1990

Optimal Superprimitivity Testing for Strings

Alberto Apostolico

Martin Farach

Costas S. Iliopoulos

Report Number:
90-1049

OPTIMAL SUPERPRIMITIVITY TESTING FOR STRINGS

Alberto Apostolico
Martin Farach
Costas S. Iliopoulos

CSD-TR-1049
November 1990

Optimal Superprimitivity Testing for Strings *

Alberto Apostolico† Martin Farach‡
Costas S. Iliopoulos§

November 28, 1990

Fibonacci Report 90.6
August 1990 - Revised November 1990

Abstract

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.

Key Words: Combinatorial Algorithms on Words, Superprimitive Strings, Period of a String, Quasiperiod of a String.

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

†Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA and Dipartimento di Matematica Pura e Applicata, University of L' Aquila, Italy, axa@cs.purdue.edu. Additional support for this author was 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 AFOSR Grant 90-0107, and by NATO Grant CRG 900293.

‡University of Maryland, Department of Computer Science, College Park, MD 20742, USA, mpf@cs.umd.edu

§University of London, Royal Holloway College, Department of Computer Science, Egham, England, csi@cs.rhbnc.ac.uk Additional support for this author was provided in part by NATO Grant CRG 900293, by UK SERC GR/F 00898, and by a Royal Society Grant.

1 Introduction

Regularities in strings model many phenomena and thus form the subject of extensive mathematical 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 z 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 string can be periodic in only one period [4]. We refer to the shortest period of a string as *the* period of that string.

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 $i \in \{1, \ldots, |z|\}$ there exists a $j \in \{1, \ldots, |w|\}$ such that there is an occurrence of w starting at position $i - j + 1$ in string x.

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 $w \neq z$, then 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 5 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 theorems 1-2 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 so are 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 otehr 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 q of x. We denote q 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.

Theorem 1 *If y is a border of x and $|y| \geq |Q(x)|$, then $Q(x)$ covers y.*

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

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

2. $|y| > 2|Q(x)|$: Then, there exists some string u such that $y = Q(x)uQ(x)$. However, since $Q(x)$ covers x, we know that every symbol in u is covered by an occurrence of $Q(x)$. Therefore, $Q(x)$ covers y.□

Theorem 2 *If y is a border of x and $|y| \geq |Q(x)|$ then $Q(y) = Q(x)$.*

Proof: By Theorem 1, $Q(y)$ covers $Q(x)$, since $Q(x)$ is a border of y. However, $Q(x)$ is superprimitive, hence $Q(y) = Q(x)$.□

Lemma 1 $Q(B(x)) = Q(x)$.

Proof: Since $B(x)$ is the border of x, then it has maximum length among all nontrivial borders and in particular it is no shorter than $Q(x)$. The claim then follows from Theorem 2.□

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 2 *$P(x)$ 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, $|P(x)| = |x| - |B(x)| + REM(|x|, |x| - |B(x)|)$, 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^2u' because u' is a prefix of u and therefore an occurrence of uu' can overlap with another to produce u^2u'. The claim follows by induction.

2. $P(x)$ covers x (by Part 1), and thus is a border of x. Also, $|P(x)| \geq |Q(x)|$. Then, by Theorem 2, $Q(P(x)) = Q(x)$.

3. The following chain of inequalities yield the claim:

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

4. We show first that $|u| = |x| - |B(x)|$. ¿From the fact that $|u^{k-1}u'|$ is a border x, we get $|B(x)| \geq |u^{k-1}u'| = |x| - |u|$. Therefore, $|u| \geq |x| - |B(x)|$.

   Let $d = |x| - |B(x)|$, and let $DIV(p/q)$ denote the integer division of p by q. Then, for $1 \leq i \leq d$, and for $c = DIV(|x|, d)$, $x_i = x_{i+d} = x_{i+2d} = \cdots = x_{i+cd}$. Therefore, if we let $v = x_1 \ldots x_d$ and $v' = x_1 \ldots x_{REM(|x|,d)}$, we have $x = v^c v'$. But then $d = |v| \geq |u|$, since u is the shortest string with this property, and therefore $|u| \leq |x| - |B(x)|$. Since $|u|$ must be both neither larger nor smaller than $|x| - |B(x)|$, we have $|u| = |x| - |B(x)|$.

   By substitution, we would like to show that $|P(x)| = |u| + REM(|x|, |u|)$. But $x = u^k u'$, so we get:

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

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 $x_1x_2\ldots x_i$ of x. This table is a well known tool of fast string searching strategies (see, e.g., [5]), in which context it is called sometimes *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 $x_1\ldots x_m$ of the input string $x_1\ldots x_n$ as follows.

FIND-CANDIDATE($x_1\ldots x_m$)
Let $b \leftarrow FL(m)$
If $b = 0$
  then Return $x_1\ldots x_m$
If $b > \frac{1}{2}n$
  then Let $b \leftarrow m - b + REM(m, m-b)$
Let $s \leftarrow$ FIND-CANDIDATE($x_1\ldots x_b$)
  If TEST-CANDIDATE($x_1\ldots x_m$,s)
    then Return s
    else Return $x_1\ldots x_m$
End FIND-CANDIDATE

TEST-CANDIDATE($x_1\ldots x_m$,s)
Compute the list $M = \{m_1, \ldots, m_t\}$ of positions of the occurrences of s in $x_1\ldots x_b$.
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

A call FIND-CANDIDATE($x_1 \ldots x_n$) actuates the algorithm on input string $x_1 \ldots x_n$. The correctness of the algorithm is centered around theorems 1 and 2. Lemma 2 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 is that, immediately prior to the execution of TEST-CANDIDATE, the string s being considered is known to be the quasiperiod of $|x_1 \ldots x_b|$. By Theorem 2, if s covers $x_1 \ldots x_n$ then s must cover $x_1 \ldots x_m$ (as well as all other P-borders of $x_1 \ldots x_n$ of length larger than m). Otherwise, the next shortest candidate quasiperiod for $x_1 \ldots x_n$ is P-border $x_1 \ldots x_m$ 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 3 *The quasiperiod w and the corresponding w-cover of a string x of n symbols can be computed in $O(n)$ time and space.*

References

1. Apostolico, A. and A. Ehrenfeucht (1990), "Efficient Detection of Quasiperiodicities in Strings", Fibonacci Report 90.5, submitted for publication.

2. Apostolico, A. and F.P. Preparata (1983), "Optimal Off-line Detection of Repetitions in a String", *Theoretical Computer Science* **22**, 297-315.

3. Lothaire, M (1983), *Combinatorics on Words*, Addison-Wesley, Reading, Mass.

4. Lyndon, R.C. and M.P. Shutzenberger (1962), "The Equation $a^M = b^N c^P$ in a Free Group", *Michigan Mathematical Journal* 9, 289-298.

5. Knuth, D.E., J.H. Morris and V.R. Pratt (1977), "Fast Pattern Matching in Strings", *SIAM Journal on Computing* 6, 2, 323-350.