Palindromic Length in Linear Time

Mikhail Rubinchik, Arseny M. Shur

Ural Federal University

Introduction

- Palindrome is a finite string w[1..n] equal to its reversal w[n] $\cdots$ w[2]w[1]
  - like the word rotator

Introduction

- Palindrome is a finite string w[1..n] equal to its reversal w[n] $\cdots$ w[2]w[1]
  - like the word rotator
- a simple and important type of repetitions in strings

Introduction

- Palindrome is a finite string w[1..n] equal to its reversal $w[n]\cdots w[2]w[1]$
  - like the word rotator
- a simple and important type of repetitions in strings
- a lot of attention in CS literature since 1970s
  - see Slisenko 1973; Manacher 1974; Knuth, Morris, Pratt 1975; Galil, Seiferas 1978 etc

Introduction

- Palindrome is a finite string w[1..n] equal to its reversal w[n] $\cdots$ w[2]w[1]
  - like the word rotator
- a simple and important type of repetitions in strings
- a lot of attention in CS literature since 1970s
  - see Slisenko 1973; Manacher 1974; Knuth, Morris, Pratt 1975; Galil, Seiferas 1978 etc
- important generalizations motivated by bioinformatics (involutive palindromes, gapped palindromes)

Definitions

- Palindromic factorization (PF) is the factorization of a string that contains only palindromes.
- abacaba = aba $\cdot$ c $\cdot$ aba is a PF, abacaba = abacaba is a PF too. But abac $\cdot$ aba is not a PF.

Definitions

- Palindromic factorization (PF) is the factorization of a string that contains only palindromes.
- abacaba $=$ aba $\cdot$ c $\cdot$ aba is a PF, abacaba $=$ abacaba is a PF too. But abac $\cdot$ aba is not a PF.

Definitions

- Palindromic factorization (PF) is the factorization of a string that contains only palindromes.
- abacaba = aba · c · aba is a PF, abacaba = abacaba is a PF too. But abac · aba is not a PF.
- Palindromic k-factorization is the a PF that contains exactly k palindromes.
- ababa = ababa is 1-factorization, a · b · aba is 3-factorization.

Definitions

- Palindromic factorization (PF) is the factorization of a string that contains only palindromes.
- abacaba = aba · c · aba is a PF, abacaba = abacaba is a PF too. But abac · aba is not a PF.
- Palindromic k-factorization is the a PF that contains exactly k palindromes.
- ababa = ababa is 1-factorization, a · b · aba is 3-factorization.
- Palindromic length (PL) of a string S is the minimal k such that the string S has a k-factorization. PL(abacaba) = 1, PL(baca) = 2 , PL(abaca) = 3

Two Problems about Palindromic Factorization

- Compute Palindromic Length online
  - The input string arrives symbol by symbol; for each new symbol the algorithm updates the palindromic length of the processed string

Two Problems about Palindromic Factorization

- Compute Palindromic Length online
  - The input string arrives symbol by symbol; for each new symbol the algorithm updates the palindromic length of the processed string
  - Simple solution: $O(n^2)$ time and $O(n)$ space by using dynamic programming. PL[i] is the palindromic length of the prefix of length i.

Two Problems about Palindromic Factorization

- Compute Palindromic Length online
  - The input string arrives symbol by symbol; for each new symbol the algorithm updates the palindromic length of the processed string
  - Simple solution: $O(n^2)$ time and $O(n)$ space by using dynamic programming. PL[i] is the palindromic length of the prefix of length i.
- k-factorization online
  - Simple solution: $O(kn^2)$ time and $O(kn)$ space by using dynamic programming. can[i][j] is the bit indicating whether a j-factorization exists for the string S[1..i].

Overview of Results on Factorization

Palindromic length

k-factorization

Overview of Results on Factorization

Palindromic length	k-factorization
2014 $O(n \log n)$	2015 $O(kn)$
Fici, Gagie, Karkkainen, Kempa	Kosolobov, Rubinchik, Shur

Overview of Results on Factorization

Palindromic length	k-factorization
2014 $O(n \log n)$ Fici, Gagie, Karkkainen, Kempa	2015 $O(kn)$ Kosolobov, Rubinchik, Shur
2016 $O(n \log n)$ Rubinchik, Shur.	2016 $O(n \log n)$ Rubinchik, Shur.

Overview of Results on Factorization

Palindromic length

2014 $O(n \log n)$
Fici, Gagie, Karkkainen, Kempa

2016 $O(n \log n)$
Rubinchik, Shur.

$O(n)$ — open problem

k-factorization

2015 $O(kn)$
Kosolobov, Rubinchik, Shur

2016 $O(n \log n)$
Rubinchik, Shur.

$O(n)$ — open problem

History of these problems

Palindromic length	k-factorization
2014 $O(n \log n)$ Fici, Gagie, Karkkainen, Kempa	2015 $O(nk)$ Kosolobov, Rubinchik, Shur
2016 $O(n \log n)$ Rubinchik, Shur.	2016 $O(n \log n)$ Rubinchik, Shur.
$O(n)$ — open problem	$O(n)$ — open problem

series

History of these problems

Palindromic length	k-factorization
	bit compression
2014 $O(n \log n)$ Fici, Gagie, Karkkainen, Kempa	2015 $O(nk)$ Kosolobov, Rubinchik, Shur
2016 $O(n \log n)$ Rubinchik, Shur.	2016 $O(n \log n)$ Rubinchik, Shur.
$O(n)$ — open problem	$O(n)$ — open problem

series

History of these problems

Palindromic length	k-factorization
	bit compression
2014 $O(n \log n)$ Fici, Gagie, Karkkainen, Kempa	2015 $O(nk)$ Kosolobov, Rubinchik, Shur
2016 $O(n \log n)$ Rubinchik, Shur.	2016 $O(n \log n)$ Rubinchik, Shur.
	series
$O(n)$ — open problem	$O(n)$ — open problem
this talk	

History of these problems

Palindromic length	k-factorization
	bit compression
2014 $O(n \log n)$ Fici, Gagie, Karkkainen, Kempa	2015 $O(nk)$ Kosolobov, Rubinchik, Shur
2016 $O(n \log n)$ Rubinchik, Shur.	2016 $O(n \log n)$ Rubinchik, Shur.
$O(n)$ — open problem	$O(n)$ — open problem

series

this talk

Palindromic series

- Let suf_1 be the largest suffix palindrome of the string, suf_2 be the second and ... suf_t the smallest suffix palindrome (one symbol).

Palindromic series

- Let suf_1 be the largest suffix palindrome of the string, suf_2 be the second and ... suf_t the smallest suffix palindrome (one symbol).
- $\mathrm{period}(\mathrm{suf}_1) \geq \mathrm{period}(\mathrm{suf}_2) \geq \ldots \geq \mathrm{period}(\mathrm{suf}_{t-1}) \geq \mathrm{period}(\mathrm{suf}_t)$

Palindromic series

- Let suf_1 be the largest suffix palindrome of the string, suf_2 be the second and ... suf_t the smallest suffix palindrome (one symbol).
- $\mathrm{period}(\mathrm{suf}_1) \geq \mathrm{period}(\mathrm{suf}_2) \geq \ldots \geq \mathrm{period}(\mathrm{suf}_{t-1}) \geq \mathrm{period}(\mathrm{suf}_t)$
- There are $O(\log n)$ different periods of suffix palindromes.

Palindromic series

- Let suf_1 be the largest suffix palindrome of the string, suf_2 be the second and ... suf_t the smallest suffix palindrome (one symbol).
- $\mathrm{period}(\mathrm{suf}_1) \geq \mathrm{period}(\mathrm{suf}_2) \geq ... \geq \mathrm{period}(\mathrm{suf}_{t-1}) \geq \mathrm{period}(\mathrm{suf}_t)$
- There are $O(\log n)$ different periods of suffix palindromes.
- Palindromic series is the set of suffix palindromes with the same period.

Palindromic series

- Let suf_1 be the largest suffix palindrome of the string, suf_2 be the second and ... suf_t the smallest suffix palindrome (one symbol).
- $\mathrm{period}(\mathrm{suf}_1) \geq \mathrm{period}(\mathrm{suf}_2) \geq \ldots \geq \mathrm{period}(\mathrm{suf}_{t-1}) \geq \mathrm{period}(\mathrm{suf}_t)$
- There are $O(\log n)$ different periods of suffix palindromes.
- Palindromic series is the set of suffix palindromes with the same period.
- All suffix palindromes can be stored within $O(\log n)$ space ($O(1)$ for each series)

Palindromic series

- Let suf_1 be the largest suffix palindrome of the string, suf_2 be the second and ... suf_t the smallest suffix palindrome (one symbol).
- $\mathrm{period}(\mathrm{suf}_1) \geq \mathrm{period}(\mathrm{suf}_2) \geq \ldots \geq \mathrm{period}(\mathrm{suf}_{t-1}) \geq \mathrm{period}(\mathrm{suf}_t)$
- There are $O(\log n)$ different periods of suffix palindromes.
- Palindromic series is the set of suffix palindromes with the same period.
- All suffix palindromes can be stored within $O(\log n)$ space ($O(1)$ for each series)
- Appending a symbol to the string, we can update the series list in $O(\log n)$ time This is the way $O(n \log n)$-time algorithms for palindromic factorization work

Bit compression for palindromic factorizations

- For k-factorization we have a $k \times n$ boolean matrix for dynamic programming.

Bit compression for palindromic factorizations

- For k-factorization we have a $k \times n$ boolean matrix for dynamic programming.
- We can replace it by an integer matrix of size $k \cdot (n/w) \leq k \cdot (n/\log n)$.
  - w is the number of bits in machine word
  - in the word-RAM model we assume that $w = O(\log n)$

Bit compression for palindromic factorizations

- For k-factorization we have a $k \times n$ boolean matrix for dynamic programming.
- We can replace it by an integer matrix of size $k \cdot (n/w) \leq k \cdot (n/\log n)$.
  - w is the number of bits in machine word
  - in the word-RAM model we assume that $w = O(\log n)$
- In [Kosolobov, Rubinchik, Shur, 2014] it was shown that this matrix can be updated in $O(kn)$ time

Bit compression for palindromic factorizations

- For k-factorization we have a $k \times n$ boolean matrix for dynamic programming.
- We can replace it by an integer matrix of size $k \cdot (n/w) \le k \cdot (n/\log n)$.
  - w is the number of bits in machine word
  - in the word-RAM model we assume that $w = O(\log n)$
- In [Kosolobov, Rubinchik, Shur, 2014] it was shown that this matrix can be updated in $O(kn)$ time
- For palindromic length, we have a size n integer array for dynamic programming
- We cannot compress it in a simple way.

Our result: Bit compression for palindromic length

Lemma

If S is a string of palindromic length k and c is a symbol, then the palindromic length of Sc is $k-1$, k, or $k+1$.

Our result: Bit compression for palindromic length

Lemma

If S is a string of palindromic length k and c is a symbol, then the palindromic length of Sc is $k-1$, k, or $k+1$.

- For string "abacabaaa", the array of palindromic lengths for all prefixes is 1, 2, 1, 2, 3, 2, 1, 2, 2.
- We can represent it like +1, -1, +1, +1, -1, -1, +1, 0. We can replace 0 to 00, +1 to 01, -1 to 10. So we can replace the integer array of size n to a bit array of size 2n.

Our result: Bit compression for palindromic length

Lemma

If S is a string of palindromic length k and c is a symbol, then the palindromic length of Sc is $k-1$, k, or $k+1$.

- For string "abacabaaa", the array of palindromic lengths for all prefixes is 1, 2, 1, 2, 3, 2, 1, 2, 2.
- We can represent it like +1, -1, +1, +1, -1, -1, +1, 0. We can replace 0 to 00, +1 to 01, -1 to 10. So we can replace the integer array of size n to a bit array of size 2n.

Our result: Bit compression for palindromic length

Lemma

If S is a string of palindromic length k and c is a symbol, then the palindromic length of Sc is $k - 1$, k, or $k + 1$.

- For string "abacabaaa", the array of palindromic lengths for all prefixes is 1, 2, 1, 2, 3, 2, 1, 2, 2.
- We can represent it like +1, -1, +1, +1, -1, -1, +1, 0. We can replace 0 to 00, +1 to 01, -1 to 10. So we can replace the integer array of size n to a bit array of size 2n.

Theorem

Palindromic length of a string can be found in O(n) time online.

Open problem

- The existence of a k-factorization can be decided in $O(kn)$ time or in $O(n \log n)$ time.

Open problem

- The existence of a k-factorization can be decided in $O(kn)$ time or in $O(n \log n)$ time.

Lemma

Given a k-factorization of string S of length n, it is possible, in $O(n)$ time, to factor S into $k + 2t$ palindromes for any positive integer t such that $k + 2t \leq n$.

Open problem

- The existence of a k-factorization can be decided in $O(kn)$ time or in $O(n \log n)$ time.

Lemma

Given a k-factorization of string S of length n, it is possible, in $O(n)$ time, to factor S into $k + 2t$ palindromes for any positive integer t such that $k + 2t \leq n$.

- We need to find "even palindromic length" and "odd palindromic length" in linear time.

Open problem

- The existence of a k-factorization can be decided in $O(kn)$ time or in $O(n \log n)$ time.

Lemma

Given a k-factorization of string S of length n, it is possible, in $O(n)$ time, to factor S into $k + 2t$ palindromes for any positive integer t such that $k + 2t \leq n$.

- We need to find "even palindromic length" and "odd palindromic length" in linear time.
- The difference between neighboring elements in the array of even (odd) palindromic lengths can be $\Omega(n)$

Open problem

- The existence of a k-factorization can be decided in $O(kn)$ time or in $O(n \log n)$ time.

Lemma

Given a k-factorization of string S of length n, it is possible, in $O(n)$ time, to factor S into $k + 2t$ palindromes for any positive integer t such that $k + 2t \leq n$.

- We need to find "even palindromic length" and "odd palindromic length" in linear time.
- The difference between neighboring elements in the array of even (odd) palindromic lengths can be $\Omega(n)$
- So we need some other trick

Open problem

- The existence of a k-factorization can be decided in $O(kn)$ time or in $O(n \log n)$ time.

Lemma

Given a k-factorization of string S of length n, it is possible, in $O(n)$ time, to factor S into $k + 2t$ palindromes for any positive integer t such that $k + 2t \leq n$.

- We need to find "even palindromic length" and "odd palindromic length" in linear time.
- The difference between neighboring elements in the array of even (odd) palindromic lengths can be $\Omega(n)$
- So we need some other trick

Open question

Is there a linear time algorithm for k-factorization.

Thank you for your attention!