Top-k Indexes Made Small and Sweet

Yufei Tao

School of Information Technology and Electrical Engineering
University of Queensland
Small

Easy to implement.

Sweet

With non-trivial theoretical guarantees.
**Computation Models**

- **RAM**
- **EM**
  - Disk block size $B$
  - Running time: number of I/Os
  - As far as this talk is concerned, think about RAM as EM with $B = 2$. 

“Enough Already” [Carey and Kossmann’97]

What to do if the database returns too many results?
“Enough Already” [Carey and Kossmann’97]

What to do if the database returns too many results?

Popular Approach: Top-k

- Returns only the $k$ elements with the highest priorities.
- Every reporting query has its top-$k$ version.
Example 1: 1d Top-\(k\) Range Reporting

Range Reporting

“Find the hotels whose prices are between 100 and 200 dollars per night.”

\[ q \]

YF Tao
Top-k Indexes Made Small and Sweet
Example 1: 1d Top-$k$ Range Reporting

Range Reporting

“Find the hotels whose prices are between 100 and 200 dollars per night.”

Top-$k$ Range Reporting

“Find the $k$ best-rated hotels whose prices are between 100 and 200 dollars per night.”
Example 2: 2d Top-

“Find the $k$ best-rated hotels in the Manhattan area”.

- 15
- 10
- 12
- 20
- 7
- 25
- 5
- 30
Example 3: 1d Top-k Stabbing

“Find the $k$ accounts with the largest balances active on 1 Jan 2016.”
Example 4: 2d Top-$k$ Halfspace Reporting

“Find the $k$ best-rated hotels satisfying

$$0.1 \cdot \text{distance} + 0.9 \cdot \text{price} \geq 5.$$
Top-\(k\) Literature

**Small** (system): A LOT OF work!
Top-k Literature

Small (system): A LOT OF work!

Sweet (theory):

- Serious attention only in the last few years.
- Best understood: 1d top-k range reporting.
- Recent efforts: more sophisticated top-k problems.
- All with sophisticated designs.
This Talk

Aims to **significantly** simplify the design of top-\(k\) structures with non-trivial theoretical guarantees for a **large** set of problems.

Our results **immediately** imply structures with excellent efficiency for all the aforementioned problems (and much more).

- No/little efforts required.

Results taken from our work in PODS’12, PODS’14, and PODS’16.
**Reporting Queries in General**

- **Input $D$**: a set of $n$ elements from a domain $\mathbb{D}$.
- **$Q$**: the set of all permissible predicates.
- A query
  - chooses a predicate $q \in Q$
  - reports the set $q(D)$ of elements in $D$ satisfying $q$.  

\[ q \]

---

YF Tao

Top-k Indexes Made Small and Sweet

11 / 34
**Top-k Reporting Queries in General**

- **Input** $D$: a set of $n$ elements from a domain $\mathbb{D}$.
  - Each $e \in D$ is associated with a real-valued weight $w(e)$.
- $Q$: the set of all permissible predicates.
- A query
  - chooses a predicate $q \in Q$ and a positive integer $k$
  - reports the $k$ elements in $q(D)$ with the highest weights.
Prioritized Reporting Queries in General

- **Input $D$**: a set of $n$ elements from a domain $\mathbb{D}$.
  - Each $e \in D$ is associated with a real valued weight $w(e)$.
- $\mathcal{Q}$: the set of all permissible predicates.
- A query
  - chooses a predicate $q \in \mathcal{Q}$ and a real-value $\tau$
  - reports the elements in $q(D)$ whose weights are $\geq \tau$.

\[ q \]
\[ \tau = 10 \]
Common Pattern in All the Theoretical Top-k Solutions

1. Design a structure for prioritized reporting (often easy).

2. Convert a top-k query to a prioritized query (usually difficult).

The Pattern is Compulsory!

Prioritized reporting can always be reduced to top-k reporting.

If there is a top-k structure of $S_{top}(n)$ space and $Q_{top}(n) + O(k/B)$ query time, then there is a prioritized structure of $O(S_{top}(n))$ space and $O(Q_{top}(n)) + O(t/B)$ query time, where $t$ is the number of elements reported.

Implication: prioritized reporting no harder than top-k reporting.

YF Tao

Top-k Indexes Made Small and Sweet
Common Pattern in All the Theoretical Top-k Solutions

1. Design a structure for prioritized reporting (often easy).
2. Convert a top-k query to a prioritized query (usually difficult).

The Pattern is Compulsory!

Prioritized reporting can always be reduced to top-k reporting.

- If there is a top-k structure of $S_{top}(n)$ space and $Q_{top}(n) + O(k/B)$ query time
- Then there is a prioritized structure of $O(S_{top}(n))$ space and $O(Q_{top}(n)) + O(t/B)$ query time, where $t$ is the number of elements reported.

YF Tao
Top-k Indexes Made Small and Sweet
Common Pattern in All the Theoretical Top-k Solutions

1. Design a structure for prioritized reporting (often easy).
2. Convert a top-k query to a prioritized query (usually difficult).

The Pattern is Compulsory!

Prioritized reporting can always be reduced to top-k reporting.

- If there is a top-k structure of $S_{top}(n)$ space and $Q_{top}(n) + O(k/B)$ query time
- Then there is a prioritized structure of $O(S_{top}(n))$ space and $O(Q_{top}(n)) + O(t/B)$ query time, where $t$ is the number of elements reported.

Implication: prioritized reporting no harder than top-k reporting.
Open: is top-\(k\) reporting harder than prioritized reporting?

Suppose that there is a prioritized structure \(S_{\text{pri}}(n)\) space and \(Q_{\text{pri}}(n) + O(t/B)\) query time. We want to leverage the structure as a black box to design a top-\(k\) structure of \(S_{\text{top}}(n)\) space and \(Q_{\text{top}}(n) + O(k/B)\) query time. How good can functions \(S_{\text{top}}(n)\) and \(Q_{\text{top}}(n)\) be?

The Grand Wish

If one can prove \(S_{\text{top}}(n) = O(S_{\text{pri}}(n))\) and \(Q_{\text{top}}(n) = O(Q_{\text{pri}}(n))\), then it will mean that the two problems actually have the same hardness!

Our Result 1

\(S_{\text{top}}(n) = O(S_{\text{pri}}(n))\) and \(Q_{\text{top}}(n) = O(Q_{\text{pri}}(n) \cdot \log B/n)\) subject to very mild conditions.

What if we want no deterioration at all?
Open: is top-k reporting harder than prioritized reporting?

Suppose that there is a prioritized structure of $S_{pri}(n)$ space and $Q_{pri}(n) + O(t/B)$ query time.

We want to leverage the structure as a BLACK BOX to design a top-k structure of $S_{top}(n)$ space and $Q_{top}(n) + O(k/B)$ query time. How good can functions $S_{top}(n)$ and $Q_{top}(n)$ be?
Open: is top-\(k\) reporting harder than prioritized reporting?

Suppose that there is a prioritized structure of \(S_{pri}(n)\) space and \(Q_{pri}(n) + O(t/B)\) query time.

We want to leverage the structure as a BLACK BOX to design a top-\(k\) structure of \(S_{top}(n)\) space and \(Q_{top}(n) + O(k/B)\) query time. How good can functions \(S_{top}(n)\) and \(Q_{top}(n)\) be?

The Grand Wish

If one can prove \(S_{top}(n) = O(S_{pri}(n))\) and \(Q_{top}(n) = O(Q_{pri}(n))\), then it will mean that the two problems actually have the same hardness!
Open: is top-$k$ reporting harder than prioritized reporting?

Suppose that there is a prioritized structure of $S_{pri}(n)$ space and $Q_{pri}(n) + O(t/B)$ query time.

We want to leverage the structure as a BLACK BOX to design a top-$k$ structure of $S_{top}(n)$ space and $Q_{top}(n) + O(k/B)$ query time. How good can functions $S_{top}(n)$ and $Q_{top}(n)$ be?

**The Grand Wish**

If one can prove $S_{top}(n) = O(S_{pri}(n))$ and $Q_{top}(n) = O(Q_{pri}(n))$, then it will mean that the two problems actually have the same hardness!

**Our Result 1**

$$S_{top}(n) = O(S_{pri}(n)) \text{ and } Q_{top}(n) = O(Q_{pri}(n) \cdot \log_B n)$$

subject to very mild conditions.
Open: is top-\(k\) reporting harder than prioritized reporting?

Suppose that there is a prioritized structure of \(S_{pri}(n)\) space and \(Q_{pri}(n) + O(t/B)\) query time.

We want to leverage the structure as a BLACK BOX to design a top-\(k\) structure of \(S_{top}(n)\) space and \(Q_{top}(n) + O(k/B)\) query time. How good can functions \(S_{top}(n)\) and \(Q_{top}(n)\) be?

The Grand Wish

If one can prove \(S_{top}(n) = O(S_{pri}(n))\) and \(Q_{top}(n) = O(Q_{pri}(n))\), then it will mean that the two problems actually have the same hardness!

Our Result 1

\[
S_{top}(n) = O(S_{pri}(n)) \text{ and } Q_{top}(n) = O(Q_{pri}(n) \cdot \log_B n)
\]

subject to very mild conditions.

What if we want no deterioration at all?
**Max Reporting Queries in General**

- **Input** $D$: a set of $n$ elements from a domain $\mathbb{D}$.
  - Each $e \in D$ is associated with a real-valued weight $w(e)$.
- **$Q$**: the set of all possible predicates.
- A query
  - chooses a predicate $q \in Q$
  - reports the element in $q(D)$ with the **highest weight**.

```
9  1  5  20 10  3  15  8  2  6
```

Obviously no harder than top-$k$ reporting.
Suppose that there is

- A **prioritized structure** of \( S_{pri}(n) \) space that answers a query in \( Q_{pri}(n) + O(t/B) \) time, and can be updated in \( U_{pri}(n) \) time;
- A **max structure** of \( S_{max}(n) \) space that answers a query in \( Q_{max}(n) \) query time, and can be updated in \( U_{max}(n) \) time.
Suppose that there is

- A **prioritized structure** of $S_{pri}(n)$ space that answers a query in $Q_{pri}(n) + O(t/B)$ time, and can be updated in $U_{pri}(n)$ time;
- A **max structure** of $S_{max}(n)$ space that answers a query in $Q_{max}(n)$ query time, and can be updated in $U_{max}(n)$ time.

**Our Result 2**

There is a top-$k$ structure with space $S_{top}(n)$, query time $Q_{top}(n) + O(k/B)$, and update time $U_{top}(n)$ time where

\[
S_{top}(n) = O(S_{pri}(n) + S_{max}(n)) \text{ in expectation}
\]

\[
Q_{top}(n) = O(Q_{pri}(n) + Q_{max}(n)) \text{ in expectation}
\]

\[
U_{top}(n) = O(U_{pri}(n) + U_{max}(n)) \text{ in expectation}
\]

subject to mild conditions.
Suppose that there is
- A prioritized structure of $S_{pri}(n)$ space that answers a query in $Q_{pri}(n) + O(t/B)$ time, and can be updated in $U_{pri}(n)$ time;
- A max structure of $S_{max}(n)$ space that answers a query in $Q_{max}(n)$ query time, and can be updated in $U_{max}(n)$ time.

**Our Result 2**

There is a top-$k$ structure with space $S_{top}(n)$, query time $Q_{top}(n) + O(k/B)$, and update time $U_{top}(n)$ time where

- $S_{top}(n) = O(S_{pri}(n) + S_{max}(n))$ in expectation
- $Q_{top}(n) = O(Q_{pri}(n) + Q_{max}(n))$ in expectation
- $U_{top}(n) = O(U_{pri}(n) + U_{max}(n))$ in expectation

subject to mild conditions.

**Implication:** in terms of expected performance, top-$k$ is as hard as solving prioritized reporting and max reporting simultaneously.
Application 1: 1d Top-k Range Reporting

9 1 5 20 10 3 15 8 2 6

$q$
Application 1: 1d Top-k Range Reporting

**Prioritized reporting**: 3-sided range reporting

External priority search tree [Arge et al.'99]:

\[ S_{pri}(n) = O(n/B), \quad Q_{pri}(n) = O(\log_B n), \quad U_{pri}(n) = O(\log_B n) \]
Application 1: 1d Top-k Range Reporting

Max reporting

\[ 9 \quad 1 \quad 5 \quad 20 \quad 10 \quad 3 \quad 15 \quad 8 \quad 2 \quad 6 \]

q

B-tree:

\[ S_{max}(n) = O(n/B), \quad Q_{max}(n) = O(\log_B n), \quad U_{max}(n) = O(\log_B n) \]
Application 1: 1d Top-k Range Reporting

\[ S_{pri}(n) = O(n/B), \quad Q_{pri}(n) = O(\log_B n), \quad U_{pri}(n) = O(\log_B n) \]
\[ S_{max}(n) = O(n/B), \quad Q_{max}(n) = O(\log_B n), \quad U_{max}(n) = O(\log_B n) \]

Our Result 2 immediately implies

\[ S_{top}(n) = O(n/B), \quad Q_{top}(n) = O(\log_B n), \quad U_{top}(n) = O(\log_B n) \]
all in expectation

Current state of the art [Tao'14, Brodal'15] (complicated structures!)

\[ S_{top}(n) = O(n/B), \quad Q_{top}(n) = O(\log_B n), \quad U_{top}(n) = O(\log_B n) \] amortized
Application 2: 1d Top-k Stabbing

\[ q \]
Prioritized reporting: Ray stabbing

Ray stabbing structure [Tao’14]:
$S_{pri}(n) = O(n/B)$, $Q_{pri}(n) = O(\log_B n)$, $U_{pri}(n) = O(\log_B n)$ amortized
Application 2: 1d Top-k Stabbing

Max reporting

Stabbing max structure [Agarwal et al.'12]:

\[
S_{\text{max}}(n) = O(n/B), \quad Q_{\text{max}}(n) = O(\log_B n), \quad U_{\text{max}}(n) = O(\log_B n)
\]

amortized
Application 2: 1d Top-k Stabbing

\[ S_{pri}(n) = O(n/B), \quad Q_{pri}(n) = O(\log_B n), \quad U_{pri}(n) = O(\log_B n) \text{ amortized} \]
\[ S_{max}(n) = O(n/B), \quad Q_{max}(n) = O(\log_B n), \quad U_{max}(n) = O(\log_B n) \text{ amortized} \]

**Our Result 1 immediately implies**

\[ S_{top}(n) = O(n/B), \quad Q_{top}(n) = O(\log_B^2 n) \text{ worst case} \]

**Our Result 2 immediately implies**

\[ S_{top}(n) = O(n/B), \quad Q_{top}(n) = O(\log_B n), \quad U_{top}(n) = O(\log_B n) \text{ amortized} \text{ all in expectation} \]

**Current state of the art**

None!
A function $f(n)$ is **geometrically converging** if it satisfies two conditions:

- For any $n \geq B$:
  \[
  \sum_{i=0}^{h} f \left( \frac{n}{c^i} \right) = O(f(n))
  \]
  for any value $c \geq 2$, where $h$ is the largest integer $i$ satisfying $n/c^i \geq B$.

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

Theorem 1.

Suppose that there is a prioritized structure of $S_{pri}(n)$ space and query cost $Q_{pri}(n) + O(t/B)$ such that $S_{pri}(n)$ is geometrically converging, and

$$Q_{pri}(n) \geq \log_B n.$$ 

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 $S_{top}(n)$ and query time $Q_{top}(n) + O(k/B)$ with

$$S_{top}(n) = O(S_{pri}(n))$$

$$Q_{top}(n) = O \left( Q_{pri}(n) \cdot \frac{\log n}{\log B + \log \frac{Q_{pri}(n)}{\log_B n}} \right)$$
Theorem 2.

Suppose that there is

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

Then, there is a top-$k$ structure of expected space $S_{top}(n)$ and expected query time $Q_{top}(n) + O(k/B)$ with

$$S_{top}(n) = O \left( S_{pri}(n) + S_{max} \left( \frac{6n}{B \cdot Q_{pri}(n)} \right) \right)$$

$$Q_{top}(n) = O \left( Q_{pri}(n) + Q_{max}(n) \right).$$

Furthermore, if the prioritized and max structures support an update in $U_{pri}(n)$ and $U_{max}(n)$ I/Os respectively, then the top-$k$ structure supports an update in $O(U_{pri}(n) + U_{max}(n))$ expected I/Os. If any of $U_{pri}(n)$ and $U_{max}(n)$ is amortized, so is the update cost of the top-$k$ structure.
A problem is \( \lambda \)-polynomially bounded if for any input \( D \) of \( n \) elements, there are at most \( n^\lambda \) distinct outcomes for \( q(D) \) over all \( q \in Q \), where \( \lambda \) is a constant.

**Lemma 3 (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 \( q \in Q \) 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 \( q(D) \).

Foundation for our Result 1.
Lemma 4.

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 $4K$.

Foundation for our Result 2.
Reduction at a Very High Level

Structure

- Recursively sparsify the current dataset.
- Create a prioritized or max structure on each sparsified dataset.
- Create a prioritized structure on the original dataset.

Query

- Key is to handle small $k$ (for worst case) or to guess the right $\tau$ (for expected case).
- Bottom up.
Open: is top-\(k\) reporting harder than prioritized reporting?

Our conjecture is no.
My Collaborators in the Top-k Project

Cheng Sheng  

Rahul Saladi
THANK YOU!

Let Us Go Small and Sweet