Efficient Range Searching for Categorical and Plain Data

Yakov Nekrich, University of Kansas

In the orthogonal range searching problem we store a set of input points S in a data structure; the answer to a query Q is a piece of information about points in $Q \cap S$, e.g., the list of all points in $Q \cap S$ or the number of points in Q. In the colored (or categorical) range searching problem the set of input points is partitioned into categories; the answer to a query is a piece of information about categories of points in a query range.

In this paper we describe several new results for one- and two-dimensional range searching problem. We obtain an optimal adaptive data structure for counting the number of objects in a three-sided range and for counting categories of objects in a one-dimensional range. We also obtain new results on color range reporting in two dimensions, approximate color counting in one dimension and some other related problems.

Categories and Subject Descriptors: F2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*

Additional Key Words and Phrases: Data Structures, Orthogonal Range Reporting, Colored Range Reporting

1. INTRODUCTION

The problem of reporting all tuples that belong to a specified query range is a fundamental problem in the database theory. This task can be also formulated as the orthogonal range reporting problem extensively studied in the data structures and computational geometry communities: a set of points S is stored in a data structure; for any query rectangle Q we should report all points $p \in S$ that are in Q. In many situations input objects are classified into categories, and we may be interested in listing only categories of objects that occur in the query range. Such scenarios can be modeled by the colored (or categorical) range reporting problem: each point p in a set S is assigned a color $col(p)$; for a query rectangle Q we must report the colors of all points that occur in Q. Data structures for regular range reporting queries can obviously answer colored reporting queries by listing all points in the query range and finding distinct colors of those points. However, the efficiency of such solution can be arbitrarily bad because the number of object categories can be significantly smaller than the total number of objects in the query range. Output-sensitive solutions of the colored reporting problem were considered in a number of papers, e.g., [Janardan and Lopez 1993; Bozanis et al. 1995; Gupta et al. 1995; 1996; Bozanis et al. 1997; Muthukrishnan 2002; Nanopoulos and Bozanis 2003; Karpinski and Nekrich 2011].

In this paper we present I/O-efficient data structures for two-dimensional colored range reporting with guaranteed worst-case performance. Our solutions achieve optimal query costs and almost the same space usage as the corresponding data structures

Author's address: Y. Nekrich, Department of Electrical Engineering and Computer Science, University of Kansas.
Part of this work was done while the author was visiting Laboratoire d'Informatique Gaspard Monge, Université Paris-Est & CNRS.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

for regular range reporting queries. We also show how the ideas used in our color reporting data structures can be used to answer counting queries for categorical and plain data.

Counting tuples or categories of tuples in a query range is another fundamental problem. This task can be modeled by orthogonal range counting (respectively by colored orthogonal range counting) problem. In the orthogonal range counting problem, we keep a set of points S in a data structure so that for any query rectangle Q the number of points in $Q \cap S$ can be computed. In the categorical range counting problem categories of points in a query range must be counted. By the lower bound in [Pătraşcu 2007], the best previously described data structures for two-dimensional range counting achieve optimal query time. However, we can obtain better results when the number of objects that must be counted is small. In the adaptive range counting problem, the query cost is a function of the total number of objects in the query range. We present adaptive data structures for counting colors in a one-dimensional query range and for counting points in a two-dimensional range that is bounded on three sides and achieve optimal time for this problems.

In some cases the exact answer to a counting query is not important; it suffices to obtain an estimate on the number of objects or object categories in the query range. In this paper we show that we can obtain an estimate on the number of object categories in a one-dimensional query range using a linear space data structure. We also show how approximate answers for some important special cases of two-dimensional counting queries can be given by linear- or almost-linear space data structures. All presented results share a common technique that will be described later in this paper.

Applications. A typical application of colored range reporting is a database that stores a set of locations on a plane and the type of soil for each location; given a two-dimensional rectangular area, we want to find the types of soil that occur in this area. Colored range reporting queries are also important in the context of text databases and databases that contain non-text strings, such as DNA sequences. Frequently we need to find documents that contain a certain string P, but documents that contain multiple occurrences of P must be reported only once. Moreover, we may ask for documents that contain P and satisfy some additional requirements. Such queries can be reduced to colored reporting queries and their variants, see e.g., [Muthukrishnan 2002; Gagie et al. 2010; Karpinski and Nekrich 2011; Nekrich and Navarro 2012]. If we only want to count the number of relevant categories (documents containing a given pattern, types of soil in locations from a given range among all locations in a geo-information system, etc) instead of enumerating them, then color range counting problem provides an appropriate model.

Color Range Reporting: Previous work. Colored range reporting queries and their variants were extensively studied in the main memory model, see e.g., [Janardan and Lopez 1993; Gupta et al. 1995; Bozanis et al. 1995; 1997; Agarwal et al. 2002; Muthukrishnan 2002; Karpinski and Nekrich 2008]. For instance, the data structures described in [Gupta et al. 1995; Janardan and Lopez 1993] support two-dimensional colored range reporting queries in $O(\log N + K)$ time[1] and use $O(N \log^2 N)$ words of space. In [Janardan and Lopez 1993] the authors also described another structure that uses $O(N \log N)$ words, but needs $O(\log^2 N + K)$ time to answer a query. Henceforth K denotes the number of elements in the answer and N is the total number of points in the data structure. Agarwal *et al.* [Agarwal et al. 2002] considered the two-dimensional colored reporting problem in the case when points lie on a $U \times U$ grid, i.e. all point coordinates are positive integers bounded by parameter U. In [Agarwal

[1]In this paper $\log N$ denotes the binary logarithm of N if the logarithm base is not specified.

et al. 2002], the authors described a data structure that uses $O(N \log^2 N)$ words of space and supports queries in $O(\log\log U + K)$ time. Three-sided queries are a special case of two-dimensional queries when the query range $Q = [a, b] \times [0, c]$ is bounded on three sides. The data structure of [Agarwal et al. 2002] supports three-sided queries in $O(\log\log U + K)$ time and uses $O(N \log N)$ space. The data structure of Shi and JáJá [Shi and JáJá 2005] answers queries in $O(\log N + K)$ time and uses optimal linear space. They also obtained an $O(N \log N)$ space structure for general two-dimensional queries with the same query time.

Even in the main memory, there are significant complexity gaps between the data structures that report points and data structures that report colors. For instance, there are data structures for three-sided and two-dimensional point reporting that use $O(N)$ and $O(N \log^{\varepsilon} N)$ words for any $\varepsilon > 0$ respectively [McCreight 1985; Chazelle 1988]. Moreover, three-sided queries can be answered in $O(K)$ time and linear space when points lie on an $N \times N$ grid. We observe that the case of reporting on a grid is important for database applications because in most cases properties of objects stored in a database are specified by integers bounded by a parameter U (frequently $U = N^{O(1)}$ or even $U = O(N)$).

In the external memory model [Aggarwal and Vitter 1988], the main memory consists of M words. Every I/O operation reads a *block* of B contiguous words from disk into the main memory (respectively, writes a block into external disk). The time complexity is measured in the number of I/O operations; the space usage is measured in the number of blocks occupied by the data structure on disk.

External memory data structures for reporting two-dimensional points were studied in a number of papers, see e.g., [Ramaswamy and Subramanian 1994; Subramanian and Ramaswamy 1995; Arge et al. 1999; Nekrich 2011]. The structure of Arge *et al.* [Arge et al. 1999] uses $O((N/B) \log_2 N / \log_2 \log_B N)$ blocks of space and answers range reporting queries in $O(\log_B N + K/B)$ I/Os; the space usage can be reduced to $O(N/B)$ blocks in case of three-sided queries. In the case when all points lie on a $U \times U$ grid, the data structure of Nekrich [Nekrich 2007] answers point reporting queries in $O(\log\log_B U + K/B)$ I/Os. In [Nekrich 2007] 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[2] for any constant $h > 0$. Recently, Larsen and Pagh [Larsen and Pagh 2012] showed that three-sided point reporting queries can be answered in $O(K/B)$ I/Os using $O(N/B)$ blocks of space.

Using the approach of [Gupta et al. 1995], we can answer one-dimensional color reporting queries using a data structure for three-sided point reporting queries. Hence, all results for three-sided reporting can be transformed into structures for color reporting in one dimension. Heuristic solutions for colored range reporting in $d > 1$ dimensions are described in [Nanopoulos and Bozanis 2003]. We are not aware of any external memory structures for colored reporting in $d > 1$ dimensions with worst-case performance guarantees that were published prior to [Nekrich 2012].

Color Range Reporting: Our Results. We present several efficient data structures for three-sided and two-dimensional colored reporting problem. These results were also reported in [Nekrich 2012]. We describe a $O(N/B)$ space data structure that answers three-sided reporting queries in $O((N/B)^{\delta} + K/B)$ I/Os for any constant $\delta > 0$. This is the first linear space data structure for this problem in external memory. Furthermore we address the problem of reporting colors in the case when all points lie on an $N \times N$ grid. We describe a data structure that answers three-

[2]Here $\log_B^{(h)} N = \log_B(\log_B^{(h-1)} N)$ for $h > 1$ and $\log_B^{(1)} N = \log_B N$.

Table I. Our results and selected previous results for three-sided and two-dimensional colored reporting.

Ref.	Query Type	Space Usage	Query Cost	Grid	Memory Model
[Gupta et al. 1995], [Janardan and Lopez 1993]	Two-Dimensional	$O((N/B)\log^2 N)$	$O(\log N + K)$		Main
[Janardan and Lopez 1993]	Two-Dimensional	$O((N/B)\log N)$	$O(\log^2 N + K)$		Main
[Agarwal et al. 2002]	Three-Sided	$O((N/B)\log N)$	$O(\log\log U + K)$	$U \times U$	Main
[Agarwal et al. 2002]	Two-Dimensional	$O((N/B)\log^2 N)$	$O(\log\log U + K)$	$U \times U$	Main
[Shi and JáJá 2005]	Three-Sided	$O(N/B)$	$O(\log N)$		Main
[Shi and JáJá 2005]	Two-Dimensional	$O((N/B)\log N)$	$O(\log N)$		Main
New	Three-Sided	$O(N/B)$	$O((N/B)^\delta + K/B)$		External
New	Three-Sided†	$O((N/B)\log\log N \log^{(3)} N)$	$O(K/B)$	$N \times N$	External
New	Three-Sided†	$O((N/B)\log\log N)$	$O((\log\log N)^\delta + K/B)$	$N \times N$	External
New	Three-Sided†	$O((N/B)\log\log N \log^{(3)} N)$	$O(\log\log_B U + K/B)$	$U \times U$	External
New	Two-Dimensional†	$O((N/B)\log N \log\log N \log^{(3)} N)$	$O(\log\log_B U + K/B)$	$U \times U$	External
New	Three-Sided	$O(N/B)$	$O(1 + K)$	$N \times N$	Main
New	Two-Dimensional	$O(N/B)$	$O(\log\log U + K)$	$U \times U$	Main
[Larsen and van Walderveen 2013]	Three-Sided	$O((N/B)\log^* N)$	$O(1 + K/B)$	$N \times N$	External
[Larsen and van Walderveen 2013]	Three-Sided	$O(N/B)$	$O(\log^{(f)} N + K/B)$	$N \times N$	External
[Larsen and van Walderveen 2013]	Three-Sided	$O(N/B)$	$O(1 + K)$	$N \times N$	Main

Note: For convenience, the space usage of main memory data structures is measured in blocks of size B. Data structures marked with † produce a list of colors that may contain duplicates; see a remark after Lemma 3.1.

sided queries in optimal $O(K/B)$ I/O operations and uses $O((N/B)\log\log N\log^{(3)} N)$ blocks of space, where $\log^{(3)} N = \log\log\log N$. We also describe another data structure that uses $O((N/B)\log\log N)$ blocks of space, but answers three-sided queries in $O((\log\log N)^\delta + K/B)$ I/Os for any constant $\delta > 0$. Our results can be extended to queries on a $U \times U$ grid. We show that there is a data structure that answers three-sided colored reporting queries on a $U \times U$ grid in $O(\log\log_B U + K/B)$ I/Os and uses $O((N/B)\log\log N\log^{(3)} N)$ blocks of space. We also show that there is a data structure that answers general two-dimensional queries in $O(\log\log_B U + K/B)$ I/Os and uses $O((N/B)\log N\log\log N\log^{(3)} N)$ blocks of space. By the predecessor lower bound [Pătraşcu and Thorup 2006], any data structure stored in $O((N/B)\log^{O(1)} N)$ blocks of space needs $\Omega(\log\log_B U + K/B)$ I/Os to answer an orthogonal range reporting query. All our results are listed in Table I. We observe that our results for colored range reporting on a grid can be extended to the case of arbitrary point coordinates using a standard technique [Gabow et al. 1984]; the space usage of all data structures remains unchanged and the query cost increases by an additive $O(\log_B N)$ factor.

Thus we achieve optimal query cost for three-sided colored reporting on an $N \times N$ grid and for two-dimensional colored range reporting on a $U \times U$ grid. Even in the main memory, the fastest previous structure [Agarwal et al. 2002] needs $O(\log\log N)$ time and $O(N\log N)$ words of space to answer three-sided queries on an $N \times N$ grid. The space usage of our structure for three-sided queries is close to optimal $O(N/B)$ blocks. It was shown in [Arge et al. 1999] that any data structure for regular range reporting which answers queries in $\log_B^{O(1)} N + O(K/B)$ I/O operations uses $\Omega((N/B)\log N/\log\log_B N)$ blocks of space. Hence, the space usage of our two-dimensional data structure is almost optimal (i.e., differs from optimal by a very small $O(\log\log N \cdot \log\log_B N\log^{(3)} N)$ factor). Even in comparison with the previous fastest main memory structures, we improve the space usage almost by a logarithmic factor.

In this extended version of [Nekrich 2012] we describe further results that can be obtained using similar techniques. We describe a main memory data structure that answers three-sided color reporting queries in $O(1 + K)$ time and uses $O(N)$ words of space. This result is valid in the word RAM model of computation that is arguably one of the most realistic and most widely studied main memory models. We can also extend this to support general two-dimensional queries in $O(\log\log U)$ time and $O(N\log N)$ space.

Color Range Reporting: Recent Parallel Work. Very recently, some new results on three-sided color reporting on $N \times N$ grid were achieved [Larsen and van Walderveen 2013]. In [Larsen and van Walderveen 2013] the authors describe a data structure that either uses $O(N/B)$ blocks and answers queries in $O(\log^{(f)} N + K/B)$ I/Os or uses $O((N/B)\log^* N)$ blocks of space and answers queries in $O(1 + K/B)$ I/Os. Besides, they also describe an $O(N)$-word data structure that answers three-sided queries in $O(1 + K)$ time in the word RAM model. These results can also be extended (with $O(\log N)$-factor increase in space usage) to two-dimensional queries. For comparison, their results for three-sided queries are included into Table I. While the results of Larsen and van Walderveen [Larsen and van Walderveen 2013] subsume our results for the color range reporting problem, their data structures rely on a technique that is quite similar to the path-range trees approach of our paper. The path-range tree data structure will be introduced in section 3; it can also be applied to other important problems described below.

Approximate Range Counting. Furthermore, we describe several new data structures for counting colored and uncolored objects in the query range. In the approximate range counting problem, we must approximate the number of objects (points, colors) in the query range. Data structures for approximate two-dimensional point counting were described in [Nekrich 2009] and, very recently, in [Chan and Wilkinson 2013]. The data structure of Chan and Wilkinson [Chan and Wilkinson 2013] uses $O(N)$ or $O(N \log\log N)$ words and answers approximate point counting queries in $O(\log^{\varepsilon} N)$ or $O(\log\log N)$ time. The structure in [Nekrich 2009] needs more space but provides a better approximation on the number of points in $O(\log\log N)$ time. We observe that $O(\log\log N)$ query time is optimal because we can use approximate counting to answer range emptiness queries, i.e. to decide whether a query range Q contains at least one point stored in the data structure. By the lower bound of [Pătraşcu and Thorup 2006] and the reduction of [Miltersen et al. 1998] we need $\Omega(\log\log N)$ time to answer a range emptiness query with an $O(N \log^f N)$-word data structure for any constant f.

On the other hand it is known that three-sided emptiness queries on $N \times N$ grid can be answered in $O(1)$ time. In this paper we show that the time necessary to answer an approximate three-sided counting query can be further reduced and describe an optimal data structure for this important special case. Our data structure uses linear space and supports approximate point counting queries on $N \times N$ grid in $O(1)$ I/Os.

We also describe a data structure for approximate color counting on $N \times N$ grid that needs $O((N/B)(\log\log N)^2)$ space. For a three-sided range Q, it returns a $(4+\varepsilon)$-approximation on the number of colors in Q in $O((\log\log N)^2)$ I/Os.

Adaptive Point Counting. Furthermore we present an adaptive data structure for three-sided point counting problem. A two-dimensional orthogonal point counting query Q asks for the number K of points in the query range Q. There are linear space data structures that answer two-dimensional counting queries in $O(\log N/\log\log N)$ time in the main memory [JáJá et al. 2004] or in $O(\log_B N)$ I/Os in external memory [Agarwal et al. 2013]. Pătraşcu showed that this query cost is optimal in the worst-case [Pătraşcu 2007]. However, we can further reduce the query cost if the number of points in the query range is small. Jørgensen and Larsen [Jørgensen and Larsen 2011] obtained an $O(N)$-word data structure that answers three-sided queries in $O(\log\log N + \log K/\log\log N)$ time. Chan and Wilkinson [Chan and Wilkinson 2013] achieved the same time and $O(N \log\log N)$ space for the general two-dimensional counting queries. Both results are valid in the word RAM model. The result of [Chan and Wilkinson 2013] matches the lower bound for two-dimensional range emptiness queries and is optimal. But, as in the case of approximate counting, we can get rid of the $\log\log N$ additive term in the query cost when a three-sided query is asked.

In this paper we obtain a main memory adaptive data structure for three-sided counting with optimal query cost and space usage. Our data structure uses $O(N)$ words of space and answers three-sided point counting queries on an $N \times U$ grid in $O(\log_w K)$ time, where $w = \log U$ is the word size and $U \geq N$. We observe that our result is a significant improvement when K is small; for instance, we obtain $O(1)$ query cost if the query range contains $O(\log^f U)$ points for some constant f. We also describe an optimal external data structure that uses linear space and answers three-sided counting queries on an $N \times U$ grid in $O(\log_B K)$ I/Os.

Counting Colors in One Dimension. Finally we consider the problem of counting the number of colors in a one-dimensional query range. One-dimensional color counting queries can be answered in $O(\log N / \log\log N)$ time in the RAM model or in $O(\log_B N)$ I/Os in the external memory model using a linear-space data structure; this result is based on reduction from [Gupta et al. 1995] and data structures for two-dimensional point counting [JáJá et al. 2004; Agarwal et al. 2013]. By the lower bound of Patrascu [Pătraşcu 2007] and the reduction in [Larsen and van Walderveen 2013], this result is optimal. However we can further improve the query cost when the number of colors in the query range is small or when it suffices to obtain an approximate estimate on the number of colors in a query range. In this paper we obtain optimal data structures both for adaptive and for approximate color counting queries in one dimension. We describe a linear space data structure that answers color counting queries in $O(\log_B K)$ I/Os in external memory model or in $O(\log_W K)$ time in the RAM model if point coordinates are bounded by $O(N)$. We also describe another linear space data structure that returns an ε-approximation on the number of points in a query range in $O(1)$ time.

Threshold Counting Queries. We also consider a variant of counting queries, further called τ-threshold counting queries. If the number of elements (points or colors) in a query range does not exceed a query parameter τ, then the exact number of points (respectively colors) is returned. Otherwise, we simply determine that the query range contains more than τ elements. Thus essentially a τ-threshold query returns $\min(K, \tau + 1)$ where K is the number of points (or colors) in the query range. The value of the integer parameter τ is specified at query time. Our data structures for adaptive three-sided point counting and one-dimensional color counting support τ-threshold counting queries in $O(\log_w \tau)$ time in the RAM model and in $O(\log_B \tau)$ I/Os in the external memory model.

Overview. We describe a linear space data structure that supports three-sided color reporting queries in section 2. In section 3 we describe a novel data structure, called path-range tree, and show how it can be used to answer colored three-sided queries. In section 4 we describe data structures that answer colored queries in the case when the set of points is small. We sum up the results for three-sided colored queries on an $N \times N$ grid and show how they can be extended to more general two-dimensional queries in section 5. In section 6 we describe a RAM data structure for three-sided queries with optimal query time and space usage. In sections 7, 7.1, and 8 we show how the approach of path-range trees can be applied to the problems of counting points and colors in the query range. Throughout this paper $p.x$ and $p.y$ denote the x- and y-coordinates of a point p and $col(p)$ denotes the color of p.

2. THREE-SIDED COLOR REPORTING IN LINEAR SPACE

In this section we show that three-sided color reporting queries on an $N \times N$ grid can be answered in $O((N/B)^\delta + (K/B))$ I/Os using a $O(N/B)$ space data structure for any $\delta > 0$. Our data structure combines the sweepline approach with a reduction of one-dimensional color queries to three-sided point reporting queries.

A data structure is called partially persistent if every update operation creates a new version of the data structure and queries on any version can be answered. That is, the t-th update operation on a (partially) persistent variant of a data structure D creates the t-th version $D(t)$. The answer to a query Q at time t is the same as the answer to Q asked to $D(t)$. A data structure is called offline partially persistent if the sequence of update operations is known in advance. We refer to the seminal paper of Driscoll *et al.* [Driscoll et al. 1989] and to a survey of Kaplan [Kaplan 2005] for an extensive description of persistent data structures.

We start by describing a partially persistent data structure that supports three-sided point reporting queries. Then we show how this result can be transformed into a one-dimensional persistent structure that supports color reporting queries. Finally, in Lemma 2.3 we use persistence to transform a one-dimensional structure into a structure that supports three-sided color queries. Lemma 2.3 is based on the sweepline technique, i.e., we convert the insertion times of one-dimensional points in a persistent data structure into the y-coordinates of two-dimensional points. In this and the following sections we will assume w. l. o. g. that all points have different x-coordinates.

LEMMA 2.1. *There exists an offline partially persistent data structure D that stores two-dimensional points and uses $O(N/B)$ space for a sequence of N insert and delete operations; for every version $D(t)$ and any three-sided range $Q = [a, b] \times [0, c]$ we can report all points from $Q \cap D(t)$ in $O((N/B)^\delta + (K/B))$ I/Os for any $\delta > 0$.*

Proof: We maintain a range tree T with node degree $(N/B)^\delta$ on the set of x-coordinates. Each leaf of T contains the x-coordinates of B points. If leaves are counted in the left-to-right order, the i-th leaf v_l contains all points p with $p.x \in rng(v_l)$ where $rng(v_l) = [(i-1)B+1, iB]$. The i-th internal node v on level ℓ contains all points p with $p.x \in rng(v)$ where $rng(v) = [(i-1)(N/B)^{\ell\delta}B + 1, i(N/B)^{\ell\delta}B]$. Every point is stored in exactly one node on each tree level. Since the height of the tree is $1/\delta$, the structure uses $O(N/B)$ blocks of space.

All points stored in a node v are maintained in a persistent tree L_v [Becker et al. 1996; Varman and Verma 1997]. The points of L_v are sorted by their y-coordinates. L_v uses $O(N_v/B)$ space for a sequence of N_v updates and supports range reporting queries on any version $L_v(t)$ in $O(\log_B N_v + K/B)$ I/Os. Hence, we can report all points $p \in L_v(t)$ such that $p.y \le c$ in $O(\log_B N + K/B)$ I/Os.

When a new point p is inserted, we identify the nodes v such that $p.x \in rng(v)$ and insert $p.y$ into persistent trees L_v. Deletions are handled in a symmetric way.

Consider a query $Q = [a, b] \times [0, c]$ at time t. We use a standard representation of the interval $[a, b]$ in a range tree. On every level ℓ we can identify $O((N/B)^\delta)$ nodes $v_{i,\ell}$ with $rng(v_{i,\ell}) \subset [a, b]$ and $rng(parent(v_{i,\ell})) \not\subset [a, b]$. There are also at most two leaves u_1 and u_2 such that $rng(u_i) \cap [a, b] \neq \emptyset$ but $rng(u_i) \not\subset [a, b]$, $i = 1, 2$. We can report all points $p \in L_{v_{i,\ell}}(t)$, $p.y \le c$, in $O(\log_B N + K/B)$ I/Os. Each $L_{u_i}(t)$, $i = 1, 2$, contains $O(B)$ points at any time t because u_i is a leaf. We can construct the list L'_i that contains all points $p \in L_{u_i}(t)$, in $O(\log_B N + 1)$ I/Os. Then, we can traverse L'_i and identify all $p \in L'_i \cap Q$ in $O(1)$ I/Os. The total cost of answering a query is $O((N/B)^\delta \log_B N + (K/B))$. If $B = N^{\Theta(1)}$, $\log_B N = O(1)$. Otherwise $(N/B)^{\delta'} \log_B N = (N/B)^\delta$ for an appropriately chosen $\delta' < \delta$. Hence, we can obtain the result of this Lemma by substituting δ' instead of δ in the above construction. □

LEMMA 2.2. *There exists an offline partially persistent data structure D that contains colored one-dimensional points and uses $O(N/B)$ space for a sequence of N insertions; for every version $D(t)$ and any one-dimensional range $Q = [a, b]$ we can report all distinct colors of points from $Q \cap D(t)$ in $O((N/B)^\delta + (K/B))$ I/Os for any constant $\delta > 0$.*

Proof: Using the technique described in [Gupta et al. 1995], we can answer one-dimensional colored range reporting queries by a data structure for three-sided point reporting queries. Suppose that S is a set of one-dimensional points. We denote by $prev(p)$ the rightmost point p_1 such that $p_1.x < p.x$ and p_1 has the same color as p. If p is the leftmost point of color ν, we assume that $prev(p) = p_d$ for a dummy point p_d with $p_d.x = 0$. We denote by $succ(p)$ the leftmost point p_2 such that $p_2.x > p.x$ and p_2 has the same color as p. For every one-dimensional point $p \in S$, the set S' contains a two-dimensional point p' such that $p'.x = p.x$, $p'.y = prev(p).x$, and $col(p') = col(p)$. If an interval $[a, b]$ contains one or more points $p \in S$ of color $\nu \in \mathcal{C}$, then the range $Q = [a, b] \times [0, a]$ contains exactly one point $p' \in S'$ such that $col(p') = \nu$.

Using Lemma 2.1 and the approach of [Gupta et al. 1995] described above, we can obtain a partially persistent data structure for one-dimensional color reporting queries that uses $O(N/B)$ space for a sequence of N insertions. Let S' be the set defined above. When a new point p is inserted into S, the set S' is updated as follows. We insert the point $p_1 = (p.x, prev(p).x)$ into S'. If $succ(p)$ exists, we delete $p_2 = (succ(p).x, prev(p).x)$ from S' and insert a new point $p_3 = (succ(p).x, p.x)$. We maintain the set S' in the data structure E implemented as in Lemma 2.1 □

LEMMA 2.3. *There exists a $O(N/B)$ space data structure that answers three-sided color range reporting queries for points on an $N \times N$ grid in $O((N/B)^{\delta} + (K/B))$ **I/Os** for any $\delta > 0$.*

Proof: Our construction is based on the combination of the sweepline technique and the result of Lemma 2.2. To construct our data structure, we sweep a horizontal line h in the $+y$ direction. Initially the y-coordinate of h is 0; when h hits a point p, we add p to the partially persistent data structure E that supports colored reporting queries in one dimension. E is implemented as in Lemma 2.2.

Let t_c denote the version of E after the point with y-coordinate c was inserted. Since points lie on an $N \times N$ grid, we can store t_c for all c in an array with N entries. $E(t_c)$ contains all points $p \in S$ with $p.y \le c$. Hence, we can answer a three-sided query $Q = [a, b] \times [0, c]$ by asking the query $Q_1 = [a, b]$ to the t_c-th version $E(t_c)$ of the data structure E. □

3. PATH-RANGE TREES

In this section we describe data structures for three-sided colored reporting with optimal or almost optimal query cost. After giving some preliminary definitions we define a new data structure, called path-range tree. Then, we show how path-range trees can be used to answer three-sided colored queries.

We associate x-coordinates of points with the leaves of a full binary tree T. The i-th leaf of T is associated with the i-th smallest x-coordinate x_i of a point from S. Recall that we consider the situation when points lie on $N \times N$ grid. Hence, in our case $x_i = i$ for $1 \le i \le N$. The *range* of the i-th leaf is an interval $[x_i, x_{i+1} - 1]$. The range of an internal node v is the union of ranges of its children. The set $S(v)$ contains all points p such that $p.x$ belongs to the range of v.

Range trees [Bentley 1980] and priority trees [McCreight 1985] are two handbook data structures used in numerous geometric applications. In the range tree, we store a set $S(v)$ in each node v. Since each point belongs to $O(\log N)$ nodes, a range tree contains $O(N \log N)$ points. In the priority tree, each node v contains only one point $p(v) \in S(v)$. We select $p(v)$ to be the lowest point from $S(v)$ such that $p(v) \neq p(w)$ for any ancestor w of v. Since each point is stored only once, a priority tree contains $O(N)$ points.

Structure. A path-range tree, combines some properties of range trees and priority trees with a new approach. While in the range tree and priority tree points are associated with tree nodes, the path-range tree is based on sets $Y(\pi)$ for selected node-to-leaf paths π; points of $Y(\pi)$ belong to nodes that hang off a path π. For every $Y(\pi)$, only a subset $L(\pi) \subset Y(\pi)$ is stored in the data structure. Points of $L(\pi)$ are chosen in such way that each color from $Y(\pi)$ occurs at most once in $L(\pi)$; points in $L(\pi)$ are stored in ascending order of their y-coordinates. A detailed description of the path-range tree is given below.

We denote by $\pi(u, v)$ (the set of nodes on) the path from the node u to its descendant v (excluding u). Sets $\pi_l(u, v)$ and $\pi_r(u, v)$ contain all nodes that satisfy the following condition: If a node $w \in \pi(u, v)$ is the right child of its parent $w' \in \pi(u, v)$, then the sibling of w belongs to $\pi_l(u, v)$; if a node $w \in \pi(u, v)$ is the left child of its parent $w' \in \pi(u, v)$, then the sibling of w belongs to $\pi_r(u, v)$. See Fig. 1 for an example. We define $\Pi_l(u, v) = \cup_{w \in \pi_l(u,v)} S(w)$ and $\Pi_r(u, v) = \cup_{w \in \pi_r(u,v)} S(w)$. Let $\mathrm{ymin}(\nu, F)$ for a color ν and a set of points F denote the point with the smallest y-coordinate among all points $p \in F$ with $col(p) = \nu$. The set $Y_l(u, v)$ (respectively, $Y_r(u, v)$) contains the point $\mathrm{ymin}(c, \Pi_l(u, v))$ (respectively, $\mathrm{ymin}(c, \Pi_r(u, v))$ for any color c that occurs in $\Pi_l(u, v)$ ($\Pi_r(u, v)$)). Thus $Y_l(u, v)$ contains exactly one point for each color that occurs in $\Pi_l(u, v)$. The list $Y_l(u, v)[1..i]$ contains i points from $Y_l(u, v)$ with the smallest y-coordinates; we assume that the points of $Y_l(u, v)[1..i]$ are sorted by their y-coordinates. $Y_r(u, v)[1..i]$ is defined in the same way with respect to $Y_r(u, v)$.

We say that a node v is on level ℓ if the shortest path from v to a leaf node consists of ℓ edges. Let $\mathcal{N}_i$ denote the set of nodes on level[3] $\log B + f(i) \log\log N$ for $f(i) = 2^i$ and $i = 1, \ldots, \log\log_{\log N}(N/B)$. A set $S(v)$, $v \in \mathcal{N}_i$, contains $O(B \log^{f(i)} N)$ points. For each $v \in \mathcal{N}_i$ and each ancestor u of v we store the lists $L_l(u, v) = Y_l(u, v)[1..m_i]$ and $L_r(u, v) = Y_r(u, v)[1..m_i]$ where $m_i = B \log^{f(i)-1} N$. We also store all points from $S(v)$ for a node $v \in \mathcal{N}_i$ and $i \geq 2$ in a $O(|S_v|/B)$ space data structure $D(v)$ that answers three-sided color reporting queries in $O((|S(v)|/B)^{1/4} + K/B)$ I/Os. This data structure is implemented as in Lemma 2.3. All points from $S(v)$ for each $v \in \mathcal{N}_1$ are stored in a structure $D'(v)$ that will be described in section 4. $D'(v)$ supports three-sided color reporting queries in $O(q(N)+K/B)$ I/Os and uses space $O((|S(v)|/B)s(N))$ for functions $s(N)$ and $q(N)$ that will be defined in section 4.

Queries. Let $Q = [a, b] \times [0, c]$ be the query range; let l_a and l_b be the the leaves that contain the largest value smaller than a and the smallest value larger than b respectively. We denote by w the lowest common ancestor of l_a and l_b. For any point p, $p.x \in [a, b]$ if and only if p belongs to some set $S(u)$ for $u \in \pi_r(w, l_a) \cup \pi_l(w, l_b)$; see Fig. 1. Therefore our goal is to examine the points from $\Pi_r(w, l_a) \cup \Pi_l(w, l_b)$ and report the distinct colors of points p with $p.y \leq c$. Our structure does not store sets $\Pi_r(w, l_a)$ and $\Pi_l(w, l_b)$ (this would take at least $\Theta(n^2)$ space). However, we will show that the information contained in sets $L_l(u, v)$, $L_r(u, v)$, and structures $D(v)$ for nodes $v \in \mathcal{N}_i$ is sufficient to examine relevant parts of the set $\Pi_r(w, l_a) \cup \Pi_l(w, l_b)$.

We show how relevant colors from the set $\Pi_l(w, l_b) = \cup_{u \in \pi_l(w,l_b)} S(u)$ can be reported. The set $\Pi_r(w, l_a)$ can be processed symmetrically. Let v_i denote the ancestor of l_b that belongs to $\mathcal{N}_i$. For $i = 1, \ldots$ we traverse the lists $L_l(w, v_i)$ until a point p_i, $p_i.y > c$, is encountered or the node v_i is not a descendant of w. In each visited node v_i, $i \geq 1$, our procedure works as follows. If $v_i = w$ or v_i is an ancestor of w, we answer the query $Q = [a, b] \times [0, c]$ to data structure $D(v_i)$ (respectively $D'(v_i)$ if $i = 1$) and stop. If v_i is a descendant of w, the list $L_l(w, v_i)$ is traversed until a point p with $p.y > c$ is found. If we reach the end of $L_l(w, v_i)$ and $p.y \leq c$ for all $p \in L_l(w, v_i)$, we increment i and proceed in the next node v_i. Otherwise we traverse $L_l(w, v_i)$ once again and report colors of

[3]To avoid tedious details, we assume throughout this paper that B, N, and $\log N$ are powers of 2.

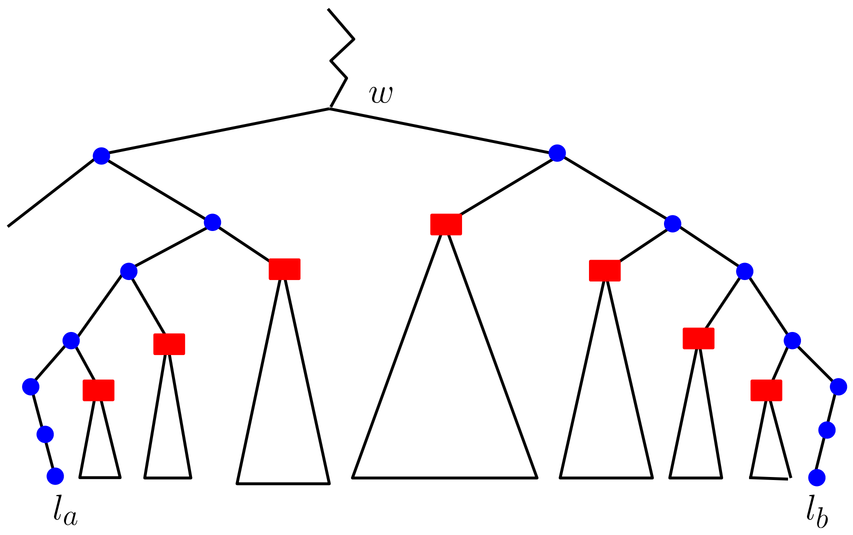

Fig. 1. Paths in a path-range tree. Node w is the lowest common ancestor of l_a and l_b. Nodes in $\pi(w, l_a)$ and $\pi(w, l_b)$ are marked with circles. Nodes in $\pi_r(w, l_a)$ and $\pi_l(w, l_b)$ are marked with rectangles. Only relevant parts of the tree are shown.

all points $p \in L_l(w, v_i)$, $p.y \le c$. Then, we answer the query $Q = [a, b] \times [0, c]$ to data structure $D(v_i)$ (respectively $D'(v_i)$ if $i = 1$) and stop.

Let t be the index of the last visited node v_t. Suppose that there is at least one point $p \in \Pi_l(w, v_t)$ of color ν, such that $p.y \le c$. Then $\text{ymin}(\nu, \Pi_l(w, v_t)) \le c$ and $L_l(w, v_t)$ contains exactly one point p' such that $col(p') = \nu$ and $p.y \le c$. Colors of all such points are reported when $L_l(w, v_t)$ is traversed for the second time. If a point p belongs to $\Pi_l(w, l_b) \setminus \Pi_l(w, v_t)$, then p belongs to $S(v_t)$. Hence, colors of all points $p \in \Pi_l(w, l_b) \setminus \Pi_l(w, v_t)$ are found by a query to $D(v_t)$.

Let K be the total number of distinct colors in $\Pi_l(w, l_b)$. As follows from the description of our search procedure, $K \ge |L_l(w, v_{t-1})| = \Theta(B \log^{f(t-1)-1} N)$. Since $|L_l(w, v_{t-1})| \ge \sum_{i=1}^{t-2} |L_l(w, v_i)|$, $K = \Omega(\sum_{i=1}^{t-1} |L_l(w, v_i)|)$. Hence, the total cost of traversing $L_l(w, v_i)$, $1 \le i < t$, is $O(K/B)$. Every point in the traversed prefix of $L_l(w, v_t)$ corresponds to a distinct color in $\Pi_l(w, v_t)$. Therefore $L(w, v_t)$ is also traversed in $O(K/B)$ I/Os. The cost of a query to the data structure $D(v_t)$ is $O(((B \log^{f(t)} N)/B)^{1/4} + K'/B)$ where $K' \le K$ is the number of distinct colors in $S(v_t) \cap Q$. Since $(\log^{f(t)} N)^{1/4} = O(\log^{f(t-1)-1} N) = O(|L(w, v_{t-1})|/B) = O(K/B)$ for $t > 1$, the query to $D(v_t)$ is answered in $O(K/B)$ I/Os. If $t = 1$, the query to $D'(v_1)$ takes $O(q(N) + K'/B)$ I/Os.

Space Usage. There are $O(N/(B \log^{f(i)} N))$ nodes in $\mathcal{N}_i$. Since $S(v)$, $v \in \mathcal{N}_i$, contains $O(B \log^{f(i)} N)$ points, all data structures $D(v)$ for $v \in \mathcal{N}_i$ and $2 \le i \le \log\log_{\log N}(N/B)$ use $O((N/B) \log\log(N/B))$ blocks of space. Each list $L_l(u, v)$ and $L_r(u, v)$ for $v \in \mathcal{N}_i$ contains $O(B \log^{f(i)-1} N)$ points. There are $O(\log N)$ lists for each $v \in \mathcal{N}_i$. Hence, all $L_l(u, v)$, $L_r(u, v)$ also use $O((N/B) \log\log(N/B))$ blocks of space. Finally, all structures $D'(v)$ for $v \in \mathcal{N}_1$ use $O((N/B)s(N))$ blocks of space.

The main result of this section can be summed up as follows.

LEMMA 3.1. *Suppose that there exists a data structure that answers three-sided colored reporting queries on a set of* $r = O(B \log^2 N)$ *points on* $r \times N$ *grid in* $O(q(N) + K/B)$ *I/Os and uses* $O(\frac{r}{B} s(r))$ *space. Then there exists a* $O(\frac{N}{B}(\log\log N + s(N)))$ *space* *data structure that answers three-sided color reporting queries on a set of N points on $N \times N$ grid in $O(q(N) + K/B)$ I/Os.*

Proof: A node $v \in \mathcal{N}_1$ has r leaf descendants. Hence x-coordinates of points stored in a structure $D'(v)$, $v \in \mathcal{N}_1$, are integers $t, t+1, \ldots, t+r$. We can replace the x-coordinate of each point p by relative x-coordinate $p.x - t + 1$. Hence, we can assume that all points $p \in S(v)$ for any node $v \in \mathcal{N}_1$ lie on $r \times N$ grid. By assumption of this Lemma each $D'(v)$ can be implemented in $O(\frac{r}{B} s(r))$ space so that queries are supported in $O(q(N) + K/B)$ I/Os. The total space usage and query cost follow from the description preceding this Lemma. $\square$

Remark The data structure of Lemma 3.1 produces a list of size $O(K)$ that contains all K colors occurring in a three-sided range and no other colors. However, the list of colors may contain duplicates. That is, the same color may appear in the list more than once, but no more than a constant number of times. If necessary, we can remove duplicates from the list in $O(\frac{K}{B} \log_{M/B} \frac{K}{B})$ additional I/Os, where M is the size of the main memory. For instance if $K = B^{O(1)}$ and $M = B^{1+\varepsilon}$ for any constant $\varepsilon > 0$, duplicates are deleted in $O(K/B)$ I/Os. The same remark also applies to results in sections 4 and 5.

4. THREE-SIDED COLOR REPORTING FOR $O(B \operatorname{LOG}^2 N)$ POINTS

Using the same idea as in section 3, we can obtain two efficient data structures for a set that contains $r = O(B \log^2 N)$ points. We construct a path-range tree on the set of r points and use the same notation as in section 3. Again a three-sided query $Q = [a, b] \times [0, c]$ is answered by reporting all distinct colors in $Q \cap \Pi_r(w, l_a)$ and $Q \cap \Pi_l(w, l_b)$. To avoid repetitions, we will describe only the differences between the new structures and the path-range tree presented in section 3.

LEMMA 4.1. *Let S be a set of $r = O(B \log^2 N)$ points on an $r \times N$ grid. There exists a $O((|S|/B) \log \log N)$ space data structure that supports three-sided colored queries on S in $O((\log \log N)^{\delta} + K/B)$ I/Os for any constant $\delta > 0$.*

Proof: The set $\mathcal{N}_i$, $i \geq 1$, contains all nodes on level $\log B + f(i) \log^{(3)} N$. The data structures $D(v)$ for $v \in \mathcal{N}_1$ are implemented using Lemma 2.3 with parameter $\delta' = \delta/2$. A set $S(v)$, $v \in \mathcal{N}_1$, contains $O(B (\log \log N)^2)$ points. Hence, $D(v)$ answers three-sided color queries in $O((|S(v)|/B)^{\delta'} + K/B) = O((\log \log N)^{\delta} + K/B)$ I/Os.

The procedure that reports all distinct colors in $\Pi_l(w, l_b) \cap Q$ visits ancestors $v_i \in \mathcal{N}_i$ of l_b for $i = 1, 2, \ldots$ and traverses lists $L_l(w, v_i)$ as in the proof of Lemma 3.1. Suppose that v_t is the last visited node. We can report all distinct colors from $L_l(w, v_t)$ in $O(K/B)$ I/Os. If $t = 1$, the query to $S(v_t)$ is answered in $O((\log \log N)^{\delta} + K/B)$ I/Os; otherwise the query to $S(v_t)$ is answered in $O(K/B)$ I/Os. Colors that occur in $\Pi_r(w, l_a) \cap Q$ can be found by a symmetric procedure. $\square$

We can improve the query cost by slightly increasing the space usage.

LEMMA 4.2. *Let S be a set of $r = O(B \log^2 N)$ points on an $r \times N$ grid. There exists a $O((|S|/B) \log \log N \log^{(3)} N)$ space data structure that supports three-sided colored queries on S in $O(K/B)$ I/Os.*

Proof: The only important difference from the structure in section 3 is in the choice of parameters. The set $\mathcal{N}_1$ consists of all nodes on level $\log B$. For $2 \leq i \leq \log^{(3)}(N/B)$ the set $\mathcal{N}_i$ contains all nodes on the level $\log B + f(i)$. Since $|S(v)| = O(B)$ for $v \in \mathcal{N}_1$, we can store $S(v)$ in one block of space and answer colored queries on $S(v)$ in $O(1)$ I/Os. A query is answered in the same way as in section 3 or in the proof of Lemma 4.1. Since a query to a data structure $D(v)$ for $v \in \mathcal{N}_1$ can be answered in $O(1)$ I/Os, the total query cost is $O(K/B)$.

All sets $S(v)$ contain $O(r \cdot \log\log r)$ points. All lists $L_l(u,v)$ and $L_r(u,v)$ for $v \in \mathcal{N}_i$ and a fixed i contain $O(r \log r)$ points. Hence, the total space usage is $O((r/B)\log r \log\log r) = O((|S|/B)\log\log N \log^{(3)} N)$.

$\square$

5. COLORED RANGE REPORTING IN TWO DIMENSIONS

Combining Lemma 3.1 and Lemma 4.2, we obtain the data structure with optimal query cost.

THEOREM 5.1. *There exists a $O((N/B)\log\log N \log^{(3)} N)$ space data structure that answers three-sided color reporting queries on an $N \times N$ grid in $O(K/B)$ I/Os.*

Combining Lemma 3.1 and Lemma 4.1, we obtain the data structure with a slightly better space usage and slightly worse query cost.

THEOREM 5.2. *There exists a $O((N/B)\log\log N)$ space data structure that answers three-sided color reporting queries on an $N \times N$ grid in $O((\log\log N)^{\delta} + K/B)$ I/Os for any $\delta > 0$.*

We can extend the result of Theorem 5.1 to a more general scenario.

THEOREM 5.3. *There exists a $O((N/B)\log\log N \log^{(3)} N)$ space data structure that answers three-sided color reporting queries on a $U \times U$ grid in $O(\log\log_B U + K/B)$ I/Os. There exists a $O((N/B)\log N \log\log N \log^{(3)} N)$ space data structure that answers two-dimensional color reporting queries on a $U \times U$ grid in $O(\log\log_B U + K/B)$ I/Os.*

Proof: Orthogonal queries on a $U \times U$ grid can be reduced to queries on an $N \times N$ grid using the rank space technique [Gabow et al. 1984]. This technique is based on replacing the point coordinates with their respective ranks. The x-rank of a point p in a set S is the number of points $p' \in S$ whose x-coordinates do not exceed $p.x$ (ties are broken arbitrarily). The y-rank of p is the number of points $p' \in S$ whose y-coordinates do not exceed $p.y$. We obtain the rank set of S by replacing the x- and y-coordinates of each point p by its x- and y-rank. We can also define the x-rank (y-rank) of a value t as the number of points $p \in S$ such that $p.x \le t$ (resp. $p.y \le t$). Gabow et al. [Gabow et al. 1984] showed that we can answer an orthogonal query $Q = [a,b] \times [c,d]$ on S by answering some orthogonal query Q' on the rank set. Essentially, we must find the x-ranks of a, b and y-ranks of c and d to obtain Q'; we refer to [Gabow et al. 1984; Alstrup et al. 2000] for details. The query cost increases by the time necessary to search for a predecessor in an ordered set of N elements and the space usage remains unchanged. We can search for a predecessor in a one-dimensional set in $O(\log_2 \log_B U)$ I/Os [Pătraşcu and Thorup 2006].

We can answer a general orthogonal colored reporting queries in two dimensions by answering two three-sided colored reporting queries. This reduction is standard in range reporting structures; it was also applied to colored range reporting in [Agarwal et al. 2002]. For completeness, we sketch this method below.

We construct a range tree T_y on the y-coordinates of points. In each node v of T_y, we define the set $S(v)$ as in section 3. All points of $S(v)$ are stored in two data structures, $P_1(v)$ and $P_2(v)$ that support three-sided queries. $P_1(v)$ answers queries of the form $[a,b] \times [0,c]$; it can be implemented using Theorem 5.1. $P_2(v)$ supports queries of the form $[a,b] \times [c,+\infty]$; it can be implemented in a symmetric way.

To answer a query $[a,b] \times [c,d]$, we identify the lowest common ancestor w of leaves l_c and l_d; l_c and l_d are the leaves that contain c and d respectively. Let w_1 and w_2 be the left and right children of w. All points in $S \cap Q$ belong to either $S(w_1) \cap ([a,b] \times [c,+\infty])$ or $S(w_2) \cap ([a,b] \times [0,d])$. Hence, we can find all colors in $S \cap Q$ by answering two three-sided color queries on $P_2(w_1)$ and $P_1(w_2)$. Thus the query cost is the same as the cost of three-sided queries. The space usage increases by $O(\log N)$ factor. Again, we can extend the result for an $N \times N$ grid to the result on a $U \times U$ grid by applying the reduction to rank space [Gabow et al. 1984]. $\square$

Using the same methods, we can also extend the result of Theorem 5.2.

THEOREM 5.4. *There exists a $O((N/B) \log\log N)$ space data structure that answers three-sided color reporting queries on a $U \times U$ grid in $O((\log\log N)^{\delta} + \log\log_B U + K/B)$ **I/Os**.*
*There exists a $O((N/B) \log N \log\log N)$ space data structure that answers two-dimensional color reporting queries on a $U \times U$ grid in $O((\log\log N)^{\delta} + \log\log_B U + K/B)$ **I/Os**.*

We can also use the rank space technique [Gabow et al. 1984] to apply all results described in Theorems 5.1-5.4 and Lemma 2.3 to the situation when point coordinates are arbitrary reals. The space usage of all data structures does not change and the query cost increases by an additive factor $O(\log_B N)$.

6. COLOR RANGE REPORTING IN THE RAM MODEL

COROLLARY 6.1. *There exists a data structure that uses $O(m \log m)$ bits and answers three-sided color range reporting queries for points on an $m \times m$ grid in time $O((m)^{\delta} + K)$ for any constant $\delta > 0$.*

Proof: We obtain this statement by setting $B = const$ and substituting N with m in Lemma 2.3. $\square$

LEMMA 6.2. *Let S be a set of $r = O((\log\log N)^2)$ points on an $r \times N$ grid. There exists a $O(|S|)$-words data structure that supports three-sided colored queries on S in $O(1 + K)$ time. The solution needs access to a universal look-up table of size $o(N)$.*

Proof: Using reduction to rank space [Gabow et al. 1984], we can reduce the problem on $r \times N$ grid to a problem on $r \times r$ grid. Since our set contains r points, we can assign an integer index between 1 and r to each color. We store a table of size $O(r)$ that enables us to identify the color with index i for every valid i. Consider sets of r points on $r \times r$ grid, such that each point is assigned some color index i, $1 \le i \le r$. There are $2^{O((\log\log N)^2)} = o(N/r^4)$ different sets of colored points on an $r \times r$ grid and r^3 different queries. An answer to each query consists of at most r colors. Hence, we can store answers to all possible queries on all possible sets in a look-up table of size $o(N)$. Given a query Q, we can obtain the list of color indexes of all points in Q using this look-up table. Using the table of indexes, we can then report the colors that correspond to indexes in $O(1)$ time per color. $\square$

LEMMA 6.3. *Let t be a pre-defined parameter, such that $\log N \le t \le N$. Suppose that there exists a data structure that answers three-sided colored reporting queries on a set of $r = O(\log^2 t)$ points on $r \times N$ grid in $O(q(t) + K/B)$ **I/Os** and uses $O(rs(r))$ space. Then there exists a $O(ts(t))$ space data structure that answers three-sided color reporting queries on a set of t points on $t \times N$ grid in $O(q(t) + K/B)$ **I/Os**.*

Proof: Let S be a set with t points. We construct a path-range tree on S with the following parameters. The set $\mathcal{N}_i$ contains all nodes on level $\lg^{(3)} t + f(i) \lg\lg t$. Thus each $v \in \mathcal{N}_i$ has $\lg^{f(i)} t \lg\lg t$ leaf descendants. For a node $v \in \mathcal{N}_i$ and an ancestor u of v, the list $L_l(u,v)$ (resp. $L_r(u,v)$) contains the first (lowest) $m_i = \log^{f(i)-1} t$ elements of $Y_l(u,v)$ ($Y_r(u,v)$). Data structures $D(v)$, $v \in \mathcal{N}_i$, support three-sided color queries on sets $S(v)$ and are implemented as in Corollary 6.1.

All lists $L_l(u,v)$ and $L_r(u,v)$ use $\sum_{i=1}^{\log\log_{\log t} t}(t/\log\log t) = O(t)$ words. All data structures $D(v)$ for $v \in \mathcal{N}_i$ use $O(t \lg\lg t)$ words. We can reduce the space usage of $D(v)$ by applying the reduction to rank space to $D(v)$ for all $v \in \mathcal{N}_i$ and $i \geq 3$. The x-coordinates of all points in $S(v)$ are consecutive integers. Hence, we can obtain the x-rank of a point from its true coordinate (the x-coordinate of $p \in S(v)$ from its x-rank in $S(v)$) in $O(1)$ time: we simply subtract the x-coordinate of the leftmost point in $S(v)$ from the x-coordinate of p (resp. we add the x-coordinate of the leftmost point in $S(v)$ to the x-rank of p). The x-rank of any integers a, b, $1 \leq a, b \leq t$, can be computed in the same way. Using the same idea, we can also retrieve the color of each point $p \in S(v)$ in $O(1)$ time from its $O(\log(|S(v)|))$-bit representation: instead of directly storing the color of a point, we store the x-rank of some point with the same color.

We need a somewhat more complicated construction for the y-ranks. Let $X(v)$ and $Y(v)$ denote the sets of x- and y-ranks of points in $S(v)$. For every node $u \in \mathcal{N}_i$ and every node $w \in \mathcal{N}_{i-1}$, such that w is a descendant of u, we store a data structure $V(u,w)$ implemented as a van Emde Boas data structure [van Emde Boas et al. 1977]. $V(u,w)$ contains the y-ranks in $S(u)$ of all $p \in S(u) \cap S(w)$. For any y, $1 \leq y \leq |S(v)|$, $V(u,w)$ can return the y-rank of the highest point $p_w \in S(w)$, such that $p_w.y \leq y$, in $O(\log\log N)$ time. If we know the y-rank of p_w in S_u, we can find the y-rank of p_w in $S(w)$ using a table look-up. We also store the y-coordinates of all points $p \in S$ in the van Emde Boas data structure V [van Emde Boas et al. 1977]. Using V, we can find for any positive $c \leq N$ the highest point $p \in S$ such that $p.y \leq c$ in $O(\log\log N)$ time. Thus we can find the y-rank of any $c \in [1, N]$ in any set S_u, $u \in \mathcal{N}_i$, in $O((\log\log N)^2)$ time.

Each data structure $D(v)$ for $v \in \mathcal{N}_i$ contains $s_i = O(\lg\lg t \lg^{f(i)} t)$ points. If every point coordinate is replaced by its rank, we can store the data structure $D(v)$ in $O(s(i)(\lg^{(3)} t + f(i)\lg\lg t)) = O(s(i)(\lg^{(3)} N + f(i)\lg\lg N))$ bits. All data structures $D(v)$ for $v \in \mathcal{N}_i$ and $i \geq 1$ require $O(t\sum_{i=1}^{\log\log_{\log t} t}(\lg^{(3)} t + f(i)\lg\lg t))$ bits. Data structures $V(u,w)$ and the look-up table for a node u use $O(|S(u)|\log|S(u)|)$ bits. Hence, all data structures $V(u,w)$ and all tables need $O(t\sum_{i=1}^{\log\log_{\log t} t}(\lg^{(3)} t + f(i)\lg\lg t)) = O(t\sum_{i=1}^{\log\log_{\log N} N}(\lg^{(3)} N + f(i)\lg\lg N))$ bits. Data structures $D(v)$ for $v \in \mathcal{N}_1 \cup \mathcal{N}_2$ contain $O(t)$ points and use linear space. Thus the total space usage of our data structure is $O(t)$ words of $\log N$ bits.

The procedure to answer a query $Q = [a,b] \times [0,c]$ is the same as in Lemma 3.1. Suppose that we need to answer a color query to a data structure $D(v)$ for $v \in \mathcal{N}_i$ and $i \geq 3$ while Q is processed. We find the x-ranks of a and b in $S(v)$ in $O(1)$ time. Using V and $V(u,w)$, we find the y-rank of c in $S(v)$ in $O((\log\log N)^2)$ time. Once the x-ranks of a, b and the y-rank of c are known, we answer the data structure on $D(v)$ as in the previous implementations of the path-range tree. Since we ask a query on a data structure $D(v)$ for $v \in \mathcal{N}_i$ and $i \geq 3$, the query answer contains at least $(\log\log N)^3$ colors. Therefore we answer a query to $D(v)$ in $O((\log\log N)^2 + K) = O(K+1)$ time. $\square$

Now we are ready to prove the main result for color reporting in the RAM model.

THEOREM 6.4. *There exists an $O(N)$-words data structure that answers three-sided color reporting queries on $N \times N$ grid in $O(K+1)$ time.*
There exists an $O(N\log N)$-words data structure that answers two-dimensional color reporting queries on $N \times N$ grid in $O(\log\log U + K)$ time.

Proof: First we set $t = \log^2 N$ in Lemma 6.3 and apply Lemma 6.2. We obtain a data structure $\mathcal{D}_1$ that answers color reporting queries for $\log^2 N$ points on $\log^2 N \times N$ grid in $O(1 + K)$ time. Then, we set $t = N$ in Lemma 6.3 and apply the data structure $\mathcal{D}_1$. We obtain a linear space data structure that answers three-sided reporting queries on $N \times N$ grid in $O(K + 1)$ time.

We can extend this result in the same way as in the proof of Theorem 5.3 and obtain a data structure that uses $O(N \log N)$ words and supports general two-dimensional queries on a $U \times U$ grid in $O(\log \log U)$ time. $\square$

7. APPROXIMATE COLOR COUNTING

In this section we show how the ideas used in path-range trees can be applied to estimating the number of different colors in a three-sided range.

THEOREM 7.1. *There exists an $O(N(\log \log N)^2)$-words data structure for points on $N \times N$ grid that returns a $(4+\varepsilon)$-approximation for the number of colors in a three-sided range in $O((\log \log N)^2)$ time.*

Proof: Let T be the full binary tree on the x-coordinates; let $\mathcal{N}_1'$ denote the set of tree nodes on level $2 \log \log N$. We define the sets $Y_l(u, v)$ and $Y_r(u, v)$ in the same way as in section 3. For every $Y_l(u, v)$, such that $v \in \mathcal{N}_1'$ and u is an ancestor of v, we store a data structure $E_l(u, v)$. $E_l(u, v)$ uses $O(\log N)$ words and enables us to estimate the number of points $p \in Y_l(u, v)$ such that $p.y \le c$. For $j = 1, 2, \ldots, \log_\varepsilon(|Y_l(u, v)|)$, $E_l(u, v)$ contains the (ε^j)-th point from $Y_l(u, v)$ (that is, the ε^j-th lowest point in $Y_l(u, v)$). Using q-heaps [Fredman and Willard 1994], we can find for any c, the highest point $p \in E_l(u, v)$ such that $p.y \le c$ in $O(1)$ time. Thus $E_l(u, v)$ enables us to estimate the number of points $p \in Y_l(u, v)$ such that $p.y \le c$. Obviously $E_l(u, v)$ also provides an estimate for the number of point colors for points $p \in \Pi_l(u, v)$, $p.y \le c$. A data structure $E_r(u, v)$ is defined in the symmetric way for the set $Y_r(u, v)$.

For each $v \in \mathcal{N}_1$, we also store a data structure $D(v)$ that supports exact color counting queries on $S(v)$. Since $|S(v)| = O(\log^2 N)$, we can implement $D(v)$ so that it uses $O(|S(v)|(\log \log N)^2)$ words and answers exact color counting queries in $O((\log \log N)^2)$ time [Gupta et al. 1995].

To estimate the number of points in a range $= [a, b] \times [0, c]$, we identify the leaves l_a and l_b that hold a and b. Let w denote the lowest common ancestor of a and b. If w is a descendant of some $v \in \mathcal{N}_1$, we answer the query using data structure $D(v)$. Otherwise, let $v_a \in \mathcal{N}_1$ and $v_b \in \mathcal{N}_1$ be the ancestors of l_a and l_b respectively. All points $p \in S$, such that $a \le p.x \le b$ belong to either $S(v_a) \cup S(v_b)$ or to $\Pi_l(w, v_a) \cup \Pi_r(w, v_b)$. We can obtain n_a, n_b, such that $n_a \le |S(v_a) \cap Q| \le (1 + \varepsilon)n_a$ and $n_b \le |S(v_b) \cap Q| \le (1 + \varepsilon)n_b$ using $D(v_1)$ and $D(v_2)$ respectively. For all $p \in \Pi_r(w, v_a) \cup \Pi_l(w, v_b)$, $a \le p.x \le b$. Hence, a point $p \in \Pi_r(w, v_a) \cup \Pi_l(w, v_b)$ is in Q iff $p.y \le c$. We can estimate the number n_a' of colors for points $p \in \Pi_r(w, v_a)$, $p.y \le c$, using $E_r(w, v_a)$. We can estimate the number n_b' of colors for points $p \in \Pi_l(w, v_b)$, $p.y \le c$, using $E_l(w, v_a)$. The sum $n_a + n_b + n_a' + n_b'$ provides an estimate for the total number of different colors in Q. Since every color may be counted up to four times, $(n_a + n_b + n_a' + n_b')/4 \le n_{\text{real}} \le (1 + \varepsilon)(n_a + n_b + n_a' + n_b')$, where n_{real} is the actual number of different colors that occur in Q. Thus $(n_a + n_b + n_a' + n_b')/4$ provides a $(4 + 4\varepsilon)$-approximation for the number of colors in a three-sided range. By substituting $\varepsilon' = \varepsilon/4$ into our proof, we obtain the result of this theorem. $\square$

7.1. Approximate Point Counting

THEOREM 7.2. *There exists an $O(N)$-words data structure for points on $N \times N$ grid that returns an ε-approximation for the number of points in a three-sided range in $O(1)$ time.*

Proof: Our data structure is the same as in Theorem 7.1. To count points instead of colors, we assume that each point in a data structure is assigned a unique color. In this way an estimation for the number of colors provides an estimation for the number of points. Data structures $D(v)$ for $v \in \mathcal{N}'_1$ are implemented as data structures for standard point counting queries. Since $|S(v)| = \log^2 N$, we can answer range counting queries on $S(v)$ in $O(1)$ time using an $O(|S(v)|)$-word data structure[JáJá et al. 2004]. In the case of point reporting we obtain a better estimation because the counts returned by queries to $D(v_a)$, $D(v_b)$, $E_r(w, v_a)$, and $E_l(w, v_b)$ can be added without increasing the approximation factor. Hence, $(n_a + n_b + n'_a + n'_b) \leq n_{\text{real}} \leq (1+\varepsilon)(n_a + n_b + n'_a + n'_b)$. □

Clearly, Theorems 7.2 and 7.1 can be re-formulated for the external memory model. The query costs remain unchanged; the space usage is $O(N/B)$ blocks for approximate point counting and $O((N/B)(\log\log N)^2)$ blocks for color counting.

8. ADAPTIVE POINT COUNTING

In this section we show how three-sided point counting queries can be answered in $O(\log K/\log W)$ time, where K is the number of points in the answer and W is the word size in bits. Results of this section are valid in the word RAM model. We assume that point coordinates are positive integers bounded by U, each point coordinate fits into a machine word, and thus $W = \log U$. We will need the following Lemma to support reduction to rank space in auxiliary data structures.

LEMMA 8.1. *Let S be a set of m points on $m \times U$ grid and suppose that the y-coordinate of each point $p \in S$ can be computed in $O(1)$ time if its x-coordinate is known. There exists a data structure that uses $O(m(\log W + \log m))$ additional bits and finds the y-rank of any c, $1 \leq c \leq N$, in $O(\log_W N)$ time, where W is the word size.*

Proof: We construct an auxiliary array A that contains the x-coordinates of all points sorted in the ascending order of their ranks: $A[i]$ contains the x-coordinate of a point $p \in S$ with y-rank i for $1 \leq i \leq m$. Let S' denote the set of points with y-rank $j\Delta$ for $\Delta = \lceil \log N/\log W \rceil$ and $j = 1, 2, \ldots, \lfloor m/\Delta \rfloor$. The y-coordinates of all $p \in S'$ are stored in a fusion tree V' [Fredman and Willard 1990]. Since S' contains $\lceil m \log W/\log N \rceil$ points on $N \times N$ grid, V' uses $O(m \log W)$ bits of space. Array A needs $O(m \log m)$ bits.

To find the y-rank r_c of some integer c, we find the highest $p' \in S'$ such that $p'.y \leq c$. Using V', this can be done in $O(\log N/\log W)$ time, see [Fredman and Willard 1990]. Suppose that the y-rank of p' equals to $t\Delta$ for some $0 \leq t \leq \lfloor m/\Delta \rfloor$. Then $t\Delta \leq r_c \leq (t+1)\Delta$. For any point with y-rank r, $t\Delta \leq r \leq (t+1)\Delta$, we can find its x-coordinate using A. When the x-coordinate of a point is known, we can compute its y-coordinate and compare it with c in $O(1)$ time. Hence, we can find the y-rank r_c of c in $O(\log(\log N/\log W)) = O(\log\log_W N)$ time by binary search. □

THEOREM 8.2. *There exists a linear space data structure that answers three-sided point counting queries on $N \times U$ grid in $O(\log_W K)$ time, where $W = \log U$ is the word size.*

Proof: We change the data structure of Lemma 6.3 to support counting points instead of reporting colors.

Again, our base structure is the path-range tree with the following parameters. The set $\mathcal{N}_i$ contains all nodes on level $\lg^{(3)} N + f(i) \lg W$. Thus each $v \in \mathcal{N}_i$ has $W^{f(i)} \lg\lg N$ leaf descendants. Let $\overline{Y}_l(u,v)$ ($\overline{Y}_r(u,v)$) denote the list of all points from $\Pi_l(u,v)$ ($\Pi_r(u,v)$) sorted by their y-coordinates in ascending order. The set $\overline{L}_l(u,v)$ ($\overline{L}_r(u,v)$) for a node $v \in \mathcal{N}_i$ contains $m_i = W^{f(i)-1}$ lowest points from $\overline{Y}_l(u,v)$ ($\overline{Y}_r(u,v)$). For every list $\overline{L}_l(u,v)$, we store the y-coordinates of all its points in a fusion tree [Fredman and Willard 1990]. Thus we can find for any c the number of points $p \in \overline{L}_l(u,v)$, $p.y \le c$, in $O(\log(|\overline{L}_l(u,v))/\log W)$ time (we assume that the y-rank of any p in $\overline{L}_l(u,v)$ is known). We store the same structure for each $\overline{L}_r(u,v)$. Furthermore, let $last_l(u,v)$ and $last_r(u,v)$ denote the highest points in $\overline{L}_l(u,v)$ and $\overline{L}_r(u,v)$ respectively.

For every set $S(v)$ where $v \in \mathcal{N}_i$, we keep a data structure $\overline{D}(v)$, described in [JáJá et al. 2004], that supports range counting queries in $O(\log(|S(v)|)/\log W)$ time[4]. For every $v \in \mathcal{N}_1$ the data structure $\overline{D}(v)$ answers range counting queries on $S(v)$ in $O(1)$ time because $S(v)$ contains $o(W^2)$ points.

Coordinates of points $p \in \overline{D}(v)$ are replaced with their ranks. We can find the x-rank of any a, $1 \le a \le N$, in any set $S(v)$ as described in the proof of Lemma 6.3. The y-coordinates of points are stored in a global array Y; $Y[x] = p.y$ for the point p such that $p.x = x$. We can obtain the x-coordinate of each point $p \in S(v)$ if its x-rank in $S(v)$ is known. We also keep a data structure of Lemma 8.1 for each $v \in \mathcal{N}_i$ and all $\mathcal{N}_i$; using this data structure, we can find the y-rank of any integer c in $O(\log N/\log W)$ time. A data structure for computing the y-ranks in a set $S(v)$ uses $O(N(\log W + \log(|S(v)|))$ bits. All data structures for computing the y-ranks use $O(N \log\log_W N \log W + N\sum_{i=\log^{(3)} N}^{\log\log_{\log N} N}(\lg^{(3)} N + f(i)\lg\lg N)) = O(N\log N)$ bits. Since point coordinates are replaced by their ranks in $\overline{D}(v)$, each $\overline{D}(v)$ needs $O(|S(v)|\log(|S(v)|))$ bits and all $\overline{D}(v)$ need $O(N\log N)$ bits. Since all lists $\overline{L}_l(u,v)$, $\overline{L}_r(u,v)$ contain $O(N)$ elements, the total space usage of all lists is $O(N)$ words.

A query $Q = [a,b] \times [0,c]$ is answered as follows. We denote by l_a and l_b the leaves that contain values a and b. We denote by u their lowest common ancestor. Our goal is to count the number of points $p \in \Pi_r(u,l_a) \cup \Pi_l(u,l_b)$, such that $p.y \le c$. Points from $\Pi_l(u,l_b)$ can be counted as follows. Let v_i denote an ancestor of l_b such that $v_i \in \mathcal{N}_i$. We visit nodes v_i, $i = 1,\ldots,$ until we reach the node v_g, such that $last(v_g,u) > c$ or v_{g+1} is an ancestor of u. We count the number of points $p \in \Pi_l(u,v_g)$, $p.y \le c$, using the fusion tree on $\overline{L}_l(u,v_g)$. We count the number of points in $p \in \Pi_l(v_g,l_b)$, $p.y \le c$, using the data structure $\overline{D}(v_g)$. The total cost of this procedure is $O((\log(|\overline{L}_l(u,v_g)|) + \log(|S(v_g)|))/\log W)$. Since $last(u,v_{g-1}) \le c$, the query range Q contains $K \ge W^{f(g-1)}$ different points. $\overline{L}_l(u,v_g)$ and $S(v_g)$ contain $W^{f(g)}$ and $W^{f(g)}\log\log N$ points respectively. Therefore points $p \in \Pi_l(u,l_b)$, $p.y \le c$, are counted in $O(\log K/\log W)$ time. Points from $\Pi_r(u,l_a)$, such that $p.y \le c$ can be counted in a symmetric way. □

We can extend this result to the external memory model.

THEOREM 8.3. *There exists a linear space data structure that answers three-sided point counting queries on $N \times U$ grid in $O(\log_B K)$ I/Os.*

Proof: If $B \ge N$, we can read S into internal memory and answer a query in $O(1)$ I/Os. If $B \le \log N$, the result of Theorem 8.2 enables us to answer queries in $O(\log_W N) = O(\log_B N)$ I/Os. If $\log N < B < N$, we use the data structure of Theorem 8.2 with the following modifications. We replace W with B in the proofs of Lemma 8.1 and Theorem 8.2. The set $\mathcal{N}_i'$ now contains all nodes on level $\lg^{(3)} N + f(i)\lg B$ so that each

[4]In [JáJá et al. 2004] the authors consider a special case $W = \log N$ and present a data structure that answers queries in $O(\log N/\log\log N)$ time. Chan and Wilkinson [Chan and Wilkinson 2013] observed that the result of JaJa et al. [JáJá et al. 2004] can be extended to the case of any $W \ge \log N$ so that queries are answered in $O(\log N/\log W)$ time, where N is the number of points in a data structure.

$v \in \mathcal{N}'_i$ has $B^{f(i)} \lg\lg N$ leaf descendants. Each list $\overline{L}_l(u,v)$ and $\overline{L}_r(u,v)$ for $v \in \mathcal{N}_i$ contains $B^{f(i)-1}$ lowest points from $Y_l(u,v)$ and $Y_r(u,v)$ respectively. For every $S(v)$ where $v \in \mathcal{N}_i$, we store a data structure that uses $O(N \log_2 B)$ bits and computes the y-rank of any integer c in $O(\log_B N)$ I/Os; this data structure is implemented as in Lemma 8.1, but S' contains every $(\log_B N)$-th element in $S(v)$ and y-coordinates of points in S' are kept in a standard B-tree. All data structures for computing y-ranks use $O(N \log\log_B N \log B + N \log N) = O(N \log N)$ bits because $N > B > \log N$.

Besides, we replace other secondary data structures with external memory data structures. We implement $\overline{D}(v)$ using the result of Agarwal et al. [Agarwal et al. 2013], so that range counting queries on each set $S(v)$ are answered in $O(\log_B(|S(v)|))$ I/Os. Each range counting data structure uses $O(|S(v)| \log(|S(v)|))$ bits; hence all $\overline{D}(v)$ use $O(N \log B) \sum_{i=1}^{\log\log_B N} f(i) = O(N \log N)$ bits. Instead of fusion tree we maintain a standard B-tree on each $\overline{L}_l(u,v)$ and $\overline{L}_r(u,v)$, so that points $p \in \overline{L}_l(u,v)$ (or $p \in \overline{L}_r(u,v)$) satisfying $p.y \le c$ for some query value c can be counted in $O(\log_B(|\overline{L}_l(u,v)|))$ (respectively $O(\log_B(|\overline{L}_r(u,v)|))$) I/Os. All B-trees also need $O(N \log N)$ bits of space.

A query is processed in the same way as in Theorem 8.2. By the same argument as in the proof of Theorem 8.2, the cost of answering a query is $O(\log_B K)$ I/Os. $\square$

Furthermore our data structures can be also used to efficiently compute a lower bound on the number of points in a three-sided range. A τ-threshold point counting query reports the number of points K in the range if $K \le \tau$ for a parameter τ determined at query time; otherwise we report that the number of points in the query range exceeds τ without computing the exact value of K.

COROLLARY 8.4. *There exists a linear space data structure that answers three-sided point counting queries on $N \times U$ grid in $O(\log_W K)$ time, where $W = \log U$ is the word size. There exists a linear space external memory data structure that answers τ-threshold three-sided point counting queries on $N \times U$ grid in $O(\log_B \tau)$ I/Os.*

Proof: The data structures and the procedures for answering a query $[a,b] \times [0,c]$ are the same as in Theorems 8.2 and 8.3 respectively. However, we stop when the first node v_g such that $|\overline{L}_l(u,v_g)| > \tau$ is reached even if all points in $\overline{L}_l(u,v_g)$ are below c. $\square$

9. COLOR COUNTING IN ONE DIMENSION

Our results on three-sided point counting have important implications for one-dimensional color counting. We employ the reduction that was used in the proof of Lemma 2.2. Recall that $prev(p)$ denotes the rightmost point p_1 of the same color as p, such that p_1 is to the left of p. Let S be a set of colored one-dimensional points. We construct a set S' that contains exactly one point p' for each $p \in S$; $p'.x = p.x$ and $p'.y = prev(p).x$. Then the number of distinct colors in $([a,b]) \cap S$ equals the number of points in $([a,b] \times [0,a]) \cap S'$. We assume that points of S are on N-grid, i.e., that coordinates of points are positive integers bounded by $O(N)$. Two following results immediately follow from our bounds for adaptive and approximate three-sided point counting queries, stated in Theorems 7.2 and 8.2.

COROLLARY 9.1. *There exists a linear space data structure that answers one-dimensional color counting queries on N-grid in $O(\log_W K)$ time. There exists a linear space external memory data structure that answers color counting queries on N-grid in $O(\log_B K)$ I/Os.*

As shown in Corollary 8.4, the data structures of Corollary 9.1 can be used to answer τ-threshold one-dimensional color counting queries in $O(\log_w \tau)$ time and $O(\log_B \tau)$ I/Os respectively.

COROLLARY 9.2. *There exists an $O(N)$-words data structure for points on N-grid that returns an ε-approximation for the number of colors in a query range in $O(1)$ time.*

The results for N-grid can be used to report colors in an array: entries of an array A are assigned colors and stored in a linear-space data structure: for a query range $[i, j]$ such that $1 \leq i \leq j \leq N$, we can compute the number of colors in $A[i]$,$A[i+1]$, $\ldots$, $A[j]$ in $O(\log_B K)$ I/Os where K is the number of distinct colors in the query range. We can also obtain an estimate on the number of colors in $A[i]$,$A[i+1]$, $\ldots$, $A[j]$ in $O(1)$ time. Furthermore we can extend the results of Corollaries 9.1 and 9.2 to the case when point coordinates are bounded by parameter U using the rank-space technique described in the proof of Theorem 5.3; the query cost increases by an additive term $O(\log\log_B U)$ and the space usage remains unchanged.

10. CONCLUSIONS

In this paper we described several optimal data structures for one-dimensional color counting and three-sided point counting queries. We also presented efficient data structures for colored range reporting in external memory. All presented data structures are static; existence of a dynamic data structure that efficiently supports two-dimensional color queries, insertions, and deletions is an interesting open question. The main memory data structure of [Gupta et al. 1995] uses $O(N \log^2 N)$ words of space and supports color reporting queries and insertions in $O(\log^2 N + K)$ and $O(\log^3 N)$ time respectively. But even in the main memory, no data structure that supports both insertions and deletions in $\log^{O(1)} N$ and answers queries in $\log^{O(1)} N + O(K)$ time is currently known.

As a side note, we remark that fully-dynamic data structures exist for a somewhat similar problem of reporting elements in a colored list [Mortensen 2006]. In this problem we maintain a set of one-dimensional points whose colors are bounded by a polylogarithmic function of N; for a query range $[a, b] \times [c, d]$, we report all points in the range $[a, b]$ such that their colors are in the interval $[c, d]$. Data structures supporting such queries are an important tool for solving some two-dimensional geometric problems [Mortensen 2006; Blelloch 2008; Giyora and Kaplan 2009]. Although reporting in a colored list bears some resemblance to color reporting queries considered in this paper, there is an important difference: in the former problem points are reported instead of colors (in other words, we enumerate all objects, not just categories of objects).

We observed at the end of section 3 that our external memory data structures for color reporting (with exception of the structure in section 2) generate a list in which the same color may occur a constant number of times. The problem of duplicate colors can be easily solved in the main memory: we can remove duplicates from a list of K colors in $O(K)$ time. In the external memory we need $O((K/B)\log_{M/B}\frac{K}{B})$ I/Os to get rid of duplicates. Since in many situations $K = B^{O(1)}$ and usually $M = B^{1+\Omega(1)}$, we can often remove duplicates without increasing the query cost. It would also be interesting to design an external memory data structure that produces a list of colors in a query range, so that each relevant color occurs exactly once and a query is answered in $\log_B^{O(1)} N + O(K/B)$ I/Os for any K and B.

REFERENCES

AGARWAL, P. K., ARGE, L., GOVINDARAJAN, S., YANG, J., AND YI, K. 2013. Efficient external memory structures for range-aggregate queries. *Comput. Geom. 46,* 3, 358–370.

AGARWAL, P. K., GOVINDARAJAN, S., AND MUTHUKRISHNAN, S. 2002. Range searching in categorical data: Colored range searching on grid. In *Proc. 10th Annual European Symposium on Algorithms (ESA 2002)*. 17–28.

AGGARWAL, A. AND VITTER, J. S. 1988. The input/output complexity of sorting and related problems. *Commun. ACM 31,* 9, 1116–1127.

ALSTRUP, S., BRODAL, G. S., AND RAUHE, T. 2000. New data structures for orthogonal range searching. In *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*. 198–207.

ARGE, L., SAMOLADAS, V., AND VITTER, J. S. 1999. On two-dimensional indexability and optimal range search indexing. In *PODS*. 346–357.

BECKER, B., GSCHWIND, S., OHLER, T., SEEGER, B., AND WIDMAYER, P. 1996. An asymptotically optimal multiversion b-tree. *VLDB J. 5,* 4, 264–275.

BENTLEY, J. L. 1980. Multidimensional divide-and-conquer. *Commun. ACM 23,* 4, 214–229.

BLELLOCH, G. E. 2008. Space-efficient dynamic orthogonal point location, segment intersection, and range reporting. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*. 894–903.

BOZANIS, P., KITSIOS, N., MAKRIS, C., AND TSAKALIDIS, A. K. 1995. New upper bounds for generalized intersection searching problems. In *Proc. 22nd International Colloquium on Automata, Languages and Programming (ICALP 95)*. 464–474.

BOZANIS, P., KITSIOS, N., MAKRIS, C., AND TSAKALIDIS, A. K. 1997. New results on intersection query problems. *Computer Journal 40,* 1, 22–29.

CHAN, T. M. AND WILKINSON, B. T. 2013. Adaptive and approximate orthogonal range counting. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*. to be published.

CHAZELLE, B. 1988. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput. 17,* 3, 427–462.

DRISCOLL, J. R., SARNAK, N., SLEATOR, D. D., AND TARJAN, R. E. 1989. Making data structures persistent. *J. Comput. Syst. Sci. 38,* 1, 86–124.

FREDMAN, M. L. AND WILLARD, D. E. 1990. Blasting through the information theoretic barrier with fusion trees. In *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC'90)*. 1–7.

FREDMAN, M. L. AND WILLARD, D. E. 1994. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *J. Comput. Syst. Sci. 48,* 3, 533–551.

GABOW, H. N., BENTLEY, J. L., AND TARJAN, R. E. 1984. Scaling and related techniques for geometry problems. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC 1984)*. 135–143.

GAGIE, T., NAVARRO, G., AND PUGLISI, S. J. 2010. Colored range queries and document retrieval. In *Proc. 17th International Symposium on String Processing and Information Retrieval (SPIRE)*. 67–81.

GIYORA, Y. AND KAPLAN, H. 2009. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Transactions on Algorithms 5,* 3.

GUPTA, P., JANARDAN, R., AND SMID, M. H. M. 1995. Further results on generalized intersection searching problems: Counting, reporting, and dynamization. *Journal of Algorithms 19,* 2, 282–317.

GUPTA, P., JANARDAN, R., AND SMID, M. H. M. 1996. Algorithms for generalized halfspace range searching and other intersection searching problems. *Comput. Geom. 6*, 1–19.

JÁJÁ, J., MORTENSEN, C. W., AND SHI, Q. 2004. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *Proc. 15th International Symposium on Algorithms and Computation (ISAAC 2004)*. 558–568.

JANARDAN, R. AND LOPEZ, M. A. 1993. Generalized intersection searching problems. *International Journal of Computational Geometry and Applications 3,* 1, 39–69.

JØRGENSEN, A. G. AND LARSEN, K. G. 2011. Range selection and median: Tight cell probe lower bounds and adaptive data structures. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*. 805–813.

KAPLAN, H. 2005. Persistent data structures. In *Handbook on Data Structures and Applications, D. Mehta and S. Sahni (Editors)*. CRC Press 2001, 241–246.

KARPINSKI, M. AND NEKRICH, Y. 2008. Searching for frequent colors in rectangles. In *Proc. 20th Annual Canadian Conference on Computational Geometry (CCCG)*.

KARPINSKI, M. AND NEKRICH, Y. 2011. Top-k color queries for document retrieval. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*. 401–411.

LARSEN, K. G. AND PAGH, R. 2012. I/o-efficient data structures for colored range and prefix reporting. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 583–592.

LARSEN, K. G. AND VAN WALDERVEEN, F. 2013. Near-optimal range reporting structures for categorical data. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*. to be published.

MCCREIGHT, E. M. 1985. Priority search trees. *SIAM J. Comput. 14,* 2, 257–276.

MILTERSEN, P. B., NISAN, N., SAFRA, S., AND WIGDERSON, A. 1998. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci. 57,* 1, 37–49.

MORTENSEN, C. W. 2006. Fully dynamic orthogonal range reporting on ram. *SIAM J. Comput. 35,* 6, 1494–1525.

MUTHUKRISHNAN, S. 2002. Efficient algorithms for document retrieval problems. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 657–666.

NANOPOULOS, A. AND BOZANIS, P. 2003. Categorical range queires in large databases. In *Proc. 8th International Symposium on Advances in Spatial and Temporal Databases, (SSTD 2003)*. 122–139.

NEKRICH, Y. 2007. External memory range reporting on a grid. In *Proc. 18th International Symposium on Algorithms and Computation (ISAAC)*. 525–535.

NEKRICH, Y. 2009. Data structures for approximate orthogonal range counting. In *Proc. 20th International Symposium on Algorithms and Computation (ISAAC 2009)*. 183–192.

NEKRICH, Y. 2011. External memory orthogonal range reporting with fast updates. In *Proc. 22nd International Symposium on Algorithms and Computation (ISAAC)*. 395–404.

NEKRICH, Y. 2012. Space-efficient range reporting for categorical data. In *Proc. 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2012)*. 113–120.

NEKRICH, Y. AND NAVARRO, G. 2012. Sorted range reporting. In *Proc. 13th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2012)*. 271–282.

PĂTRAŞCU, M. 2007. Lower bounds for 2-dimensional range counting. In *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*. 40–46.

PĂTRAŞCU, M. AND THORUP, M. 2006. Time-space trade-offs for predecessor search. In *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC 2006)*. 232–240.

RAMASWAMY, S. AND SUBRAMANIAN, S. 1994. Path caching: A technique for optimal external searching. In *Proc. 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. 25–35.

SHI, Q. AND JÁJÁ, J. 2005. Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. *Inf. Process. Lett. 95,* 3, 382–388.

SUBRAMANIAN, S. AND RAMASWAMY, S. 1995. The p-range tree: A new data structure for range searching in secondary memory. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 378–387.

VAN EMDE BOAS, P., KAAS, R., AND ZIJLSTRA, E. 1977. Design and implementation of an efficient priority queue. *Mathematical Systems Theory 10*, 99–127.

VARMAN, P. J. AND VERMA, R. M. 1997. An efficient multiversion access structure. *IEEE Trans. Knowledge and Data Engeneering 9,* 3, 391–409.