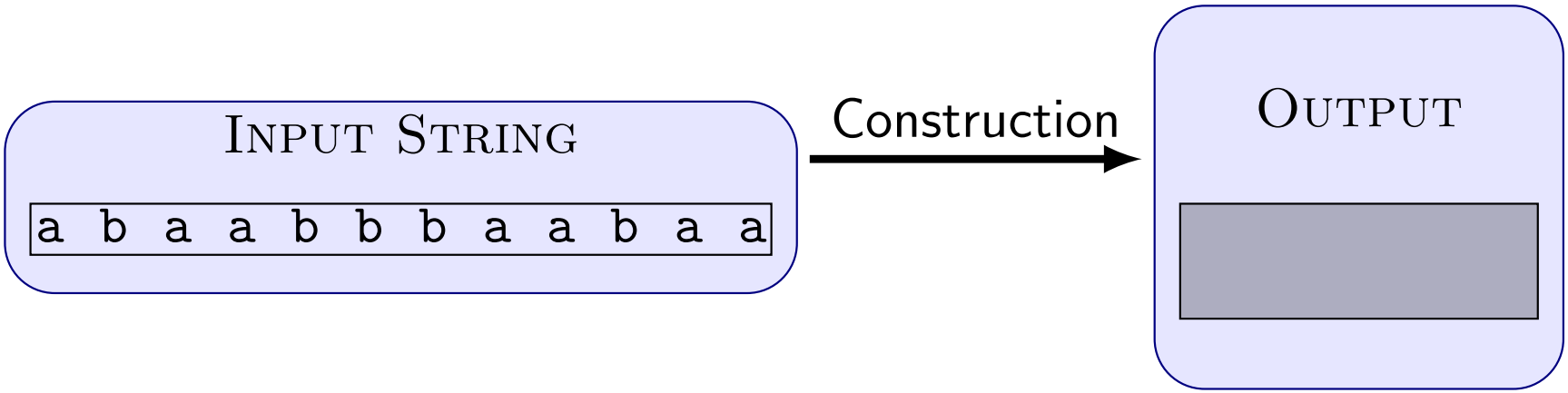
Universal Reconstruction of a String

Paweł Gawrychowski, Tomasz Kociumaka, Jakub Radoszewski,
Wojciech Rytter and Tomasz Waleń

University of Warsaw, Poland

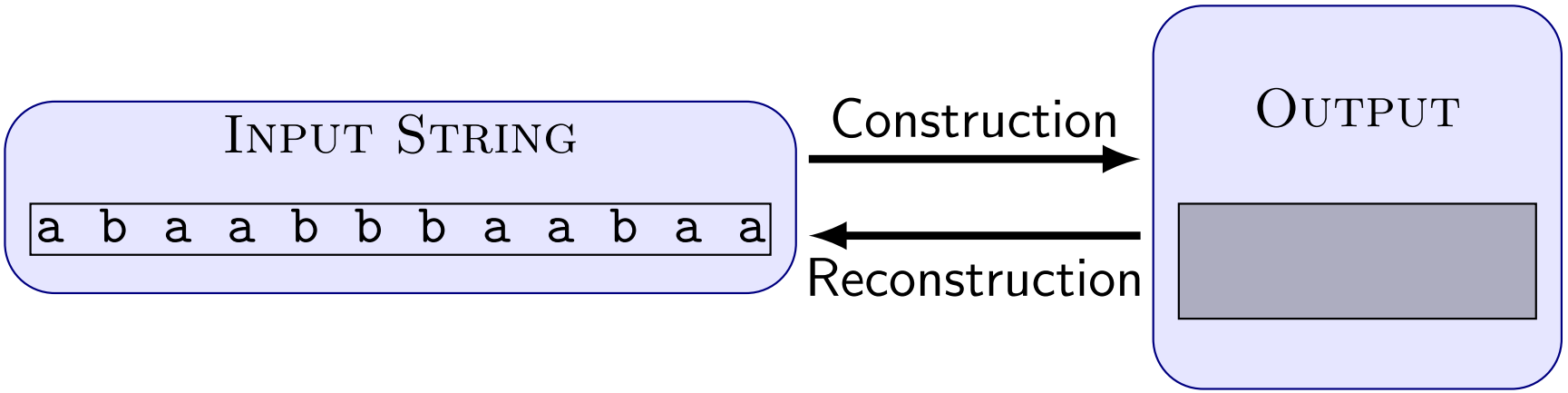
WADS 2015
Victoria, BC, Canada
August 5, 2015

Reconstruction Problems



Construction (typically in text algorithms):
Given a string w compute $\mathbf{F}(w)$.

Reconstruction Problems



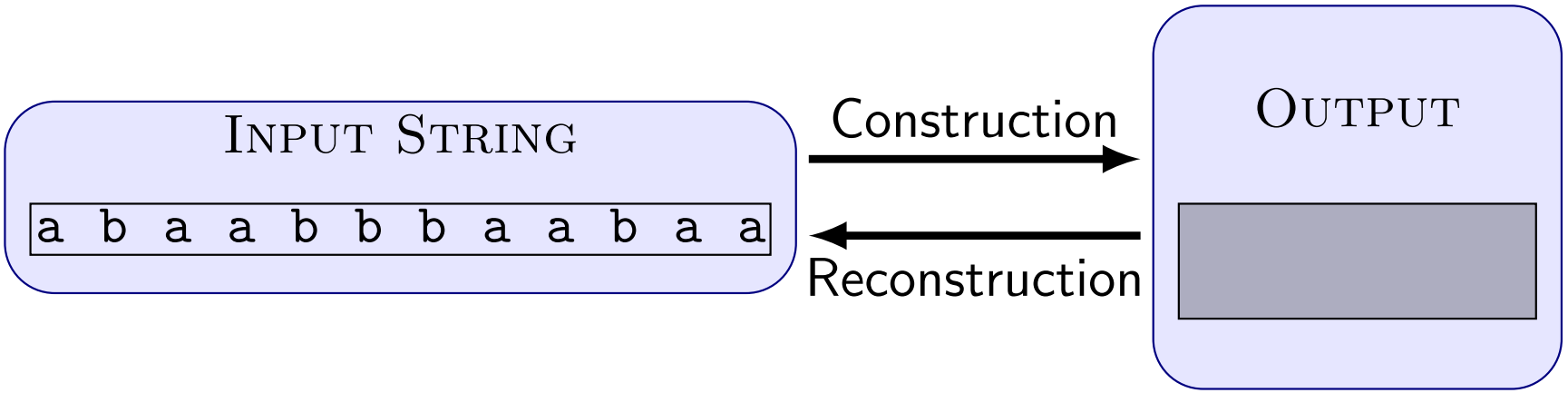
Construction (typically in text algorithms):

Given a string w compute $\mathbf{F}(w)$.

Reconstruction:

Given F find a string w satisfying $F = \mathbf{F}(w)$.

Reconstruction Problems



Construction (typically in text algorithms):

Given a string w compute $\mathbf{F}(w)$.

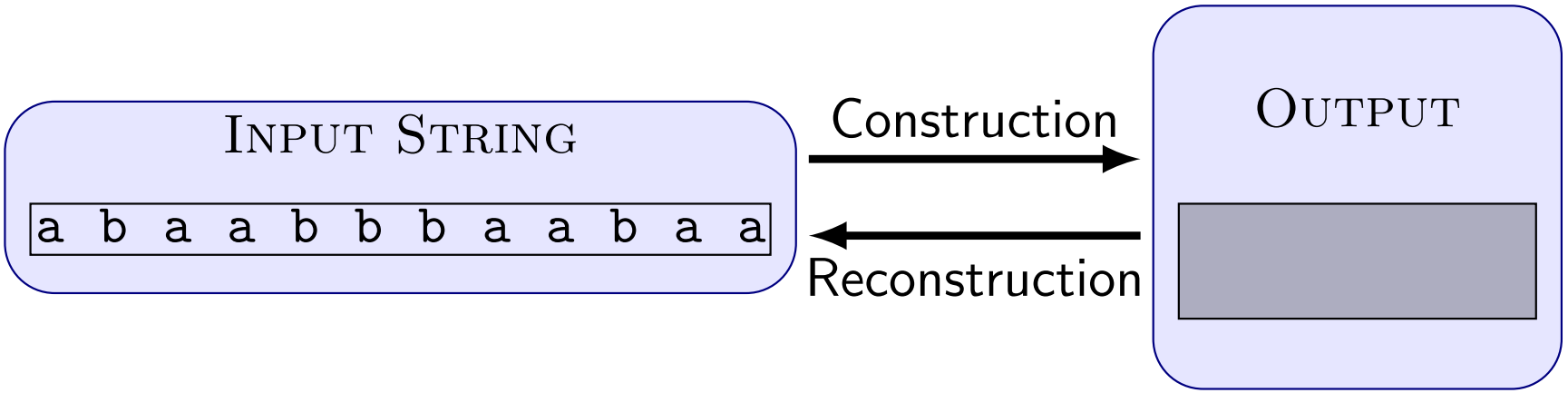
Reconstruction:

Given F find a string w satisfying $F = \mathbf{F}(w)$.

Verification:

Given F decide if there exists w satisfying $F = \mathbf{F}(w)$.

Reconstruction Problems



Construction (typically in text algorithms):

Given a string w compute $\mathbf{F}(w)$.

Reconstruction:

Given F find a string w satisfying $F = \mathbf{F}(w)$.

Verification:

Given F decide if there exists w satisfying $F = \mathbf{F}(w)$.

Main motivation:

- develop a good understanding of $\mathbf{F}$,
- aid computer experiments for combinatorics on words.

Systems of Substring Equations

A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a solution of E of length n (a string w of length n satisfying E).

Systems of Substring Equations

A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a solution of E of length n (a string w of length n satisfying E).

$$E = \{ \qquad\qquad n = 9 \qquad\qquad \}$$

w :	?	?	?	?	?	?	?	?	?
	0	1	2	3	4	5	6	7	8

Systems of Substring Equations

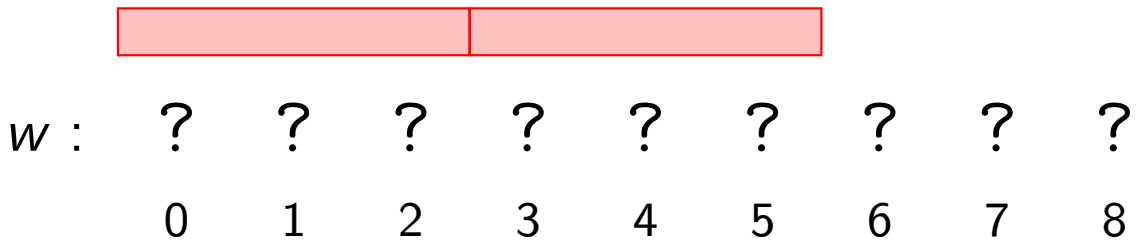
A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a solution of E of length n (a string w of length n satisfying E).

$$n = 9$$

$$E = \{w[0..2] = w[3..5] \}$$



Systems of Substring Equations

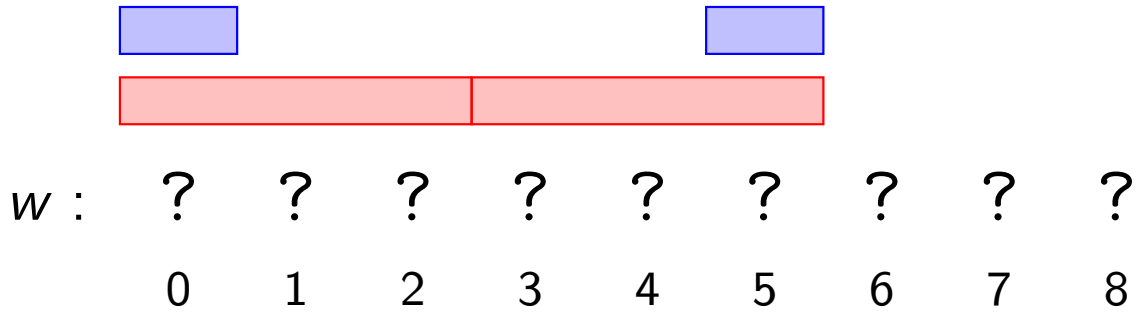
A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a solution of E of length n (a string w of length n satisfying E).

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5] \}$$



Systems of Substring Equations

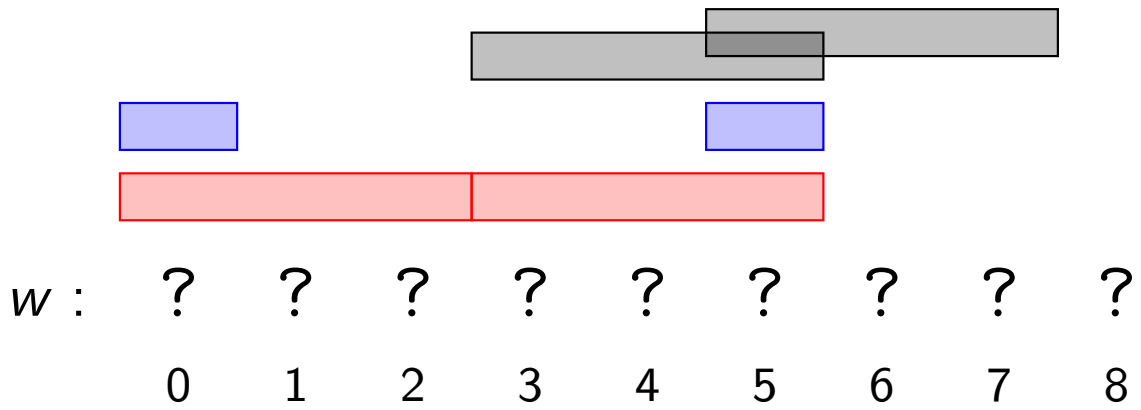
A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a solution of E of length n (a string w of length n satisfying E).

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



Systems of Substring Equations

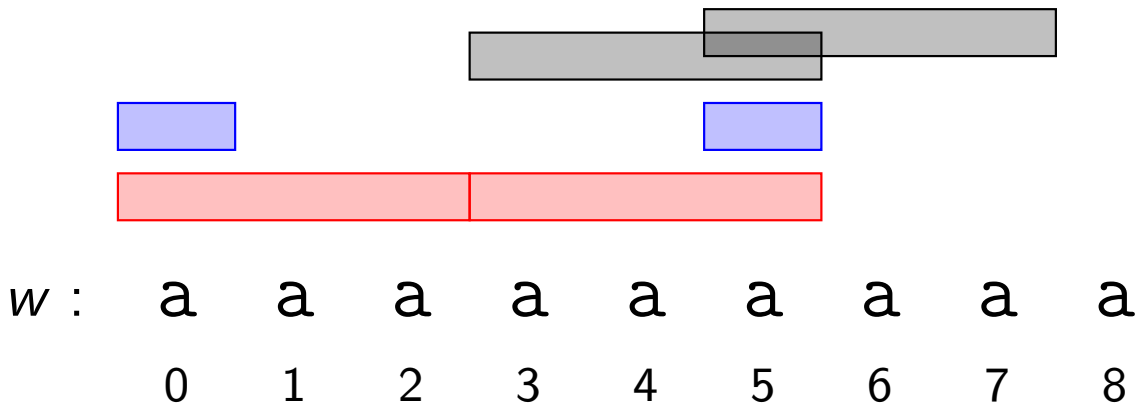
A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a solution of E of length n (a string w of length n satisfying E).

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



Systems of Substring Equations

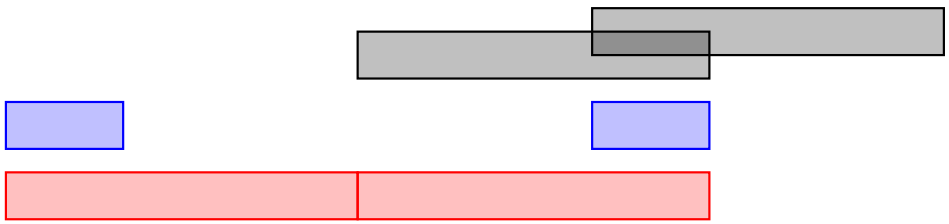
A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a generic solution of E of length n (a string w of length n over max. alph. satisfying E).

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



$\Phi(E) = w$:

a	b	a	a	b	a	b	a	c
0	1	2	3	4	5	6	7	8

Systems of Substring Equations

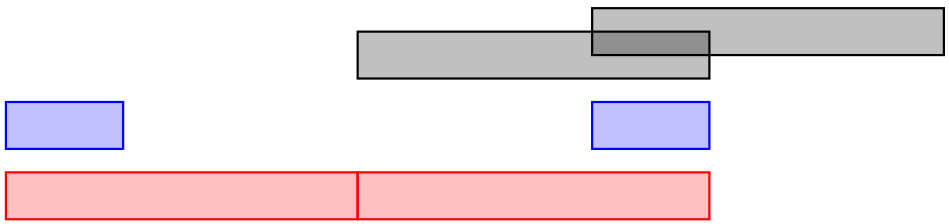
A substring equation is a constraint of the form $w[p..q] = w[p'..q']$ where $q - p = q' - p'$.

Problem

Given a system E of substring equations find a generic solution of E of length n (a string w of length n over max. alph. satisfying E).

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



$\Phi(E) = w$:

a	b	a	a	b	a	b	a	c
0	1	2	3	4	5	6	7	8

Solutions of E = images of $\Phi(E)$ via letter-to-letter morphisms.

Our Results

Theorem

A generic solution of E can be computed in $\mathcal{O}(n + |E|)$ time.

Our Results

Theorem

A generic solution of E can be computed in $\mathcal{O}(n + |E|)$ time.

Several structures can be characterized using substring equations:

- border table (Morris-Pratt failure table),
- PREF table,
- maximal repetitions (runs),
- maximal palindromes.

Our Results

Theorem

A generic solution of E can be computed in $\mathcal{O}(n + |E|)$ time.

Several structures can be characterized using substring equations:

- border table (Morris-Pratt failure table),
- PREF table,
- maximal repetitions (runs),
- maximal palindromes.

Our contribution for reconstruction algorithms:

- universal $\mathcal{O}(n)$-time algorithm,
- first $\mathcal{O}(n)$-time algorithm for maximal repetitions:
  - previously $\mathcal{O}(n^2)$.

Our Results

Theorem

A generic solution of E can be computed in $\mathcal{O}(n + |E|)$ time.

Several structures can be characterized using substring equations:

- border table (Morris-Pratt failure table),
- PREF table,
- maximal repetitions (runs),
- maximal palindromes.

Our contribution for reconstruction algorithms:

- universal $\mathcal{O}(n)$-time algorithm,
- first $\mathcal{O}(n)$-time algorithm for maximal repetitions:
  - previously $\mathcal{O}(n^2)$.

Details later in the talk.

Solving Systems of Substring Equations

Positions Graph

A substrings equation represents several letter equations.

$$n = 9$$

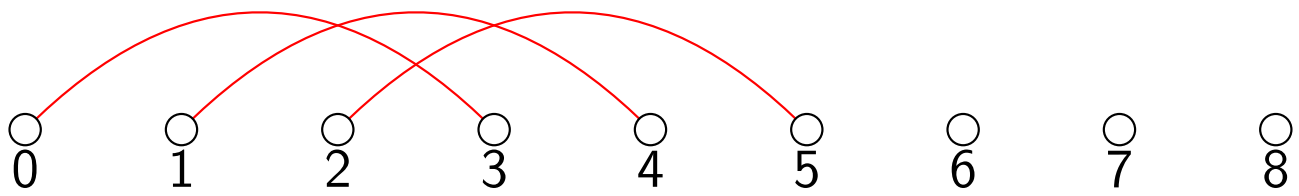
$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$

Positions Graph

A substrings equation represents several letter equations.

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



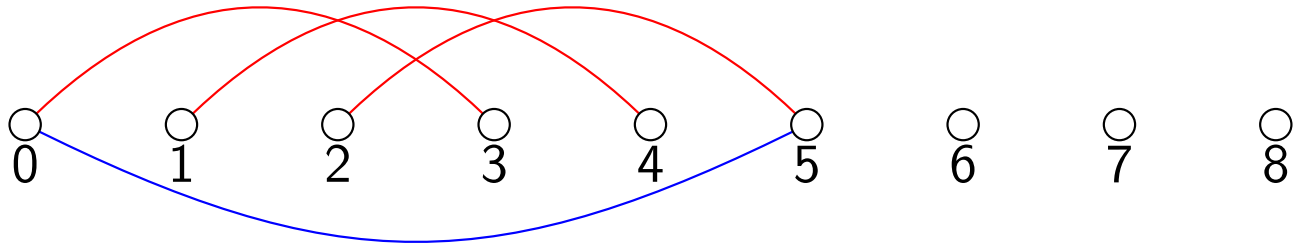
$w[0] = w[3]$ $\quad$ $w[1] = w[4]$ $\quad$ $w[2] = w[5]$

Positions Graph

A substrings equation represents several letter equations.

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



$w[0] = w[3]$ $\quad$ $w[1] = w[4]$ $\quad$ $w[2] = w[5]$

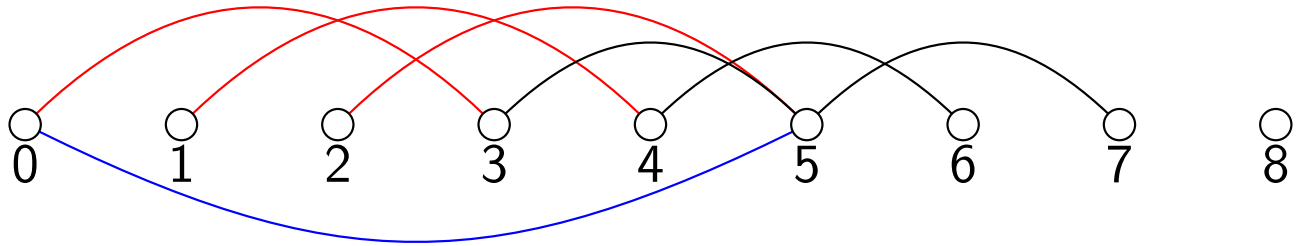
$w[0] = w[5]$

Positions Graph

A substrings equation represents several letter equations.

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



$w[0] = w[3]$ $\quad$ $w[1] = w[4]$ $\quad$ $w[2] = w[5]$

$w[0] = w[5]$

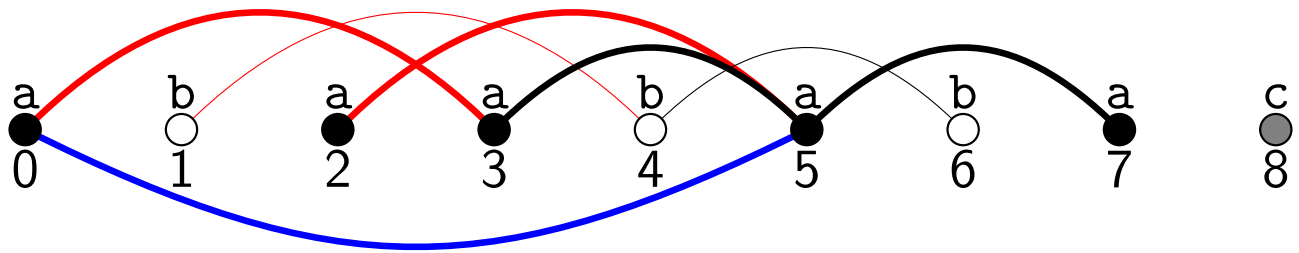
$w[3] = w[5]$ $\quad$ $w[4] = w[6]$ $\quad$ $w[5] = w[7]$

Positions Graph

A substrings equation represents several letter equations.

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



$w[0] = w[3]$ $\quad$ $w[1] = w[4]$ $\quad$ $w[2] = w[5]$

$w[0] = w[5]$

$w[3] = w[5]$ $\quad$ $w[4] = w[6]$ $\quad$ $w[5] = w[7]$

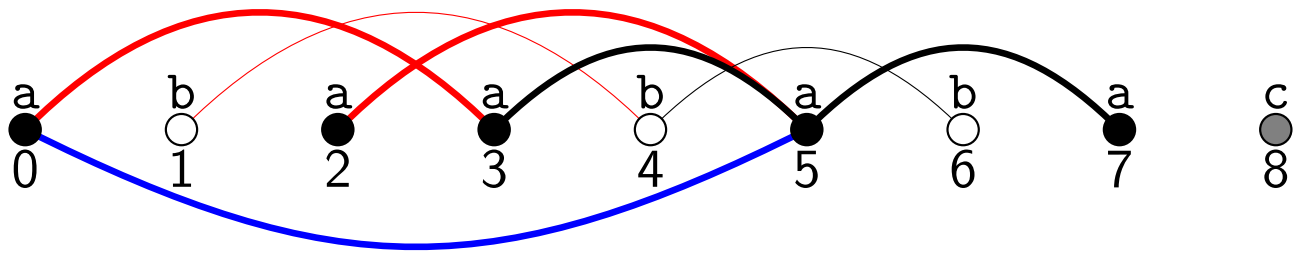
Connected components = distinct letters.

Positions Graph

A substrings equation represents several letter equations.

$$n = 9$$

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



$w[0] = w[3]$ $\quad$ $w[1] = w[4]$ $\quad$ $w[2] = w[5]$

$w[0] = w[5]$

$w[3] = w[5]$ $\quad$ $w[4] = w[6]$ $\quad$ $w[5] = w[7]$

Connected components = distinct letters.

Observation

A system can be solved in $\mathcal{O}(n + \|E\|)$ time ($\|E\|$ = total length).

Split Operation

How to make equations shorter?

Split Operation

How to make equations shorter?

Definition

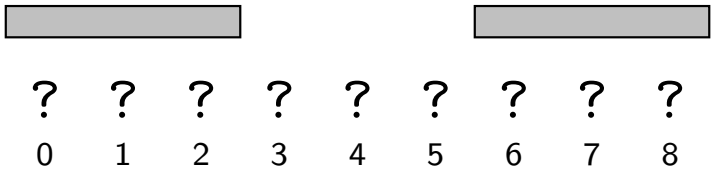
Systems E and E' are equivalent ($E \equiv E'$) if $\Phi(E) = \Phi(E')$.

Split Operation

How to make equations shorter?

Definition

Systems E and E' are equivalent ($E \equiv E'$) if $\Phi(E) = \Phi(E')$.

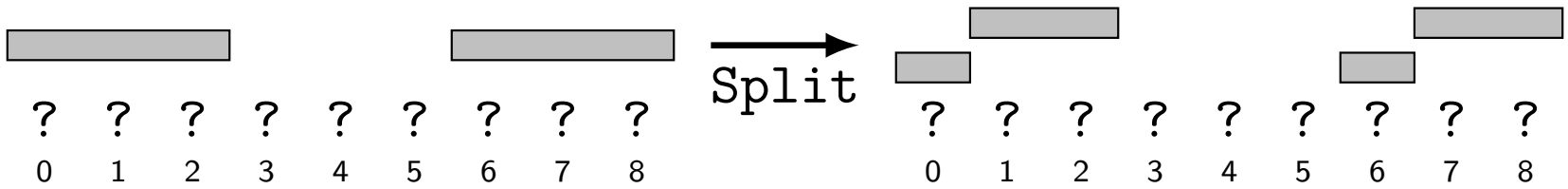


Split Operation

How to make equations shorter?

Definition

Systems E and E' are equivalent ($E \equiv E'$) if $\Phi(E) = \Phi(E')$.

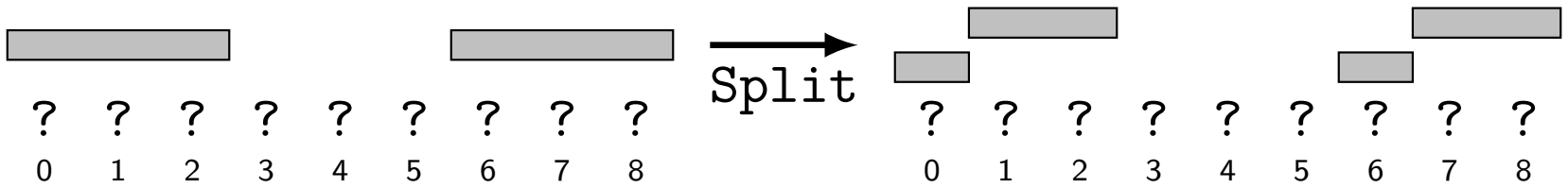


Split Operation

How to make equations shorter?

Definition

Systems E and E' are equivalent ($E \equiv E'$) if $\Phi(E) = \Phi(E')$.



Split purposes:

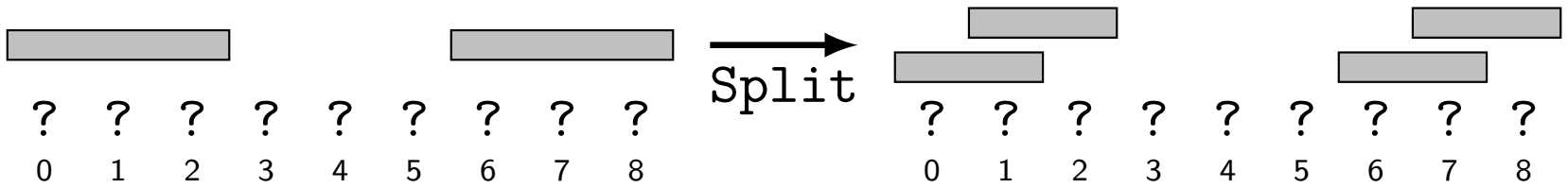
- reduce length,

Split Operation

How to make equations shorter?

Definition

Systems E and E' are equivalent ($E \equiv E'$) if $\Phi(E) = \Phi(E')$.



Split purposes:

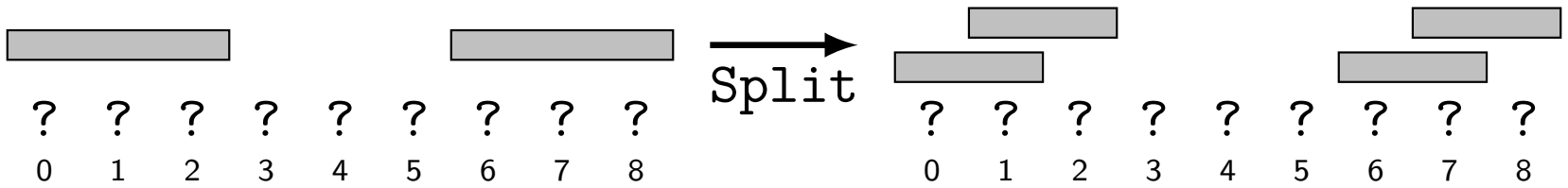
- reduce length,
- control lengths (e.g. make them powers of two).

Split Operation

How to make equations shorter?

Definition

Systems E and E' are equivalent ($E \equiv E'$) if $\Phi(E) = \Phi(E')$.



Split purposes:

- reduce length,
- control lengths (e.g. make them powers of two).

Side effect:

- increase the number of equations.

Reduce Operation

How to reduce the number of equations?

Reduce Operation

How to reduce the number of equations?

Definition

An equation $w[p..q] = w[p'..q']$ is redundant if $E \setminus \{w[p..q] = w[p'..q']\} \equiv E$.

Reduce Operation

How to reduce the number of equations?

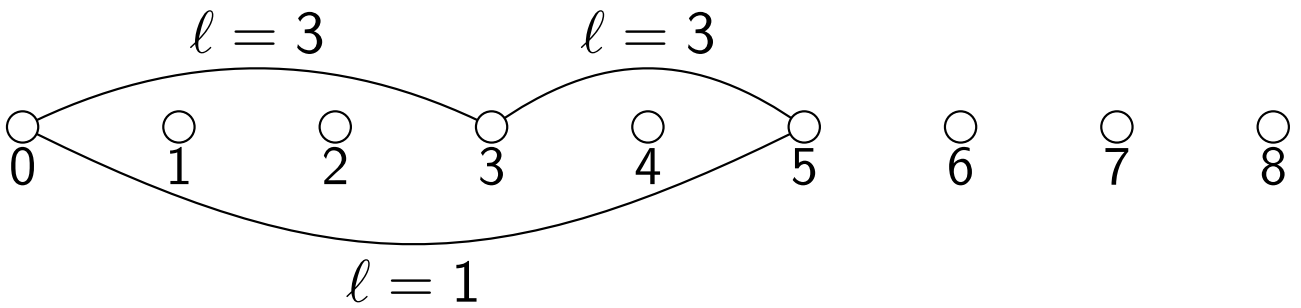
Definition

An equation $w[p..q] = w[p'..q']$ is redundant if $E \setminus \{w[p..q] = w[p'..q']\} \equiv E$.

Equations graph:

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$

$G(E)$



Reduce Operation

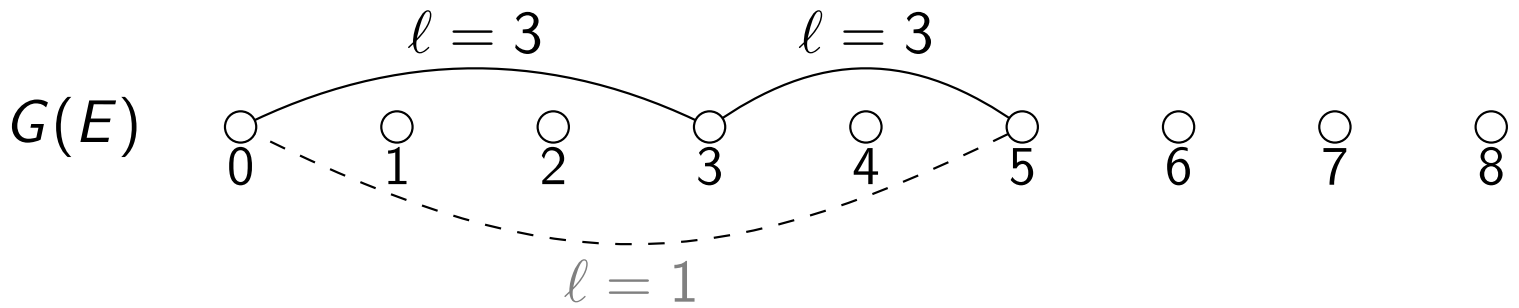
How to reduce the number of equations?

Definition

An equation $w[p..q] = w[p'..q']$ is redundant if $E \setminus \{w[p..q] = w[p'..q']\} \equiv E$.

Equations graph:

$$E = \{w[0..2] = w[3..5], w[0..0] = w[5..5], w[3..5] = w[5..7]\}$$



Lemma

$E \equiv MST(G(E))$ where MST is the max.-weight spanning forest.

$\mathcal{O}(n \log n + |E|)$-time Algorithm

Alternately apply

- `Split` to decrease the maximum length (by a factor of two),
- `Reduce` to remove redundant equations.

$\mathcal{O}(n \log n + |E|)$-time Algorithm

Alternately apply

- `Split` to decrease the maximum length (by a factor of two),
- `Reduce` to remove redundant equations.

1. $F_{\log n+1} = \emptyset$
2. For $k := \lfloor \log n \rfloor$ downto 0
   1. $E_k := \{w[p..q] = w[p'..q'] \in E : 2^k \leq |w[p..q]| < 2^{k+1}\}$
   2. `Split` equations in $F_{k+1} \cup E_k$ into equations of length 2^k
   3. `Reduce` the resulting system to obtain F_k (using BFS/DFS)
3. Return $\Phi(F_0)$ (equations in F_0 are of length 1)

$\mathcal{O}(n \log n + |E|)$-time Algorithm

Alternately apply

- `Split` to decrease the maximum length (by a factor of two),
- `Reduce` to remove redundant equations.

1. $F_{\log n+1} = \emptyset$
2. For $k := \lfloor \log n \rfloor$ downto 0
   1. $E_k := \{w[p..q] = w[p'..q'] \in E : 2^k \leq |w[p..q]| < 2^{k+1}\}$
   2. `Split` equations in $F_{k+1} \cup E_k$ into equations of length 2^k
   3. `Reduce` the resulting system to obtain F_k (using BFS/DFS)
3. Return $\Phi(F_0)$ (equations in F_0 are of length 1)

Total time complexity: $\mathcal{O}(\sum_k(|E_k| + |F_k|))$

$\mathcal{O}(n \log n + |E|)$-time Algorithm

Alternately apply

- `Split` to decrease the maximum length (by a factor of two),
- `Reduce` to remove redundant equations.

1. $F_{\log n+1} = \emptyset$
2. For $k := \lfloor \log n \rfloor$ downto 0
   1. $E_k := \{w[p..q] = w[p'..q'] \in E : 2^k \leq |w[p..q]| < 2^{k+1}\}$
   2. `Split` equations in $F_{k+1} \cup E_k$ into equations of length 2^k
   3. `Reduce` the resulting system to obtain F_k (using BFS/DFS)
3. Return $\Phi(F_0)$ (equations in F_0 are of length 1)

Total time complexity: $\mathcal{O}(\sum_k (|E_k| + |F_k|))$

$$\sum_{k=0}^{\log n} |E_k| = |E|$$

$\mathcal{O}(n \log n + |E|)$-time Algorithm

Alternately apply

- `Split` to decrease the maximum length (by a factor of two),
- `Reduce` to remove redundant equations.

1. $F_{\log n+1} = \emptyset$
2. For $k := \lfloor \log n \rfloor$ downto 0
   1. $E_k := \{w[p..q] = w[p'..q'] \in E : 2^k \leq |w[p..q]| < 2^{k+1}\}$
   2. `Split` equations in $F_{k+1} \cup E_k$ into equations of length 2^k
   3. `Reduce` the resulting system to obtain F_k (using BFS/DFS)
3. Return $\Phi(F_0)$ (equations in F_0 are of length 1)

Total time complexity: $\mathcal{O}(\sum_k (|E_k| + |F_k|))$

$$\sum_{k=0}^{\log n} |E_k| = |E| \qquad \sum_{k=0}^{\log n} |F_k| \leq \sum_{k=0}^{\log n} n = \mathcal{O}(n \log n)$$

$\mathcal{O}(n \log n + |E|)$-time Algorithm

Alternately apply

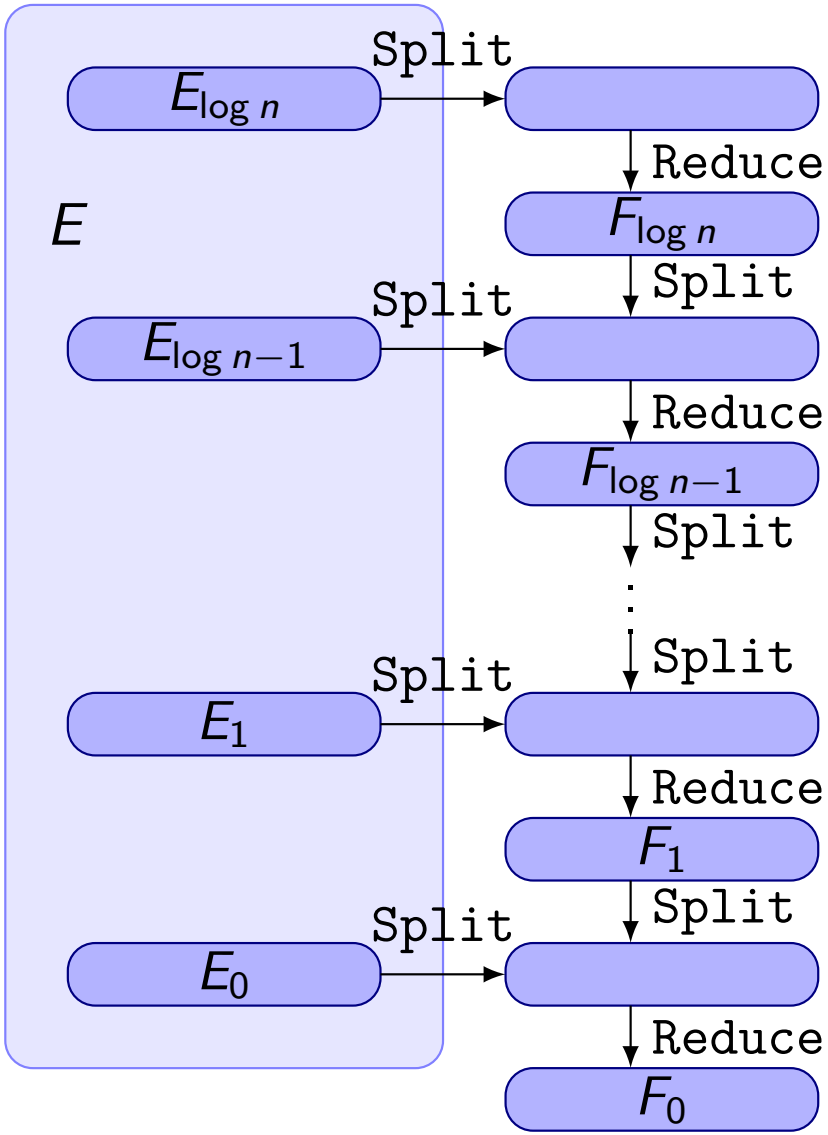
- `Split` to decrease the maximum length (by a factor of two),
- `Reduce` to remove redundant equations.

1. $F_{\log n+1} = \emptyset$
2. For $k := \lfloor \log n \rfloor$ downto 0
   1. $E_k := \{w[p..q] = w[p'..q'] \in E : 2^k \leq |w[p..q]| < 2^{k+1}\}$
   2. `Split` equations in $F_{k+1} \cup E_k$ into equations of length 2^k
   3. `Reduce` the resulting system to obtain F_k (using BFS/DFS)
3. Return $\Phi(F_0)$ (equations in F_0 are of length 1)

Total time complexity: $\mathcal{O}(\sum_k(|E_k| + |F_k|)) = \mathcal{O}(|E| + n \log n)$

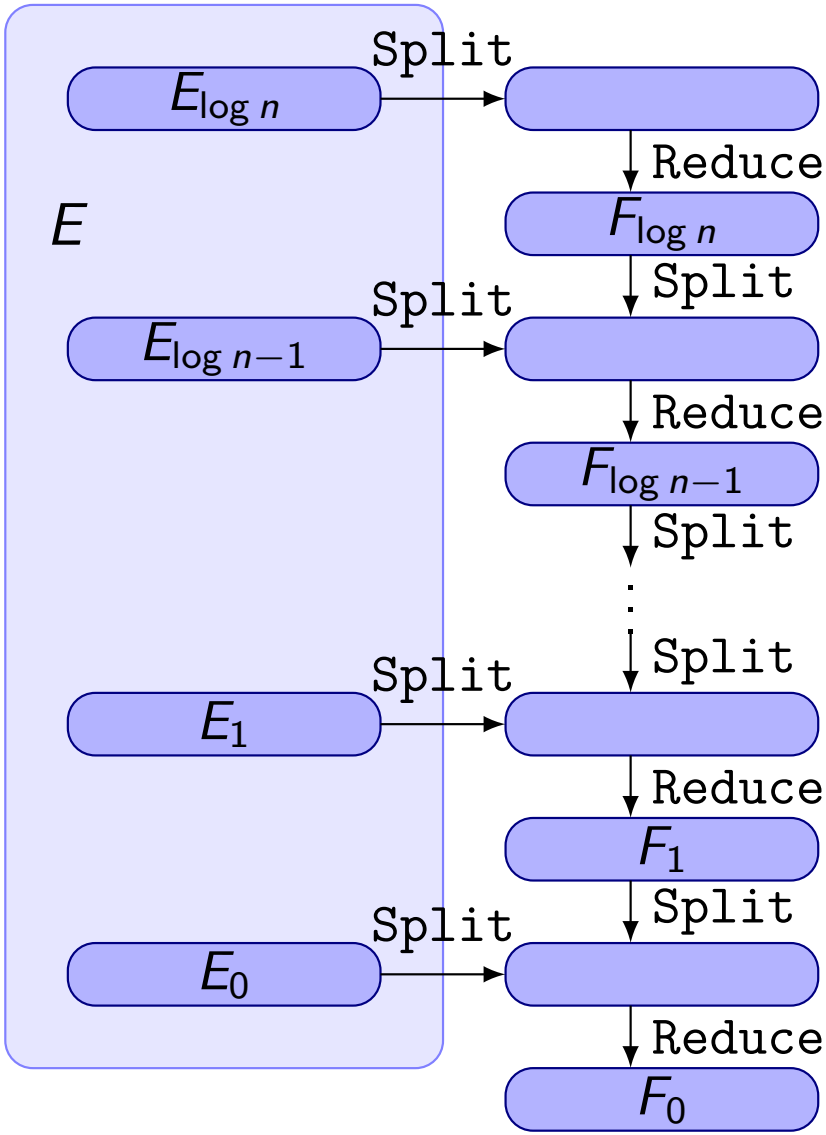
$$\sum_{k=0}^{\log n} |E_k| = |E| \qquad \sum_{k=0}^{\log n} |F_k| \leq \sum_{k=0}^{\log n} n = \mathcal{O}(n \log n)$$

Speed-up Idea

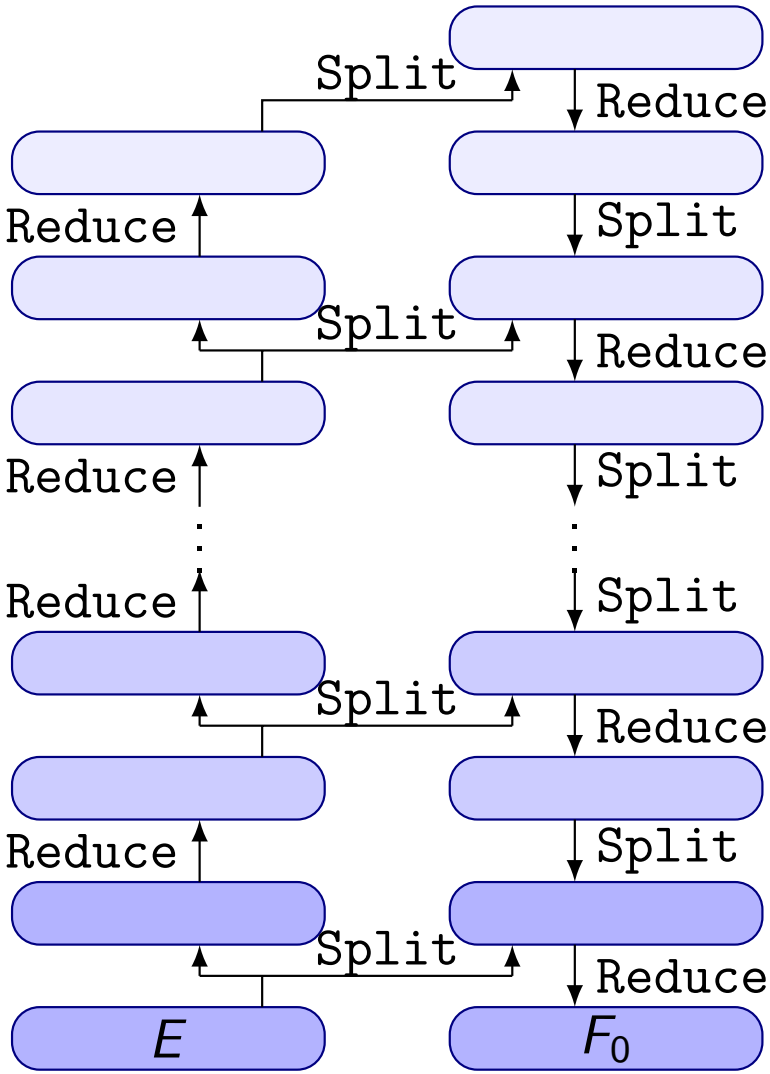


Issue: many long equations.

Speed-up Idea

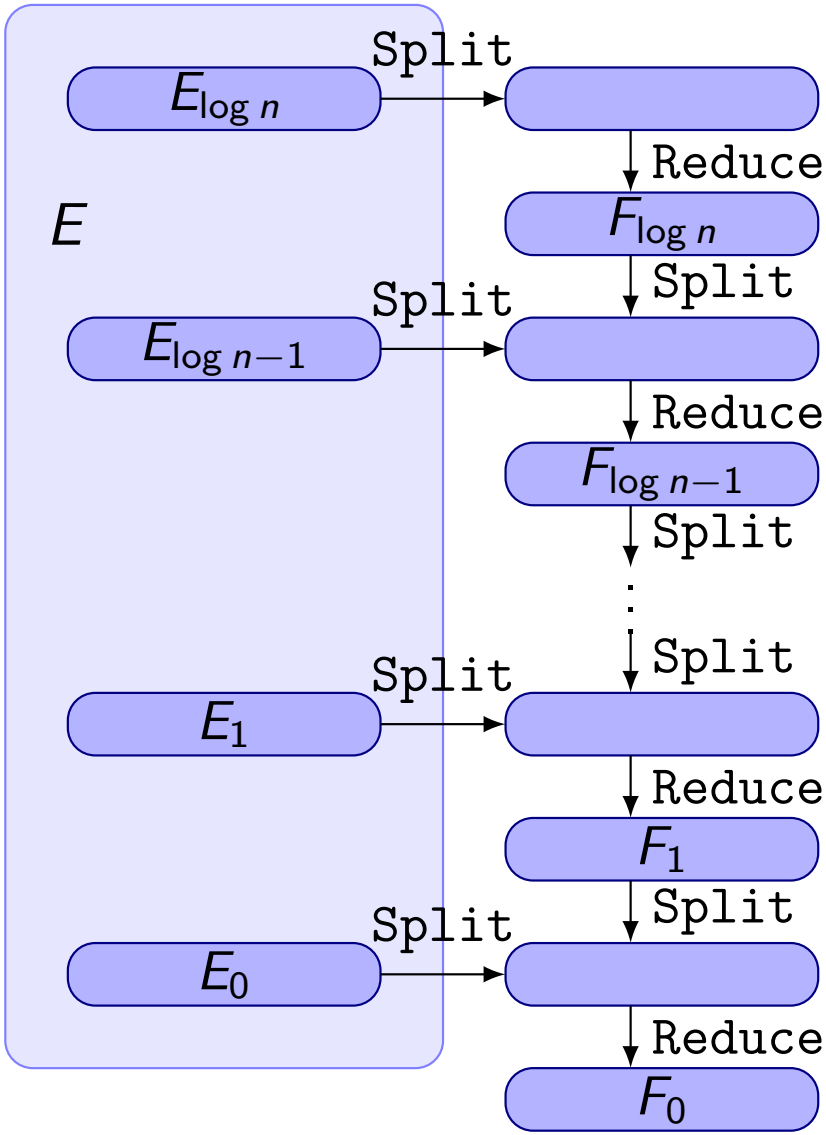


Issue: many long equations.

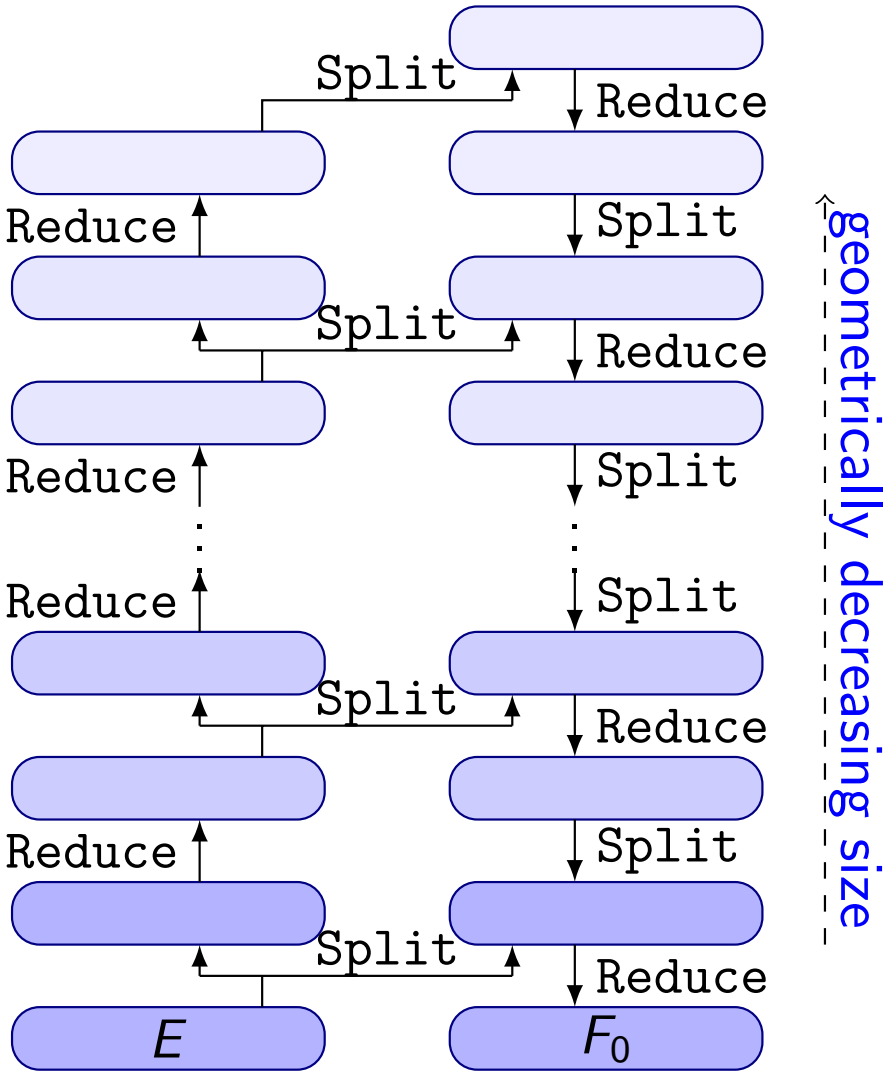


Idea: few positions of long equations.

Speed-up Idea



Issue: many long equations.



Idea: few positions of long equations.

Special Positions

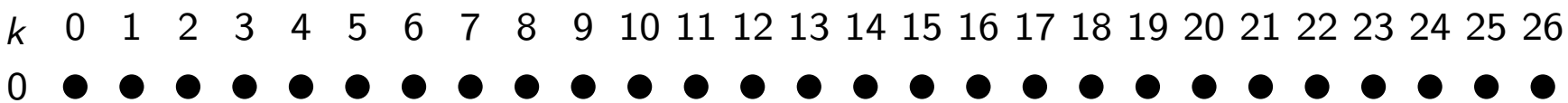
Definition

p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.

Special Positions

Definition

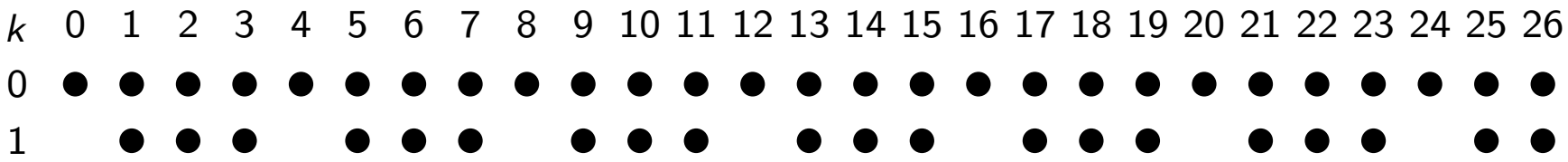
p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



Special Positions

Definition

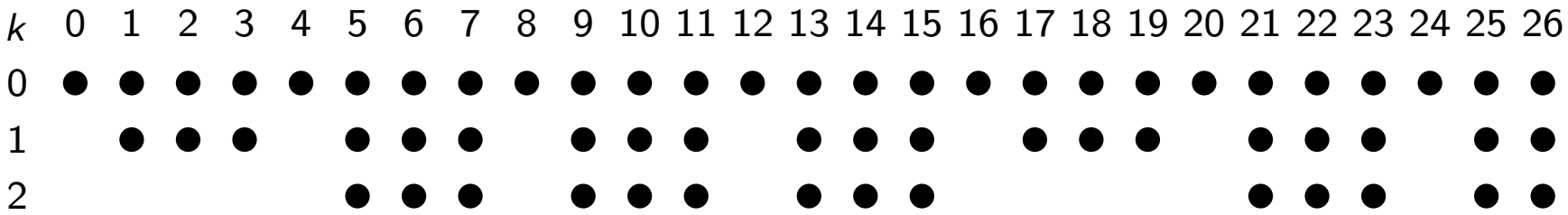
p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



Special Positions

Definition

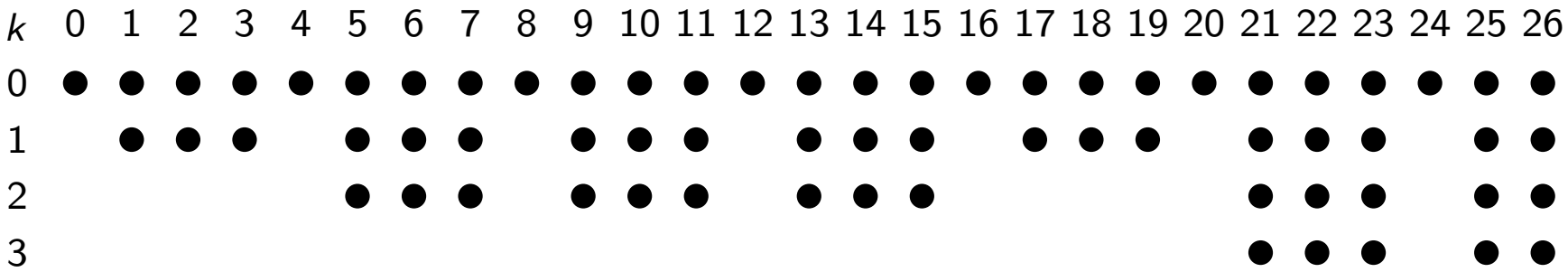
p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



Special Positions

Definition

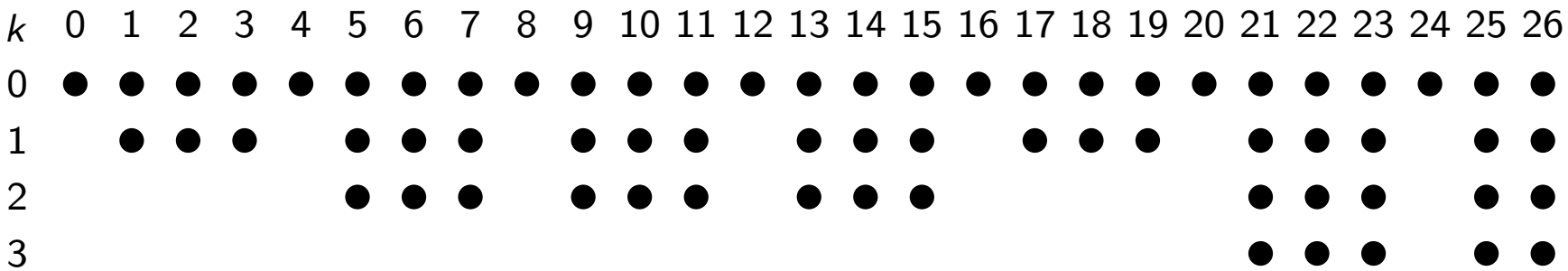
p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



Special Positions

Definition

p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.

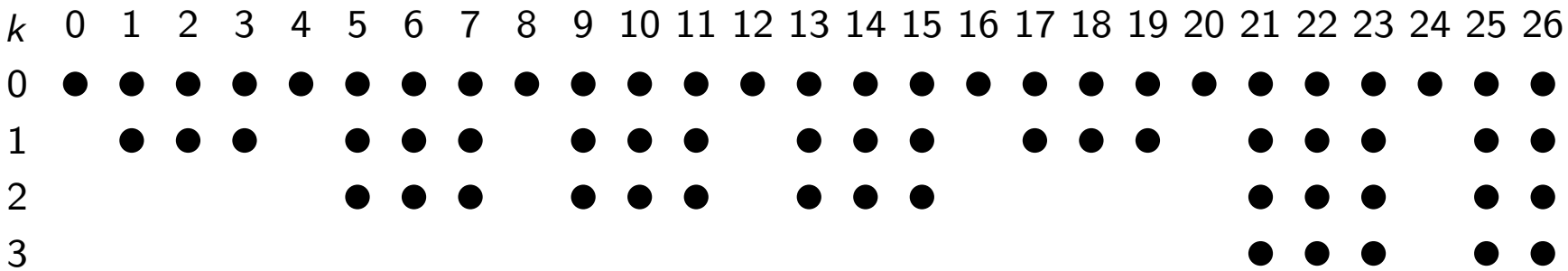


At most $(\frac{3}{4})^k n$ non-negative integers smaller than n are k-special.

Special Positions

Definition

p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



At most $(\frac{3}{4})^k n$ non-negative integers smaller than n are k-special.

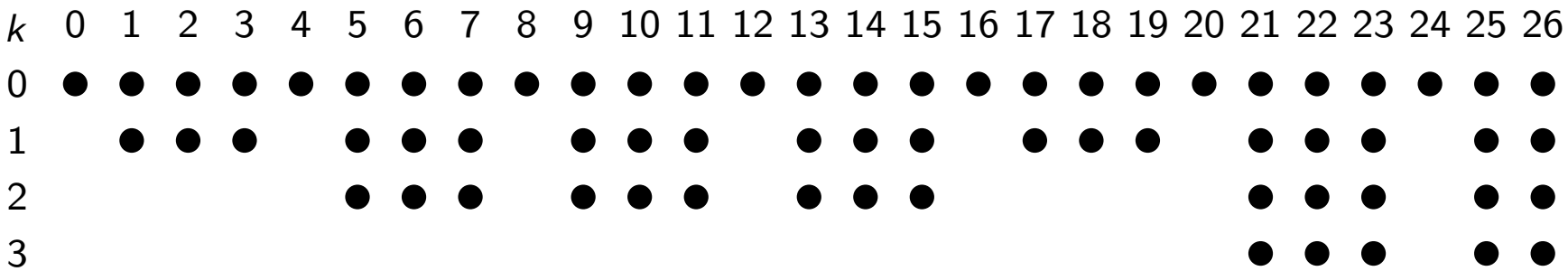
Observation

If p and p' are k-special, then for some $r \in \{0, 1, 2\}$ both $p + r4^k$ and $p' + r4^k$ are $(k+1)$-special.

Special Positions

Definition

p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



At most $(\frac{3}{4})^k n$ non-negative integers smaller than n are k-special.

Observation

If p and p' are k-special, then for some $r \in \{0, 1, 2\}$ both $p + r4^k$ and $p' + r4^k$ are $(k+1)$-special.

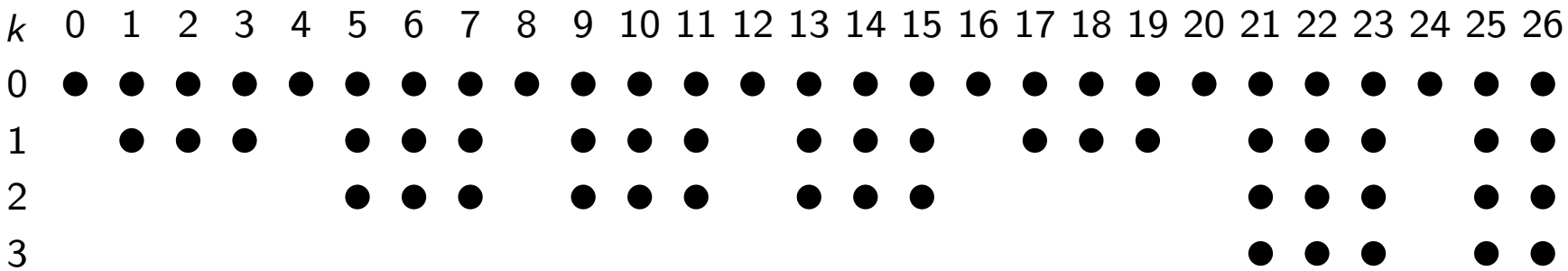
Example for $k = 1$:

$(0032)_4, (1001)_4$

Special Positions

Definition

p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



At most $(\frac{3}{4})^k n$ non-negative integers smaller than n are k-special.

Observation

If p and p' are k-special, then for some $r \in \{0, 1, 2\}$ both $p + r4^k$ and $p' + r4^k$ are $(k+1)$-special.

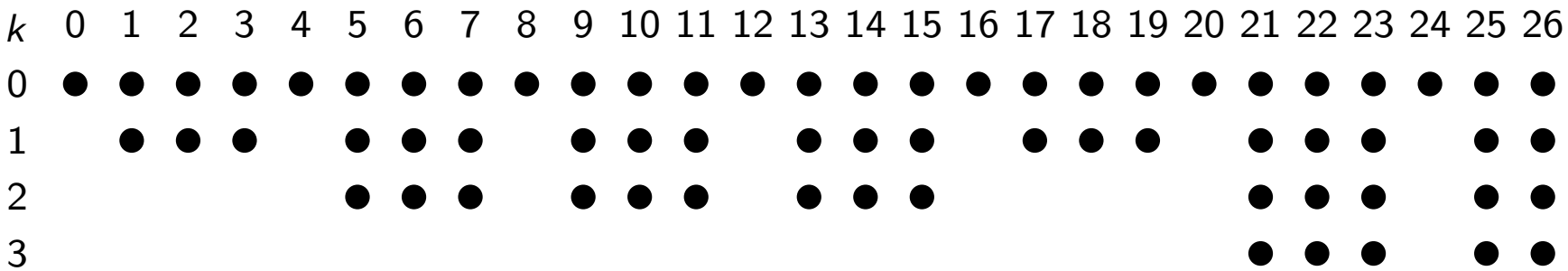
Example for $k = 1$:

$$(0032)_4, (1001)_4 \quad \rightsquigarrow \quad (0102)_4, (1011)_4$$

Special Positions

Definition

p is k-special if k least significant digits of $(p)_4$ are non-zero.
$w[p..q] = w[p'..q']$ is k-special if p and p' are k-special.



At most $(\frac{3}{4})^k n$ non-negative integers smaller than n are k-special.

Observation

If p and p' are k-special, then for some $r \in \{0, 1, 2\}$ both $p + r4^k$ and $p' + r4^k$ are $(k+1)$-special.

Example for $k = 1$:

$$(0032)_4, (1001)_4 \quad \rightsquigarrow \quad (0102)_4, (1011)_4 \quad \rightsquigarrow \quad (0112)_4, (1021)_4$$

Operations on k-Special Equations

`Split`: cut the leading 4^k characters several times.

Operations on k-Special Equations

`Split`: cut the leading 4^k characters several times.

Lemma (Special `Split`)

Every k-special equation can be split into at most two k-special equations of length $\leq 4^k$ and one $(k+1)$-special equation.

Operations on k-Special Equations

`Split`: cut the leading 4^k characters several times.

Lemma (Special `Split`)

Every k-special equation can be split into at most two k-special equations of length $\leq 4^k$ and one $(k+1)$-special equation.

Lemma (Simple `Split`)

Every k-special equation of length $\leq 4^{k+1}$ can be split into at most four k-special equations of length $\leq 4^k$.

Operations on k-Special Equations

`Split`: cut the leading 4^k characters several times.

Lemma (Special `Split`)

Every k-special equation can be split into at most two k-special equations of length $\leq 4^k$ and one $(k+1)$-special equation.

Lemma (Simple `Split`)

Every k-special equation of length $\leq 4^{k+1}$ can be split into at most four k-special equations of length $\leq 4^k$.

`Reduce`: size bounded by the number of special positions:

Lemma (`Reduce`)

If E consists of k-special equations, then $|MST(G(E))| \leq (\frac{3}{4})^k n$.

Operations on k-Special Equations

`Split`: cut the leading 4^k characters several times.

Lemma (Special `Split`)

Every k-special equation can be split into at most two k-special equations of length $\leq 4^k$ and one $(k+1)$-special equation.

Lemma (Simple `Split`)

Every k-special equation of length $\leq 4^{k+1}$ can be split into at most four k-special equations of length $\leq 4^k$.

`Reduce`: size bounded by the number of special positions:

Lemma (`Reduce`)

If E consists of k-special equations, then $|MST(G(E))| \leq (\frac{3}{4})^k n$.

Implementation:

1994: Fredman & Willard Linear-time MST for integer weights.

$\mathcal{O}(n)$-time Algorithm

Recursive function:

input reduced system of k-special equations.

output reduced system of k-special equations of length $\leq 4^k$.

$\mathcal{O}(n)$-time Algorithm

Recursive function:

input reduced system of k-special equations.

output reduced system of k-special equations of length $\leq 4^k$.

1. `Split` each equation into:
   - equations of lengths $\leq 4^k$ (F),
   - $(k+1)$-special equations (G).
2. `Reduce` G and recursively process it to obtain $(k+1)$-special equations of lengths $\leq 4^{k+1}$.
3. `Split` each of these equations into k-special equations of lengths $\leq 4^k$.
4. `Reduce` and return the resulting system combined with F.

$\mathcal{O}(n)$-time Algorithm

Recursive function:

input reduced system of k-special equations.

output reduced system of k-special equations of length $\leq 4^k$.

1. `Split` each equation into:
   - equations of lengths $\leq 4^k$ (F),
   - $(k+1)$-special equations (G).
2. `Reduce` G and recursively process it to obtain $(k+1)$-special equations of lengths $\leq 4^{k+1}$.
3. `Split` each of these equations into k-special equations of lengths $\leq 4^k$.
4. `Reduce` and return the resulting system combined with F.

Running time: $\mathcal{O}((\frac{3}{4})^k n)$.

$\mathcal{O}(n)$-time Algorithm

Recursive function:

input reduced system of k-special equations.

output reduced system of k-special equations of length $\leq 4^k$.

1. `Split` each equation into:
   - equations of lengths $\leq 4^k$ (F),
   - $(k+1)$-special equations (G).
2. `Reduce` G and recursively process it to obtain $(k+1)$-special equations of lengths $\leq 4^{k+1}$.
3. `Split` each of these equations into k-special equations of lengths $\leq 4^k$.
4. `Reduce` and return the resulting system combined with F.

Running time: $\mathcal{O}((\frac{3}{4})^k n)$.

Main procedure:

1. Reduce the input family E and call the function with $k = 0$.

$\mathcal{O}(n)$-time Algorithm

Recursive function:

input reduced system of k-special equations.

output reduced system of k-special equations of length $\leq 4^k$.

1. `Split` each equation into:
   - equations of lengths $\leq 4^k$ (F),
   - $(k+1)$-special equations (G).
2. `Reduce` G and recursively process it to obtain $(k+1)$-special equations of lengths $\leq 4^{k+1}$.
3. `Split` each of these equations into k-special equations of lengths $\leq 4^k$.
4. `Reduce` and return the resulting system combined with F.

Running time: $\mathcal{O}((\frac{3}{4})^k n)$.

Main procedure:

1. Reduce the input family E and call the function with $k = 0$.

Running time: $\mathcal{O}(|E| + n)$.

More Practical Variant

Linear-time algorithm of Fredman and Willard is very impractical.
Is it necessary to use it?

More Practical Variant

Linear-time algorithm of Fredman and Willard is very impractical.
Is it necessary to use it? No!

Solution:

1. Stronger invariants on equations:
   - strongly k-special equations (length 2^p for $p \geq 2k$).

More Practical Variant

Linear-time algorithm of Fredman and Willard is very impractical.
Is it necessary to use it? No!

Solution:

1. Stronger invariants on equations:
   - strongly k-special equations (length 2^p for $p \geq 2k$).
2. Single initial `Split` phase to make lengths powers of two.

More Practical Variant

Linear-time algorithm of Fredman and Willard is very impractical.
Is it necessary to use it? No!

Solution:

1. Stronger invariants on equations:
   - strongly k-special equations (length 2^p for $p \geq 2k$).
2. Single initial `Split` phase to make lengths powers of two.
3. More sophisticated Special `Split` implementation:
   - so that lengths are still powers of two.

More Practical Variant

Linear-time algorithm of Fredman and Willard is very impractical.
Is it necessary to use it? No!

Solution:

1. Stronger invariants on equations:
   - strongly k-special equations (length 2^p for $p \geq 2k$).
2. Single initial `Split` phase to make lengths powers of two.
3. More sophisticated Special `Split` implementation:
   - so that lengths are still powers of two.

Profit:

1. MST computation based on Dijkstra-Jarník-Prim algorithm:
   - priority queue for weights $2^0, \ldots, 2^{\log n}$,
     - constant-time operations (least significant set bit extraction).

Applications: Universal Reconstruction Framework

Border Table

Definition

A border of a string occurs both as its prefix and as its suffix.

a	b	a	a	b	c	b	a	a	b	a	a
0	1	2	3	4	5	6	7	8	9	10	11

Border Table

Definition

A border of a string occurs both as its prefix and as its suffix.

a	b	a	a	b	c	b	a	a	b	a	a
0	1	2	3	4	5	6	7	8	9	10	11

Border Table

Definition

A border of a string occurs both as its prefix and as its suffix.

a	b	a	a	b	c	b	a	a	b	a	a
0	1	2	3	4	5	6	7	8	9	10	11

Definition

The border table stores for each prefix the length of its longest proper border.

Border Table

Definition

A border of a string occurs both as its prefix and as its suffix.

w :	a	b	a	a	b	c	b	a	a	b	a	a
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

Definition

The border table stores for each prefix the length of its longest proper border.

Border Table

Definition

A border of a string occurs both as its prefix and as its suffix.

w:	a	b	a	a	b	c	b	a	a	b	a	a
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

Definition

The border table stores for each prefix the length of its longest proper border.

Linear-time algorithms:

1970: Morris & Pratt construction

2002: Franek et al. verification

2009: Duval et al. reconstruction

Border Table

Definition

A border of a string occurs both as its prefix and as its suffix.

w :	a	b	a	a	b	b	b	a	a	b	a	a
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

Definition

The border table stores for each prefix the length of its longest proper border.

Linear-time algorithms:

1970: Morris & Pratt construction

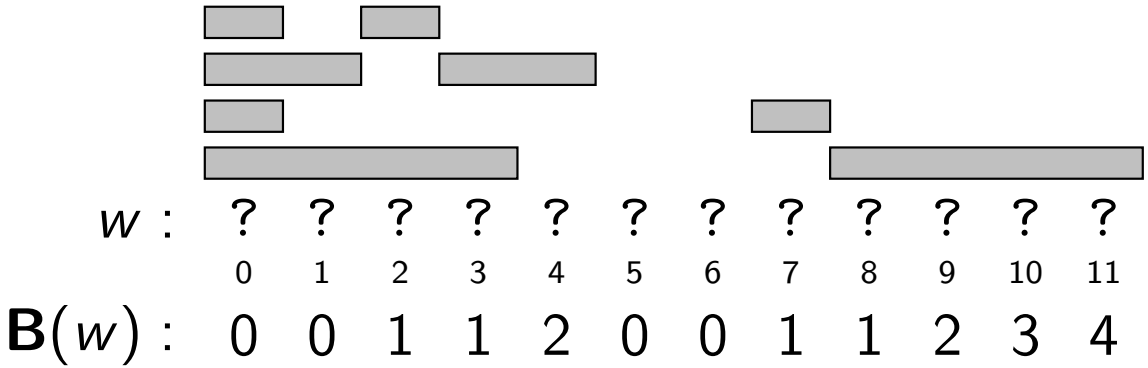
2002: Franek et al. verification

2009: Duval et al. reconstruction (smallest possible alphabet)

Universal Reconstruction: Border Table

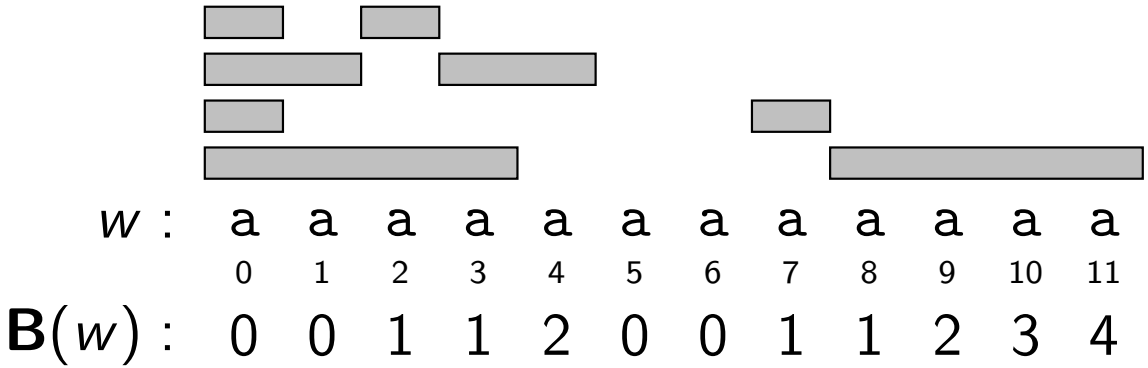
w :	?	?	?	?	?	?	?	?	?	?	?	?
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

Universal Reconstruction: Border Table



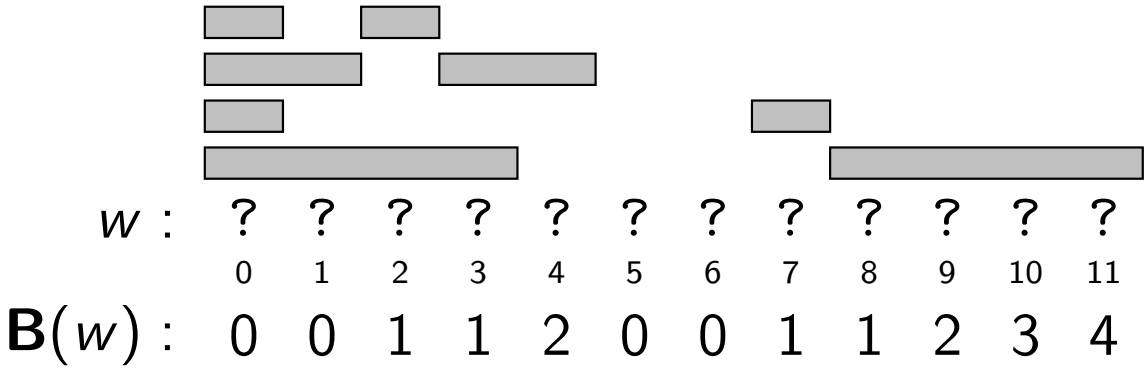
equations $w[0..B[i]-1] = w[i-B[i]+1..i]$

Universal Reconstruction: Border Table



equations $w[0..B[i]-1] = w[i-B[i]+1..i]$

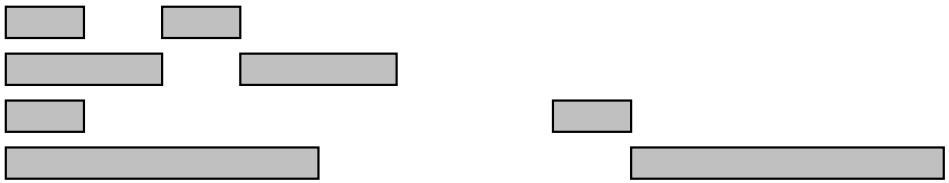
Universal Reconstruction: Border Table



equations $w[0..B[i]-1] = w[i-B[i]+1..i]$

non-equations $w[0..j-1] \neq w[i-j+1..i]$ for $B[i] < j < i$

Universal Reconstruction: Border Table

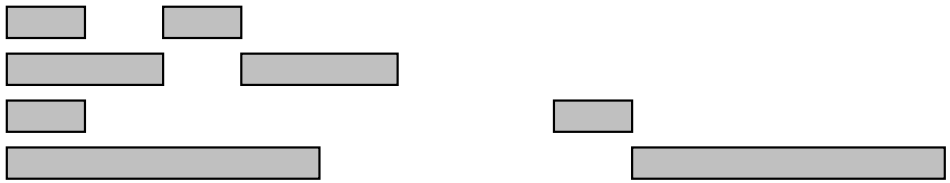


$\Phi(E) = w$:	a	b	a	a	b	c	d	a	a	b	a	a
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

equations $w[0..B[i]-1] = w[i-B[i]+1..i]$

non-equations $w[0..j-1] \neq w[i-j+1..i]$ for $B[i] < j < i$

Universal Reconstruction: Border Table



$\Phi(E) = w$:	a	b	a	a	b	c	d	a	a	b	a	a
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

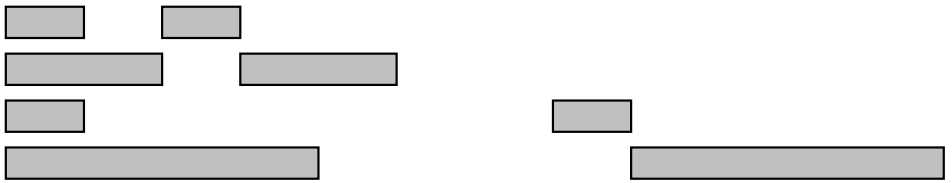
equations $w[0..B[i]-1] = w[i-B[i]+1..i]$

non-equations $w[0..j-1] \neq w[i-j+1..i]$ for $B[i] < j < i$

Observation

A generic solution satisfies all satisfiable non-equations.

Universal Reconstruction: Border Table



$\Phi(E) = w$:	a	b	a	a	b	c	d	a	a	b	a	a
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

equations $w[0..B[i]-1] = w[i-B[i]+1..i]$

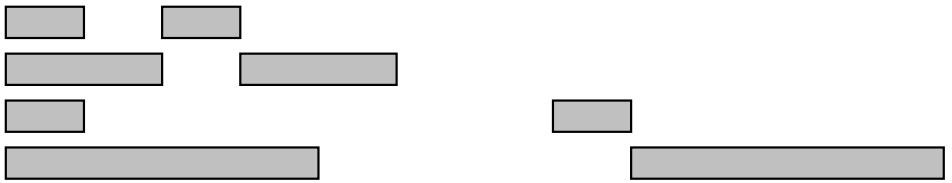
non-equations $w[0..j-1] \neq w[i-j+1..i]$ for $B[i] < j < i$

Observation

A generic solution satisfies all satisfiable non-equations.

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.

Universal Reconstruction: Border Table



$\Phi(E) = w$:	a	b	a	a	b	c	d	a	a	b	a	a
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{B}(w)$:	0	0	1	1	2	0	0	1	1	2	3	4

equations $w[0..B[i]-1] = w[i-B[i]+1..i]$

non-equations $w[0..j-1] \neq w[i-j+1..i]$ for $B[i] < j < i$

Observation

A generic solution satisfies all satisfiable non-equations.

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.
2. Applies to structures characterized using a conjunction of:
   - explicit substring equations $w[p..q] = w[p'..q']$
   - implicit substring non-equations $w[p..q] \neq w[p'..q']$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$

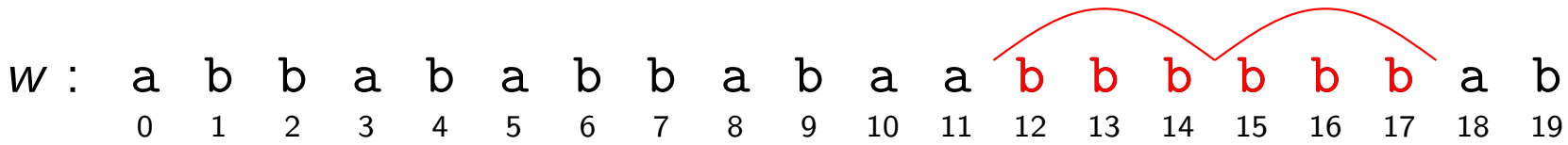
w :	a	b	b	a	b	a	b	b	a	b	a	a	b	b	b	b	b	b	a	b
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



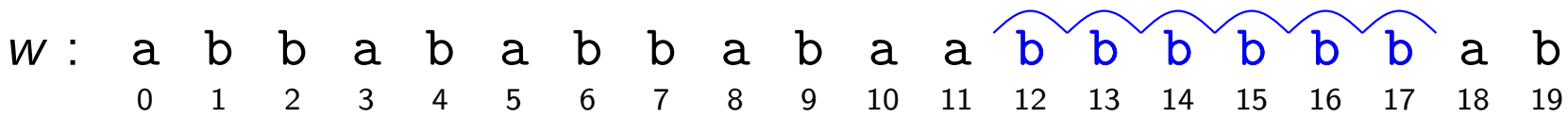
$$(12, 17, 3) \notin \mathbf{R}(w)$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



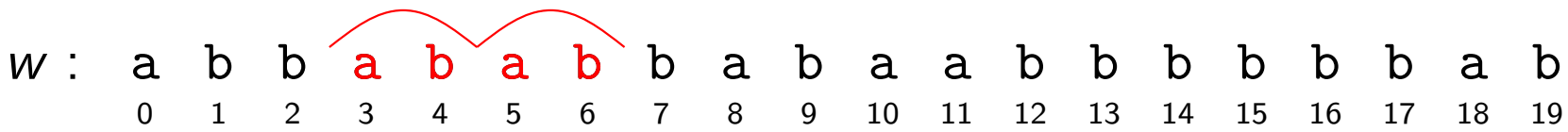
$$(12, 17, 3) \notin \mathbf{R}(w) \qquad (12, 17, 1) \in \mathbf{R}(w)$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



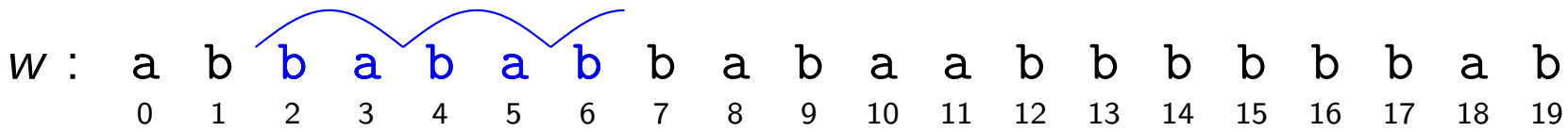
$$(3, 6, 2) \notin \mathbf{R}(w)$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



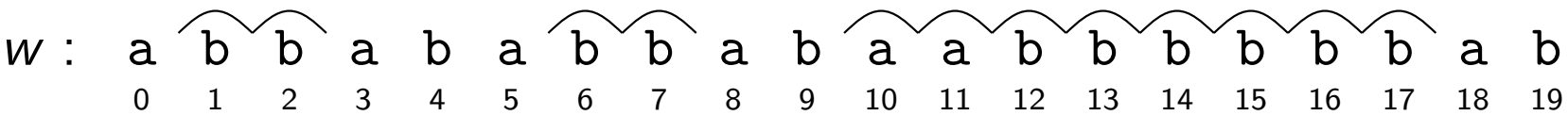
$$(3, 6, 2) \notin \mathbf{R}(w) \qquad (2, 6, 2) \in \mathbf{R}(w)$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



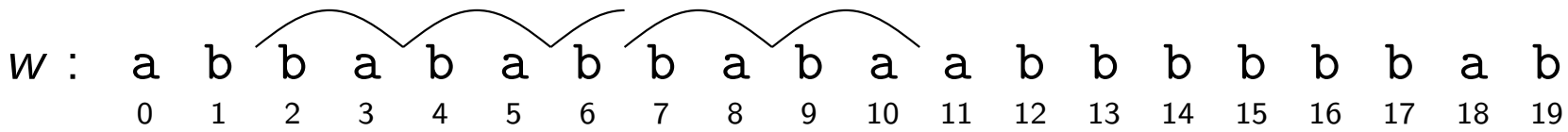
$$\mathbf{R}(w) = \{(1, 2, 1), (6, 7, 1), (10, 11, 1), (12, 17, 1),$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



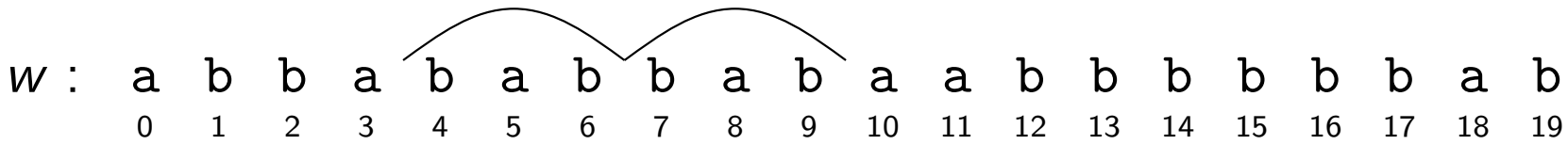
$$\mathbf{R}(w) = \{(1, 2, 1), (6, 7, 1), (10, 11, 1), (12, 17, 1), (2, 6, 2), (7, 10, 2)\}$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



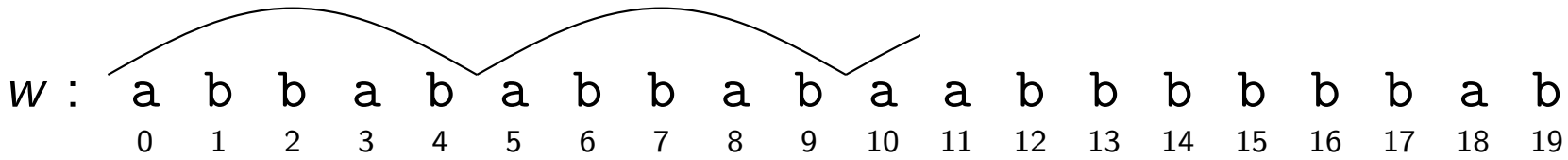
$$\mathbf{R}(w) = \{(1, 2, 1), (6, 7, 1), (10, 11, 1), (12, 17, 1),$$
$$(2, 6, 2), (7, 10, 2), (4, 9, 3)$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$



$$\mathbf{R}(w) = \{(1,2,1), (6,7,1), (10,11,1), (12,17,1),$$
$$(2,6,2), (7,10,2), (4,9,3), (0,10,5)\}$$

Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

$$\mathbf{R}(w) = \{(i, j, p) : w[i..j] \text{ is a maximal repetition of period } p\}.$$

w :	a	b	b	a	b	a	b	b	a	b	a	a	b	b	b	b	b	b	a	b
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

$$\mathbf{R}(w) = \{(1,2,1), (6,7,1), (10,11,1), (12,17,1), (2,6,2), (7,10,2), (4,9,3), (0,10,5)\}$$

Theorem (Bannai et al.; SODA 2015)

A string of length n has at most $n-1$ maximal repetitions.

Maximal Repetitions: Construction & Reconstruction

1996: Kolpakov & Kucherov Linear-time construction

2010: Matsubara et al. Quadratic-time reconstruction

Maximal Repetitions: Construction & Reconstruction

1996: Kolpakov & Kucherov Linear-time construction

2010: Matsubara et al. Quadratic-time reconstruction

2015: this work Linear-time reconstruction

Maximal Repetitions: Construction & Reconstruction

1996: Kolpakov & Kucherov Linear-time construction

2010: Matsubara et al. Quadratic-time reconstruction

2015: this work Linear-time reconstruction

Example:

$$|w| = 20 \qquad \mathbf{R}(w) = \{(1,2,1),(6,7,1),(10,11,1),(12,17,1),(2,6,2),(7,10,2),(4,9,3),(0,10,5)\}$$

w :	a	b	b	a	b	a	b	b	a	b	a	a	c	c	c	c	c	c	d	e
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Maximal Repetitions: Construction & Reconstruction

1996: Kolpakov & Kucherov Linear-time construction

2010: Matsubara et al. Quadratic-time reconstruction

2015: this work Linear-time reconstruction

Example:

$$|w| = 20 \qquad \mathbf{R}(w) = \{(1,2,1),(6,7,1),(10,11,1),(12,17,1),(2,6,2),(7,10,2),(4,9,3),(0,10,5)\}$$

w :	a	b	b	a	b	a	b	b	a	b	a	a	c	c	c	c	c	c	d	e
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

2010: Matsubara et al. NP-hard to minimize the alphabet size

Universal Reconstruction: Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.
2. Applies to structures characterized using a conjunction of:
   - explicit substring equations $w[p..q] = w[p'..q']$
   - implicit substring non-equations $w[p..q] \neq w[p'..q']$

Universal Reconstruction: Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

equations $w[i..j-p] = w[i+p..j]$ (p is a period of $w[i..j]$),

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.
2. Applies to structures characterized using a conjunction of:
   - explicit substring equations $w[p..q] = w[p'..q']$
   - implicit substring non-equations $w[p..q] \neq w[p'..q']$

Universal Reconstruction: Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

equations $w[i..j-p] = w[i+p..j]$ (p is a period of $w[i..j]$),

non-equations $w[i..j]$ is indeed a maximal repetition of period p:

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.
2. Applies to structures characterized using a conjunction of:
   - explicit substring equations $w[p..q] = w[p'..q']$
   - implicit substring non-equations $w[p..q] \neq w[p'..q']$

Universal Reconstruction: Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

equations $w[i..j-p] = w[i+p..j]$ (p is a period of $w[i..j]$),

non-equations $w[i..j]$ is indeed a maximal repetition of period p:

- $w[i..j]$ does not have a period shorter than p,

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.
2. Applies to structures characterized using a conjunction of:
   - explicit substring equations $w[p..q] = w[p'..q']$
   - implicit substring non-equations $w[p..q] \neq w[p'..q']$

Universal Reconstruction: Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

equations $w[i..j-p] = w[i+p..j]$ (p is a period of $w[i..j]$),

non-equations $w[i..j]$ is indeed a maximal repetition of period p:

- $w[i..j]$ does not have a period shorter than p,
- $w[i..j]$ cannot be extended preserving p.

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.
2. Applies to structures characterized using a conjunction of:
   - explicit substring equations $w[p..q] = w[p'..q']$
   - implicit substring non-equations $w[p..q] \neq w[p'..q']$

Universal Reconstruction: Maximal Repetitions

Definition

A fragment $w[i..j]$ is a repetition of period p in a string w if $\mathrm{per}(w[i..j]) = p$ and $|w[i..j]| \geq 2p$. It is a maximal repetition (a run) if it cannot be extended preserving the period.

equations $w[i..j-p] = w[i+p..j]$ (p is a period of $w[i..j]$),

non-equations $w[i..j]$ is indeed a maximal repetition of period p:

- $w[i..j]$ does not have a period shorter than p,
- $w[i..j]$ cannot be extended preserving p.

non-equations every square of length $2q$ can be extended to a maximal repetition with period p dividing q.

1. Verify non-equations using an oracle:
   - construction algorithm for the structure.
2. Applies to structures characterized using a conjunction of:
   - explicit substring equations $w[p..q] = w[p'..q']$
   - implicit substring non-equations $w[p..q] \neq w[p'..q']$

Conclusions & Open Problems

Our contributions:

- A linear-time algorithm solving systems of equations.
- Formulation of general reconstruction setting.
- Several applications:
  - border table,
  - maximal repetitions,
  - PREF table,
  - maximal palindromes (extended setting for reversed equations).

Conclusions & Open Problems

Our contributions:

- A linear-time algorithm solving systems of equations.
- Formulation of general reconstruction setting.
- Several applications:
  - border table,
  - maximal repetitions,
  - PREF table,
  - maximal palindromes (extended setting for reversed equations).

Open problems:

- Further applications of substring equation systems?
- Generic algorithm for wider class of problems?
  - Include strong border table (KMP failure table).
- When alphabet minimization is possible?

Questions?

Thank you for your attention!