# A Linear Time Algorithm for Seeds Computation

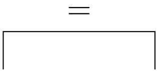
**Tomasz Kociumaka**, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, Tomasz Waleń

University of Warsaw

**SODA 2012** Kyoto, January 18, 2012

# Periodicity and quasiperiodicity

Periodicity:

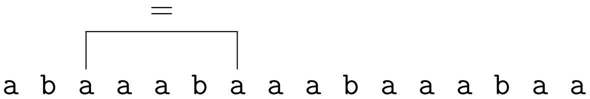



a b a a a b a a a b a a a b a a

One of the key concepts in text algorithms.

# Periodicity and quasiperiodicity

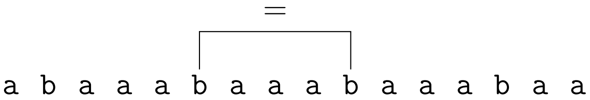
Periodicity:




One of the key concepts in text algorithms.

# Periodicity and quasiperiodicity

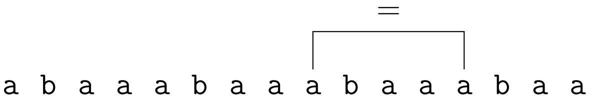
Periodicity:




One of the key concepts in text algorithms.

# Periodicity and quasiperiodicity

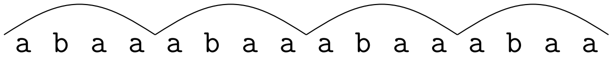
Periodicity:




One of the key concepts in text algorithms.

# Periodicity and quasiperiodicity

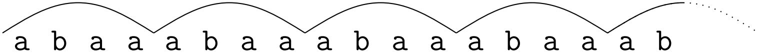
Periodicity:




One of the key concepts in text algorithms.

# Periodicity and quasiperiodicity

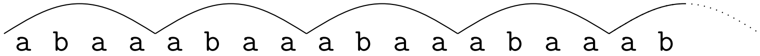
Periodicity:



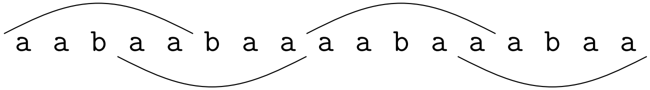

One of the key concepts in text algorithms.

# Periodicity and quasiperiodicity

Periodicity:

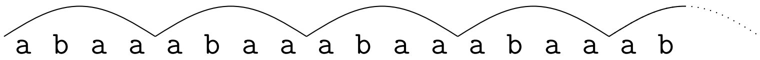



Quasiperiodicity:

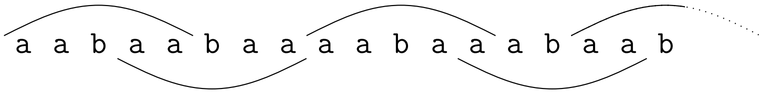


# Periodicity and quasiperiodicity

Periodicity:




Quasiperiodicity:



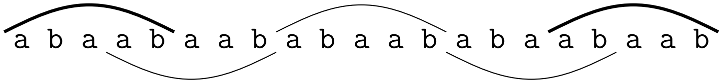
# Covers and seeds

Cover:

a b a a b a a b a b a a b a b a a b a a b

# Covers and seeds

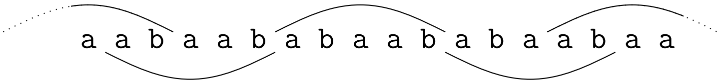
Cover:




Each letter of the word is covered by an occurrence of the cover.

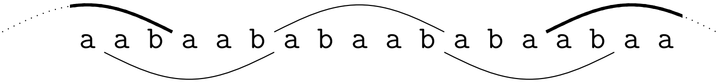
# Covers and seeds

Seed:



# Covers and seeds

Seed:




Each letter of the word is covered by an occurrence of the seed. The occurrences can be external.

# The main problem

**Problem (Shortest-Seed)**

*Given a word $w$ of length $n$ over an alphabet $\Sigma$, compute the shortest seed of $w$.*

# The main problem

## Problem (Shortest-Seed)

*Given a word $w$ of length $n$ over an alphabet $\Sigma$, compute the shortest seed of $w$.*

## Problem (All-Seeds)

*Given a word $w$ of length $n$ over an alphabet $\Sigma$, compute an $O(n)$-sized representation of all the seeds of $w$.*

# The main problem

**Problem (Shortest-Seed)**

*Given a word $w$ of length $n$ over an alphabet $\Sigma$, compute the shortest seed of $w$.*

**Problem (All-Seeds)**

*Given a word $w$ of length $n$ over an alphabet $\Sigma$, compute an $O(n)$-sized representation of all the seeds of $w$.*

**Theorem (Our result)**

*The All-Seeds Problem for $\Sigma = \{0, 1, \ldots, n^{O(1)}\}$ can be solved in $O(n)$ time.*

# Background

- Seeds were introduced in 1993 by Iliopoulos, Moore & Park.
- In the same paper $O(n \log n)$-time algorithm for the All-Seeds Problem over a fixed-size alphabet is given.

# Background

- Seeds were introduced in 1993 by Iliopoulos, Moore & Park.
- In the same paper $O(n \log n)$-time algorithm for the All-Seeds Problem over a fixed-size alphabet is given.
- No $o(n \log n)$ algorithm even for the Shortest-Seed Problem for binary alphabet up to now.

# Background

- Seeds were introduced in 1993 by Iliopoulos, Moore & Park.
- In the same paper $O(n \log n)$-time algorithm for the All-Seeds Problem over a fixed-size alphabet is given.
- No $o(n \log n)$ algorithm even for the Shortest-Seed Problem for binary alphabet up to now.
- W.F. Smyth stated finding a linear algorithm for the All-Seeds Problem as a hard open problem in his survey (2000).

# Background

- An $O(\log n)$-time PRAM algorithm for $n$ processors, Ben-Amran et al., SODA 1994.

# Background

- An $O(\log n)$-time PRAM algorithm for $n$ processors, Ben-Amran et al., SODA 1994.
- For covers linear algorithms for similar problems are known:
    - shortest covers of each prefix (Breslauer, 1992)
    - all covers (Moore & Smyth, SODA 1994)

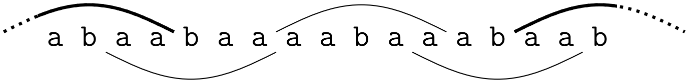
# Background

- An $O(\log n)$-time PRAM algorithm for $n$ processors, Ben-Amran et al., SODA 1994.
- For covers linear algorithms for similar problems are known:
    - shortest covers of each prefix (Breslauer, 1992)
    - all covers (Moore & Smyth, SODA 1994)
- Variants of seeds have been studied:
    - approximate seeds (Christodoulakis et al., 2003)
    - $\lambda$-seeds (Guo, Zhang & Iliopoulos, 2006)

# Constraints for seeds

Two different types of constraints

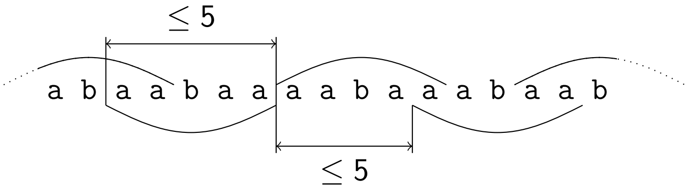
- Border constraints, easier



# Constraints for seeds

Two different types of constraints

- Border constraints, easier
- Maxgap constrains, harder




Maxgap is a maximal distance between the starting positions of two consecutive occurrences of a given subword.

# Quasiseeds

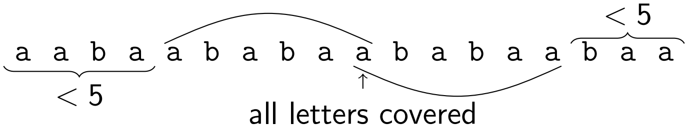
- The All-Seeds Problem can be linearly reduced to computing the maxgaps of all subwords (encoded in a suffix tree).
- No $o(n \log n)$ algorithm known.

# Quasiseeds

- The All-Seeds Problem can be linearly reduced to computing the maxgaps of all subwords (encoded in a suffix tree).
- No $o(n \log n)$ algorithm known.

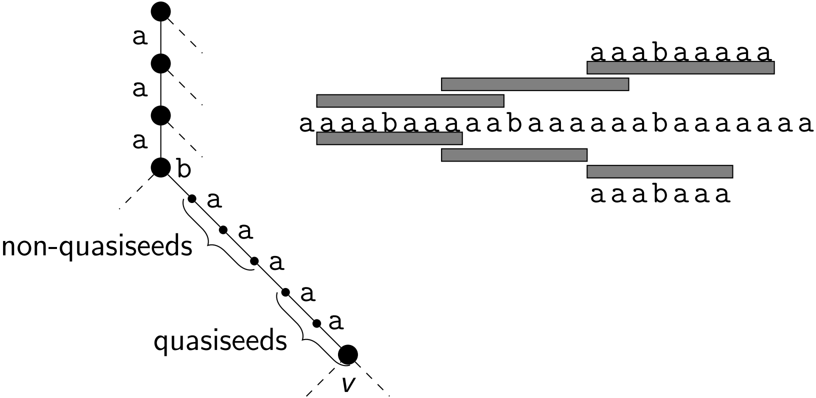
**Definition (Quasiseed)**

A subword $v$ is a *quasiseed* of $w$ if there are less than $|v|$ letters both before its first occurrence and after the last one and each letter between those two occurrences is covered by an occurrence of $v$.



# Useful properties of quasiseeds

An $O(n)$ representation on the suffix tree.



# Useful properties of quasiseeds

**Lemma (Restricted-Quasiseeds)**

*Given an integer $d$ and a word $w$ of length $n$, the representation of all quasiseeds of length in $\{d, d+1, \ldots, 2d\}$ can be found in $O(n)$ time.*

# Useful properties of quasiseeds

**Lemma (Restricted-Quasiseeds)**

*Given an integer $d$ and a word $w$ of length $n$, the representation of all quasiseeds of length in $\{d, d+1, \ldots, 2d\}$ can be found in $O(n)$ time.*

- The All-Seeds Problem can be linearly reduced to computing (the representation of) all quasiseeds.

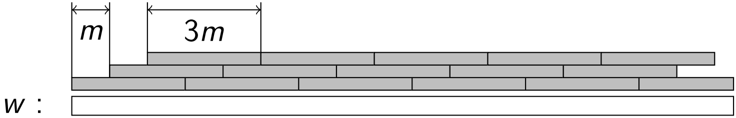
# Main problem

**Problem (All-Quasiseeds)**

*Given a word of length $n$, compute the representation of all its quasiseeds.*

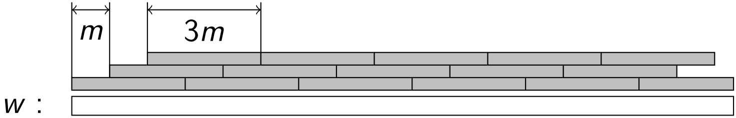
# Recursive structure of the algorithm

Interval $m$-staircase



# Recursive structure of the algorithm

Interval $m$-staircase




**Lemma (Short Quasiseeds)**

*A subword $v$ of length $< m$ is a quasiseed of $w$ if and only if it is a quasiseed of each subword corresponding to an $m$-staircase interval.*

# Recursive structure of the algorithm

- The total length of the intervals in the staircase (size of the staircase) is about $3n$.

# Recursive structure of the algorithm

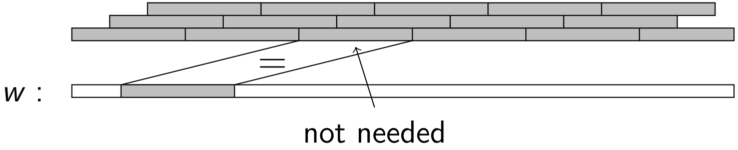
- The total length of the intervals in the staircase (size of the staircase) is about $3n$.
- If it were $\frac{1}{2}n$, the recursion could yield a linear algorithm.

# Recursive structure of the algorithm

- The total length of the intervals in the staircase (size of the staircase) is about $3n$.
- If it were $\frac{1}{2}n$, the recursion could yield a linear algorithm.
- We need to reduce the staircase.

# Recursive structure of the algorithm

- The total length of the intervals in the staircase (size of the staircase) is about $3n$.
- If it were $\frac{1}{2}n$, the recursion could yield a linear algorithm.
- We need to reduce the staircase.



# Recursive structure of the algorithm

Outline:

1. Find an appropriate reduced staircase
2. Find the long quasiseeds (non-recursively)
3. Find the short quasiseeds (recursive calls)
4. Merge the results of those calls

# Recursive structure of the algorithm

Outline:

1. Find an appropriate reduced staircase
2. Find the long quasiseeds (non-recursively)
3. Find the short quasiseeds (recursive calls)
4. Merge the results of those calls

**Main issue**: How to find an appropriate $m$, so that simultaneously:

- the reduced staircase is small,
- long quasiseeds can be found in O(n).

Due to the Restricted-Quasiseeds Lemma,
$m = \Theta(n)$ would suffice for the second part.

# Recursive structure of the algorithm

Outline:

1. Find an appropriate reduced staircase
2. Find the long quasiseeds (non-recursively)
3. Find the short quasiseeds (recursive calls)
4. Merge the results of those calls

**Main issue**: How to find an appropriate $m$, so that simultaneously:

- the reduced staircase is small,
- long quasiseeds can be found in O(n).

Due to the Restricted-Quasiseeds Lemma,
$m = \Theta(n)$ would suffice for the second part.

- Merging is not as easy as it may seem (RMQ and static find-union).

# $f$-factorization

- A variant of a well known LZ-factorization

# $f$-factorization

- A variant of a well known LZ-factorization

**Definition ($f$-factorization)**

An $f$-factorization $f_1 f_2 \ldots f_k$ of $w$ is constructed greedily: $f_i$ is either just the first occurrence of a letter or the longest prefix of the remaining suffix that is a subword of $f_1 \ldots f_{i-1}$.

$$\texttt{a} \,|\, \texttt{b} \,|\, \texttt{a} \,|\, \texttt{a b a} \,|\, \texttt{b a a b a} \,|\, \texttt{b a b} \,|\, \texttt{c} \,|\, \texttt{a}$$

# $f$-factorization

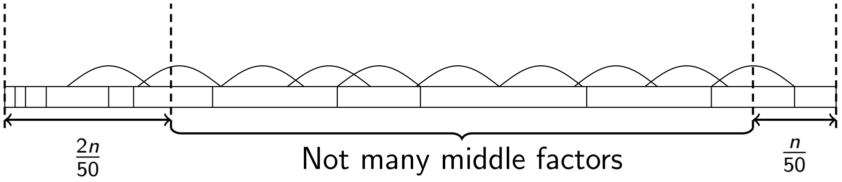
**Theorem (Crochemore, 1983; Crochemore et al. 2009)**

*The $f$-factorization over (constant) integer alphabet can be computed in $O(n)$ time.*

# Quasiseeds, staircase and factorization

**Lemma**

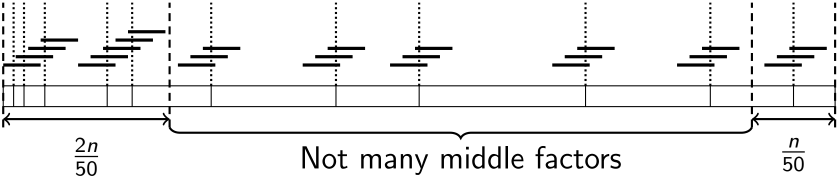
*Let $F$ be the $f$-factorization of $w$ ($|w| = n$) and $v$ be a quasiseed of $w$, $|v| < \frac{n}{50}$. Then at most $\left\lfloor \frac{2n}{|v|} \right\rfloor - 1$ factors from $F$ lie within $[\frac{2n}{50}, \frac{49n}{50}]$.*



# Quasiseeds, staircase and factorization

**Lemma**

*Let $F$ be the $f$-factorization of $w$ ($|w| = n$) and $v$ be a quasiseed of $w$, $|v| < \frac{n}{50}$. Then at most $\left\lfloor \frac{2n}{|v|} \right\rfloor - 1$ factors from $F$ lie within $[\frac{2n}{50}, \frac{49n}{50}]$.*




Stairs lying within a single factor are not necessary.

# Key lemmas

The algorithm does not know the quasiseed, but can find the number of middle factors.

# Key lemmas

The algorithm does not know the quasiseed, but can find the number of middle factors.
Let $g$ be the number of middle factors of the word $w$, $|w| = n > 200$.

**Lemma**

*There is no quasiseed $v$ of $w$ such that:*

$$\frac{2n}{g+1} < |v| \leq \frac{n}{50}.$$

# Key lemmas

The algorithm does not know the quasiseed, but can find the number of middle factors.
Let $g$ be the number of middle factors of the word $w$, $|w| = n > 200$.

**Lemma**

*There is no quasiseed $v$ of $w$ such that:*

$$\frac{2n}{g+1} < |v| \leq \frac{n}{50}.$$

**Lemma**

*If $m \leq \frac{n}{50(g+1)}$ then the size of the reduced staircase is $< \frac{n}{2}$.*

# Final structure of the algorithm

1. Find an $f$-factorization and the number of middle factors ($g$)
2. $m := \left\lfloor \frac{n}{50(g+1)} \right\rfloor$
3. Compute the reduced staircase
4. Compute the long quasiseeds (belonging to two ranges of fixed ratio)
5. If $m > 0$ compute the short quasiseeds by recursive calls and merge the results

# Conclusions

- We have presented a linear algorithm for the All-Quasiseeds Problem (over integer alphabet).
- This yields a linear algorithm for the All-Seeds Problem (over integer alphabet).

# Conclusions

- We have presented a linear algorithm for the All-Quasiseeds Problem (over integer alphabet).
- This yields a linear algorithm for the All-Seeds Problem (over integer alphabet).

Thank you!