

Optimal superprimitivity testing for strings *

Alberto Apostolico **

Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA
and Dipartimento di Matematica Pura e Applicata, University of L'Aquila, L'Aquila, Italy

Martin Farach

Department of Computer Science, University of Maryland, College Park, MD 20742, USA

Costas S. Iliopoulos ***

Department of Computer Science, Royal Holloway College, University of London, Egham, United Kingdom

Communicated by L. Boasson

Received 3 January 1991

Revised 12 April 1991

Abstract

Apostolico, A., M. Farach and C.S. Iliopoulos, Optimal superprimitivity for strings, Information Processing Letters 39 (1991) 17–20.

A string w covers another string z if every position of z is within some occurrence of w in z . Clearly, every string is covered by itself. A string that is covered only by itself is *superprimitive*. We show that the property of being superprimitive is testable on a string of n symbols in $O(n)$ time and space.

Keywords: Combinatorial problems, design of algorithms, analysis of algorithms, algorithms on words, superprimitive strings, period of a string, quasiperiod of a string

1. Introduction

Regularities in strings model many phenomena and thus form the subject of extensive mathemati-

cal studies (see, e.g., [3]). Some regularities, e.g., square substrings, are *avoidable* in the sense that we can build indefinitely long strings that are immune from that regularity; others are *unavoidable*. Perhaps the most conspicuous regularities in strings are those that manifest themselves in the form of repeated subpatterns. Recall that a word x is *primitive* if setting $x = s^k$ implies $k = 1$. A primitive string w is a *period* of another string z if $z = w^c w'$ for some integer $c > 0$ and w' a possibly empty prefix of w . A string x is *periodic* if z has a period w such that $|w| \leq |z|/2$. It is a well-known fact of combinatorics on words that a periodic string has precisely one period [4]. In general, we refer to the shortest period of a string as *the period* of that string.

* This research was supported, through the Leonardo Fibonacci Institute, by the Istituto Trentino di Cultura, Trento, Italy.

** Additional support provided in part by the National Research Council of Italy, by NSF Grant CCR-89-00305, by NIH Library of Medicine Grant R01 LM05118, by ASOFR Grant 90-0107, by the Italian Ministry of Research and by NATO Grant CRG 900293.

*** Additional support provided in part by NATO Grant CRG 900293, by UK SERC GR/F, and by a Royal Society Grant.

In this paper, we concentrate on another form of regularity in strings, called *quasiperiodicity*, which was recently introduced and studied in [1]. The following definitions clarify this notion.

Definition. A string w *covers* another string z if for every $i \in \{1, \dots, |z|\}$ there exists a $j \in \{1, \dots, |w|\}$ such that there is an occurrence of w starting at position $i - j + 1$ in string z .

Informally, a string w covers another string z if every position of z occurs within some occurrence of w in z . Clearly, every string is covered by itself.

Definition. If z is covered by $w \neq z$, the z is *quasiperiodic*, and the ordered sequence of all occurrences of w in z is called the *w-cover* of z .

For example, the string

$z = abaababababaaba$

is quasiperiodic since it can be obtained by the concatenation and superposition of five instances of $w = aba$. A periodic string is always quasiperiodic, but the converse is not true.

Definition. A string z is *superprimitive* if it is not quasiperiodic.

Clearly, a superprimitive string is also primitive. However, the converse is not true. For example, aba is superprimitive and also primitive, but $abaabaab$ is primitive but not superprimitive, since the string $abaab$ covers it. Clearly, for any string z there is always some superprimitive string q that covers z . String q is a *quasiperiod* for z . It turns out ([1], cf. also Corollary 3 below) that every string has a unique quasiperiod. It is easy to check that if a string contains some quasiperiodic substring then it must also contain a *square*, i.e., a substring in the form ww . As is well known [3], squares are avoidable regularities in strings, whence also quasiperiodicities are such. Finding the period of a string (hence, in particular, checking whether that string is periodic or has a square prefix) takes linear time by known methods (see, e.g., [5]). On the other hand, there are optimal $\Theta(n \log n)$ algorithms for detecting all squares in

a string x of n symbols (see, e.g., [2]). In [1], it is shown that all maximal quasiperiodic substrings of a string x of n symbols can be identified in time $O(n \log^2 n)$. A natural question concerns then the complexity of finding the quasiperiod of a string.

In this paper, we give an optimal, linear-time algorithm for testing whether a string is superprimitive. If x is not superprimitive, our algorithm returns the quasiperiod s of x . We also denote s by $Q(x)$. Thus, a string x is superprimitive if $|Q(x)| = |x|$. Note that the original string x can be produced by repeated duplication and concatenation (with possible overlap) of $Q(x)$.

2. Some combinatorial properties

Recall that a string u is a *border* of string x if u is simultaneously a prefix and a suffix of x . A border u of x is *nontrivial* if $u \neq x$. The longest nontrivial border of x is denoted by $B(x)$. By convention, we refer to $B(x)$ as *the* border of x and to any border as *a* border of x . Let s_x be a quasiperiod of x .

Theorem 1. If y is a border of x and $|y| \geq |s_x|$, then s_x covers y .

Proof. Since $|y| \geq |s_x|$ and s_x is a border of x , then s_x is also a border of y . We distinguish two cases:

Case 1: $|y| \leq 2|s_x|$. Then, every symbol of y is covered by at least one of the two occurrences of s_x that start at positions 1 and $|y| - |s_x| + 1$ of y , respectively.

Case 2: $|y| > 2|s_x|$. Then, there exists some string u such that $y = s_x u s_x$. However, since s_x covers x , we know that every symbol in u is covered by an occurrence of s_x . Therefore, s_x covers y . \square

Theorem 2. If y is a border of x and $|y| \geq |s_x|$, then, for any quasiperiod s_y of y , $s_y = s_x$.

Proof. Assume first that $|s_y| \geq |s_x|$. Since s_y is a border of y and y is a border of x , then s_y is a border of x . Having assumed $|s_y| \geq |s_x|$, we

have by Theorem 1 that s_x covers s_y . Assume now $|s_y| \leq |s_x|$. Then, s_x is a border of y and $|s_x| \geq |s_y|$, whence, by Theorem 1, s_y covers s_x . In either case, one has that $s_x = s_y$, since s_x and s_y are both superprimitive by definition of a quasiperiod. \square

Corollary 3. *Any word x has a unique quasiperiod (denoted by $Q(x)$).*

Proof. Assume that x has two distinct quasiperiods, denoted s_x and s'_x , and assume to fix the ideas that $|s_x| > |s'_x|$. Let y be a border of x such that $|y| \geq |s_x|$. By Theorem 2, we have that $s_y = s_x$ and $s'_y = s'_x$. Therefore, $s_x = s'_x$. \square

Lemma 4. $Q(B(x)) = Q(x)$.

Proof. Since $B(x)$ is the border of x , then it has maximum length among all nontrivial borders and in particular it is not shorter than $Q(x)$. The claim then follows from Theorem 2. \square

Let $P(x)$ be defined for string x as follows. If x is primitive, then $P(x) = x$. If x is periodic, then let $x = u^k u'$, where u is the period of x and $u' \neq u$ is a prefix of u . Then, $P(x) = uu'$.

Lemma 5. $P(x)$ has the following properties:

- (1) $P(x)$ covers x .
- (2) $Q(P(x)) = Q(x)$.
- (3) If x is periodic, $|P(x)| < \frac{2}{3}|x|$.
- (4) If x is periodic,

$$|P(x)| = |x| - |B(x)| + \text{REM}(|x|, |x| - |B(x)|),$$

where REM is the remainder function of integer division.

Proof. If x is not periodic, then (1) and (2) follow trivially. Assume that x is periodic and let $x = u^k u'$, where u is the period of x .

(1) The string $P(x) = uu'$ clearly covers $u^2 u'$ because u' is a prefix of u and therefore an occurrence of uu' can overlap with another to produce $u^2 u'$. The claim follows by induction.

(2) Clearly, $P(x)$ is a border of x . Recall that $Q(x)$ is unique. If $P(x)$ is superprimitive, then,

since $P(x)$ covers x (by (1)), one has that $P(x) = Q(x)$. If, on the other hand, $P(x)$ is not superprimitive, then $|P(x)| > |Q(x)|$. In either case, $|P(x)| \geq |Q(x)|$ whence, by Theorem 2, $Q(P(x)) = Q(x)$.

(3) The following chain of inequalities yields the claim:

$$\begin{aligned} |u'| &< |u|, \\ 3|u| + |u'| &< 4|u| \leq 2k|u|, \quad k \geq 2, \\ 3|u| + 3|u'| &< 2k|u| + 2|u'|, \\ 3|uu'| &< 2|u^k u'|, \\ |P(x)| = |uu'| &< \frac{2}{3}|u^k u'| = \frac{2}{3}|x|. \end{aligned}$$

(4) It is a well-known property of periodic words (see, e.g., Proposition 1.3.4 in [3]) that $|u| = |x| - |B(x)|$. We show next by substitution that $|P(x)| = |u| + \text{REM}(|x|, |u|)$. Since $x = u^k u'$, we get:

$$\begin{aligned} P(x) &= uu', \\ |P(x)| &= |u| + |u'|, \\ |P(x)| &= |u| + \text{REM}(k|u| + |u'|, |u|), \\ |P(x)| &= |u| + \text{REM}(|u^k u'|, |u|), \\ |P(x)| &= |u| + \text{REM}(|x|, |u|). \quad \square \end{aligned}$$

3. The algorithm

The algorithm to find the quasiperiod of a string x consists of a succession of stages in each of which smaller and smaller prefixes of x are considered. Upon completion of the first stage, either x is determined to be superprimitive or a border of x having the same candidate quasiperiod as x is identified. This border is guaranteed to have length at most $\frac{2}{3}|x|$. We then recurse on this border. The amount of work done at each stage is linear in the length of the border being considered and such a length is reduced by a constant fraction at each stage. Therefore, the total work is linear in $|x|$.

The algorithm contains a preprocessing phase which computes a table called FL where $FL(i)$ is the length of the border of the i th prefix $x_1 x_2 \dots x_i$.

of x . This table is a well-known tool of fast string searching strategies (see, e.g., [5]), in which context it is sometimes called *failure function*. Building FL requires time linear in $|x|$. In our construction, FL is used to determine the borders of various prefixes of x and to find the periods of these prefixes. Note that we only need to construct one global copy of FL . The recursive body of our procedure handles a border $x_1 \dots x_m$ of the input string $x_1 \dots x_n$ as follows.

FIND-CANDIDATE($x_1 \dots x_m$)

Let $b \leftarrow FL(m)$

If $b = 0$

then Return $x_1 \dots x_m$

If $b > \frac{1}{2}n$

then Let $b \leftarrow m - b + REM(m, m - b)$

Let $s \leftarrow \text{FIND-CANDIDATE}(x_1 \dots x_b)$

if $\text{TEST-CANDIDATE}(s, x_1 \dots x_m)$

then Return s

else Return $x_1 \dots x_m$

End FIND-CANDIDATE

TEST-CANDIDATE($s, x_1 \dots x_b$)

Compute the list $M = \{m_1, \dots, m_t\}$ of positions of the occurrences of s in $x_1 \dots x_b$.

For each adjacent pair of matches, m_i and m_{i+1} , do the following:

If $m_{i+1} - m_i > |s|$

Return *FALSE*

Return *TRUE*

End TEST-CANDIDATE

The algorithm is actuated on input string $x_1 \dots x_n$ by calling **FIND-CANDIDATE**($x_1 \dots x_n$). The correctness of the algorithm is centered around Theorems 1 and 2. Lemma 5 is used in the procedure **FIND-CANDIDATE** only to reduce the work. Let a *P-border* of x be any of the borders of x considered by the procedure. The basic invariant condition at each step of the recursion where $b \neq 0$ is that, immediately prior to the execution of **TEST-CANDIDATE**, the string s being considered is known to be the quasiperiod of $x_1 \dots x_b$. By Theorem 2, if s covers $x_1 \dots x_n$, then s must cover $x_1 \dots x_m$ (as well as all other *P-borders* of $x_1 \dots x_n$ of length larger than m).

Otherwise, the next shortest candidate quasiperiod for $x_1 \dots x_n$ is the *P-border* $x_1 \dots x_m$ itself.

Consider now the time complexity of the procedure. As is well known (see, e.g., [5]), the table FL can be computed in linear time. This table is computed only once so that the preprocessing takes time linear in $|x|$. At each stage of the recursion, all operations of **FIND-CANDIDATE** except for the execution of the **TEST-CANDIDATE** take constant time. However, the list M can be computed by any linear-time string searching algorithm, e.g., that in [5], after which **TEST-CANDIDATE** also takes time linear in the border of x being considered. Since the lengths of the borders considered at successive stages are in a fixed fraction progression, the total work involved in all executions of **TEST-CANDIDATE** also adds up to time linear in $|x|$. Note that the algorithm also generates the cover of x by its quasiperiod.

Theorem 6. *The quasiperiod w and the corresponding w -cover of a string x of n symbols can be computed in $O(n)$ time and space.*

Acknowledgment

The study reported in this paper was stimulated by discussions with A. Ehrenfeucht, to whom we wish to express our gratitude. We are also grateful to the referee for exposing and correcting an inconsistency in our earlier draft, and for other helpful comments.

References

- [1] A. Apostolico and A. Ehrenfeucht, Efficient detection of quasiperiodicities in strings, *Fibonacci Report* 90.6 (1990), submitted for publication.
- [2] A. Apostolico and F.P. Preparata, Optimal off-line detection of repetitions in a string, *Theoret. Comput. Sci.* **22** (1983) 297–315.
- [3] M. Lothaire, *Combinatorics on Words* (Addison-Wesley, Reading, MA, 1983).
- [4] R.C. Lyndon and M.P. Shützenberger, The equation $a^M = b^{N_c}P$ in a free group, *Michigan Math. J.* **9** (1962) 289–298.
- [5] D.E. Knuth, J.H. Morris and V.R. Pratt, Fast pattern matching in strings, *SIAM J. Comput.* **6** (2) (1977) 323–350.