Lower Bound Framework for Differentially Private and Oblivious Data Structures

Giuseppe Persiano and **Kevin Yeo**

Google | Columbia University | Hippocratica Civitas Studium Salerni

Outline

Oblivious and Differentially Private Data Structures

Lower Bound Framework

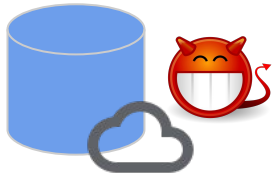
New Proof Techniques

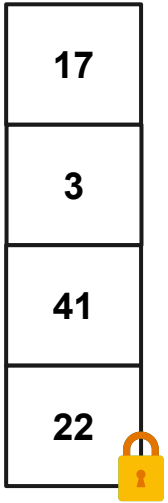
Oblivious and Differentially Private Data Structures

Oblivious RAM

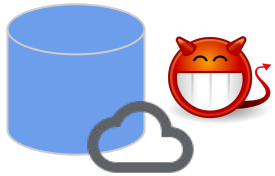
Oblivious RAM

Oblivious RAM

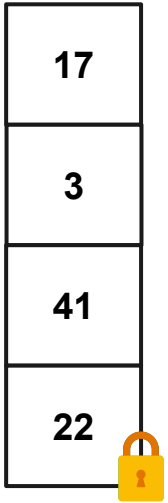




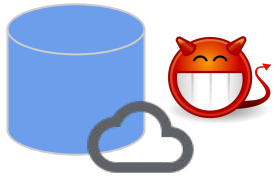
Oblivious RAM

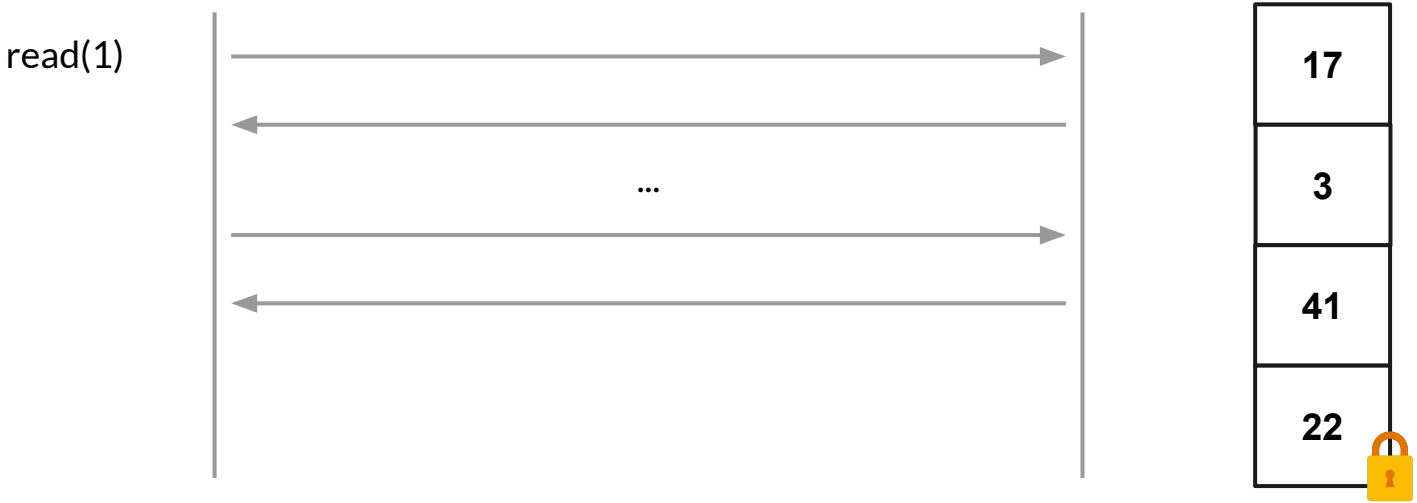


read(1)

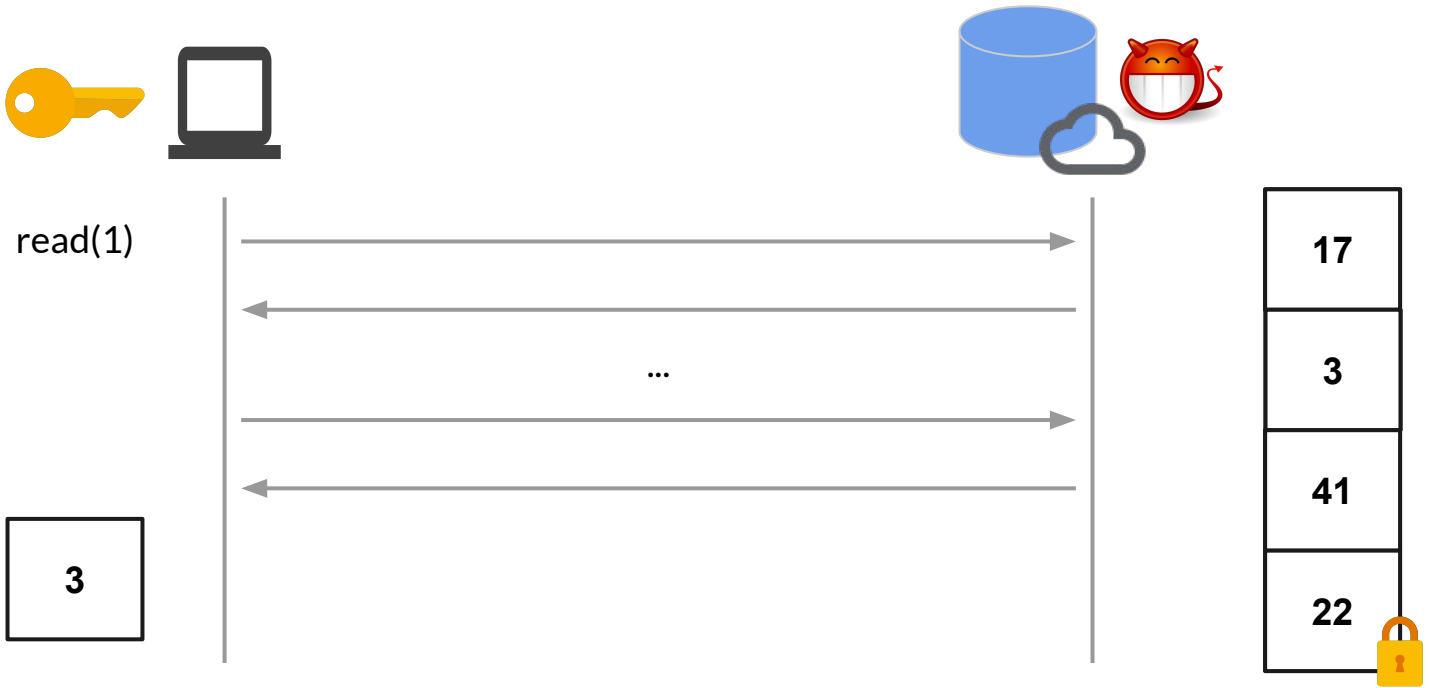


Oblivious RAM

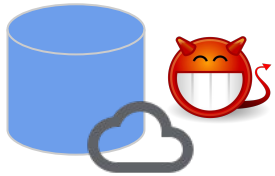


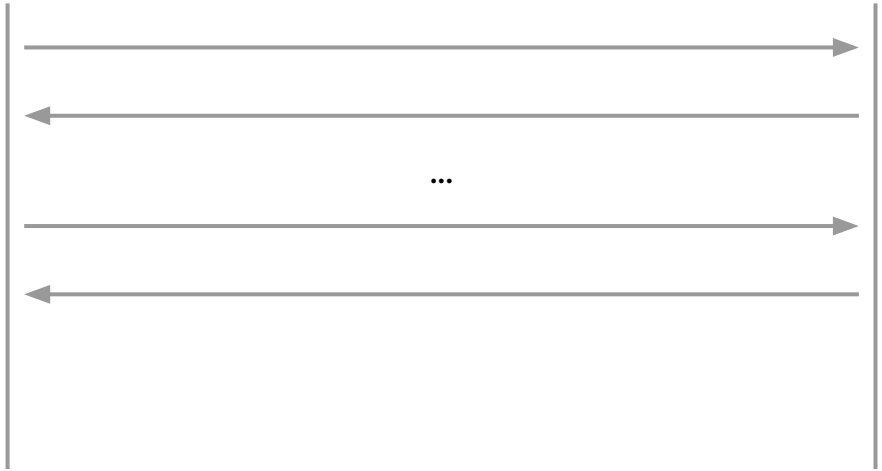


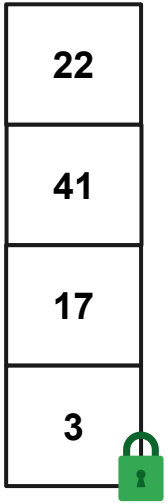
Oblivious RAM



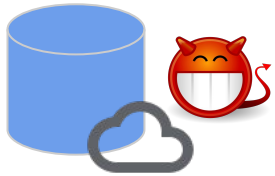
Oblivious RAM





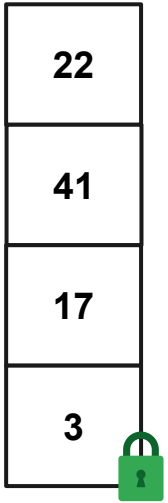


Oblivious RAM



read(1)

write(2, X)

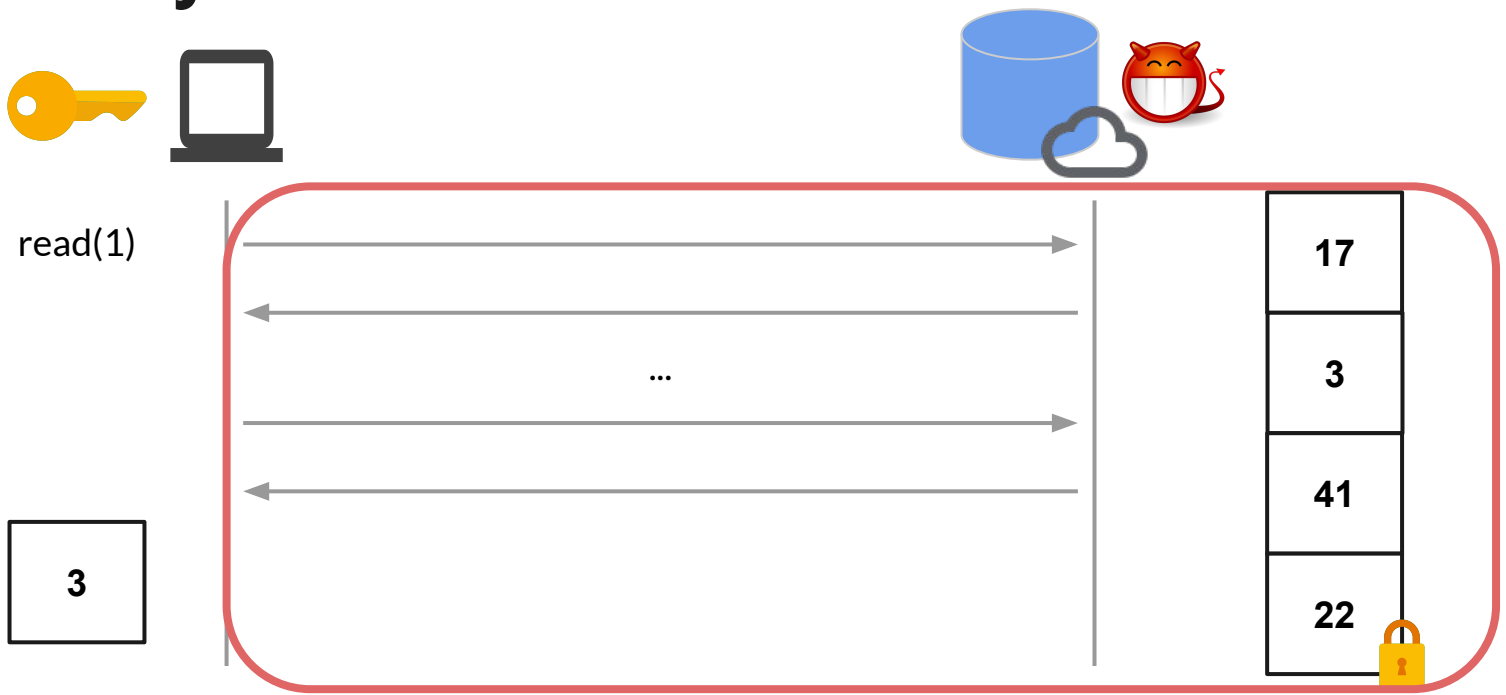


Oblivious RAM Security

Definition. For any sequence of equal-length operations O_1 and O_2, the adversary's view $ObvDS(O_1)$ and $ObvDS(O_2)$ from a construction ODS must be indistinguishable to a computational adversary A:

$$Pr[A(\ ObvDS(O_1)\ ) = 1] \cong Pr[A(\ ObvDS(O_2)\ ) = 1]$$

Adversary's View



Differentially Private RAM

Oblivious RAM

read(10), write(17), read(2), read(3), write(54)

Differentially Private RAM

read(10), write(17), read(2), read(3), write(54)

Differentially Private RAM

Oblivious RAM	Differentially Private RAM
read(10), write(17), read(2), read(3), write(54)	read(10), write(17), read(2), read(3), write(54)
write(13), write(93), read(9), read(7), read(62)	

Differentially Private RAM

Oblivious RAM	Differentially Private RAM
read(10), write(17), read(2), read(3), write(54)	read(10), write(17), read(2), read(3), write(54)
write(13), **write(93)**, **read(9)**, **read(7)**, **read(62)**	read(10), write(17), read(2), **write(18)**, write(54)

Differentially Private RAM Security

Definition. For any sequence of equal-length operations O_1 and O_2 that differ in at most one operation, the adversary's view $DPDS(O_1)$ and $DPDS(O_2)$ from an (ε, δ)-Differentially Private data structure construction DPDS must satisfy the following for any computational adversary A:

$$Pr[A(\ DPDS(O_1)\ ) = 1] \leq e^{\varepsilon}\ Pr[A(\ DPDS(O_2)\ ) = 1] + \delta$$

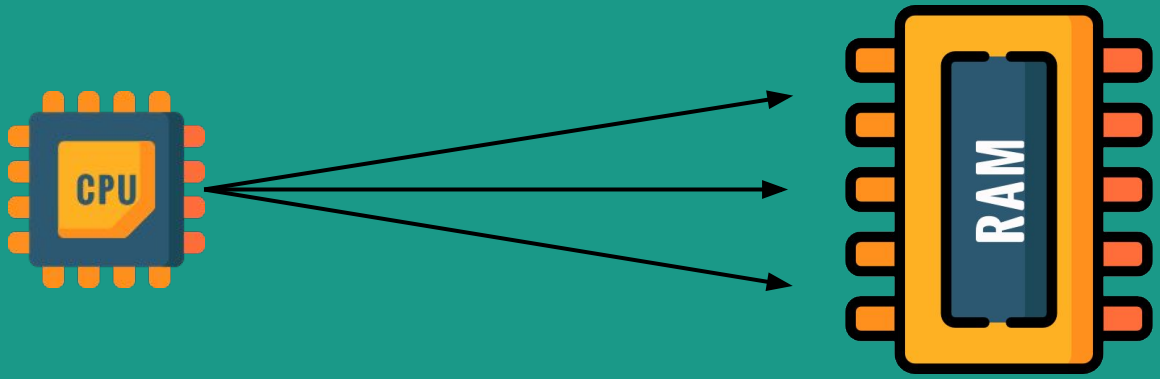
Other Data Structures

- RAMs (Arrays)
- Sets
  - Enable checking membership of entries
- Predecessor and Successor
  - Return largest element smaller than a query input
- Disjoint Sets (Union-Find)
  - Maintain sets of sets enabling merging and querying

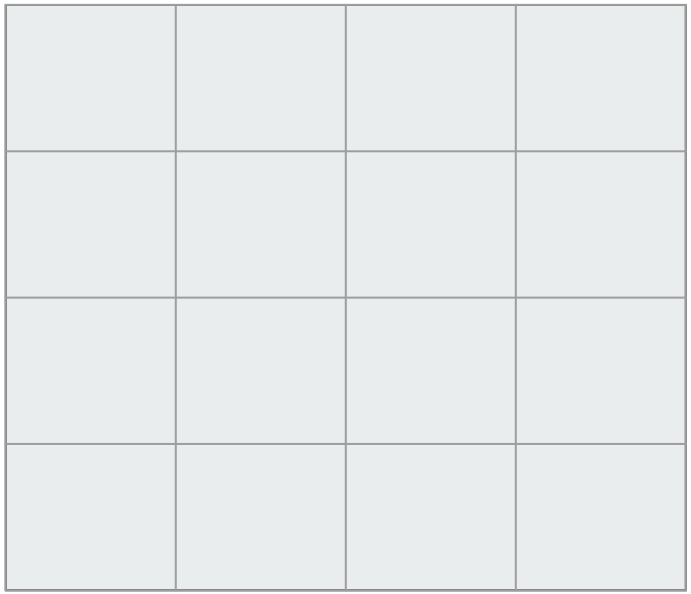
Parameters

- **w**: Size of memory cells in the system
- **b:** Size of query outputs
  - RAMs (Arrays)**:** b = array entry size
  - Sets: b = 1
  - Predecessor and Successor: b = O(log |U|)
  - Disjoint Sets (Union-Find): b = O(log n)

Cell Probe Model



Cell Probe Model



Cell Probe Model

- Only cost is probing (read/write) a server cell of **w** bits

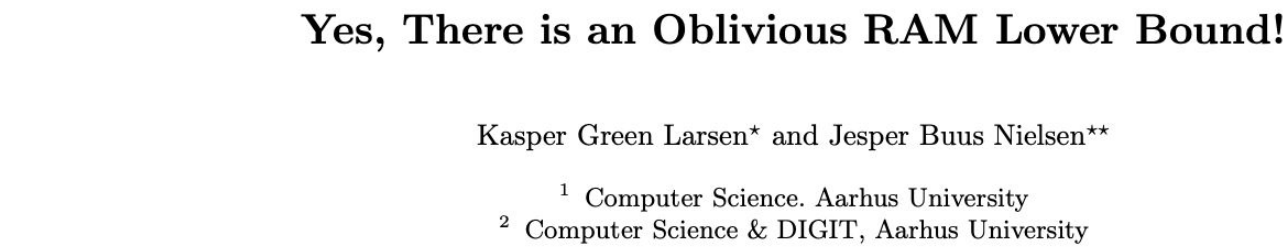
Cell Probe Model

Cell Probe Model

- Computation is free
- Generating randomness (or accessing a random oracle) is free
- Accessing client storage is free
- Very weak cost model → Very strong lower bounds

ORAM Lower Bound (b ≥ w)

Theorem [Larsen, Nielsen '18]. In the cell probe model, oblivious RAMs require $\Omega(b/w * \log N)$ overhead.



Yes, There is an Oblivious RAM Lower Bound!

Kasper Green Larsen* and Jesper Buus Nielsen**

[1] Computer Science. Aarhus University
[2] Computer Science & DIGIT, Aarhus University

Abstract. An Oblivious RAM (ORAM) introduced by Goldreich and Ostrovsky [JACM'96] is a (possibly randomized) RAM, for which the memory access pattern reveals no information about the operations performed. The main performance metric of an ORAM is the bandwidth overhead, i.e., the multiplicative factor extra memory blocks that must be accessed to hide the operation sequence. In their seminal paper in-

ORAM Lower Bound (b < w)

Theorem [Komargodski, Lin '21]. In the cell probe model, oblivious RAMs require $\Omega(\log N/\log(w/b))$ overhead.

A Logarithmic Lower Bound for Oblivious RAM
(for all parameters)

Ilan Komargodski* Wei-Kai Lin†

June 13, 2021

Abstract

An Oblivious RAM (ORAM), introduced by Goldreich and Ostrovsky (J. ACM 1996), is a (probabilistic) RAM that hides its access pattern, i.e., for every input the observed locations

DPRAM Lower Bound (b ≥ w)

Theorem [Persiano, Y '19]. In the cell probe model, for any $\varepsilon = O(1)$ and $\delta \leq 1/3$, Differentially Private RAMs must use $\Omega(b/w * \log n)$ overhead.

DPRAM Lower Bound (b < w)

Open Problem [Komargodski and Lin ‘21]: Can we prove logarithmic lower bounds for DPRAMs for the setting of b < w?

DPRAM Lower Bounds

Theorem. In the cell probe model, for any $\varepsilon = O(1)$ and $\delta = O(1)$, DPRAMs must use overhead $\Omega(\log n/\log(w/b))$.

- First DPRAM logarithmic lower bound when $b < w$
- Resolves open problem of Komargodski and Lin [Crypto'21]
- First DPRAM lower bounds in multi-server setting (for any b, w)

Rich Line of Prior Works

Oblivious RAMs: [Larsen, Nielsen ‘18], [HKKS ‘19], [Komargodski, Lin’ 21]

Differentially Private RAMs: [Persiano, Y ‘19]

Weaker Privacy Notions:

- Encrypted Search Leakage [PPY‘20]
- Multi-Server [LSY ‘20]

Other Data Structures:

- Stacks, Queues, Heaps, Search Trees [JLN ‘19]
- Near-Neighbor Search [LMWY‘20]

Lower Bound Framework

Theorem. Any data structure problem P satisfying the following two conditions requires overhead ~Ω(log n) in the cell probe model.

1. **[Large Information Retrieval]:** Queries are sufficiently complex to enable retrieving information
2. **[Event Transfer Probability]:** Certain events that are adversarially observable must transfer across operational sequences.

New Lower Bounds from Framework

- RAMs (Arrays)
  - Plaintext: O(1) -> **Differential Privacy: ~Ω(log n)**
- Sets
  - Plaintext: O(1) -> **Differential Privacy: ~Ω(log n)**
- Predecessor and Successor
  - Plaintext: O(log log n) -> **Differential Privacy: ~Ω(log n)**
- Disjoint Sets (Union-Find)
  - Plaintext: ~O(1) -> **Differential Privacy: ~Ω(log n)**

Lower Bound Framework

New Lower Bounds from Framework

- RAMs (Arrays)
  - Plaintext: O(1) -> **Differential Privacy: ~Ω(log n)**
- Sets
  - Plaintext: O(1) -> **Differential Privacy: ~Ω(log n)**
- Predecessor and Successor
  - Plaintext: O(log log n) -> **Differential Privacy: ~Ω(log n)**
- Disjoint Sets (Union-Find)
  - Plaintext: ~O(1) -> **Differential Privacy: ~Ω(log n)**

Lower Bound for DP Sets

Theorem. In the cell probe model, for any $\varepsilon = O(1)$ and $\delta = O(1)$, DP set data structures must use overhead $\sim\Omega(\log n)$.

Large Information Retrieval

Definition. A data structure problem P has large information retrieval if there exists a random update sequence $U = (u_1, u_2, \ldots, u_n)$ such that for any consecutive subset of L updates $u_a, \ldots, u_{a+L-1}$, there exists a query set Q of size O(L) whose answers have high entropy with respect to $u_a, \ldots, u_{a+L-1}$.

Formally, let A(U, Q) be the answers of all queries in Q immediately executed after U. Then,

$$H(A(U, Q) \mid u_1, \ldots, u_{a-1}, u_{a+L}, \ldots, u_n) / L = \Omega(v)$$

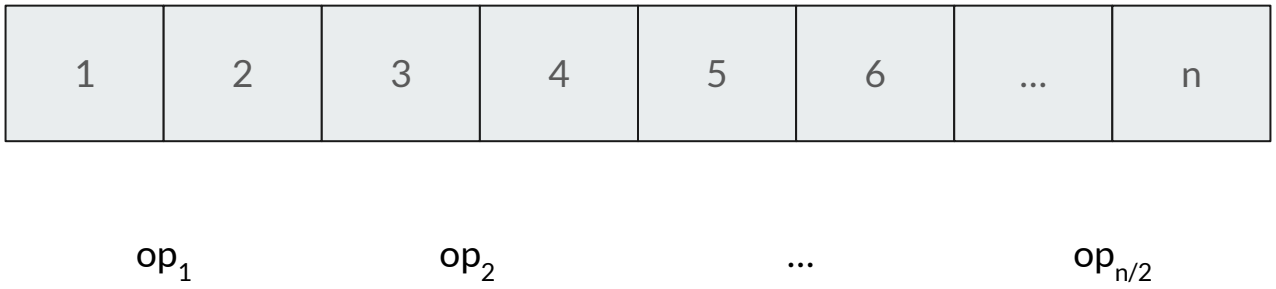
for some $v >= 0$.

Large Information Retrieval

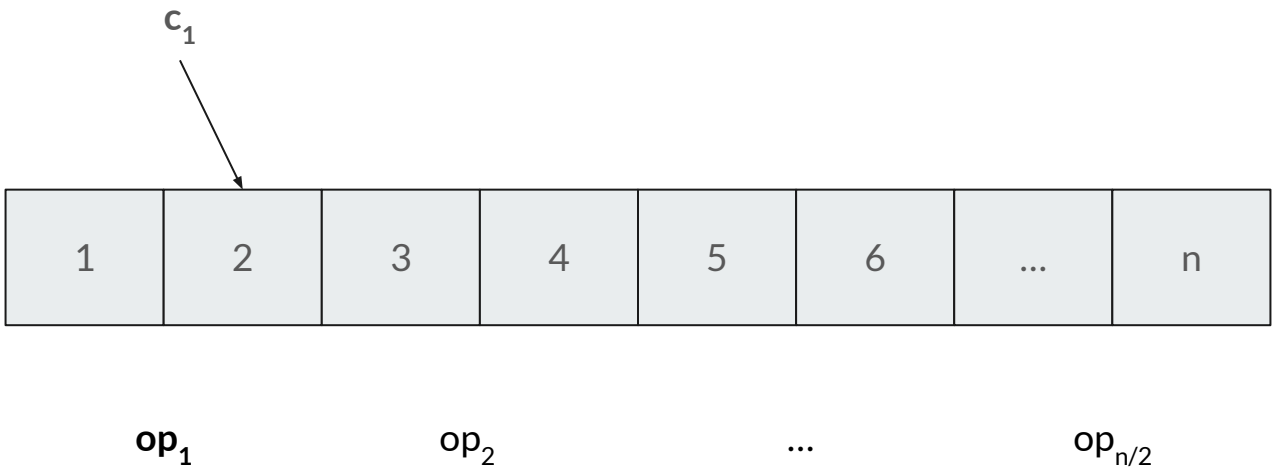
Large Information Retrieval

op_1

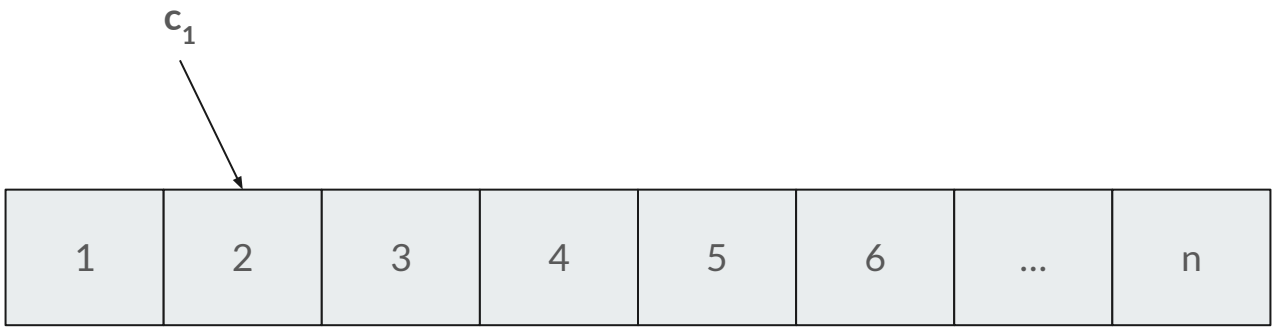
Large Information Retrieval



Large Information Retrieval

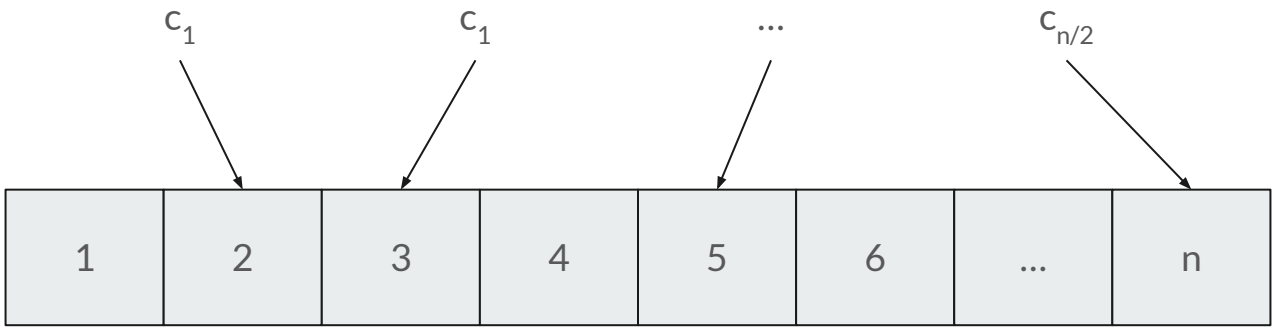


Large Information Retrieval



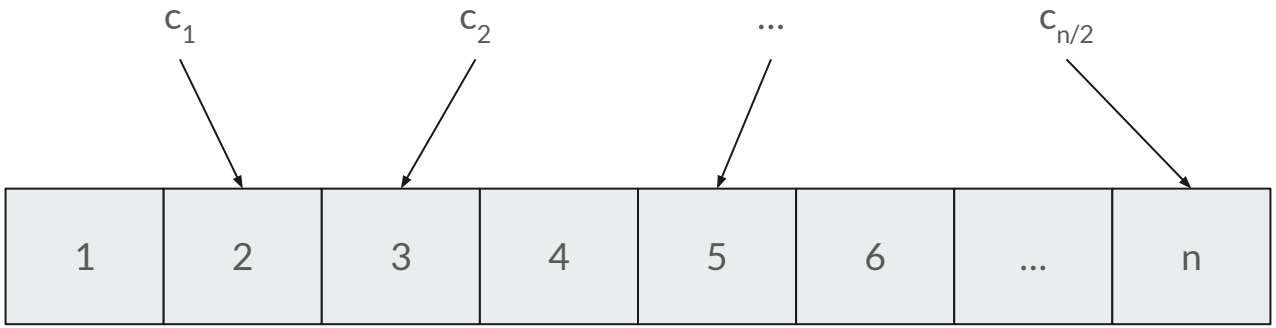
insert(2) op_2 ... $op_{n/2}$

Large Information Retrieval



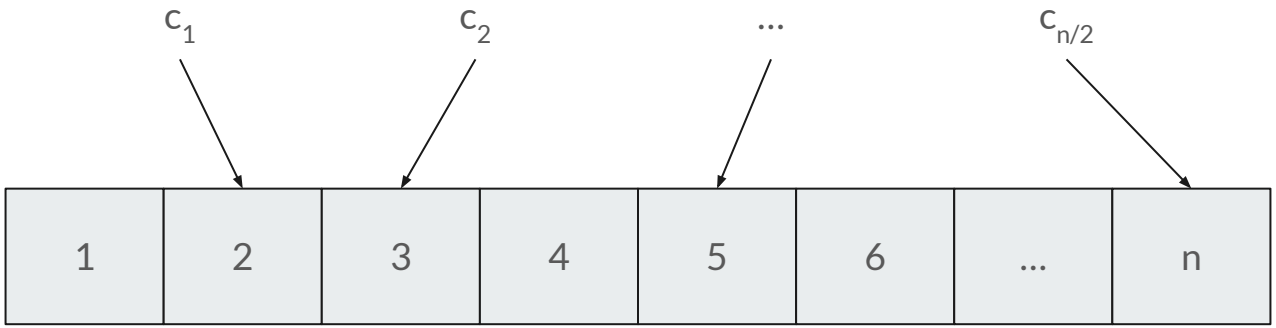
insert(2) op_2 ... $op_{n/2}$

Large Information Retrieval



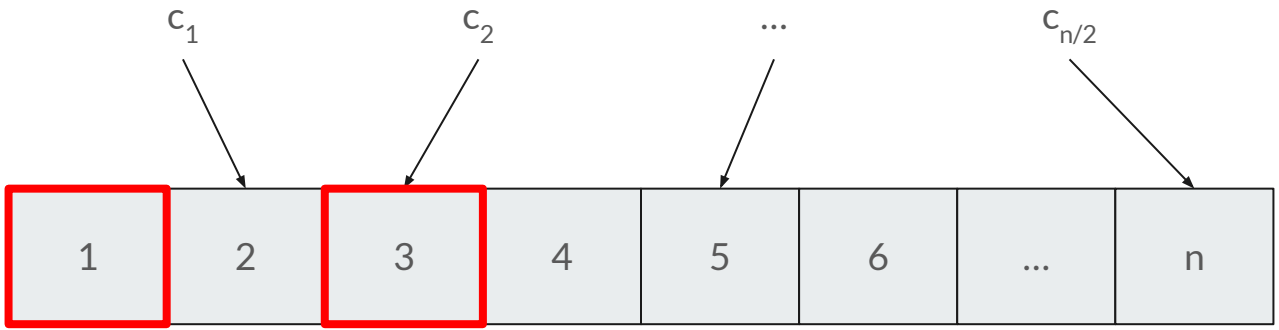
insert(2) insert(3) ... insert(n)

Large Information Retrieval



insert(2) **insert(3)** ... insert(n)

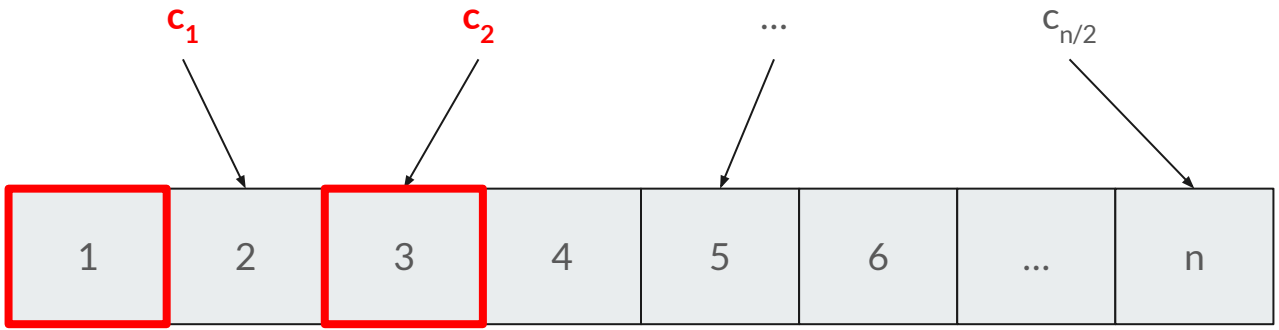
Large Information Retrieval



insert(2) **insert(3)** ... insert(n)

query(1) **query(3)**

Large Information Retrieval



insert(2)	insert(3)	...	insert(n)
query(1)	query(3)		

Lower Bound for DP Sets

Proof.

Step 1: Large Information Retrieval

- $U = (\text{insert}(1 + c_1), \text{insert}(3 + c_2), \ldots, \text{insert}(n - 1 + c_{(n-1)/2})$ where each c_i is a random coin flip (bit)
- For any consecutive L updates: $\text{insert}(2a + c_a), \text{insert}(2a + 2 + c_{a+1}), \ldots, \text{insert}(2a + 2(L-1) + c_{a+L-1})$, construct query set $Q = \{2a, 2a + 2, \ldots, 2a + 2(L-1)\}$ of L queries.
- Answers to Q enable retrieving all coin flips $c_a, c_{a+1}, \ldots, c_{a+L-1}$.
- $H(A(U, Q) \mid u_1, \ldots, u_{a-1}, u_{a+L}, \ldots, u_n) = L$

Event Transfer Probability

Definition. For any update sequence U and query q, let E(U, q) be an event observable by an adversary. Furthermore, suppose that $\Pr[E(U, q)] = \Omega(1)$. Then, for any query q', it must be that

$$\Pr[E(U, q')] = \Omega(\Pr[E(U, q)]).$$

Lower Bound for DP Sets

Proof.

Step 2: Event Transfer Probability

- By DP, for events E(U, q) and E(U, q') for update sequence U and queries q and q', it must satisfy

$$\Pr[E(U, q) = 1] \leq e^{\varepsilon} \Pr[E(U, q')] + \delta$$

- In other words,

$$(\Pr[E(U, q) = 1] / e^{\varepsilon}) - \delta \leq \Pr[E(U, q')]$$

Lower Bound for DP Sets

Generic lemmas for:
- Oblivious
- Differential Privacy
- Multi-Server
- Encrypted Search

Proof.

Step 2: Event Transfer Probability

- By DP, for events E(U, q) and E(U, q') for update sequence U and queries q and q', it must satisfy

$$\Pr[E(U, q) = 1] \leq e^{\varepsilon} \Pr[E(U, q')] + \delta$$

- In other words,

$$(\Pr[E(U, q) = 1] / e^{\varepsilon}) - \delta \leq \Pr[E(U, q')]$$

Lower Bound for DP Sets

Theorem. In the cell probe model, for any $\varepsilon = O(1)$ and $\delta = O(1)$, DP set data structures must use overhead $\sim\Omega(\log n)$.

For sets, we showed that one can extract $v = 1$ bit for each of the L updates on average.

Lower Bound for DP Sets

Theorem. In the cell probe model, for any $\varepsilon = O(1)$ and $\delta = O(1)$, DP set data structures must use overhead $\sim\Omega(\log n)$.

For sets, we showed that one can extract $v = 1$ bit for each of the L updates on average.

General Theorem. Any data structure problem P with **b**-bit query outputs satisfying the two conditions requires overhead **$\sim\Omega(v/b * \log n)$** in the cell probe model.

Is this framework tight?

Question. Are there any data structures that would have logarithmic lower bounds for all choices of b and w that do not fit into the framework?

Stacks and Queues

- Cannot satisfy Large Information Retrieval property of our framework.

Stacks and Queues

- Cannot satisfy Large Information Retrieval property of our framework.
- **Theorem [JLN ‘20]**: Oblivious stacks and queues require Ω(log n) overhead when b >= w
- **Theorem [Komargodski and Shi ‘21]**: DP stacks and queues require O(b/w * loglog n) overhead.

Stacks and Queues

- Cannot satisfy Large Information Retrieval property of our framework.
- **Theorem [JLN '20]**: Oblivious stacks and queues require Ω(b/w * log n) overhead.
- **Theorem [Komargodski and Shi '21]**: DP stacks and queues require O(b/w * loglog n) overhead.
- **Open Problem:** What is the right overhead for oblivious stacks and queues when b < w?

Oblivious Stacks and Queues

Theorem. Oblivious stacks and queues have $O(b/w * \log n)$ overhead.

- Inherent that it cannot satisfy our framework's conditions.
- Sub-constant overhead possible when $b/w = o(1/\log n)$
- First separation result for online oblivious data structures between settings of $b = w$ and $b < w$

New Proof Techniques

Chronogram

The Cell Probe Complexity of Dynamic Data Structures

Michael L. Fredman [1]

Bellcore and
U.C. San Diego

Michael E. Saks [2]

U.C. San Diego,
Bellcore and
Rutgers University

1. Summary of Results

Dynamic data structure problems involve the representation of data in memory in such a way as to permit certain types of modifications of the data (**updates**) and certain types of questions about the data (**queries**). This paradigm encompasses many fundamental problems in computer science.

The purpose of this paper is to prove new lower and upper bounds on the time per operation to implement solutions to some familiar dynamic data structure problems including list representation, subset ranking, partial sums, and the set union problem . The main features of our lower bounds are:

(1) They hold in the *cell probe* model of computation (A. Yao [18]) in which the time complexity of a sequential computation is defined to be the number of words of memory that are accessed. (The number of bits b in a single word of memory is a parameter of the model). All other computations are free. This model is at least as powerful as a random access machine and allows for unusual representation of data, indirect addressing etc. This contrasts with most previous lower bounds which are

register size from $\log n$ to polylog(n) only reduces the time complexity by a constant factor. On the other hand, decreasing the register size from $\log n$ to 1 increases time complexity by a $\log n$ factor for one of the problems we consider and only a $\log\log n$ factor for some other problems.

The first two specific data structure problems for which we obtain bounds are:

List Representation. This problem concerns the represention of an ordered list of at most n (not necessarily distinct) elements from the universe $U = \{1, 2, ..., n\}$. The operations to be supported are **report**(k), which returns the k^{th} element of the list, **insert**(k, u) which inserts element u into the list between the elements in positions $k-1$ and k, **delete**(k), which deletes the k^{th} item.

Subset Rank. This problem concerns the representation of a subset S of $U = \{1, 2, ..., n\}$. The operations that must be supported are the updates "**insert** item j into the set" and "**delete** item j from the set" and the queries **rank**(j), which returns the number of elements in S that are less than or equal to j.

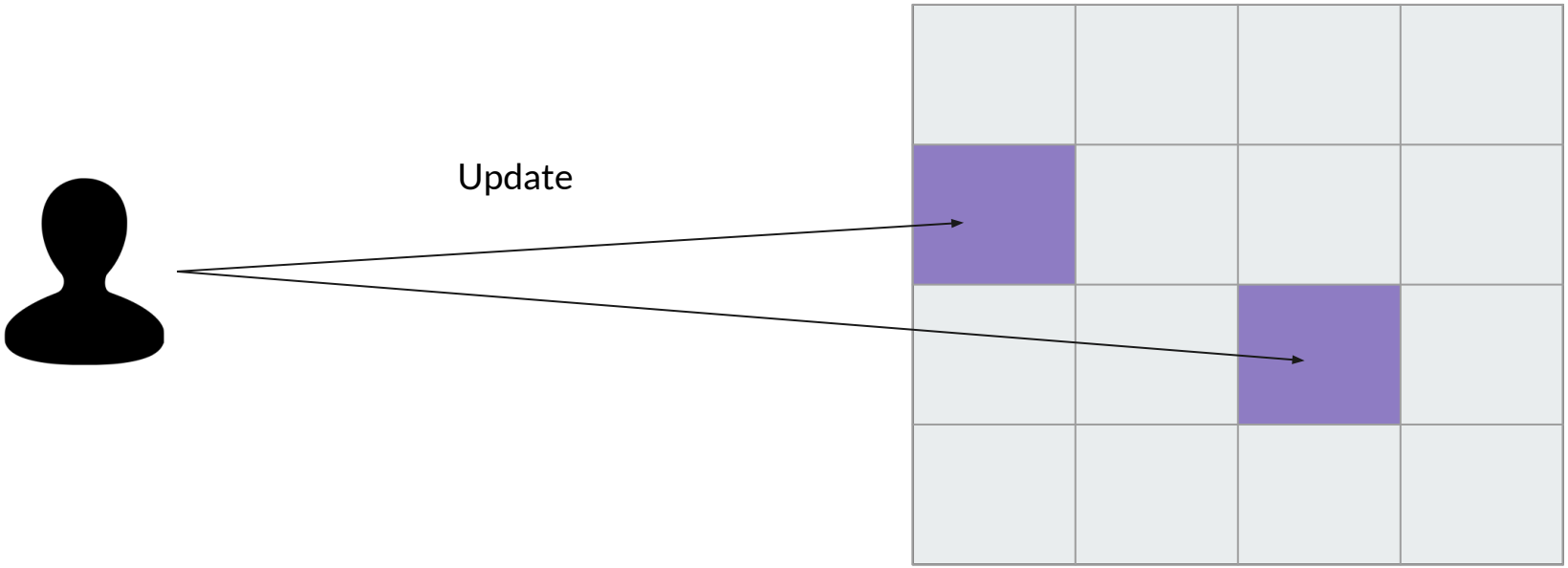
Chronogram

- Leveraged to prove Ω(b/w * log n) lower bound for DPRAMs [Persiano, Y ‘20]
- Used for super-logarithmic lower bounds for (statistically) oblivious near-neighbor search [LMWY ‘20]

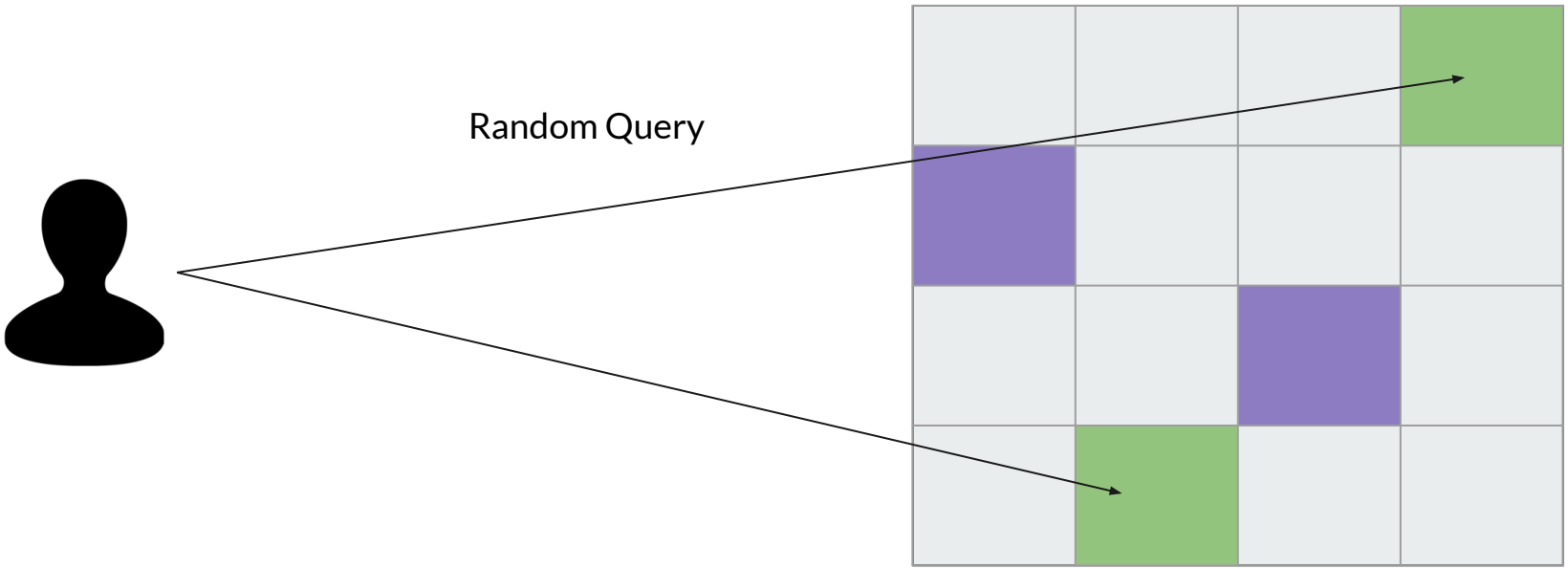
New Proof Techniques

1. Improved encoding of data structure answers when queries and updates are too-efficient
2. Showed that majority of answers can be encoded for “free” as they are unchanged by a too-efficient-to-be-true data structure.

New Proof Techniques



New Proof Techniques



New Proof Techniques

1. Improved encoding of data structure answers when queries and updates are too-efficient
2. Showed that majority of answers can be encoded for “free” as they are unchanged by a too-efficient-to-be-true data structure.

Questions?

Email: kwlyeo@cs.columbia.edu or giuper@gmail.com