Out-of-Order Membership in Regular Languages

Antoine Amarilli
Inria Lille

Sebastien Felipe Labbe
ULille

Charles Paperman
ULille

ICALP 2026

The Membership Problem

Definition

Fix a language $L \subseteq \Sigma^*$. Given a word w, decide whether $w \in L$.

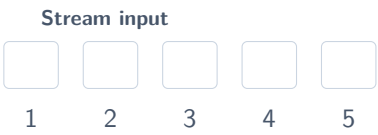
Example

$$\Sigma = \{a, b\} \qquad L = \Sigma^* aa \Sigma^*$$

$$aabab \in L \qquad ababa \notin L$$

Membership: Left-to-Right Streaming

Fixed: $\Sigma = \{a, b\} \quad L = \Sigma^* aa \Sigma^*$



Membership: Left-to-Right Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Membership: Left-to-Right Streaming

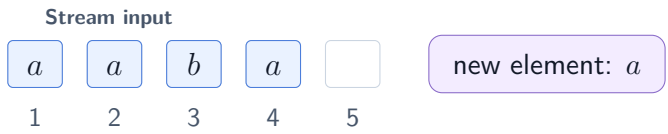
Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Membership: Left-to-Right Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

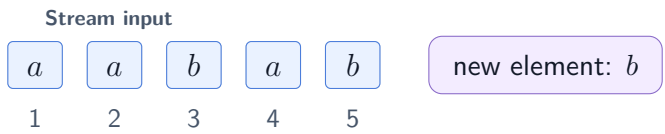
Membership: Left-to-Right Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$



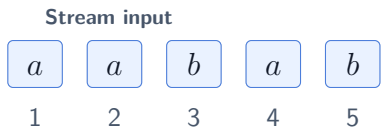
Membership: Left-to-Right Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$



Membership: Left-to-Right Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

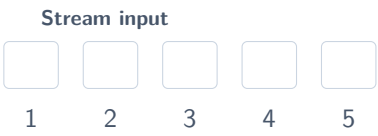


Decision: Is the word $aabab$ in L?

Membership: Out-of-Order Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



Membership: Out-of-Order Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

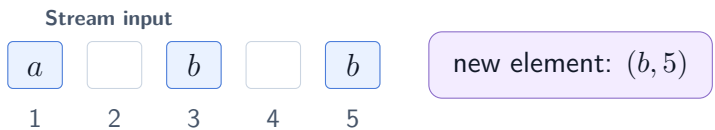
Setting: adversarial arrival order

Membership: Out-of-Order Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$ **Setting:** adversarial arrival order

Membership: Out-of-Order Streaming

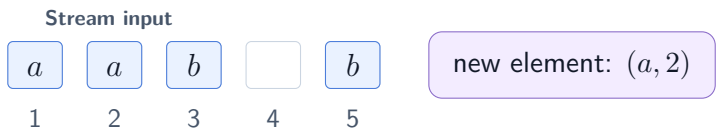
Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$ **Setting:** adversarial arrival order



Membership: Out-of-Order Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

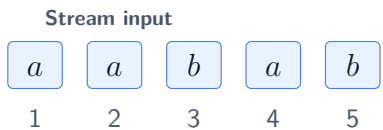
Setting: adversarial arrival order



Membership: Out-of-Order Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

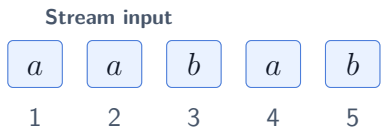
Setting: adversarial arrival order



Membership: Out-of-Order Streaming

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

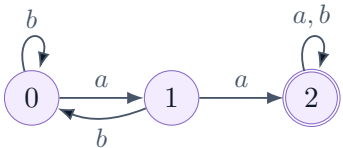
Setting: adversarial arrival order



Decision: Is the word $aabab$ in L?

Regular Languages

We focus on regular languages, which are recognized by deterministic finite automata.



$$L = \Sigma^* aa \Sigma^*$$

Each letter defines a transition function.

	a	b
0	1	0
1	2	0
2	2	2

letter a

$0 \mapsto 1$
$1 \mapsto 2$
$2 \mapsto 2$

letter b

$0 \mapsto 0$
$1 \mapsto 0$
$2 \mapsto 2$

Data complexity of Out-of-Order Membership in Regular Languages

- **Model:** out-of-order membership.
- **Languages:** regular languages.
- **Complexity:** data complexity.
  - **Fixed:** the language L is fixed.
  - **Input:** a word w of length n, with n known in advance.
  - We study **time per element** and **space** as functions of n.

General Upper Bound: Linear Space

Fixed: $\Sigma = \{a, b\}$ $L = \Sigma^* aa \Sigma^*$

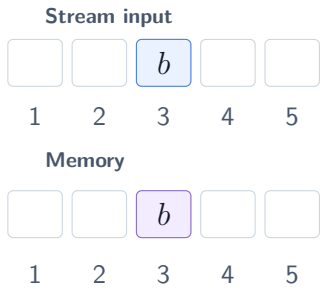
Setting: adversarial arrival order

Theorem: every fixed regular language has $O(n)$ time per element and $O(n)$ space.

General Upper Bound: Linear Space

Fixed: $\Sigma = \{a, b\}$ $L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



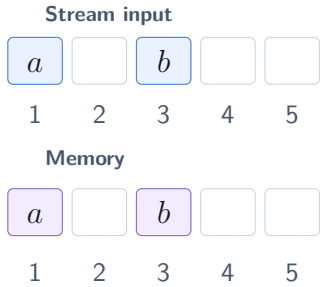
new element: $(b, 3)$

Theorem: every fixed regular language has $O(n)$ time per element and $O(n)$ space.

General Upper Bound: Linear Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



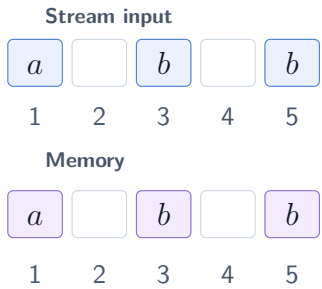
new element: $(a, 1)$

Theorem: every fixed regular language has $O(n)$ time per element and $O(n)$ space.

General Upper Bound: Linear Space

Fixed: $\Sigma = \{a, b\}$ $L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



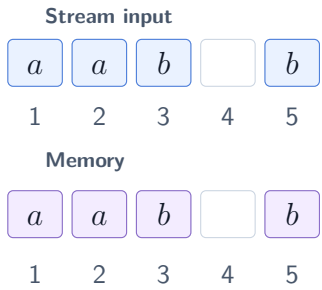
new element: $(b, 5)$

Theorem: every fixed regular language has $O(n)$ time per element and $O(n)$ space.

General Upper Bound: Linear Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



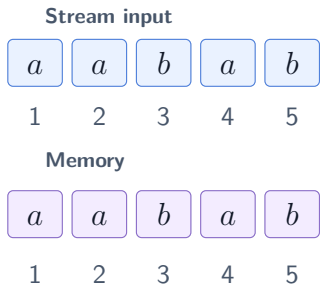
new element: $(a, 2)$

Theorem: every fixed regular language has $O(n)$ time per element and $O(n)$ space.

General Upper Bound: Linear Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



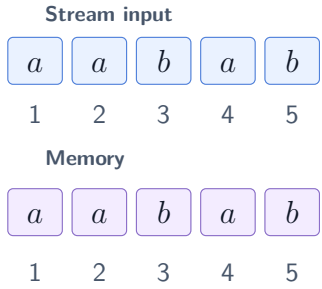
new element: $(a, 4)$

Theorem: every fixed regular language has $O(n)$ time per element and $O(n)$ space.

General Upper Bound: Linear Space

Fixed: $\Sigma = \{a, b\}$ $L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



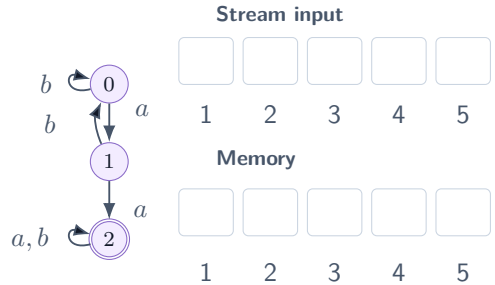
Decision: Run the automaton on the fully written down word $aabab$.

Theorem: every fixed regular language has $O(n)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



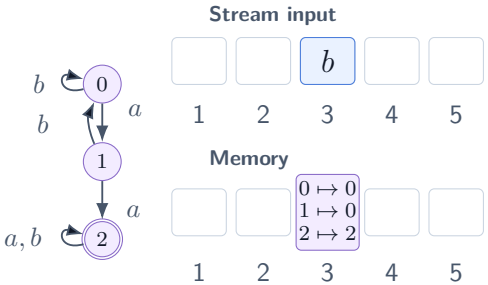
Store transition functions.

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



new element: $(b, 3)$

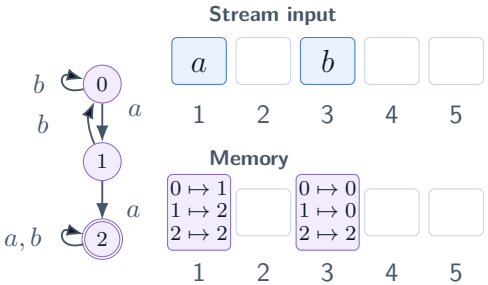
Store transition functions.

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order

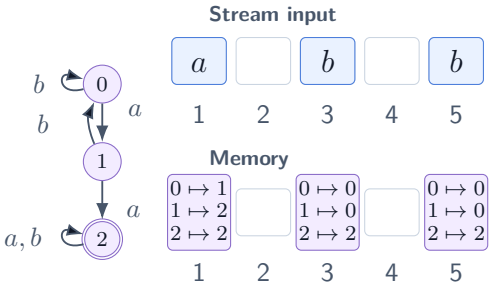


Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



new element: $(b, 5)$

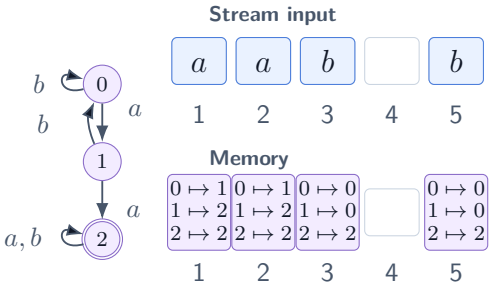
Store transition functions.

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



new element: $(a, 2)$

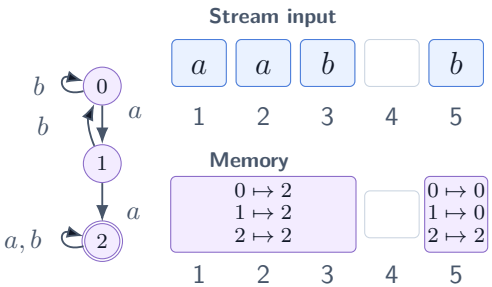
Store transition functions.

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



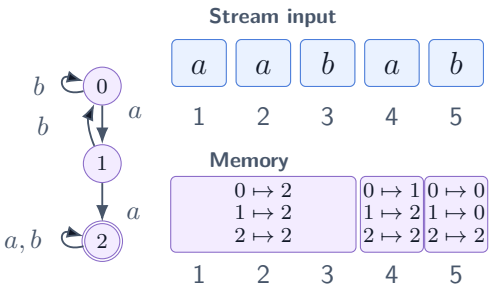
Compose adjacent functions.

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $L = \Sigma^* aa\Sigma^*$

Setting: adversarial arrival order



new element: $(a, 4)$

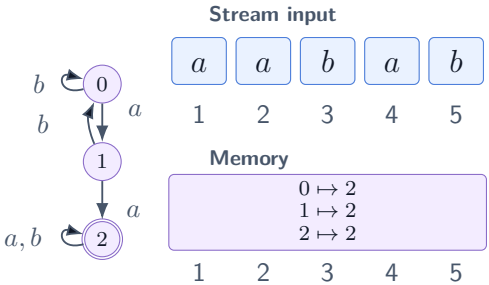
Store transition functions.

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



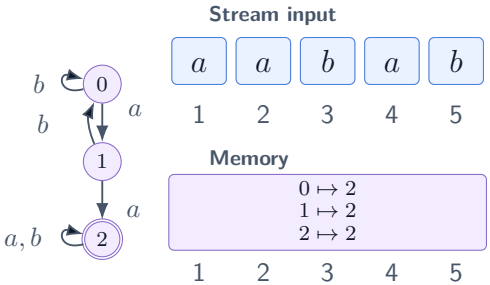
Final block summarizes word.

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

General Upper Bound: Constant Time per Element

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



Final block summarizes word.

Decision: accept iff initial state is mapped to final state

Theorem: every fixed regular language has $O(1)$ time per element and $O(n)$ space.

Space Regimes: Constant vs. Linear

The general upper bound gives $O(1)$ time per element and $O(n)$ bits of memory.

Can this space bound be improved for every language, or at least for some?

Example: $\Sigma^* aa \Sigma^*$ has $\Omega(n)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

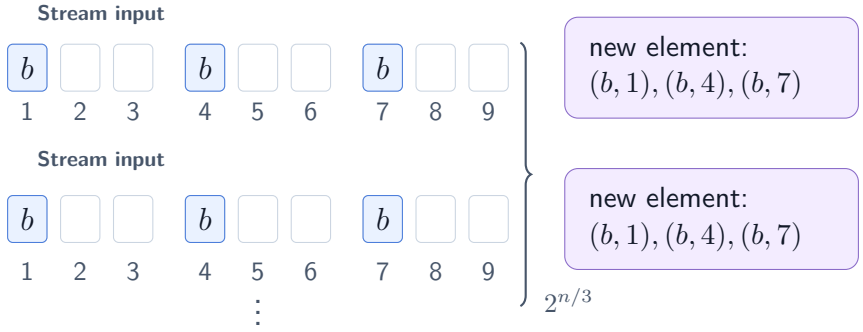
Setting: adversarial arrival order

Theorem: out-of-order membership for $\Sigma^* aa \Sigma^*$ requires $\Omega(n)$ space.

Example: $\Sigma^* aa \Sigma^*$ has $\Omega(n)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order

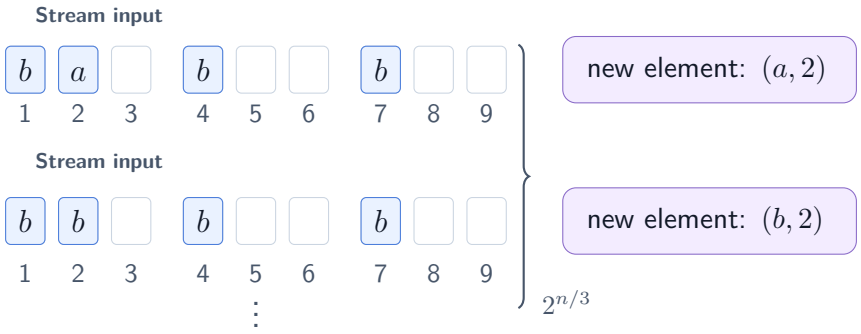


Theorem: out-of-order membership for $\Sigma^* aa \Sigma^*$ requires $\Omega(n)$ space.

Example: $\Sigma^* aa \Sigma^*$ has $\Omega(n)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order

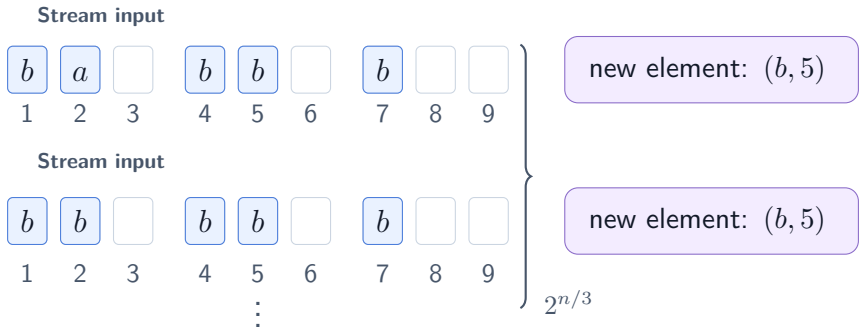


Theorem: out-of-order membership for $\Sigma^* aa \Sigma^*$ requires $\Omega(n)$ space.

Example: $\Sigma^* aa \Sigma^*$ has $\Omega(n)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order

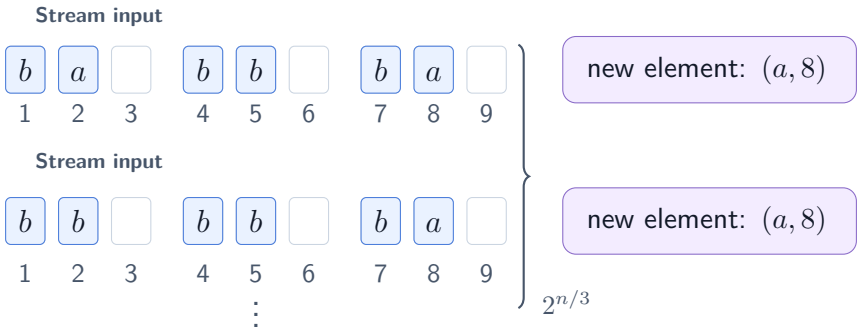


Theorem: out-of-order membership for $\Sigma^* aa \Sigma^*$ requires $\Omega(n)$ space.

Example: $\Sigma^* aa \Sigma^*$ has $\Omega(n)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order

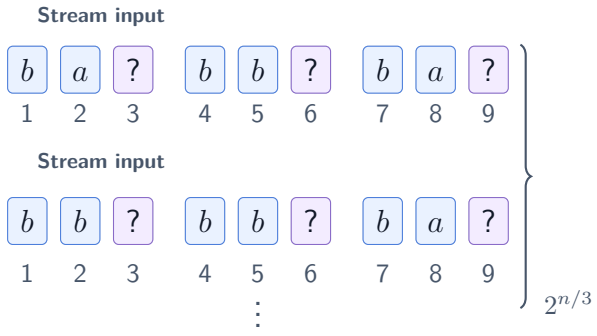


Theorem: out-of-order membership for $\Sigma^* aa \Sigma^*$ requires $\Omega(n)$ space.

Example: $\Sigma^* aa \Sigma^*$ has $\Omega(n)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order

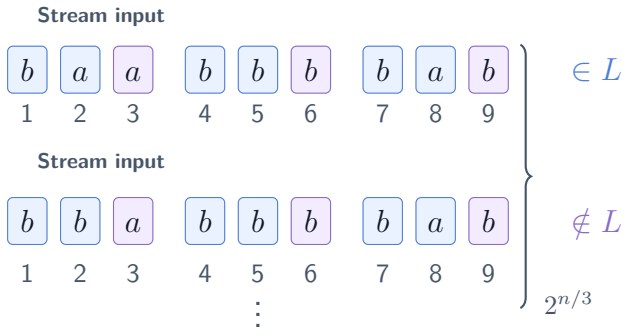


Theorem: out-of-order membership for $\Sigma^* aa \Sigma^*$ requires $\Omega(n)$ space.

Example: $\Sigma^* aa \Sigma^*$ has $\Omega(n)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = \Sigma^* aa \Sigma^*$

Setting: adversarial arrival order



Theorem: out-of-order membership for $\Sigma^* aa \Sigma^*$ requires $\Omega(n)$ space.

Space Regimes: Constant vs. Linear

The general upper bound gives $O(1)$ time per element and $O(n)$ bits of memory.

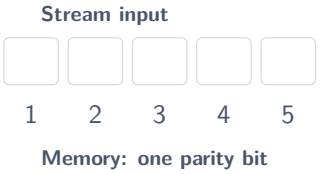
Can this space bound be improved for every language, or at least for some?

- For consecutive a's $(\Sigma^* aa \Sigma^*)$: no, there is an $\Omega(n)$ lower bound.

Example: $b^*(ab^*ab^*)^*$ is in $O(1)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order



On a: flip the bit.

Theorem: out-of-order membership for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Example: $b^*(ab^*ab^*)^*$ is in $O(1)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order

Stream input

new element: $(b, 3)$

Memory: one parity bit

even

On a: flip the bit.

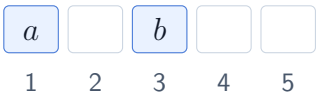
Theorem: out-of-order membership for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Example: $b^*(ab^*ab^*)^*$ is in $O(1)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order

Stream input



Memory: one parity bit

odd

new element: $(a, 1)$

On a: flip the bit.

Theorem: out-of-order membership for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Example: $b^*(ab^*ab^*)^*$ is in $O(1)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order

Stream input

new element: $(b, 5)$

Memory: one parity bit

odd

On a: flip the bit.

Theorem: out-of-order membership for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Example: $b^*(ab^*ab^*)^*$ is in $O(1)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order

Stream input

new element: $(a, 2)$

Memory: one parity bit

even

On a: flip the bit.

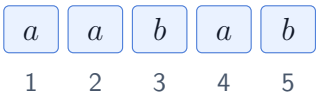
Theorem: out-of-order membership for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Example: $b^*(ab^*ab^*)^*$ is in $O(1)$ space

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order

Stream input



new element: $(a, 4)$

Memory: one parity bit

odd

On a: flip the bit.

Decision: accept iff memory state is even

Theorem: out-of-order membership for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Space Regimes: Constant vs. Linear

The general upper bound gives $O(1)$ time per element and $O(n)$ bits of memory.

Can this space bound be improved for every language, or at least for some?

- For consecutive a's $(\Sigma^* aa \Sigma^*)$: no, there is an $\Omega(n)$ lower bound.
- For parity $(b^*(ab^*ab^*)^*)$: yes, there is an $O(1)$ upper bound.

Main Question: Classify Space

Problem

Under **out-of-order membership**, which **regular languages** have which **space data complexity**?

At least two behaviours:

$$L = b^*(ab^*ab^*)^* \quad : \quad O(1) \text{ space}$$
$$L = \Sigma^* aa \Sigma^* \quad : \quad \Omega(n) \text{ bits of memory}$$

Which languages are constant? Which are linear? Are there other regimes?

Problem variants

Out-of-order membership variant	position content
Original problem	$\Sigma \quad (a, 1)$

Problem variants

Out-of-order membership variant	position content
Original problem	$\Sigma \quad (a, 1)$
Semigroup variant	$\Sigma^+ \quad (aba, 3)$
Monoid variant	$\Sigma^* \quad (\varepsilon, 4)$

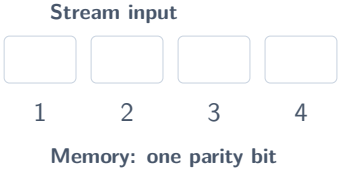
Problem variants

Out-of-order membership variant	position content
Original problem	$\Sigma \quad (a, 1)$
Semigroup variant	$\Sigma^+ \quad (aba, 3)$
Monoid variant	$\Sigma^* \quad (\varepsilon, 4)$

Monoid Variant: Example Stream

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^* position content



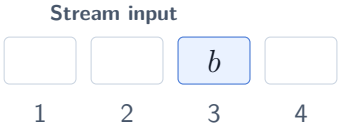
Flip if the new word has an odd number of a's.

Theorem: the monoid variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Monoid Variant: Example Stream

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^* position content



new element: $(b, 3)$

Flip if the new word has an odd number of a's.

Theorem: the monoid variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Monoid Variant: Example Stream

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^* position content

Stream input

aa		b	
1	2	3	4

Memory: one parity bit

even

new element: $(aa, 1)$

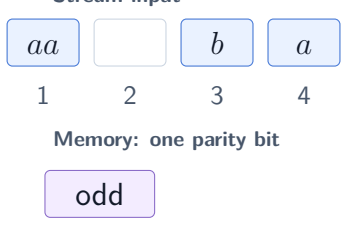
Flip if the new word has an odd number of a's.

Theorem: the monoid variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Monoid Variant: Example Stream

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^* position content



new element: $(a, 4)$

Flip if the new word has an odd number of a's.

Theorem: the monoid variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Monoid Variant: Example Stream

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^* position content

Stream input

aa	ε	b	a
1	2	3	4

Memory: one parity bit

odd

new element: $(\varepsilon, 2)$

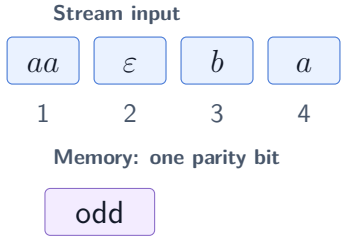
Flip if the new word has an odd number of a's.

Theorem: the monoid variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Monoid Variant: Example Stream

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^* position content



Flip if the new word has an odd number of a's.

Decision: accept iff final memory state is even

Theorem: the monoid variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Monoid Variant: $O(1)$ Space

Definition

$L \subseteq \Sigma^*$ is in $\mathbf{Com}$ if adjacent letters can always be swapped without changing membership:

$$\forall u, v \in \Sigma^*,\ \forall a, b \in \Sigma, \quad uabv \in L \iff ubav \in L.$$

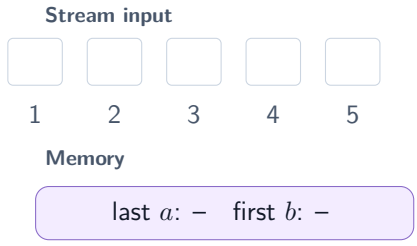
Theorem

A regular language has constant space data complexity for out-of-order membership in the monoid variant if and only if it is in $\mathbf{Com}$.

Monoid Variant: a^*b^* is in $O(\log n)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a^*b^*$

Setting: adversarial arrival order, Σ^* position content

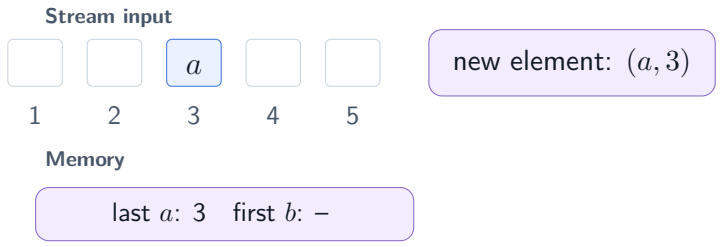


Theorem: the monoid variant for a^*b^* is in $O(\log n)$ space.

Monoid Variant: a^*b^* is in $O(\log n)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a^*b^*$

Setting: adversarial arrival order, Σ^* position content

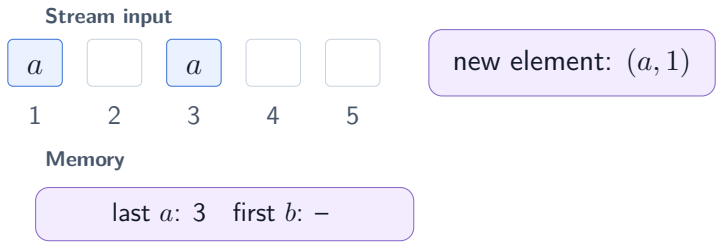


Theorem: the monoid variant for a^*b^* is in $O(\log n)$ space.

Monoid Variant: a^*b^* is in $O(\log n)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a^*b^*$

Setting: adversarial arrival order, Σ^* position content

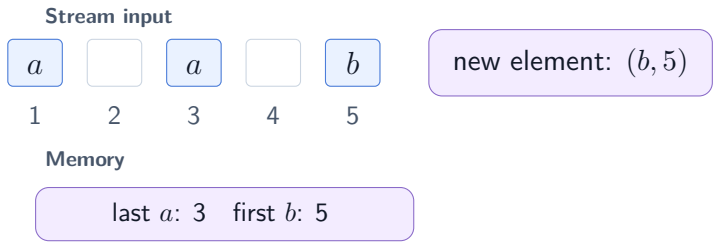


Theorem: the monoid variant for a^*b^* is in $O(\log n)$ space.

Monoid Variant: a^*b^* is in $O(\log n)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a^*b^*$

Setting: adversarial arrival order, Σ^* position content

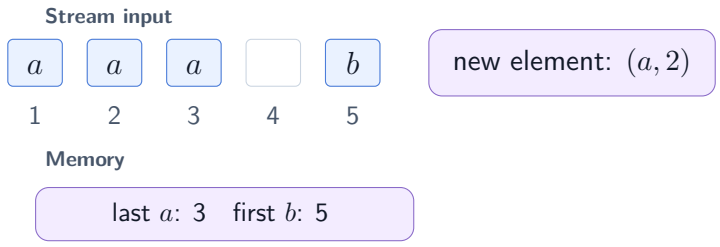


Theorem: the monoid variant for a^*b^* is in $O(\log n)$ space.

Monoid Variant: a^*b^* is in $O(\log n)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a^*b^*$

Setting: adversarial arrival order, Σ^* position content

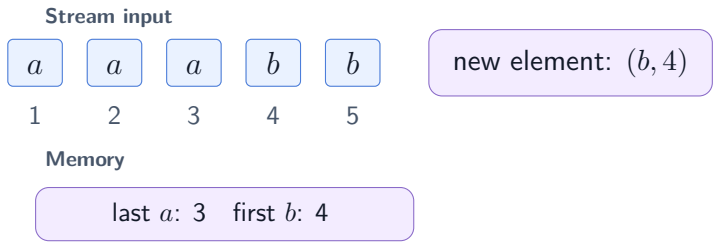


Theorem: the monoid variant for a^*b^* is in $O(\log n)$ space.

Monoid Variant: a^*b^* is in $O(\log n)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a^*b^*$

Setting: adversarial arrival order, Σ^* position content

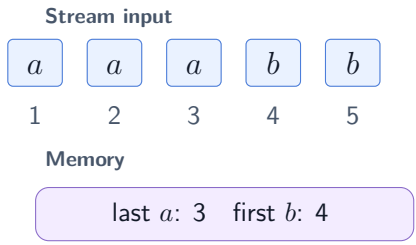


Theorem: the monoid variant for a^*b^* is in $O(\log n)$ space.

Monoid Variant: a^*b^* is in $O(\log n)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a^*b^*$

Setting: adversarial arrival order, Σ^* position content



Decision: accept iff last $a <$ first b

Theorem: the monoid variant for a^*b^* is in $O(\log n)$ space.

Monoid Variant: $O(\log n)$ Space

Definition

$L \subseteq \Sigma^*$ is in $\mathbf{FL}$ if there exists a constant k such that membership is unchanged when we keep only the first k and last k positions of each letter:

$$w \in L \iff \mathrm{FL}_k(w) \in L.$$

Theorem

A regular language has logarithmic space data complexity for out-of-order membership in the monoid variant if and only if $L \in \mathbf{FL} \vee \mathbf{Com}$.

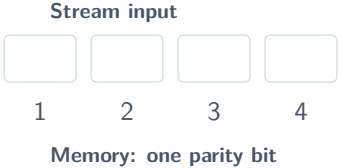
Problem variants

Out-of-order membership variant	position content
Original problem	$\Sigma \quad (a, 1)$
Semigroup variant	$\Sigma^+ \quad (aba, 3)$
Monoid variant	$\Sigma^* \quad (\varepsilon, 4)$

Semigroup Variant: Nonempty Words

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^+ position content



Flip if the new word has an odd number of a's.

Theorem: the semigroup variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Semigroup Variant: Nonempty Words

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^+ position content

Stream input

new element: $(b, 3)$

Memory: one parity bit

even

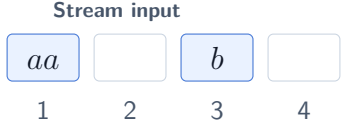
Flip if the new word has an odd number of a's.

Theorem: the semigroup variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Semigroup Variant: Nonempty Words

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^+ position content



new element: $(aa, 1)$

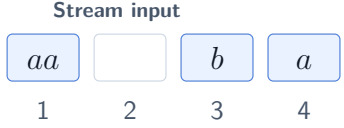
Flip if the new word has an odd number of a's.

Theorem: the semigroup variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Semigroup Variant: Nonempty Words

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^+ position content



new element: $(a, 4)$

Flip if the new word has an odd number of a's.

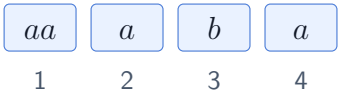
Theorem: the semigroup variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Semigroup Variant: Nonempty Words

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^+ position content

Stream input



new element: $(a, 2)$

Memory: one parity bit

even

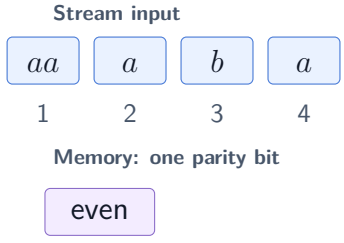
Flip if the new word has an odd number of a's.

Theorem: the semigroup variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Semigroup Variant: Nonempty Words

Fixed: $\Sigma = \{a, b\}$ $\quad L = b^*(ab^*ab^*)^*$

Setting: adversarial arrival order, Σ^+ position content



Flip if the new word has an odd number of a's.

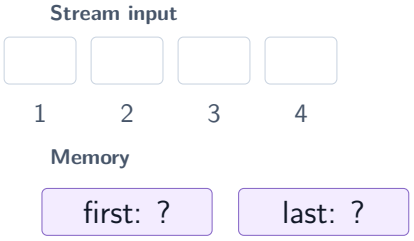
Decision: accept iff final memory state is even

Theorem: the semigroup variant for $b^*(ab^*ab^*)^*$ is in $O(1)$ space.

Semigroup Variant: $a\Sigma^*b$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a\Sigma^*b$

Setting: adversarial arrival order, Σ^+ position content

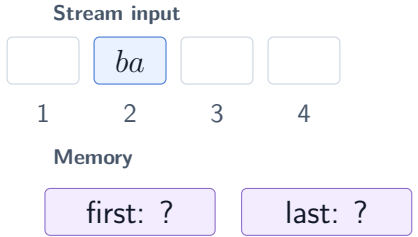


Theorem: the semigroup variant for $a\Sigma^*b$ is in $O(1)$ space.

Semigroup Variant: $a\Sigma^*b$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a\Sigma^*b$

Setting: adversarial arrival order, Σ^+ position content

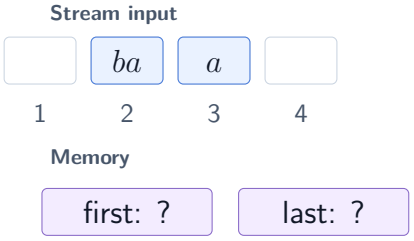


Theorem: the semigroup variant for $a\Sigma^*b$ is in $O(1)$ space.

Semigroup Variant: $a\Sigma^*b$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a\Sigma^*b$

Setting: adversarial arrival order, Σ^+ position content

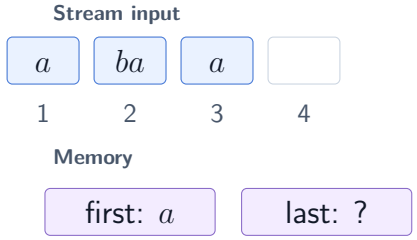


Theorem: the semigroup variant for $a\Sigma^*b$ is in $O(1)$ space.

Semigroup Variant: $a\Sigma^*b$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a\Sigma^*b$

Setting: adversarial arrival order, Σ^+ position content

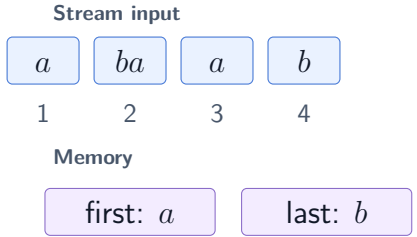


Theorem: the semigroup variant for $a\Sigma^*b$ is in $O(1)$ space.

Semigroup Variant: $a\Sigma^*b$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a\Sigma^*b$

Setting: adversarial arrival order, Σ^+ position content

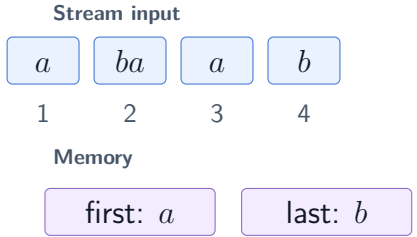


Theorem: the semigroup variant for $a\Sigma^*b$ is in $O(1)$ space.

Semigroup Variant: $a\Sigma^*b$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = a\Sigma^*b$

Setting: adversarial arrival order, Σ^+ position content



Decision: Accept iff first starts with a and last ends with b.

Theorem: the semigroup variant for $a\Sigma^*b$ is in $O(1)$ space.

Semigroup Variant: $O(1)$ Space

Definition

$L \subseteq \Sigma^+$ is in $\mathbf{LI}$ if there exists a constant k such that membership only depends on the first and last k letters:

$$\forall p, s \in \Sigma^k,\ \forall x \in \Sigma^*, \quad pxs \in L \Leftrightarrow ps \in L.$$

Theorem

A regular language has constant space data complexity for out-of-order membership in the semigroup variant if and only if it is in $\mathbf{LI} \vee \mathbf{Com}$.

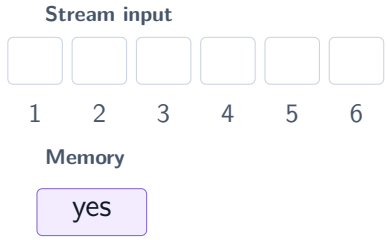
Problem variants

Out-of-order membership variant	position content
Original problem	$\Sigma \quad (a, 1)$
Semigroup variant	$\Sigma^+ \quad (aba, 3)$
Monoid variant	$\Sigma^* \quad (\varepsilon, 4)$

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $L = (ab)^*$

Setting: adversarial arrival order, Σ position content



Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $L = (ab)^*$

Setting: adversarial arrival order, Σ position content

Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $L = (ab)^*$

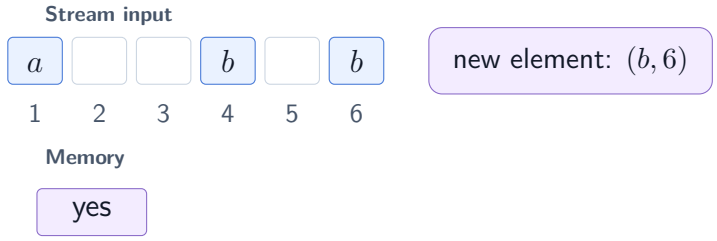
Setting: adversarial arrival order, Σ position content

Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $L = (ab)^*$

Setting: adversarial arrival order, Σ position content

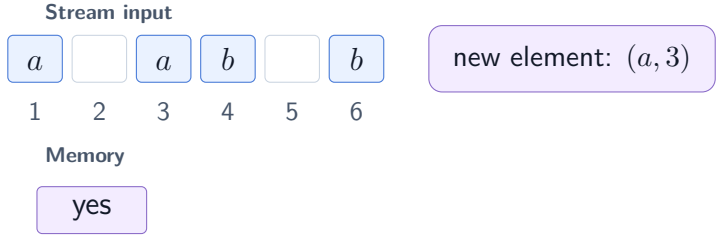


Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $L = (ab)^*$

Setting: adversarial arrival order, Σ position content

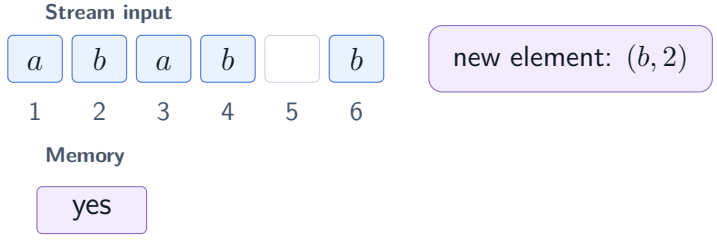


Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $L = (ab)^*$

Setting: adversarial arrival order, Σ position content

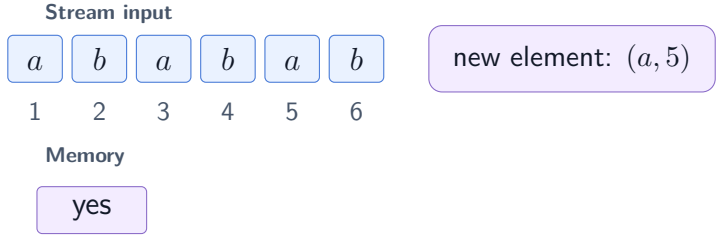


Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $L = (ab)^*$

Setting: adversarial arrival order, Σ position content

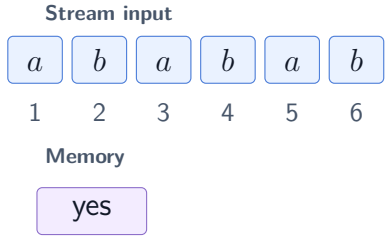


Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Original Problem: $(ab)^*$ is in $O(1)$ Space

Fixed: $\Sigma = \{a, b\}$ $\quad L = (ab)^*$

Setting: adversarial arrival order, Σ position content



Decision: Accept iff memory still says yes.

Theorem: out-of-order membership for $(ab)^*$ is in $O(1)$ space.

Complexity Landscape

Space	Monoid	Semigroup	Language
$O(1)$	$\mathbf{Com}$ $b^*(ab^*ab^*)^*$	$\mathbf{LI} \vee \mathbf{Com}$ $a\Sigma^*b$	$(\mathbf{LI} \vee \mathbf{Com}) * \mathbf{Mod}$?
$O(\log n)$	$\mathbf{FL} \vee \mathbf{Com}$ a^*b^*	$\mathbf{MA} \vee \mathbf{LI} \vee \mathbf{Com}$? $a^*b^*a^*$	$(\mathbf{MA} \vee \mathbf{LI} \vee \mathbf{Com}) * \mathbf{Mod}$?
$O(\sqrt{n})$	–	$\mathbf{FLA} \vee \mathbf{Com}$? $\Sigma^*a\Sigma^*b\Sigma^*c\Sigma^*$	$(\mathbf{FLA} \vee \mathbf{Com}) * \mathbf{Mod}$?
$O(n)$	$\mathbf{all}$ $\Sigma^*aa\Sigma^*$	$\mathbf{all}$ $(ab)^*$	$\mathbf{all}$

Open Problem: $\Sigma^* a \Sigma^* b \Sigma^* c \Sigma^*$ - abc subword

Fixed: $\Sigma = \{a, b, c\}$ $\quad L = \Sigma^* a \Sigma^* b \Sigma^* c \Sigma^*$

Setting: adversarial arrival order, Σ^+ position content

Known bounds

- $\Omega(\log n)$ lower bound
- $O(\sqrt{n})$ upper bound (not published)