Approximate Range Counting Under Differential Privacy[*]

Ziyue Huang[†] Ke Yi[‡]

February 25, 2021

Abstract

Range counting under differential privacy has been studied extensively. Unfortunately, lower bounds based on discrepancy theory suggest that large errors have to be introduced in order to preserve privacy: Essentially for any range space (except axis-parallel rectangles), the error has to be polynomial. In this paper, we show that by allowing a standard notion of geometric approximation where points near the boundary of the range may or may not be counted, the error can be reduced to logarithmic. Furthermore, our approximate range counting data structure can be used to solve the approximate nearest neighbor (ANN) problem and k-NN classification, leading to the first differentially private algorithms for these two problems with provable guarantees on the utility.

1 Introduction

Differential privacy (DP) [7] is a rigorous notion of privacy-preserving data publishing, which has been widely adopted. Essentially, it ensures that no individual can substantially change the probability distribution on the published results of an analysis. A central problem studied under differential privacy is *counting queries.* Given a set of points $P \subseteq \mathcal{U}$, where $\mathcal{U}$ is the universe, the goal is to release a differentially private data structure of P, so that for any query $Q \subseteq \mathcal{U}$ from a certain query family $\mathcal{Q}$, we can find the count[1] $|P \cap Q|$. To ensure the privacy of P, some noise has to be injected to the query answers, so the key challenge is to minimize the error under a given privacy budget (ε, δ); please see Section 2.1 for a more formal DP definition and the meaning of the privacy parameters ε, δ.

When $\mathcal{Q}$ consists of arbitrary subsets of $\mathcal{U}$, then the optimal error achievable is $\sqrt{n} \cdot \mathrm{poly}(\frac{1}{\varepsilon}, \log \frac{|\mathcal{U}|\cdot|\mathcal{Q}|}{\delta})$ [12]. In the other extreme when $\mathcal{Q} = \mathcal{U}$, which are called *point queries*, also known as the *histogram problem*, the error is $O(\frac{1}{\varepsilon} \cdot \min\{\log |\mathcal{U}|, \log \frac{1}{\delta}\})$ [21]. This is a large gap. Thus, there have been a lot of interest in studying specific query families, and particularly, identifying those that yield polylogarithmic errors. In this paper, we study query families in constant-dimensional Euclidean space. More precisely, the dataset P consists of n points from $\mathcal{U} = [u]^d$, where $[u]$ denotes integers from 1 to u, while $\mathcal{Q}$ consists of geometric ranges of a certain type, e.g., (axis-parallel) rectangles, halfspaces, simplices, spheres, or arbitrarily-shaped regions. In the geometric setting, $\mathcal{Q}$ is often called a *range space.*

1.1 Previous work

For $d = 1$, the only interesting range space is the set of intervals. For this case, Dwork et al. [8] present an $(\varepsilon, 0)$-DP algorithm with error $O\left(\frac{1}{\varepsilon}\log^{1.5} u\right)$, as well as a lower bound

[*]This work has been supported by HKRGC under grants 16202317, 16201318, and 16201819.

[†]Department of Computer Science and Engineering, HKUST. `zhuangbq@cse.ust.hk`.

[‡]Department of Computer Science and Engineering, HKUST. `yike@cse.ust.hk`.

[1]We allow P to be a multiset, i.e., the same point from $\mathcal{U}$ may appear multiple times in P. In this case, $|P \cap Q|$ adds up all the multiplicities of points of P that are in Q.

of $\Omega(\frac{1}{\varepsilon}\log u)^2$. Bun et al. [4] show that under (ε, δ)-DP, this lower bound can be broken, achieving error $2^{(1+o(1))\log^* u} \cdot \log(1/\delta)/\varepsilon$. They also show an $\Omega(\log^* u \cdot \log(1/\delta)/\varepsilon)$ lower bound for $\exp(-\varepsilon n/\log^* n) \le \delta \le 1/n^2$. Note that these lower bounds justify the restriction that the points must have integer coordinates. In addition, they hold irrespective of the space/time of the data structure. They solely rely on the privacy requirement.

For $d \ge 2$, Chan et al. [5] extend the algorithm of [8] to answer axis-parallel rectangle queries with error $O\left(\frac{1}{\varepsilon}\log^{1.5d} u\right)$, which is later improved to $O\left(\frac{1}{\varepsilon}(\log u + \log^{1.5d} n)\right)$ [9]. These results show that rectangles are "easy", namely, they admit polylogarithmic errors. Unfortunately, it turns out they are essentially the only easy cases. Muthukrishnan and Nikolov [17] and Nikolov et al. [18] build an equivalence between the error and the *discrepancy* of the range space $\mathcal{Q}$. Rectangle range counting has polylogarithmic errors essentially because the discrepancy of rectangles is polylogarithmic [16]. On the other hand, since the discrepancy of other natural range spaces, such as halfspaces, simplices, and spheres, is all $n^{\phi(d)}$, where $\phi(d) \in [\frac{1}{4}, \frac{1}{2}]$ is a constant depending on the particular range space and d [16], they are ruled out for having polylogarithmic errors. Making things worse, if we allow nonconvex ranges, the discrepancy becomes $\sqrt{n}$, which means that the aforementioned solution that works for arbitrary query families [12] is already optimal, namely, geometry doesn't help.

However, a series of differentially private range counting data structures [6, 14, 22, 15, 19, 20, 23] have been proposed by practitioners, mostly based on hierarchical space decompositions. They work well on many real-world datasets, but perform badly on high-discrepancy point sets, as predicted by the lower bounds [17, 18].

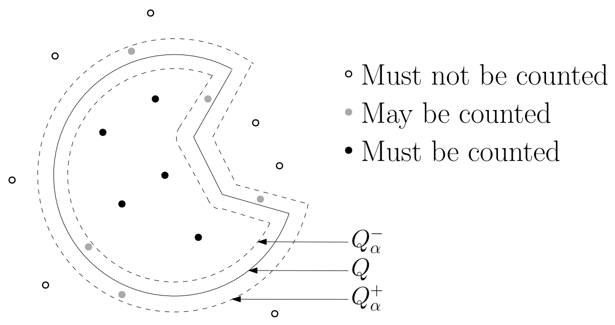


Figure 1: Approximate range counting.

1.2 Our results

This paper provides the theoretical justification that geometry *does* help, at least in constant dimensions. However, in order to circumvent the polynomial discrepancy lower bounds, we have to introduce some relaxation. Specifically, we consider *approximate range counting* as defined in [1]. The *diameter* of a range Q is the largest distance between any two points in Q. Given a range Q of diameter w and a constant *fuzziness parameter* $0 < \alpha < 1$, the inner range Q_α^- is defined as the region of points whose distance from any point exterior to Q is at least αw, while the outer range Q_α^+ is those points whose distance from a point interior to Q is at most αw. Then any count between $|P \cap Q_\alpha^-|$ and $|P \cap Q_\alpha^+|$ is considered a valid answer to Q, i.e., points in $Q_\alpha^+ - Q_\alpha^-$ may or may not be counted (see Figure 1). In fact, Blum et al. [3] used a similar notion of geometric approximation with differential privacy, but their error is still polynomial.

We show that polylogarithmic errors are achievable under this notion of approximation. In particular, we show that (1) under $(\varepsilon, 0)$-DP, the error is $O\left(\frac{1}{\varepsilon}\log u\right)$, with a lower bound of

[2] All upper bounds stated in this paper hold for any single query with constant probability, while lower bounds hold for the maximum error of all queries with constant probability.

$\Omega\left(\frac{1}{\varepsilon}\log u\right)$; and (2) under (ε,δ)-DP, the error can be reduced to $O(\frac{1}{\varepsilon}(\log\log u+\log\frac{1}{\delta}+\log^{1.5} n))$, with a lower bound of $\Omega\left(\frac{1}{\varepsilon}\log^* u\cdot\log(1/\delta)\right)$ for $\exp(-\varepsilon n/\log^* n)\le\delta\le 1/n^2$. These results hold for *any* range space in constant dimensions (even if the ranges are nonconvex). Compared with the $\sqrt{n}$ lower bound mentioned above, we obtain this exponential improvement exactly due to geometry—the notion of fuzziness has no counterpart in arbitrary query families over an abstract set system. Technically, the fuzziness allows a packing argument (see Theorem 2.4), which we borrow from the non-private setting [1].

In practice, this notion of fuzziness is often acceptable, considering that either data or the range (or both) are often imprecise themselves. For example, if the public want to know how many people got COVID-19 in their neighborhood, whether the cases near the boundary are included or not is not very important. Another way of comparing our result with prior work on DP range counting (which does not allow fuzziness) is that we have an extra $|P\cap(Q_\alpha^+-Q_\alpha^-)|$ term in our error bound. This term is incomparable to $\sqrt{n}$, so our result doesn't contradict the discrepancy-based lower bound [17, 18]. However, this error term is small on most real-world datasets, which explains why small errors are often observed in many practical solutions.

Our data structures are naturally based on approximate range counting structures in the non-private setting, in particular, the BBD-tree [1]. However, some non-trivial modifications and analyses are needed to make the BBD-tree private, especially for reducing the dependence on u in the (ε,δ)-DP case.

Our approximate range counting data structure can also be used to solve the approximate nearest neighbor (ANN) problem and k-NN classification, yielding the first DP algorithms for these two problems with provable guarantees on the utility. For the ANN problem, returning the NN itself (or any other point in P) is not private; instead, we return a distance that approximates the nearest distance. Returning only the distance can still be useful in many applications, e.g., if the user only wants to know whether there is a point in P that is sufficiently close to her query point. Also, only the distance to the query point is needed in k-NN classification; please see Section 4 for details.

2 Preliminaries

2.1 Differential privacy

Let $P\sim P'$ denote two neighboring databases, i.e., one contains one more point than the other.

Definition 2.1 (Differential Privacy [10]). *For $\varepsilon>0$ and $\delta\ge 0$, a randomized algorithm $\mathcal{M}$ is (ε,δ)-differentially private if for any neighboring databases $P\sim P'$ and any $S\subseteq\mathsf{Range}(\mathcal{M})$,*

$$\mathsf{Pr}[\mathcal{M}(P)\in S]\le e^{\varepsilon}\cdot\mathsf{Pr}[\mathcal{M}(P')\in S]+\delta$$

The case when $\delta=0$ is also referred to as *pure differential privacy.* In practice, ε is usually a constant ranging from 0.1 to 10, while δ must be much smaller than $1/n$. For a numeric query f, the most common technique for designing DP mechanisms is by masking the result with Laplace noise calibrated to the *sensitivity* of the query $\Delta_f=\max_{P\sim P'}|f(P)-f(P')|$. We use $\mathrm{Lap}(\lambda)$ to denote the Laplace distribution with parameter λ, which has probability density function $\mathsf{Pr}[X=z]=\frac{1}{2\lambda}\exp(-|z|/\lambda)$ and variance $2\lambda^2$. And the fact that $\mathsf{Pr}[|X|>t\cdot\lambda]\le\exp(-t)$ is often useful to analyze the accuracy of the Laplace mechanism.

Theorem 2.1 (Laplace Mechanism [10]). *For a numeric query f over a database P, an $(\varepsilon,0)$-differentially private mechanism is to output $f(P)+X$, where $X\sim Lap(\Delta_f/\varepsilon)$.*

We need the following properties of DP mechanisms.

Theorem 2.2 (Basic Composition [10])**.** *Let $\mathcal{M}_i$ be an (ε, δ)-differentially private mechanism for all $i \in [k]$, then $\mathcal{M}(P) = (\mathcal{M}_1(P), \mathcal{M}_2(P), \ldots, \mathcal{M}_k(P))$ is $(k\varepsilon, k\delta)$-differentially private.*

Theorem 2.3 (Group Privacy [10])**.** *Let $\mathcal{M}$ be an $(\varepsilon, 0)$-differentially private mechanism. For any two databases P and P' that differ by at most k individuals and any $S \subseteq \mathsf{Range}(\mathcal{M})$,*

$$\mathsf{Pr}[\mathcal{M}(P) \in S] \leq e^{k\varepsilon} \cdot \mathsf{Pr}[\mathcal{M}(P') \in S]$$

2.2 The BBD-tree

The *balanced box-decomposition tree (BBD-tree)* [2] is a hierarchical space decomposition[3] where each node is associated with a region of space called a *cell*. We define a *box* to be an axis-parallel rectangle whose aspect ratio is either 1 or 2. Recall that points in P have integer coordinates. To avoid the ambiguity of points lying on the boundary of a box, the boxes' corners all have coordinates in the form $x + \frac{1}{2}$ for integer x. Each cell in the BBD-tree is either a box or the region between two boxes, one enclosed within the other. Thus each cell comprises of an outer box and an optional inner box. For a cell c, its size is the length of the longest side of its outer box. We use $\text{size}(c)$ to denote its size and $\text{count}(c)$ to denote the number of points in P lying inside c. The aspect ratio of each cell in the BBD-tree is bounded by 2, which allows us to bound the number of cells that intersect any range of a given diameter.

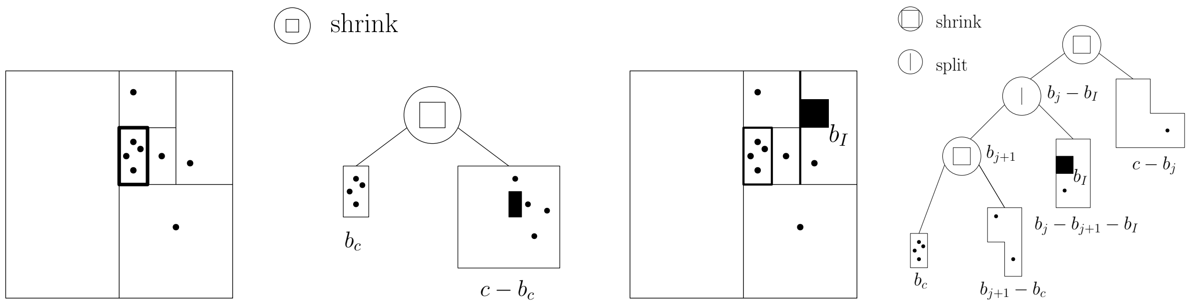
Theorem 2.4 ([1])**.** *Consider any space decomposition of height h, consisting of cells of bounded aspect ratios. Let C be any subset of its cells, where each cell has size at least s and they all intersect a range of diameter w. Then $|C| = O(h + (w/s)^d)$. If the cells are also disjoint, then $|C| = O((w/s)^d)$.*

Note that the quadtree has this property (where the aspect ratios are all 1), but it has a large height $h = O(\log u)$. On the other hand, the BBD-tree has only $O(\log n)$ height. To achieve this without violating the aspect ratio constraint, the key idea is to give a little more flexibility to the shape of the cells.

The BBD-tree is a binary tree and is constructed by the repeated application of two partitioning operations alternatively on each level, *split* and *shrink*. Starting from the root node whose corresponding cell is the entire space $[u]^d$, we recursively divide each cell c by either split or shrink, until $\text{size}(c)$ or $\text{count}(c)$ is at most 1.

To split a cell, we bisect it along its longest side (it is guaranteed that the bisecting hyperplane does not intersect c's inner box if it has one). The resulting cells have aspect ratio of either 1 or 2. Now consider the shrink operation. If c does not have an inner box, the shrink operation is performed by repeatedly applying split operations and recursing on the child box b_c with the majority of points in c, until $\text{count}(b_c \cap c) \leq \frac{2}{3}\text{count}(c)$. Then all intermediate splits are discarded, and the two children of c are b_c and $c - b_c$ (i.e., c and b_c are the outer and inner boxes of the cell $c - b_c$; see Figure 2a). For the case when there is an inner box b_I in c, we first follow the same procedure to obtain the box b_c. If $b_I \subseteq b_c$, then we shrink c to $b_c - b_I$ (the other child is thus $c - b_c$). Otherwise, suppose the majority boxes obtained during the series of splits are $b_1, \ldots, b_k = b_c$. Let b_j be the smallest box in the sequence that contains b_I. Then we first shrink c to $b_j - b_I$ (the other child is $c - b_j$), then split $b_j - b_I$ to b_{j+1} and $b_j - b_{j+1} - b_I$, then shrink b_{j+1} to b_c (the other child is $b_{j+1} - b_c$), as shown in Figure 2b. The shrink operations ensure that $\text{count}(c)$ decreases by a constant factor as we descend a constant number of levels, leading to an $O(\log n)$ height of the BBD-tree.

[3]As a hierarchical space decomposition naturally corresponds to a tree, where each node corresponds to a cell, we will use the terms "node" and "cell" interchangeably.



(a) shrink without an existing inner box (b) shrink with an existing inner box b_I

Figure 2: The shrink operation in the BBD-tree.

3 Approximate range counting

In this section, we describe and analyze our private range counting data structure. It is similar to the existing heuristic solutions [6, 19, 23], namely, we also release a hierarchical space decomposition that is differentially private and can be used to answer any range query. Each cell c in the space decomposition is associated with a count masked by Laplace noise, which we denote by noisy_count(c). However, there are two differences, which are important for achieving theoretical guarantees: First, we use a different query procedure, which is described in Section 3.1. Second, we use the BBD-tree as our space decomposition, and we show how to make it private in Sections 3.2 and 3.3.

3.1 Query procedure

Our query procedure is similar to the one in [1]. Given a space decomposition and a range Q, we visit its cells in a top-down manner, summing up the noisy counts of cells as we go along. We stop exploring further at a cell in the following two cases: (1) if $c \cap Q^-_\alpha = \emptyset$, we just skip c; and if (2) if $c \subseteq Q^+_\alpha$, we add noisy_count(c) and then skip c. If we do not skip a cell, we visit its children recursively. The detailed query procedure is given in Algorithm 1.

Algorithm 1 Approximate Range Counting Procedure: Query(Q, α, c)

```
Input: a range $Q$; range approximation factor $\alpha$; a cell $c$ in the space decomposition.
Output: an answer to the $\alpha$-approximate range query $Q$.
1: if $c \cap Q^-_\alpha = \emptyset$ then return 0
2: if $c \subseteq Q^+_\alpha$ then return noisy_count($c$)
3: if $c$ is a leaf cell then return 0
4: $s \leftarrow 0$
5: for each child $c'$ of $c$ do
6:     $s \leftarrow s + \mathrm{Query}(Q, \alpha, c')$
7: return s
```

It is clear that line 1 in Algorithm 1 does not introduce any error. Line 2 introduces a zero-mean error with magnitude proportional to the noise level, while line 3 introduces a bias equal to count(c). We will account for these two sources of errors when analyzing the accuracy of our private BBD-tree.

3.2 Pure differential privacy

For pure DP, we use the BBD-tree with only split operations, which results in a full (binary) quadtree. The quadtree has u^d leaves and $O(\log u)$ height. It is safe to release this tree structure as it does not depend on P. It is clear that the sensitivity of $\mathrm{count}(c)$ is 1 for each cell c and each point contributes to the counts of $d \log u$ nodes on a root-to-leaf path. Then by the basic composition theorem, adding noise drawn from $\mathrm{Lap}(d \log u/\varepsilon)$ to each $\mathrm{count}(c)$ is sufficient to preserve $(\varepsilon, 0)$-DP.

Now we analyze the error of an approximate range counting query. The proofs of the following theorem, as well as some others, are given in the appendix.

Theorem 3.1. *There is an $(\varepsilon, 0)$-differentially private space decomposition, such that any α-approximate range counting query can be answered within error $O\left(\varepsilon^{-1}\alpha^{-d/2} \log u \log \frac{1}{\beta}\right)$ with probability at least $1-\beta$. Moreover, it can be pruned to $O(n)$ nodes with probability at least $1-\beta$, and in this case the error is $O\left(\varepsilon^{-1}\alpha^{-d} \log u \log \frac{n}{\beta}\right)$.*

The $\log u$ dependency of the error follows from the height of the tree. One may wonder if we could make the shrink operation differentially private, thus reducing its height to $O(\log n)$. Unfortunately, we show that this is not possible under pure DP, by presenting an $\Omega(\log u)$ lower bound on the error for any pure DP approximate range counting algorithm (see Section 5).

3.3 (ε, δ)-differential privacy

By adopting (ε, δ)-DP, we can improve the error dependence on u from $\log u$ to $\log \log u$. The key, as mentioned above, is to make the shrink operations private. Recall that the original shrink algorithm on a cell c uses a series of splits until the majority box contains at most $\frac{2}{3}\mathrm{count}(c)$ points. To make this comparison private and ensure that the shrink algorithm still decreases $\mathrm{count}(c)$ by a constant factor, we can add noise proportional to $\mathrm{count}(c)$. However, directly using $\mathrm{count}(c)$ to calibrate the noise would not be differentially private. In fact, even the total count, i.e., n, cannot be released; so we use $\tilde{n} := n + \frac{4}{\varepsilon} \log \frac{2}{\beta} + \mathrm{Lap}(\frac{4}{\varepsilon})$ instead. The idea is to make $O(\log \tilde{n})$ "guesses" for the right noise level, which is captured by a parameter h. At $h = 0$, the magnitude of noise is $\tilde{n}^{O(1)}/\varepsilon$, and it will exponentially decrease as h increases; at $h = O(\log \tilde{n})$, the noise magnitude will become $O(1/\varepsilon)$.

We now describe how to construct a private BBD-tree. We still use shrink and split operations on alternating levels of the tree. The split operation is the same as in the standard BBD-tree, while the private shrink operation is shown in Algorithm 2 for a given cell c with noise level h. The shrink operation on the root node is invoked with $h = 0$; on a non-root node c, we will use $h = h_p + 1$, where h_p is the noise level used in the shrink operation at the grandparent of c. We stop further subdividing c when $\mathrm{size}(c) = 1$, when we are about to invoke the shrink algorithm with $h = H := \lceil 2.5 \log \tilde{n} \rceil$, or after the shrink algorithm returns `LEAF`.

Algorithm 2 works as follows. If $\mathrm{count}(c)$ is too small compared with the the current noise level (determined probabilistically), the algorithm will return `AGAIN`. In this case, we invoke the algorithm again with $h \leftarrow h + 1$, decreasing the noise magnitude by a constant factor. Otherwise it will split c repeatedly and recurse on the majority box. Compared with the non-private shrink operation, we make the following changes. Let b_c be the current majority box. First, we add noise before checking if $\mathrm{count}(b_c \cap c) \leq \frac{2}{3}\mathrm{count}(c)$. As a result, we cannot guarantee that $\mathrm{count}(c)$ will decrease by a constant factor, but it will be the case with high probability. Second, in the non-private case, we can just return b_c when $\mathrm{count}(b_c \cap c) \leq \frac{2}{3}\mathrm{count}(c)$ after the last split. We can no longer do this, as being the majority box is also sensitive information which depends on P (this was not sensitive before the splits stop, as the count of the majority box was large enough to hide the presence or absence of one data point). Instead, we return

b_c or its sibling after a noisy comparison. Finally, when c has an inner box b_I, the non-private shrink algorithm will never try to split b_I because the count would have become $0 \leq \frac{2}{3}\text{count}(c)$ already. However, as the comparison with $\frac{2}{3}\text{count}(c)$ is now probabilistic, we need to guard against this from happening. In particular, when the algorithm tries to split b_I, it will return `LEAF`. If the algorithm does not return `AGAIN` or `LEAF`, it will return a box b_c. Then we shrink c to b_c as in the standard BBD-tree as described in Section 2.2; this may involve an interleaving split in case c has an inner box b_I that is disjoint from b_c.

Algorithm 2 Private Shrink Operation; Shrink(c, h, β)

```
Input: an input cell $c = b_o - b_I$ ($b_I$ is optional) with noise level $h$; the failure probability $\beta$.
Output: a child box $b_c$, AGAIN, or LEAF.
 1: $\varepsilon_h \leftarrow \left(\frac{3}{4}\right)^{H-h} \cdot \varepsilon/100$
 2: Draw $\zeta \sim \text{Lap}(1/\varepsilon_h)$. Let $\widetilde{R} \leftarrow \text{count}(c) + \zeta$.
 3: if $\widetilde{R} < 370 \cdot \left(\log \frac{2\tilde{n}d}{\beta\delta} + \log\log u\right)/\varepsilon_h$ then return AGAIN
 4: Draw $\gamma \sim \text{Lap}(1/\varepsilon_h)$. Let $\widetilde{T} \leftarrow \frac{2}{3}\text{count}(c) + \gamma$.
 5: $b_c \leftarrow b_o$
 6: repeat
 7:     $b_l, b_r \leftarrow \text{split}(b_c)$
 8:     if $\text{count}(b_l \cap c) > \text{count}(b_r \cap c)$ then $b_c \leftarrow b_l$ else $b_c \leftarrow b_r$
 9:     if $b_I$ exists and $b_c = b_I$ then return LEAF
10:     Draw $\eta \sim \text{Lap}(1/\varepsilon_h)$. Let $\tilde{\theta} \leftarrow \text{count}(b_c \cap c) + \eta$.
11: until $\tilde{\theta} < \widetilde{T}$ or $\text{size}(b_c) = 1$
12: Draw $\xi_l, \xi_r \sim \text{Lap}(1/\varepsilon_h)$.
13: if $\text{count}(b_l \cap c) + \xi_l > \text{count}(b_r \cap c) + \xi_r$ then $b_c \leftarrow b_l$ else $b_c \leftarrow b_r$
14: return $b_c$
```

Finally, after the BBD-tree has been constructed, we associate each cell c a noisy count, obtained by adding noise drawn from $\text{Lap}(40\log\tilde{n}/\varepsilon)$ to count(c). The following theorem guarantees its privacy and establishes some of its key properties that will be useful for proving its utility.

Theorem 3.2. *The private BBD-tree preserves (ε, δ)-differential privacy. Furthermore, for $n > \frac{8}{\varepsilon}\log\frac{2}{\beta}$, with probability at least $1-\beta$, it has height $O(\log n)$ and $O(n)$ nodes, and every leaf cell c has either size$(c) = 1$ or count$(c) = O\left(\varepsilon^{-1} \cdot \left(\log\frac{n}{\beta\delta} + \log\log u\right)\right)$.*

Proof. Let $\ell_h = 10 \cdot (\log\frac{2\tilde{n}d}{\beta\delta} + \log\log u)/\varepsilon_h$. Note that in a private shrink operation (Algorithm 2) on an input cell c with noise level h, all the noises are generated from $\text{Lap}(1/\varepsilon_h)$. We say that a Laplace noise is *bounded* if its absolute value does not exceed ℓ_h.

We begin with the following lemma on one invocation of Algorithm 2:

Lemma 3.1. *If all its Laplace noises are bounded, then Algorithm 2 (1) will not return* `LEAF`*; (2) if it returns* `AGAIN`*, then we have count$(c) \leq 38\ell_h$; (3) otherwise we must have count$(c) \geq 36\ell_h$, and the algorithm returns a child box b_c which satisfies either of the following two properties: (a) size$(b_c) > 1$ and $\frac{1}{4}$count$(c) \leq$ count$(b_c \cap c) \leq \frac{3}{4}$count$(c)$; (b) size$(b_c) = 1$ and count$(b_c \cap c) \geq \frac{1}{4}$count$(c)$.*

Proof. (2) is easy to show. If Algorithm 2 returns `AGAIN`, since all noises are bounded, then it follows from line 3 that count$(c) \leq 37\ell_h - \zeta \leq 38\ell_h$. The first part of (3) is also true, since if the algorithm does not return `AGAIN`, we must have count$(c) \geq 37\ell_h - \zeta \geq 36\ell_h$.

Next, we prove the second part of (3). Let $b_c^{(1)}, b_l^{(1)}, b_r^{(1)}, \tilde{\theta}^{(1)}, \ldots, b_c^{(k)}, b_l^{(k)}, b_r^{(k)}, \tilde{\theta}^{(k)}$ denote the values of the variables $b_c, b_l, b_r, \tilde{\theta}$ at the end of each iteration in the repeat-until loop; note that $b_c^{(i)}$ is always the majority box between $b_l^{(i)}$ and $b_r^{(i)}$, $i = 1, \ldots, k$. Let b_c be the final output child box which is either $b_l^{(k)}$ or $b_r^{(k)}$ after a noisy comparison (line 13). We have $k \le d \log u$ as every d splits will decrease size($b_c^{(i)}$) by a factor of 2. To prove the second part of (3), we only need to consider the case size(b_c) > 1. In this case, we have $\tilde{\theta}^{(k)} < \widetilde{T}$ and $\tilde{\theta}^{(k-1)} \ge \widetilde{T}$ according to line 11. Given $\tilde{\theta}^{(k)} < \widetilde{T}$, we have

$$\text{count}(b_c \cap c) \le \text{count}(b_c^{(k)} \cap c) \le \frac{2}{3}\text{count}(c) - \eta^{(k)} + \gamma \le \frac{2}{3}\text{count}(c) + 2\ell_h \le \frac{3}{4}\text{count}(c),$$

where the first inequality is because b_c is either $b_l^{(k)}$ or $b_r^{(k)}$, while $b_c^{(k)}$ is the majority box between $b_l^{(k)}$ and $b_r^{(k)}$; the second inequality follows from the definition of $\tilde{\theta}^{(k)}$ and $\widetilde{T}$; the third inequality follows from the boundedness of $\eta^{(k)}$ and γ; and the fourth inequality is due to count$(c) \ge 36\ell_h$. On the other hand, given $\tilde{\theta}^{(k-1)} > \widetilde{T}$, following similar arguments, we have

$$\text{count}(b_c^{(k-1)} \cap c) \ge \frac{2}{3}\text{count}(c) - \eta^{(k-1)} + \gamma \ge \frac{2}{3}\text{count}(c) - 2\ell_h.$$

Assume w.l.o.g. that $b_c = b_l^{(k)}$. Then,

$$\begin{aligned} 2 \cdot \text{count}(b_c \cap c) &= \text{count}(b_l^{(k)} \cap c) + \text{count}(b_l^{(k)} \cap c) \\ &\ge \text{count}(b_l^{(k)} \cap c) + (\text{count}(b_r^{(k)} \cap c) - \xi_l + \xi_r) \\ &= \text{count}(b_c^{(k-1)} \cap c) - \xi_l + \xi_r \\ &\ge \text{count}(b_c^{(k-1)} \cap c) - 2\ell_h \\ &\ge \frac{2}{3}\text{count}(c) - 4\ell_h, \end{aligned}$$

where the first inequality follows from line 13, and the second inequality follows from the boundedness of ξ_l and ξ_r. Because count$(c) \ge 36\ell_h$, we have

$$\text{count}(b_c \cap c) \ge \frac{1}{3}\text{count}(c) - 2\ell_h \ge \frac{1}{4}\text{count}(c),$$

proving (3).

For (1), observe that $b_c^{(i)}$ is not identical to b_I (if it exists) for any $i \in [k]$, since

$$\text{count}(b_c^{(i)} \cap c) \ge \frac{1}{2}\text{count}(b_c^{(i-1)} \cap c) \ge \frac{1}{3}\text{count}(c) - \ell_h > 0,$$

where the first inequality is because $b_c^{(i)}$ is the majority box after a split on $b_c^{(i-1)}$; the second inequality follows from $\tilde{\theta}^{(i-1)} > \widetilde{T}$ and the boundedness of $\eta^{(i-1)}$ and γ; and the third inequality is due to count$(c) \ge 36\ell_h$. Thus the algorithm will not return `LEAF`. ■

We are now ready to prove the utility properties of a private BBD-tree. If the event F that $n \le \tilde{n} \le 2n$ holds, the $O(\log n)$ height trivially follows from the construction algorithm. Due to the exponential tail of the Laplace distribution and $n > \frac{8}{\varepsilon} \log \frac{2}{\beta}$, we have $\Pr[F] \ge 1 - \beta/2$. To prove that the tree has $O(n)$ nodes and the property of the leaves stated in the theorem, consider the event E that the Laplace noises used in all invocations of Algorithm 2 when constructing the private BBD-tree are bounded, conditioned upon which the properties stated in Lemma 3.1 hold.

Lemma 3.2. $\Pr[E] \geq 1 - \beta\delta/2$.

Proof. In one invocation of Algorithm 2 on a cell c, we draw $O(d \log u)$ independent Laplace noises from $\mathrm{Lap}(1/\varepsilon_h)$, so they are all bounded with probability at least $1 - \beta\delta/(2\tilde{n}^{10})$, due to the exponential tail of Laplace distribution and a union bound. To bound the noise values in all the invocations, first consider the case when Algorithm 2 returns `AGAIN`. When this happens, we invoke it on c again with noise level $h \leftarrow h + 1$. This step is logically equivalent to making two copies of c, say c' and c'', such that c' is the only child of c and c'' is the only child of c'. Recall that we alternate between split and shrink operations, so we will later perform a shrink on c'' with $h \leftarrow h + 1$. Then in this logically equivalent BBD-tree, we only invoke one split operation or one shrink operation (or just make a copy of itself). Recall that we stop the construction when we reach $H = 2.5 \log \tilde{n}$, and one shrink operation may generate 4 levels of the tree (in case an interleaving split is needed), so the height of the logical tree is at most $4H = 10 \log \tilde{n}$. Thus, it has at most $\tilde{n}^{10}$ nodes, namely, at most $\tilde{n}^{10}$ invocations of Algorithm 2. Then the lemma is proved by a union bound. ■

Lemma 3.3. *Conditioned upon $E \wedge F$, the private BBD-tree has $O(n)$ nodes and every leaf cell c has either size$(c) = 1$ or count$(c) = O\left(\varepsilon^{-1} \cdot \left(\log \frac{n}{\beta\delta} + \log\log u\right)\right)$.*

Proof. Conditioned upon E, all the shrink operations have the three properties stated in Lemma 3.1. For any leaf c of size larger than 1, consider the path from root to c in the BBD-tree. There are exactly H invocations of Algorithm 2 on the path, as none of them has returned `LEAF`. At most $\log_{4/3} n < H$ (as the event F holds) of them have returned a child box, since every such shrink operation decreases the count by a factor at least $\frac{1}{4}$. This means that there is at least one invocation of Algorithm 2 that returned `AGAIN`.

Let c^* be the smallest cell on this path on which Algorithm 2 returned `AGAIN`, and let h^* be its noise level. By Lemma 3.1,

$$\mathrm{count}(c^*) \leq 38\ell_{h^*} = O\left(\left(\frac{4}{3}\right)^{H-h^*} \cdot \left(\log \frac{nd}{\beta\delta} + \log\log u\right)/\varepsilon\right).$$

Each of the remaining $H - h^*$ shrink operations must have reduced the count by a factor of at least $\frac{1}{4}$, so

$$\mathrm{count}(c) = O\left(\left(\frac{3}{4}\right)^{H-h^*} \cdot \ell_{h^*}\right) = O\left(\left(\log \frac{nd}{\beta\delta} + \log\log u\right)/\varepsilon\right).$$

To see that the BBD-tree has $O(n)$ nodes, consider the parent or grandparent, depending on which one is on the shrink level, of any leaf. Algorithm 2 might have been invoked on this node multiple times, but the last one must have returned a child box, implying that it contained at least ℓ_H data points by Lemma 3.1. Then we conclude that the private BBD-tree has $O(n)$ nodes since there are $O(\log n)$ levels and each level has at most $O(n/\ell_H)$ nodes. ■

Finally, we prove its privacy.

Lemma 3.4. *The private BBD-tree preserves (ε, δ)-differential privacy.*

Proof. Note that releasing $\tilde{n}$ preserves $(\varepsilon/4, 0)$-DP. Furthermore, there are two parts of the BBD-tree: its structure and the noisy counts associated with its cells. Since the height of the tree is at most $10 \log \tilde{n}$, the released noisy counts preserves $(\varepsilon/4, 0)$-DP by a standard argument. Below, we show that the tree structure is $(\varepsilon/2, \delta)$-DP. Then by the basic composition theorem, the private BBD-tree achieves (ε, δ)-DP.

The structure depends on the input point set P, as well as the Laplace noises generated internally during all the invocations of Algorithm 2. Let $\mathcal{M}(P, \Gamma)$ denote the tree structure constructed on point set P using noises Γ. We will show that for any two neighboring data sets P, P', and any tree structure y,

$$\mathsf{Pr}[\mathcal{M}(P, \Gamma) = y \wedge E] \leq e^{\varepsilon/2} \cdot \mathsf{Pr}[\mathcal{M}(P', \Gamma) = y], \tag{1}$$

where the probability is computed over the randomness of Γ, Then, for any set of output structures S, we have

$$\mathsf{Pr}[\mathcal{M}(P, \Gamma) \in S] \leq \mathsf{Pr}[\mathcal{M}(P, \Gamma) \in S \wedge E] + \mathsf{Pr}[\bar{E}] \leq e^{\varepsilon/2} \cdot \mathsf{Pr}[\mathcal{M}(P', \Gamma) \in S] + \delta,$$

namely, $\mathcal{M}$ is $(\varepsilon/2, \delta)$-DP.

To prove Eq. (1), it suffices to demonstrate an injection f from $\{\Gamma : \mathcal{M}(P, \Gamma) = y \wedge E\}$ to $\{\Gamma : \mathcal{M}(P', \Gamma) = y\}$ such that $\mathsf{Pr}(\Gamma) \leq e^{\varepsilon/2} \cdot \mathsf{Pr}(f(\Gamma))$. In order to achieve this, f can only change a small number of Laplace noises by a constant magnitude each. Note that the presence or absence of a data point will only affect the cells on a root-to-leaf path of y. For any Γ such that $\mathcal{M}(P, \Gamma) = y$ and all noises in Γ are bounded (i.e., event E holds), we follow the path starting from the root while making changes to Γ so that $\mathcal{M}(P', f(\Gamma)) = y$. Since the split operations do not depend on P, we only need to consider all the H shrink operations on that path.

For a cell c, we use $\text{count}(c)$ to denote its count in P while $\text{count}'(c)$ its count in P'. Note that $|\text{count}(c) - \text{count}'(c)| \leq 1$ for any c. We use ζ, η, $\widetilde{R}$, $\widetilde{T}$, $\ldots$ to denote the noises in Γ, and ζ', η', $\widetilde{R}'$, $\widetilde{T}'$, $\ldots$ for their counterparts in $f(\Gamma)$. The injection is defined as follows for each invocation of Algorithm 2:

1. Set $\zeta' := \zeta + \text{count}(c) - \text{count}'(c)$. Then we have $\widetilde{R}' = \widetilde{R}$, which makes the decisions at line 2 on P and P' are the same. Note that $|\zeta' - \zeta| = |\text{count}(c) - \text{count}'(c)| \leq 1$.

2. Recall the definitions of $b_c^{(\cdot)}, b_l^{(\cdot)}, b_r^{(\cdot)}, \eta^{(\cdot)}, \tilde{\theta}^{(\cdot)}$ in the proof of Lemma 3.1. Set $\widetilde{T}' := \widetilde{T} - 1$, ${\eta'}^{(i)} := \eta^{(i)}$ for $1 \leq i < k$, and ${\tilde{\theta}'}^{(k)} := \tilde{\theta}^{(k)} - 2$. Note that $|\gamma' - \gamma| \leq 2$ in order to achieve $\widetilde{T}' := \widetilde{T} - 1$ since $|\text{count}(c) - \text{count}'(c)| \leq 1$. We already have $\tilde{\theta}^{(i)} \geq \widetilde{T}$ for $1 \leq i < k$ and $\tilde{\theta}^{(k)} < \widetilde{T}$ on P. For $1 \leq i < k$, since $\tilde{\theta}^{(i)} \geq \widetilde{T}$, we have $\text{count}(b_c^{(i)}) > \frac{2}{3}\text{count}(c) - 2\ell_h > \frac{1}{2}\text{count}(c) + \ell_h$ by $|\eta^{(i)}| \leq \ell_h$, $|\gamma| \leq \ell_h$ and $\text{count}(c) \geq 36\ell_h$. This implies that the difference between $\text{count}(b_l^{(i)})$ and $\text{count}(b_r^{(i)})$ is at least ℓ_h, then the presence or absence of a single data point will not affect the choice of the majority box $b_c^{(i)}$ on line 8, thus we have $|{\tilde{\theta}'}^{(i)} - \tilde{\theta}^{(i)}| \leq 1$ by $|\text{count}(b_c^{(i)}) - \text{count}'(b_c^{(i)})| \leq 1$ and ${\eta'}^{(i)} = \eta^{(i)}$. Since $b_c^{(k-1)}$ in the invocations on P and P' are the same, we can conclude that $|\text{count}'({b_c}^{(k)}) - \text{count}(b_c^{(k)})| \leq 1$, implying that $|{\eta'}^{(k)} - \eta^{(k)}| \leq 3$ in order to achieve ${\tilde{\theta}'}^{(k)} := \tilde{\theta}^{(k)} - 2$. Given $\tilde{\theta}^{(i)} \geq \widetilde{T}$ for $1 \leq i < k$ and $\tilde{\theta}^{(k)} < \widetilde{T}$, by the above mapping, we will have ${\tilde{\theta}'}^{(i)} \geq \widetilde{T}'$ for $1 \leq i < k$ and ${\tilde{\theta}'}^{(k)} < \widetilde{T}'$. This implies that the repeat-until loop in the invocations on P and P' are the same.

3. Set $\xi_l' := \text{count}(b_l^{(k)}) + \xi_l - \text{count}'(b_l^{(k)})$, $\xi_r' := \text{count}(b_r^{(k)}) + \xi_r - \text{count}'(b_r^{(k)})$. This makes the decision on line 13 in the invocations on P and P' the same. We have $|\xi_l' - \xi_l| \leq 1$ since $|\text{count}'(b_l^{(k)}) - \text{count}(b_l^{(k)})| \leq 1$, and similarly $|\xi_r' - \xi_r| \leq 1$.

Note that the injection defined above changes the values of the noises drawn from $\text{Lap}(1/\varepsilon_h)$ by at most 8, so the ratio between $\mathsf{Pr}(\Gamma)$ and $\mathsf{Pr}(f(\Gamma))$ is bounded by $\exp(8 \cdot \varepsilon_h)$ for the invocation of Algorithm 2 at noise level h. Across all H invocations, we have

$$\mathsf{Pr}(\Gamma) \leq \exp\left(8 \cdot \sum_{h=0}^{H} \varepsilon_h\right) \cdot \mathsf{Pr}(f(\Gamma)) \leq \exp(\varepsilon/2) \cdot \mathsf{Pr}(f(\Gamma)),$$

and Eq. (1) is proved. ■

Then Theorem 3.2 follows from Lemma 3.2, 3.3, and 3.4.

□

Now we analyze the error when using the private BBD-tree to answer approximate range queries.

Theorem 3.3. *Given an (ε, δ)-differentially private BBD-tree and any α-approximate range query Q, for $n > \frac{8}{\varepsilon} \log \frac{2}{\beta}$, with probability at least $1 - \beta$, Algorithm 1 answers Q with an error of* $O\left(\varepsilon^{-1} \cdot \left(\alpha^{-d}(\log \frac{n}{\beta\delta} + \log\log u) + \log^{1.5} n \log \frac{1}{\beta} + \alpha^{-d/2} \log n \log \frac{1}{\beta}\right)\right)$.

4 Applications

4.1 Approximate nearest neighbor

In the *approximate nearest neighbor (ANN)* problem, the goal is to find a data point $p \in P$ such that $\mathrm{dist}(p, q) \leq (1 + \alpha)\mathrm{dist}(p^*, q)$ for any given query point q, where p^* is the nearest neighbor of q in P and $0 < \alpha < 1$ is the approximation ratio. In the non-private setting, there is a standard approach using BBD-tree to solve ANN problem [2]. In the private setting, however, there are two differences. First, returning a data point in P will clearly breach privacy. Thus, we must settle for a slightly weaker target, namely, we will aim to return only the approximate distance to the NN, but not the NN itself. Second, the non-private BBD-tree has no errors, while its private versions do. This introduces some technical complications. In particular, we can no longer measure the approximation ratio of the ANN compared with the nearest neighbor p^*, but the τ-th nearest neighbor. More precisely, our goal is to return a distance r such that $r_{(1)} \leq r \leq (1 + \alpha)r_{(\tau)}$, where $r_{(\tau)}$ is the distance between q and its τ-th nearest neighbor in P. We call τ the *rank error*; obviously, smaller τ means better utility.

We present a more general algorithm for approximately finding the distance between a query point q and its k-th nearest neighbor in P, given an approximate range counting synopsis with error κ for $(\alpha/10)$-approximate spherical range queries. Note that since the whole space is $[u]^d$, the smallest and the largest possible distance between two distinct points are 1 and $\sqrt{d}u$ respectively. Let $B(q, w)$ denote the open ball with radius w centered at q. From the approximate range counting synopsis, we obtain approximate counts for $|P \cap B(q, 1/2)|, |P \cap B(q, (1 + \alpha/3)/2)|, |P \cap B(q, (1 + \alpha/3)^2/2)|, \ldots, |P \cap B(q, (1 + \alpha/3)^t/2)|$, where $t = \lceil \log_{1+\alpha/3}(2\sqrt{d}u) \rceil$; denote these approximate counts as $a_0, a_1, a_2, \ldots, a_t$. Let $i = \min\{i \mid a_i > k + \kappa\}$. Then we output $r = (1 + \alpha/10)(1 + \alpha/3)^i/2$ if $i \neq 0$; otherwise we return $r = 0$.

Lemma 4.1. *For any $k \leq n - 2\kappa$, the above algorithm returns an r such that $r_{(k)} \leq r \leq (1 + \alpha)r_{(k+2\kappa)}$.*

In the private setting, we use our private BBD-trees presented in Section 3.2 and 3.3 as the approximate range counting synopsis.

Theorem 4.1. *There is an (ε, δ)-differentially private synopsis such that, with probability at least $1 - \beta$, the distance to the ANN for any query point can be found with a rank error of*

$O(\tau_{\varepsilon,\delta})$, where $t = \lceil \log_{1+\alpha/3}(2\sqrt{d}u) \rceil$, and

$$\tau_{\varepsilon,\delta} = \begin{cases} \varepsilon^{-1}\alpha^{-d/2} \log u \log \frac{t}{\beta}, & \delta = 0; \\ \varepsilon^{-1} \cdot \left(\alpha^{-d}(\log \frac{n}{\beta\delta} + \log\log u) + \log^{1.5} n \log \frac{t}{\beta} + \alpha^{-d/2} \log n \log \frac{t}{\beta}\right), & \delta > 0. \end{cases}$$

4.2 k-NN classification

In k-NN classification, we are given m classes of points $P_1, P_2, \ldots, P_m$, and i is called the label of P_i. Given a query point q, the k-NN algorithm returns the label that is most common among its k nearest neighbors in $P = P_1 \cup \cdots \cup P_m$.

To obtain a private k-NN algorithm, we release an $(\varepsilon/2, \delta/2)$-private BBD-tree for P, as well as an $(\varepsilon/2, \delta/2)$-private BBD-tree for each P_i. This preserves an overall (ε, δ)-DP, since the presence or absence of one point affects only two private BBD-trees. For a given query point q, we invoke the algorithm in Section 4.1 on P with approximation factor $\alpha/3$, obtaining an r such that $r_{(k)} \le r \le (1+\alpha/3) r_{(k+O(\tau_{\varepsilon,\delta}))}$. Next, we query the private BBD-trees with approximation factor $\alpha/10$ to obtain approximate counts for $|P_1 \cap B(q,r)|, \ldots, |P_m \cap B(q,r)|$. Finally, we return the label i with the largest estimated $|P_i \cap B(q,r)|$.

Our private k-NN algorithm has the following utility guarantee. Intuitively, it says that as long as the correct label wins the majority in the k nearest neighbors of q by a poly-logarithmic margin, while the distances measured are allowed an $(1 \pm \alpha)$-factor approximation, then the algorithm will find it with high probability.

Theorem 4.2. *There exists absolute constants C_1 and C_2 such that, with probability at least $1-\beta$, our private k-NN synopsis can be used to find the correct label i for any query point q, provided that $|P_i \cap B(q,(1-\alpha)r_{(k)})| - |P_j \cap B(q,(1+\alpha)r_{(k+C_1\cdot\tau_{\varepsilon,\delta})})| \ge C_2 \cdot \tau_{\varepsilon,\delta}$ for all $j \ne i$.*

Note that our algorithm releases a private k-NN classifier, which can be used to classify any query point. On the other hand, the private k-NN classifier in [11] only releases the classification results (with no utility guarantees) for a given set of queries, while no more queries can be accepted afterwards. Thus we solve a much more general problem than theirs.

5 Lower bounds

Observe that the errors in Theorem 3.1 and 4.1 for pure DP have an $O(\log u)$ term. In this section, we show that this is unavoidable even in one dimension. In particular, we prove the following lower bound on the one-dimensional ANN problem with a constant approximation factor $\alpha = 0.1$ and a constant failure probability $\beta = 1/3$ via a packing argument [13]. This in turn implies a lower bound for approximate range counting.

Theorem 5.1. *Any $(\varepsilon, 0)$-differentially private one-dimensional ANN algorithm that, with probability at least $2/3$, returns an r such that $r_{(1)} \le r \le 1.1 \cdot r_{(\tau)}$ for every query point q, must have a rank error $\tau = \Omega\left(\frac{1}{\varepsilon}\log u\right)$.*

Corollary 5.1. *Any $(\varepsilon, 0)$-differentially private one-dimensional approximate range counting algorithm that, with probability at least $2/3$, answers all 0.01-approximate range queries with error at most $\kappa_{\varepsilon,0}$, must have $\kappa_{\varepsilon,0} = \Omega\left(\frac{1}{\varepsilon}\log u\right)$.*

We can also show a lower bound under (ε, δ)-DP by a reduction from exact (i.e., no fuzziness) range counting [4]:

Theorem 5.2. *Any (ε, δ)-differentially private one-dimensional approximate range counting algorithm that, with probability at least $2/3$, answers all 0.01-approximate range queries with error at most $\kappa_{\varepsilon,\delta}$, must have $\kappa_{\varepsilon,\delta} = \Omega\left(\frac{1}{\varepsilon} \cdot (\log^* u) \cdot \log(1/\delta)\right)$, for $\exp(-\varepsilon n/\log^* n) \le \delta \le 1/n^2$.*

Note that this lower bound means that (1) some dependency on u is inevitable even under (ε, δ)-DP, justifying the restriction that the points have integer coordinates; and (2) a logarithmic dependency on n/δ is also necessary.

References

[1] Sunil Arya and David M Mount. Approximate range searching. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 172–181, 1995.

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

[3] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 609–618, 2008.

[4] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially private release and learning of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 634–649. IEEE, 2015.

[5] TH Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *International Colloquium on Automata, Languages, and Programming*, pages 405–417. Springer, 2010.

[6] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*, pages 20–31. IEEE, 2012.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[8] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.

[9] Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N Rothblum. Pure differential privacy for rectangle queries via private partitions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 735–751. Springer, 2015.

[10] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[11] Mehmet Emre Gursoy, Ali Inan, Mehmet Ercan Nergiz, and Yucel Saygin. Differentially private nearest neighbor classification. *Data Mining and Knowledge Discovery*, 31(5):1544–1575, 2017.

[12] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE Annual Symposium on Foundations of Computer Science*, pages 61–70. IEEE, 2010.

[13] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 705–714, 2010.

[14] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

[15] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. A data-and workload-aware algorithm for range queries under differential privacy. *Proceedings of the VLDB Endowment*, 7(5):341–352, 2014.

[16] J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer-Verlag, Berlin, 1999.

[17] Shanmugavelayutham Muthukrishnan and Aleksandar Nikolov. Optimal private halfspace counting via discrepancy. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1285–1292, 2012.

[18] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360, 2013.

[19] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *2013 IEEE 29th international conference on data engineering (ICDE)*, pages 757–768. IEEE, 2013.

[20] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment*, 6(14):1954–1965, 2013.

[21] Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

[22] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering*, 23(8):1200–1214, 2010.

[23] Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*, pages 155–170, 2016.

A Proof of Theorem 3.1

Proof. We need the following technical lemma.

Lemma A.1. *(Measure Concentration [5]) Let X_i's be independent random variables drawn from Lap(λ_i), then for any* $0 < \beta < 1$,

$$\Pr\left[\left|\sum_i X_i\right| > 4\sqrt{\sum_i \lambda_i^2 \log\frac{2}{\beta}}\right] \le \beta.$$

As mentioned, there are two sources of errors: the noisy counts from line 2 and the leaf nodes from line 3 in Algorithm 1. Since the split operation terminates only when the size of the cell is 1, which means that the precise location of the points in the cell is known, the condition in line 3 is actually never true[4]. Thus, it suffices to only account for the noises from line 2.

Consider any cell c whose noisy count has been added in line 2, and denote its parent cell as parent(c). Observe that parent(c) must intersect both Q_α^- and the complement of Q_α^+ (otherwise we would not have visited c), which means that size(parent$(c)) = \Omega(\alpha w)$. Because size$(c) \geq \frac{1}{2}$size(parent(c)), we have size$(c) = \Omega(\alpha w)$ as well. In addition, all cells whose noisy counts have been added in line 2 must be disjoint. Then Theorem 2.4 states that there are $O((w/\alpha w)^d) = O(\alpha^{-d})$ such cells. Each cell has an noise that is independently drawn from $\mathrm{Lap}(d \log u/\varepsilon)$. By Lemma A.1, the absolute value of the aggregated Laplace noise is $O\left(\varepsilon^{-1}\alpha^{-d/2} \log u \log \frac{1}{\beta}\right)$ with probability at least $1-\beta$.

Let $T := 2\varepsilon^{-1} d \log u \log \frac{n}{\beta}$. During the construction of the private space decomposition, which is a full binary tree, we can stop subdividing a node when its noisy count (perturbed by a Laplace noise drawn from $\mathrm{Lap}(2d\log u/\varepsilon)$) is below $2T$; whenever the number of nodes exceeds n we report failure, otherwise after the construction each node is associated with a refreshed noisy count (perturbed by a fresh Laplace noise drawn from $\mathrm{Lap}(2d\log u/\varepsilon)$). By the basic composition theorem, this procedure preserves $(\varepsilon, 0)$-DP. The error due to line 2 is the same as in the last paragraph. Next, we analyze the error due to line 3. For n Laplace noises used during the construction, by a union bound they have bounded absolute value T with probability at least $1-\beta$. Conditioned upon this event in the following analysis. Then all nodes, except leaf nodes, contain at least T data points (otherwise the stopping condition will be true). In addition, the nodes on each level are disjoint, so across all levels there are at most $n/T \cdot \log u < n$ nodes. Moreover, each leaf node has $O(T)$ data points since the stopping condition is true. And because there are $O(\alpha^{-d})$ leaf nodes from line 3, the error due to line 3 is $O\left(\alpha^{-d}T\right)$. □

B Proof of Theorem 3.3

Proof. As in the proof of Theorem 3.1, the error comes from the noisy counts from line 2 and the leaf nodes whose size is greater than 1 from line 3 in Algorithm 1. Denote the former set of nodes as C_1 and the latter C_2. Assume the event that $n \leq \tilde{n} \leq 2n$ holds in the following analysis, which happens with probability at least $1-\beta/2$ due to the exponential tail of the Laplace distribution.

Each node in C_1 has noise independently drawn from $\mathrm{Lap}(40\log\tilde{n}/\varepsilon)$. By Lemma A.1, the aggregated noise has bounded absolute value $O\left(\sqrt{|C_1|}\log\frac{1}{\beta}\log n/\varepsilon\right)$ with probability at least $1-\beta/4$. To bound $|C_1|$, consider their parent cells. As argued in the proof of Theorem 3.1, their parent cells have size at least $\Omega(\alpha w)$, but not necessarily disjoint. So there are $O\left(\log n + (1/\alpha)^d\right)$ of them by Theorem 2.4. $|C_1|$ is at most twice this size.

By Theorem 3.2, with probability at least $1-\beta/4$, the number of points lying in every leaf cell in C_2 is $O\left(\varepsilon^{-1}\cdot\left(\log\frac{n}{\beta\delta} + \log\log u\right)\right)$. Each leaf cell in C_2 has size $\Omega(\alpha w)$ because it must intersect both Q_α^- and the complement of Q_α^+. Furthermore, all leaf cells are disjoint. Thus, by Theorem 2.4, we have $|C_2| = O(\alpha^{-d})$.

Then the theorem follows by applying a union bound and summing these two parts of error. □

[4]When checking the conditions in line 1 and 2 of Algorithm 1, we use the precise location of the points inside the cell if the cell size is 1.

C Proof of Lemma 4.1

Proof. We know such i should exist, since $a_t \geq n - \kappa \geq k + \kappa$. First assume $i \neq 0$. Given $a_i \geq k + \kappa$, by the definition of approximate range counting, we have $|P \cap B(q, (1+\alpha/3)^i/2)^+_{\alpha/10}| \geq a_i - \kappa \geq k$, implying that

$$r = (1+\alpha/10)(1+\alpha/3)^i/2 \geq r_{(k)}.$$

Given $a_{i-1} \leq k + \kappa$, we have $|P \cap B(q, (1+\alpha/3)^{i-1}/2)^-_{\alpha/10}| \leq a_{i-1} + \kappa \leq k + 2\kappa$, meaning $r_{(k+2\kappa)} \geq (1-\alpha/10)(1+\alpha/3)^{i-1}/2$. Thus

$$r = (1+\alpha/10)(1+\alpha/3)^i/2 \leq (1+\alpha)(1-\alpha/10)(1+\alpha/3)^{i-1}/2 \leq (1+\alpha)r_{(k+2\kappa)}.$$

For the special case $i = 0$, we know $r_{(k)} = 0$. Then outputting $r = 0$ is fine. □

D Proof of Theorem 4.1

Proof. The privacy guarantee simply follows from the post-processing property of differential privacy. Now we argue for the rank error. For any query point, there are t spherical range queries asked by the algorithm above. When $\delta = 0$, setting the failure probability in Theorem 3.1 to β/t and applying a union bound, the error guarantee is $O(\tau_{\varepsilon,\delta})$ which holds for t spherical range queries with probability at least $1-\beta$. When $\delta > 0$, we analyze the errors due to line 2 and line 3 in Algorithm 1 as in the proof of Theorem 3.3. By Theorem 3.2, with probability at least $1-\beta/2$, every leaf cell c in the private BBD-tree has either size$(c) = 1$ or count$(c) = O(\varepsilon^{-1} \cdot (\log \frac{n}{\beta\delta} + \log\log u))$, then the error due to line 3 in Algorithm 1 is $O(\varepsilon^{-1}\alpha^{-d}(\log \frac{n}{\beta\delta} + \log\log u))$ which holds for all range queries. By Lemma A.1 and applying a union bound, the error due to line 2 in Algorithm 1 is $O(\log^{1.5} n \log \frac{t}{\beta} + \alpha^{-d/2} \log n \log \frac{t}{\beta})$ which holds for t spherical range queries with probability at least $1-\beta/2$. Thus, the error guarantee is $O(\tau_{\varepsilon,\delta})$ which holds for t spherical range queries with probability at least $1-\beta$. Then we run the algorithm above with $k = 1$, which leads to a rank error of $O(\tau_{\varepsilon,\delta})$ by Lemma 4.1. □

E Proof of Theorem 4.2

Proof. By Theorem 4.1, there exists an absolute constant C_1 such that $r_{(k)} \leq r \leq (1+\alpha/3)r_{(k+C_1\cdot\tau_{\varepsilon,\delta})}$. Let a_i denote the approximate count for $|P_i \cap B(q,r)|$ for $i = 1, 2, \ldots, m$. As argued in the proof of Theorem 4.1, the error guarantee is $C_2' \cdot \tau_{\varepsilon,\delta}$ which holds for $a_1, a_2, \ldots, a_m$ with probability at least $1-\beta$, where C_2' is an absolute constant. Taking $C_2 = 2C_2'$, then since $|P_i \cap B(q, (1-\alpha)r_{(k)})| - |P_j \cap B(q, (1+\alpha)r_{(k+C_1\cdot\tau_{\varepsilon,\delta})})| \geq C_2 \cdot \tau_{\varepsilon,\delta}$ for all $j \neq i$, we have

$$\begin{aligned}
a_i &\geq |P_i \cap B(q, (1-\frac{\alpha}{10})r)| - C_2' \cdot \tau_{\varepsilon,\delta} \\
&\geq |P_i \cap B(q, (1-\alpha)r_{(k)})| - C_2' \cdot \tau_{\varepsilon,\delta} \\
&\geq |P_j \cap B(q, (1+\alpha)r_{(k+C_1\cdot\tau_{\varepsilon,\delta})})| + C_2' \cdot \tau_{\varepsilon,\delta} \\
&\geq |P_j \cap B(q, (1+\frac{\alpha}{10})(1+\frac{\alpha}{3})r_{(k+C_1\cdot\tau_{\varepsilon,\delta})})| + C_2' \cdot \tau_{\varepsilon,\delta} \\
&\geq |P_j \cap B(q, (1+\frac{\alpha}{10})r)| + C_2' \cdot \tau_{\varepsilon,\delta} \\
&\geq a_j
\end{aligned}$$

Then the theorem follows. □

F Proof of Theorem 5.1

Proof. Let $\mathcal{A}$ be the purported ANN algorithm with error τ. We first construct the hard point sets and queries. The set of queries is $Q = \{q = 1 + 10i \mid i = 2, 3, \ldots, t\}$ where $t = \lfloor u/10 \rfloor - 1$. Then we construct the same number of point sets P_i, $i = 2, 3, \ldots, t$. These point sets are similar but they will have well-separated answers for Q. Specifically, P_i consists of $n - \tau$ points at coordinate 1 and τ points at $10i$. Note that for $j \neq i$, P_j and P_i only differ by τ points, while $\mathcal{A}$ must return different distances on the query $q = 1 + 10j$ to guarantee an error of at most τ. In particular, the answer must be between 1 and $1 + 0.1$ on P_j while it should be least 9 on P_i.

Next, we use a packing argument. Let $\mathcal{A}(P, Q) \in \mathbb{R}^{|Q|}$ be the answers for Q when applying $\mathcal{A}$ on P. Let $\mathcal{P}_i \subseteq \mathbb{R}^{|Q|}$ be all the allowed answers (up to error τ) for Q on each data set P_i, then we have $\mathsf{Pr}[\mathcal{A}(P_i, Q) \in \mathcal{P}_i] \geq 2/3$ for every i according to the probability guarantee of $\mathcal{A}$. Also, as argued above, all the $\mathcal{P}_i$'s must be disjoint. Now we have

$$\begin{aligned}
1 &\geq \sum_{i=1}^{t} \mathsf{Pr}[A(P_1, Q) \in \mathcal{P}_i] \\
&\geq \sum_{i=1}^{t} e^{-\tau\varepsilon} \cdot \mathsf{Pr}[A(P_i, Q) \in \mathcal{P}_i] \\
&\geq \sum_{i=1}^{t} \frac{2}{3} e^{-\tau\varepsilon} \\
&\geq \frac{u}{20} e^{-\tau\varepsilon}.
\end{aligned}$$

The first inequality follows from the fact that all the $\mathcal{P}_i$'s are disjoint. The second inequality is according to the group privacy property of DP (Theorem 2.3), since any pair of our constructed data sets differ at most τ points. The third inequality follows from the accuracy guarantee of the given algorithm $\mathcal{A}$. Then we conclude that $\tau = \Omega\left(\frac{1}{\varepsilon}\log u\right)$ after rearranging. □

G Proof of Corollary 5.1

Proof. By the reduction in Section 4.1, any 0.01-approximate range counting algorithm with error κ can be used to construct an 0.1-approximate ANN algorithm with error $O(\kappa)$. Then the corollary follows. □

H Proof of Theorem 5.2

Proof. We will show that any dataset P can be transformed to another dataset $T(P)$ such that the answer of any threshold query over P is the same as the answer of an 0.01-approximate threshold query over $T(P)$. The transformation T is defined as mapping each integer x in P to 2^x in $T(P)$. Note that the universe size has increased from u to $U := 2^u$ after the transformation. Then for any threshold query $[1, y]$ over P, its answer is the same as the 0.01-approximate threshold query $[-1.5 \cdot 2^y, 1.5 \cdot 2^y]$ over $T(P)$, because: (1) for any point x in P lying inside $[1, y]$, we have $2^x \leq 2^y < 1.5 \cdot 2^y - 0.03 \cdot 2^y$ and $2^x \geq 2^1 > -1.5 \cdot 2^y + 0.03 \cdot 2^y$; (2) for any point z in P lying outside $[1, y]$, we have $2^z \geq 2^{y+1} > 1.5 \cdot 2^y + 0.03 \cdot 2^y$.

Therefore, by the reduction from exact range counting to approximate range counting, along with the existing lower bound of $\Omega((\log^* u) \cdot \log(1/\delta)/\varepsilon)$ for exact range counting proved in [4], we have $\kappa_{\varepsilon,\delta} = \Omega((\log^* u) \cdot \log(1/\delta)/\varepsilon) = \Omega((\log^* \log U) \cdot \log(1/\delta)/\varepsilon) = \Omega((\log^* U) \cdot \log(1/\delta)/\varepsilon)$. □