Lower bounds for Streaming Problems

Raphaël Clifford

Joint work with
Markus Jalsenius and Benjamin Sach
The CPU does not remember anything in between operations.
The CPU does not remember anything in between operations.

The CPU has unlimited computational power.
Data Structure Lower Bounds

Yao - FOCS '78

Predecessor (static)
- Ajtai - Combinatorica '88 (incorrect) (Communication complexity)
- Miltersen - STOC’ 94
- Miltersen, Nisan, Safra, Wigdersen - STOC ’95
- Beame, Fich - STOC ’99
- Sen - ICALP ’01

Dynamic problems (partial sums, union find)
- Fredman, Saks - STOC ’89 (Chronogram technique)
- Ben-Amram, Galil - FOCS ’91
- Miltersen, Subramanian, Vitter, Tamassia - TCS ’94
- Husfeldt, Rauhe, Skyum - SWAT ’96 (planar connectivity)
- Fredman, Henzinger - Algorithmica ’98 (non-determinism)
- Alstrup, Husfeldt, Rauhe - FOCS ’98 (marked ancestor)
- Alstrup, Husfeldt, Rauhe - SODA ’01 (2D NN)
- Alstrup, Ben-Amram, Rauhe - STOC ’99 (union-find)
Data Structure Lower Bounds

Yao - FOCS ’78
Predecessor (static)
- Ajtai - Combinatorica ’88 (incorrect) (Communication complexity)
- Miltersen - STOC’ 94
- Miltersen, Nisan, Safra, Wigdersen - STOC ’95
- Beame, Fich - STOC ’99
- Sen - ICALP ’01

Dynamic problems (partial sums, union find)
- Fredman, Saks - STOC ’89 (Chronogram technique)
- Ben-Amram, Galil - FOCS ’91
- Miltersen, Subramaniam - TCS ’94
- Husfeldt, Rauhe, Skyum - SWAT ’96 (planar connectivity)
- Fredman, Henzinger - Algorithmica ’98 (non-determinism)
- Alstrup, Husfeldt, Rauhe - FOCS ’98 (marked ancestor)
- Alstrup, Husfeldt, Rauhe - SODA ’01 (2D NN)
- Alstrup, Ben-Amram, Rauhe - STOC ’99 (union-find)

Best lower bound
\[ \Omega \left( \frac{\log n}{\log \log n} \right) \]
Data Structure Lower Bounds

Yao - FOCS ’78

Predecessor (static)
- Ajtai - Combinatorica ’88 (incorrect) (Communication complexity)
- Miltersen - STOC’ 94
- Miltersen, Nisan, Safra, Wigderson - STOC ’95
- Beame, Fich - STOC ’99
- Sen - ICALP ’01

Dynamic problems (partial sums, union find)
- Fredman, Saks - STOC ’89 (Chronogram technique)
- Ben-Amram, Galil - FOCS ’91
- Miltersen, Subramanian, Vitter, Tamassia - TCS ’94
- Husfeldt, Rauhe, Skyum - SWAT ’96 (planar connectivity)
- Fredman, Henzinger - Algorithmica ’98 (non-determinism)
- Alstrup, Husfeldt, Rauhe - FOCS ’98 (marked ancestor)
- Alstrup, Husfeldt, Rauhe - SODA ’01 (2D NN)
- Alstrup, Ben-Amram, Rauhe - STOC ’99 (union-find)

First \( \Omega (\log n) \) lower bound using information transfer.

M. Pătraşcu and E. Demaine

Tight bounds for the partial-sums problem

SODA 2004
Convolution

Stream of numbers from $[q]$

Fixed vector $V \in [q]^n$

Output dot product (modulo $q$):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x(i + \text{leftmost aligned index})$$
Convolution

Stream of numbers from $[q]$:

Fixed vector $V \in [q]^n$

Output dot product (modulo $q$):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x(i + \text{leftmost aligned index})$$
Convolution

Stream of numbers from $[q]$

Fixed vector $V \in [q]^n$

Output dot product (modulo $q$):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x(i + \text{leftmost aligned index})$$
### Convolution

Stream of numbers from \([q]\)

<table>
<thead>
<tr>
<th></th>
<th>(x_0)</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
<th>(x_6)</th>
<th>(x_7)</th>
<th>(x_8)</th>
<th>(x_9)</th>
<th>(x_{10})</th>
</tr>
</thead>
</table>

Fixed vector \(V \in [q]^n\)

Output dot product (modulo \(q\)):

\[
V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x(i + \text{leftmost aligned index})
\]
Convolution

Stream of numbers from \([q]\)

<table>
<thead>
<tr>
<th>(x_0)</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
<th>(x_6)</th>
<th>(x_7)</th>
<th>(x_8)</th>
<th>(x_9)</th>
<th>(x_{10})</th>
<th>(x_{11})</th>
</tr>
</thead>
</table>

Fixed vector \(V \in [q]^n\)

Output dot product (modulo \(q\)):

\[
V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x(i + \text{leftmost aligned index})
\]
Convolution

Stream of numbers from \([q]\)

Fixed vector \(V \in [q]^n\)

Output dot product (modulo \(q\)):

\[
V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x (i + \text{leftmost aligned index})
\]
Convolution

Stream of numbers from $[q]$

Fixed vector $V \in [q]^n$

Output dot product (modulo $q$):

$$V \cdot \text{(last } n \text{ digits of stream)} = \sum_{i=0}^{n-1} v_i x(i + \text{leftmost aligned index})$$
Convolution

Stream of numbers from \([q]\)

\[
\begin{array}{cccccccccccc}
& x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} \\
\end{array}
\]

Fixed vector \(V \in [q]^n\)

Output dot product (modulo \(q\)):

\[
V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{i + \text{leftmost aligned index}}
\]

Lower bound: \(\Omega\left(\frac{\delta}{w}\log n\right)\)

\(\delta = \log q\), word size \(w\).

C., Jalsenius. Lower Bounds for Online Integer Multiplication and Convolution in the Cell-Probe Mode. ICALP 2011
Previous bounds

M. J. Fischer and L. J. Stockmeyer
Fast on-line integer multiplication
STOC ’73

C., K. Efremenko, B. Porat and E. Porat
A black box for online approximate pattern matching
• $O(\log^2 n)$ time per arriving symbol (pair)
Previous bounds

M. J. Fischer and L. J. Stockmeyer
Fast on-line integer multiplication
STOC ’73

C., K. Efremenko, B. Porat and E. Porat
A black box for online approximate pattern matching

- $O(\log^2 n)$ time per arriving symbol (pair)

Offline cell probe complexity is linear!
⇒
**online** upper bound of $O(\log n)$
Previous bounds

M. J. Fischer and L. J. Stockmeyer
Fast on-line integer multiplication
STOC ’73

C., K. Efremenko, B. Porat and E. Porat
A black box for online approximate pattern matching

- $O(\log^2 n)$ time per arriving symbol (pair)

Better online lower bound
$\Rightarrow$

**super linear** lower bound for
**offline** convolution and multiplication
Yao’s minimax principle

A lower bound on the expected running time for

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Random</td>
</tr>
</tbody>
</table>

implies that the same lower bound holds for

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomised</td>
<td>Worst case</td>
</tr>
</tbody>
</table>
Yao’s minimax principle

A lower bound on the expected running time for

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Random</td>
</tr>
</tbody>
</table>

implies that the same lower bound holds for

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Worst case</td>
</tr>
</tbody>
</table>
Information transfer

- Fixed value
- Unknown value chosen uniformly at random from $[q]$
Information transfer

Fixed value

Unknown value chosen uniformly at random from $[q]$
Information transfer

Fixed value

Unknown value chosen uniformly at random from $[q]$
Information transfer

- Fixed value
- Unknown value chosen uniformly at random from \([q]\)

Memory cells
Information transfer

Fixed value

Unknown value
chosen uniformly at random from $[q]$
Information transfer

Unknown value chosen uniformly at random from $[q]$. 

Fixed value

Memory cells
Information transfer

- Fixed value
- Unknown value chosen uniformly at random from $[q]$
- Cell written during the $?\$-inputs
Information transfer

- Unknown value chosen uniformly at random from $[q]$.
- Fixed value.
- Cell written during the $t$-inputs.
Information transfer

- Fixed value
- Unknown value chosen uniformly at random from $[q]$
- Cell written during the $t$-inputs
Information transfer

- Fixed value
- Unknown value chosen uniformly at random from $[q]$ 
- Memory cells: Cell written during the $t$-inputs
Information transfer

Cells read during the next $\ell$ inputs

Unknown value chosen uniformly at random from $[q]$?

Fixed value

Cell written during the $t$-inputs

Cells read during the next $\ell$ inputs
Information transfer

Cells read during the next $\ell$ inputs

Unknown value chosen uniformly at random from $[q]$

Fixed value

Cell written during the $t$-inputs

Memory cells

Cells read during the next $\ell$ inputs
Information transfer

Fixed value

Unknown value chosen uniformly at random from $[q]$

Cell written during the $\ell$-inputs

Cells read during the next $\ell$ inputs
Information transfer

Cell written during the \( \ell \)-inputs

Cells read during the next \( \ell \) inputs

Fixed value

Unknown value chosen uniformly at random from \([q] \)

Memory cells

Diagram showing the transfer of information with fixed and unknown values.
Information transfer

Cell written during the $\ell$-inputs

Cells read during the next $\ell$ inputs

Fixed value

Unknown value chosen uniformly at random from $[q]$
Information transfer

The cells in $IT(t, \ell)$ provide sufficient information in order to give correct output during inputs.

The memory cells contain:
- Fixed value
- Unknown value chosen uniformly at random from $[q]$

Not including cells that were overwritten before being read.
Information transfer

The conditional entropy

\[ H(\text{the outputs during } \cdot \text{ all fixed}) \leq w + 2w \cdot \mathbb{E}[|IT(t, \ell)| \mid \text{ all fixed}] \]

\( w \) bits per cell
Information transfer

The conditional entropy

\[ H(\text{the outputs during } | IT(t, \ell)| \mid \text{all fixed}) \leq w + 2w \cdot \mathbb{E} [\| IT(t, \ell) \| \mid \text{all fixed}] \]

\( w \) bits per cell

Fixed value

Unknown value
chosen uniformly
at random from \([q]\)
Information transfer

Fixed value

Unknown value chosen uniformly at random from $[q]$

The conditional entropy

$$H(\text{the outputs during } |IT(t, \ell)| | \text{all fixed}) \leq w + 2w \cdot \mathbb{E}[|IT(t, \ell)| | \text{all fixed}]$$

$w$ bits per cell
How much information about $\ell$ do we need in order to give correct outputs during $\ell$?

Information transfer
How much information about \( ? ? ? ? ? \) do we need in order to give correct outputs during \( \ell \)?

 Depends on the fixed vector
Output is always 0 (no information)
Information transfer

Contributes to the dot product with the same value at each alignment

\( \delta = \log q \) bits of information
if the position is a power of 2
if the position is a power of 2
if the position is a power of 2
if the position is a power of 2
if the position is a power of 2
Information transfer

if the position is a power of 2
if the position is a power of 2

\( R \) = a recovered value
(recall that \( ? \) is chosen uniformly at random from \( [q] \), hence contributes with \( \delta = \log q \) bits to the entropy)
if the position is a power of 2

\( R \) = a recovered value

(recall that ? is chosen uniformly at random from \([q]\), hence contributes with \( \delta = \log q \) bits to the entropy)
if the position is a power of 2

\[ R \] = a recovered value

(recall that \[ ? \] is chosen uniformly at random from \[ [q] \], hence contributes with \( \delta = \log q \) bits to the entropy)
if the position is a power of 2

\( R \) = a recovered value

(recall that \( ? \) is chosen uniformly at random from \([q]\), hence contributes with \( \delta = \log q \) bits to the entropy)
if the position is a power of 2

$R$ = a recovered value

(recall that $?$ is chosen uniformly at random from $[q]$, hence contributes with $\delta = \log q$ bits to the entropy)

Conclusion: If $\ell$ is a power of 2 then we recover $\frac{\ell}{2}$ values
The conditional entropy

\[ H(\text{the outputs during } | \text{ all fixed}) \geq \frac{\ell}{2} \delta \]

Conclusion: If \( \ell \) is a power of 2 then we recover \( \frac{\ell}{2} \) values
The conditional entropy

\[ H(\text{the outputs during } t \mid \text{all fixed}) \geq \frac{\ell}{2} \delta \]

The conditional information transfer

\[ \mathbb{E}[|IT(t, \ell)| \mid \text{all fixed}] \geq \frac{\delta}{4w} \ell - \frac{1}{2} \]

\( w \) bits per cell
Suppose that all values (□ and ?) from the stream are chosen uniformly at random from $[q]$.

By linearity of expectation...

The conditional information transfer

$$\mathbb{E}[|IT(t, \ell)| \mid \text{all } \square \text{ fixed}] \geq \frac{\delta}{4w} \ell - \frac{1}{2}$$

$w$ bits per cell
Suppose that all values (■ and ?) from the stream are chosen uniformly at random from [q].

By linearity of expectation...

The conditional information transfer

\[ \mathbb{E} \left[|IT(t, \ell)| \right] \mid \text{all } \square \text{ fixed} \geq \frac{\delta}{4w} \ell - \frac{1}{2} \]

\( w \) bits per cell
Feed the algorithm with \( n \) values chosen uniformly at random from \([q]\).
Feed the algorithm with $n$ values chosen uniformly at random from $[q]$. 
Feed the algorithm with $n$ values chosen uniformly at random from $[q]$. 
Feed the algorithm with $n$ values chosen uniformly at random from $[q]$. 

$IT(t = 1, \ell = 2)$
Total number of cell reads

Feed the algorithm with $n$ values chosen uniformly at random from $[q]$. 

$IT(t = 5, \ell = 1)$
Feed the algorithm with \( n \) values chosen uniformly at random from \([q]\).
Total number of cell reads

0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0

Feed the algorithm with $n$ values chosen uniformly at random from $[q]$.

$IT(t = 5, \ell = 2)$
Feed the algorithm with \( n \) values chosen uniformly at random from \([q]\).
Total number of cell reads

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

Feed the algorithm with \( n \) values chosen uniformly at random from \([q]\).
The number of cell reads during the $n$ inputs is at least

$$\sum_{\text{internal node } v} |IT(t_v, \ell_v)|$$

Feed the algorithm with $n$ values chosen uniformly at random from $[q]$.
The number of cell reads during the $n$ inputs is at least

$$\sum_{\text{internal node } v} |IT(t_v, \ell_v)|$$

random from $[q]$.

No double counting of a cell read!
The number of cell reads during the $n$ inputs is at least

$$\sum_{\text{internal node } v} |IT(t_v, \ell_v)|$$

The expected number of cell reads is at least

$$\mathbb{E} \left[ \sum_{\text{internal node } v} |IT(t_v, \ell_v)| \right] = \sum_{\text{internal node } v} \mathbb{E} [ |IT(t_v, \ell_v)| ] \geq \sum_{\text{internal node } v} \frac{\delta}{4w} \ell_v - \frac{1}{2} = \Omega \left( \frac{\delta}{w} \cdot n \log n \right)$$
Total number of cell reads

The number of cell reads during the \( n \) inputs is at least

\[
\sum_{\text{internal node } v} |IT(t_v, \ell_v)|
\]

The expected number of cell reads is at least

\[
\mathbb{E} \left[ \sum_{\text{internal node } v} |IT(t_v, \ell_v)| \right] = \sum_{\text{internal node } v} \mathbb{E} \left[ |IT(t_v, \ell_v)| \right]
\]

So...

The amortised time lower bound per output is

\[
\Omega \left( \frac{\delta}{w} \log n \right)
\]
Multiplication in a stream

Paterson, Fischer and Meyer
An Improved Overlap Argument for On-Line Multiplication
SIAM-AMS Proceedings, 1974
For binary numbers on

- Multitape Turing machine: $\Omega(n \log n)$
- BAM or ”bounded activity machine”:

$$\Omega\left(\frac{n \log n}{\log \log n}\right)$$

C., Jalsenius
Lower Bounds for Online Integer Multiplication and Convolution in the Cell-Probe Mode. ICALP 2011

Time lower bound: $\Omega\left(\frac{\delta}{w} \cdot n \log n\right)$
Hamming distance

Stream of symbols from alphabet $\Sigma$

Fixed string $S$  

Output Hamming distance between $S$ and last $n$ symbols of stream.
Hamming distance

Stream of symbols from alphabet $\Sigma$

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ | ? |

Fixed string $S$ →

$S_0$ $S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$

$n$

Output Hamming distance between $S$ and last $n$ symbols of stream.

Lower bound: $\Omega\left(\frac{\delta}{w} \log n\right)$

$\delta = \log |\Sigma|$

C., Jalsenius, Sach. Tight Cell-Probe Bounds for Online Hamming Distance Computation. SODA 2013
The hard instance - a first attempt

Try a similar approach to before:

\[ \ell = 8 \]

\[
\begin{array}{cccccccc}
\text{?} & \text{?} & \text{?} & \text{?} & \text{R} & \text{R} & \text{R} & \text{R} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{array}
\]

0 = a symbol occurring only in the fixed string

1 if the position is a power of 2

We can only infer whether \( \text{R} \) is the symbol 1 or not, i.e. only one bit of information.
Hamming distance

More difficult than convolution:

- Appear to get at most 1 bit of information per symbol.
- Too large alphabet implies large Hamming distance (on random input), i.e. low entropy.
- Too small an alphabet implies low entropy per symbol.
- No obvious worst case pattern.
A harder instance

Substring $P$ at every power of two position, and 0 elsewhere (a symbol that does not occur in the stream).
A harder instance

Substring $P$ at every power of two position, and 0 elsewhere (a symbol that does not occur in the stream).

**Lemma**
There is a $P$ s.t. sliding it over all $2|P|$ length strings $T$ (over alphabet $\Sigma \setminus \{0\}$) generates $|\Sigma|^{\Theta(|\Sigma|)}$ distinct Hamming array outputs.
A harder instance

Substring $P$ at every power of two position, and 0 elsewhere (a symbol that does not occur in the stream).

**Lemma**
There is a $P$ s.t. sliding it over all $2|P|$ length strings $T$ (over alphabet $\Sigma \setminus \{0\}$) generates $|\Sigma|^{\Theta(|\Sigma|)}$ distinct Hamming array outputs.

Great news! Highest entropy we can hope for.
The hard instance

Each $T_j$ is drawn uniformly from a set $\mathcal{T}$ of size $|\Sigma|^\Theta(|\Sigma|)$. Any two strings in $\mathcal{T}$ give distinct Hamming outputs with $P$. 
The hard instance

Each $T_j$ is drawn uniformly from a set $\mathcal{T}$ of size $|\Sigma|^\Theta(|\Sigma|)$. Any two strings in $\mathcal{T}$ give distinct Hamming outputs with $P$.

Recover $\Theta(\ell)$ symbols from a window of $\ell$ unknown input symbols. Entropy:

$$\Theta\left(\frac{\ell}{2|\Sigma|} \cdot \log |\Sigma|^\Theta(|\Sigma|)\right) = \Theta(\ell \cdot \log |\Sigma|) = \Theta(\ell \delta)$$

$\delta = \log |\Sigma|$
The hard instance

Each $T_j$ is drawn uniformly from a set $\mathcal{T}$ of size $|\Sigma|^\Theta(|\Sigma|)$.

Hence lower bound $\Omega\left(\frac{\delta}{w} \log n\right)$

Recover $\Theta(\ell)$ symbols from a window of $\ell$ unknown input symbols. Entropy:

$$\Theta\left(\frac{\ell}{2|\Sigma|} \cdot \log |\Sigma|^\Theta(|\Sigma|)\right) = \Theta(\ell \cdot \log |\Sigma|) = \Theta(\ell \delta)$$

$\delta = \log |\Sigma|$
The string $P$

Proof overview of the lemma.

- Partition $P$ into blocks, each using a unique symbol.

\[ \mu = \left| \Sigma \right|^{1/3} \]  

◊ is a symbol that only occurs in $T$
The string $P$

**Proof overview of the lemma.**
- Partition $P$ into blocks, each using a unique symbol.
- Symbols of $T$ will slide over $P$, and match sums will correspond to sums of binary vectors.

$\mu = \sqrt[3]{\sum}$

⋄ is a symbol that only occurs in $T$
The string $P$

- For each window of $\mu$ outputs, one can obtain $\mu^\Theta(\mu)$ distinct vector sums. (Proof involves cyclic binary codes.)

$\diamond$ is a symbol that only occurs in $T$.
The string $P$

- For each window of $\mu$ outputs, one can obtain $\mu^{\Theta(\mu)}$ distinct vector sums. (Proof involves cyclic binary codes.)
- Thus, over the whole of $T$ there are $|\Sigma|^{\Theta(|\Sigma|)}$ possible distinct Hamming array outputs.

$\diamond$ is a symbol that only occurs in $T$. 

$\mu = |\Sigma|^{1/3}$
What next?

Entirely new techniques appear to be needed again for seemingly related problems. For example:

- Edit distance (outputs can be encoded in $O(n)$ bits)
- Decision problems (entropy is very low)
What next?

Entirely new techniques appear to be needed again for seemingly related problems. For example:

- Edit distance (outputs can be encoded in $O(n)$ bits)
- Decision problems (entropy is very low)

Thank you!