Available online at www.sciencedirect.com
ScienceDirect

Information and Computation 205 (2007) 1459–1469

Information and Computation

www.elsevier.com/locate/ic

The number of runs in a string[☆]

Wojciech Rytter[1]

Institute of Informatics, Warsaw University, 02–097 Warsaw, Banacha 2, Poland

Received 14 June 2006; revised 15 January 2007
Available online 8 February 2007

Abstract

A *run* in a string is a nonextendable (with the same minimal period) periodic segment in a string. The set of runs corresponds to the structure of internal periodicities in a string. Periodicities in strings were extensively studied and are important both in theory and practice (combinatorics of words, pattern-matching, computational biology). Let $\rho(n)$ be the maximal number of runs in a string of length n. It has been shown that $\rho(n) = O(n)$, the proof was very complicated and the constant coefficient in $O(n)$ has not been given explicitly. We demystify the proof of the linear upper bound for $\rho(n)$ and propose a new approach to the analysis of runs based on the properties of subperiods: the periods of periodic parts of the runs We show that $\rho(n) \leq 3.44\ n$ and there are at most $O.67n$ runs with periods larger than 87. This supports the conjecture that the number of all runs is smaller than n. We also give a completely new proof of the linear bound and discover several new interesting "periodicity lemmas".

Keywords: Run; String; Periodicity

1. Introduction

We consider finite strings over a finite arbitrary alphabet. The set of all runs in a string corresponds to the structure of its regularities. Initial interest was mostly in repetitions of the type xx (so called *squares*), [1,12]. The number of squares, with *primitive* x, is $\Omega(n \log n)$, hence the number of periodicities of this type is not linear. Then, it has been discovered that the number of runs (also called maximal repetitions or repeats) is linear and consequently linear time algorithms for runs were investigated [10,9]. The result of [10] was one of the deepest results related to combinatorics and algorithmics of strings. However, the most intriguing question remained the asymptotically tight bound for the number of runs. The first bound was quite complicated and has not given any *concrete* constant coefficient in $O(n)$ notation. This subject has been studied in [15,16,2]. A beautiful construction showing the lower bound of approximately 0.927 n has been given in [2].

[☆] The preliminary version of this paper appeared in [14] Symposium on Theoretical Aspects of Computer Science, STACS 2006, with the weaker result.

Email address: rytter@mimuw.edu.pl

[1] The research was supported by the Grant KBN N206 004 32/0806.

0890-5401/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2007.01.007

The exact number of runs has been considered for special strings: *Fibonacci words* and (more generally) *Sturmian words*, [8,7,13]. In this paper, we make a step towards better understanding of the structure of runs. The proof of the linear upper bound is simplified and small *explicit* constant coefficient is given in $O(n)$ notation.

A period p of a word w is any positive integer p such that $w[i] = w[i+p]$ whenever both sides of this equation are defined. Let $per(w)$ denote the size of the smallest period of w. We say that a word w is **periodic** iff $per(w) \leqslant \frac{|w|}{2}$. A word w is said to be *primitive* iff w is not of a form z^k, where z is a finite word and $k \geqslant 2$ is a natural number.

A **run** in a string w is an interval $\alpha = [i \dots j]$ such that $w[i \dots j]$ is a periodic word with the period $p = per(w[i \dots j])$ and this period is not extendable to the left or to the right of $[i \dots j]$. In other words, $[i \dots j]$ is a run iff $|j - i + 1| \geqslant 2p$, $i = 1$ or $w[i-1] \neq w[i-1+p]$ and $j = n$ or $w[j+1] \neq w[j+1-p]$. A run α can be properly included as an interval in another run β, but in this case $per(\alpha) < per(\beta)$.

The value of the run $\alpha = [i \dots j]$ is $val(\alpha) = w[i \dots j]$. When it creates no ambiguity we identify sometimes runs with their values although two different runs could correspond to the identical subwords, if we disregard positions of these runs. Hence runs are also called maximal *positioned* repetitions.

Denote by $RUNS(w)$ the set of runs of w, see Fig. 1 for an example.

Denote: $\rho(n) = \max\{|RUNS(w)| : |w| = n\}$. The most interesting and open conjecture about the runs is: $\rho(n) < n$.

We make a small step towards proving validity of this conjecture and show that $\rho(n) \leqslant 3.44\, n$. The proof of linear upper bound in [10] does not give any explicit constant coefficient at all.

Components of a run.

Each value of the run α is a string $x^k y = w[i \dots j]$, where $|x| = per(\alpha) \geqslant 1$, $k \geqslant 2$ is an integer and y is a proper prefix of x (possibly empty).
The subword x is called the periodic part of the run and denoted by $PerPart(\alpha) = x$. Denote

$$SquarePart(\alpha) = w[i \dots i + 2\, per(\alpha) - 1],\ center(\alpha) = i + |x|$$

The position i is said to be the *occurrence* of this run and is denoted by $first(\alpha)$. We write $\alpha \prec \beta$ iff $first(\alpha) < first(\beta)$.
Define also $dist(\alpha, \beta) = |first(\alpha)) - first(\beta)|$.

Example. In Fig. 2 we have: $first(\alpha) = 2$, $first(\beta) = 4$, $PerPart(\gamma) = (aba)^4 ab$; and $center(\alpha) = 22$, $center(\beta) = center(\gamma) = 21$

In the paper, the crucial role is played by the runs α with highly periodic $PerPart(\alpha)$. Denote **subperiod** $(\alpha) = per(PerPart(\alpha))$.

Example. In Fig. 2 we have:
$subperiod(\alpha) = subperiod(\beta) = subperiod(\gamma) = 3.$

We say that a word w is **highly periodic** (*h-periodic*) if $per(w) \leqslant \frac{|w|}{4}$. A word which is not highly periodic is said to be *weakly periodic*.

Observe that a word can be periodic but at the same time weakly periodic. Also, according to the definition, weakly periodic word can be not periodic.

Algorithmic aspects An efficient algorithm for the computation of all runs was given in [10]. Its basic component is a special decomposition of the string into blocks using a version of Lempel-Ziv compression (see [3,4]) Essentially

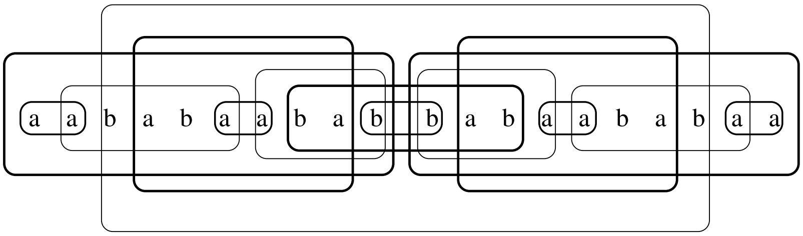

Fig. 1. The structure of $RUNS((aabab)^2(babaa)^2)$.