Data Streams, Dyck Languages, and Detecting Dubious Data Structures

Amit Chakrabarti
Dartmouth College

Graham Cormode
AT&T Research Labs

Ranganath Kondapally
Dartmouth College

Andrew McGregor
University of Massachusetts, Amherst
• At each step, user inserts a value into the memory or asks that the \textit{smallest} value is extracted:
• At each step, user inserts a value into the memory or asks that the smallest value is extracted:

\[ \text{ins}(5) \]
At each step, user **inserts** a value into the memory or asks that the *smallest* value is **extracted**:

\[
\text{ins}(5) \quad \text{ins}(3)
\]
At each step, user inserts a value into the memory or asks that the smallest value is extracted:

\[ \text{ins}(5) \text{ ins}(3) \]
At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

\texttt{ins(5) ins(3) ext(3)}
• At each step, user inserts a value into the memory or asks that the smallest value is extracted:

ins(5) ins(3) ext(3) ins(6)
• At each step, user inserts a value into the memory or asks that the smallest value is extracted:

\[
\text{ins}(5) \ \text{ins}(3) \ \text{ext}(3) \ \text{ins}(6) \ \text{ins}(7)
\]
• At each step, user inserts a value into the memory or asks that the smallest value is extracted:

ins(5)  ins(3)  ext(3)  ins(6)  ins(7)
At each step, user inserts a value into the memory or asks that the smallest value is extracted:

\[ \text{ins}(5) \text{ ins}(3) \text{ ext}(3) \text{ ins}(6) \text{ ins}(7) \text{ ext}(5) \]
At each step, user inserts a value into the memory or asks that the smallest value is extracted:

ins(5) ins(3) ext(3) ins(6) ins(7) ext(5)
At each step, user inserts a value into the memory or asks that the smallest value is extracted:

ins(5) ins(3) ext(3) ins(6) ins(7) ext(5) ext(6)
At each step, user inserts a value into the memory or asks that the smallest value is extracted:

\[
\text{ins}(5) \text{ ins}(3) \text{ ext}(3) \text{ ins}(6) \text{ ins}(7) \text{ ext}(5) \text{ ext}(6)
\]
• At each step, user inserts a value into the memory or asks that the smallest value is extracted:

\[
\text{ins}(5) \quad \text{ins}(3) \quad \text{ext}(3) \quad \text{ins}(6) \quad \text{ins}(7) \quad \text{ext}(5) \quad \text{ext}(6) \quad \text{ext}(7)
\]
• At each step, user inserts a value into the memory or asks that the smallest value is extracted:

\[
\text{ins}(5) \hspace{1em} \text{ins}(3) \hspace{1em} \text{ext}(3) \hspace{1em} \text{ins}(6) \hspace{1em} \text{ins}(7) \hspace{1em} \text{ext}(5) \hspace{1em} \text{ext}(6) \hspace{1em} \text{ext}(7)
\]

? **Challenge:** Without remembering all the interaction, can you verify the priority queue performed correctly?
- At each step, user **inserts** a value into the memory or asks that the *smallest* value is **extracted**:

  \[
  \text{ins}(5) \quad \text{ins}(3) \quad \text{ext}(3) \quad \text{ins}(6) \quad \text{ins}(7) \quad \text{ext}(5) \quad \text{ext}(6) \quad \text{ext}(7)
  \]

  **Challenge:** Without remembering all the interaction, can you verify the priority queue performed correctly?

- **Motivation:** Want to use cheap commodity hardware.

  [Blum, Evans, Gemmell, Kannan, Naor ’94]
PQ Language Problem
PQ Language Problem

- Let PQ be set of legitimate transcripts of a priority queue that starts and ends empty.
PQ Language Problem

- Let PQ be set of legitimate transcripts of a priority queue that starts and ends empty.

\[
\text{ins}(5), \text{ins}(3), \text{ext}(3), \text{ins}(7), \text{ext}(5), \text{ext}(7) \in PQ
\]
PQ Language Problem

- Let PQ be set of legitimate transcripts of a priority queue that starts and ends empty.

  \[
  \text{ins}(5), \text{ins}(3), \text{ext}(3), \text{ins}(7), \text{ext}(5), \text{ext}(7) \in \text{PQ} \\
  \text{ins}(5), \text{ext}(3), \text{ins}(3), \text{ins}(7), \text{ext}(7), \text{ext}(5) \notin \text{PQ}
  \]
PQ Language Problem

- Let PQ be set of legitimate transcripts of a priority queue that starts and ends empty.
  
  \[
  \text{ins}(5), \text{ins}(3), \text{ext}(3), \text{ins}(7), \text{ext}(5), \text{ext}(7) \in \text{PQ}
  \]
  
  \[
  \text{ins}(5), \text{ext}(3), \text{ins}(3), \text{ins}(7), \text{ext}(7), \text{ext}(5) \notin \text{PQ}
  \]

- **PQ Problem**: Given streaming access to length N transcript, determine if it’s in PQ using o(N) space.
PQ Language Problem

- Let PQ be set of legitimate transcripts of a priority queue that starts and ends empty.
  
  ins(5), ins(3), ext(3), ins(7), ext(5), ext(7) ∈ PQ

  ins(5), ext(3), ins(3), ins(7), ext(7), ext(5) ∉ PQ

- **PQ Problem:** Given *streaming* access to length N transcript, determine if it’s in PQ using o(N) space.

- *In this talk...*
  
  i. We’ll design an algorithm that uses O(√N) space!
PQ Language Problem

- Let PQ be set of legitimate transcripts of a priority queue that starts and ends empty.
  
  \[
  \text{ins}(5), \text{ins}(3), \text{ext}(3), \text{ins}(7), \text{ext}(5), \text{ext}(7) \in PQ
  \]
  
  \[
  \text{ins}(5), \text{ext}(3), \text{ins}(3), \text{ins}(7), \text{ext}(7), \text{ext}(5) \notin PQ
  \]

- **PQ Problem:** Given *streaming* access to length N transcript, determine if it’s in PQ using \(o(N)\) space.

- *In this talk...*
  
  i. We’ll design an algorithm that uses \(O(\sqrt{N})\) space!
  
  ii. Prove it’s optimal via a communication lower bound.
PQ Language Problem

• Let PQ be set of legitimate transcripts of a priority queue that starts and ends empty.
  
  \[
  \text{ins}(5), \text{ins}(3), \text{ext}(3), \text{ins}(7), \text{ext}(5), \text{ext}(7) \in PQ
  \]

  \[
  \text{ins}(5), \text{ext}(3), \text{ins}(3), \text{ins}(7), \text{ext}(7), \text{ext}(5) \notin PQ
  \]

• **PQ Problem:** Given streaming access to length N transcript, determine if it’s in PQ using \( o(N) \) space.

• **In this talk...**
  
  i. We’ll design an algorithm that uses \( O(\sqrt{N}) \) space!

  ii. Prove it’s optimal via a communication lower bound.

  iii. Explore connections with other problems...
I. Memory Checking
II. Lower Bounds
III. Parenthesis and Passes
I. Memory Checking
PQ Algorithm
PQ Algorithm

- **Thm:** There exists a $O(\sqrt{N} \log N)$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.
PQ Algorithm

• Theorem: There exists a $O(\sqrt{N \log N})$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

“Can verify terabytes of transcript with only megabytes!”
PQ Algorithm

- **Thm:** There exists a $O(\sqrt{N \log N})$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

  "Can verify terabytes of transcript with only megabytes!"

- **Prelim:** Easy to check that set of values inserted equals set of values extracted using fingerprinting.
PQ Algorithm

• **Thm:** There exists a $O(\sqrt{N \log N})$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

  “Can verify terabytes of transcript with only megabytes!”

• **Prelim:** Easy to check that set of values inserted equals set of values extracted using fingerprinting.

$$\prod_{u \text{ inserts}} (x - u) \overset{?}{=} \prod_{u \text{ extracts}} (x - u)$$
PQ Algorithm

• Thm: There exists a $O(\sqrt{N \log N})$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

  “Can verify terabytes of transcript with only megabytes!”

• Prelim: Easy to check that set of values inserted equals set of values extracted using fingerprinting.

$$\prod_{u \text{ inserts}} (r - u) \overset{?}{=} \prod_{u \text{ extracts}} (r - u)$$
PQ Algorithm

• **Thm:** There exists a $O(\sqrt{N \log N})$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

  “Can verify terabytes of transcript with only megabytes!”

• **Prelim:** Easy to check that set of values inserted equals set of values extracted using fingerprinting.

  $$\prod_{u \text{ inserts}} (r - u) \overset{?}{=} \prod_{u \text{ extracts}} (r - u)$$

! **For this talk:** Assume inserted elements are distinct and that inserts come before their corresponding extract. I.e., we’re trying to identify the following bad pattern:

  $$\text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u)$$ for some $u < v$
Epochs and Local Bad Patterns...
Epochs and Local Bad Patterns...
• Split length $N$ sequence into $\sqrt{N}$ epochs of length $\sqrt{N}$
• Split length N sequence into $\sqrt{N}$ epochs of length $\sqrt{N}$
• Split length $N$ sequence into $\sqrt{N}$ epochs of length $\sqrt{N}$

• **Defn:** Bad pattern $\text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u)$ is *local* if $\text{ins}(u)$ and $\text{ext}(v)$ occur in same epoch and *long-range* otherwise.
• Split length $N$ sequence into $\sqrt{N}$ epochs of length $\sqrt{N}$

• **Defn:** Bad pattern $\text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u)$ is *local* if $\text{ins}(u)$ and $\text{ext}(v)$ occur in the same epoch and *long-range* otherwise.

• Using $O(\sqrt{N})$ space, we can buffer each epoch and check for local bad patterns.
Catching Long-Range Bad Patterns... 1/2
Catching Long-Range Bad Patterns... 1/2
• Maintain the max value extracted between end of i-th epoch and current time. Call it f(i).
Maintain the max value extracted between end of i-th epoch and current time. Call it $f(i)$. 

Catching Long-Range Bad Patterns... 1/2
Catching Long-Range Bad Patterns... 1/2

- Maintain the max value extracted between end of i-th epoch and current time. Call it f(i).
- Maintain the max value extracted between end of i-th epoch and current time. Call it f(i).
• Maintain the max value extracted between end of i-th epoch and current time. Call it $f(i)$. 

Catching Long-Range Bad Patterns... 1/2
Maintain the max value extracted between end of i-th epoch and current time. Call it f(i).
• Maintain the max value extracted between end of i-th epoch and current time. Call it \( f(i) \).
• Maintain the max value extracted between end of i-th epoch and current time. Call it \( f(i) \).

• **Defn:** Each \( \text{ins}(u) \) or \( \text{ext}(u) \) is *adopted* by earliest epoch \( k \) with \( f(k) \leq u \).
• **Lemma:** If $\text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u)$ is a long-range bad pattern then $\text{ins}(u)$ and $\text{ext}(u)$ are adopted by different epochs.
Lemma: If $\text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u)$ is a long-range bad pattern then $\text{ins}(u)$ and $\text{ext}(u)$ are adopted by different epochs.

Proof:
  i. Let $\text{ins}(u)$ be adopted by k-th epoch.
Catching Long-Range Bad Patterns... 2/2

- **Lemma:** If \( \text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u) \) is a long-range bad pattern then \( \text{ins}(u) \) and \( \text{ext}(u) \) are adopted by different epochs.

- **Proof:**
  i. Let \( \text{ins}(u) \) be adopted by \( k \)-th epoch.
  ii. After \( v \) is extracted \( f(k) \geq v > u \).
**Lemma:** If $\text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u)$ is a long-range bad pattern then $\text{ins}(u)$ and $\text{ext}(u)$ are adopted by different epochs.

**Proof:**

i. Let $\text{ins}(u)$ be adopted by $k$-th epoch.

ii. After $v$ is extracted $f(k) \geq v > u$.

iii. $\text{ext}(u)$ can no longer be adopted by $k$-th epoch.
• **Lemma**: If \( \text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u) \) is a long-range bad pattern then \( \text{ins}(u) \) and \( \text{ext}(u) \) are adopted by different epochs.

• **Proof**:
  i. Let \( \text{ins}(u) \) be adopted by k-th epoch.
  ii. After \( v \) is extracted \( f(k) \geq v > u \).
  iii. \( \text{ext}(u) \) can no longer be adopted by k-th epoch.

• **Lemma**: If there are no bad patterns, every \( \text{ins}(u) \) and \( \text{ext}(u) \) pair get adopted by the same epoch.
Lemma: If \( \text{ins}(u) \ldots \text{ext}(v) \ldots \text{ext}(u) \) is a long-range bad pattern then \( \text{ins}(u) \) and \( \text{ext}(u) \) are adopted by different epochs.

Proof:
i. Let \( \text{ins}(u) \) be adopted by \( k \)-th epoch.
   ii. After \( v \) is extracted \( f(k) \geq v > u \).
   iii. \( \text{ext}(u) \) can no longer be adopted by \( k \)-th epoch.

Lemma: If there are no bad patterns, every \( \text{ins}(u) \) and \( \text{ext}(u) \) pair get adopted by the same epoch.

Algorithm: Using fingerprints to check: for each epoch \( k \)

\[
\{u : \text{ins}(u) \text{ adopted by } k\} = \{u : \text{ext}(u) \text{ adopted by } k\}.
\]
Conclusions
Conclusions

- **Thm:** There exists a $O(\sqrt{N \log N})$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

  "Can verify terabytes of transcript with only megabytes"
Conclusions

• **Thm:** There exists a $O(\sqrt{N \log N})$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

  “Can verify terabytes of transcript with only megabytes”

• **Extensions:** Sub-linear space streaming recognition of other data structures like stacks, double-ended queues...
I. Memory Checking

II. Lower Bounds

III. Parenthesis and Passes
II. Lower Bounds
Communication Complexity
Many space lower bounds in data stream model are based on reductions from communication complexity.
Communication Complexity

• Many space lower bounds in data stream model are based on reductions from communication complexity.

• **Augmented Index**: Alice has $x \in \{0,1\}^n$ and Bob has a prefix $y \in \{0,1\}^{k-1}$ of $x$ and $c \in \{0,1\}$. Bob wants to check if $c=x_k$. 

\[ \begin{align*} 
x & \quad \text{Alice has} \quad y, c \quad \text{Bob has} \\
\end{align*} \]
• Many space lower bounds in data stream model are based on reductions from communication complexity.

• **Augmented Index:** Alice has $x \in \{0,1\}^n$ and Bob has a prefix $y \in \{0,1\}^{k-1}$ of $x$ and $c \in \{0,1\}$. Bob wants to check if $c = x_k$.

• **Thm:** Any 1/3-error, one-way protocol from Alice to Bob for $A_{1n}$ requires $\Omega(n)$ bits sent. [Miltersen et al. JCSS ’98]
Multi-player Augmented Index
Multi-player Augmented Index

We now have 2m players $A_1, \ldots, A_m, B_1, \ldots, B_m$ where each $A_i$ and $B_i$ have an instance $(x^i, y^i, c^i)$ of $A\|n$.
We now have $2m$ players $A_1, \ldots, A_m, B_1, \ldots, B_m$ where each $A_i$ and $B_i$ have an instance $(x_i, y_i, c_i)$ of $\text{AI}_n$.

Want to determine if any of the $\text{AI}_n$ instances are false using private messages communicated in the order

$$A_1 \rightarrow B_1 \rightarrow A_2 \rightarrow B_2 \rightarrow \ldots \rightarrow A_m \rightarrow B_m \rightarrow A_m \rightarrow A_{m-1} \rightarrow \ldots \rightarrow A_1$$
We now have 2m players $A_1, \ldots, A_m, B_1, \ldots, B_m$ where each $A_i$ and $B_i$ have an instance $(x_i, y_i, c_i)$ of $A_{1n}$.

Want to determine if any of the $A_{1n}$ instances are false using private messages communicated in the order:

$$A_1 \rightarrow B_1 \rightarrow A_2 \rightarrow B_2 \rightarrow \ldots \rightarrow A_m \rightarrow B_m \rightarrow A_m \rightarrow A_{m-1} \rightarrow \ldots \rightarrow A_1$$

**Thm:** Any $1/3$-error protocol has a $\Omega(\min m, n)$ bit message.
Multi-player Augmented Index

- We now have 2m players $A_1, \ldots, A_m, B_1, \ldots, B_m$ where each $A_i$ and $B_i$ have an instance $(x_i, y_i, c_i)$ of $\text{AI}_n$

- Want to determine if any of the $\text{AI}_n$ instances are false using private messages communicated in the order

  $A_1 \rightarrow B_1 \rightarrow A_2 \rightarrow B_2 \rightarrow \ldots \rightarrow A_m \rightarrow B_m \rightarrow A_m \rightarrow A_{m-1} \rightarrow \ldots \rightarrow A_1$

- **Thm:** Any 1/3-error protocol has a $\Omega(\min m,n)$ bit message.

- **Corollary:** Any algorithm for PQ requires $\Omega(\sqrt{N})$ space.
Reduction from multi-player AI to PQ...

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...

```
0010 0, 0 0100 01, 1 1010 , 1 1010 0100 0010
```

```
ins(12.0) ins(11.1) ins(10.0) ins(9.0)
```

"Ascension Problem" [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...

```
0010 0,0 0100 01,1 1010 ,1 1010 0100 0010

ins(12.0) ext(9.0)
ins(11.1) ext(10.0)
ins(10.0) ins(10.0)
ins(9.0) ins(9.0)
```

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...

"Ascension Problem" [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...
Reduction from multi-player AI to PQ...

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]
Reduction from multi-player AI to PQ...
• **Thm:** Any algorithm for recognizing PQ with probability at least 2/3 requires $\Omega(\sqrt{N})$ space.
• **Thm:** Any algorithm for recognizing PQ with probability at least 2/3 requires \( \Omega(\sqrt{N}) \) space.

• **Proof:**

  i. Let \( \mathcal{A} \) be a stream algorithm using \( s \) bits of space.
Reduction from multi-player AI to PQ...

• **Thm:** Any algorithm for recognizing PQ with probability at least 2/3 requires $\Omega(\sqrt{N})$ space.

• **Proof:**
  
  i. Let $\mathcal{A}$ be a stream algorithm using $s$ bits of space.
  
  ii. Use $\mathcal{A}$ to construct a protocol with $s$ bit messages: Players run $\mathcal{A}$ on their input and send memory state to next player.
Reduction from multi-player AI to PQ...

- **Thm:** Any algorithm for recognizing PQ with probability at least 2/3 requires $\Omega(\sqrt{N})$ space.

- **Proof:**
  
  i. Let $\mathcal{A}$ be a stream algorithm using $s$ bits of space.
  
  ii. Use $\mathcal{A}$ to construct a protocol with $s$ bit messages: Players run $\mathcal{A}$ on their input and send memory state to next player.
  
  iii. Therefore, $s = \Omega(\min m, n)$ for length $mn$ sequence.
I. Memory Checking

II. Lower Bounds

III. Parenthesis and Passes
III. Parenthesis and Passes
Fictional Quote:

“After Ammu died (after the last time she came back to Ayemenem (she had been swollen with cortisone and a rattle in her chest that sounded like a faraway man shouting), Rahel drifted.”
DYCK Language Problem
DYCK Language Problem

- $\text{DYCK}_2$ is the set of strings of properly nested brackets when there are two different types of brackets:
  
  $((([])([]))) \in \text{DYCK}_2$  
  
  $([[]]) \not\in \text{DYCK}_2$
DYCK Language Problem

• DYCK₂ is the set of strings of properly nested brackets when there are two different types of brackets:

  ((([])([]))) ∈ DYCK₂  \quad  [[[[]][]]) \notin DYCK₂

• **DYCK Problem**: Given streaming access to length N string, determine if it’s in DYCK₂ using o(N) space.
DYCK Language Problem

- $\text{DYCK}_2$ is the set of strings of properly nested brackets when there are two different types of brackets:
  
  $$(([])([])) \in \text{DYCK}_2 \quad \text{and} \quad ([[]][])) \notin \text{DYCK}_2$$

- **DYCK Problem**: Given *streaming* access to length $N$ string, determine if it’s in $\text{DYCK}_2$ using $o(N)$ space.

- **Previous result**: $O(\sqrt{N})$ space suffices. [Magniez, Mathieu, Nayak ’10]
DYCK Language Problem

- \( \text{DYCK}_2 \) is the set of strings of properly nested brackets when there are two different types of brackets:
  
  \[ ((([])([]))) \in \text{DYCK}_2 \quad \text{and} \quad ((([]))[]) \notin \text{DYCK}_2 \]

- **DYCK Problem**: Given *streaming* access to length \( N \) string, determine if it’s in \( \text{DYCK}_2 \) using \( o(N) \) space.

- **Previous result**: \( O(\sqrt{N}) \) space suffices. [Magniez, Mathieu, Nayak ’10]

- **But...** If you’re allowed a forward pass followed by a backwards pass, space can be reduced to \( O(\log N) \)!
DYCK Language Problem

- $\text{DYCK}_2$ is the set of strings of properly nested brackets when there are two different types of brackets:
  
  $((([])[[]])) \in \text{DYCK}_2 \\ ([[][[]]) \notin \text{DYCK}_2$

- **DYCK Problem:** Given streaming access to length $N$ string, determine if it’s in $\text{DYCK}_2$ using $o(N)$ space.

- **Previous result:** $O(\sqrt{N})$ space suffices. [Magniez, Mathieu, Nayak ’10]

- **But...** If you’re allowed a forward pass followed by a backwards pass, space can be reduced to $O(\log N)$!

  “How useful is reading backwards? Do we also get a space saving if you can take multiple forward passes?”
Space/Pass Trade-Offs
Space/Pass Trade-Offs

- If you increase the number of passes $p$, for some problems the space required can be dramatically reduced...
• If you increase the number of passes $p$, for some problems the space required can be dramatically reduced...

• *Example 1:* Necessary and sufficient space to find the median of $n$ values $\Theta(n^{1/p})$. 
Space/Pass Trade-Offs

• If you increase the number of passes $p$, for some problems the space required can be dramatically reduced...

• **Example 1:** Necessary and sufficient space to find the median of $n$ values $\Theta(n^{1/p})$.

• **Example 2:** Necessary and sufficient space to find an increasing subsequence of length $k$ is $\Theta(k^{1+\frac{1}{2p-1}})$.
$\text{DYCK}_2 \iff \text{PQ}$
There's a reduction from $\text{DYCK}_2$ to $\text{PQ}$ and our bounds extend to multi-pass algorithms.
There's a reduction from \( \text{DYCK}_2 \) to PQ and our bounds extend to multi-pass algorithms.

**Thm:** Any \( p \) pass algorithm for \( \text{DYCK}_2 \) that only uses forward passes requires \( \Omega(\sqrt{N/p}) \).
DYCK\(_2 \Leftrightarrow\) PQ

- There’s a reduction from DYCK\(_2\) to PQ and our bounds extend to multi-pass algorithms.

- **Thm:** Any \(p\) pass algorithm for DYCK\(_2\) that only uses forward passes requires \(\Omega(\sqrt{N/p})\).

  “Reading backwards can be very helpful!”
DYCK$_2$ $\iff$ PQ

- There’s a reduction from DYCK$_2$ to PQ and our bounds extend to multi-pass algorithms.

- **Thm:** Any $p$ pass algorithm for DYCK$_2$ that only uses forward passes requires $\Omega(\sqrt{N/p})$.

  “Reading backwards can be very helpful!”

- **Open Problem:** Stream complexity of recognizing other languages and examples of backwards phenomena?
Memory Checking: Sub-linear space recognition of various data-structure transcript languages!

Theory of Stream Computation: Forward and backward pass better than many forward passes!

Further Work: Annotations, stream language recognition, ...

Thanks!
I. Memory Checking

II. Lower Bounds

III. Parenthesis and Passes
IV. Augmented Index Bound
Information Complexity

[Chakrabarti, Shi, Wirth, Yao ’01]
Information Complexity

[Chakrabarti, Shi, Wirth, Yao ’01]

- **Entropy and Mutual Information:**

  \[
  H(X) = -\sum \Pr[X = x] \lg \Pr[X = x] \\
  H(X|Y) = -\sum \Pr[X = x, Y = y] \lg \Pr[X = x|Y = y]
  \]

  \[
  I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
  \]

  \[
  I(X;Y|Z) = H(X|Z) - H(X|Y,Z)
  \]
Information Complexity

[Chakrabarti, Shi, Wirth, Yao ’01]

- **Entropy and Mutual Information:**

\[
H(X) = -\sum \text{Pr}[X = x] \lg \text{Pr}[X = x]
\]

\[
H(X|Y) = -\sum \text{Pr}[X = x, Y = y] \lg \text{Pr}[X = x|Y = y]
\]

\[
I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
\]

\[
I(X; Y|Z) = H(X|Z) - H(X|Y, Z)
\]
Information Complexity

[Chakrabarti, Shi, Wirth, Yao '01]

- **Entropy and Mutual Information:**

\[
H(X) = -\sum \Pr[X = x] \log \Pr[X = x]
\]

\[
H(X|Y) = -\sum \Pr[X = x, Y = y] \log \Pr[X = x|Y = y]
\]

\[
I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
\]

\[
I(X; Y|Z) = H(X|Z) - H(X|Y, Z)
\]

- **Information cost method:** Consider mutual information between random input for a communication problem and the communication transcript:

\[
I(\text{transcript}; \text{input})
\]
Information Complexity

[Chakrabarti, Shi, Wirth, Yao ’01]

- **Entropy and Mutual Information:**

\[
H(X) = -\sum \text{Pr}[X = x] \log \text{Pr}[X = x]
\]

\[
H(X|Y) = -\sum \text{Pr}[X = x, Y = y] \log \text{Pr}[X = x|Y = y]
\]

\[
I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
\]

\[
I(X;Y|Z) = H(X|Z) - H(X|Y, Z)
\]

- **Information cost method:** Consider mutual information between random input for a communication problem and the communication transcript:

\[
I(\text{transcript}; \text{input}) \leq \text{length of transcript}
\]
Information Complexity

[Chakrabarti, Shi, Wirth, Yao ’01]

• **Entropy and Mutual Information:**

\[
H(X) = - \sum \Pr[X = x] \log \Pr[X = x]
\]

\[
H(X|Y) = - \sum \Pr[X = x, Y = y] \log \Pr[X = x|Y = y]
\]

\[
I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
\]

\[
I(X; Y|Z) = H(X|Z) - H(X|Y, Z)
\]

• **Information cost method:** Consider mutual information between random input for a communication problem and the communication transcript:

\[
I(\text{transcript}; \text{input}) \leq \text{length of transcript}
\]

• Can restrict to partial transcript and subsets of input: useful for proving direct-sum arguments.
Information Complexity of $\text{AI}_n$
**Information Complexity of $\text{Al}_n$**

- **Defn:** Let $P$ be a protocol for $\text{Al}_n$ using public random string $R$. Let $T$ be the transcript and $(X, K, C) \sim \xi$. Define

\[
\begin{align*}
\text{icost}^A_\xi(P) & = I(T : X | K, C, R) \\
\text{icost}^B_\xi(P) & = I(T : K, C | X, R)
\end{align*}
\]
Information Complexity of $\text{Al}_n$

- **Defn:** Let $P$ be a protocol for $\text{Al}_n$ using public random string $R$. Let $T$ be the transcript and $(X, K, C) \sim \xi$. Define

\[
\begin{align*}
\text{icost}_\xi^A(P) &= I(T : X \mid K, C, R) \\
\text{icost}_\xi^B(P) &= I(T : K, C \mid X, R)
\end{align*}
\]

- **Thm:** Let $P$ be a randomized protocol for $\text{Al}_n$ with error $1/3$ under the uniform distribution $\mu$. Then,

\[
\text{icost}_\mu_0^A(P) = \Omega(n) \quad \text{or} \quad \text{icost}_\mu_0^B(P) = \Omega(1)
\]

where $\mu_0$ is $\mu$ conditioned on $X_K=C$. 
MULTI-AI_{m,n} versus AI_n
MULTI-AI_{m,n} versus AI_{n}

- **Defn:** Let Q be a protocol for MULTI-AI_{m,n} using public random string R. Let T be transcript and (X^i, K^i, C^i)_{i\in[m]} \sim \xi.

\[
\text{icost}_\xi(Q) = I(T_m : K^1, C^1, \ldots, K^m, C^m \mid X^1, \ldots, X^m, R)
\]

where T_m is the set of messages sent by B_m.
MULTI-AI\textsubscript{m,n} versus AI\textsubscript{n}

• **Defn:** Let Q be a protocol for MULTI-AI\textsubscript{m,n} using public random string \( R \). Let \( T \) be transcript and \( (X^i,K^i,C^i)_{i\in[m]} \sim \xi \).

\[
\text{icost}_\xi(Q) = I(T_m : K^1, C^1, \ldots, K^m, C^m \mid X^1, \ldots, X^m, R)
\]

where \( T_m \) is the set of messages sent by \( B_m \).

• **Thm (Direct Sum):** If there exists a \( p \)-round, \( s \)-bit, \( \varepsilon \)-error protocol \( Q \) for MULTI-AI\textsubscript{m,n} then there exists a \( p \)-round, \( \varepsilon \)-error randomized protocol \( P \) for AI\textsubscript{n} where

i. Alice sends at most \( ps \) bits

ii. \( m \cdot \text{icost}_{\mu_0}^B(P) \leq \text{icost}_{\mu_0}^{\otimes m}(Q) \)
Putting it all together...
Putting it all together...

- **Thm:** Any $p$-round, $s$-bit, $1/3$-error protocol $Q$ for \textsc{Multi-Al}_{m,n} requires $ps=\Omega(\min m,n)$. 
Putting it all together...

- **Thm:** Any p-round, s-bit, 1/3-error protocol Q for MULTI-$\text{AI}_{m,n}$ requires $ps=\Omega(\min m,n)$.

- **Proof:**
  
  i. By direct sum theorem, there exists $\varepsilon$-error, p-pass protocol P for $\text{AI}_n$ such that:

  $$p \cdot s \geq \text{icost}_{\mu_0^m}(Q) \geq m \cdot \text{icost}_{\mu_0}^B(P)$$

  $$p \cdot s \geq \text{icost}_{\mu_0}^A(P)$$
Putting it all together...

- **Thm:** Any p-round, s-bit, $1/3$-error protocol $Q$ for MULTI-$\text{AI}_m,n$ requires $ps=\Omega(\min m,n)$.

- **Proof:**
  
  i. By direct sum theorem, there exists $\varepsilon$-error, $p$-pass protocol $P$ for $\text{AI}_n$ such that:
  
  \[ p \cdot s \geq \text{icost}_{\mu_0 \otimes m}(Q) \geq m \cdot \text{icost}_{\mu_0}^B(P) \]
  
  \[ p \cdot s \geq \text{icost}_{\mu_0}^A(P) \]
  
  ii. By information complexity of $\text{AI}_n$
  
  \[ \max(m \cdot \text{icost}_{\mu_0}^B(P), \text{icost}_{\mu_0}^A(P)) = \Omega(\min(m, n)) \]
Summary

**Memory Checking:** Sub-linear space recognition of various data-structure transcript languages!

**Theory of Stream Computation:**
Forward and backward pass better than many forward passes!

**Further Work:** Annotations, stream language recognition, ...

Thanks!