Cache-Oblivious Range Reporting With Optimal Queries Requires Superlinear Space

Peyman Afshani[1,*] Chris Hamilton[2,†] Norbert Zeh[2,‡]

[1] MADALGO[§], Dept. of Computer Science, Aarhus University, IT Parken, Aabogade 34
[2] Faculty of Computer Science, Dalhousie University, Halifax, NS, B3H 1W5, Canada
peyman@madalgo.au.dk, {chamilton,nzeh}@cs.dal.ca

Draft of August 24, 2009

Abstract

We consider a number of range reporting problems in two and three dimensions and prove lower bounds on the amount of space used by any cache-oblivious data structure for these problems that achieves the optimal query bound of $O(\log_B N + K/B)$ block transfers, 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 $f(\log_B N, K/B)$, for any monotonically increasing function $f(\cdot,\cdot)$, the data structure has to use $\Omega(N(\log\log N)^{\varepsilon})$ space. This lower bound holds even for the expected size of any Las-Vegas-type data structure that achieves an expected query bound of at most $f(\log_B N, K/B)$ block transfers. The exponent ε depends on the function f and on the range of permissible block sizes.

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 deterministic I/O-efficient data structures with the optimal query bound in the worst case and using linear or $O(N \log^* N)$ space are known for the above problems. The second consequence is the non-existence of linear-space cache-oblivious persistent B-trees with optimal 1-d range reporting queries.

1 Introduction

Range reporting is a well studied fundamental problem in computational geometry. Given a set S of points in $\mathbb{R}^d$, the goal is to preprocess S so that, for a query range q, all points in $S \cap q$ can be reported efficiently. Typical types of query ranges include axis-aligned boxes, circles, simplices, and halfspaces. To indicate the type of permissible queries, the problem is then referred to more specifically as orthogonal, circular, simplex or halfspace range reporting. *Three-sided range reporting* is a special case of planar orthogonal range reporting that considers axis-aligned boxes whose top boundaries are fixed at $y = +\infty$. *Dominance reporting* is another important special case of orthogonal range reporting: 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. These different query types are illustrated in Figure 1.

*Part of this work was done while visiting Dalhousie University. Work was supported in part by the Danish National Research Foundation and the Danish Strategic Research Council.

†Supported by a Killam Graduate Scholarship.

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

§Center for Massive Data Algorithmics—a center of the Danish National Research Foundation.

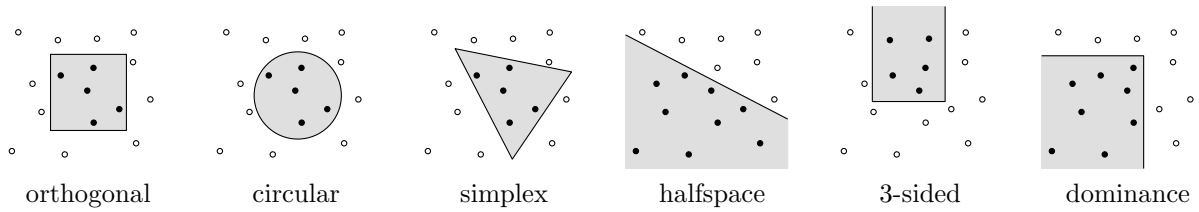


Figure 1: Standard query types.

Most previous work on this type of problem 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.).

A number of models have been proposed to model the non-uniform access costs in real memory hierarchies. See [31] for a survey. The two most widely adopted ones are the *input/output model* (or I/O model) [6] and the *cache-oblivious model* [18]. Their success is due to the balance they provide between simplicity, in order to allow the design and analysis of sophisticated algorithms, and accuracy in predicting the performance of algorithms on real memory hierarchies.

The I/O model considers two levels of memory: a fast *internal memory* with the capacity to hold M data items, and a slow but conceptually unlimited *external memory*. 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 items; the complexity of an algorithm is the number of such *block transfers* it performs.

The cache-oblivious model provides a simple framework for designing 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, to perform the minimum number of block transfers possible 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, it is simultaneously optimal at all levels of the memory hierarchy. See [18] for a more detailed discussion of this model and for a justification of the optimality assumption of the paging algorithm.

In this paper, we study three-sided range reporting, 3-d dominance reporting, and 3-d halfspace range reporting in the cache-oblivious model. We prove that any cache-oblivious data structure for these problems that achieves the optimal (or in fact a much weaker) query bound has to use asymptotically more space than a structure with the same query bound in the I/O model.

1.1 Related Work

In the I/O model, much work has focused on orthogonal range reporting. A number of linear-space data structures have been 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 higher dimensions [9, 19, 20, 22, 26, 28], where K is the number of reported points. The same bounds have been 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 [25] with a query bound of $O(\log_B N + K/B)$ for three-sided queries and using linear space.

Query type	Model	Space	Query bound	References
2-d three-sided	internal memory	N	$\log N + K$	[25]
	I/O model	N	$\log_B N + K/B$	[10]
	cache-oblivious model	$N \log N$	$\log_B N + K/B$	[3, 4, 7, 11]
3-d dominance	internal memory	N	$\log N + K$	[1, 24]
	I/O model	N	$\log_B N + K/B$	[1]
	cache-oblivious model	$N \log N$	$\log_B N + K/B$	[3]
3-d halfspace	internal memory	N	$\log N + K$	[2]
	I/O model	$N \log^* N$	$\log_B N + K/B$	[2]
	cache-oblivious model	$N \log N$	$\log_B N + K/B$	[3]

Table 1: A summary of known upper bounds for three-sided range reporting, 3-d dominance reporting and 3-d halfspace range reporting. For the sake of clarity, O-notation has been omitted.

In the cache-oblivious model, three data structures have been 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], works only if $\log \log B$ is an integer. The second and third data structures, by Arge et al. [7] and Arge and Zeh [11], remove this restriction. The data structure by Arge and Zeh is obtained using standard techniques from a linear-space data structure for optimal 2-d dominance queries proposed in the same paper.

For 3-d dominance reporting, Vengroff and Vitter [30] presented a data structure with a query bound of $O((\log \log \log_B N) \log(N/B) + K/B)$ block transfers and using $O(N \log(N/B))$ space in the I/O model. The query bound can be reduced to $O(\log_B N + K/B)$ by choosing the parameters in the data structure more carefully [31]. 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 also in the cache-oblivious model. In [3], we show that the optimal query bound can indeed be achieved by a cache-oblivious data structure, but our data structure uses $O(N \log N)$ space.

Halfspace range reporting in 3-d has a longer history, in part 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 data structure with an expected query time of $O(\log N + K)$ [16]. Building on these ideas, Agarwal et al. [5] obtained an $O(N \log N)$-space data structure with an expected query bound of $O(\log_B N + K/B)$ block transfers in the I/O model. Further research led to the development of internal-memory data structures with the optimal query bound in the worst case and using $O(N \log \log N)$ space [17, 27]. The same improvements can be carried over to the I/O model. Recently, Afshani and Chan [2] described a linear-space data structure with the optimal query bound in internal memory and an $O(N \log^* N)$-space data 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 data 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 data structures achieving the same query bound in the cache-oblivious model use $O(N \log N)$ space. This raises the question whether linear-space cache-oblivious data 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 $f(\log_B N, K/B)$, for any monotonically increasing function $f(\cdot, \cdot)$, has to use $\Omega(N(\log \log N)^{\varepsilon})$ space. This lower bound holds even for the expected size of Las-Vegas-type data structures, that is, data structures with randomized construction and query algorithms that guarantee correct query answers but achieve the desired query bound only in the expected case. The exponent ε depends on the function f and on the range of permissible block sizes.

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 reporting bound of $O(\log_B N + K/B)$ block transfers in the worst or expected case, while such a data structure exists in the I/O model [13].

There have been previous results showing that the cache-oblivious model is less powerful than the I/O model. Brodal and Fagerberg [15] established a lower bound on the amount of main memory (as a function of B) necessary for optimal cache-oblivious sorting, while Bender et al. [14] proved that cache-oblivious searching has to cost a constant factor more than the search bound achieved in the I/O model using B-trees [12]. In contrast, our result establishes a gap between the resource consumption of cache-oblivious and I/O-efficient data structures that grows with the input size.

The key to obtaining our result is the construction of a hard point set and of a set of hard queries over this point set in combination with 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 [10, 21, 23, 29] involved the construction of a hard point set together with a set 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 the 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, thus, force 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 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 deterministic cache-oblivious data structure that supports three-sided range reporting queries using $f(\log_B N, K/B)$ block transfers in the worst case. This result is summarized in the following theorem. The lower bounds for 3-d dominance reporting and 3-d halfspace range reporting are obtained from this result using reductions and are discussed in Section 4. The same section discusses how to extend the lower bound to Las-Vegas-type randomized data structures.

Theorem 1. *Let $f(\cdot,\cdot)$ be a monotonically increasing function, and $0 < \delta \leq 1/2$ a constant. Any cache-oblivious data structure capable of answering three-sided range reporting queries using at most $f(\log_B N, K/B)$ block transfers in the worst case, for every block size $B \leq N^{2\delta}$, must use $\Omega(N(\log\log N)^{\varepsilon})$ space, where $\varepsilon = 1/f(\delta^{-1}, 1)$.*

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 previous lower bounds for 4-sided range reporting in the I/O model [10, 21, 23, 29], our proof ignores the cost of locating the points to be reported by a query and shows that the above space bound is necessary even to ensure only that the points are stored in at most $f(\log_B N, K/B)$ blocks.

Our first lemma shows that it suffices to focus on the case $\delta = 1/2$—that is, to allow arbitrarily large block sizes $B \leq N$—in the proof of Theorem 1.

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

Proof. Consider a particular choice of N, f, and δ in Theorem 1, and let $N' = N^{2\delta}$ and $f'(x, y) = f(x/(2\delta), y)$. 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 data structure storing S' and capable of answering the queries in Q' using at

most $f'(\log_B N', K/B)$ block transfers, for block sizes up to N', must use $\Omega(N'(\log\log N')^\varepsilon)$ space, where $\varepsilon = 1/f'(2,1)$.

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 we place side by side. We also construct a query set Q as the union of query sets $Q_1, Q_2, \ldots, Q_m$, where Q_i is a copy of the 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 the point set S_i using $f(\log_B N, K/B) = f((\log_B N^{2\delta})/(2\delta), K/B) = f'(\log_B N', K/B)$ 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 $S_1, S_2, \ldots, S_m$, the space occupied by the entire set S is $\Omega(N(\log\log N)^\varepsilon)$, for $\varepsilon = 1/f'(2,1) = 1/f(\delta^{-1}, 1)$. □

By Lemma 1, it suffices to prove Theorem 1 for arbitrarily large block sizes. As already mentioned, the key to achieving this is to construct a point set S along with a set Q of queries over S such that any data structure capable of answering the queries in Q in the desired query bound has to use super-linear space.

The construction of the point set S is recursive; see Figure 2(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 a $t \times 2^{t-1}$-grid of subcells and distributing the points over those subcells. This process continues recursively as long as each grid cell contains more than $\sqrt{N}$ points.

The construction of the query set Q follows the recursive construction of the point set S. Each query at the top level comprises a union of cells of T chosen so that each top-level query outputs roughly the same number of points. For each grid cell, we construct a set of queries over the subgrid in this cell in a similar fashion. We repeat this construction recursively until we reach the last level in the recursive construction of the point set S.

The main idea now is to prove that, by choosing the queries in Q appropriately, we can force that there exists at least one grid cell in T at least half of whose points are duplicated $\Omega(t^\varepsilon)$ times. The next level of recursion ensures that at least one of the subcells in each of the remaining cells of T 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. (The recursion depth depends on the choice of t, and $t \approx \log\log N$ is the largest value we can choose to ensure that the recursion is sufficiently deep.)

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. In particular, it allows us to choose a block size no less than $\sqrt{N}$ and such that each query has an output size no greater than B. With this choice of parameters, every query needs to be answerable using $f(\log_B N, K/B) \leq f(2,1) = \mathrm{O}(1)$ 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 of the points contained in each query.

We divide the details of our proof into two parts. In Section 2.1, we discuss the construction of the point set S and of the query set Q more precisely 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 then 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 recursive 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 2(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

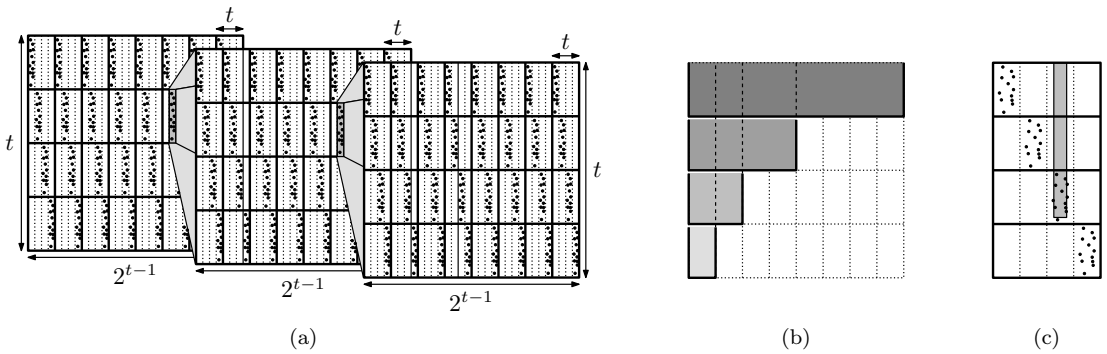


Figure 2: (a) The recursive construction of the point set. Fat solid lines bound grid cells, dotted lines separate subcolumns. (b) The set of queries in Q_T. Only one query is shown for each level of Q_T. (c) Queries at recursive levels output only points from their subgrids.

same procedure recursively to the set of points assigned to each cell. The recursion stops when the smallest cell in the current subgrid 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 2(b). It is easily verified that every query in Q_T contains between $2^{t-1}N_1$ and $(2^t - 1)N_1$ points. 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 each cell, adding a query set $Q_{T'}$ to Q, for each subgrid T' in the recursive construction of S. 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 clearly 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 2(c).

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

Lemma 2. *Assume that the following is true for every subgrid T' and its corresponding query set $Q_{T'}$:*

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

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

Proof. We construct a set of disjoint cells such that at least half of the points in each cell are duplicated $\Omega(t^\varepsilon)$ times and the total number of points in these cells is $\Omega(N)$. This proves that the points in these cells alone use $\Omega(Nt^\varepsilon)$ space.

To construct this set of cells, we apply the following recursive selection process, starting with $T' = T$. For the current grid T' storing N' points, we choose a block size of $B = N'/2^{t-1}$. (Remember that a cache-oblivious data structure must be able to cope with any block size.) Then, by (*), there exists a cell of T' at least half of whose points are duplicated $\Omega(t^\varepsilon)$ times. We choose one of these cells, C', and add it to the set of selected cells. Then we recurse on the subgrids in each of the remaining cells unless the current grid T' is already at the lowest level of recursion in the construction of the point set S.

The set of cells selected in this fashion is easily seen to be disjoint, as we recursively select cells only from subgrids in cells not selected at the current level. We call a point *selected* if it is contained in one of the selected cells. Our goal is to show that there are $\Omega(N)$ selected points in T.

The cell C' selected in a grid T' contains at least $N'/4^t$ points. This implies that at most $N'(1-4^{-t})^{r'+1}$ points from T' are not selected by the recursive selection process, where r' denotes the minimum recursion depth inside the cells of T'. Indeed, if $r' = 0$, exactly the points in C' are selected, leaving at most $N'' = N'(1-4^{-t})$ points in T' unselected. If $r' > 0$, we observe that, by induction, at most $N''(1-4^{-t})^{r'} = N'(1-4^{-t})^{r'+1}$ points in the cells of T' other than C' are not selected, while all points in C' are selected.

Since the recursion stops when the smallest cell of the current subgrid contains at most $\sqrt{N}$ points, the recursion depth inside each cell of the top-level grid T is at least

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

for N sufficiently large. Hence, the number of points in T that are not selected is at most

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

that is, at least $(1-1/e)N = \Omega(N)$ points in T are selected. 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 c in the ith row of T is contained in exactly one level-i query in Q_T; we denote this query by q_c. Moreover, 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 $B = (2^t - 1)N_1 \geq \sqrt{N}$, we ensure that every query outputs at most B points and, hence, must be answerable using $\alpha = f(2,1) = \mathrm{O}(1)$ block transfers.

So consider a layout of the points in T in memory that achieves this query bound. We represent the layout by assigning colour sets to points and queries. Every memory block is represented using a unique colour. The colour set $C(p)$ of a point p comprises the colours of those blocks that hold a copy of p. The colour set $C(q)$ of a query q comprises the colours of the blocks accessed to answer query q. Our goal now can be rephrased as showing that there exists a cell in T at least half of whose points have $\Omega(t^\varepsilon)$ colours. We do this in two steps. In Lemma 3, we show that there exists a set of $r = \Omega(t)$ cells in the same column of T such that half of the points in each cell are "exposed" in a sense defined below. Then we show that there exists at least one cell among these r cells such that every exposed point in this cell has at least $r^\varepsilon = \Omega(t^\varepsilon)$ colours.

So consider a sequence $\langle c_1, c_2, \ldots, c_r \rangle$ of cells in the same column of T and numbered from top to bottom. For $1 \leq i \leq r$, let $q_i = q_{c_i}$. We say that a query q_j *covers* a point $p \in c_i$ with $i > j$ if $C(p) \cap C(q_j) \neq \emptyset$. Point p is said to be *exposed* if none of the queries $q_1, q_2, \ldots, q_{i-1}$ covers it; cell c_i is exposed if at least half of its points are exposed.

Lemma 3. *There exists a column in T containing a sequence of $\Omega(t)$ exposed 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 $1, h+1, 2h+1, \ldots$. As each cell in row $ih+1$ contains $N_1 2^{ih}$ points, and there are 2^{t-1} cells in each row, the total number of points in row $ih+1$ is $X_i := N_1 2^{t+ih-1}$.

Next we bound the number of points in row $ih+1$ covered by queries at levels $1, h+1, \ldots, (i-1)h+1$ of the query set Q_T. Level $jh+1$ contains 2^{jh} queries. Hence, the total number of queries at levels $1, h+1, \ldots, (i-1)h+1$ is $\sum_{j=0}^{i-1} 2^{jh} < 2^{(i-1)h+1}$. The colour set of each such query contains at most α colours, and there are at most $B = N_1 2^t$ points with the same colour. Hence, at most $Y_i := \alpha N_1 2^{t+(i-1)h+1} = (4\alpha/2^h)X_i$ points have a colour belonging to the colour set of at least one of these queries and, thus, can be covered by these queries.

Now choose $h = \lceil \log(16\alpha) \rceil = \Theta(1)$. Then $Y_i \leq X_i/4$. This implies that at least half of the cells in row $ih+1$ are exposed. Since this applies to all rows in T_h, there exists a column in T_h at least half of whose cells are exposed. Since T_h has $\lfloor t/h \rfloor = \Omega(t)$ rows, this column—and, hence, the corresponding column of T—contains a sequence of $\Omega(t)$ exposed cells. □

By Lemma 3, there exists a sequence $\langle c_1, c_2, \ldots, c_r\rangle$ of $r = \Omega(t)$ exposed cells. Now, for all $1 \le i \le r$, we choose p_i to be an exposed point in cell c_i with the minimum number of colours. It suffices to show that there exists an index i such that p_i has at least $r^\varepsilon = \Omega(t^\varepsilon)$ colours, as this implies that all exposed points in c_i have at least this many colours and at least half the points in c_i are exposed.

In addition to the point sequence $\langle p_1, p_2, \ldots, p_r\rangle$, consider the query sequence $\langle q_1, q_2, \ldots, q_r\rangle$, where $q_i = q_{c_i}$, for all $1 \le i \le r$. The point-query sequence $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_r, q_r)\rangle$ has the following two properties:

(i) For $i \le j$, $C(p_i) \cap C(q_j) \neq \emptyset$ (because $p_i \in q_j$).

(ii) For $i > j$, $C(p_i) \cap C(q_j) = \emptyset$ (because p_i is exposed).

We call a colouring of the points and queries in the sequence *proper* if it satisfies these two conditions. The sequence is said to be (α, f)-*coloured* if $|C(p_i)| \le f$ and $|C(q_i)| \le \alpha$, for all $1 \le i \le r$. Finally, we say that a point-query sequence is (α, f)-*colourable* if it has a proper (α, f)-colouring.

Now let $\ell(\alpha, f)$ be the length of the shortest point-query sequence that is not $(\alpha, f-1)$-colourable, and let $L(\alpha, f)$ be the length of the longest point-query sequence that is (α, f)-colourable. Clearly, $\ell(\alpha, f) = L(\alpha, f-1)+1$. Another way to express $\ell(\alpha, f)$ is as the length of the shortest point-query sequence such that α-colouring the queries forces at least one point to have at least f colours if the colouring is to be proper. The next lemma shows that $\ell(\alpha, f) \le f^\alpha$. Thus, a properly coloured sequence of length r contains at least one point with at least $r^{1/\alpha}$ colours if all the queries are α-coloured, which is exactly what we need to prove.

Lemma 4. *For $\alpha \ge 0$ and $f \ge 1$, $\ell(\alpha, f) = \binom{\alpha+f-1}{\alpha} \le f^\alpha$.*

Proof. We prove first that $\ell(\alpha, f)$ satisfies the recurrence relation

$$\ell(\alpha, f) = \begin{cases} 1 & f = 1 \text{ or } \alpha = 0 \\ \ell(\alpha, f-1) + \ell(\alpha-1, f) & f > 1 \text{ and } \alpha > 0. \end{cases} \tag{1}$$

The base case ($f = 1$ or $\alpha = 0$) is fairly obvious: a sequence of length 1 is neither $(0, f)$-colourable, for any f, nor $(\alpha, 0)$-colourable, for any α, while a sequence of length 0 is (α, f)-colourable, for any α and f.

For the inductive step ($f > 1$ and $\alpha > 0$), we prove that $L(\alpha, f) = \ell(\alpha, f) + L(\alpha-1, f)$. This implies that

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

First we prove that $L(\alpha, f) \le \ell(\alpha, f) + L(\alpha-1, f)$. To this end, we consider a point-query sequence $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_r, q_r)\rangle$, where $r = L(\alpha, f)$. Since $L(\alpha-1, f) \ge 0$, for all α and f, $r \le \ell(\alpha, f)$ would immediately imply $r \le \ell(\alpha, f) + L(\alpha-1, f)$. Thus, we can assume that $r > r' := \ell(\alpha, f)$.

Now let C be a proper (α, f)-colouring of S. Since the restriction of the colouring C to the point-query sequence $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_{r'}, q_{r'})\rangle$ is a proper (α, f)-colouring of this sequence, the definition of $\ell(\alpha, f)$ implies that there exists a point p_k, $1 \le k \le r'$, such that $|C(p_k)| = f$ and $C(p_k) \subseteq \bigcup_{j=1}^{r'} C(q_j)$. We use this to construct a proper $(\alpha-1, f)$-colouring C' of the point-query sequence $\langle (p_{r'+1}, q_{r'+1}), (p_{r'+2}, q_{r'+2}), \ldots, (p_r, q_r)\rangle$, thereby showing that its length $r - r'$ is at most $L(\alpha-1, f)$, which implies that $L(\alpha, f) = r = r' + (r - r') \le \ell(\alpha, f) + L(\alpha-1, f)$.

To obtain such a colouring C', we define $C'(p_i) = C(p_i)$ and $C'(q_i) = C(q_i) \setminus C(p_k)$, for all $r' < i \le r$. By property (i) of C, $C(q_i) \cap C(p_k) \neq \emptyset$, for $r' < i \le r$ and, hence, $|C'(q_i)| < |C(q_i)| \le \alpha$, while $|C'(p_i)| = |C(p_i)| \le f$. Thus, C' is an $(\alpha-1, f)$-colouring of the sequence $\langle (p_{r'+1}, q_{r'+1}), (p_{r'+2}, q_{r'+2}), \ldots, (p_r, q_r)\rangle$. Next we show that C' is proper.

First observe that, for $r' < j < i \leq r$, $C'(p_i) \cap C'(q_j) \subseteq C(p_i) \cap C(q_j) = \emptyset$, by property (ii) of C. Hence, C' satisfies property (ii).

For $r' < i \leq j \leq r$, we have $C(p_i) \cap C(p_k) \subseteq C(p_i) \cap \bigcup_{h=1}^{r'} C(q_h) = \emptyset$, by property (ii) of C. Hence, $C'(p_i) \cap C'(q_j) = C(p_i) \cap (C(q_j) \setminus C(p_k)) = C(p_i) \cap C(q_j) \neq \emptyset$, by property (i) of C. Thus, C' satisfies property (i). This concludes the proof of the inequality $L(\alpha, f) \leq \ell(\alpha, f) + L(\alpha - 1, f)$.

Next we prove that $L(\alpha, f) \geq \ell(\alpha, f) + L(\alpha - 1, f)$ by proving that a sequence $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_r, q_r) \rangle$ of length $r = \ell(\alpha, f) + L(\alpha - 1, f) = L(\alpha, f - 1) + L(\alpha - 1, f) + 1$ is (α, f)-colourable.

We divide the sequence into three subsequences $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_{r_1}, q_{r_1}) \rangle$, $\langle (p_{r_1+1}, q_{r_1+1}) \rangle$, and $\langle (p_{r_1+2}, q_{r_1+2}), (p_{r_1+3}, q_{r_1+3}), \ldots, (p_r, q_r) \rangle$, where $r_1 = L(\alpha, f - 1)$ and $r_3 = r - r_1 - 1 = L(\alpha - 1, f)$. By the choice of r_1 and r_3, the sequence $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_{r_1}, q_{r_1}) \rangle$ has a proper $(\alpha, f-1)$-colouring C_1, while the sequence $\langle (p_{r_1+2}, q_{r_1+2}), (p_{r_1+3}, q_{r_1+3}), \ldots, (p_r, q_r) \rangle$ has a proper $(\alpha - 1, f)$-colouring C_3. We can assume that the two colourings use different colours. We define a colouring C of the sequence $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_r, q_r) \rangle$ as

$$C(p_i) = \begin{cases} C_1(p_i) \cup \{\gamma\} & 1 \leq i \leq r_1 \\ \{\gamma\} & i = r_1 + 1 \\ C_3(p_i) & r_1 + 2 \leq i \leq r \end{cases}$$

and

$$C(q_i) = \begin{cases} C_1(q_i) & 1 \leq i \leq r_1 \\ \{\gamma\} & i = r_1 + 1 \\ C_3(q_i) \cup \{\gamma\} & r_1 + 2 \leq i \leq r \end{cases},$$

where γ is a new colour not used by either C_1 or C_3. Since C_1 is an $(\alpha, f - 1)$-colouring of the sequence $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_{r_1}, q_{r_1}) \rangle$ and C_3 is an $(\alpha - 1, f)$-colouring of $\langle (p_{r_1+2}, q_{r_1+2}), (p_{r_1+3}, q_{r_1+3}), \ldots, (p_r, q_r) \rangle$, C is an (α, f)-colouring of $\langle (p_1, q_1), (p_2, q_2), \ldots, (p_r, q_r) \rangle$. Next we show that C is proper.

First consider property (i). For a point p_i and a query q_j with $1 \leq i \leq j \leq r$, we distinguish three cases. If $j \leq r_1$, then $C(p_i) \supset C_1(p_i)$ and $C(q_j) = C_1(q_j)$. Hence, since $C_1(p_i) \cap C_1(q_j) \neq \emptyset$ (by property (i) of the colouring C_1), we have $C(p_i) \cap C(q_j) \neq \emptyset$. If $i \leq r_1 + 1 \leq j$, we have $\gamma \in C(p_i) \cap C(q_j)$. If $r + 1 < i$, then $C(p_i) = C_3(p_i)$ and $C(q_j) \supset C_3(q_j)$. Hence, since $C_3(p_i) \cap C_3(q_j) \neq \emptyset$ (by property (i) of the colouring C_3), we have $C(p_i) \cap C(q_j) \neq \emptyset$.

Next we verify property (ii). Consider a point p_i and a query q_j with $1 \leq j < i \leq r$. If $i \leq r_1$, we have $C(p_i) \cap C(q_j) = \emptyset$ because $C_1(p_i) \cap C(q_j) = C_1(p_i) \cap C_1(q_j) = \emptyset$ (by property (ii) of the colouring C_1) and $\gamma \notin C(q_j)$. If $j \leq r_1 < i$, we have $C(q_j) \cap C(p_i) = \emptyset$ because $\gamma \notin C(q_j)$ and colourings C_1 and C_3 use different colours. Finally, if $r_1 + 1 \leq j$, then $C(p_i) \cap C(q_j) = \emptyset$ because $C(p_i) \cap C_3(q_j) = C_3(p_i) \cap C_3(q_j) = \emptyset$ (by property (ii) of the colouring C_3) and $\gamma \notin C(p_i)$.

This shows that C is a proper (α, f)-colouring of the sequence $\langle (p_i, q_i) \rangle_{i=1}^{r}$ and, hence, that $\ell(\alpha, f) + L(\alpha - 1, f) = r \leq L(\alpha, f)$.

Having established the correctness of (1), it remains to derive a closed form for $\ell(\alpha, f)$. We do this using substitution. For the base case, we have $\ell(0, f) = 1 = \binom{f-1}{0}$ and $\ell(\alpha, 1) = 1 = \binom{\alpha}{\alpha}$. For the inductive step, we obtain

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

This finishes the proof. □

To summarize, Theorem 1 is established by invoking Lemma 3 to prove that there exists a sequence of $r = \Omega(t)$ exposed cells in T. Using Lemma 4 and the discussion leading up to it, this implies that at least half of the points in one of the cells in this sequence are duplicated at least $r^\varepsilon = \Omega(t^\varepsilon)$ times, for $\varepsilon = 1/\alpha$ and $\alpha = f(2,1)$. Since the same argument can be applied recursively to every subgrid in the recursive construction of the point set S, we can now invoke Lemma 2 to prove Theorem 1 for $\delta = 1/2$. By Lemma 1, this implies Theorem 1 for any $0 < \delta \leq 1/2$.

3 Tightness of the Lower Bound

In this section, we show how to lay out the points in the set S constructed in the previous section so that the queries in Q can be answered using at most $\alpha + 1 + K/B$ block transfers, for any block size B. The layout uses $\mathrm{O}(Nt^{1/\alpha})$ space, which implies that the lower bound of Theorem 1 is tight for the point set S and query set Q we constructed, up to the dependence of ε on $f(\cdot,\cdot)$. This is summarized in the following theorem.

Theorem 2. *There exists an $\mathrm{O}(N(\log\log N)^{1/\alpha})$-space layout of the point set S such that, for any block size B, any query in Q can be answered by examining at most $\alpha + 1 + K/B$ blocks.*

Note that our goal is to show only that we cannot obtain a stronger lower bound for the point and query sets we constructed; the construction in this section does not yield a general data structure whose query and space bounds match the lower bound of Theorem 1. In particular, we ignore the cost of locating the blocks storing the points contained in a particular query, and the points in a query not in Q may be scattered across more than $\alpha + 1 + K/B$ blocks.

The layout. To construct a layout of the point set S, let f be the smallest integer such that $\ell(\alpha, f+1) \geq t$; that is, $f = \mathrm{O}(t^{1/\alpha}) = \mathrm{O}((\log\log N)^{1/\alpha})$. To simplify the argument, we can assume that $\ell(\alpha, f+1) = t$, which we can achieve by padding each grid with empty rows at the bottom. Our layout consists of f copies of S, which immediately implies that the layout uses $Nf = \mathrm{O}(N(\log\log N)^{1/\alpha})$ space.

In the first copy, we divide the rows of the top-level grid T into $\alpha + 1$ groups $G_0, G_1, \ldots, G_\alpha$, ordered from top to bottom. The size of group G_i is $\ell(\alpha - i, f)$. The cells in each group are laid out in column-major order. This is illustrated in Figure 3(a). Note that the groups $G_0, G_1, \ldots, G_\alpha$ cover all the rows of T because $t = \ell(\alpha, f+1) = \sum_{i=0}^{\alpha} \ell(i, f)$; the second equality follows from the classic equality $\binom{n+1}{m+1} = \sum_{i=0}^{n} \binom{i}{m}$. To complete the layout of the points in the first copy of S, we have to determine the order in which to arrange the points in each cell of T. Since each cell of T is divided recursively into $t \times 2^{t-1}$ subgrids, we can divide the rows of each such subgrid T' into groups $G'_0, G'_1, \ldots, G'_\alpha$ in the same fashion as just described and lay out the cells in each group in column-major order. This continues recursively until we reach subgrids T' whose cells have not been divided further in the construction of the point set S. For each such subgrid, we arrange the points in each cell in an arbitrary order.

For $1 < k \leq f$, the layout of the points in the kth copy of S is obtained by dividing the groups used to define the $(k-1)$st copy into subgroups and laying out the cells in each subgroup in column-major order. In particular, a group G in the $(k-1)$st copy consisting of $\ell(\beta, f-k+2)$ rows is divided into $\beta + 1$ subgroups $G_0, G_1, \ldots, G_\beta$, with G_i covering $\ell(i, f-k+1)$ rows. This is illustrated in Figure 3(a) for the second copy of S. Similar to the argument for the first copy of S, we have $\ell(\beta, f-k+2) = \sum_{i=0}^{\beta} \ell(i, f-k+1)$, that is, the subgroups cover the rows of G exactly. To determine the order in which to lay out the points in each cell of T, we apply the same refinement process to the groups in each subgrid T' in the recursive construction of S.

By continuing in this fashion until $k = f$, we obtain that every group in the fth copy of S consists of $\ell(\beta, 1) = 1$ rows, for some β. Hence, the fth copy of S stores the cells in each row of T (and of any subgrid T') in x-sorted order. This is illustrated on the right of Figure 3(a).

Answering queries. Since the layout arranges the cells of each subgrid T' in the recursive construction of S in the same fashion as the cells of T, it suffices to prove that any query $q \in Q_T$ can be answered using

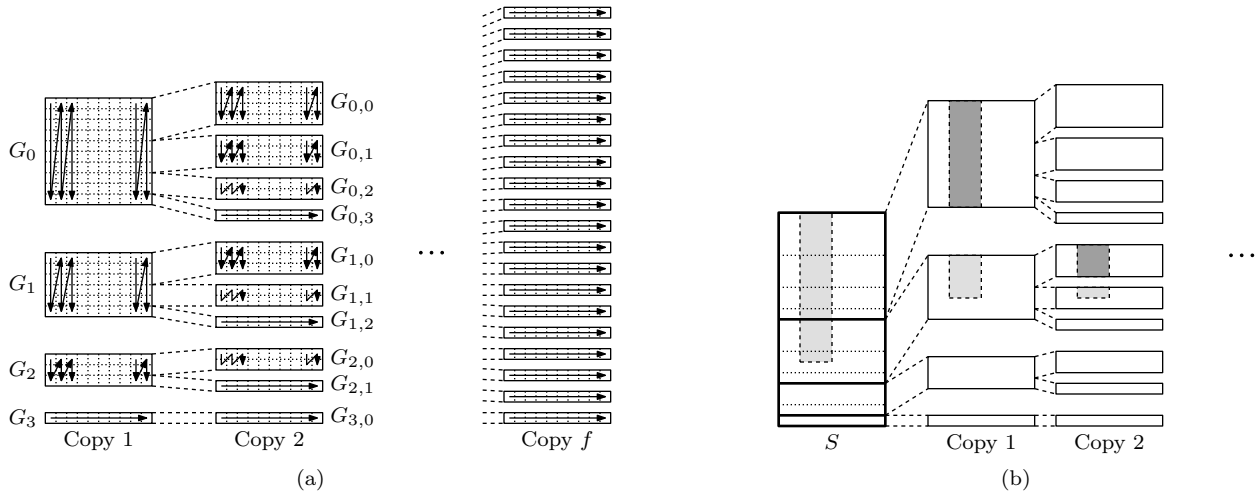


Figure 3: (a) An $O(N(\log\log N)^{\varepsilon})$-space layout for the point set S (layout shown for $\alpha = 3$). (b) Answering a query on the first two levels of the layout. The dark portion of the query in each copy of S can be answered by scanning a subsequence of the column-major layout of the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

$\alpha + 1 + K/B$ block transfers. To this end, we show that the points in query q are divided into at most $\alpha + 1$ contiguous subsequences in the layout; scanning these takes at most $\alpha + 1 + K/B$ block transfers.

Assume that query q is at the ith level of Q_T, that is, its bottom boundary coincides with the bottom boundary of the ith row of T. If this row is the bottom-most row of some group G_{j_1} in the first copy of S, observe that the points in q that belong to any group G_j, $0 \le j \le j_1$, are stored contiguously in the first copy of S. Hence, the points in q are divided into $j_1 + 1$ contiguous subsequences.

If row i belongs to group G_{j_1} but is not its bottom row, we use the first copy of S to report all points of q that belong to groups $G_0, G_1, \ldots, G_{j_1-1}$ and use subsequent copies of S to report all points in q that belong to G_{j_1}. We say that group G_{j_1} is the *partial group of* q in the first copy of S. By the arguments in the previous paragraph, the points of q that belong to groups $G_0, G_1, \ldots, G_{j_1-1}$ form j_1 contiguous subsequences in the first copy of S.

To see how subsequent copies of S are used to collect the remaining points of q, consider the kth copy and let G be the partial group of q in the $(k-1)$st copy. This group has $\ell(\beta, f - k + 2)$ rows, for some β. Then we consider the subgroup G_{j_k} of G containing row i. As for the first copy of S, if row i is the bottom-most row of group G_{j_k}, we use groups $G_0, G_1, \ldots, G_{j_k}$ to report all points in G that belong to q. These points are stored in $j_k + 1$ contiguous subsequences of the kth copy. If row i is not the bottom-most row of group G_{j_k}, then G_{j_k} is once again a partial group of q, and we report only those points in q that belong to $G_0, G_1, \ldots, G_{j_k-1}$ using the kth copy of S, and use subsequent copies of S to report the points in q that belong to G_{j_k}. Figure 3(b) illustrates this recursive partitioning of query q for the first two copies of S. Note that this procedure terminates at the latest when reaching the last copy of S because this copy stores all rows in x-sorted order, that is, the group G_{j_f} cannot be partial.

It remains to bound the number of contiguous subsequences into which the points of q are divided by this procedure. If the procedure terminates after inspecting the first r copies of S, the points in q are divided into $1 + \sum_{k=1}^{r} j_k$ contiguous subsequences, j_k in the kth copy of S, for $1 \le k < r$, and $j_r + 1$ in the rth copy. For all $1 < k \le r$, if the partial group G of q in the $(k-1)$st copy of S has size $\ell(\beta, f - i + 2)$, then $j_k \le \beta$ and G_{j_k} has $\ell(\beta - j_k, f - i + 1)$ rows. Since $j_1 \le \alpha$, this implies that $1 + \sum_{k=1}^{r} j_k \le \alpha + 1$ using a simple inductive argument.

4 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 cache-oblivious persistent B-trees with optimal 1-d range queries. The lower bounds in Section 4.1, for 3-d dominance reporting and persistent B-trees, are obtained using direct reductions from three-sided range reporting. In Section 4.2, we prove the same lower bound for 3-d halfspace range reporting, but this requires more care, as there is no direct reduction from three-sided range reporting to 3-d halfspace range reporting. Finally, in Section 4.3, we show that all lower bounds proved in Sections 2, 4.1, and 4.2 also hold for the expected size of Las-Vegas-type data structures that achieve an expected query bound of at most $f(\log_B N, K/B)$ block transfers.

4.1 3-D Dominance Reporting and Persistent B-Trees

The first result in this section is a lower bound on the space consumption of any cache-oblivious data structure for 3-d dominance reporting, as summarized in the following theorem.

Theorem 3. *Let $f(\cdot,\cdot)$ be a monotonically increasing function, and $0 < \delta \leq 1/2$ a constant. Any cache-oblivious data structure capable of answering 3-d dominance reporting queries using at most $f(\log_B N, K/B)$ block transfers in the worst case, for every block size $B \leq N^{2\delta}$, must use $\Omega(N(\log\log N)^{\varepsilon})$ space, where $\varepsilon = 1/f(\delta^{-1}, 1)$.*

Proof. A simple geometric transformation reduces three-sided range queries to 3-d dominance queries. We map each input point $p = (x_p, y_p) \in S$ to the point $\phi(p) = (-x_p, x_p, -y_p)$ in $\mathbb{R}^3$. The 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 $\phi(q) = (-l, r, -b)$. In other words, any 3-sided dominance reporting structure can be used as a 3-sided range reporting structure and, thus, is subject to the lower bound of Theorem 1. □

A similar reduction shows the following result.

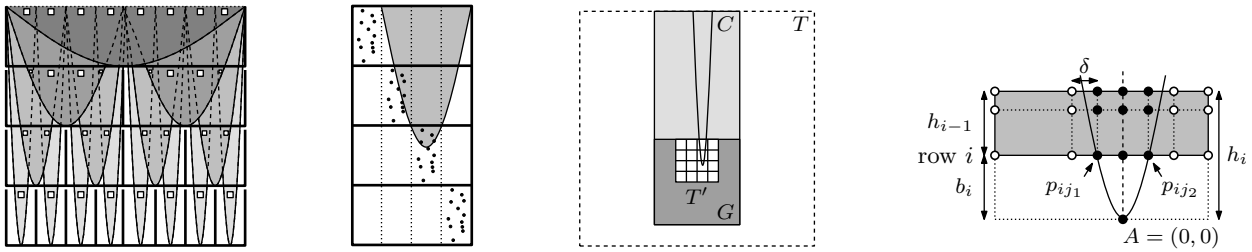
Theorem 4. *Let $f(\cdot,\cdot)$ be a monotonically increasing function, and $0 < \delta \leq 1/2$ a constant. Any (partially) persistent cache-oblivious B-tree capable of answering 1-d range range reporting queries on any previous version of the tree using at most $f(\log_B N, K/B)$ block transfers in the worst case, for every block size $B \leq N^{2\delta}$, must use $\Omega(N(\log\log N)^{\varepsilon})$ space to represent a sequence of N update operations, where $\varepsilon = f(\delta^{-1}, 1)$.*

Proof. Consider a persistent cache-oblivious B-tree T that uses $S(N)$ space to represent a sequence of N update operations and supports 1-d range reporting queries on any version of the tree using at most $f(\log_B N, K/B)$ block transfers. Using this, we can obtain a three-sided range reporting structure with query bound at most $f(\log_B N, K/B)$: 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 reporting 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. □

4.2 Halfspace Range Reporting in Three Dimensions

In this section, we extend 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 5. *Let $f(\cdot,\cdot)$ be a monotonically increasing function, and $0 < \delta \leq 1/2$ a constant. Any cache-oblivious data structure capable of answering 3-d halfspace range reporting queries using at most*



(a) (b) (c) (d)

Figure 4: (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., $T_{2,6}$). (c) Placement of a subgrid within a grid box G that is nested inside a column box C. (d) Incremental embedding of a subgrid T' inside a grid box.

$f(\log_B N, K/B)$ block transfers in the worst case, for every block size $B \leq N^{2\delta}$, must use $\Omega(N(\log\log N)^{\varepsilon})$ space, where $\varepsilon = 1/f(\delta^{-1}, 1)$.

To prove Theorem 5, we follow the recursive construction of S and Q from Section 2 and, for every subgrid T' in the construction, replace the three-sided queries in $Q_{T'}$ with parabolic ones as shown in Figure 4(a). If we embed the subgrids of T' inside the indicated squares, each query outputs the same set of points in T' as its corresponding three-sided query. However, we also have to ensure that the queries in $Q_{T'}$ output only points from T'. For three-sided queries, we achieved this by ensuring that there are no points in the x-range of T' but outside its y-range. As shown in Figure 4(b), this is not sufficient for parabolic queries, as parabolic queries over T' cannot be confined to the x-range of T'. Instead, we use the following construction, which extends a construction from [2].

Given a grid cell and a subgrid T' to be embedded inside this cell, we use G to denote the portion of the cell where the points in T' are to be placed (the white squares in Figure 4(a)). We call G the *grid box* of T'. Every parabolic query over T' is guaranteed to output only points from T' if it leaves the x-range of G only above the top boundary of the top-level grid T. We represent this using a *column box* C that shares its left, right, and bottom boundaries with G and whose top boundary is the top boundary of T; see Figure 4(c). The parabolic queries over T' must intersect only the top boundary edge of C. Once we have obtained an embedding of T' inside G and a set of parabolic queries over T' that output the same set of points as the three-sided queries in $Q_{T'}$ and intersect only the top boundary of C, we can define grid and column boxes for the subgrids of T' as above and apply this construction recursively. Thus, to prove Theorem 5, it suffices to prove that, given *any* grid box G and *any* column box $C \supseteq G$, we can embed T' inside G and construct appropriate parabolic queries over T' that intersect only the top boundary of C.

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 of T' contains exactly one point, as we can embed the subgrid represented by each such point in a sufficiently small neighbourhood of the point without altering the properties of the construction. We can further assume that the bottom boundaries of C and G coincide, that C is twice as wide as G, and that G is horizontally centred inside C, as this can be enforced by shrinking G and C appropriately without relaxing the constraints placed by these two boxes on the embedding of T' and on the queries constructed over T'. We denote the widths and heights of G and C by w_G, w_C, h_G, and h_C, respectively.

We construct T' row by row, placing the points in each row at the same y-coordinate and spacing them evenly in x-direction. The top row of T' coincides with the top boundary of G, and we centre T' horizontally in G. The points in each row have distance $\delta > 0$ between every pair of consecutive points. After placing

the points in the ith row, we construct parabolic queries for all queries at level i in $Q_{T'}$ that output exactly those points in rows 1 through i contained in their three-sided counterparts in $Q_{T'}$. The next row of points is then placed infinitesimally below the horizontal line through the apexes of these level-i queries. This is illustrated in Figure 4(d).

Once we have placed the points of all rows of T' in this fashion, we obtain an embedding of T' inside a box of width $\delta(2^{t-1}-1)$ and height h_t. We derive a bound on h_t as a function of δ. By choosing δ small enough, we can ensure that the constructed box (and, hence, T') is completely contained in G.

To discuss this construction in detail, consider the construction of the ith row. Let B_{i-1} be the smallest box containing all points already placed in rows 1 and $i-1$ and such that its bottom boundary passes through the apexes of all level-$(i-1)$ queries we have just constructed. The width of B_{i-1} is $\delta(2^{t-1}-1)$, and we denote its height by h_{i-1}. For $i=1$, we assume that $h_0=0$, that is, that B_0 is a line segment contained in the top boundary of G. To construct the ith row of T', we evenly distribute 2^{t-1} points along a horizontal line segment of the same width as B_{i-1} and infinitesimally below the bottom boundary of B_{i-1}. This ensures that no point in the ith row is contained in any query at a level less than i. To construct the queries at the ith level, we observe that each such query q' has to output points $p_{i'j'}$ with $1 \leq i' \leq i$ and $(k'-1)2^{t-i} < j' \leq k'2^{t-i}$, for some $0 < k' \leq 2^{i-1}$, where p_{ij} denotes the jth point in the ith row. For ease of notation, let $j_1 = (k'-1)2^{t-i}+1$ and $j_2 = k'2^{t-i}$. We construct the parabola q' so that it passes through points p_{ij_1} and p_{ij_2} and such that points p_{1,j_1-1} and p_{1,j_2+1}, as well as the two top corners of C, lie below q'. This ensures that q' outputs exactly the desired set of points and intersects only the top boundary edge of C.

To obtain a bound on the height h_t of the final box B_t, we first derive a bound on the distance b_i between the apex A of q' and the ith row of points, for all $1 \leq i \leq t$, and then use the recurrence $h_i > h_{i-1} + b_i$. To ease the exposition, we assume that the apex A of q' is at the origin, so that q' is given by an equation of the form $y = a_i x^2$. Let $d_i = \delta(2^{t-i}-1)/2$ be half the distance between points p_{ij_1} and p_{ij_2}. Since A has distance b_i from the ith row of points and q' passes through points p_{ij_1} and p_{ij_2}, q' must satisfy

$$b_i = a_i d_i^2. \tag{2}$$

In order for points p_{1,j_1-1} and p_{1,j_2+1} to lie below q', q' must satisfy

$$h_{i-1} + b_i < a_i(d_i+\delta)^2, \tag{3}$$

as the distance of A from the first row is infinitesimally greater than $h_{i-1}+b_i$ and the distance between two neighbouring columns of T' is δ. Finally, if we can satisfy (2) and (3) while placing A inside G, then q' intersects only the top boundary edge of C if

$$h_C < a_i \frac{w_G^2}{4}, \tag{4}$$

as the height of C is h_C and the distance between G and either of the two vertical boundary edges of C is $w_G/2$.

By substituting (2) into (3) and rearranging the result, we obtain that (3) holds if

$$a_i > \frac{h_{i-1}}{2d_i\delta + \delta^2}. \tag{5}$$

Inequalities (4) and (5) are both satisfied if

$$a_i > \frac{h_{i-1}}{2d_i\delta + \delta^2} + \frac{4h_C}{w_G^2}, \tag{6}$$

which gives

$$b_i > \left(\frac{h_{i-1}}{2d_i\delta + \delta^2} + \frac{4h_C}{w_G^2}\right) d_i^2 \tag{7}$$

by substituting (6) into (2). By using the equation $d_i = \delta(2^{t-i} - 1)/2$ and simplifying appropriately, we obtain that (7) holds if

$$b_i > 2^{t-i-2} h_{i-1} + \frac{4^{t-i}\delta^2 h_C}{w_G^2}. \tag{8}$$

By substituting this into the recurrence $h_i > h_{i-1} + b_i$, we obtain that we can satisfy (8) as long as

$$h_i > h_{i-1}(1 + 2^{t-i-2}) + \frac{4^{t-i}\delta^2 h_C}{w_G^2},$$

which holds if we set $h_0 = 0$ and $h_i = 4^{it}\delta^2 h_C / w_G^2$ and as long as $t \geq 2$. Thus, by choosing δ no greater than $\min\left(w_G/2^t, \sqrt{h_G w_G^2/(4^{t^2} h_C)}\right)$, we can ensure that all points of T' lie inside G, and that all parabolas in $Q_{T'}$ have their apexes inside G, output the same set of points as their corresponding three-sided queries, and intersect only the top boundary edge of C. This completes the proof. □

4.3 Las-Vegas-Type Data Structures

The lower bounds we have proved so far apply to data structures with deterministic construction algorithms and worst-case query bounds. In many cases, however, it is significantly easier to obtain efficient randomized data structures of the Las Vegas kind, that is, with expected query bounds resulting from randomness in their construction and in the query procedure (see, e.g., [17]). Therefore, we obtain a much stronger statement of the difficulty of cache-oblivious range searching if we can show that the lower bounds shown in the previous sections apply also to this type of data structure.

Formally, we consider data structures whose construction algorithm and query procedure both have access to a sequence of random bits. The random bits used during the construction influence the shape of the data structure, while the random bits used by the query procedure influence which blocks are read to answer a given query. In a Las-Vegas-type data structure, the random bits may influence the costs of individual queries, but the answer provided by a query must always be correct. The following theorem extends the lower bounds we have proved for deterministic data structures with worst-case query bounds to randomized data structures of the Las Vegas kind.

Theorem 6. *Let $f(\cdot,\cdot)$ be a monotonically increasing function, and $0 < \delta \leq 1/2$ a constant. Any data structure for three-sided range reporting, 3-d dominance reporting or 3-d halfspace range reporting constructed by a randomized algorithm and capable of answering queries using at most $f(\log_B N, K/B)$ block transfers in the expected sense, for every block size $B \leq N^{2\delta}$, must use expected $\Omega(N(\log\log N)^{\varepsilon})$ space, where $\varepsilon = 1/(4f(\delta^{-1}, 1))$.*

A cache-oblivious persistent B-tree that supports 1-d range reporting queries on any previous version of the tree using at most $f(\log_B N, K/B)$ block transfers in the expected sense, for every block size $B \leq N^{2\delta}$, must use expected $\Omega(N(\log\log N)^{\varepsilon})$ space to represent a sequence of N update operations.

To prove Theorem 6, it suffices to prove it for three-sided range queries and $\delta = 1/2$. As before, Lemma 1 then extends the result to smaller values of δ, and the reductions in Sections 4.1 and 4.2 extend the lower bound to 3-d dominance reporting, 3-d halfspace range reporting, and cache-oblivious persistent B-trees.

We start with the observation that we can eliminate the randomness in the query procedure. In particular, we assume that, given a data structure constructed by the randomized construction procedure and a query q, the query procedure is able to identify, at no cost, the smallest set of blocks in the data structure that contain all the points in q. Clearly, a space lower bound in this model implies the same lower bound for data structures with randomized query procedures.

Now assume that the construction procedure makes use of b random bits. Depending on the values of these bits, it produces one of $m := 2^b$ data structures $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$. Our goal is to prove that the average size of these data structures is $\Omega(N(\log\log N)^{\varepsilon})$. To do this, we consider the point set S and the query set Q constructed in Section 2.1. We consider a query $q \in Q$ *cheap* on data structure $\mathcal{D}_k$ if answering q using

$\mathcal{D}_k$ requires at most $4f(2,1)$ block transfers, and *expensive* otherwise. Since the expected cost of any query $q \in Q$ is at most $f(2,1)$, it must be cheap on at least $3m/4$ of the data structures.

Now consider a tree $\mathcal{T}$ representing the recursive construction of S. Its root is the top-level grid T, and its internal nodes represent the subgrids constructed recursively. We associate a copy $\mathcal{T}_k$ of $\mathcal{T}$ with each data structure $\mathcal{D}_k$ and refer to the copy of a node $T' \in \mathcal{T}$ in $\mathcal{T}_k$ as T'_k. This copy corresponds to the representation of the points in T' in $\mathcal{D}_k$.

In Section 2.2, we proved that, for a deterministic data structure $\mathcal{D}$ with worst-case query bounds, every grid $T' \in \mathcal{T}$ contains a cell at least half of whose points are duplicated $\Omega(t^\varepsilon)$ times. The next lemma shows that the same property holds for at least $m/4$ of its copies $T'_1, T'_2, \ldots, T'_m$.

Lemma 6. *Every node $T' \in \mathcal{T}$ has at least $m/4$ copies among $T'_1, T'_2, \ldots, T'_m$ such that each has a child at least half of whose points are duplicated $\Omega(t^\varepsilon)$ times.*

Proof. For a grid T'_k, we call the ith row of T'_k *cheap* if at least half of the level-i queries in $Q_{T'}$ are cheap on $\mathcal{D}_k$; otherwise, the row is *expensive*. As each query is cheap on at least $3m/4$ of the data structures $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$, every row of T' is cheap in at least $m/2$ of the copies $T'_1, T'_2, \ldots, T'_m$. Thus, the total number of cheap rows over all copies of T' is at least $tm/2$, which implies that there are at least $m/4$ copies of T' that have at least $t/3$ cheap rows each.

Now recall the notation and terminology from Section 2.2; the colourings of points and queries refer to data structure $\mathcal{D}_k$ now. For a sequence of cells $\langle c_1, c_2, \ldots, c_r \rangle$ in the same column of grid T'_k and its corresponding query sequence $\langle q_1, q_2, \ldots, q_r \rangle$, we call a point $p \in c_i$ *weakly exposed* if none of the queries $q_1, q_2, \ldots, q_{i-1}$ covers it, that is, if it is exposed in the sense defined in Section 2.2. Point p is *strongly exposed* if it is weakly exposed and the query q_i is cheap on $\mathcal{D}_k$. As in Section 2.2, we call a cell of T'_k weakly or strongly exposed if at least half its points are weakly or strongly exposed. To prove the lemma, it suffices to show that each copy T'_k of T' with at least $t/3$ cheap rows contains a sequence $\langle c_1, c_2, \ldots, c_r \rangle$ of cells in the same column whose length is $r = \Omega(t)$ and all of whose cells are strongly exposed. Since each query q_i is α-coloured in this case, for $\alpha = 4f(2,1)$, the lemma then follows by application of Lemma 4.

Let T'' be the subgrid of T'_k consisting of only the cheap rows of T'_k, and, similarly to the proof of Lemma 3, let T''_h be the subgrid of T'' consisting of rows $1, h+1, 2h+1, \ldots$. Using the same arguments as in the proof of Lemma 3, at most a $(4\alpha/2^h)$-fraction of the points in each row of T''_h can be covered by cheap queries at higher levels of T''_h. Thus, for $h = \lceil \log(32\alpha) \rceil = \Theta(1)$, at least a 7/8-fraction of the points in each row of T''_h are weakly exposed. Since a cell is weakly exposed if at least half of its points are weakly exposed, this implies that at least a 3/4-fraction of the cells in each row are weakly exposed. It follows that at least a 1/4-fraction of the cells in each row are strongly exposed because every row of T''_h is cheap. This, however, implies that there exists a column of T''_h such that at least a 1/4-fraction of its cells are strongly exposed. Since the height of T''_h is at least $t/(3h) = \Omega(t)$, this shows that T'_k contains a sequence of $\Omega(t)$ strongly exposed cells in the same column, and the lemma follows. □

Using Lemma 6, we can now prove that the average size of data structures $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$ is $\Omega(N(\log\log N)^\varepsilon)$. To this end, we call a node $T'_k \in \mathcal{T}_k$ *accounted for* if there exists an ancestor T''_k of T'_k in $\mathcal{T}_k$ for which we can guarantee that at least half the points in T''_k are duplicated $\Omega(t^\varepsilon)$ times. We call a point accounted for if it is contained in a grid T'_k that is accounted for. Our goal now is to show that there exists a level i in $\mathcal{T}$ such that at least a constant fraction of the points in all level-i nodes of trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m$ are accounted for. Since at least half of the accounted-for points are duplicated $\Omega(t^\varepsilon)$ times, this shows that the total size of data structures $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$ is $\Omega(mNt^\varepsilon)$, and their average size is $\Omega(Nt^\varepsilon) = \Omega(N(\log\log N)^\varepsilon)$, as claimed in Theorem 6.

For a given level i in $\mathcal{T}$, we consider all nodes at level i in $\mathcal{T}$ that have more than $7m/8$ unaccounted-for copies in trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m$. Let N_i be the total number of points in these nodes of $\mathcal{T}$. By Lemma 6, each such node T' has at least $m/8$ copies in trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m$ that are unaccounted for and such that each has a child that is accounted for at level $i+1$. If T' contains N' points, then any child of T' contains at least $N'/4^t$ points, as argued in Section 2.1. Hence, at least $(m/8) \cdot (N_i/4^t)$ new points are accounted for at depth $i+1$.

For $N_i > N/2$, this shows that at least $nM/(16 \cdot 4^t) = mN/(16\sqrt{\log N})$ new points are accounted for at depth $i+1$. Since there are only mN points in total, this implies that there can be at most $16\sqrt{\log N}$ levels in $\mathcal{T}$ that satisfy $N_i > N/2$. However, we have shown in Section 2.1 that the height of $\mathcal{T}$ is at least $\log N / \log\log N$, which is greater than $16\sqrt{\log N}$, for N sufficiently large. Hence, there exists a level i in $\mathcal{T}$ with $N_i \leq N/2$, and at least $N/2$ of the points in S are accounted for in at least $m/8$ of the trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m$. Since at least half of the accounted for points are duplicated $\Omega(t^\varepsilon)$ times, the total size of data structures $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$ is thus $\Omega(mNt^\varepsilon)$, as claimed. This concludes the proof of Theorem 6.

5 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 between the two models had been obtained (with considerable technical difficulty and using sophisticated techniques), our result is the first one that proves a gap that grows with the input size. Our proof of the lower bound continues to hold even if the data structure is aware of the block sizes we use. That is, the lower bound holds even for cache-aware multi-level memory hierarchy models. In that case, however, our proof makes the somewhat unrealistic assumption that the memory hierarchy has $\sqrt{\log N}$ levels.

Our lower bound still leaves a sizable gap to the $O(N \log N)$ space bound required by the currently best cache-oblivious data structures for the problems we considered. As our analysis in Section 3 shows, the result we obtained is in fact the best possible with the point and query sets we considered. Thus, it remains open whether stronger lower bounds can be obtained using harder point sets or whether the $O(N \log N)$ space bound of the currently best data structures for these problems can be lowered. In particular, it seems plausible that $O(N \log^\varepsilon N)$-space data structures for these problems may exist.

References

[1] Afshani, P.: On dominance reporting in 3D. In: Proceedings of the 16th European Symposium on Algorithms, *Lecture Notes in Computer Science*, vol. 5193, pp. 41–51. Springer-Verlag (2008). DOI http://www.springerlink.com/content/r853v710340481u3

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

[3] Afshani, P., Hamilton, C., Zeh, N.: A general approach for cache-oblivious range reporting and approximate range counting. In: Proceedings of the 25th ACM Symposium on Computational Geometry, pp. 287–295 (2009)

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

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

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

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

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

[9] Arge, L., de Berg, M., Haverkort, H.J., Yi, K.: 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, pp. 347–358 (2004)

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

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

[12] Bayer, R., McCreight, E.M.: Organization and maintenance of large ordered indices. Acta Informatica **1**, 173–189 (1972)

[13] Becker, B., Gschwind, S., Ohler, T., Seeger, B., Widmayer, P.: An asymptotically optimal multiversion B-tree. The VLDB Journal **5**(4), 264–275 (1996)

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

[15] Brodal, G.S., Fagerberg, R.: On the limits of cache-obliviousness. In: Proceedings of the 35th ACM Aymposium on Theory of Computing, pp. 307–315 (2003)

[16] Chan, T.M.: 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, pp. 586–595 (1998)

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

[18] Frigo, M., Leiserson, C.E., Prokop, H., Ramachandran, S.: Cache-oblivious algorithms. In: Proceedings of the 40th IEEE Symposium on Foundations of Computer Science, pp. 285–397 (1999)

[19] Grossi, R., Italiano, G.F.: Efficient cross-tree for external memory. In: J. Abello, J.S. Vitter (eds.) External Memory Algorithms and Visualization, pp. 87–106. American Mathematical Society (1999)

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

[21] Hellerstein, J.M., Koutsoupias, E., Papadimitriou, C.H.: On the analysis of indexing schemes. In: Proceedings of the 16th ACM Symposium on Principles of Database Systems, pp. 249–256 (1997)

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

[23] Koutsoupias, E., Taylor, D.S.: Tight boundns for 2-dimensional indexing schemes. In: Proceedings of the 17th ACM Symposium on Principles of Database Systems, pp. 52–58 (1998)

[24] Makris, C., Tsakalidis, A.: Algorithms for three-dimensional dominance searching in linear space. Information Processing Letters **66**(6), 277–283 (1998). DOI http://dx.doi.org/10.1016/S0020-0190(98)00075-1

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

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

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

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

[29] Samoladas, V., Miranker, D.P.: A lower bound theorem for indexing schemes and its application to multidimensional range queries. In: Proceedings of the 17th ACM Symposium on Principles of Database Systems, pp. 44–51 (1998)

[30] Vengroff, D.E., Vitter, J.S.: Efficient 3-D range searching in external memory. In: Proceedings of the 28th ACM Aymposium on Theory of Computing, pp. 192–201 (1996)

[31] Vitter, J.S.: 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`