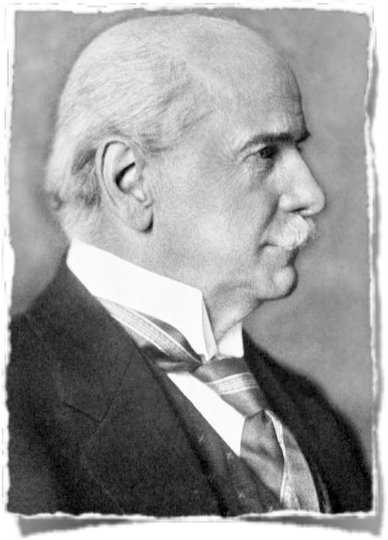
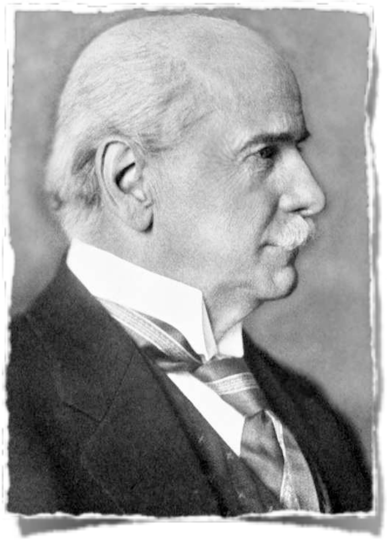
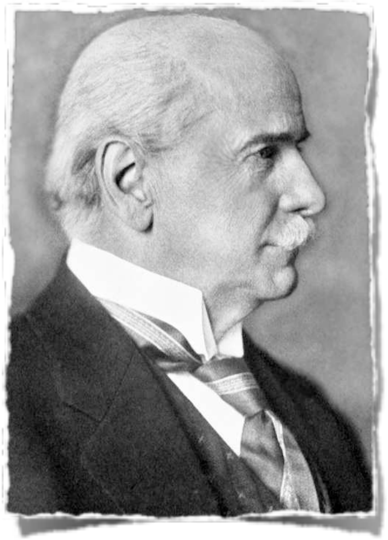
Data Streams, Dyck Languages, and *Detecting Dubious Data Structures*

Amit Chakrabarti *Dartmouth College*
Graham Cormode *AT&T Research Labs*
Ranganath Kondapally *Dartmouth College*
Andrew McGregor *University of Massachusetts, Amherst*



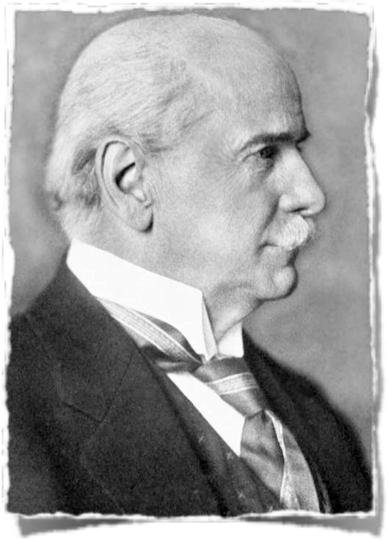


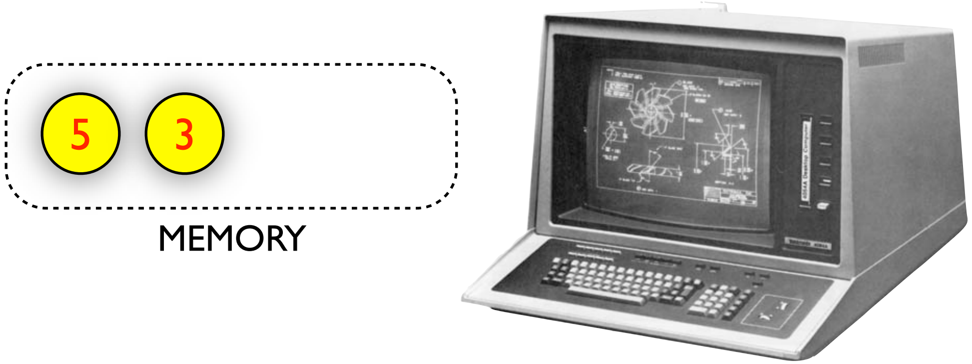
- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:



- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

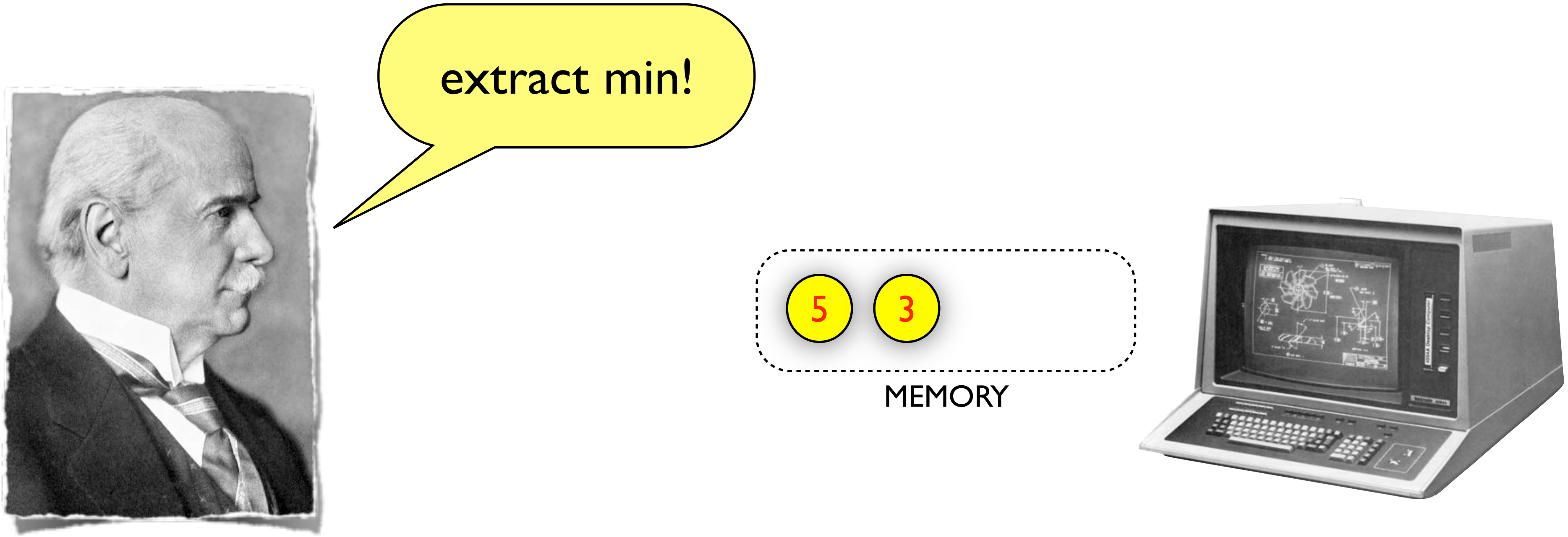
  ins(5)





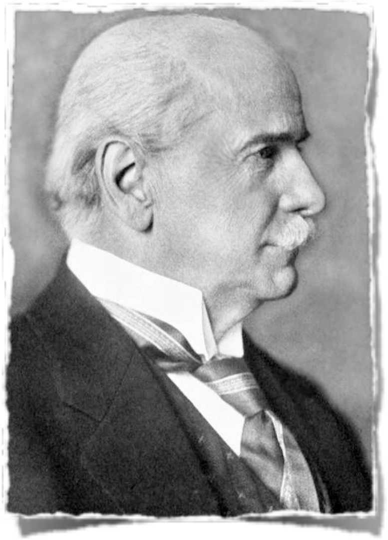
- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

  ins(5) ins(3)



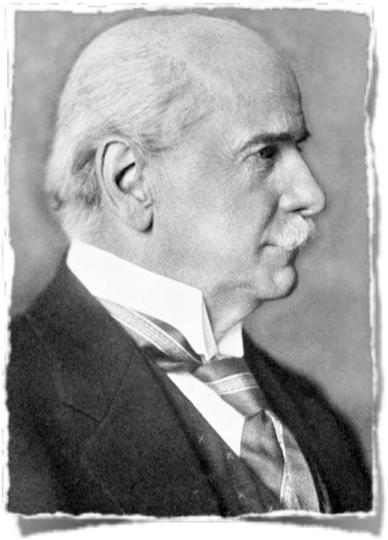
- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

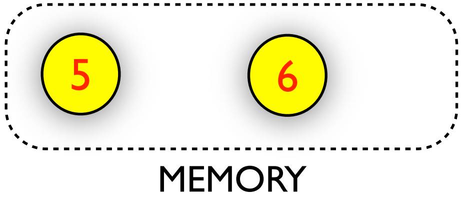
ins(5) ins(3)



- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

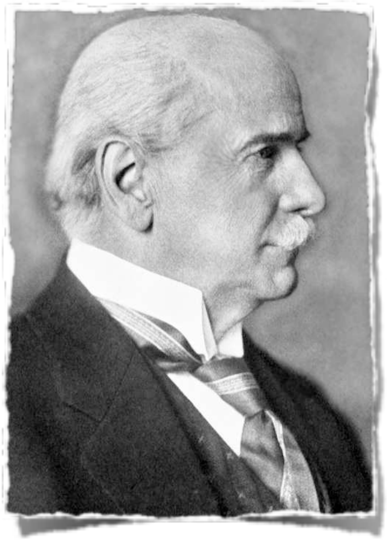
  ins(5) ins(3) ext(3)

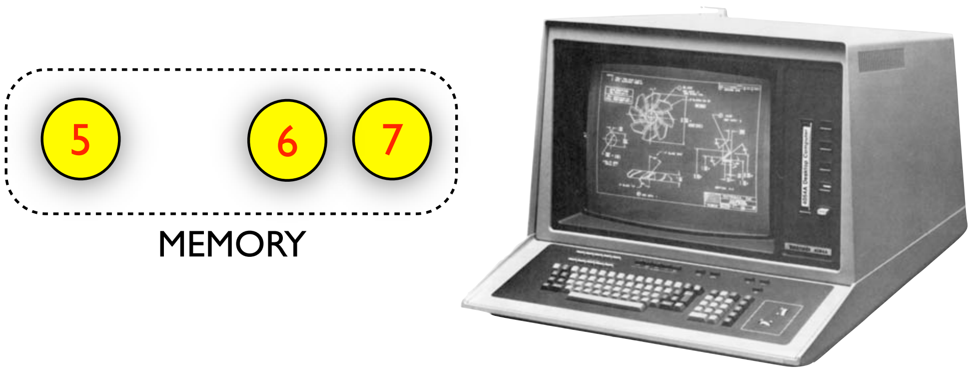




- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

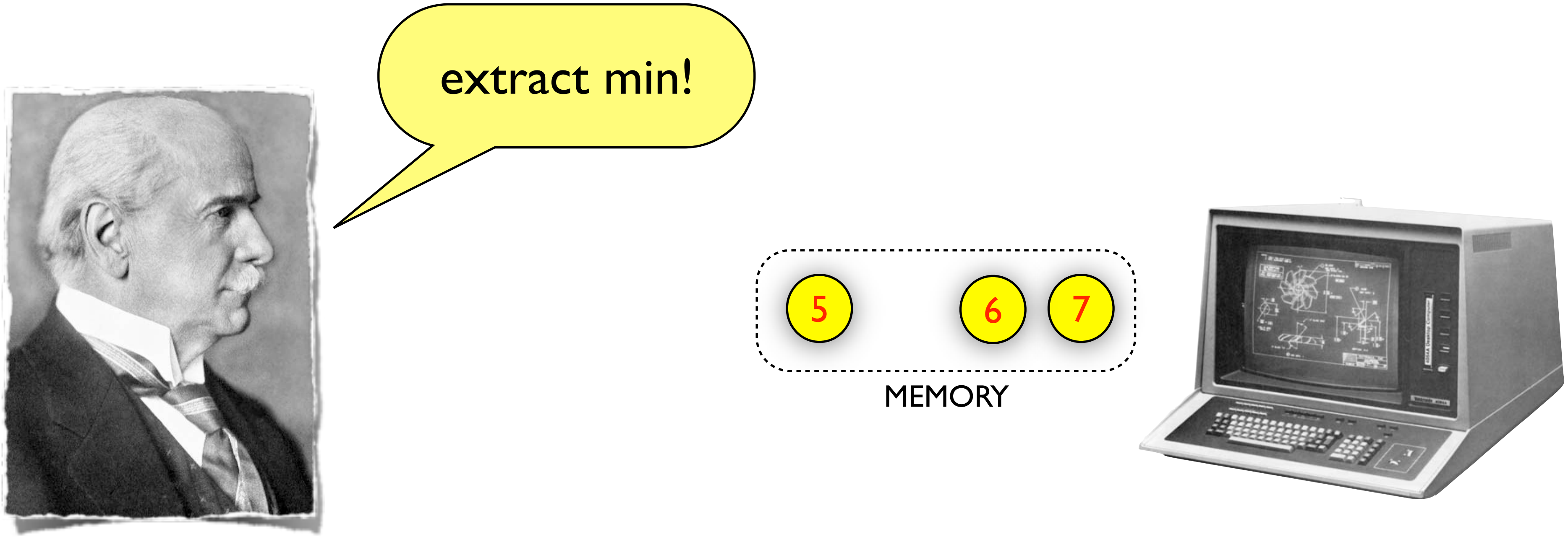
  ins(5) ins(3) ext(3) ins(6)





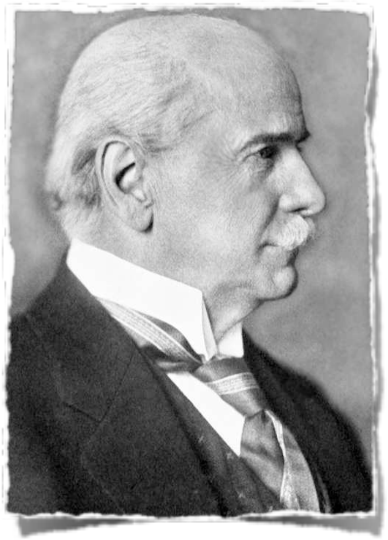
- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

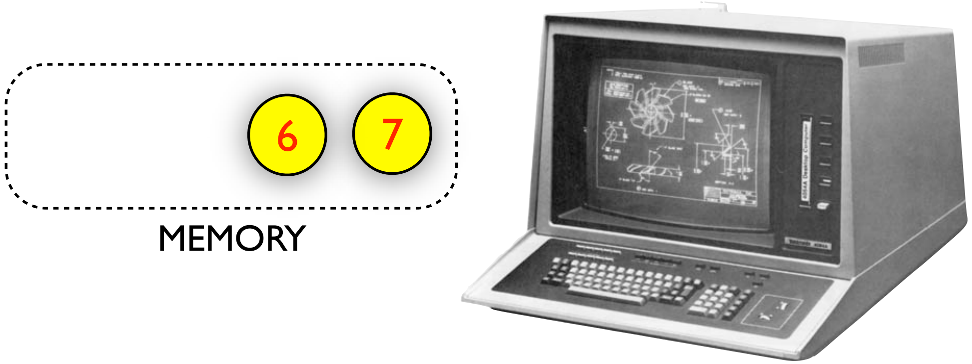
ins(5) ins(3) ext(3) ins(6) ins(7)



- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

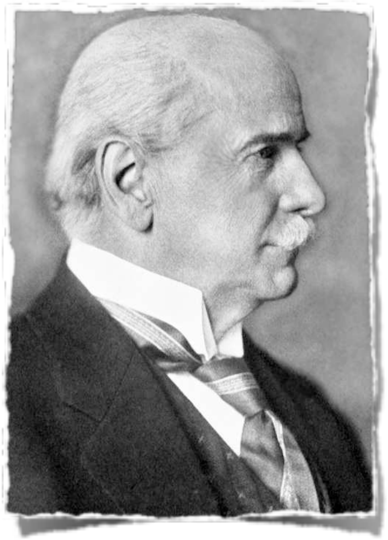
ins(5) ins(3) ext(3) ins(6) ins(7)

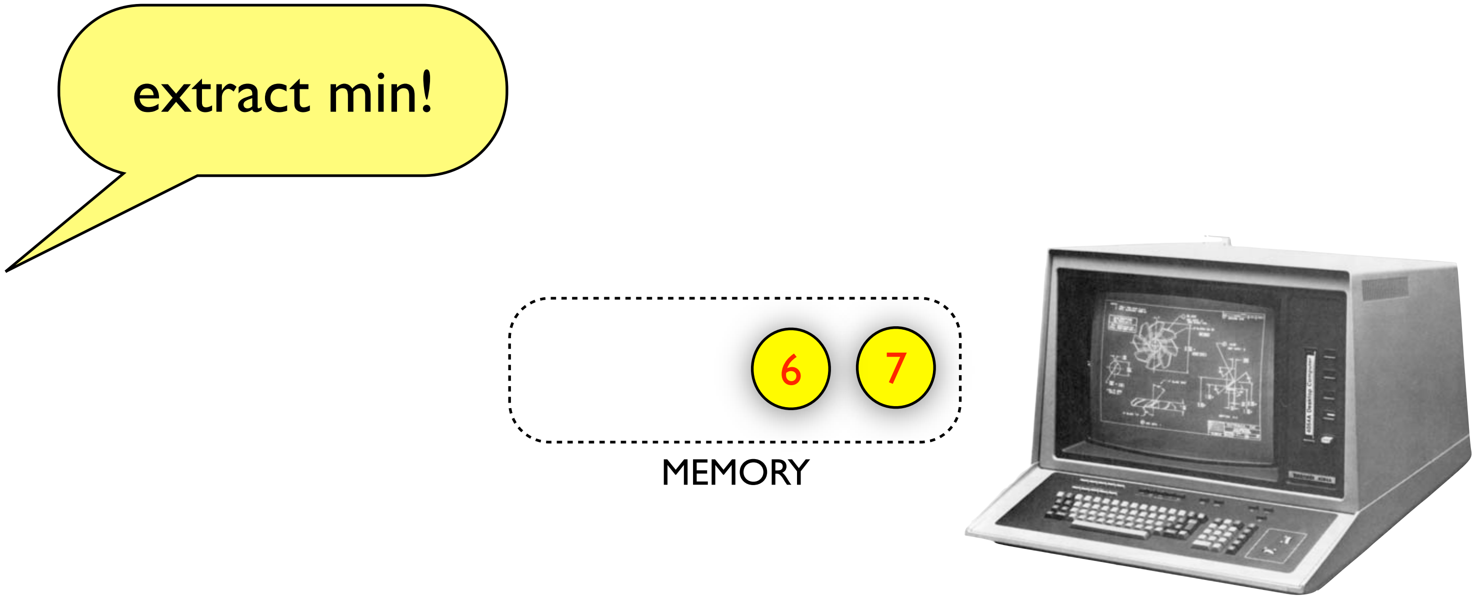




- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

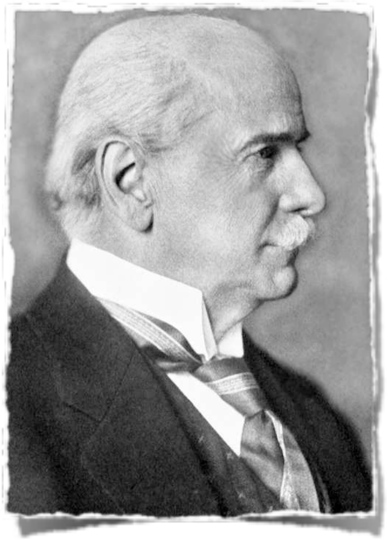
ins(5) ins(3) ext(3) ins(6) ins(7) ext(5)

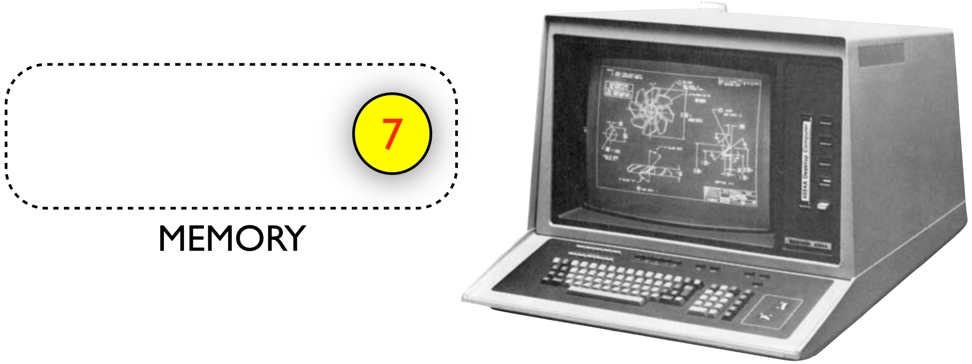




- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

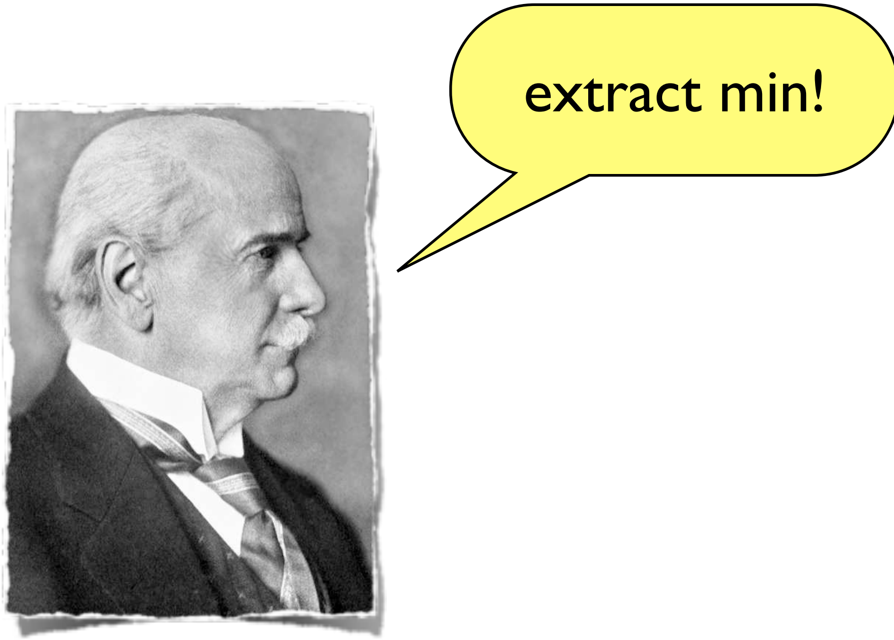
  ins(5) ins(3) ext(3) ins(6) ins(7) ext(5)





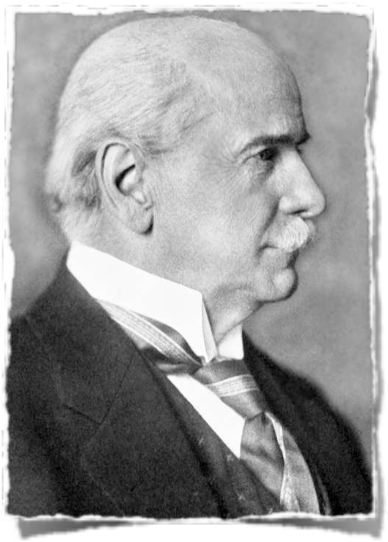
- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

  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:

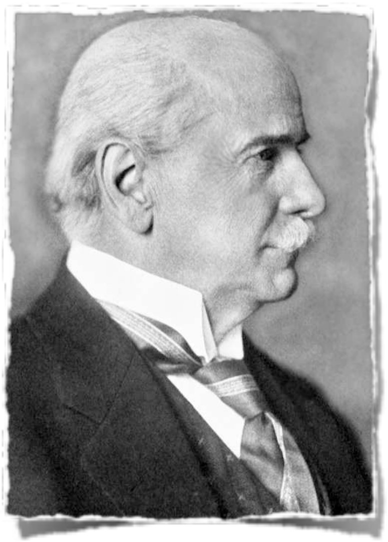
  ins(5) ins(3) ext(3) ins(6) ins(7) ext(5) ext(6)

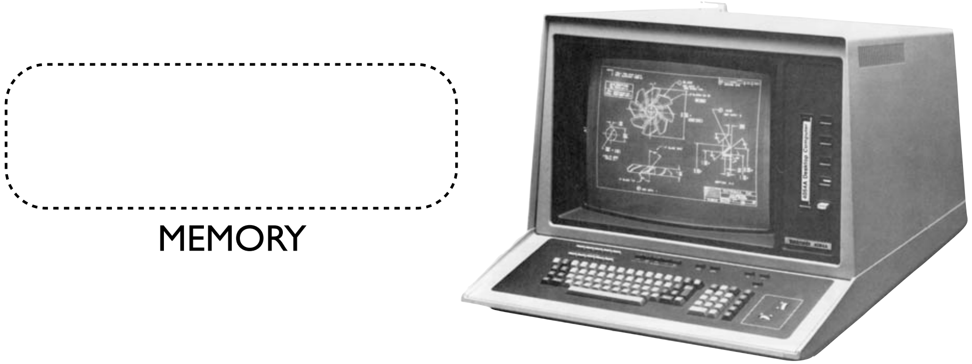


MEMORY

- 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) ext(7)

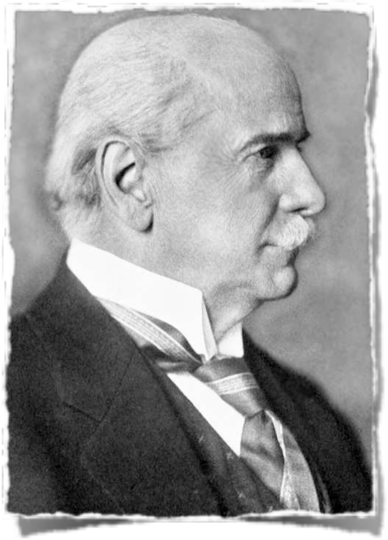


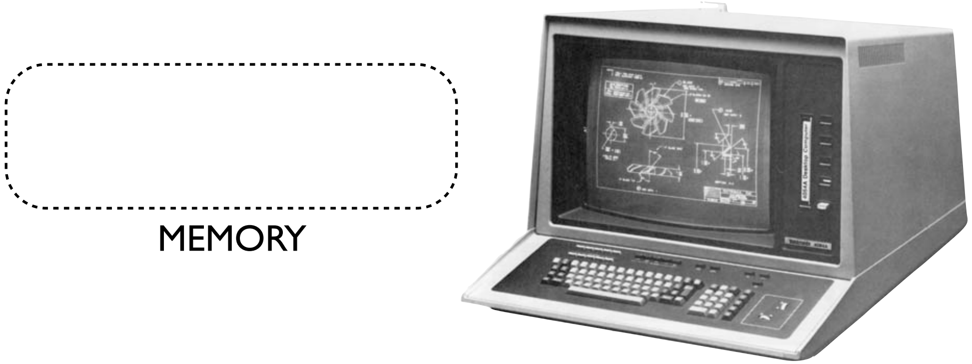


- At each step, user inserts a value into the memory or asks that the *smallest* value is extracted:

ins(5) ins(3) ext(3) ins(6) ins(7) ext(5) ext(6) 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:

  ins(5) ins(3) ext(3) ins(6) ins(7) ext(5) ext(6) 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.

  ins(5), ins(3), ext(3), ins(7), ext(5), ext(7) $\in$ PQ

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) $\in$ PQ

ins(5), ext(3), ins(3), ins(7), ext(7), ext(5) $\notin$ PQ

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) $\in$ PQ

  ins(5), ext(3), ins(3), ins(7), ext(7), ext(5) $\notin$ 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) $\in$ PQ

  ins(5), ext(3), ins(3), ins(7), ext(7), 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!

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) $\in$ PQ

  ins(5), ext(3), ins(3), ins(7), ext(7), 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.

  ins(5), ins(3), ext(3), ins(7), ext(5), ext(7) $\in$ PQ

  ins(5), ext(3), ins(3), ins(7), ext(7), 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...*

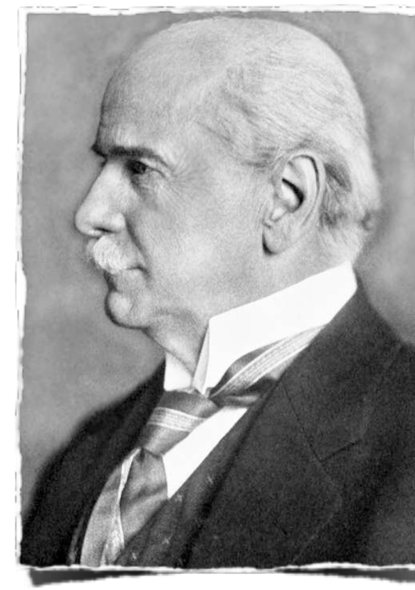
  i. We'll design an algorithm that uses $O(\sqrt{N})$ space!

  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

1. 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

- *<u>Thm:</u>* 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.

$$\prod_{u \text{ inserts}} (x - u) \stackrel{?}{=} \prod_{u \text{ extracts}} (x - u)$$

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.

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

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.

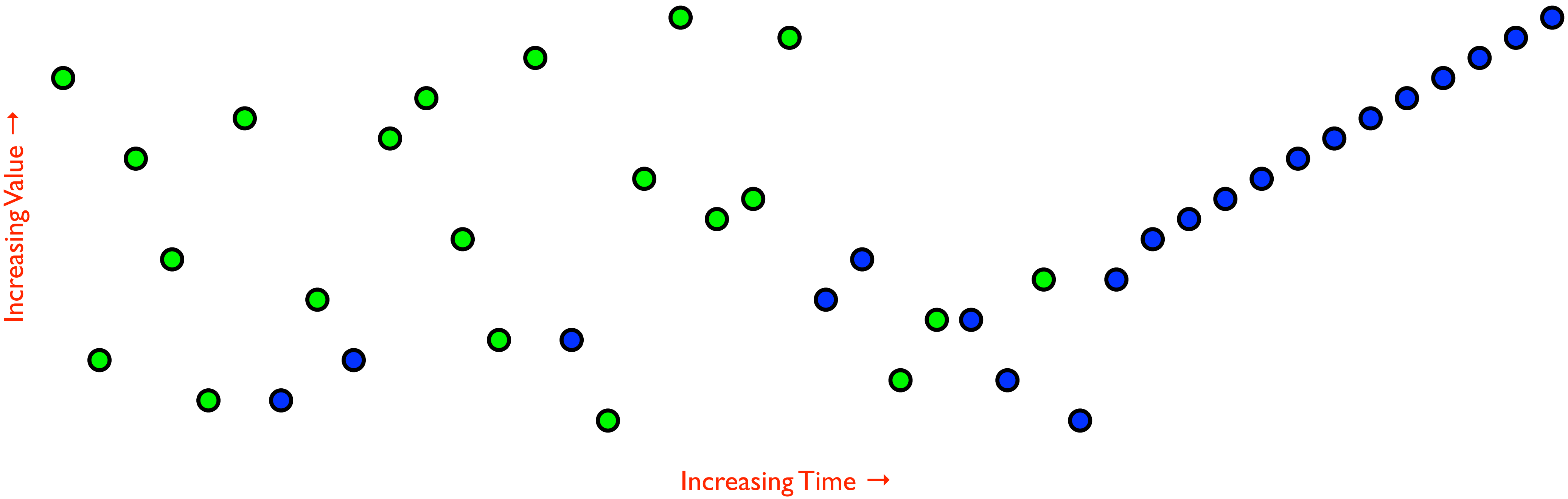
$$\prod_{u \text{ inserts}} (r - u) \stackrel{?}{=} \prod_{u \text{ extracts}} (r - u)$$

! *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:*

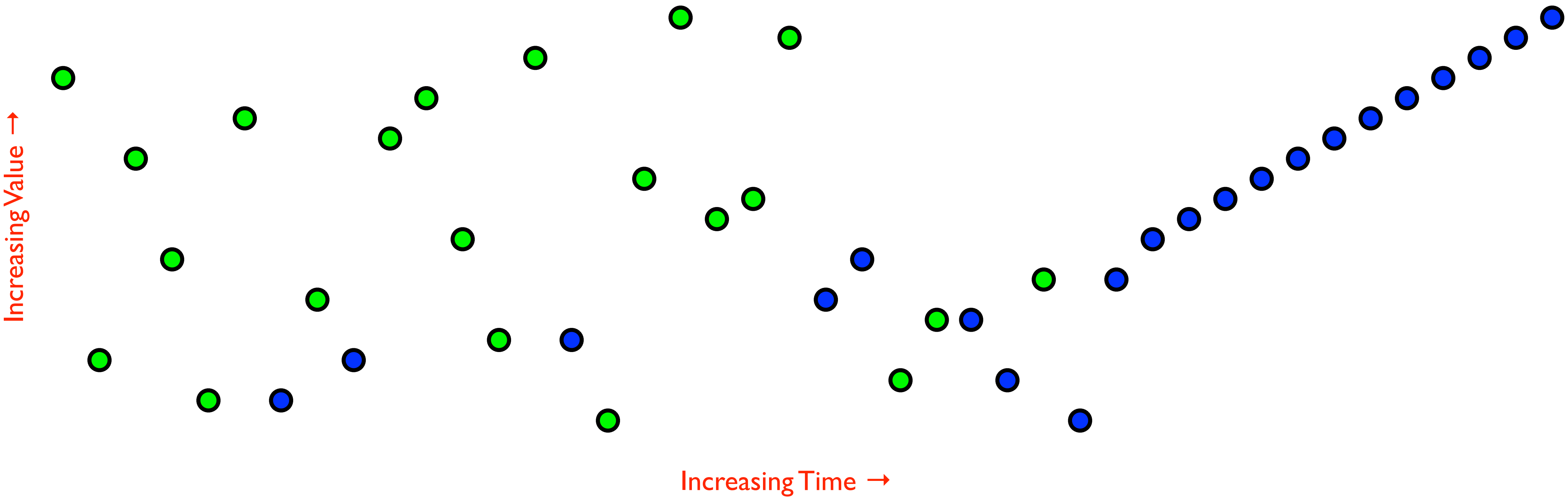
ins(u) ... ext(v) ... ext(u) for some u < v

Epochs and Local Bad Patterns...

Epochs and Local Bad Patterns...

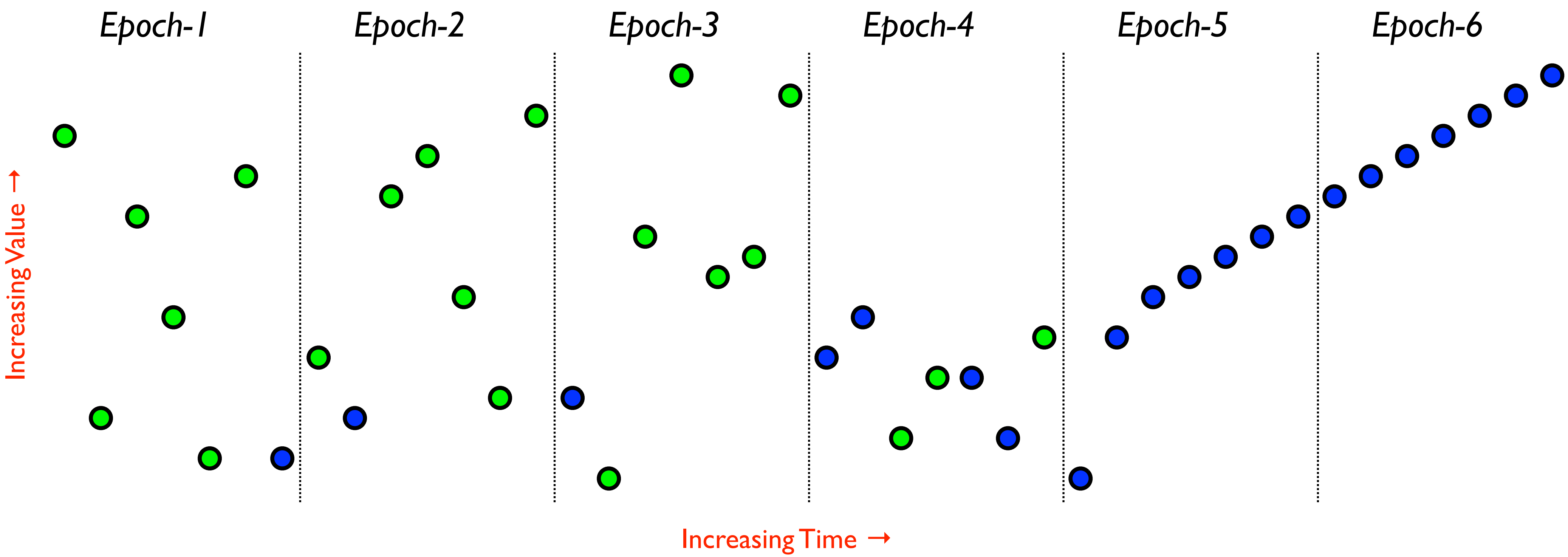


Epochs and Local Bad Patterns...



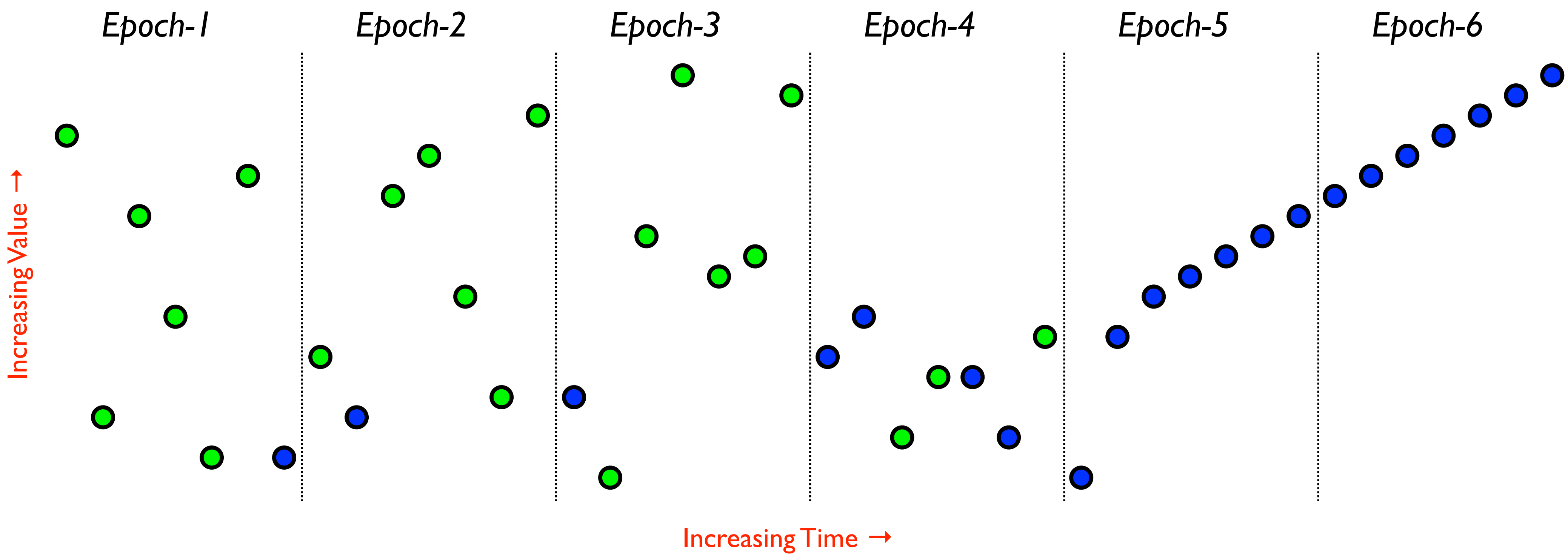
- Split length N sequence into $\sqrt{N}$ epochs of length $\sqrt{N}$

Epochs and Local Bad Patterns...



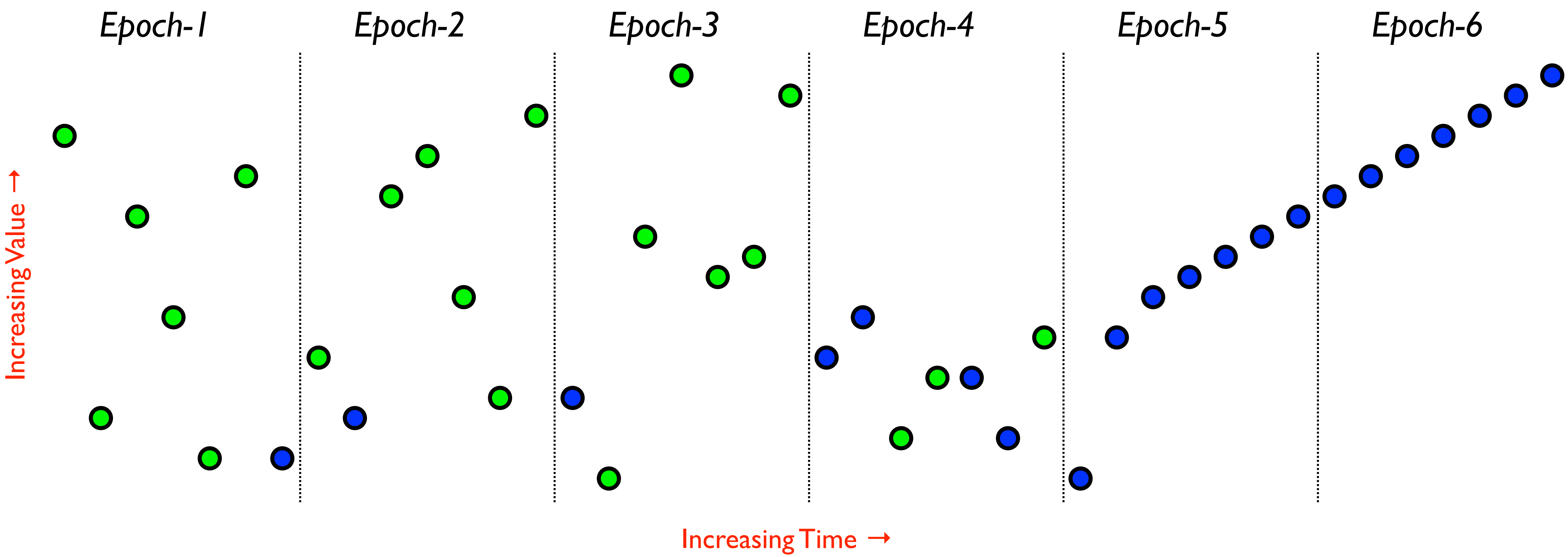
- Split length N sequence into $\sqrt{N}$ epochs of length $\sqrt{N}$

Epochs and Local Bad Patterns...



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

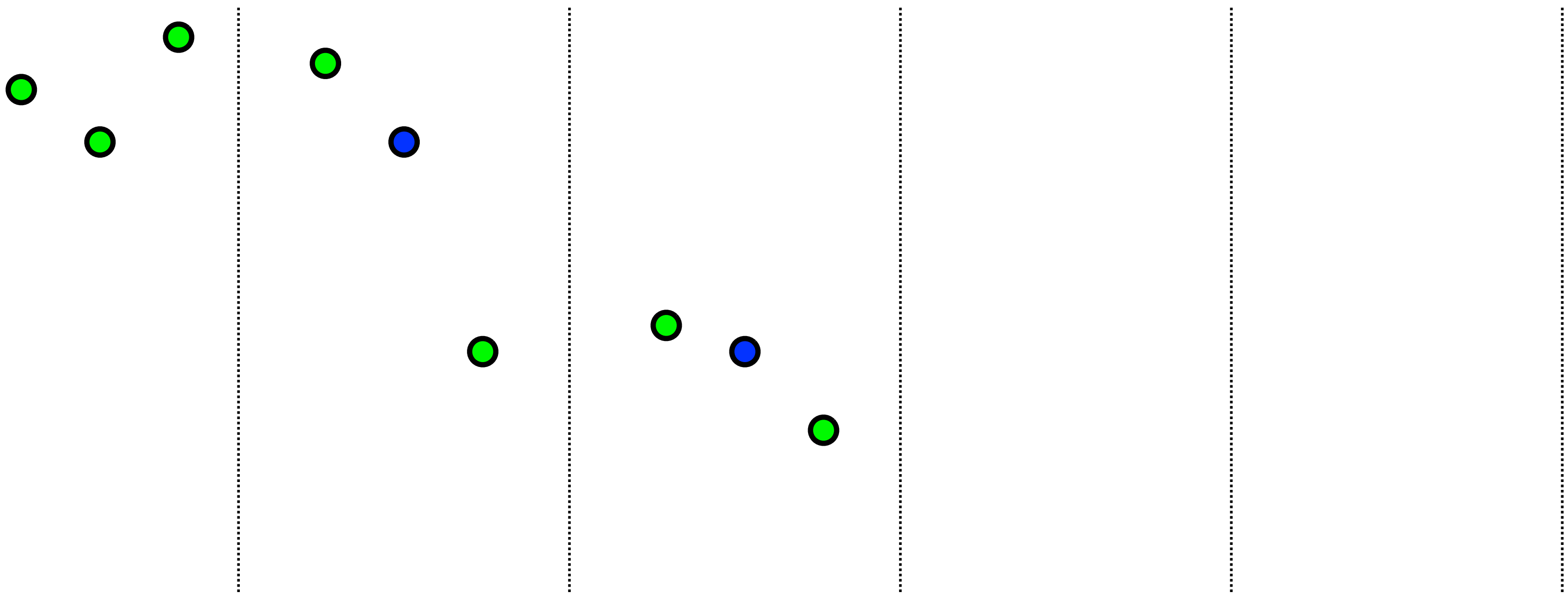
Epochs and Local Bad Patterns...



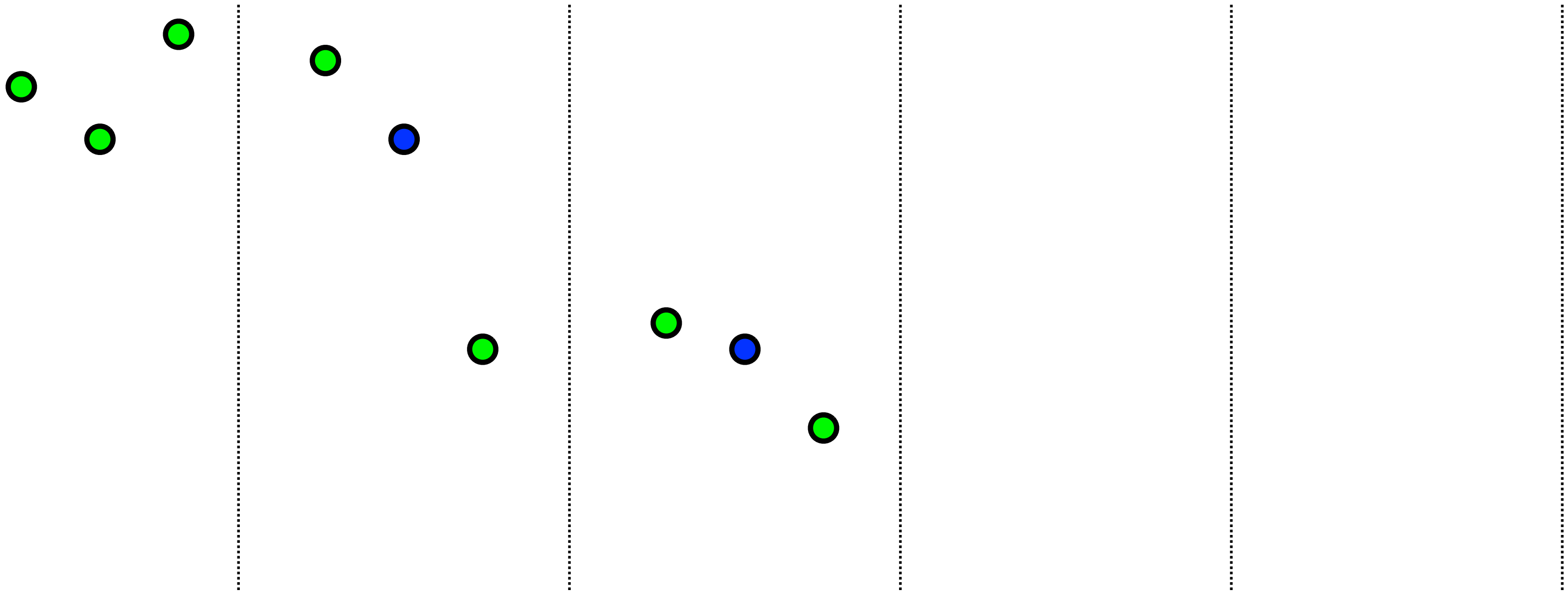
- Split length N sequence into $\sqrt{N}$ epochs of length $\sqrt{N}$
- *Defn:* Bad pattern ins(u) ... ext(v) ... ext(u) is *local* if ins(u) and ext(v) occur in same epoch and *long-range* otherwise.
- Using $O(\sqrt{N})$ space, we can buffer each epoch and check for local bad patterns.

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

Catching Long-Range Bad Patterns… 1/2

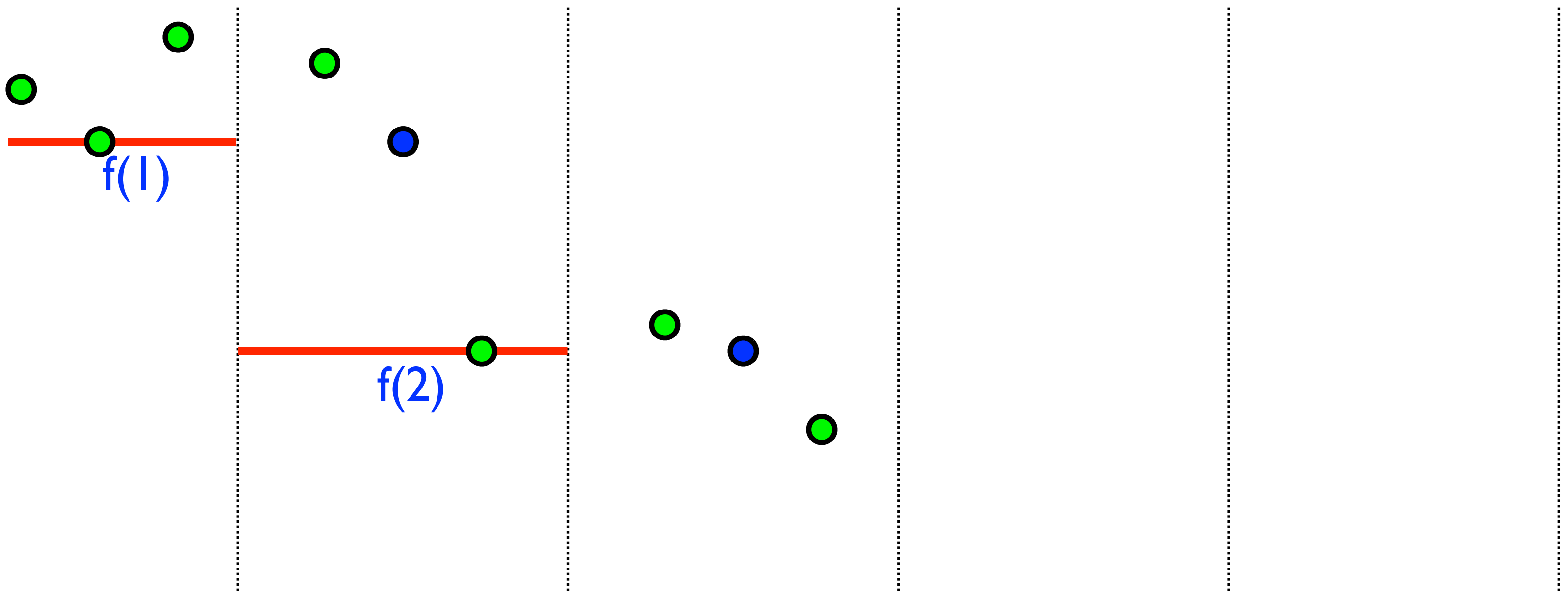


Catching Long-Range Bad Patterns... 1/2



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

Catching Long-Range Bad Patterns… 1/2

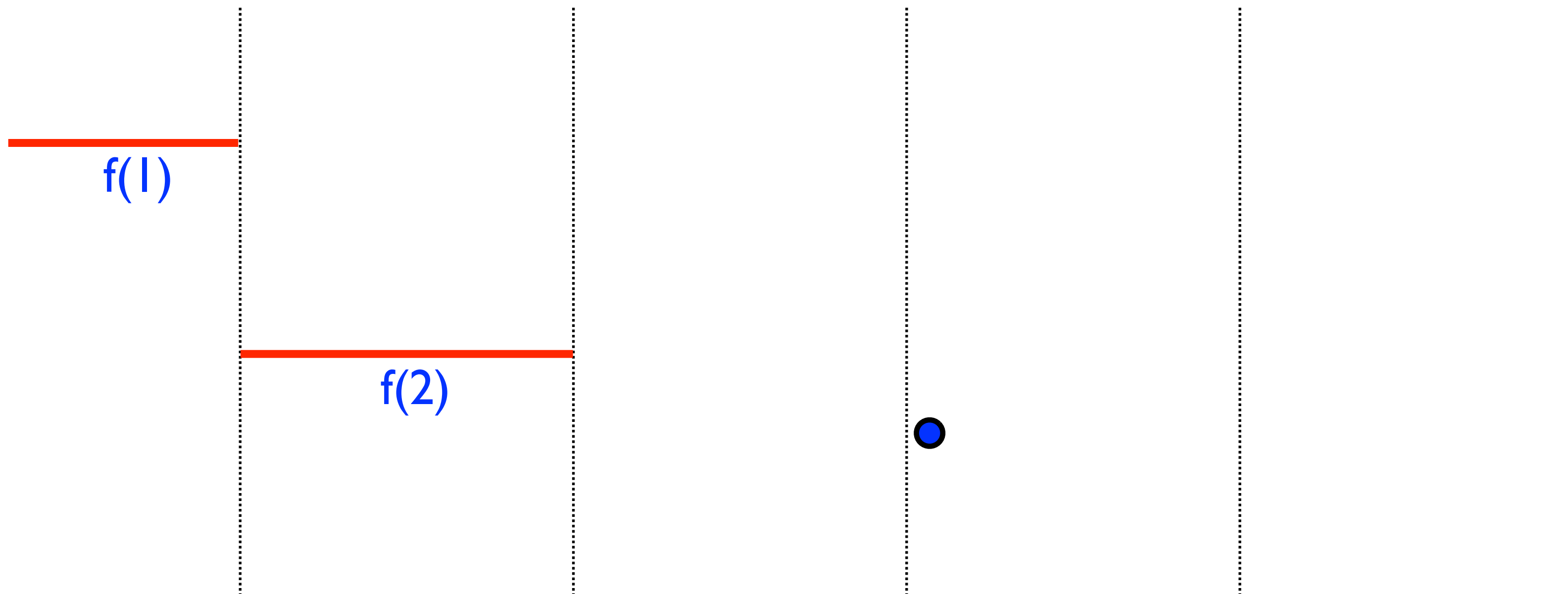


- Maintain the max value extracted between end of i-th epoch and *current time*. Call it f(i).

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

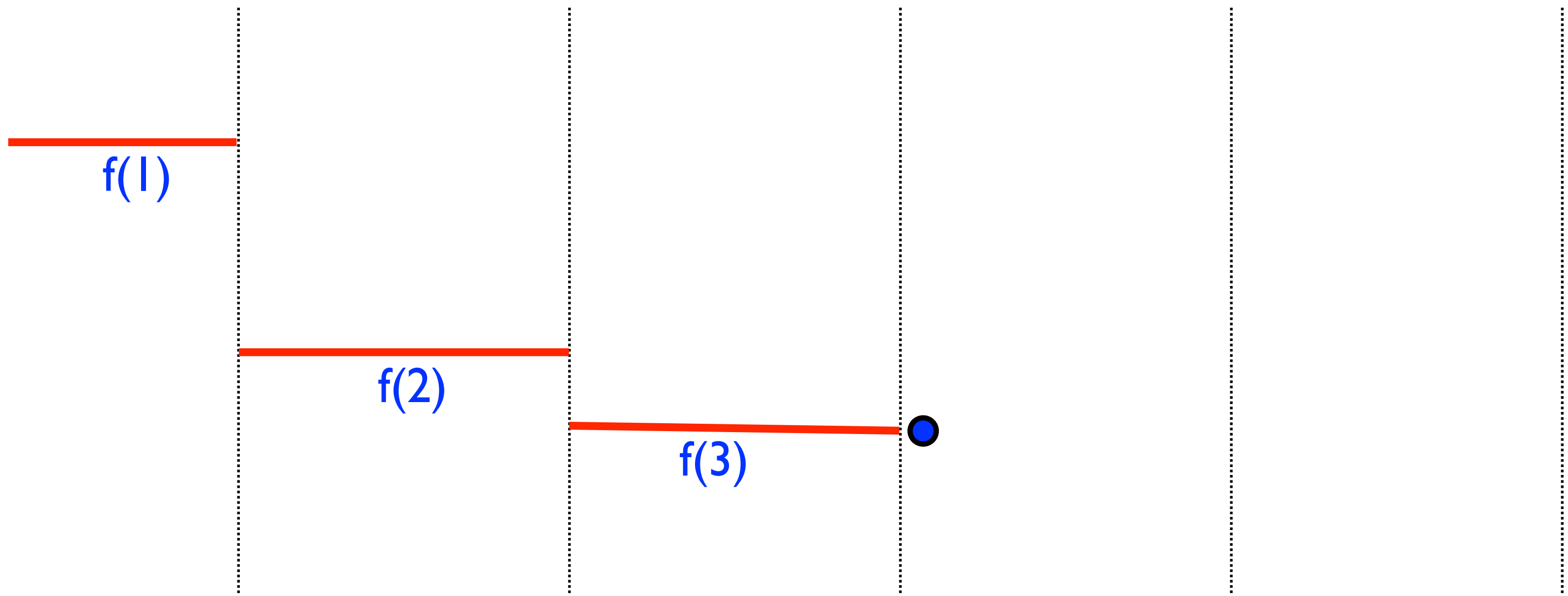
- Maintain the max value extracted between end of i-th epoch and *current time*. Call it f(i).

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



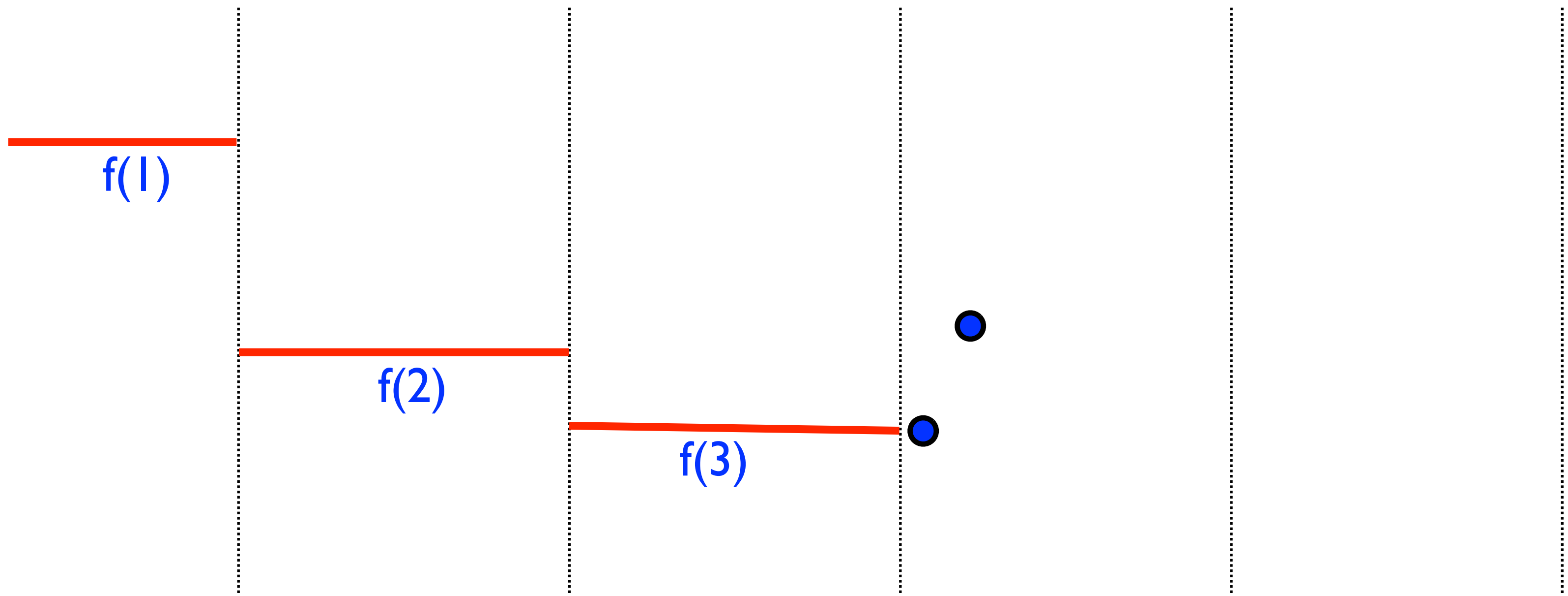
- Maintain the max value extracted between end of i-th epoch and *current time*. Call it f(i).

Catching Long-Range Bad Patterns… 1/2



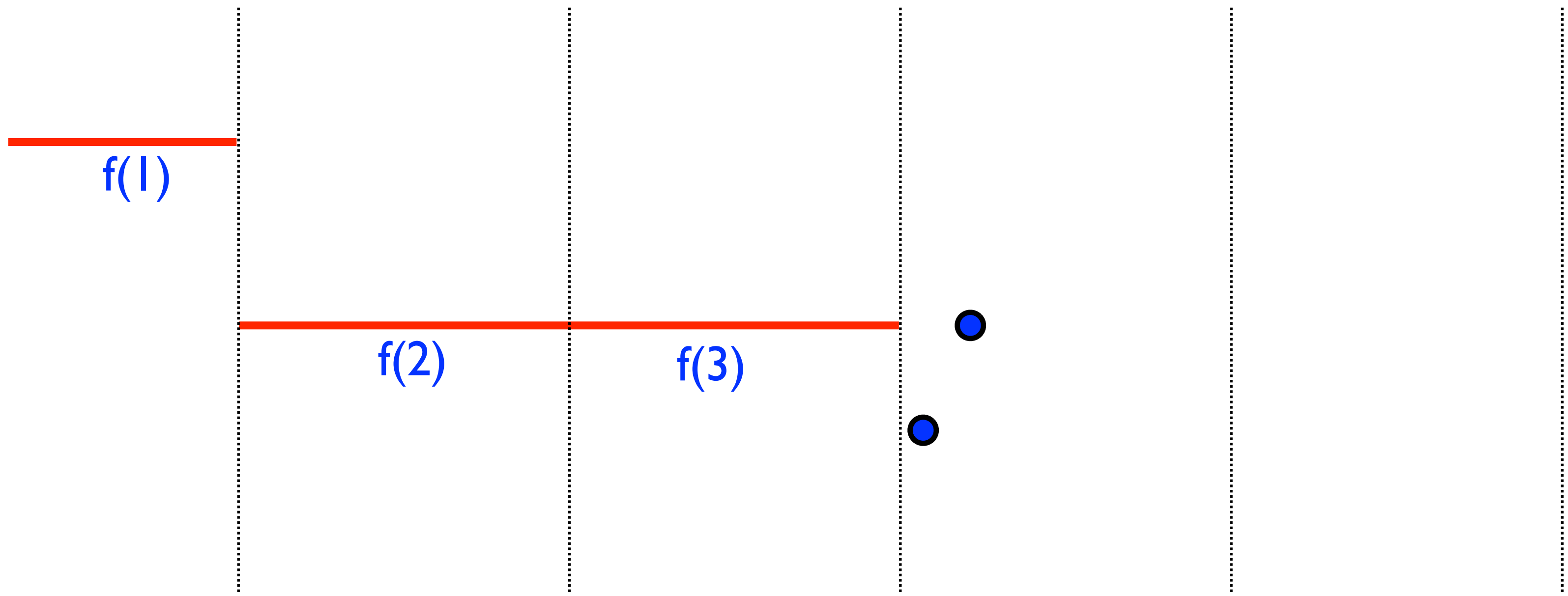
- Maintain the max value extracted between end of i-th epoch and *current time*. Call it f(i).

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



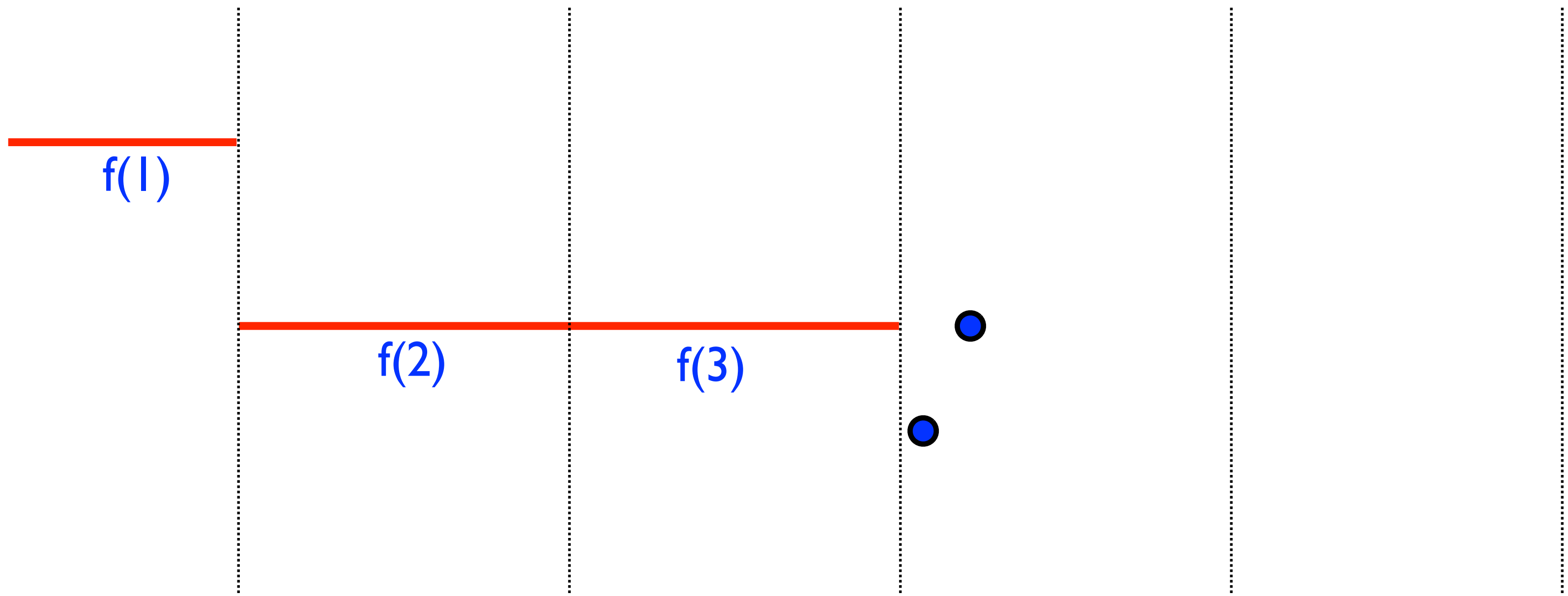
- Maintain the max value extracted between end of i-th epoch and *current time*. Call it f(i).

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).

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).
- *Defn:* Each ins(u) or ext(u) is *adopted* by earliest epoch k with $f(k) \leq u$.

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

Catching Long-Range Bad Patterns… 2/2

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

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

- *Lemma:* If ins(u) ... ext(v) ... ext(u) is a long-range bad pattern then ins(u) and ext(u) are adopted by different epochs.
- *Proof:*
  i. Let ins(u) be adopted by k-th epoch.

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

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

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

- *Lemma:* If ins(u) ... ext(v) ... ext(u) is a long-range bad pattern then ins(u) and ext(u) are adopted by different epochs.
- *Proof:*
  1. Let ins(u) be adopted by k-th epoch.
  2. After v is extracted $f(k) \geq v > u$.
  3. ext(u) can no longer be adopted by k-th epoch.

Catching Long-Range Bad Patterns… 2/2

- *Lemma:* If ins(u) … ext(v) … ext(u) is a long-range bad pattern then ins(u) and ext(u) are adopted by different epochs.
- *Proof:*
  i. Let ins(u) be adopted by k-th epoch.
  ii. After v is extracted $f(k) \geq v > u$.
  iii. ext(u) can no longer be adopted by k-th epoch.
- *Lemma:* If there are no bad patterns, every ins(u) and ext(u) pair get adopted by the same epoch.

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

- *Lemma:* If ins(u) ... ext(v) ... ext(u) is a long-range bad pattern then ins(u) and ext(u) are adopted by different epochs.
- *Proof:*
  i. Let ins(u) be adopted by k-th epoch.
  ii. After v is extracted $f(k) \geq v > u$.
  iii. ext(u) can no longer be adopted by k-th epoch.
- *Lemma:* If there are no bad patterns, every ins(u) and ext(u) pair get adopted by the same epoch.
- *Algorithm:* Using fingerprints to check: for each epoch k

  {u : ins(u) adopted by k} = {u : ext(u) 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

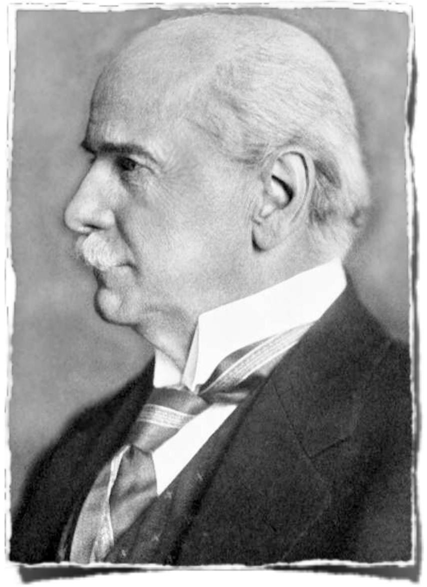
- *Thm:* There exists a $O(\sqrt{N} \log N)$ space algorithm with $O(\log N)$ amortized update time for recognizing PQ.

  "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

Communication Complexity

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

Communication Complexity

x

y, c

- 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 $c = x_k$.

Communication Complexity

x

y, c

- 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 AI_n requires $\Omega(n)$ bits sent. [Miltersen et al. JCSS '98]

Multi-player Augmented Index

Multi-player Augmented Index

x^1 y^1, c^1 x^2 y^2, c^2 x^3 y^3, c^3

- 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 AI_n

Multi-player Augmented Index

x^1 y^1, c^1 x^2 y^2, c^2 x^3 y^3, c^3 x^3 x^2 x^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 AI_n
- Want to determine if any of the AI_n instances are false using private messages communicated in the order

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

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 AI_n
- Want to determine if any of the AI_n instances are false using private messages communicated in the order

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

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

Multi-player Augmented Index

x^1 y^1, c^1 x^2 y^2, c^2 x^3 y^3, c^3 x^3 x^2 x^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 AI_n
- Want to determine if any of the 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.

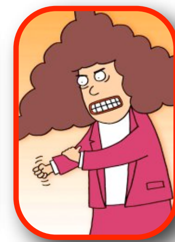
Reduction from multi-player AI to PQ…


0010


0, 0


0100

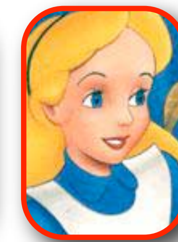

01,1


1010


,1


1010

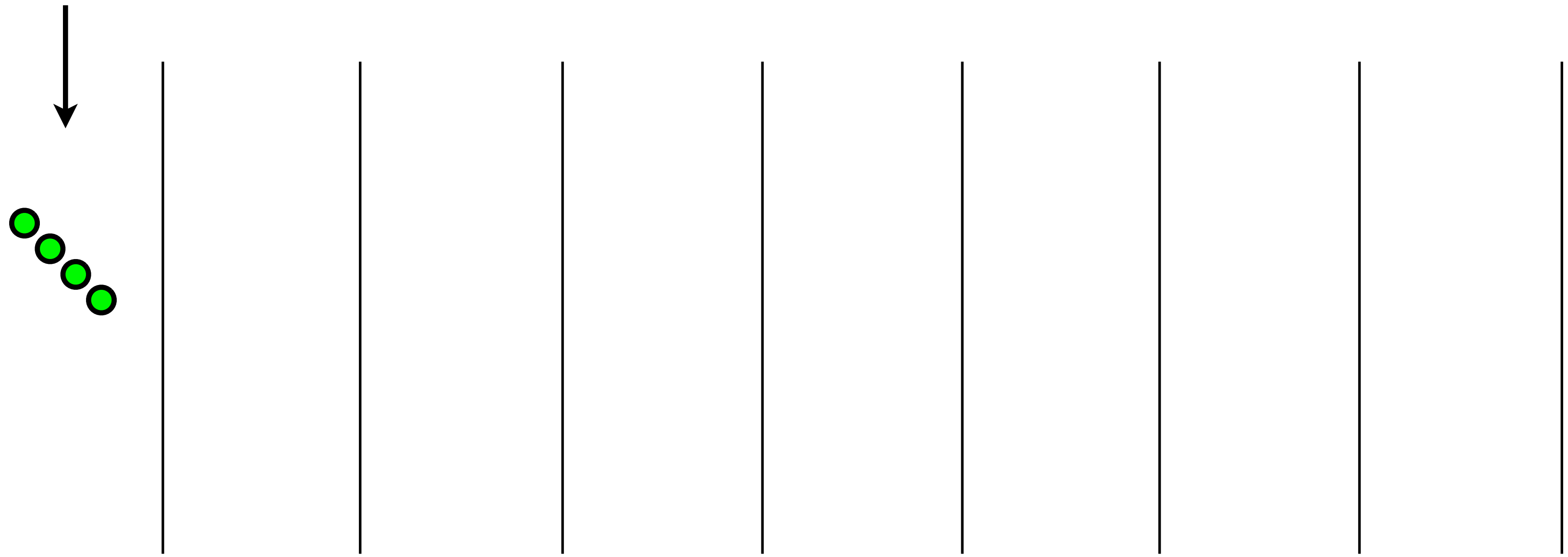

0100


0010

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]

Reduction from multi-player AI to PQ...

0010 | 0, 0 | 0100 | 01,1 | 1010 | ,1 | 1010 | 0100 | 0010

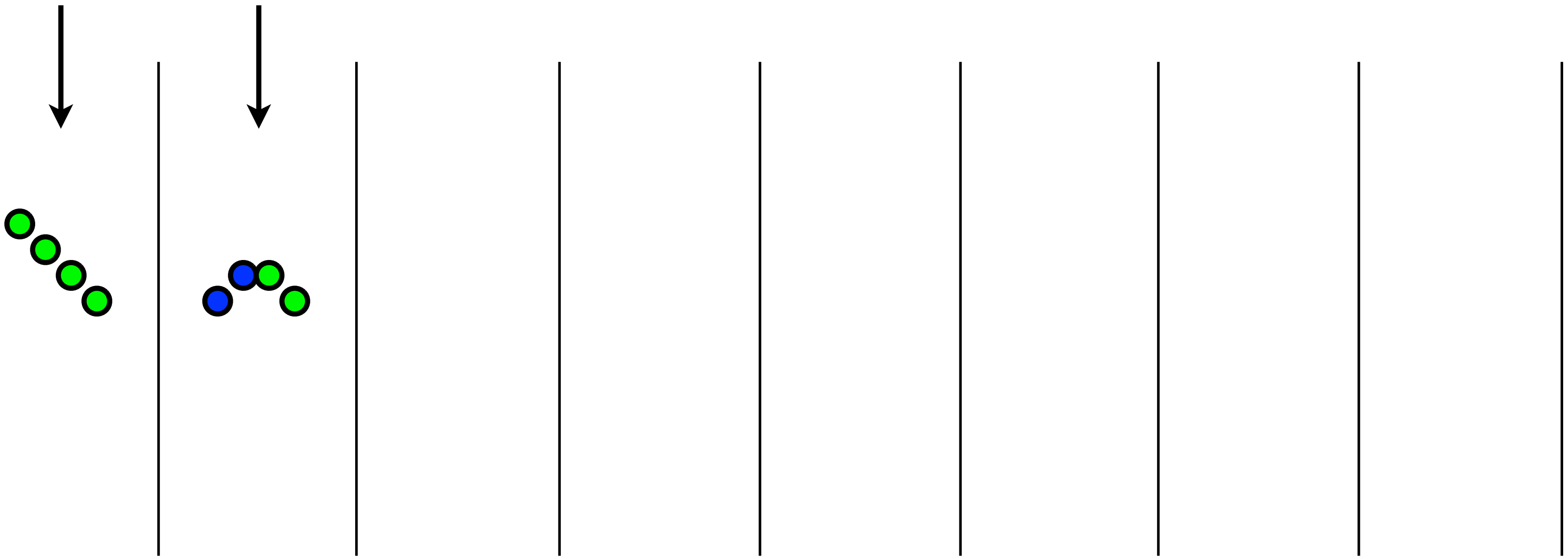


ins(12.0)
ins(11.1)
ins(10.0)
ins(9.0)

“Ascension Problem” [Magniez, Mathieu, Nayak ’10]

Reduction from multi-player AI to PQ...

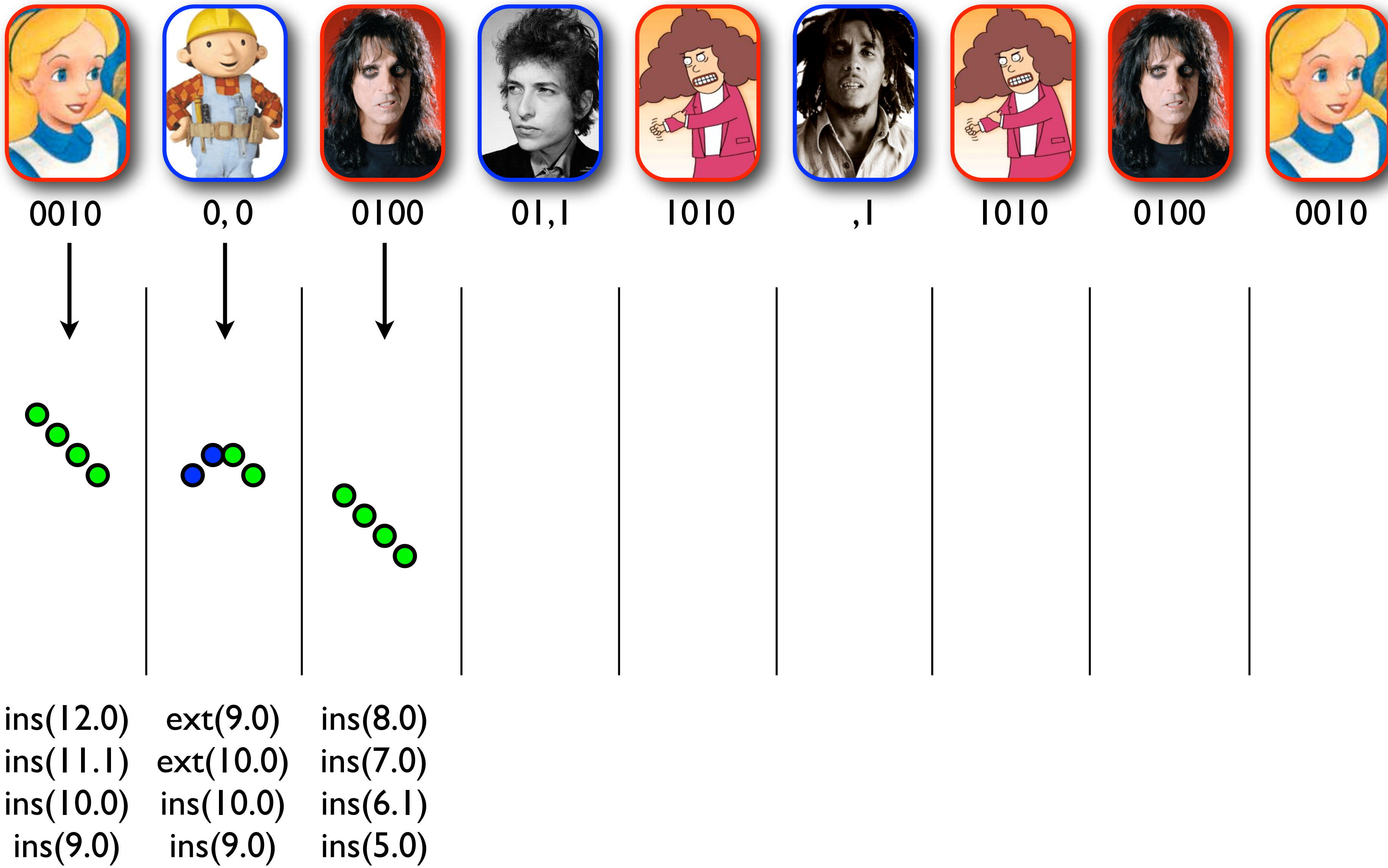
0010   0, 0   0100   01,1   1010   ,1   1010   0100   0010



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]

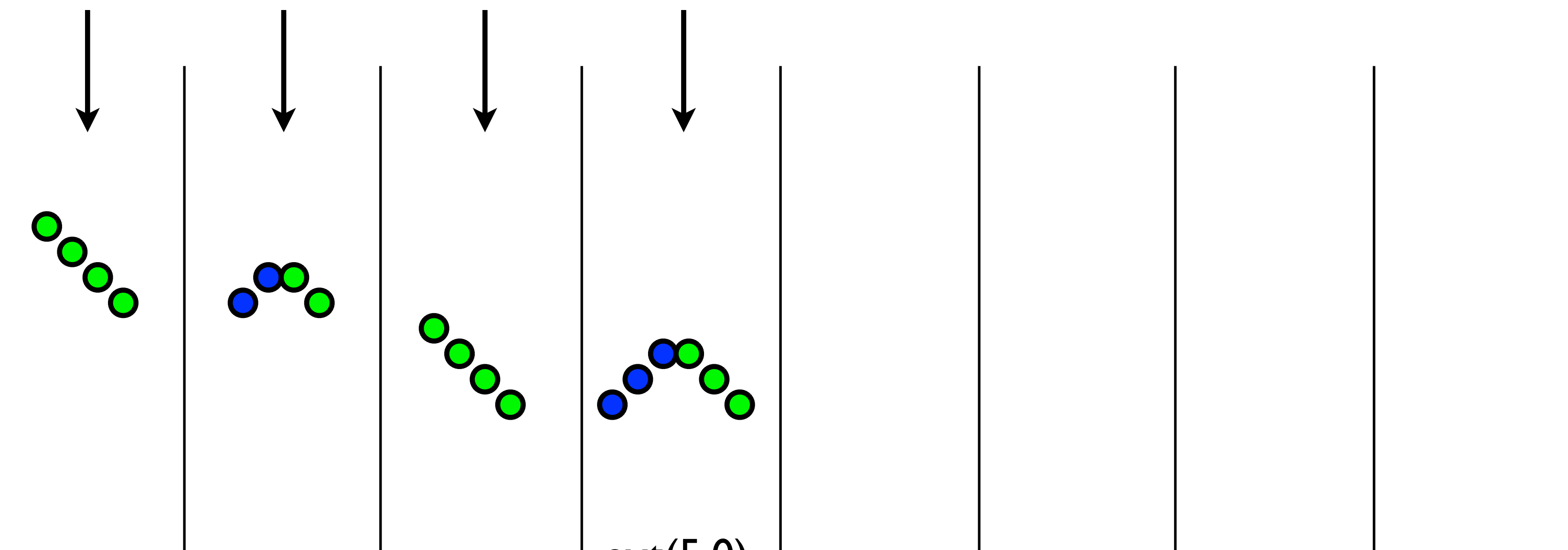
Reduction from multi-player AI to PQ...



"Ascension Problem" [Magniez, Mathieu, Nayak '10]

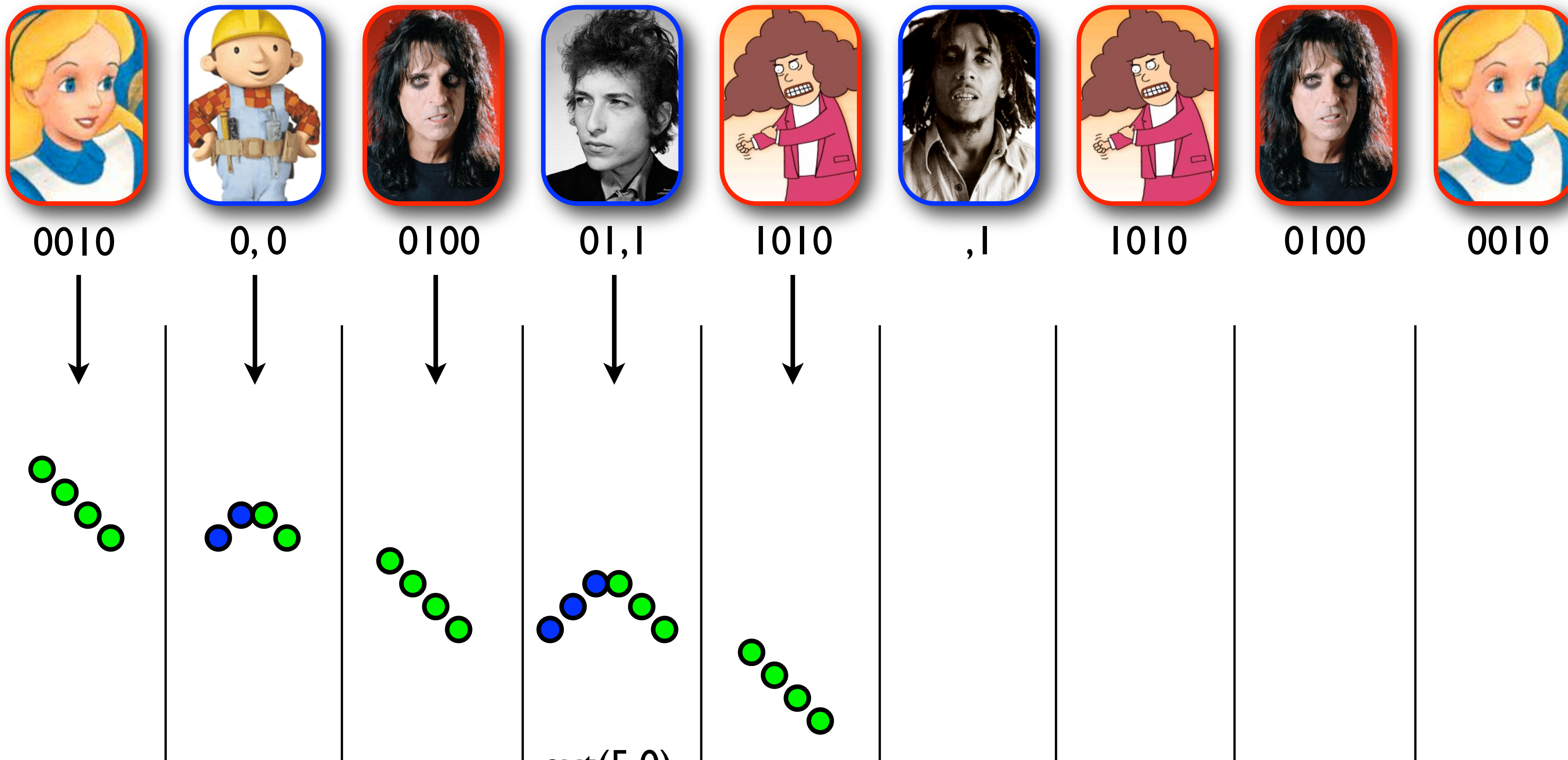
Reduction from multi-player AI to PQ...

0010	0, 0	0100	01,1	1010	,1	1010	0100	0010
			ext(5.0)					
ins(12.0)	ext(9.0)	ins(8.0)	ext(6.1)					
ins(11.1)	ext(10.0)	ins(7.0)	ext(7.1)					
ins(10.0)	ins(10.0)	ins(6.1)	ins(7.1)					
ins(9.0)	ins(9.0)	ins(5.0)	ins(6.1)					
			ins(5.0)					



"Ascension Problem" [Magniez, Mathieu, Nayak '10]

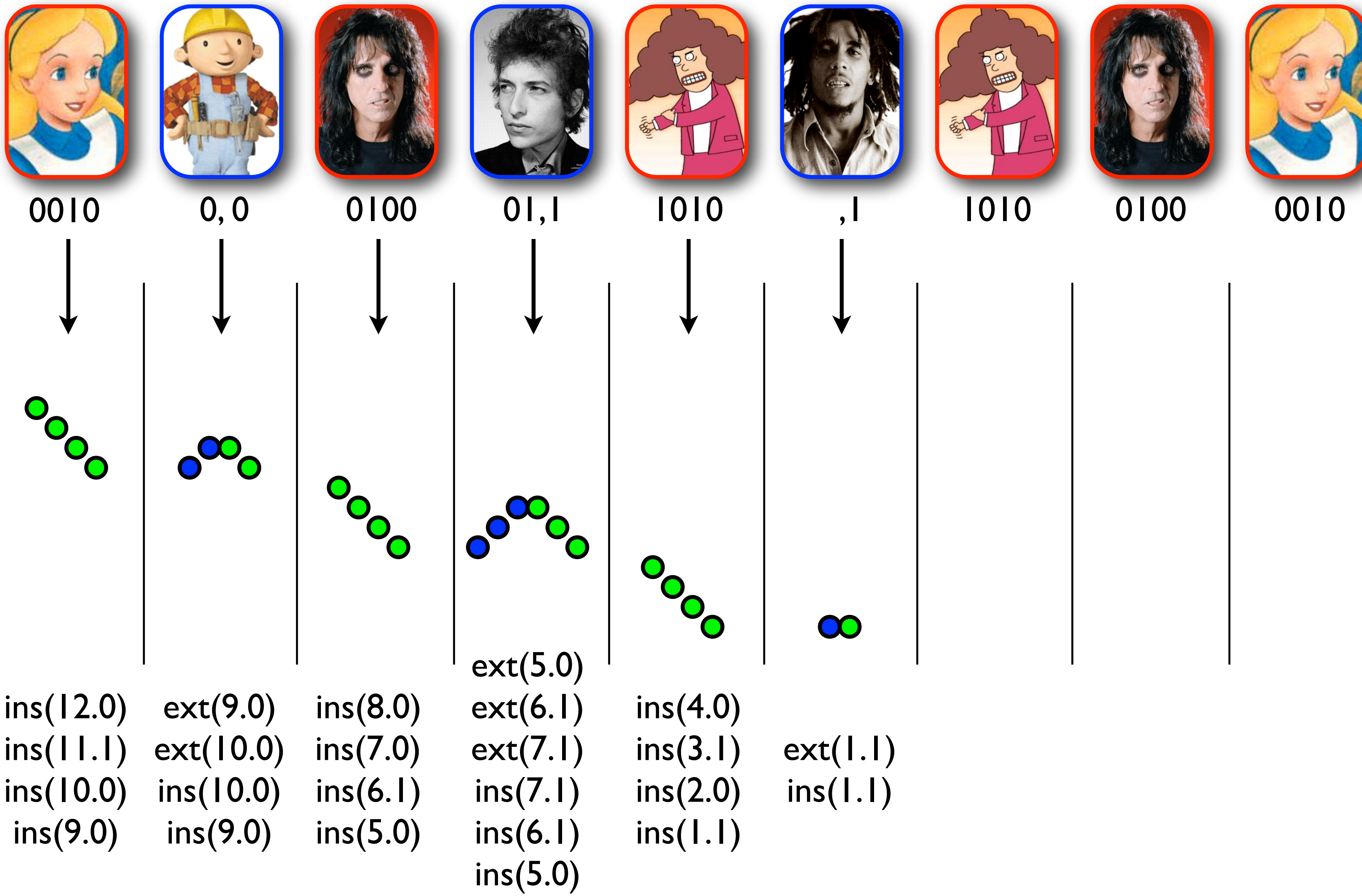
Reduction from multi-player AI to PQ...



			ext(5.0)	
ins(12.0)	ext(9.0)	ins(8.0)	ext(6.1)	ins(4.0)
ins(11.1)	ext(10.0)	ins(7.0)	ext(7.1)	ins(3.1)
ins(10.0)	ins(10.0)	ins(6.1)	ins(7.1)	ins(2.0)
ins(9.0)	ins(9.0)	ins(5.0)	ins(6.1)	ins(1.1)
			ins(5.0)	

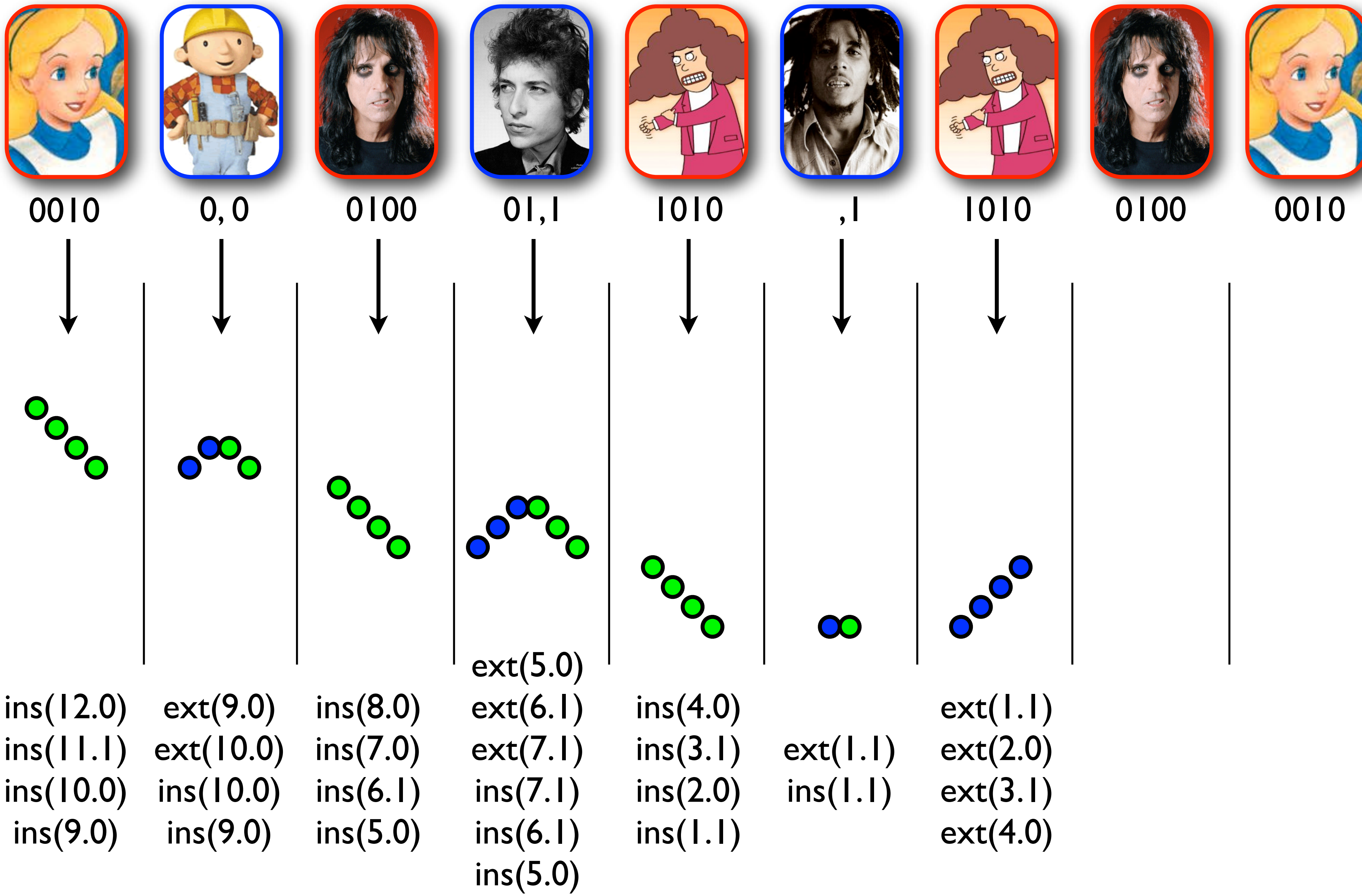
"Ascension Problem" [Magniez, Mathieu, Nayak '10]

Reduction from multi-player AI to PQ...



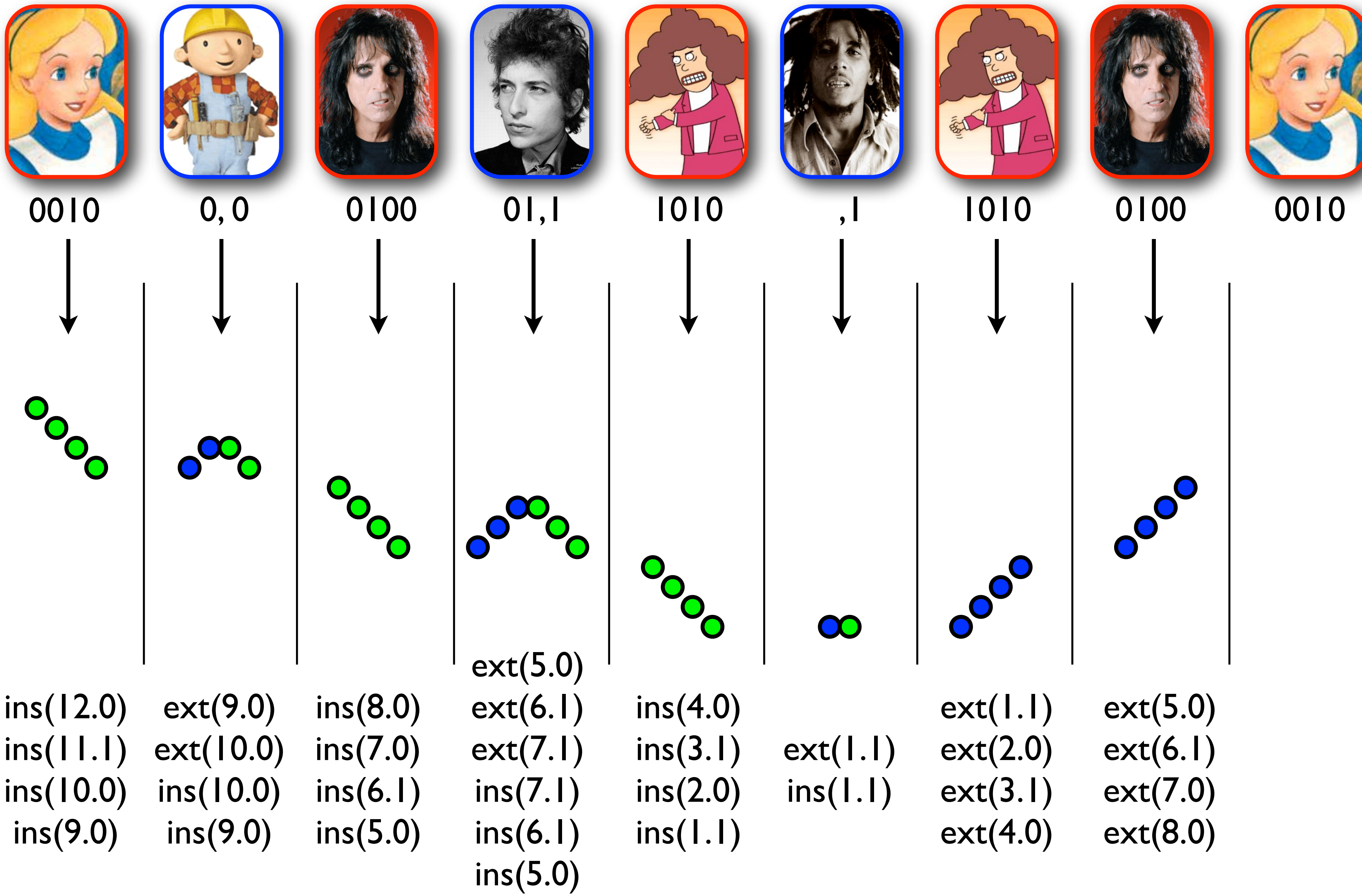
"Ascension Problem" [Magniez, Mathieu, Nayak '10]

Reduction from multi-player AI to PQ...



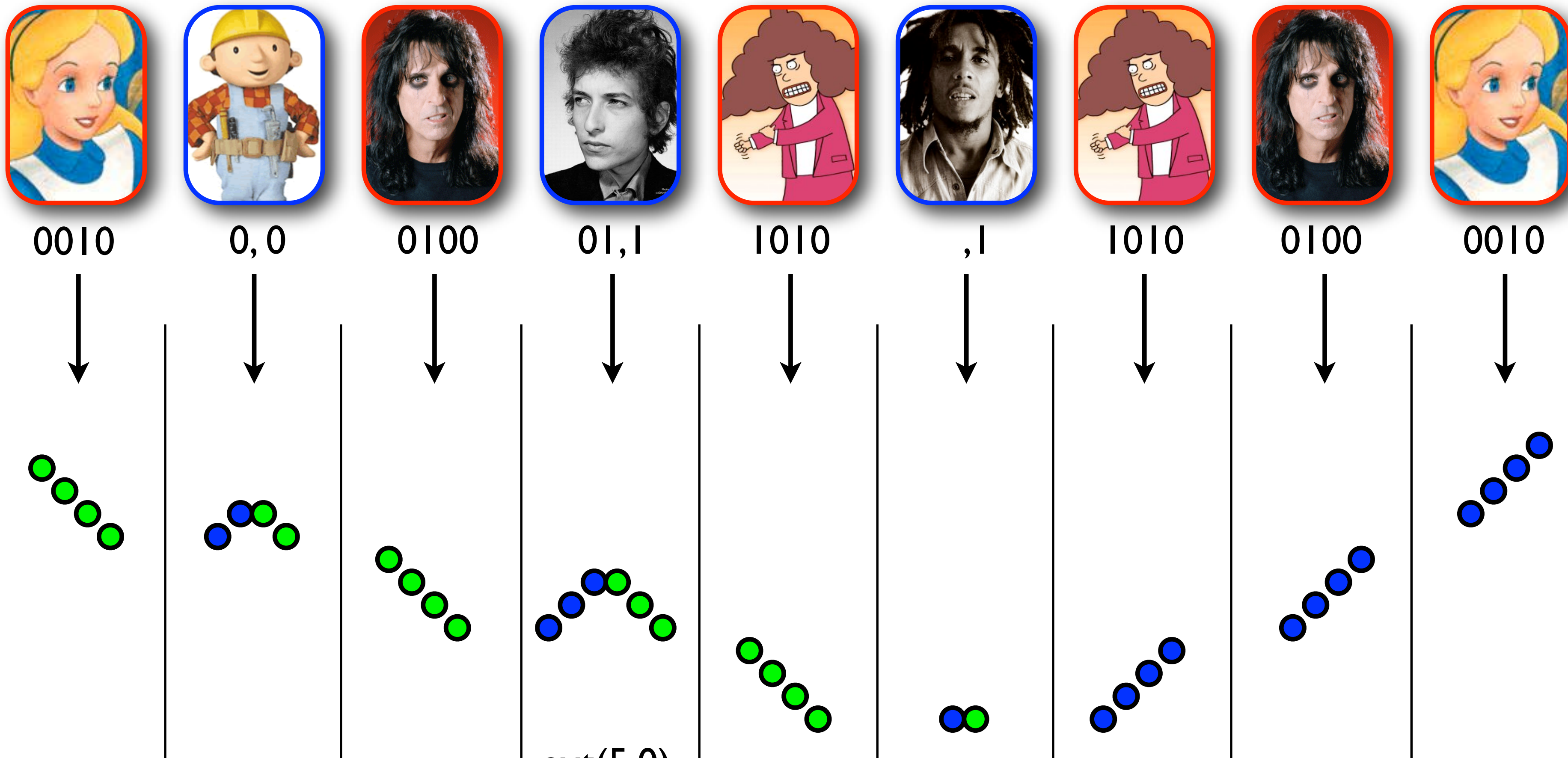
“Ascension Problem” [Magniez, Mathieu, Nayak '10]

Reduction from multi-player AI to PQ...



"Ascension Problem" [Magniez, Mathieu, Nayak '10]

Reduction from multi-player AI to PQ...



			ext(5.0)					
ins(12.0)	ext(9.0)	ins(8.0)	ext(6.1)	ins(4.0)		ext(1.1)	ext(5.0)	ext(9.0)
ins(11.1)	ext(10.0)	ins(7.0)	ext(7.1)	ins(3.1)	ext(1.1)	ext(2.0)	ext(6.1)	ext(10.0)
ins(10.0)	ins(10.0)	ins(6.1)	ins(7.1)	ins(2.0)	ins(1.1)	ext(3.1)	ext(7.0)	ext(11.1)
ins(9.0)	ins(9.0)	ins(5.0)	ins(6.1)	ins(1.1)		ext(4.0)	ext(8.0)	ext(12.0)
			ins(5.0)					

"Ascension Problem" [Magniez, Mathieu, Nayak '10]

Reduction from multi-player AI to PQ...

Reduction from multi-player AI to PQ...

- <u>*Thm:*</u> Any algorithm for recognizing PQ with probability at least 2/3 requires $\Omega(\sqrt{N})$ 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 $\mathfrak{A}$ 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 $\mathfrak{A}$ be a stream algorithm using s bits of space.

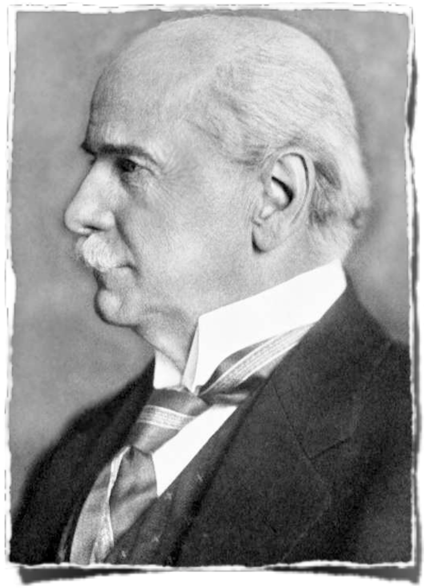
  ii. Use $\mathfrak{A}$ to construct a protocol with s bit messages: Players run $\mathfrak{A}$ on their input and send memory state to next player.

Reduction from multi-player AI to PQ...

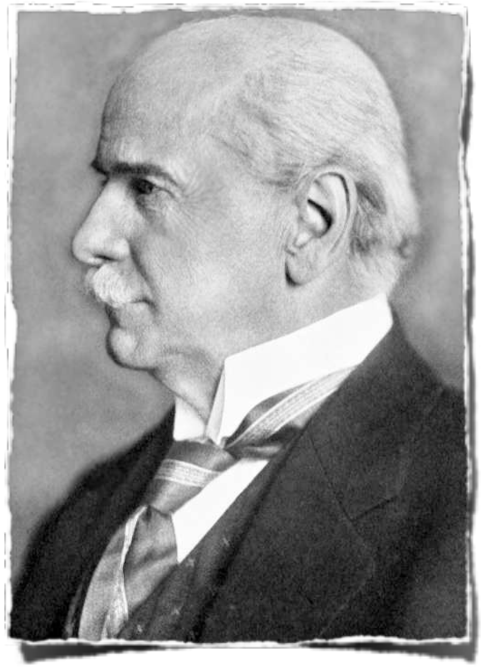
- *Thm:* Any algorithm for recognizing PQ with probability at least 2/3 requires $\Omega(\sqrt{N})$ space.
- *Proof:*
  1. Let $\mathfrak{A}$ be a stream algorithm using s bits of space.
  2. Use $\mathfrak{A}$ to construct a protocol with s bit messages: Players run $\mathfrak{A}$ on their input and send memory state to next player.
  3. Therefore, $s = \Omega(\min m,n)$ for length mn sequence.

I. Memory Checking

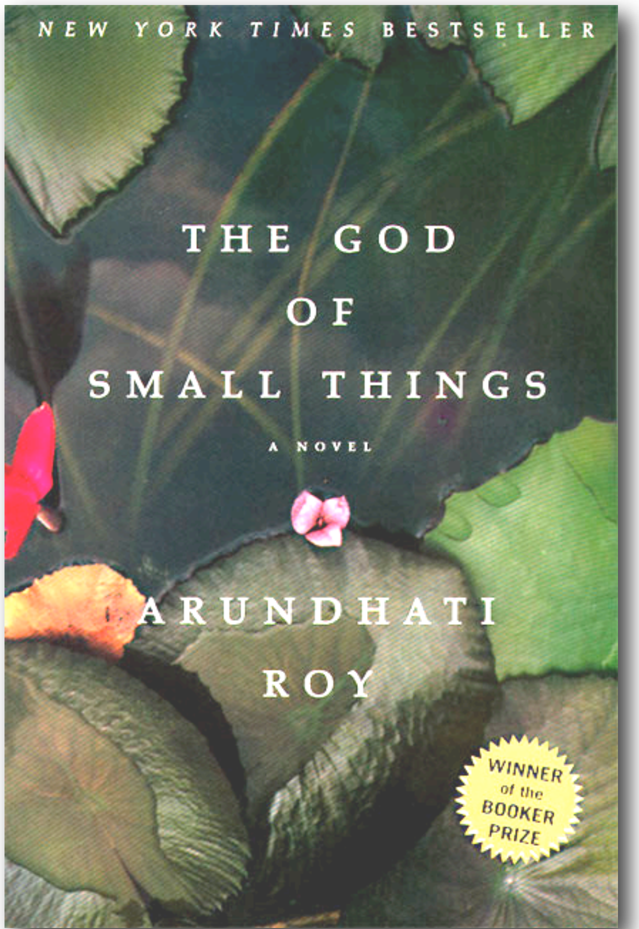
II. Lower Bounds

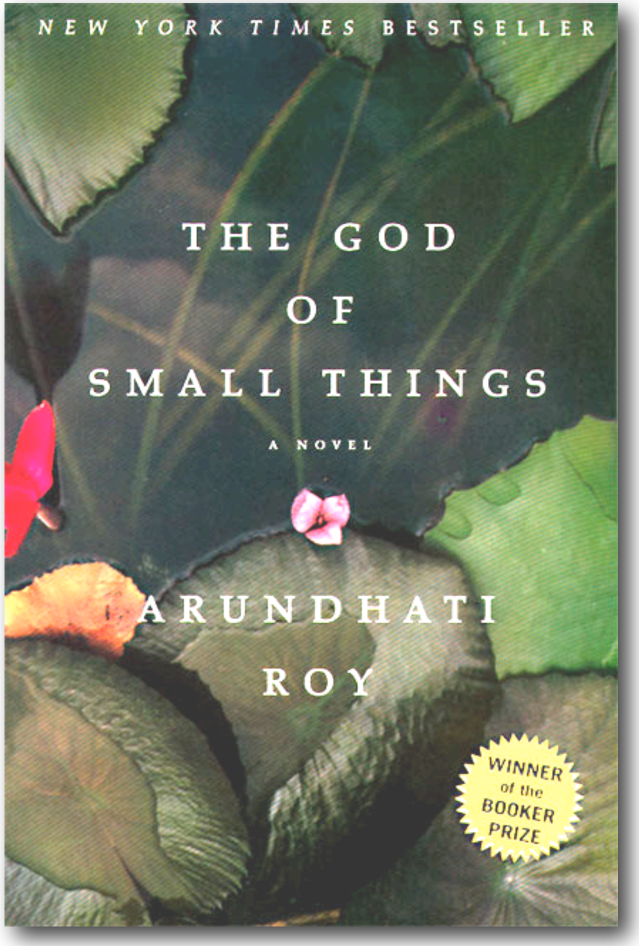


III. Parenthesis and Passes



III. Parenthesis and Passes


NEW YORK TIMES BESTSELLER
THE GOD OF SMALL THINGS
A NOVEL
ARUNDHATI ROY
WINNER of the BOOKER PRIZE



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

- DYCK_2 is the set of strings of properly nested brackets when there are two different types of brackets:

$$((([])()[])) \in \mathrm{DYCK}_2 \qquad ([([]])[])) \notin \mathrm{DYCK}_2$$

DYCK Language Problem

- DYCK_2 is the set of strings of properly nested brackets when there are two different types of brackets:

  $((([])()[])) \in \mathrm{DYCK}_2$ $([([]])[])) \notin \mathrm{DYCK}_2$

- DYCK *Problem:* Given *streaming* access to length N string, determine if it's in DYCK_2 using $o(N)$ space.

DYCK Language Problem

- DYCK_2 is the set of strings of properly nested brackets when there are two different types of brackets:

  $((([])()[])) \in \mathrm{DYCK}_2$ $([([]])[])) \notin \mathrm{DYCK}_2$

- *DYCK Problem:* Given *streaming* access to length N string, determine if it's in DYCK_2 using $o(N)$ space.

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

DYCK Language Problem

- $DYCK_2$ is the set of strings of properly nested brackets when there are two different types of brackets:

  $(((\,[\,]\,)(\,)[\,]\,)) \in DYCK_2$ $\qquad$ $([(\,[\,]\,])[\,]\,)) \notin DYCK_2$

- *DYCK Problem:* Given *streaming* access to length N string, determine if it's in $DYCK_2$ 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

- 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 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...

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 $\Theta(n^{1/p})$.

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 $\Theta(k^{1+\frac{1}{2^p-1}})$.

$\mathrm{DYCK}_2 \Rightarrow \mathrm{PQ}$

$DYCK_2 \Rightarrow PQ$

- There's a reduction from $DYCK_2$ to PQ and our bounds extend to multi-pass algorithms.

$DYCK_2 \Rightarrow PQ$

- 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 $\Omega(\sqrt{N}/p)$.

$\text{DYCK}_2 \Rightarrow \text{PQ}$

- 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!"

$DYCK_2 \Rightarrow PQ$

- 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?

Summary

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

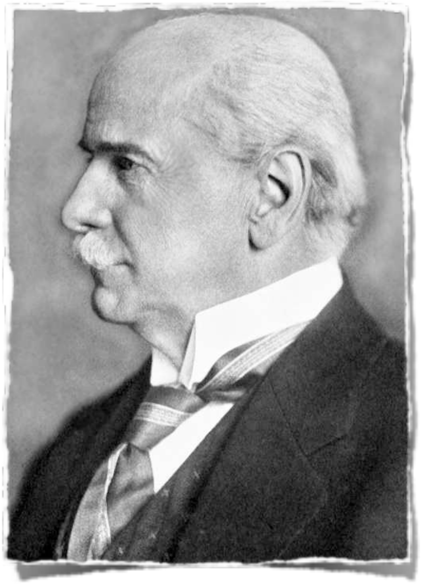
Theory of Stream Computation: Forward and backward pass better than many forward passes!

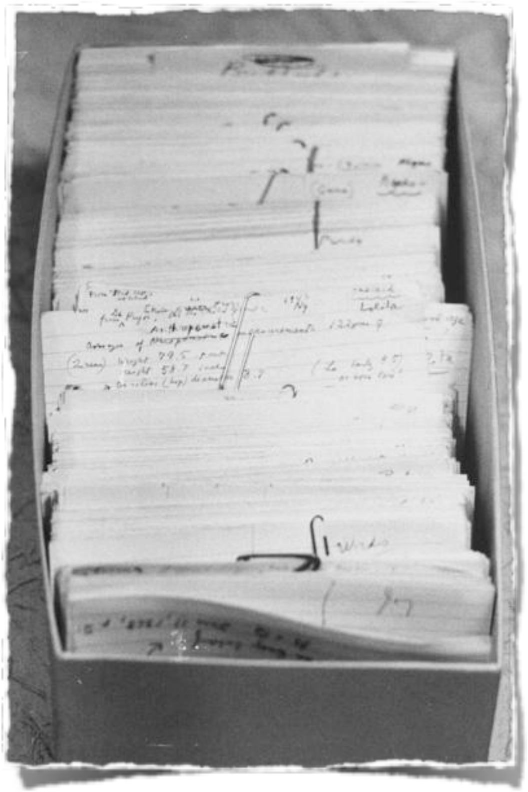
Further Work: Annotations, stream language recognition, ...

Thanks!

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:***

$$
\begin{aligned}
H(X) &= -\Sigma \Pr[X = x] \lg \Pr[X = x] \\
H(X|Y) &= -\Sigma \Pr[X = x, Y = y] \lg \Pr[X = x | Y = y]
\end{aligned}
$$

Information Complexity

[Chakrabarti, Shi, Wirth, Yao '01]

- *Entropy and Mutual Information:*

$$
\begin{aligned}
H(X) &= -\Sigma \Pr[X = x] \lg \Pr[X = x] \\
H(X|Y) &= -\Sigma \Pr[X = x, Y = y] \lg \Pr[X = x | Y = y] \\
I(X;Y) &= H(X) - H(X|Y) \;=\; H(Y) - H(Y|X) \\
I(X;Y|Z) &= H(X|Z) - H(X|Y,Z)
\end{aligned}
$$

Information Complexity

[Chakrabarti, Shi, Wirth, Yao '01]

- *Entropy and Mutual Information:*

$$\begin{aligned} H(X) &= -\Sigma \Pr[X = x] \lg \Pr[X = x] \\ H(X|Y) &= -\Sigma \Pr[X = x, Y = y] \lg \Pr[X = x | Y = y] \\ I(X;Y) &= H(X) - H(X|Y) \;=\; H(Y) - H(Y|X) \\ I(X;Y|Z) &= H(X|Z) - H(X|Y,Z) \end{aligned}$$

- *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:***

$$
\begin{aligned}
H(X) &= -\Sigma \Pr[X = x] \lg \Pr[X = x] \\
H(X|Y) &= -\Sigma \Pr[X = x, Y = y] \lg \Pr[X = x | Y = y] \\
I(X;Y) &= H(X) - H(X|Y) \; = \; H(Y) - H(Y|X) \\
I(X;Y|Z) &= H(X|Z) - H(X|Y,Z)
\end{aligned}
$$

- ***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:*

$$
\begin{aligned}
H(X) &= -\Sigma \Pr[X = x] \lg \Pr[X = x] \\
H(X|Y) &= -\Sigma \Pr[X = x, Y = y] \lg \Pr[X = x | Y = y] \\
I(X; Y) &= H(X) - H(X|Y) \;=\; H(Y) - H(Y|X) \\
I(X; Y|Z) &= H(X|Z) - H(X|Y, Z)
\end{aligned}
$$

- *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 AI_n

Information Complexity of AI_n

- ***Defn:*** Let P be a protocol for AI_n using public random string R. Let T be the transcript and (X, K, C)~ξ. Define

$$
\begin{aligned}
\mathrm{icost}_{\xi}^{A}(\mathrm{P}) &= I(T : X \mid K, C, R) \\
\mathrm{icost}_{\xi}^{B}(\mathrm{P}) &= I(T : K, C \mid X, R)
\end{aligned}
$$

Information Complexity of AI_n

- *Defn:* Let P be a protocol for AI_n using public random string R. Let T be the transcript and (X, K, C)~ξ. Define

$$\begin{aligned} \mathrm{icost}_{\xi}^{A}(P) &= I(T : X \mid K, C, R) \\ \mathrm{icost}_{\xi}^{B}(P) &= I(T : K, C \mid X, R) \end{aligned}$$

- *Thm:* Let P be a randomized protocol for AI_n with error 1/3 under the uniform distribution μ. Then,

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

where μ_0 is μ 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 $(X^i,K^i,C^i)_{i\in[m]}\sim\xi$.

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

where T_m is the set of messages sent by B_m.

MULTI-AI$_{m,n}$ versus AI$_n$

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

$$\mathrm{icost}_\xi(Q) = \mathrm{I}(\mathrm{T}_m : \mathrm{K}^1, \mathrm{C}^1, \ldots, \mathrm{K}^m, \mathrm{C}^m \mid \mathrm{X}^1, \ldots, \mathrm{X}^m, \mathrm{R})$$

where T_m is the set of messages sent by B_m.

- *Thm (Direct Sum):* If there exists a p-round, s-bit, ε-error protocol Q for MULTI-AI$_{m,n}$ then there exists a p-round, ε-error randomized protocol P for AI$_n$ where

i. Alice sends at most ps bits

ii. $m \cdot \mathrm{icost}^B_{\mu_0}(P) \leq \mathrm{icost}_{\mu_0^{\otimes m}}(Q)$

Putting it all together...

Putting it all together...

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

Putting it all together...

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

i. By direct sum theorem, there exists ε-error, p-pass protocol P for AI$_n$ such that:

$$
\begin{aligned}
p \cdot s &\geq \mathrm{icost}_{\mu_0^{\otimes m}}(Q) \geq m \cdot \mathrm{icost}^{B}_{\mu_0}(P) \\
p \cdot s &\geq \mathrm{icost}^{A}_{\mu_0}(P)
\end{aligned}
$$

Putting it all together...

- *Thm:* Any p-round, s-bit, 1/3-error protocol Q for MULTI-$\mathrm{AI}_{m,n}$ requires ps=Ω(min m,n).
- *Proof:*
  i. By direct sum theorem, there exists ε-error, p-pass protocol P for AI_n such that:

$$p \cdot s \geq \mathrm{icost}_{\mu_0^{\otimes m}}(Q) \geq m \cdot \mathrm{icost}^B_{\mu_0}(P)$$
$$p \cdot s \geq \mathrm{icost}^A_{\mu_0}(P)$$

  ii. By information complexity of AI_n

$$\max(m \cdot \mathrm{icost}^B_{\mu_0}(P), \mathrm{icost}^A_{\mu_0}(P)) = \Omega(\min(m, n))$$

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!