

Computing maximal-exponent factors in an overlap-free word

Golnaz Badkobeh, Maxime Crochemore

► **To cite this version:**

Golnaz Badkobeh, Maxime Crochemore. Computing maximal-exponent factors in an overlap-free word . Journal of Computer and System Sciences, Elsevier, 2016. hal-01806284

HAL Id: hal-01806284

<https://hal-upec-upem.archives-ouvertes.fr/hal-01806284>

Submitted on 1 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Computing maximal-exponent factors in an overlap-free word[☆]

Golnaz Badkobeh^a, Maxime Crochemore^{a,b,*}

^a*King's College London, UK*

^b*Université Paris-Est, France*

Abstract

The exponent of a word is the quotient of its length over its smallest period. The exponent and the period of a word can be computed in time proportional to the word length. We design an algorithm to compute the maximal exponent of all factors of an overlap-free word. Our algorithm runs in linear-time on a fixed-size alphabet, while a naive solution of the question would run in cubic time. The solution for non overlap-free words derives from algorithms to compute all maximal repetitions, also called runs, occurring in the word.

We also show there is a linear number of occurrences of maximal-exponent factors in an overlap-free word. Their maximal number lies between $0.66n$ and $2.25n$ in a word of length n . The algorithm can additionally locate all of them in linear time.

Keywords: Word, string, repetition, power, repeat, periodicity, word exponent, return word, algorithm, automaton.

2000 MSC: 68W40, 68R15, 68Q45

[☆]An extended abstract of a preliminary version of the article was presented at SPIRE'2012 [1].

*Corresponding author

Email addresses: golnaz.badkobeh@gmail.com (Golnaz Badkobeh),
maxime.crochemore@kcl.ac.uk (Maxime Crochemore)

URL: <http://www.inf.kcl.ac.uk/pg/badkobeh/> (Golnaz Badkobeh),
<http://www.inf.kcl.ac.uk/staff/mac/> (Maxime Crochemore)

1. Introduction

We consider the question of computing the maximal exponent of factors (substrings) of a given word (string). The exponent of a word is the quotient of the word length over the word smallest period. For example `alfalfa` has period 3 and exponent $7/3$, and `restore` has period 5 and exponent $7/5$. A word with exponent e is also called an e -power. The exponent indicates better than the period the degree of repetitiveness of factors occurring in a word.

In this article we focus on factors whose exponent is at most 2. Such factors can uniquely be written as uvu where u is the longest border of uvu , that is, the longest proper prefix that is also a suffix of the factor. Note that the exponent is 1 if and only if u is the empty word, while it is 2 if and only if v is the empty word. Consistently with the existing literature a word whose exponent is 1, the minimal possible exponent, admits only the empty word as a border and is called border-free. A word is called a square when its exponent is a positive even integer. In this article, a factor whose exponent is smaller than 2 is called a repeat, while a factor whose exponent is at least 2 is called a repetition or a periodic factor. In other words, in the former case the factor u repeats at two distant positions.

The study of repeats in a word is relevant to long-distance interactions between separated occurrences of the same segment (the u part) in the word. Although occurrences may be far away from each other, they may interact when, for example, the word is folded as it is the case for genomic sequences. A very close problem to considering those repeats is that of computing maximal pairs (positions of the two occurrences of u) with gaps constraints as described by Gusfield [2] and later improved by Brodal et al. [3].

From a combinatorial point of view, the question is related to return words: z is a return word associated with u if u is a prefix of zu and u has no internal occurrence in zu . For instance, if u has only two occurrences in uvu (as a prefix and a suffix) then uv is a return word for u . The word then links two consecutive occurrences of u . The analysis of return words provide characterisations for word morphisms and infinite words. For example, a binary infinite Sturmian word, generalisation of Fibonacci word, is characterised by the fact that every factor (occurring infinitely many times) admits exactly two return words (see [4] and references therein).

The notion of maximal exponent is central to questions related to the avoidability of powers in infinite words. An infinite word is said to avoid

38 e -powers (resp. e^+ -powers) if the exponents of its finite factors are smaller
 39 than e (resp. no more than e). Dejean [5] introduced the repetitive threshold
 40 $\text{RT}(a)$ of an a -letter alphabet: the smallest rational number for which there
 41 exists an infinite word on a letters whose finite factors have exponent at most
 42 $\text{RT}(a)$. In other words, the maximal exponent of factors of such a word is
 43 $\text{RT}(a)$, the minimum possible. The word is also said to be $\text{RT}(a)^+$ -power free.
 44 It is known from Thue [6] that $r(2) = 2$, Dejean [5] proved that $r(3) = 7/4$
 45 and stated the exact values of $\text{RT}(a)$ for every alphabet size $a > 3$. Dejean's
 46 conjecture was eventually proved in 2009 after partial proofs given by several
 47 authors (see [7, 8] and references therein).

48 The exponent of a word can be calculated in linear time using basic string
 49 matching that computes the smallest period associated with the longest border
 50 of the word (see [9]). A straightforward consequence provides a $O(n^3)$ -
 51 time solution to compute the maximal exponent of all factors of a word of
 52 length n since there are potentially of the order of n^2 factors. However, a
 53 quadratic time solution is also a simple application of basic string matching.
 54 By contrast, our solution runs in linear time on a fixed-size alphabet.

55 When a word contains runs, that is, maximal periodicities of exponent at
 56 least 2, computing their maximal exponent can be achieved in linear time by
 57 adapting the algorithm of Kolpakov and Kucherov [10] that computes all the
 58 runs occurring in the word. Their result relies on the fact that there exists a
 59 linear number of runs in a word [10] (see [11, 12] for precise bounds). Nev-
 60 ertheless, this does not apply to square-free words, which we are considering
 61 here.

62 Our solution works indeed on overlap-free words, not only on square-free
 63 words, that is, on words whose maximal exponent of factors is at most 2.
 64 Thus, we are looking for factors w of the form uvu , where u is the longest
 65 border of w . In order to accomplish this goal, we exploit two main tools: the
 66 Suffix Automaton of some factors and a specific factorisation of the whole
 67 word.

68 The Suffix Automaton (see [9]) is used to search for maximal-exponent
 69 factors in a product of two words due to its ability to locate occurrences of
 70 all factors of a pattern. Here, we enhance the automaton to report the right-
 71 most occurrences of those factors. Exploiting only the Suffix Automaton in a
 72 balanced divide-and-conquer manner produces a $O(n \log n)$ -time algorithm.

73 In order to eliminate the log factor we additionally exploit a word factori-
 74 sation, namely the f-factorisation (see [9]), a type of LZ77 factorisation (see
 75 [13]) fit for word algorithms. It has now become common to use this factorisa-

76 tion to derive efficient or even optimal algorithms. The f-factorisation allows
77 one to skip larger and larger parts of the words during an online processing.
78 For our purpose, it is composed of factors occurring before their current po-
79 sition with no overlap. The factorisation can be computed in $O(n \log a)$ -time
80 (where a is the alphabet size) using a Suffix Tree or a Suffix Automaton, and
81 in linear time on an integer alphabet using a Suffix Array [14].

82 The running time of the proposed algorithm depends additionally on the
83 repetitive threshold of the underlying alphabet size of the word. The thresh-
84 old restricts the context of the search for a second occurrence of u associated
85 with a factor uvu .

86 We show a very surprising property of factors whose exponent is max-
87 imal in an overlap-free word: there are no more than a linear number of
88 occurrences of them, although the number of occurrences of maximal (i.e.
89 non-extensible) factors can be quadratic.

90 We show a lower bound of $0.66n$ and an upper bound of $2.25n$ on their
91 maximal number for a word of length n . They improve on the bounds given
92 in a preliminary version [1] of the article. The lower bound is based on a
93 result of Pansiot [15] on the repetitive threshold of four-letter alphabets.

94 As a consequence, the algorithm can be modified to output all occurrences
95 of maximal-exponent factors of an overlap-free word in linear time.

96 The question would have a simple solution by computing MinGap on
97 each internal node of the Suffix Tree of the input word, as is discussed in the
98 conclusion. MinGap of a node is the smallest difference between the positions
99 assigned to leaves of the subtree rooted at the node. Unfortunately, the best
100 algorithms for MinGap computation, equivalent to MaxGap computation,
101 run in time $O(n \log n)$ (see [16, 17, 18] and the discussion in [19]).

102 A remaining question to the present study is to unify the algorithmic
103 approaches for locating runs in non overlap-free words and maximal-exponent
104 factors in overlap-free words.

105 The plan of the article is as follows. After defining the problem in the
106 next section we present the general scheme of the algorithm that relies on
107 the f-factorisation of the input word in Section 3. The sub-function operat-
108 ing a Suffix Automaton is described in Section 4 and the complexity of the
109 complete algorithm is studied in Section 5. In Section 6 we prove lower and
110 upper bounds on the number of occurrences of maximal-exponent factors. A
111 conclusion follows.

112 2. Maximal-exponent factors

113 We consider words (strings) on a finite alphabet A of size a . If x is a
114 word of length $|x| = m$, $x[i]$ denotes its letter at position i , $0 \leq i < m$. A
115 factor of x is of the form $x[i]x[i+1] \dots x[j]$ for two positions i and j and is
116 denoted by $x[i..j]$ (it is the empty word if $j < i$). It is a prefix of x if $i = 0$
117 and a suffix of x if $j = m - 1$.

118 The word x has period p , $0 < p \leq m$, if $x[i] = x[i+p]$ whenever both
119 sides of the equality are defined, i.e. for $i = 0, \dots, m-p-1$. The period of x ,
120 $\text{period}(x)$, is its smallest period and its exponent is $\text{exp}(x) = m/\text{period}(x)$.
121 For example, $\text{exp}(\text{restore}) = 7/5$, $\text{exp}(\text{mama}) = 2$ and $\text{exp}(\text{alfalfa}) = 7/3$.
122 An overlap-free word contains no factor of exponent larger than 2, that is,
123 no factor of the form $bwbwb$ for a letter b and a word w .

124 We consider a fixed overlap-free word y of length n and deal with its
125 factors having the maximal exponent among all factor exponents. They are
126 called **maximal-exponent factor** or MEF for short. They have exponent
127 at most 2 since y is overlap-free.

128 A MEF w in y is of the form uvu , where u is its longest border (longest
129 factor that is both a prefix and a suffix of w). Then $\text{period}(w) = |uv|$ and
130 $\text{exp}(w) = |uvu|/|uv| = 1 + |u|/\text{period}(w)$. By convention, in the following we
131 allow a border-free factor to be considered as a MEF of exponent 1, though
132 it contains no repeat in the common sense since the repeating element u is
133 empty and it can appear only if no letter in y appears more than once, i.e.
134 if its length is no more than the alphabet size.

135 First note that a MEF uvu contains only two occurrences of u since this
136 would produce a factor with a larger exponent. Second, any occurrence of
137 the MEF uvu is maximal in the sense that it cannot be extended with the
138 same period. That is, the two occurrences of u are followed by two distinct
139 letters and preceded by two distinct letters. These remarks are stated in
140 Lemmas 3 and 2 respectively.

141 The maximality of occurrences of repetitions in non overlap-free words
142 implies their linear number but unfortunately this property does not hold for
143 MEF occurrences.

144 3. Computing the maximal exponent of factors

145 The core result of the article is an algorithm, `MAXEXPFAC`, that com-
146 puts the maximal exponent of factors of the overlap-free word y . The algo-
147 rithm has to look for factors of the form uvu , for two words u and v , u being

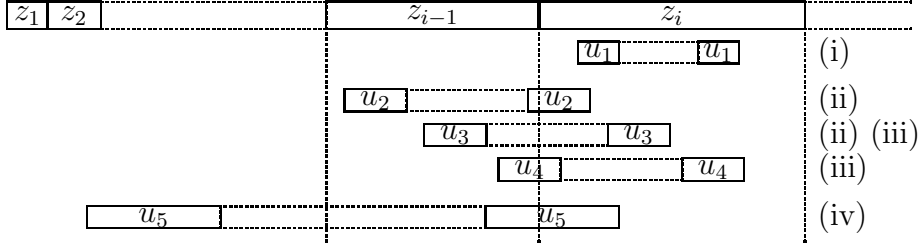


Figure 1: The only four possible locations of a factor uvu involving phrase z_i of the factorisation of the word: (i) internal to z_i ; (ii) the first occurrence of u is internal to z_{i-1} ; (iii) the second occurrence of u is internal to z_i ; (iv) the second occurrence of u is internal to $z_{i-1}z_i$.

148 the longest border of uvu . The aim of this algorithm is accomplished with
 149 the help of Algorithm MAXEXP, designed in the next section, which detects
 150 those factors occurring within the concatenation of two words.

151 Algorithm MAXEXPFAC relies on the f-factorisation of y , a type of LZ77
 152 factorisation [13] defined as follows. It is a sequence of non-empty words,
 153 z_1, z_2, \dots, z_k , called phrases and satisfying $y = z_1z_2 \cdots z_k$ where z_i is the
 154 longest prefix of $z_i z_{i+1} \cdots z_k$ occurring in $z_1z_2 \cdots z_{i-1}$. When this longest
 155 prefix is empty, z_i is the first letter of $z_i z_{i+1} \cdots z_k$, thus it is a letter that does
 156 not occur previously in y . This definition is equivalent to the definition in
 157 [9], in which a phrase z_i can overlap with its previous occurrence, because
 158 the word y is overlap-free. We adapt the factorisation to the purpose of our
 159 problem by defining z_1 as the longest prefix of y in which no letter occurs
 160 more than once. Then, $|z_1| \leq a$ and $\text{MAXEXPFAC}(z_1) = 1$. Note that
 161 $\text{MAXEXPFAC}(z_1z_2) > 1$ if $z_1 \neq y$.

162 When the factorisation of y is computed, Algorithm MAXEXPFAC processes
 163 the phrases sequentially, from z_2 to z_k . After z_1, z_2, \dots, z_{i-1} have
 164 been processed, the variable e stores the maximal exponent of factors of
 165 $z_1z_2 \cdots z_{i-1}$. Then, the next factors to be considered are those involving
 166 phrase z_i . Such factors uvu can either be internal to z_i or involve other
 167 phrases. However, the crucial property of the factorisation is that the second
 168 occurrence of u is only to be searched for in $z_{i-1}z_i$ because it cannot contain
 169 a phrase as this would contradict the definition of the factorisation.

170 We further distinguish four possible cases according to the position of the
 171 factor uvu as follows (see Figure 1):

- 172 (i) The two occurrences of u are contained in z_i .
- 173 (ii) The first occurrence of u is contained in z_{i-1} and the second ends in z_i .

- 174 (iii) The first occurrence of u starts in z_{i-1} and the second occurrence is
 175 contained in z_i .
 176 (iv) The first occurrence of u starts in $z_1 \cdots z_{i-2}$ and the second occurrence
 177 is contained in $z_{i-1}z_i$.

178 Case (i) needs no action and other cases are dealt with calls to Algorithm
 179 MAXEXP as described in the code below where \tilde{x} denotes the reverse of
 180 word x . For any two words z and w and a positive rational number e ,
 181 MAXEXP(z, w, e) is the maximal exponent of factors in zw whose occurrences
 182 start in z and end in w , and whose exponent is at least e ; it is e itself if there
 183 is no such factor.

MAXEXPFAC(y)

```

1      ( $z_1, z_2, \dots, z_k$ )  $\leftarrow$  f-factorisation of  $y$ 
2       $\triangleright z_1$  is the longest prefix of  $y$  in which no letter repeats
3       $e \leftarrow 1$ 
4      for  $i \leftarrow 2$  to  $k$  do
184    5           $e \leftarrow$  MAXEXP( $z_{i-1}, z_i, e$ )
        6           $e \leftarrow$  MAXEXP( $\tilde{z}_i, \tilde{z}_{i-1}, e$ )
        7          if  $i > 2$  then
        8               $e \leftarrow$  MAXEXP( $z_{i-1}\tilde{z}_i, z_1 \cdots \tilde{z}_{i-2}, e$ )
        9      return  $e$ 

```

185 Note that variable e can be initialised to the repetitive threshold $\text{RT}(a)$
 186 (see Introduction) when the alphabet of word y is of size a and if the word is
 187 long enough. The maximal length of words containing no factor of exponent
 188 at least $\text{RT}(a)$ is 3 for $a = 2$, 38 for $a = 3$, 121 for $a = 4$, and $a + 1$ for $a \geq 5$
 189 (see [5]).

190 Another technical remark is that the instruction at line 6 can be tuned to
 191 deal only with type (iii) factors of the form u_4vu_4 (see Figure 1), i.e. factors
 192 for which the first occurrence of the border starts in z_{i-1} and ends in z_i ,
 193 because line 5 finds those of the form u_3vu_3 . However, this has no influence
 194 on the asymptotic running time.

195 **Theorem 1.** *For any overlap-free word input, MAXEXPFAC computes the*
 196 *maximal exponent of factors occurring in the word.*

197 **Proof.** We consider a run of MAXEXPFAC(y). Let e_1, e_2, \dots, e_k be the
 198 successive values of the variable e , where e_i is the value of e just after the

199 execution of lines 5–8 for index i . The initial value $e_1 = 1$ is the maximal
 200 exponent of factors in z_1 as a consequence of its definition. We show that e_i
 201 is the maximal exponent of factors occurring in $z_1z_2 \cdots z_i$ if e_{i-1} is that of
 202 $z_1z_2 \cdots z_{i-1}$, for $2 \leq i \leq k$.

203 To do so, since e_i is at least e_{i-1} (use of max at lines 5–8), all factors
 204 occurring in $z_1z_2 \cdots z_{i-1}$ are taken into account and we only have to consider
 205 factors coming from the concatenation of $z_1z_2 \cdots z_{i-1}$ with z_i , that is, factors
 206 of the form uvu where the second occurrence of u ends in z_i . As discussed
 207 above and illustrated in Figure 1, only four cases are to be considered because
 208 the second occurrence of u cannot start in $z_1z_2 \cdots z_{i-2}$ without contradicting
 209 the definition of z_{i-1} .

210 Line 5 deals with Case (ii) by the definition of MAXEXP. Similarly, line
 211 6 is for Case (iii), and line 8 for Case (iv).

212 If a factor occurs entirely in z_i , Case (i), by the definition of z_i it occurs
 213 also in $z_1z_2 \cdots z_{i-1}$, which is reported by e_{i-1} .

214 Therefore, all relevant factors are considered in the computation of e_i ,
 215 which is then the maximal exponent of factors occurring in $z_1z_2 \cdots z_i$. This
 216 implies that the exponent e_k returned by the algorithm is the exponent of
 217 $z_1z_2 \cdots z_k = y$ as stated. ■

218 4. Locating repeats in a product

219 In this section, we describe Algorithm MAXEXP applied to (z, w, e) for
 220 computing the maximal exponent of factors in zw that end in w , whose left
 221 border occurs in z , and whose exponent is at least e . MAXEXP is called in
 222 the main algorithm of the previous section.

223 To locate factors under consideration, the algorithm examines positions
 224 j on w and for each computes the longest potential border of a factor, a
 225 longest suffix u of $zw[0..j]$ occurring in z . The algorithm is built upon an
 226 algorithm that finds all of them using the Suffix Automaton of word z .

227 The Suffix Automaton of z , denoted $\mathcal{S}(z)$, is used to locate borders of
 228 factors. It is the minimal deterministic finite automaton whose language
 229 is the set of suffixes of z (see [9, Section 6.6] for more description and for
 230 efficient construction). An example is given in Figure 2. The data structure
 231 has an initial state denoted $\text{initial}(\mathcal{S})$ and a state called $\text{last}(\mathcal{S})$ that is the
 232 accepting state of z itself (it is the only state with no outgoing arcs). In
 233 addition to the transition function goto (represented by arcs in the figure) it

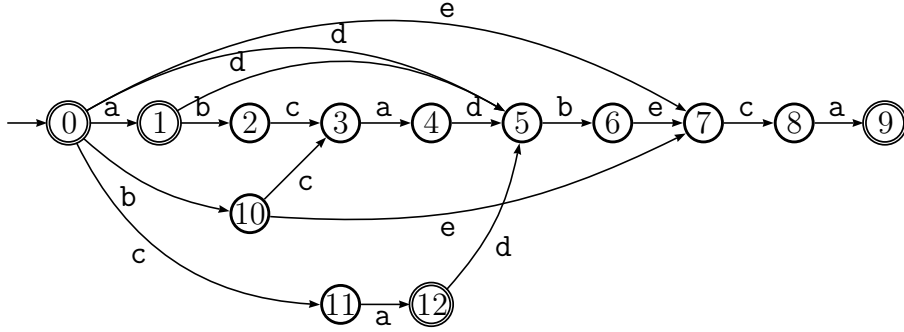


Figure 2: Suffix Automaton of **abcadbca**. Suffix links: $F[1] = 0, F[2] = 10, F[3] = 11, F[4] = 1, F[5] = 0, F[6] = 10, F[7] = 0, F[8] = 11, F[9] = 12, F[10] = 0, F[11] = 0, F[12] = 1$. Maximal incoming word lengths: $L[0] = 0, L[1] = 1, L[2] = 2, L[3] = 3, L[4] = 4, L[5] = 5, L[6] = 6, L[7] = 7, L[8] = 8, L[9] = 9, L[10] = 1, L[11] = 1, L[12] = 2$. Minimal extension lengths: $sc[0] = 0, sc[1] = 0, sc[2] = 7, sc[3] = 6, sc[4] = 5, sc[5] = 4, sc[6] = 3, sc[7] = 2, sc[8] = 1, sc[9] = 0, sc[10] = 3, sc[11] = 1, sc[12] = 0$.

234 contains the failure link F_z and the length function L_z , both defined on the
 235 set of states. The link is defined as follows: let $p = \text{goto}(\text{initial}(\mathcal{S}(z)), x)$ for
 236 $x \in A^+$; then $F_z(p) = \text{goto}(\text{initial}(\mathcal{S}(z)), x')$, where x' is the longest suffix of
 237 x for which this latter state is not p . As for the length function, $L_z(p)$ is the
 238 maximal length of words x for which $p = \text{goto}(\text{initial}(\mathcal{S}(z)), x)$.

239 The next two lemmas show that, after u is located with the Suffix Au-
 240 tomaton, although some of its suffixes may have an exponent higher than e ,
 241 we can discard many of them.

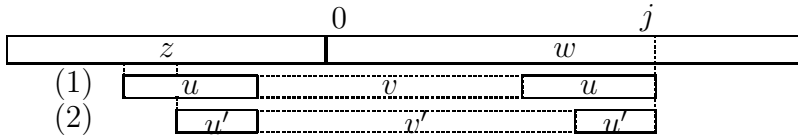


Figure 3: When u and its suffix u' end at the same right-most position on z , factor (1) has a larger exponent than factor (2).

242 Figure 3 illustrates the proof of the following lemma.

243 **Lemma 2.** *Let u' be a suffix of u . If they are both associated with the same*
 244 *state of $\mathcal{S}(z)$ the maximal exponent of a $u'v'u'$ is not greater than the maximal*
 245 *exponent of its associated uvu factor.*

246 **Proof.** The hypothesis implies that the right-most occurrence of u' ends
 247 at the same positions on z as u (see Figure 3). Then, $u'v'u'$ and uvu have

248 the same period $|vu| = |v'u'|$ but since $u'v'u'$ is not longer than uvu , the
 249 exponent of $u'v'u'$ is not greater than that of uvu . ■

250 Note that a suffix u' of u may have an internal occurrence in uvu , which
 251 would lead to a factor having a larger exponent. For example, let $z = \text{abadba}$
 252 and $w = \text{cdaba}$. The factor abadbacdaba with border aba has exponent $11/8$
 253 while the suffix ba of aba infers the factor bacdaba of greater exponent $7/5$.

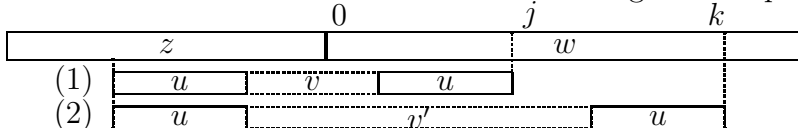


Figure 4: Factor (1) ending at position j has a larger exponent than factor (2) ending at position $k > j$.

254 The proof of the following lemma can be deduced from the remark in
 255 Figure 4.

256 **Lemma 3.** *If u occurs at end positions j and k on w with $k > j$, the factor*
 257 *$wv'u$ ending at k cannot be a maximal-exponent factor.*

258 **Proof.** To have a maximal exponent the first occurrence of u in $wv'u$ should
 259 end at the right-most position on z . But then there is a factor sharing
 260 the same first occurrence of u and with a closer second occurrence of u
 261 (see Figure 4). Therefore $1 + |u|/|uv| > 1 + |u|/|wv'|$, which completes the
 262 statement proof. ■

263 The properties stated in the previous lemmas are used by Algorithm
 264 MAXEXP to avoid some exponent calculations as follows. Let uvu be a
 265 factor ending at j on $zw[0..j]$ and for which u is the longest word associated
 266 with state $q = \text{goto}(\text{initial}(\mathcal{S}), u)$, where goto is the transition function of
 267 the automaton. Then next occurrences of u and of any of its suffixes cannot
 268 produce factors with an exponent larger than that of uvu . State q is then
 269 marked to inform the next steps of the algorithm that it has been visited.

270 We need another function, sc_z , defined on states of $\mathcal{S}(z)$ as follows: $sc_z(p)$
 271 is the minimal length of paths from p to a terminal state; in other terms, if
 272 $p = \text{goto}(\text{initial}(\mathcal{S}(z)), x)$, then $sc_z(p) = |x'|$ where x' is the shortest word for
 273 which xx' is a suffix of z . With this precomputed extra element, computing
 274 an exponent is a mere division (see Figure 5).

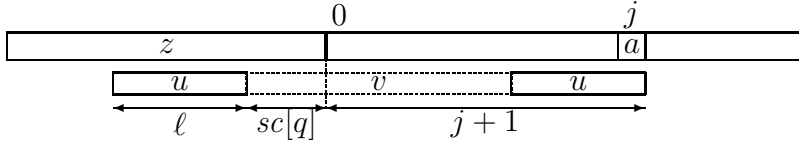


Figure 5: The maximal exponent of all factors in question bordered by u , longest factor of z ending at j , is $(\ell + sc[q] + j + 1)/(sc[q] + j + 1)$.

j	0	1	2	3	4	5	6	7	8	9
$w[j]$	d	e	c	a	d	b	e	c	a	d
q	12	5	7	8	9	5	6	7	8	9
ℓ	2	3	1	2	3	3	4	5	6	7
exp	8/5	5/4	3/2	7/4	4/3	13/9	14/9	5/3	16/9	17/14
			5/4			10/9				

Figure 6: Computing exponents when searching zw for factors uvu . The first occurrence of u is in z and the second ends in zw . The Suffix Automaton of $z = \text{abcdbeca}$ with function sc is in Figure 2. The search is done by parsing $w = \text{decadbecad}$ with the automaton. Exponents of factors are given by the expression $(\ell + sc[q] + j + 1)/(sc[q] + j + 1)$. The last line is for exponents corresponding to suffixes of u . The maximal exponent of all factors is $7/4$.

MAXEXP(z, w, e)

```

1   $\mathcal{S} \leftarrow$  Suffix Automaton of  $z$ 
2  mark initial( $\mathcal{S}$ )
3   $(q, \ell) \leftarrow (F[\text{last}(\mathcal{S})], L[F[\text{last}(\mathcal{S})]])$ 
4  for  $j \leftarrow 0$  to  $\min\{\lfloor |z|/(e-1) - 1 \rfloor, |w| - 1\}$  do
5      while goto( $q, w[j]$ ) = NIL and  $q \neq \text{initial}(\mathcal{S})$  do
6           $(q, \ell) \leftarrow (F[q], L[F[q]])$ 
7      if goto( $q, w[j]$ )  $\neq$  NIL then
8           $(q, \ell) \leftarrow (\text{goto}(q, w[j]), \ell + 1)$ 
9           $(q', \ell') \leftarrow (q, \ell)$ 
10         while  $q'$  unmarked do
11              $e \leftarrow \max\{e, (\ell' + sc[q'] + j + 1)/(sc[q'] + j + 1)\}$ 
12             if  $\ell' = L[q']$  then
13                 mark  $q'$ 
14              $(q', \ell') \leftarrow (F[q'], L[F[q']])$ 
15  return  $e$ 

```

Figure 6 illustrates a computation done by the algorithm using the Suffix

277 Automaton of Figure 2.

278 Note that the potential overflow when computing $\lfloor |z|/(e-1) - 1 \rfloor$ can
279 easily be fixed in the algorithm implementation.

280 **Theorem 4.** *Algorithm MAXEXP, applied to words z and w and to the*
281 *rational number e , produces the maximal exponent of factors in zw that end*
282 *in w , whose left border occurs in z , and whose exponent is at least e .*

283 **Proof.** In the algorithm, position j on w stands for a potential ending
284 position of a relevant factor. First, we show that the algorithm does not
285 require to examine more values of j than those specified at line 4. The
286 exponent of a factor uvu is $|uvu|/|vu|$. Since we are looking for factors
287 satisfying $|uvu|/|vu| \geq e$, the longest possible such factor has period $j+1$
288 and border z . Then $(|z|+j+1)/(j+1) > e$ implies $j < |z|/(e-1) - 1$ (which
289 is conventionally set to $+\infty$ if $e = 1$). Since j is a position on w , $j < |w|$, which
290 completes the first statement.

291 Second, given a position j on w , we show that the algorithm examines all
292 the possible concerned factors having an exponent at least e and ending at j .
293 The following property related to variables q , state of \mathcal{S} , and ℓ is known from
294 [9, Section 6.6]: let u be the longest suffix of $zw[0..j]$ that is a factor of z ,
295 then $q = \text{goto}(\text{initial}(\mathcal{S}), u)$ and $\ell = |u|$. The property is also true just after
296 execution of line 3 for z alone due to the initialisation of the two variables.

297 Then, word u is the border of a factor ending in w and whose left border
298 occurs in z . Lines 9 to 14 check the exponents associated with u and its
299 suffixes. If q' is unmarked, the exponent is computed as explained before (see
300 Figure 5). If the condition at line 11 is met, which means that u is the longest
301 word satisfying $q' = \text{goto}(\text{initial}(\mathcal{S}), u)$, due to Lemma 3 the algorithm does
302 not need to check the exponent associated with later occurrences of u , nor
303 with the suffixes of u since they have been checked before. Due to Lemma
304 2, suffixes of u ending at the same right-most position on z do not have a
305 larger exponent. Therefore the next suffix whose associated exponent has to
306 be checked is the longest suffix leading to a different state of \mathcal{S} : it is $F(q')$
307 and the length of the suffix is $L(F(q'))$ by definition of F and L .

308 Finally note the initial state of \mathcal{S} is marked because it corresponds to an
309 empty word u , that is a factor of exponent 1, which is not larger than the
310 values of e .

311 This proves the algorithm runs through all possible relevant factors and
312 completes the proof. ■

313 **5. Complexity analysis**

314 In this section we analyse the running time and memory usage of our
 315 algorithms.

316 **Proposition 5.** *Applied to words z and w and to the rational number e ,
 317 Algorithm MAXEXP requires $O(|z|)$ space in addition to inputs and runs in
 318 total time $O(|z| + \min\{\lfloor |z|/(e-1) - 1 \rfloor, |w| - 1\})$ on a fixed size alphabet. It
 319 performs less than $2|z| + \min\{\lfloor |z|/(e-1) - 1 \rfloor, |w| - 1\}$ exponent computations.*

320 **Proof.** The space is used mostly for storing the automaton, which is known
 321 to have no more than $2|z|$ states and $3|z|$ edges (see [9]). It can be stored
 322 in linear space if edges are implemented by successor lists, which adds a
 323 multiplicative $\log a$ factor on transition time.

324 It is known from [9, Section 6.6] that the algorithm runs in linear time
 325 on a fixed alphabet, including the automaton construction with elements F ,
 326 L and sc , if we exclude the time for executing lines 9 to 14.

327 It remains to enumerate the number of times line 11 is executed. It is
 328 done once for each position j associated with an unmarked state. If it is done
 329 more than once for a given position, then the second value of q' comes from
 330 the failure link. A crucial observation is that condition at line 12 holds for
 331 such a state. Therefore, since $\mathcal{S}(z)$ has no more than $2|z|$ states, the total
 332 number of extra executions of line 11 is at most $2|z|$, which gives the stated
 333 result. ■

334 The proof of the linear running time of Algorithm MAXEXPFAC addition-
 335 ally relies on a combinatorial property of words. It is the notion of repetitive
 336 threshold $\text{RT}(a)$ for an alphabet of size a mentioned in Introduction.

337 **Theorem 6.** *Applied to any overlap-free word of length n on a fixed-size
 338 alphabet, Algorithm MAXEXPFAC runs in time $O(n)$ and requires $O(n)$ extra
 339 space.*

340 **Proof.** Computing the f-factorisation (z_1, z_2, \dots, z_k) of the input takes time
 341 and space $O(n)$ on a fixed-size alphabet using any suffix data structure. (It
 342 can even be done in time $O(n)$ on an integer alphabet, see [14].)

343 The next instructions execute in linear extra space from Proposition 5.
 344 Line 5 takes time $O(|z_{i-1}| + \min\{\lfloor |z_{i-1}|/(e-1) - 1 \rfloor, |z_i| - 1\})$, which is
 345 bounded by $O(|z_{i-1}| + |z_{i-1}|/(e-1) - 1)$, for $i = 2, \dots, k$. For a long enough

346 input, e is eventually at least $\text{RT}(a)$ where a is the input alphabet. The time
 347 is then bounded by $O(|z_{i-1}| + |z_{i-1}|/(\text{RT}(a) - 1) - 1)$, thus $O(|z_{i-1}|)$ because
 348 $\text{RT}(a)$ is a constant. The contribution of Line 5 to the total runtime is then
 349 $O(\sum_{i=2}^k |z_{i-1}|)$.

350 Similarly it is $O(\sum_{i=2}^k |z_i|)$ for Line 6 and $O(\sum_{i=2}^k |z_{i-1}z_i|)$ for Line 8. Thus
 351 the overall runtime is bounded by $O(\sum_{i=1}^k |z_i|)$, which is $O(n)$ as expected. ■

352 6. Counting maximal-exponent factors

353 This section is devoted to the combinatorial aspects of maximal-exponent
 354 factors (MEF). We exhibit upper and lower bounds on their maximal number
 355 of occurrences in an overlap-free word.

356 The upper bound shows there is no more than a linear number of MEF
 357 occurrences in a word according to its length. In addition, the lower bound
 358 proves that this is optimal up to a multiplicative factor that remains to be
 359 discovered.

360 Note that on the alphabet $\{\mathbf{a}, \mathbf{a}_1, \dots, \mathbf{a}_n\}$ the word $\mathbf{aa}_1\mathbf{aa}_2\mathbf{a}\dots\mathbf{aa}_n\mathbf{a}$ of
 361 length $2n + 1$ has a quadratic number of maximal factors. Indeed all occur-
 362 rences of factors of the form \mathbf{awa} for a non-empty word w are non extensible.
 363 But only the n factors of the form \mathbf{aca} for a letter c have the maximal expo-
 364 nent $3/2$.

365 6.1. Upper bound

366 Before giving an upper bound, we start with a simple property of MEFs,
 367 which does not lead to their linear number, but is used later to tune the
 368 upper bound.

369 **Lemma 7.** *Consider two occurrences of MEFs with the same border length*
 370 *b starting at respective i and j on the word y , $i < j$. Then, $j - i > b$.*

371 **Proof.** The two MEFs having the same border length, since they have the
 372 same exponent, they have also the same period and the same length. Let b
 373 be their border length and p their period.

374 Assume *ab absurdo* $j - i \leq b$. The word $y[i..i+b-1] = y[i+p..i+p+b-1]$
 375 is the border of the first MEF. The assumption implies that $y[i+b] = y[i+p+$
 376 $b]$ because these letters belong to the border of the second MEF. It means
 377 the first MEF can be extended with the same period, producing a larger
 378 exponent, a contradiction. Therefore, $j - i > b$ as stated. ■

379 If we count the occurrences of MEFs by their border lengths after Lemma 7
 380 we get an initial part of the harmonic series, a quantity that is not linear
 381 with respect to the length of y .

382 To get a linear upper bound on the number of occurrences of MEFs we
 383 introduce the notion of δ -MEFs, for a positive real number δ , as follows. A
 384 MEF uvu is a δ -MEF if its border length $b = |u| = |uvu| - \text{period}(uvu)$
 385 satisfies $2\delta < b \leq 4\delta$. Then any MEF is a δ -MEF for some $\delta \in \Delta$, where
 386 $\Delta = \{1/4, 1/2, 1, 2, 2^2, 2^3, \dots\}$. This is the technique used for example in
 387 [11, 12] to count runs in words.

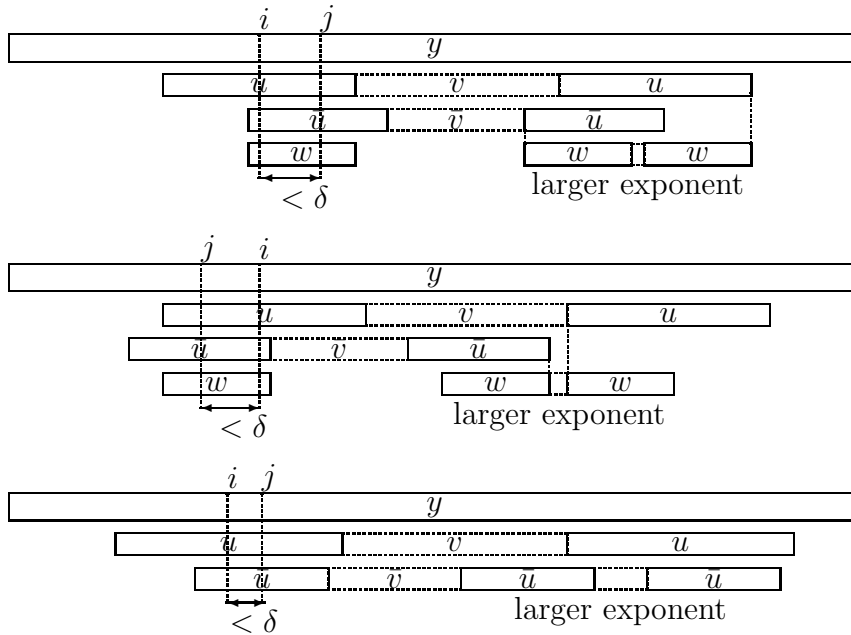


Figure 7: Two δ -MEFs, uvu and $\bar{u}\bar{v}\bar{u}$, having mid-positions of their left borders at close positions induce a factor with a larger exponent, a contradiction.

388 The proof of the next lemma is illustrated by Figure 7. We define the
 389 mid-position of an occurrence of a factor x whose first letter is at position i
 390 on y by $i + \lfloor |x|/2 \rfloor - 1$.

391 **Lemma 8.** *Let uvu and $\bar{u}\bar{v}\bar{u}$ be two occurrences of δ -MEFs in y whose left*
 392 *borders mid-positions are at respective positions i and j on y . Then, $|j - i| \geq$*
 393 *δ .*

394 **Proof.** We consider w.l.o.g. $|u| \geq |\bar{u}|$. Assume ab absurdo $|j - i| < \delta$ (see
 395 Figure 7).

396 Since both $|u| > 2\delta$ and $|\bar{u}| > 2\delta$, the two occurrences of left borders
 397 overlap. Let w be the overlap. It can be a suffix of u and a prefix of \bar{u} , or it
 398 can be a suffix of \bar{u} and a prefix of u , or w can be \bar{u} itself, the shorter of two
 399 borders, when it occurs inside u . The three cases are displayed in this order
 400 on Figure 7.

401 Let $p = |uv|$ be the period of uvu and $p' = |\bar{u}\bar{v}|$ be that of $\bar{u}\bar{v}\bar{u}$. The
 402 exponent of the two factors is $e = 1 + |u|/p = 1 + |\bar{u}|/p'$, which implies
 403 $p - p' = (|u| - |\bar{u}|)/(e - 1)$.

Note that w , the overlap of the two left borders, occurs at least at two
 other positions. For example, in the first case, it occurs as a suffix of the right
 border of u and as a prefix of the right border of \bar{u} . Due to the periodicity
 of the two factors, uvu and $\bar{u}\bar{v}\bar{u}$, the last two occurrences of w are $p - p'$
 positions apart. Therefore the factor z starting with one occurrence and
 ending with the other has exponent at least (it can be larger if w is not the
 longest border of z):

$$1 + \frac{|w|}{p - p'} = 1 + \frac{|w|(e - 1)}{(|u| - |\bar{u}|)}.$$

404 Now, from inequalities $2\delta < |\bar{u}| \leq |u| \leq 4\delta$ and the definition of w , we
 405 have both $|w| > |u|/2$ and $|u| - |\bar{u}| < |u|/2$. Then $|w| > |u| - |\bar{u}|$ and since
 406 $e - 1 > 0$ the exponent of z is then larger than e , a contradiction. Therefore
 407 $|j - i| \geq \delta$ as stated. ■

408 A direct consequence of the previous lemma is the linear number of MEF
 409 occurrences. Because Lemma 8 implies that the number of δ -MEF occur-
 410 rences in y is no more than n/δ . And since values of δ in Δ cover all border
 411 lengths, the total number of occurrences of MEFs is bounded by

$$\sum_{\delta \in \Delta} \frac{n}{\delta} = n \left(4 + 2 + 1 + \frac{1}{2} + \left(\frac{1}{2}\right)^2 + \dots \right) < 8n.$$

412 The next statement refines the above upper bound by combining results
 413 of Lemmas 7 and 8.

414 **Theorem 9.** *There are less than $2.25n$ occurrences of maximal-exponent*
 415 *factors in a word of length n .*

416 **Proof.** According to Lemma 7 there are less than

$$\sum_{b=1}^{b=5} \frac{n}{b+1} = 1.45n$$

417 occurrences of MEFs with border length at most 5.

418 We then apply Lemma 8 with values of $\delta \in \Gamma$ that cover all remaining
 419 border lengths of MEFs: $\Gamma = \{(5/2), 5, 10, 20, \dots\}$. It gives the upper bound

$$\sum_{\delta \in \Gamma} \frac{n}{\delta} = \frac{1}{5} \left(2 + 1 + \frac{1}{2} + \left(\frac{1}{2}\right)^2 + \dots \right) n = \frac{4}{5}n$$

420 for the number of occurrences of MEFs with border length at least 6. There-
 421 fore the global upper bound we obtain is $2.25n$. ■

422 Note that the border length 5 minimises the expression

$$\left(\sum_{b=1}^{b=k} \frac{n}{b+1} \right) + \frac{1}{k} \left(2 + 1 + \frac{1}{2} + \left(\frac{1}{2}\right)^2 + \dots \right) n = \left(\sum_{b=1}^{b=k} \frac{n}{b+1} \right) + \frac{4n}{k}$$

423 with respect to k , which means the technique is unlikely to produce a smaller
 424 bound. By contrast, experiments show that the number of occurrences of
 425 MEFs is smaller than n and not even close to n , at least for small values of
 426 n . The following table displays the maximal number of MEFs for overlap-
 427 free word lengths $n = 5, 6, \dots, 20$ and for alphabet sizes 2, 3 and 4. It also
 428 displays (second element of pairs) the associated maximal exponent. In the
 429 binary case we already know that it is 2 since squares are unavoidable in
 430 words whose length is greater than 3.

n	5	6	7	8	9	10	11	12
binary	2	3	4	5	5	6	6	8
ternary	(2, 1.5)	(3, 1.5)	(4, 2)	(5, 2)	(5, 2)	(6, 1.5)	(6, 2)	(8, 2)
4-ary	(2, 1.5)	(3, 1.5)	(4, 2)	(5, 2)	(5, 2)	(6, 1.5)	(7, 1.5)	(8, 2)
	13	14	15	16	17	18	19	20
	8	9	9	11	11	12	12	14
	(8, 2)	(9, 2)	(9, 2)	(11, 2)	(11, 2)	(12, 2)	(12, 2)	(14, 2)
	(8, 1.5)	(9, 1.5)	(10, 1.5)	(11, 2)	(12, 1.5)	(12, 1.5)	(13, 1.5)	(14, 1.5)

433 *6.2. Lower bound*

434 We now deal with a lower bound on the maximal number of occurrences
 435 of maximal-exponent factors. We first consider an infinite word whose factors
 436 have maximal exponent $3/2$ and then show that its prefixes contain a linear
 437 number of occurrences of these factors.

There exists an infinite word on the four-letter alphabet $A_4 = \{a, b, c, d\}$ whose maximal exponent of its factors is $7/5$. The existence of such a word was proved by Pansiot [15] and it is easy to see that the exponent value cannot be smaller for an infinite word on A_4 . Indeed, the result is part of the conjecture of Dejean [5] who stated the repetitive threshold for all alphabet sizes; the proof of this conjecture was eventually completed by Rao [7] and by Currie and Rampersad [8]. Here is an example of such a word given by Pansiot [15]:

$$\mathbf{p} = \text{bacdabcadcbacdbcabdacbad} \dots$$

From the word \mathbf{p} we define \mathbf{q} on the alphabet $A_5 = \{a, b, c, d, e\}$ by inserting letter e in between any two consecutive letters. That is, for each integer $i \geq 0$,

$$\begin{aligned} \mathbf{q}[2i] &= e \\ \mathbf{q}[2i + 1] &= \mathbf{p}[i] \end{aligned}$$

or in other words $\mathbf{q} = f(\mathbf{p})$, where f is the morphism defined by $f(a) = ea$, for any letter $a \in A_4$. The word \mathbf{q} is:

$$\mathbf{q} = \text{ebeaecedeaebeceaeedecebeaecedebeceaebedeaecebeaed} \dots$$

438 Let uvu be a factor of \mathbf{p} , where u is its longest border and then $|uv|$ is its
 439 smallest period. By the choice of \mathbf{p} , we have $\exp(uvu) = |uvu|/|uv| \leq 7/5$.
 440 In addition, we know that the period length of all $7/5$ -powers in \mathbf{p} is at
 441 least 10 (see [20]). Thus the induced factor $f(uvu)e$ in \mathbf{q} has exponent
 442 $(2|uvu| + 1)/2|uv|$, which is $29/20$ when uvu is a $7/5$ -power. This value is
 443 less than $3/2$.

444 As another example, consider the factor abcda of \mathbf{p} . It has exponent $5/4$
 445 and its induced factor in \mathbf{q} , $f(\text{abcda})e = \text{eaebeceae}$, has exponent $11/8$,
 446 which is less than $3/2$ again. By contrast, the factor abca of \mathbf{p} has exponent
 447 $4/3$ and its induced factor in \mathbf{q} , eaebeceae has exponent $9/6 = 3/2$.

448 The next lemma shows that very few factors of \mathbf{q} have exponent $3/2$, the
 449 maximal value.

450 **Lemma 10.** *Let w be a factor of \mathbf{q} , then $\exp(w) \leq 3/2$. Additionally*
 451 *$\exp(w) = 3/2$ when $w = f(uvu)\mathbf{e}$ with either $uvu = v = \mathbf{a}$ or $u = \mathbf{a}$ and*
 452 *$v = \mathbf{bc}$ up to a permutation of letters.*

453 **Proof.** Let w be a factor with maximal exponent among the factors of \mathbf{q} .
 454 Its first letter is \mathbf{e} because otherwise its length could be increased by one unit
 455 without changing the period, which would increase the exponent. Similarly,
 456 its last letter is \mathbf{e} . Then, w is of the form $f(uvu)\mathbf{e}$ for a factor uvu of \mathbf{p} whose
 457 longest border is u .

Assume that $\exp(w) \geq 3/2$. Then

$$\frac{2|uvu| + 1}{2|uv|} \geq 3/2,$$

which gives

$$2|u| + 1 \geq |uv|.$$

Also, since uvu is a factor of \mathbf{p} , it satisfies

$$|uvu|/|uv| \leq 7/5,$$

which implies

$$\frac{5}{2}|u| \leq |uv|.$$

Therefore

$$\frac{5}{2}|u| \leq 2|u| + 1,$$

458 which is only possible for $|u| = 0, 1$, or 2 .

459 If $|u| = 0$, $|v| = |uv| = 1$, and the induced factor in \mathbf{q} is of the form \mathbf{eae} ,
 460 for a letter $a \in A_4$, and has exponent $3/2$.

461 If $|u| = 1$, $|uv| = 3$, and then uvu is of the form \mathbf{abca} up to a permutation
 462 of letters, inducing a factor of exponent $3/2$ in \mathbf{q} .

463 Finally, if $|u| = 2$, $|uv| = 5$ and $\exp(uvu) = 7/5$. But as recalled above,
 464 no factor of \mathbf{p} with that exponent has period 5. This case is impossible,
 465 which concludes the proof. ■

466 The conclusion of the previous lemma is that the maximal exponent of
 467 factors is $3/2$. The lower bound on the occurrence number of $3/2$ -powers in
 468 \mathbf{q} requires another property of \mathbf{p} , which is used in the proof of the following
 469 corollary.

470 **Corollary 11.** *The number of occurrences of maximal-exponent factors in*
471 *prefixes of \mathbf{q} tends to $2n/3$ with the prefix length n .*

472 **Proof.** From the previous lemma, maximal-exponent factors in \mathbf{q} are
473 induced by factors of the form \mathbf{a} or \mathbf{abca} , up to a permutation of the four
474 letter of A_4 , in \mathbf{p} .

475 It is clear from the definition of \mathbf{q} that at every two of its positions occur
476 one of the factors \mathbf{eae} , \mathbf{ebe} , \mathbf{ece} , \mathbf{ede} . Their occurrence number then tends
477 to $n/2$.

478 Turning to the other factors of exponent $3/2$, it is known that the six
479 factors of the form \mathbf{abca} appear at every three positions in \mathbf{p} . Indeed, an
480 occurrence of \mathbf{abca} , can extend to \mathbf{abcad} and \mathbf{abcadb} but not to \mathbf{abcadb}
481 whose suffix \mathbf{bcadb} has exponent $6/4 = 3/2 > 7/5$. Therefore, the induced
482 factors of exponent $3/2$ occur at every six positions in \mathbf{q} , leading to a limit
483 of $n/6$.

484 Summing up the two limits, the occurrence numbers of $3/2$ -powers in
485 prefixes of \mathbf{q} tend to $n/2 + n/6 = 2n/3$ as stated. \blacksquare

486 7. Conclusion

487 The result of Section 6 implies that Algorithm MAXEXPFACTOR can be mod-
488 ified to output all the MEFs occurring in the input word in the same asymp-
489 totic time. Indeed, the only occurrences of MEFs that are skipped by the
490 algorithm when computing the maximal exponent are those occurring inside
491 a phrase of the f -factorisation (Case (i) of Section 3). However storing the
492 previous occurrences of MEFs and listing them can be done in time propor-
493 tional to their number, which does not affect the asymptotic running time of
494 the algorithm and yields the next statement.

495 **Corollary 12.** *All the occurrences of maximal-exponent factors of a word*
496 *can be listed in linear time with respect to its length.*

497 The present work triggers the study of a uniform solution to compute
498 both repetitions (of exponent at least 2) and repeats. However, exponent
499 2 seems to reflect a transition phase in the combinatorics of these studied
500 objects. For instance, the number of repetitions in a word can be of the order
501 of $n \log n$ and the number of maximal periodicities (runs) is linear, while the
502 number of maximal occurrences of factor uvu can be quadratic.

503 An interesting question is to select factors related to repeats that occur
504 only a linear number of times or slightly more. An attempt has been achieved
505 in [21] where it is shown that the number of maximal repetitions of any
506 exponent more than $1 + \epsilon$ is bounded by $\frac{1}{\epsilon}n \ln n$. See also the discussions at
507 the end of [10] and of [22].

508 Other interesting problems are the exact evaluation of the maximal num-
509 ber of occurrences of MEF and the calculation of the maximal number of
510 (distinct) MEFs occurring in a word.

511 8. Acknowledgements

512 We warmly thank G. Kucherov and R. Kolpakov for interesting discus-
513 sions on repetitions, runs and repeats in words. We also thank the reviewers
514 whose comments greatly helped us improve the presentation of this article.

515 References

- 516 [1] G. Badkobeh, M. Crochemore, C. Toopsuwan, Computing the maximal-
517 exponent repeats of an overlap-free string in linear time, in: L. Calderon-
518 Benavides, C. Gonzalez-Caro, E. Chavez, N. Ziviani (Eds.), Symposium
519 on String Processing and Information Retrieval, Vol. 7608 of Lecture
520 Notes in Computer Science, Springer, 2012, pp. 61–72.
- 521 [2] D. Gusfield, Algorithms on Strings, Trees and Sequences: Computer
522 Science and Computational Biology, Cambridge University Press, Cam-
523 bridge, 1997.
- 524 [3] G. S. Brodal, R. B. Lyngsø, C. N. S. Pedersen, J. Stoye, Finding max-
525 imal pairs with bounded gap, in: M. Crochemore, M. Paterson (Eds.),
526 Combinatorial Pattern Matching, Vol. 1645 of Lecture Notes in Com-
527 puter Science, Springer, 1999, pp. 134–149.
- 528 [4] L. Vuillon, A characterization of Sturmian words by return words, Eur.
529 J. Comb. 22 (2) (2001) 263–275.
- 530 [5] F. Dejean, Sur un théorème de Thue, J. Comb. Theory, Ser. A 13 (1)
531 (1972) 90–99.
- 532 [6] A. Thue, Über unendliche Zeichenreihen, Norske Vid. Selsk. Skr. I Math-
533 Nat. Kl. 7 (1906) 1–22.

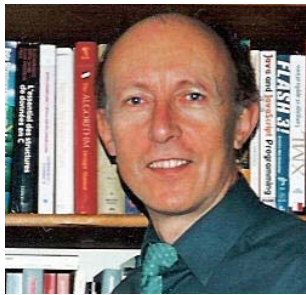
- 534 [7] M. Rao, Last cases of Dejean’s conjecture, *Theor. Comput. Sci.* 412 (27)
535 (2011) 3010–3018.
- 536 [8] J. D. Currie, N. Rampersad, A proof of Dejean’s conjecture, *Math.*
537 *Comput.* 80 (274) (2011) 1063–1070.
- 538 [9] M. Crochemore, C. Hancart, T. Lecroq, *Algorithms on Strings*, Cam-
539 bridge University Press, 2007, 392 pages.
- 540 [10] R. Kolpakov, G. Kucherov, On maximal repetitions in words, *J. Discret.*
541 *Algorithms* 1 (1) (2000) 159–186.
- 542 [11] W. Rytter, The number of runs in a string, *Inf. Comput.* 205 (9) (2007)
543 1459–1469.
- 544 [12] M. Crochemore, L. Ilie, L. Tinta, The “runs” conjecture, *Theor. Com-*
545 *put. Sci.* 412 (27) (2011) 2931–2941.
- 546 [13] J. Ziv, A. Lempel, A universal algorithm for sequential data compres-
547 sion, *IEEE Trans. Inf. Theory* 23 (1977) 337–343.
- 548 [14] M. Crochemore, G. Tischler, Computing longest previous non-
549 overlapping factors, *Inf. Process. Lett.* 111 (2011) 291–295.
- 550 [15] J.-J. Pansiot, A propos d’une conjecture de F. Dejean sur les répétitions
551 dans les mots, in: J. Díaz (Ed.), *Automata, Languages and Program-*
552 *ming*, 10th Colloquium, Barcelona, Spain, July 18-22, 1983, *Proceed-*
553 *ings*, Vol. 154 of *Lecture Notes in Computer Science*, Springer, 1983,
554 pp. 585–596.
- 555 [16] O. Berkman, C. S. Iliopoulos, K. Park, The subtree max gap problem
556 with application to parallel string covering, *Inf. Comput.* 123 (1) (1995)
557 127–137.
- 558 [17] C. S. Iliopoulos, D. W. G. Moore, K. Park, Covering a string, *Algorith-*
559 *mica* 16 (3) (1996) 288–297.
- 560 [18] G. S. Brodal, C. N. S. Pedersen, Finding maximal quasiperiodicities
561 in strings, in: R. Giancarlo, D. Sankoff (Eds.), *Combinatorial Pattern*
562 *Matching*, Vol. 1848 of *Lecture Notes in Computer Science*, Springer,
563 2000, pp. 397–411.

- 564 [19] M. Christou, M. Crochemore, C. S. Iliopoulos, M. Kubica, S. P. Pissis,
565 J. Radoszewski, W. Rytter, B. Szreder, T. Walen, Efficient seeds com-
566 putation revisited, in: R. Giancarlo, G. Manzini (Eds.), *Combinatorial*
567 *Pattern Matching*, Vol. 6661 of *Lecture Notes in Computer Science*,
568 Springer, Berlin, 2011, pp. 350–363.
- 569 [20] G. Badkobeh, M. Crochemore, M. Rao, Finite repetition threshold for
570 large alphabets, *RAIRO - Theor. Inf. and Applic.* (2014) to appear.
- 571 [21] R. Kolpakov, G. Kucherov, P. Ochem, On maximal repetitions of arbi-
572 trary exponent, *Inf. Process. Lett.* 110 (7) (2010) 252–256.
- 573 [22] M. Crochemore, L. Ilie, Maximal repetitions in strings, *J. Comput. Syst.*
574 *Sci.* 74 (2008) 796–807.



575

576 Dr. Golnaz Badkobeh completed her PhD in Computer Science at King's
577 College London in 2013 after obtained a BSc in Mathematics at University
578 of Bristol in 2008. She has been working in the field of Combinatorics on
579 Words and string processing algorithms, areas in which she has published
580 eight scholarly research articles in her academic career. She currently has
581 three more articles under review.



582

583 Maxime Crochemore is presently Professor at King's College London and
584 Emeritus Professor of Université Paris-Est. His research interests are in the
585 design and analysis of algorithms. His major achievements are on word al-
586 gorithms, which includes pattern matching, text indexing, coding, and text
587 compression. He also works on the combinatorial background of these sub-
588 jects and on their applications to bio-informatics. He has co-authored several
589 textbooks on algorithms and published more than 200 articles. He partici-
590 pated in a large number of international projects on algorithms and super-
591 vised to completion more than twenty PhD students.