New Results on Three-Sided Skyline Range Counting and Reporting

Suruchi Kushwaha
Michigan Technological University, Houghton, MI, USA

Yakov Nekrich
Michigan Technological University, Houghton, MI, USA

Abstract

In the orthogonal skyline range counting (resp. reporting) problem we store the set of points P in a data structure so that for any query range Q the number of points (resp. the list of all points) on the skyline of $Q \cap P$ can be found efficiently. In this paper we study two-dimensional range counting and reporting problems in the case when the query range is bounded on three sides.

We describe a linear-space data structure that answers *top-open* three-sided skyline counting queries in $O(\log \log N)$ time, where N is the number of points stored in the data structure. We also show that *bottom-open* three-sided skyline counting queries are as difficult as general four-sided queries and any data structure that uses $O(N \log^c N)$ space for a constant c requires $\Omega(\log N / \log \log N)$ time to answer such queries.

Next, we turn to skyline *color* range queries. In this variant of the problem each point in P is assigned a color and we must count (resp. report) the distinct colors of point on the skyline of $Q \cap P$. We describe an $O(N)$-space data structure that answers top-open three-sided color counting queries in $O(\log N / \log \log N)$ time. Finally, we study top-open three-sided skyline color reporting in the EM model and describe a data structure that uses linear space and answers queries in $O(\frac{k}{B} + 1)$ I/Os where k is the number of colors on the skyline. This is the first external-memory data structure with optimal query cost and space usage for this problem.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

Keywords and phrases Data Structures, Range Searching, Skyline Queries

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.27

1 Introduction

A two-dimensional point $p = (p.x, p.y)$ is said to be dominated by a point $q = (q.x, q.y)$ if $q.x > p.x$ and $q.y \geq p.y$, or $q.x \geq p.x$ and $q.y > p.y$. For a set of points P, the skyline of P is the subset of points that are not dominated by any other point in P. A skyline range searching query Q on a set of points P asks for (some information) about the skyline of points in $P \cap Q$.

In this paper we study different variants of two-dimensional skyline range searching in the case when the query range is bounded on three sides. See Figure 1. We present new results for three-sided skyline range counting queries and three-sided skyline color range counting queries. We also describe a data structure for three-sided skyline color range reporting in the external memory model.

Previous Work

A skyline range counting query Q asks for the number of points on the skyline of $Q \cap P$. Orthogonal skyline range counting queries were considered in [10, 6, 3]. Brodal and Larsen [3] described a $O(N)$-space data structure with $O(\log N / \log \log N)$ query time for a four-sided range. In [3] the authors also showed that this bound is tight: any data structure that uses $N \log^{O(1)} N$ space supports skyline range counting queries in $\Omega(\log N / \log \log N)$ time.

A skyline range reporting query Q asks for the list of all points on the skyline of $P \cap Q$. Data structures for orthogonal skyline range reporting queries were studied in [5, 3, 11, 9]. Das et al. [5] presented a data structure that uses $O(N \log N / \log\log N)$ space and supports four-sided skyline range reporting queries in $O(\log N / \log\log N + k)$ time, where k is the number of reported skyline points. Brodal and Larsen [3] imporved their result and described a data structure that requires query time $O(\log N / \log\log N + k)$ and space usage $O(N \log^{\varepsilon} N)$.

In the external memory model [17], Kejlberg-Rasmussen et al. [11] present a linear space data structure that answers four-sided skyline reporting queries in $O((N/B)^{\varepsilon} + k/B)$ I/Os, where k is the number of reported points and B is the block size. In the same paper [11] the authors present a linear-space data structure that answers top-open three-sided skyline reporting queries in $O(1 + k/B)$ I/Os when all points are on $N \times N$ grid.

Ganguly et al. [9] studied the colored version of skyline range reporting problem in the external-memory model. In this variant of the problem each point is assigned a color (or category). The answer to a query Q is the list of colors that occur on the skyline of $P \cap Q$. For the case when all points are on the $N \times N$ grid, Ganguly et al. [9] describe a data structure that uses $O(N \log^* N)$ space and answers top-open three-sided queries in $O(1 + k/B)$ I/Os and a data structure that uses $O(N)$ space and answers queries in $O((\log^* N)^2 + k/B)$ I/Os, where k is the number of reported colors.

Our Results

In this paper we describe a linear-space data structure that answers *top-open* three-sided skyline range counting queries on $N \times N$ grid in time $O(\log\log N)$. We also show that any data structure with $O(N \log^c N)$ space for some constant $c > 0$ needs $\Omega(\log N / \log\log N)$ time to answer *bottom-open* three-sided skyline range counting queries. This result demonstrates that there is a fundamental gap between top-open queries (i.e., when the query range is unbounded in $+y$ direction) and bottom-open queries (i.e., when the query range is unbounded in $-y$ direction) with respect to skyline counting queries. Moreover, due to the lower bound on orthogonal range counting [16], we need $\Omega(\log N / \log\log N)$ time to answer two-dimensional range counting queries using $O(N\text{polylog(N)})$ space. This lower bound is also valid when the query range is a top-open three-sided range. Our result thus shows that counting points on a skyline of a query range can be fundamentally faster than counting all points in a query range.

Next we turn to the skyline color range counting problem. We describe a data structure that answers top-open three-sided skyline color counting queries in $O(\log N / \log\log N)$ time. That is, our data structure calculates, for any top-open three-sided query range Q, the number of distinct colors of points on the skyline of Q. This is the first data structure for the skyline color counting problem.

Finally we consider top-open three-sided color range reporting in the external-memory model. We improve upon the previous result by Ganguly et al. [9] and describe a data structure that uses $O(N)$ space and supports queries in $O(\frac{k}{B} + 1)$ I/Os, where k is the number of reported colors and B is the block size. Our result demonstrates that three-sided skyline color reporting can be implemented with optimal space and query cost. For comparison, the best currently known solution for the three-sided color reporting problem (i.e., for the case when *all* colors in a three-sided query range must be reported) either uses super-linear space or has super-constant query cost [15]. Thus our result shows that skyline reporting can be more efficient than general color reporting in the case of top-open three-sided ranges (with respect to the best currently known results).

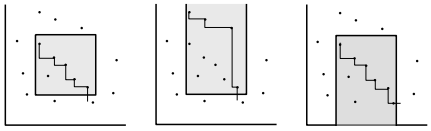
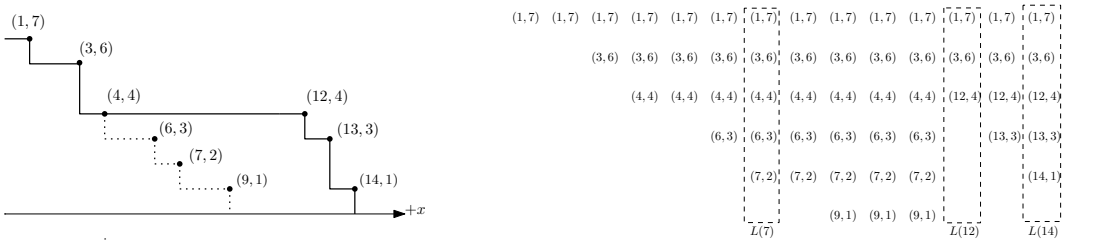


Figure 1 Examples of four-sided (left) top-open three-sided (center) and bottom-open three-sided (right) skyline queries.



(a) Example skyline points.

(b) Example skyline points in partially persistent list.

Figure 2 Skyline points in our data structure.

For simplicity we will assume throughout this paper that point coordinates are in the rank space [8, 1], i.e., all point coordinates are distinct integers from $\{1, \ldots, N\}$. We reserve N to denote the total number of points in a data structure and B will denote the block size in the external memory model.

2 Top-Open Skyline Counting

In this section we describe a simple linear-space data structure that answers three-sided skyline counting queries in $O(\log \log N)$ time. Our solution uses the sweepline technique: a vertical sweepline l is moved in $+x$ direction and a subset of points to the left of l is stored in a partially persistent list L. When a point p is hit by the sweepline, the list L is updated as follows: we remove all points $p' \in L$ with $p'.y \leq p.y$; then we add p to L. Points in L are stored in decreasing order of their y-coordinates. We will denote by $L(a)$ the version of L after removing all p' and inserting the point p with $p.x = a$. See Figure 2 for an example.

▶ **Lemma 1.** *When the vertical sweepline hits a point p, p is appended at the end of L.*

Proof. Follows directly from the description. When a point p is inserted into L, all points p' with $p'.y < p.y$ must be removed from L. ◀

The *rank* of a point p in the list L, denoted by $\text{rank}(p)$, is equal to the number of points that precede p in L (plus one). By Lemma 1, when a point p is inserted into $L(p.x)$, the rank of p equals the size of $L(p.x)$; the rank of any point p is not changed while p is in L.

▶ **Lemma 2.** *A point p is in $L(a)$ iff p is on the the skyline of $Q_a = (-\infty, a] \times (-\infty, +\infty)$.*

Proof. If p is not on the skyline of Q_a, then p is dominated by some point $p' \in Q_a$. Since $p.x < p'.x \leq a$ and $p.y \leq p'.y$, p was removed from $L(a')$ for some $a' \leq p'.x \leq a$ and $p \notin L(a)$. If p is on the skyline of Q_a, then there is no point p'' such that $p.x \leq p''.x \leq a$ and $p.y \leq p''.y$. By definition of L, $p \in L(a)$. ◀

If a point p precedes a point q in L, we will denote this by $p \prec q$.

▶ **Lemma 3.** *If $p \prec q$ in L, then $p.x < q.x$ and $p.y > q.y$.*

Proof. Consider the version $L(q.x)$ of L.
By Lemma 1, q is appended at the end of $L(q.x)$ and p precedes q in $L(q.x)$; since L is sorted by y-coordinates, $p.y > q.y$. By Lemma 2, p and q are on the skyline of $(-\infty, a] \times (-\infty, +\infty)$. Since $p.y \geq q.y$ and p does not dominate q, $p.x < q.x$. ◀

▶ **Lemma 4.** *A point p is on the skyline of $Q = [a, b] \times [c, +\infty)$ iff p is on the skyline of $(-\infty, b] \times (-\infty, +\infty)$, $p.x \geq a$ and $p.y \geq c$.*

Proof. Consider a point p on the skyline of $[a, b] \times [c, +\infty)$. Then $p.x \geq a$ and $p.y \geq c$. There is no point $p' \in [a, b] \times [c, +\infty)$ that dominates p. Any point p' that dominates p must satisfy $p'.x \geq a$ and $p'.y \geq c$. Hence, there is no point in $(-\infty, b] \times (-\infty, +\infty)$ that dominates p. Conversely, consider a point p on the skyline of $(-\infty, b] \times (-\infty, +\infty)$ such that $p.x \geq a$ and $p.y \geq c$. There is no point p' that dominates p in the region $(-\infty, b] \times (-\infty, +\infty)$. Hence there is no point p' that dominates p in $[a, b] \times [c, +\infty)$. Since p is in Q and p is not dominated by any other $p' \in Q$, p is on the skyline of Q. ◀

▶ **Theorem 5.** *There exists an $O(N)$-space data structure that answers top-open three-sided skyline counting queries on $N \times N$ grid in time $O(\log \log N)$.*

Proof. We use the partially persistent predecessor data structure by Chan [4] that supports queries in $O(\log \log N)$ time and uses $O(N)$ space where N is the total number of elements stored in the data structure. We employ two instances of Chan's data structure, one that supports queries on x-coordinates of points in L and one that supports queries on y-coordinates of points in L. Suppose that we must answer a skyline counting query on a range $Q = [a, b] \times [c, +\infty)$. By Lemmas 2 and 4, a point p is on the skyline of $[a, b] \times [c, +\infty)$ iff $p \in L(b)$, $p.x \geq a$ and $p.y \leq b$. Points in $L(b)$ are sorted in increasing order by their x-coordinates and in decreasing order by their y-coordinates. Let p_c denote the point in $L(b)$ with the smallest y-coordinate such that $p_c.y \geq c$. Let p_a denote the point in $L(b)$ with the largest x-coordinate such that $p_c.x \geq a$. By Lemma 3, the skyline of Q consists of all points p in $L(b)$ such that $p_a \preceq p \preceq p_c$. The number of such points is $\text{rank}(p_c) - \text{rank}(p_a) + 1$, where $\text{rank}(q)$ denotes the rank of a point q in L. ◀

Using the reduction to rank space technique, we can extend the result of Theorem 5 to the general case of N points on a $U \times U$ grid; see e.g., [1]. In this general case, the space usage of the data structure remains $O(N)$ and the query time is increased by an additive term $q_{pred}(N) = O(\min(\sqrt{\log N / \log \log N}, \log \log U))$, where $q_{pred}(N)$ is the time needed to answer a predecessor query [13].

3 Adjusted Right Boundary

For a given set of points P and query range Q the range predecessor of $P \cap Q$ is the point p_s with the largest x-coordinate in $P \cap Q$. The *adjusted right boundary* of $P \cap Q$, denoted by $\texttt{adj}(P, Q)$, is the x-coordinate of p_s. Thus $\texttt{adj}(P, Q)$ is the x-coordinate of the rightmost point in $P \cap Q$. For simplicity we will denote $\texttt{adj}(P, Q)$ as $\texttt{adj}(Q)$, since P is the entire input set. Using the adjusted right boundary, we can convert a skyline query on a top-open three-sided range $Q = [a, b] \times [c, +\infty)$ into a skyline query on an one-dimensional range $Q' = [a, b'] \times (-\infty, +\infty)$ where $b' = \texttt{adj}(Q)$. We will use the partially persistent list L, introduced in Section 2 to describe our result in this section.

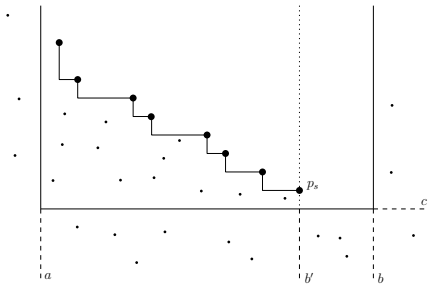


Figure 3 An example of adjusted right boundary; $b' = \texttt{adj}(Q)$ for $Q = [a, b] \times [c, +\infty)$.

▶ **Lemma 6.** *A point p is on the skyline of $Q = [a, b] \times [c, +\infty)$ iff p is on the skyline of $Q' = [a, b'] \times (-\infty, +\infty)$ where $b' = \texttt{adj}(Q)$.*

Proof. Suppose that a point p is on the skyline of Q, i.e there is no other point $q \in Q$ that dominates p. By definition of $\texttt{adj}(Q)$, b' is the largest x-coordinate of a point in Q. Hence, $p.x \leq b'$ and therefore p must be in Q'. There is no point in $[a, b'] \times [c, +\infty)$ that dominates p because $[a, b'] \times [c, +\infty) \subset [a, b] \times (-\infty, +\infty)$ and there is no point with y-coordinate less than c that dominates p because $p.y \geq c$. Therefore, p is on the skyline of Q'. Now, suppose the point p is on the skyline of Q'. Since $p.x \leq b'$ and $b' \leq b$, $p.x \leq b$. Let p_r be the point such that $b' = p_r.x$. Then p_r is the point with the largest x-coordinate in Q and $p_r.y \geq c$. Since $p.x \leq p_r.x$ and p is on the skyline of Q', $p.y \geq p_r.y \geq c$. There is also no point q with $q.x \in [b', b]$ and $q.y \geq c$. Therefore $p \in Q$ and is also on the skyline of Q. ◀

The following lemma enables us to compute the range predecessor query in a top-open three-sided range.

▶ **Lemma 7.** *Let $Q = [a, b] \times [c, +\infty)$ and let p_c denote the point that is the successor of c in $L(b)$ with respect to y-coordinates. If Q is not empty, then p_c is the range predecessor of Q.*

Proof. According to Lemma 2, $L(b)$ is the skyline of points in $(-\infty, b] \times (-\infty, +\infty)$, hence $p_c.x \leq b$. Since, p_c is the successor of c in $L(b)$ with respect to y-coordinates, $p_c.y \geq c$ and by Lemma 3, $p_c.x$ must be the point with largest x-coordinate in $L(b)$ among the points that have y-coordinate greater than c. Therefore, p_c is the point with largest x-coordinate in the range $(-\infty, b] \times [c, +\infty)$. Since Q is not empty, $p_c.x \geq a$. Hence, $p_c \in Q$ and is also the range predecessor of Q. ◀

To find the adjusted right boundary of a top-open three-sided query range $Q = [a, b] \times [c, +\infty)$, we visit the version $L(b)$ of the partially persistent list, i.e. the version that is created when the vertical sweepline hits $x = b$. By Lemma 7, we can find the adjusted right boundary in $O(\log \log N)$ time using the persistent predecessor data structure from [4].

4 Bottom-open Three-sided Queries

In this section we show that a four-sided skyline range searching query $Q = [a, b] \times [c, d]$ can be reduced to a bottom-open three-sided skyline range searching query $Q' = [a, b'] \times (-\infty, d]$ where b' is the x-coordinate of the rightmost point in Q (aka the range predecessor of Q). In other words the skyline of Q is the same as the skyline of the bottom-open range $Q' = [a, b'] \times (-\infty, d]$, where $b' = p_s.x$.

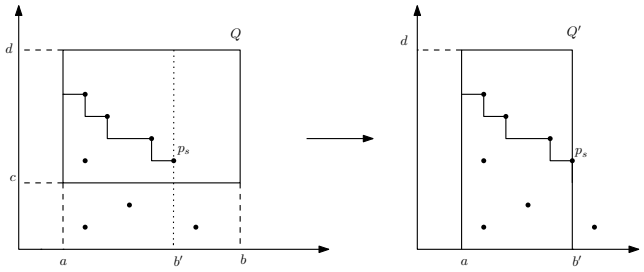


Figure 4 Reduction of a four-sided skyline query to a bottom-open three-sided skyline query.

▶ **Lemma 8.** *Let p_s denote the rightmost point in a four-sided range $Q = [a, b] \times [c, d]$. A point p is on the skyline of Q iff p is on the skyline of $Q' = [a, b'] \times (-\infty, d]$ where $b' = p_s.x$.*

Proof. Consider some point p on the skyline of Q. Since p_s is the rightmost point in Q, $p.x \leq p_s.x$ and hence p is in Q'. Since p is on the skyline of Q and $b' \leq b$ there is no point in $[a, b'] \times [c, d]$ that dominates p. No point in $[a, b'] \times (-\infty, c]$ can dominate p because $p.y \geq c$. Hence, no point in Q' dominates p and p is on the skyline of Q'. Now, suppose a point p is on the skyline of Q'. Since $b' = p_s.x$, $p.x \leq p_s.x$. The point p is not dominated by p_s because p is on the skyline of Q' and hence $p.y \geq p_s.y \geq c$ since p_s is the rightmost point of Q. This implies that $p \in Q$ and, since $[b', b] \times [c, d]$ is empty, p is also on the skyline of Q. ◀

We can use any data structure for bottom-open three-sided skyline queries to answer four-sided skyline queries, with an additional cost of finding the rightmost point in the four-sided range. Through this reduction, we can demonstrate that solving a bottom-open three-sided skyline query is as hard as solving a four-sided skyline query.

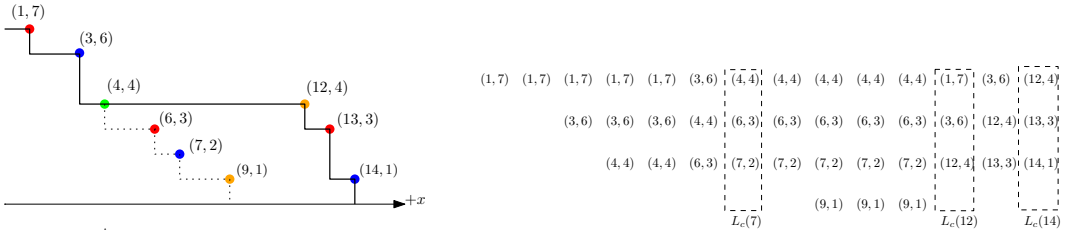
▶ **Theorem 9.** *If we can answer a bottom-open three-sided skyline counting query in time $O(Q(N))$ and $O(S(N))$ space, then we can answer a four-sided skyline counting query in time $O(Q(N) + Q'(N))$ and $O(S(N) + S'(N))$ space where $Q'(N)$ and $S'(N)$ are the time and space required for answering a range predecessor query.*

Using the result of Nekrich and Navarro [14], the range predecessor query can be answered in $O(N)$ space and $O(\log^{\varepsilon} N)$ time. Hence we can set $S'(N) = O(N)$ and $Q'(N) = O(\log^{\varepsilon} N)$ in Theorem 9. As a direct corollary of this result, we establish the lower bound of $\Omega(\log N / \log\log N)$ for bottom-open three-sided skyline counting queries. This lower bound is also valid when all points are on $N \times N$ grid.

▶ **Corollary 10.** *Any data structure that uses $O(N\text{polylog}(N))$ space needs $\Omega(\log N / \log\log N)$ time to answer a bottom-open three-sided skyline counting query.*

Proof. In [3], Brodal and Larsen prove the $\Omega(\log N / \log\log N)$ lower bound for four-sided skyline counting queries. By Theorem 9 and [14], a data structure that supports three-sided bottom-open queries in $O(S(N))$ space and $O(Q(N))$ time can be used to answer four-sided queries in $O(S(N)+N)$ space and $O(Q(N)+\log^{\varepsilon} N)$ time. Hence, the lower bound of Brodal and Larsen [3] is also valid for bottom-open three-sided queries. ◀

We remark that Corollary 10 is valid even if points are in the rank space. On the other hand, we have shown in Theorem 5 that top-open three-sided skyline counting queries can be answered in $O(\log\log N)$ time using linear space. Theorem 5 and Corollary 10 show that there is a significant complexity gap between the top-open and bottom-open variants of three-sided skyline counting.



(a) Example color skyline points. **(b)** Example color skyline points in partially persistent list.

Figure 5 Color skyline points in our data structure.

5 Top-Open Skyline Color Counting

In this section, we present a linear space data structure for skyline color counting queries on a top-open three-sided range in $O(\log N/\log\log N)$ time. We use the adjusted right boundary, defined in Section 3, to reduce a top-open three-sided query to a two-sided query; then we count the distinct colors on the skyline of a two-sided range. The data structure will use the persistent list approach, similar to Section 2. But now our task is more challenging because distinct colors of points on the skyline must be counted and there could be several points of the same color. Therefore we will employ a different persistent list and combine this approach with the reduction of three-sided to two-sided queries from Section 3.

Let L denote the partially persistent list that was defined in Section 2 and let L_c be the sub-list of the partially persistent list L where for every version k and each color α that occurs in $L(k)$, $L_c(k)$ stores the bottommost point of color α from $L(k)$. Thus, as the vertical sweepline l moves along $+x$ direction, we keep only the bottommost point of each distinct color on the skyline to the left of l in the partially persistent list L_c. Our definition of the sublist is motivated by the fact that we must count the number of distinct colors on the skyline. Thus, L_c must contain exactly one point for every color on the skyline of the respective two-sided range. The correctness of our method will be proved in Lemma 11, 12, and 13.

We distinguish between three different kinds of updates on L_c. When l hits a point p with $p.x = a$, we *erase* all points $p' \in L_c$ such that $p.y \geq p'.y$. An erased point p' is permanently deleted from L_c as it will no longer be on the skyline; that is, p' will not be reinserted into any later versions of the list L_c. When a point p' is erased, we reinsert the bottommost previously removed point p'_r such that $p'_r.color = p'.color$ and $p'_r.y > p.y$ (provided such point p'_r exists). If there is a point $q \in L_c$ such that $q.color = p.color$ and $q.y > p.y$, we *remove* q. A removed point q is still on the skyline but is not the bottommost point of its color. Hence it must be deleted from the current version of L_c, but it can be reinserted into a later version of L_c. Then we add p to L_c. We will denote by $L_c(a)$ the version of L_c after erasing all p', removing q, and inserting p.

▶ **Lemma 11.** *There is at least one point of color α on the skyline of $Q = (-\infty, b] \times (-\infty, \infty)$ iff there is exactly one point of color α in $L_c(b)$.*

Proof. Suppose there is one or more points $p_1, p_2, p_3, ..., p_t$ where $t \geq 1$ of color α on the skyline of Q. Then by Lemma 2, $p_1, p_2, p_3, ..., p_t \in L(b)$ and by definition the bottommost point is in $L_c(b)$. Therefore $L_c(b)$ contains exactly one point of color α. Conversely, let p_α be the point of color α in $L_c(b)$. Then p_α is also in $L(b)$ because $L_c(b) \subset L(b)$. Hence, by Lemma 2, p_α is on the skyline of the query range Q and there is at least one point of color α in Q. ◀

▶ **Lemma 12.** *There is at least one point of color α on the skyline of $Q = [a, b] \times (-\infty, +\infty)$ iff there is exactly one point $p \in L_c(b)$ such that $p.x \geq a$.*

Proof. Suppose there are one or more points $p_1, p_2, p_3, ..., p_t$ where $t \geq 1$ of color α on the skyline of Q. Then by Lemma 2, $p_1, p_2, p_3, ..., p_t \in L(b)$. Let p_b denote the bottommost point of color α in $L(b)$. Since $p_b \in Q$, $p_b.x \geq a$. By definition, only $p_b \in L_c(b)$. Therefore, $L_c(b)$ contains exactly one point of color α such that $p.x \geq a$. Conversely, consider the point $p \in L_c(b)$ with $p.x \geq a$. Then by definition, $p \in L(b)$ and by Lemma 2, p is on the skyline of $(-\infty, b] \times (-\infty, +\infty)$. Finally, using a special case of Lemma 4, where $c = -\infty$, p is on the skyline of Q. ◀

▶ **Lemma 13.** *There is at least one point of color α on the skyline of $Q = [a, b] \times [c, +\infty)$ iff there is exactly one point p of color α in $L_c(b')$ such that $p.x \geq a$ and $b' = \texttt{adj}(Q)$.*

Proof. Suppose there is at least one point p of color α on the skyline of Q. By Lemma 6, p_b is on the skyline of $[a, b'] \times (-\infty, +\infty)$ where $b' = \texttt{adj}(Q)$. Hence by Lemma 13, there is exactly one point of color α in $L_c(b')$. Conversely, suppose the point p of color α is in $L_c(b')$. By Lemma 11, p is on the skyline of $(-\infty, b') \times (-\infty, +\infty)$. Given that $p.x \geq a$, hence by Lemma 6, p is on the skyline of Q. ◀

We can count the number of points $r \in L_c(b')$ with $r.x \geq a$ and $b' = \texttt{adj}(Q)$ by answering a partially persistent range counting query. We use the data structure of Munro et. al. [12] that answers such queries in $O(\log m / \log \log m)$ time and uses $O(m)$ space, where m is the total number of updates in L_c. We will show below that $m = O(N)$. Thus we can count the number of points $r \in L_c(b')$ with $r.x \geq a$ in $O(\log m / \log \log m) = O(\log N / \log \log N)$ time and the total query time is $O(\log N / \log \log N)$.

Space Usage

To demonstrate that the space requirement for the data structure is $O(N)$, we need to evaluate the number of updates for L_c. Suppose that every update costs \$1 and each point is initially assigned \$4 before being inserted into the partially persistent list. We will show below that this overall budget of $\$4N$ is sufficient to execute all necessary updates of L_c. The insert operation costs \$1, leaving the point with \$3. At any time, a point $p \in L_c$ can be either removed or erased from the list. When p is removed, \$1 is spent; thus each removed point has \$2. A removed point p is reinserted into L_c only when a point p_o with the same color and smaller y-coordinate is erased from L_c. The erased point, with \$3, spends \$1 on the erase operation; the remaining \$2 are transferred from p_o to the reinserted point p. Thus p spends \$1 on an insertion and receives \$2 from p_o. Hence when a point is re-inserted into p_b, it has a budget of \$3. In summary, the list L_c is updated $m \leq 4N$ times. Since the total number of updates is $m = O(N)$, the space used by L_c and the range counting data structure is also bounded by $O(N)$.

▶ **Theorem 14.** *There exists an $O(N)$-space data structure that answers top-open three-sided skyline color counting queries in $O(\log N / \log \log N)$ time.*

6 Top-Open Skyline Color Reporting

In this section we describe an external memory data structure that uses linear space and supports top-open three-sided skyline color reporting queries in $O(\frac{k}{B}) + 1$ I/Os, where k is the number of reported colors. Our data structure uses the persistent list L_c defined in

Section 5. By Lemma 12, a color occurs on the skyline of $Q = [a, b] \times [c, +\infty)$ iff there is a point p of that color in $L_c(b')$, such that $p.x \geq a$ and b' is the adjusted right boundary of Q. Thus, a query can be answered by accessing the version $L_c(b')$ of L_c. However, it is not clear whether we can compute the adjusted right boundary in $O(1)$ time. In this paper we use a different approach that relies on the results from previous sections. We will show that we can either (1) calculate b' in $O(\frac{k}{B})$ I/Os when the number of colors k is sufficiently large or (2) calculate the adjusted right boundary in $O(1)$ time for an appropriately selected small subset of points.

The following two lemmas will be used in this section.

▶ **Lemma 15.** *Let S be a set of integers such that $|S| = O(B \log^2 N)$. There exists a data structure that uses space $O(|S|)$ and supports predecessor queries on S in $O(1)$ I/Os.*

▶ **Lemma 16.** *Let P' be a set of two-dimensional points such that $|P'| = O\big(B \log^2 N\big)$. There exists a data structure that uses space $O(|P'|)$ and answers range predecessor queries on P' in $O(1)$ I/Os.*

Both lemmas can be easily proved by combining standard data structures; for completeness, we provide a proof in Appendix.

Data Structure

We divide the points into *chunks* of size $\Delta = B \log^2 N$ according to their x-coordinates. Thus, the i-th chunk contains points p, such that $i\Delta < p.x \leq (i+1)\Delta$ and the total number of chunks is $\frac{N}{\Delta}$. Each chunk is stored in the data structure of Lemma 16.

Chunks are also grouped into a number of (intersecting) sets, further called *canonical sets*. For every i such that $1 \leq i < \frac{N}{\Delta}$ and for every j such that $(i + 2^j)\Delta \leq N$, the canonical set $I_{i,j}$ consists of all points with x-coordinates in $[i\Delta + 1, (i + 2^j)\Delta]$. For each canonical set I, we store the list $TS(I)$ that contains $B \log N$ topmost points with distinct colors that occur on the skyline of I. More formally: Let $V(I)$ denote the (conceptual) list of all points on the skyline of I, sorted by y-coordinates in decreasing order. Let $V'(I)$ denote the sub-list of $V(I)$ that contains the topmost point of every color α that occurs in $V(I)$. The set $TS(I)$ contains the first $B \log N$ points from $V'(I)$. The y-coordinates of points in $TS(I)$ are stored[1] in the data structure of Lemma 15.

To find $\texttt{adj}(Q)$ for a top-open three-sided query range $Q = [a, b] \times [c, +\infty)$, we first identify the chunks containing a and b. Let the chunks containing a and b be C_a and C_b respectively. The query $Q = [a, b] \times [c, +\infty)$ is then divided into three parts: the left marginal query $Q_{lmarg} = [a, a_1 - 1] \times [c, +\infty)$, the right marginal query $Q_{rmarg} = [b_1 + 1, b] \times [c, +\infty)$ and the middle query $Q_{mid} = [a_1, b_1] \times [c, +\infty)$; the parameters a_1 and b_1 are chosen so that $a_1 - 1$ is the right x-boundary of the chunk C_a and $b_1 + 1$ is the left x-boundary of the chunk C_b. See Fig 6 for illustration. The marginal queries fit into individual chunks and the middle query spans an integer number of chunks. We will search for the appropriate adjusted right boundary in Q_{rmarg}, Q_{middle} and Q_{lmarg} (in this order).

The following lemma provides an explanation for our method. We will say that a query range Q is *empty* if $P \cap Q = \emptyset$.

[1] We remark that lists $V(I)$ and $V'(I)$ are not stored in our data structure. These lists were introduced in order to define $TS(I)$.

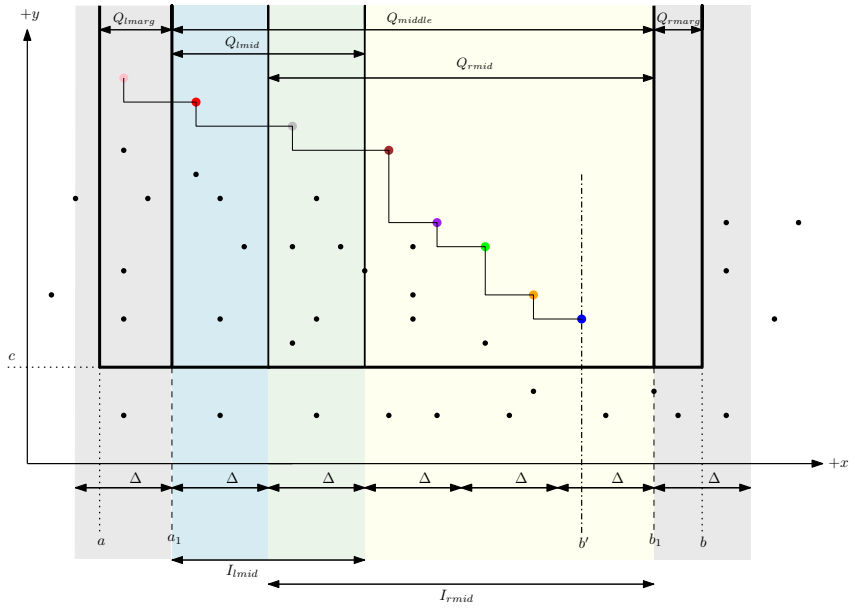


Figure 6 Procedure for finding $\texttt{adj}(Q)$. A query range Q is decomposed into Q_{lmarg}, Q_{middle}, and Q_{rmarg}. Q_{middle} is then decomposed into two overlapping query ranges, Q_{lmid} and Q_{rmid}.

▶ **Lemma 17.** *Let $Q = [a, b] \times [c, +\infty)$ and consider any r such that $a \leq r \leq b$. If $Q_r = [r, b] \times [c, +\infty)$ is empty, then the adjusted right boundary of $Q_l = [a, r-1] \times [c, +\infty)$ is the adjusted right boundary of Q.*
If $[r, b] \times [c, +\infty)$ is not empty, then the adjusted right boundary of $[r, b] \times [c, +\infty)$ is the adjusted right boundary of Q.

Proof. By definition, $\texttt{adj}(Q)$ is the largest x-coordinate of a point in Q, see Section 3. If the right region $[r, b] \times [c, +\infty)$ is empty the rightmost point must lie in the left region $Q_l = [a, r-1] \times [c, +\infty)$. Hence, $\texttt{adj}(Q) = \texttt{adj}([a, r-1] \times [c, +\infty))$. But if the right region $Q_r = [r, b] \times [c, +\infty)$ is not empty, there must be a point p in the right region such that $p.x > p'.x$ for any point $p' \in Q_l$; hence $\texttt{adj}(Q)$ is the same as $\texttt{adj}([r, b] \times [c, +\infty))$. ◀

Lemma 17 can be applied iteratively and we can divide the query range into more than two parts in the same manner. Next, we present two more lemmas that will be used to describe the method for searching the adjusted right boundary.

▶ **Lemma 18.** *Consider a canonical set $I = [x_1, x_2]$ and a query range $Q(I) = [x_1, x_2] \times [c, +\infty)$ for some c. Let $right(TS(I)) = (r_x, r_y)$ denote the lowermost point in $TS(I)$. If $c \leq r_y$, then there are at least $B \log N$ distinct colors on the skyline of $Q(I)$, where B is the size of the block.*

Proof. By definition, $TS(I)$ contains $B \log N$ topmost points with distinct colors from the skyline of I. Each such point $p = (p_x, p_y) \in TS(I)$ satisfies $p_x \in [x_1, x_2]$ and $p_y \geq r_y$. Since $c \leq r_y \leq p_y$, we conclude that $p \in Q(I)$. Therefore all points from $TS(I)$ are also in $Q(I)$. Since points in $TS(I)$ are on the skyline of I, they are also on the skyline of $Q(I)$. Since all points in $TS(I)$ have different colors, the skyline of $Q(I)$ must contain at least $B \log N$ distinct colors. ◀

▶ **Lemma 19.** *Consider a canonical set $I = [x_1, x_2] \times (-\infty, +\infty)$ and a query range $Q = [x_1, x_2] \times [c, +\infty)$ such that $c > r_y$, where $right(TS(I)) = (r_x, r_y)$ is the rightmost (i.e., the lowermost) point in $TS(I)$. A color α occurs on the skyline of $Q \cap P$ if and only if it occurs on the skyline of $Q' \cap P$, where $Q' = [x_1, b'] \times (-\infty, +\infty)$ and $b' = \texttt{adj}(TS(I), Q)$.*

Proof. Let p be the topmost point of color α on the skyline of $Q \cap P$. By definition of $\texttt{adj}(TS(I), Q)$, we have $p_x \leq b'$, and since b' is the rightmost x-coordinate among the topmost points of distinct colors on the skyline of Q, there is no point of color α in $[b', x_2] \times [c, +\infty)$ that dominates p. Also, since $p_y \geq c$, no point in $[x_1, b'] \times (-\infty, c]$ dominates p. Hence, p is on the skyline of $Q' \cap P$ and the color α occurs on the skyline of $Q' \cap P$. Conversely, let p' be the topmost point of color α on the skyline of $Q' \cap P$. Since $p' \in Q' \cap P$, $p_x \leq b'$. By definition of $\texttt{adj}(TS(I), Q)$, $p_y \geq c$ and $b' \leq x_2$. This implies $p'_x \leq b' \leq x_2$ and hence $p' \in Q \cap P$. As p' is already the topmost point of its color and lies to the left of b', no point in $[b', x_2] \times [c, +\infty) \subseteq Q$ can dominate it. Therefore, p' also lies on the skyline of $Q \cap P$ and the color α must also be on the skyline of $Q \cap P$. ◀

In what follows, we describe the method for searching the adjusted right boundary in detail (see Figure 6 for a visual illustration).

`Step 1.` We calculate $\texttt{adj}(P, Q_{rmarg})$ or determine that $Q_{rmarg} \cap P = \emptyset$ using the data structure for the chunk that contains Q_{rmarg}. By Lemma 17, $\texttt{adj}(P, Q) = \texttt{adj}(P, Q_{rmarg})$ if $Q_{rmarg} \cap P \neq \emptyset$, and using Lemma 16 we can find $\texttt{adj}(P, Q_{rmarg})$ in $O(1)$ time. If Q_{rmarg} is empty, we proceed with Step 2.

`Step 2.` Let $h = \lfloor \frac{b_1 - a_1 + 1}{\Delta} \rfloor$. We divide Q_{middle} into two query ranges, Q_{lmid} and Q_{rmid}. Let $Q_{lmid} = [a_1, a_1 + 2^h\Delta - 1] \times [c, +\infty)$ and $Q_{rmid} = [b_1 - 2^h\Delta + 1, b_1] \times [c, +\infty)$. Let I_{lmid} denote the canonical set that contains all points with x-coordinates in $[a_1, a_1 + 2^h\Delta - 1]$. Let I_{rmid} denote the canonical set that contains all points with x-coordinates in $[b_1 - 2^h\Delta + 1, b_1]$. We observe that $Q_{lmid} \cap P \subseteq I_{lmid}$ and $Q_{rmid} \cap P \subseteq I_{rmid}$ for two canonical sets $I_{lmid} = [a_1, a_1 + 2^h\Delta - 1] \times (-\infty, +\infty)$ and $I_{rmid} = [b_1 - 2^h\Delta + 1, b_1] \times (-\infty, +\infty)$.

`Step 2(a).` Let p_t denote the topmost point in $TS(I_{rmid})$ and let p_l denote the lowermost point in $TS(I_{rmid})$. Since points in $TS(I_{rmid})$ are sorted by y-coordinates, we can identify p_t and p_l in $O(1)$ I/Os. We distinguish between three different cases.

`Case 1.` $\boldsymbol{p_t.y < c}$: In this case $Q_{rmid} \cap P = \emptyset$ and we proceed with Step 3.

`Case 2.` $\boldsymbol{p_l.y \geq c}$: In this case we find $\texttt{adj}(Q)$ in $O(\log_B N) = O(\log N)$ I/Os, but we can show that $k \geq B \log N$. By Lemma 18, the total number of colors on the skyline of Q_{rmid} is at least $B \log N$. Since $Q_{rmid} \subseteq Q$, $k \geq B \log N$, where k is the total number of colors on the skyline of Q. Since $k \geq B \log N$, the adjusted right boundary $\texttt{adj}(Q_{rmid}, P)$ is found in $O(\log_B N) = O(\frac{k}{B})$ I/Os.

`Case 3.` $\boldsymbol{p_t.y \geq c}$ `and` $\boldsymbol{p_l.y < c}$: In this case it is sufficient to find $\texttt{adj}\big(TS(I_{\text{rmid}}), Q\big)$ as can be concluded from Lemma 19. Since $TS(I_{rmid})$ contains $B \log N$ points, we can find $\texttt{adj}(TS(I_{rmid}), Q)$ in $O(1)$ I/Os using Lemma 15.

`Step 3.` If $Q_{rmid} \cap P = \emptyset$, we process Q_{lmid} in the same way as was described for Q_{rmid} in Step 2.

`Step 4.` If $Q_{lmid} \cap P = \emptyset$, we find $\texttt{adj}(P, Q) = \texttt{adj}(P, Q_{lmarg})$ in $O(1)$ I/Os, in the same way as described for Q_{rmarg} in Step 1.

Space Usage

Each individual chunk contains $B \log^2 N$ points, and there are N/Δ chunks, where $\Delta = B \log^2 N$. Therefore, the total space usage of chunk data structures is $O(N)$. Additionally, $TS(I)$ for every canonical set I stores $O(B \log N)$ points, and there are $O\big(\frac{N}{\Delta} \log \frac{N}{\Delta}\big)$ canonical

sets because for every chunk there are $O\big(\log(N/\Delta)\big)$ sets of exponentially increasing size starting at the left boundary of that chunk. In total, the data structure occupies $O(N)$ words.

Our procedure for transforming a three-sided top-open query range into a two-sided query range can be summarized in the following lemma.

▶ **Lemma 20.** *For any query range $Q = [a, b] \times [c, +\infty)$, we can find some value b' that satisfies the following property: a color α occurs on a skyline of $Q \cap P$ iff it occurs on the skyline of $Q' \cap P$ for $Q' = [a, b'] \times (-\infty, +\infty)$. The cost of finding b' is $O(k/B)$ I/Os, where k is the number of colors on the skyline of Q and the underlying data structure uses space $O(N)$.*

Color Reporting Queries

The distinct colors on the skyline of $Q = [a, b] \times [c, +\infty)$ can be reported as follows. We find the value of b', such that a color occurs on the skyline of Q iff it occurs on the skyline of $Q' = [a, b'] \times (-\infty, +\infty)$. By Lemma 20 this can be done in $O(k/B)$ I/Os where k is the number of colors on the skyline of Q. Then we access the partially persistent list L_c at the version b', i.e $L_c(b')$, and traverse the list in order of decreasing x-coordinates. We stop when a point p with $p.x < a$ is reached. For each point we report its color. The time required to examine all such points and report their colors is $O(k'/B)$ I/Os, where k' is the number of points $p \in L_c(b')$ with $p.x \geq a$. By Lemma 12, for every color on the skyline of Q' there is exactly one point p with that color, such that $p \in L_c(b')$ and $p.x \geq a$.

▶ **Theorem 21.** *In the external memory model, there exists a $O(N)$-space data structure that answers top-open three-sided skyline color reporting queries on an $N \times N$ grid in $O(\frac{k}{B}) + 1$ I/Os.*

The general case can be reduced to the case when points are on $N \times N$ grid at the cost of increasing the query cost by $q_{pred}(N)$; see the remark after Theorem 5.

7 Conclusion

In this paper we show that there is a gap between top-open and bottom-open three-sided skyline counting queries: while bottom-open queries have the same $\Theta(\log N/\log\log N)$ time complexity as four-sided skyline counting queries, top-open queries can be answered in $O(\log\log N)$ time. Our technique also lead to improved results on (top-open) three-sided skyline color counting and (top-open) three-sided range reporting in external memory.

One important open question related to our work concerns the complexity of four-sided skyline range reporting. The best data structure for four-sided range reporting in 2-D answers queries in $O(\log\log N + k)$ time. This time is known to be optimal. On the other hand, the fastest currently known data structure for four-sided skyline range reporting answers queries is $O(\log N/\log\log N + k)$ time. Is there a data structure that uses $N\log^{O(1)} N$ space and supports four-sided skyline reporting queries in $O(\log\log N + k)$ time?

References

1 Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2000. doi:10.1109/SFCS.2000.892088.

2 Bruno Becker, Stephan Gschwind, Thomas Ohler, Bernhard Seeger, and Peter Widmayer. An asymptotically optimal multiversion b-tree. *The VLDB Journal*, pages 264–275, 1996. doi:10.1007/s007780050028.

3 Gerth Stølting Brodal and Kasper Green Larsen. Optimal planar orthogonal skyline counting queries. In *14th Scandinavian Symposium and Workshops (SWAT)*, pages 110–121, 2014. doi:10.1007/978-3-319-08404-6_10.

4 Timothy M. Chan. Persistent predecessor search and orthogonal point location on the word RAM. *ACM Transactions on Algorithms*, pages 22:1–22:22, 2013. doi:10.1145/2483699.2483702.

5 Ananda Swarup Das, Prosenjit Gupta, Anil Kishore Kalavagattu, Jatin Agarwal, Kannan Srinathan, and Kishore Kothapalli. Range aggregate maximal points in the plane. In *6th International Workshop on Algorithms and Computation (WALCOM)*, 2012. doi:10.1007/978-3-642-28076-4_8.

6 Ananda Swarup Das, Prosenjit Gupta, and Kannan Srinathan. Counting maximal points in a query orthogonal rectangle. In *7th International Workshop on Algorithms and Computation (WALCOM)*, 2013. doi:10.1007/978-3-642-36065-7_8.

7 Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, pages 424–436, 1993. doi:10.1016/0022-0000(93)90040-4.

8 Harold N. Gabow, Jon Louis Bentley, and Robert E. Tarjan. Scaling and related techniques for geometry problems. In *Proceedings of the 16th annual ACM Symposium on Theory of computing (STOC)*, pages 135–143, 1984. doi:10.1145/800057.808675.

9 Arnab Ganguly, Daniel Gibney, Sharma V. Thankachan, and Rahul Shah. I/O-optimal categorical 3-sided skyline queries. *Theoretical Computer Science*, pages 132–144, 2021. doi:10.1016/j.tcs.2021.10.010.

10 Anil Kishore Kalavagattu, Jatin Agarwal, Ananda Swarup Das, and Kishore Kothapalli. On counting range maxima points in plane. In *Combinatorial Algorithms, 23rd International Workshop (IWOCA)*, pages 263–273, 2012. doi:10.1007/978-3-642-35926-2_28.

11 Casper Kejlberg-Rasmussen, Yufei Tao, Konstantinos Tsakalidis, Kostas Tsichlas, and 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)*, pages 103–114, 2013. doi:10.1145/2463664.2465225.

12 J. Ian Munro, Yakov Nekrich, and Sharma V. Thankachan. Range counting with distinct constraints. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG)*, 2015. URL: http://research.cs.queensu.ca/cccg2015/CCCG15-papers/44.pdf.

13 Gonzalo Navarro and Javiel Rojas-Ledesma. Predecessor search. *ACM Computing Surveys*, 53(5):105:1–105:35, 2021. doi:10.1145/3409371.

14 Yakov Nekrich and Gonzalo Navarro. Sorted range reporting. In *13th Scandinavian Symposium and Workshops (SWAT)*, pages 271–282. Springer, 2012. doi:10.1007/978-3-642-31155-0_24.

15 Manish Patil, Sharma V. Thankachan, Rahul Shah, Yakov Nekrich, and Jeffrey Scott Vitter. Categorical range maxima queries. In *33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 266–277. ACM, 2014. doi:10.1145/2594538.2594557.

16 Mihai Patrascu. Lower bounds for 2-dimensional range counting. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 40–46, 2007. doi:10.1145/1250790.1250797.

17 Jeffrey Scott Vitter. External memory algorithms and data structures: Dealing with massive data. *ACM Computing surveys (CsUR)*, pages 209–271, 2001. doi:10.1145/384192.384193.

A Proofs of Lemmas 15 and 16

▶ **Lemma 15.** *Let S be a set of integers such that $|S| = O(B \log^2 N)$. There exists a data structure that uses space $O(|S|)$ and supports predecessor queries on S in $O(1)$ I/Os.*

Proof. If $B \geq \log N$, then $|S| \leq B^2$. In this case we store S in a B-tree so that a predecessor query is answered in $O(\log_B |S|) = O(1)$ I/Os. If $B < \log N$, $|S| \leq \log^2 N$. In this case we store S in a fusion tree [7], so that a predecessor query is answered in $O(\log_{\log n} |S|) = O(\frac{\log |S|}{\log \log n}) = O(1)$. ◀

▶ **Lemma 16.** *Let P' be a set of two-dimensional points such that $|P'| = O(B \log^2 N)$. There exists a data structure that uses space $O(|P'|)$ and answers range predecessor queries on P' in $O(1)$ I/Os.*

Proof. According to Lemma 7, the rightmost point in a query range Q can be found using a partially persistent data structure. If $B \geq \log N$, we store the points in a partially persistent B-Tree [2]. If $B < \log N$, we store the points in a partially persistent variant of the fusion tree. ◀