Separating RAM and Multitape Turing Machines with Short Random Oracles



Lijie Chen
OpenAI & UC Berkeley



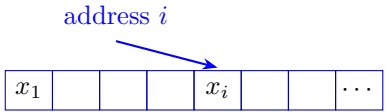
Yichuan Wang
UC Berkeley

FOCS 2026 November 9, 2026

RAM vs Multitape TM

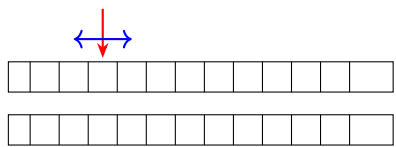
Random-access machine (RAM)

- $O(\log n)$-bit words.
- Access any addressed word in one step.



Multitape Turing machine (MTM)

- A fixed number of tapes. ($O(1)$ many)
- Each head moves at most one cell per step.

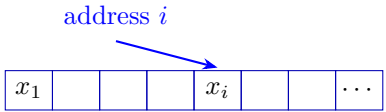


Standard simulation: T-time RAM can be simulated in $\widetilde{O}(T^2)$-time Multitape-TM. [Cook–Reckhow '73]

RAM vs Multitape TM

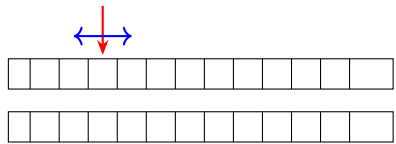
Random-access machine (RAM)

- $O(\log n)$-bit words.
- Access any addressed word in one step.



Multitape Turing machine (MTM)

- A fixed number of tapes. ($O(1)$ many)
- Each head moves at most one cell per step.

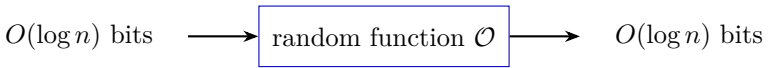


Standard simulation: T-time RAM can be simulated in $\widetilde{O}(T^2)$-time Multitape-TM. [Cook–Reckhow '73]

Can we replace the quadratic overhead by $T^{1.99}$?

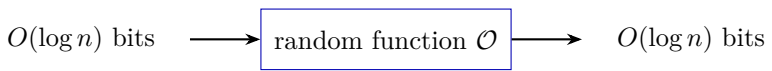
Our Result: A Nearly Quadratic Separation with Short Random Oracles

Both machines receive the same **short** random oracle:



Our Result: A Nearly Quadratic Separation with Short Random Oracles

Both machines receive the same **short** random oracle:

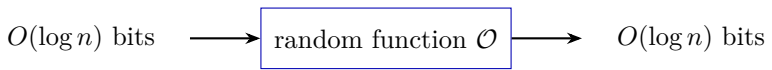


Our Theorem (informal)

There is a 1-bit-output function $\mathsf{Hard}^{\mathcal{O}}(x)$ computable in $\widetilde{O}(n)$ RAM time, but requires $\widetilde{\Omega}(n^2)$ Multitape-TM time to success w.p. $1 - o(1)$ over $x, \mathcal{O}$.

Our Result: A Nearly Quadratic Separation with Short Random Oracles

Both machines receive the same **short** random oracle:



Our Theorem (informal)

There is a 1-bit-output function $\mathsf{Hard}^{\mathcal{O}}(x)$ computable in $\widetilde{O}(n)$ RAM time, but requires $\widetilde{\Omega}(n^2)$ Multitape-TM time to success w.p. $1 - o(1)$ over $x, \mathcal{O}$.

(Technically we can prove $1/2 + o(1)$ hardness, but for simplicity we'll talk about $1 - o(1)$ today)

More on RAM vs Multitape vs One Tape vs Space

Source		Simulation bound	
T-time RAM	$\subseteq$	$\widetilde{O}(T^2)$-time multitape	
T-time RAM	$\subseteq$	$\widetilde{O}(T^2)$-time one-tape	
T-time multitape	$\subseteq$	$\widetilde{O}(T^2)$-time one-tape	
T-time RAM	$\subseteq$	$\widetilde{O}(T^{2/3})$ space	[OpenAI '26]
T-time multitape	$\subseteq$	$\widetilde{O}(T^{1/2})$ space	[Williams '25]
T-time one-tape	$\subseteq$	$\widetilde{O}(T^{2/5})$ space	[OpenAI '26]

More on RAM vs Multitape vs One Tape vs Space

Source		Simulation bound	
T-time RAM	$\subseteq$	$\widetilde{O}(T^2)$-time multitape	
T-time RAM	$\subseteq$	$\widetilde{O}(T^2)$-time one-tape	
T-time multitape	$\subseteq$	$\widetilde{O}(T^2)$-time one-tape	
T-time RAM	$\subseteq$	$\widetilde{O}(T^{2/3})$ space	[OpenAI '26]
T-time multitape	$\subseteq$	$\widetilde{O}(T^{1/2})$ space	[Williams '25]
T-time one-tape	$\subseteq$	$\widetilde{O}(T^{2/5})$ space	[OpenAI '26]

Oracles and Space: Williams's simulation extends to short oracle queries, but not to length-n queries.

Hard Function: Pointer Chasing

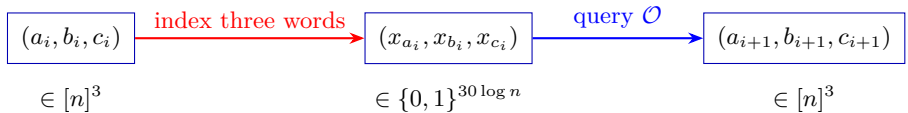
Input: n words $x = (x_1, \ldots, x_n)$, each $|x_i| = 10 \log n$.
(From here on, n counts words)

Hard Function: Pointer Chasing

Input: n words $x = (x_1, \ldots, x_n)$, each $|x_i| = 10 \log n$.
(From here on, n counts words)

Oracle: $\mathcal{O}\colon \{0,1\}^{30 \log n} \to [n]^3$

- Start at $(a_1, b_1, c_1) := (1, 1, 1)$;



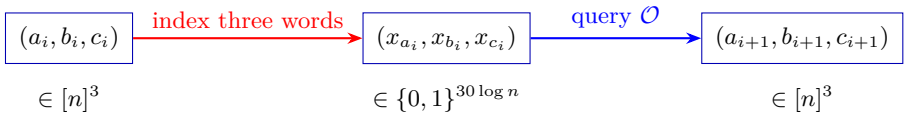
- Repeat the update $n - 1$ times;
- Output **the last bit** of the final tuple (a_n, b_n, c_n).

Hard Function: Pointer Chasing

Input: n words $x = (x_1, \ldots, x_n)$, each $|x_i| = 10 \log n$.
(From here on, n counts words)

Oracle: $\mathcal{O}\colon \{0,1\}^{30 \log n} \to [n]^3$

- Start at $(a_1, b_1, c_1) := (1, 1, 1)$;



- Repeat the update $n - 1$ times;
- Output **the last bit** of the final tuple (a_n, b_n, c_n).

Easy for a RAM:

$$(n-1) \text{ rounds } \times \ O(\log n) \text{ time each round } = \ O(\text{input length}).$$

Proof Step 0: Fixed Input, Random Oracle

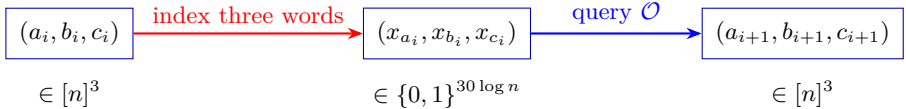
Fix a deterministic Multitape TM M running in $o(n^2)$ time.

Fix one incompressible input $x = (x_1, \ldots, x_n)$

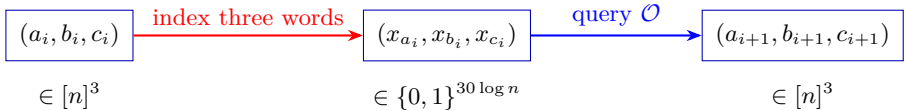
Each $|x_i| = 10 \log n$, and $\mathsf{K}(x) \geq |x| - O(\log n)$.
K: Kolmogorov complexity.

Only consider randomness over the oracle $\mathcal{O}$.

Proof Step 1: The Queries Must be Made, and in Order

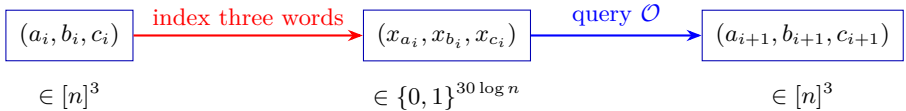


Proof Step 1: The Queries Must be Made, and in Order



1. **Unlikely to skip a required query.**
   The incomplete queries leave the output bit nearly unbiased.

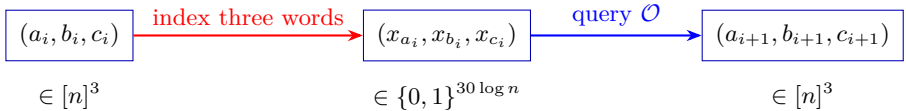
Proof Step 1: The Queries Must be Made, and in Order



1. **Unlikely to skip a required query.**
   The incomplete queries leave the output bit nearly unbiased.

2. **Unlikely to make a later query in advance.**
   A guess become a future key with probability only $1/n^2$.
   Take union bound over all oracle queries (at most $o(n^2)$ many), the probability is $< o(1)$. (3-tuple is needed to avoid collision)

Proof Step 1: The Queries Must be Made, and in Order



1. **Unlikely to skip a required query.**
   The incomplete queries leave the output bit nearly unbiased.

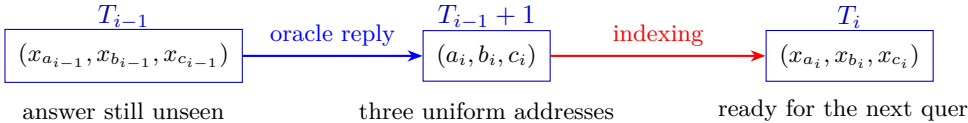
2. **Unlikely to make a later query in advance.**
   A guess become a future key with probability only $1/n^2$.
   Take union bound over all oracle queries (at most $o(n^2)$ many), the probability is $< o(1)$. (3-tuple is needed to avoid collision)

What this gives us: Answer correctly w.p. $1 - o(1)$ requires *completing the required queries in order* w.p. $1 - o(1)$.

Proof Step 2 (Core): Large Gap Between Queries

Stopping times $0 = T_0 < T_1 < T_2 < \cdots < T_{n-1}$**:**

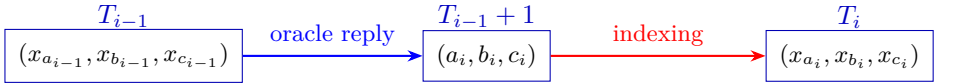
T_i $(1 \le i \le n-1)$ is the timestamp of the first query of $\mathcal{O}(x_{a_i}, x_{b_i}, x_{c_i})$.



Proof Step 2 (Core): Large Gap Between Queries

Stopping times $0 = T_0 < T_1 < T_2 < \cdots < T_{n-1}$:

T_i $(1 \le i \le n-1)$ is the timestamp of the first query of $\mathcal{O}(x_{a_i}, x_{b_i}, x_{c_i})$.

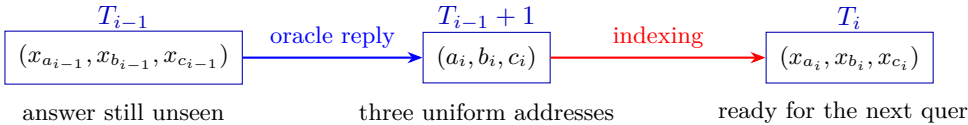


Core Lemma

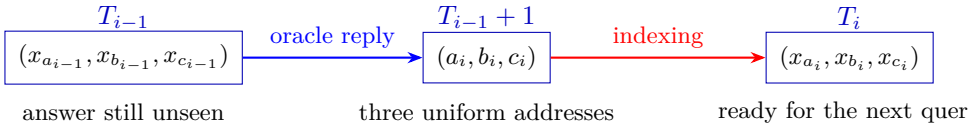
For any $2 \le i \le n-1$,

$$\Pr_{\mathcal{O}}[T_i - T_{i-1} < o(n)] \le 0.99.$$

Proof Step 2 (Core): Large Gap Between Queries

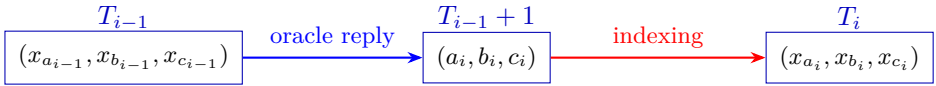


Proof Step 2 (Core): Large Gap Between Queries

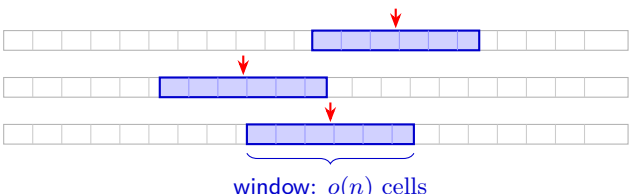


Fix computation history up to T_{i-1}.
What if $T_i - T_{i-1}$ is always $< o(n)$?

Proof Step 2 (Core): Large Gap Between Queries

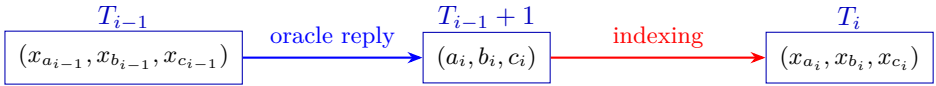


Fix computation history up to T_{i-1}.
What if $T_i - T_{i-1}$ is always $< o(n)$?

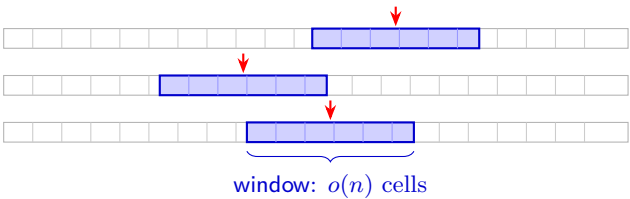


During time T_{i-1} to T_i, only contents in window is visible to the TM

Proof Step 2 (Core): Large Gap Between Queries



Fix computation history up to T_{i-1}.
What if $T_i - T_{i-1}$ is always $< o(n)$?



During time T_{i-1} to T_i, only contents in window is visible to the TM

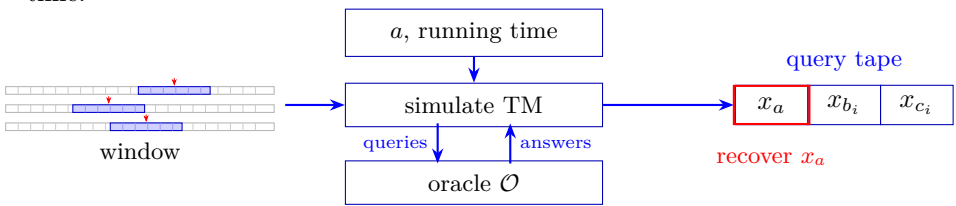
Proof Idea: Given window. For any oracle answer (a_i, b_i, c_i), we can simulate the TM and retrieve $(x_{a_i}, x_{b_i}, x_{c_i})$;
Enumerate all possible answers (a_i, b_i, c_i), we can retrieve the whole x;
Thus, we compress x into $o(n)$ bits, contradicting $\mathsf{K}(x)$ is large.

Proof Step 2 (Core): Compressing x using window (Details)

Proof Step 2 (Core): Compressing x using window (Details)

Short Description of x_a:

1. Restore window. Set the first address in $\mathcal{O}$'s pending reply to a.
2. Run for $T_i - T_{i-1}$ steps.
3. Read x_a from the query tape at that time.



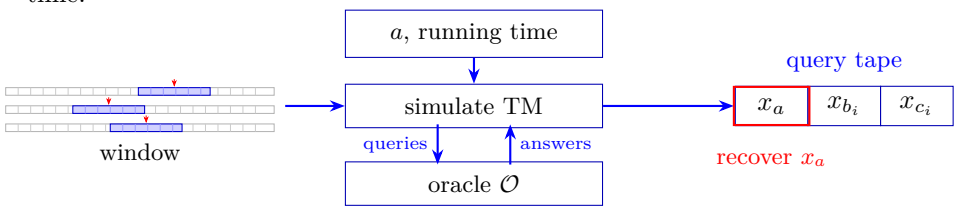
Proof Step 2 (Core): Compressing x using window (Details)

Short Description of x_a:

1. Restore window. Set the first address in $\mathcal{O}$'s pending reply to a.
2. Run for $T_i - T_{i-1}$ steps.
3. Read x_a from the query tape at that time.

Description length:

1. window ($o(n)$ bits)
2. a ($O(\log n)$ bits)
3. $T_i - T_{i-1}$ ($O(\log n)$ bits)
4. $\mathcal{O}$



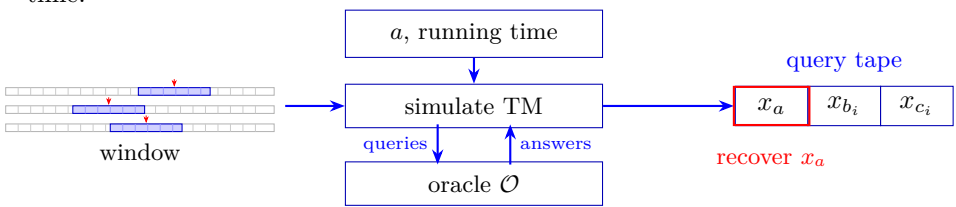
Proof Step 2 (Core): Compressing x using window (Details)

Short Description of x_a:

1. Restore window. Set the first address in $\mathcal{O}$'s pending reply to a.
2. Run for $T_i - T_{i-1}$ steps.
3. Read x_a from the query tape at that time.

Description length:

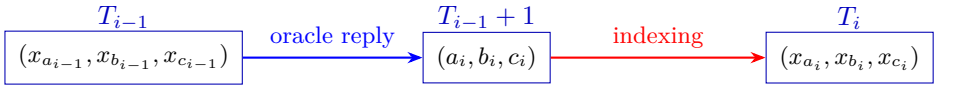
1. window $(o(n)$ bits)
2. a $(O(\log n)$ bits)
3. $T_i - T_{i-1}$ $(O(\log n)$ bits)
4. $\mathcal{O}$



What We Get

$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] < o(n)$$

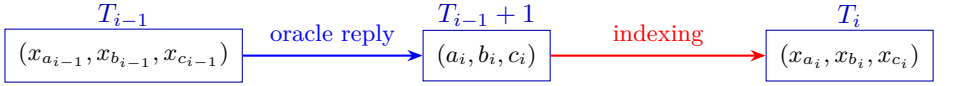
Proof Step 2 (Core): Contradicting High $\mathsf{K}(x)$



If $T_i - T_{i-1}$ is always $< o(n)$:

$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] < o(n)$$

Proof Step 2 (Core): Contradicting High $\mathsf{K}(x)$



If $T_i - T_{i-1}$ is always $< o(n)$:

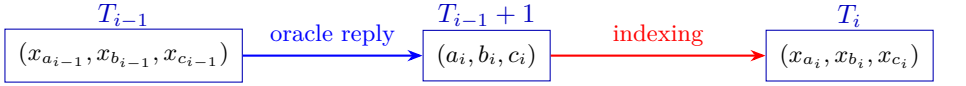
$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] < o(n)$$

Fact: a random string does not save description length

$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] \geq \mathsf{K}(x) - O(\log(n))$$

Proof: Use Symmetry of Information of Kolmogorov Complexity

Proof Step 2 (Core): Contradicting High $\mathsf{K}(x)$



If $T_i - T_{i-1}$ is always $< o(n)$:

$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] < o(n)$$

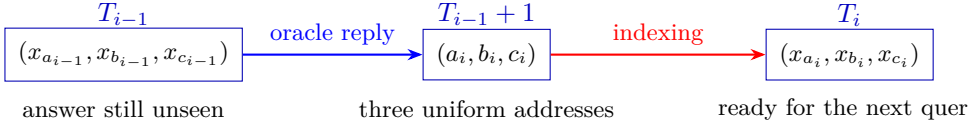
Fact: a random string does not save description length

$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] \geq \mathsf{K}(x) - O(\log(n))$$

Proof: Use Symmetry of Information of Kolmogorov Complexity

(If $T_i - T_{i-1}$ is $< o(n)$ with high probability, we can still compress *most of* x_a-s, and get a compression of x.)

Proof Step 2 (Core): Contradicting High $\mathsf{K}(x)$



If $T_i - T_{i-1}$ is always $< o(n)$:

$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] < o(n)$$

Fact: a random string does not save description length

$$\mathbb{E}_{\mathcal{O}}[\mathsf{K}(x \mid \mathcal{O})] \geq \mathsf{K}(x) - O(\log(n))$$

Proof: Use Symmetry of Information of Kolmogorov Complexity

(If $T_i - T_{i-1}$ is $< o(n)$ with high probability, we can still compress *most of* x_a-s, and get a compression of x.)

$\implies$ $\mathsf{K}(x)$ is small, contradiction!

Summary: Short Oracle Queries Force Nearly Quadratic Multitape Time

Our result: Near quadratic separation of RAM and Multitape-TM with *short* random oracle (query on $O(\log n)$ bits)

Proof Idea: For a random query answer a, unavoidably need to move $\Omega(n)$ steps to retrieve x_a. (Because x is incompressable.)

Summary: Short Oracle Queries Force Nearly Quadratic Multitape Time

Our result: Near quadratic separation of RAM and Multitape-TM with *short* random oracle (query on $O(\log n)$ bits)

Proof Idea: For a random query answer a, unavoidably need to move $\Omega(n)$ steps to retrieve x_a. (Because x is incompressable.)

Open problem: Can we replace $\mathcal{O}$ with a cryptographic assumption?

Thank you!