Maximal Palindromic Factorization

Ali Alatabbi[1] Costas S. Iliopoulos[1] M. Sohel Rahman[2]

Ali.Alatabbi@kcl.ac.uk, csi@dcs.kcl.ac.uk, msrahman@cse.buet.ac.bd

[1]King's College London, United Kingdom.

[2]AℓEDA Group, Department of CSE, BUET, Dhaka-1000, Bangladesh.

Introduction

- The detection of palindromes is a classical and well-studied problem in computer science, language theory and algorithm design with a lot of variants arising out of different practical scenarios.
- Interestingly, in the seminal Knuth-Morris-Pratt paper presenting the well-known string matching algorithm [Knuth et al. (1977)], a problem related to palindrome recognition was also considered.
- Manacher discovered an on-line sequential algorithm that finds all *initial* palindromes in a string [Manacher (1975)].
- Gusfield gave a linear-time algorithm to find all *maximal* palindromes (a notion we define shortly) in a string [Gusfield (1997)].
- Porto and Barbosa gave an algorithm to find all *approximate* palindromes in a string [Porto and Barbosa (2002)].
- In word combinatorics, for example, studies have investigated the inhabitation of palindromes in Fibonacci words or Sturmian words in general [Droubay (1995)], [Droubay and Pirillo (1999)], [Glen (2006)].

Introduction

- Words with palindromic structure are important in DNA and RNA sequences.
- Palindromes play an important role in regulation of gene activity and other cell processes because these are often observed near promoters, introns and specific untranslated regions.
- Finding common palindromes in two gene sequences can be an important criterion to compare them, and also to find common relationships between them.
- However, in those applications, the reversal of palindromes should be combined with the complementarity concept on nucleotides

Introduction

Generic factorization process plays an important role in String Algorithms. The obvious advantage of such process is that when processing a string online, the work done on an element of the factorization can usually be skipped because already done on its previous occurrence.

Suppose, we are given a set of strings $\mathcal{S} = \{s_1, s_2, \ldots, s_k\}$, such that s_i is a substring of s and $1 \leq i \leq k$.
A factorization $\mathcal{F}$ of s with respect to $\mathcal{S}$ refers to a decomposition of s such that $s = s_{i_1} s_{i_2} \ldots s_{i_\ell}$ where $s_{i_j} \in \mathcal{S}$ and ℓ is minimum.
In this context the set $\mathcal{S}$ is referred to as the factorization set.

Notation and terminology

- A string or sequence is a succession of zero or more symbols from an alphabet Σ of cardinality σ, where σ expresses the number of distinct characters in the alphabet.
- The empty string is the empty sequence (of zero length) and is denoted by ϵ.
- The set of all strings over the alphabet Σ including ϵ is denoted by Σ^*.
- The set of all non-empty strings over the alphabet Σ is denoted by Σ^+. $\Sigma^* = \Sigma^+ \cup \epsilon$.
- A string s of length $|s| = n$ is represented by $s[1 \ldots n]$. The i-th symbol of s is denoted by $s[i]$.
- A string y is a factor of s if $s = xyz$ for $x, z \in \Sigma^*$; it is a prefix of s if x is empty and a suffix of s if z is empty.
- We denote by $s[i \ldots j]$ the factor of s that starts at position i and ends at position j.
- We denote by $\tilde{s}$ the reversal of s, i.e., $\tilde{s} = s[n]\ s[n-1]\ \ldots\ s[1]$.

Notation and terminology

- A palindrome is a symmetric string that reads the same forward and backward. More formally, s is called a palindrome if and only if $s = \tilde{s}$.
- The empty string ϵ is assumed to be a palindrome. Also note that a single character is a palindrome by definition.
- The radius of a palindrome s is $\frac{|s|}{2}$.
- Given a string s of length n, suppose $s[i \ldots j]$,with $1 \leq i \leq j \leq n$ is a palindrome, i.e., $s[i \ldots j]$ is a palindromic substring of s. Then, the center of the palindromic substring $s[i \ldots j]$ is $\lfloor \frac{i+j}{2} \rfloor$.
- The total number of occurrences of all palindromes in a word can be quadratic in the length of the word, as is the case for the word a^n.

Notation and terminology

- A palindromic factor $s[i \ldots j]$ is called the maximal palindrome at the center $\lfloor \frac{i+j}{2} \rfloor$ if no other palindromes at the center $\lfloor \frac{i+j}{2} \rfloor$ have a larger radius than $s[i \ldots j]$, i.e., if $s[i-1] \neq s[j+1]$, where $i = 1$, or $j = n$.
- A maximal palindrome $s[i \ldots j]$ is called a suffix (prefix) palindrome of s if and only if $j = n$ ($i = 1$). We denote by $(c, r)_s$ the maximal palindromic factor of a string s whose center is c and radius is r; or we simply drop the subscript and use (c, r) when the string s is clear from the context.
- The set of all center-distinct maximal palindromes of a string s is denoted by $\mathcal{MP}(s)$. Further, for the string s, we denote the set of all *prefix palindromes* (*suffix palindromes*) as $\mathcal{PP}(s)$ ($\mathcal{SP}(s)$).

Notation and terminology

- A palindromic factor $s[i \dots j]$ is called the maximal palindrome at the center $\lfloor \frac{i+j}{2} \rfloor$ if no other palindromes at the center $\lfloor \frac{i+j}{2} \rfloor$ have a larger radius than $s[i \dots j]$, i.e., if $s[i-1] \neq s[j+1]$, where $i = 1$, or $j = n$.
- A maximal palindrome $s[i \dots j]$ is called a suffix (prefix) palindrome of s if and only if $j = n$ ($i = 1$). We denote by $(c, r)_s$ the maximal palindromic factor of a string s whose center is c and radius is r; or we simply drop the subscript and use (c, r) when the string s is clear from the context.
- The set of all center-distinct maximal palindromes of a string s is denoted by $\mathcal{MP}(s)$. Further, for the string s, we denote the set of all *prefix palindromes* (*suffix palindromes*) as $\mathcal{PP}(s)$ ($\mathcal{SP}(s)$).

Proposition

The position i could be the center of at most two maximal palindromic factors, therefore; $\mathcal{MP}(s)[i]$ contains at most two elements, where $1 \leq i \leq n$, hence; there are at most $2n$ elements in $\mathcal{MP}(s)$.

Problem Definition

Problem (Maximal Palindromic Factorization (MPF))

Given a string s, find the maximal palindromic factorization of s, that is a factorization of s where the factorization set is $\mathcal{MP}(s)$.

Problem Definition

Problem (Maximal Palindromic Factorization (MPF))

Given a string s, find the maximal palindromic factorization of s, that is a factorization of s where the factorization set is $\mathcal{MP}(s)$.

We use the following result from Manacher (1975).

Theorem (Manacher (1975))

For any string s of length n, $\mathcal{MP}(s)$ can be computed in $O(n)$ time.

In what follows, we assume that the elements of $\mathcal{MP}(s)$ are sorted in increasing order of centers c. Actually, the algorithm of Manacher (1975) computes the elements of $\mathcal{MP}(s)$ in this order. Clearly, the set $\mathcal{PP}(s)$ and $\mathcal{SP}(s)$ can be computed easily during the computation of $\mathcal{MP}(s)$.

The Algorithm

On the other hand, we use $\mathcal{MPL}(s)[i]$ to denote the set of the lengths of all maximal palindromes ending at position i,where $1 \leq i \leq n$ in s.

$$\mathcal{MPL}(s)[i] = \{2\ell - 1 \mid s[i-\ell+1 \ldots i+\ell-1] \in \mathcal{MP}(s)\} \cup \{2\ell' \mid s[i-\ell' \ldots i+\ell'-1] \in \mathcal{MP}(s)\} \quad (1)$$

where $1 \leq i \leq n$, with 2ℓ and $2\ell' + 1$ are the lengths of the odd and even palindromic factors respectively.

Proposition

The set $\mathcal{MPL}(s)$ (Equation 1) can be computed in linear time from the set $\mathcal{MP}(s)$.

The Algorithm

$\mathcal{U}(s)[i]$ stores the position j such that $j+1$ is the starting position of a maximal palindromic factors ending at i and j is the end of another maximal palindromic substring.

Clearly, this can be easily computed once we have $\mathcal{MPL}(s)$ computed.

$$\mathcal{U}[i][j] = i - \mathcal{MPL}(s)[i][j] \tag{2}$$

The Algorithm

$\mathcal{U}(s)[i]$ stores the position j such that $j+1$ is the starting position of a maximal palindromic factors ending at i and j is the end of another maximal palindromic substring.

Clearly, this can be easily computed once we have $\mathcal{MPL}(s)$ computed.

$$\mathcal{U}[i][j] = i - \mathcal{MPL}(s)[i][j] \tag{2}$$

Given the list $\mathcal{U}(s)$ for a string s, we define a directed graph $\mathcal{G}_s = (\mathcal{V}, \mathcal{E})$ as follows. We have $\mathcal{V} = \{i \mid 1 \leq i \leq n\}$ and $\mathcal{E} = \{(i,j) \mid j \in \mathcal{U}(s)[i]\}$. Note that (i,j) is a directed edge where the direction is from i to j.

Now we can present the steps of our algorithm for computing the maximal palindromic factorization of a given string s of length n.

The Algorithm

MPF Algorithm: Maximal Palindromic Factorization Algorithm
Input: A String s of length n
Output: Maximal Palindromic Factorization of s

1: Compute the set of maximal palindromes $\mathcal{MP}(s)$ and identify the set of prefix palindromes $\mathcal{PP}(s)$.
2: Compute the list $\mathcal{MPL}(s)$.
3: Compute the list $\mathcal{U}(s)$.
4: Construct the graph $\mathcal{G}_s = (\mathcal{V}, \mathcal{E})$.
5: Do a breadth first search on $\mathcal{G}_s$ assuming the vertex n as the source.
6: Identify the shortest path $P \equiv n \rightsquigarrow v$ such that v is the end position of a palindrome belonging to $\mathcal{PP}(s)$. Suppose $P \equiv \langle n = p_k \ p_{k-1} \ \ldots \ p_2 \ p_1 = v \rangle$.
7: Return $s = s[1..p_1] \ s[p_1 + 1..p_2] \ \ldots \ s[p_{k-1} + 1..p_k]$.

The Algorithm

MPF Algorithm: Maximal Palindromic Factorization Algorithm
Input: A String s of length n
Output: Maximal Palindromic Factorization of s

1: Compute the set of maximal palindromes $\mathcal{MP}(s)$ and identify the set of prefix palindromes $\mathcal{PP}(s)$.
2: Compute the list $\mathcal{MPL}(s)$.
3: Compute the list $\mathcal{U}(s)$.
4: Construct the graph $\mathcal{G}_s = (\mathcal{V}, \mathcal{E})$.
5: Do a breadth first search on $\mathcal{G}_s$ assuming the vertex n as the source.
6: Identify the shortest path $P \equiv n \rightsquigarrow v$ such that v is the end position of a palindrome belonging to $\mathcal{PP}(s)$. Suppose $P \equiv \langle n = p_k \ p_{k-1} \ \ldots \ p_2 \ p_1 = v \rangle$.
7: Return $s = s[1..p_1] \ s[p_1 + 1..p_2] \ \ldots \ s[p_{k-1} + 1..p_k]$.

Theorem

Given a string s of length n, MPF Algorithm correctly computes the maximal palindromic factorization of s in $O(n)$ time.

Analysis

Running time:
In Step 1 the computation of $\mathcal{MP}(s)$ can be done using the algorithm of Manacher (1975) in $O(n)$ time. Also, $\mathcal{PP}(s)$ and $\mathcal{SP}(s)$ can be computed easily while computing $\mathcal{MP}(s)$. The computation of $\mathcal{MPL}(s)$ and $\mathcal{U}(s)$ in Step 2 and Step 3 can be done in linear time once $\mathcal{MP}(s)$ is computed.

There are in total n number of vertices in $\mathcal{G}_s$. The number of edges $|\mathcal{E}|$ of $\mathcal{G}_s$ depends on $\mathcal{U}(s)$. But it is easy to realize that the summation of the number of elements in all the positions of $\mathcal{U}(s)$ cannot exceed the total number of maximal palindromes. Now, since there can be at most $2n+1$ centers, there can be just as many maximal palindromes in s. Therefore we have $|\mathcal{E}| = O(n)$.

Hence, the graph construction (Step 4) as well as the breadth first search (Step 5) can be done in $O(|\mathcal{V}| + |\mathcal{E}|) = O(n)$ time.

Finally, the identification of the desired path in Step 6 can also be done easily if we do some simple bookkeeping during the breadth first search because we already have computed the sets $\mathcal{PP}(s)$ and $\mathcal{SP}(s)$ in Step 1. Hence the total running time of the algorithm is $O(n)$.

An Illustrative Example

Suppose we are given a string $s = abbcbbcbbbcbb$. We will proceed as follows:

First we compute the set $\mathcal{MP}(s)$. For example, at position $i = 9$ there are 2 palindromes of lengthes 2 and 9 centered at position 9 of s.

Secondly, we compute the set $\mathcal{MPL}(s)$. For example, at position $i = 9$ there are 3 palindromes of lengthes 2, 5 and 8 ending at position 9 of s.

Finally, we compute $\mathcal{U}(s)$.

Now, we can construct the graph $\mathcal{G}_s$. For example, we can see that from vertex $i = 9$ we have 3 directed edges, namely, $(9, 7), (9, 4)$ and $(9, 1)$. Our desired shortest path is $P = \langle 13, 4, 3, 1 \rangle$. So, the maximal palindromic factorization of $s = abbcbbcbbbcbb$ is as follows:

$$s[1..1]s[2..3]s[4..4]s[5..13] = a\ bb\ c\ bbcbbbcbb$$

.

An Illustrative Example

Table : Steps for computing $\mathcal{U}(s)$ and $\mathcal{MPL}(s)$ for $s = abbcbbcbbbcbb$

i	$\mathcal{MP}[i]$	$\mathcal{MPL}[i]$	$\mathcal{U}[i]$
1	$\mathcal{MP}[1] = \{(1,1)\}$	$\mathcal{MPL}[1] = \{1\}$	$\mathcal{U}[1] = \{0\}$
2	$\mathcal{MP}[2] = \{(2,2)\}$	$\mathcal{MPL}[2] = \{.\}$	$\mathcal{U}[2] = \{.\}$
3	$\mathcal{MP}[3] = \{(3,1)\}$	$\mathcal{MPL}[3] = \{2\}$	$\mathcal{U}[3] = \{1\}$
4	$\mathcal{MP}[4] = \{(4,5)\}$	$\mathcal{MPL}[4] = \{.\}$	$\mathcal{U}[4] = \{.\}$
5	$\mathcal{MP}[5] = \{(5,8)\}$	$\mathcal{MPL}[5] = \{.\}$	$\mathcal{U}[5] = \{.\}$
6	$\mathcal{MP}[6] = \{(6,1)\}$	$\mathcal{MPL}[6] = \{5\}$	$\mathcal{U}[6] = \{1\}$
7	$\mathcal{MP}[7] = \{(7,5)\}$	$\mathcal{MPL}[7] = \{.\}$	$\mathcal{U}[7] = \{.\}$
8	$\mathcal{MP}[8] = \{(8,2)\}$	$\mathcal{MPL}[8] = \{.\}$	$\mathcal{U}[8] = \{.\}$
9	$\mathcal{MP}[9] = \{(9,2)(9,9)\}$	$\mathcal{MPL}[9] = \{2,5,8\}$	$\mathcal{U}[9] = \{7,4,1\}$
10	$\mathcal{MP}[10] = \{(10,1)\}$	$\mathcal{MPL}[10] = \{2\}$	$\mathcal{U}[10] = \{8\}$
11	$\mathcal{MP}[11] = \{(11,5)\}$	$\mathcal{MPL}[11] = \{.\}$	$\mathcal{U}[11] = \{.\}$
12	$\mathcal{MP}[12] = \{(12,2)\}$	$\mathcal{MPL}[12] = \{.\}$	$\mathcal{U}[12] = \{.\}$
13	$\mathcal{MP}[13] = \{(13,1)\}$	$\mathcal{MPL}[13] = \{1,2,5,9\}$	$\mathcal{U}[13] = \{12,11,8,4)\}$

An Illustrative Example

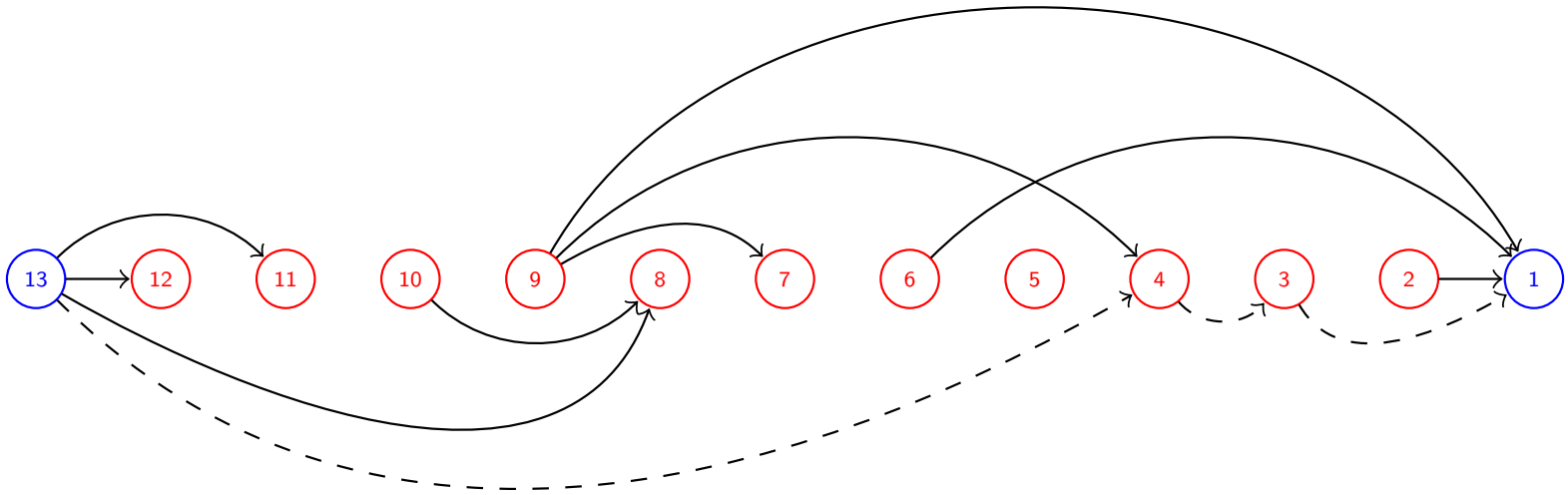

Figure : The graph $\mathcal{G}_s$ for $s = abbcbbcbbbcbb$

Conclusion

In this paper, we answer a recent question raised during StringMasters, Verona, Italy - 2013: **does there exist an algorithm to compute the maximal palindromic factorization of a finite string?**

We answer the previous question affirmatively by providing a linear-time algorithm that computes the *maximal palindromic factorization (MPF)* of a string.

An immediate target will be extending the algorithm to biological palindromes, where the word reversal is defined in conjunction with the complementarity of nucleotide letters: $c \leftrightarrow g$ and $a \leftrightarrow t$ (or $a \leftrightarrow u$, in case of RNA).

The proposed algorithm can be extended to find *maximal distinct palindromic factorization set*. We will focus on this problem in a future work.

Also we will work on studying palindromic cover of string and how can it be modeled using graphs.

The End

Thank You.

Bibliography

X. Droubay and G. Pirillo. Palindromes and Sturmian words. *Theoretical Computer Science*, 223: 73–85, 1999.

Xavier Droubay. Palindromes in the fibonacci word. *Inf. Process. Lett.*, 55(4):217–221, 1995.

Amy Glen. Occurrences of palindromes in characteristic sturmian words. *Theor. Comput. Sci.*, 352 (1-3):31–46, 2006.

Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, 1997.

D. E. Knuth, J. H. Morris Jr., and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6(2):323–350, 1977.

Glenn Manacher. A new linear-time on-line algorithm for finding the smallest initial palindrome of a string. *Journal of the ACM*, 22:346 – 351, July 1975.

Alexandre H L Porto and Valmir C Barbosa. Finding approximate palindromes in strings. *Pattern Recognition*, 2002.