Sorted Range Reporting Revisited

Gelin Zhou

David R. Cheriton School of Computer Science,
University of Waterloo, Canada.
g5zhou@uwaterloo.ca

Abstract. We consider the two-dimensional sorted range reporting problem. Our data structure requires $O(n \log \log n)$ words of space and $O(\log \log n + k \log \log n)$ query time, where $k$ is the number of points in the query range. This data structure improves a recent result of Nekrich and Navarro [8] by a factor of $O(\log \log n)$ in query time, and matches the state of the art for unsorted range reporting [1].

1 Introduction

The orthogonal range searching problems is well-known in the communities of computational geometry and data structures. For these problems, we need maintain a point set $S$ in $d$-dimensional space, such that certain functions over points in a given query rectangle $Q$ can be computed efficiently. In this paper we study a variant of the two-dimensional orthogonal range reporting problem, for which points in the query range are sorted in increasing order of their $x$-coordinates. In addition, our data structures can work in an online fashion: points from $S \cap Q$ are reported in increasing order of $x$-coordinates until the query-answering procedure is terminated or all points in $S \cap Q$ are output.

The sorted range reporting problem was proposed by Nekrich and Navarro [8]. Let $k$ denote the number of points in the given query range. Their linear space data structure requires $O(\log \epsilon n + k \log \epsilon n)$ query time, and their data structure with optimal query time occupies $O(n \log \epsilon n)$ words of space. These results match the state of the art for unsorted range reporting [1]. However, their data structure using $O(n \log \log n)$ words of space requires $O(\log^2 \log n + k \log^2 \log n)$ query time, which is slower than the corresponding result for unsorted range reporting [1] by a factor of $O(\log \log n)$. In this paper we present a data structure using the same amount of space but only $O(\log \log n + k \log \log n)$ query time.

The sorted range reporting problem is closely related to the orthogonal range successor problem (sometimes referred to as the range next-value problem) [4,9,8]. Nekrich and Navarro’s data structures [8] can be used directly to support range successor queries. Their linear space data structure requires $O(\log^2 n)$ query time, and their data structure achieving the optimal $O(\log \log n)$ query time occupies $O(n \log \epsilon n)$ words of space. Our data structure requires only $O(n \log \log n)$ words of space to achieve the optimal query time.

We assume that the given point set is in an $n \times n$ grid, or rank space. Every two points have different $x/y$-coordinates. The underlying computational model throughout this work is the standard word RAM model with word size $w = \Omega(\log n)$.

The rest of this paper is organized as follows: In Section 2 we review Chan, Larsen and Patrascu’s data structures [1] for (unsorted) range reporting and Nekrich and Navarro’s results [8] for the sorted variant. In Section 3 we present our data structures for sorted range reporting and range successor queries.

2 Preliminary

For completeness, we describe Chan et al.’s work [1] for unsorted range reporting in addition to Nekrich and Navarro’s work [8] for sorted range reporting. We may modify their presentations for the sake of consistency.

* This work is a course project for CS 763 - Computational Geometry, instructed by Timothy Chan.

1 Increasing/decreasing $x/y$-coordinate ordering can be easily supported via coordinate changes.
2.1 Unsorted Range Reporting

Chan et al.’s data structures \[1\] for range reporting are based on the wavelet tree \[2,6\]. A conceptual range tree \( T \) is built on \([1..n]\), where \( n \) is assumed to be a power of two. Every node \( v \) in \( T \) has an associated range. Let \( S(v) \) denote the set of points whose \( y \)-coordinates are in this range. It is clear that \( S(v) \) contains a point only for every leaf node \( v \) at the bottom of \( T \). An internal node \( v \) has two children \( v_l \) and \( v_r \), whose associated ranges are a disjoint union of that of \( v \). Points in \( S(v) \) are conceptually listed as \( S(v)[1], S(v)[2], \cdots \) in increasing order of \( x \)-coordinates. For each of these point, we write down a 0-bit if this point is also in \( S(v_l) \), or a 1-bit otherwise. These bits are concatenated and maintained as a bit vector, such that rank/select operations can be performed in constant time \[3\].

To achieve efficient query time, Chan et al. \[1\] formulated the following two operations as the ball-inheritance problem: Given a node \( v \) in \( T \) and a range \([a..b]\) on the \( x \)-axis, noderange\((v, a, b)\) returns the range \([a_v..b_v]\) such that \( S(v)[a_v] \) is the first one whose \( x \)-coordinate is \( \geq a \) and \( S(v)[b_v] \) is the last one whose \( x \)-coordinate is \( \leq b \). Given a node \( v \) in \( T \) and an index \( 1 \leq i \leq |S(v)| \), point\((v, i)\) returns the coordinates of \( S(v)[i] \). The following lemma addresses their results.

**Lemma 1** \([\text{2[1]}]\). Using \( O(nf(n)) \) words of space, one can support operations noderange\((v, a, b)\) and point\((v, i)\) with \( O(g(n) + \lg lg n) \) and \( O(g(n)) \) query time, respectively, where

\[
1. \quad f(n) = O(1) \text{ and } g(n) = O(\lg^* n);
2. \quad f(n) = O(\lg \lg n) \text{ and } g(n) = O(\lg \lg n);
3. \quad f(n) = O(\lg^* n) \text{ and } g(n) = O(1).
\]

A range reporting query \( Q = [a..b] \times [c..d] \) over the point set \( S \) can be answered as follows. First we find node \( v \) in \( T \), which is the lowest common ancestor of the leaf nodes that correspond to \( c \) and \( d \). Let \( v_l \) and \( v_r \) denote the children of \( v \). It is clear that the associated range of \( v \) contains \([c..d]\), and the associated ranges of \( v_l \) and \( v_r \) both intersect \([c..d]\). Therefore, \( S \cap Q \) is decomposed into \( S(v_l) \cap ([a..b] \times \infty] \) and \( S(v_r) \cap ([a..b] \times (-\infty,d]) \). We consider how to support \( S(v_r) \cap ([a..b] \times \infty] \) only. We compute \([a_v..b_v]\) using noderange\((v_r, a, b)\). Thus, we need only report all the points in \( S(v_r)[a_v..b_v] \) whose \( y \)-coordinates \( \leq d \). To perform this step efficiently, an index for range minimum queries over \( S(v_r) \) is built. This index returns the position of the point with the smallest \( y \)-coordinate in \( S(v_r)[i..j] \) using constant time and \( 2|S(v_r)| + o(|S(v_r)|) \) bits of space \[5\]. The following paragraph shows how to report \( k \) points in \( kg(n) \) time:

Initially, we set \([i..j] = [a_v..b_v]\). We query \([i..j]\) on the index for range minimum queries, and \( l \) is returned. Using Lemma \[1\] we verify if the \( y \)-coordinate of \( S(v_r)[l] \) is no greater than \( d \). We terminate if the \( y \)-coordinate is greater. Otherwise, we report \( S(v_r)[l] \) and recurse on \([i..l - 1]\) and \([l + 1..j]\).

It is noteworthy that the first point returned by this algorithm is the lowest point in \( S(v_r) \cap Q \). On the other hand, the first point returned by the other 3-sided query is the highest point in \( S(v_l) \cap Q \). We cannot simply obtain the lowest or highest point in \( S \cap Q \).

The following lemma summarizes the discussions in this section.

**Lemma 2** \([\text{[1]}]\). Using \( O(nf(n)) \) words of space, one can support two-dimensional range reporting queries with \( O(\lg \lg n + g(n) + kg(n)) \) query time, where

\[
1. \quad f(n) = O(1) \text{ and } g(n) = O(\lg^* n);
2. \quad f(n) = O(\lg \lg n) \text{ and } g(n) = O(\lg \lg n);
3. \quad f(n) = O(\lg^* n) \text{ and } g(n) = O(1).
\]

2.2 Suboptimal Range Successor

Nekrich and Navarro \[8\] showed that Chan et al.’s data structures \[1\] can support range successor queries after certain modifications. For simplicity, we swap \( x \)-/\( y \)-coordinates and return the lowest point (i.e., the one with the smallest \( y \)-coordinate) in the query range \( Q = [a..b] \times [c..d] \). Let \( \pi \) denote the path from the root of the conceptual range tree \( T \) to the leaf that corresponds to \( c \). The basic idea is to find the lowest node \( v_f \) on \( \pi \) such that \( S(v_f) \cap Q \neq \emptyset \). Let \( v \) denote the lowest common ancestor of the leaf nodes that
correspond to $c$ and $d$. It is clear that, for every node $u$ on $\pi$ that is deeper than $v$, its associated range contains $c$ but not $d$. It implies that $S(u) \cap Q = S(u) \cap ([a..b] \times [c..\infty))$. One can determine if $S(u) \cap Q \neq \phi$ by determining if the 3-sided query contains any point. Therefore $v_f$ can be computed using binary search on $\pi$. This requires to compute $O(\log \log n)$ 3-sided emptiness queries.

If $v_f$ is a leaf node, then the only point in $S(v_f)$ is the answer. If $v_f$ is an internal node, the child of $v_f$ on $\pi$ must be the left child. Otherwise the associated range of the left child would not intersect $[c..d]$, and the assumption on $v_f$ would be contracted. Since the left child of $v_f$ does not correspond to any point in the query range $Q$, the right child must correspond to at least a point in $Q$. Using the algorithm described in the previous section, we can find such a point using range minimum queries. The first point returned is by chance the lowest one.

The following lemma summarizes the above discussions.

**Lemma 3 ([8]).** Using $O(n f(n))$ words of space, one can support two-dimensional range successor queries with $O(g(n) \log \log n)$ query time, where

1. $f(n) = O(1)$ and $g(n) = O(\log^* n)$;
2. $f(n) = O(\log \log n)$ and $g(n) = O(\log \log n)$.

One can support sorted range reporting queries using range successor queries. Suppose the lowest point in $Q = [a..b] \times [c..d]$ has $y$-coordinate $p$. We can find the second lowest point in $Q$ by querying $Q' = [a..b] \times [p + 1..d]$. The procedure is repeated until the query range contains no point. Thus we have the following lemma:

**Lemma 4 ([8]).** Using $O(n f(n))$ words of space, one can support two-dimensional sorted range reporting queries with $O(\log \log n (g(n) + k g(n)))$ query time, where

1. $f(n) = O(1)$ and $g(n) = O(\log^* n)$;
2. $f(n) = O(\log \log n)$ and $g(n) = O(\log \log n)$.

## 3 Optimal Range Successor Queries in Less Space

We present the main result in this paper: a data structure for range successor queries using $O(n \log \log n)$ words of space and $O(\log \log n)$ query time. This data structure is obtained by modifying Nekrich and Navarro’s third data structure [8] for sorted range reporting. We consider 3-sided range successor queries, for which the leftmost point in $S \cap ([a..b] \times [\infty..d])$ is returned. A preliminary result of Nekrich and Navarro is addressed in the following lemma.

**Lemma 5 (Lemma 5 in [8]).** Given a set of $n$ points, one can support 3-sided range successor queries using $O(n \log^3 n)$ bits of space and $O(\log \log n)$ query time.

Now we describe our data structures. We first build the same data structures as the second variant of Lemma 2. Let $Q' = [a..b] \times [\infty..d]$ denote a 3-sided query range. We further construct auxiliary data structures on every $S(v)$ such that the leftmost point in $S(v) \cap Q'$ can be returned in $O(\log \log n)$ time, using $O(|S(v)| \log \log n)$ bits of additional space.

As we have mentioned, points in $S(v)$ are conceptually listed in increasing order of $x$-coordinates. These points are divided into blocks $B_1(v), B_2(v), \ldots$ of size $|\log^3 n|$ (the last block may contain less). $D(v)$ stores the lowest point in each block explicitly, and is maintained using Lemma 3 such that 3-sided range successor queries on $D(v)$ can be supported in $O(\log \log n)$ time. This auxiliary data structure occupies $O(|D(v)| \log^3 |D(v)|) = O(|S(v)|)$ bits of space.

For some constant $0 < \epsilon < 1$, points in every block $B_i(v)$ are further divided into sub-blocks $SB_{1,1}(v), SB_{1,2}(v), \ldots$ of size $|\log^\epsilon n|$ (the last sub-block may contain less). In addition, their $x/y$-coordinates are rewritten as the $x/y$-ranks within this block. A rank can be represented in $O(\log \log n)$ bits, and all the ranks require $O(|S(v)| \log \log n)$ bits over all blocks. $E_i(v)$ stores the lowest point in every sub-block explicitly, and is also maintained using Lemma 5 to support 3-sided range successor queries on $E_i(v)$ in $O(\log \log \log n)$ time. This
auxiliary data structure requires only \( O(|E_i(v)| \log^3 |E_i(v)|) = O(|B_i(v)| \log^3 n / |E_i(v)|) = o(|B_i(v)|) \) bits of additional space. Thus the space cost of all \( E_i(v) \)'s is \( o(|S(v)|) \) bits.

Now we show how to answer the 3-sided range successor query \( Q' = [a..b] \times [-\infty..d] \) over \( S(v) \). First we compute \([a_v..b_v]\) using nonderange\((v,a,b)\), which requires \( O(\log \log n) \) time. Let \( B_i(v) \) and \( B_j(v) \) be the blocks that contains \( a_v \) and \( b_v \), respectively. Thus \([a_v..b_v]\) spans over blocks \( B_i(v), \ldots, B_j(v) \). We first attempt to find the leftmost point in \( B_i(v) \cap Q' \) (we will show how to do it later). If such a point exists, our algorithm terminates and the point is returned. Otherwise, we query \( D(v) \) to find the leftmost block among \( B_{i+1}(v), \ldots, B_{j-1}(v) \) that intersect \( Q' \). Let \( B_i(v) \) denote the block. We query \( B_i(v) \cap Q' \) and returns the result. If such \( l \) does not exist, we query \( B_j(v) \cap Q' \).

The remaining issue is to find the leftmost point in \( B_l(v) \) that is contained in a given 3-sided range \( Q' \) efficiently. We first compute the ranks of \( a, b \) and \( d \) within this block. Let them be \( a', b' \) and \( d' \), respectively. \( a' \) and \( b' \) can be computed directly from \( a_v \) and \( b_v \). The computation of \( d' \) requires succinct indices for predecessor search \([1]\), using \( O(\log \log n) \) time. Similar to the procedure described in the previous paragraph, we find the leftmost point in \( B_i(v) \cap ([a'..b'] \times [-\infty..d']) \) in \( O(\log \log n) \) time. The only difference is that we find the leftmost point within a sub-block using table-lookup. This can be done using constant time and a global lookup table of \( o(n) \) bits of additional space, since there are only \( 2^{|\log^\epsilon n|} \times O(\log \log n) = O(n^{1-\delta}) \) different sub-blocks, for some positive constant \( \delta \).

Summarizing the discussion above, we can find the leftmost point in \( S(v) \cap Q' \) in \( O(\log \log n) \) time. We also construct auxiliary data structures on every \( S(v) \) for 3-sided range successor queries of the form \([a..b] \times [c..\infty] \). Combining both parts and following the approach in Section \([2]\), we can find the leftmost point in a 4-sided query range in \( O(\log \log n) \) time.

**Theorem 1.** Using \( O(n \log \log n) \) words of space, one can support two-dimensional range successor queries in \( O(\log \log n) \) time. Repeatedly using this data structure, one can support two-dimensional sorted range reporting queries in \( O(\log \log n + k \log \log n) \) time, where \( k \) is the size of output.

**References**

1. Chan, T.M., Larsen, K.G., Patrascu, M.: Orthogonal range searching on the ram, revisited. In: Symposium on Computational Geometry. pp. 1–10 (2011)
2. Chazelle, B.: A functional approach to data structures and its use in multidimensional searching. SIAM J. Comput. 17(3), 427–462 (1988)
3. Clark, D.R., Munro, J.I.: Efficient suffix trees on secondary storage (extended abstract). In: SODA. pp. 383–391 (1996)
4. Crochemore, M., Iliopoulos, C.S., Kubica, M., Rahman, M.S., Walen, T.: Improved algorithms for the range next value problem and applications. In: STACS. pp. 205–216 (2008)
5. Fischer, J., Heun, V.: Space-efficient preprocessing schemes for range minimum queries on static arrays. SIAM J. Comput. 40(2), 465–492 (2011)
6. Grossi, R., Gupta, A., Vitter, J.S.: High-order entropy-compressed text indexes. In: SODA. pp. 841–850 (2003)
7. Grossi, R., Orlandi, A., Raman, R., Rao, S.S.: More haste, less waste: Lowering the redundancy in fully indexable dictionaries. In: STACS. pp. 517–528 (2009)
8. Nekrich, Y., Navarro, G.: Sorted range reporting. In: SWAT. pp. 271–282 (2012)
9. Yu, C.C., Hon, W.K., Wang, B.F.: Improved data structures for the orthogonal range successor problem. Comput. Geom. 44(3), 148–159 (2011)