Véges szavak általánosított részszó-bonyolultsága

KÁSA Zoltán

Sapientia Erdélyi Magyar Tudományegyetem
Kolozsvár–Marosvásárhely–Csíkszereda
Matematika-Informatika Tanszék, Marosvásárhely

Budapest, 2010. szept. 9

Értelmezés

Legyenek n, d_1, d_2 *és* s *pozitív egész számok, és* $u = x_1x_2\ldots x_n \in \Sigma^n$ *egy* Σ *ábécé feletti szó. A* $v = x_{i_1}x_{i_2}\ldots x_{i_s}$ *szó, ahol*
$i_1 \geq 1$,
$d_1 \leq i_{j+1} - i_j \leq d_2$, *ha* $j = 1, 2, \ldots, s-1$,
$i_s \leq n$,
az u *szó* s *hosszúságú* (d_1, d_2)*-részszava.*

Például, az *aabcade* szóban a $(2, 4)$-részszavak:

a, *ab*, *ac*, *aba*, *aa*, *acd*, *abd*, *aae*, *abae*, *ace*, *abe*,
ad,
b, *ba*, *bd*, *bae*, *be*,
c, *cd*, *ce*,
ae,
d,
e.

Értelmezés

Egy adott szó összes, egymástól különböző (d_1, d_2)-részszavának számát az adott szó (d_1, d_2)-bonyolultságának nevezzük.

$d_1 = 1$

- A. Iványi, On the d-complexity of words, *Annales Univ. Sci. Budapest., Sect. Computatorica,* **8** (1987) 69–90.
- Z. Kása, On the d-complexity of strings, *Pure Math. Appl.*, **9,** 1–2 (1998) 119–128.

$d_2 = n - 1$

- Z. Kása, Super-d-complexity of finite words, *8th MaCs*, Komárno, July 14–17, 2010.

Let Σ be an alphabet, Σ^n the set of all n-length words over Σ, Σ^* the set of all finite word over Σ.

Definition

Let n, d and s be positive integers, and $u = x_1 x_2 \ldots x_n \in \Sigma^n$. A d-subword of length s of u is defined as $v = x_{i_1} x_{i_2} \ldots x_{i_s}$ where
$i_1 \geq 1$,
$1 \leq i_{j+1} - i_j \leq d$ for $j = 1, 2, \ldots, s-1$,
$i_s \leq n$.

$u = bear$

2-subwords:

b, e, a, r
be, ba, ea, er, ar
bea, ber, bar, ear
$bear$

Definition

Let n, d and s be positive integers, and $u = x_1 x_2 \ldots x_n \in \Sigma^n$. A super-$d$-subword of length s of u is defined as $v = x_{i_1} x_{i_2} \ldots x_{i_s}$ where
$i_1 \geq 1$,
$d \leq i_{j+1} - i_j < n$ for $j = 1, 2, \ldots, s-1$,
$i_s \leq n$.

u= abcdef

super-2-subwords:

a, ac, ad, ae, af, ace, acf, adf,
b, bd, be, bf, bdf,
c, ce, cf,
d, df,
e,
f

Definition

The super-d-complexity of a word is the number of all its different super-d-subwords.

$u = abcdef$

super-2-subwords:

a, ac, ad, ae, af, ace, acf, adf,
b, bd, be, bf, bdf,
c, ce, cf,
d, df,
e,
f

The super-2-complexity of this word is 20.

Super-d-complexity of rainbow words

Words with different letters are called rainbow words.
The super-d-complexity of an n-length rainbow word: $S(n, d)$.
Let us denote by $b_{n,d}(i)$ the number of super-d-subwords which begin in the position i in an n-length rainbow word.

u=abcdef

$b_{6,2}(1) = 8$: *a, ac, ad, ae, af, ace, acf, adf*
$b_{6,2}(2) = 5$, $b_{6,2}(3) = 3$, $b_{6,2}(4) = 2$, $b_{6,2}(5) = 1$, $b_{6,2}(6) = 1$.

$b_{n,d}(i) = 1 + b_{n,d}(i+d) + b_{n,d}(i+d+1) + \cdots + b_{n,d}(n),$
for $n > d, 1 \leq i \leq n - d$,

$b_{n,d}(1) = 1 \text{ for } n \leq d.$

The super-d-complexity of rainbow words can be computed by the formula:

$$S(n,d) = \sum_{i=1}^{n} b_{n,d}(i).$$

This can be expressed also as

$$S(n,d) = \sum_{k=1}^{n} b_{k,d}(1),$$

because of the formula

$$S(n+1,d) = S(n,d) + b_{n+1,d}(1).$$

In the case $d = 1$ the complexity $S(n,1)$ can be computed easily: $S(n,1) = 2^n - 1$. This is equal to the n-complexity of n-length rainbow words.

Computing super-d-complexity of rainbow words

- by recursive algorithms
- by mathematical formulas
- by graph algorithms

```
for k ← 1 to n
    do b_k ← −1
B(n, d, i)
```

−1	−1	−1	−1	−1	−1	−1	−1

```
B(n, d, i)

p ← 1
for k ← i + d to n
    do if b_k = −1
          then B(n, d, k)
       p ← p + b_k
b_i ← p
return
```

$B(8, 2, 1)$: $b_7 = 1$, $b_8 = 1$, $b_5 = 3$, $b_6 = 2$, $b_3 = 8$, $b_4 = 5$, $b_1 = 21$.

21	−1	8	5	3	2	1	1

Lemma

$b_{n,2}(1) = F_n$, where F_n is the nth Fibonacci number.

Theorem

$S(n,2) = F_{n+2} - 1$, where F_n is the nth Fibonacci number.

Let us denote by $M_{n,d} = b_{n,d}(1)$,

$$M_{n,d} = M_{n-1,d} + M_{n-d,d}, \quad \text{for } n \geq d \geq 2,$$
$$M_{0,d} = 0,\ M_{1,d} = 1,\ \ldots,\ M_{d-1,d} = 1.$$

Let us name this sequence *d*-middle sequence. Because of the $M_{n,2} = F_n$ equality, the *d*-middle sequence can be considered as a generalization of the Fibonacci sequence.

The next algorithm computes $M_{n,d}$, by using an array $M_0, M_1, \ldots, M_{d-1}$ to store the necessary previous elements:

MIDDLE(n, d)

$M_0 \leftarrow 0$
for $i \leftarrow 1$ **to** $d - 1$
  do $M_i \leftarrow 1$
for $i \leftarrow d$ **to** n
  do $M_{i \bmod d} \leftarrow M_{(i-1) \bmod d} + M_{(i-d) \bmod d}$
    print $M_{i \bmod d}$
return

Using the generating function $M_d(z) = \sum_{n \geq 0} M_{n,d} z^n$, the following closed formula results:

$$M_d(z) = \frac{z}{1 - z - z^d}.$$

This can be used to compute the sum $s_{n,d} = \sum_{n=1}^{n} M_{i,d}$, which is the coefficient of z^{n+d} in the expansion of the function

$$\frac{z^d}{1 - z - z^d} \cdot \frac{1}{1 - z} = \frac{z^d}{1 - z - z^d} + \frac{z}{1 - z - z^d} - \frac{z}{1 - z}.$$

So $s_{n.d} = M_{n+(d-1),d} + M_{n,d} - 1 = M_{n+d,d} - 1$. Therefore

$$\sum_{i=1}^{n} M_{i,d} = M_{n+d,d} - 1.$$

Theorem

$S(n,d) = M_{n+d,d} - 1$, where $n > d$ and $M_{n,d}$ is the nth elements of d-middle sequence.

Computing super-d-complexity by mathematical formulas

Theorem

$$S(n,d) = \sum_{k\geq 0} \binom{n-(d-1)k}{k+1}, \text{ for } n \geq 2, d \geq 1.$$

Theorem

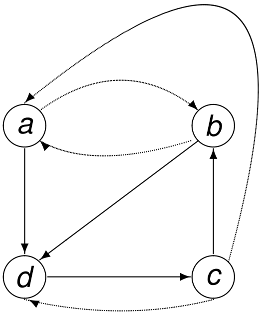
$$b_{n+1,d}(1) = \sum_{k\geq 0} \binom{n-(d-1)k}{k}, \text{ for } n \geq 1, d \geq 1.$$

$$\sum_{k\geq 0} \binom{n-(d-1)k}{k+1} = \sum_{i=1}^{n} \sum_{k\geq 0} \binom{i-1-(d-1)k}{k},$$

and from this

$$\sum_{i=1}^{n} \binom{i-1-(d-1)k}{k} = \binom{n-(d-1)k}{k+1}$$

Latin négyzet



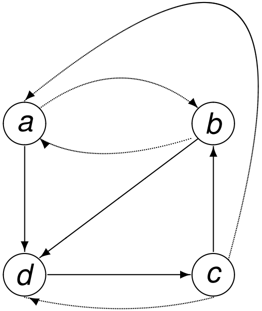
$$L = \begin{pmatrix} 0 & ab & 0 & ad \\ ba & 0 & 0 & bd \\ ca & cb & 0 & cd \\ 0 & 0 & dc & 0 \end{pmatrix} \qquad L^* = \begin{pmatrix} 0 & b & 0 & d \\ a & 0 & 0 & d \\ a & b & 0 & d \\ 0 & 0 & c & 0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0 & 0 & adc & abd \\ 0 & 0 & bdc & bad \\ cba & cab & 0 & \left\{ \begin{matrix} cad \\ cbd \end{matrix} \right\} \\ dca & dcb & 0 & 0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0 & adcb & abdc & 0 \\ bdca & 0 & badc & 0 \\ 0 & 0 & 0 & \left\{ \begin{matrix} cbad \\ cabd \end{matrix} \right\} \\ dcba & dcab & 0 & 0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} \left\{ \begin{matrix} adcba \\ abdca \end{matrix} \right\} & 0 & 0 & 0 \\ 0 & \left\{ \begin{matrix} bdcab \\ badcb \end{matrix} \right\} & 0 & 0 \\ 0 & 0 & \left\{ \begin{matrix} cbadc \\ cabdc \end{matrix} \right\} & 0 \\ 0 & 0 & 0 & \left\{ \begin{matrix} dcbad \\ dcabd \end{matrix} \right\} \end{pmatrix}$$

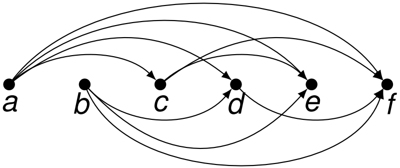
Az L^3 elemei szerint a gráfban 8 Hamilton-út van, L^4 szerint pedig két Hamilton-kör (a mátixban ezek mindegyike négyszer jelenik meg, hisz egy kör bármelyik csúccsal kezdődhet).



adcba, *abdca*

Computing super-d-complexity by graph algorithms

$G = (V, E)$, with
$V = \{a_1, a_2, \ldots, a_n\}$,
$E = \{(a_i, a_j) \mid j - i \geq d,\ i = 1, 2, \ldots, n, j = 1, 2, \ldots, n\}$.



Graph for super-2-subwords when $n = 6$.

The adjacency matrix $A = (a_{ij})_{\substack{i=\overline{1,n} \\ j=\overline{1,n}}}$ of the graph is defined by:

$$a_{ij} = \begin{cases} 1, & \text{if } j - i \geq d, \\ 0, & \text{otherwise,} \end{cases} \qquad \text{for } i = 1, 2, \ldots, n, j = 1, 2, \ldots, n.$$

$$R = I + A + A^2 + \cdots + A^k, \text{ where } A^{k+1} = O \,(\text{the null matrix}).$$

The super-d-complexity of a rainbow word is then

$$S(n, d) = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij}.$$

Matrix R can be better computed using a variant of the well-known Warshall algorithm:

WARSHALL(A, n)

$W \leftarrow A$
for $k \leftarrow 1$ **to** n
  do for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
      do $w_{ij} \leftarrow w_{ij} + w_{ik} w_{kj}$
return W

From W we obtain easily $R = I + W$.

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

After applying the Warshall algorithm:

$$W = \begin{pmatrix} 0 & 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad R = \begin{pmatrix} 1 & 0 & 1 & 1 & 2 & 3 \\ 0 & 1 & 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and then $S(6,2) = 20$, the sum of elements in R.

The Warshall algorithm combined with the Latin square method can be used to obtain all nontrivial (with length at least 2) super-d-subwords of a given n-length rainbow word $a_1a_2\cdots a_n$. Let us consider a matrix $\mathcal{A}$ with the elements A_{ij} which are set of strings. Initially this matrix is defined as:

$$A_{ij} = \begin{cases} \{a_ia_j\}, & \text{if } j - i \geq d, \\ \emptyset, & \text{otherwise}, \end{cases} \qquad \text{for } i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, n.$$

If $\mathcal{A}$ and $\mathcal{B}$ are sets of strings, $\mathcal{AB}$ will be formed by the set of concatenation of each string from $\mathcal{A}$ with each string from $\mathcal{B}$:

$$\mathcal{AB} = \big\{ab \,\big|\, a \in \mathcal{A}, b \in \mathcal{B}\big\}.$$

If $s = s_1s_2\cdots s_p$ is a string, let us denote by $'s$ the string obtained from s by eliminating the first character: $'s = s_2s_3\cdots s_p$. Let us denote by $'A_{ij}$ the set A_{ij} in which we eliminate from each element the first character. In this case $'\mathcal{A}$ is a matrix with elements $'A_{ij}$.

Starting with the matrix $\mathcal{A}$ defined as before, the algorithm to obtain all nontrivial super-d-subwords is the following:

WARSHALL-LATIN($\mathcal{A}, n$)

```
𝒲 ← 𝒜
for k ← 1 to n
    do for i ← 1 to n
        do for j ← 1 to n
            do if W_ik ≠ ∅ and W_kj ≠ ∅
                then W_ij ← W_ij ∪ W_ik ' W_kj
return 𝒲
```

The set of nontrivial super-d-subwords is $\bigcup_{i,j \in \{1,2,\ldots,n\}} W_{ij}$.

For $n = 8$, $d = 3$ the initial matrix is:

$$\begin{pmatrix} \emptyset & \emptyset & \emptyset & \{ad\} & \{ae\} & \{af\} & \{ag\} & \{ah\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \{be\} & \{bf\} & \{bg\} & \{bh\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{cf\} & \{cg\} & \{ch\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{dg\} & \{dh\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{eh\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}.$$

The result of the algorithm in this case is:

$$\begin{pmatrix}
\emptyset & \emptyset & \emptyset & \{ad\} & \{ae\} & \{af\} & \{ag, adg\} & \{ah, adh, aeh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \{be\} & \{bf\} & \{bg\} & \{bh, beh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{cf\} & \{cg\} & \{ch\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{dg\} & \{dh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{eh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset
\end{pmatrix}.$$

The general case

In the general case for any word $w \in \Sigma^*$, let us denote the super-d-complexity by $S_w(d)$. We have

$$\left\lceil \frac{|w|}{d} \right\rceil \leq S_w(d) \leq S(|w|, d),$$

where $|w|$ is the length of w. The minimal value is obtained for a trivial word $w = a \ldots a$, and the maximal one for a rainbow word.

The algorithm WARSHALL-LATIN can be used for nonrainbow words too, with the remark that repeating subwords must be eliminated. For the word *aabbbaaa* and $d = 3$ the result is: *aa*, *ab*, *aba*, *ba*. (nontrivial subwords only)

w	$S_w(2)$
00000	3
00001	5
00010	5
00011	5
00100	6
00101	6
00110	6
00111	5
01000	5
01001	7
01010	7
01011	6
01100	6
01101	7
01110	7
01111	5

Max $S_w(2)$ is 7.

Let us denote by $f(m,n,d)$ the maximal value of the super-d-complexity of all words of length n over an alphabet of m letters:

$$f(m,n,d) = \max_{\substack{w \in \Sigma^n \\ m = |\Sigma|}} \Big(S_w(d)\Big).$$

	2	3	4	5	6	7	8	9	10
7	14	7	6	5	3	-	-	-	-
8	19	10	6	6	5	3	-	-	-
9	26	13	7	6	6	5	3	-	-
10	35	15	10	6	6	6	5	3	-
11	47	19	13	7	6	6	6	5	3

Theorem

$f(2, n, n-1) = 3$ for $n \geq 3$.
$f(2, n, n-2) = 5$ for $n \geq 4$.

If $\left\lceil \frac{n}{2} \right\rceil \leq d \leq n-3$ then $f(2, n, d) = 6$ for $n \geq 6$.

If n is even, then $f\left(2, n, \frac{n-2}{2}\right) = 10$ for $n \geq 6$.

If n is odd, then $f\left(2, n, \frac{n-1}{2}\right) = 7$ for $n \geq 5$.

$f(2,n,2) = f(2,n-1,2) + f(2,n-2,2) - f(2,n-4,2)$ *for* $n \geq 7$.

- W. Ebeling, R. Feistel, *Physik der Selbstorganisation und Evolution,* Akademie-Verlag, Berlin, 1982.
- C. Elzinga, S. Rahmann, H. Wung, Algorithms for subsequence combinatorics, *Theor. Comput. Sci.*, **409,** 3 (2008) 394–404.
- C. H. Elzinga, Complexity of categorial time series, *Sociological Methods & Research*, **38,** 3 (2010) 463–481. http://home.fsw.vu.nl/ch.elzinga/Complexity%20Preliminary.pdf
- O. G. Troyanskaya, O. Arbell, Y. Koren, G. M. Landau, A. Bolshoy, Sequence complexity profiles of prokaryotic genomic sequences: A fast algorithm for calculating linguistic complexity, *Bioinformatics,* **18,** 5 (2002) 679–688.
- N. J. A. Sloane, The on-line encyclopedia of integer sequences, `http://www.research.att.com/~njas/sequences/`