Efficient Top-k Indexing via General Reductions

Saladi Rahul
University of Minnesota
USA
sala0198@umn.edu

Yufei Tao
University of Queensland
Australia
taoyf@itee.uq.edu.au

ABSTRACT

Let D be a set of n elements each associated with a real-valued weight, and $\mathbb{Q}$ be the set of all possible predicates allowed on those elements. Given a predicate in $\mathbb{Q}$ and integer k, a *top-k query* returns the k elements with the largest weights among the elements of D satisfying q. The corresponding data structure problem aims to store D in small space to allow every query to be answered efficiently. It is already known that, before settling the problem, one *must* be able to solve two degenerated accompanying problems: (i) *prioritized reporting*: given a predicate $q \in \mathbb{Q}$ and a real value τ, return all the elements of D satisfying q and having weights at least τ; (ii) *max reporting*: top-k queries with k fixed to 1.

In this paper we prove general reductions in external memory that explore the opposite direction. Our first reduction shows that, (under mild conditions) any prioritized reporting structure yields a static top-k structure with only a slow-down in query time by a factor of $O(\log_B n)$, where B is the block size. Our second reduction shows that if one additionally has a max reporting structure, then combining the two structures yields a top-k structure with no performance slow down (in space, query, and update) in expectation. These reductions significantly simplify the design of top-k structures, as we showcase on numerous problems including halfspace reporting, circular reporting, interval stabbing, point enclosure, and 3d dominance. All the techniques proposed work directly in the RAM model as well.

Keywords

Top-k; Reductions; Data Structures; Algorithms; External Memory; RAM

1. INTRODUCTION

Reporting queries—which constitute a main type of operations in database systems—can be abstracted at a high level as follows. The input is a set D of n elements drawn from a certain domain $\mathbb{D}$. Let $\mathbb{Q}$ be the set of all possible *predicates* that are allowed on the elements of $\mathbb{D}$. A *query* chooses a predicate $q \in \mathbb{Q}$, and reports all the elements $e \in D$ such that e satisfies q—we denote the set of such elements as $q(D)$.

The result size $|q(D)|$ of a reporting query is often exceedingly large, especially in today's big-data era. In many applications, on the other hand, a user is interested only in those elements of $q(D)$ with high "priorities". This motivates the augmentation of each element e in D with a *weight* $w(e) \in \mathbb{R}$ (where $\mathbb{R}$ represents the real domain). Accordingly, every reporting query spawns two variants:

- **Prioritized Reporting:** Given a predicate $q \in \mathbb{Q}$ and a real value τ, a query reports all the elements $e \in q(D)$ such that $w(e) \geq \tau$.
- **Top-k Reporting:** Given a predicate $q \in \mathbb{Q}$ and an integer $k \geq 1$, a query reports the k elements in $q(D)$ with the highest weights; specially, if $|q(D)| < k$, then the entire $q(D)$ is reported.

For each query type, the corresponding indexing problem aims to store D in a data structure of small space to guarantee attractive query efficiency. Note that a structure for prioritized reporting must support queries with any q and τ, while a structure for top-k reporting must support queries with any q and k.

1.1 Computation Models and Math Conventions

We will carry out most of our analysis in the standard external memory (EM) model [8]. A machine is equipped with M words of memory, and a disk that has been formatted into *blocks* of B words each (this paper assumes $B \geq 64$). The values of M and B satisfy $M \geq 2B$. An I/O either reads a disk block into memory, or writes B words of memory into a disk block. The *time* of an algorithm is measured in the number of I/Os performed, while the *space* of a structure is measured in the number of disk blocks occupied.

By setting M and B to appropriate constants, all our EM results also hold in the classic RAM model, whose definition can be found in almost every textbook on algorithms, e.g., [16].

We consider that each element is stored in $O(1)$ words, and that the elements of D have distinct weights, as is a standard assumption in the top-k literature (whose purpose is to avoid ambiguity in the top-k result caused by tie-breaking in weights).

All complexities by default hold in the worst case. We

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

PODS'16, June 26-July 01, 2016, San Francisco, CA, USA

© 2016 ACM. ISBN 978-1-4503-4191-2/16/06...$15.00

DOI: http://dx.doi.org/10.1145/2902251.2902290

(i) define $\log^*(n)$ as the number of times that we need to perform $\log_2(.)$ on n to get a value no more than 2, and (ii) use notation $\tilde{O}(.)$ to hide a factor polylogarithmic to n when such a factor is insignificant. All the logarithm bases are equal to 2 if omitted.

Finally, we say that a function $f(n)$ is *geometrically converging* if it satisfies two conditions:

- For any $n \geq B$:

$$\sum_{i=0}^{h} f\left(\frac{n}{c^i}\right) \quad = \quad O(f(n))$$

  for any value $c \geq 2$, where h is the largest integer i satisfying $n/c^i \geq B$.

- For any $n < B$, $f(n) = O(1)$.

1.2 Motivation

Recent years have witnessed considerable research aiming to design top-k data structures for a large variety of problems (see Section 2 for a survey), which differ in their concrete specifications of $\mathbb{D}$ and $\mathbb{Q}$. Most solutions, interestingly, share a common two-step process:

- First, design a structure for prioritized reporting.
- Second, reduce top-k reporting to prioritized reporting.

Typically, the prioritized-reporting structure is easy to obtain (if not readily available), such that the challenge lies in the second step.

The community has come to realize that this process is *necessary*. Specifically, it has been shown [26, 28, 29] that *prioritized reporting can be reduced to top-k reporting*. Suppose that there is a structure that consumes $\mathcal{S}_{top}(n)$ space on n elements, and answers a top-k query in $\mathcal{Q}_{top}(n) + O(k/B)$ I/Os. Then, there is a prioritized-reporting structure of $\mathcal{S}_{pri}(n)$ space that answers a query in $\mathcal{Q}_{pri}(n) + O(t/B)$ I/Os—where t is the number of reported elements—such that

$$\begin{aligned} \mathcal{S}_{pri}(n) &= O(\mathcal{S}_{top}(n)) \\ \mathcal{Q}_{pri}(n) &= O(\mathcal{Q}_{top}(n)). \end{aligned}$$

The reduction does not rely on the underlying problems, i.e., prioritized reporting is no harder than top-k reporting, regardless of $\mathbb{D}$ and $\mathbb{Q}$. Therefore, if one does not even have a structure for the former, there is no hope for the latter.

What is much more intriguing is the opposite: *how much harder is top-k reporting than prioritized reporting*? To phrase this differently, let us assume that we already have a structure of $\mathcal{S}_{pri}(n)$ space that answers a prioritized query in $\mathcal{Q}_{pri} + O(t/B)$ time. We want to use the structure as a *black box* to design a top-k structure of space $\mathcal{S}_{top}(n)$ and query cost $\mathcal{Q}_{top} + O(k/B)$. The question is how good the functions $\mathcal{S}_{top}(n)$ and $\mathcal{Q}_{top}(n)$ can be.

Resolving the question has an important technical implication. Since prioritized reporting structures are already known for numerous problems, such a black-box reduction will give us top-k structures *immediately* for all those problems! Unfortunately, not much progress has been made towards this direction. The best understanding is the rather primitive relationships below[1] [28]:

$$\mathcal{S}_{top}(n) = O(\mathcal{S}_{pri}(n)) \quad (1)$$

$$\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n)\log_2 n) + O\left(\frac{k}{B}\log_2 n\right) \quad (2)$$

The multiplicative term $\log_2 n$ on k/B essentially prevents the reduction from producing any structure with linear output-sensitive cost $O(k/B)$.

Ideally, we would like to have $\mathcal{S}_{top}(n) = O(\mathcal{S}_{pri}(n))$ and $\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n))$—in order words, no performance deterioration—which would prove the asymptotic equivalence of prioritized and top-k reporting. If such a strong reduction still remains elusive, the next obvious question is what other "assisting problem" needs to be settled to make a no-deterioration reduction possible.

1.3 Our Results 1: Reductions

Our first contribution is to show that, under mild conditions, there only needs to be an $O(\log_B n)$ gap in the query cost between top-k and prioritized reporting:

THEOREM 1. Suppose that there is a prioritized structure of $\mathcal{S}_{pri}(n)$ space and query cost $\mathcal{Q}_{pri}(n) + O(t/B)$ such that $\mathcal{S}_{pri}(n)$ is geometrically converging, and

$$\mathcal{Q}_{pri}(n) \quad \geq \quad \log_B n.$$

Furthermore, suppose that the problem is *polynomially bounded*, namely, for any input D of n elements, there are only $n^{O(1)}$ distinct outcomes for $q(D)$ over all the possible predicates $q \in \mathbb{Q}$.

Then, there is a top-k structure of space $\mathcal{S}_{top}(n)$ and query time $\mathcal{Q}_{top}(n) + O(k/B)$ with

$$\mathcal{S}_{top}(n) = O(\mathcal{S}_{pri}(n)) \quad (3)$$

$$\mathcal{Q}_{top}(n) = O\left(\mathcal{Q}_{pri}(n) \cdot \frac{\log n}{\log B + \log \frac{\mathcal{Q}_{pri}(n)}{\log_B n}}\right) \quad (4)$$

It is worth mentioning that most of the known reporting problems are polynomially bounded. Consider, for example, D to be a set of n points in $\mathbb{R}^2$. In *halfspace reporting*, given a halfspace q (the region on one side of a line), we want to report the set $q(D)$ of points in $D \cap q$. It is easy to see that $O(n^2)$ different subsets $q(D)$ exist, ranging over all possible q, because there are only $\binom{n}{2}$ different lines passing two points in D.

Two remarks are in order:

- Since the denominator in (4) is at least $\log B$, it holds that $\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n) \cdot \log_B n)$.
- If $\mathcal{Q}_{pri}(n) \geq (n/B)^\epsilon$ for an arbitrarily small constant $\epsilon > 0$, (4) becomes $\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n))$. In other words, top-$k$ reporting is asymptotically as difficult as prioritized reporting for "hard" queries.

Our techniques for establishing Theorem 1 are drastically different from those behind (1) and (2), which are obtained by binary search on the weight threshold τ [28]. We prove

[1] Assuming that $\mathcal{S}_{pri}(n)$ is geometrically converging.

Theorem 1 mainly with two new ideas. The first one is to show the existence of a core-set that allows us to process queries with large k through prioritized reporting. The second one is a recursive mechanism that "fine-tunes" the results of queries with small k using nested core-sets.

An important special case of top-k reporting is *max reporting* where all queries have $k = 1$. Before solving queries of all k, one must at least be able to design an efficient structure for max reporting. In other words, just like prioritized reporting, max reporting is also a *necessary* step towards settling top-k reporting. Our second contribution is to show that the two necessary structures are *sufficient* as well:

THEOREM 2. Suppose that there is

- A prioritized-reporting structure of $\mathcal{S}_{pri}(n)$ space that answers a query in $\mathcal{Q}_{pri}(n) + O(t/B)$ I/Os;
- A max-reporting structure of $\mathcal{S}_{max}(n)$ space that answers a query (i.e., $k = 1$) in $\mathcal{Q}_{max}(n)$ I/Os. It is required that
  - $\mathcal{S}_{max}(n) = O(n^2/B)$ for any $n \geq B$.
  - $\mathcal{S}_{max}(n)$ is geometrically converging.

Then, there is a top-k structure of expected space $\mathcal{S}_{top}(n)$ and expected query time $\mathcal{Q}_{top}(n) + O(k/B)$ with

$$\mathcal{S}_{top}(n) = O\left(\mathcal{S}_{pri}(n) + \mathcal{S}_{max}\left(\frac{6n}{B \cdot Q_{pri}(n)}\right)\right) \quad (5)$$

$$\mathcal{Q}_{top}(n) = O\left(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n)\right). \quad (6)$$

Furthermore, if the prioritized and max structures support an update in $\mathcal{U}_{pri}(n)$ and $\mathcal{U}_{max}(n)$ I/Os respectively, then the top-k structure supports an update in $O(\mathcal{U}_{pri}(n) + \mathcal{U}_{max}(n))$ expected I/Os. If any of $\mathcal{U}_{pri}(n)$ and $\mathcal{U}_{max}(n)$ is amortized, the update cost of the top-k structure is amortized expected. In the above, every expectation is taken over the random choices made by our algorithms.

Three remarks are in order:

- The above reduction is optimal in the sense that there is no performance degradation (in expectation): the space, query, and update costs of the top-k structure are all determined by the worse between the prioritized and max structures.
- Somewhat surprisingly, $\mathcal{S}_{top}(n)$ may even be smaller than $O(\mathcal{S}_{max}(n))$. For instance, consider $\mathcal{S}_{pri}(n) = O(n/B)$, $\mathcal{S}_{max}(n) = O((n/B)\log_B n)$, while $\mathcal{Q}_{pri}(n) \geq \log_B n$—which will give $\mathcal{S}_{top}(n) = O(n/B)$. This implies, interestingly, that one does not need to try very hard to minimize the space of the max structure: our reduction comes with "bootstrapping power" that automatically reduces the expected space of the final top-k structure.
- The condition $\mathcal{S}_{max}(n) = O(n^2/B)$ essentially captures all the max structures useful in practice. The power 2 is not compulsory at all; it can be replaced by any larger constant by adjusting the other constants in our analysis appropriately.

We establish Theorem 2 by sampling from the input set D so that the element with the highest weight in the sample set is approximately the k-th in D. This, at the conceptual level, may be reminiscent of a method by Aronov and Har-Peled [9] that reduces approximate counting to *emptiness queries*. However, our approach differs substantially in both algorithmic and technical details, a quick proof of which is the following fact: the counting structure of [9] suffers from performance degradation by a logarithmic factor compared to the emptiness structure, while as pointed out earlier Theorem 2 incurs no performance degradation.

Both theorems are applicable in RAM by setting B to appropriate constants.

1.4 Our Results 2: New Top-k Structures

The reduction results in the preceding subsection lead to top-k structures for a large number of problems. We discuss several representatives that (i) are of immediate interest to database systems, (ii) are either new or improve previous work, and (iii) nicely demonstrate the applications of Theorems 1 and 2.

Halfspace and Circular Range Reporting. In *halfspace reporting*, $\mathbb{D}$ is the set of points in $\mathbb{R}^d$, where d is a fixed integer. Each predicate in $\mathbb{Q}$ specifies a halfspace, i.e., all the $\boldsymbol{x}$ satisfying $\boldsymbol{x} \cdot \boldsymbol{q} \geq c$, where $\boldsymbol{q}$ and c are the query parameters ($\boldsymbol{x}$ and $\boldsymbol{q}$ are d-dimensional vectors, and c a real value). An element $e \in \mathbb{D}$ satisfies the predicate if e falls in the halfspace. The importance of halfspace reporting—in general, searching with linear constraints—has long been recognized in the database community; e.g., see [6]. We obtain:

THEOREM 3. For top-k halfspace reporting:

- When $d = 2$, there is a RAM structure of $O(n \log n)$ space and $O(\log n + k)$ query time, both in expectation.
- When $d \geq 4$, there is a RAM structure of $O(n \log n)$ space and $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor}) + O(k)$ query time, both in the worst case.
- When $d \geq 4$, there is an EM structure of $O(n/B)$ space and $O((n/B)^{1-1/\lfloor d/2 \rfloor + \epsilon} + k/B)$ query time, both in the worst case, where $\epsilon > 0$ is an arbitrarily small constant.

The first bullet compares favorably to a structure obtained by combining [5], [15], and [28] that uses $O(n \log n)$ space, and answers a query in $O(\log^2 n + k)$ time. For $d \geq 4$, [28] gave a RAM structure that, when matching our query time, requires $O(n^{1+\epsilon})$ space for some constant $\epsilon > 0$; no EM structures were known previously.

Circular range reporting is a closely related problem. Again, $\mathbb{D}$ is the set of points in $\mathbb{R}^d$, but each predicate in $\mathbb{Q}$ specifies a ball in $\mathbb{R}^d$, i.e., all the $\boldsymbol{x}$ satisfying $dist(\boldsymbol{x}, \boldsymbol{q}) \leq r$, where $dist$ is the Euclidean distance between vectors $\boldsymbol{x}$ and $\boldsymbol{q}$, and r is a positive real value (query parameters are $\boldsymbol{q}$ and r). Circular reporting is fundamental in spatial databases and similarity retrieval; e.g., see [36]. By the standard "lifting trick" [17], we obtain directly from Theorem 3:

COROLLARY 1. For top-k circular reporting with $d \geq 3$, there is:

- A RAM structure of $O(n \log n)$ space and $\tilde{O}(n^{1-1/\lfloor (d+1)/2 \rfloor}) + O(k)$ query time, both in the worst case.
- An EM structure of $O(n/B)$ space and $O((n/B)^{1-1/\lfloor (d+1)/2 \rfloor + \epsilon} + k/B)$ query time, both in the worst case, where $\epsilon > 0$ is an arbitrarily small constant.

The RAM structure improves a structure of [28], which again requires $O(n^{1+\epsilon})$ space to match our query time. No EM structures were known for this problem.

Interval Stabbing. This problem is among the most classic problems in the database area; e.g., see [21]. Here, $\mathbb{D}$ is the set of all intervals in $\mathbb{R}$, namely, $\mathbb{D} = \{[x, y] \mid x, y \in \mathbb{R}\}$. Each predicate in $\mathbb{Q}$ specifies a point q, such that an element $e = [x, y]$ in $\mathbb{D}$ satisfies the predicate if $q \in [x, y]$. We obtain:

THEOREM 4. *For top-k interval stabbing, there is an EM structure of*

- *$O(n/B)$ space and $O(\log_B n + k/B)$ query time, both in expectation. The structure can be updated in $O(\log_B n)$ I/Os amortized expected per insertion and deletion.*
- *$O(n/B)$ space and $O(\log_B^2 n + k/B)$ query time, both in the worst case.*

The second bullet improves a structure of [28] that uses $O((n/B) \log n)$ space to ensure query time $O(\log n \cdot \log_B n + k/B)$.

2D Point Enclosure. In this problem, $\mathbb{D}$ is the set of all rectangles in $\mathbb{R}^2$, namely, $\mathbb{D} = \{[x_1, x_2] \times [y_1, y_2] \mid x_1, x_2, y_1, y_2 \in \mathbb{R}\}$. Each predicate in $\mathbb{Q}$ specifies a point $q \in \mathbb{R}^2$, such that an element $e \in \mathbb{D}$ satisfies the predicate if $q \in e$.

To emphasize on the relevance of a top-k query of this type to databases, let us consider a dating website, where a person registers requirements on her/his ideal significant other: age in $[x_1, x_2]$, and height in $[y_1, y_2]$. A reasonable top-k point enclosure query from, say, a lady is:

> "Find the 10 gentlemen with the highest salaries such that my age and height fall into their preferred ranges."

We obtain:

THEOREM 5. *For top-k point enclosure, there is a RAM structure of*

- *$O(n \log^*(n))$ space and $O(\log n \log \log n + k)$ query time, both in expectation.*
- *$O(n \log^*(n))$ space and $O(\log^2 n \frac{\log \log n}{\log \log \log n} + k)$ query time, both in the worst case.*

Both structures compare favorably to a structure obtained by combining [28] and [13] that uses $O(n \log n)$ space, and answers a query in $O(\log^2 n + k)$ time.

3D Dominance. In this problem, $\mathbb{D}$ is the set of points in $\mathbb{R}^3$. Each predicate in $\mathbb{Q}$ specifies a point $q = (x, y, z)$, such that an element $e = (e_x, e_y, e_z)$ in $\mathbb{D}$ satisfies the predicate if $e_x \leq x$, $e_y \leq y$, and $e_z \leq z$. A practical top-k query of this type is

> "Find the 10 best-rated hotels whose (i) prices are at most x dollars per night, (ii) distances from the town center are at most y km, and (iii) security rating is at least z."

We obtain:

THEOREM 6. *For top-k 3D dominance, there is a RAM structure of $O(n \log n / \log \log n)$ space and $O(\log^{1.5} n + k)$ query time, both in expectation.*

This compares favorably to a structure obtained by combining [28] and [5] that uses $O(n \log n)$ space and $O(\log^2 n + k)$ query time.

2. PREVIOUS WORK

The existing research on top-k can be classified into two categories: (i) general reductions (to which this paper belongs) and (ii) exploration of individual problems.

The first category, to our knowledge, contains only the work by Rahul and Janardan [28]. Besides the reduction giving (1) and (2), they also showed that top-k reporting can be converted to approximate counting[2] and conventional reporting queries. Specifically, consider a reporting problem defined by $\mathbb{D}$ and $\mathbb{Q}$ as formulated in Section 1. Given a predicate $q \in \mathbb{Q}$, an *approximate counting* query returns a value that is guaranteed to be between $|q(D)|$ and $c|q(D)|$ where $c > 1$ is a constant integer fixed for all queries. Suppose that we have:

- A structure that uses space $\mathcal{S}_{rep}(n)$ and answers a query in the original (unweighted) problem in $\mathcal{Q}_{rep}(n) + O(t/B)$ time, where t is the number of elements reported;
- A structure that uses $\mathcal{S}_{cnt}(n)$ space and answers an approximate counting query in $\mathcal{Q}_{cnt}(n)$ time.

Then, there is a top-k structure with space $\mathcal{S}_{top}(n)$ and query time $\mathcal{Q}_{top}(n) + O(k/B)$ such that

$$\begin{aligned} \mathcal{S}_{top}(n) &= O((\mathcal{S}_{rep}(n) + \mathcal{S}_{cnt}(n)) \log_2 n) \\ \mathcal{Q}_{top}(n) &= O((\mathcal{Q}_{rep}(n) + \mathcal{Q}_{cnt}(n)) \log_2 n). \end{aligned}$$

The reduction also works in RAM by setting B to an appropriate constant. The competing results in Section 1.4 are obtained from the above reduction by plugging in suitable reporting and counting structures.

[2] [28] requires *exact* counting; our discussion here in fact improves that of [28].

Regarding the second category, [20] is an excellent survey on top-k research that focuses on system implementation that does not necessarily have attractive performance guarantees. In the theory line, the work of [10] appears to be the first attempt to incorporate top-k features into conventional reporting queries. Since then, the topic has grown into a sizable literature. The most extensively studied (and hence, the best understood) problem is *top-k range reporting*, whose 1D version was studied in [3, 11, 12, 33, 35], and 2D (orthogonal) version in [28, 29]. See also [25] for a colored version of the problem in 1D. The work [30] investigated more sophisticated colored top-k versions of several computational geometry problems. Finally, top-k queries on text retrieval problems have been considered in [19, 24, 25, 32]; see also a recent survey [22].

3. REDUCTION WITH WORST-CASE EFFICIENCY

This section serves as a proof of Theorem 1. We will need the Chernoff bounds given in the appendix.

3.1 Top-k Core-Set

Rank Sampling. Let S be a set of elements. By independently sampling each element of S with probability p, we obtain a *p-sample set* R. Furthermore, let us further assume that the elements of S are distinct, and drawn from an ordered domain. We say that an element $e \in S$ has *rank* i, if e is the i-th greatest in S. Intuitively, given an integer k that is reasonably large, the element with rank kp in R ought to have rank roughly k in S. The next lemma formalizes this intuition.

LEMMA 1. *Let S be a set of n elements, and R be a p-sample set of S. Suppose that integer $k \geq 1$ and real value $\delta \in (0,1)$ satisfy $kp \geq 3\ln(3/\delta)$ and $n \geq 4k$. Then, the following hold simultaneously with probability at least $1-\delta$:*

- $|R| > 2kp$
- *The element with rank $\lceil 2kp \rceil$ in R has rank between k and $4k$ in S.*

PROOF. The first bullet fails with probability

$$\begin{aligned} \mathbf{Pr}[|R| \leq 2kp] &= \mathbf{Pr}[|R| \leq (2k/n) \cdot np] \\ &\leq \mathbf{Pr}[|R| \leq (1/2)np] \\ \text{(Chernoff bound (16))} &\leq \exp(-np/12) \\ &\leq \exp(-kp/3) \\ &\leq \delta/3. \end{aligned}$$

Let e be the element with rank $\lceil 2kp \rceil$ in R, and $\hat{k}$ be the rank of e in S. Next, we bound the probability of the event $\hat{k} > 4k$. For $i \in [1, 4k]$, define x_i to be 1 if the i-th greatest element in S is sampled, or 0 otherwise. Let $X = \sum_{i=1}^{4k} x_i$, and thus, $\mathbf{E}[X] = 4kp \geq 12\ln(3/\delta)$. Event $\hat{k} > 4k$ implies $X \leq \lceil 2kp \rceil - 1$. We have:

$$\begin{aligned} \mathbf{Pr}[\hat{k} > 4k] &\leq \mathbf{Pr}[X \leq \lceil 2kp \rceil - 1] \\ &= \mathbf{Pr}[X < 2kp] \\ &= \mathbf{Pr}[X < (1/2) \cdot \mathbf{E}[X]] \\ \text{(Chernoff bound (16))} &\leq \exp(-\mathbf{E}[X]/12). \\ &\leq \delta/3. \end{aligned}$$

Finally, we bound the probability of the event $\hat{k} < k$. Define $Y = \sum_{i=1}^{k} x_i$, and thus, $\mathbf{E}[Y] = kp \geq 3\ln(3/\delta)$. Event $\hat{k} < k$ implies that $Y \geq \lceil 2kp \rceil$. We have:

$$\begin{aligned} \mathbf{Pr}[\hat{k} < k] &\leq \mathbf{Pr}[Y \geq 2kp] \\ &= \mathbf{Pr}[Y \geq 2\mathbf{E}[Y]] \\ \text{(Chernoff bound (17))} &\leq \exp(-\mathbf{E}[Y]/3) \\ &\leq \delta/3. \end{aligned}$$

By the union bound, the two bullets in the lemma hold simultaneously with probability at least $1-\delta$. □

Core-Set. As stated in Theorem 1, suppose that the underlying problem is polynomially bounded. More specifically, we say that the problem is *λ-polynomially bounded* if for any input D of n elements, there are at most n^λ distinct outcomes for $q(D)$ over all $q \in \mathbb{Q}$, where λ is a constant.

Given a subset R of D, we say that an element $e \in R$ has *weight rank i* in R if it has the i-th greatest weight in R. The next lemma proves the existence of a small-size core-set that approximately captures a specific rank for all the "large" queries whose predicates are satisfied by many elements.

LEMMA 2 (TOP-K CORE-SET LEMMA). *For any integer $K \geq 4\lambda \ln n$, there is a subset R of D such that*

- $|R| \leq 12\lambda \cdot (n/K)\ln n$.
- *For any $q \in \mathbb{Q}$ satisfying $|q(D)| \geq 4K$, it holds that*
  - $|q(R)| > 8\lambda \ln n$
  - *The element with weight rank $\lceil 8\lambda \ln n \rceil$ in $q(R)$ has weight rank between K and $4K$ in $q(D)$.*

PROOF. Set $p = 4(\lambda/K)\ln n$, and $\delta = 1/(2n^\lambda)$. These values ensure:

$$Kp = 4\lambda \ln n \geq 3\ln(3/\delta). \tag{7}$$

Let R be a p-sample set of D. We will prove that R satisfies all the conditions in the lemma with a non-zero probability.

Fix a $q \in \mathbb{Q}$ satisfying $|q(D)| \geq 4K$. Clearly, $q(R)$ is a p-sample set of $q(D)$. Applying Lemma 1 on $S = q(D)$ (the application is enabled by (7)), we know that with probability at least $1-\delta$, the following hold simultaneously:

- $|q(R)| > 2Kp = 8\lambda \ln n$.
- The element with rank $\lceil 2Kp \rceil$ has rank between K and $4K$ in $q(D)$.

By λ-polynomially boundedness and the union bound, the above holds for all queries with probability at least $1 - \delta n^\lambda = 1/2$.

Finally, as $|R|$ equals np in expectation, by Markov's inequality, $|R| \leq 3np = 12\lambda(n/K)\ln n$ with probability at least $2/3$. It thus follows that all the conditions of the lemma hold with probability at least $1 - (1/2 + 1/3) > 0$. □

3.2 Structure

We proceed to explain how to use a prioritized reporting structure to design a top-k reporting structure on a problem that is λ-polynomially bounded for a constant λ. Recall that the former structure consumes space $\mathcal{S}_{pri}(n)$ space on n

elements, and answers a prioritized query in $\mathcal{Q}_{pri}(n)+O(t/B)$ I/Os.

Define:

$$g = \frac{\mathcal{Q}_{pri}(n)}{\log_B n} \quad (8)$$

$$f = 12\lambda B \cdot \mathcal{Q}_{pri}(n). \quad (9)$$

Note that $g \geq 1$ (as required by Theorem 1), and for $B \geq 64$, both the following are fulfilled:

$$\frac{12\lambda}{f} \cdot \ln n \leq \frac{1}{g\sqrt{B}} \quad (10)$$

$$f \geq \lceil 8\lambda \ln n \rceil. \quad (11)$$

To prove Theorem 1, next we first describe our solution to top-k queries with $k \leq f$, and then, queries with larger k.

Queries with $k \leq f$. It suffices, in fact, to consider $k = f$. Given a query q with $k < f$, we first treat it as a top-f query, i.e., retrieving the set of f elements with the greatest weights in $q(D)$. Then, the final result of q can be easily obtained by performing *k-selection* [8] on these elements in $O(f/B) = O(\mathcal{Q}_{pri}(n))$ I/Os. Apart from this, the cost depends only on the top-f query.

Given a top-f query q, we answer it directly using a prioritized structure on D, if $|q(D)| \leq 4f$. We do *not* need any counting structure for estimating $q(D)$. Instead, we can achieve the purpose by issuing a *prioritized query* with predicate q and threshold $\tau = -\infty$ in a *cost monitoring* manner:

- Either the query terminates by itself
- Or we terminate it manually as soon as $4f+1$ elements have been reported.

In both cases, the number of I/Os performed by the query is at most $\mathcal{Q}_{pri}(n) + O(f/B) = O(\mathcal{Q}_{pri}(n))$. In the former case, we obtain the final result of the top-f query by performing k-selection on the elements fetched by the prioritized query. In the latter case, it must hold that $|q(D)| > 4f$; we answer such queries with a structure constructed as follows.

Take a core-set R_1 of D using Lemma 2 with $K = f$, and build a prioritized structure on R_1. This process is then carried out recursively: for every $i \geq 2$, we take a core-set R_{i+1} of R_i with the same $K = f$, and build a prioritized structure on R_{i+1}. The recursion ends at some $i = h$ where $|R_h| \leq 4f$.

For convenience, let us treat D as R_0. For each R_i ($0 \leq i \leq h-1$), it holds that $f \geq 4\lambda \ln n \geq 4\lambda \ln |R_i|$; hence, by Lemma 2:

$$|R_{i+1}| \leq \frac{12\lambda \cdot |R_i|}{f} \ln |R_i| \leq \frac{12\lambda \cdot |R_i|}{f} \ln n$$

$$\text{(by (10))} \leq \frac{|R_i|}{g\sqrt{B}}. \quad (12)$$

The total space of all the prioritized structures is therefore $O(\mathcal{S}_{pri}(n))$ by the fact that $\mathcal{S}_{pri}(n)$ is geometrically converging. Furthermore, (12) indicates that

$$h = O(\log_{g\sqrt{B}} n).$$

We now explain inductively how to answer a top-f query on any R_i for $i \in [0, h]$ in no more than

$$c \cdot (h - i + 1) \cdot \mathcal{Q}_{pri}(n) \quad (13)$$

I/Os, for some constant $c \geq 1$. At the bottom level $i = h$, the purpose can be easily achieved by scanning the entire R_h in $O(f/B)$ I/Os, which is at most $c_1 \cdot \mathcal{Q}_{pri}(n)$ for some constant $c_1 \geq 1$. Assuming that this can be done for all $i \geq j+1$, consider $i = j$, at which level we distinguish two scenarios:

- $|q(R_j)| \leq 4f$: Answer the query in the cost monitoring manner as explained earlier using the prioritized structure on R_j. The cost is $\mathcal{Q}_{pri}(|R_j|) + O(f/B) \leq c_2 \cdot \mathcal{Q}_{pri}(n)$ for some constant $c_2 \geq 1$.
- $|q(R_j)| > 4f$: According to Lemma 2, $q(R_{j+1})$ must have size at least $\lceil 8\lambda \ln |q(R_j)| \rceil$. By (11), it must hold:

  $$f \geq \lceil 8\lambda \ln |q(R_j)| \rceil.$$

  Therefore, we can retrieve the element e with weight rank $\lceil 8\lambda \ln |q(R_j)| \rceil$ in $q(R_{j+1})$ by issuing a top-f query on R_{j+1} in at most

  $$c \cdot (h - i) \cdot \mathcal{Q}_{pri}(n)$$

  I/Os. Lemma 2 indicates that the weight rank of e in $q(R_j)$ is between f and $4f$. We deploy the prioritized structure on R_j to fetch all the elements of $q(R_j)$ with weights at least $w(e)$ in $\mathcal{Q}_{pri}(|R_j|) + O(f/B) \leq c_2 \cdot \mathcal{Q}_{pri}(n)$ I/Os. The result of the top-f query can be obtained from these objects with "k-selection" in no more than $c_3 \cdot \mathcal{Q}_{pri}(n)$ I/Os for some constant $c_3 \geq 1$.

We choose c to be $\max\{c_1, c_2 + c_3\}$, which ensures:

$$c \cdot (h - i) \cdot \mathcal{Q}_{pri}(n) + (c_2 + c_3) \cdot \mathcal{Q}_{pri}(n)$$
$$\leq c \cdot (h - i + 1) \cdot \mathcal{Q}_{pri}(n)$$

and hence, completing our claim in (13). It thus becomes clear that the cost of answering a query on D is

$$O(h \cdot \mathcal{Q}_{pri}(n)) = O(\mathcal{Q}_{pri}(n) \cdot \log_{g\sqrt{B}} n).$$

Plugging in the definition equation (8) of g gives the claimed complexity in Theorem 1.

Queries with $k > f$. We apply Lemma 2 to take a core-set $R[i]$ of D with $K = 2^{i-1}f$, for $i = 1, 2, ..., h$, where h is the largest integer i satisfying $2^{i-1}f \leq n$. It is easy to verify from (9) that

$$h = O(\log(n/B)). \quad (14)$$

Our structure has two components:

- A prioritized structure on D.
- On each $R[i]$ where $1 \leq i \leq h$, build a *top-f structure*, namely, the structure we just explained for answering queries with $k \leq f$. Since $|R[i]| \leq 12\lambda(n/(2^{i-1}f)) \ln n$, all these top-$f$ structures use in total

  $$O\left(\sum_{i=1}^{h} \mathcal{S}_{pri}\left(\frac{12\lambda \cdot n \ln n}{2^{i-1}f}\right)\right)$$
  $$= O(\mathcal{S}_{pri}(n) + h) = O(\mathcal{S}_{pri}(n))$$

  space, where the derivation used the facts that (i) $\mathcal{S}_{pri}(n)$ is geometrically converging, (ii) $\mathcal{S}_{pri}(n)$ obviously needs to be $\Omega(n/B)$, and hence, by (14), $h = O(\mathcal{S}_{pri}(n))$.

The total space of our structure is therefore $O(\mathcal{S}_{pri}(n))$.

Now consider a top-k query q with $f < k \leq n$. First, if $k \geq n/2$, we answer it by simply scanning the entire D in $O(n/B) = O(k/B)$ I/Os. Next, we consider $k < n/2$.

Identify the smallest $i \in [1, h]$ such that $2^{i-1}f \geq k$. Fix the value of K to $2^{i-1}f$ in the rest of the section. Note that $k \leq K < 2k$. We then proceed as follows:

- If $|q(D)| \leq 4K$, we answer the query with cost monitoring through the prioritized structure on D in $\mathcal{Q}_{pri}(n) + O(K/B)$ I/Os.
- If $|q(D)| > 4K$, Lemma 2 indicates that $q(R[i])$ has size at least $\lceil 8\lambda \ln n \rceil$, and that the element e with weight rank $\lceil 8\lambda \ln n \rceil$ in $q(R[i])$ has weight rank between K and $4K$ in $q(D)$.

  Retrieve e by issuing a top-f query on $R[i]$. By searching the top-f structure on $R[i]$, the query finishes in $O(\mathcal{Q}_{pri}(n) \log_{g\sqrt{B}} n)$ I/Os, as proved earlier. Extract from the prioritized structure on D the elements in $q(D)$ whose weights are at least $w(e)$; this entails $\mathcal{Q}_{pri}(n) + O(K/B)$ I/Os. Finally, the query result can be produced with k-selection in $O(K/B)$ I/Os.

Overall, the query performs $O(\mathcal{Q}_{pri}(n) \log_{g\sqrt{B}} n + K/B)$ I/Os. This completes the whole proof of Theorem 1.

4. REDUCTION WITH EXPECTED EFFICIENCY

In this section, we provide a reduction to establish the correctness of Theorem 2. Our discussion focuses on $n \geq B \cdot \mathcal{Q}_{max}(n)$; otherwise, a top-$k$ query can be trivially answered by performing k-selection on the whole D in $O(n/B) = O(\mathcal{Q}_{max}(n))$ I/Os.

Rank Sampling, Again. Our current rank sampling lemma, i.e., Lemma 1, falls short for the subsequent discussion. That lemma governs the behavior of the sample with rank $\lceil 2kp \rceil$ which—given the working condition $kp \geq 3\ln(3/\delta)$—is strictly greater than 1. Intuitively, this is problematic because, to apply max (i.e., top-1) queries, we need to understand the behavior of rank 1 in the sample set, i.e., the largest sample.

Fortunately, we are able to get around the obstacle by proving an alternative result, which is less general than Lemma 1, but serves exactly the purpose that Lemma 1 does not.

LEMMA 3. *Let S be a set of n elements, and $K \geq 2$ a real value satisfying $n \geq 4K$. For a $(1/K)$-sample set R of S, the following hold simultaneously with probability at least 0.09:*

- $|R| \geq 1$
- *The largest element in R has rank in S greater than K but at most $4K$.*

PROOF. The first bullet fails only if none of the elements in D were sampled, which occurs with a probability

$$(1 - 1/K)^n \leq (1 - 1/K)^{4K} \leq 1/e^4$$

where the last inequality used the fact that $(1 - x)^{1/x} < 1/e$ for all $x \geq 0$.

Let e be the largest element in R (note: this should be distinguished from the base of natural logarithm; the semantics of each occurrence of "e" should be clear from the context throughout the paper). Denote by $\hat{K}$ the rank of e in D. Next, we bound the probability of the event $\hat{K} > 4K$, which occurs only if none of the $4K$ largest elements in D were sampled. Hence:

$$\mathbf{Pr}[\hat{K} > 4K] = (1 - 1/K)^{4K} \leq 1/e^4.$$

Finally, we bound the probability of the event $\hat{K} \leq K$, which occurs only if at least one of the K largest elements in D was sampled. Hence:

$$\mathbf{Pr}[\hat{K} \leq K] = 1 - (1 - 1/K)^K.$$

Applying the fact that $(1 - 1/x)^x \geq 1/e^2$ for all $x \geq 2$, we know:

$$\mathbf{Pr}[\hat{K} \leq K] \leq 1 - 1/e^2.$$

The union bound now shows that the probability of violating at least one bullet of Lemma 3 is at most

$$2/e^4 + (1 - 1/e^2) < 0.91.$$

We thus complete the proof. □

Structure. We now describe how to design a top-k structure from (i) a prioritized structure, which uses $\mathcal{S}_{pri}(n)$ space on n elements and answers a prioritized query in $\mathcal{Q}_{pri}(n) + O(t/B)$ I/Os, and (ii) a max structure, which uses $\mathcal{S}_{max}(n)$ space and answers a max query in $\mathcal{Q}_{max}(n)$ I/Os.

Fix a constant $\sigma = 1/20$. For each integer $i \geq 1$, define:

$$K_i \quad = \quad B \cdot \mathcal{Q}_{max}(n) \cdot (1 + \sigma)^{i-1}.$$

Let h be the largest i such that $K_i \leq n/4$; clearly, $h = O(\log(n/B))$. We create a prioritized structure on D. Also, for each $i \in [1, h]$, we take a $(1/K_i)$-sample set R_i of D, and create a max structure on R_i.

Query. Let us first eliminate queries with $k < B \cdot \mathcal{Q}_{max}(n)$. Given such a query q, we first treat it as a top-$(B \cdot \mathcal{Q}_{max}(n))$ query, i.e., extracting the $B \cdot \mathcal{Q}_{max}(n)$ elements with the greatest weights in $q(D)$. Then, the final result of the original query can be obtained by performing k-selection on those elements. The total cost is $O(\mathcal{Q}_{max}(n))$ plus the time of the top-$(B \cdot \mathcal{Q}_{max}(n))$ query.

Let us now focus on a top-k query q with $k \geq B \cdot \mathcal{Q}_{max}(n)$. If $k > K_h$, the query is answered naively by reading the whole D in $O(n/B)$ I/Os, which is $O(k/B)$ because $k > K_h \geq n/(4(1 + \sigma)) = \Omega(n)$.

If $k \leq K_h$, identify the smallest i such that $K_i \geq k$; note that $K_i = \Theta(k)$. Setting $j = i$, we carry out a *round* with the steps below:

1. If $|q(D)| \leq 4K_j$, solve the query with the prioritized structure on D in the cost-monitoring manner (see Section 3.2), which costs $\mathcal{Q}_{pri}(n) + O(K_j/B)$ I/Os. The algorithm declares the round *succeeded* and terminates.
2. Otherwise, identify the element e in $q(R_j)$ with the maximum weight from the max structure on R_j in $\mathcal{Q}_{max}(n)$ I/Os. In the special case where $q(R_j)$ is empty, treat e as a dummy element with $w(e) = -\infty$.

3. Perform a prioritized query on D with q and threshold $\tau = w(e)$ in a cost-monitoring manner:
   (a) Either the query terminates by itself—let S be the set of elements retrieved,
   (b) Or we terminate it as soon as $4K_j + 1$ elements have been reported.

   In both cases, the cost is $\mathcal{Q}_{pri}(n) + O(K_j/B)$.
4. Declare this round *failed* if either of the following is true:
   - Case 3(a) occurred, but $|S| \leq K_j$.
   - Case 3(b) occurred.

   Otherwise, declare this round *succeeded*.
5. If succeeded, perform k-selection on S to produce the k elements in $q(D)$ with the largest weights, and terminate the algorithm by returning them as the final answer.
6. Otherwise (i.e., failed), increase j by 1.
   (a) If $j \leq h$, start the next round from Step 1.
   (b) Else (i.e., $j = h + 1$), answer the top-k query naively by reading the whole D in $O(n/B) = O(K_h/B)$ I/Os; the algorithm then terminates. This is the only scenario where termination can happen in a failed round.

To analyze the cost of the algorithm, notice that a round fails only if (i) $|q(|D|)| > 4K_j$ (otherwise, Line 1 terminates the algorithm), and (ii) one of the two bullets in Step 4 is true. Thus, Lemma 3 tells us that failure happens with probability at most 0.91, noticing that $q(R_i)$ is a $(1/K_j)$-sample set of $q(D)$. This implies that round j—for a specific $j \geq i$—is executed only with probability 0.91^{j-i}, namely, only when all the preceding rounds have failed. Also observe that round j, regardless of whether it fails, performs at most

$$\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + O(K_j/B)$$

I/Os. Thus, the expected cost of the algorithm is bounded by

$$\begin{aligned} & \sum_{j=i}^{h} O\Big(\Big(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + \frac{K_j}{B}\Big) \cdot 0.91^{j-i}\Big) \\ = \; & O\Big(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + \sum_{j=i}^{h} \frac{K_j}{B} 0.91^{j-i}\Big) \qquad (15) \end{aligned}$$

Notice that $K_j = K_i \cdot (1+\sigma)^{j-i} = O(k) \cdot (1+\sigma)^{j-i}$. Plugging these into (15) shows that the expected cost is

$$O\left(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + \frac{k}{B} \sum_{j=i}^{h} ((1+\sigma) \cdot 0.91)^{j-i}\right)$$

which is $O(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + k/B)$ because $(1+\sigma) \cdot 0.91 < 1$.

Space. The prioritized structure on D obviously takes up $\mathcal{S}_{pri}(n)$ space. We claim that all the max structures occupy $o(n/B) + O(\mathcal{S}_{max}(\frac{6n}{B \cdot \mathcal{Q}_{max}(n)}))$ expected space in total, which implies the space result in Theorem 2 (because $\mathcal{S}_{pri}(n) = \Omega(n/B)$).

The claim is fairly intuitive because $\mathbf{E}[|R_i|] = n/K_i$ geometrically decreases as i increases, which, together with the fact that $\mathcal{S}_{max}(n)$ is geometrically converging, seems to yield the claim immediately. The complication, however, is that $\mathcal{S}_{max}(n)$ may be a convex function, such that $\mathbf{E}[\mathcal{S}_{max}(|R_i|)]$ is *not* necessarily $O(\mathcal{S}_{max}(\mathbf{E}[|R_i|]))$. Next, we show how to circumvent this obstacle.

We will prove that all the max structures occupy $o(n/B) + O(\mathcal{S}_{max}(\frac{6n}{B \cdot \mathcal{Q}_{max}(n)}))$ space in total with probability at least $1 - 1/n^2$. Combining this with the fact that all those structures obvious demand no more than $O(\mathcal{S}_{max}(n) \cdot h) = O((n^2/B) \cdot \log(n/B))$ gives the target claim.

Let i^* be the largest i such that $K_i \leq n/(3 \ln n)$. Consider an $i \in [1, i^*]$. Since $|R_i|$ is the sum of n independent Bernoulli variables each of which equals 1 with probability $1/K_i$, by Chernoff bound (17), we have:

$$\begin{aligned} \mathbf{Pr}[|R_i| \geq 6 \cdot \mathbf{E}[|R_i|]] &\leq \exp(-\mathbf{E}[|R_i|]) \\ &= \exp(-n/K_i) \leq 1/n^3. \end{aligned}$$

Therefore, with probability at least $1 - h/n^3$, the max structures on $R_1, R_2, ..., R_{i^*}$ use at most

$$\begin{aligned} & \sum_{i=1}^{i^*} O\left(\mathcal{S}_{max}\left(\frac{6n}{B \cdot \mathcal{Q}_{max}(n) \cdot (1+\sigma)^{i-1}}\right)\right) \\ = \; & O\left(h + \mathcal{S}_{max}\left(\frac{6n}{B \cdot \mathcal{Q}_{max}(n)}\right)\right) \end{aligned}$$

space overall.

Let us now concentrate on $i \in [i^* + 1, h]$. Notice that there are only $O(\log \log n)$ such values of i. Also, by definition of i^*, we know that $\mathbf{E}[|R_i|] = n/K_i$ is in the range from 4 to $O(\log n)$. Again, by Chernoff bound (17), we have :

$$\begin{aligned} \mathbf{Pr}[|R_i| \geq (\ln n^4) \cdot \mathbf{E}[|R_i|]] &\leq \exp(-(\ln n^4) \cdot \mathbf{E}[|R_i|]/6) \\ &\leq \exp(-(\ln n^{4 \cdot 2/3}) \\ &= 1/n^{8/3}. \end{aligned}$$

Hence, with probability at least $1 - O(\log \log n)/n^{8/3}$, it holds that for all $i \in [i^* + 1, h]$:

$$|R_i| \leq 4 \ln n \cdot \mathbf{E}[|R_i|] = O(\log^2 n).$$

By the fact that $\mathcal{S}_{max}(n) = O(1 + n^2/B)$, the max structures on R_{i^*}, R_{i^*+1}, ..., R_h together consume no more than $O(h + (\log \log n \cdot \log^4 n)/B) = o(n/B)$ space. We thus conclude that, with probability at least $1 - h/n^3 - O(\log \log n)/n^{8/3} > 1 - 1/n^2$, all the max structures use $o(n/B) + O(\mathcal{S}_{max}(\frac{6n}{B \cdot \mathcal{Q}_{max}(n)}))$ space.

Update. It remains to discuss how to support insertions and deletions on the input set D. This is in fact fairly easy, if one observes that each element $e \in D$ has in expectation only $O(1)$ copies in the entire structure—recall that the sampling rate of R_i equals $1/K_i$, which geometrically decreases as i increases. Hence, the insertion of e triggers one insertion into the prioritized structure, and one insertion into $O(1)$ max structures in expectation. The total cost is thus $O(\mathcal{U}_{pri} + \mathcal{U}_{max})$ expected. Also, we can record in $O(1)$ expected words which max structures include e. By hashing, this "bookkeeping" itself can be maintained in $O(1)$ expected I/Os as e is inserted/deleted, without increasing the overall

space complexity. In this way, a deletion of e can also be supported in $O(\mathcal{U}_{pri} + \mathcal{U}_{max})$ expected I/Os.

The above argument still works even if one or both of $\mathcal{U}_{pri}$ and $\mathcal{U}_{max}$ are amortized. This completes the whole proof of Theorem 2.

5. NEW STRUCTURES FROM REDUCTIONS

Next, we utilize our reductions to derive the new top-k structures claimed in Section 1.4, specifically, in Theorems 3-6. While those theorems were presented essentially in descending order of their importance to databases, next we will prove them in a different order: from the least sophisticated to the most.

5.1 Top-k Interval Stabbing (Theorem 4)

The theorem now becomes almost a pleasure to prove:

- The prioritized-reporting version of the problem has been studied by Tao [34] (where the version is called *ray stabbing*), who gave an $O(n/B)$-size structure that answers a query in $O(\log_B n + t/B)$ I/Os, and supports an update in $O(\log_B n)$ amortized I/Os.
- The max-reporting version has been studied by Agarwal et al. [7], who gave an $O(n/B)$-size structure that answers a query in $O(\log_B n)$ I/Os, and supports an update in $O(\log_B n)$ amortized I/Os.

Therefore, the first and second bullets of Theorem 4 follow from Theorems 2 and Theorem 1, respectively.

5.2 Top-k Point Enclosure (Theorem 5)

The prioritized-reporting version of the problem has been studied by Rahul [27], who gave a structure of $O(n \log^* n)$ size and $O(\log n \log\log n + t)$ query time.

Next, we explain how to solve the max-reporting version with a structure that uses $O(n \log n)$ space and answers a query in $O(\log n)$ time. Based on the above, the first and second bullets of Theorem 5 follow from Theorems 2 and Theorem 1, respectively. Note that the application of Theorem 2 demonstrates the "bootstrapping power" as remarked in Section 1.4.

1D Stabbing Max. Section 5.1 already mentioned a dynamic structure for solving the max-reporting version of interval stabbing. In fact, if the goal is to design a *static* structure, that problem can be settled with a very simple structure of $O(n)$ space and $O(\log n)$ time. Although this should be folklore, we give the details nonetheless because it will be helpful later.

Let D be a set of n weighted intervals in $\mathbb{R}$. The $2n$ endpoints of the intervals divide $\mathbb{R}$ into at most $2n + 1$ disjoint subintervals. With each subinterval I, we associate the maximum weight of all the intervals in D that span I. Given a value $q \in \mathbb{R}$, a query returns the maximum weight of the intervals of D containing q. This is precisely the weight associated with the subinterval containing q. Finding the subinterval is essentially predecessor search, which can be carried out in $O(\log n)$ time by performing binary search on the endpoints.

2D Stabbing Max (Point Enclosure Max). Now we return to the max-reporting version of point enclosure. The input is a set D of n weighted axis-parallel rectangles. Create a segment tree T on the x-projections of those rectangles. For each node u of T, define D_u to be the set of segments assigned to u. Build a 1D stabbing max structure on D_u. The overall space is clearly $O(n \log n)$.

Given a point $q = (x, y)$, a query returns the maximum weight of the rectangles of D containing q. To process the query, we descend a root-to-leaf path Π of T according to x, and then, on each node $u \in \Pi$, issue a 1D stabbing max query on D_u with y. The final answer is the maximum of the results of these 1D queries. The algorithm takes $O(\log^2 n)$ time, which can be improved to $O(\log n)$ with fractional cascading [14] because, as mentioned earlier, each 1D query performs nothing but predecessor search on a sorted list.

5.3 Top-k 3D Dominance (Theorem 6)

The prioritized-reporting version of the problem has been studied by Afshani et al. [2] (where the version is called *4D dominance reporting*), who gave a structure with size $O(n \log n / \log\log n)$ and query time $O(\log^{1.5} n + t)$.

Next, we explain how to solve the max-reporting version with a structure that uses $O(n)$ space and answers a query in $O(\log^{1.5} n)$ time. Plugging in these bounds into Theorem 2 proves Theorem 6.

In this setting, D is a set of n weighted points in $\mathbb{R}^3$. Let $e_1, e_2, ..., e_n$ be the sequence of points in descending order of weight. With each point e_i, we associate a region ρ_i in $\mathbb{R}^3$ satisfying the following constraint: any point $q = (x, y, z)$ belongs to ρ_i if and only if e_i is the point with the maximum weight in $(-\infty, x] \times (-\infty, y] \times (-\infty, z]$. The region assignment below ensures the constraint for all points $e_i = (e_{ix}, e_{iy}, e_{iz})$:

- $\rho_1 = [e_{1x}, \infty) \times [e_{1y}, \infty) \times [e_{1z}, \infty)$.
- For $i \in [2, n]$:

$$\rho_i = [e_{ix}, \infty) \times [e_{iy}, \infty) \times [e_{iz}, \infty) \setminus \bigcup_{j=1}^{i-1} \rho_j.$$

Each non-empty region ρ_i is decomposed into axis-parallel disjoint cuboids by performing a *vertical decomposition*. If ρ_i has n_i vertices, then the number of cuboids in the decomposition of ρ_i will be $O(n_i)$. It can be verified [1] that $\sum_{i=1}^{n} n_i = O(n)$.

Therefore, the max reporting problem can be transformed to a point location problem: Given a query point q, find the cuboid (if any) containing q from a set of $O(n)$ disjoint axis-parallel cuboids. Rahul [27] presented a structure of size $O(n)$ to answer such a query in $O(\log^{1.5} n)$ time.

5.4 Top-k Halfspace Reporting: $d = 2$ (Theorem 3: 1st Bullet)

We will show:

- The prioritized-reporting version of the problem can be settled by an $O(n \log n)$-size structure that answers a query in $O(\log n + t)$ time.
- The max-reporting version can be settled by an $O(n)$-size structure that answers a query in $O(\log n)$ time.

Plugging in these results into Theorem 2 proves the first bullet of Theorem 3.

Prioritized Reporting. Chazelle et al. [15] settled the *original* 2D halfspace reporting problem. Specifically, they showed that n points in $\mathbb{R}^2$ can be stored in an $O(n)$-size structure such that, given a halfspace q, all the t input points falling in q can be reported in $O(\log n + t)$ time. Their query algorithm, in fact, starts with finding the predecessor of some query value (that depends on q) in a pre-computed list of real values. This accounts for the $O(\log n)$ term. Once the predecessor is found, the rest of the algorithm finishes in $O(1 + t)$ time.

Next, we leverage the above structure to attack the prioritized-reporting version. In this setting, the input is a set D of n weighted points in $\mathbb{R}^2$. Create a balanced binary search tree T on their weights, with each weight stored in a leaf, which is associated with the corresponding point in D. For each node u of T, denote by D_u the set of points stored in the subtree of u. Create a halfspace reporting structure of [15] on D_u. The total space is $O(n \log n)$.

Given a halfspace q and a threshold τ, a query returns all the points $e \in D$ such that $e \in q$ and weight $w(e) \geq \tau$. We answer it as follows. First, collect the *canonical* set $U(\tau)$ of nodes $u_1, u_2, ..., u_m$ with the smallest m such that D_{u_1}, D_{u_2}, ..., D_{u_m} are disjoint, and their union equals $\{e \in D \mid w(e) \geq \tau\}$. It is rudimentary to find these $m = O(\log n)$ nodes in $O(\log n)$ time. Then, perform a halfspace reporting query using q on D_{u_i}, for each $i \in [1, m]$. The final answer is the union of the outputs of all these m queries.

As explained earlier, each halfspace reporting query spends $O(\log n)$ time on a predecessor search, which makes the total query time $O(\log^2 n + t)$. A standard application of fractional cascading reduces the time to $O(\log n + t)$.

Max Reporting. The input is again a set D of n weighted points. Given a halfspace q, a query returns the maximum weight of the points of D covered by q. By standard duality, we consider instead the following equivalent *stabbing max* problem. The input is a set D' of n weighted halfspaces in $\mathbb{R}^2$. The goal is to store D' in a data structure such that, given a point q' in $\mathbb{R}^2$, we can efficiently report the maximum weight of the halfspaces of D' containing q'. Below we describe an $O(n)$-size structure with $O(\log n)$ query time.

Using the idea in Section 5.3, we can transform the problem into a point location problem on a planar subdivision of complexity $O(n)$. Let $e'_1, e'_2, ..., e'_n$ be the halfspaces of D', in descending order of weight. For each e'_i, define a region ρ_i in $\mathbb{R}^2$ satisfying the following constraint: any point q belongs to ρ_i if and only if e'_i is the halfspace with the maximum weight among all the halfspaces containing q. The region assignment below ensures the constraint:

- $\rho_1 = e'_1$.
- For $i \in [2, n]$, $\rho_i = e'_i \setminus \bigcup_{j=1}^{i-1} \rho_j$.

ρ_1, ρ_2, ..., ρ_n are disjoint polygons such that the planar subdivision they induce has at most n vertices. To see this, imagine generating ρ_i in ascending order of i as follows. If e'_i falls entirely in $\bigcup_{j=1}^{i-1} \rho_j$, then e'_i introduces no vertex on the subdivision. Otherwise, at least one point p on the boundary line of e'_i must be outside $\bigcup_{j=1}^{i-1} \rho_j$. Walk from p along the boundary line towards one direction, and stop as soon as hitting the boundary line of any of the halfspaces already considered. The stopping point is a new vertex on the subdivision. Similarly, walking from p towards the other direction will determine another new vertex.

Given a query point q', it suffices to find the polygon of the subdivision containing q'. This can be done in $O(\log n)$ time with an $O(n)$-size structure [31].

5.5 Top-k Halfspace Reporting: $d \geq 4$ (Theorem 3: 2nd and 3rd Bullets)

The subsequent discussion demonstrates the power of Theorem 1 in showing the asymptotic equivalence between top-k reporting and prioritized reporting, when the query time is large.

RAM. We will show that the prioritized-reporting version of the problem can be settled by an $O(n \log n)$-size structure that answers a query in $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor}) + O(t)$ time. Plugging this into Theorem 1 yields the second bullet of Theorem 3.

In the *original* halfspace reporting problem, we want to store n points in $\mathbb{R}^d$ in a structure such that, given any halfspace q in $\mathbb{R}^d$, all the input points falling in q can be reported efficiently. Afshani and Chan [4] gave a structure of $O(n)$ space and query time $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor}) + O(t)$.

Recall that Section 5.4 presented a prioritized reporting structure in 2D space, where there is a 2D halfspace reporting structure on each D_u. To obtain a prioritized reporting structure for $\mathbb{R}^d$, we simply replace that 2D structure with the d-dimensional halfspace reporting structure of [4]. A prioritized query is answered in the way as described in Section 5.4, excluding the part about fractional cascading. The claimed space and query bounds follow from the same analysis as in Section 5.4.

EM. We will show that the prioritized-reporting version of the problem can be settled by an $O(n/B)$-size structure that answers a query in $O((n/B)^{1-1/\lfloor d/2 \rfloor + \epsilon} + t/B)$ time. Plugging this into Theorem 1 yields the third bullet of Theorem 3.

For the *original* halfspace reporting problem, Agarwal et al. [6] gave a structure of $O(n/B)$ space and query time $O((n/B)^{1-1/\lfloor d/2 \rfloor + \epsilon'} + t/B)$ for any arbitrarily small constant $\epsilon' > 0$, which we utilize below to design the required structure for prioritized reporting.

The input is a set D of n weighted points in $\mathbb{R}^d$, which we denote as $e_1, e_2, ..., e_n$ in descending order of weight. Set $f = (n/B)^{\epsilon/2}$. Build a B-tree T on the weights of the n points with leaf capacity B and internal fanout f. Store each point together with its weight in the corresponding leaf. For each node u of T, denote by D_u the set of points stored in the subtree of u; we create a structure of [6] on D_u by setting $\epsilon' = \epsilon/2$. The overall space consumption is $O(n/B)$—noticing that T has $O(1)$ levels.

To answer a query with halfspace q and threshold τ, we collect the canonical set $U(\tau)$ of nodes $u_1, u_2, ..., u_m$ with the smallest m such that $D_{u_1}, ..., D_{u_m}$ are disjoint, and their union equals $\{e \in D \mid w(e) \geq \tau\}$. It is rudimentary to find these $m = O(f)$ nodes in $O(1 + f/B)$ I/Os. We then perform a halfspace reporting query using q on D_{u_i}, for all $i \in [1, m]$. The final answer is the union of the outputs of all these m

queries. The query cost is

$$\begin{aligned} & O\left(m \cdot (n/B)^{1-1/\lfloor d/2 \rfloor + \epsilon'} + t/B\right) \\ = \; & O\left((n/B)^{1-1/\lfloor d/2 \rfloor + \epsilon} + t/B\right) \end{aligned}$$

I/Os.

6. REFERENCES

[1] Peyman Afshani. On dominance reporting in 3D. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 41–51, 2008.

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

[3] Peyman Afshani, Gerth Stolting Brodal, and Norbert Zeh. Ordered and unordered top-k range reporting in large data sets. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 390–400, 2011.

[4] Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009.

[5] Peyman Afshani, Chris H. Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry*, 43(8):700–712, 2010.

[6] Pankaj K. Agarwal, Lars Arge, Jeff Erickson, Paolo Giulio Franciosa, and Jeffrey Scott Vitter. Efficient searching with linear constraints. *Journal of Computer and System Sciences (JCSS)*, 61(2):194–216, 2000.

[7] Pankaj K. Agarwal, Lars Arge, Haim Kaplan, Eyal Molad, Robert Endre Tarjan, and Ke Yi. An optimal dynamic data structure for stabbing-semigroup queries. *SIAM Journal of Computing*, 41(1):104–127, 2012.

[8] Alok Aggarwal and Jeffrey Scott Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM (CACM)*, 31(9):1116–1127, 1988.

[9] Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal of Computing*, 38(3):899–921, 2008.

[10] Iwona Bialynicka-Birula and Roberto Grossi. Rank-sensitive data structures. In *String Processing and Information Retrieval (SPIRE)*, pages 79–90, 2005.

[11] Gerth Stolting Brodal. External memory three-sided range reporting and top-k queries with sublogarithmic updates. In *Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 47, pages 23:1–23:14. 2016.

[12] Gerth Stolting Brodal, Rolf Fagerberg, Mark Greve, and Alejandro Lopez-Ortiz. Online sorted range reporting. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 173–182, 2009.

[13] Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal of Computing*, 17(3):427–462, 1988.

[14] Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986.

[15] Bernard Chazelle, Leonidas J. Guibas, and D. T. Lee. The power of geometric duality. *BIT Numerical Mathematics*, 25(1):76–90, 1985.

[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[17] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[18] Torben Hagerup and Christine Rub. A guided tour of chernoff bounds. *Information Processing Letters (IPL)*, 33(6):305–308, 1990.

[19] Wing-Kai Hon, Rahul Shah, Sharma V. Thankachan, and Jeffrey Scott Vitter. Space-efficient frameworks for top-k string retrieval. *Journal of the ACM (JACM)*, 61(2):9:1–9:36, 2014.

[20] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4), 2008.

[21] Hans-Peter Kriegel, Marco Potke, and Thomas Seidl. Managing intervals efficiently in object-relational databases. In *Proceedings of Very Large Data Bases (VLDB)*, pages 407–418, 2000.

[22] Moshe Lewenstein. Orthogonal range searching for text indexing. In *Space-Efficient Data Structures, Streams, and Algorithms*, pages 267–302, 2013.

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

[24] S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002.

[25] Gonzalo Navarro and Yakov Nekrich. Top-k document retrieval in optimal time and linear space. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1066–1077, 2012.

[26] Manish Patil, Sharma V. Thankachan, Rahul Shah, Yakov Nekrich, and Jeffrey Scott Vitter. Categorical range maxima queries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 266–277, 2014.

[27] Saladi Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\mathbb{R}^3$. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 200–211, 2015.

[28] Saladi Rahul and Ravi Janardan. A general technique for top-k geometric intersection query problems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 26(12):2859–2871, 2014.

[29] Saladi Rahul and Yufei Tao. On top-k range reporting in 2d space. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 265–275, 2015.

[30] Biswajit Sanyal, Prosenjit Gupta, and Subhashis Majumder. Colored top-k range-aggregate queries.

Information Processing Letters (IPL), 113(19-21):777–784, 2013.

[31] Neil Sarnak and Robert Endre Tarjan. Planar point location using persistent search trees. *Communications of the ACM (CACM)*, 29(7):669–679, 1986.

[32] Rahul Shah, Cheng Sheng, Sharma V. Thankachan, and Jeffrey Scott Vitter. Top-k document retrieval in external memory. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 803–814, 2013.

[33] Cheng Sheng and Yufei Tao. Dynamic top-k range reporting in external memory. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, 2012.

[34] Yufei Tao. Stabbing horizontal segments with rays. In *Proceedings of Symposium on Computational Geometry (SoCG)*, 2012.

[35] Yufei Tao. A dynamic I/O-efficient structure for one-dimensional top-k range reporting. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 256–265, 2014.

[36] David A. White and Ramesh Jain. Similarity indexing with the SS-tree. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 516–523, 1996.

Appendix: Chernoff Bounds

Let $X_1, ..., X_n$ be independent Bernoulli variables such that $\mathbf{Pr}[X_i = 1] = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^{n} p_i$. It holds for any $\alpha \in (0, 1)$ that

$$\mathbf{Pr}[X \leq (1-\alpha)\mu] \quad \leq \quad e^{-\alpha^2 \mu/3}. \tag{16}$$

For any $\alpha \geq 2$, it holds that

$$\mathbf{Pr}[X \geq \alpha\mu] \quad \leq \quad e^{-\alpha\mu/6}. \tag{17}$$

The above inequalities can be found in many papers and textbooks, e.g., [18, 23]