Fast Data Structures for Orthogonal Range Reporting

Marek Karpinski* Yakov Nekrich†

Abstract

In this paper we present new results for planar and multi-dimensional orthogonal range reporting. We describe a dynamic data structure for planar orthogonal range reporting with query time $O(\log n/\log\log n + k)$ and space $O(n\log^{\varepsilon} n)$ for any $\varepsilon > 0$ and k the answer size. This is the first dynamic data structures with sublogarithmic query time for that problem.

We also present a static data structure for three-dimensional range reporting queries with $O(\log n/\log\log n + k)$ query time. For three-dimensional range reporting on a U^3 grid, we present a data structure with query time $O((\log\log U)^2\log\log\log U + k\log\log U)$; this leads to three-dimensional range reporting in $O(\sqrt{\log n/\log\log n} + k\log\log n)$ time. The last results can be extended to d-dimensional range reporting yielding also the best up to now query times for that case.

Our results depend on a new *reductions* method. This method could be of independent interest.

Keywords: Algorithms and Data Structures, Range Reporting, Planar Orthogonal Range Reporting, Multi-Dimensional Range Reporting

1 Introduction

The orthogonal range reporting problem is to maintain a set of points S so that for an arbitrary d-dimensional query rectangle Q all points from S that belong to Q can be reported. This problem has been studied extensively; surveys of the previous results are given in [2] and [12].

*Dept. of Computer Science, University of Bonn. E-mail marek@cs.uni-bonn.de. Work partially supported by a DFG grant, Max-Planck Research Prize, and IST grant 14036 (RAND-APX).

†Dept. of Computer Science, University of Bonn. E-mail yasha@cs.uni-bonn.de. Work partially supported by IST grant 14036 (RAND-APX).

In this paper we present several new results for planar (two-dimensional) and three-dimensional orthogonal range reporting. The methods used have some common features. These results can be extended to d-dimensional range reporting.

The results of this paper are valid in the unit-cost RAM model with the word size logarithmic in the number of elements.

1.1 Planar Dynamic Range Reporting

Several space efficient static data structures for this problem with logarithmic query time are described in Chazelle [10]. In particular, [10] describes a data structure with $O(n \log^{\varepsilon} n)$ space and $O(\log n + k)$ time and a data structure with $O(\log n + k \log\log \frac{4n}{k+1})$ query time and $O(n \log\log n)$ space; here and further $k = |S \cap Q|$ is the size of the answer. Using a dynamization of the fractional cascading technique of Chazelle and Guibas [9], Mehlhorn and Näher [17] described a dynamic data structure with query time $O(\log n \log\log n + k)$, update time $O(\log n \log\log n)$ and space $O(n \log n)$. Mortensen [19] described a data structure that requires $O(n \log n / \log\log n)$ space and supports queries and updates in $O(\log n + k)$ and $O(\log n)$ time respectively. In [20] the space requirements for the dynamic case are further reduced: the data structures of [20] use either $O(n \log^{\varepsilon} n)$ space and support queries in $O(\log n + k)$ time, or $O(n \log\log n)$ space and $O(\log n + k \log\log n)$ query time, thus matching the two above mentioned results of Chazelle [10] for the static case.

In the case of orthogonal range reporting on an $n \times n$ grid, there exist static data structures with sublogarithmic query time. For instance, Overmars [22] described a data structure with query time $O(\log\log n + k)$ using space $O(n \log n)$; in [3] a data structure with query time $O(\log\log n + k)$ and space $O(n \log^{\varepsilon} n)$ is described. Using the reduction to rank space technique(see, e.g., [10]) and data structures for predecessor queries, a sublogarithmic time can be achieved for the general case static data structures. Combining the fusion trees of Willard with the result of [3], we obtain a $O(\log n / \log\log n + k)$ query time and $O(n \log^{\varepsilon} n)$ space data structure; another data structure with $O(\log n / \log\log n + k)$ query time is described in [26]. Combining the exponential search trees (see [4], [5],[6]) with [3], we obtain a data structure with $O(\sqrt{\log n / \log\log n} + k)$ query time.

In the case of dynamic planar orthogonal range reporting queries, the fastest previously known dynamic data structures (cf. [19], [20]) have query time $\Omega(\log n + k)$. In this paper we present a dynamic data structure with

$O(\log n/\log\log n+k)$ query time for planar range reporting queries. To the best of our knowledge, this is the first data structure with $o(\log n+k)$ query time.

The main idea of our data structure is a reduction of a planar range reporting query to two three-sided queries.

1.2 Three-Dimensional Range Reporting

The static three-dimensional range reporting data structure with query time $O(\log^2 n + k)$ and space $O(n\log^2 n/\log\log n)$ was described by Chazelle [8]. Willard [26] has improved the query time to $O(\log^2 n/\log\log n + k)$ using fusion trees. Overmars [22] describes a data structure with query time $O(\log n\log\log n + k)$ and space $O(n\log^2 n)$. In [23] the query time was reduced to $O(\log n\log^{**} n)$. In [3] a $O(n\log^{1+\varepsilon} n)$ space and $O(\log n)$ time data structure was presented. To our knowledge, all previous data structures for the general three-dimensional range reporting required logarithmic query time, even in the case when non-constant penalties for each point in the answer are allowed. In this paper we present the first data structure for three-dimensional range reporting with sublogarithmic query time; our data structure supports queries in $O(\log n/\log\log n + k)$ time and uses $O(n\log^{2+\varepsilon} n)$ space for any $\varepsilon > 0$. If penalties for each reported point are allowed, we construct a data structure with query time $O((\log\log U)^2\log\log\log U + k\log\log U)$ and $O(n\log^3 n)$ space for range searching on a U^3 grid. Applying the standard reduction to rank space technique, we obtain a $O(\sqrt{\log n/\log\log n} + k\log\log n)$ time data structure for range reporting in $\mathbb{R}^3$. Finally, the last results can be extended to static d-dimensional range reporting: we present data structures with $O((\log n/\log\log n)^{d-2} + k)$ query time and $O(n\log^{d-1+\varepsilon} n)$ space for any $\varepsilon > 0$, and $O((\log n/\log\log n)^{d-3}(\log\log n)^2\log\log\log n + k\log\log n)$ query time and $O(n\log^d n)$ space respectively.

Our approach depends on a reduction of a three-dimensional range reporting query to several dominance reporting queries.

1.3 Main Results

We start with a more precise formulation of our main results.

Theorem 1 *There is a data structure D that supports planar orthogonal range reporting queries in $O(\log n/\log\log n + k)$ time and updates in $O(\log^2 n)$ amortized time, where k is the size of the answer. D can be*

constructed in $O(n \log n)$ time and requires $O(n \log^{\varepsilon} n)$ space for arbitrary $\varepsilon > 0$.

Theorem 2 *There is a data structure D' that supports planar orthogonal range reporting queries in $O(\log n / \log\log n + k \log\log n)$ time and updates in $O(\log^2 n)$ amortized time, where k is the size of the answer. D' can be constructed in $O(n \log n)$ time and requires $O(n \log\log n)$ space.*

The result of Theorem 1 is a $\log\log n$ factor improvement in query time compared to [20] and an improvement in terms of both space and query time compared to [19].

A *three-sided* range reporting query is a special case of the planar range reporting query, in which one side of the query rectangle is constrained to lie on one of the axes. Our result is based on two general reductions. The first reduction converts a dynamic data structure for three-sided queries into a dynamic data structure for planar range reporting. The second reduction converts a dynamic data structure with space $\Theta(n \log^p n)$ into a dynamic data structure with space $O(n \log^{\varepsilon} n)$ for arbitrary constants $p > 0$, $\varepsilon > 0$. The precise description of reductions is given in Theorems 7 and 8. These reductions are of interest on their own, since they elucidate the connection between data structures for three-sided range queries and (space efficient) data structures for general planar range reporting queries.

In the case of three-dimensional range reporting, we achieve the following results:

Theorem 3 *There exists a data structure that uses space $O(n \log^{2+\varepsilon} n)$ and answers three-dimensional orthogonal range reporting queries in $O(\log n / \log\log n + k)$ time.*

Theorem 4 *There exists a data structure that uses space $O(n \log^{2+\varepsilon} n)$ and answers three-dimensional orthogonal range reporting queries in $O(\sqrt{\log n / \log\log n} + k \log\log n)$ time.*

Applying the technique from [3] and reduction to rank space, we can obtain the following results for $d > 3$ dimensions.

Theorem 5 *For $d \geq 3$, there exists a data structure that uses space $O(n \log^{d-1+\varepsilon} n)$ and answers d-dimensional range reporting queries in $O((\log n / \log\log n)^{d-2} + k)$ time. There also exists a data structure that uses space $O(n \log^d n)$ and answers d-dimensional range reporting queries in $O((\log n / \log\log n)^{d-3} (\log\log n)^2 \log\log\log n + k \log\log n)$ time.*

Our results for three-dimensional range reporting are based on reducing a three-dimensional orthogonal range reporting to dominance reporting queries.

In section 2 we describe a data structure for dynamic orthogonal range reporting on a polynomially bounded grid. In section 2.2 we show how the space usage of this data structure can be reduced and generalize the result for planar orthogonal range reporting. In section 3 we describe the results for three-dimensional range reporting.

2 An $O(\log n/\log\log n + k)$ Time Dynamic Data Structure

In this section we describe a data structure for planar range reporting queries on a $U \times U$ grid, where $U = n^{O(1)}$. This data structure achieves query time $O(\log n/\log\log n + k)$ and requires $O(n\log^2 n)$ space.

We will use the following lemma from [26]:

Lemma 1 *There is a linear space dynamic data structure that supports three-sided queries in $O(\log n/\log\log n + k)$ time, updates in $O(\log n/\log\log n)$ amortized time, and can be constructed in $O(n)$ time.*

Following [26], we call the data structure from Lemma 1 *a fusion priority tree.*

Theorem 6 *There exists a dynamic data structure A that supports planar orthogonal range reporting queries on a $U \times U$ grid in time $O(\log n/\log\log n + k)$ and update operations in $O(\log^2 n/\log\log n)$ amortized time. A uses $O(n\log n)$ words of memory and can be constructed in $O(n\log n)$ time.*

`Main Idea` The main idea of our approach is the recursive division of the universe along the horizontal axis into intervals of equal size. That is, a $U \times U$ universe is divided into two $U/2 \times U$ rectangles; each of the $U/2 \times U$ rectangles is further subdivided into two $U/4 \times U$ rectangles, and so on. Given an arbitrary query $[a,b]\times[c,d]$, we can express $[a,b]$ as $[a,u]\cup[u+1,b]$, so that $[a,u]$ and $[u+1,b]$ belong to two adjacent horizontal intervals (say, $[a,u]\times \subset I_1$ and $[u+1,b] \subset I_2$) in our hierarchy. Using this representation, we can answer the query $[a,b]\times[c,d]$ by answering two three-sided queries: $([a,u+1)\times[c,d])\cap(I_1\times U)$ and $((u,b]\times[c,d])\cap(I_2\times U)$.

`Description of the Data Structure` We assume w.l.o.g that U is a power of 2. For ease of description we assume that all point coordinates

belong to the interval $[0, U-1]$. Let $U^0 = [0, U-1]$. We divide U^0 into two equal size intervals $U^1 = [0, U/2-1]$ and $U^2 = [U/2, U-1]$. U^2 and U^3 are divided in the same way. The lower and upper bounds of the interval U^i are denoted by l^i and r^i. This division continues as long as the interval U^i contains elements from P_x and the size of the interval is bigger than 1: if $U^i = [l^i, u^i]$, so that $r^i - l^i > 0$, and $P_x \cap U^i \neq \emptyset$ U^i is divided into $U^{2i} = [l^i, l_i + (r^i - l^i + 1)/2 - 1]$ and $U^{2i+1} = [l_i + (r^i - l^i + 1)/2, r^i]$. Intervals U^{2i} and U^{2i+1} are called *the children intervals* of U^i. We denote by P_x^i the x-coordinates of all points that belong to interval U^i: $P_x^i = U^i \cap P_x$. We say that an interval U^j is empty (non-empty) if $U^j \cap P_x = \emptyset$($U^j \cap P_x \neq \emptyset$). We say that an interval U^j is on level l if $(r^j - l^j + 1) = \dfrac{U}{2^l}$. We say that interval U^j *splits* $[a, b]$, if $l^j \in [a, b]$ and $r^j \notin [a, b]$, or $r^j \in [a, b]$ and $l^j \notin [a, b]$.

For each non-empty set P_x^i (i.e., for each non-empty interval U^i) two data structures for three-sided range reporting queries are stored. Data structures for P_x^i contain all points in $[l^i, r^i] \times P_y$ and support queries $(l^i - 1, b] \times [c, d]$ and $[a, r^i + 1) \times [c, d]$. Using Lemma 1, such queries can be answered in $O(\log n / \log\log n + k)$ time. Each point belongs to $O(\log n)$ sets P_x^i; hence, the total space used by all data structures is $O(n \log n)$.

For each level l we store in a linear space data structure R^l the upper and lower bounds of all non-empty intervals on level l. R^l supports predecessor queries, so that intervals on level l that contain or split the query interval $[a, b]$ can be found efficiently. The total space used by all R^l is also $O(n \log n)$.

`Range Reporting` To answer an arbitrary planar query $[a, b] \times [c, d]$, we find two adjacent intervals U^j, U^{j+1}, so that both U^j and U^{j+1} split $[a, b]$. Those intervals can be found as follows. Let l be such that $U/2^l \geq (b-a+1) > U/2^{l+1}$. There are at most two intervals on level l that split $[a, b]$. Using data structure R^l, we can find such intervals in $O(\log n / \log\log n)$ time. If $[a, b]$ does not intersect with any non-empty interval on level l, then $[a, b] \cap P_x = \emptyset$, and no points must be reported. If $[a, b]$ is contained in one interval U^i, then both children of U^i split $[a, b]$. If two adjacent intervals U^i and U^{i+1} split $[a, b]$, then $[a, b] = [a, r^i] \cup [l^{i+1}, b]$. Therefore a query $[a, b] \times [c, d]$ can be answered by answering at most two three-sided queries as follows: If U^i is non-empty, we report all points in $([a, r^i + 1) \times [c, d]) \cap P_x^i$. If U^{i+1} is non-empty, we report all points in $((l^{i+1} - 1, b] \times [c, d]) \cap P_x^{i+1}$. Both queries can be answered in $O(\log n / \log\log n + k)$ time.

`Update Operations` When we insert or delete an element e, it must be inserted into or deleted from $O(\log n)$ data structures for three-sided range queries. This takes time $O(\log^2 n / \log\log n)$. Besides that, $O(\log n)$ empty intervals U^j may become non-empty (in the case of an insertion), or

$O(\log n)$ non-empty intervals may become empty (in the case of a deletion). The bounds l^i and r^i of those intervals must be inserted into or deleted from the corresponding data structures R^l. This incurs an additional cost of $O(\log^2 n/\log\log n)$.

2.1 A reduction from three-sided to planar range reporting queries

The result of Theorem 6 can be generalized as follows.

Theorem 7 *Suppose there is a data structure B for three-sided queries with query time $O(t(n)+k)$, where $t(n) = \Omega(\sqrt{\log n/\log\log n})$, and (amortized) update time $u(n)$ that uses space $s(n)$ and can be constructed in $k(n)$ time. Then there exists a data structure A for planar range reporting queries on a $U \times U$ grid where $U = n^{O(1)}$. A supports queries in time $O(t(n)+k)$ and updates in (amortized) time $O(u(n)\log n)$; A uses space $O(s(n)\log n)$ and can be constructed in $O(k(n)\log n)$ time.*

The proof of Theorem 7 is analogous to the proof of Theorem 6.

2.2 A Space Efficient Data Structure

In this section we describe a general method for decreasing the space requirements of dynamic data structures for orthogonal range queries. The first result described in the introduction follows from the combination of Theorem 8 and Theorem 6.

Theorem 8 *Let $t(n) = \Omega(\sqrt{\log n/\log\log n})$. Suppose there exist a data structure A for planar orthogonal range reporting queries on a $U \times U$ grid for $U = n^{O(1)}$, so that A has query time $O(t(n)+k)$ and update time $O(\log^2 n)$, and requires space $O(n\log^p n)$ for any $p > 0$, and a linear space data structure B for three-sided queries with query time $O(t(n)+k)$ and update time $O(\log^2 n)$.*
Then there is a data structure D that supports planar range reporting queries in $O(t(n)+k)$ time and requires $O(n\log^\varepsilon n)$ space for any $\varepsilon > 0$. There is also a data structure D' that supports planar range reporting queries in $O(t(n)+k\log\log n)$ time and requires $O(n\log\log n)$ space. Both D and D' support updates in amortized time $O(\log^2 n)$. If A can be constructed in $O(n\log^p n)$ time, and B can be constructed in $O(n)$ time, then both D and D' can be constructed in $O(n\log n)$ time.

This theorem is a generalization of the result presented in [20] and can be proven in a similar way. For completeness we provide a sketch of the proof in the Appendix.

3 Faster Three-Dimensional Range Reporting

In this section we present fast static data structures for three-dimensional range reporting queries.

We show that three-dimensional orthogonal range reporting queries can be "reduced" to three-dimensional dominance reporting queries, i.e., if there is a data structure that supports dominance reporting queries in $O(\log n/\log\log n)$ time, then a data structure for the general case of orthogonal range reporting queries can be constructed.

In this section we will use the *reduction to rank space* technique (see e.g., [8]). Using this technique, a set of three-dimensional points $P \subset \mathbb{R}^3$ can be translated into a set $\hat{P} \subset [0, n-1]^3$. For a point $p \in P$, let $p = (p_x, p_y, p_z)$. Each point $p \in P$ is translated into $t(p) = \hat{p} \in \hat{P}$, so that $\hat{p}_x = \text{rank}(p_x, P_x)$, $\hat{p}_y = \text{rank}(p_y, P_y)$, and $\hat{p}_z = \text{rank}(p_z, P_z)$, where $\text{rank}(a, S) = |\{y \in S | y < a\}|$. A query $[a_x, b_x] \times [a_y, b_y] \times [a_z, b_z]$ can be translated into a query $[\hat{a}_x, \hat{b}_x] \times [\hat{a}_y, \hat{b}_y] \times [\hat{a}_z, \hat{b}_z]$, so that $\hat{a}_x = \text{rank}(pred(a_x, P_x), P_x)$, $\hat{b}_x = \text{rank}(pred(b_x, P_x), P_x)$, $\hat{a}_y = \text{rank}(pred(a_y, P_y), P_y)$, $\hat{b}_y = \text{rank}(pred(b_y, P_y), P_y)$, $\hat{a}_z = \text{rank}(pred(a_z, P_z), P_z)$, $\hat{b}_z = \text{rank}(pred(b_z, P_z), P_z)$ where $pred(a, S) = \max(y \in (S \cup \{-\infty\} | y \leq a))$ Obviously, $p \in (P \cap ([a_x, b_x] \times [a_y, b_y] \times [a_z, b_z])) \Leftrightarrow \hat{p} \in (\hat{P} \cap ([\hat{a}_x, \hat{b}_x] \times [\hat{a}_y, \hat{b}_y] \times [\hat{a}_z, \hat{b}_z]))$. Thus, if we answer six predecessor queries, we can reduce an arbitrary three-dimensional query into a query on $[0, n-1]^3$. Predecessor queries can be answered in $O(\sqrt{\log n/\log\log n})$ time or in $O(\log\log U)$ time where U is the size of the universe (see e.g., [6], [4], [5]).

We will use the following notation. Reporting all points in a product of a two-dimensional rectangle $[a, b] \times [c, d]$ and a half-open interval $(-\infty, e,]$ will be further called a *five-sided query*; reporting all points in a product of an interval $[a, b]$ and two half-open intervals will be further called a *four-sided query*. Reporting all points in a product of three half-open intervals (each of those intervals can be open to the left or to the right) is called a three-dimensional *generalized dominance query*. A three-dimensional generalized dominance query is equivalent to a three-dimensional dominance query. Using the linear space and $O(\log n/\log\log n)$ time data structure of [14], we obtain the following

Fact 1 *There exists a linear space data structure that supports generalized dominance reporting queries in* $O(\log n/\log\log n + k)$ *time.*

Using the data structure from [16], we obtain

Fact 2 *There exists a linear space data structure that supports generalized dominance reporting queries on* U^3 *grid in* $O((\log\log U)^2 \log\log\log U + k\log\log U)$ *time.*

Our reduction consists of three stages: First, we show how a four-sided range query can be reduced to a generalized dominance reporting query. Then, we show that a five-sided range reporting can be reduced to four-sided range reporting. Last, we demonstrate that a three-dimensional range reporting query can be reduced to a five-sided query.

Let P be the set of points stored in a data structure; let P_x, P_y, and P_z be the sets of x-,y-, and z-coordinates of points in P.

Theorem 9 *Suppose there is a data structure for generalized dominance reporting queries with* $O(t(n) + kp(n))$ *time and* $O(s(n))$ *space, so that* $T(n) = \Omega(v(n))$ *and* $v(n)$ *is the time necessary to answer a predecessor query.*
Then there exists a data structure that:
(a) answers four-sided queries in $O(t(n)+kp(n))$ *time and uses* $O(s(n)\log n)$ *space.*
(b) answers five-sided queries in $O(t(n)+kp(n))$ *time and uses* $O(s(n)\log^2 n)$ *space.*
(c) answers three-dimensional orthogonal range queries in $O(t(n) + kp(n))$ *time and uses* $O(s(n)\log^3 n)$ *space.*

Proof: In this proof we assume that all coordinates are in the rank space.
(a) We divide P_x into intervals U^i (and sets P_x^i) in the same way as in Theorem 6. For each set P_x^i two data structures for three-dimensional generalized dominance reporting queries are stored. Data structures for P_x^j support queries $(l^j-1, a]\times(-\infty, b]\times(-\infty, c]$ and queries $[a, r^j+1)\times(-\infty, b]\times(-\infty, c]$. Since each point is stored in $O(\log n)$ data structures for generalized dominance reporting, and each of those data structures requires $O(s(n))$ space, the total space required by our construction is $O(s(n)\log n)$.

Consider a four-sided query $[a, b]\times(-\infty, c]\times(-\infty, d]$. To answer this query, we find two adjacent intervals U^i, U^{i+1}, such that U^i and U^{i+1} are on the same level, and both U^i and U^{i+1} split $[a, b]$. We can do it using the same procedure as in Theorem 6. Since $[a, b] = [a, r^i + 1)\cup(l^{i+1} - 1, b]$, all points

from $[a,b]\times(-\infty,c]\times(-\infty,d]$ are either in $([a,r^i+1)\times(-\infty,c]\times(-\infty,d])\cap P_x^i$ or in $((l^{i+1}-1,b]\times(-\infty,c]\times(-\infty,d])\cap P_x^{i+1}$. By reporting all points in the above queries we answer the four-sided query $[a,b]\times(-\infty,c]\times(-\infty,d]$.

(b) We divide P_y into sets P_y^i in the same way as in part **(a)**. For each set P_y^i we store two data structures for four-sided range queries from part **(a)**. These data structure allow us to answer queries $[a,b]\times(l^i-1,d]\times[e,+\infty)$ and $[a,b]\times[c,r^i+1)\times[e,+\infty)$. Since every point is stored in $\lceil\log n\rceil$ data structures, and each data structure for four-sided range queries requires $O(s(n)\log n)$ space, the total space required by our construction is $O(s(n)\log^2 n)$. Given a five-sided query $[a,b]\times[c,d]\times[e,+\infty)$, we find two adjacent intervals U^i and U^{i+1} such that both U^i and U^{i+1} split $[c,d]$ using the same algorithm as in part **(a)**. Then, we can answer the five-sided query by answering two four-sided queries $([a,b]\times[c,r^i+1)\times[e,+\infty))\cap P_y^i$ and $([a,b]\times(l^{i+1}-1,d]\times[e,+\infty))\cap P_y^{i+1}$.

(c) Again, P_z is divided into sets P_z^i in the same way as P_x and P_y above. We store data structures for five-sided queries for each set P_z^i: the queries $[a,b]\times[c,d]\times[e,r^i+1)$ and $[a,b]\times[c,d]\times(l^i-1,f]$ are supported. Given a query $[a,b]\times[c,d]\times[e,f]$, we can find two adjacent U^i, U^{i+1}, such that both U^i and U^{i+1} split $[e,f]$. After this, the query is answered by reporting all points in $([a,b]\times[c,d]\times[e,r^i+1))\cap P_z^i$ and $([a,b]\times[c,d]\times(l^{i+1}-1,f])\cap P_z^{i+1}$. This data structure uses $O(s(n)\log^3 n)$ space because each point is stored in $O(\log n)$ data structures from part **(b)**. □

Corollary 1 *There exists a data structure that answers three-dimensional orthogonal range reporting queries in $O(\sqrt{\log n/\log\log n}+k\log\log n)$ time. All data structures use space $O(n\log^3 n)$.*

This Corollary can be obtained by a combination of the reduction to rank space technique, Fact 2, and Theorem 9 . In this case, the query time is dominated by the search for predecessors.

We can further reduce the space requirements of the first data structure.

Theorem 10 *Suppose there is a data structure that supports three-dimensional dominance queries in $O(t(n)+kp(n))$ time and uses $O(n)$ space for $t(n)=\Omega(\sqrt{\log n/\log\log n})$. Then there exists a data structure A that uses space $O(n\log^{2+\varepsilon} n)$ for any $\varepsilon>0$ and answers three-dimensional range reporting queries in $O(t(n)+kp(n))$ time.*

Proof: We start with a description of the data structure A. We divide P_x into $n^{1/3}$ intervals $[x_{i-1},x_i]$, so that each three-dimensional rectangle

$[x_{i-1}, x_i] \times P_y \times P_z$ contains $n^{2/3}$ points. We divide P_y into $n^{1/3}$ intervals $[y_{i-1}, y_i]$, so that each three-dimensional rectangle $P_x \times [y_{i-1}, y_i] \times P_z$ contains $n^{2/3}$ points. We divide P_z into $n^{1/3}/\log^3 n$ intervals $[z_{i-1}, z_i]$, so that each $P_x \times P_y \times [z_{i-1}, z_i]$ contains $n^{2/3}\log^3 n$ points. Three-dimensional rectangles $[x_{i-1}, x_i] \times P_x \times P_z$ will be further called x-slices; three-dimensional rectangles $P_x \times [y_{i-1}, y_i] \times P_z$ and $P_x \times P_y \times [z_{i-1}, z_i]$ will be called y-slices and z-slices respectively.

For each x-slice $[x_{i-1}, x_i] \times P_x \times P_z$ two data structures for five-sided range queries are stored: they support queries $(x_{i-1} - 1, b] \times [c, d] \times [e, f]$ and $[a, x_i + 1) \times [c, d] \times [e, f]$. Data structures for a y-slice $P_x \times [y_{i-1}, y_i] \times P_z$ support queries $[a, b] \times (y_{i-1} - 1, d] \times [e, f]$ and $[a, b] \times [c, y_i + 1) \times [e, f]$; data structures for a z-slice $P_x \times P_y \times [z_{i-1}, z_i]$ support queries $[a, b] \times [c, d] \times (z_{i-1} - 1, f]$ and $[a, b] \times [c, d] \times [e, z_i + 1)$.

The data structure A_t contains points (i, j, k), such that the cell $[x_{i-1}, x_i] \times [y_{j-1}, y_j] \times [z_{k-1}, z_k]$ contains at least one point. A_t supports three-dimensional range reporting queries. The maximal number of points in A_t is $O(n/\log^3 n)$; hence, we can use Theorem 9 to implement A_t in $O(n)$ space. For each non-empty cell $[x_{i-1}, x_i] \times [y_{j-1}, y_j] \times [z_{k-1}, z_k]$, we also store the list of points contained in this cell.

For each $[x_{i-1}, x_i] \times P_y \times P_z$, $P_x \times [y_{i-1}, y_i] \times P_z$ and $P_x \times P_y \times [z_{i-1}, z_i]$ a recursively defined data structure is stored. We call the data structure that contains all points a *level 0 data structure*, and data structures that contain all points in an x-, $y-$, or z-slice of a level l data structure are called level $l+1$ data structures. Observe that the total number of elements in all level l data structures increases by a factor 3 with each level. On every $r = \delta \log\log n$-th level we apply reduction to rank space. We will explain later how the parameter δ can be chosen. Let D^l be a data structure on ir-th recursion level, and let S^{l+1} be the set of points in an arbitrary x-slice, y-slice, or z-slice of D^l. We store a table that allows us to find the coordinates of a point in S^{l+1} in the rank space of S^{l+1}; using hash functions, this table can be implemented in $O(|S^{l+1}|\log(|S^{l+1}|))$ bits. We also store for each point in the rank space of S^{l+1} its "original" coordinates in D^l. The total space used by the tables for all slices is $O(|D^l|\log(|D^l|))$ bits.

Consider an arbitrary query $[a, b] \times [c, d] \times [e, f]$. If the query rectangle $[a, b] \times [c, d] \times [e, f]$ is contained in one slice, we transfer the query to a data structure for the corresponding slice. Otherwise, let $x_{i_1-1} < a < x_{i_1}$, $x_{i_2-1} < b < x_{i_2}$, $y_{j_1-1} < c < y_{j_1}$, $y_{j_2-1} < d < y_{j_2}$, and $z_{k_1-1} < e < z_{k_1}$, $z_{k_2-1} < f < z_{k_2}$. We report all points in $[x_{i_1}, x_{i_2-1}] \times [y_{j_1}, y_{j_2-1}] \times [z_{k_1}, z_{k_2-1}]$ by finding all non-empty cells $(i, j, k) \in [i_1, i_2 - 1] \times [j_1, j_2 - 1] \times [k_1, k_2 - 1]$ in A_t and reporting all points in those cells. The rest of the points in

$[a,b] \times [c,d] \times [e,f]$ can be found by answering six five-sided queries: $[a,b] \times [c,d] \times (z_{k_2-1}-1, f]$, $[a,b] \times [c,d] \times [e, z_{k_1}+1)$, $[a,b] \times (y_{j_2-1}-1, d] \times [z_{k_1}, z_{k_2-1}]$, $[a,b] \times [c, y_{j_1}+1) \times [z_{k_1}, z_{k_2-1}]$, $(x_{i_2-1}-1, b] \times [y_{j_1}, y_{j_2-1}] \times [z_{k_1}, z_{k_2-1}]$, $[a, x_{i_1}+1) \times [y_{j_1}, y_{j_2-1}] \times [z_{k_1}, z_{k_2-1}]$.

Query time can be computed by the formula $q(n,k) = O(t(n)+k'p(n))+ q(n^{2/3}\log^3 n, k'')$, where $q(n,k)$ is the query time, $k'+k''=k$, and k is the size of the answer. Hence, $q(n,k) = O(t(n)+kp(n))$.

The space used by our data structure can be computed as follows. Let $s^l(n)$ be the maximal number of points stored in a level l data structure; $s^l(n)$ can be estimated with a recursive formula $s^l(n) = (s^{l-1}(n))^{2/3}\log^3(s^{l-1}(n))$; hence, $s^l(n) = O((s^{l-1}(n))^{2/3+\gamma})$ for an arbitrary constant γ. The maximum number of levels is $\log_{(2+3\gamma)/3}(1/2)\log\log n = O(\log\log n)$. Consider an arbitrary group of $\delta\log\log n$ levels: $ir, ir+1, \ldots, ir+\delta\log\log n-1$, $i = 0, 1, \ldots, \lceil \log_{(3/(2+3\gamma)} 2/\delta\rceil - 1$. Each point is stored in 3^{ir} data structures on level ir. Therefore, the total number of elements in all data structures on level ir is $n3^{ir}$. Since reduction to rank space was applied on level ir, each point is stored with $O(s^{ir}(n))$ bits. A data structure on level ir requires $O(s^{ir}(n)\log^3(s^{ir}(n)))$ bits. Each point in each data structure on level ir takes $O(\log^3(s^{ir}(n)))$ bits, and $s^{ir}(n) = O(n^{((2+3\gamma)/3)^{ir}})$. Therefore the total number of bits used is (up to a constant factor) $n((2+\gamma)/3)^{3ir}3^{ir}\log^3 n$ For a sufficiently small γ, $((2+\gamma)/3)^3 < 1/3$ and $n((2+\gamma)/3)^{3ir}3^{ir}\log^3 n < n\log^3 n$; hence, all data structures on level ir use $O(n\log^3 n)$ bits.

The space used by the data structures on levels $ir+1, \ldots, ir+\delta\log\log n-1$ increases by a factor 3 with each level. Hence the total space used by all recursive data structures on levels $ir, \ldots, ir+\delta\log\log n-1$ is $O(n\log^3 n)\sum_{i=0}^{\delta\log\log n-1} 3^i$. We can choose δ so that $\sum_{i=0}^{\delta\log\log n-1} 3^i = O(\log^\varepsilon n)$ for an arbitrary $\varepsilon > 0$. Hence all data structures on levels $ir+1, \ldots, ir+\delta\log\log n-1$ use $O(n\log^{3+\varepsilon} n)$ bits. Since there is a constant number of groups of levels, all recursively defined data structures require $O(n\log^{2+\varepsilon} n)$ words of $\log n$ bits.

□

Theorem 4 follows from Corollary 1 and Theorem 10. Theorem 3 follows from Fact 1 and Theorem 10.

It was shown in [3] that given a $O(s(n))$ space and $O(t(n)+kp(n))$ time data structure for d-dimensional range reporting queries, we can construct for any $\varepsilon > 0$ a $O(t(n)(\log n/\log\log n)^m + kp(n))$ time and $O(s(n)\log^{m+\varepsilon} n)$ space data structure that supports $(d+m)$-dimensional orthogonal range reporting queries. Applying the technique of [3] to Fact 1 combined with Theorem 10 we obtain the first result of Theorem 5. Combining reduction to rank space, Fact 2, and Theorem 9, we obtain the second result of Theorem

5.

4 Summary

In this paper we presented the first dynamic data structure for planar orthogonal range reporting and static data structures for three-dimensional orthogonal range reporting with sublogarithmic query time.

Our results are based on some important reductions. Using those reductions a dynamic data structure for three-sided queries with query time $O(t(n)+k)$, where $t(n) = \Omega(\sqrt{\log n/\log\log n})$ (i.e., $t(n)$ is not asymptotically faster than the lower bound on the predecessor queries), can be converted into a data structure for general planar range reporting queries with only a small increase in space and without changing query time. We can also reduce three-dimensional orthogonal range reporting queries to three-dimensional dominance reporting queries.

Several modifications of the presented reductions can be proven in a similar way. For instance, given a linear space data structure for three-sided queries on the $n\times n$ grid with query time $O(t(n)+k)$ for an arbitrary function $t(n)$, a dynamic data structure for orthogonal range reporting on the $n \times n$ grid with query time $O(t'(n)+k)$, such that $t'(n) = O(t'(n^{2/3}) + t(n^{2/3}))$, and space $O(n\log^{\varepsilon} n)$ can be constructed.

References

[1] P. K. Agarwal, L. Arge, A. Danner, B. Holland-Minkley "Cache-oblivious Data Structures for Orthogonal Range Searching."Proc. 9th ACM Symp. on Computational Geometry (2003), 237-245.

[2] P. K. Agarwal and J. Erickson "Geometric range searching and its relatives" In "Advances in Discrete and Computational Geometry", vol. 23 of Contemporary Mathematics, 1–56. AMS Press, Providence, RI, 1999. Available at `http://citeseer.ist.psu.edu/article/agarwal99geometric.html`.

[3] S. Alstrup, G. S. Brodal, T. Rauhe " New Data Structures for Orthogonal Range Searching", Proc. 41st IEEE FOCS(2000), 198-207.

[4] A. Andersson, *Faster Deterministic Sorting and Searching in Linear Space*, Proc 37th IEEE FOCS (1996), 135-141.

[5] A. Andersson, M. Thorup, *Tight(er) worst-case bounds on dynamic searching and priority queues*, Proc. 32nd ACM STOC(2000), 335-342.

[6] P. Beame, F. E. Fich, *Optimal Bounds for the Predecessor Problem and Related Problems*, J. Comput. Syst. Sci. (65), 2002, 38-72.

[7] M. A. Bender, E. D. Demaine, M. Farach-Colton "Cache-Oblivious B-Trees", Proc. 41st IEEE FOCS(2000), 399-409.

[8] B. Chazelle "Filtering Search: A New Approach To Query Answering", SIAM J. on Computing (15), 1986, 703-724.

[9] B. Chazelle, L. J. Guibas " Fractional Cascading: I. A Data Structuring Technique", Algorithmica (1), 1986, 133-162

[10] B. Chazelle "A Functional Approach to Dynamic Data Structures", SIAM J. on Computing (17), 1988, 427-462.

[11] B. Chazelle "Lower Bounds for Orthogonal Range Search II. The Arithmetic Model", J. of the ACM (37), 1990, 439 - 463.

[12] J. L. Chiang, R. Tamassia " Dynamic Algorithms in Computational Geometry", Technical Report CS-91-24, Dept. of Computer Science, Brown University, 1991.

[13] A. Itai, A. G. Konheim, M. Rodeh "A Sparse Table Implementation of Priority Queues", Proc. 8th ICALP(1981), 417-431.

[14] J. JaJa, C. W. Mortensen, Q. Shi " Space-Efficient and Fast Algorithms for Multidimensional Dominance Reporting and Counting" Proc. 15th ISAAC(2004), 558-568.

[15] G. S. Lueker " A Data Structure for Orthogonal Range Queries", Proc. 19th ACM FOCS(1978), 28-34.

[16] C. Makris, A. K. Tsakalidis "Algorithms for Three-Dimensional Dominance Searching in Linear Space" Information Processing Letters (66), 1998, 277-283.

[17] K. Mehlhorn and S. Näher "Dynamic Fractional Cascading" , Algorithmica (5), 1990, 215-241.

[18] E.M. McCreight "Priority Search Trees", SIAM J. on Computing (14), 1985, 257-276.

[19] C. W. Mortensen "Fully Dynamic Two Dimensional Orthogonal Range and Line Segment Intersection Reporting in Logarithmic Time" Proc. 14th ACM-SIAM Symposium on Discrete Algorithms(2003), 618-627.

[20] Y. Nekrich, "Space efficient dynamic orthogonal range reporting", Proc. 21st ACM Symp. on Computational Geometry (2005), 306-313.

[21] M. H. Overmars "Design of Dynamic Data Structures" Springer-Verlag New York, Inc., Secaucus, NJ, 1987.

[22] M. H. Overmars "Efficient Data Structures for Range Searching on a Grid", J. Algorithms (9),1988, 254-275.

[23] S. Subramanian, S. Ramaswamy "The P-range Tree: A New Data Structure for Range Searching in Secondary Memory", Proc. 6th SODA (1995), 378-387 .

[24] D. E. Willard "New Data Structures for Orthogonal Range Queries", SIAM J. on Computing (14), 1985, 232-253.

[25] D. E. Willard "Applications of Range Query Theory to Relational Data Base Join and Select Operations", Journal of Computer and System Sciences (52), 1996, 157-169.

[26] D. E. Willard, Examining Computational Geometry, Van Emde Boas Trees, and Hashing from the Perspective of the Fusion Tree. SIAM J. Comput. (29), 2000, 1030-1049.

Appendix I. Proof of Theorem 8

Proof Sketch : The set of points S is divided into columns C_i and rows R_i, so that the number of elements in every column (row) is between $\sqrt{n \log^p n}/2$ and $2\sqrt{n \log^p n}$. We store lists of points in all $K_{ij} = C_i \cap R_j$; data structure D_t contains points (i, j) for $K_{ij} \neq \emptyset$. Given A, D_t can be implemented in $O(n)$ space, so that queries and updates are supported in time $O(t(n) + k)$ and $O(\log^2 n)$ respectively. For each column and row two data structures for three-sided range queries are stored. Using those data structures B, every query of the kind $(r_{i-1}, d] \times [a, b]$ or $[d, r_i) \times [a, b]$ or $(c_{j-1}, b] \times [c, d]$ and $[b, c_j) \times [c, d]$ can be answered in $O(t(n)+k)$ time. If the number of elements in a row or a column exceeds $t(n)$ a recursively defined data structure is stored for this row or column.

Consider a query $[a, b] \times [c, d]$. We find i_{min}, i_{max}, j_{min}, and j_{max}, such that $c_{i_{min}-1} < a$, $b < c_{i_{max}}$, $r_{j_{min}-1} < c$ and $d < r_{j_{max}}$. Again, we distinguish between two cases. If $c_{j-1} < a$ and $b < c_j$ for some j, or $r_{i-1} < c$ and $d < r_i$ for some i, then the query is transferred to a data structure corresponding to column C_j or row R_i. Otherwise we answer four three-sided range queries to report all elements from marginal columns and rows, and we identify all non-empty rectangles K_{ij} by answering query $[i_{min}, i_{max} - 1] \times [j_{min}, j_{max} - 1]$. In the first case, the size of the data structure is reduced from n to $\sqrt{n \log^p n}$, and in the second case the query is answered in $O(t(n) + k)$ time. Let query time $q(n, k) = O(t'(n) + k)$; $t'(n)$ can be estimated as $t'(n) = \max(O(t(n)), t'(\sqrt{n \log^p n}))$. $t'(n) < O(t(n)) + t'(\sqrt{n \log^p n})$. Since $\sqrt{n \log^p n} = o(n^{(b+1)/2b})$ for any integer b, $t'(n) < O(t(n)) + t'(n^{(b+1)/2b})$. For $t(n) = \Omega(\sqrt{\log n / \log \log n})$, $t'(n) = O(t(n))$. Hence, queries can be answered in $O(t(n) + k)$ time.

We call the data structure that contains all points a *level 0 data structure*, and data structures that contain all points in a column or row of a level l data structure are called level $l + 1$ data structures. It can be shown that the maximal recursion level $l_{max} = \log \log n + c$, where c is a constant. Every point is stored in D $O(\log n)$ times: once in a level 0 data structure, twice in level 1 data structures, 2^l times in level l data structures. The space requirements can be reduced from $O(n \log n)$ to $O(n \log^\varepsilon n)$ for any $\varepsilon > 0$ by using the *dynamic range reduction to extended rank space* technique described in [20]. If we apply this technique on a recursion level l, every point in each level l data structure can be stored with only $O(\log(s^l(n)))$ bits, where $s^l(n)$ is the maximal number of points in a level l data structure. To avoid penalties for each point in the answer, we apply range reduction on every $\varepsilon \log \log n$-th level. Then every group of $\varepsilon \log \log n$ levels takes $O(n \log^{1+\varepsilon} n)$

bits, and the whole data structure D requires $O(n \log^{\varepsilon} n)$ words of $\log n$ bits. Alternatively, the dynamic range reduction can be applied on each recursive level. Then all data structures on level l, $l = 1, 2, \ldots, \log\log n + c$, use $O(n \log n)$ bits, and the total space can be reduced to $O(n \log\log n)$ words. But in this case the reverse range reduction must be applied up to $\Theta(\log\log n)$ times to restore the original point coordinates, and for each point in the answer an $O(\log\log n)$ penalty must be paid. Therefore, the data structure D' uses $O(n \log\log n)$ words of memory, and supports queries in time $O(t(n) + k \log\log n)$.

Update time can be estimated with help of a recursive formula: $u(n) \leq O(\log^2 n) + 2u(\sqrt{n \log^p n}) < O(\log^2 n) + 2u(n^{(b+1)/2b})$ for an arbitrary integer $b > 1$. Therefore, $u(n) = O(\log^2 n)$.

To construct D or D', we must construct a data structure A with $n/\log^p n$ elements, fusion priority trees for rows and columns, and $2\sqrt{n/\log^p n}$ data structures for rows and columns with $\sqrt{n \log^p n}$ elements each. Let $c'(n)$ be the construction time of A and $c(n)$ the construction time of D. Then for $c(n)$ a recursion $c(n) = c'(n/\log^p n) + 2\sqrt{n/\log^p n}\, c(\sqrt{n \log^p n}) + O(n)$ is valid. Let $v(n) = c(n)/n$. As $c'(n/\log^p n) = O(n)$, $v(n) = 2v(\sqrt{n \log^p n}) + O(1)$. Since the number of recursive level is $\log\log n + c$ for a constant c, $v(n) = O(\log n)$, and $c(n) = O(n \log n)$. □