Sliding a Suffix Array Along a Text

Anisa Al-Hafeedh[1], Laurent Mouchard[2], Mikaël Salson[2], Bill Smyth[3]

[1]University of Oman

[2]LITIS, University of Rouen, France

[3]McMaster University, Hamilton, Ontario, Canada

5 February 2010

litis

Introduction

Suffix Array

- Text index introduced in 1990.
- Lexicographically sorted suffixes.
- Matching a pattern of length m in a text T of length n in $O(m \log n)$ worst-case time.

Introduction

Suffix Array

- Text index introduced in 1990.
- Lexicographically sorted suffixes.
- Matching a pattern of length m in a text T of length n in $O(m \log n)$ worst-case time.

Example

$$T = \overset{0}{\mathrm{C}}\ \overset{1}{\mathrm{T}}\ \overset{2}{\mathrm{C}}\ \overset{3}{\mathrm{T}}\ \overset{4}{\mathrm{A}}\ \overset{5}{\mathrm{C}}\ \overset{6}{\mathrm{A}}\ \overset{7}{\mathrm{C}}\ \overset{8}{\mathrm{T}}\ \overset{9}{\mathrm{A}}\ \overset{10}{\$}$$

$$\mathrm{SA}(T) = 10\ 9\ 4\ 6\ 5\ 7\ 2\ 0\ 8\ 3\ 1$$

Problem

Question

Consider a sliding window, and a suffix array computed on it.
How to update the suffix array when the window slides?

litis

Problem

Question

Consider a sliding window, and a suffix array computed on it.
How to update the suffix array when the window slides?

Solution

M. Salson, T. Lecroq, M. Léonard, and L. Mouchard.
Dynamic extended suffix array.
Journal of Discrete Algorithms, 2009.

litis

Problem

Question

Consider a sliding window, and a suffix array computed on it.
How to update the suffix array when the window slides?

Solution

M. Salson, T. Lecroq, M. Léonard, and L. Mouchard.
Dynamic extended suffix array.
Journal of Discrete Algorithms, 2009.

Problem

No constant-time access to a random position in the suffix array.

UNIVERSITÉ DE ROUEN

litis

Problem

Question

Consider a sliding window, and a suffix array computed on it.
How to update the suffix array when the window slides?

Solution

M. Salson, T. Lecroq, M. Léonard, and L. Mouchard.
Dynamic extended suffix array.
Journal of Discrete Algorithms, 2009.

Problem

No constant-time access to a random position in the suffix array.

Ideas

- Suffix array should be stored using an array (not using a tree).

UNIVERSITÉ DE ROUEN

litis

Problem

Question

Consider a sliding window, and a suffix array computed on it.
How to update the suffix array when the window slides?

Solution

M. Salson, T. Lecroq, M. Léonard, and L. Mouchard.
Dynamic extended suffix array.
Journal of Discrete Algorithms, 2009.

Problem

No constant-time access to a random position in the suffix array.

Ideas

- Suffix array should be stored using an array (not using a tree).
- Avoid decrements by computing a *displaced* suffix array.

UNIVERSITÉ DE ROUEN

litis

Problem

Question

Consider a sliding window, and a suffix array computed on it.
How to update the suffix array when the window slides?

Solution

M. Salson, T. Lecroq, M. Léonard, and L. Mouchard.
Dynamic extended suffix array.
Journal of Discrete Algorithms, 2009.

Problem

No constant-time access to a random position in the suffix array.

Ideas

- Suffix array should be stored using an array (not using a tree).
- Avoid decrements by computing a *displaced* suffix array.

Displaced suffix array

Values range from k to $k + n - 1$ instead of 1 to n for the original suffix array.

- k: starting position of the window in the text;
- n: length of the window.

Algorithm

Big picture

- Store a list *Disp* of modifications, in position order, that have to be computed for updating the displaced suffix array.
- Update the displaced suffix array accordingly.

Algorithm

Big picture

- Store a list *Disp* of modifications, in position order, that have to be computed for updating the displaced suffix array.
- Update the displaced suffix array accordingly.

What is stored in the list?

Three types of modification:

- insertion of value j at position i, denoted by (i, j);
- deletion of the value at position i, denoted by $(i, 0)$;
- substitution of the value at position i by j, denoted by $(i, -j)$.

Algorithm

Remark

When sliding a window by s characters, we delete a prefix of length s and insert a suffix of the same length.

Algorithm

Remark

When sliding a window by s characters, we delete a prefix of length s and insert a suffix of the same length.

Deletion of the prefix

Suffixes starting at positions $[k, \ldots, k+s-1]$ should not exist anymore in SA. Binary search over the suffix array SA to identify positions of these suffixes in SA. Add the corresponding entries to *Disp*.

Algorithm

Remark

When sliding a window by s characters, we delete a prefix of length s and insert a suffix of the same length.

Deletion of the prefix

Suffixes starting at positions $[k, \ldots, k+s-1]$ should not exist anymore in SA. Binary search over the suffix array SA to identify positions of these suffixes in SA. Add the corresponding entries to *Disp*.

Insertion of the suffix

Suffixes starting at positions $[k+n, \ldots, k+n+s-1]$ do not exist yet in SA. Binary search over SA to find their positions of insertion. Add the corresponding entries to *Disp*.

litis

Algorithm

Remark

When sliding a window by s characters, we delete a prefix of length s and insert a suffix of the same length.

Deletion of the prefix

Suffixes starting at positions $[k, \ldots, k+s-1]$ should not exist anymore in SA. Binary search over the suffix array SA to identify positions of these suffixes in SA. Add the corresponding entries to *Disp*.

Insertion of the suffix

Suffixes starting at positions $[k+n, \ldots, k+n+s-1]$ do not exist yet in SA. Binary search over SA to find their positions of insertion. Add the corresponding entries to *Disp*.

Reorder suffixes

Proceed from right to left, starting by suffix at position $k+n-1$. Let j be the position of the current suffix in SA. Compute using binary search the new position $j' \geq j$ of this suffix.
Remark: $j' - j$ is not greater than at the previous step.

UNIVERSITÉ DE ROUEN

Example: creating *Disp* list

$$T = \begin{array}{c} \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \\ \mathsf{G\,A\,C\,T\,C\,T\,A\,C\,A\,C\,T\,A\,T\,C\,A} \end{array}$$

Example: creating *Disp* list

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	G	A	C	T	C	T	A	C	A	C	T	A	T	C	A

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp

Example: creating *Disp* list

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	G	A	C	T	C	T	A	C	A	C	T	A	T	C	A

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp

Example: creating *Disp* list

$$T = \underset{}{\overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15}{\mathrm{G\,A\,\underbrace{C\,T}_{\text{deleted}}\,C\,T\,A\,C\,A\,C\,T\,A\,\underbrace{T\,C}_{\text{inserted}}\,A}}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp

Example: creating *Disp* list

$$T = \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \\ G & A & \underbrace{C}_{\text{deleted}} & \underbrace{T}_{\text{deleted}} & C & T & A & C & A & C & T & A & \underbrace{T}_{\text{inserted}} & \underbrace{C}_{\text{inserted}} & A & \end{array}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp

Deleting suffix 3.

Example: creating *Disp* list

$$T = \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ G & A & \underbrace{C}_{\text{deleted}} & \underbrace{T}_{} & C & T & A & C & A & C & T & A & \underbrace{T}_{} & \underbrace{C}_{\text{inserted}} & A \end{array}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(7, 0)$

Deleting suffix 3.

Example: creating *Disp* list

$$T = \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ \mathrm{G} & \mathrm{A} & \underbrace{\mathrm{C}}_{\text{deleted}} & \mathrm{T} & \mathrm{C} & \mathrm{T} & \mathrm{A} & \mathrm{C} & \mathrm{A} & \mathrm{C} & \mathrm{T} & \mathrm{A} & \underbrace{\mathrm{T}}_{\text{inserted}} & \mathrm{C} & \mathrm{A} \end{array}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(7, 0)$

Deleting suffix 4.

Example: creating *Disp* list

$$T = \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \\ G & A & \underbrace{C}_{\text{deleted}} & \underbrace{T}_{} & C & T & A & C & A & C & T & A & \underbrace{T}_{\text{inserted}} & \underbrace{C}_{} & A & \$ \end{array}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp		$(7, 0)$	$(10, 0)$

Deleting suffix 4.

Example: creating *Disp* list

$$T = \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \\ G & A & \underbrace{C}_{\text{deleted}} & \underbrace{T}_{} & C & T & A & C & A & C & T & A & \underbrace{T}_{\text{inserted}} & \underbrace{C}_{} & A & \end{array}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp		$(7, 0)$	$(10, 0)$

Inserting suffix 14: C.

Example: creating *Disp* list

$$T = \underset{\text{deleted}}{\underbrace{\text{GACT}}}\text{CTACACTA}\underset{\text{inserted}}{\underbrace{\text{TC}}}\text{A}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

14

Disp $(4, 14)$ $(7, 0)$ $(10, 0)$

Inserting suffix 14: C.

Example: creating *Disp* list

$$T = \underset{\text{deleted}}{\text{GACT}}\text{CTACACTA}\underset{\text{inserted}}{\text{TC}}\text{A}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(4, 14)$ $(7, 0)$ $(10, 0)$

Inserting suffix 13: TC.

Example: creating *Disp* list

$$T = \underset{}{\overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15}{G\ A\ \underbrace{C\ T}_{\text{deleted}}\ C\ T\ A\ C\ A\ C\ T\ A\ \underbrace{T\ C}_{\text{inserted}}\ A}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
									↑	
									13	

Disp $(4, 14)$ $(7, 0)$ $(10, 0)$ $(10, 13)$

Inserting suffix 13: TC.

Example: creating *Disp* list

$$T = \underset{}{\overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15}{G\ A\ \underbrace{C\ T}_{\text{deleted}}\ C\ T\ A\ C\ A\ C\ T\ A\ \underbrace{T\ C}_{\text{inserted}}\ A}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp	$(4, 14)$	$(7, 0)$	$(10, -13)$

Inserting suffix 13: TC.

Example: creating *Disp* list

$$T = \underset{}{\overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15}{\mathrm{G\ A\ \underbrace{C\ T}_{\text{deleted}}\ C\ T\ A\ C\ A\ C\ T\ A\ \underbrace{T\ C}_{\text{inserted}}\ A}}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp	$(4, 14)$	$(7, 0)$	$(10, -13)$

Reordering suffix 12: ATC?

Example: creating *Disp* list

$$T = \begin{array}{c} \scriptstyle 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15 \\ \mathrm{G\,A\,\underbrace{C\,T}_{\text{deleted}}\,C\,T\,A\,C\,A\,C\,T\,A\,\underbrace{T\,C}_{\text{inserted}}\,A} \end{array}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(4, 14)$ $(7, 0)$ $(10, -13)$

Reordering suffix 12: ATC?

Example: creating *Disp* list

$T =$ G A C T C T A C A C T A T C A

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

C T deleted; T C inserted

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(1, 0)$ $(4, 12)$ $(4, 14)$ $(7, 0)$ $(10, -13)$

Reordering suffix 12: ATC?

Example: creating *Disp* list

$$T = \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \\ G & A & \underbrace{C}_{\text{deleted}} & \underbrace{T}_{\text{deleted}} & C & T & A & C & A & C & T & A & \underbrace{T}_{\text{inserted}} & \underbrace{C}_{\text{inserted}} & A & \end{array}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(1, 0)$ $(4, 12)$ $(4, 14)$ $(7, 0)$ $(10, -13)$

Reordering suffix 11: TATC?

Example: creating *Disp* list

$$T = \underset{}{\overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15}{G\ A\ \underbrace{C\ T}_{\text{deleted}}\ C\ T\ A\ C\ A\ C\ T\ A\ \underbrace{T\ C}_{\text{inserted}}\ A}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(1, 0)$ $(4, 12)$ $(4, 14)$ $(7, 0)$ $(8, 0)$ $(10, 11)$ $(10, -13)$

Reordering suffix 11: TATC?

Example: creating *Disp* list

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	G	A	C	T	C	T	A	C	A	C	T	A	T	C	A

deleted (positions 3–4), inserted (positions 13–14)

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Disp $(1,0)$ $(4,12)$ $(4,14)$ $(5,0)$ $(7,-10)$ $(8,0)$ $(10,11)$ $(10,-13)$

Reordering suffix 10: CTATC?

Example: creating *Disp* list

$$T = \underset{}{\overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15}{G\ A\ \underbrace{C\ T}_{\text{deleted}}\ C\ T\ A\ C\ A\ C\ T\ A\ \underbrace{T\ C}_{\text{inserted}}\ A}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

↑

Disp $(1,0)$ $(4,12)$ $(4,14)$ $(5,0)$ $(7,-10)$ $(8,0)$ $(10,11)$ $(10,-13)$

Reordering suffix 9: ACTATC?

Example: updating SA using *Disp*

Disp	$(1,0)$	$(4,12)$	$(4,14)$	$(5,0)$	$(7,-10)$	$(8,0)$	$(10,11)$	$(10,-13)$

Example: updating SA using *Disp*

$$\underset{\textit{Disp}}{} \underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$$

Example: updating SA using *Disp*

Disp	$\underbrace{(1,0)\ \ (4,12)}$	$\underbrace{(4,14)\ \ (5,0)}$	$\underbrace{(7,-10)}$	$\underbrace{(8,0)\ \ (10,11)}$	$\underbrace{(10,-13)}$
	Left shift	Right shift	Static	Left shift	Static

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Example: updating SA using *Disp*

Disp	$\underbrace{(1,0)\ \ (4,12)}$	$\underbrace{(4,14)\ \ (5,0)}$	$\underbrace{(7,-10)}$	$\underbrace{(8,0)\ \ (10,11)}$	$\underbrace{(10,-13)}$
	Left shift	Right shift	Static	Left shift	Static

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4

Example: updating SA using *Disp*

Disp	$\underbrace{(1,0)\ \ (4,12)}$	$\underbrace{(4,14)\ \ (5,0)}$	$\underbrace{(7,-10)}$	$\underbrace{(8,0)\ \ (10,11)}$	$\underbrace{(10,-13)}$
	Left shift	Right shift	Static	Left shift	Static

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9								

Example: updating SA using *Disp*

Disp	$\underbrace{(1,0)\ \ (4,12)}$	$\underbrace{(4,14)\ \ (5,0)}$	$\underbrace{(7,-10)}$	$\underbrace{(8,0)\ \ (10,11)}$	$\underbrace{(10,-13)}$
	Left shift	Right shift	Static	Left shift	Static

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12							

Example: updating SA using *Disp*

Disp

$$\underbrace{(1,0)\ \ (4,12)}_{\text{Left shift}}\ \ \underbrace{(4,14)\ \ (5,0)}_{\text{Right shift}}\ \ \underbrace{(7,-10)}_{\text{Static}}\ \ \underbrace{(8,0)\ \ (10,11)}_{\text{Left shift}}\ \ \underbrace{(10,-13)}_{\text{Static}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12							

Example: updating SA using *Disp*

Disp $\underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12		8					

Example: updating SA using *Disp*

Disp

$\underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8					

Example: updating SA using *Disp*

Disp $\underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8					

Example: updating SA using *Disp*

Disp $\underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8		10			

Example: updating SA using *Disp*

Disp

$$\underbrace{(1,0)\ \ (4,12)}_{\text{Left shift}}\ \ \underbrace{(4,14)\ \ (5,0)}_{\text{Right shift}}\ \ \underbrace{(7,-10)}_{\text{Static}}\ \ \underbrace{(8,0)\ \ (10,11)}_{\text{Left shift}}\ \ \underbrace{(10,-13)}_{\text{Static}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8		10			

Example: updating SA using *Disp*

Disp $\underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8		10	6		

Example: updating SA using *Disp*

Disp

$$\underbrace{(1,0)\ \ (4,12)}_{\text{Left shift}}\ \ \underbrace{(4,14)\ \ (5,0)}_{\text{Right shift}}\ \ \underbrace{(7,-10)}_{\text{Static}}\ \ \underbrace{(8,0)\ \ (10,11)}_{\text{Left shift}}\ \ \underbrace{(10,-13)}_{\text{Static}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8		10	6	11	

Example: updating SA using *Disp*

Disp $\underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8		10	6	11	

Example: updating SA using *Disp*

$$\textit{Disp} \quad \underbrace{(1,0)\ \ (4,12)}_{\text{Left shift}}\ \ \underbrace{(4,14)\ \ (5,0)}_{\text{Right shift}}\ \ \underbrace{(7,-10)}_{\text{Static}}\ \ \underbrace{(8,0)\ \ (10,11)}_{\text{Left shift}}\ \ \underbrace{(10,-13)}_{\text{Static}}$$

	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8		10	6	11	13

Example: updating SA using *Disp*

$$\underset{\textit{Disp}}{} \underbrace{(1,0)\ (4,12)}_{\text{Left shift}}\ \underbrace{(4,14)\ (5,0)}_{\text{Right shift}}\ \underbrace{(7,-10)}_{\text{Static}}\ \underbrace{(8,0)\ (10,11)}_{\text{Left shift}}\ \underbrace{(10,-13)}_{\text{Static}}$$

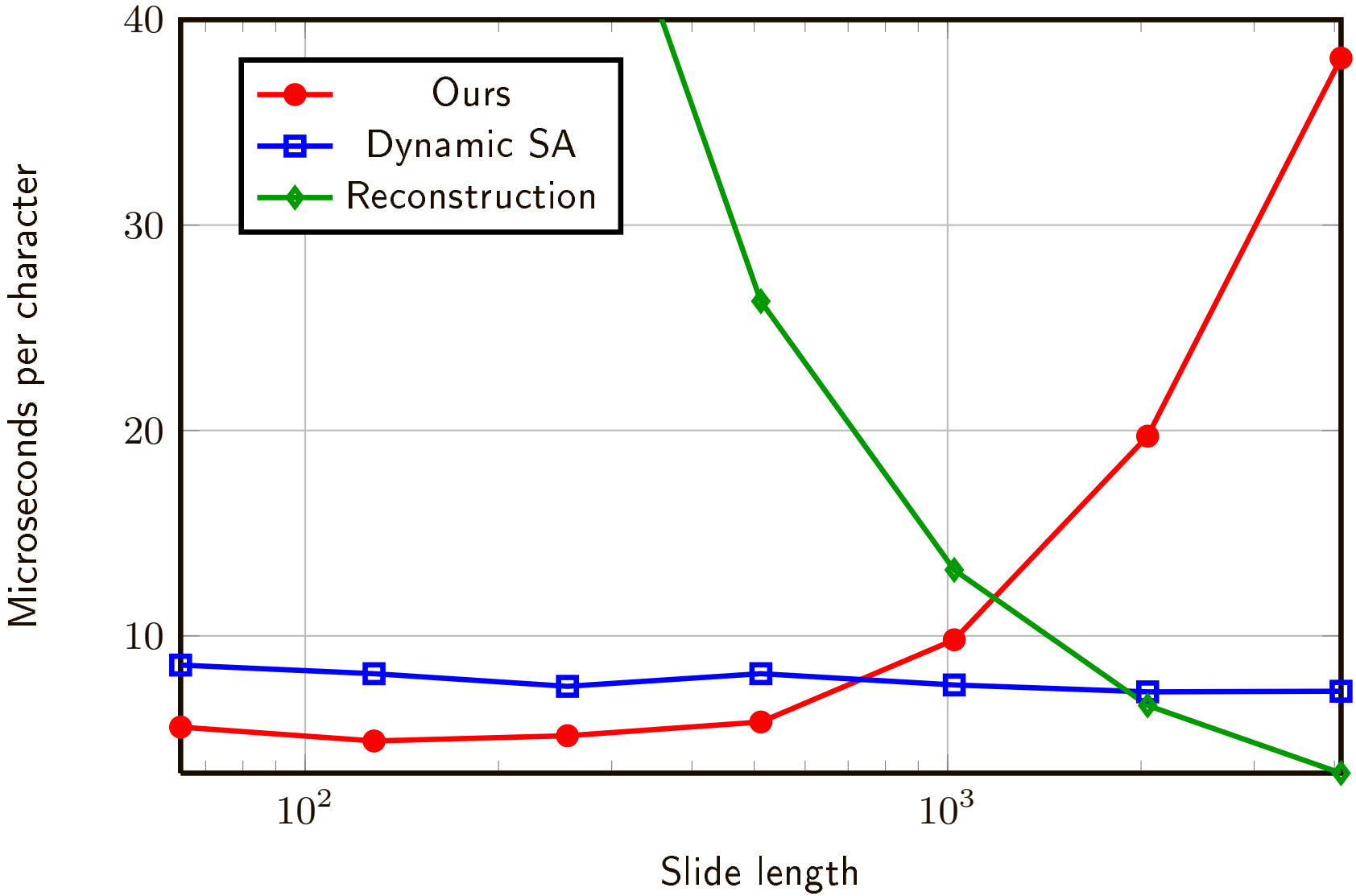
	1	2	3	4	5	6	7	8	9	10
SA	12	7	9	8	10	5	3	11	6	4
	7	9	12	14	8		10	6	11	13

$$T = \overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15}{\text{GACTCTACACTATCA}}$$

	1	2	3	4	5	6	7	8	9	10
SA	7	9	12	14	8	5	10	6	11	13

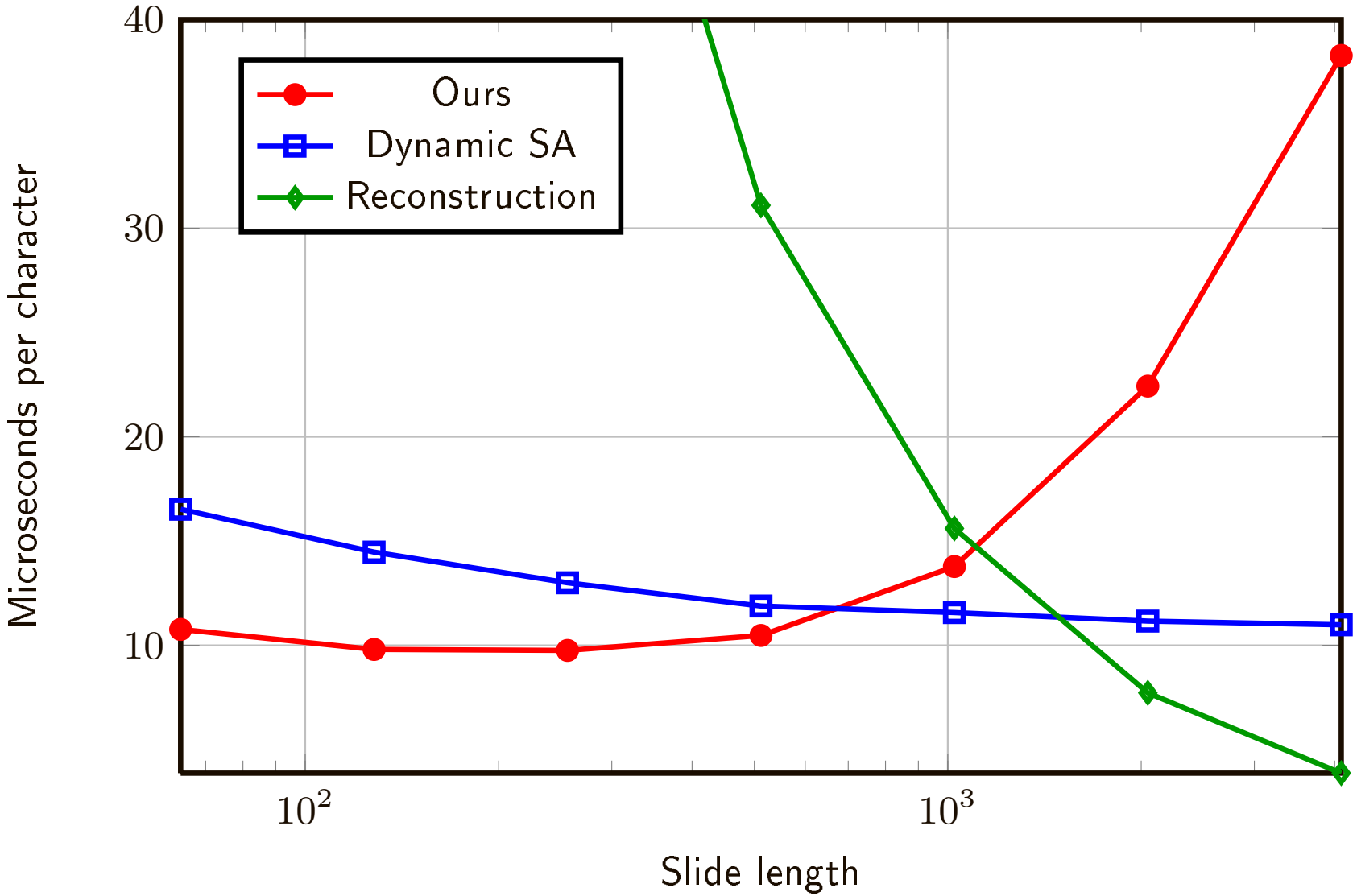
Experiments

On a genome sequence with a 32K-length window.



Experiments

On a XML file with a 32K-length window.



Conclusions and Perspectives

Conclusions

- Constant-time access to any value of the suffix array (rather than logarithmic for the dynamic suffix array);
- More efficient than reconstruction or dynamic suffix array for "small" slides (< 1000)

Perspectives

- Fast Lempel-Ziv factorization?
- Implement and test this LZ approach.