ARTICLE IN PRESS

European Journal of Combinatorics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

European Journal of Combinatorics

journal homepage: www.elsevier.com/locate/ejc

# Computing primitively-rooted squares and runs in partial words☆

F. Blanchet-Sadri [a], J. Lazarow [b], J. Nikkel [c], J.D. Quigley [d], X. Zhang [e]

[a] *Department of Computer Science, University of North Carolina, P.O. Box 26170, Greensboro, NC 27402–6170, USA*
[b] *Department of Mathematics, University of Texas at Austin, 1 University Station C1200 Austin, TX 78712–0233, USA*
[c] *Department of Mathematics, Vanderbilt University, 1326 Stevenson Center, Nashville, TN 37240, USA*
[d] *Department of Mathematics, University of Illinois — Urbana–Champaign, Altgeld Hall, 1409 W. Green Street, Urbana, IL 61801, USA*
[e] *Department of Mathematics, Princeton University, Fine Hall, Washington Road, Princeton, NJ 08544, USA*

ARTICLE INFO

*Article history:*
Available online xxxx

ABSTRACT

This paper deals with two types of repetitions in strings: *squares*, which consist of two adjacent occurrences of substrings, and *runs*, which are periodic substrings that cannot be extended further to the left or right while maintaining the period. We show how to compute all the primitively-rooted squares in a given partial word, which is a sequence that may have undefined positions, called holes or wildcards, that match any letter of the alphabet over which the sequence is defined. We also describe an algorithm for computing all primitively-rooted runs in a given partial word and extend previous analyses on the number of runs to partial words.


## 1. Introduction

*Repetitions* in strings, or words, have been extensively studied, both from the algorithmic point of view and the combinatorial point of view (see, for example, [10,14]). Applications can be found in many important areas such as computational biology, text compression, natural language processing, to name a few [12,30]. Repetitions are characterized by their periods, lengths, and starting positions.

☆ This is a journal version of Blanchet-Sadri et al. (2015) [10].

*E-mail addresses:* blanchet@uncg.edu (F. Blanchet-Sadri), jlazarow@utexas.edu (J. Lazarow), jordan.k.nikkel@vanderbilt.edu (J. Nikkel), jdquigl2@illinois.edu (J.D. Quigley), xufanz@princeton.edu (X. Zhang).

http://dx.doi.org/10.1016/j.ejc.2017.07.020
0195-6698/

There are many equivalent characterizations of repetitions, and in this paper a repetition in a word $w$ is a triple $(f, l, p)$, where $w[f..l]$ is $p$-periodic and the *exponent* of the repetition, $\frac{l-f+1}{p}$, is at least 2.

*Squares* are the special case of repetitions when $\frac{l-f+1}{p}$ is 2. In other words, a square in a word is a factor $uu$ for some word $u$, called the *root* of the square. It is *primitively-rooted* or *PR* if $u$ is *primitive*, i.e., $u$ is not a power of another word. There can be as many as $\Theta(n \log n)$ occurrences of *primitively-rooted* squares in a word of length $n$, and several $O(n \log n)$ time algorithms have been developed for finding all the repetitions [2,11,31]. A major breakthrough was to compute them in $O(n)$ time; this was achieved in two steps: (1) all repetitions are encoded in *maximal repetitions* or *runs* and (2) there is a linear bound on the number of runs [27,28].

A repetition $(f, l, p)$ is maximal, or is a run, if it is non-extendible, i.e., neither $(f - 1, l, p)$ nor $(f, l + 1, p)$ are repetitions in the word $w$. Note that since every run has exponent at least 2, the square at the beginning of the run uniquely defines the run. This is because that square contains the starting position and period of the run, and the property of the run being maximal gives a unique ending position. A *PR-run* in $w$ is a maximal repetition $(f, l, p)$ with a primitive root of length $p$. If $w = 00011010110101101010$, then $w[2..18] = (01101)^{\frac{17}{5}}$ is a PR-run with period 5, root 01101, and exponent $\frac{17}{5}$ (indexing in the word starts at 0). The maximum number of runs in a string of length $n$ is bounded from above by $cn$, for some constant $c$. A first proof was given by Kolpakov and Kucherov [27,28] who provided an $O(n)$ time algorithm for detecting all runs. However, no constant $c$ can be deduced from their proof. Kolpakov and Kucherov's "runs conjecture" that the maximum number of runs in a string of length $n$ is less than $n$ has generated bounds that have been improved over the years (upper bounds [13,15,22,23,34,35] and lower bounds [19–21,33,36]). Recently, Bannai et al. [4,5] studied runs through combinatorics of Lyndon words and finally settled the "runs" conjecture (another simple proof appears in [16]). The number of runs being less than $n$ has applications to the analysis of any optimal algorithm for computing all repetitions.

*Partial words*, also referred to as *strings with don't-cares* which allow for incomplete or corrupted data [1,18], are sequences that may contain undefined positions, called holes and represented by $\diamond$'s, that are *compatible* with, or *match*, any letter in the alphabet. *Total words* are partial words without holes. Here, a *factor* is a consecutive sequence of symbols in a partial word $w$, while a *subword* is a total word compatible with a factor in $w$. A factor $uv$ is a square if some *completion*, i.e., a filling in of the holes with letters from the alphabet, turns $uv$ into a total word that is a square; equivalently, $u$ and $v$ are compatible.

Repetitions in partial words have also recently been studied, both from the algorithmic point of view and the combinatorial point of view (see, for example, [7–9,17,24,32]). However, no work has been dedicated to computing all occurrences of PR-squares and PR-runs in partial words. The known algorithms for detecting them in total words do not extend easily to partial words, one of the most important culprits being that the compatibility relation is not transitive, as opposed to the equality relation being transitive, e.g., 0 is compatible with $\diamond$ and $\diamond$ is compatible with 1, but 0 is not compatible with 1.

So we adopt an approach, for our algorithms, based on the *large divisors* of the length $n$ of the input partial word, i.e., divisors of $n$, distinct from $n$, whose multiples are either $n$ itself or not divisors of $n$. Every distinct prime divisor $i$ of $n$ gives rise to exactly one large divisor of $n$, namely $\frac{n}{i}$, and hence the number of large divisors of $n$, $\omega(n)$, is the number of distinct prime divisors of $n$, e.g., 30 has large divisors 6, 10, and 15, giving $\omega(30) = 3$. Recently, a formula for the number of primitive partial words was given in terms of the large divisors [7]. In fact, the maximum number of holes in a primitive partial word of length $n$ over a $k$-letter alphabet is $n - \omega(n) - 1$, for all $n, k \geq 2$, and this bound is tight.

The contents of our paper are as follows: In Section 2, we review a few basic concepts on partial words such as periodicity and primitivity. In Section 3, we present efficient algorithms for computing all occurrences of PR-square factors and PR-runs in any partial word over a $k$-letter alphabet. In Section 4, we describe an efficient algorithm for counting the number of occurrences of PR-square subwords. In Section 5, we extend previous analyses on PR-run occurrences in total words to the case of PR-run occurrences in partial words. In particular, we show a linear upper bound on the number of runs for partial words with a constant number of holes. We also show how to recursively count the exact number of microruns, those of small period, for partial words. Finally in Section 6, we conclude with some open problems.

Please cite this article in press as: F. Blanchet-Sadri, et al., Computing primitively-rooted squares and runs in partial words, European Journal of Combinatorics (2017), http://dx.doi.org/10.1016/j.ejc.2017.07.020.