On Generalized Planar Skyline and Convex Hull Range Queries

Nadeem Moidu, Jatin Agarwal, Sankalp Khare,
Kishore Kothapalli, and Kannan Srinathan

Center for Security, Theory and Algorithmic Research (CSTAR),
IIIT Hyderabad (India)

Abstract. We present output sensitive techniques for the generalized reporting versions of the planar range maxima problem and the planar range convex hull problem. Our solutions are in the pointer machine model, for orthogonal range queries on a static point set. We solve the planar range maxima problem for two-sided, three-sided and four-sided queries. We achieve a query time of $O(\log n + c)$ using $O(n)$ space for the two-sided case, where n denotes the number of stored points and c the number of colors reported. For the three-sided case, we achieve query time $O(\log^2 n + c \log n)$ using $O(n)$ space while for four-sided queries we answer queries in $O(\log^3 n + c \log^2 n)$ using $O(n \log n)$ space. For the planar range convex hull problem, we provide a solution that answers queries in $O(\log^2 n + c \log n)$ time, using $O(n \log^2 n)$ space.

1 Introduction

Generalized intersection searching was introduced by Janardan and Lopez [14]. Since then there has been a considerable amount of work on generalized searching and reporting problems [9][10][11][3][12][13][1][7][22][20]. A comprehensive survey of developments in the area can be found in [8]. In the generalized version of a problem, points are associated with colors. Colors capture the idea of membership, dividing objects into groups based on some common property. Such categorization has practical applications in databases, spatial information systems and other areas where objects are separable into classes and queries involve membership checking in these classes.

We present here solutions for the generalized (colored) range-query versions of two classic problems in computational geometry – *Convex Hull* and *Skyline*[1] – in the two dimensional setting, for a static set of points. Both these problems, in our knowledge, have not been tackled before in literature. Generalized intersection searching problems are broadly divided into two kinds, *reporting* and *counting*. In the former, the goal is to report the distinct colors whose points fall in the query range, while in the latter the goal is to count the number of such colors. Our solutions are for the reporting versions of both problems.

[1] We will use the terms *maxima* and *skyline* interchangeably in this paper.

S.P. Pal and K. Sadakane (Eds.): WALCOM 2014, LNCS 8344, pp. 34–43, 2014.

1.1 Generalized Planar Range Maxima Queries

A point $p = (x, y)$ is said to *dominate* another point $p' = (x', y')$ if both $x \geq x'$ and $y \geq y'$ are true. Given a set of points P in the plane, the set $m \subseteq P$ of all points which are not dominated by any other point in P is known as the maximal set. In the planar range maxima problem, given a range query q, we are asked to report the maximal set of the points in $P \cap q$. Note that range maxima queries are sometimes referred to as range skyline queries. We present a solution for the *reporting* version of the problem, where the points must be preprocessed in a way such that given an orthogonal range query q, we can efficiently report the distinct colors on the maximal set of $P \cap q$.

1.2 Generalized Planar Range Convex Hull Queries

In the planar convex hull range query problem, given a set of points P and a query region q, the goal is to find the convex hull of the points in $P \cap q$. We present a solution for generalized *reporting* of the convex hull in a orthogonal range q, for which we must preprocess the points in a way such that we can report the distinct colors that constitute the convex hull of the points in $P \cap q$.

2 Previous Work

Kalavagattu et al. [15] studied the problem of counting and reporting points belonging to the maxima in an orthogonal range query on a static set of points. Brodal et al. [5] presented a solution for the dynamic version of the same problem. Rahul et al. [21] solved a similar problem where the maxima and the range query are based on different sets of dimensions. The colored version of the range maxima problem has not been studied before.

Brass et al. [4] studied the range convex hull problem and presented a solution using range trees and a method similar to the gift wrapping algorithm [19]. Moidu et al. [18] presented a more efficient solution, using a novel approach using a modified version of the range tree. The colored version of the range convex hull problem has also not been studied before.

3 Preliminaries

We assume that all points are distinct and have integer co-ordinates. The more general setting can be transformed to one with integer co-ordinates by reduction to rank space using standard methods [6][2]. Let $\mathcal{C}$ be the set of all colors. Let $n = |P|$. Clearly, $|\mathcal{C}| \leq n$ (if $|\mathcal{C}| = n$, then the problem becomes a case of standard intersection searching, without colors). We encode the colors as integers from 1 to n for notational and operational convenience.

For both the problems under consideration, let c be the number of distinct colors intersected by the query range. Our answer, therefore, will have size $O(c)$. To be output sensitive, we would therefore like our solutions to take time which

is a function of c (and not a function of the total number of points on the maxima or convex hull). This rules out any method involving the computation of all maximal or convex hull points followed by some processing on them.

All queries studied in the following sections are orthogonal and axis-parallel, unless explicitly specified otherwise. All results are in the pointer machine model.

3.1 Generalized Reporting in One Dimension

Given a set of points in one dimension where each point has a color (not necessarily unique) associated with it, we want to preprocess the point-set such that given a query range we can efficiently report the distinct colors in that range.

Gupta et al. [11] showed a transformation which reduces the generalized one dimensional range reporting problem into the standard grounded range reporting problem in two dimensions and solved the problem using a *priority search tree* (PST) [17]. Thus, for a static set of points in one dimension, the c distinct colors intersected by a query range can be reported using a $O(n)$ space and $O(n \log n)$ preprocessing time data structure $\mathcal{D}$ which answers queries in $O(\log n + c)$ time per query. We use this result in our solutions for both maxima and convex hull.

3.2 Heavy-light Decomposition

Heavy-light decomposition is a technique which allows us to break down a rooted tree into a set of mutually disjoint paths. It was first used in literature by Tarjan [25] while the exact phrase was coined by Sleator and Tarjan when they used the technique in their analysis of link-cut trees [23][24].

Let T be a rooted n-ary tree. Let size(v) be the number of nodes in the subtree rooted at a node v. An edge (p, q), where q is a child of p, is labeled *heavy* if size(q) $> \frac{1}{2} \cdot$ size(p) and *light* otherwise. A tree with edges labeled in this manner is said to be *decomposed*. The following properties can be shown easily for a heavy-light decomposed tree:

- At most 1 edge from a node to its children can be heavy.
- Each connected set of heavy edges forms a vertex-disjoint path. We call such a path a *heavy path*.
- A path (v, u) in the tree, where u is an ancestor of v, will consist of $O(\log n)$ light edges and $O(\log n)$ heavy paths.

4 Generalized Range Maxima Queries

Due to the nature of information conveyed by the maximal chain of a set of points, it is common to perform orthogonal range queries that are unbounded in one or two directions. We present separate solutions to three representative types of range queries for the maxima problem.

4.1 Two Sided Queries

We begin with the simplest kind of maxima queries, where the query range is unbounded in two directions. We solve for query ranges of the type $[x_l, \infty) \times [y_l, \infty)$. Notice that in such queries, the range maxima is nothing but a continuous subset of the maximal chain of the entire point-set. Therefore for this type of two-sided query, reporting the generalized range maxima is equivalent to the generalized reporting problem in one dimension. This can be solved using the method described in Section 3.1. The remaining three types of two-sided queries, however, cannot be solved using this approach.

Theorem 1. *Let P be a set of colored points in two dimensions. P can be preprocessed into a $O(n)$ space and $O(n \log n)$ preprocessing time data structure such that given a two-sided query q unbounded in the positive x and positive y directions, the c distinct colors in the maxima of $P \cap q$ can be reported in $O(\log n + c)$ time.*

4.2 Three Sided Queries

In a three sided query, the maximal chain need not be a continuous subset of the maximal chain for the entire point set. We solve the problem for three sided queries unbounded in the positive x direction, i.e. queries of the type $[x_l, \infty) \times [y_l, y_h]$, where $y_l < y_h$.

We construct a one dimensional range tree on the x co-ordinates of the points in P. Let us call this tree T_x. Let $S(v)$ be the subtree of an internal node v in T_x. At each internal node v, we store a pointer to the point in $S(v)$ having the maximum y co-ordinate.

For a point $p = (p_x, p_y)$, let next (p) be the point with maximum y co-ordinate in its south-east quadrant, i.e. in the range $[p_x, \infty) \times (-\infty, p_y]$. We construct a graph where each point in P is a vertex and there is an edge from each p_i to next(p_i). We create a dummy vertex p_{null}. For all such points p_i for which no next(p_i) value exists in P, we add an edge (p_i, p_{null}) to the graph. The next(.) values can be computed in $O(n \log n)$ time using the tree T_x described above. Each vertex in this graph, except p_{null} has exactly one outgoing edge (edge pointing towards its next(.) point) and there are no cycles. Therefore, this graph is a tree. We call it $\mathcal{T}$. A maximal chain, as reported by a three sided query, will be a path in $\mathcal{T}$.

We now decompose $\mathcal{T}$ using heavy-light decomposition. We preprocess all heavy paths with the data structure $\mathcal{D}$ of Section 3.1. For light edges we do not perform any preprocessing.

Theorem 2. *Let P be a set of colored points in two dimensions. P can be processed into a $O(n)$ space and $O(n \log n)$ preprocessing time data structure such that given a three sided query q, unbounded on the right, the c distinct colors in the maxima of $P \cap q$ can be reported in $O(\log^2 n + c \log n)$ time per query.*

Proof. The preprocessing stage involves:

1. The building of the tree T_x, which is a one dimensional range tree. It takes $O(n)$ space and can be built in $O(n \log n)$ time.
2. The building of the tree $\mathcal{T}$, which takes $O(n \log n)$ time. The number of nodes in $\mathcal{T}$ is the same as the number of points in P, therefore $\mathcal{T}$ has size $O(n)$. In addition, for each heavy path in $\mathcal{T}$ we build the data structure $\mathcal{D}$ of section 3.1. If the length of a heavy path is l_h, it takes $O(l_h)$ space and $O(l_h \log l_h)$ time to build the required data structure. Clearly the overall space requirement is $O(n)$ while the time required is $O(n \log n)$.

Thus, for preprocessing, the total space requirement is $O(n)$ and the total time required is $O(n \log n)$.

Given a query $[x_l, \infty) \times [y_l, y_h]$, let $p_a \in P$ be the point with the maximum y co-ordinate lying in the range $[x_l, \infty) \times [-\infty, y_h]$. This point can be found in $O(\log n)$ time by an orthogonal range successor query on the tree T_x. The point p_a will be one end point of the maximal chain in $P \cap q$. Let p_b be the other end point and $\mathcal{P}_{mc}$ be the path in $\mathcal{T}$ from p_a to p_b. It can be shown that p_b will be an ancestor of p_a in the tree $\mathcal{T}$. Therefore, it follows from Section 3.2 that the path $\mathcal{P}_{mc}$ will consist of $O(\log n)$ light edges and $O(\log n)$ heavy paths.

For heavy paths, we can report the distinct colors using the preprocessed data structure $\mathcal{D}$ of Section 3.1. This takes $O(\log n + c)$ time per heavy path. Since there are $O(\log n)$ heavy paths, the total time required is $O(\log^2 n + c \log n)$ For the remaining points, i.e. those that are not part of a heavy path, we simply report the colors when such a point is encountered. This takes a total of $O(\log n)$ time since there are $O(\log n)$ light edges. Thus reporting the colors on the maximal chain for a three sided query can be done in $O(\log^2 n + c \log n)$. However, there still remains the issue of duplicates.

To ensure that each color in the maxima is reported only once, we leverage the fact that colors are encoded as integers from 1 to $\mathcal{C}$. For every query we initialize a bit-array B, of size $\mathcal{C}$, with all $B[i]$ set to 0. As colors are reported from $\mathcal{T}$, for every reported color k we check the value of $B[k]$. If $B[k] = 0$, we output color k and set $B[k]$ to 1, else we do not output color k and move on. Since there are a total of $O(c)$ colors output by $\mathcal{T}$, the aggregation process will not be a dominating factor in the query time. □

4.3 Four Sided Queries

Four sided (rectangular) orthogonal range queries can be answered using a range tree together with the structure for three sided queries (Section 4.2).

Theorem 3. *Let P be a set of colored points in two dimensions. P can be processed into a $O(n \log n)$ space and $O(n \log^2 n)$ preprocessing time data structure such that given a four sided query q, the c distinct colors in the maxima of $P \cap q$ can be reported in $O(\log^3 n + c \log^2 n)$ time per query.*

Proof. Let T_x be a one dimensional range tree on the x co-ordinates of all points in P. Let $S(v)$ be the subtree of an internal node v in T_x. At each internal node v, we populate the structures described in Section 4.2 for three sided

queries, using the points in $S(v)$ as the input. This takes $O(|S(v)|)$ space and $O(|S(v)| \log |S(v)|)$ time per internal node v.

Now consider the set S_d of all nodes lying at depth d in the primary tree T_x. The total time required to preprocess all nodes in S_d will be:

$$\sum_{v \in S_d} O(|S(v)| \log |S(v)|) \leq \sum_{v \in S_d} O(|S(v)| \log n) = O(\log n \sum_{v \in S_d} |S(v)|) = O(n \log n) \tag{1}$$

Likewise, the total space required to store the preprocessed data structures on each node in S_d will be:

$$\sum_{v \in S_d} O(|S(v)|) = O(\sum_{v \in S_d} |S(v)|) = O(n) \tag{2}$$

There will be $O(\log n)$ levels in the tree T_x. From equations (1) and (2) it follows that preprocessing the entire tree will take $O(n \log^2 n)$ time and $O(n \log n)$ space.

Given a query $q = [x_l, x_h] \times [y_l, y_h]$, we first query T_x with the range $[x_l, x_h]$. This gives us $O(\log n)$ canonical nodes. We process these canonical nodes from right to left, starting with the canonical node whose x range has upper boundary x_h. On the first canonical node, we do a three sided query $[x_l, \infty] \times [y_l, y_h]$. Let y_m be the y co-ordinate of the point with maximum y on the maximal chain of the points covered by this canonical node. Points whose y co-ordinate is less than y_m in the remaining canonical nodes cannot lie on the maximal chain of the query region. So on the next canonical node, we do a three sided query $[x_m, \infty] \times [y_m, y_h]$, and so on. Each of these queries will return at most c colors. The time taken per canonical node will be $O(\log^2 n + c \log n)$. Since there are $O(\log n)$ canonical nodes, the total time per query will be $O(\log^3 n + c \log^2 n)$. Note that even though the colors output by each canonical node will be distinct within themselves, the overall set may have duplicates. To remove them, we will once again use the bit array method described in the proof of Theorem 2. □

5 Generalized Range Convex Hull Queries

We present an output sensitive algorithm to report the distinct colors on the convex hull of the points lying in a query range. We use the modified two dimensional range tree proposed by Moidu et al. [18] together with the generalized one dimensional range reporting structure of Section 3.1.

5.1 Preprocessing

We build a modified range tree $\mathcal{R}$, as described in [18], on the set of points P and supplement it with additional preprocessed data structures to support generalized range reporting of the convex hull.

Constructing $\mathcal{R}$ takes $O(n \log n)$ time. Within $\mathcal{R}$, we call the primary range tree, built using the x co-ordinates of the points, T_x. Each vertex v_i of T_x has a secondary tree associated with it, which is built using the y co-ordinates of the

points rooted at v_i. We call this tree $T_y(v_i)$. At each canonical node in every secondary tree $T_y(v_i)$, we precompute the convex hull of the points rooted at $T_y(v_i)$.

Lemma 1. *The convex hulls of the points rooted at the canonical nodes in $\mathcal{R}$ can be computed in a total of $O(n \log^2 n)$ time.*

Proof. Consider a node v_x in the primary tree T_x. Let the number of points rooted at v_x be n_{v_x}. Therefore the total number of points in the tree $T_y(v_x)$ will also be n_{v_x}.

Instead of computing the convex hull of the points rooted at each node of $T_y(v_x)$ ab initio, we will proceed in a bottom-up fashion starting at the deepest (lowermost) level, merging the convex hulls of the children of each canonical node to get the convex hull of the points at the canonical node itself. Merging of every pair of child hulls to form the parent hull takes $O(\log(n_1 + n_2))$ time using the method described by Kirkpatrick and Snoeyink [16] to compute outer tangents for disjoint convex polyhedra. Here n_1 and n_2 are the number of points in the respective child convex hulls. Clearly, even in the worst case, $n_1 + n_2 \leq n_{v_x}$ (for tree $T_y(v_x)$). Therefore each merge step takes at most $O(\log n_{v_x})$ time. Notice that the total number of merge operations required to compute the convex hulls for all canonical nodes in $T_y(v_x)$ is the same as the total number of canonical nodes (non-leaf nodes) present in $T_y(v_x)$, i.e. $O(n_{v_x})$. Therefore, the total preprocessing time required to compute the convex hull of each canonical node in a secondary tree $T_y(v_x)$ will be $O(n_{v_x} \log n_{v_x})$, where $n_{v_x} = |T_y(v_x)|$.

Consider the set S_d of all nodes lying at depth d in the primary tree T_x. The time required to preprocess the secondary trees corresponding to each node $s \in S_d$ will be $O(n_s \log n_s)$, where n_s is the number of nodes rooted at s. The total time required to preprocess all nodes in S_d will be

$$\sum_{s \in S_d} O(n_s \log n_s) \leq \sum_{s \in S_d} O(n_s \log n) = O(\log n \sum_{s \in S_d} n_s) = O(n \log n) \qquad (3)$$

There are a total of $\log n$ levels in the tree T_x. Therefore, preprocessing the entire tree at the cost of $O(n \log n)$ per level will take $O(n \log^2 n)$ time. □

Once the convex hulls of all canonical nodes have been computed, we preprocess each canonical convex hull (convex hull of the points rooted at the canonical node) for generalized range reporting. To do this, we first linearize the list of convex hull points and then preprocess it using the data structure $\mathcal{D}$ described in Section 3.1. At each canonical node, we store the convex hull points in counterclockwise order in an array. We store a pointer from each convex hull point to its index in the array, allowing lookups in both directions (array index to convex hull point and vice-versa). Using the array indices as one dimensional co-ordinates, we preprocess the array for generalized one dimensional range reporting using the data structure $\mathcal{D}$.

Lemma 2. *The canonical convex hulls in $\mathcal{R}$ can be preprocessed for generalized range reporting in $O(n \log^3 n)$ time.*

Proof. The data structure $\mathcal{D}$ takes $O(n \log n)$ time to build. We build one instance of $\mathcal{D}$ for every canonical node in $\mathcal{R}$. By a similar analysis as was performed in proving Lemma 1, it can be shown that the preprocessing of all canonical convex hulls in $\mathcal{R}$ will take $O(n \log^3 n)$ time. □

Lemma 3. *The data structure $\mathcal{R}$ occupies $O(n \log^2 n)$ space.*

Proof. The modified range tree of [18] utilizes $O(n \log n)$ space, as it only stores a constant amount of extra information per node. Our data structure $\mathcal{R}$, however, stores two additional items at each canonical node – the canonical convex hull and the data structure for generalized reporting. Both these structures require $O(n)$ space. Analysis similar to what is shown in the proof of Lemma 1 will show that the total space requirement of the data structure $\mathcal{R}$ is $O(n \log^2 n)$. □

5.2 Query

Moidu et al., in [18], show that using their modified range tree, given a query region q, in $O(\log^2 n)$ time we can

1. Identify $O(\log n)$ canonical nodes whose convex hulls, when merged, form the convex hull of the points in $P \cap q$. They call these nodes *candidate* nodes (or blocks).
2. Find, for each candidate node, the start and end points of a continuous segment of its convex hull which, after merging, becomes part of the convex hull of $P \cap q$. Let us call these points p_s and p_e.

For every candidate node n_c, once we obtain the points p_s and p_e, we can find the corresponding indices l and r, using the arrays populated in the preprocessing stage for generalized reporting (refer Section 5.1). We then do the query $[l, r]$ on the preprocessed generalized reporting data structure $\mathcal{D}$. For each of the $O(\log n)$ candidate nodes, the corresponding instance of $\mathcal{D}$ outputs the distinct colors present in the points which that node contributes to the convex hull of $P \cap q$. The set of colors returned by querying each candidate node can at most be of size c, where c is the total number of distinct colors in the convex hull of $P \cap q$. However, the same color can be output by more than one candidate node. To ensure that each color is reported only once, we once again use the bit-array method of Section 4.2 which ensures that there are no duplicates in the final output.

Lemma 4. *The data structure $\mathcal{R}$ answers range queries in $O(\log^2 n + c \log n)$ time.*

Proof. For every range query of the form $[x_l, x_h] \times [y_l, y_h]$, our query algorithm performs a one dimensional generalized range reporting query on the instance of the data structure $\mathcal{D}$ stored in each of the $O(\log n)$ candidate canonical nodes. $\mathcal{D}$ has a query time of $O(\log n + c)$, where c is the number of colors reported. Hence our method requires $O(\log^2 n + c \log n)$ time per query. □

Thus we have the following result:

Theorem 4. *Let P be a set of colored points in two dimensions. P can be preprocessed into a $O(n \log^2 n)$ space and $O(n \log^3 n)$ preprocessing time data structure such that given an orthogonal range query q, the c distinct colors in the convex hull of $P \cap q$ can be reported in $O(\log^2 n + c \log n)$ time.*

6 Future Work

Designing output sensitive algorithms for generalized intersection searching problems is an challenging problem since the query time must depend on the number of colors and not on the number of points in the result. Both problems discussed in the preceding sections present interesting possibilities in the design of solutions for generalized geometric range aggregate query problems. An immediate open question is that of proposing more efficient solutions for the problems introduced here. Improvements in the runtime by a logarithmic factor should be possible.

Furthermore, we have not tackled the counting versions of the generalized range maxima and convex hull where the result is simply the number of colors. Developing efficient algorithms for the counting versions remains an open line of pursuit. All results here are for a static set of points. Developing solutions for the dynamic case is still an open problem. Solutions to these problems in higher dimensions are also yet to be proposed.

References

1. Agarwal, P.K., Govindarajan, S., Muthukrishnan, S.M.: Range searching in categorical data: Colored range searching on grid. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 17–28. Springer, Heidelberg (2002)
2. Alstrup, S., Brodal, G.S., Rauhe, T.: New data structures for orthogonal range searching. In: FOCS, pp. 198–207. IEEE Computer Society (2000)
3. Bozanis, P., Kitsios, N., Makris, C., Tsakalidis, A.K.: New upper bounds for generalized intersection searching problems. In: Fülöp, Z. (ed.) ICALP 1995. LNCS, vol. 944, pp. 464–474. Springer, Heidelberg (1995)
4. Brass, P., Knauer, C., Shin, C.S., Smid, M., Vigan, I.: Range-aggregate queries for geometric extent problems. In: CATS: 19th Computing: Australasian Theory Symposium (2013)
5. Brodal, G.S., Tsakalidis, K.: Dynamic planar range maxima queries. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 256–267. Springer, Heidelberg (2011)
6. Chazelle, B.: A functional approach to data structures and its use in multidimensional searching. SIAM J. Comput. 17(3), 427–462 (1988)
7. Gagie, T., Kärkkäinen, J., Navarro, G., Puglisi, S.J.: Colored range queries and document retrieval. Theor. Comput. Sci. 483, 36–50 (2013)
8. Gupta, P., Janardan, R., Smid, M.: Computational geometry: Generalized intersection searching. In: Mehta, D., Sahni, S. (eds.) Handbook of Data Structures and Applications, Chapman & Hall/CRC, Boca Raton, FL, pp. 1–17. CRC Press (2005)
9. Gupta, P., Janardan, R., Smid, M.H.M.: Efficient algorithms for generalized intersection searching on non-iso-oriented objects. In: Symposium on Computational Geometry, pp. 369–378 (1994)

10. Gupta, P., Janardan, R., Smid, M.H.M.: Algorithms for generalized halfspace range searching and other intersection searching problems. Comput. Geom. 5, 321–340 (1995)
11. Gupta, P., Janardan, R., Smid, M.H.M.: Further results on generalized intersection searching problems: Counting, reporting, and dynamization. J. Algorithms 19(2), 282–317 (1995)
12. Gupta, P., Janardan, R., Smid, M.H.M.: Algorithms for generalized halfspace range searching and other intersection searching problems. Comput. Geom. 6, 1–19 (1996)
13. Gupta, P., Janardan, R., Smid, M.H.M.: A technique for adding range restrictions to generalized searching problems. Inf. Process. Lett. 64(5), 263–269 (1997)
14. Janardan, R., Lopez, M.A.: Generalized intersection searching problems. Int. J. Comput. Geometry Appl. 3(1), 39–69 (1993)
15. Kalavagattu, A.K., Agarwal, J., Das, A.S., Kothapalli, K.: On counting range maxima points in plane. In: Smyth, B. (ed.) IWOCA 2012. LNCS, vol. 7643, pp. 263–273. Springer, Heidelberg (2012)
16. Kirkpatrick, D., Snoeyink, J.: Computing common tangents without a separating line. In: Sack, J.-R., Akl, S.G., Dehne, F., Santoro, N. (eds.) WADS 1995. LNCS, vol. 955, pp. 183–193. Springer, Heidelberg (1995)
17. McCreight, E.M.: Priority search trees. SIAM J. Comput. 14(2), 257–276 (1985)
18. Moidu, N., Agarwal, J., Kothapalli, K.: Planar convex hull range query and related problems. In: CCCG, Carleton University, Ottawa, Canada (2013)
19. O'Rourke, J.: Computational Geometry in C. Cambridge University Press (1998), http://cs.smith.edu/~orourke/books/compgeom.html
20. Rahul, S., Bellam, H., Gupta, P., Rajan, K.: Range aggregate structures for colored geometric objects. In: CCCG. pp. 249–252 (2010)
21. Rahul, S., Janardan, R.: Algorithms for range-skyline queries. In: Cruz, I.F., Knoblock, C.A., Kröger, P., Tanin, E., Widmayer, P. (eds.) SIGSPATIAL/GIS, pp. 526–529. ACM (2012)
22. Shi, Q., JáJá, J.: Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. Inf. Process. Lett. 95(3), 382–388 (2005)
23. Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. In: STOC, pp. 114–122. ACM (1981)
24. Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. J. Comput. Syst. Sci. 26(3), 362–391 (1983)
25. Tarjan, R.E.: Applications of path compression on balanced trees. J. ACM 26(4), 690–715 (1979)