1\leq i\leq r$, let p_i be an alive point in b_i that has the minimum number of colours among all alive points in b_i. The sequence of points $p_1,p_2,\ldots,p_r$ has the following properties:

(i) For all $1\leq i\leq r$, there exists a query $q_i=q_{b_i}$ containing points $p_1,p_2,\ldots,p_i$, but not $p_{i+1},p_{i+2},\ldots,p_r$.

(ii) For all $1\leq j<i\leq r$, $C(p_i)\cap F(q_j)=\emptyset$. This is true because point p_i is alive.

Next we prove a lower bound on the size of the colour set $C(p_i)$ of the point p_i with the greatest number of colours among points $p_1,p_2,\ldots,p_r$. Let $\ell(\alpha,f)$ be the length r of the shortest sequence that forces at least one of the points $p_1,p_2,\ldots,p_r$ to have at least f colours if each of the queries $q_1,q_2,\ldots,q_r$ is α-coloured. We prove that $\ell(\alpha,f)\leq f^\alpha$.

Since $r = \Omega(t)$, this implies that some point p_i has $\Omega(t^{1/\alpha})$ colours, that is, is duplicated $\Omega(t^{1/\alpha})$ times. However, p_i is the point in b_i with minimum duplication among all alive points in b_i, and at least half of the points in b_i are alive. Thus, at least half of the points in b_i are duplicated $\Omega(t^{1/\alpha})$ times. It remains to prove that $\ell(\alpha, f) \leq f^\alpha$.

Our central claim is that $\ell(\alpha, f)$ satisfies the following recurrence relation.

$$\ell(\alpha, f) \leq \begin{cases} 1 & f = 1 \\ f & \alpha = 1 \\ \ell(\alpha, f-1) + \ell(\alpha-1, f) & \text{otherwise} \end{cases}$$

The two base cases are fairly obvious: A sequence of length one suffices to ensure that there is a point with at least one colour. If $\alpha = 1$, all points in a query q_i must have the same colour. Since p_1 is contained in all queries $q_1, q_2, \ldots, q_r$, a sequence of length r forces p_1 to have r colours. Hence, $\ell(1, f) = f$. For the inductive step ($\alpha > 1$ and $f > 1$), consider the longest possible point sequence $p_1, p_2, \ldots, p_r$ and the corresponding query sequence $q_1, q_2, \ldots, q_r$ such that we can α-colour each of the queries, while assigning at most $f-1$ colours to each point. If $r \leq \ell(\alpha, f-1)$, we are done. So assume that $r > \ell(\alpha, f-1)$, and let $r' = \ell(\alpha, f-1)$. α-Colouring each of the queries $q_1, q_2, \ldots, q_{r'}$ forces at least one point p_i with $1 \leq i \leq r'$ to have at least $f-1$ colours from $\bigcup_{j=1}^{r'} F(q_j)$. For each query q_j with $r' < j \leq r$, let q'_j be the restriction of q_j to points $p_{r'+1}, p_{r'+2}, \ldots, p_r$. Since query q_j is α-coloured and contains point p_i, and none of the colours in $C(p_i) \subseteq \bigcup_{j=1}^{r'} F(q_j)$ belongs to $C(p_{i'})$, for any $i' > r'$, query q'_j is $(\alpha-1)$-coloured. Since each of the points $p_{r'+1}, p_{r'+2}, \ldots, p_r$ receives at most $f-1$ colours, this implies that $r - r' < \ell(\alpha-1, f)$ and $r < \ell(\alpha, f-1) + \ell(\alpha-1, f)$. In other words, in a list of length at least $\ell(\alpha, f-1) + \ell(\alpha-1, f)$, α-colouring all the queries forces at least one point to have at least f colours.

It is not difficult to show that the above recurrence is in fact an equality that solves to $\ell(\alpha, f) = \binom{f+\alpha-1}{\alpha} \leq f^\alpha$. We refer the reader to Appendix A, where we prove this. □

Lemmas 2 and 4 together imply Theorem 1 for $\delta = 1/2$, $\varepsilon = 1/\alpha$, and $\alpha = 3a \cdot 2^c$. By Lemma 1, this implies Theorem 1 for any $0 < \delta \leq 1/2$. In Appendix B, we show that the analysis obtained in this section is the best possible for the point set S and query set Q we have defined. Thus, in order to prove a stronger lower bound than the one stated in Theorem 1, it would be necessary to construct a harder point set or set of queries.

3. FURTHER LOWER BOUNDS

In this section, we show that the proof technique from Section 2 can be used to obtain the same lower bound as in Theorem 1 for other range searching problems and to obtain a lower bound on the space consumption of a cache-oblivious persistent B-tree with worst-case optimal 1-d range queries.

3.1 3-D Dominance Reporting and Persistent B-Trees

The following theorem establishes the same lower bound as in Theorem 1 for 3-d dominance reporting. The proof technique is a simple reduction that shows that any data structure that supports 3-d dominance reporting queries can be used to answer three-sided range reporting queries without altering the query or space bound.

THEOREM 2. *Any cache-oblivious data structure for 3-d dominance reporting that achieves a query bound of at most $a(\log_B N)^c(1+K/B)$ block transfers on a memory hierarchy with block sizes up to $N^{2\delta}$, for a constant $0 < \delta \leq 1/2$, must use $\Omega(N(\log\log N)^\varepsilon)$ space, where $\varepsilon = \delta^c/(3a)$.*

PROOF. A simple geometric transformation reduces three-sided range queries to 3-d dominance queries: map an input point $p = (x_p, y_p) \in S$ to the point $\phi(p) = (-x_p, x_p, -y_p)$ in $\mathbb{R}^3$. Then a point p belongs to the three-sided query range $q = [l, r] \times [b, +\infty)$ if and only if $\phi(p)$ is dominated by the point $\phi(q) = (-l, r, -b)$. □

A similar reduction shows the following result.

THEOREM 3. *Consider a sequence of N updates on a (partially) persistent cache-oblivious B-tree. If the tree supports 1-d range queries on any previous version of the tree using at most $a(\log_B N)^c(1 + K/B)$ block transfers in the worst case, for block sizes up to $N^{2\delta}$, where $0 < \delta \leq 1/2$, then there exists an update sequence that forces the tree to use $\Omega(N(\log\log N)^\varepsilon)$ space to represent all N versions of the tree, where $\varepsilon = \delta^c/(3a)$.*

PROOF. Consider a persistent cache-oblivious B-tree T that uses $S(N)$ space to represent the versions of T produced by a sequence of N update operations and supports 1-d range queries on any version of the tree using at most $a(\log_B N)^c(1 + K/B)$ block transfers. Using T, we can obtain a three-sided range reporting structure with query bound at most $a(\log_B N)^c(1 + K/B)$: we insert the points one by one into the tree, in order of decreasing y-coordinates. To answer a three-sided range reporting query $q = [l, r] \times [b, +\infty)$, we ask a 1-d range query with query range $[l, r]$ on the version of T that was current at y-coordinate b. By combining this observation with Theorem 1, we obtain the claimed space lower bound. □

3.2 Halfspace Range Reporting in Three Dimensions

Our final result extends the technique from Section 2 to 3-d halfspace range reporting. In this case, we are not able to obtain a general geometric transformation that provides a reduction from three-sided range reporting in the plane to halfspace range reporting. We can, however, distort the point set in the proof of Theorem 1 so that each query in the query set Q we constructed in Section 2 can be replaced with a parabolic range query that outputs the exact same set of points. Since parabolic range queries can be reduced to halfspace range queries (see, e.g., [2]), this proves the following theorem.

THEOREM 4. *Any cache-oblivious data structure for 3-d halfspace range reporting that achieves a query bound of at most $a(\log_B N)^c(1+K/B)$ block transfers on a memory hierarchy with block sizes up to $N^{2\delta}$, for a constant $0 < \delta \leq 1/2$, must use $\Omega(N(\log\log N)^\varepsilon)$ space, where $\varepsilon = \delta^c/(3a)$.*

It suffices again to consider the case $\delta = 1/2$, as the general case, $0 < \delta \leq 1/2$, then follows from Lemma 1.

Consider the grid T, the set of queries Q_T, and the related notation used in Section 2. The basic idea of our construction is as follows. For every subgrid T' in our construction, we want to replace the three-sided range queries with

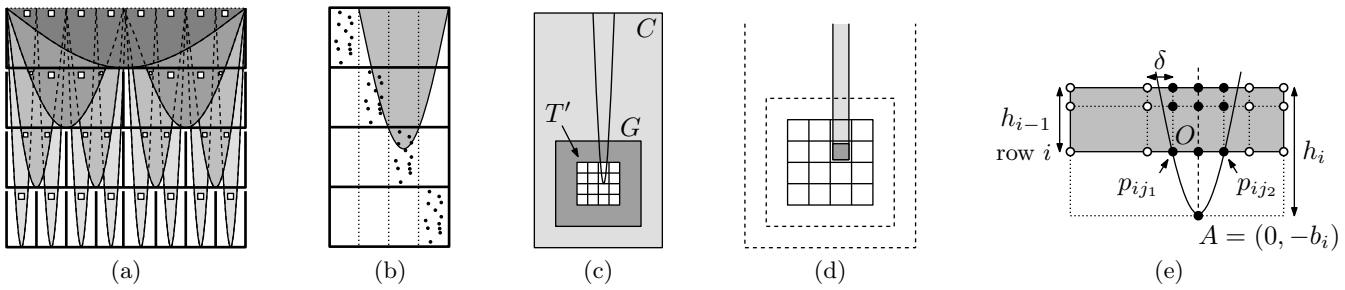

Figure 2: (a) Replacing three-sided queries with parabolic ones. The white squares are the areas where the subgrids in each grid cell are to be placed. (b) A naively constructed query in the subgrid in cell $T_{3,6}$ also reports points in other cells (e.g., $T_{2,6}$). Only the 6th column is shown. (c) Placement of a subgrid within a grid box G that is nested inside the column box C. (d) Recursive definition of grid and column boxes for the subgrids in the cells of a grid. (e) Incremental embedding of a subgrid T' inside a grid box.

parabolic ones as shown in Figure 2(a). This is easy to do at any given level of recursion. Recall, however, that every column of T' is divided into subcolumns, and every three-sided range query over a given subgrid T'' in a cell of T' has to stay within the subcolumn of T' containing T'', in order to prevent it from reporting points outside of T''; see Figure 1(c). A naive construction of the parabolas does not have this property, as illustrated in Figure 2(b). Instead, we use the following construction, which extends a construction from [2].

For every subgrid T' we have to construct, we are given two boxes C and G such that $G \subseteq C$. We call C the *column box* of T' and G the *grid box* of T'. Our goal is to place T' inside G and define parabolic range queries over T' that output the same set of points from T' as the three-sided range queries in $Q_{T'}$ and do not intersect the boundary of C, except at the top edge; see Figure 2(c). Intuitively, once we have constructed a grid T', each of the subgrids T'' in its cells is to be placed into the appropriate subcolumn in this cell, which is represented by the grid box of T''. Each query in $Q_{T''}$ has to stay within the subcolumn containing T'', which is represented by the column box. The top boundary of each column box is the top boundary of the top-level grid T. A query will in fact leave its allocated column somewhere above this top boundary of T, but this does not add any output points to the query, as all points are placed within T.

This discussion implies that Theorem 4 follows if we can prove that, given any column box C and grid box G with $G \subseteq C$, we can embed T' inside G while ensuring that the parabolic queries constructed over T' intersect only the top edge of C. Indeed, once such an embedding has been obtained, we can easily define appropriate grid and column boxes for the subgrids to be placed into the cells of T', as shown in Figure 2(d).

LEMMA 5. *Given a column box C and a grid box $G \subseteq C$, a subgrid T' can be embedded inside G so that, for every three-sided range query $q \in Q_{T'}$, there exists a parabolic range query q' that reports the same set of points as q and intersects only the top boundary of C.*

PROOF. For the sake of this proof, we can assume that each grid cell in T' contains exactly one point, as we can choose to embed the subgrid represented by each such point in a sufficiently small neighbourhood of the point without altering the properties of the construction. We construct T' row by row, placing the points in each row at the same y-coordinate and spacing them evenly in x-direction. More precisely, we choose the width of each column of T to be some parameter δ to be chosen later, and we place the points in each row in the centre of their respective columns. When constructing row i, we assume that the points placed in rows 1 through $i-1$ have been placed into a box of width $\delta 2^{t-1}$ and height h_{i-1}. For $i = 1$, this box has height $h_0 = 0$, that is, is a line. We evenly distribute the points in row i along the bottom edge of this box, as shown in Figure 2(e).

Next we construct the queries at the ith level of the query set Q_T. If we denote the jth point in row i by p_{ij}, each such query q has to output points $p_{i'j'}$ with $1 \le i' \le i$ and $(k-1)2^{t-i} < j' \le k2^{t-i}$, for some $0 < k \le 2^{i-1}$. To construct q, let $j_1 = (k-1)2^{t-i} + 1$ and $j_2 = k2^{t-i}$. We construct a parabola through points p_{ij_1} and p_{ij_2} and with a sufficiently low apex A that it contains only those points in rows 1 through i whose x-coordinates are between those of p_{ij_1} and p_{ij_2} (the black points in Figure 2(e)) and that it intersects only the top boundary of C. Let b_i be the distance of A from row i. We construct such a parabola for every query at level i in Q_T and define the bottom boundary of the box containing rows 1 through i to be infinitesimally below the apexes of these queries, that is, $h_i = h_{i-1} + b_i + \varepsilon$. This ensures that none of the level-i queries outputs a point in a row below row i. By the end of the construction, we have placed T into a box of size $\delta 2^{t-1} \times h_t$. We have to show that, by making δ small enough, we can ensure that this box fits inside G.

Consider a level-i query q and, to ease exposition, let us assume that row i coincides with the line $y = 0$ and the apex A of q is on the line $x = 0$. Let d_i denote half the distance between points p_{ij_1} and p_{ij_2} defined above. Then $d_i = \delta 2^{t-i-1}$. Since the apex $A = (0, -b_i)$ of parabola q belongs to the line $x = 0$, q can be described by an equation of the form $y = a_i x^2 - b_i$. Since q passes through points p_{ij_1} and p_{ij_2}, we have $b_i = a_i d_i^2$. To ensure that no point in rows 1 through $i-1$ outside the x-range of points p_{ij_1} and p_{ij_2} belongs to q, we must have

$$b_i + h_{i-1} < a_i(d_i + \delta)^2,$$

as the distance between two points in adjacent columns is δ and all points in rows 1 through $i-1$ are at most $b_i + h_{i-1}$

above the apex of q. Substituting $b_i = a_i d_i^2$, this gives

$$a_i > h_{i-1}/(2d_i\delta + \delta^2).$$

By choosing $a_i = a + h_{i-1}/(2d_i\delta + \delta^2)$, for a sufficiently large parameter a that depends on the ratio between the width of G and the height of C, we simultaneously satisfy this constraint on a_i and ensure that q becomes a thin sliver that intersects only the top boundary of C. By substituting this into the equation for b_i and replacing d_i with $\delta 2^{t-i-1}$, we obtain that $b_i = a\delta^2 4^{t-i-1} + 4^{t-i-1}h_{i-1}/(1 + 2^{t-i}) < a\delta^2 4^{t-i-1} + 2^{t-i}h_{i-1}$. A choice of $\delta = \varepsilon/(\sqrt{a}2^{t-1})$ yields $b_i < \varepsilon + 2^{t-i}h_{i-1}$ and, thus, $h_i < 2\varepsilon + (2^{t-i} + 1)h_{i-1}$. From this, we obtain $h_i < \varepsilon 2^{it+1}$. In particular, $h_t < \varepsilon 2^{t^2+1}$. By choosing $\varepsilon \leq h/2^{t^2+1}$, where h is the height of G, the grid thus fits into G. (Of course, we also have to ensure that it is not wider than G, but that is trivial.) □

4. CONCLUSIONS

In this paper, we have provided another separation result between the cache-oblivious model and the I/O model by proving an $\Omega((\log\log N)^{\varepsilon})$ gap between the space bounds of range reporting data structures with optimal query bounds in the two models. While previous separation results had been obtained (using considerable technical difficulties and sophisticated techniques), our result is the first one that proves a gap that grows with the input.

Nevertheless, our lower bound is far from the $O(N \log N)$ space bound required by the currently best cache-oblivious structures for the problems we considered. As our analysis in Appendix B shows, the result we obtained is in fact the best possible with the point and query sets we considered. Hence, better lower bounds are possible only through construction of a different point set or query set. On the other hand, it is worthwhile to investigate whether the $O(N \log N)$ space bound of the best known data structures for the problems we studied can be lowered; in particular, an interesting open question is whether $O(N \log^{\varepsilon} N)$ space suffices to achieve the optimal query bound.

5. REFERENCES

[1] P. Afshani. On dominance reporting in 3D. In *Proceedings of the 16th European Symposium on Algorithms*, volume 5193 of *Lecture Notes in Computer Science*, pages 41–51. Springer-Verlag, 2008.

[2] P. Afshani and T. M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 180–186, 2009.

[3] P. Afshani, C. Hamilton, and N. Zeh. A general approach for cache-oblivious range reporting and approximate range counting. In *Proceedings of the 25th ACM Symposium on Computational Geometry*, 2009.

[4] P. K. Agarwal, L. Arge, A. Danner, and B. Holland-Minkley. Cache-oblivious data structures for orthogonal range searching. In *Proceedings of the 19th ACM Symposium on Computational Geometry*, pages 237–245, 2003.

[5] P. K. Agarwal, L. Arge, J. Erickson, P. G. Franciosa, and J. S. Vitter. Efficient searching with linear constraints. *Journal of Computer and System Sciences*, 61:194–216, 2000.

[6] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.

[7] L. Arge, G. S. Brodal, R. Fagerberg, and M. Laustsen. Cache-oblivious planar orthogonal range searching and counting. In *Proceedings of the 21st ACM Symposium on Computational Geometry*, pages 160–169, 2005.

[8] L. Arge, M. de Berg, and H. J. Haverkort. Cache-oblivious R-trees. In *Proceedings of the 21st ACM Symposium on Computational Geometry*, pages 170–179, 2005.

[9] L. Arge, M. de Berg, H. J. Haverkort, and K. Yi. The priority R-tree: A practically efficient and worst-case optimal R-tree. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 347–358, 2004.

[10] L. Arge, V. Samoladas, and J. S. Vitter. On two-dimensional indexability and optimal range search indexing. In *Proceedings of the 18th Symposium on Principles of Database Systems*, pages 346–357, 1999.

[11] L. Arge and N. Zeh. Simple and semi-dynamic structures for cache-oblivious orthogonal range searching. In *Proceedings of the 22nd ACM Symposium on Computational Geometry*, pages 158–166, 2006.

[12] M. A. Bender, G. S. Brodal, R. Fagerberg, D. Ge, S. He, H. Hu, J. Iacono, and A. López-Ortiz. The cost of cache-oblivious searching. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 271–282, 2003.

[13] G. S. Brodal and R. Fagerberg. On the limits of cache-obliviousness. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 307–315, 2003.

[14] T. M. Chan. Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 586–595, 1998.

[15] T. M. Chan. Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000.

[16] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 285–397, 1999.

[17] R. Grossi and G. F. Italiano. Efficient cross-tree for external memory. In J. Abello and J. S. Vitter, editors, *External Memory Algorithms and Visualization*, pages 87–106. American Mathematical Society, 1999.

[18] R. Grossi and G. F. Italiano. Efficient splitting and merging algorithms for order decomposable problems. *Information and Computation*, 154(1):1–33, 1999.

[19] K. V. R. Kanth and A. K. Singh. Optimal dynamic range searching in non-replicated index structures. In *Proceedings of the International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 257–276. Springer-Verlag, 1999.

[20] C. Makris and A. Tsakalidis. Algorithms for three-dimensional dominance searching in linear space. *Information Processing Letters*, 66(6):277–283, 1998.

[21] E. M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–76, 1985.

[22] O. Procopiuc, P. K. Agarwal, L. Arge, and J. S. Vitter. Bkd-tree: A dynamic scalable kd-tree. In *Proceedings of the 8th International Symposium on Advances in Spatial and Temporal Databases*, volume 2750 of *Lecture Notes in Computer Science*, pages 46–65. Springer-Verlag, 2003.

[23] E. A. Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the 15th ACM Symposium on Computational Geometry*, pages 390–399, 1999.

[24] J. Robinson. The K-D-B tree: A search structure for large dimensional dynamic indexes. In *Proceedings of the SIGMOD International Conference on Management of Data*, pages 10–18, 1981.

[25] D. E. Vengroff and J. S. Vitter. Efficient 3-D range searching in external memory. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 192–201, 1996.

[26] J. S. Vitter. External memory algorithms and data structures: dealing with massive data. *ACM Computing Surveys*, 33(2), 2001. Updated version at `http://www.cs.purdue.edu/~jsv/Papers/Vit.IO_survey.pdf`

APPENDIX

A. A CLOSED FORM FOR $\ell(\alpha, f)$

In this appendix, we obtain a closed form for the minimum length $\ell(\alpha, f)$ of any point sequence $p_1, p_2, \ldots, p_r$ with corresponding queries $q_1, q_2, \ldots, q_r$ as in Lemma 4 such that at least one point p_i has to have f colours if each query q_j is to be α-coloured.

To this end, given a sequence $p_1, p_2, \ldots, p_r$ of points and a sequence of queries $q_1, q_2, \ldots, q_r$ such that query q_i reports points $p_1, p_2, \ldots, p_i$, we consider an assignment of colour sets $C(p_i)$ and $F(q_j)$ to the points and queries. We say the assignment is *valid* if

(i) For every query q_j and every point p_i with $1 \leq i \leq j$, we have $C(p_i) \cap F(q_j) \neq \emptyset$ and

(ii) For every query q_j and every point p_i with $i > j$, we have $C(p_i) \cap F(q_j) = \emptyset$.

This just reflects the conditions we imposed on the colour sets of alive points in Lemma 4. We call this assignment of colours to points and queries an (α, f)-*colouring* if the maximal number of colours assigned to any point is f and the maximal number of colours assigned to any query is α. We say a point sequence is (α, f)-*colourable* if there exists a valid (α, f)-colouring of the point sequence and its corresponding query sequence. The shortest sequence such that α-colouring all queries forces at least one point to have f colours is the shortest sequence that is (α, f)-colourable but not $(\alpha, f-1)$-colourable. The next lemma provides a closed form for the length $\ell(\alpha, f)$ of such a sequence.

LEMMA 6. *The minimum length of any point sequence that is (α, f)-colourable but not $(\alpha, f-1)$-colourable is*

$$\ell(\alpha, f) = \binom{f+\alpha-1}{\alpha}.$$

PROOF. Let $L(\alpha, f)$ be the *maximum* length of an (α, f)-colourable sequence. Consider a point sequence $p_1, p_2, \ldots, p_r$ of length $r = L(\alpha, f-1)$. We may add a new point p_{r+1} to the end of this sequence and give it a new unique colour γ. By adding this colour to the colour set of each of the points $p_1, p_2, \ldots, p_r$ and setting $F(q_{r+1}) = \{\gamma\}$, we obtain a valid (α, f)-colouring of the sequence $p_1, p_2, \ldots, p_{r+1}$ and its corresponding query sequence. Now consider appending to this a sequence $p_{r+2}, p_{r+3}, \ldots, p_s$ of length $L(\alpha-1, f)$ such that the colour sets assigned to the points and queries in this sequence are pairwise disjoint from the colour sets assigned to the points and queries in the original sequence. This sequence by itself is $(\alpha-1, f)$-colourable. By adding the colour γ to the colour set of each query q_j with $r+2 \leq j \leq s$, we obtain a valid (α, f)-colouring of the concatenation of the two sequence. This proves that $L(\alpha, f) \geq L(\alpha, f-1) + L(\alpha-1, f) + 1$.

Adding a single point to the end of a sequence of length $L(\alpha, f)$ forces at least one point in the sequence to have $f+1$ colours if every query is to receive at most α colours. Hence, $\ell(\alpha, f+1) = L(\alpha, f) + 1$. Substituting this into the above inequality yields

$$\begin{aligned} \ell(\alpha, f) - 1 &\geq \ell(\alpha, f-1) + \ell(\alpha-1, f) - 2 + 1 \\ \ell(\alpha, f) &\geq \ell(\alpha, f-1) + \ell(\alpha-1, f). \end{aligned}$$

Combining this with the inequality for $\ell(\alpha, f)$ from Lemma 4 yields the recurrence relation $\ell(\alpha, f) = \ell(\alpha, f-1) + \ell(\alpha-1, f)$. Substituting the closed form from our hypothesis into this recurrence relation yields

$$\begin{aligned} \ell(\alpha, f) &= \ell(\alpha, f-1) + \ell(\alpha-1, f) \\ &= \binom{f+\alpha-2}{\alpha} + \binom{f+\alpha-2}{\alpha-1} = \binom{f+\alpha-1}{\alpha}. \end{aligned}$$

The two base cases of $\ell(1, f) = f$ and $\ell(\alpha, 1) = 1$ are easily verified as well. Thus, the closed form is correct by induction. □

B. TIGHTNESS OF THE LOWER BOUND

In this appendix, we argue that the lower bound in Theorem 1 is tight for the point set S and query set Q we constructed. We do this by showing that there exists a layout of S that uses $O(N(\log\log N)^{\varepsilon})$ space and allows each of the queries in Q to be answered by accessing at most $\alpha + K/B$ blocks *for any* B.

Using the recurrence relation for $\ell(\alpha, f)$ from Lemma 6, we can immediately show that $\ell(\alpha, f) = \sum_{i=0}^{\alpha} \ell(i, f-1)$. We use this property to define a layout for S.

The layout. Let f be the smallest integer such that $\ell(\alpha, f+1) > t$; that is, $f = O(t^{1/\alpha}) = O((\log\log N)^{\varepsilon})$. We make f copies of S and show that, using these f copies, any query in Q with output size K can be answered by accessing at most $\alpha + K/B$ blocks. In the first copy of S, we layout the cells in the first $\ell(\alpha, f)$ rows of T in column-major order, followed by the cells of the next $\ell(\alpha-1, f)$ rows, which are also in

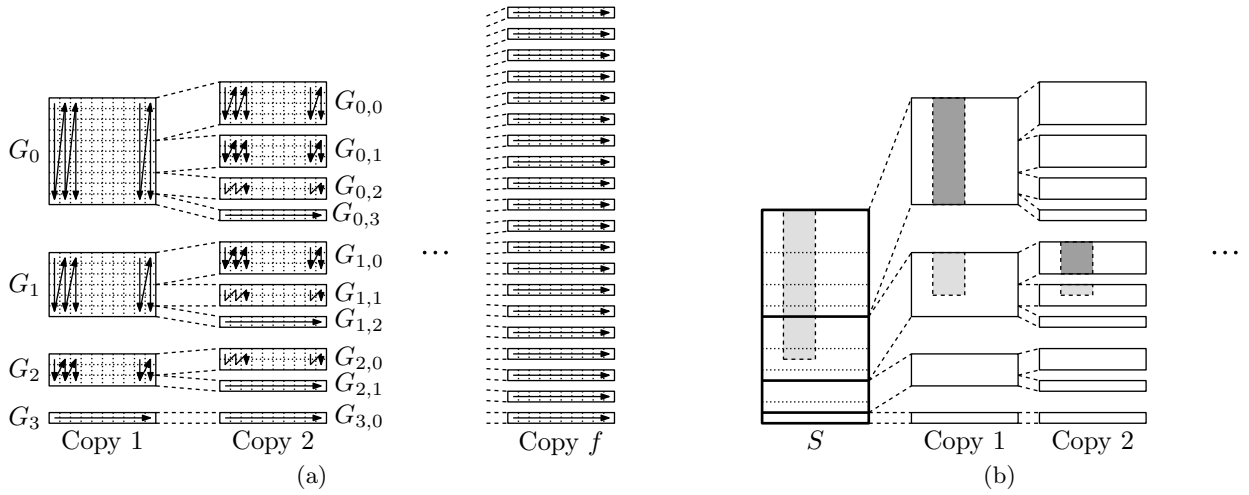

Figure 3: (a) An $O(N(\log\log N)^{\epsilon})$-space layout for point set S. (b) Answering a query on the first two levels of the layout. The dark portion of the query in each copy of S can be answered by scanning a subsequence of the column-major layout of the respective group. The light portion is answered using subsequent copies of S.

column-major order, and so on. We refer to these groups of rows as $G_0, G_1, \ldots, G_\alpha$. This layout is illustrated in Figure 3(a). The points in each cell are arranged by applying this strategy recursively. Note that, by the choice of f, these row groups cover all the rows of T.

The second copy of S now considers subgroups $G_{i,j}$ of the row groups G_i in the first copy. Group G_i contains $\ell(\alpha-i, f)$ rows, which we divide into subgroups $G_{i,0}, G_{i,1}, \ldots, G_{i,\alpha-i}$ consisting of $\ell(\alpha-i, f-1), \ell(\alpha-1-1, f-1), \ldots, \ell(0, f-1)$ rows, respectively. The cells in each row group are again laid out in column-major order; see Figure 3(a). The points in each grid cell are laid out recursively by dividing the groups of rows in the subgrids of those cells in the same fashion.

The third copy now divides the row groups of the second copy into subgroups of size $\ell(\cdot, f-2)$, and so on until the last copy has row groups of size $\ell(\cdot, 1)$, which ensures that the rows of the top-level grid and of each subgrid are stored in x-sorted order; see Figure 3(a).

Answering queries. Since this layout has the same structure for each subgrid T' in our construction of point set S, it suffices to discuss how we answer the queries in Q_T. Consider a query q at level i of Q_T. Its bottom boundary coincides with the bottom boundary of the ith row of T. Now let $0 \le j_1 \le \alpha$ be maximal such that $i_1 = \sum_{j=1}^{j_1} \ell(\alpha-j+1, f) \le i$, that is, the last row, i_1, of the first j_1 groups in the first copy of S is not below row i. The cells in each of the groups $G_0, G_1, \ldots, G_{j_1-1}$ that belong to q are stored consecutively, and we simply scan the appropriate subsequence of cells in each group to report its points. This requires $j_1 + K_1/B$ block transfers, where K_1 is the number of reported points. If $i_1 = i$, this reports all the points in q, and we are done. So assume that $i > i_1$.

In this case, we use the layout of the rows in group G_{j_1} in the second copy of S. Let $0 \le j_2 \le \alpha - j_1$ be maximal so that $i_2 = i_1 + \sum_{j=1}^{j_2} \ell(\alpha - j_1 - j + 1, f-1) \le i$, that is, the last row of the first j_2 row groups of G_{j_1} is not below row i. Then we can again scan the appropriate portion of each of these groups to report the points in these groups that belong to q, which requires $j_2 + K_2/B$ block transfers, where K_2 is the number of reported points. If $i > i_2$, we continue to the subgroups of G_{j_1,j_2} in the third copy, and so on. This procedure succeeds in reporting all points of q at the latest when reaching the fth copy of S, as each row of T is stored by itself in x-sorted order in this copy. Figure 3(b) shows the groups reported in the first two copies of S in the layout for an example query.

Assume now that our query inspects the first r copies of S before terminating. Then the query cost is $\sum_{h=1}^{r}(j_h + K_h/B) = \sum_{h=1}^{r} j_h + K/B$. It is easily verified, however, that, for all $1 < h \le r$, we have $j_h \le \alpha - \sum_{h'<h} j_{h'}$. Thus, $\sum_{h=1}^{r} j_h \le \alpha$, and the query accesses at most $\alpha + K/B$ blocks.