An On-Line String Superprimitivity Test

Dany Breslauer*
Columbia University

CUCS-022-92

Abstract

A string w *covers* another string z if every symbol of z is within some occurrence of w in z. A string is called *superprimitive* if it is covered only by itself, and *quasiperiodic* if it is covered by some shorter string. We present an on-line linear-time algorithm that tests if each prefix of an input string is superprimitive while the string is given a symbol at a time.

1 Introduction

Quasiperiodicity, as defined by Apostolico and Ehrenfeucht [1], is an avoidable regularity of strings that is strongly related to other regularities such as periods and squares [4]. Apostolico, Farach and Iliopoulos [2] gave a linear time recursive algorithm that tests if a string is superprimitive. Apostolico and Ehrenfeucht [1] presented an algorithm that finds all maximal quasiperiodic substrings of a string.

In this paper we present a new algorithm that tests if a string is superprimitive. The algorithm is simpler and more efficient than the algorithm of Apostolico, Farach and Iliopoulos [2]. It is also *on-line*, which means that it tests if each prefix of the input string is superprimitive as soon that the prefix is given. The algorithm works under the general alphabet assumption where the only access to the input string is by comparisons of pairs of symbols. Similarly to the algorithm given by Apostolico, Farach and Iliopoulos [2] the new algorithm uses the Knuth-Morris-Pratt [3] string matching algorithm. However, that string matching algorithm is used only once.

We proceed with some definitions of terms that will be used in the sequel. A string w *covers* a string z if for every position $i \in \{1, \cdots, |z|\}$ of z there exists an occurrence of w starting at some position j of z such that $1 \leq j \leq i \leq j + |w| - 1 \leq |z|$. A string z is called *superprimitive* if it is covered only by itself, and *quasiperiodic* if it is covered by a string w such that $w \neq z$. A superprimitive string w that covers a string z is called a *quasiperiod* of z.

*Partially supported by an IBM Graduate Fellowship.

A string z has a *period* w if z is a prefix of w^k for some integer k. Alternatively, a string w is a period of a string z if $z = w^l v$ and v is a possibly empty prefix of w. The shortest period of a string z is called *the period* of z. Clearly, a string is always a period of itself.

We say that a non-empty string w is a *border* of a string z if z starts and ends with an occurrence of w. That is, $z = uw$ and $z = wv$ for some possibly empty strings u and v. Clearly, a string is always a border of itself. This border is called the trivial border.

We describe next few facts that will be used in our algorithm.

Fact 1.1 *A string z has a period of length π, such that $\pi < |z|$, if and only if it has a non-trivial border of length $k - \pi$.*

Proof: Immediate from the definitions of a border and a period. $\square$

Fact 1.2 *If a string w covers a string z then w is a border of z.*

Proof: Since the first symbol of z must be covered by w, the string z must start with an occurrence of w. Since the last symbol of z must also be covered by w, the string z must also end with an occurrence of w. That is, w is a border of z. $\square$

Note that by the last lemma any cover of a string z can be represented by a single integer that is the length the border of z.

Fact 1.3 *If a string w covers a string z and another string v, such that $|w| \leq |v|$, is a border of z then w covers also v.*

Proof: Given any prefix of z, it is covered by w except possibly at most the last $|w| - 1$ symbols of the prefix. Similarly, given any suffix of z, it is covered by w except possibly at most the first $|w| - 1$ symbols of the suffix. Since v is a border of z it is both a prefix and a suffix and it must be covered by w. $\square$

Fact 1.4 *If a string z is covered by two strings w and v, such that $|w| \leq |v|$, then w covers v. Therefore, a string cannot have two different quasiperiods w and v.*

Proof: By Fact 1.2 if v covers z then v is a border of z. By Fact 1.3 w covers v and if $w \neq v$ then v is quasiperiodic. $\square$

2 The on-line algorithm

In this section we describe an on-line algorithm that tests if each prefix $\mathcal{S}_{[1..k]}$ of an input string $\mathcal{S}_{[1..n]}$ is quasiperiodic. Similarly to the algorithm of Apostolico, Farach and Iliopoulos [2], if a prefix $\mathcal{S}_{[1..k]}$ is not superprimitive the algorithm finds the quasiperiod of that prefix.

The new algorithm will use the pattern processing step of the Knuth-Morris-Pratt [3] string matching algorithm. In the pattern processing step, the Knuth-Morris-Pratt algorithm computes the periods of all prefixes of the pattern while the pattern is given on-line. We do not include the details of the Knuth-Morris-Pratt algorithm, but rather summarize its properties in the following theorem:

Theorem 2.1 *(Knuth, Morris and Pratt [3]) There exists an on-line algorithm that finds the period of each prefix of a string $\mathcal{S}_{[1..n]}$ while the string is given a symbol at a time. The algorithm takes $O(n)$ time and uses linear auxiliary space. The number of comparisons performed by the algorithm is at most $2n$.*

```
– The Q[i] array stores the quasiperiod of any prefix S[1..i].
– The R[i] array stores the longest prefix of S, up to k, that is
–     covered by a superprimitive prefix S[1..i].
    k = 1
    while k ≤ n do
        Compute the period π_k of S[1..k] using the Knuth-Morris-Pratt
            string matching algorithm.
        – If the prefix S[1..k] has a non-trivial border check if the quasiperiod
        –     of that border covers the whole prefix.
        if π_k < k and R[Q[k − π_k]] ≥ k − Q[k − π_k] then
            – If the quasiperiod of the border S[1..k−π_k] covers the whole
            –     prefix S[1..k], then it is also the quasiperiod of S[1..k].
            Q[k] = Q[k − π_k]
            R[Q[k]] = k
        else
            – If the prefix S[1..k] does not have any non-trivial border
            –     or if the quasiperiod of the border does not cover the whole
            –     prefix S[1..k] then it is superprimitive.
            Q[k] = k
            R[k] = k
        end
        k = k + 1
    end
```

Figure 1: The new algorithm.

The new algorithm scans the input string $\mathcal{S}_{[1..n]}$ one symbol at a time. It maintains two arrays: $\mathcal{Q}[i]$ and $\mathcal{R}[i]$. The $\mathcal{Q}[i]$ array stores the quasiperiod of any prefix $\mathcal{S}_{[1..i]}$, for $1 \leq i \leq k$. The $\mathcal{R}[i]$ array stores only for superprimitive prefixes $\mathcal{S}_{[1..i]}$, the longest prefix of $\mathcal{S}_{[1..k]}$ that is covered by $\mathcal{S}_{[1..i]}$. Note that a prefix $\mathcal{S}_{[1..i]}$ is superprimitive if and only if $\mathcal{Q}[i] = i$. When the algorithm proceeds to the next symbol $\mathcal{S}_{[k]}$ it has to compute the quasiperiod of $\mathcal{S}_{[1..k]}$ and store it in $\mathcal{Q}[k]$.

As soon that the next input symbol $\mathcal{S}_{[k]}$ is reached, the algorithm calls the Knuth-Morris-Pratt algorithm to find the period length π_k of the prefix $\mathcal{S}_{[1..k]}$. The only comparisons of input symbols are performed in these calls to the Knuth-Morris-Pratt algorithm. Once the period of the prefix $\mathcal{S}_{[1..k]}$ is given, the algorithm can proceed to compute the values of $\mathcal{Q}[k]$ and $\mathcal{R}[\mathcal{Q}[k]]$ based only on the values of π_k, $\mathcal{Q}[1..k-1]$ and $\mathcal{R}[1..k-1]$. There are two cases:

1. If $\pi_k = k$ then by Fact 1.1 the prefix $\mathcal{S}_{[1..k]}$ has no non-trivial border. By Fact 1.2 any string that covers $\mathcal{S}_{[1..k]}$ must also be a border. Thus, the prefix $\mathcal{S}_{[1..k]}$ is covered only by itself and therefore it is superprimitive. In this case we define $\mathcal{Q}[k] = k$ and $\mathcal{R}[k] = k$.

2. If $\pi_k < k$ then by Fact 1.1 the prefix $\mathcal{S}_{[1..k]}$ has a border of length $k - \pi_k$. Since π_k is the shortest period of $\mathcal{S}_{[1..k]}$ the border of length $k - \pi_k$ is the longest non-trivial border of the prefix $\mathcal{S}_{[1..k]}$.

   If the prefix $\mathcal{S}_{[1..k]}$ is quasiperiodic, then by Fact 1.3 the border $\mathcal{S}_{[1..k-\pi_k]}$ must also be covered by the same quasiperiod. The algorithm checks if the quasiperiod of the border $\mathcal{S}_{[1..k-\pi_k]}$ can cover the whole prefix $\mathcal{S}_{[1..k]}$. If a cover of the prefix $\mathcal{S}_{[1..k]}$ by the quasiperiod of $\mathcal{S}_{[1..k-\pi_k]}$ is given, by removing the last occurrence of that quasiperiod we get a cover of a shorter prefix of $\mathcal{S}_{[1..k]}$. This means that the quasiperiod of $\mathcal{S}_{[1..k-\pi_k]}$ covers a prefix of $\mathcal{S}_{[1..k]}$ that is long enough that with one more occurrence of that quasiperiod, the whole prefix $\mathcal{S}_{[1..k]}$ is covered.

   Thus, all the algorithm has to do is to check if the quasiperiod of the border $\mathcal{S}_{[1..k-\pi_k]}$ can cover a prefix of $\mathcal{S}_{[1..k]}$ that is long enough. That is, if $\mathcal{R}[\mathcal{Q}[k-\pi_k]] \geq k - \mathcal{Q}[\pi_k]$, then that quasiperiod covers $\mathcal{S}_{[1..k]}$ and we define $\mathcal{Q}[k] = \mathcal{Q}[k - \pi_k]$ and update $\mathcal{R}[\mathcal{Q}[k]] = k$. Otherwise, the prefix $\mathcal{S}_{[1..k]}$ is superprimitive and we define $\mathcal{Q}[k] = k$ and $\mathcal{R}[k] = k$.

Theorem 2.2 *The algorithm that is described above and in Figure 1 takes $O(n)$ time and uses linear auxiliary space. The number of comparisons of input symbols is at most $2n$.*

Proof: The algorithm performs comparisons only in the calls to the Knuth-Morris-Pratt algorithm. By Theorem 2.1 the calls to that algorithm take $O(n)$ time and at most $2n$ comparisons. The rest of the work takes also $O(n)$ time. By Theorem 2.1 the Knuth-Morris-Pratt algorithm uses linear auxiliary space. The additional space to store the $\mathcal{R}$ and $\mathcal{Q}$ arrays is also linear. □

References

[1] A. Apostolico and A. Ehrenfeucht. Efficient Detection of Quasiperiodicities in Strings. Technical Report 90.5, The Leonadro Fibonacci Institute, Trento, Italy, 1990.

[2] A. Apostolico, M. Farach, and C. S. Iliopoulos. Optimal superprimitivity testing for strings. *Inform. Process. Lett.*, 39:17–20, 1991.

[3] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6:322–350, 1977.

[4] M. Lothaire. *Combinatorics on Words.* Addison-Wesley, Reading, MA. U. S.A., 1983.