Internal Pattern Matching Queries in a Text and Applications

Tomasz Kociumaka Jakub Radoszewski
Wojciech Rytter Tomasz Waleń

University of Warsaw, Poland

SODA 2015
San Diego, California, USA
January 4, 2015

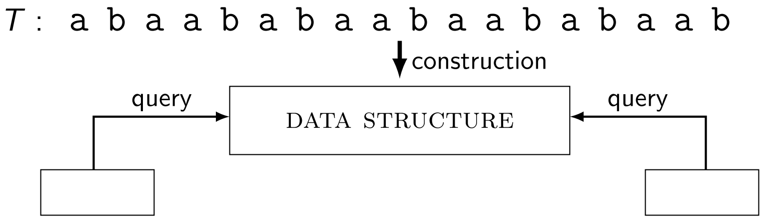
T : a b a a b a b a a b a a b a b a a b

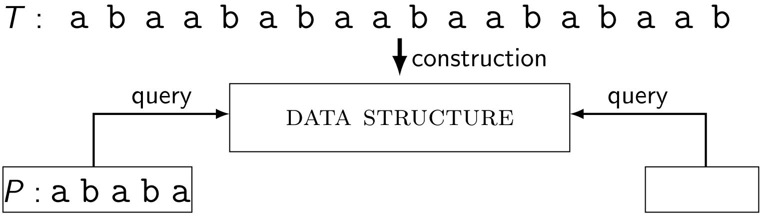
T : `a b a a b a b a a b a a b a b a a b`

↓ construction

DATA STRUCTURE

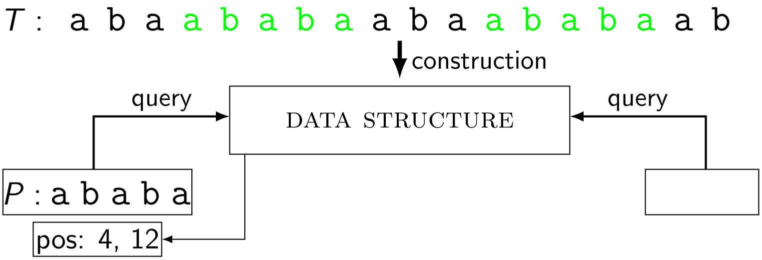
Classic Data Structures for Text Processing





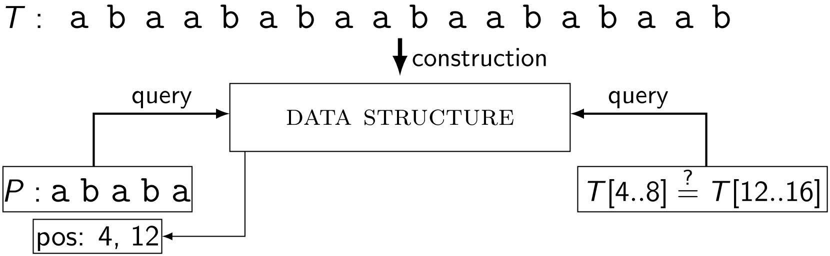
Indexing Subwords

Given a pattern P, find all occurrences of P in T.



Indexing Subwords

Given a pattern P, find all occurrences of P in T.



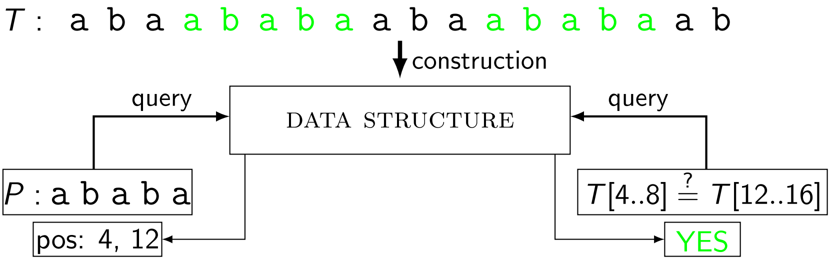
Indexing Subwords

Given a pattern P, find all occurrences of P in T.

Subword Equality

Given two subwords x, y of T, decide whether $x = y$.

Classic Data Structures for Text Processing

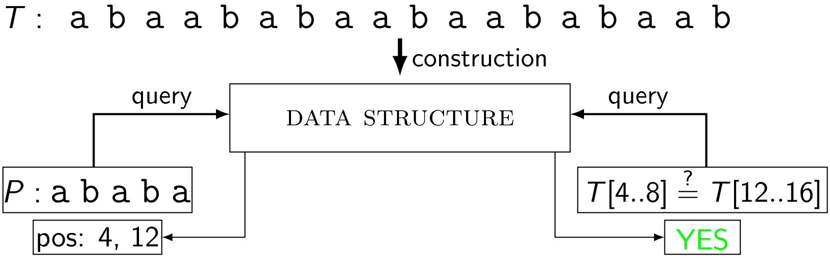


Indexing Subwords

Given a pattern P, find all occurrences of P in T.

Subword Equality

Given two subwords x, y of T, decide whether $x = y$.



Indexing Subwords

Given a pattern P, find all occurrences of P in T.

Subword Equality

Given two subwords x, y of T, decide whether $x = y$.

Indexing queries:

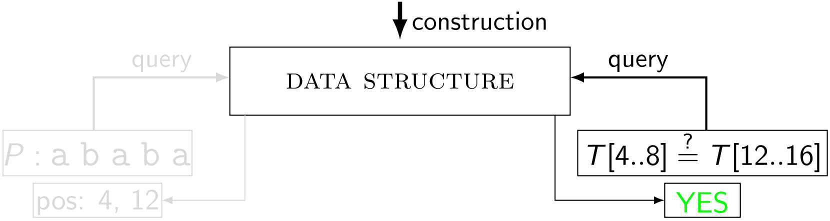
- a string is part of the query input.

Internal queries:

- concern subwords of T only (identified by positions).

Classic Data Structures for Text Processing

T : a b a a b a b a a b a a b a b a a b



Indexing Subwords

Given a pattern P, find all occurrences of P in T.

Indexing queries:

- a string is part of the query input.

Subword Equality

Given two subwords x, y of T, decide whether $x = y$.

Internal queries: $\longleftarrow$ this talk

- concern subwords of T only (identified by positions).

Internal queries

Solve a problem for inputs being subwords of a given text T.

Internal Queries

Internal queries

Solve a problem for inputs being subwords of a given text T.

- $\mathcal{O}(1)$ query input size allows for $\mathcal{O}(1)$ query time,
  - $\Omega(|P|)$ required if the "pattern" P is in the input.

Internal Queries

Internal queries

Solve a problem for inputs being subwords of a given text T.

- $\mathcal{O}(1)$ query input size allows for $\mathcal{O}(1)$ query time,
  - $\Omega(|P|)$ required if the "pattern" P is in the input.

Motivation:

- All input data must be known in advance

Internal Queries

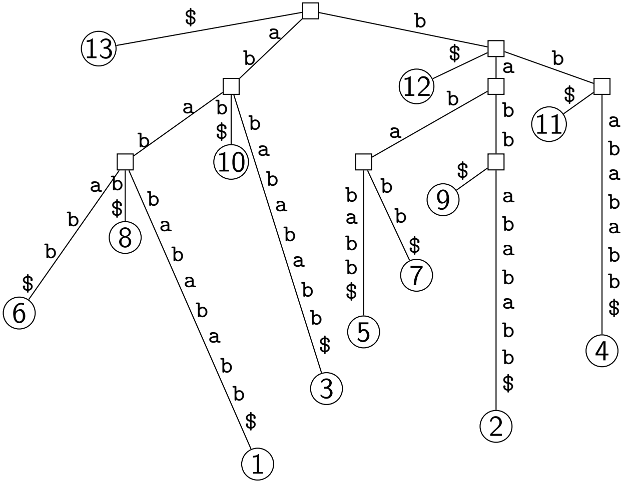
Internal queries

Solve a problem for inputs being subwords of a given text T.

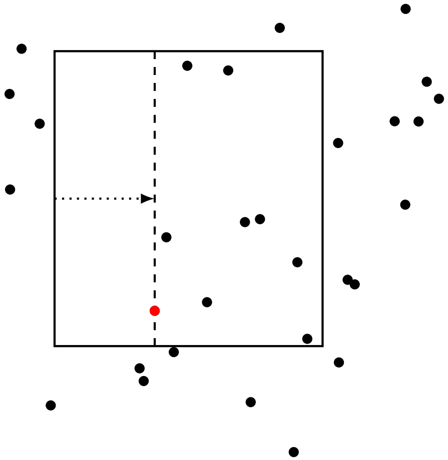
- $\mathcal{O}(1)$ query input size allows for $\mathcal{O}(1)$ query time,
  - $\Omega(|P|)$ required if the "pattern" P is in the input.

Motivation:

- All input data must be known in advance
- Primitives for algorithms and data structures
  - linear size and $\mathcal{O}(1)$ query time crucial for applicability,
  - efficient construction important for applications in algorithms.


$
ab
b
13
ab
b$
babababb$
10
3
abb$
b$
babababb$
6
8
1
$
a
b
12
ba
bb
babb$
bb$
5
7
$
abababb$
9
2
$
abababb$
11
4

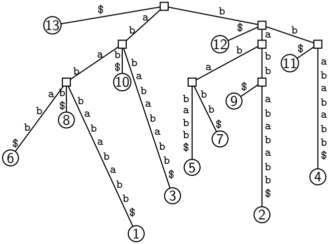
Suffix tree



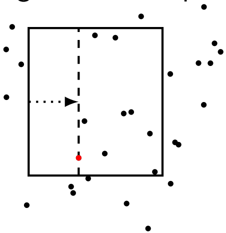
Range successor queries

Internal Queries: Previous Techniques

Suffix tree

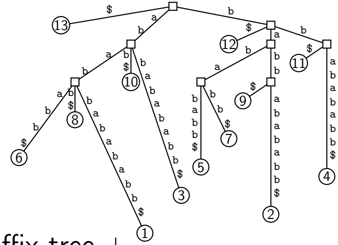


Range successor queries



Internal Queries: Previous Techniques

Suffix tree



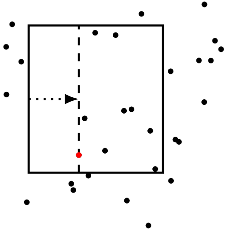
Suffix tree +
problem-specific tools:

- Longest common prefix
- Min/Max suffix

\+ efficient, often $\mathcal{O}(1)$ time queries with $\mathcal{O}(n)$ space,

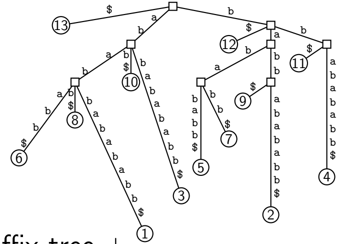
– limited use.

Range successor queries



Internal Queries: Previous Techniques

Suffix tree



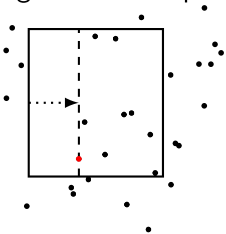
Suffix tree +
problem-specific tools:

- Longest common prefix
- Min/Max suffix

\+ efficient, often $\mathcal{O}(1)$ time queries with $\mathcal{O}(n)$ space,

– limited use.

Range successor queries



Suffix tree +
orthogonal range queries:

- Pattern matching-related
- Period queries

\+ wide applicability,

– lower bounds for query time, slow construction.

Fact

Let $Occ(x,y)$ be the positions in y where occurrences of x start. $Occ(x,y)$ forms an arithmetic progression if $|y| \le 2|x|$.

x : $\boxed{\texttt{a b a b a b a}}$ y : $\boxed{\texttt{b b a b a b a b a b a b a a}}$

$$Occ(x,y) =$$

Fact

Let $Occ(x, y)$ be the positions in y where occurrences of x start. $Occ(x, y)$ forms an arithmetic progression if $|y| \leq 2|x|$.

x : $\boxed{\texttt{abababa}}$ y : $\boxed{\texttt{bbabababababaa}}$

$$Occ(x, y) = \{3,$$

Fact

Let $Occ(x, y)$ be the positions in y where occurrences of x start. $Occ(x, y)$ forms an arithmetic progression if $|y| \le 2|x|$.

x : a b a b a b a y : b b a b a b a b a b a b a a

$$Occ(x, y) = \{3, 5,$$

Fact

Let $Occ(x,y)$ be the positions in y where occurrences of x start.
$Occ(x,y)$ forms an arithmetic progression if $|y| \le 2|x|$.

x : `ababab a` y : `bbababababababaa`

$$Occ(x,y) = \{3,5,7\}$$

Fact

Let $Occ(x,y)$ be the positions in y where occurrences of x start. $Occ(x,y)$ forms an arithmetic progression if $|y| \le 2|x|$.

T : x : a b a b a b a y : b b a b a b a b a b a b a a

$$Occ(x,y) = \{3,5,7\}$$

Problem (INTERNAL PATTERN MATCHING QUERIES)

Given subwords x and y of T with $|y| \le 2|x|$, report all occurrences of x in y (as an arithmetic progression).

Fact

Let $Occ(x, y)$ be the positions in y where occurrences of x start. $Occ(x, y)$ forms an arithmetic progression if $|y| \leq 2|x|$.

T : x : a b a b a b a y : b b a b a b a b a b a b a a

$$Occ(x, y) = \{3, 5, 7\}$$

Problem (Internal Pattern Matching Queries)

Given subwords x and y of T with $|y| \leq 2|x|$, report all occurrences of x in y (as an arithmetic progression).

- $\Theta\left(\frac{|y|}{|x|}\right)$ space is necessary to encode $Occ(x, y)$,
- $Occ(x, y)$ can be computed with $\Theta\left(\frac{|y|}{|x|}\right)$ IPM Queries.

IPM Queries: Our Results

Problem (Internal Pattern Matching Queries)

Given subwords x and y of T with $|y| \le 2|x|$, report all occurrences of x in y (as an arithmetic progression).

Theorem

IPM Queries *can be answered in $\mathcal{O}(1)$ time by a data structure of size $\mathcal{O}(n)$, which can be constructed in $\mathcal{O}(n)$ time.*

IPM Queries: Our Results

Problem (Internal Pattern Matching Queries)

Given subwords x and y of T with $|y| \leq 2|x|$, report all occurrences of x in y (as an arithmetic progression).

Theorem

IPM Queries *can be answered in $\mathcal{O}(1)$ time by a data structure of size $\mathcal{O}(n)$, which can be constructed in $\mathcal{O}(n)$ time.*

Technical assumptions:

- Word-RAM model with word-size $w = \Omega(\log n)$.

Problem (Internal Pattern Matching Queries)

Given subwords x and y of T with $|y| \le 2|x|$, report all occurrences of x in y (as an arithmetic progression).

Theorem

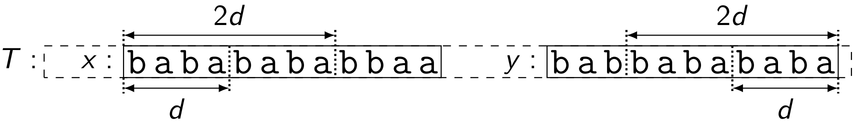
IPM Queries *can be answered in $\mathcal{O}(1)$ time by a data structure of size $\mathcal{O}(n)$, which can be constructed in $\mathcal{O}(n)$ time.*

Technical assumptions:

- Word-RAM model with word-size $w = \Omega(\log n)$.
- Construction:
  - randomized (Las Vegas, expected time),

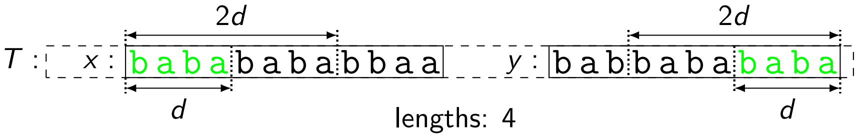
Applications

Prefix-Suffix and Period Queries



Problem (Prefix-Suffix Queries)

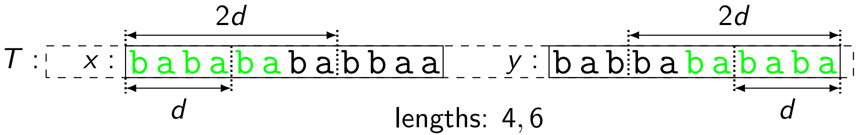
Given subwords x and y of T and $d \in \mathbb{N}$, report all prefixes of x of length between d and $2d$ that are also suffixes of y.



Problem (Prefix-Suffix Queries)

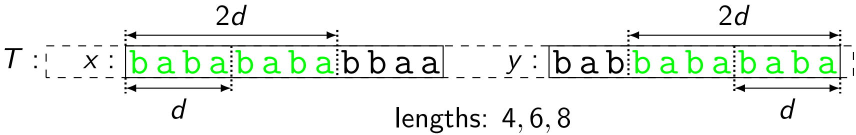
Given subwords x and y of T and $d \in \mathbb{N}$, report all prefixes of x of length between d and $2d$ that are also suffixes of y.

Prefix-Suffix and Period Queries



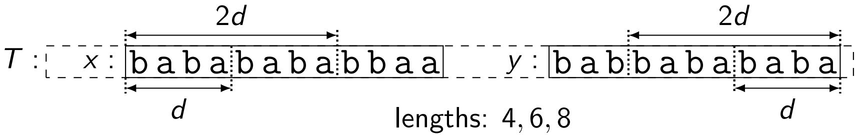
Problem (Prefix-Suffix Queries)

Given subwords x and y of T and $d \in \mathbb{N}$, report all prefixes of x of length between d and $2d$ that are also suffixes of y.



Problem (PREFIX-SUFFIX QUERIES)

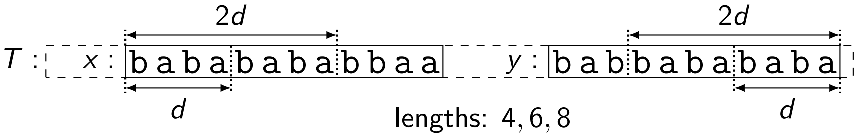
Given subwords x and y of T and $d \in \mathbb{N}$, report all prefixes of x of length between d and $2d$ that are also suffixes of y.



Problem (Prefix-Suffix Queries)

Given subwords x and y of T and $d \in \mathbb{N}$, report all prefixes of x of length between d and $2d$ that are also suffixes of y.

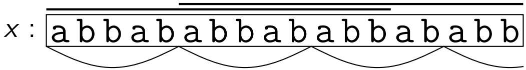
- $\mathcal{O}(1)$ time using IPM Queries.



Problem (PREFIX-SUFFIX QUERIES)

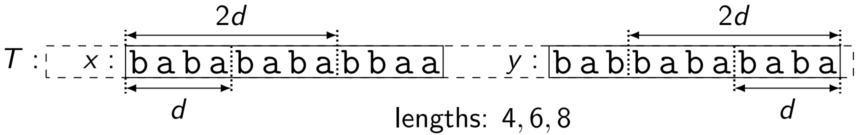
Given subwords x and y of T and $d \in \mathbb{N}$, report all prefixes of x of length between d and $2d$ that are also suffixes of y.

- $\mathcal{O}(1)$ time using IPM QUERIES.



Problem (PERIOD QUERIES)

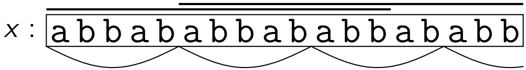
Given a subword x of T, report all periods of x.



Problem (Prefix-Suffix Queries)

Given subwords x and y of T and $d \in \mathbb{N}$, report all prefixes of x of length between d and $2d$ that are also suffixes of y.

- $\mathcal{O}(1)$ time using IPM Queries.



Problem (Period Queries)

Given a subword x of T, report all periods of x.

- $\mathcal{O}(\log |x|)$ time using IPM Queries.

Applications in Substring Compression

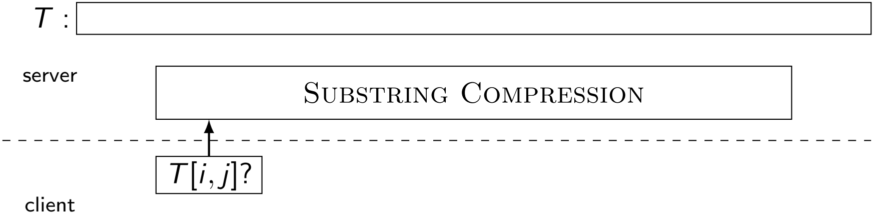
T :

server

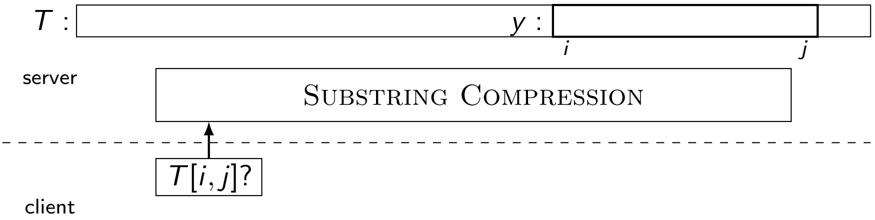
SUBSTRING COMPRESSION

client

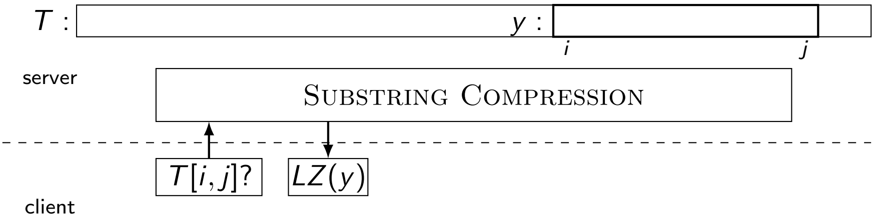
Applications in Substring Compression



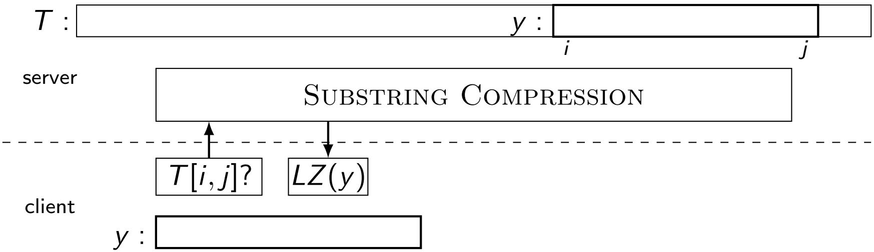
Applications in Substring Compression



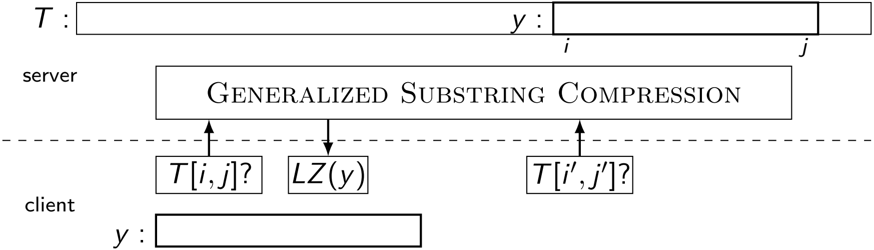
Applications in Substring Compression



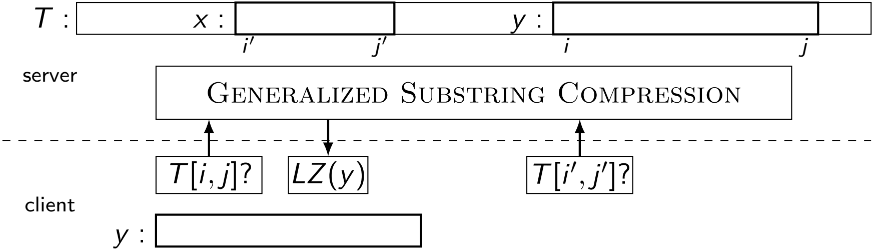
Applications in Substring Compression



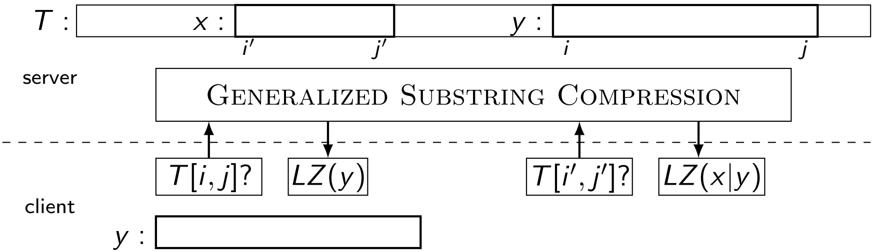
Applications in Substring Compression



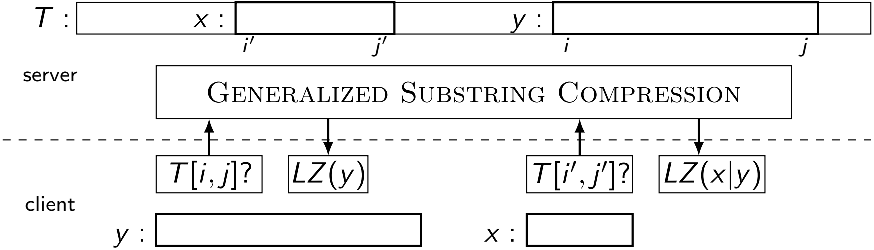
Applications in Substring Compression



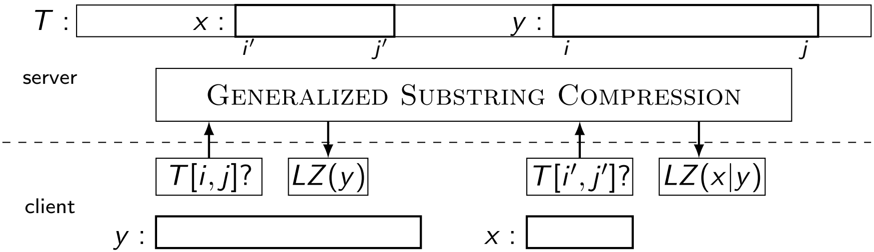
Applications in Substring Compression



Applications in Substring Compression



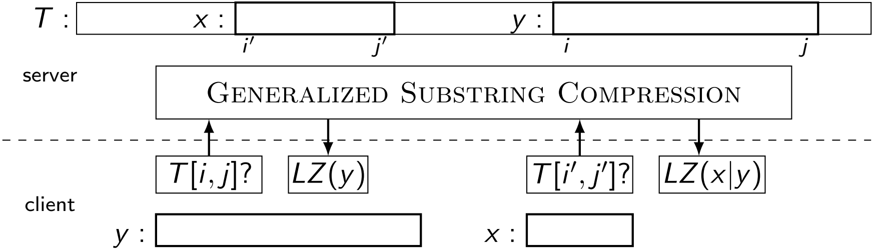
Applications in Substring Compression



Problem (Cormode & Muthukrishnan; SODA 2005)

Given subwords x and y, compute $LZ(x|y)$, i.e., the section of the Lempel-Ziv (LZ77) factorization LZ($y\$x$) that corresponds to x.

Applications in Substring Compression



Problem (Cormode & Muthukrishnan; SODA 2005)

Given subwords x and y, compute $LZ(x|y)$, i.e., the section of the Lempel-Ziv (LZ77) factorization $LZ(y\$x)$ that corresponds to x.

- data structure of $\mathcal{O}(n)$ size,
- query time improved from $\mathcal{O}(C \log \frac{|x|}{C} \log^{\varepsilon} n)$ to $\mathcal{O}(C \log\log \frac{|x|}{C} \log^{\varepsilon} n)$ where C is the output size,
- combines orthogonal range queries with IPM QUERIES.

Other Applications

Problem (CYCLIC EQUIVALENCE QUERIES)

Decide whether given subwords x and y are cyclic shifts of each other ($x = uv$ and $y = vu$ for some strings u and v).

- $\mathcal{O}(1)$ query time using IPM QUERIES.

Problem (Cyclic Equivalence Queries)

Decide whether given subwords x and y are cyclic shifts of each other ($x = uv$ and $y = vu$ for some strings u and v).

- $\mathcal{O}(1)$ query time using IPM Queries.

Definition (δ-subrepetition)

A δ-*subrepetition* is a fragment x such that $\mathrm{per}(x) \leq \frac{|x|}{1+\delta}$ and x cannot be extended (to the left or right) preserving $\mathrm{per}(x)$.

- Improved algorithm for finding δ-subrepetitions in a word:
  - first (expected) linear-time algorithm for $\delta = \Theta(1)$.

Other Applications

Problem (Cyclic Equivalence Queries)

Decide whether given subwords x and y are cyclic shifts of each other ($x = uv$ and $y = vu$ for some strings u and v).

- $\mathcal{O}(1)$ query time using IPM Queries.

Definition (δ-subrepetition)

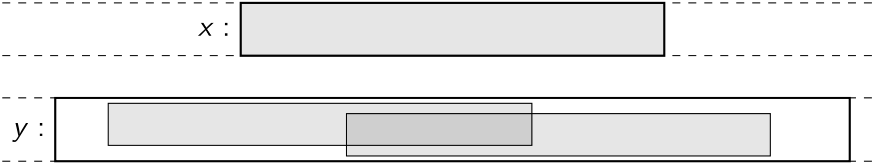
A δ-*subrepetition* is a fragment x such that $\mathrm{per}(x) \le \frac{|x|}{1+\delta}$ and x cannot be extended (to the left or right) preserving $\mathrm{per}(x)$.

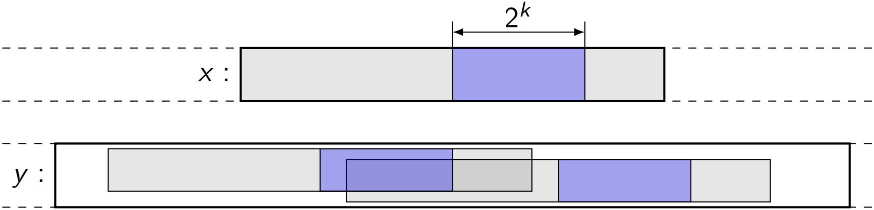
- Improved algorithm for finding δ-subrepetitions in a word:
  - first (expected) linear-time algorithm for $\delta = \Theta(1)$.

- More applications through wavelet suffix trees:
  - substring suffix rank & selection,
  - substring compression using Burrows-Wheeler transform.

High-level Ideas

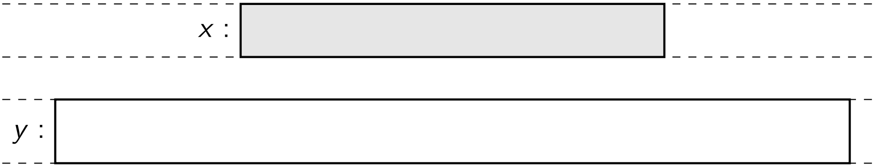
IPM Queries: Outline





Main idea:

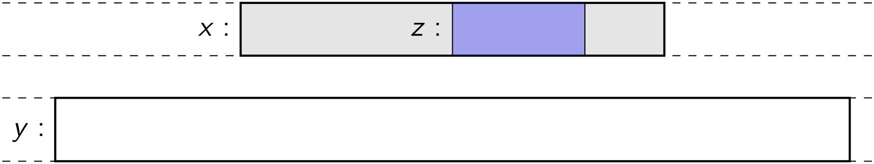
- Design (locally consistent) representative assignment
  - representatives of length 2^k with $\frac{|x|}{4} < 2^k \le \frac{|x|}{2}$,
  - store information about location of representative only.



Main idea:

- Design (locally consistent) representative assignment
  - representatives of length 2^k with $\frac{|x|}{4} < 2^k \leq \frac{|x|}{2}$,
  - store information about location of representative only.

Query algorithm:

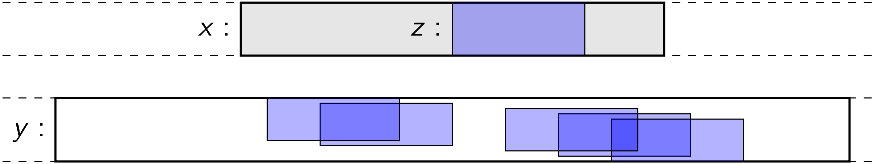


Main idea:

- Design (locally consistent) representative assignment
  - representatives of length 2^k with $\frac{|x|}{4} < 2^k \leq \frac{|x|}{2}$,
  - store information about location of representative only.

Query algorithm:

1. Compute the representative z assigned to x.

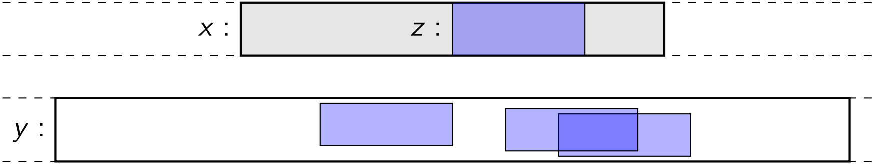


Main idea:

- Design (locally consistent) representative assignment
  - representatives of length 2^k with $\frac{|x|}{4} < 2^k \le \frac{|x|}{2}$,
  - store information about location of representative only.

Query algorithm:

1. Compute the representative z assigned to x.
2. Find occurrences of z within y.

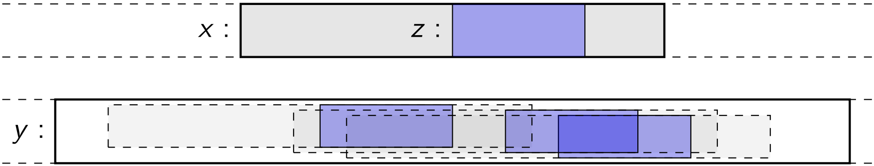


Main idea:

- Design (locally consistent) representative assignment
  - representatives of length 2^k with $\frac{|x|}{4} < 2^k \leq \frac{|x|}{2}$,
  - store information about location of representative only.

Query algorithm:

1. Compute the representative z assigned to x.
2. Find occurrences of z (as representatives) within y.

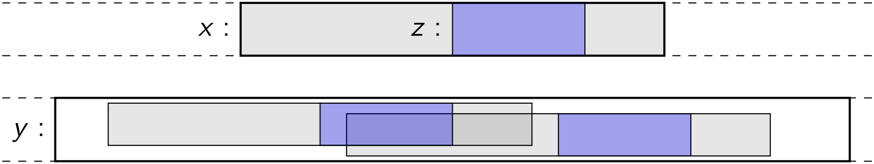


Main idea:

- Design (locally consistent) representative assignment
  - representatives of length 2^k with $\frac{|x|}{4} < 2^k \le \frac{|x|}{2}$,
  - store information about location of representative only.

Query algorithm:

1. Compute the representative z assigned to x.
2. Find occurrences of z (as representatives) within y.
3. Check which occurrences of z extend to occurrences of x.

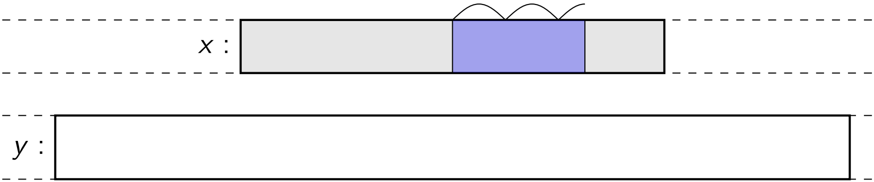


Main idea:

- Design (locally consistent) representative assignment
  - representatives of length 2^k with $\frac{|x|}{4} < 2^k \leq \frac{|x|}{2}$,
  - store information about location of representative only.

Query algorithm:

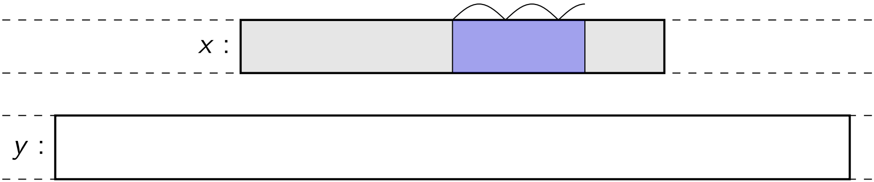
1. Compute the representative z assigned to x.
2. Find occurrences of z (as representatives) within y.
3. Check which occurrences of z extend to occurrences of x.



Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Periodic Representatives

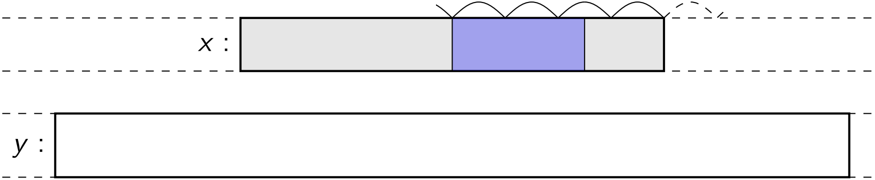


Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

Periodic Representatives



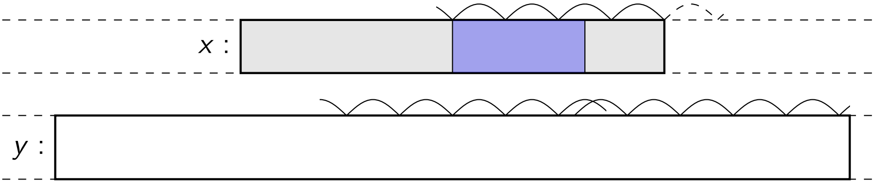
Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

1. Extend z to a run. (Suppose it does not cover x.)

Periodic Representatives



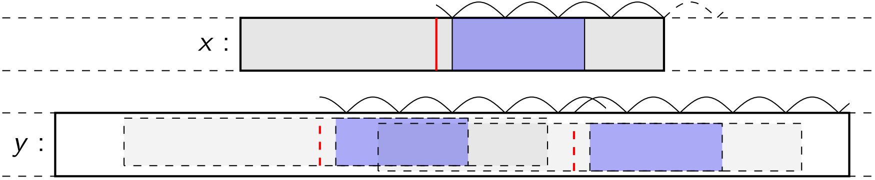
Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

1. Extend z to a run. (Suppose it does not cover x.)
2. Find runs of the same period in y.

Periodic Representatives

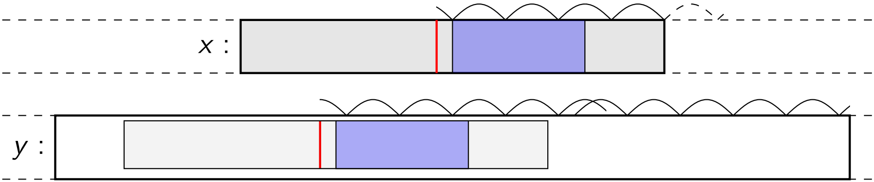


Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

1. Extend z to a run. (Suppose it does not cover x.)
2. Find runs of the same period in y.
3. Generate candidates using run endpoints.



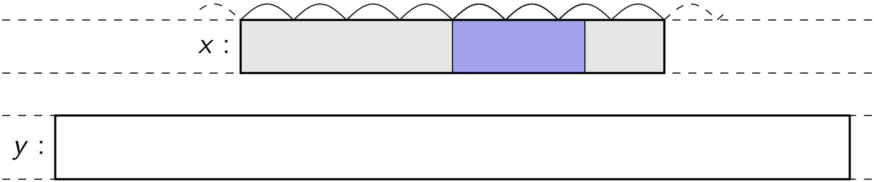
Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

1. Extend z to a run. (Suppose it does not cover x.)
2. Find runs of the same period in y.
3. Generate candidates using run endpoints.
4. Check which candidates are indeed occurrences.

Periodic Representatives



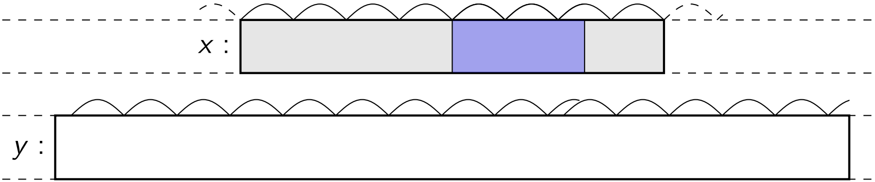
Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

1. Extend z to a run. (Suppose it covers x.)

Periodic Representatives



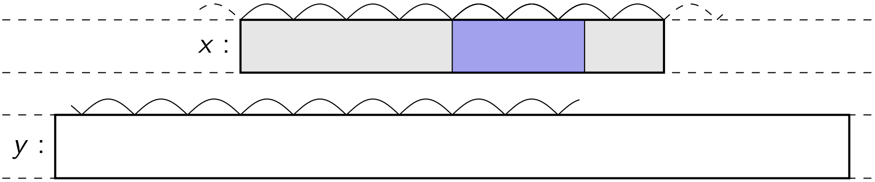
Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

1. Extend z to a run. (Suppose it covers x.)
2. Find runs of the same period in y.

Periodic Representatives



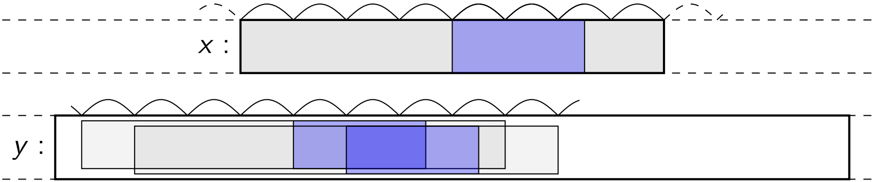
Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

1. Extend z to a run. (Suppose it covers x.)
2. Find runs of the same period in y.
3. Synchronize runs using Lyndon roots.

Periodic Representatives



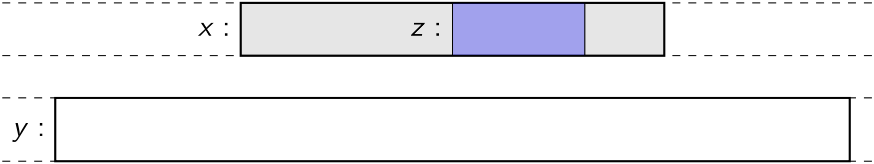
Representative: any periodic subword of appropriate length.

- use combinatorial and algorithmic tools for repetitions.

Query algorithm:

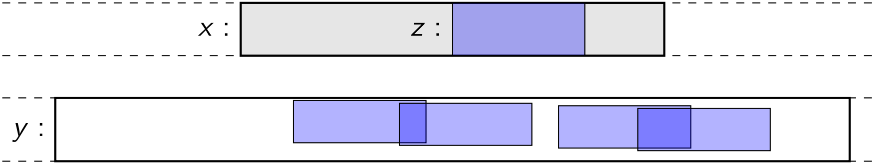
1. Extend z to a run. (Suppose it covers x.)
2. Find runs of the same period in y.
3. Synchronize runs using Lyndon roots.
4. Output occurrences in bulk.

Non-periodic Representatives



Non-periodic representatives:

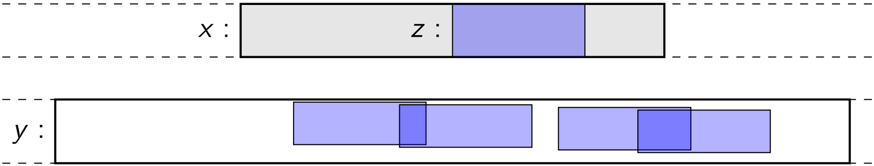
Non-periodic Representatives



Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Non-periodic Representatives

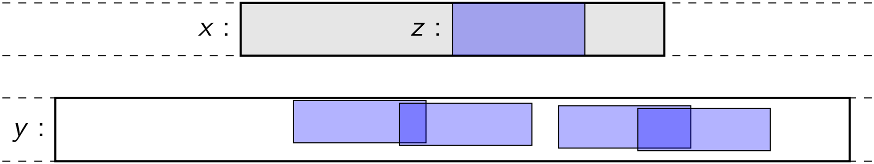


Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

Non-periodic Representatives



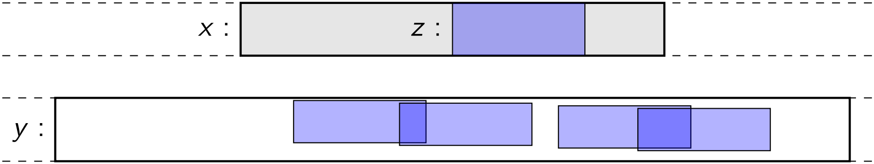
Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

- minimal subword of appropriate length wrt. a random order
  - guaranteed local consistence,

Non-periodic Representatives



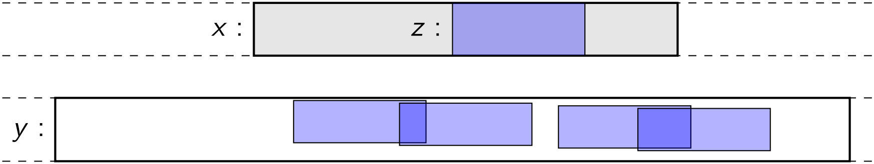
Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

- minimal subword of appropriate length wrt. a random order
  - guaranteed local consistence,
  - neighbouring patterns often share the representative,

Non-periodic Representatives



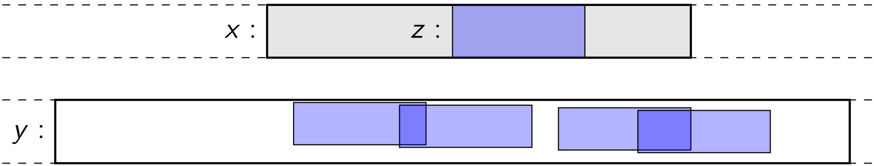
Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

- minimal subword of appropriate length wrt. a random order
  - guaranteed local consistence,
  - neighbouring patterns often share the representative,

Non-periodic Representatives



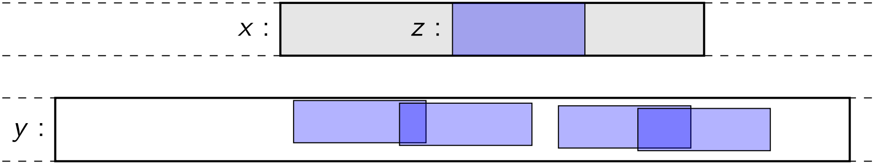
Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

- minimal subword of appropriate length wrt. a random order
  - guaranteed local consistence,
  - neighbouring patterns often share the representative,

Non-periodic Representatives



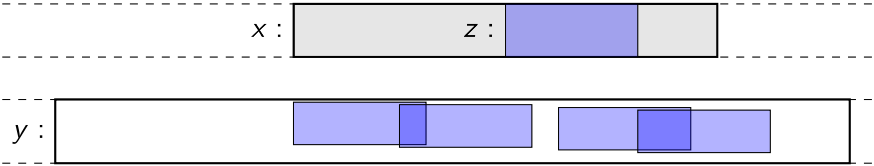
Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

- minimal subword of appropriate length wrt. a random order
  - guaranteed local consistence,
  - neighbouring patterns often share the representative,

Non-periodic Representatives



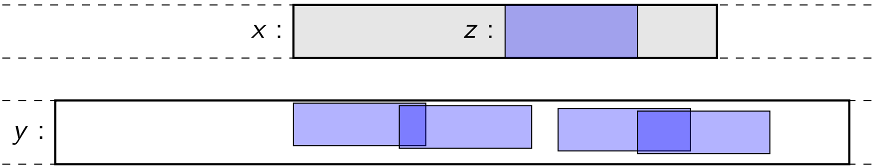
Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

- minimal subword of appropriate length wrt. a random order
  - guaranteed local consistence,
  - neighbouring patterns often share the representative,

Non-periodic Representatives



Non-periodic representatives:

- occurrences of z in y cannot overlap too much;
  - $\mathcal{O}(1)$ occurrences within y.

Representative assignment:

- minimal subword of appropriate length wrt. a random order
  - guaranteed local consistence,
  - neighbouring patterns often share the representative,
  - in total $\mathcal{O}(n)$ representatives with $\mathcal{O}(n)$ occurrences as a representative.

Conclusions & Open Problems

Our results:

- a $\mathcal{O}(n)$-size data structure for IPM QUERIES with $\mathcal{O}(1)$-time queries and $\mathcal{O}(n)$-expected time construction,
- several applications of IPM QUERIES:
  - further internal queries,
  - faster algorithms.

Conclusions & Open Problems

Our results:

- a $\mathcal{O}(n)$-size data structure for IPM Queries with $\mathcal{O}(1)$-time queries and $\mathcal{O}(n)$-expected time construction,
- several applications of IPM Queries:
  - further internal queries,
  - faster algorithms.

Open problems:

- More applications of IPM Queries.

Conclusions & Open Problems

Our results:

- a $\mathcal{O}(n)$-size data structure for IPM Queries with $\mathcal{O}(1)$-time queries and $\mathcal{O}(n)$-expected time construction,
- several applications of IPM Queries:
  - further internal queries,
  - faster algorithms.

Open problems:

- More applications of IPM Queries.
- Design a deterministic construction algorithm. . .
  - or an algorithm running in $\mathcal{O}(n)$ time w.h.p.

Conclusions & Open Problems

Our results:

- a $\mathcal{O}(n)$-size data structure for IPM QUERIES with $\mathcal{O}(1)$-time queries and $\mathcal{O}(n)$-expected time construction,
- several applications of IPM QUERIES:
  - further internal queries,
  - faster algorithms.

Open problems:

- More applications of IPM QUERIES.
- Design a deterministic construction algorithm...
  - or an algorithm running in $\mathcal{O}(n)$ time w.h.p.
- Can shortest periods be computed in $o(\log |x|)$ time?
  - Is there any lower bound for this problem?

Thank you for your attention!