Shared-Constraint Range Reporting

Sudip Biswas[1], Manish Patil[1], Rahul Shah[1], and Sharma V. Thankachan[2]

1 **Louisiana State University, USA**
{sbiswa7,mpatil,rahul}@csc.lsu.edu
2 **Georgia Institute of Technology, USA**
sharma.thankachan@gatech.edu

Abstract

Orthogonal range reporting is one of the classic and most fundamental data structure problems. (2,1,1) query is a 3 dimensional query with two-sided constraint on the first dimension and one sided constraint on each of the 2nd and 3rd dimension. Given a set of N points in three dimension, a particular formulation of such a $(2, 1, 1)$ query (known as four-sided range reporting in three-dimension) asks to report all those K points within a query region $[a, b] \times (-\infty, c] \times [d, \infty)$. These queries have overall 4 constraints. In Word-RAM model, the best known structure capable of answering such queries with optimal query time takes $O(N \log^{\epsilon} N)$ space, where $\epsilon > 0$ is any positive constant. It has been shown that any external memory structure in optimal I/Os must use $\Omega(N \log N / \log \log_B N)$ space (in words), where B is the block size [Arge et al., PODS 1999]. In this paper, we study a special type of $(2, 1, 1)$ queries, where the query parameters a and c are the same i.e., $a = c$. Even though the query is still four-sided, the number of independent constraints is only three. In other words, one constraint is shared. We call this as a *Shared-Constraint Range Reporting* (SCRR) problem. We study this problem in both internal as well as external memory models. In RAM model where coordinates can only be compared, we achieve linear-space and $O(\log N + K)$ query time solution, matching the best-known three dimensional dominance query bound. Whereas in external memory, we present a linear space structure with $O(\log_B N + \log \log N + K/B)$ query I/Os. We also present an I/O-optimal (i.e., $O(\log_B N + K/B)$ I/Os) data structure which occupies $O(N \log \log N)$-word space. We achieve these results by employing a novel divide and conquer approach. SCRR finds application in database queries containing sharing among the constraints. We also show that SCRR queries naturally arise in many well known problems such as top-k color reporting, range skyline reporting and ranked document retrieval.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases data structure, shared constraint, multi-slab, point partitioning

Digital Object Identifier 10.4230/LIPIcs.ICDT.2015.277

1 Introduction

Orthogonal range searching is one of the central data structure problems which arises in various fields. Many database applications benefit from the structures which answer range queries in two or more dimensions. Goal of orthogonal range searching is to design a data structure to represent a given set of N points in d-dimensional space, such that given an axis-aligned query rectangle, one can efficiently list all points contained in the rectangle. One simple example of orthogonal range searching data structure represents a set of N points in 1-dimensional space, such that given a query interval, it can report all the points falling within the interval. A balanced binary tree taking linear space can support such queries in optimal

18th International Conference on Database Theory (ICDT'15).
Editors: Marcelo Arenas and Martín Ugarte; pp. 277–290

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$O(\log N + K)$ time. Orthogonal range searching gets harder in higher dimensions and with more constraints. The hardest range reporting, yet having a linear-space and optimal time (or query I/Os in external memory) solution is the three dimensional dominance reporting query, also known as $(1, 1, 1)$ query [1] with one-sided constraint on each dimension. Here the points are in three-dimensions and the query asks to report all those points within an input region $[q_1, \infty) \times [q_2, \infty) \times [q_3, \infty)$. A query of the form $[q_1, q_2] \times [q_3, \infty)$ is known as $(2, 1)$ query, which can be seen as a particular case of $(1, 1, 1)$ query. However, four (and higher) sided queries are known to be much harder and no linear-space solution exists even for the simplest two dimensional case which matches the optimal query time of three dimensional dominance reporting. In Word-RAM model, the best result (with optimal query time) is $O(N \log^{\epsilon} N)$ words [10], where N is the number of points and $\epsilon > 0$ is an arbitrary small positive constant. In external memory, there exists an $\Omega(N \log N / \log \log_B N)$-space lower bound (and a matching upper bound) for any two-dimensional four-sided range reporting structure with optimal query I/Os [6]. Therefore, we cannot hope for a linear-space or almost-linear space structure with $O(\log_B N + K/B)$ I/Os for orthogonal range reporting queries with four or more constraints. The model of computation we assume is a unit-cost RAM with word size logarithmic in n. In RAM model, random access of any memory cell and basic arithmetic operations can be performed in constant time.

Motivated by database queries with constraint sharing and several well known problems (More details in Section 2), we study a special four sided range reporting query problem, which we call as the *Shared-Constraint Range Reporting* (SCRR) problem. Given a set $\mathcal{P}$ of N three dimensional points, the query input is a triplet (a, b, c), and our task is to report all those points within a region $[a, b] \times (-\infty, a] \times [c, \infty)$. We can report points within any region $[a, b] \times (-\infty, f(a)] \times [c, \infty)$, where $f(\cdot)$ is a pre-defined monotonic function (using a simple transformation). The query is four sided with only three independent constraints. Many applications which model their formulation as 4-sided problems actually have this sharing among the constraints and hence better bounds can be obtained for them using SCRR data structures. Formally, we have the following definition.

▶ **Definition 1.** A SCRR query $Q_{\mathcal{P}}(a, b, c)$ on a set $\mathcal{P}$ of three dimensional points asks to report all those points within the region $[a, b] \times (-\infty, a] \times [c, \infty)$.

The following theorems summarize our main results.

▶ **Theorem 2** (SCRR in Ram Model). *There exists a linear space RAM model data structure for answering SCRR queries on the set $\mathcal{P}$ in $O(\log N + K)$ time, where $N = |\mathcal{P}|$ and K is the output size.*

▶ **Theorem 3** (Linear space SCRR in External Memory). *SCRR queries on the set $\mathcal{P}$ can be answered in $O(\log_B N + \log \log N + K/B)$ I/Os using an $O(N)$-word structure, where $N = |\mathcal{P}|$, K is the output size and B is the block size.*

▶ **Theorem 4** (Optimal Time SCRR in External Memory). *SCRR queries on the set $\mathcal{P}$ can be answered in optimal $O(\log_B N + K/B)$ I/Os using an $O(N \log \log N)$-word structure, where $N = |\mathcal{P}|$, K is the output size and B is the block size.*

Our Approach. Most geometric range searching data structures use point partitioning scheme with appropriate properties, and recursively using the data structure for each partition. Our paper uses a novel approach of partitioning the points which seem to fit SCRR problem very well. Our data structure uses rank-space reduction on the given point-set, divide the SCRR query data structure based on small and large output size, takes advantage

of some existent range reporting data structure to obtain efficient solution and then bootstrap the data structure for smaller ranges.

Related Work. The importance of two-dimensional three-sided range reporting is mirrored in the number of publications on the problem. The general two-dimensional orthogonal range searching has been extensively studied in internal memory [2, 3, 4, 11, 12, 13, 9, 7]. The best I/O model solution to the three-sided range reporting problem in two-dimensions is due to Arge et al. [6], which occupies linear space and answers queries in $O(\log_B N + K/B)$ I/Os. Vengroff and Vitter [20] addressed the problem of dominance reporting in three dimensions in external memory model and proposed $O(N \log N)$ space data structure that can answer queries in optimal $O(\log_B N + K/B)$ I/Os. Recently, Afshani [1] improved the space requirement to linear space while achieving same optimal I/O bound. For the general two-dimensional orthogonal range reporting queries in external memory settings Arge et al. [6] gave $O((N/B)\log_2 N/\log_2 \log_B N)$ blocks of space solution achieving optimal $O(\log_B N + K/B)$ I/Os. Another external memory data structure is by Arge et al. [5] where the query I/Os is $O(\sqrt{N/B} + k/B)$ and the index space is linear. In the case when all points lie on a $U \times U$ grid, the data structure of Nekrich [19] answers range reporting queries in $O(\log\log_B U + K/B)$ I/Os. In [19] the author also described data structures for three-sided queries that use $O(N/B)$ blocks of space and answer queries in $O(\log\log_B U + K/B)$ I/Os on a $U \times U$ grid and $O(\log_B^{(h)} N)$ I/Os on an $N \times N$ grid for any constant $h > 0$. Very recently, Larsen and Pagh [17] showed that three-sided point reporting queries can be answered in $O(1 + K/B)$ I/Os using $O(N/B)$ blocks of space.

Outline. In section 2, we show how SCRR arises in database queries and relate SCRR problem to well known problems of colored range reporting, ranked document retrieval, range skyline queries and two-dimensional range reporting. In section 3 we discuss rank-space reduction of the input point-set to make sure no two points share the same x-coordinate. In section 4 we introduce a novel way to partition the point-set for answering SCRR queries which works efficiently for larger output size. Section 5 explains how to answer SCRR queries for smaller output size. Using these two data structures, section 6 obtains linear space and $O(\log N + K)$ time data structure for SCRR queries in RAM model thus proving theorem 2. Section 7 discusses SCRR queries in external memory, which includes a linear space but sub-optimal I/O and an optimal I/O but sub-optimal space data structures.

2 Applications

In this section, we show application of SCRR in database queries and list some of the well known problems, which could be directly reduced to SCRR. We start with two simple examples to illustrate shared constraint queries in database:

1. National Climatic Data Center contains data for various geographic locations. Sustained wind speed and gust wind speed are related to the mean wind speed for a particular time. Suppose we want to retrieve the stations having ($sustained\_wind\_speed, gust\_wind\_speed$) satisfying criteria 1: $mean\_wind\_speed < sustained\_wind\_speed < max\_wind\_speed$ and criteria 2: $gust\_wind\_speed < mean\_wind\_speed$. Here $mean\_wind\_speed$ and max_wind_speed comes as query parameters. Note that both these criteria have one constraint shared, thus effectively reducing number of independent constraints by one. By representing each station as the 2-dimensional point ($sustained\_wind\_speed$, $gust\_wind\_speed$), this query translates into the orthogonal range query specified by the

(unbounded) axis-aligned rectangle $[mean\_wind\_speed : max\_wind\_speed] \times (-\infty : mean\_wind\_speed]$.

2. Consider the world data bank which contains data for Gross domestic product (gdp), and we are interested in those countries that have gdp within the range of minimum and maximum gdp among all countries and gdp growth is greater than certain proportion of the minimum gdp. Our query might look like: $min\_gdp < gdp < max\_gdp$ and $c \times min\_gdp < gdp\_growth$, where c is a constant. Here min_gdp and max_gdp comes as query parameters. The constraint on gdp_growth is proportional to the lower constraint of gdp, which means the number of independent constraint is only two. This query can be similarly converted to orthogonal range reporting problem by representing each country as the point (gdp, gdp_growth), and asking to report all the points contained in the (unbounded) axis-aligned rectangle $[min\_gdp : max\_gdp] \times [c \times min\_gdp : \infty)$.

We can take advantage of such sharing among constraints to construct more effective data structure for query answering. This serves as a motivation for SCRR data structures. Below we show the relation between SCRR and some well known problems.

Colored Range Reporting. In colored range reporting, we are given an array A, where each element is assigned a color, and each color has a priority. For a query $[a, b]$ and a threshold c (or a parameter K) we have to report all distinct colors with priority $\geq c$ (or K colors with highest priority) within $A[a, b]$ [15]. We use the chaining idea by muthukrishnan [18] to reduce the colored range reporting to SCRR problem.

We map each element $A[i]$ to a weighted point (x_i, y_i) such that (1) $x_i = i$, (2) y_i is the highest $j < i$ such that both $A[i]$ and $A[j]$ have the same color (if such a y_i does not exist then $y_i = -\infty$) and (3) its weight w_i is same as the priority of color associated with $A[i]$. Then, the colored range reporting problem is equivalent to the following SCRR problem: report all points in $[a, b] \times (-\infty, a)$ with weight $\geq c$. By maintaining a additional linear space structure, for any given a, b and K, a threshold c can be computed in constant time such that number of colors reported is at least K and at most $\Omega(K)$ (we defer details to the full version). Then, by finding the Kth color with highest color among this (using selection algorithm) and filtering out colors with lesser priority, we shall obtain the top-K colors in additional $O(K/B)$ I/Os or $O(K)$ time.

Document Retrieval Problems. In string databases or in string retrieval systems, we have a collection $\mathcal{D}$ of documents (strings) of total length N. Define $score(P, d)$, the *score* of a document d with respect to a pattern P, which is a function of the locations of all P's occurrences in d. Then our goal is to preprocess $\mathcal{D}$ and maintain a structure such that, given a query pattern P and a threshold c, all those documents d_i with $score(P, d_i) \geq c$ can be retrieved efficiently. Hon et. al. [14] showed that the document retrieval problem can be reduced to the following problem: Given a collection of N intervals (y_i, x_i) with weights w_i and a query (a, b, c), output all the intervals such that $y_i \leq a \leq x_i \leq b$ and $w_i \geq c$. This is precisely the SCRR problem that we have investigated in this article.

Range Skyline Queries. Given a set S of N points in two-dimensions, a point (x_i, y_i) is said to be dominated by a point (x_j, y_j) if $x_i < x_j$ and $y_i < y_j$. Skyline of S is subset of S which consists of all the points in S which are not dominated by any other point in S. In Range-Skyline problem, the emphasis is to quickly generate those points within a query region R, which are not dominated by any other point in R. There exists optimal solutions

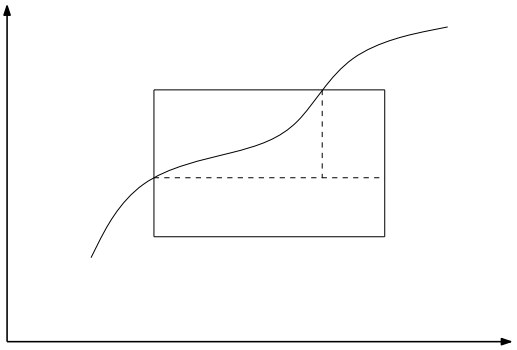

Figure 1 Special Two-dimensional Range Reporting Query.

in internal as well as external memory models for the case where R is a three-sided region of the form $[a, b] \times [c, +\infty)$ [16, 8].

We can reduce the range skyline query to SCRR by mapping each two-dimensional input point $p_i = (x_i, y_i)$ to a three-dimensional point x'_i, y'_i, z'_i as follows: (1) $x'_i = x_i$, (2) y'_i is the the x-coordinate of the leftmost point dominating p_i and (3) $z'_i = y_i$. Then range skyline query with three-sided region $[a, b] \times [c, +\infty)$ as input can be answered by reporting the output of SCRR query $[a, b] \times (-\infty, a] \times [c, +\infty)$.

Two-dimensional Range Reporting. Even though general four-sided queries are known to be hard as noted earlier, we can efficiently answer "special" four-sided queries efficiently. Any four-sided query with query rectangle R with one of its corners on the line $x = y$ can be viewed as a SCRR query. In fact any query rectangle R which intersect with $x = y$ line (or a predefined monotonic curve) can be reduced to SCRR (Figure 1).

3 Rank-Space Reduction of Points

We use rank-space reduction on the given point-set. Although rank-space reduction does not save any space for our data structure, it helps to avoid predecessor/successor search while querying and facilitate our partitioning technique. Without loss of generality, we assume that the points $p_i = (x_i, y_i, z_i) \in \mathcal{P}$ satisfy the following conditions: $x_i \leq x_{i+1}$ for all $i \in [1, N-1]$ and also $y_i \leq x_i$ for all $i \in [1, N]$. Note that $x_i \leq x_{i+1}$ can be ensured by sorting the point-set with respect to their x-coordinates and any point not satisfying $y_i \leq x_i$ can not be answer of our SCRR query, so we can remove them from consideration. In this section, we describe how to transform each point $p_i = (x_i, y_i, z_i) \in \mathcal{P}$ to a point $p'_i = (x'_i, y'_i, z'_i) \in \mathcal{P}'$ with the following additional properties guaranteed:

- Points in $\mathcal{P}'$ are on an $[1, N] \times [1, N] \times [1, N]$ grid (i.e., $x_i, y_i, z_i \in [1, N]$)
- $x'_i < x'_{i+1}$ for all $i \in [1, N-1]$. If $y_i \leq y_j$ (resp., $z_i \leq z_j$), then $y'_i \leq y'_j$ (resp., $z'_i \leq z'_j$) for all $i, j \in [1, N-1]$.

Such a mapping is given below: (1) The x-coordinate of the transformed point is same as the rank of that point itself. i.e., $x'_i = i$ (ties are broken arbitrarily), (2) Let $y_i \in (x_{k-1}, x_k]$, then $y'_i = k$, (3) Replace each z_i by the size of the set. i.e., $z'_i = \{j | z_j \leq z_i, j \in [1, N]\}$. We now prove the following lemma.

▶ **Lemma 5.** *If there exists an $S(N)$-space structure for answering SCRR queries on $\mathcal{P}'$ in optimal time in RAM model (or I/Os in external memory), then there exists an $S(N)+O(N)$-space structure for answering SCRR queries on $\mathcal{P}$ in optimal time (or I/Os).*

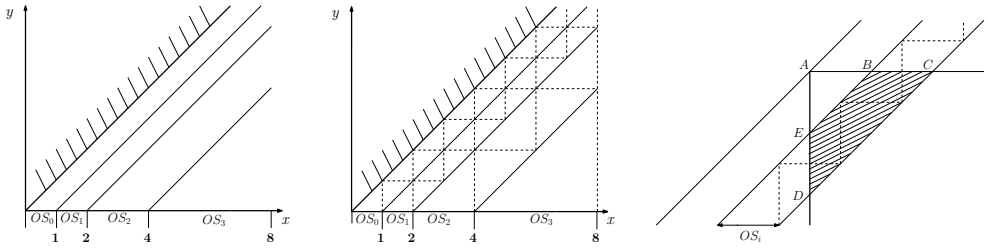

Figure 2 Point partitioning schemes: (a) Oblique slabs (b) Step partitions.

Proof. Assume we have an $S(N)$ space structure for SCRR queries on $\mathcal{P}'$. Now, whenever a query $Q_{\mathcal{P}}(a, b, c)$ comes, our first task is to identify the parameters a', b' and c' such that a point p_j is an output of $Q_{\mathcal{P}}(a, b, c)$ if and only if p'_j is an output of $Q_{\mathcal{P}}(a', b', c')$ and vice versa. Therefore, if point p'_j is an output for $Q_{\mathcal{P}}(a', b', c')$, we can simply output p_j as an answer to our original query. Based on our rank-space reduction, a', b' and c' are given as follows: (1) $x_{a'-1} < a \leq x_{a'}$ (assume $x'_0 = 0$), (2) $x_{b'} \leq b < x_{b'+1}$ (assume $x'_{N+1} = N + 1$), (3) Let z_j be the successor of c, then $c' = z'_j$.

By maintaining a list of all points in $\mathcal{P}$ in the sorted order of their x-coordinate values (along with a B-tree or binary search over it), we can compute a' and b' in $O(\log N)$ time(or $O(\log_B N)$ I/Os). Similarly, c' can also be computed using another list, where the points in $\mathcal{P}$ are arranged in the sorted order of z-coordinate value. The space occupancy of this additional structure is $O(N)$. Notice that this extra $O(\log N)$ time or $O(\log_B N)$ I/Os is optimal if we do not assume any thing about the coordinate values of points in $\mathcal{P}$. ◀

4 The Framework

In this section we introduce a new point partitioning scheme which will allow us to reduce the SCRR query into a logarithmic number of disjoint planar 3-sided or three dimensional dominance queries. From now onwards, we assume points in $\mathcal{P}$ to be in rank-space (Section 3). We begin by proving the result summarized in following theorem.

▶ **Lemma 6.** *By maintaining an $O(|\mathcal{P}|)$-word structure, any SCRR query $Q_{\mathcal{P}}(\cdot, \cdot, \cdot)$ can be answered in $O(\log^2 N + K)$ time in the RAM model, where K is the output size.*

For simplicity, we treat each point $p_i \in \mathcal{P}$ as a weighted point (x_i, y_i) in an $[1, N] \times [1, N]$ grid with z_i as its weight. The proposed framework utilizes divide-and-conquer technique based on the following partitioning schemes:

- *Oblique Slabs*: We partition the $[1, N] \times [1, N]$ grid into multi-slabs $OS_0, OS_1, ..., OS_{\lceil \log N \rceil}$ induced by lines $x = y + 2^i$ for $i = 0, 1, ...\lceil \log N \rceil$ as shown in figure 2(a). To be precise, OS_0 is the region between the lines $x = y$ and $x = y+1$ and OS_i for $i = 1, 2, 3, ..., \lceil \log N \rceil$ be the region between lines $x = y + 2^{i-1}$ and $x = y + 2^i$.
- *Step Partitions:* Each slab OS_i for $i = 1, 2, ...$ is further divided into regions with right-angled triangle shape (which we call as *tiles*) using axis parallel lines $x = (2^{(i-1)} * (1 + j))$ and $y = 2^{(i-1)} * j$ for $j = 1, 2, ...$ as depicted in Figure 2(b). OS_0 is divided using axis parallel lines $x = j$ and $y = j$ for $j = 1, 2, ...$Notice that the (axis parallel) boundaries of these triangles within any particular oblique slab looks like a step function.

Our partitioning scheme ensures property summarized by following lemma.

▶ **Lemma 7.** *Any region $[a, b] \times [1, a]$ intersects with at most $O(\log N)$ tiles.*

Proof. Let ϕ_i be the area of a tile in the oblique slab OS_i. Note that $\phi_0 = \frac{1}{2}$ and $\phi_i = \frac{1}{2}(2^{i-1})^2$ for $i \in [1, \lceil \log N \rceil]$. And let A_i be the area of the overlapping region between OS_i and the query region $[a, b] \times [1, a]$. Now our task is to simply show A_i/ϕ_i is a constant for all values of i. Assume $b = n$ in the extreme case. Then the overlapping region between OS_i and $[a, n] \times [1, a]$ will be trapezoid in shape and its area is given by $\phi_{i+1} - \phi_i$ (See Figure 2(c) for a pictorial proof). Therefore number of tiles needed for covering this trapezoidal region is $A_i/\phi_i = O(1)$. Which means the entires region can be covered by $O(\log N)$ tiles ($O(1)$ per oblique slab). ◀

In the light of the above lemma, a given SCRR query $Q_{\mathcal{P}}(a, b, c)$ can be decomposed into $O(\log N)$ subqueries of the type $Q_{\mathcal{P}_t}(a, b, c)$. Here $\mathcal{P}_t$ be the set of points within the region covered by a tile t. In the next lemma, we show that each of the $Q_{\mathcal{P}_t}(a, b, c)$ can be answered in optimal time (i.e., $O(\log |\mathcal{P}_t|)$ plus $O(1)$ time per output). Therefore, in total $O(N)$-space, we can maintain such structures for every tile t with at least one point within it. Then by combining with the result in lemma 7, the query $Q_{\mathcal{P}}(a, b, c)$ can be answered in $O(\log N * \log N + K) = O(\log^2 N + K)$ time, and lemma 6 follows.

▶ **Lemma 8.** *Let $\mathcal{P}_t$ be the set of points within the region covered by a tile t. Then a SCRR query $Q_{\mathcal{P}_t}(a, b, c)$ can be answered in $O(\log |\mathcal{P}_t| + k)$ time using a linear-space (i.e., $O(|\mathcal{P}_t|)$ words) structure, where k is the output size.*

Proof. The first step is to maintain necessary structure for answering all possible axis aligned three-dimensional dominance queries over the points in $\mathcal{P}_t$, which takes linear-space (i.e., $O(|\mathcal{P}_t|)$ words or $O(|\mathcal{P}_t| \log |\mathcal{P}_t|)$ bits). Let α and β be the starting and ending position of the interval obtained by projecting tile t to x-axis (see Figure 3). Then if the tile t intersects with the query region $[a, b] \times [1, a]$, then we have the following cases (see Figure 3):

1. $\alpha \leq a \leq \beta \leq b$: In this case, all points in $p_i \in \mathcal{P}_t$ implicitly satisfy the condition $x_i \leq b$. Therefore $Q_{\mathcal{P}_t}(a, b, c)$ can be obtained by a three sided query with $[a, N] \times [1, a] \times [c, N]$ as the input region or a two dimensional dominance query with $[a, N] \times [1, N] \times [c, N]$ as the input region (Figure 3(a)).
2. $a \leq \alpha \leq \beta \leq b$: In this case, all points in $p_i \in \mathcal{P}_t$ implicitly satisfy the condition $x_i \in [a, b]$. Therefore, $Q_{\mathcal{P}_t}(a, b, c)$ can be obtained by a two dimensional dominance query with $[1, N] \times [1, a] \times [c, N]$ as the input region (Figure 3(b)).
3. $a \leq \alpha \leq b \leq \beta$: In this case, all points in $p_i \in \mathcal{P}_t$ implicitly satisfy the condition $x_i \geq a$. Therefore $Q_{\mathcal{P}_t}(a, b, c)$ can be obtained by a three dimensional dominance query with $[1, b] \times [1, a] \times [c, N]$ as the input region (Figure 3(c)).
4. $\alpha \leq a \leq b \leq \beta$: Notice that the line between the points (a, a) and (b, a) are completely outside (and above) the tile t. Therefore, all points in $p_i \in \mathcal{P}_t$ implicitly satisfy the condition $y_i \leq a$. Therefore, $Q_{\mathcal{P}_t}(a, b, c)$ can be obtained by a three sided query with $[a, b] \times [1, N] \times [c, N]$ as the input region (Figure 3(d)).

Note that tiles can have two orientations. We have discussed four cases for one of the tile orientations. Cases for other orientation is mirror of the above four cases and can be handled easily. ◀

5 Towards $O(\log N + K)$ Time Solution

Our result in lemma 6 is optimal for $K \geq \log^2 N$. In this section, we take a step forward to achieve more efficient time solution for smaller values of K using multi-slab ideas. Using a parameter Δ (to be fixed later), we partition the $[1, N] \times [1, N]$ grid into $L = \lceil N/\Delta \rceil$ vertical

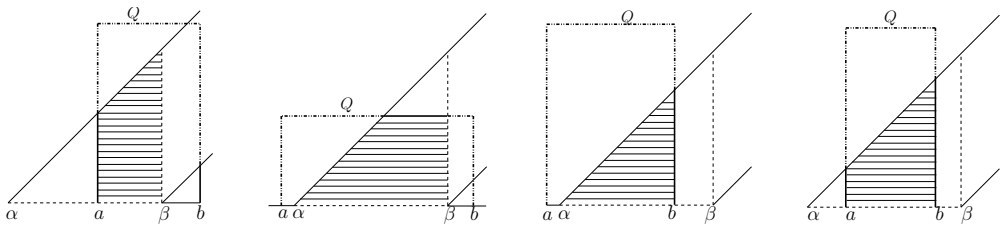

Figure 3 $Q_{\mathcal{P}}(a, b, c)$ and tile t intersections.

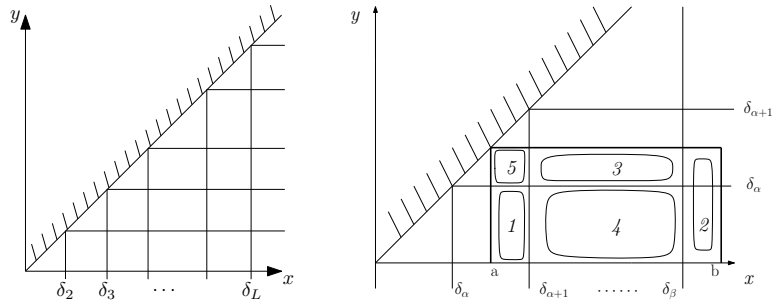

Figure 4 Divide-and-conquer scheme using Δ

slabs (Figure 4(a)). Multi-slabs $VS_0, VS_1, ..., VS_L$ are the slabs induced by lines $x = i\Delta$ for $i = 0, 1, ..., L$. Denote by δ_i $(i \in [0, L])$ the minimum x-coordinate in VS_i. For notational convenience, we define $\delta_{L+1} = \infty$. By slight abuse of notation, we use VS_i to represent the set of points in the corresponding slab.

A query $Q_{\mathcal{P}}$ with (a, b, c) as an input is called *inter-slab* query if it overlaps two or more vertical slabs, otherwise if it is entirely contained within a single vertical slab we call it an *intra-slab* query. In this section, we propose a data structure that can answer inter-slab queries optimally.

▶ **Lemma 9.** *Inter-slab SCRR queries can be answered in $O(\log N + K)$ time in RAM model using a data structure occupying $O(N)$ words space, where K represents the number of output.*

Proof. Given a query $Q_{\mathcal{P}}(a, b, c)$ such that $a \leq b$, let α, β be integers that satisfy $\delta_\alpha \leq a < \delta_{\alpha+1}$ and $\delta_\beta \leq b < \delta_{\beta+1}$. The x interval of an inter-slab query i.e. $[a, b]$ spreads across at least two vertical slabs. Therefore, $Q_{\mathcal{P}}$ can be decomposed into five subqueries $Q^1_{\mathcal{P}}$, $Q^2_{\mathcal{P}}$, $Q^3_{\mathcal{P}}$, $Q^4_{\mathcal{P}}$ and $Q^5_{\mathcal{P}}$ as illustrated in Figure 4(b). These subqueries are defined as follows.

- $Q^1_{\mathcal{P}}$ is the part of $Q_{\mathcal{P}}$ which is in $[\delta_\alpha, \delta_{\alpha+1}) \times [1, \delta_\alpha) \times [c, N]$.
- $Q^2_{\mathcal{P}}$ is the part of $Q_{\mathcal{P}}$ which is in $[\delta_\beta, \delta_{\beta+1}) \times [1, \delta_{\alpha+1}) \times [c, N]$.
- $Q^3_{\mathcal{P}}$ is the part of $Q_{\mathcal{P}}$ which is in $[\delta_{\alpha+1}, \delta_\beta) \times [\delta_\alpha, \delta_{\alpha+1}) \times [c, N]$.
- $Q^4_{\mathcal{P}}$ is the part of $Q_{\mathcal{P}}$ which is in $[\delta_{\alpha+1}, \delta_\beta) \times [1, \delta_\alpha) \times [c, N]$.
- $Q^5_{\mathcal{P}}$ is the part of $Q_{\mathcal{P}}$ which is in $[\delta_\alpha, \delta_{\alpha+1}) \times (\delta_\alpha, \delta_{\alpha+1}) \times [c, N]$.

If $\alpha + 1 = \beta$ then we only need to consider subqueries $Q^1_{\mathcal{P}}$, $Q^2_{\mathcal{P}}$ and $Q^5_{\mathcal{P}}$. Each of these subqueries can now be answered as follows.

Answering $Q^1_{\mathcal{P}}$. The subquery $Q^1_{\mathcal{P}}$ can be answered by retrieving all points in $VS_\alpha \cap [a, N] \times [1, N]$ with weight $\geq c$. This is a two-dimensional dominance query in VS_α. This can be achieved by maintaining a three-dimensional dominance query structure for RAM model [1] for the points in VS_i for $i = 1, \ldots, L$ separately. The query time will be $O(\log |VS_\alpha| + K_1) = O(\log N + K_1)$, where K_1 is the output size and index space is $O(\sum_{i=1}^{L} |VS_i|) = O(N)$ words.

Answering $Q^2_{\mathcal{P}}$. To answer subquery $Q^2_{\mathcal{P}}$ we will retrieve all points in $VS_\beta \cap [1, b) \times [1, a)$ with weight $\geq c$. By maintaining a collection of three-dimensional dominance query structures [1] occupying linear space overall, $Q^2_{\mathcal{P}}$ can be answered in $O(\log N + K_2)$ time, where K_2 is the output size.

Answering $Q^3_{\mathcal{P}}$. To answer subquery $Q^3_{\mathcal{P}}$, we begin by partitioning the set of points $\mathcal{P}$ into L horizontal slabs $HS_1, HS_2, \ldots, HS_L$ induced by lines $y = i\Delta$, such that $HS_i = \mathcal{P} \cap [\delta_{i+1}, N] \times [\delta_i, \delta_{i+1})$. The subquery $Q^3_{\mathcal{P}}$ can now be answered by retrieving all points in $HS_\alpha \cap [1, \delta_\beta) \times [1, a)$ with weight $\geq c$. This can be achieved by maintaining a three-dimensional dominance query structure [1] for the points in HS_i for $i = 1, ..., L$ separately. Since each point in S belongs to at most one HS_i the overall space can be bounded by $O(N)$ words and the query time can be bounded by $O(\log |HS_\alpha|) + K_3) = O(\log N + K_3)$ time, with K_3 being the number of output.

Answering $Q^5_{\mathcal{P}}$. To answer subquery $Q^5_{\mathcal{P}}$ we will retrieve all points in $VS_\alpha \cap (a, N] \times [1, a)$ with weight $\geq c$. By maintaining a collection of three-dimensional dominance query structures occupying linear space overall as described in earlier subsections, $Q^5_{\mathcal{P}}$ can be answered in $O(\log |VS_\alpha| + K_5) = O(\log N + K_5)$ time, where K_5 is the output size.

Answering $Q^4_{\mathcal{P}}$. We begin by describing a naive way of answering $Q^4_{\mathcal{P}}$ by using a collection of three-dimensional dominance query structures built for answering $Q^1_{\mathcal{P}}$. We query VS_i to retrieve all the points in $VS_i \cap [1, N] \times [1, \delta_\alpha)$ with weight $\geq c$ for $i = \alpha + 1, ..., \beta - 1$. Such a query execution requires $O((\beta - \alpha + 1) \log N + K_4)$ time, where K_4 is the output size. We are required to spend $O(\log N)$ time for each vertical slab even if the query on a particular VS_i does not produce any output. To answer subquery $Q^4_{\mathcal{P}}$ in $O(\log N + K_4)$ time, we make following crucial observations: (1) All three boundaries of $Q^4_{\mathcal{P}}$ are on the partition lines, (2) The left boundary of $Q^4_{\mathcal{P}}$ (i.e., line $x = \delta_{\alpha+1}$) is always the successor of the top boundary (i.e., line $y = \delta_\alpha$), (3) The output size is bounded by $O(\log^2 N)$.

We use these observations to construct following data structure: Since the top left and bottom right corner of $Q^4_{\mathcal{P}}$ falls on the partition lines, there are at most $(N/\Delta)^2$ possible different rectangles for $Q^4_{\mathcal{P}}$. For each of these we store at most top-$O(\log^2 N)$ points in sorted order of their weight. Space requirement of this data structure is $O((N/\Delta)^2 \log^2 N)$ words. Query algorithm first identifies the rectangle that matches with $Q^4_{\mathcal{P}}$ among $(N/\Delta)^2$ rectangles and then simply reports the points with weight greater than c in optimal time. Finally to achieve linear space, we choose $\Delta = \sqrt{N} \log N$.

Thus, we can obtain $K = K_1 + K_2 + K_3 + K_4 + K_5$ output in $O(K)$ time. Also, in the divide and conquer scheme, the point sets used for answering subqueries $Q^1_{\mathcal{P}}, ..., Q^5_{\mathcal{P}}$ are disjoint, hence all reported answers are unique. Now given a query $Q_{\mathcal{P}}(a, b, c)$, if the subquery $Q^4_{\mathcal{P}}$ in the structure just described returns $K_4 = \log^2 N$ output, it suggest that output size $K > \log^2 N$. Therefore, we can query the structure in lemma 6 and still retrieve all output in optimal time. This completes the proof of lemma 9. ◀

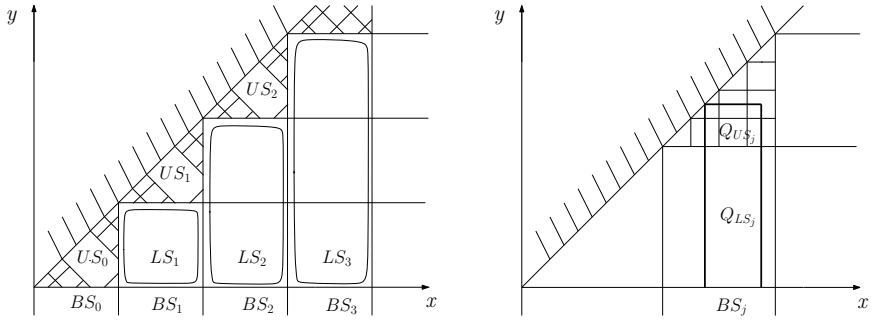

Figure 5 Optimal time SCRR query data structure.

6 Linear Space and $O(\log N + K)$ Time Data Structure in RAM Model

In this section, we show how to obtain our $O(\log N + K)$ time result stated in Theorem 2 via bootstrapping our previous data structure.

We construct $\psi_1, \psi_2, ..., \psi_{\lceil \log\log N \rceil}$ levels of data structures, where $\psi_1 = \sqrt{N}\log N$ and $\psi_i = \sqrt{\psi_{i-1}}\log\psi_{i-1}$, for $i = 2, 3, ..., \lceil \log\log N \rceil$. At each level, we use multi-slabs to partition the points. More formally, at each level ψ_i, the $[1, N] \times [1, N]$ grid is partitioned into $g = \lceil N/\psi_i \rceil$ vertical slabs or multi-slabs. At level ψ_i, multi-slabs $BS_0, BS_1, ..., BS_g$ are the slabs induced by lines $x = j\psi_i$ for $j = 0, 1, ..., g$. Each multi-slab BS_j is further partitioned into disjoint upper partition US_j and lower partition LS_j (Figure 5). Below we describe the data structure and query answering in details.

For a multi-slab BS_j and a $SCRR$ query, US_j can have more constraints than LS_j, making it more difficult to answer query on US_j. Our idea is to exclude the points of US_j from each level, and build subsequent levels based on only these points. Query answering for LS_j can be done using the inter-slab query data structure and three-sided range reporting data structure.

At each level ψ_i, we store the data structure described in Lemma 9 capable of answering inter-slab queries by taking slab width $\Delta = \sqrt{\psi_i}\log\psi_i$. Also we store separate three-sided range reporting data structures for each LS_j, $j = 0, 1, ..., g$. The points in US_j at level ψ_i (for $i = 1, \cdots, \lceil \log\log N \rceil - 1$) are removed from consideration. These removed points are considered in subsequent levels. Note that the inter-slab query data structure stored here is slightly different from the data structure described in lemma 9, since we removed the points of US_j (region 5 of figure 4b). $\psi_{\lceil \log\log N \rceil}$ is the bottom level and no point is removed from here. Level ψ_{i+1} contains only the points of all the US_j partitions from previous level ψ_i, and again the upper partition points are removed at level ψ_{i+1}. More specifically, level ψ_1 contains an inter-slab query data structure and $\sqrt{N}\log N$ number of separate two-dimensional three-sided query data structures over each of the lower partitions LS_j. Level ψ_2 contains $\sqrt{N}\log N$ number of data structures similar to level ψ_1 corresponding to each of the upper partitions US_j of level ψ_1. Subsequent levels are constructed in a similar way. No point is repeated at any level, two-dimensional three-sided query data structures and inter-slab query data structures take linear space, giving linear space bound for the entire data structure.

A SCRR query $Q_{\mathcal{P}}$ can be either an inter-slab or an intra-slab query at level ψ_i (illustrated in figure 6). An intra-slab SCRR query $Q_{\mathcal{P}}$ can be divided into Q_{US_i} and Q_{LS_i}. Q_{LS_i} is

three-sided query in LS_i, which can be answered by the three-sided range reporting structure stored at LS_i. Q_{US_i} is issued as a SCRR query for level ψ_{i+1} and can be answered by traversing at most $\lceil \log\log N \rceil$ levels. An inter-slab SCRR query $Q_{\mathcal{P}}$ can be decomposed into 5 sub-queries: Q_1, Q_2, Q_3, Q_4 and Q_5. Q_1, Q_2, Q_3 and Q_4 can be answered using the optimal inter-slab query data structure of lemma 9 in similar way described in details in section 5. Again Q_5 is issued as a SCRR query for level ψ_{i+1} and can be answered in subsequent levels. At each level $O(\log \psi_i + K_i)$ time is needed where K_i is the output size obtained at level ψ_i. Since no point is repeated at any level, all reported answers are unique. At most $\log\log N$ levels need to be accessed for answering $Q_{\mathcal{P}}$. Total time is bounded by $O(\sum_{i=1}^{\log\log N}(\log \psi_i) + \log\log N + \sum_{i=1}^{\log\log N}(K_i)) = O(\log N + K)$, thus proving theorem 2.

7 SCRR Query in External Memory

In this section we discuss two external memory data structures for SCRR query, one achieving optimal I/O and another achieving linear space. Both these data structures are obtained by modifying our RAM model data structure. We use the multi-slab ideas similar to section 5. We assume points in $\mathcal{P}$ to be in rank-space. We begin by stating external memory variants of lemma 6 and lemma 9.

▶ **Lemma 10.** *By maintaining an $O(N)$-word structure, any SCRR query $Q_{\mathcal{P}}(\cdot,\cdot,\cdot)$ can be answered in $O(\log^2(N/B) + K/B)$ I/Os, where K is the output size.*

Proof. We use $\lceil \log(N/B) \rceil$ number of oblique slabs induced by lines $x = y + 2^i B$ for $i = 0, 1, ...\lceil \log(N/B) \rceil$. Each oblique slab is partitioned into tiles using axis parallel lines $x = (2^{(i-1)} * (1 + j))B$ and $y = 2^{(i-1)} * jB$ for $j = 1, 2, ...$ It can be easily shown that any SCRR query $Q_{\mathcal{P}}(a, b, d)$ intersects with at most $O(\log(N/B))$ tiles, each of which can be resolved in linear space and optimal I/Os using three-dimensional query structure [1] in each tile, achieving $O(\log^2(N/B) + K/B)$ total I/Os. ◀

▶ **Lemma 11.** *Inter-slab SCRR queries can be answered in $O(\log_B N + K/B)$ I/Os using a data structure occupying $O(N)$ words space, where K represents the number of outputs.*

Proof. This can be achieved by using a data structure similar to the one described in lemma 9 with $\Delta = \sqrt{NB}\log_B N$. We use external memory counterparts for three sided and three dimensional dominance reporting and for answering query Q_p^4 we maintain top $O(log_B^2 N)$ points from each of the rectangle. ◀

7.1 Linear Space Data Structure

The linear space data structure is similar to the RAM model linear space and optimal time structure described in section 6. Major difference is we use $\psi_i = \sqrt{\psi_{i-1}B}\log_B \psi_{i-1}$ for bootstrapping and use the external memory counterparts of the data structures.

We construct $\psi_1, \psi_2, ..., \psi_{\lceil \log\log N \rceil}$ levels of data structures, where $\psi_1 = \sqrt{NB}\log_B N$ and any $\psi_i = \sqrt{\psi_{i-1}B}\log_B \psi_{i-1}$. At each level ψ_i, the $[1, N] \times [1, N]$ grid is partitioned into $g = \lceil N/\psi_i \rceil$ vertical slabs. Multi-slabs BS_j, upper partition US_j and lower partition LS_j for $j = 0, 1, ..., g$ are defined similar to section 6.

At each level ψ_i, we store the data structure described in Lemma 11 capable of answering inter-slab queries in optimal I/Os by taking slab width $\Delta = \sqrt{\psi_i B}\log_B \psi_i$. Also we store separate three-sided external memory range reporting data structures for each LS_j, $j = 0, 1, ..., g$. In order to maintain linear space, the points in US_j at level ψ_i (for $i = 1, \cdots, \lceil \log\log N \rceil - 1$)

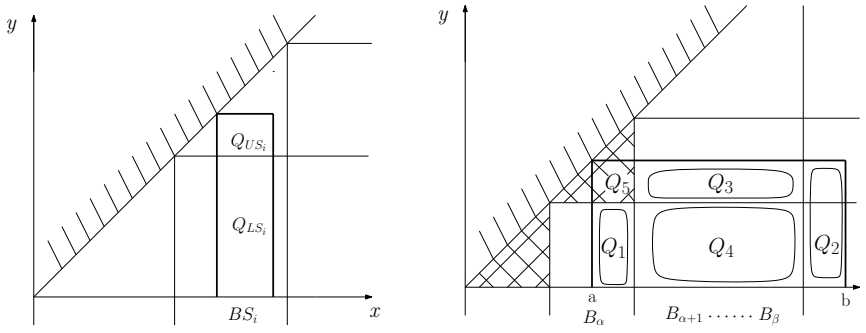

Figure 6 Intra-slab and Inter-slab query for linear space data structure.

are removed from consideration. These removed points are considered in subsequent levels. $\psi_{\lceil \log\log N \rceil}$ is the bottom level and no point is removed from here. Level ψ_{i+1} contains only the points of all the US_j partitions from previous level ψ_i, and again the upper partition points are removed at level ψ_{i+1}. External memory two-dimensional three-sided query data structures and optimal inter-slab query data structures for external memory take linear space, and we avoided repetition of points in different levels, thus the overall data structure takes linear space.

Query answering is similar to section 6. If the SCRR query $Q_{\mathcal{P}}$ is intra-slab, then $Q_{\mathcal{P}}$ can be divided into Q_{US_i} and Q_{LS_i}. Q_{LS_i} is three-sided query in LS_i, which can be answered by the three-sided range reporting structure stored at LS_i. Q_{US_i} is issued as a SCRR query for level ψ_{i+1} and can be answered by traversing at most $\lceil \log\log N \rceil$ levels. An inter-slab SCRR query $Q_{\mathcal{P}}$ can be decomposed into 5 sub-queries: Q_1, Q_2, Q_3, Q_4 and Q_5. Q_1, Q_2, Q_3 and Q_4 can be answered using the optimal inter-slab query data structure of lemma 11 in similar way described in details in section 5. Again Q_5 is issued as a SCRR query for level ψ_{i+1} and can be answered in subsequent levels. At each level $O(\log_B \psi_i + K_i/B)$ I/Os are needed where K_i is the output size obtained at level ψ_i. Since no point is repeated at any level, all reported answers are unique. At most $\log\log N$ levels need to be accessed for answering $Q_{\mathcal{P}}$. Total I/O is bounded by $O(\sum_{i=1}^{\log\log N}(\log_B \psi_i) + \log\log N + \sum_{i=1}^{\log\log N}(K_i/B))$ $= O(\log_B N + \log\log N + K/B)$ thus proving theorem 3.

7.2 I/O Optimal Data Structure

I/O optimal data structure is quite similar to the linear space data structure. The major difference is the points in US_j at level ψ_i (for $i = 1, \cdots, \lceil \log\log N \rceil - 1$) are not removed from consideration. Instead at each US_j we store external memory data structure capable of answering inter-slab query optimally(Lemma 11). All the points of US_j $(j = 0, 1, ..., g)$ of level ψ_i (for $i = 1, \cdots, \lceil \log\log N \rceil - 1$) are repeated in the next level ψ_{i+1}. This will ensure that we will have to use only one level to answer the query. For LS_j $(j = 0, 1, ..., g)$, we store three-sided query structures. Since there are $\log\log N$ levels, and at each level data structure uses $O(N)$ space, total space is bounded by $O(N \log\log N)$. To answer a query $Q_{\mathcal{P}}$, we query the structures associated with ψ_i such that $Q_{\mathcal{P}}$ is an intra-slab query for ψ_i and is inter-slab for ψ_{i+1}. We can decompose the query $Q_{\mathcal{P}}$ into Q_{US} and Q_{LS}, where $Q_{US}(Q_{LS})$ falls completely within $US_j(LS_j)$. Since, Q_{US} is an inter-slab query for the inter-slab query data structure stored at US_j, it can be answered optimally. Also Q_{LS} is a simple three-sided query for which output can be retrieved in optimal time using the three-sided structure of LS_j. This completes the proof for Theorem 4.

8 Conclusions

In many applications which require range queries, some of the input constraints are shared and not really independent. We give first non trivial indexes for handling such cases breaking the currently known $O(N \log^{\epsilon} N)$ space barrier for four-sided queries in Word-RAM model. In Word-RAM model, we obtained linear space and optimal time index for answering SCRR queries. Our optimal I/O index in external memory takes $O(N \log \log N)$ words of space and answer queries optimally. We also present a linear space index for external memory. We leave it as an open problem to achieve optimal space bounds, avoiding the $O(\log \log N)$ blowup in external memory model. Also it will be interesting to see whether such results can be obtained in Cache Oblivious model.

Acknowledgements. This work is supported by US NSF Grants CCF–1017623, CCF–1218904.

References

1 Peyman Afshani. On dominance reporting in 3d. In *ESA*, pages 41–51, 2008.

2 Peyman Afshani, Lars Arge, and Kasper Dalgaard Larsen. Orthogonal range reporting in three and higher dimensions. In *FOCS*, pages 149–158, 2009.

3 Peyman Afshani, Lars Arge, and Kasper Dalgaard Larsen. Orthogonal range reporting: query lower bounds, optimal structures in 3-d, and higher-dimensional improvements. In *Symposium on Computational Geometry*, pages 240–246, 2010.

4 Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *FOCS*, pages 198–207, 2000.

5 Lars Arge, Mark de Berg, Herman J. Haverkort, and Ke Yi. The priority r-tree: A practically efficient and worst-case optimal r-tree. In *SIGMOD Conference*, pages 347–358, 2004.

6 Lars Arge, Vasilis Samoladas, and Jeffrey Scott Vitter. On two-dimensional indexability and optimal range search indexing. In *PODS*, pages 346–357, 1999.

7 Jon Louis Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980.

8 Gerth Stølting Brodal and Kasper Green Larsen. Optimal planar orthogonal skyline counting queries. *CoRR*, abs/1304.7959, 2013.

9 Timothy M. Chan, Kasper Green Larsen, and Mihai Patrascu. Orthogonal range searching on the ram, revisited. In *Symposium on Computational Geometry*, pages 1–10, 2011.

10 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17(3):427–462, 1988.

11 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17(3):427–462, 1988.

12 Bernard Chazelle. Lower bounds for orthogonal range searching i. the reporting case. *J. ACM*, 37(2):200–212, 1990.

13 Bernard Chazelle. Lower bounds for orthogonal range searching ii. the arithmetic model. *J. ACM*, 37(3):439–463, 1990.

14 Wing-Kai Hon, Rahul Shah, Sharma V. Thankachan, and Jeffrey Scott Vitter. Space-efficient frameworks for top-k string retrieval. *J. ACM*, 61(2):9, 2014.

15 Marek Karpinski and Yakov Nekrich. Top-k color queries for document retrieval. In *SODA*, pages 401–411, 2011.

16 Casper Kejlberg-Rasmussen, Yufei Tao, Konstantinos Tsakalidis, Kostas Tsichlas, and Jeonghun Yoon. I/o-efficient planar range skyline and attrition priority queues. In *PODS*, pages 103–114, 2013.

17 Kasper Green Larsen and Rasmus Pagh. I/o-efficient data structures for colored range and prefix reporting. In *SODA*, pages 583–592, 2012.

18 S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 657–666, 2002.

19 Yakov Nekrich. External memory range reporting on a grid. In *ISAAC*, pages 525–535, 2007.

20 Darren Erik Vengroff and Jeffrey Scott Vitter. Efficient 3-d range searching in external memory. In *STOC*, pages 192–201, 1996.