A New Lower Bound for Semigroup Orthogonal Range Searching

Peyman Afshani
Aarhus University, Denmark
peyman@cs.au.dk

Abstract

We report the first improvement in the space-time trade-off of lower bounds for the orthogonal range searching problem in the semigroup model, since Chazelle's result from 1990. This is one of the very fundamental problems in range searching with a long history. Previously, Andrew Yao's influential result had shown that the problem is already non-trivial in one dimension [14]: using m units of space, the query time $Q(n)$ must be $\Omega(\alpha(m,n) + \frac{n}{m-n+1})$ where $\alpha(\cdot,\cdot)$ is the inverse Ackermann's function, a very slowly growing function. In d dimensions, Bernard Chazelle [9] proved that the query time must be $Q(n) = \Omega((\log_\beta n)^{d-1})$ where $\beta = 2m/n$. Chazelle's lower bound is known to be tight for when space consumption is "high" i.e., $m = \Omega(n \log^{d+\varepsilon} n)$.

We have two main results. The first is a lower bound that shows Chazelle's lower bound was not tight for "low space": we prove that we must have $mQ(n) = \Omega(n(\log n \log\log n)^{d-1})$. Our lower bound does not close the gap to the existing data structures, however, our second result is that our analysis is tight. Thus, we believe the gap is in fact natural since lower bounds are proven for idempotent semigroups while the data structures are built for general semigroups and thus they cannot assume (and use) the properties of an idempotent semigroup. As a result, we believe to close the gap one must study lower bounds for non-idempotent semigroups or building data structures for idempotent semigroups. We develope significantly new ideas for both of our results that could be useful in pursuing either of these directions.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Computational geometry

Keywords and phrases Data Structures, Range Searching, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.3

Related Version A full version of the paper is available at [1], `https://arxiv.org/abs/1903.07967`.

Funding *Peyman Afshani*: supported by DFF (Det Frie Forskningsräd) of Danish Council for Indepndent Reserach under grant ID DFF−7014−00404.

1 Introduction

Orthogonal range searching in the semigroup model is one of the most fundamental data structure problems in computational geometry. In the problem, we are given an input set of points to store in a data structure where each point is associated with a weight from a semigroup $\mathcal{G}$ and the goal is to compute the (semigroup) sum of all the weights inside an axis-aligned box given at the query time. Disallowing the "inverse" operation in $\mathcal{G}$ makes the data structure very versatile as it is then applicable to a wide range of situations (from computing weighted sum to computing the maximum or minimum inside the query). In fact, the semigroup variant is the primary way the family of range searching problems are introduced, (see the survey [4]).

Here, we focus only on static data structures. We use the convention that $Q(n)$, the query time, refers to the worst-case number of semigroup additions required to produce the query answer $S(n)$, space, refers to the number of semigroup sums stored by the data structure. By storage, denoted by $S^+(n)$, we mean space but not counting the space used by the input,

© Peyman Afshani;
licensed under Creative Commons License CC-BY

i.e., $S(n) = n + S^+(n)$. So we can talk about data structures with sublinear space, e.g., with 0 storage the data structure has to use the input weights only, leading to the worst-case query time of n.

1.1 The Previous Results

Orthogonal range searching is a fundamental problem with a very long history. The problem we study is also very interesting from a lower bound point of view where the goal is to understand the fundamental barriers and limitations of performing basic data structure operations. Such a lower bound approach was initiated by Fredman in early 80s and in a series of very influential papers (e.g., see [10, 11, 12]). Among his significant results, was the lower bound [11, 12] that showed a sequence of n insertions, deletions, and queries requires $\Omega(n \log n)$ time to run.

Arguably, the most surprising result of these early efforts was given by Andrew Yao who in 1982 showed that even in one dimension, the static case of the problem contains a very non-trivial, albeit small, barrier. In one dimension, the problem essentially boils down to adding numbers: store an input array A of n numbers in a data structure s.t., we can add up the numbers from $A[i]$ to $A[j]$ for i and j given at the query time. The only restriction is that we should use only additions and not subtractions (otherwise, the problem is easily solved using prefix sums). Yao's significant result was that answering queries requires $\Omega(\alpha(S(n), n) + n/S^+(n))$ additions, where $\alpha(\cdot, \cdot)$ is the inverse Ackermann function. This bound implies that if one insists on using $O(n)$ storage, the query bound cannot be reduced to constant, but even using a miniscule amount of extra storage (e.g., a $\log^* \log^* n$ factor extra storage) can reduce the query bound to constant. Furthermore, using a bit less than n storage, e.g., by a $\log^* \log^* n$ factor, will once again yield a more natural (and optimal) bound of $n/S^+(n)$. Despite its strangeness, it turns out there are data structures that can match the exact lower bound (see also [5]). After Tarjan's famous result on the union-find problem [13], this was the second independent appearance of the inverse Ackermann function in the history of algorithms and data structures.

Despite the previous attempts, the problem is still open even in two dimensions. At the moment, using range trees [7, 8] on the 1D structures is the only way to get two or higher dimensional results. In 2D for instance, we can have $S^+(n) = O(n/\log n)$ with query bound $Q(n) = O(\log^3 n)$, or $S^+(n) = O(n)$ with query bound $Q(n) = O(\log^2 n)$, or $S^+(n) = O(n \log n)$ with query bound $O(\alpha(cn, n) \log n)$, for any constant c. In general and in d dimensions, we can build a structure with $S^+(n) = O(n \log^{d-1} n)$ units of storage and with $Q(n) = O(\alpha(cn, n) \log^{d-1} n)$ query bound, for any constant c. We can reduce the space complexity by any factor t by increasing the query bound by another factor t. Also, strangely, if t is asymptotically larger than $\alpha(n, n)$, then the inverse Ackermann term in the query bound disappears. Nonetheless, a surprising result of Chazelle [9] shows that the reverse is not true: the query bound must obey $Q(n) = \Omega((\log_{S(n)/n} n)^{d-1})$ which implies using polylogarithmic extra storage only reduces the query bound by a $(\log \log n)^{d-1}$ factor. Once again, using range tree with large fan out, one can build a data structure that uses $O(n \log^{2d-2+\varepsilon} n)$ storage, for any positive constant ε, and achieves the query bound of $O((\log_{\log n} n)^{d-1})$. This, however leaves a very natural and important open problem: *Is Chazelle's lower bound the only barrier? Is it possible to achieve $O(n)$ space and $O(\log^{d-1} n)$ query time?*

Idempotence and random point sets. A semigroup is idempotent if for every $x \in \mathcal{G}$, we have $x + x = x$. All the previous lower bounds are in fact valid for idempotent semigroups. Furthermore, Chazelle's lower bound uses a uniform (or randomly placed) set of points which shows the lower bound does not require pathological or fragile input constructions.

Furthermore, his lower bound also holds for dominance ranges, i.e., d-dimensional boxes in the form of $(-\infty, a_1] \times \cdots \times (-\infty, a_d]$. These little perks result in a very satisfying statement: problem is still difficult even when $\mathcal{G}$ is "nice" (idempotent), and when the point set is "nice" (uniformly placed) and when the queries are simple ("dominance queries").

1.2 Our Results

We show that for any data structure that uses $S^+(n)$ storage and has query bound of $Q(n)$, we must have $S^+(n) \cdot Q(n) = \Omega(n(\log n \log\log n)^{d-1})$. This is the first improvement to the storage-time trade-off curve for the problem since Chazelle's result in 1990. It also shows that Chazelle's lower bound is not the only barrier. Observe that our lower bound is strong at a different corner of parameter space compared to Chazelle's: ours is strongest when storage is small whereas Chazelle's is strongest when the storage is large. Furthermore, we also keep most of the desirable properties of Chazelle's lower bound: our lower bound also holds for idempotent semigroups and uniformly placed point sets. However, we have to consider more complicated queries than just dominance queries which ties to our second main result. We show that our analysis is tight: given a "uniformly placed" point set and an idempotent semigroup $\mathcal{G}$, we can construct a data structure that uses $O(n)$ storage and has the query bound of $O((\log n \log\log n)^{d-1})$. As a corollary, we provide an almost complete understanding of orthogonal range searching queries with respect to a uniformly placed point set in an idempotent semigroup.

Challenges. Our results and specially our lower bound require significantly new ideas. To surpass Chazelle's lower bound, we need to go beyond dominance queries which requires wrestling with complications that ideas such as range trees can introduce. Furthermore, in our case, the data structure can actually improve the query time by a factor f by spending a factor f extra space. This means, we are extremely sensitive to how the data structure can "use" its space. As a result, we need to capture the limits of how intelligently the data structure can spend its budge of "space" throughout various subproblems.

Implications. It is natural to conjecture that the uniformly randomly placed point set should be the most difficult point set for orthogonal queries. Because of this, we conjecture that our lower bounds are almost tight. This opens up a few very interesting open problems. See Section 5.

2 Preliminaries

The Model of Computation. Let P be an input set of n points with weights from a semigroup $\mathcal{G}$. Our model of computation is the same as the one used by the previous lower bounds, e.g., [9]. There has been quite some work dedicated to building a proper model for lower bounds in the semigroup model. We will not delve into those details and we only mention the final consequences of the efforts. The data structure stores a number of sums where each sum s is the sum of the weights of a subset $s_P \subset P$. With a slight abuse of the notation, we will use s to refer both to the sum as well as to the subset s_P. The number of stored sums is the space complexity of the data structure. If a sum contains only one point, then we call it a **singleton** and we use $S^+(n)$ to denote the storage occupied by sums that are not singletons. Now, consider a query range r containing a subset $r_P = r \bigcap P$. The query algorithm must find k stored subsets $s_1, \dots, s_k$ such that $r_P = \cup_{i=1}^k s_i$. For a given query r, the smallest such integer k is the query bound of the query. The query bound of the data structure is the worst-case query bound of any query. Observe that the data structure does not disallow covering any point more than once and in fact, for idempotent semigroups this poses no problem. All the known lower bounds work in this way, i.e., they allow covering a point inside the query multiple times. However, if the semigroup is not idempotent, then covering a point more than once could lead to incorrect results. Since data structures work for general semigroups, they ensure that $s_1, \dots, s_k$ are disjoint.

Definitions and Notations. A d-dimensional dominance query is determined by one point $(x_1, \dots, x_d)$ and it is defined as $(-\infty, x_1] \times \dots \times (-\infty, x_d]$.

▶ **Definition 1.** *We call a set $P \subset \mathbb{R}^d$* **well-distributed** *if the following properties hold: (i) P is contained in the d-dimensional unit cube. (ii) The volume of any rectangle that contains $k \geq 2$ points of P is at least $\varepsilon_d k/|P|$ for some constant ε_d that only depends on the dimension. (iii) Any rectangle that has volume v, contains at most $\lceil v|P|/\varepsilon_d \rceil$ points of P.*

▶ **Lemma 2** ([2, 9, 3])**.** *For any constant d and any given value of n, there exists a well-distributed point set in $\mathbb{R}^d$ containing $\Theta(n)$ points.*

3 The Lower Bound

This section is devoted to the proof of our main theorem which is the following.

▶ **Theorem 3.** *If P is a well-distributed point set of n points in $\mathbb{R}^d$, any data structure that uses $S^+(n)$ storage, and answers $(2d-1)$-sided queries in $Q(n)$ query bound requires that $S^+(n) \cdot Q(n) = \Omega(n(\log n \log\log n)^{d-1})$.*

Let $\mathcal{Q}$ be the unit cube in $\mathbb{R}^d$. Throughout this section, the input point set is a set P of n well-distributed points in $\mathcal{Q}$. Let $\mathcal{D}$ be a data structure that answers semigroup orthogonal range searching queries on P.

3.1 Definitions and Set up

We consider queries that have two boundaries in dimensions 1 to $d-1$ but only have an upper bound in dimension d. For simplicity, we rename the axes such that the d-th axis is denoted by Y and the first $d-1$ axes are denoted by $X_1, \dots, X_{d-1}$. Thus, each query is in the form of $[x_1', x_1] \times \dots [x_{d-1}', x_{d-1}] \times (-\infty, y]$. The point $(x_1, \dots, x_{d-1}, y)$ is defined as the **dot of q** and is denoted by $\mathrm{Dot}(q)$. For every $1 \leq i \leq d-1$, the line segment that connects $\mathrm{Dot}(q)$ to the point $(x_1, \dots, x_{i-1}, x_i', x_{i+1}, \dots, x_{d-1}, y)$ is called the i-th marker of q and it is denoted by $t_i(s)$.

The tree T_i. For each dimension $i = 1, \dots, d-1$, we define a balanced binary tree T_i of height $h = \log n$ as follows. Informally, we cut $\mathcal{Q}$ into 2^h congruent boxes with hyperplanes perpendicular to axis X_i which form the leaves of T_i. To be more specific, every node in T_i is assigned a box $r(v) \subset \mathcal{Q}$. The root of T_i is assumed to have depth 0 and it is assigned $\mathcal{Q}$. For every node v, we divide $r(v)$ into two congruent "left" and "right" boxes with a hyperplane $\ell(v)$, perpendicular to X_i axis. The left box is assigned to left child of v and similarly the right box is assigned to the right child of v. We do not do this if $r(v)$ has volume less than $1/n$; these nodes become the leaves of T. Observe that all trees T_i, $1 \leq i \leq d-1$ have the same height h. The volume of $r(v)$ for a node v at depth j is 2^{-j}.

Embedding the problem in $\mathbb{R}^{2d-1}$. The next idea is to embed our problem in $\mathbb{R}^{2d-1}$. Consistent with the previous notation, the first d axes are $X_1, \ldots, X_{d-1}$ and Y. We label the next axis $Z_1, \ldots, Z_{d-1}$. We now represent T_i geometrically as follows. Consider h the height of T_i. For each i, $1 \le i \le d-1$, we now define a **representative diagram Γ_i** which is a axis-aligned decomposition of the unit (planar) square Q_i in a coordinate system where the horizontal axis is X_i and the vertical axis is Z_i. As the first step of the decomposition, cut Q_i into h equal-sized sub-rectangles using $h-1$ horizontal lines. Next, we will further divide each sub-rectangle into small regions and we will assign every node v of T_i to one of these regions. This is done as follows. The root v of T_i is assigned the topmost sub-rectangle as its region, $\gamma(v)$. Assume v is assigned a rectangle $\gamma(v)$ as its region. We create a vertical cut starting from the middle point of the lower boundary of $\gamma(v)$ all the way down to the bottom of the rectangle Q_i. The children of v are assigned to the two rectangles that lie immediately below $\gamma(v)$. See Figure 1.

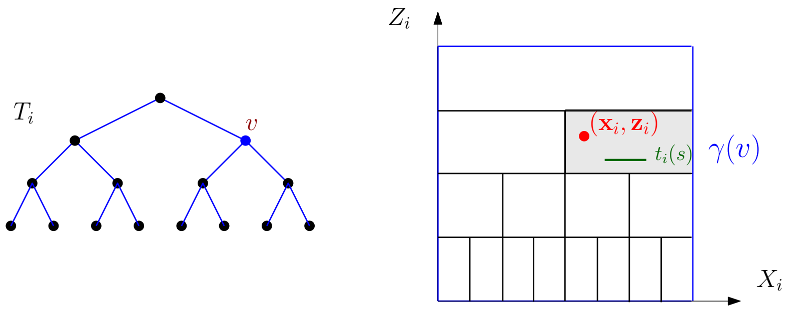

Figure 1 A tree and its representative diagram. The region of a node v is highlighted in grey.

Placing the Sums. Consider a semigroup sum s stored by the data structure $\mathcal{D}$. Our lower bound will also apply to semigroups that are idempotent which means without loss of generality, we can assume that our semigroup is idempotent. As a result, we can assume that each semigroup sum s stored by the data structure has the same shape as the query. Let $b(s)$ be the smallest box that is unbounded from below (along the Y axis) that contains all the points of s. If s does not include a point, p, inside $b(s)$, we can just p to s. Any query that can use s must contain the box $b(s)$ which means adding p to s can only improve things. Each sum s is placed in one node of T_i for every $1 \le i \le d-1$. The details of this placement are as follows.

A node v_i in T_i stores any sum s such that the i-th marker of s, $t_i(s)$, intersects $\ell(v_i)$ with v being the *highest* node with this property. Geometrically, this is equivalent to the following: we place s at a node v if $\gamma(v)$ is the *lowest* region that fully contains the segment $t_i(s)$ (or to be precise, the projection of $t_i(s)$ onto the Z_iX_i plane). For example, in Figure 1(right), the sum s is placed at v in T_i since $t_i(s)$, the green line segment, is completely inside $\gamma(v)$ with v being the lowest node of this property. Remember that s is placed at some node in each tree T_i, $1 \le i \le d-1$ (i.e., it is placed $d-1$ times in total).

Notations and difficult queries. We will adopt the convention that random variables are denoted with bold math font. The difficult query is a $2d-1$ sided query chosen randomly as follows. The query is defined as $[\mathbf{x}'_1, \mathbf{x}_1] \times \cdots \times [\mathbf{x}'_{d-1}, \mathbf{x}_{d-1}] \times (-\infty, \mathbf{y}]$ where $\mathbf{x}'_i, \mathbf{x}_i$ and $\mathbf{y}$ are also random variables (to be described). $\mathbf{y}$ is chosen uniformly in $[0,1]$. To choose the remaining coordinates, we do the following. We place a random point $(\mathbf{x}_i, \mathbf{z}_i)$ uniformly inside the representative plane Γ_i (i.e., choose $\mathbf{x}_i$ and $\mathbf{z}_i$ uniformly in $[0,1]$). Let $\mathbf{v}_i$ be the random variable denoting the node in T_i s.t., region $\gamma(\mathbf{v}_i)$ contains the point $(\mathbf{x}_i, \mathbf{z}_i)$. $\mathbf{x}'_i$ is the X_i-coordinate of the left boundary of $\gamma(\mathbf{v}_i)$. Let $\boldsymbol{\ell}_i$ be the depth of $\mathbf{v}_i$ in T_i. We denote the point $(\mathbf{x}_1, \ldots, \mathbf{x}_{d-1}, \mathbf{y})$ by $\mathbf{q}$ and denote the $2d-1$ sided query by $\mathrm{Dom}_{\mathbf{v}_1,\ldots,\mathbf{v}_{d-1}}(\mathbf{q})$. See Figure 1(right). Note that a query $\mathrm{Dom}_{v_1,\ldots,v_{d-1}}(q)$ is equivalent to a dominance query defined by point q in $r(v_1) \cap \cdots \cap r(v_{d-1})$. To simplify the presentation and to stop redefining these concepts, we will reserve the notations introduced in this paragraph to only represent the concepts introduced here.

▶ **Observation 4.** *A necessary condition for being able to use a sum s to answer $Dom_{v_1,\ldots,v_{d-1}}(q)$ is that s is stored at the subtree of v_i, for every $1 \le i \le d-1$.*

Proof. Due to how we have placed the sums, the sums stored at the ancestors of v_i contain at least one point that lies outside $r(v_i)$ and since $\mathrm{Dom}(q)$ is entirely contained inside $r(v_i)$ those sums cannot be used to answer the query. ◀

Subproblems. Consider a query $\mathrm{Dom}(q) = \mathrm{Dom}_{v_1,\ldots,v_{d-1}}(q)$. We now define subproblems of $\mathrm{Dom}(q)$. A subproblem is represented by an array of $d-1$ integral indices $\mathfrak{j} = (j_1, \ldots, j_{d-1})$ and it is denoted as $\mathfrak{j}$-subproblem. The state of $\mathfrak{j}$-subproblem of a query $\mathrm{Dom}_{v_1,\ldots,v_{d-1}}(q)$ could either be *undefined*, or it could refer to covering a particular subset of points inside the query. In particular, given $\mathrm{Dom}(q)$, a $\mathfrak{j}$-subproblem is undefined if for some $1 \le i \le d-1$, there is no node $u_i \in T_i$ with the following properties: u_i has depth $\ell_i + j_i$, u_i has a right sibling u'_i with $r(u'_i)$ containing the query point q. See Figure 2. However, if such nodes u_i exist for all $1 \le i \le d-1$, then the $\mathfrak{j}$-subproblem of $\mathrm{Dom}(q)$ is **well-defined** and it refers to the problem of covering all the points inside the region $\mathrm{Dom}(q) \cap r(u_1) \cap \cdots \cap r(u_{d-1})$; observe that this is equivalent to covering all the points inside the region $r(u_1) \cap \cdots \cap r(u_{d-1})$ that have Y-coordinate at most y. Further observe that for u_i to exist in T_i, it needs to pass two **checks**: **(check I)** $\ell_i + j_i \le h$ as otherwise, there are no nodes with depth $\ell_i + j_i$ and **(check II)** a node u_i at depth $\ell_i + j_i$ has a right sibling u'_i with $r(u'_i)$ containing q. The nodes $u_1, \ldots, u_{d-1}$ are called the **defining nodes** of the $\mathfrak{j}$-subproblem. Thus, the random variable $\mathbf{v}_i$ defines the random variable $\mathbf{u}_i$ where $\mathbf{u}_i$ could be either undefined or it could be a node in T_i. Clearly, the distribution of $\mathbf{u}_i$ is independent of the distributions of $\mathbf{u}_j$ and $\mathbf{v}_j$ for $i \ne j$ as $\mathbf{u}_i$ only depends on $\mathbf{v}_i$.

▶ **Observation 5.** *Consider a well-defined $\mathfrak{j} = (j_1, \ldots, j_{d-1})$ subproblem of a query $Dom_{v_1,\ldots,v_{d-1}}(q)$ and its defining nodes $u_1, \ldots, u_{d-1}$. To solve the $\mathfrak{j}$-subproblem (i.e., to cover the points inside the subproblem), the data structure can use a sum s only if for every $1 \le i \le d-1$, we either have case (i) where s is stored at ancestors of u_i but not the ancestors of v_i or case (ii) where s is stored at the subtree of u_i. If a sum s violates one of these two conditions for some i, then it cannot be used to answer the $\mathfrak{j}$-subproblem. See Figure 2.*

Proof. See the full paper for the proof [1]. ◀

3.2 The Main Lemma

In this subsection, we prove a main lemma which is the heart of our lower bound proof. To describe this lemma, we first need the following notations. Consider a well-defined $\mathfrak{j}$-subproblem of a query $\mathrm{Dom}_{v_1,\ldots,v_{d-1}}(q)$ where $j_i \le \frac{h}{2}$ for $1 \le i \le d-1$. As discussed, this subproblem corresponds to covering all the points in the region $r(u_1) \cap \cdots \cap r(u_{d-1})$ whose Y-coordinate is below y, the Y-coordinate of point q; thus, the $\mathfrak{j}$-subproblem of the query

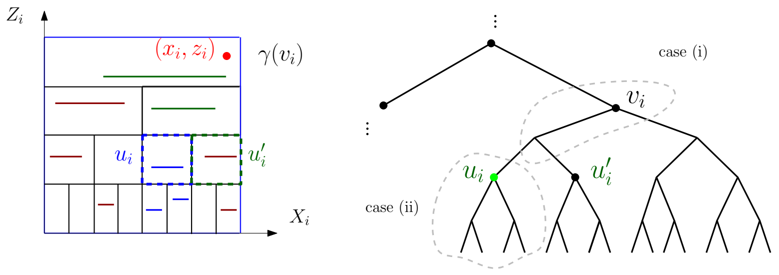

Figure 2 u_i is a defining node. The blue line segments correspond to $t_i(s)$ of a sum s that is placed in the subtree of u_i. The green ones correspond to those placed at ancestors of u_i but not at ancestors of v_i. The red ones correspond to sums that cannot be used to answer the subproblem.

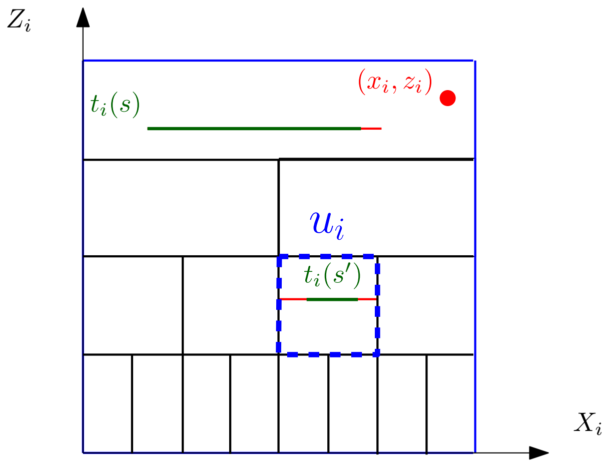

Figure 3 The extensions of two sums that can be used to answer a subproblem.

can be represented as the problem of covering all the points inside the box $[a_1, b_1] \times \cdots \times [a_{d-1}, b_{d-1}] \times (-\infty, y]$ where a_i and b_i correspond to the left and the right boundaries of the slab $r(u_i)$. Let $0 < \lambda$ be a parameter. Consider the region $[a_1, b_1] \times \cdots \times [a_{d-1}, b_{d-1}] \times [y - \beta, y]$ in which β is chosen such that the region contains λ points; as our pointset is well-distributed, this implies that the volume of the region is $\Theta(\lambda/n)$. We call this region **the $\boldsymbol{\lambda}$-top box**. The **$\boldsymbol{\lambda}$-top**, denoted by $\mathrm{Top}(\mathfrak{j}, \lambda)$, is then the problem of covering all the points inside the λ-top box of the $\mathfrak{j}$-subproblem. With a slight abuse of the notation, we will use $\mathrm{Top}(\mathfrak{j}, \lambda)$ to refer also to the set of points inside the λ-top box. If there are not enough points in the λ-top box, the λ-top is undefined, otherwise, it is well-defined. These of course also depend on the query but we will not write the dependency on the query as it will clutter the notation. Furthermore, observe that when the query is random, then $\mathrm{Top}(\mathfrak{j}, \lambda)$ becomes a random variable which is either undefined or it is some subset of points.

Extensions of sums. Due to technical issues, we slightly extend the number of points each sum covers. Consider a sum s stored at a subtree of v_i such that s can be used to answer the $\mathfrak{j}$-subproblem. By Observation 4, s is either placed at the subtree of u_i or on the path connecting v_i to u_i. We extend the X_i range of the sum s (i.e., the projection of s on the X_i) to include the left and the right boundary of the node u_i along the X_i-dimension. We do this for all $d - 1$ first dimensions to obtain an extension $e(s)$ of sum s. We allow the data structure to cover any point in $e(s)$ using s.

▶ **Lemma 6** (The Main Lemma). *Consider a* $\mathfrak{j} = (j_1, \ldots, j_{d-1})$ *subproblem of a random query* $Dom_{\mathbf{v}_1,\ldots,\mathbf{v}_{d-1}}(\mathbf{q})$, *for* $1 \le j_i \le h/2$. *Let* $\lambda = \frac{\delta h^{d-1}}{j_1 j_2 \cdots j_{d-1}} \cdot \frac{n}{S^+(\mathcal{A})}$ *where* δ *is a small enough constant and* $S^+(\mathcal{A})$ *is the storage of the data structure. Let* $\mathbf{S}_\mathfrak{j}$ *be the set of sums* s *such that (i)* s *is contained inside the query* $Dom_{\mathbf{v}_1,\ldots,\mathbf{v}_{d-1}}(\mathbf{q})$, *and (ii)* $e(s)$ *covers at least* C *points from* $Top(\mathfrak{j}, \lambda)$, *meaning,* $|e(s) \cap Top(\mathfrak{j}, \lambda)| \ge C$ *where* C *is a large enough constant.*

With $\Omega(1)$ *probability, the* $\mathfrak{j}$*-subproblem and the* $Top(\mathfrak{j}, \lambda)$ *are well-defined. Furthermore conditioned on both of these being well-defined, with probability* $1 - O(\sqrt{\delta/\varepsilon_d})$, *the nodes* $\mathbf{v}_1, \cdots, \mathbf{v}_{d-1}$ *will be sampled as nodes* $v_1, \cdots, v_{d-1}$, *s.t., the following holds:* $\mathbb{E}[\sum_{s \in \mathbf{S}_\mathfrak{j}} |e(s) \cap Top(\mathfrak{j}, \lambda)|] < \frac{|Top(\mathfrak{j},\lambda)|}{C'}$ *where the expectation is over the random choices of* $\mathbf{y}$ *and* C' *is another large constant.*

Let us give some intuition on what this lemma says and why it is critical for our lower bound. For simplicity assume $S^+(\mathcal{A}) = n$ and assume we sample $\mathbf{v}_1, \cdots, \mathbf{v}_{d-1}$ as the first step, and and then sample $\mathbf{y}$ as the last step. The above lemma implies that if we focus on one particular subproblem, the sums in the data structure cannot cover too many points; to see this consider the following. The lemma first says that after the first step, with positive constant probability, $\mathfrak{j}$-subproblem and $\mathrm{Top}(\mathfrak{j}, \lambda)$ are well-defined. Furthermore, here is a very high chance that our random choices will "lock us" in a "doomed" state, after sampling $v_1, \cdots, v_{d-1}$. Then, when considering the random choices of $\mathbf{y}$, sums that cover at least C points in total cover a very small fraction of the points. As a result, we will need $\Omega(\lambda/C) = \Omega(\frac{\delta \log^{d-1} n}{C j_1 j_2 \cdots j_{d-1}})$ sums to cover the points inside the λ-top of the subproblem. Summing these values over all possible subproblems, $j_i, 1 \le j_i \le h/2$, $1 \le i \le d-1$ will create a lot of Harmonic sums of the type $\sum_{x=1}^{h/2} x = O(\log \log n)$ which will eventually lead to our lower bound. In particular, we will have $\sum_{j_i, 1 \le j_i \le h/2} \Omega(\frac{h^{d-1}}{j_1 j_2 \cdots j_{d-1}}) = (\log n \log \log n)^{d-1}$. There is however, one very big technical issue that we will deal with later: a sum can cover very few points from each subproblem but from very many subproblems! Without solving this technical issue, we only get the bound $\max_{j_i, 1 \le j_i \le h/2} \Omega(\frac{h^{d-1}}{j_1 j_2 \cdots j_{d-1}}) = (\log n)^{d-1}$ which offers no improvements over Chazelle's lower bound. Thus, while solving this technical issue is important, nonetheless, it is clear that the lemma we will prove in this section is also very critical.

As this subsection is devoted to the proof of the above lemma, we will assume that we are considering a fixed $\mathfrak{j}$-subproblem and thus the indices $j_1, \ldots, j_{d-1}$ are fixed.

3.2.1 Notation and Setup

By Observation 5, only a particular set of sums can be used to answer the $\mathfrak{j}$-subproblem of a query. Consider a sum s that can be used to answer the subproblem of some query. By the observation, we must have that s must either satisfy case (i) or case (ii) for every tree T_i, $1 \le i \le d-1$. Over all indices i, $1 \le i \le d-1$, they describe $2^{d-1} = O(1)$ different cases. This means that we can partition $\mathbf{S}_\mathfrak{j}$ into 2^{d-1} different **equivalent classes** s.t., for any two sums s_1 and s_2 in an equivalent class, either they both satisfy case (i) or they both satisfy case (ii) in Observation 5 and for any dimension i. Since 2^{d-1} is a constant, it suffices to show that our lemma holds when only considering sums of particular equivalent class. In particular, let $S'_\mathfrak{j}$ be the subset of eligible sums that all belong to one equivalent class. Now, it suffices to show that $\mathbb{E}[\sum_{s \in S'_\mathfrak{j}} |e(s) \cap \mathrm{Top}(\mathfrak{j}, \lambda)|] < \frac{|\mathrm{Top}(\mathfrak{j},\lambda)|}{2^d C'}$. since summing these over all 2^{d-1} equivalent classes will yield the lemma. Furthermore, w.l.o.g and by renaming the X-axes, we can assume that there exists a fixed value t, $0 \le t \le d-1$, such that for every sum $s \in S'_\mathfrak{j}$, for dimensions $1 \le i \le t$, s satisfies case (i) in T_i and for $t < i \le d-1$, s is within case (ii). Note that if $t = 0$, then it implies that we have no instances of case (i) and for $t = d-1$ we have no instances of case (ii).

The probability distribution of subproblems. To proceed, we need to understand the distribution of the subproblems. This is done by the following observation.

▶ **Observation 7.** *Consider a* $\mathfrak{j} = (j_1, \ldots, j_{d-1})$ *subproblem of a random query* $Dom_{\mathbf{v}_1,\ldots,\mathbf{v}_{d-1}}(\mathbf{q})$ *defined by random variables* $\mathbf{u}_1, \ldots, \mathbf{u}_{d-1}$*. We can make the following observations. (i) the distribution of the random variable* $j_i + \boldsymbol{\ell}_i$ *is uniform among the integers* $j_i + 1, \ldots, h + j_i$*. (ii) With probability* j_i/h*,* $\mathbf{u}_i$ *will be undefined because it fails (Check I). (iii) If (Check I) does not fail for* $\mathbf{u}_i$*, there is exactly 0.5 probability that* $\mathbf{u}_i$ *is undefined. (iv) For a fixed* j_i*, the probability distribution,* μ_i*, of* $\mathbf{u}_i$ *is as follows: with probability* $1 - \frac{1-j_i/h}{2}$*,* $\mathbf{u}_i$ *is undefined. Otherwise,* $\mathbf{u}_i$ *is a node in* T_i *sampled in the following way: sample a random integer (depth)* $\boldsymbol{\ell}'$ *uniformly among integers in* $j_i + 1, \ldots, h$ *and select a random node uniformly among all the nodes at depth* $\boldsymbol{\ell}'$ *that have a right sibling.*

Proof. See the full paper for the proof [1]. ◀

Partial Queries. Observe that w.l.o.g., we can assume that we first generate the dimensions 1 to t of the query, and then the dimensions $t+1$ to $d-1$ of the query, and then the value $\mathbf{y}$. A partial query is one where only the dimensions 1 to t have been generated. This is equivalent to only sampling t random points $(\mathbf{x}_i, \mathbf{z}_i)$ for $1 \le i \le t$. To be more specific, assume we have set $\mathbf{v}_i = v_i$, for $1 \le i \le t$ where each v_i is a node in T_i. Then, the partial query is equivalent to the random query $\text{Dom}_{v_1,\ldots,v_t,\mathbf{v}_{t+1},\ldots,\mathbf{v}_{d-1}}(\mathbf{q})$ and in which the first t coordinates of $\mathbf{q}$ are known (not random). Thus, we can still talk about the $\mathfrak{j}$-subproblem of a partial query; it could be that the $\mathfrak{j}$-subproblem is already known to be undefined (this happens when one of the nodes u_i, $1 \le i \le t$ is known to be undefined) but otherwise, it is defined by defining nodes $u_1, \ldots, u_t$ and the random variables $\mathbf{u}_{t+1}, \ldots, \mathbf{u}_{d-1}$; these latter random variables could later turn out to be undefined and thus rendering the $\mathfrak{j}$-subproblem of the query undefined.

After sampling a partial query, we can then talk about **eligible** sums: a sum s is eligible if it could potentially be used to answer the $\mathfrak{j}$-subproblem once the full query has been generated. Note that the emphasis is on answering the $\mathfrak{j}$-subproblem. This means, there are multiple ways for a sum to be ineligible: if $\mathfrak{j}$-subproblem is already known to be undefined then there are no eligible sums. Otherwise, the defining nodes $u_1, \cdots, u_t$ are well-defined. In this case, if it is already known that s is outside the query, or it is already known that s cannot cover any points from the $\mathfrak{j}$-subproblem then s becomes ineligible. Final and the most important case of ineligibility is when s is placed at a node w_i which is a descendant of node $u_i \in T_i$ for some $1 \le i \le t$. If this happens, even though s can be potentially used to answer the $\mathfrak{j}$-subproblem, it can do so from a different equivalent class, as the reader should remember that we only consider sums that are stored in the path that connects u_i to v_i for $1 \le i \le t$. If a sum passes all these, then it is eligible. Clearly, once the final query is generated, the set $S'_{\mathfrak{j}}$ is going to be a subset of the eligible sums.

▶ **Definition 8.** *Given a partial query* $Dom_{v_1,\ldots,v_t,\mathbf{v}_{t+1},\ldots,\mathbf{v}_{d-1}}(\mathbf{q})$*, and considering a fixed* $\mathfrak{j}$*-subproblem, we define the potential function* $\Phi_{v_1,\ldots,v_t}$ *to be the number of eligible sums.*

▶ **Lemma 9.** *We have*

$$\mathbb{E}(\Phi_{\mathbf{v}_1,\ldots,\mathbf{v}_t} \cdot \prod_{i=1}^{t} \frac{h2^{\boldsymbol{\ell}_i + j_i}}{j_i}) \le O(S^+(\mathcal{D})).$$

Proof. See the full paper for the proof [1]. ◀

By the above lemma, we except only few eligible sums for a random partial query. Let $\mathcal{BAD}_1$ be the "bad" event that the nodes $\mathbf{v}_1, \dots, \mathbf{v}_t$ are sampled to be nodes $v_1, \dots, v_t$ such that $\Phi_{v_1,\dots,v_t} \cdot \prod_{i=1}^{t} \frac{h2^{\ell_i + j_i}}{j_i} > |S^+(\mathcal{D})/\varepsilon|$. By Markov's inequality and Lemma 9, $\Pr[\mathcal{BAD}_1] = O(\varepsilon)$.

Now, fix $\mathbf{v}_1 = v_1, \dots, \mathbf{v}_t = v_t$. In the rest of the proof we will assume these values are fixed and we are going to generate the rest of the query. Next, we define another potential function.

▶ **Definition 10.** *The potential $\Psi_{w_{t+1},\dots,w_{d-1}}$ for $w_{t+1} \in T_{t+1}, \dots, w_{d-1} \in T_{d-1}$, where the depth of w_i in T_i is d_i is defined as follows. First define $\#_{x_{t+1},\dots,x_{d-1}}$ for nodes $x_i \in T_i$ to be the number of eligible sums s such that s is placed at x_i for $t+1 \le i \le d-1$. Given the nodes $w_{t+1}, \cdots, w_{d-1}$, and for non-negative integers $k_{t+1}, \dots, k_{d-1}$, we define $\#_{k_{t+1},\dots,k_{d-1}}$ as the sum of all $\#_{w'_{t+1},\dots,w'_{d-1}}$ over all nodes w'_i where w'_i has depth $d_i + k_i$ in T_i and w'_i is a descendant of w_i. We define the potential function as follows.*

$$\Psi_{w_{t+1},\dots,w_{d-1}} = \sum_{k_{t+1}=0}^{\infty} \cdots \sum_{k_{d-1}=0}^{\infty} \frac{\#_{k_{t+1},\dots,k_{d-1}}}{2^{k_{t+1}+\cdots+k_{d-1}}}.$$

▶ **Lemma 11.** *Having fixed the nodes $v_1, \dots, v_t$, we have,*

$$\mathbb{E}[\Psi_{\mathbf{u}_{t+1},\dots,\mathbf{u}_{d-1}} \cdot \prod_{i=t+1}^{d-1} (h2^{\boldsymbol{\ell}_i + j_i})] = O(\Phi_{v_1,\dots,v_t})$$

where $\boldsymbol{\ell}_i$ is the depth of $\mathbf{v}_i$, $\mathbf{u}_i$ is the defining node of the j-subproblem, the expectation is taken over the random choices of $\mathbf{v}_i$, $t+1 \le i \le d-1$ and the potential is defined to be zero if any of the nodes $\mathbf{u}_i$ is undefined.

Proof. See the full paper for the proof [1]. ◀

Now we define the second bad event $\mathcal{BAD}_2$ to be the event that $\Psi_{u_{t+1},\dots,u_{d-1}} \cdot (h2^{\ell_{t+1}+j_{t+1}}) \cdot \ldots (h2^{\ell_{d-1}+j_{d-1}}) \ge \Phi_{v_1,\dots,v_t}/\varepsilon$. By Markov's inequality and Lemma 11, $\Pr[\mathcal{BAD}_2] = O(\varepsilon)$.

Proof of the main lemma. We now prove our main lemma (Lemma 6 at page 7).

Remember that we will focus on one equivalent class $\mathbf{S}'_{\mathrm{j}}$ of $\mathbf{S}_{\mathrm{j}}$. Observe that the summation $\sum_{s \in \mathbf{S}'_{\mathrm{j}}} |e(s) \cap \mathrm{Top}(\mathrm{j}, \lambda)|$ counts how many times a point in $\mathrm{Top}(\mathrm{j}, \lambda)$ is covered by extensions of sums that cover at least C points of the $\mathrm{Top}(\mathrm{j}, \lambda)$ and this only takes into account the random choices of $\mathbf{y}$ as the nodes $v_1, \cdots, v_{d-1}$ have been fixed. As a result, $\mathbf{S}'_{\mathrm{j}}$ is a random variable that only depends on $\mathbf{y}$. To make this clear, let $\mathcal{M}_{\mathrm{j}}$ be the set that includes all the sums that can be part of $\mathbf{S}'_{\mathrm{j}}$ over all the random choices of $\mathbf{y}$. As a result, $\mathbf{S}'_{\mathrm{j}}$ is a random subset of $\mathcal{M}_{\mathrm{j}}$. Observe that every sum $s \in \mathcal{M}_{\mathrm{j}}$ has the property that it is stored in some node on the path from u_i to v_i for $1 \le i \le t$ and at the subtree of u_i for $t+1 \le i \le d-1$. Since $\mathrm{Top}(\mathrm{j}, \lambda)$ has exactly, λ points, we can label them from one to λ under some global ordering of the points (e.g., lexicographical ordering). Thus, let $f(g)$ be the x-th point in $\mathrm{Top}(\mathrm{j}, \lambda)$, $1 \le g \le \lambda$. Also, let $m(g)$ be the number of sums $s \in \mathbf{S}'_{\mathrm{j}}$ s.t., $e(s)$ contains $f(g)$. Then, we can do the following rewriting:

$$\sum_{s \in \mathbf{S}'_{\mathrm{j}}} |e(s) \cap \mathrm{Top}(\mathrm{j}, \lambda)| = \sum_{g=1}^{\lambda} m(g).$$

By linearity of expectation,

$$\mathbb{E}[\sum_{s \in \mathbf{S}'_{\mathrm{j}}} |e(s) \cap \mathrm{Top}(\mathrm{j}, \lambda)|] = \sum_{g=1}^{\lambda} \mathbb{E}[m(g)] = \sum_{s \in \mathcal{M}_{\mathrm{j}}} \sum_{g=1}^{\lambda} \Pr[s \text{ covers } f(g), s \in \mathbf{S}'_{\mathrm{j}}]. \quad (1)$$

In the rest of the proof, we bound the right hand side of Eq. 1 and note that the probability is over the choices of $\mathbf{y}$. Consider a particular outcome of our random trials in which the random variable $\mathbf{v}_i$ has been set to node v_i, for $1 \le i \le d-1$ in which none of the bad events $\mathcal{BAD}_1$ and $\mathcal{BAD}_2$ have happened. Set the parameter ε used in the definition of these bad events to $\varepsilon = \sqrt{\delta/\varepsilon_d}$. Thus, none of the bad events happen with probability at least $1 - O(\sqrt{\delta/\varepsilon_d})$, conditioned on the event that the $\mathfrak{j}$-subproblem of the query is defined. Note that can we assume the random variable $\mathbf{y}$ has not been assigned yet. This is a valid assumption since the subproblem of a query only depend on the selection of the nodes $v_1, \dots, v_{d-1}$ and not on the Y-coordinate of the query.

As $\mathcal{BAD}_1$ has not occurred, we have $\Phi_{v_1,\dots,v_t} \cdot \prod_{i=1}^{t} \frac{h2^{\ell_i+j_i}}{j_i} \le |S(\mathcal{D})/\varepsilon|$. As $\mathcal{BAD}_2$ has not occurred either, we know that $\Psi_{u_{t+1},\dots,u_{d-1}} \cdot \prod_{i=t+1}^{d-1}(h2^{\ell_i+j_i}) < \Phi_{v_1,\dots,v_t}/\varepsilon$. Thus,

$$\Psi_{v_{t+1},\dots,v_{d-1}} < \frac{\Phi_{v_1,\dots,v_t}}{\varepsilon \prod_{i=t+1}^{d-1}(h2^{\ell_i+j_i})} \le \frac{|S^+(\mathcal{D})|}{\varepsilon \prod_{i=1}^{t} \frac{h2^{\ell_i+j_i}}{j_i}} \cdot \frac{1}{\varepsilon \prod_{i=t+1}^{d-1}(h2^{\ell_i+j_i})}$$
$$= \frac{|S^+(\mathcal{D})| \prod_{i=1}^{t} j_i}{\varepsilon^2 h^{d-1} \prod_{i=1}^{d-1} 2^{j_i+\ell_i}}. \qquad (2)$$

The experiment. To bound the sum at the Eq. 1, we will use the above inequality combined with the following experiment. We select a random point $\mathbf{p}$ from $\mathrm{Top}(\mathfrak{j}, \lambda)$ by sampling an integer $\mathbf{g} \in [1, \cdots, \lambda]$ and considering $f(\mathbf{g})$. We compute the probability that $f(\mathbf{g})$ can be covered by the extension of a sum in $\mathbf{S}'_{\mathfrak{j}}$ where the probability is computed over the choices of $\mathbf{g}$ and the Y-coordinate of the query $\mathbf{y}$.

We now look at the side lengths of the box $\mathrm{Top}(\mathfrak{j}, \lambda)$. The i-th side length of λ-top box is $\frac{1}{2^{\ell_i+j_i}}$ for $1 \le i \le d-1$; this is because the $\mathfrak{j}$-subproblem was defined by nodes u_i where u_i has depth $\ell_i + j_i$. Let β be the side length of $\mathrm{Top}(\mathfrak{j}, \lambda)$ along the Y-axis. As β is chosen such that $\mathrm{Top}(\mathfrak{j}, \lambda)$ contains λ points and the pointset well distributed, the volume of λ-top box is $\Theta(\lambda/n)$. This implies, it suffices to pick $\beta = \Theta(\frac{\lambda}{n} \prod_{i=1}^{d-1} 2^{\ell_i+j_i})$. Now remember that the Y-coordinate of the top boundary of the λ-top box is y and the Y-coordinate of its lower boundary is $y - \beta$.

Consider a sum $s \in \mathcal{M}_{\mathfrak{j}}$. Now consider the smallest box enclosing $e(s)$; w.l.o.g., we use the notation $e(s)$ to refer to this box. For $t+1 \le i \le d-1$, the i-th side length of $e(s)$ is $2^{-\ell_i-j_i-\zeta_i(s)}$ because s was placed at node $w_i \in T_i$ which is below u_i and thus our extensions extends the i-dimension of the box to match that of w_i. However, for $1 \le i \le t$, the i-th side length of $e(s)$ is $2^{-\ell_i-j_i}$. We have

$$\mathrm{Vol}(e(s) \cap \mathrm{Top}(\mathfrak{j}, \lambda)) \le \beta \prod_{i=1}^{t} 2^{-\ell_i-j_i} \prod_{i=t+1}^{d-1} 2^{-\ell_i-j_i-\zeta_i(s)} = \Theta(\frac{\lambda}{n} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}). \qquad (3)$$

Observe that we have assumed s covers at least C points inside $\mathrm{Top}(\mathfrak{j}, \lambda)$. However, our point set is well-distributed which implies the number of points covered by s is at most $\frac{n}{\varepsilon_d}\mathrm{Vol}(e(s) \cap \mathrm{Top}(\mathfrak{j}, \lambda))$ which by Eq. 3 is bounded by $O(\frac{\lambda}{\varepsilon_d} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)})$. We are picking the point $f(\mathbf{g})$ randomly among the λ points inside the $\mathrm{Top}(\mathfrak{j}, \lambda)$ which implies the probability that $f(\mathbf{g})$ gets covered is at most

$$O(\frac{1}{\varepsilon_d} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}). \qquad (4)$$

Note that above inequality is only with respect to the *random choices of* $\mathbf{g}$ and ignores the probability of $s \in \mathbf{S}'_{\mathfrak{j}}$. However, the only necessary condition for a sum $s \in \mathcal{M}_{\mathfrak{j}}$ to be in $\mathbf{S}'_{\mathfrak{j}}$

is that its Y-coordinate falls within the top and bottom boundaries of $\mathrm{Top}(\mathfrak{j},\lambda)$ along the Y-axis. The probability of this event is at most β by construction. As this probability is indepdenent of choice of $\mathbf{p}$, we have

$$\Pr[s \text{ covers } f(\mathbf{g}), s \in \mathbf{S}'_{\mathfrak{j}}] = O(\frac{\beta}{\varepsilon_d}\prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}). \tag{5}$$

Now we consider the definition of the potential function Ψ to realize that we have

$$\sum_{k_{t+1}=0}^{\infty}\cdots\sum_{k_{d-1}=0}^{\infty}\frac{\#_{k_{t+1},\dots,k_{d-1}}}{2^{k_{t+1}+\cdots+k_{d-1}}} = \Psi_{v_{t+1},\dots,v_{d-1}} = \sum_{s\in S'_{\mathfrak{j}}}\prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}. \tag{6}$$

The left hand side is the definition of the potential function Ψ where as the right hand side counts exactly the same concept: a sum s placed at depth $\ell_i + j_i + \zeta_i(s)$ of T_i and at a descendant of v_i, for $t+1 \le i \le d-1$, contributes exactly $\prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}$ to the potential Ψ.

Remember that $m(\mathbf{p})$ is the number of sums that cover a random point $\mathbf{p}$ selected uniformly among the points inside $\mathrm{Top}(\mathfrak{j},\lambda)$. We have

$$\sum_{s\in\mathcal{M}_{\mathfrak{j}}} \Pr[s \text{ covers } f(\mathbf{g}), s\in \mathbf{S}'_{\mathfrak{j}}] = \sum_{s\in\mathcal{M}_{\mathfrak{j}}} O\left(\frac{\beta\prod_{i=t+1}^{d-1}2^{-\zeta_i(s)}}{\varepsilon_d}\right) = \qquad \text{(from Eq. 5)}$$

$$O\left(\frac{\beta\Psi_{v_{t+1},\dots,v_{d-1}}}{\varepsilon_d}\right) = O\left(\frac{\beta}{\varepsilon_d}\cdot\frac{|S^+(\mathcal{D})|\prod_{i=1}^{t} j_i}{\varepsilon^2 h^{d-1}\prod_{i=1}^{d-1}2^{j_i+\ell_i}}\right) = \qquad \text{(from Eq. 6 and Eq. 2)}$$

$$O\left(\frac{\frac{\lambda}{n}\prod_{i=1}^{d-1}2^{\ell_i+j_i}}{\varepsilon_d}\cdot\frac{|S^+(\mathcal{D})|\prod_{i=0}^{t} j_i}{\varepsilon^2 h^{d-1}\prod_{i=1}^{d-1}2^{j_i+\ell_i}}\right) = \qquad \text{(from definition of } \beta)$$

$$O\left(\frac{\lambda}{n\varepsilon_d}\cdot\frac{|S^+(\mathcal{D})|\prod_{i=0}^{t} j_i}{\varepsilon^2 h^{d-1}}\right) = O\left(\frac{\frac{\delta h^{d-1}}{j_1 j_2\cdots j_{d-1}}\cdot\frac{n}{S^+(\mathcal{A})}}{n\varepsilon_d}\cdot\frac{|S^+(\mathcal{D})|\prod_{i=0}^{t} j_i}{\varepsilon^2 h^{d-1}}\right) =$$

$$\text{(from the definition of } \lambda)$$

$$O\left(\frac{\delta}{\varepsilon_d\varepsilon^2}\right) < \frac{1}{2^d C'}. \qquad \text{(from simplification and picking } \delta = O(\varepsilon_d\varepsilon^2 C'^{-1}2^{-d}) \text{ small enough)}$$

Observe that $\Pr[s \text{ covers } f(\mathbf{g}), s\in\mathbf{S}'_{\mathfrak{j}}] = \frac{1}{\lambda}\sum_{g=1}^{\lambda}\Pr[s \text{ covers } f(g), s\in\mathbf{S}'_{\mathfrak{j}}]$. Now our Main Lemma follows from plugging this in Eq. 1.

3.3 The Lower Bound Proof

Our proof strategy is to use Lemma 6 to show that the query algorithm is forced to use a lot of sums that only cover a constant number of points inside the query, leading to a large query time.

▶ **Theorem 12.** *Let P be a well-distributed point set containing $\Theta(n)$ points in $\mathbb{R}^d$. Answering semigroup queries on P using $S^+(n)$ storage and with $Q(n)$ query bound requires that $S^+(n)\cdot Q(n) = \Omega(n(\log n\log\log n)^{d-1})$.*

We pick a random query according to the distribution defined in the previous subsection. By Lemma 6, every $\mathfrak{j}$-subproblem for $1 \le j_i \le h/2$, has a constant probability of being well-defined. Let $\mathcal{W}$ be the set of all the well-defined subproblems. For a $\mathfrak{j}$-subproblem, let $\lambda_{\mathfrak{j}}$ be the value λ as it is defined in Lemma 6. Observe that if a $\mathfrak{j}$-subproblem for $\mathfrak{j} = (j_1,\cdots,j_{d-1})$, is well-defined, then $\mathrm{Top}(\mathfrak{j},\lambda_{\mathfrak{j}})$ contains $\lambda_{\mathfrak{j}} = \frac{\delta h^{d-1}}{j_1 j_2\cdots j_{d-1}}\cdot\frac{n}{S^+(\mathcal{A})}$ points. However, if $\mathfrak{j}$-subproblem or $\mathrm{Top}(\mathrm{j}, \lambda_{\mathrm{j}})$ is not well-defined, then we consider $\mathrm{Top}(\mathrm{j}, \lambda_{\mathrm{j}})$ to contain 0 points. We define the top of the query, $\mathrm{Top}(q)$, to be the set of points $\cup_{\mathrm{j}=(j_1,\ldots,j_{d-1}),1\le j_1,\ldots,j_{d-1}\le h/2}\mathrm{Top}(\mathrm{j}, \lambda_{\mathrm{j}})$. As each j-subproblem and $\mathrm{Top}(\mathrm{j}, \lambda_{\mathrm{j}})$, for $j_i \le h/2$ has a constant probability of being well-defined, we have

$$\begin{aligned}
\mathbb{E}[|\mathrm{Top}(q)|] &= \sum_{\mathrm{j}=(j_1,\ldots,j_{d-1}),1\le j_1,\ldots,j_{d-1}\le \frac{h}{2}} \mathbb{E}[\mathrm{Top}(\mathrm{j}, \lambda_{\mathrm{j}})] \\
&= \Theta(1) \sum_{j_1=1}^{h/2} \cdots \sum_{j_{d-1}=1}^{h/2} \frac{\delta h^{d-1}}{j_1 j_2 \cdots j_{d-1}} \cdot \frac{n}{S^+(\mathcal{A})} \\
&= \sum_{j_1=1}^{h/2} \cdots \sum_{j_{d-2}=1}^{h/2} \frac{\delta \Theta(\log h) h^{d-1}}{j_1 j_2 \cdots j_{d-2}} \cdot \frac{n}{S^+(\mathcal{A})} \\
&= \ldots = \Theta\left(\frac{\delta \log^{d-1} h h^{d-1} n}{S^+(\mathcal{A})}\right). \qquad (7)
\end{aligned}$$

In the full paper [1], we show that we can find a subset $\mathrm{Top}(q)$ that contain at least a constant fraction its points, s.t., every sum can cover at most a constant number of points in this subset. As a result, the total number of sums required to cover the points in $\mathrm{Top}(q)$ is asymptotically the same as Eq. 7, our claimed lower bound. This would complete the proof.

4 The Upper Bounds

In the full version of our paper [1], we prove the following theorem that shows the analysis of our lower bound from the previous section is almost tight.

▶ **Theorem 13.** *For a set P of n points placed uniformly randomly inside the unit cube in $\mathbb{R}^d$, one can build a data structure that uses $O(n)$ storage such that a $(d+k)$-sided query can be answered with the expected query bound of $O(\log^{d-1} n(\log\log n)^k)$, for $1 \le k \le d-1$.*

If P is well-distributed, then the query bound can be made worst-case.

Due to lack of space, the technical parts of the proof appear in the full version only. However, the main idea is to simulate the phenomenon we have captured in our lower bound: the idea that one can store sums such that the sums from different subproblems "help" each other. To do that, we define the notion of "collectively well-distributed" point sets. Intuitively, collectively well-distributed point sets is a collection of point sets $\mathcal{P}$ where each element of $\mathcal{P}$ is a well-distributed point set but importantly, certain unions of the point sets in $\mathcal{P}$ are also well-distributed point sets. See Fig. 4 for an example. It turns out that we can turn any properly "collectively well-distributed" point set $\mathcal{P}$ that contains $O(n)$ points, into a data structure that uses $O(n)$ space (i.e., we form a constant number of sums per point in $\mathcal{P}$) and has the desired query time.

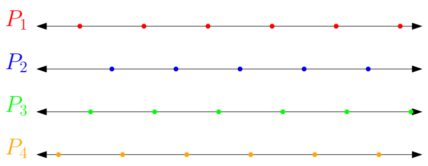

Figure 4 The point sets P_1, P_2, P_3, P_4 are well-distributed. For any continuous set of integers $I \subset \{1,\ldots,4\}$, $\cup_{i\in I} P_i$ is also well-distributed but $P_1 \cup P_3$ might not be well-distributed.

5 Conclusions

In this paper we considered the semigroup range searching problem from a lower bound point of view. We improved the best previous lower bound trade-off offered by Chazelle by analysing a well-distributed point set for $(2d-1)$-sided queries for an idempotent semigroup. Furthermore, we showed that our analysis is tight which leads us to suspect that we have found an (almost) optimal lower bound for idempotent semigroups as we believe it is unlikely that a more difficult point set exists. Thus, two prominent open problems emerge: (i) Can we improve the known data structures under the *extra* assumption that the semigroup is idempotent? (ii) Can we improve our lower bound under the *extra* assumption that the semigroup is not idempotent? Note that the effect of idempotence on other variants of range searching was studied at least once before [6].

References

1 Peyman Afshani. A New Lower Bound for Semigroup Orthogonal Range Searching, 2019. arXiv:1903.07967.

2 Peyman Afshani, Lars Arge, and Kasper Green Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *Symposium on Computational Geometry (SoCG)*, pages 323–332, 2012. doi:10.1145/2261250.2261299.

3 Peyman Afshani and Anne Drimel. On the complexity of range searching among curves. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 898–917, 2017.

4 Pankaj K. Agarwal. Range searching. In J. E. Goodman, J. O'Rourke, and C. Toth, editors, *Handbook of Discrete and Computational Geometry.* CRC Press, Inc., 2016.

5 N. Alon and B. Schieber. OPTIMAL PREPROCESSING FOR ANSWERING ON-LINE PRODUCT QUERIES. Technical Report 71/87, Tel-Aviv University, 1987.

6 Sunil Arya, Theocharis Malamatos, and David M. Mount. On the Importance of Idempotence. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 564–573, 2006.

7 Jon Louis Bentley. Decomposable searching problems. *Information Processing Letters (IPL)*, 8(5):244–251, 1979.

8 Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM (CACM)*, 23(4):214–229, 1980.

9 Bernard Chazelle. Lower bounds for orthogonal range searching: part II. The arithmetic model. *Journal of the ACM (JACM)*, 37(3):439–463, 1990.

10 Michael L. Fredman. The inherent complexity of dynamic data structures which accommodate range queries. In *Proc. 21stProceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 191–199, October 1980.

11 Michael L. Fredman. A Lower Bound on the Complexity of Orthogonal Range Queries. *Journal of the ACM (JACM)*, 28(4):696–705, 1981.

12 Michael L. Fredman. Lower Bounds on the Complexity of Some Optimal Data Structures. *SIAM Journal of Computing*, 10(1):1–10, 1981.

13 Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences (JCSS)*, 18(2):110–127, 1979.

14 Andrew C. Yao. Space-time Tradeoff for Answering Range Queries (Extended Abstract). In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, STOC '82, pages 128–136. ACM, 1982.