# Internal Pattern Matching in Small Space and Applications

**Gabriel Bathie**
with Panagiotis Charalampopoulos and Tatiana Starikovskaya

June 25, 2024

## Circular Pattern Matching (CPM)

**Goal:** find in $T$ all occurrences of **rotations** of $P$.

$$\mathsf{P} = \mathsf{abcde} \rightarrow \begin{cases} \mathsf{rot}^1(P) = \mathsf{bcdea} \\ \mathsf{rot}^2(P) = \mathsf{cdeab} \end{cases} \qquad \mathsf{T} = \mathsf{becdeabaefgbcdeac}$$

$\rightarrow$ Algorithms for CPM?

**Related problem:** Long(est) Common Substring (LCS)

## Reduction to LCS

Occurrences of rotations of $P$ in $T$ are **exactly** the common substrings of length $m = |P|$ of $P^2$ and $T$.

$$\mathsf{P} \cdot \mathsf{P} = \mathsf{abcdeabcde} \quad \mathsf{T} = \mathsf{becdeabaefg} \ldots$$

# About Longest Common Substring (LCS)

- LCS: can be solved in $O(n)$ time and space using suffix trees [Wei73],
- [KSV14]: read-only algorithm with space $O(s)$ and time $O(n^2/s)$,
- $\Omega(n)$ space lower bound in streaming,
- [MRRS21]: semi-streaming algorithm with space $O(1)$ and time $O(n^2)$,
  - **Semi-streaming:** Read-only access to $P$, streaming access to $T$.
- No known algorithm with $T \cdot S = n^{2-o(1)}$.

**Q°:** Can we extend the trade-off of [KSV14] to semi-streaming?

## Our Results

Semi-streaming algorithm for LCS / CPM
with space $\tilde{O}(s)$ and time $\tilde{O}(n^2/s)$ for $\sqrt{n} \leq s \leq n$.

Here, $n = |P|$ and $|T| = O(n)$.

**Why $s \geq \sqrt{n}$?** "long" vs "short" common substrings.
Two algorithms in time $\tilde{O}(n^2/s)$:

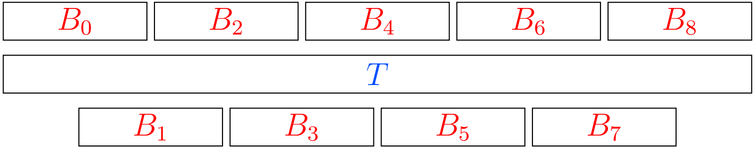
- space $O(s)$ for length $\leq s$
- space $O(n/s)$ for length $\geq s$,

$\Rightarrow s \geq \sqrt{n}$ to have $T \cdot S = O(n^2)$.

# Short common substrings

Find common substrings of length $\ell \leq s$ in $O(s)$ space and $O(n^2/s)$ time:

- Cover $T$ with blocks of length $2s$, overlapping by $s-1$ letters,




- for each block $B$, build its suffix tree: $O(s)$ space,
- Run $P$ through the suffix tree to find LCS: $O(n)$ time,
- there are $O(n/s)$ such blocks: $O(n^2/s)$ time in total.

Analysis: each substring of length $\ell \leq s$ is contained in exactly one block.

# Long common substrings: Internal Pattern Matching

## Internal Pattern Matching (IPM)

Given $i, j$ and a letter $a$, return an occurrence of $T[i..j] \cdot a$ in $T$, if any.

$\rightarrow$ Data structure problem

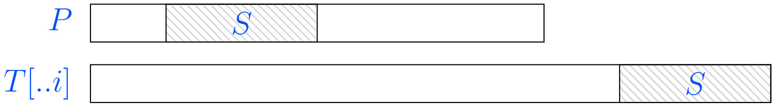
## Main Result

Data structure for IPM using $O(n/s)$ space and $\tilde{O}(1)$ time per query,
**restricted to queries with** $j - i \geq s$.

- Uses $s$-partioning sets of Kosolobov and Sivukhin [KS24] to find $O(n/s)$ "good" positions,
- Build forward and reverse sparse suffix tree for these positions,
- Reduce queries to 2D-range emptiness.

# From IPM to LCS in semi-streaming

**Algorithm**: Maintain the longest suffix $S$ of the current window $T[..i]$ that is a substring of $P$.



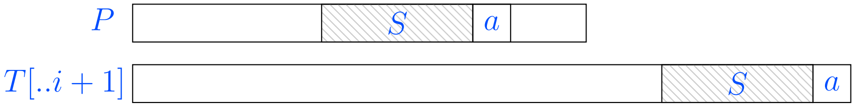

When receiving $a = T[i+1]$, use IPM queries to find if $S \cdot a$ occurs in $P$.

- If yes, we have a new longest suffix.
- Otherwise, use binary search to find the longest suffix $S'$ of $S$ that occurs in $P$.

Only works for $|S| \geq s$.
$\rightarrow$ start when the other algorithm find a common substring of length $s$

# From IPM to LCS in semi-streaming

**Algorithm**: Maintain the longest suffix $S$ of the current window $T[..i]$ that is a substring of $P$.



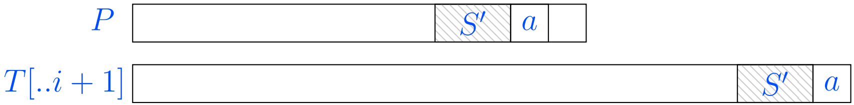

When receiving $a = T[i+1]$, use IPM queries to find if $S \cdot a$ occurs in $P$.

- If yes, we have a new longest suffix.
- Otherwise, use binary search to find the longest suffix $S'$ of $S$ that occurs in $P$.

Only works for $|S| \geq s$.
$\rightarrow$ start when the other algorithm find a common substring of length $s$

# From IPM to LCS in semi-streaming

**Algorithm**: Maintain the longest suffix $S$ of the current window $T[..i]$ that is a substring of $P$.




When receiving $a = T[i+1]$, use IPM queries to find if $S \cdot a$ occurs in $P$.

- If yes, we have a new longest suffix.
- Otherwise, use binary search to find the longest suffix $S'$ of $S$ that occurs in $P$.

Only works for $|S| \geq s$.
$\rightarrow$ start when the other algorithm find a common substring of length $s$

# Summary

- Circular pattern matching: closely related to LCS.
- LCS: Solved using Internal Pattern Matching.

**Main Result**

Data structure for $\geq s$-IPM using $O(n/s)$ space and $\tilde{O}(1)$ time per query.

**Applications**

LCS and CPM in semi-streaming using $O(s)$ space and $\tilde{O}(n^2/s)$ time for $s \geq \sqrt{n}$.

**Open problem**: No known reduction from LCS to CPM.
Can we solve CPM faster than $T \cdot S = n^2$?

# References I


Dmitry Kosolobov and Nikita Sivukhin.
Construction of sparse suffix trees and LCE indexes in optimal time and space.
In *Proc. of CPM*, 2024.

Tomasz Kociumaka, Tatiana Starikovskaya, and Hjalte Wedel Vildhøj.
Sublinear space algorithms for the longest common substring problem.
In *Proc. of ESA*, pages 605–617, 2014.

Tung Mai, Anup Rao, Ryan A Rossi, and Saeed Seddighin.
Optimal space and time for streaming pattern matching.
*arXiv preprint arXiv:2107.04660*, 2021.

# References II

Peter Weiner.
Linear pattern matching algorithms.
In *14th SWAT, 1973*, pages 1–11. IEEE, 1973.