LEARNING-AUGMENTED RANGE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Multidimensional range search underlies spatial databases, geographic information systems, and location-based services, where queries often concentrate on a few popular regions. Standard structures such as skip quadtrees provide efficient worst-case guarantees but do not exploit this recurring demand to accelerate common queries. Workload predictions offer an opportunity to reduce search costs, provided that performance remains reliable when predictions are inaccurate or query patterns change.

We introduce LASQ, a learning-augmented skip quadtree that combines these benefits. The key idea is to predict the frequencies of *critical cubes*, the quadtree cells whose searches dominate the cost of answering a range query, and make frequently needed cubes easier to reach. For fixed dimension and approximation parameter ε, LASQ correctly answers $(1+\varepsilon)$-approximate weighted range-sum queries with expected cost $O(1 + H(f, \hat{f}))$, averaged over the query distribution. Here, $H(f, \hat{f}) = \sum_u f_u \log(1/\hat{f}_u)$ is an analogue of cross-entropy for the true and predicted critical-cube frequencies. Accurate predictions yield the corresponding entropy-like bound, while every query takes $O(\log n)$ expected time regardless of prediction accuracy. The structure uses $O(n)$ expected space and requires $O(n \log n)$ expected preprocessing time. Experiments on real and synthetic spatial data demonstrate substantial reductions in the cubes examined per query over the standard skip quadtree with accurate predictions, alongside resilience to prediction errors.

1 INTRODUCTION

Spatial databases and geographic information systems frequently need to retrieve or summarize data within a specified region, for example to locate nearby amenities or count visits within a neighborhood. On large datasets, answering every such query by scanning all stored points is expensive, making efficient spatial indexing essential. These tasks are captured by *range searching*, a fundamental problem in computational geometry and spatial data management (Agarwal & Erickson, 1999). The goal is to preprocess a set S of n points in $\mathbb{R}^d$ so that, given a query region R, one can quickly report the points of $S \cap R$ or compute an aggregate such as their count or total weight. Classical data structures for this problem include k-d trees (Bentley, 1975), quadtrees (Finkel & Bentley, 1974; Samet, 2006), and R-trees (Guttman, 1984).

Exact range searching offers little hope of polylogarithmic query time once space is limited to $O(n)$: Brönnimann et al. (1993) give a lower bound of $\Omega(n^{1-2/(d+1)})$ query time (ignoring logarithmic factors) for halfspace range queries under linear space in the arithmetic model, a bound that applies to spherical range queries as well. This is why, following Arya & Mount (2000), we consider an *approximate* version of the problem, in which points within relative distance ε of the query boundary may be counted either way. Linear-space structures then answer a query in $O(\log n + \varepsilon^{1-d})$ time (Arya & Mount, 2000; Duncan et al., 2001); among them, the *skip quadtree* of Eppstein et al. (2005) is particularly simple, layering a skip list's random sampling (Pugh, 1990) on top of a quadtree.

These worst-case guarantees do not reflect how often different regions are queried. In practice, queries on check-in or map data often concentrate around a few popular locations, suggesting that frequently accessed regions should be easier to reach. The standard skip quadtree does not exploit this information: it builds its hierarchy by promoting every point to the next level with the same probability, regardless of the query workload. For example, adapting a one-dimensional example

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of Zeynali et al. (2024), m searches for the same deep cube can require $\Theta(m \log n)$ expected time. Locating that cube once and making it directly accessible could instead reduce this cost to $O(m + \log n)$. This example illustrates the potential benefit of adapting a search structure to its workload. When the query distribution is known in advance, such benefits are well established: optimal search trees (Knuth, 1971; Mehlhorn, 1975) achieve expected search costs governed by the distribution's entropy, and analogous guarantees hold for planar point location (Iacono, 2004; Arya et al., 2007). These constructions, however, require the query distribution to be known.

A more flexible approach uses *predictions* of the workload, such as frequencies counted in an earlier query log (Fu et al., 2025) or produced by a learned estimator (Hsu et al., 2019). Algorithms with predictions (Lykouris & Vassilvitskii, 2021; Purohit et al., 2018; Mitzenmacher & Vassilvitskii, 2022) aim to be *consistent*, i.e. near-optimal when the predictions are accurate, and *robust*, i.e. never much worse than the best prediction-free algorithm, however wrong the predictions are. For one-dimensional dictionaries this has been achieved with learned priorities in search trees (Lin et al., 2022; Chen et al., 2025) and, optimally, with the `RobustSL` skip list of Zeynali et al. (2024); Fu et al. (2025) also give learning-augmented k-d trees for looking up individual points.

In this paper, we introduce `LASQ`, a learning-augmented skip quadtree that uses workload predictions to accelerate multidimensional range search while remaining robust to prediction errors. One important challenge is deciding what to predict. A range query is resolved by a small collection of *critical cubes*, which may lie at different spatial scales, and its cost is dominated by the searches that locate them. Our key idea is to predict how often each cube is critical and use these predictions to organize the skip-quadtree hierarchy. Cubes predicted to be critical more frequently are promoted to higher levels, making them easier to reach.

1.1 Contributions

We develop a prediction model tailored to range search and show how it leads to a data structure with provable consistency and robustness guarantees. Our main contributions are as follows:

- **A prediction model for range search.** We predict how often each quadtree cube is critical for a random query (§ 3.1). We prove that the total frequency over all cubes is bounded by a constant, a property that supports the design and analysis of `LASQ`.
- **A learning-augmented skip quadtree.** We introduce `LASQ`, a skip quadtree whose hierarchy is guided by predicted critical-cube frequencies (§ 3). It uses $O(n)$ expected space and can be constructed in $O(n \log n)$ expected time.
- **Consistency and robustness.** We prove that `LASQ` correctly answers every $(1+\varepsilon)$-approximate weighted range-sum query (Theorem 14). For fixed d and ε, its expected query cost, averaged over the query distribution, is $O(1 + H(f, \hat{f}))$, where $H(f, \hat{f}) = \sum_u f_u \log(1/\hat{f}_u)$ for the true and predicted critical-cube frequencies f and $\hat{f}$. Accurate predictions yield an entropy-like bound, while every query takes $O(\log n)$ expected time regardless of prediction accuracy. These guarantees extend to point-location queries and to reporting queries, with an additive cost proportional to the output size for reporting (§ 5).
- **Experiments.** We evaluate `LASQ` on real and synthetic spatial data. With accurate predictions, its queries examine 1.9–4.4× fewer cubes than the skip quadtree and 1.6–6.3× fewer than a quadtree; as the predictions are moved towards their opposite, the cost grows smoothly, and on the real datasets it stays below the skip quadtree even with fully reversed predictions (§ 6).

1.2 Related work

Approximate range searching. Arya & Mount (2000) introduced approximate range searching; their balanced box-decomposition trees (Arya et al., 1998) answer queries in $O(\log n + \varepsilon^{1-d})$ time with linear space. Balanced aspect-ratio trees (Duncan et al., 2001) and the skip quadtree (Eppstein et al., 2005) match this bound.

Distribution-sensitive search and learned indexes. Entropy-optimal point-location structures (Iacono, 2004; Arya et al., 2007) exploit a known query distribution to locate points in planar subdivisions. Biased range trees (Dujmović et al., 2012) similarly adapt to a known distribution,

supporting a restricted class of range queries in two dimensions using $O(n \log n)$ space. Instance-optimal geometric algorithms (Afshani et al., 2017) adapt to the input geometry itself. Learned multidimensional and spatial indexes (Nathan et al., 2020; Ding et al., 2020; Li et al., 2020; Qi et al., 2020) adapt their layout to the data and, in some cases, to the query workload, with an emphasis on empirical performance. Our work provides provable consistency and robustness guarantees for approximate range queries in any fixed dimension, using $O(n)$ expected space and potentially inaccurate workload predictions.

Learning-augmented search structures. Learning-augmented search trees and skip lists (Lin et al., 2022; Chen et al., 2025; Zeynali et al., 2024) provide consistency and robustness guarantees for dictionary lookups, using predicted access frequencies of individual keys. The learning-augmented k-d trees of Fu et al. (2025) extend this setting to multidimensional point lookups, but their guarantees do not cover range queries. A range query may involve several critical cubes, so these single-target guarantees do not directly yield bounds on its cost. We are not aware of earlier work that provides a learning-augmented structure for multidimensional range search with both consistency and robustness guarantees. Our construction builds on the skip-quadtree framework and the frequency-based promotion ideas of `RobustSL`. Appendix D discusses these connections and further related work in detail.

2 PRELIMINARIES

This section states the problem and recalls the compressed quadtree and the skip quadtree of Eppstein et al. (2005), on which `LASQ` is built.

2.1 PROBLEM STATEMENT

We focus on weighted range-sum queries. Given a set S of n points in $\mathbb{R}^d$, each with a nonnegative weight, we preprocess S to quickly compute the total weight of the points in a query region, up to an error allowed only near its boundary. Section 5 extends our results to range searching, where the goal is to report the points in the query region. Our structures use axis-parallel cubes obtained by repeatedly halving a root cube, with the following conventions.

Assumption 1 (Conventions)**.** The dimension $d \geq 2$ is a constant. A *cube* is an axis-parallel d-dimensional box with all sides equal, taken half-open, $\prod_{j=1}^{d}[a_j, a_j + s)$, so that the cubes of one side length used by the quadtree partition space.

We also normalize the input so that the root cube is the unit cube.

Assumption 2 (Ambient space)**.** Let Q_0 denote the quadtree on S. We take its root cube, and hence the ambient space, to be $[0, 1)^d$, so every cube of Q_0 has side length $s = 2^{-i}$ for some integer $i \geq 0$. This is without loss of generality: any bounded point set and bounded query domain can be translated and uniformly scaled to fit inside $[0, 1)^d$, without changing the combinatorial quantities bounded by the lemmas below.

We state our results for queries that are Euclidean balls; Theorem 21 extends them to convex query regions. Since exact answers are too expensive (Section 1), each ball comes with a thin annulus around it in which points may be counted either way.

Definition 3 (Query region and annulus)**.** A *query* specifies a center $v \in \mathbb{R}^d$ and a radius $w \geq 0$, defining the *query region* $R = B(v, w) = \{x \in \mathbb{R}^d : \|x - v\|_2 \leq w\}$. For the data structure's fixed approximation parameter $\varepsilon \in (0, 1)$, the *annulus* $A = B(v, (1 + \varepsilon)w) \setminus R$ is the region of thickness εw just outside R. When $w = 0$, we have $R = \{v\}$ and $A = \varnothing$; this degenerate case will be used for point location (Remark 20).

An answer must count all points of R and possibly any point of the annulus, but no point beyond it.

Definition 4 (Weighted range-sum query)**.** The input is a point set $S \subset \mathbb{R}^d$ (with $|S| = n$) together with weights $\mathrm{wt} : S \to \mathbb{R}_{\geq 0}$. Given a query (v, w) with region $R = B(v, w)$ and annulus A (Definition 3), a *correct answer* is any value total satisfying $\sum_{x \in S \cap R} \mathrm{wt}(x) \leq \mathrm{total} \leq \sum_{x \in S \cap (R \cup A)} \mathrm{wt}(x)$,

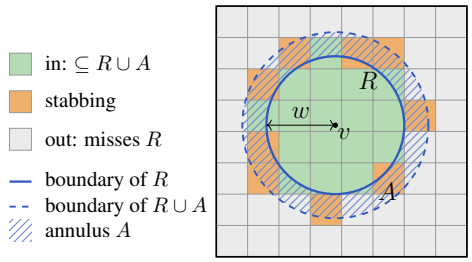


Figure 1: A query (v, w) with the cubes of one grid scale classified as in, stabbing, or out (Definition 6).

i.e. total may only be wrong about points within relative distance ε of R's boundary. A data structure *solves the* $(1+\varepsilon)$*-approximate weighted range-sum query problem* if it returns a correct answer to every such query.

2.2 Compressed quadtrees and critical cubes

Our base structure is the compressed quadtree Q_0 of Eppstein et al. (2005), built on a point set $S \subset \mathbb{R}^d$ with $|S| = n$; $|Q_0| = O(n)$.

Definition 5 (Quadrants and interesting cubes)**.** The *quadrants* of a cube p are the 2^d sub-cubes obtained by bisecting p along every coordinate axis. A cube is *interesting* if it is the root or at least two of its quadrants contain points of S.

The compressed quadtree Q_0 consists of the interesting cubes. Each stores one pointer per quadrant, leading to the next interesting cube within that quadrant, to a single point of S, or to nothing if the quadrant is empty. See Figure 10 in Appendix E for an illustration.

Definition 6 (In, out, and stabbing cubes)**.** Fix a query with region R and annulus A. Each cube $u \in Q_0$ belongs to exactly one of three types: I) *In:* $u \subseteq R \cup A$; II) *Out:* $u \cap R = \emptyset$ and $u \not\subseteq R \cup A$; III) *Stabbing:* u intersects both R and the complement of $R \cup A$. Thus, a stabbing cube reaches from inside the query region to beyond the annulus. A cube contained entirely in the annulus is classified as in. A point stored as a child in Q_0 is treated as a degenerate cube: it is in if it lies in $R \cup A$ and out otherwise, and is never stabbing. See Figure 1.

For an in cube, we include the total weight of its points; for an out cube, we include nothing. Only stabbing cubes require descending further into the quadtree. The next definition identifies where to continue this search, allowing us to skip chains of cubes with identical intersections with $R \cup A$.

Definition 7 (Critical cubes)**.** A stabbing cube u is *critical* if every stabbing child w in Q_0 satisfies $w \cap (R \cup A) \subsetneq u \cap (R \cup A)$. In particular, a stabbing cube with no stabbing children is critical.

For any stabbing cube u, let $\mathrm{crit}(u)$ be the smallest stabbing cube in Q_0 containing $u \cap (R \cup A)$. These candidate cubes form a nonempty chain, so $\mathrm{crit}(u)$ is well defined. It is a critical cube at or below u; see Lemma 28 and Figure 11.

Along the path from u to $\mathrm{crit}(u)$, the intersection with $R \cup A$ remains unchanged. At $\mathrm{crit}(u)$, each child is either in or out, requiring no further refinement, or is stabbing with a strictly smaller intersection with $R \cup A$.

2.3 The skip quadtree

The compressed quadtree has $O(n)$ nodes, but its depth can be $\Theta(n)$, for instance when the points lie at geometrically shrinking distances from one point, so walking down from the root can take linear time. The skip quadtree of Eppstein et al. (2005) fixes this the way a skip list fixes a linked list (Pugh, 1990). Starting from $S_0 := S$, each point of S_{i-1} survives into S_i independently with probability $1/2$, and level Q_i is the compressed quadtree of S_i over the same root cube. A cube with two nonempty quadrants for S_i still has them for $S_{i-1} \supseteq S_i$, so every cube of Q_i is also a cube of Q_{i-1} and can keep links to its copies one level below and above; the expected total size is $O(n)$. To locate a point x, a search starts at the root of the sparsest nonempty level, moves to the child in x's quadrant for as long as one exists, then follows the link one level down and continues, until it stops in Q_0 at the smallest cube containing x (Figure 7 in Appendix E). Each level costs $O(1)$ expected steps, so a search takes $O(\log n)$ time.

A range query keeps a set of stabbing cubes, initially the root; for each such cube u it finds $\mathrm{crit}(u)$ by the same level-by-level search (with "contains x" replaced by "is stabbing and contains $u \cap (R \cup A)$"), counts the in children of $\mathrm{crit}(u)$, drops the out ones, and continues with the stabbing ones, for a total query time of $O(\log n + \varepsilon^{1-d})$ (Eppstein et al., 2005). Since every point reaches level i with probability 2^{-i} whatever the workload, a cube that is critical for almost every query costs as much to reach as one that almost never is. LASQ instead builds its levels from predictions, so that frequently critical cubes are reached quickly.

3 The Learning-Augmented Skip Quadtree

The preceding definitions identify the critical cubes at which a query must explore the quadtree further. The main challenge is to locate these cubes efficiently. Our data structure, LASQ, uses predictions to speed up this search, with the goal of answering common queries faster while preserving worst-case guarantees when the predictions are inaccurate. This section describes the structure and its query algorithm; Section 4 proves its guarantees.

3.1 Prediction model

LASQ uses predictions of how often each square of Q_0 is critical for a query. Fix a distribution $\mathcal{Q}$ from which the queries answered by the data structure are drawn.

Definition 8 (True critical-stabbing frequency)**.** For each cube $u \in Q_0$, its *true critical-stabbing frequency* is the probability that it is critical for a query drawn from $\mathcal{Q}$: $f_u := \Pr_{(v,w)\sim\mathcal{Q}}\left[\, u \text{ is a critical stabbing square for } (v, w) \,\right]$.

Unlike a dictionary lookup, which accesses a single key, a range query can make several squares critical, possibly at different scales. Thus, the frequencies f_u need not sum to 1. Their sum is nevertheless bounded by a constant $C = C(d, \varepsilon)$ (Lemma 16), a fact used in Section 4.

LASQ does not know the true frequencies f_u. Instead, it is built using a *predicted frequency* $\hat{f}_u$ for each square. These predictions are untrusted: subject to the budget below, they may be arbitrarily inaccurate or even adversarial. The robustness guarantees hold for any such predictions, while the consistency guarantees capture the benefit of accurate predictions.

Assumption 9 (Predicted frequency budget)**.** The predicted frequencies satisfy $\hat{f}_u \in [0, 1]$ and $\sum_{u\in Q_0} \hat{f}_u \le C$, where $C = C(d, \varepsilon)$ is the constant from Lemma 16.

Assumption 10 (Oracle access)**.** The predicted critical-stabbing frequency $\hat{f}_u$ is available in $O(1)$ time per cube $u \in Q_0$, e.g. via a precomputed table.

The expected query cost of LASQ is $O(1 + H_n(f, \hat{f}))$ (Theorem 14), where $H_n(f, \hat{f}) := \sum_{u\in Q_0} f_u \min\{\log(1/\hat{f}_u), \log n\}$, and $\log(1/0) := \infty$. In particular, $H_n(f, \hat{f}) \le H(f, \hat{f}) := \sum_u f_u \log(1/\hat{f}_u)$, and $H_n(f, \hat{f}) \le C \log n$ is finite even when some predicted frequencies are zero.

Learnability of the predictions. Let $\bar{f}_u$ be the fraction of m independent sample queries $q_1, \ldots, q_m \sim \mathcal{Q}$ (independent of later queries and of the construction) for which u is critical (proof in Appendix A.11).

Theorem 11 (PAC learnability of the predictions)**.** *$\bar{f}$ always satisfies Assumption 9. Moreover, for $\gamma \in (0, 1]$ and $\eta \in (0, 1)$, if $m \ge 8\, n^{\gamma} \ln(C n^{\gamma}/\eta)$, then with probability at least $1 - \eta$ over the sample, $H_n(f, \bar{f}) \le \frac{1}{\gamma} H_n(f, f) + C \log 2$.*

So with probability $1 - \eta$, building the structure of this section from $\bar{f}$ loses only an $O(1/\gamma)$ factor against exact predictions (Corollary 47 in Appendix A.11).

3.2 Frequency classes

We organize the squares of Q_0 into classes according to their predicted frequencies. These classes guide promotion in a hierarchy, with the aim of making squares predicted to be critical more often

faster to reach. Fix two algorithm parameters $\theta, \rho \in (0,1)$. The parameter θ determines the frequency thresholds that separate the classes, while ρ controls randomized promotion alongside the guaranteed promotions determined by these classes (Section 3.3). We use n as the scale parameter, since $|Q_0| = O(n)$.

The class thresholds decrease doubly exponentially: $\theta, \theta^2, \theta^4, \ldots$. We stop at the first threshold below $n^{-1/2}$, defining $K := \min\{\, c \geq 0 : \theta^{2^c} < n^{-1/2} \,\} = \Theta(\log\log n)$, so that $n^{-1/2} \leq \theta^{2^{K-1}} < n^{-1/4}$ and $2^{K-1} = \Theta(\log n)$. A cube u with $\hat{f}_u \geq \theta$ is in *class* 0; for $1 \leq c \leq K-1$, u is in *class* c if $\theta^{2^c} \leq \hat{f}_u < \theta^{2^{c-1}}$; every other cube falls into the catch-all *class* K (Appendix A.4 gives the closed form of K and an equivalent description of the classes). The doubling keeps the number of classes at $\Theta(\log\log n)$, and it makes the sizes of the classes grow so fast that a search through classes $0, \ldots, c$ costs no more than a constant times the last one (Section 4).

3.3 Construction

We build levels by processing classes in order of increasing predicted frequency, $c = K, K-1, \ldots, 0$ (Section 3.2), so that each class's promotion starts exactly where the previous one left off. Fix $\mathcal{D}_{K+1} := 0$. For each class c, in this order, every cube $u \in Q_0$ of class c draws an independent $\mathcal{X}_u(\rho)$, a geometric random variable with parameter $1-\rho$, and is assigned a *promotion depth*

$$D_u := \begin{cases} \mathcal{X}_u(\rho) & \text{if } c = K, \\ \mathcal{D}_{c+1} + 1 + \mathcal{X}_u(\rho) & \text{if } c < K; \end{cases}$$

so the catch-all class K starts at level 0, exactly like points of an ordinary skip list, and every other class starts one level above the previous class's maximum; once every cube of class c has a depth, fix

$$\mathcal{D}_c := \max\Big(\mathcal{D}_{c+1},\ \max_{u \in \text{class } c} D_u\Big)$$

($\mathcal{D}_c := \mathcal{D}_{c+1}$ if class c is empty) before moving on to class $c-1$. Every cube of class $c < K$ therefore stops strictly above $\mathcal{D}_{c+1}$, so each class stops within its own range of levels, $(\mathcal{D}_{c+1}, \mathcal{D}_c]$ (for class K, $[0, \mathcal{D}_K]$), whose boundaries are set by the random promotion itself rather than fixed in advance (Figure 6 in Appendix E).

Let $M_i := \{\, u \in Q_0 : D_u \geq i \,\}$ be the cubes marked to persist to level i; since D_u does not depend on i, $M_i \supseteq M_{i+1}$ for every i. Every cube's depth D_u is finite, so $M_i = \emptyset$ for all sufficiently large i; LASQ only ever builds levels up to the topmost i with $M_i \neq \emptyset$, its *topmost nonempty level*.

Definition 12 (Prediction-augmented levels)**.** Q_0 is the (unmodified) base compressed quadtree. For $i > 0$, a cube $q \in Q_0$ is a node of Q_i iff q is the root, or $q \in M_i$, or q has at least two quadrants each containing a node of Q_i.

So q is kept at level i if it is marked or is a branch point between cubes kept at level i; the levels are nested (Lemma 33; see Figure 9 in Appendix E). Each node of Q_0 caches the total weight $\mathrm{W}(q) := \sum_{x \in S \cap q} \mathrm{wt}(x)$ of its subtree. Each node of Q_i ($i > 0$) stores links to its copies in Q_0 and in Q_{i-1} (Section 3.4), and to its children within Q_i:

Definition 13 (Q_i as a rooted tree)**.** For $i > 0$, $v \in Q_i$, and each quadrant of v in Q_0 whose subtree contains a node of Q_i: v's child in that quadrant is the *topmost* such node – the lowest common ancestor, in Q_0, of every node of Q_i in that subtree. This node is itself always a node of Q_i (Lemma 34), so these per-quadrant links make Q_i a rooted tree in its own right, not just a subset of Q_0's nodes.

We call the resulting structure LASQ (Learning-Augmented Skip Quadtree). Appendix A.7 describes how to build it in $O(n \log n)$ expected time.

3.4 Range Search Query Algorithm

Given a query (v, w), LASQ runs Algorithm 1, the query algorithm of the skip quadtree (Section 2.3) on LASQ's levels. If the root is not stabbing, the answer is immediate (lines 3–4). Otherwise the query keeps a set $\mathcal{A}$ of *active* stabbing cubes, starting with the root. For each one, u, it jumps to $\mathrm{crit}(u)$ (line 8), since no cube in between splits the query, and classifies the children of $\mathrm{crit}(u)$

Algorithm 1 Range query in LASQ.

```
1: procedure RangeQuery(v, w)
2:     r ← root of Q_0
3:     if r is in then return W(r)
4:     else if r is out then return 0
5:     total ← 0;  A ← {r}
6:     while A ≠ ∅ do
7:         remove a cube u from A
8:         q ← FindCritical(u)
9:         for each child u' of q in Q_0 do
10:            if u' is in then
11:                total ← total + W(u')
12:            else if u' is stabbing then
13:                add u' to A
14:    return total
```

```
1: procedure FindCritical(u)
       ▷ y is good if y is stabbing and y ⊇ u ∩ (R ∪ A)
2:     i ← topmost nonempty level
3:     x ← root of Q_i
4:     loop
5:         while x has a good child y in Q_i do
6:             x ← y
7:         if i = 0 or x has no good child in Q_0 then
8:             return x                          ▷ x = crit(u)
9:         x ← copy of x in Q_{i-1};  i ← i − 1
```

(lines 9–13): an in child contributes its cached weight in $O(1)$ time, an out child nothing, and a stabbing child becomes active.

FindCritical(u) is where the predictions pay off. Starting at the root of the topmost nonempty level, it follows child links within the current level Q_i (Definition 13) to good children (lines 5–6); we call each such move a *hop*. When the current cube x has no good child in Q_i, the search checks its children in Q_0, since Q_i may be a sparser view (line 7): if none is good, $x = \mathrm{crit}(u)$ and the search returns it; otherwise it drops to the copy of x one level down (line 9). At level 0 the two tests coincide, so the search ends in Q_0 at the latest. Frequently critical cubes sit at high levels, where they are reached after few hops (Figure 8 in Appendix E).

4 Analysis

This section states the guarantees of LASQ and outlines their proofs; the full proofs and the auxiliary lemmas they use are in Appendix A.

Theorem 14 (Main result)**.** *Given a point set $S \subset \mathbb{R}^d$ with $|S| = n$, LASQ (Section 3) can be built in $O(n \log n)$ expected time and uses $O(n)$ expected space (Lemmas 43 and 44). Queried via the algorithm of Section 3.4, it solves the $(1+\varepsilon)$-approximate weighted range-sum query problem of Definition 4 correctly on every query, with expected cost $O(1+H_n(f,\hat{f}))$ averaged over $(v,w) \sim \mathcal{Q}$, for $H_n(f,\hat{f})$ as in Section 3.1, so in particular $O(1+H(f,\hat{f}))$; and expected cost $O(\log n)$ for every individual query, regardless of prediction accuracy; the $O(\cdot)$ notation treats d and ε as constants.*

A proof sketch appears at the end of this section; the full proof appears in Appendix A.10.

Correctness. FindCritical(u) always returns $\mathrm{crit}(u)$ (Lemma 45), and the cubes whose weights are added are pairwise disjoint, lie inside $R \cup A$, and together contain every point of $S \cap R$ (Lemma 46), which is exactly what Definition 4 asks for.

Few critical cubes per query. The analysis rests on the fact that a query has only a constant number of critical cubes. This is Lemma 7 of Eppstein et al. (2005), but its proof has a gap (Appendix A.2), so we prove it from scratch.

Lemma 15 (Critical cubes per query)**.** *There is a constant $C = C(d,\varepsilon)$ such that, for every query (v,w), the number of critical stabbing cubes is at most C.*

Proof in Appendix A.3.

Stabbing cubes with side length $s \le 8w$ lie in an $O(s)$-thick shell around ∂R and satisfy $s \ge \varepsilon w/\sqrt{d}$; a volume argument therefore bounds their number by $O(\varepsilon^{1-d})$. Larger critical cubes descend from cubes whose intersection with $R \cup A$ meets several children, of which there are only $O(1)$. Averaging over the query distribution gives the frequency budget, while Assumption 9 similarly bounds the predicted mass and hence the size of each class. The proofs of the next two lemmas are deferred to Appendix A.4.

Lemma 16 (True frequency budget)**.** *For the constant $C = C(d, \varepsilon)$ of Lemma 15, $\sum_{u \in Q_0} f_u \le C$.*

Lemma 17 (Class-size bound)**.** *For every class $c \in [0, K-1]$, the number N_c of cubes in class c satisfies $N_c \le C \cdot \theta^{-2^c}$.*

Cost of one search. Unlike Lemma 16, Lemma 17 holds for whatever predictions were used, which is what makes the following bound robust. The proof of this lemma appears in Appendix A.6.

Lemma 18 (Predicted-frequency critical-cube search)**.** *Let u be a stabbing cube and $q = \mathrm{crit}(u)$, the cube* FINDCRITICAL(u) *in Algorithm 1 returns (Lemma 45). Searching for q from the root of the topmost nonempty level of* LASQ *downward takes: $O(1 + \log(1/\hat{f}_q))$ expected comparisons if q is in class $c_q < K$ (Section 3.2); $O(\log n)$ expected comparisons if q is in class K.*

The levels of LASQ split into one *layer* per class: layer c consists of the levels $\mathcal{D}_{c+1} + 1, \ldots, \mathcal{D}_c$ at which class-c cubes stop. A search for a class-c cube is over once it has passed layers $0, \ldots, c$. The width of layer c is the maximum of N_c geometric random variables, hence $O(\log N_c)$ in expectation, and by Lemma 17 the widths of layers $0, \ldots, c$ add up to $O(2^c)$, which is $O(1 + \log(1/\hat{f}_q))$ by the class thresholds. Within one level, every hop exposes a cube that reached that level but not the next, and these cubes fail independently, so each level costs $O(1)$ expected steps (Appendix A.6).

Space and construction. Each level has at most twice as many nodes as cubes marked for it, and Lemma 17 keeps the total number of marks $O(n)$ (Lemmas 44 and 43).

Proof sketch of Theorem 14. Construction time, space, and correctness are shown above. By Lemma 15, a query calls FINDCRITICAL at most $C = O(1)$ times, and classifying the children of each critical cube costs $O(2^d) = O(1)$. By Lemma 18 and the class thresholds, the search for a critical cube q costs $O(1 + \min\{\log(1/\hat{f}_q), \log n\})$ in expectation, and in particular $O(\log n)$ whatever the predictions, which gives robustness. For consistency, the query and the construction are independent sources of randomness, so the expected cost over $(v, w) \sim \mathcal{Q}$ is $\sum_u f_u \cdot O(1 + \min\{\log(1/\hat{f}_u), \log n\}) = O(1 + H_n(f, \hat{f}))$, using $\sum_u f_u \le C$ (Lemma 16). The full proof is in Appendix A.10. □

5 EXTENSIONS

Beyond aggregate weights, one often wants the points themselves, the simplest query of all is locating a point, and queries need not be balls. The same structure and query algorithm handle all three, with the same guarantees.

Theorem 19 (Reporting queries)**.** LASQ *also answers* reporting *queries: given a query (v, w), it outputs a set O of points with $S \cap R \subseteq O \subseteq S \cap (R \cup A)$. Each such query costs the bounds of Theorem 14 plus an additive $O(T)$, where $T = |O|$ is the output size; in particular, the consistency and robustness guarantees of Theorem 14 carry over, up to the $O(T)$ term.*

Proof. Run Algorithm 1, but wherever it adds the cached weight of an in cube u (lines 3 and 11), traverse the subtree of u in Q_0 and output its points instead. By Lemma 46 (and trivially if the root is in or out), the counted cubes are pairwise disjoint, lie in $R \cup A$, and cover $S \cap R$, so $S \cap R \subseteq O \subseteq S \cap (R \cup A)$ and no point is output twice. Every internal node of a subtree of Q_0 has at least two children and its leaves are the points of $S \cap u$, so the subtree has $O(|S \cap u|)$ nodes; as the counted cubes are disjoint, the traversals cost $O(T)$ in total, and the rest of the query is unchanged. □

Remark 20 (Point location)**.** Range search also covers point location. For the degenerate query $(v, 0)$, where $R = \{v\}$ and $A = \emptyset$, the only critical cube is the smallest cube of Q_0 containing v, so a weighted range-sum query with unit weights decides whether $v \in S$ at the cost of a single critical-cube search: $O(1 + \log(1/\hat{f}_q))$ for the cube q found, and $O(\log n)$ regardless of the predictions. Appendix B gives the details.

Theorem 21 (Convex query regions)**.** *Suppose that each query is a convex region $R \subset \mathbb{R}^d$ of diameter at most $2w$, with annulus $A := \{x \notin R : \mathrm{dist}(x, R) \le \varepsilon w\}$, and that whether a cube is in, out, or stabbing can be decided in $O(1)$ time. Then Theorems 14 and 19 hold unchanged, with $C = C(d, \varepsilon)$ replaced by another constant depending only on d and ε.*

Table 1: Cubes examined per query: mean over 10 runs of the mean over 10^3 test queries, $\pm$ standard deviation over runs ($\varepsilon = 0.1$).

	Gowalla	OSM	Deep forest
Quadtree	17.6 ± 0.8	13.2 ± 0.7	90.8 ± 2.7
Skip quadtree	41.0 ± 0.9	33.8 ± 1.5	27.7 ± 5.3
LASQ	$\mathbf{9.4 \pm 1.1}$	$\mathbf{8.3 \pm 0.9}$	$\mathbf{14.4 \pm 6.2}$

Proof sketch. The shape of R matters in two places: stabbing cubes of side s lie within distance $s\sqrt{d}$ of the boundary of R, a strip of volume $O(w^{d-1}s)$ for convex R (Lemma 25), and the macro-regime bound (Lemma 31) uses only that $R \cup A$ is convex with diameter below $4w$. See Appendix B.1. □

6 Experiments

We compare LASQ with the skip quadtree of Eppstein et al. (2005) and a point-region quadtree with one point per leaf, all implemented in Python, on 125,669 check-in locations of *Gowalla* (Cho et al., 2011; Leskovec & Krevl, 2014) and 27,832 *OpenStreetMap* amenities (OpenStreetMap contributors, 2026) around the Great Lakes (fixed random 10% samples of the distinct locations), and a synthetic *deep forest* of 10^4 points whose compressed quadtree has mean depth about 300. Each dataset is run 10 times. A run draws a popularity ranking of the points and, from it, 10^3 training and 10^3 test queries: small areas (a few meters wide on the real data) centered at data points drawn with Zipf-skewed popularity (exponent 1.5); for the deep forest, each run also draws a new point set. LASQ uses $\theta = 0.3$, $\rho = 0.2$ and, unless stated otherwise, predictions equal to the critical-stabbing frequencies measured on the training queries; every structure is measured on the test queries, with $\varepsilon = 0.1$. We count the cubes each query's search moves onto (not those it only tests), and report the mean over the runs of the per-run mean, $\pm$ the standard deviation over runs. Appendix C gives the full setup and further results; code and data are in the supplementary material.

As Table 1 shows, LASQ examines the fewest cubes on every dataset, 1.9–4.4× fewer than the skip quadtree and 1.6–6.3× fewer than the quadtree. In LASQ, queries at popular locations finish in a few steps (median 5.2 cubes on Gowalla, against 41.2 for the skip quadtree), while rare ones pay a full top-down search.

To vary the quality of the predictions, we move them smoothly towards their *opposite*: $\hat{f} = (1 - \beta)f + \beta f^{\text{opp}}$, where f are the training frequencies and f^{opp} gives the largest of them to the least often critical cube, the second largest to the second least, and so on, so every $\hat{f}$ has the same sum (Assumption 9). As the prediction error grows (Figure 2), the cost of LASQ grows smoothly, by at most 16% up to $\beta = 0.5$. On the real datasets it stays below the skip quadtree even with fully reversed predictions and passes the quadtree only for $\beta > 0.8$; on the deep forest it passes the skip quadtree for $\beta > 0.8$ and stays far below the quadtree. LASQ also examines the fewest cubes with every query radius multiplied by 0.1, 2, 5 or 10 (Appendix C).

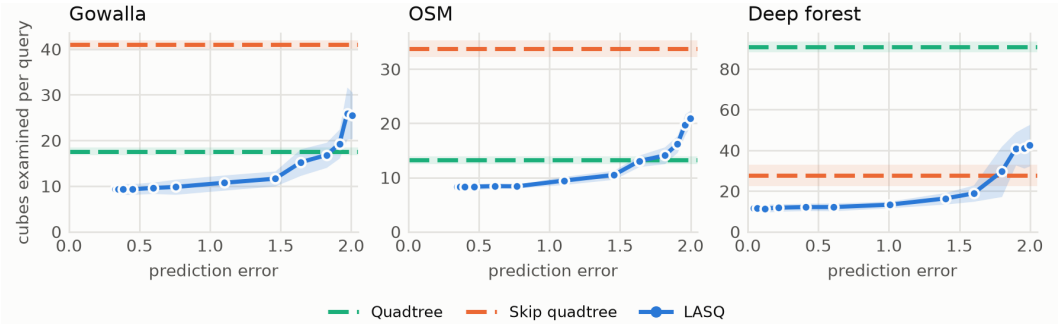


Figure 2: Cubes examined per query by LASQ as its predictions move from the training frequencies to their opposite ($\beta = 0$ to 1, left to right), against the prediction error $\|\hat{f} - f\|_1 / \|f\|_1$, with f measured on the test queries (at $\beta = 0$ the error is the gap between training and test frequencies, so the curves need not start at 0): mean over 5 runs, shaded $\pm$ standard deviation over runs. Dashed: the quadtree and the skip quadtree, which use no predictions (Table 1); $\varepsilon = 0.1$.

AI USE STATEMENT

The authors first produced a complete, polished draft of this paper and then used generative AI tools (OpenAI (2026) and Anthropic (2026)) to help refine grammar and wording throughout. Anthropic Claude also served as a coding assistant while we refactored the code that generates our empirical results and figures. No generative AI tools were involved in formulating the problem, developing theorems or proofs, designing algorithms or experiments, implementing methods, generating or cleaning data, or interpreting results. All AI-assisted output was reviewed by the authors: text edits were compared against the original draft, and the refactored code was confirmed to reproduce the original results under fixed random seeds. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Every theoretical result is presented together with its assumptions in Sections 2–5, and full proofs are given in Appendices A and B. Section 6 and Appendix C describe the experimental setup, including datasets, query workloads, parameter settings, and baselines. The supplementary material provides the implementation of every data structure, the data and query generators, and the scripts used to produce all tables and plots.

REFERENCES

Peyman Afshani, Jérémy Barbay, and Timothy M. Chan. Instance-optimal geometric algorithms. *Journal of the ACM*, 64(1):3:1–3:38, 2017.

Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pp. 1–56. American Mathematical Society, 1999.

Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc P. Renault. Online computation with untrusted advice. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, pp. 52:1–52:15, 2020.

Anthropic. Claude. `https://claude.ai`, 2026. Large language model.

Sunil Arya and David M. Mount. Approximate range searching. *Computational Geometry: Theory and Applications*, 17(3–4):135–152, 2000.

Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

Sunil Arya, Theocharis Malamatos, and David M. Mount. Space-time tradeoffs for approximate spherical range counting. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 535–544, 2005.

Sunil Arya, Theocharis Malamatos, David M. Mount, and Ka Chun Wong. Optimal expected-case planar point location. *SIAM Journal on Computing*, 37(2):584–610, 2007.

Amitabha Bagchi, Adam L. Buchsbaum, and Michael T. Goodrich. Biased skip lists. *Algorithmica*, 42(1):31–48, 2005.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Marshall Bern, David Eppstein, and Shang-Hua Teng. Parallel construction of quadtrees and quality triangulations. In *Proceedings of the 3rd Workshop on Algorithms and Data Structures (WADS)*, volume 709 of *Lecture Notes in Computer Science*, pp. 188–199. Springer-Verlag, 1993.

Prosenjit Bose, Karim Douïeb, and Stefan Langerman. Dynamic optimality for skip lists and B-trees. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1106–1114, 2008.

Hervé Brönnimann, Bernard Chazelle, and János Pach. How hard is halfspace range searching. *Discrete & Computational Geometry*, 10(2):143–155, 1993.

Bernard Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM*, 37(2):200–212, 1990.

Jingbang Chen, Xinyuan Cao, Alicia Stepin, and Li Chen. On the power of learning-augmented search trees. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7739–7762, 2025.

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1082–1090, 2011.

Jialin Ding, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. Tsunami: A learned multi-dimensional index for correlated data and skewed workloads. *Proceedings of the VLDB Endowment*, 14(2):74–86, 2020.

Vida Dujmović, John Howat, and Pat Morin. Biased range trees. *Algorithmica*, 62(1):21–37, 2012.

Christian A. Duncan, Michael T. Goodrich, and Stephen Kobourov. Balanced aspect ratio trees: Combining the advantages of k-d trees and octrees. *Journal of Algorithms*, 38(1):303–333, 2001.

David Eppstein, Michael T. Goodrich, and Jonathan Z. Sun. The skip quadtree: A simple dynamic data structure for multidimensional data. In *Proceedings of the Twenty-First Annual Symposium on Computational Geometry (SCG)*, pp. 296–305, 2005.

Paolo Ferragina and Giorgio Vinciguerra. The PGM-index: A fully-dynamic compressed learned index with provable worst-case bounds. *Proceedings of the VLDB Endowment*, 13(8):1162–1175, 2020.

Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

Chunkai Fu, Brandon G. Nguyen, Jung Hoon Seo, Ryan Zesch, and Samson Zhou. Learning-augmented search data structures. In *International Conference on Learning Representations (ICLR)*, 2025.

Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pp. 47–57, 1984.

Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations (ICLR)*, 2019.

John Iacono. Expected asymptotically optimal planar point location. *Computational Geometry: Theory and Applications*, 29(1):19–22, 2004.

Donald E. Knuth. Optimum binary search trees. *Acta Informatica*, 1(1):14–25, 1971.

Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*, pp. 489–504, 2018.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014.

Pengfei Li, Hua Lu, Qian Zheng, Long Yang, and Gang Pan. LISA: A learned index structure for spatial data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 2119–2133, 2020.

Honghao Lin, Tian Luo, and David P. Woodruff. Learning augmented binary search trees. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13431–13440, 2022.

Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM*, 68(4):24:1–24:25, 2021.

Kurt Mehlhorn. Nearly optimal binary search trees. *Acta Informatica*, 5(4):287–295, 1975.

Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022.

Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

J. Ian Munro, Thomas Papadakis, and Robert Sedgewick. Deterministic skip lists. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 367–375, 1992.

Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. Learning multi-dimensional indexes. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 985–1000, 2020.

OpenAI. ChatGPT. `https://chatgpt.com`, 2026. Large language model.

OpenStreetMap contributors. OpenStreetMap. `https://www.openstreetmap.org`, 2026. Data retrieved via the Overpass API (`https://overpass-api.de`).

William Pugh. Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Jianzhong Qi, Guanli Liu, Christian S. Jensen, and Lars Kulik. Effectively learning spatial indices. *Proceedings of the VLDB Endowment*, 13(12):2341–2354, 2020.

Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

Raimund Seidel and Cecilia R. Aragon. Randomized search trees. *Algorithmica*, 16(4/5):464–497, 1996.

Daniel D. Sleator and Robert E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32 (3):652–686, 1985.

Ali Zeynali, Shahin Kamali, and Mohammad Hajiesmaili. Robust learning-augmented dictionaries. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58470–58483, 2024.

A OMITTED PROOFS

This appendix proves the results stated in Section 4, together with the auxiliary lemmas they rely on. The proofs assume the unit cube (Assumption 2); in practice all coordinates are bounded by some known maximum M, and dividing every coordinate and every query radius by M reduces to this case without changing any proof.

By Assumption 2, every cube of Q_0 has side length $s = 2^{-i}$ for some integer i; for a real threshold that is not itself of this form, round up to the smallest valid quadtree side length above it, which only strengthens any "at least s" bound.

A.1 PACKING BOUNDS FOR THE BASE QUADTREE AND FOR STABBING CUBES

Lemma 22 (Cubes of one size are disjoint)**.** *For every valid side length s, the cubes of Q_0 of side length exactly s are pairwise disjoint.*

Proof. By Assumptions 1 and 2, the quadtree subdivision at side length s is a regular grid of pairwise disjoint half-open cubes of side s, and every cube of Q_0 of side length s is one of these cubes. Distinct cubes of Q_0 are distinct cubes, hence disjoint. □

Lemma 23 (Fixed-scale packing bound)**.** *Let $R \subset \mathbb{R}^d$ be a query region contained in an axis-aligned box of side length $2w$. Fix a valid quadtree side length $s = 2^{-i}$. Then the number of cubes of Q_0 of side length exactly s that intersect R is at most*

$$\left(\frac{2w}{s} + 2\right)^d.$$

Proof. Let B be an axis-aligned box of side length $2w$ containing R (exists by assumption). Since $R \subseteq B$, any cube of Q_0 that intersects R also intersects B, so it suffices to bound the number of side-s cubes intersecting B.

Cubes of Q_0 of side length s are pairwise disjoint (Lemma 22).

Let u be such a cube with $u \cap B \neq \emptyset$, and pick $x \in u \cap B$. Since u has side length s, every $y \in u$ satisfies $\|x - y\|_\infty \leq s$, so u is contained in B^+, the box obtained by expanding B by s on every side, i.e. B^+ has side length $2w + 2s$.

The cubes of side length s that intersect B are therefore pairwise disjoint subsets of B^+, each of volume s^d; if there are N of them,

$$N \cdot s^d \;\leq\; \mathrm{vol}(B^+) = (2w + 2s)^d,$$

which gives

$$N \;\leq\; \left(\frac{2w + 2s}{s}\right)^d = \left(\frac{2w}{s} + 2\right)^d. \qquad \square$$

Summing this fixed-scale bound over the dyadic scales $s' \in \{s, 2s, 4s, \dots\}$ that lie between s and $8w$ (rather than over *all* larger scales – see Appendix A.3) bounds the number of cubes of Q_0 of side length between s and $8w$ that intersect R:

Corollary 24 (Packing bound, sizes in $[s, 8w]$)**.** *There is a constant $c = c(d)$ such that the number of cubes of Q_0 of side length between s and $8w$ that intersect R is at most $c \cdot (w/s)^d$.*

Proof. By Lemma 23, the dyadic scale $s' = 2^k s$ (for integer $k \geq 0$ with $s' \leq 8w$) contributes at most $(2w/s' + 2)^d$ cubes. Since $s' \leq 8w$ gives $2w/s' \geq 1/4$, we have $2w/s' + 2 \leq 9 \cdot (2w/s')$, so this scale contributes at most $9^d(2w/s')^d = 9^d(2w/s)^d \cdot 2^{-kd}$ cubes. Summing the geometric series over $k = 0, 1, 2, \dots$ gives at most $9^d(2w/s)^d \cdot \frac{1}{1-2^{-d}} = O((w/s)^d)$. □

A tighter bound for stabbing cubes. The bound above is loose because it also charges for cubes that lie entirely inside R: those never need to be individually examined by a range query (a query can just count/report all of R wholesale), so a tight analysis should only pay for cubes that straddle R's boundary.

Lemma 25 (Fixed-scale stabbing bound)**.** *Fix a valid quadtree side length $0 < s \le 8w$. The number of stabbing cubes of Q_0 of side length exactly s is at most $c_d \cdot (w/s)^{d-1}$ for a constant c_d depending only on d.*

Proof. Let u be a stabbing cube of side length s; any two points of u are within $s\sqrt{d}$ of each other in the Euclidean norm.

Outer confinement. Since $u \cap R \neq \emptyset$, pick $x \in u \cap R$, so $\|x - v\|_2 \le w$. Every $y \in u$ satisfies $\|y - v\|_2 \le \|x - v\|_2 + \|x - y\|_2 \le w + s\sqrt{d}$, so $u \subseteq B(v, r_+)$ for $r_+ := w + s\sqrt{d}$.

Inner exclusion. Since $u \cap \overline{R \cup A} \neq \emptyset$, pick $y \in u$ with $\|y - v\|_2 \ge (1+\varepsilon)w$. Every $x \in u$ satisfies $\|x - v\|_2 \ge \|y - v\|_2 - \|x - y\|_2 \ge (1+\varepsilon)w - s\sqrt{d}$, so u does not intersect $B(v, r_-)$ for $r_- := \max(0, (1+\varepsilon)w - s\sqrt{d})$.

Combining the two, $u \subseteq B(v, r_+) \setminus B(v, r_-)$. A short calculation (using $s \le 8w$) gives $r_+ - r_- \le 2s\sqrt{d}$ in both cases of the max, and $r_+ \le (1 + 8\sqrt{d})w$. Writing V_d for the volume of the unit ball, and using that $x \mapsto x^{d-1}$ is increasing,

$$\begin{aligned}\operatorname{vol}\big(B(v, r_+) \setminus B(v, r_-)\big) = V_d(r_+^d - r_-^d) &= V_d \int_{r_-}^{r_+} d\, x^{d-1}\, dx \\ &\le d\, V_d\, r_+^{d-1}(r_+ - r_-) \;\le\; c'_d \cdot w^{d-1} s,\end{aligned}$$

for a constant c'_d depending only on d.

By Lemma 22, stabbing cubes of side length exactly s are pairwise disjoint, and each is contained in this shell, so if there are N of them,

$$N \cdot s^d \;\le\; c'_d \cdot w^{d-1} s, \qquad \text{i.e.} \qquad N \;\le\; c'_d \cdot \Big(\frac{w}{s}\Big)^{d-1}. \qquad \square$$

Corollary 26 (Stabbing cubes, sizes in $[s, 8w]$)**.** *There is a constant $c = c(d)$ such that the number of stabbing cubes of Q_0 of side length between s and $8w$ is at most $c \cdot (w/s)^{d-1}$.*

Proof. Sum Lemma 25 over dyadic scales $s' = 2^k s \le 8w$: scale $s' = 2^k s$ contributes at most $c_d(w/s)^{d-1} \cdot 2^{-k(d-1)}$ cubes. Since $d \ge 2$, this is a decreasing geometric series in k, summing to $c_d(w/s)^{d-1} \cdot \frac{1}{1 - 2^{-(d-1)}} = O((w/s)^{d-1})$. $\square$

This matches the exponent of Eppstein et al.'s Lemma 7 and Lemma 8 (up to the constant), for the capped range $s \le s' \le 8w$.

A.2 Lemma 7 of Eppstein et al. (2005)

Lemma 7 of Eppstein et al. (2005) states that, for a query ball of radius r, the number of critical cubes of side length at least s is $c\,(s/r)^{1-d}$ for a constant c. Its proof considers the inclusion tree of these critical cubes and claims that every non-branching node covers a part of the annulus A, not covered by any other non-branching node, of volume at least $s^{d-1}\varepsilon r/c$. The following configuration (Figure 3) shows that this claim does not hold.

Let P be a cube of Q_0 whose top-left quadrant P_1 holds a single point, whose top-right quadrant P_2 and bottom-left quadrant hold no points, and whose bottom-right quadrant Q holds two points, in two of its lower quadrants; then Q is a cube of Q_0, and the children of P are Q and the point in P_1. Place the query so that R reaches into P only around the common corner of Q and P_2 on the right side of P, and $R \cup A$ meets P only in a small neighborhood of that corner; the query region itself need not be small, and may lie mostly outside P. Then Q is stabbing, and its two children, points away from that corner, are not, so Q is critical. The cube P is stabbing; its child Q is stabbing but covers strictly less of R and of $R \cup A$, since it misses the part inside P_2; and its other child is

out. So P is critical too. In the inclusion tree of critical cubes of side length at least s, the node P has the single child Q, so it is non-branching, yet the part of A it covers beyond Q lies in the small neighborhood of the corner, so its volume can be far below $s^{d-1}\varepsilon r/c$. This is why we prove a constant bound on the number of critical cubes from scratch (Lemma 15).

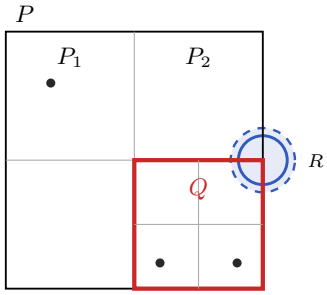


Figure 3: A counterexample to the proof of Lemma 7 of Eppstein et al. (2005). The query R (dashed: $R \cup A$) meets P only near one corner. Both P and Q (thick) are critical, and P is a non-branching node of the inclusion tree, but the part of A that P covers beyond Q is confined to that corner, so its volume is not bounded below in terms of the side length s of Q.

A.3 Removing the cap: large critical cubes

Definition 27 (Coverage-critical cube)**.** A cube $u \in Q_0$ is *coverage-critical* if $u \cap (R \cup A) \neq \emptyset$ and $u \cap (R \cup A)$ is not contained in any single child of u in Q_0. This notion is used only to count critical stabbing cubes in Appendix A.3. The two notions differ: a coverage-critical cube need not be stabbing (an *in* cube with at least two children, for instance, is coverage-critical), and for a degenerate query ($w = 0$, $v \in S$), the cube whose child is the single point v is critical but not coverage-critical. Figure 4 contrasts the notions.

Corollary 26 is still capped at $8w$. We now bound the number of critical stabbing cubes of side length greater than $8w$, using the coverage-critical cubes of Definition 27. Throughout, write $B' := R \cup A = B(v, (1+\varepsilon)w)$, a ball of diameter $2(1+\varepsilon)w < 4w$ (a single point when $w = 0$). As usual for quadtrees, two cubes of Q_0 are either nested or disjoint, and a cube of side length s has its corners at integer multiples of s (Assumption 2).

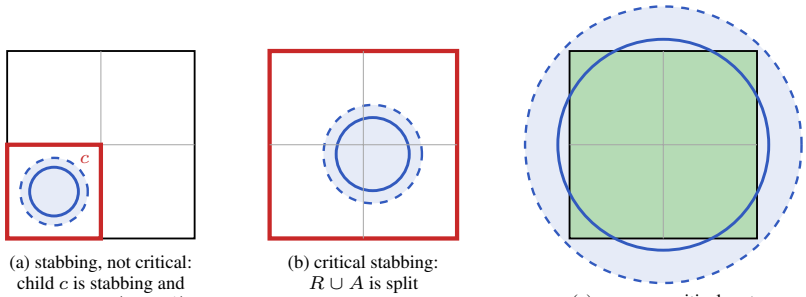
Lemma 28 (The critical cube below a stabbing cube)**.** *Let u be a stabbing cube, and let $P(u)$ be the set of stabbing cubes of Q_0 that contain $u \cap B'$. Then $P(u)$ is a path in Q_0 starting at the root, and its last (smallest) cube,* $\mathrm{crit}(u)$*, satisfies* $\mathrm{crit}(u) \subseteq u$ *and* $\mathrm{crit}(u) \cap B' = u \cap B'$*, and is critical. Moreover, a cube $x \in P(u)$ equals* $\mathrm{crit}(u)$ *if and only if no child of x in Q_0 belongs to $P(u)$.*

Proof. $P(u)$ is nonempty since $u \in P(u)$. Any two cubes of $P(u)$ both contain the nonempty set $u \cap R$, so they are nested; hence $P(u)$ is a chain. It is closed under taking ancestors: an ancestor of a cube in $P(u)$ contains $u \cap B'$ and is stabbing, since a superset of a stabbing cube still meets R and still reaches outside B'. A chain closed under ancestors is a path from the root, so its smallest cube $\mathrm{crit}(u)$ is well defined, and $x \in P(u)$ is the smallest one if and only if no child of x lies in $P(u)$ (if $x \neq \mathrm{crit}(u)$, the child of x towards $\mathrm{crit}(u)$ lies in $P(u)$).

Since $u \in P(u)$, $\mathrm{crit}(u) \subseteq u$, so $\mathrm{crit}(u) \cap B' \subseteq u \cap B' \subseteq \mathrm{crit}(u) \cap B'$, and the two are equal. Finally, let c be a stabbing child of $\mathrm{crit}(u)$. Then $c \cap B' \subseteq \mathrm{crit}(u) \cap B'$, and equality would mean $c \supseteq u \cap B'$, i.e. $c \in P(u)$, contradicting the minimality of $\mathrm{crit}(u)$. So every stabbing child covers strictly less of B', and $\mathrm{crit}(u)$ is critical (Definition 7). □

Lemma 29 (Critical cubes with a stabbing child)**.** *A critical stabbing cube with at least one stabbing child is coverage-critical.*

Proof. Let q be critical with a stabbing child c. Then $q \cap B' \supseteq c \cap R \neq \emptyset$. Suppose $q \cap B'$ were contained in a single child c' of q. If $c' = c$, then $c \cap B' = q \cap B'$, so the stabbing child c does not cover strictly less of B' than q, contradicting criticality. If $c' \neq c$, then c and c' are disjoint, but $\emptyset \neq c \cap R \subseteq q \cap B' \subseteq c'$, a contradiction. □



(a) stabbing, not critical: child c is stabbing and contains $q \cap (R \cup A)$

(b) critical stabbing: $R \cup A$ is split among several children

(c) coverage-critical, not stabbing: $q \subseteq R \cup A$ is *in*

Figure 4: Three cubes q and a query. (a) q is stabbing but not critical: its stabbing child c covers all of $q \cap (R \cup A)$, so the search continues into c. (b) q is a critical stabbing cube (Definition 7). (c) q is coverage-critical (Definition 27) but not stabbing, since it lies inside $R \cup A$.

Lemma 30 (Parent critical cube)**.** *Suppose the root of Q_0 is stabbing, and let q be a critical stabbing cube other than* crit(*root*)*. Then $q = \mathrm{crit}(z)$ for a stabbing cube $z \supseteq q$ whose parent p in Q_0 is a critical stabbing cube.*

Proof. Let z be the largest cube of Q_0 that contains q and satisfies $z \cap B' = q \cap B'$ (it exists, as q itself qualifies). Then z is stabbing, as a superset of q. We first show $q = \mathrm{crit}(z)$. Indeed $q \in P(z)$, since q is stabbing and contains $z \cap B' = q \cap B'$. A child c of q in $P(z)$ would be stabbing with $c \cap B' = q \cap B'$, contradicting that q is critical; so $q = \mathrm{crit}(z)$ by Lemma 28.

If z were the root, then $q = \mathrm{crit}(\text{root})$, which is excluded. So z has a parent p, which is stabbing as a superset of z. By the maximality of z, $p \cap B' \neq z \cap B'$, so the child z of p covers strictly less of B' than p. Any other child c of p is disjoint from z, so it misses the nonempty set $z \cap R \subseteq p \cap B'$, and also covers strictly less of B' than p. Hence p is critical. □

Lemma 31 (Macro-regime bound)**.** *There is a constant c_d depending only on d such that, for every query, the number of coverage-critical cubes of Q_0 of side length strictly greater than $8w$ is at most c_d.*

Proof. Let $\mathcal{F}$ be the set of such cubes. Since cubes of Q_0 are nested or disjoint, $\mathcal{F}$ ordered by inclusion is a forest. We bound its depth and its branching.

Depth. Let $u_1 \supsetneq u_2 \supsetneq \cdots \supsetneq u_m$ be a chain in $\mathcal{F}$; we show $m \leq d + 1$. For $j < m$, let c_j be the child of u_j in Q_0 that contains u_{j+1}. Then B' meets c_j (it meets $u_{j+1} \subseteq c_j$, by Definition 27), and, since u_j is coverage-critical, some point $x_j \in B' \cap u_j$ lies outside c_j. As c_j is an axis-parallel cube, x_j lies strictly outside one of the closed half-spaces bounding c_j; let H_j be the hyperplane bounding it. Because B' is convex and meets c_j on one side of H_j and contains x_j on the other, B' meets H_j. Because $x_j \in u_j$, the cube u_j is not contained in the half-space of H_j that contains c_j.

These hyperplanes are distinct. Suppose $H_j = H_k$ for some $j < k$. Both $c_k \subseteq u_k \subseteq u_{j+1} \subseteq c_j$ lie in the half-space of H_j that contains c_j, which is therefore also the half-space of H_k that contains c_k. But then u_k lies in that half-space, contradicting the choice of H_k.

Each H_j has the form $\{x : x_\ell = t\}$ for a coordinate ℓ and a value t that is an integer multiple of the side length of c_j. Every c_j contains the cube u_{j+1}, whose side length is greater than $8w$, so all these values t are integer multiples of s^*, the smallest valid side length greater than $8w$ (side lengths are powers of 2). Two distinct such values in the same coordinate differ by at least $s^* > 8w$, while B' spans an interval of length less than $4w$ in each coordinate (a single value if $w = 0$). So B' meets at most one of them per coordinate, and at most d of the distinct hyperplanes $H_1, \ldots, H_{m-1}$ overall. Hence $m - 1 \leq d$.

Branching. The roots of $\mathcal{F}$, as well as the children in $\mathcal{F}$ of any one cube, are pairwise disjoint cubes, each of side length at least s^* and each meeting B'. Each of them is a union of cells of the grid of side length s^*, and contains a cell that meets B'; disjoint cubes contain different cells. A set of diameter less than s^* spans an interval of length less than s^* in each coordinate and so meets at most

two cells per coordinate, hence at most 2^d cells. So there are at most 2^d roots, and each cube has at most 2^d children in $\mathcal{F}$.

A forest with at most 2^d roots, branching at most 2^d, and depth at most $d+1$ has at most $\sum_{k=1}^{d+1} 2^{dk} \leq 2^{d(d+1)+1}$ nodes; take $c_d := 2^{d(d+1)+1}$. □

Lemma 32 (Stabbing cubes are not too small)**.** *For a query with $w > 0$, every stabbing cube has side length at least $\varepsilon w/\sqrt{d}$.*

Proof. A stabbing cube u of side length s contains a point x with $\|x - v\|_2 \leq w$ and a point y with $\|y - v\|_2 \geq (1+\varepsilon)w$, so by the triangle inequality $\|x - y\|_2 \geq \varepsilon w$. Since x and y lie in a cube of side length s, $\|x - y\|_2 \leq s\sqrt{d}$. Hence $s \geq \varepsilon w/\sqrt{d}$. □

Lemma 15 (Critical cubes per query)**.** *There is a constant $C = C(d, \varepsilon)$ such that, for every query (v, w), the number of critical stabbing cubes is at most C.*

Proof. If the root of Q_0 is not stabbing, there are no stabbing cubes at all: if the root misses R, so does every cube, and if the root lies in B', so does every cube. So assume the root is stabbing, and split the critical stabbing cubes by side length.

Side length at most $8w$. If $w = 0$ there are none. Otherwise, by Lemma 32 they all have side length between $\varepsilon w/\sqrt{d}$ and $8w$, so Corollary 26 with $s = \varepsilon w/\sqrt{d}$ bounds their number by $c \cdot (\sqrt{d}/\varepsilon)^{d-1}$.

Side length greater than $8w$. Let q be such a cube, other than $\mathrm{crit}(\text{root})$. By Lemma 30, $q = \mathrm{crit}(z)$ for a stabbing child z of a critical stabbing cube p. The cube p has a stabbing child, so it is coverage-critical by Lemma 29, and its side length is greater than that of q, hence greater than $8w$. By Lemma 31 there are at most c_d choices for p; each has at most 2^d children z; and z determines $q = \mathrm{crit}(z)$. So there are at most $1 + 2^d c_d$ such cubes, counting $\mathrm{crit}(\text{root})$.

Altogether, the count is at most

$$C \;:=\; c \cdot \left(\frac{\sqrt{d}}{\varepsilon}\right)^{d-1} + 1 + 2^d c_d,$$

a constant depending only on d and ε. □

A.4 Frequency budget and class sizes

The classes in detail. The number of classes of Section 3.2 has the closed form

$$K \;:=\; \min\{\, c \geq 0 \,:\, \theta^{2^c} < n^{-1/2} \,\} \;=\; \max\left(0,\; 1 + \left\lfloor \log\left(\frac{\log n}{2\log(1/\theta)}\right)\right\rfloor\right) \;=\; \Theta(\log\log n).$$

Standard floor bounds on this closed form pin 2^{K-1} to within a factor of 2 of $\log n/(2\log(1/\theta))$ on both sides, i.e.

$$n^{-1/2} \;\leq\; \theta^{2^{K-1}} \;<\; n^{-1/4},$$

so $2^{K-1} = \Theta(\log n)$. Writing c_u for the class of a cube u,

$$\theta^{2^{c_u}} \;\leq\; \max\big(\hat{f}_u,\, n^{-1/2}\big) \;<\; \theta^{2^{c_u-1}} \qquad (c_u \geq 1),$$

with "$\leq$" in place of "$<$" on the right when $c_u = K$ and $\theta^{2^{K-1}} = n^{-1/2}$ exactly, where the $n^{-1/2}$ floor is inert (the max equals $\hat{f}_u$ itself) for every class but K: class $c \leq K-1$ already requires $\hat{f}_u \geq \theta^{2^{K-1}} \geq n^{-1/2}$.

Lemma 16 (True frequency budget)**.** *For the constant $C = C(d, \varepsilon)$ of Lemma 15, $\sum_{u \in Q_0} f_u \leq C$.*

Proof. Writing $\nVdash_u$ for the indicator that u is a critical stabbing cube for a query drawn from $\mathcal{Q}$,

$$\sum_{u \in Q_0} f_u = \sum_{u \in Q_0} \Pr[\nVdash_u = 1] = \mathbb{E}\Big[\sum_{u \in Q_0} \nVdash_u\Big]$$
$$= \mathbb{E}\big[\#\{\text{critical stabbing cubes for the query}\}\big] \le C,$$

since by Lemma 15 the count is at most C for every query. □

Lemma 17 (Class-size bound)**.** *For every class $c \in [0, K-1]$, the number N_c of cubes in class c satisfies $N_c \le C \cdot \theta^{-2^c}$.*

Proof. Every u in class $c < K$ has $\hat{f}_u \ge \theta^{2^c}$ directly (the $n^{-1/2}$ floor of Section 3.2 only clamps class K), so $N_c \cdot \theta^{2^c} \le \sum_{u \in \text{class } c} \hat{f}_u \le \sum_{u \in Q_0} \hat{f}_u \le C$ by Assumption 9. Unlike Lemma 16, this holds for whatever $\hat{f}$ was actually used to build LASQ, accurate or not – which is what lets the class-size bound, and everything downstream of it, survive adversarial predictions. □

A.5 Coherence of the prediction-augmented levels

Lemma 33 (Coherence)**.** *For every $i \ge 1$, every node of Q_i is also a node of Q_{i-1}.*

Proof. For $i = 1$ there is nothing to show, since $Q_0 = Q_0$ contains every cube. For $i \ge 2$, we show that every node q of Q_i is a node of Q_{i-1}, by induction on the height of q in Q_0 (leaves first). If q is the root, it is a node of every level. If $q \in M_i$, then $q \in M_{i-1}$ (since $M_i \subseteq M_{i-1}$), so q is a node of Q_{i-1}. Otherwise q has two quadrants each containing a node of Q_i; these nodes are proper descendants of q, hence of smaller height, so by the induction hypothesis they are nodes of Q_{i-1}. Then q has two quadrants each containing a node of Q_{i-1}, and is a node of Q_{i-1} by Definition 12. □

Lemma 34 (Q_i tree structure)**.** *Fix $i > 0$, $v \in Q_i$, and a quadrant of v in Q_0 whose subtree contains a node of Q_i. The lowest common ancestor, in Q_0, of every node of Q_i in that subtree is itself a node of Q_i.*

Proof. Let T be the (by assumption, nonempty) set of nodes of Q_i in the quadrant's subtree, and w their lowest common ancestor in Q_0. If $w \in T$, this is immediate. Otherwise, w has at least two quadrants each containing a node of T: if all of T lay within a single quadrant of w, that quadrant's child would itself be a common ancestor of T strictly lower than w, contradicting w being the *lowest* common ancestor. So w has at least two quadrants each containing a node of Q_i, hence $w \in Q_i$ by Definition 12. □

A.6 Cost of a critical-cube search

This subsection proves Lemma 18.

Partition the levels of LASQ into $K+1$ *layers* by class, following the construction of Section 3.3: layer 0 is levels $\mathcal{D}_1 + 1$ through the top, and for $c \ge 1$, layer c is levels $\mathcal{D}_{c+1} + 1$ through $\mathcal{D}_c$ (with $\mathcal{D}_{K+1} := 0$). By construction, every cube of class c has its promotion depth in $(\mathcal{D}_{c+1}, \mathcal{D}_c]$ (in $[0, \mathcal{D}_K]$ for class K, whose cubes of depth 0 appear only in Q_0) – i.e. it lies entirely within layer c and no other layer – and every cube of class $c' \le c$ reaches at least level $\mathcal{D}_{c+1} + 1$, the top of layer c (since its own promotion base $\mathcal{D}_{c'+1}$ satisfies $\mathcal{D}_{c'+1} \ge \mathcal{D}_{c+1}$). So by the time a top-down search reaches the bottom of layer c, every cube of class $\le c$ is present, and nothing from a class $> c$ is.

Lemma 35 (Layer width)**.** *For every class $c \in \{0, \dots, K\}$ with N_c cubes and every $t \ge 0$,*

$$\Pr[\mathcal{D}_c - \mathcal{D}_{c+1} \ge \log_{1/\rho} N_c + t + 1] \;\le\; \rho^t \qquad (N_c \ge 1);$$

in particular $\mathcal{D}_c - \mathcal{D}_{c+1} = O(\log(N_c + 1))$, both in expectation and with probability $1 - N_c^{-\Omega(1)}$, treating ρ as a constant.

Proof. If $N_c = 0$, $\mathcal{D}_c = \mathcal{D}_{c+1}$ and both bounds are trivial. Otherwise, writing $\{\mathcal{X}_u(\rho)\}_{u\in\text{class } c}$ for the N_c independent geometric draws of Section 3.3, $\mathcal{D}_c - \mathcal{D}_{c+1} = 1 + \max_{u\in\text{class } c} \mathcal{X}_u(\rho)$ for $c < K$, and $\mathcal{D}_K - \mathcal{D}_{K+1} = \max_{u\in\text{class } K} \mathcal{X}_u(\rho)$ for $c = K$: the (possibly $+1$-shifted) maximum of N_c i.i.d. geometric random variables – exactly the quantity governing a skip list's height. That such a maximum is $O(\log N_c)$, both in expectation and with high probability (via a union bound over the geometric tail $\Pr[\mathcal{X}_u(\rho) \geq t] = \rho^t$), is a standard fact; see Motwani & Raghavan (1995) for the general bound and Pugh (1990) for the original skip-list argument this specializes. Concretely, $\mathcal{D}_c - \mathcal{D}_{c+1} \leq 1 + \max_u \mathcal{X}_u(\rho)$ in both cases, and a union bound over the N_c draws gives $\Pr[\max_u \mathcal{X}_u(\rho) \geq \log_{1/\rho} N_c + t] \leq N_c\, \rho^{\log_{1/\rho} N_c + t} = \rho^t$. □

Lemma 36 (Doubly exponential sums)**.** *For every integer $c \geq 0$, $\sum_{c'=0}^{c} \theta^{-2^{c'}} \leq \theta^{-2^c}/(1-\theta)$.*

Proof. Dividing by θ^{-2^c}, the sum becomes $\sum_{c'=0}^{c} \theta^{2^c - 2^{c'}}$. The exponents $2^c - 2^{c'}$ for $c' = c, c-1, \ldots, 0$ are distinct nonnegative integers, so the sum is at most $\sum_{m\geq 0} \theta^m = 1/(1-\theta)$. □

Lemma 37 (Height of the top layers)**.** *For every class $0 \leq c \leq K-1$, $\mathbb{E}[\mathcal{D}_0 - \mathcal{D}_{c+1}] = O(2^c)$. Moreover, $\mathbb{E}[\mathcal{D}_0] = O(\log n)$.*

Proof. $\mathcal{D}_0 - \mathcal{D}_{c+1} = \sum_{c'=0}^{c} (\mathcal{D}_{c'} - \mathcal{D}_{c'+1})$ telescopes, so by linearity of expectation and Lemma 35, $\mathbb{E}[\mathcal{D}_0 - \mathcal{D}_{c+1}] = O\big(\sum_{c'=0}^{c} \log(N_{c'}+1)\big)$. For $c' \leq c \leq K-1$, Lemma 17 gives $N_{c'}+1 \leq 2C\,\theta^{-2^{c'}}$, so

$$\sum_{c'=0}^{c} \log(N_{c'}+1) \leq \sum_{c'=0}^{c} \big(\log(2C) + 2^{c'} \log(1/\theta)\big)$$
$$= (c+1)\log(2C) + (2^{c+1}-1)\log(1/\theta) = O(2^c).$$

For the second claim, $\mathcal{D}_0 = (\mathcal{D}_0 - \mathcal{D}_K) + (\mathcal{D}_K - \mathcal{D}_{K+1})$ since $\mathcal{D}_{K+1} = 0$. The first part has expectation $O(2^{K-1}) = O(\log n)$ by the first claim with $c = K-1$ and the definition of K (Section 3.2); the second has expectation $O(\log(N_K+1)) = O(\log n)$ by Lemma 35 and $N_K \leq |Q_0| = O(n)$. □

Lemma 38 (Cubes at a level of layer c)**.** *Fix a class c and a level i with $\mathcal{D}_{c+1} < i \leq \mathcal{D}_c$. Every cube of a class $c' < c$ has depth at least $\mathcal{D}_c + 1 \geq i + 1$, and every cube of a class $c' > c$ has depth at most $\mathcal{D}_{c+1} \leq i - 1$. In particular, the only cubes with depth exactly i are cubes of class c.*

Proof. $\mathcal{D}$ is non-increasing in the class index. A cube u of class $c' < c$ has $D_u = \mathcal{D}_{c'+1} + 1 + \mathcal{X}_u \geq \mathcal{D}_{c'+1} + 1 \geq \mathcal{D}_c + 1$. A cube of class $c' > c$ has depth at most $\mathcal{D}_{c'} \leq \mathcal{D}_{c+1}$ by the definition of $\mathcal{D}_{c'}$ as a maximum. □

One more structural fact about the levels is needed, for the construction-time and space bounds below: how a level's size relates to its marked cubes.

Lemma 39 (Level size bound)**.** *Fix $i \geq 0$ with $M_i \neq \emptyset$. Viewing Q_i as the rooted tree of Definition 13, $|Q_i| \leq 2|M_i|$.*

Proof. By Definition 12, every node of Q_i other than the root of Q_0 is either in M_i or has at least two children in this tree – a non-root node with at most one child and not in M_i is, by the same definition, simply not a node of Q_i at all (it is skipped over, the same way a non-branching cube is skipped when building Q_0 itself). Call a node of Q_i *branching* if it has at least two children, and let B and L denote the number of branching nodes and leaves (nodes with no children) of this tree, respectively.

Every leaf is in M_i. A non-root leaf is not branching, so it is in M_i by the dichotomy above. The root cannot be an unmarked leaf: if M_i contains a cube other than the root, that cube is a node of Q_i within one of the root's quadrants, so the root has a child there (Definition 13) and is not a leaf; if M_i contains nothing but the root, the root is trivially in M_i. Either way, $L \leq |M_i|$.

For any nonempty rooted tree, let n_j denote the number of nodes with exactly j children, so summing over all nodes' child-counts gives $|Q_i| = \sum_j n_j$. Separately, every edge is the parent-side

of exactly one child, so summing child-counts also counts every edge exactly once (not twice): $\sum_j j\, n_j$ equals the number of edges, which is $|Q_i| - 1$ since a tree has one fewer edge than nodes, so $\sum_j j\, n_j + 1 = \sum_j n_j$. Isolating n_0,

$$n_0 \;=\; 1 + \sum_{j\geq 2} (j-1)\, n_j \;\geq\; 1 + \sum_{j\geq 2} n_j \;=\; 1 + B,$$

i.e. $B \leq L - 1$ (writing $L = n_0$).

Combining, every node of Q_i is the root, in M_i, or branching, so

$$|Q_i| \;\leq\; 1 + |M_i| + B \;\leq\; 1 + |M_i| + (L-1) \;=\; |M_i| + L \;\leq\; |M_i| + |M_i| \;=\; 2|M_i|. \quad \square$$

Lemma 40 (Per-level hop bound)**.** *Fix a stabbing cube u, a level $i \geq 0$, and the class c whose layer contains level i, i.e. $\mathcal{D}_{c+1} < i \leq \mathcal{D}_c$ (for $i = 0$, take $c = K$). Let $\mathcal{F}_{i,c}$ denote the geometric draws of all cubes outside class c together with the set M_i. Let h_i be the number of nodes of $Q_i \cap P(u)$ strictly below the deepest node of $Q_{i+1} \cap P(u)$ (below the root if Q_{i+1} is empty); if* FINDCRITICAL(u) *descends within level i, it makes exactly h_i hops there. Then, for every $j \geq 1$,*

$$\Pr[h_i \geq j \mid \mathcal{F}_{i,c}] \;\leq\; (j+1)(1-\rho)^{j-1}, \qquad \textit{hence} \qquad \mathbb{E}[h_i \mid \mathcal{F}_{i,c}] \;\leq\; \frac{1}{\rho^2} + \frac{1}{\rho}.$$

Proof. Write $B' := R \cup A$ and $W := Q_i \cap P(u)$, where $P(u)$ is the path of Lemma 28. Since the root of Q_0 is a node of every level and $P(u)$ starts at the root, W is a path in Q_0 from the root downward.

The hops at level i are the nodes of W below the entry point. Let $x \in W$ and suppose some node of W lies below x. Let w' be the highest such node, and y the child of x in Q_i in the quadrant of x containing w' (Definition 13). Then y is the lowest common ancestor of the nodes of Q_i in that quadrant, so y is an ancestor of w' or w' itself; as $P(u)$ is closed under ancestors, $y \in W$, and hence $y = w'$. So a descent within level i that starts at a node of W hops through the following nodes of W one by one, and stops at the deepest node of W. By the same argument at level $i+1$, and since $Q_{i+1} \subseteq Q_i$ (Lemma 33), the descent at level $i+1$ stops at the deepest node of $Q_{i+1} \cap P(u)$, which is where the descent at level i starts. So a descent at level i makes exactly h_i hops. Number the nodes of W other than the root from the bottom up as $w_1, w_2, \ldots, w_L$. Then $h_i \geq j$ if and only if none of $w_1, \ldots, w_j$ is a node of Q_{i+1}.

Conditioning. Given $\mathcal{F}_{i,c}$, the set M_i is fixed, hence so are Q_i, W, and $w_1, \ldots, w_L$; only membership in M_{i+1} is still random. By Lemma 38 (and, for $i = 0$, since every cube outside class K has depth at least 1), every cube of a class $c' < c$ reaches level $i+1$, and every cube of M_i outside class c is of such a class. A class-c cube u' is in M_i iff its draw satisfies $\mathcal{X}_{u'} \geq i - \mathcal{D}_{c+1} - 1$ (with $\mathcal{D}_{K+1} = 0$ and no -1 for $c = K$), where $\mathcal{D}_{c+1}$ is fixed by the draws of the classes $> c$. The draws of class c are independent geometric random variables, so by memorylessness, given $\mathcal{F}_{i,c}$, each class-c cube of M_i reaches level $i+1$ with probability exactly ρ, independently of the others. Hence the events "$m \notin M_{i+1}$", for distinct cubes $m \in M_i$, are independent given $\mathcal{F}_{i,c}$, and each has probability at most $1 - \rho$.

Marks. Each w_k is not the root, so by Definition 12 it is in M_i or has at least two children in Q_i. If $w_k \in M_i$, let $m_k := w_k$. Otherwise w_k has a child in Q_i in a quadrant not containing w_{k-1} (for $k = 1$, in any quadrant); call that quadrant *off-path*, and let m_k be a lowest node of Q_i in it, which has no child in Q_i and hence is in M_i by Definition 12. The marks are determined by $\mathcal{F}_{i,c}$, and they are pairwise distinct: for $k' < k$, the mark $m_{k'}$ lies in the subtree of w_{k-1}, while m_k is w_k itself or lies in an off-path quadrant of w_k.

If $w_1, \ldots, w_j$ all miss Q_{i+1}, all but at most one of $m_1, \ldots, m_j$ miss M_{i+1}. If $w_k \in M_i$, then $m_k = w_k \notin Q_{i+1} \supseteq M_{i+1}$. Otherwise, suppose $m_k \in M_{i+1}$. Then the off-path quadrant of w_k contains a node of Q_{i+1}; as w_k is neither the root nor in Q_{i+1}, Definition 12 allows no other quadrant of w_k to contain a node of Q_{i+1}. In particular the subtree of w_{k-1}, which contains $m_{k'}$ for every $k' < k$, has no node of Q_{i+1}, so $m_{k'} \notin M_{i+1}$ for all $k' < k$. Hence at most one index k has $m_k \in M_{i+1}$.

Tail bound. By the last two paragraphs,

$$\begin{aligned}\Pr[h_i \geq j \mid \mathcal{F}_{i,c}] &\leq \Pr\left[\text{at least } j-1 \text{ of } m_1, \ldots, m_j \text{ miss } M_{i+1} \,\middle|\, \mathcal{F}_{i,c}\right] \\ &\leq (1-\rho)^j + j\,(1-\rho)^{j-1} \;\leq\; (j+1)(1-\rho)^{j-1},\end{aligned}$$

by a union bound over the one index that may be missing. Summing, $\mathbb{E}[h_i \mid \mathcal{F}_{i,c}] = \sum_{j\geq 1} \Pr[h_i \geq j \mid \mathcal{F}_{i,c}] \leq \sum_{m\geq 0}(m+2)(1-\rho)^m = 1/\rho^2 + 1/\rho$. □

Lemma 18 (Predicted-frequency critical-cube search)**.** *Let u be a stabbing cube and $q = \mathrm{crit}(u)$, the cube* FINDCRITICAL(u) *in Algorithm 1 returns (Lemma 45). Searching for q from the root of the topmost nonempty level of* LASQ *downward takes: $O(1+\log(1/\hat{f}_q))$ expected comparisons if q is in class $c_q < K$ (Section 3.2); $O(\log n)$ expected comparisons if q is in class K.*

Proof. By construction (Section 3.3), the cubes whose promotion depths end inside layer c are exactly the N_c cubes of class c: cubes of classes $c' < c$ pass through every level of layer c, and cubes of classes $c' > c$ stop below it. So the width $\mathcal{D}_c - \mathcal{D}_{c+1}$ of layer c depends only on the draws of class c (Lemma 35), and a top-down search for q has reached a level containing q by the time it has passed through layers $0, \ldots, c_q$ (and, if $c_q = K$, level 0), so it works only at those levels. At each level it works on, it makes h_i hops (Lemma 40), one check against Q_0, and one drop, each costing $O(1)$ comparisons. For a level $i \geq 1$ and a class $c \leq c_q$, the event $L_{i,c} := \{\mathcal{D}_{c+1} < i \leq \mathcal{D}_c\}$ is determined by $\mathcal{F}_{i,c}$: $\mathcal{D}_{c+1}$ depends only on the classes $> c$, and, given $\mathcal{D}_{c+1} < i$, $\mathcal{D}_c \geq i$ holds iff some class-c cube is in M_i. Hence, by Lemma 40,

$$\mathbb{E}\Big[\sum_{i\geq 1}\sum_{c\leq c_q} \mathbb{1}[L_{i,c}]\,(h_i+2)\Big] = \sum_{i\geq 1}\sum_{c\leq c_q} \mathbb{E}\big[\mathbb{1}[L_{i,c}]\,\mathbb{E}[h_i+2 \mid \mathcal{F}_{i,c}]\big]$$
$$= O\Big(\sum_{i\geq 1}\sum_{c\leq c_q} \Pr[L_{i,c}]\Big) = O\big(\mathbb{E}[\mathcal{D}_0 - \mathcal{D}_{c_q+1}]\big),$$

since the events $L_{i,c}$ for $c \leq c_q$ cover exactly the levels of layers $0, \ldots, c_q$. Level 0, searched only when $c_q = K$, adds $O(1)$ more in expectation by the same lemma. So the search costs $O(\mathbb{E}[\mathcal{D}_0 - \mathcal{D}_{c_q+1}])$ for $c_q < K$, and $O(1 + \mathbb{E}[\mathcal{D}_0])$ for $c_q = K$.

For $c_q = K$, Lemma 37 gives $O(\log n)$. For $c_q < K$, it gives $O(2^{c_q})$. If $c_q = 0$ this is $O(1)$. If $c_q \geq 1$, then $\hat{f}_q < \theta^{2^{c_q-1}}$ (Section 3.2), so

$$\log \hat{f}_q < 2^{c_q-1}\log\theta \quad\Longrightarrow\quad 2^{c_q} < \frac{2\log(1/\hat{f}_q)}{\log(1/\theta)} = O(\log(1/\hat{f}_q)).$$

Either way, the cost is $O(1+\log(1/\hat{f}_q))$. □

So a well-predicted critical cube (small c_q) is found in time proportional to the log of its own rarity, $O(1+\log(1/\hat{f}_q))$, while the class-K fallback guarantees $O(\log n)$ in the worst case regardless of how wrong the predictions are. The two combine into a single bound that stays finite when $\hat{f}_q = 0$; here and below, $\log(1/0) := \infty$, so that $\min\{\log(1/\hat{f}_q), \log n\} = \log n$ for such a cube.

Corollary 41 (Search cost, every class)**.** *For every $q \in Q_0$, searching for q takes $O(1 + \min\{\log(1/\hat{f}_q), \log n\})$ expected comparisons; in particular, $O(\log n)$ expected comparisons whatever $\hat{f}_q$ is.*

Proof. For $c_q < K$, Lemma 18 gives $O(1+\log(1/\hat{f}_q))$. Here $\hat{f}_q \geq \theta^{2^{c_q}} > 0$ by the definition of class $c_q < K$ (Section 3.2), so $\log(1/\hat{f}_q) \leq 2^{c_q}\log(1/\theta) \leq 2^{K-1}\log(1/\theta) = O(\log n)$ (using $c_q \leq K-1$ and $2^{K-1} = O(\log n)$, from the definition of K in Section 3.2). Hence $\log(1/\hat{f}_q) = O(\min\{\log(1/\hat{f}_q), \log n\})$, and the bound follows.

For $c_q = K$, Lemma 18 gives $O(\log n)$. By Section 3.2, class K is exactly the cubes with $\hat{f}_q < \theta^{2^{K-1}}$, and $\theta^{2^{K-1}} < n^{-1/4}$ by K's own definition. So $\hat{f}_q < n^{-1/4}$ (possibly $\hat{f}_q = 0$), hence $\log(1/\hat{f}_q) > \frac{1}{4}\log n$ and $\min\{\log(1/\hat{f}_q), \log n\} \geq \frac{1}{4}\log n$. Thus $O(\log n) = O(\min\{\log(1/\hat{f}_q), \log n\})$.

Finally, $\min\{\log(1/\hat{f}_q), \log n\} \leq \log n$ in both cases. □

A.7 CONSTRUCTION TIME

Concretely, LASQ is built in four passes: construct Q_0 by standard compressed-quadtree methods (Bern et al., 1993); sort Q_0's cubes by $\hat{f}_u$ (Assumption 10); scan the sorted order once to assign each cube's class c_u, since the class thresholds above are thresholds on $\max(\hat{f}_u, n^{-1/2})$, a monotone function of $\hat{f}_u$ for fixed n; and scan it a second time, now in class order $c = K, K-1, \ldots, 0$ (the same order the sort already presents), drawing each cube's $\mathcal{X}_u(\rho)$ and computing its depth D_u while updating the running class maximum $\mathcal{D}_c$. Each level Q_i is then materialized from Q_{i-1} by contracting cubes that are neither marked (M_i) nor branching (Definition 12).

Both the construction-time bound below and the space bound of Appendix A.8 reduce to bounding $\sum_u D_u$, the total promoted depth across all cubes; the following identity is what connects that sum to the levels' marked sets M_i.

Lemma 42 (Depth-sum identity)**.** $\sum_{i\geq 1} |M_i| = \sum_{u\in Q_0} D_u$.

Proof. Writing $\mathbb{1}[\cdot]$ for the indicator function, $u \in M_i$ iff $D_u \geq i$ (Section 3.3), so

$$\sum_{i\geq 1} |M_i| \;=\; \sum_{i\geq 1} \sum_{u\in Q_0} \mathbb{1}[D_u \geq i] \;=\; \sum_{u\in Q_0} \sum_{i\geq 1} \mathbb{1}[D_u \geq i] \;=\; \sum_{u\in Q_0} D_u,$$

swapping the order of summation – valid since both are finite sums: every D_u is finite, and $M_i = \emptyset$ once i exceeds the topmost nonempty level $\mathcal{D}_0$. □

Lemma 43 (Construction time)**.** *Given a point set $S \subset \mathbb{R}^d$ with $|S| = n$ and, under Assumption 10, oracle access to $\hat{f}_u$ for each resulting cube u, LASQ can be built in $O(n \log n)$ expected time.*

Proof. Building Q_0 takes $O(n \log n)$ time by standard methods (Bern et al., 1993). Sorting its $O(n)$ cubes by $\hat{f}_u$ takes $O(n \log n)$, and each of the two subsequent linear scans (Section 3.3: classifying, then drawing depths) costs $O(1)$ per cube, $O(n)$ total – so it remains to bound the cost of materializing the levels.

Every cube's depth satisfies $D_u \leq \mathcal{D}_0$, the topmost nonempty level, so $\sum_{u\in Q_0} D_u \leq |Q_0| \cdot \mathcal{D}_0 = O(n) \cdot \mathcal{D}_0$. Since $\mathcal{D}_0 = \sum_{c=0}^{K} (\mathcal{D}_c - \mathcal{D}_{c+1})$ telescopes over the $K + 1 = O(\log \log n)$ classes, Lemma 37 gives $\mathbb{E}[\mathcal{D}_0] = O(\log n)$. So $\mathbb{E}\big[\sum_u D_u\big] = O(n \log n)$. Building level Q_i from Q_{i-1} – contracting cubes that are neither marked (M_i) nor branching, per Definition 12 and Lemma 33's $Q_i \subseteq Q_{i-1}$ – takes $O(|Q_{i-1}|)$ time; equivalently, each cube u costs $O(1)$ work at each of its D_u levels, so materializing every level from 1 to $\mathcal{D}_0$ costs $O\big(\sum_u D_u\big) = O(n \log n)$ in expectation – consistent with summing Lemma 39's $|Q_i| \leq 2|M_i|$ over all levels and applying Lemma 42.

Summing all steps, construction takes $O(n \log n)$ expected time. □

A.8 SPACE COMPLEXITY

Lemma 44 (Space complexity)**.** *LASQ uses $O(n)$ expected space.*

Proof. LASQ's space is $|Q_0| + \sum_{i\geq 1} |Q_i|$; $|Q_0| = O(n)$ (Section 2), and summing Lemma 39 over all levels and applying Lemma 42, $\sum_{i\geq 1} |Q_i| \leq 2 \sum_{i\geq 1} |M_i| = 2 \sum_u D_u$. We bound $\mathbb{E}[\sum_u D_u]$ class by class.

Fix a level i in layer c (Appendix A.6), i.e. $\mathcal{D}_{c+1} < i \leq \mathcal{D}_c$. By Lemma 38, all $\sum_{c'<c} N_{c'}$ cubes of the classes $c' < c$ lie in M_i, and no cube of a class $c' > c$ does. So

$$|M_i| \;=\; |M_i \cap \text{class } c| \;+\; \sum_{c'<c} N_{c'} \qquad (i \text{ in layer } c).$$

Summing over the $\mathcal{D}_c - \mathcal{D}_{c+1}$ levels of layer c: the first part sums to $\sum_{u\in\text{class } c} (D_u - \mathcal{D}_{c+1}) = \sum_{u\in\text{class } c} (\mathbb{1}[c < K] + \mathcal{X}_u(\rho))$, of expectation $O(N_c)$ (treating ρ as a constant); the second part is $(\mathcal{D}_c - \mathcal{D}_{c+1}) \cdot \sum_{c'<c} N_{c'}$, and since $\sum_{c'<c} N_{c'}$ is determined by the predictions alone (independent

of the randomness in $\mathcal{D}_c - \mathcal{D}_{c+1}$), Lemma 35 gives its expectation as $O(\log(N_c+1)) \cdot \sum_{c'<c} N_{c'}$. So

$$\mathbb{E}\Big[\sum_{i \in \text{layer } c} |M_i|\Big] \;=\; O(N_c) \;+\; O(\log(N_c+1)) \cdot \sum_{c'<c} N_{c'}.$$

Summing over $c = 0, \ldots, K$, the first term totals $O(\sum_c N_c) = O(n)$, since the classes partition Q_0. It remains to bound $\sum_{c=0}^{K} O(\log(N_c+1)) \sum_{c'<c} N_{c'}$, splitting at $c = K$.

For $1 \le c \le K-1$: every $c' \le c-1$ satisfies $c' \le K-2 < K$, so Lemma 17 applies, and by Lemma 36, $\sum_{c'<c} N_{c'} \le C \sum_{c'=0}^{c-1} \theta^{-2^{c'}} = O(\theta^{-2^{c-1}})$. Combined with $\log(N_c+1) = O(2^c)$ (Lemma 17 again), each term is $O(2^c \theta^{-2^{c-1}})$; bounding $2^c \le 2^{K-1}$ and applying Lemma 36 once more, the sum over $c = 1, \ldots, K-1$ is $O\big(2^{K-1}\theta^{-2^{K-2}}\big)$.

For $c = K$: $\sum_{c'<K} N_{c'} = O(\theta^{-2^{K-1}})$ by Lemma 17 and Lemma 36, and $\log(N_K+1) = O(\log n)$ using only the trivial bound $N_K \le n$.

By Section 3.2, $n^{-1/2} \le \theta^{2^{K-1}} < n^{-1/4}$ and $2^{K-1} = O(\log n)$, so $\theta^{-2^{K-1}} \le n^{1/2}$ and $\theta^{-2^{K-2}} = \sqrt{\theta^{-2^{K-1}}} \le n^{1/4}$. The $c < K$ total is therefore $O(2^{K-1}\theta^{-2^{K-2}}) = O(n^{1/4}\log n)$, and the $c = K$ term is $O(\log n) \cdot O(\sqrt{n}) = O(\sqrt{n}\log n)$ – both $o(n)$.

Altogether, $\mathbb{E}[\sum_u D_u] = O(n) + O(n^{1/4}\log n) + O(\sqrt{n}\log n) = O(n)$, so LASQ's total expected space is $|Q_0| + 2\mathbb{E}[\sum_u D_u] = O(n)$. □

A.9 Correctness of the query algorithm

Lemma 45 (FindCritical returns $\mathrm{crit}(u)$)**.** *For every stabbing cube u,* FindCritical(u) *in Algorithm 1 returns* $\mathrm{crit}(u)$*. Throughout the call, the current cube x lies on the path $P(u)$ of Lemma 28.*

Proof. Initially x is the root of Q_0 (Definition 12 keeps the root at every level), which lies in $P(u)$ since it contains u and is stabbing as a superset of u. A hop (line 6) moves to a cube that is stabbing and contains $u \cap (R \cup A)$, i.e. to a cube of $P(u)$, and line 9 keeps the same cube. So $x \in P(u)$ throughout.

The call returns only at line 8. If the test of line 7 passed because no child of x in Q_0 lies in $P(u)$, then $x = \mathrm{crit}(u)$ by Lemma 28. If it passed because $i = 0$, the descent in Q_0 just stopped, which means exactly that no child of x in Q_0 lies in $P(u)$, and again $x = \mathrm{crit}(u)$. The call terminates: each hop moves strictly down in Q_0, each drop decreases i, and at $i = 0$ the call returns. □

Lemma 46 (Query invariant)**.** *Suppose the root of Q_0 is stabbing, and consider the loop of lines 6–13 of Algorithm 1. Call an in cube* counted *once line 11 has added its weight. Then at every iteration:*

- *(a) every cube in $\mathcal{A}$ is stabbing, and every counted cube is contained in $R \cup A$;*
- *(b) the cubes in $\mathcal{A}$ and the counted cubes are pairwise disjoint, and every point of $S \cap R$ lies in one of them.*

Moreover, the cubes $\mathrm{crit}(u)$ *computed at line 8 over the whole run are pairwise distinct critical stabbing cubes, so the loop runs at most C times (Lemma 15) and terminates.*

Proof. Initially $\mathcal{A}$ holds only the root, which is stabbing and contains all of S, and nothing is counted, so (a) and (b) hold.

Consider an iteration that removes u from $\mathcal{A}$ and computes $q = \mathrm{crit}(u)$ (Lemma 45). By Lemma 28, $q \subseteq u$ and $q \cap (R \cup A) = u \cap (R \cup A)$. The children of q in Q_0 are pairwise disjoint subsets of q, and every point of $S \cap q$ lies in exactly one of them (Definition 5). The iteration counts the in children, adds the stabbing ones to $\mathcal{A}$, and drops the out ones. (a) holds for the new cubes by their classification. For (b), the new cubes are disjoint subsets of u, and u was disjoint from every other cube in $\mathcal{A}$ and every counted cube. A point $x \in S \cap R \cap u$ lies in $u \cap (R \cup A) = q \cap (R \cup A)$,

hence in some child c of q; c meets R, so it is not out, and it is either counted or added to $\mathcal{A}$. So (b) is maintained.

For distinctness, note that every cube added to $\mathcal{A}$ is a child of $\mathrm{crit}(u)$ for some earlier u, hence contained in u. So once u is removed, every cube removed later is contained either in a child of $\mathrm{crit}(u)$ (a cube strictly smaller than $\mathrm{crit}(u)$) or in a cube that was in $\mathcal{A}$ together with u, which is disjoint from u by (b). In the first case the later $\mathrm{crit}(u') \subseteq u'$ is strictly smaller than $\mathrm{crit}(u)$, and in the second case it is disjoint from $\mathrm{crit}(u) \subseteq u$; either way the two differ. Each $\mathrm{crit}(u)$ is a critical stabbing cube (Lemma 28), and there are at most C of these (Lemma 15), so the loop runs at most C times. □

A.10 Proof of the main theorem

Theorem 14 (Main result)**.** *Given a point set $S \subset \mathbb{R}^d$ with $|S| = n$, LASQ (Section 3) can be built in $O(n \log n)$ expected time and uses $O(n)$ expected space (Lemmas 43 and 44). Queried via the algorithm of Section 3.4, it solves the $(1+\varepsilon)$-approximate weighted range-sum query problem of Definition 4 correctly on every query, with expected cost $O(1+H_n(f,\hat{f}))$ averaged over $(v,w) \sim \mathcal{Q}$, for $H_n(f,\hat{f})$ as in Section 3.1, so in particular $O(1+H(f,\hat{f}))$; and expected cost $O(\log n)$ for every individual query, regardless of prediction accuracy; the $O(\cdot)$ notation treats d and ε as constants.*

Proof. Construction time. This is Lemma 43.

Space. This is Lemma 44.

Correctness. If the root of Q_0 is in, Algorithm 1 returns $\mathrm{W}(\text{root}) = \sum_{x \in S} \mathrm{wt}(x)$ at line 3, and $S \subseteq R \cup A$, so this is a correct answer. If the root is out, $S \cap R = \emptyset$, and the answer 0 of line 4 is correct. Otherwise the root is stabbing, and by Lemma 46 the loop terminates; when it does, $\mathcal{A} = \emptyset$, so by parts (a) and (b) of that lemma the counted cubes are pairwise disjoint, contained in $R \cup A$, and together contain every point of $S \cap R$. Their total weight total therefore satisfies the two inequalities of Definition 4.

Per-query cost. Fix a query (v,w). By Lemma 46, FindCritical is called at most C times, each time returning a different critical stabbing cube q; for each one, the work of lines 9–13 beyond the search itself – examining q's children in Q_0 to classify them as in/out/stabbing – is $O(2^d) = O(1)$, since every cube of Q_0 has at most 2^d children and d is a constant. So the query's total cost is dominated by the $C = O(1)$ critical-cube searches themselves, each $O(1 + \min\{\log(1/\hat{f}_q), \log n\})$ expected comparisons by Corollary 41:

$$\mathbb{E}[\text{query cost}] \;=\; O\Big(\sum_{q \text{ critical for } (v,w)} \big(1 + \min\{\log(1/\hat{f}_q), \log n\}\big)\Big),$$

the expectation over LASQ's construction (Section 3.3). Here, as throughout, the $O(\cdot)$ notation treats d and ε as constants; since $C = O(\varepsilon^{1-d})$ (Lemma 15), in terms of ε the bound is $O(\varepsilon^{1-d} \log n)$ per query.

Consistency. Corollary 41 holds with one constant a for every stabbing cube u: whatever the query, FindCritical(u) costs at most $a(1 + \min\{\log(1/\hat{f}_q), \log n\})$ expected comparisons, where $q = \mathrm{crit}(u)$. By the per-query bound above, for every fixed query (v,w),

$$\mathbb{E}_{\text{constr}}[\text{cost}(v,w)] \;\le\; a' + a \sum_{u \in Q_0} \mathbb{1}[u \text{ is critical for } (v,w)] \,\big(1 + \min\{\log(1/\hat{f}_u), \log n\}\big)$$

for a constant a', since distinct searches return distinct critical cubes (Lemma 46). Which cubes are critical depends only on the query, and the query is drawn from $\mathcal{Q}$ independently of the construction, so taking the expectation over $(v,w) \sim \mathcal{Q}$ and using Definition 8 gives

$$\mathbb{E}_{(v,w),\,\text{constr}}[\text{cost}(v,w)] \;\le\; a' + a \sum_{u \in Q_0} f_u \big(1 + \min\{\log(1/\hat{f}_u), \log n\}\big) \;=\; O\big(1 + H_n(f,\hat{f})\big),$$

using $\sum_u f_u \le C$ (Lemma 16), where $H_n(f,\hat{f}) = \sum_{u \in Q_0} f_u \min\{\log(1/\hat{f}_u), \log n\}$ as in Theorem 14. Since the minimum is at most $\log(1/\hat{f}_u)$, this is also $O(1 + H(f,\hat{f}))$ for $H(f,\hat{f}) :=$

$\sum_{u\in Q_0} f_u \log(1/\hat{f}_u)$, which is infinite if some cube with $f_u > 0$ has $\hat{f}_u = 0$, while $H_n(f,\hat{f}) \le C\log n$ is always finite. The quantity $H(f,\hat{f})$ is analogous to the cross-entropy between f and $\hat{f}$, but it is not a cross-entropy of probability distributions, since neither f nor $\hat{f}$ needs to sum to 1. With accurate predictions, $\hat{f} = f$, it equals $\sum_u f_u \log(1/f_u)$.

Robustness. By Corollary 41, every critical cube's search costs $O(\log n)$ expected comparisons, regardless of $\hat{f}_q$; summing over the $C = O(1)$ critical cubes as in the Per-query cost paragraph, every individual query's expected cost is $O(\log n)$, whatever $\hat{f}$ was used to build LASQ – accurate, inaccurate, or adversarial. The expectation here is over LASQ's own randomized construction (Section 3.3) alone, for a single fixed choice of $\hat{f}$; no assumption on prediction quality is needed beyond Assumption 9. □

A.11 Learning the predictions

This subsection proves Theorem 11 and Corollary 47. Recall that the learned predictions are the empirical frequencies

$$\bar{f}_u \;:=\; \frac{1}{m}\,\big|\{\, j \in \{1,\dots,m\} : u \text{ is a critical stabbing cube for } q_j \,\}\big| \qquad (u \in Q_0),$$

where $q_1,\dots,q_m \sim \mathcal{Q}$ are independent, and independent of the query being answered and of LASQ's construction.

Theorem 11 (Learnability of the predictions)**.** *$\bar{f}$ always satisfies Assumption 9. Moreover, for $\gamma \in (0,1]$ and $\eta \in (0,1)$, if $m \ge 8\, n^{\gamma}\ln(Cn^{\gamma}/\eta)$, then with probability at least $1-\eta$ over the sample, $H_n(f,\bar{f}) \le \frac{1}{\gamma} H_n(f,f) + C\log 2$.*

Proof. Budget. Clearly $\bar{f}_u \in [0,1]$. Each sample query has at most C critical stabbing cubes (Lemma 15), so

$$\sum_{u\in Q_0} \bar{f}_u \;=\; \frac{1}{m}\sum_{j=1}^{m} \big|\{u : u \text{ is critical for } q_j\}\big| \;\le\; C.$$

Frequently critical cubes are estimated within a factor of 2. Let $\tau := n^{-\gamma}$ and $U := \{u \in Q_0 : f_u \ge \tau\}$. Since $\sum_u f_u \le C$ (Lemma 16), $|U| \le C/\tau$. For $u \in U$, $m\bar{f}_u$ is a sum of m independent indicators, each equal to 1 with probability f_u (Definition 8), so its mean is $\mu := mf_u \ge m\tau$, and the Chernoff bound (Motwani & Raghavan, 1995, Theorem 4.2) gives

$$\Pr\big[\bar{f}_u < f_u/2\big] \;=\; \Pr\big[m\bar{f}_u < \mu/2\big] \;\le\; e^{-\mu/8} \;\le\; e^{-m\tau/8} \;\le\; \frac{\tau\eta}{C},$$

where the last step is the assumption on m. A union bound over the at most C/τ cubes of U shows that, with probability at least $1-\eta$, $\bar{f}_u \ge f_u/2$ for every $u \in U$. Assume this event from now on.

Comparing the two sums. Write $\ell(x) := \min\{\log(1/x), \log n\}$, with $\ell(0) = \log n$, so that $H_n(f,\hat{f}) = \sum_u f_u \ell(\hat{f}_u)$. For $u \in U$, $\bar{f}_u \ge f_u/2 > 0$ gives $\log(1/\bar{f}_u) \le \log(1/f_u) + \log 2$, hence $\ell(\bar{f}_u) \le \ell(f_u) + \log 2 \le \ell(f_u)/\gamma + \log 2$. For $u \notin U$ with $f_u > 0$, we have $1/f_u > n^{\gamma}$, so $\ell(f_u) \ge \gamma \log n$, while $\ell(\bar{f}_u) \le \log n$ in any case; hence $\ell(\bar{f}_u) \le \ell(f_u)/\gamma$. Cubes with $f_u = 0$ contribute nothing to either sum. Therefore

$$H_n(f,\bar{f}) \;=\; \sum_u f_u\, \ell(\bar{f}_u) \;\le\; \frac{1}{\gamma}\sum_u f_u\, \ell(f_u) \;+\; \log 2 \sum_{u\in U} f_u \;\le\; \frac{1}{\gamma}\, H_n(f,f) \;+\; C\log 2. \quad \square$$

Corollary 47 (LASQ with learned predictions)**.** *Under the assumptions of Theorem 11, with probability at least $1-\eta$, LASQ built with $\hat{f} = \bar{f}$ has expected query cost $O\big(\frac{1}{\gamma}(1 + H_n(f,f))\big)$, and every query has expected cost $O(\log n)$.*

Proof. Condition on the sample, which fixes $\bar{f}$. By Theorem 11, $\bar{f}$ satisfies Assumption 9, and the sample is independent of the query and of the construction, so Theorem 14 applies with $\hat{f} = \bar{f}$.

It gives expected cost $O(1 + H_n(f, \bar{f}))$, which on the event of Theorem 11 is $O\big(1 + C \log 2 + \frac{1}{\gamma} H_n(f, f)\big) = O\big(\frac{1}{\gamma}(1 + H_n(f, f))\big)$, and expected cost $O(\log n)$ for every individual query, for every sample. □

For constant γ, Corollary 47 needs $m = O(n^\gamma \log(n/\eta))$ samples to lose a factor $O(1/\gamma)$. For instance, $\gamma = 1/2$ gives a factor 2 with $O(\sqrt{n} \log(n/\eta))$ samples, matching the resolution of the classes of Section 3.2, which treat all cubes with $\hat{f}_u$ below about $n^{-1/2}$ alike.

Computing $\bar{f}$. The critical cubes of a sample query can be listed by running Algorithm 1 on it, with LASQ built from any predictions (for example $\hat{f} \equiv 0$): every critical stabbing cube is returned by some call of FINDCRITICAL. Indeed, $\mathrm{crit}(\mathrm{root})$ is returned by the first call. For any other critical stabbing cube q, Lemma 30 gives $q = \mathrm{crit}(z)$ for a stabbing child z of a larger critical stabbing cube p; by induction on side length, p is returned by some call (Lemma 45), after which z is added to $\mathcal{A}$ (line 13), and the call on z returns $\mathrm{crit}(z) = q$. By Theorem 14, this takes $O(\log n)$ expected time per sample query, so $\bar{f}$ (stored as counts in a hash table over the cubes) is computed in $O(m \log n)$ expected time.

B DETAILS FOR THE EXTENSIONS

Details for Remark 20. Range search also generalizes point location. Fix $v \in \mathbb{R}^d$ and take the degenerate query $(v, 0)$ (Definition 6), so $R = \{v\}$ and $A = \emptyset$: a cube u is stabbing iff $v \in u$, since a cube containing v, having positive side length, always reaches beyond the single point v too. Among these, u is critical iff its child in v's quadrant is *not* a further cube of Q_0: if it were, that child would still contain v, hence still be stabbing and still cover all of $R \cup A = \{v\}$ – not strictly less – so u would fail Definition 7; if it is a single point or nothing instead, no child witnesses a failure, in this quadrant or (since they don't contain v) any other. So the unique critical cube is exactly Q_0's own point-location result for v: the cube reached by descending from the root along v's quadrant at each step, stopping at the first quadrant pointer that is not a further cube. A weighted range-sum query on $(v, 0)$ with unit weights therefore decides whether $v \in S$, and the reporting variant of Theorem 19 outputs v if so. Either way, the cost is that of exactly one critical-cube search, $O(1 + \min\{\log(1/\hat{f}_q), \log n\})$ for the cube q found (Corollary 41) – so a frequently looked-up point is found faster, with the same $O(\log n)$ guarantee regardless, exactly as for range search generally. □

B.1 CONVEX QUERY REGIONS

We prove Theorem 21. Throughout, R is a convex region of diameter at most $2w$ and $A = \{x \notin R : \mathrm{dist}(x, R) \le \varepsilon w\}$, so $R \cup A = R \oplus B(0, \varepsilon w)$ is convex with diameter at most $2(1 + \varepsilon)w < 4w$. We go through the results that mention the query region.

Lemma 23 only uses that R lies in an axis-aligned box of side length $2w$, which holds since R has diameter at most $2w$. *Lemma 32* only uses that a stabbing cube contains a point of R and a point at distance more than εw from R. *Lemma 31* only uses that $R \cup A$ is convex, meets every coverage-critical cube, and spans an interval of length less than $4w$ in each coordinate. Lemmas 28–30, the correctness lemmas, and the whole cost analysis do not refer to the shape of R at all. It remains to prove the analogue of Lemma 25.

Lemma 48 (Stabbing cubes of one size, convex regions)**.** *For $0 < s \le 8w$, the number of stabbing cubes of side length exactly s is at most $c'_d (w/s)^{d-1}$ for a constant c'_d depending only on d.*

Proof. Let u be a stabbing cube of side length s and $t := s\sqrt{d}$, so any two points of u are within distance t. We claim that every $z \in u$ is within distance t of the boundary ∂R. Pick $x \in u \cap R$ and $y \in u$ outside $R \cup A$. If $z \notin R$, then $\mathrm{dist}(z, R) \le \|z - x\| \le t$. If $z \in R$, the segment from z to $y \notin R$ crosses ∂R at distance at most $\|z - y\| \le t$ from z. So u lies in $N_t := \{z : \mathrm{dist}(z, \partial R) \le t\}$.

We bound the volume of N_t, writing $S(\cdot)$ for surface area and using that $S(K) \le S(L)$ for convex bodies $K \subseteq L$. The part of N_t outside R is $(R \oplus B(0, t)) \setminus R$; since the derivative of $\tau \mapsto \mathrm{vol}(R \oplus B(0, \tau))$ is $S(R \oplus B(0, \tau))$ and $R \oplus B(0, \tau)$ lies in a ball of radius $2w + \tau$, this part has

volume at most $t \cdot dV_d(2w+t)^{d-1}$, where V_d is the volume of the unit ball. The part inside R is the set of points of R within distance t of ∂R; its volume is $\int_0^t S(R_{-\tau})\,d\tau \le t\,S(R) \le t \cdot dV_d(2w)^{d-1}$, where $R_{-\tau} \subseteq R$ is the convex set of points at distance at least τ from ∂R. As $t = s\sqrt{d} \le 8\sqrt{d}\,w$, both are $O(w^{d-1}s)$. Cubes of side length s are disjoint (Lemma 22) and have volume s^d, so there are $O((w/s)^{d-1})$ of them. □

With Lemma 48 in place of Lemma 25, Corollary 26 and hence Lemma 15 hold with new constants depending only on d and ε, and the proofs of Theorems 14 and 19 go through verbatim.

C Additional experiments

This appendix gives the full experimental setup and the results that do not fit in Section 6.

Structures and measures. All structures are implemented in Python and answer the same $(1+\varepsilon)$-approximate weighted range-sum queries (Definition 4). The *quadtree* is a point-region quadtree with one point per leaf and cached subtree weights. The *skip quadtree* is the randomized structure of Eppstein et al. (2005) (each point survives to the next level with probability $1/2$), whose range query finds each critical cube by their level-by-level search (Eppstein et al., 2005, Lemma 9). `LASQ` (Section 3) uses $\theta = 0.3$ and $\rho = 0.2$ on every dataset and answers queries with Algorithm 1. For every query we count the cubes its search moves onto: for `LASQ` and the skip quadtree, the first cube of each critical-cube search and every cube it then hops, climbs or drops to; for the quadtree, the nodes it enters, that is, the nodes that are neither outside the query nor added whole from their parent. A cube that is only tested, such as a child checked by FindCritical or a level checked before dropping, is not counted, in any structure. Each dataset is run 10 times (5 for the prediction-error experiment), with independent queries, randomness of the skip quadtree and of `LASQ`, and, for the deep forest, point set. We report the mean over runs of the per-run mean over the 10^3 test queries, $\pm$ the standard deviation of these per-run means. Every answer was checked to lie between the exact weight within w and within $(1+\varepsilon)w$ of the center; there were no incorrect answers.

Datasets.

- *Gowalla*: the 6.44M check-ins of the Gowalla location network (Cho et al., 2011), from the SNAP collection (Leskovec & Krevl, 2014), worldwide; check-ins at the same location are merged into one point whose weight is their number, giving 1,256,692 weighted points, of which we use a fixed random 10%, 125,669 points.
- *OSM*: every OpenStreetMap (OpenStreetMap contributors, 2026) node tagged `amenity` in the box of latitude 41.5 to 46.5 and longitude -84 to -73 (the Great Lakes region), 278,316 distinct points, of which we use a fixed random 10%, 27,832 points.
- *Deep forest* (synthetic): a random tree of interesting cubes is drawn first and the points are then placed in it. Starting from the root, each cube holding $m > 1$ points sends a few of them (geometrically many, mean 1.5) into one random quadrant and the rest into another, after a random number of empty levels, so the tree consists of long spines with short side branches; with probability $1/100$ a cube instead splits its points evenly, starting a new spine. Each run draws a new point set of $n = 10^4$ points; for the first, Q_0 has mean depth 307 (maximum 371). Coordinates are exact integers.

Real point sets are normalized to $[0,1]^2$. On synthetic data with shallow trees, such as uniform or clustered points, Q_0 has depth $O(\log n)$: every search is short, so there is little for skip levels or predictions to shorten, and we use the deep forest instead.

Queries. Each run draws a random ranking of the points and, from it, 10^3 training and 10^3 test queries, independently. A query's center is a data point drawn with Zipf-skewed popularity (the point of rank k has probability proportional to $1/k^{1.5}$), and its radius is $\max(0, N(\mu, (\mu/2)^2))$ times the side of the data's bounding box: $\mu = 10^{-7}$ for Gowalla (about 4 m), $\mu = 10^{-6}$ for OSM (about 1 m), and $\mu = 10^{-24}$ for the deep forest. These are small areas around popular locations, with a few critical cubes per query. `LASQ` learns its predictions from the training queries only, and all measurements are on the test queries. Unless stated otherwise, $\varepsilon = 0.1$.

Search cost in detail. Table 1 shows the main result. LASQ examines the fewest cubes on all three datasets: $4.4\times$ fewer than the skip quadtree and $1.9\times$ fewer than the quadtree on Gowalla, $4.1\times$ and $1.6\times$ fewer on OSM, and $1.9\times$ and $6.3\times$ fewer on the deep forest. The typical query gains most: the median query examines 5.2 cubes on Gowalla and 4.0 on OSM, against 41.2 and 33.7 for the skip quadtree and 17.5 and 13.2 for the quadtree. The gap between LASQ's mean and median reflects this split: queries at popular locations, whose critical cubes were promoted to high levels, finish after a few steps, while queries at rarely queried locations pay close to a full top-down search. The costs of the skip quadtree and the quadtree, by contrast, are nearly the same for every query, since neither depends on the workload. On the real datasets the quadtree is shallow and enters few nodes, while the skip quadtree's search climbs to high levels before descending; on the deep forest, the long spines make the quadtree's walk expensive.

Prediction error in detail. Figure 2 gives LASQ the predictions $\hat{f} = (1-\beta)f + \beta f^{\mathrm{opp}}$ for $\beta \in \{0, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95, 0.98, 1\}$, where f are the critical-stabbing frequencies measured on the training queries. The opposite f^{opp} ranks the cubes of Q_0 from least to most often critical in training (the cubes never critical, most of Q_0, in random order) and gives the k-th largest value of f to the k-th cube of this ranking. So the cubes that were popular in training are predicted never to be critical, a random set of cubes that never were is predicted to be popular, and $\sum_u \hat{f}_u = \sum_u f_u$ for every β. The prediction error is $\|\hat{f} - f^{\mathrm{test}}\|_1 / \|f^{\mathrm{test}}\|_1$, where f^{test} are the frequencies measured on the test queries; it is at most 2, reached when the two have disjoint supports. At $\beta = 0$ it is 0.33 on Gowalla, 0.34 on OSM and 0.04 on the deep forest, the difference between the training and test queries.

The cost of LASQ grows smoothly with the error: from 9.3 to 10.7 cubes on Gowalla, 8.3 to 9.4 on OSM and 11.5 to 13.3 on the deep forest between $\beta = 0$ and $\beta = 0.5$, then faster as the popular cubes lose their high levels to the wrongly promoted ones. At $\beta = 1$, LASQ examines 25.5 cubes on Gowalla and 20.9 on OSM, above the quadtree (17.6 and 13.2) but well below the skip quadtree (41.0 and 33.8), and 42.5 on the deep forest, $1.5\times$ the skip quadtree (27.7) but about half the quadtree (90.8).

Query radius. Tables 2 to 5 repeat Table 1 with the radius of every query (training and test) multiplied by 0.1, 2, 5 and 10; LASQ learns its predictions from the training queries of each radius. LASQ examines the fewest cubes at every radius. With radii $10\times$ smaller, a query has fewer critical cubes and LASQ's advantage grows slightly: $4.8\times$ fewer cubes than the skip quadtree on Gowalla and $4.2\times$ on OSM. As the radius grows, its margin over the quadtree on the real datasets shrinks, from 17.6 against 9.4 at the base radius to 18.2 against 15.4 at $10\times$ on Gowalla, and from 13.2 against 8.3 to 13.4 against 10.7 on OSM: a larger query has more critical cubes, each found by a separate search from the top level (Algorithm 1) and each individually less frequent, so predicted to a lower class.

Table 2: Cubes examined per query with every query radius multiplied by 0.1: mean over 10 runs of the mean over 10^3 test queries, $\pm$ standard deviation over runs ($\varepsilon = 0.1$).

	Gowalla	OSM	Deep forest
Quadtree	17.5 ± 0.7	13.2 ± 0.7	94.2 ± 2.3
Skip quadtree	40.9 ± 0.8	33.7 ± 1.5	27.9 ± 6.6
LASQ	$\mathbf{8.6 \pm 0.9}$	$\mathbf{8.1 \pm 0.8}$	$\mathbf{13.4 \pm 5.2}$

Table 3: Cubes examined per query with every query radius multiplied by 2: mean over 10 runs of the mean over 10^3 test queries, $\pm$ standard deviation over runs ($\varepsilon = 0.1$).

	Gowalla	OSM	Deep forest
Quadtree	17.7 ± 0.9	13.2 ± 0.8	89.9 ± 2.9
Skip quadtree	41.1 ± 0.9	33.8 ± 1.5	27.6 ± 5.7
LASQ	$\mathbf{10.0 \pm 1.2}$	$\mathbf{8.5 \pm 0.8}$	$\mathbf{14.5 \pm 7.1}$

Table 4: Cubes examined per query with every query radius multiplied by 5: mean over 10 runs of the mean over 10^3 test queries, $\pm$ standard deviation over runs ($\varepsilon = 0.1$).

	Gowalla	OSM	Deep forest
Quadtree	17.9 ± 1.0	13.3 ± 0.8	88.8 ± 3.1
Skip quadtree	41.3 ± 0.8	33.8 ± 1.5	28.1 ± 6.5
LASQ	$\mathbf{12.3 \pm 1.9}$	$\mathbf{9.4 \pm 0.9}$	$\mathbf{13.1 \pm 6.0}$

Table 5: Cubes examined per query with every query radius multiplied by 10: mean over 10 runs of the mean over 10^3 test queries, $\pm$ standard deviation over runs ($\varepsilon = 0.1$).

	Gowalla	OSM	Deep forest
Quadtree	18.2 ± 1.0	13.4 ± 0.9	87.8 ± 3.1
Skip quadtree	41.6 ± 0.9	34.1 ± 1.4	28.6 ± 6.9
LASQ	$\mathbf{15.4 \pm 3.1}$	$\mathbf{10.7 \pm 1.2}$	$\mathbf{14.3 \pm 6.9}$

Effect of ε. Figure 5 varies ε. A correct answer may count points in the annulus A (Definition 4), so its relative error $(\text{total} - W_R)/W_R$, where W_R is the exact weight within distance w of the center, is nonnegative; ε bounds the distance of the counted points, not this error. The observed mean error grows with ε but stays small: at $\varepsilon = 0.1$ it is 0.02% on Gowalla (99.9% of the answers are exact), 0.04% on OSM (99.96% exact), and $1 \cdot 10^{-4}$ on the deep forest (91% exact); even at $\varepsilon = 1$ it is at most 0.5%. The error comes only from the few queries whose annulus contains a point.

D Extended related work

Range searching. Range searching has been studied extensively; see Agarwal & Erickson (1999) for a survey and Samet (2006) for multidimensional data structures in general. Space-partitioning trees such as k-d trees (Bentley, 1975), quadtrees and compressed quadtrees (Finkel & Bentley, 1974; Bern et al., 1993), and R-trees (Guttman, 1984) are the structures most used in practice. Strong lower bounds limit exact range searching: on a pointer machine, reporting orthogonal ranges in polylogarithmic time (plus the output size) requires $\Omega(n(\log n/\log\log n)^{d-1})$ space (Chazelle, 1990), and linear-space structures for halfspace and spherical ranges need polynomial query time (Brönnimann et al., 1993).

Approximate range searching. Arya & Mount (2000) introduced $(1+\varepsilon)$-approximate range searching and answered queries in $O(\log n + \varepsilon^{1-d})$ time with linear space using balanced box-decomposition (BBD) trees (Arya et al., 1998); balanced aspect-ratio (BAR) trees achieve the same bound with convex cells (Duncan et al., 2001), and Arya et al. (2005) study the trade-off between space and query time for approximate spherical range counting. The skip quadtree of Eppstein et al. (2005) matches these bounds while supporting insertions and deletions in $O(\log n)$ time, and is the prediction-free structure that LASQ builds on (Section 2.3).

Distribution-sensitive search. When the access distribution is known, optimal binary search trees match its entropy (Knuth, 1971; Mehlhorn, 1975), and entropy is also a lower bound for comparison-based search (Mehlhorn, 1975); splay trees achieve this without knowing the distribution (Sleator & Tarjan, 1985). Skip lists (Pugh, 1990), their deterministic variants (Munro et al., 1992), and treaps (Seidel & Aragon, 1996) balance by random or regular sampling; biased skip lists (Bagchi et al., 2005) give items with known weights search times logarithmic in their inverse weight, and within a class of weakly balanced skip lists, a self-adjusting skip list attains the working-set bound, which is optimal for that class (Bose et al., 2008). In geometry, entropy-optimal planar point location for a known query distribution (Iacono, 2004; Arya et al., 2007) and biased range trees for two-sided orthogonal range counting (Dujmović et al., 2012) pursue similar goals, while instance-optimal geometric algorithms (Afshani et al., 2017) adapt to the input instance rather than to a query distribution. Unlike this line of work, we do not assume the query distribution is known: LASQ only receives predictions of it, which may be arbitrarily wrong.

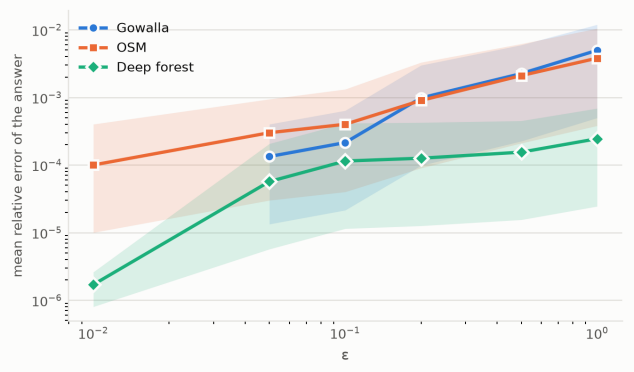


Figure 5: LASQ for ε from 0.01 to 1 on the queries of Figure 2, mean over 10 runs: mean relative error of the answer, $(\text{total} - W_R)/W_R$ with W_R the exact weight within w; shaded: $\pm$ one standard deviation over runs (clipped on the log scale). On Gowalla the error is 0 at $\varepsilon = 0.01$, which the log scale cannot show.

Learning-augmented algorithms and learned indexes. Algorithms with predictions (Mitzenmacher & Vassilvitskii, 2022) use machine-learned advice to improve performance on typical inputs while retaining worst-case guarantees, starting with online problems such as caching (Lykouris & Vassilvitskii, 2021) and scheduling (Purohit et al., 2018), and Angelopoulos et al. (2020) study the trade-off between consistency and robustness. On the data-structure side, learned indexes (Kraska et al., 2018) replace index structures with models of the data distribution, some with provable worst-case bounds (Ferragina & Vinciguerra, 2020); related ideas give learned Bloom filters (Mitzenmacher, 2018) and learned frequency sketches (Hsu et al., 2019). Learned multidimensional and spatial indexes (Nathan et al., 2020; Ding et al., 2020; Li et al., 2020; Qi et al., 2020) adapt their layout to the data and, in some cases, to the query workload, and are evaluated empirically, without worst-case query guarantees.

Learning-augmented search structures. For one-dimensional dictionaries, Lin et al. (2022) assign learned priorities to treaps, with guarantees under Zipfian access distributions, and Chen et al. (2025) obtain static optimality for arbitrary distributions with composite priorities. RobustSL (Zeynali et al., 2024) and, independently, Fu et al. (2025) augment skip lists with predicted access frequencies and are both consistent and robust; Fu et al. (2025) also build learning-augmented k-d trees. All of these structures answer queries for individual keys or points, whose predicted frequencies form a probability distribution over the items. A range query, in contrast, is resolved by several critical cubes at once, and its cost is not tied to any single item. We are not aware of earlier work that gives a learning-augmented structure for multidimensional approximate range search with both consistency and robustness guarantees.

Relation to the two closest works. LASQ builds on the skip quadtree of Eppstein et al. (2005) (compressed quadtrees linked into levels, critical cubes, and the in/out/stabbing recursion for range queries) and on the doubly exponential frequency classes of RobustSL (Zeynali et al., 2024), which were designed for one-dimensional dictionaries with point-access frequencies. Bringing predictions to range search required new ingredients: the critical-cube frequency model and its budget (Lemma 16), which replaces the fact that point-access frequencies sum to 1; the promotion of quadtree cubes rather than items, with levels that remain nested (Definition 12, Lemma 33); layers whose floors are the realized maxima of the previous classes (Section 3.3), which keeps classes from overlapping without a separate overflow argument; a new proof that a query has $O(1)$ critical cubes (Lemma 15, Appendix A.2); and the learnability of the predictions (Theorem 11).

E Additional figures and summary of definitions

This appendix collects figures illustrating the construction and the query algorithm, and, in Table 6, the main notions defined in this paper.

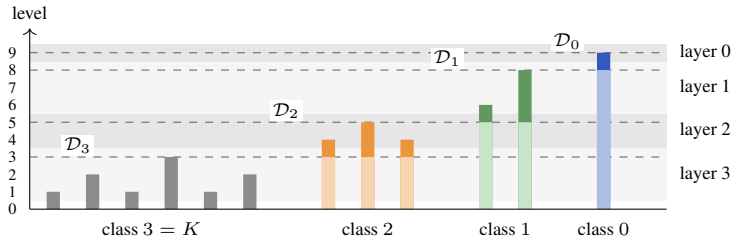


Figure 6: Promotion depths D_u of the cubes of four classes, $K = 3$ (Section 3.3). Classes are processed from the least frequent (K) to the most frequent (0); each class starts just above the realized maximum $\mathcal{D}_{c+1}$ of the previous one (light part of each bar) and adds a geometric number of extra levels. Each class therefore occupies its own layer of levels, and frequently predicted cubes reach the top.

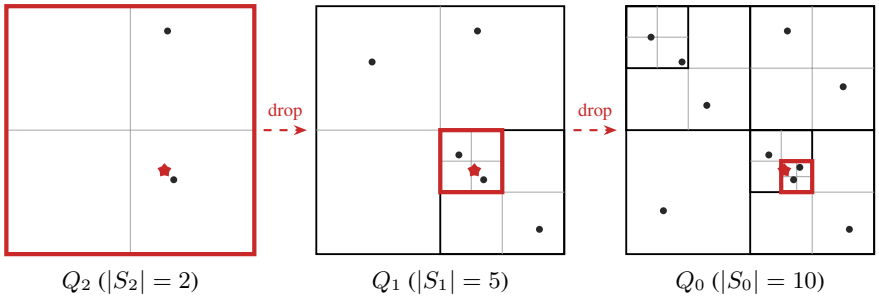


Figure 7: A skip quadtree with levels Q_2, Q_1, Q_0 built on nested random samples $S_2 \subseteq S_1 \subseteq S_0$ (Section 2.3). Locating the star starts in the sparse top level and drops down level by level; the thick cube is where the search stops within each level, and it only shrinks from one level to the next.

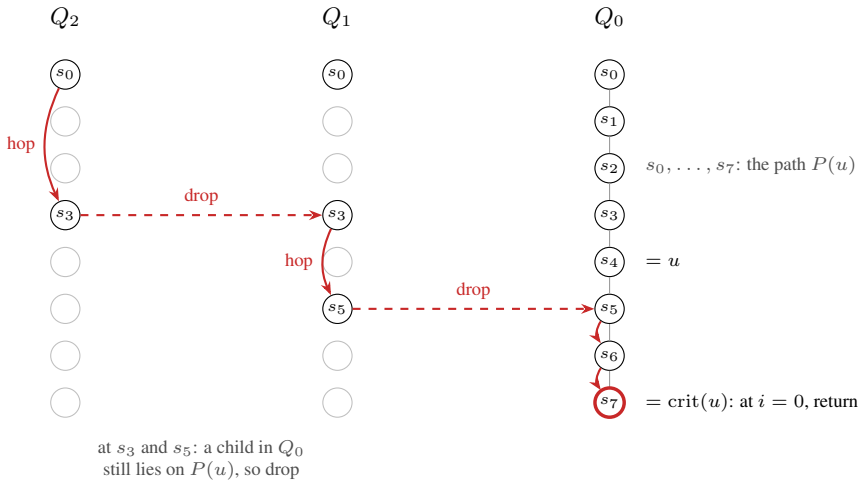


Figure 8: FINDCRITICAL(u) (Algorithm 1) on the path $P(u) = s_0, \ldots, s_7$ of Lemma 28, with $u = s_4$. Gray circles are cubes of Q_0 absent from a level. The search hops along $P(u)$ within Q_2, drops to Q_1 and Q_0 when no child in the level continues the path, and returns $s_7 = \mathrm{crit}(u)$ once it stops at level 0.

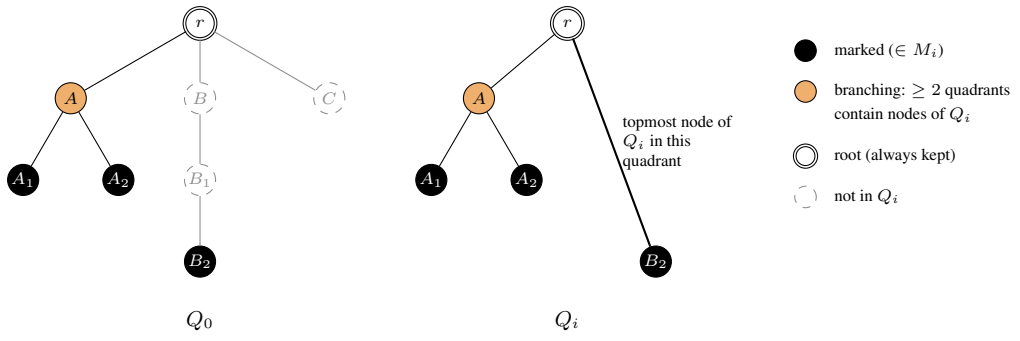


Figure 9: Building a level Q_i from the marked cubes $M_i = \{A_1, A_2, B_2\}$ (Definitions 12 and 13). A is kept because two of its quadrants contain nodes of Q_i; B, B_1 and C are dropped, and the root links directly to B_2, the topmost node of Q_i in that quadrant.

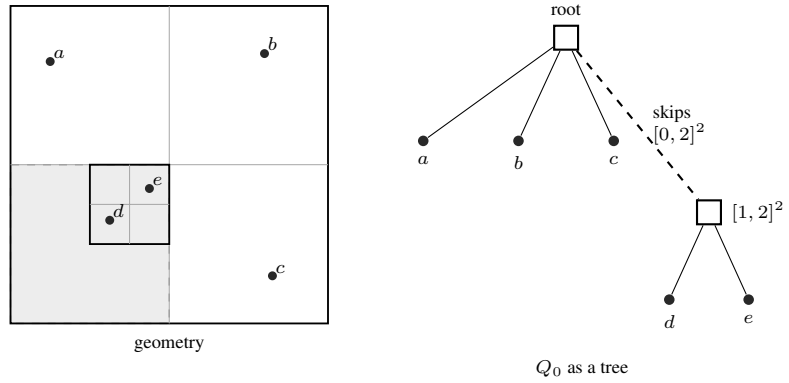


Figure 10: A compressed quadtree Q_0 on five points in $[0, 4]^2$ (Definition 5). Solid cubes are interesting. The shaded cube $[0, 2]^2$ has a single nonempty quadrant, so it is not interesting and is skipped: the root links directly to $[1, 2]^2$, the smallest cube separating d and e.

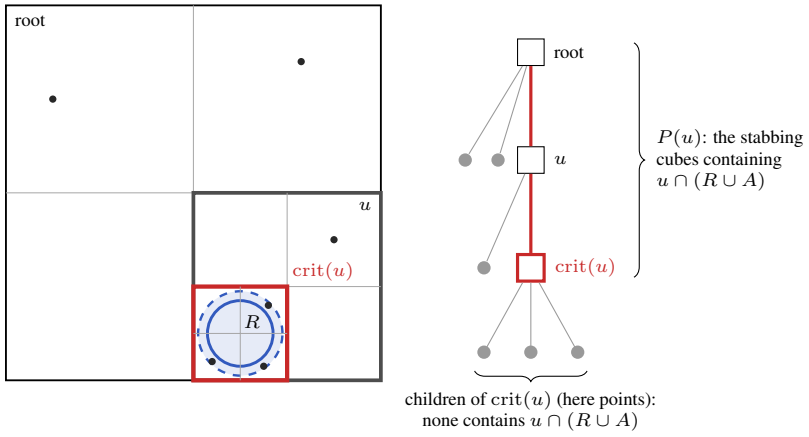


Figure 11: The critical cube below a stabbing cube u (Definition 7). The stabbing cubes containing $u \cap (R \cup A)$ (shaded disc) form a path $P(u)$ from the root; its last cube $\mathrm{crit}(u)$ is critical, since none of its children contains all of $u \cap (R \cup A)$.

Table 6: Summary of definitions.

Term	Meaning	Defined in
Quadrant	one of the 2^d half-size sub-cubes of a cube	Def. 5
Interesting cube	the root, or a cube with ≥ 2 nonempty quadrants; Q_0 keeps exactly these	Def. 5
Query region, annulus	$R = B(v, w)$, and the shell A of thickness εw around it	Def. 3
In / out cube	lies inside $R \cup A$ / misses R (and is not in)	Def. 6
Stabbing cube	meets R and reaches outside $R \cup A$	Def. 6
Critical stabbing cube	stabbing, and no stabbing child covers all of its part of $R \cup A$	Def. 7
Critical cube below u	$\mathrm{crit}(u)$: the smallest stabbing cube containing $u \cap (R \cup A)$	Def. 7
Coverage-critical cube	its part of $R \cup A$ is nonempty and not inside a single child	Def. 27 (App.)
Critical-stabbing frequency	probability that a cube is critical for a random query (true f_u, predicted $\hat{f}_u$)	Def. 8
Correct answer	a total between the weight in R and the weight in $R \cup A$	Def. 4
Frequency class	group of cubes with similar predicted frequency, $\theta^{2^c} \leq \hat{f}_u < \theta^{2^{c-1}}$	Sec. 3.2
Promotion depth	highest level a cube is kept at	Sec. 3.3
Marked cube (at level i)	a cube whose promotion depth is at least i	Sec. 3.3
Level Q_i	marked cubes at level i plus the branching cubes needed to connect them	Def. 12
Topmost nonempty level	highest level with a marked cube; where searches start	Sec. 3.3
Layer	the range of levels occupied by one frequency class	App. A.6
Active cube	a stabbing cube the query still has to resolve	Sec. 3.4
Hop	one move to a child within a level during a search	Sec. 3.4