Information Processing Letters 116 (2016) 171–174

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Two-dimensional range successor in optimal time and almost linear space

Gelin Zhou

David R. Cheriton School of Computer Science, University of Waterloo, Canada

ARTICLE INFO

Article history:
Received 30 March 2015
Accepted 5 September 2015
Available online 21 October 2015
Communicated by X. Wu

Keywords:
Data structures
Computational geometry
Range successor
Sorted range reporting
Planar orthogonal skyline reporting

ABSTRACT

In this article, we revisit the problem of supporting two-dimensional range successor queries. We present a data structure with $O(n \lg\lg n)$ words of space and $O(\lg\lg n)$ query time. This improves the work of Nekrich and Navarro (2012) by a factor of $\lg\lg n$ in query time, or a factor of $\lg^{\epsilon} n$ in space cost, where ϵ is an arbitrary positive constant. Our data structure matches the state of the art for two-dimensional range emptiness queries and achieves the optimal query time. This structure has fruitful applications in computational geometry and text indexing.

1. Introduction

The problems of supporting range aggregate queries are well-known in the communities of computational geometry and data structures. In these problems, we maintain an input point set S, so that, for any query range Q, certain functions over $S \cap Q$ can be computed efficiently. In this article we study the support for two-dimensional orthogonal *range successor* queries, or *range next-value* queries [12], for which the input point set is on the plane, and a query asks for the leftmost point in the orthogonal query range. This type of queries generalizes both *predecessor search* queries and two-dimensional orthogonal *range emptiness* queries.

As pointed out by Nekrich and Navarro [11] and Lewenstein [10], range successor queries have fruitful applications in the field of text indexing. However, only a few results have been presented for this problem [5,12,11]. The best ones have been obtained by Nekrich and Navarro [11]. Their linear space data structure requires $O(\lg^{\epsilon} n)$ query time, their structure using $O(n \lg\lg n)$ words of space requires $O(\lg^2 \lg n)$ query time, and their structure achieving the optimal $O(\lg\lg n)$ query time occupies $O(n \lg^{\epsilon} n)$ words of space, where ϵ is an arbitrary positive constant.

As a variant of the classical unsorted *range reporting* problem, the *sorted range reporting* problem is closely related to range successor queries. In the sorted range reporting problem, points in the orthogonal query range are reported in increasing order of their x-coordinates.[1] In addition, the query-answering procedure should work in an online fashion: points are reported in increasing order of x-coordinates until the procedure is terminated or all points in the query range are reported. This problem was proposed by Nekrich and Navarro [11]. Let k denote the number of points in the given query range. Their linear space data structure requires $O(\lg^{\epsilon} n + k \lg^{\epsilon} n)$ query time, and their data structure with the optimal $O(\lg\lg n + k)$ query time occupies $O(n \lg^{\epsilon} n)$ words of space. These results match the state of the art for unsorted range reporting [2]. However, their data structure using $O(n \lg\lg n)$ words of space requires $O(\lg^2 \lg n + k \lg^2 \lg n)$ query time,

[1] Increasing/decreasing x/y-coordinate ordering can be easily supported via coordinate changes.

E-mail address: g5zhou@uwaterloo.ca.

http://dx.doi.org/10.1016/j.ipl.2015.09.002
0020-0190/© 2015 Elsevier B.V. All rights reserved.

which is slower than the corresponding result for unsorted range reporting [2] by a factor of $\lg\lg n$.

Planar orthogonal skyline reporting queries play an important role in database systems for multi-criteria optimization [9]. As pointed out by Brodal and Larsen [1], this type of queries can be solved by calling range successor queries recursively. Thus Nekrich and Navarro's [11] range successor structures can directly support planar orthogonal skyline reporting queries within the same amount of space cost and $k+1$ times the amount of query time, where k is the size of output. Brodal and Larsen [1] further designed two data structures based on a different idea. Their first structure occupies $O(n\lg^{\epsilon} n)$ words of space and answers a query within $O(\lg n/\lg\lg n+k)$ query time, and the second one supports queries using $O(\lg n/\lg\lg n + k\lg\lg n)$ query time and $O(n\lg\lg n)$ words of space.

In this article we present a word-RAM data structure for range successor queries that uses $O(n\lg\lg n)$ words of space and the optimal $O(\lg\lg n)$ query time. Previous data structures using the same amount of space require $O(\lg^2\lg n)$ query time, and previous data structures achieving the same query time occupy $O(n\lg^{\epsilon} n)$ words of space [11]. Our data structure also supports sorted range reporting queries and planar orthogonal skyline reporting queries in $O(\lg\lg n + k\lg\lg n)$ time, while previous data structures using the same amount of space require $O(\lg^2\lg n + k\lg^2\lg n)$ query time for sorted reporting [11], and $O(\lg n/\lg\lg n + k\lg\lg n)$ or $O(\lg^2\lg n + k\lg^2\lg n)$ query time for skyline reporting [1,11].

The underlying computational model throughout this work is the standard word RAM model with word size $w = \Omega(\lg n)$. 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 rest of this article is organized as follows: In Section 2 we review Chan, Larsen and Pătraşcu's data structures [2] for unsorted range reporting and Nekrich and Navarro's results [11] for the sorted variant. In Section 3 we present our data structures for range successor queries.

2. Preliminary

For completeness, we describe Chan et al.'s work [2] for unsorted range reporting in addition to Nekrich and Navarro's work [11] for sorted range reporting. We may modify their presentations for the sake of consistency.

2.1. Unsorted range reporting

Chan et al.'s data structures [2] for range reporting are based on the *wavelet tree* [3,7]. A conceptual range tree $\mathcal{T}$ is built on $[1..n]$, where n is assumed to be a power of two. Every node v in $\mathcal{T}$ has an associated range. Let $S(v)$ denote the set of points whose y-coordinates are in this range; clearly $S(v)$ contains only one point for every leaf node v at the bottom of $\mathcal{T}$. An internal node v has two children v_ℓ and v_r, whose associated ranges form a disjoint union of v's associated range. Points in $S(v)$ are conceptually listed as $S(v)[1], S(v)[2], \ldots$ in increasing order of x-coordinates. For each of these points, we write down a 0-bit if this point is also in $S(v_\ell)$, 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 [4].

To achieve efficient query time, Chan et al. [2] formulated the following two operations as the *ball-inheritance problem*: Given a node v in $\mathcal{T}$ 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 no smaller than a and $S(v)[b_v]$ is the last one whose x-coordinate is no greater than b. Given a node v in $\mathcal{T}$ and an index $1 \le i \le |S(v)|$, $point(v, i)$ returns the coordinates of $S(v)[i]$. The following lemma addresses their results.

Lemma 2.1. *(See [3,2].) Using $O(nf(n))$ words of space, one can support $noderange(v, a, b)$ and $point(v, i)$ within $O(g(n) + \lg\lg n)$ and $O(g(n))$ query time, respectively, where*

1. *$f(n) = O(1)$ and $g(n) = O(\lg^{\epsilon} n)$;*
2. *$f(n) = O(\lg\lg n)$ and $g(n) = O(\lg\lg n)$; or*
3. *$f(n) = O(\lg^{\epsilon} n)$ 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 $\mathcal{T}$, 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, $S \cap Q$ can be decomposed into $S(v_\ell) \cap ([a..b] \times [c..\infty])$ and $S(v_r) \cap ([a..b] \times [-\infty..d])$. It is sufficient to show how to support $S(v_r) \cap ([a..b] \times [-\infty..d])$ only. Setting $[a_{v_r}..b_{v_r}]$ to be $noderange(v_r, a, b)$, we only need to report the points in $S(v_r)[a_{v_r}..b_{v_r}]$ whose y-coordinates are no greater than d. To perform this step efficiently, an indexing structure for range minimum queries over $S(v_r)$ is built. Given $1 \le i \le j \le |S(v)|$, this structure returns the position of the point with the smallest y-coordinate in $S(v_r)[i..j]$ using $O(1)$ time and $2|S(v_r)| + o(|S(v_r)|)$ bits of space [6]. The following procedure reports k points in $O(kg(n))$ time:

Setting $[i..j] = [a_{v_r}..b_{v_r}]$ initially, we query the smallest y-coordinate in $S(v_r)[i..j]$, which is assumed to be of index p. Using Lemma 2.1, we retrieve the y-coordinate of $S(v_r)[p]$. The procedure terminates if the y-coordinate is greater than d; otherwise $S(v_r)[p]$ is reported and recursive calls on $[i..p-1]$ and $[p+1..j]$ are made.

The following lemma summarizes the discussions in this section.

Lemma 2.2. *(See [2].) 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 k is the output size,*

1. *$f(n) = O(1)$ and $g(n) = O(\lg^{\epsilon} n)$;*
2. *$f(n) = O(\lg\lg n)$ and $g(n) = O(\lg\lg n)$; or*
3. *$f(n) = O(\lg^{\epsilon} n)$ and $g(n) = O(1)$.*

Remark. It is noteworthy that the first point returned by this algorithm is the lowest point in $S(v_r) \cap Q$. However, the first point returned by the other 3-sided query is the highest point in $S(v_\ell) \cap Q$. That being said, this algorithm cannot directly retrieve the lowest or highest point in $S \cap Q$.

2.2. Suboptimal range successor

Nekrich and Navarro [11] showed that Chan et al.'s data structures [2] could support range successor queries after certain modifications. For simplicity, x-/y-coordinates are swapped and a query returns the lowest point (i.e., the one with the smallest y-coordinate) in the query range $Q = [a..b] \times [c..d]$. Let π denote the path from the root of the conceptual range tree $\mathcal{T}$ to the leaf that corresponds to c. The basic idea is to find the lowest node v_f on π such that $S(v_f) \cap Q \neq \phi$. 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 π 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 π. This requires to compute $O(\lg\lg n)$ 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 π 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 contradicted. 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 2.3. *(See [11].) Using $O(nf(n))$ words of space, one can support two-dimensional range successor queries with $O(g(n)\lg\lg n)$ query time, where*

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

3. Range successor in optimal query time

Our data structure for range successor queries is obtained by modifying Nekrich and Navarro's [11] approach to sorted range reporting. More precisely, instead of performing binary searches on the root-to-leaf path π, we will consider and support 3-sided range successor queries, for which the leftmost point in $S \cap ([a..b] \times [-\infty..d])$ is returned. With a different partition strategy for points in $S(v)$ and appropriate use of Grossi et al.'s [8] predecessor search structure, our index for 3-sided range successor queries requires less space than that of Nekrich and Navarro. The following lemma shows a preliminary result for our data structure.

Lemma 3.1. *(See Lemma 5 in [11].) Given a set of n points, one can support 3-sided range successor queries using $O(n\lg^3 n)$ bits of space and $O(\lg\lg n)$ query time.*

Now we describe our data structures. We first build the same data structures as the second variant of Lemma 2.2. 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 $O(\lg\lg n)$ time, using $O(|S(v)|\lg\lg n)$ 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 $\lceil \lg^3 n \rceil$ (the last block may contain less). $D(v)$ stores the lowest point in each block explicitly, and is maintained using Lemma 3.1 such that 3-sided range successor queries on $D(v)$ can be supported in $O(\lg\lg n)$ time. This auxiliary data structure occupies $O(|D(v)|\lg^3|D(v)|) = O(|S(v)|)$ bits of space.

For some constant $0 < \epsilon < 1$, the points in every block $B_i(v)$ are further divided into sub-blocks $SB_{i,1}(v)$, $SB_{i,2}(v), \ldots$ of size $\lceil \lg^\epsilon n \rceil$ (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 $O(\lg\lg n)$ bits, and all the ranks require $O(|S(v)|\lg\lg n)$ bits over all blocks. $E_i(v)$ stores the lowest point in every sub-block explicitly, and is also maintained using Lemma 3.1 to support 3-sided range successor queries on $E_i(v)$ in $O(\lg\lg\lg n)$ time. This auxiliary data structure requires only $O(|E_i(v)|\lg^3|E_i(v)|) = O(|B_i(v)|\lg^3\lg n/\lceil \lg^\epsilon n \rceil) = o(|B_i(v)|)$ bits of additional space. Thus the space cost over 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 *noderange*(v, a, b), which requires $O(\lg\lg n)$ time. Let $B_{i_1}(v)$ and $B_{i_2}(v)$ be the block that contains a_v and b_v, respectively. That being said, $[a_v..b_v]$ spans over blocks $B_{i_1}(v), \ldots, B_{i_2}(v)$. We only consider the case in which $i_1 < i_2$; the remaining cases can be handled similarly. We first attempt to find the leftmost point in $B_{i_1}(v) \cap Q'$ (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 $B_{i_1+1}(v), \ldots, B_{i_2-1}(v)$ that intersects Q'. Let $B_t(v)$ denote the block. We query $B_t(v) \cap Q'$ and obtain the result. If such $B_t(v)$ does not exist, we query $B_{i_2}(v) \cap Q'$.

The remaining issue is to find the leftmost point of the intersection of a single block $B_t(v)$ and the given 3-sided range Q' efficiently, for which we have the following lemma:

Lemma 3.2. *The leftmost point in $B_t(v) \cap Q'$ can be found using $O(\lg\lg n)$ time and $O(|B_t(v)|\lg\lg n)$ bits of extra space in addition to a global lookup table of size $o(n)$ bits.*

Proof. We first compute the ranks of a, b and d within this block. Let them be a', b' and d', respectively. Here a' and b' can be computed directly from a_v and b_v in constant time. The computation of d' could be done by performing binary search on the points of $B_t(v)$. However, this would require $O(\lg^2 \lg n)$ time, since $point(v, i)$ would be called $O(\lg \lg n)$ times. Instead, we make use of Grossi et al.'s [8, Lemma 3.3] succinct indices for the y-coordinates of points in $B_t(v)$. As mentioned in Chan et al.'s work [2], this succinct index requires $O(\lg \lg n)$ bits of extra space per point, and supports predecessor search using $O(\lg \lg n)$ time in addition to $O(1)$ calls to $point(v, i)$. Hence, the value of d' can be determined in $O(\lg \lg n)$ time.

Let $SB_{t,j_1}(v)$ and $SB_{t,j_2}(v)$ be the sub-blocks that contain a' and b', respectively. As the procedure described previously, we only consider the case in which $j_1 < j_2$. Let Q_t denote the query $[a'..b'] \times [-\infty..d']$ within the block $B_t(v)$. We find and return the leftmost point in $SB_{t,j_1}(v) \cap Q_t$ if such a point exists. Otherwise, we query $E_t(v)$ to find the leftmost sub-block among $SB_{t,j_1+1}(v), \ldots, SB_{t,j_2-1}(v)$ that intersects Q_t. Let $SB_{t,p}(v)$ denote the block; clearly the answer is the leftmost point in $SB_{t,p}(v) \cap Q_t$. If such $B_t(v)$ does not exist, we query $SB_{t,j_2}(v) \cap Q_t$. In all these cases, we need only to find the leftmost point within a sub-block. This can be done using $O(1)$ time and a global lookup table of $o(n)$ bits of additional space, since there are only $2^{\lceil \lg^\epsilon n \rceil \times O(\lg \lg n)} = O(n^{1-\delta})$ different sub-blocks, for some constant $\delta > 0$. $\square$

Summarizing the discussion above, we can find the leftmost point in $S(v) \cap Q'$ in $O(\lg \lg n)$ time using $O(\lg \lg n)$ bits of extra space per point and a global lookup table of size $o(n)$ bits. 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 2.1, we can find the leftmost point in a 4-sided query range in $O(\lg \lg n)$ time. Therefore we have the following theorem:

Theorem 3.3. *Under the word RAM model with word size $w = \Omega(\lg n)$, a set of n planar points in rank space can be maintained using $O(n \lg \lg n)$ words of space, so that two-dimensional range successor queries can be supported in $O(\lg \lg n)$ query time. Repeatedly using this data structure, two-dimensional sorted range reporting queries and planar orthogonal skyline reporting queries can be answered in $O(\lg \lg n + k \lg \lg n)$ time, where k is the size of output.*

References

[1] Gerth Stølting Brodal, Kasper Green Larsen, Optimal planar orthogonal skyline counting queries, in: Algorithm Theory – SWAT 2014 – Proceedings of the 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2–4, 2014, 2014, pp. 110–121.

[2] Timothy M. Chan, Kasper Green Larsen, Mihai Pătraşcu, Orthogonal range searching on the RAM, revisited, in: Proceedings of the 27th ACM Symposium on Computational Geometry, Paris, France, June 13–15, 2011, 2011, pp. 1–10.

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

[4] David R. Clark, J. Ian Munro, Efficient suffix trees on secondary storage (extended abstract), in: Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Atlanta, Georgia, 28–30 January, 1996, 1996, pp. 383–391.

[5] Maxime Crochemore, Costas S. Iliopoulos, Marcin Kubica, Mohammad Sohel Rahman, Tomasz Walen, Improved algorithms for the range next value problem and applications, in: STACS 2008, Proceedings of the 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21–23, 2008, 2008, pp. 205–216.

[6] Johannes Fischer, Volker Heun, Space-efficient preprocessing schemes for range minimum queries on static arrays, SIAM J. Comput. 40 (2) (2011) 465–492.

[7] Roberto Grossi, Ankur Gupta, Jeffrey Scott Vitter, High-order entropy-compressed text indexes, in: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, USA, January 12–14, 2003, 2003, pp. 841–850.

[8] Roberto Grossi, Alessio Orlandi, Rajeev Raman, S. Srinivasa Rao, More haste, less waste: lowering the redundancy in fully indexable dictionaries, in: Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, Freiburg, Germany, February 26–28, 2009, 2009, pp. 517–528.

[9] Casper Kejlberg-Rasmussen, Yufei Tao, Konstantinos Tsakalidis, Kostas Tsichlas, Jeonghun Yoon, I/o-efficient planar range skyline and attrition priority queues, in: Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA, June 22–27, 2013, 2013, pp. 103–114.

[10] Moshe Lewenstein, Orthogonal range searching for text indexing, in: Space-Efficient Data Structures, Streams, and Algorithms – Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday, 2013, pp. 267–302.

[11] Yakov Nekrich, Gonzalo Navarro, Sorted range reporting, in: Algorithm Theory – SWAT 2012 – Proceedings of the 13th Scandinavian Symposium and Workshops, Helsinki, Finland, July 4–6, 2012, 2012, pp. 271–282.

[12] Chih-Chiang Yu, Wing-Kai Hon, Biing-Feng Wang, Improved data structures for the orthogonal range successor problem, Comput. Geom. 44 (3) (2011) 148–159.