Internal Dictionary Matching

Panagiotis Charalampopoulos[1,2], Tomasz Kociumaka[2,3], Manal Mohamed[1], Jakub Radoszewski[2,4], Wojciech Rytter[2], Tomasz Waleń[2]

[1]King's College London, UK

[2]University of Warsaw, Poland

[3]Bar-Ilan University, Israel

[4]Samsung R&D Institute Poland

ISAAC 2019

Introduction

Introduction

Dictionary Matching Problem

Introduction

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.

Introduction

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Introduction

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Aho–Corasick automaton [1975]:
space $\mathcal{O}(||\mathcal{D}||)$, $t_{query} = \mathcal{O}(|T| + |output|)$.

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Aho–Corasick automaton [1975]:
space $\mathcal{O}(||\mathcal{D}||)$, $t_{query} = \mathcal{O}(|T| + |output|)$.

Internal Pattern Matching

Introduction

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Aho–Corasick automaton [1975]:
space $\mathcal{O}(||\mathcal{D}||)$, $t_{query} = \mathcal{O}(|T| + |output|)$.

Internal Pattern Matching

Input: Text T.

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Aho–Corasick automaton [1975]:
space $\mathcal{O}(||\mathcal{D}||)$, $t_{query} = \mathcal{O}(|T| + |output|)$.

Internal Pattern Matching

Input: Text T.
Query: Occurrences of a substring x of T in another substring y.

Introduction

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Aho–Corasick automaton [1975]:
space $\mathcal{O}(||\mathcal{D}||)$, $t_{query} = \mathcal{O}(|T| + |output|)$.

Internal Pattern Matching

Input: Text T.
Query: Occurrences of a substring x of T in another substring y.

Keller et al. [2014]: space $\tilde{\mathcal{O}}(n)$, $t_{query} = \mathcal{O}(\log\log n + |output|)$.

Introduction

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Aho–Corasick automaton [1975]:
space $\mathcal{O}(||\mathcal{D}||)$, $t_{query} = \mathcal{O}(|T| + |output|)$.

Internal Pattern Matching

Input: Text T.
Query: Occurrences of a substring x of T in another substring y.

Keller et al. [2014]: space $\tilde{\mathcal{O}}(n)$, $t_{query} = \mathcal{O}(\log\log n + |output|)$.
Kociumaka et al. [2015]: space $\mathcal{O}(n)$, $t_{query} = \mathcal{O}(|y|/|x|)$.

Introduction

Dictionary Matching Problem

Input: Dictionary $\mathcal{D}$ of total length $||\mathcal{D}||$.
Query: Occurrences of patterns from $\mathcal{D}$ in a text T.

Aho–Corasick automaton [1975]:
space $\mathcal{O}(||\mathcal{D}||)$, $t_{query} = \mathcal{O}(|T| + |output|)$.

Internal Pattern Matching

Input: Text T.
Query: Occurrences of a substring x of T in another substring y.

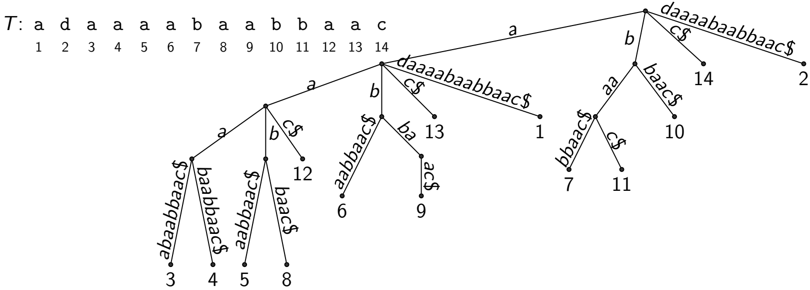
Keller et al. [2014]: space $\tilde{\mathcal{O}}(n)$, $t_{query} = \mathcal{O}(\log\log n + |output|)$.
Kociumaka et al. [2015]: space $\mathcal{O}(n)$, $t_{query} = \mathcal{O}(|y|/|x|)$.

Other internal queries: longest common prefix of two suffixes of T, periods of a substring of T, etc.

IPM → 2D Range Reporting (Mäkinen and Navarro)

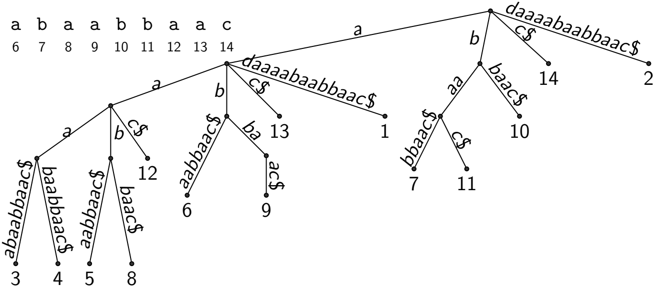
T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

IPM → 2D Range Reporting (Mäkinen and Navarro)



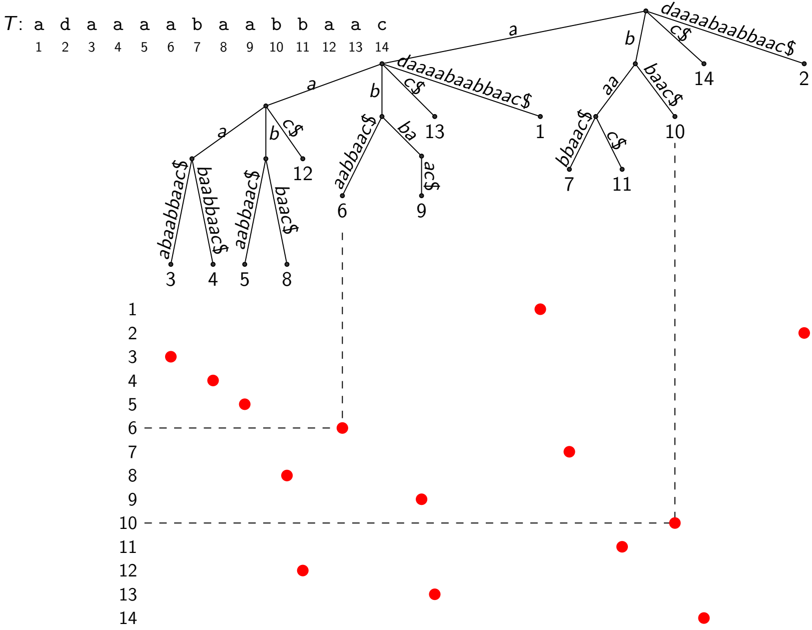
IPM → 2D Range Reporting (Mäkinen and Navarro)

T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

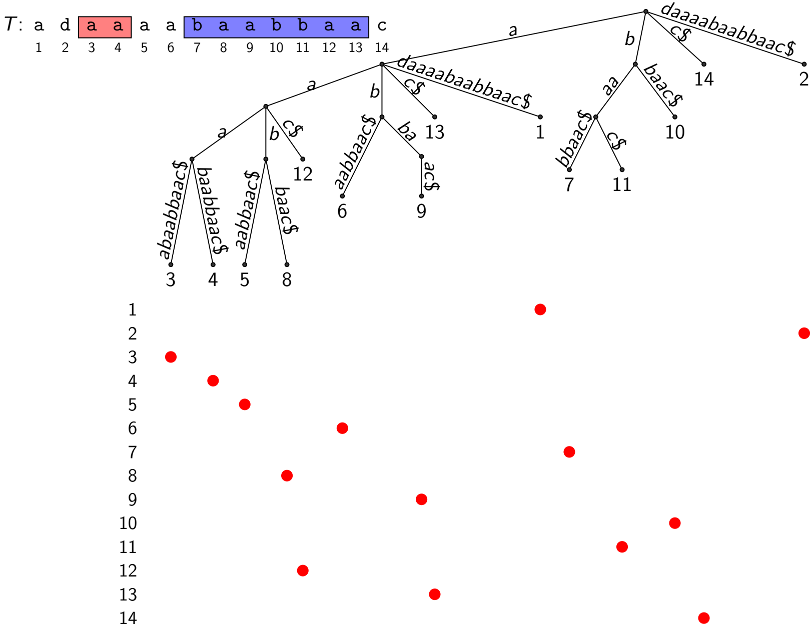


1
2
3
4
5
6
7
8
9
10
11
12
13
14

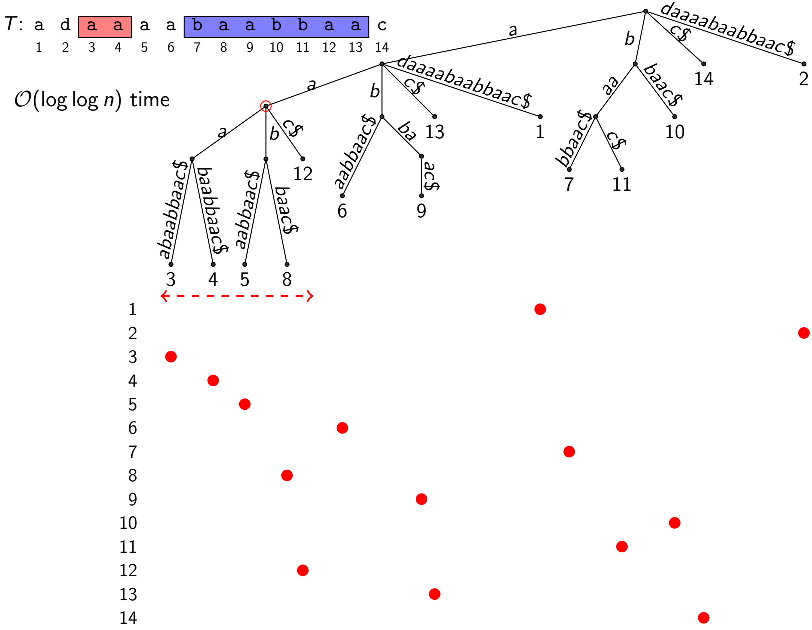
IPM → 2D Range Reporting (Mäkinen and Navarro)



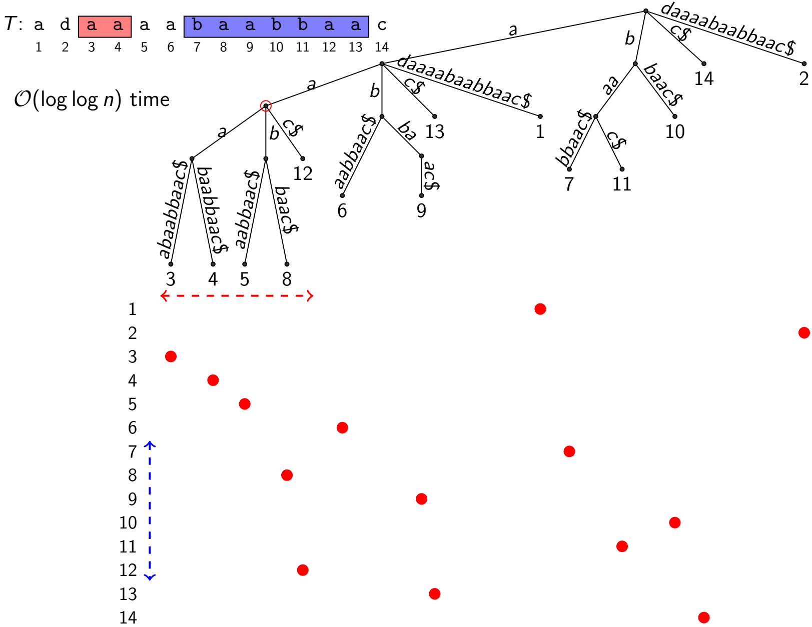
IPM → 2D Range Reporting (Mäkinen and Navarro)



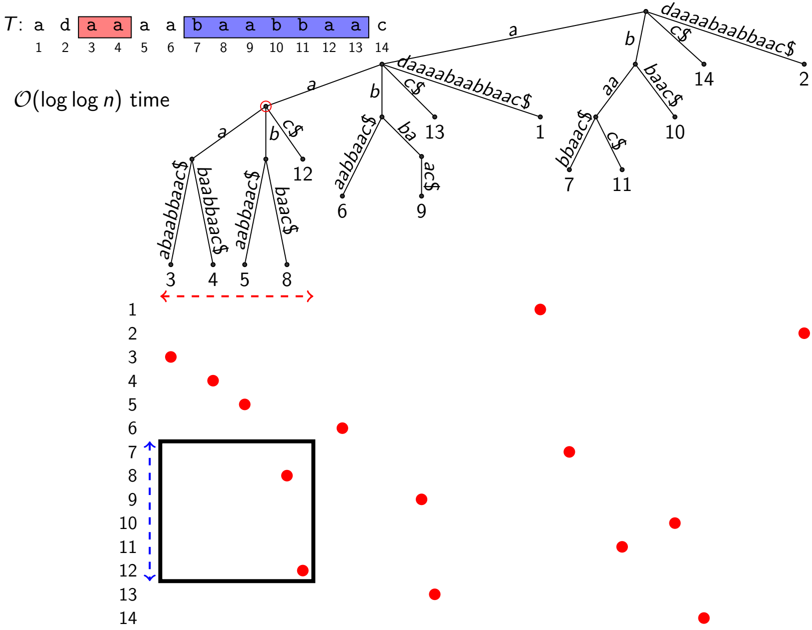
IPM → 2D Range Reporting (Mäkinen and Navarro)



IPM → 2D Range Reporting (Mäkinen and Navarro)



IPM → 2D Range Reporting (Mäkinen and Navarro)



Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.

$\mathcal{D}$:

a a a a a a

a b b a c

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.

$\mathcal{D}$:

a a   a a a a

a b b a   c

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell\,..\,r]$ of T.

$\mathcal{D}$:

a a   a a a a

a b b a   c

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.

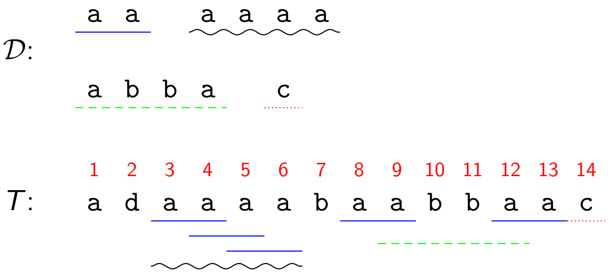
$\mathcal{D}$: a a   a a a a

a b b a   c

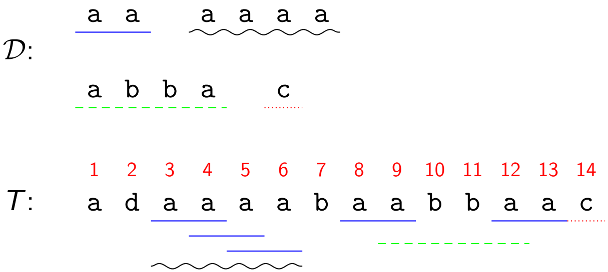
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell\,..\,r]$ of T.



Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.

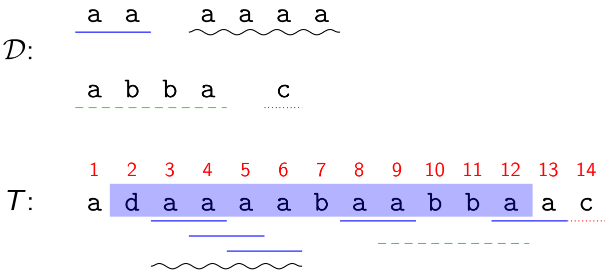


Exists(i, j)

- Decide whether at least one pattern $P \in \mathcal{D}$ occurs in $T[i \mathinner{.\,.} j]$.

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \,..\, r]$ of T.

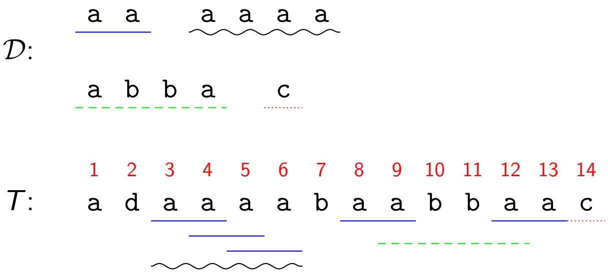


$\textsc{Exists}(i, j)$

- Decide whether at least one pattern $P \in \mathcal{D}$ occurs in $T[i \,..\, j]$.
- $\textsc{Exists}(2{,}12) = \text{true}$

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \,..\, r]$ of T.

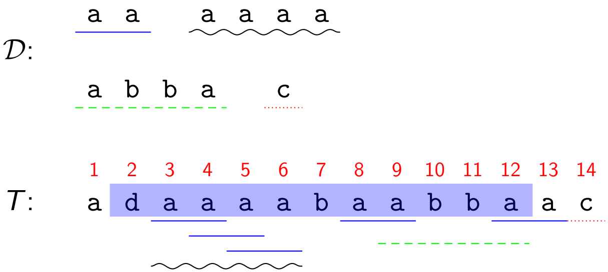


Report(i, j)

- Report all occurrences of all patterns of $\mathcal{D}$ in $T[i \,..\, j]$.

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell\,..\,r]$ of T.

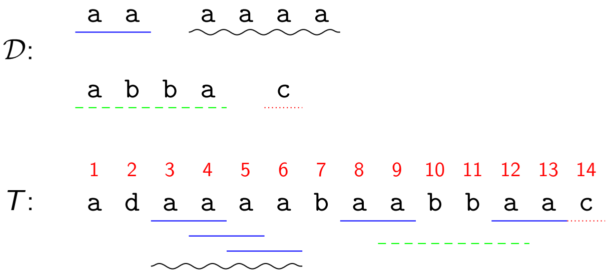


Report(i, j)

- Report all occurrences of all patterns of $\mathcal{D}$ in $T[i\,..\,j]$.
- Report(2,12) = (aa,3),(aaaa,3),(aa,4),(aa,5),(aa,8), (abba,9)

Problem Definition

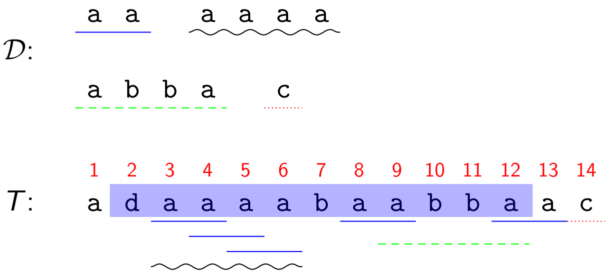
Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.



ReportDistinct(i, j)

- Report all patterns P of $\mathcal{D}$ that occur in $T[i \mathinner{.\,.} j]$.

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \,.\,.\, r]$ of T.

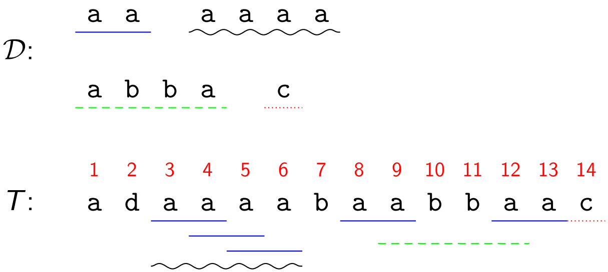


ReportDistinct(i, j)

- Report all patterns P of $\mathcal{D}$ that occur in $T[i \,.\,.\, j]$.
- ReportDistinct(2,12) $=$ `aa, aaaa, abba`

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.

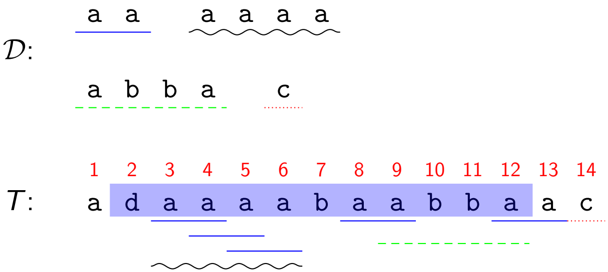


Count(i, j)

- Count the number of all occurrences of all the patterns of $\mathcal{D}$ in $T[i \mathinner{.\,.} j]$.

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.

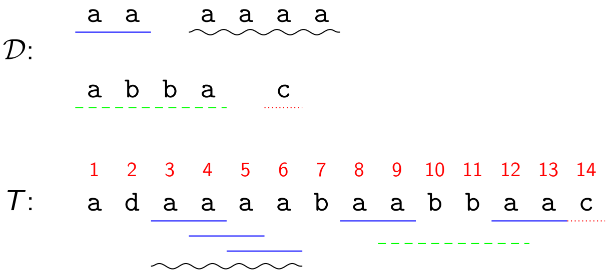


Count(i, j)

- Count the number of all occurrences of all the patterns of $\mathcal{D}$ in $T[i \mathinner{.\,.} j]$.
- Count(2,12) $=$ 6

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.

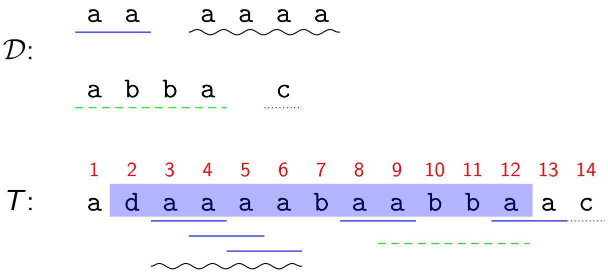


CountDistinct(i, j)

- Count all patterns of $\mathcal{D}$ that occur in $T[i \mathinner{.\,.} j]$.

Problem Definition

Input: A text T of length n and a dictionary $\mathcal{D}$ consisting of d patterns, each given as a substring $T[\ell \mathinner{.\,.} r]$ of T.



CountDistinct(i,j)

- Count all patterns of $\mathcal{D}$ that occur in $T[i \mathinner{.\,.} j]$.
- CountDistinct(2,12) $=$ 3

Our Results

Query	Preprocessing time	Space	Query time
$\textsc{Exists}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
$\textsc{Report}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(1+\lvert output\rvert)$
$\textsc{ReportDistinct}(i,j)$	$\mathcal{O}(n\log n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(\log n+\lvert output\rvert)$
$\textsc{Count}(i,j)$	$\mathcal{O}(\frac{n\log n}{\log\log n}+d\log^{3/2} n)$	$\mathcal{O}(n+d\log n)$	$\mathcal{O}(\frac{\log^2 n}{\log\log n})$
	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(1+\lvert output\rvert)$

Our Results

Query	Preprocessing time	Space	Query time
$\textsc{Exists}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
$\textsc{Report}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(1+\lvert output\rvert)$
$\textsc{ReportDistinct}(i,j)$	$\mathcal{O}(n\log n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(\log n+\lvert output\rvert)$
$\textsc{Count}(i,j)$	$\mathcal{O}(\frac{n\log n}{\log\log n}+d\log^{3/2} n)$	$\mathcal{O}(n+d\log n)$	$\mathcal{O}(\frac{\log^2 n}{\log\log n})$
	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(1+\lvert output\rvert)$

Dynamic dictionary

Our Results

Query	Preprocessing time	Space	Query time
$\textsc{Exists}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
$\textsc{Report}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(1+\lvert output\rvert)$
$\textsc{ReportDistinct}(i,j)$	$\mathcal{O}(n\log n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(\log n+\lvert output\rvert)$
$\textsc{Count}(i,j)$	$\mathcal{O}(\frac{n\log n}{\log\log n}+d\log^{3/2} n)$	$\mathcal{O}(n+d\log n)$	$\mathcal{O}(\frac{\log^2 n}{\log\log n})$
	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(1+\lvert output\rvert)$

Dynamic dictionary

- Conditional lower bound: $t_{upd} \times t_{query}$ cannot be $\mathcal{O}(n^{1-\epsilon})$ for any constant $\epsilon > 0$, unless the Online Boolean Matrix-Vector Multiplication conjecture is false.

Our Results

Query	Preprocessing time	Space	Query time
$\textsc{Exists}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
$\textsc{Report}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(1+\lvert output\rvert)$
$\textsc{ReportDistinct}(i,j)$	$\mathcal{O}(n\log n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(\log n+\lvert output\rvert)$
$\textsc{Count}(i,j)$	$\mathcal{O}(\frac{n\log n}{\log\log n}+d\log^{3/2} n)$	$\mathcal{O}(n+d\log n)$	$\mathcal{O}(\frac{\log^2 n}{\log\log n})$
	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(n+d)$	$\tilde{\mathcal{O}}(1+\lvert output\rvert)$

Dynamic dictionary

- Conditional lower bound: $t_{upd} \times t_{query}$ cannot be $\mathcal{O}(n^{1-\epsilon})$ for any constant $\epsilon > 0$, unless the Online Boolean Matrix-Vector Multiplication conjecture is false.
- Upper bound: $t_{upd} = \tilde{\mathcal{O}}(n^{\alpha})$, $t_{query} = \tilde{\mathcal{O}}(n^{1-\alpha} + \lvert output\rvert)$ for any $0 < \alpha < 1$, for all our queries.

ReportDistinct(i, j)

$\textsc{ReportDistinct}(i, j)$

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

ReportDistinct(i, j)

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.

Partition $\mathcal{D}$ into $\mathcal{D}_0, \dots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

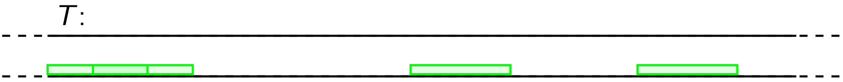
Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.

T:

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.



Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

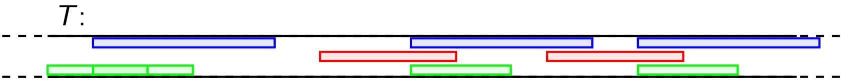
$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary.}$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

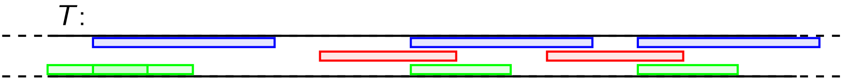
Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.



Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.

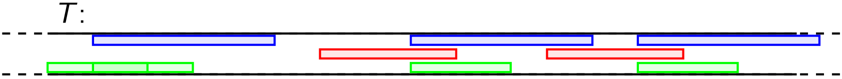


$C_k[a] = \text{col}(P_a)$, P_a the longest pattern occurring at position a

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.



⋯	g	b	⋅⋅	⋅⋅	⋅⋅	⋅⋅	r	⋅⋅	b	⋅⋅	⋅⋅	r	⋅⋅	b	⋅⋅	⋅⋅	⋅⋅	⋯

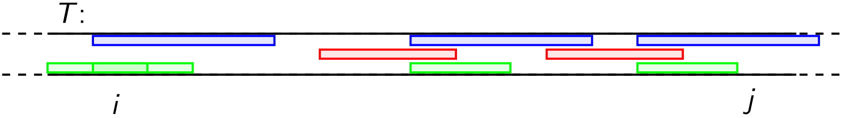
$C_k[a] = \text{col}(P_a)$, P_a the longest pattern occurring at position a

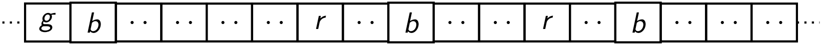
ReportDistinct(i, j)

Partition $\mathcal{D}$ into $\mathcal{D}_0, \dots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.



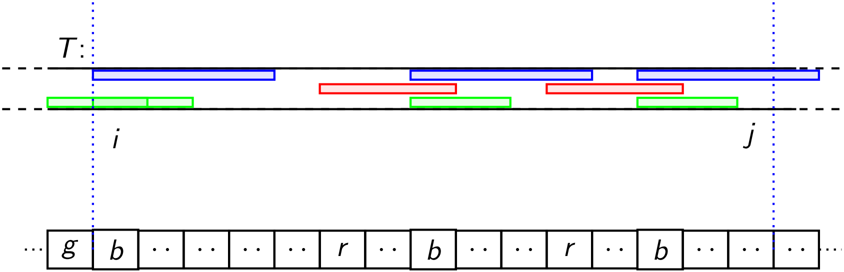


$C_k[a] = \text{col}(P_a)$, P_a the longest pattern occurring at position a

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.

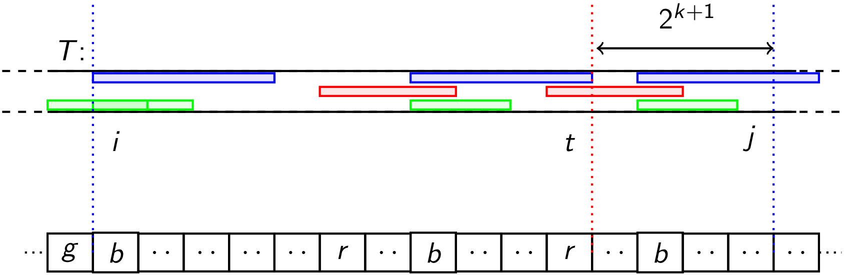


$C_k[a] = \text{col}(P_a)$, P_a the longest pattern occurring at position a

Partition $\mathcal{D}$ into $\mathcal{D}_0,\ldots,\mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color $\text{col}(P)$ to each pattern $P \in \mathcal{D}_k$.

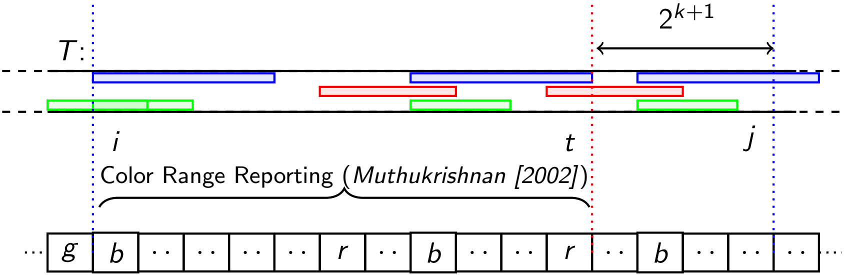


$C_k[a] = \text{col}(P_a)$, P_a the longest pattern occurring at position a

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.

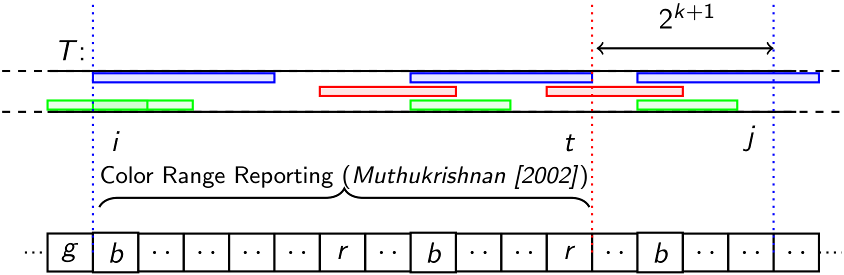


$C_k[a] = \text{col}(P_a)$, P_a the longest pattern occurring at position a

Partition $\mathcal{D}$ into $\mathcal{D}_0, \ldots, \mathcal{D}_{\lfloor \log n \rfloor}$, where

$$\mathcal{D}_k = \{P \in \mathcal{D} : 2^k \leq |P| < 2^{k+1}\} \text{ is the } k\text{-dictionary}.$$

Assign a color col(P) to each pattern $P \in \mathcal{D}_k$.



$C_k[a] = \text{col}(P_a)$, P_a the longest pattern occurring at position a

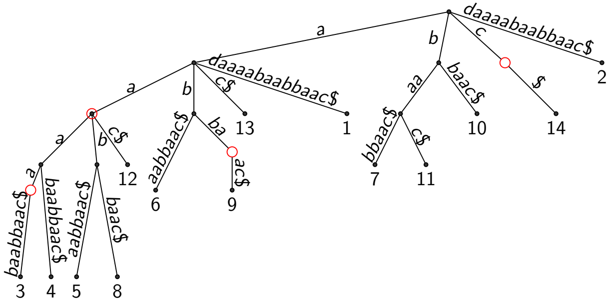
Every $P \in \mathcal{D}_k$ that occurs in some position $a \in [i, t]$ is a prefix of one of the reported patterns.

$(T, \mathcal{D})$-trie

$T = adaaaabaabbaac$ and $\mathcal{D} = \{aa, aaaa, abba, c\}$.

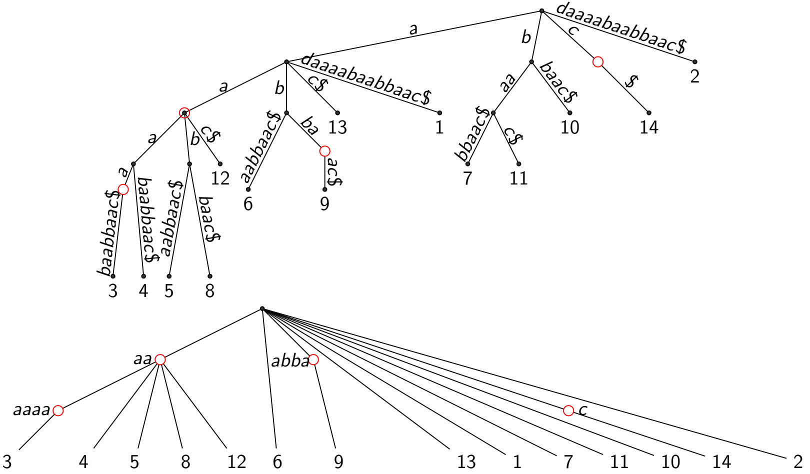
$(T, \mathcal{D})$-trie

$T = adaaaabaabbaac$ and $\mathcal{D} = \{aa, aaaa, abba, c\}$.
Suffix tree of T:



$(T, \mathcal{D})$-trie

$T = adaaaabaabbaac$ and $\mathcal{D} = \{aa, aaaa, abba, c\}$.
Suffix tree of T:



A tree representing the patterns. Prefixes in $\mathcal{D} \Leftrightarrow$ Ancestors.

Periodicity

We still have to report patterns occurring in $T[t+1\,..\,j]$, $j - t = 2^{k+1}$.

Periodicity

We still have to report patterns occurring in $T[t+1\,..\,j]$, $j - t = 2^{k+1}$.

Period

A positive integer q is a **period** of a string S if $S[i] = S[i+q]$ for all $i = 1, \ldots, |S| - q$.

Periodicity

We still have to report patterns occurring in $T[t+1\,..\,j]$, $j-t=2^{k+1}$.

Period

A positive integer q is a **period** of a string S if $S[i]=S[i+q]$ for all $i=1,\ldots,|S|-q$.

The smallest period $\mathrm{per}(S)$ is **the period** of S.

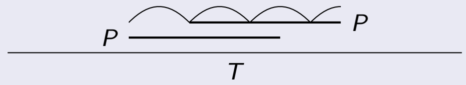
Periodicity

We still have to report patterns occurring in $T[t+1\,..\,j]$, $j - t = 2^{k+1}$.

Period

A positive integer q is a **period** of a string S if $S[i] = S[i+q]$ for all $i = 1, \ldots, |S| - q$.

The smallest period $\mathrm{per}(S)$ is **the period** of S.



Two occurrences of P in T at distance $q < |P| \Rightarrow \mathrm{per}(P) \leq q$.

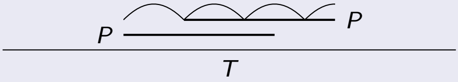
Periodicity

We still have to report patterns occurring in $T[t+1\,..\,j]$, $j - t = 2^{k+1}$.

Period

A positive integer q is a **period** of a string S if $S[i] = S[i+q]$ for all $i = 1, \ldots, |S| - q$.

The smallest period $\text{per}(S)$ is **the period** of S.



Two occurrences of P in T at distance $q < |P| \Rightarrow \text{per}(P) \leq q$.

Decompose $\mathcal{D}_k$ into (periodic) patterns P with $\text{per}(P) \leq 2^k/3$ and the remaining (aperiodic) ones.

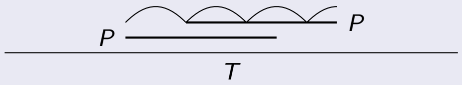
Periodicity

We still have to report patterns occurring in $T[t+1\,..\,j]$, $j-t=2^{k+1}$.

Period

A positive integer q is a **period** of a string S if $S[i]=S[i+q]$ for all $i=1,\ldots,|S|-q$.

The smallest period $\mathrm{per}(S)$ is **the period** of S.



Two occurrences of P in T at distance $q<|P| \Rightarrow \mathrm{per}(P)\leq q$.

Decompose $\mathcal{D}_k$ into (periodic) patterns P with $\mathrm{per}(P)\leq 2^k/3$ and the remaining (aperiodic) ones.

Use $\textsc{Report}(t,j)$ for aperiodic patterns!

Each of them occurs in $T[t+1\,..\,j]$ a constant number of times.

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a \mathinner{.\,.} b]$ is periodic with period q;

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a \,..\, b]$ is periodic with period q;
- the interval $[a \,..\, b]$ cannot be extended to the left nor to the right without violating the above property.

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a\,..\,b]$ is periodic with period q;
- the interval $[a\,..\,b]$ cannot be extended to the left nor to the right without violating the above property.

c	a	b	a	a	b	a	a	b	a	a	b	a	a	b	b	b	a
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

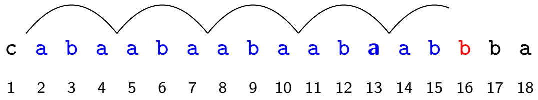
run $(2, 15, 3)$

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a\,..\,b]$ is periodic with period q;
- the interval $[a\,..\,b]$ cannot be extended to the left nor to the right without violating the above property.



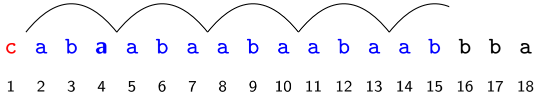
run $(2, 15, 3)$

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a \mathinner{.\,.} b]$ is periodic with period q;
- the interval $[a \mathinner{.\,.} b]$ cannot be extended to the left nor to the right without violating the above property.

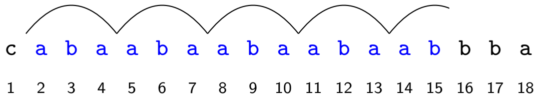


run $(2, 15, 3)$

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a\,..\,b]$ is periodic with period q;
- the interval $[a\,..\,b]$ cannot be extended to the left nor to the right without violating the above property.

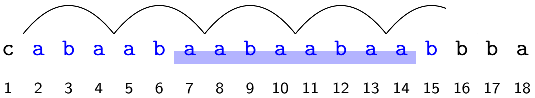


run $(2, 15, 3)$

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a\,..\,b]$ is periodic with period q;
- the interval $[a\,..\,b]$ cannot be extended to the left nor to the right without violating the above property.



run $(2, 15, 3)$

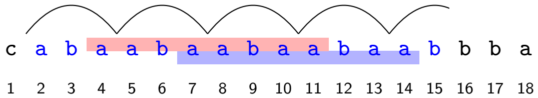
- A pattern P from the periodic k-dictionary occurs exactly once in the first $\text{per}(R) = \text{per}(P)$ positions of a run R.

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a \mathinner{.\,.} b]$ is periodic with period q;
- the interval $[a \mathinner{.\,.} b]$ cannot be extended to the left nor to the right without violating the above property.



run $(2, 15, 3)$

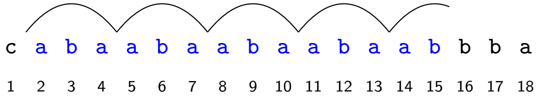
- A pattern P from the periodic k-dictionary occurs exactly once in the first $\text{per}(R) = \text{per}(P)$ positions of a run R.

Runs

Run

A **run** in a string S is a triple (a, b, q) such that:

- the substring $S[a\,..\,b]$ is periodic with period q;
- the interval $[a\,..\,b]$ cannot be extended to the left nor to the right without violating the above property.



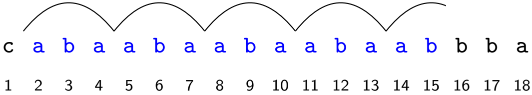
run $(2, 15, 3)$

- Two runs with periods $\leq 2^k/3$ cannot overlap a lot $\Rightarrow$ a constant number of them has $\geq 2^k$ overlap with $T[t+1\,..\,j]$.

Run

A **run** in a string S is a triple (a, b, q) such that:

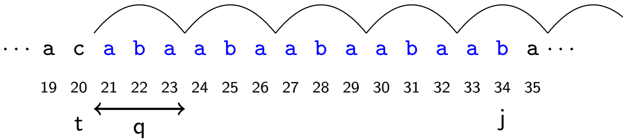
- the substring $S[a\,..\,b]$ is periodic with period q;
- the interval $[a\,..\,b]$ cannot be extended to the left nor to the right without violating the above property.



run $(2, 15, 3)$

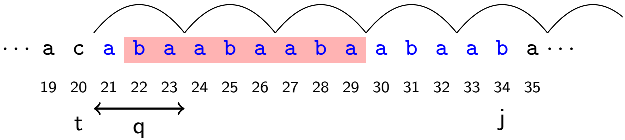
- A pattern P from the periodic k-dictionary occurs exactly once in the first $\text{per}(R) = \text{per}(P)$ positions of a run R.
- Two runs with periods $\leq 2^k/3$ cannot overlap a lot $\Rightarrow$ a constant number of them has $\geq 2^k$ overlap with $T[t+1\,..\,j]$.

We are left with finding patterns occurring in the first $q = \text{per}(R)$ positions of each relevant run R.



Processing a Periodic k-Dictionary

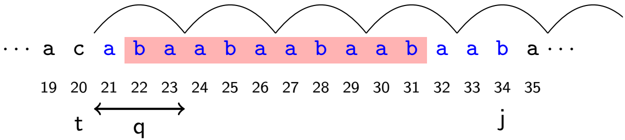
We are left with finding patterns occurring in the first $q = \text{per}(R)$ positions of each relevant run R.



- Find the interesting positions and then report all patterns using the $(T, \mathcal{D})$-trie.

Processing a Periodic k-Dictionary

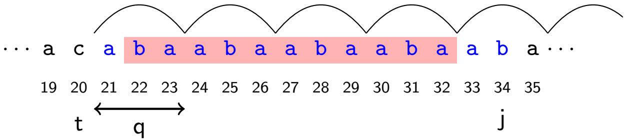
We are left with finding patterns occurring in the first $q = \text{per}(R)$ positions of each relevant run R.



- Find the interesting positions and then report all patterns using the $(T, \mathcal{D})$-trie.

Processing a Periodic k-Dictionary

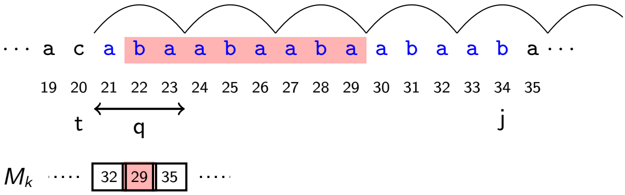
We are left with finding patterns occurring in the first $q = \text{per}(R)$ positions of each relevant run R.



- Find the interesting positions and then report all patterns using the $(T, \mathcal{D})$-trie.

Processing a Periodic k-Dictionary

We are left with finding patterns occurring in the first $q = \text{per}(R)$ positions of each relevant run R.



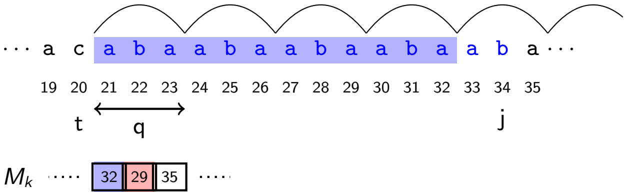
- Find the interesting positions and then report all patterns using the $(T, \mathcal{D})$-trie.

- Repeatedly use range minimum queries on array:

  $M_k(a) =$ position where the shortest pattern occurring at a ends.

Processing a Periodic k-Dictionary

We are left with finding patterns occurring in the first $q = \text{per}(R)$ positions of each relevant run R.



- Find the interesting positions and then report all patterns using the $(T, \mathcal{D})$-trie.

- Repeatedly use range minimum queries on array:

  $M_k(a) =$ position where the shortest pattern occurring at a ends.

Wrap-up

$\mathcal{O}(\log n)$ subdictionaries.

Wrap-up

$\mathcal{O}(\log n)$ subdictionaries.

$\mathcal{D}_k$ requires: space $\mathcal{O}(n + d_k)$, $t_{query} = \mathcal{O}(1 + |output|)$.

Wrap-up

$\mathcal{O}(\log n)$ subdictionaries.

$\mathcal{D}_k$ requires: space $\mathcal{O}(n + d_k)$, $t_{query} = \mathcal{O}(1 + |output|)$.

Overall: space $\mathcal{O}(n \log n + d)$, $t_{query} = \mathcal{O}(\log n + |output|)$.

Wrap-up

$\mathcal{O}(\log n)$ subdictionaries.

$\mathcal{D}_k$ requires: space $\mathcal{O}(n + d_k)$, $t_{query} = \mathcal{O}(1 + |output|)$.

Overall: space $\mathcal{O}(n \log n + d)$, $t_{query} = \mathcal{O}(\log n + |output|)$.

Grammar Compressed Strings: Straight-Line Programs

Definition

A straight-line program (aka straight-line grammar) –
a context-free grammar generating a single string.

Grammar Compressed Strings: Straight-Line Programs

Definition

A straight-line program (aka straight-line grammar) –
a context-free grammar generating a single string.

$X_0 \to \mathtt{b}$

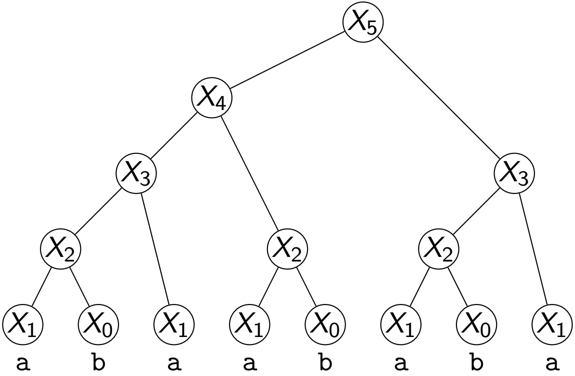
$X_1 \to \mathtt{a}$

$X_2 \to X_1 X_0$

$X_3 \to X_2 X_1$

$X_4 \to X_3 X_2$

$X_5 \to X_4 X_3$



Definition

A straight-line program (aka straight-line grammar) –
a context-free grammar generating a single string.

$X_0 \rightarrow \mathtt{b}$

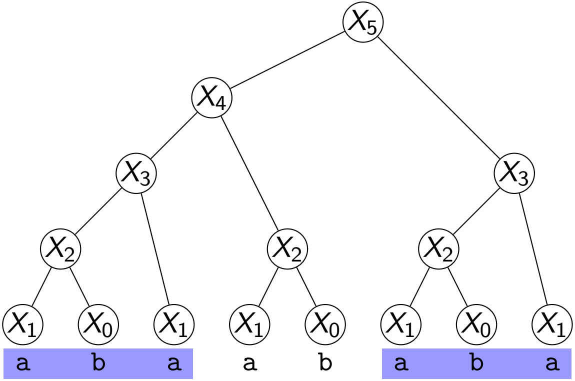
$X_1 \rightarrow \mathtt{a}$

$X_2 \rightarrow X_1 X_0$

$X_3 \rightarrow X_2 X_1$

$X_4 \rightarrow X_3 X_2$

$X_5 \rightarrow X_4 X_3$



Grammar Compressed Strings: Straight-Line Programs

Definition

A straight-line program (aka straight-line grammar) –
a context-free grammar generating a single string.

$X_0 \to \mathtt{b}$

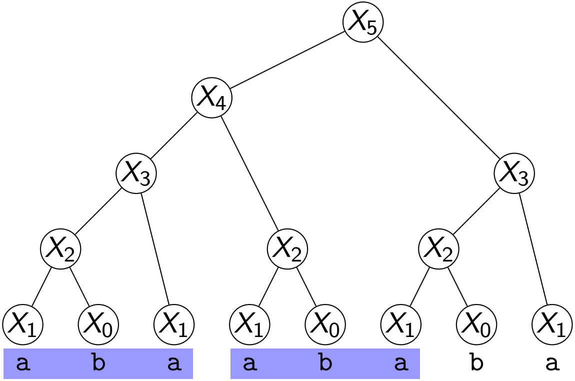
$X_1 \to \mathtt{a}$

$X_2 \to X_1 X_0$

$X_3 \to X_2 X_1$

$X_4 \to X_3 X_2$

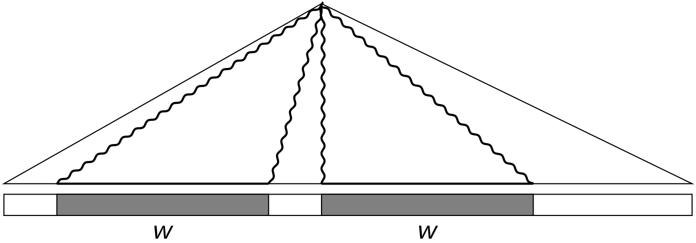
$X_5 \to X_4 X_3$



Locally Consistent Parsing

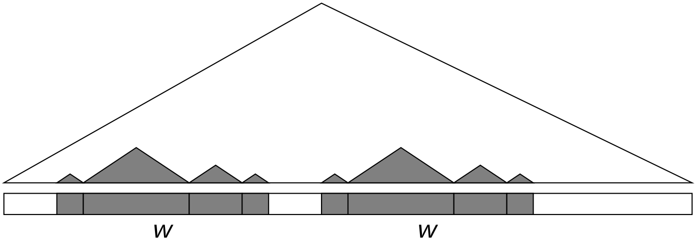
Locally Consistent Parsing: equal fragments are parsed similarly.

Locally Consistent Parsing: equal fragments are parsed similarly.



Locally Consistent Parsing

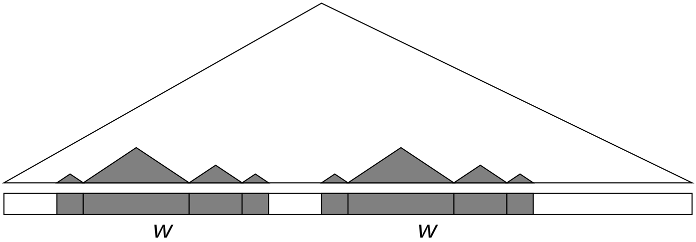
Locally Consistent Parsing: equal fragments are parsed similarly.



- Some nodes in the parse tree of w are preserved no matter the context.

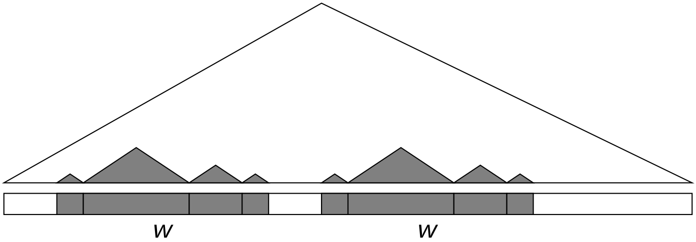
Locally Consistent Parsing

Locally Consistent Parsing: equal fragments are parsed similarly.



- Some nodes in the parse tree of w are preserved no matter the context.
- Topmost such nodes form a small layer: $\mathcal{O}(\log n)$ nodes.

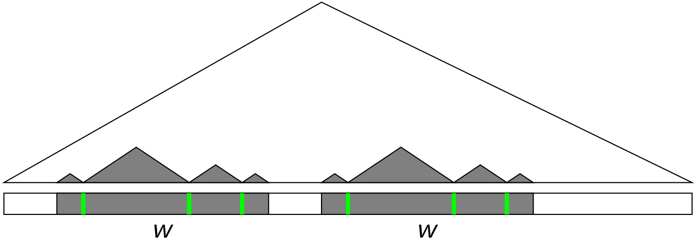
Locally Consistent Parsing: equal fragments are parsed similarly.



- Some nodes in the parse tree of w are preserved no matter the context.
- Topmost such nodes form a small layer: $\mathcal{O}(\log n)$ nodes.
- These nodes define $\mathcal{O}(\log n)$ breakpoints for each substring; they partition it into phrases.

Locally Consistent Parsing

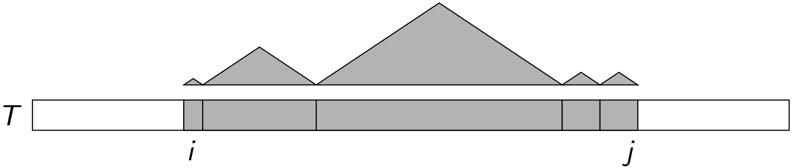
Locally Consistent Parsing: equal fragments are parsed similarly.



- Some nodes in the parse tree of w are preserved no matter the context.
- Topmost such nodes form a small layer: $\mathcal{O}(\log n)$ nodes.
- These nodes define $\mathcal{O}(\log n)$ breakpoints for each substring; they partition it into phrases.

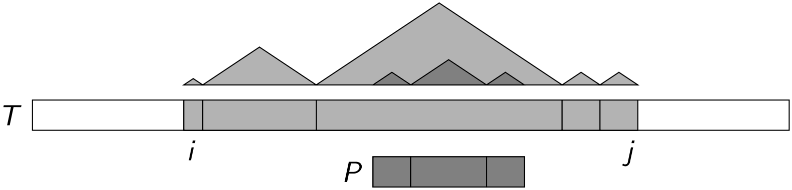
COUNT(i, j)

We have the following property for occurrences of P in $T[i\,..\,j]$.



COUNT(i, j)

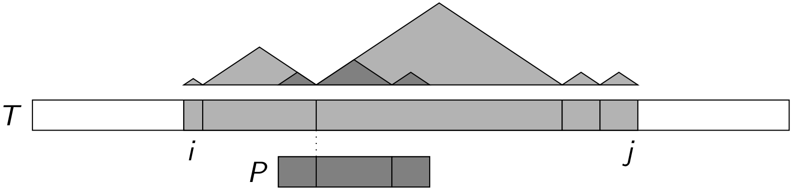
We have the following property for occurrences of P in $T[i \mathinner{.\,.} j]$.



Either fully contained in a phrase – we can precompute all such counts in a bottom-up manner.

COUNT(i, j)

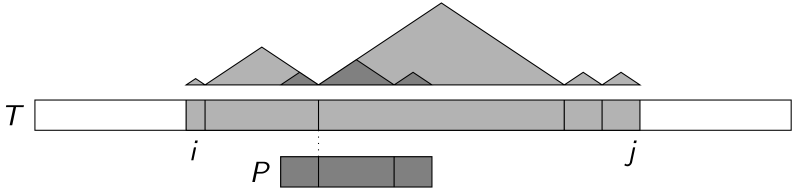
We have the following property for occurrences of P in $T[i\,..\,j]$.



Or a breakpoint of P is aligned with a breakpoint of $T[i\,..\,j]$, an anchor.

$\textsc{Count}(i, j)$

We have the following property for occurrences of P in $T[i \,..\, j]$.



Or a breakpoint of P is aligned with a breakpoint of $T[i \,..\, j]$, an anchor.

Breakpoints-Anchor IDM

- Input: T, $\mathcal{D}$ and a set B_P of breakpoints for each $P \in \mathcal{D}$.
- Query: $\textsc{Count}_\alpha(i, j)$: the number of fragments of $T[i \,..\, j]$ that match some $P \in \mathcal{D}$ and some breakpoint from B_P is aligned with anchor α.

Breakpoints-Anchor IDM and Wrap-up

Breakpoints-Anchor IDM

- Space and preprocessing time: $\tilde{\mathcal{O}}(n + \sum_{P \in \mathcal{D}} |B_P|)$
- Query time: $\mathcal{O}(\log n / \log \log n)$

Breakpoints-Anchor IDM and Wrap-up

Breakpoints-Anchor IDM

- Space and preprocessing time: $\tilde{\mathcal{O}}(n + \sum_{P \in \mathcal{D}} |B_P|)$
- Query time: $\mathcal{O}(\log n / \log \log n)$

$|B_P| = \mathcal{O}(\log n)$ for all $P \in \mathcal{D}$.

Breakpoints-Anchor IDM and Wrap-up

Breakpoints-Anchor IDM

- Space and preprocessing time: $\tilde{\mathcal{O}}(n + \sum_{P \in \mathcal{D}} |B_P|)$
- Query time: $\mathcal{O}(\log n / \log \log n)$

$|B_P| = \mathcal{O}(\log n)$ for all $P \in \mathcal{D}$.
We need to ask $\mathcal{O}(\log n)$ such queries; one for each anchor α.

Breakpoints-Anchor IDM and Wrap-up

Breakpoints-Anchor IDM

- Space and preprocessing time: $\tilde{\mathcal{O}}(n + \sum_{P \in \mathcal{D}} |B_P|)$
- Query time: $\mathcal{O}(\log n / \log \log n)$

$|B_P| = \mathcal{O}(\log n)$ for all $P \in \mathcal{D}$.
We need to ask $\mathcal{O}(\log n)$ such queries; one for each anchor α.

$\text{COUNT}(i, j)$

- Space and preprocessing time: $\tilde{\mathcal{O}}(n + d)$
- Query time: $\mathcal{O}(\log^2 n / \log \log n)$

Breakpoints-Anchor IDM and Wrap-up

Breakpoints-Anchor IDM

- Space and preprocessing time: $\tilde{\mathcal{O}}(n + \sum_{P \in \mathcal{D}} |B_P|)$
- Query time: $\mathcal{O}(\log n / \log \log n)$

$|B_P| = \mathcal{O}(\log n)$ for all $P \in \mathcal{D}$.
We need to ask $\mathcal{O}(\log n)$ such queries; one for each anchor α.

$\textsc{Count}(i, j)$

- Space and preprocessing time: $\tilde{\mathcal{O}}(n + d)$
- Query time: $\mathcal{O}(\log^2 n / \log \log n)$

Note on parsing

Simplified view of the parsing – the properties above are too good to be true. We use the parsing from the recompression technique due to Artur Jeż, and its interpretation by Tomohiro I.

Follow up work on CountDistinct(i, j):

Preprocessing time	**Space**	**Query time**	**Variant**
$\tilde{\mathcal{O}}(n + d)$	$\tilde{\mathcal{O}}(n + d)$	$\tilde{\mathcal{O}}(1)$	2-approx
$\tilde{\mathcal{O}}(n^2/m + d)$	$\tilde{\mathcal{O}}(n^2/m^2 + d)$	$\tilde{\mathcal{O}}(m)$	exact
$\tilde{\mathcal{O}}(nd/m + d)$	$\tilde{\mathcal{O}}(nd/m + d)$	$\tilde{\mathcal{O}}(m)$	exact

Table: m is an arbitrary parameter.

The End

Query	Preprocessing time	Space	Query time
$\textsc{Exists}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
$\textsc{Report}(i,j)$	$\mathcal{O}(n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(1+\lvert output \rvert)$
$\textsc{ReportDistinct}(i,j)$	$\mathcal{O}(n\log n+d)$	$\mathcal{O}(n+d)$	$\mathcal{O}(\log n+\lvert output \rvert)$
$\textsc{Count}(i,j)$	$\mathcal{O}(\frac{n\log n}{\log\log n}+d\log^{3/2} n)$	$\mathcal{O}(n+d\log n)$	$\mathcal{O}(\frac{\log^2 n}{\log\log n})$

Thank you! Questions?

References

- O. Keller, T. Kopelowitz, S. Landau Feibish, M. Lewenstein: Generalized substring compression, Theor. Comput. Sci. (2014).
- T. Kociumaka, J. Radoszewski, W. Rytter, T. Waleń: Internal Pattern Matching Queries in a Text and Applications, SODA 2015.
- V. Mäkinen, G. Navarro: Position-Restricted Substring Searching, LATIN 2006.
- S. Muthukrishnan: Efficient algorithms for document retrieval problems, SODA 2002.
- A. Jeż: Recompression: A Simple and Powerful Technique for Word Equations, J. ACM (2016).
- A. Jeż: Faster Fully Compressed Pattern Matching by Recompression, ACM Trans. Algorithms (2015).
- T. I: Longest Common Extensions with Recompression, CPM 2017.
- P. Charalampopoulos, T. Kociumaka, M. Mohamed, J. Radoszewski, W. Rytter, J. Straszyński, T. Waleń, W. Zuba: Counting Distinct Patterns in Internal Dictionary Matching, CPM 2020.