Contents

29. Perfect Arrays

Sequences of elements of given sets of symbols have a great importance in different branches of natural sciences. For example in biology the 4-letter set $\{A, C, G, T\}$ containing the nucleotids (adenine, cytosine, guanine and thymine) and the 20-letter set $\{a, c, d, e, f, g, h, i, k, l, m, \ n, p, q, r, s, t, v, w, z\}$ containing the amino-acids (alanine, cysteine, asparagin-acid, glutamine-acid, phenyl, glycine, histidine, isoleucine, lysine, leucine, methionine, asparagine, proline, glutamine, arginine, serine, threonine, valine, triptophan, tyrosine) play leading role [31].
Arrays with elements from given sets of symbols have various applications, e.g. in frequency allocation of multibeam satellites [15], in designing mask configuration for spectrometers [21] and in cryptography [44]. In [42] possible applications in picture coding and processing are suggested.

Complexity is a basic characteristic of arrays of symbols, since affects the cost of storage and reproduction, and the quantity of information stored in the arrays. The usual complexity measures are based on the time (or memory) needed for generating or recognizing them.

Cyclic sequences in which every possible sequence of a fixed length occurs exactly once have been studied for more than a hundred years [19]. This mathematical problem was extended to two-dimensional arrays by Reed and Stewart [51] who gave an example of 4×4 sized *perfect map*. Fan et al. in 1985 [17] proved the existence of binary prerfect maps for larger sizes.

Complexity is a basic characteristic of symbol arrays, since affects the cost of storage and reproduction, and the quantity of information stored in the symbol sequences. The usual complexity measures of symbol sequences are based on the time (or memory) needed for generating or recognizing them.

In this chapter we

29.1. Basic concepts

Let $\mathbb{Z}$ be the set of integers. For $u, \ v \in \mathbb{Z}$ we denote the set $\{j \in \mathbb{Z} \mid u \leq j \leq v\}$ by $[u..v]$ and the set $\{j \in \mathbb{Z} \mid j \geq u\}$ by $[u..\infty]$. Let $d \in [1..\infty]$ and $k, \ n \in [2..\infty]$, $b_i, \ c_i, \ j_i \in [1..\infty]$ $(i \in [1..d])$ and $a_i, \ k_i \in [2..\infty]$ $(i \in [1..d])$. Let $\mathbf{a} = \langle a_1, a_2, \ldots, a_d \rangle$, $\mathbf{b} = \langle b_1, b_2, \ldots, b_d \rangle$, $\mathbf{c} = \langle c_1, c_2, \ldots, c_d \rangle$, $\mathbf{j} = \langle j_1, j_2, \ldots, j_d \rangle$ and $\mathbf{k} = \langle k_1, k_2, \ldots, k_d \rangle$

be vectors of length d, $\mathbf{n} = \langle n_1, n_2, \ldots \rangle$ an infinite vector with $2 \le n_1 < n_2 < \cdots$.

A ***d-dimensional n-ary array A*** is a mapping $A : [1..\infty]^d \to [0, n-1]$.

If there exist a vector $\mathbf{b}$ and an array M such that

$$\forall \mathbf{j} \in [1..\infty]^d : A[\mathbf{j}] = M[(j_1 \bmod b_1) + 1, (j_2 \bmod b_2) + 1, \ldots, (j_d \bmod b_d) + 1],$$

then A is a **b**-***periodic array*** and M is a ***period*** of A.

The **a**-***sized subarrays*** of A are the **a**-periodic n-ary arrays.

Although our arrays are infinite we say that a **b**-periodic array is **b**-***sized.***

Indexset A_{index} of a **b**-periodic array A is the Cartesian product

$$A_{index} = [1..b_1] \times [1..b_2] \times \cdots \times [1..b_d]\,.$$

A d dimensional **b**-periodic n-ary array A is called $(n, d, \mathbf{a}, \mathbf{b})$-***perfect,*** if all possible n-ary arrays of size **a** appear in A exactly once as a subarray.

Here n is the alphabet size, d gives the number of dimensions of the "window" and the perfect array M, the vector **a** characterizes the size of the window, and the vector **b** is the size of the perfect array M.

An $(n, d, \mathbf{a}, \mathbf{b})$-perfect array A is called **c**-***cellular,*** if c_i divides b_i for $i \in [1..d]$. A cellular array consists of $b_1/c_1 \times b_2/c_2 \times \cdots \times b_d/c_d$ disjoint subarrays of size **c**, called ***cells.*** In each cell the element with smallest indices is called the ***head*** of the cell. The contents of the cell is called ***pattern.***

The product of the elements of a vector **a** is called the ***volume*** of the vector and is denoted by $|\mathbf{a}|$. The number of elements of the perfect array M is called the ***volume*** of M and is denoted by $|M|$.

If $b_1 = b_2 = \cdots = b_d$, then the $(n, d, \mathbf{a}, \mathbf{b})$-perfect array A is called **symmetric**. If A is symmetric and $a_1 = a_2 = \cdots = a_d$, then A is called **doubly symmetric.** If A is doubly symmetric and

1. $d = 1$, then A is called a ***double sequence;***
2. $d = 2$, then A is called a ***double square;***
3. $d = 3$, then A is called a ***double cube.***

According to this definition, all perfect sequences are doubly symmetric. In the case of symmetric arrays we use the notion $(n, d, \mathbf{a}, b)$ and in the case of doubly symmetric arrays we use (n, d, a, b) instead of $(n, d, \mathbf{a}, \mathbf{b})$.

The first known result originates from Flye-Sainte who proved the existence of $(2, 1, a, 2^a)$-perfect sequences for all possible values of a in 1894.

One dimensional perfect arrays are often called de Bruijn or Good sequences. Two dimensional perfect arrays are called also perfect maps or de Bruijn tori.

Even De Bruijn sequences—introduced by Antal Iványi and Zoltán Tóth in 1988—are useful in construction of perfect arrays when the size of the alphabet is an even number and the window size is 2×2. Their definition is as follows.

If n is an even integer then an $(n, 1, 2, n^2)$-perfect sequence $M = (m_1, m_2, \ldots, m_{n^2})$ is called ***even,*** if $m_i = x$, $m_{i+1} = y$, $x \ne y, m_j = y$ and $m_{j+1} = x$ imply $j - i$ is even.

Iványi and Tóth in 1988 and later Hurlbert and Isaak in 1994 provided a constructive proof of the existence of even sequences. The later algorithm is stronger since it constructs a universal infinite sequence whos prefixes are even sequences for the corresponding alphabet size.

Lexicographic indexing of an array $M = [m_{j_1 j_2 \ldots j_d}] = [m_{\mathbf{j}}]$ $(1 \leq j_i \leq b_i)$ for $i \in [1..d]$ means that the index $I(m_{\mathbf{j}})$ is defined as

$$I(m_{\mathbf{j}}) = j_1 - 1 + \sum_{i=2}^{d} \left((j_i - 1) \prod_{m=1}^{i-1} b_m \right) .$$

The concept of perfectness can be extended to infinite arrays in various ways. In ***growing arrays*** introduced by G. Hurlbert and G. Isaak in 1994 the window size is fixed, the alphabet size is increasing and the prefixes grow in all d directions.

Let a and d be positive integers with $a \geq 2$ and $\mathbf{n} = \langle n_1, n_2, \ldots \rangle$ be a strictly increasing sequence of positive integers. An array $M = [m_{i_1 i_2 \ldots i_d}]$ is called ***$(\mathbf{n}, d, a)$-growing,*** if the following conditions hold:

1. $M = [m_{i_1 i_2 \ldots i_d}]$ $(1 \leq i_j < \infty)$ for $j \in [1..d]$;
2. $m_{i_1 i_2 \ldots i_d} \in [0..n-1]$;
3. the prefix $M_k = [m_{i_1 i_2 \ldots i_d}]$ $(1 \leq i_j \leq n_k^{a^d/d} for\ j \in [1..d])$ of M is $(n_k, d, a, n_k^{a^d/d})$-perfect array for $k \in [0..\infty]$.

For the growing arrays we use the terms growing sequence, growing square and growing cube.

For $a, n \in [2..\infty]$ the ***new alphabet size*** $N(n, a)$ is

$$N(n, a) = \begin{cases} n, & \textit{if any prime divisor of } a \textit{ divides } n\,, \\ nq, & \textit{otherwise}\,, \end{cases} \tag{29.1}$$

where q is the product of the prime divisors of a not dividing n.

Note, that alphabet size n and new alphabet size N have the property that $n \mid N$.

29.2. Necessary condition and earlier results

Since in the period M of a perfect array A each element is the head of a pattern, the volume of M equals the number of the possible patterns. Since each pattern—among others the pattern containing only zeros—can appear only once, any size of M is greater then the corresponding size of the window. So we have the following necessary condition due to Cock, further Hurlbert and Isaak: If M is an $(n, d, \mathbf{a}, \mathbf{b})$-perfect array, then

$$|\mathbf{b}| = n^{|\mathbf{a}|} \tag{29.2}$$

and

$$b_i > a_i \text{ for } i \in [1..d]\,. \tag{29.3}$$

Different construction algorithms and other results concerning one and two dimensional perfect arrays can be found in the fourth volume of *The Art of Computer Programming* written by D. E. Knuth [37]. E.g. a (2,1,5,32)-perfect array [37, page

22], a 36-length even sequence whose 4-length and 16-length prefixes are also even sequences [37, page 62], a (2,2,2,4)-perfect array [37, page 38] and a (4,2,2,16)-perfect array [37, page 63].

It is known [7, 37] that in the one-dimensional case the necessary condition (29.2) is sufficient too. There are many construction algorithms, like the ones of Cock [10], Fan, Fan, Ma and Siu [17], Martin [43] or any algorithm for constructing of directed Euler cycles [41].

Chung, Diaconis and Graham [9] posed the problem to give a necessary and sufficient condition of the existence of $(n, 2, \mathbf{a}, \mathbf{b})$-perfect arrays.

The conditions (2) and (3) are sufficient for the existence of (2,2,**a**,**b**)-perfect arrays [17] and (n,2,a,b)-perfect arrays [46]. Later Paterson in [47, 48] supplied further sufficient conditions.

Hurlbert and Isaak [26] gave a construction for one and two dimensional growing arrays.

Carla Savage [53] analysed De Bruijn sequences as special Gray codes

29.3. One-dimensional perfect arrays

In the construction of one-dimensional perfect arrays we use the following algorithms.

Algorithm Martin generates one-dimensional perfect arrays. Its inputs are the alphabet size n and the window size a. Its output is an n-ary perfect sequence of length n^a. The output begins with a zeros and always continues with the maximal permitted element of the alphabet.

29.3.1. Pseudocode of the algorithm Quick-Martin

A natural implementation of Martin's algorithm can be found in the chapter *Complexity of words* of this book. The following effective implementation of Martin is due to M. Horváth and A. Iványi[23].

```
Quick-Martin($n, a$)

1 for $i = 0$ to $n^{a-1} - 1$
2     $C[i] = n - 1$
3 for $i = 1$ to $a$
4     $w[i] = 0$
5 for $i = a + 1$ to $n^a$
6     $k = w[i - a + 1]$
7     for $j = 1$ to $a - 1$
8         $k = kn + w[i - a + j]$
9     $w[i] = C[k]$
10    $C[k] = C[k] - 1$
11 return $\mathbf{w}$
```

This algorithm runs in $\Theta(an^a)$ time.

29.3.2. Pseudocode of the algorithm Optimal-Martin

The following implementation of Martin algorithm requires even smaller running time than Quick-Martin.

```
Optimal-Martin($n, a$)

1 for $i = 0$ to $n^{a-1} - 1$
2       $C[i] = n - 1$
3 for $i = 1$ to $a$
4       $w[i] = 0$
5 for $i = a + 1$ to $n^a$
6       $k = w[i - a + 1]$
7       for $j = 1$ to $a - 1$
8               $k = kn + w[i - a + j]$
9       $w[i] = C[k]$
10      $C[k] = C[k] - 1$
11 return $\mathbf{w}$
```

The running time of any algorithm which constructs a one-dimensional perfect array is $\Omega(n^a)$, since the sequence contains n^a elements. The running time of Optimal-Martin is $\Theta(n^a)$.

29.3.3. Pseudocode of the algorithm Shift

Algorithm Shift proposed by Cock in 1988 [10] is a widely usable algorithm to construct perfect arrays. We use it to transform cellular $(N, d, a, \mathbf{b})$-perfect arrays into $(N, d + 1, a, \mathbf{c})$-perfect arrays.

```
Shift($N, d, a, P_d, P_d$)

1 Martin($N^{a^d}, a - 1, \mathbf{w}$)
2 for $j = 0$ to $N^{a^d - a^{d-1}} - 1$
3       transform $w_i$ to an $a^d$ digit $N$-ary number
4       produce the $(j + 1)$-st layer of the output $P_{d+1}$ by multiple shifting
        the $j$th layer of $P_d$ by the transformed number (the first $a$ digits
        give the shift size for the first direction, then the next $a^2 - a$ digits
        in the second direction etc.)
5 return $P_{d+1}$
```

29.3.4. Pseudocode of the algorithm Even

If N is even, then this algorithm generates the N^2-length prefix of an even growing sequence [26].

```
Even($N$)
1 if $N == 2$
2      $w[1] = 0$
```

```
3     $w[2] = 0$
4     $w[3] = 1$
5     $w[4] = 1$
6     return w
7 for $i = 1$ to $N/2 - 1$
8       for $j = 0$ to $2i - 1$
9             $w[4i^2 + 2j + 1] = j$
10      for $j = 0$ to $i - 1$
11            $w[4i^2 + 2 + 4j] = 2i$
12      for $j = 0$ to $i - 1$
13            $w[4i^2 + 4 + 4j] = 2i + 1$
14      for $j = 0$ to $4i - 1$
15            $w[4i^2 + 4i + 1 + j] = w[4i^2 + 4i - j]$
16      $w[4i^2 + 8i + 1] = 2i + 1$
17      $w[4i^2 + 8i + 2] = 2i$
18      $w[4i^2 + 8i + 3] = 2i$
19      $w[4i^2 + 8i + 4] = 2i + 1$
20 return w
```

Algorithm Even [26] produces even de Bruijn sequences.

29.4. Two-dimensional perfect arrays

Chung, Diaconis and Graham posed the problem to give a necessary and sufficient condition of the existence of $(n, 2, \mathbf{a}, \mathbf{b})$-perfect arrays.

As Fan, Fan and Siu proved in 1985, the conditions (2) and (3) are sufficient for the existence of (2,2,**a**,**b**)-perfect arrays. Paterson proved the same in 1994 for $(n, 2, a, b)$-perfect arrays. Later Paterson supplied further sufficient conditions.

Hurlbert and Isaak in 1993 gave a construction for one and two dimensional growing arrays.

29.4.1. Pseudocode of the algorithm Mesh

The following implementation of Mesh is was proposed by Iványi and Tóth in 1988.
Mesh$(N, \mathbf{w}, S)$

```
1 for $i = 1$ to $N^2$
2       for $j = 1$ to $N^2$
3             if    $i + j$ is even
4                   $S[i, j] = w[i]$
5             else $S[i, j] = w[j]$
6 return $S$
```

29.4.2. Pseudocode of the algorithm Cellular

This is an extension and combination of the known algorithms Shift, Martin, Even and Mesh.

Cellular results cellular perfect arrays. Its input data are n, d and $\mathbf{a}$, its output is an $(N, d, \mathbf{a}, \mathbf{b})$-perfect array, where $b_1 = N^{a_1}$ and $b_i = N^{a_1a_2\ldots a_i - a_1a_2\ldots a_{i-1}}$ for $i = 2, 3, \ldots, d$. Cellular consists of five parts:

1. *Calculation* (line 1 in the pseudocode) determining the new alphabet size N using formula (29.1);

2. *Walking* (lines 2–3) if $d = 1$, then construction of a perfect symmetric sequence S_1 using algorithm Martin (walking in a de Bruijn graph);

3. *Meshing* (lines 4–6) if $d = 2$, N is even and $a = 2$, then first construct an N-ary even perfect sequence $\mathbf{e} = \langle e_1, e_2, \ldots, e_{N^2} \rangle$ using Even, then construct an $N^2 \times N^2$ sized N-ary square S_1 using meshing function (**??**);

4. *Shifting* (lines 7–12) if $d > 1$ and (N is odd or $a > 2$), then use Martin once, then use Shift $d - 1$ times, receiving a perfect array P;

5. *Combination* (lines 13–16) if $d > 2$, N is even and $a = 2$, then construct an even sequence with Even, construct a perfect square by Mesh and finally use of Shift $d - 2$ times, results a perfect array P.

```
Cellular($n, d, a, N, A$)

1 $N = N(n, a)$
2 if $d = 1$
3     Martin($N, d, a, A$)
4 return $A$
5 if $d == 2$ and $a == 2$ and $N$ is even
6     Mesh($N, a, A$)
7     return $A$
8 if $N$ is odd or $a \neq 2$
9     Martin($N, a, P_1$)
10    for $i = 1$ to $d - 1$
11        Shift($N, i, P_i, P_{i+1}$)
12        $A = P_1$
13    return $A$
14 Mesh($N, a, P_1$)
15 for $i = 2$ to $d - 1$
16     Shift($N, i, P_i, P_{i+1}$)
17 $A = P_d$
18 return $A$
```

29.5. Three-dimensional perfect arrays

29.5.1. Pseudocode of the algorithm Colour

Colour transforms cellular perfect arrays into larger cellular perfect arrays. Its input data are

- $d \geq 1$ – the number of dimensions;
- $N \geq 2$ – the size of the alphabet;
- $\mathbf{a}$ – the window size;
- $\mathbf{b}$ – the size of the cellular perfect array A;
- A – a cellular $(N, d, \mathbf{a}, \mathbf{b})$-perfect array.
- $k \geq 2$ – the multiplication coefficient of the alphabet;
- $\langle k_1, k_2, \ldots, k_d \rangle$ – the extension vector having the property $k^{|\mathbf{a}|} = k_1 \times k_2 \times \cdots \times k_d$.

  The *output* of Colour is
- a (kN)-ary cellular perfect array P of size $\mathbf{b} = \langle k_1 a_1, k_2 a_2, \ldots, k_d a_d \rangle$.

  Colour consists of three steps:

1. *Blocking:* (line 1) arranging $k^{|\mathbf{a}|}$ copies (blocks) of a cellular perfect array A into a rectangular array R of size $\mathbf{k} = k_1 \times k_2 \times \cdots \times k_d$ and indexing the blocks lexicographically (by $0,\ 1, \ldots, k^{|\mathbf{a}|} - 1$);
2. *Indexing:* (line 2) the construction of a lexicographic indexing scheme I containing the elements $0, 1, \ldots k^{|a|} - 1$ and having the same structure as the array R, then construction of a colouring matrix C, transforming the elements of I into k-ary numbers consisting of $|\mathbf{a}|$ digits;
3. *Colouring:* (lines 3-4) colouring R into a symmetric perfect array P using the colouring array C that is adding the N-fold of the j-th element of C to each cell of the j-th block in R (considering the elements of the cell as lexicographically ordered digits of a number).

The output P consists of blocks, blocks consist of cells and cells consists of elements. If $e = P[\mathbf{j}]$ is an element of P, then the lexicographic index of the block containing e is called the ***blockindex*** of e, the lexicographic index of the cell containing e is called the ***cellindex*** and the lexicographic index of e in the cell is called ***elementindex***. E.g. the element $S_2[7, 6] = 2$ in Table 3 has blockindex 5, cellindex 2 and elementindex 1.

Input parameters are $N,\ d,\ a,\ k,\ \mathbf{k}$, a cellular $(N, d, a, \mathbf{b})$-perfect array A, the output is a $(kN, d, \mathbf{a}, \mathbf{c})$-perfect array P, where $\mathbf{c} = \langle a_1 k_1, a_2 k_2, \ldots, a_d k_d \rangle$.

```
Colour($N, d, \mathbf{a}, k, \mathbf{k}, A, P$)

1 arrange the copies of $P$ into an array $R$ of size
  $k_1 \times k_2 \times \cdots \times k_d$ blocks
2 construct a lexicographic indexing scheme $I$ containing the elements
  of $[0..k^{a^d} - 1]$ and having the same structure as $R$
3 construct an array $C$ transforming the elements of $I$ into $k$-ary
  numbers of $v$ digits and multiplying them by $N$
```

```
4 produce the output $S$ adding the $j$-th ($j \in [0..k^{a^d} - 1]$) element of $C$
  to each cell of the $j$-th block in $R$ for each block of $R$
5 return $S$
```

29.5.2. Pseudocode of the algorithm Growing

Finally, algorithm Growing generates a prefix S_r of a growing array G. Its input data are r, the number of required doubly perfect prefixes of the growing array G, then n, d and $\mathbf{a}$. It consists of the following steps:

1. *Initialization*: construction of a cellular perfect array P using Cellular;
2. *Resizing:* if the result of the initialization is not doubly symmetric, then construction of a symmetric perfect array S_1 using Colour, otherwise we take P as S_1;
3. *Iteration*: construction of the further $r - 1$ prefixes of the growing array G repeatedly, using Colour.

Input parameters of Growing are n, d, a and r, the output is a doubly symmetric perfect array S_r, which is the rth prefix of an $(\mathbf{n}, d, a)$-growing array.

```
Growing($n, d, a, r, S_r$)
01 Cellular($n, d, a, N, P$)
02 calculation of $N$ using formula (29.1)
03 if $P$ is symmetric
04    $S_1 = P$
05 if $P$ is not symmetric
06    $n_1 = N^{d/\gcd(d,a^d)}$
07    $k = n_1/N$
08    $k_1 = (n_1)^{a^d/3}/N^a$
09    for $i = 2$ to $d$
10        $k_i = (n_1)^{a^d/d}/N^{a^i - a^{i-1}}$
11        Colour($n_1, d, a, k, \mathbf{k}, P, S_1$)
12 $k = N^d/\gcd(d, a^d)$
13 for $i = 1$ to $d$
14     $k_i = (n_2)^{a^d/d}/N^{a^i - a^{i-1}}$
15 for $i = 2$ to $r$
16     $n_i = N^{di/\gcd(d,a^d)}$
17     Colour($n_i, d, \mathbf{a}, k, \mathbf{k}, S_{i-1}, S_i$)
18 return $S_r$
```

29.6. Examples of constructing growing arrays using colouring

In this section particular constructions are presented.

column/row	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	1	0	1	1
4	0	1	1	1

column/row	1	2	3	4
1	0	1	2	3
2	4	5	6	7
3	8	9	10	11
4	12	13	14	15

Figure 29.1 a) A (2,2,4,4)-square b) Indexing scheme I of size 4×4

29.6.1. Construction of growing sequences

As the first example let $n = 2$, $a = 2$ and $r = 3$. Cellular calculates $N = 2$ and Martin produces the cellular (2,1,2,4)-perfect sequence $P = 00|11$.

Since P is symmetric, $S_1 = P$. Now Growing chooses multiplication coefficient $k = n_2/n_1 = 2$, extension vector $\mathbf{k} = \langle 4 \rangle$ and uses Colour to construct a 4-ary perfect sequence.

Colour arranges $k_1 = 4$ copies into a 4 blocks sized arrray receiving

$$R = 00|11 \;||\; 00|11 \;||\; 00|11 \;||\; 00|11. \tag{29.4}$$

Colouring receives the indexing scheme $I = 0\ 1\ 2\ 3$, and the colouring matrix C transforming the elements of I into a digit length k-ary numbers: $C = 00 \;||\; 01 \;||\; 10 \;||\; 11$.

Finally we colour the matrix R using C – that is multiply the elements of C by n_1 and adding the j-th ($j = 0, 1, 2, 3$) block of $C_1 = n_1 C$ to both cells of the j-th copy in R:

$$S_2 = 00|11 \;||\; 02|13 \;||\; 20|31 \;||\; 22|33. \tag{29.5}$$

Since $r = 3$, we use Colour again with $k = n_3/n_2 = 2$ and get the (8,1,2,64)-perfect sequence S_3 repeating S_2 4 times, using the same indexing array I and colouring array $C' = 2C$.

Another example is $a = 2$, $n = 3$ and $r = 2$. To guarantee the cellular property now we need a new alphabet size $N = 6$. Martin produces a (6,1,2,36)-perfect sequence S_1, then Colour results a (12,1,2,144)-perfect sequence S_2.

29.6.2. Construction of growing squares

Let $n = a = 2$ and $r = 3$. Then $N(2, 2) = 2$. We construct the even sequence $W_4 = e_1 e_2 e_3 e_4 = 0\ 0\ 1\ 1$ using Even and the symmetric perfect array A in Table 29.1.a using the meshing function (**??**). Since A is symmetric, it can be used as S_1. Now the greatest common divisor of a and a^d is 2, therefore indeed $n_1 = N^{2/2} = 2$.

Growing chooses $k = n_1/N = 2$ and Colour returns the array R repeating the array A $k^2 \times k^2 = 4 \times 4$ times.

Colour uses the indexing scheme I containing k^4 indices in the same 4×4 arrangement as it was used in R. Figure **??**.b shows I.

Transformation of the elements of I into 4-digit k-ary form results the colouring

column/row	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	0	1	1
3	0	1	0	1	0	1	0	1
4	0	0	0	1	1	0	1	1
5	1	0	1	0	1	0	1	0
6	0	0	0	1	1	0	1	1
7	1	1	1	1	1	1	1	1
8	0	0	0	1	1	0	1	1

Figure 29.2 Binary colouring matrix C of size 8×8

matrix C represented in Table 29.2.

Colouring of array R using the colouring array $2C$ results the (4,2,2,16)-square S_2 represented in Table 29.3.

In the next iteration Colour constructs an 8-ary square repeating S_2 4×4 times, using the same indexing scheme I and colouring by $4C$. The result is S_3, a $(8, 2, 2, 64)$-perfect square.

29.6.3. Construction of growing cubes

If $d = 3$, then the necessary condition (2) is $b^3 = (n)^{a^3}$ for double cubes, implying n is a cube number or a is a multiple of 3. Therefore, either $n \geq 8$ and then $b \geq 256$, or $a \geq 3$ and so $b \geq 512$, that is, the smallest possible perfect double cube is the (8, 3, 2, 256)-cube.

As an example, let $n = 2$, $a = 2$ and $r = 2$. Cellular computes $N = 2$, Mesh constructs the $(2, 2, 2, 4)$-perfect square in Table 29.5.a, then Shift uses Martin with $N = 16$ and $a = 1$ to get the shift sizes for the layers of the $(2, 3, 2, \mathbf{b})$-perfect output P of Cellular, where $\mathbf{b} = \langle 4, 4, 16\rangle$. Shift uses P as zeroth layer and the jth ($j \in [1 : 15]$) layer is generated by cyclic shifting of the previous layer downwards by w_i (div 4) and right by w_i (mod 4), where $\mathbf{w} = \langle 0\ 15\ 14\ 13\ 12\ 11\ 10\ 9\ 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1\rangle$. 8 layers of P are shown in Figure 29.4.

Let A_3 be a $4 \times 4 \times 16$ sized perfect, rectangular matrix, whose 0th layer is the matrix represented in Figure 29.5, and the $(2, 3, \mathrm{a}, \mathrm{b})$-perfect array P in Figure 29.4, where $\mathrm{a} = (2, 2, 2)$ and $\mathrm{b} = (4, 4, 8)$.

Growing uses Colour to retrieve a doubly symmetric cube. $n_1 = 8$, thus $b = 256$, $k = n_1/N = 4$ and $\mathbf{k} = \langle 256/4, 256/4, 256/64\rangle$, that is we construct the matrix R repeating P $64 \times 64 \times 16$ times.

I has the size $64 \times 64 \times 16$ and $I[i_1, i_2, i_3] = 64^2(i_1 - 1) + 64(i_2 - 1) + i_3 - 1$. Colour gets the colouring matrix C by transforming the elements of I into 8-digit 4-ary numbers – and arrange the elements into $2 \times 2 \times 2$ sized cubes in lexicographic order – that is in order (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1). Finally colouring results a double cube S_1.

S_1 contains 2^{24} elements therefore it is presented only in electronic form (on the homepage of the corresponding author).

column/row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
3	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
4	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3
5	0	2	0	3	0	2	0	3	0	2	0	3	0	2	0	3
6	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
7	1	2	1	3	1	2	1	3	1	2	1	3	1	2	1	3
8	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3
9	2	0	2	1	2	0	2	1	2	0	2	1	2	0	2	1
10	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
11	3	0	3	1	3	0	3	1	3	0	3	1	3	0	3	1
12	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3
13	2	2	2	3	2	2	2	3	2	2	2	3	2	2	2	3
14	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
15	3	2	3	3	3	2	3	3	3	2	3	3	3	2	3	3
16	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3

Figure 29.3 A (4,2,2,16)-square generated by colouring

Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
0 0 0 1	0 0 0 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1
0 0 1 0	0 0 1 0	0 0 0 1	0 0 0 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1
1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1
0 1 1 1	0 1 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	0 1 1 1

Figure 29.4 8 layers of a (2,3,2,16)-perfect array

If we repeat the colouring again with $k = 2$, then we get a 64-ary $65536 \times 64536 \times 64536$ sized double cube S_2.

29.6.4. Construction of a four-dimensional double hypercube

In 4 dimensions the smallest b's satisfying (**??**) are $b = 16$ and $b = 81$. But we do not know algorithm which can construct $(2, 4, 2, 16)$-perfect or $(3, 4, 2, 81)$-perfect hypercube. The third chance is the $(4, 4, 2, 256)$-perfect hypercube. Let $n = 2$ and $a = 2$. Cellular calculates $N = 2$, then calls Optimal-Martin receiving the cellular $(2, 1, 2, 4)$-perfect sequence $00|11$. Then Cellular calls Mesh which constructs the cellular $(2, 2, 2, 4)$-perfect square shown in Figure 29.5a.

Now Shift calls Optimal-Martin with $n = 1$ and $a = 1$ to get the shift sizes for the layers of the $(2, 3, 2, \mathbf{b})$-perfect output P of Cellular, where $\mathbf{b} = \langle 4, 4, 16\rangle$. Shift uses P as zeroth layer and the jth layer is generated by cyclic shifting of the previous layer downwards by w_i (div 4) and right by w_i (mod 4), where $\mathbf{w} =$

column/row	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	1	0	1	1
4	0	1	1	1

Figure 29.5 A (2,2,4,4)-square

Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
0 0 0 1	0 0 0 1	0 1 0 0	1 1 0 1	1 1 0 1	0 1 0 0	1 1 0 1	0 0 1 0
0 0 1 0	0 0 1 0	0 1 1 1	1 0 1 1	0 1 0 0	1 0 0 0	1 0 1 1	0 1 0 0
1 0 1 1	1 0 1 1	1 1 1 0	1 0 0 0	1 0 0 0	1 1 1 0	1 0 0 0	0 1 1 1
0 1 1 1	0 1 1 1	0 0 1 0	0 0 0 1	1 1 1 0	1 1 0 1	0 0 0 1	1 1 1 0
Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Layer 14	Layer 15
1 0 1 1	0 0 0 1	1 1 1 0	1 1 0 1	1 0 0 0	0 1 0 0	1 0 0 0	0 0 1 0
0 1 1 1	0 0 1 0	0 0 1 0	1 0 1 1	1 1 1 0	1 0 0 0	0 0 0 1	0 1 0 0
0 0 0 1	1 0 1 1	0 1 0 0	1 0 0 0	1 1 0 1	1 1 1 0	1 1 0 1	0 1 1 1
0 0 1 0	0 1 1 1	0 1 1 1	0 0 0 1	0 1 0 0	1 1 0 1	1 0 1 1	1 1 1 0

Figure 29.6 16 layers of the $(2, 3, 2, 16)$-perfect output of Shift

$\langle 0\ 15\ 14\ 13\ 1211\ 10\ 9\ 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1\rangle$. The layers of the $(2, 3, 2, \langle 4\ 4\ 16\rangle)$-perfect array are shown in Figure 29.6.

Up to this point the construction is the same as in [23], but now $d = 4$, therefore we use Shift again to get a $(2, 4, 2, 256)$-perfect prism, then we fill an empty $256 \times 256 \times 256 \times 256$ cube with $4 \times 4 \times 16 \times 256$-sized prisms and finally colouring results the required 4-dimensional hypercube.

Exercises

29.6-1

29.7. The main existence theorem of doubly symmetric perfect arrays

The main result of this chapter can be formulated as follows.

In the talk—using the algorithms Cellular, Optimal-Martin, Even, Mesh, Shift, Colour [23, 33, 34, 35, 37]—we prove the following theorem and illustrate it by the construction of two hypercubes.

Theorem 29.1 *If $n \ge 2$, $d \ge 1$, $a \ge 2$, and $b \ge 2$ satisfy* **??** *and*

a) $d \mid a^d$ and $(un)^{a^d/d} \ge n^{a^d} - a^{d-1}$, then there exists a $(un, d, a, (un)^{a^d/d})$-perfect array;

b) $(vn)^{a^d} \geq n^{a^d - a^{d-1}}$, then there exists a $\left((vn)^d, d, a, (vn)^{a^d}\right)$-perfect array,where u and v are suitable positive integers.

The proof is based on the algorithms Cellular, Optimal-Martin, Even, Mesh, Shift, Colour [23, 33, 34, 35, 37] and on the following two lemmas.

Lemma 29.2 (Cellular lemma) Horváth, Iványi [23, 34] *If $n \geq 2$, $d \geq 1$ and $a \geq 2$, then algorithm* Cellular *produces a cellular $(N, d, a, \mathbf{b})$-perfect array A, where N is determined by formula (29.1), $b_1 = N^a$ and $b_i = N^{a^i - a^{i-1}}$ $(i \in [2..d])$.*

Proof It is known that algorithms Even+Mesh and Martin+Shift result perfect outputs.

Since Mesh is used only for even alphabet size and for 2×2 sized window, the sizes of the constructed array are even numbers and so the output array is cellular.

In the case of Shift we exploit that all prime divisors of a divide the new alphabet size N, and $b_i = N^{(a-1)(a^{i-1})}$ and $(a-1)(a^{i-1}) \geq 1$. ■

Lemma 29.3 (Indexing lemma) Horváth, Iványi [23, 34] *If $n \geq 2$, $d \geq 2$, $k \geq 2$, C is a d dimensional $\mathbf{a}$-cellular array with $|\mathbf{b}| = k^{|\mathbf{a}|}$ cells and each cell of C contains the corresponding cellindex as an $|\mathbf{a}|$ digit k-ary number, then any two elements of C having the same elementindex and different cellindex are heads of different patterns.*

Proof Let P_1 and P_2 be two such patterns and let us suppose they are identical. Let the head of P_1 in the cell have cellindex g and head of P_2 in the cell have cellindex h (both cells are in array C). Let $g - h = u$.

We show that $u = 0 \pmod{k^{|b|}}$. For example in Table 2 let the head of P_1 be $(2, 2)$ and the head of P_2 be $(2, 6)$. Then these heads are in cells with cellindex 0 and 2 so here $u = 2$.

In both cells, let us consider the position containing the values having local value 1 of some number (in our example they are the elements (3,2) and (3,6) of C.) Since these elements are identical, then $k|u$. Then let us consider the positions with local values k (in our example they are (3,1) and (3,5).) Since these elements are also identical so $k^2|u$. We continue this way up to the elements having local value $k^{|b|}$ and get $k^{|b|}|u$, implying $u = 0$.

This contradicts to the conditon that the patterns are in different cells. ■

Lemma 29.4 (Colouring lemma) *If $k \geq 2$, $k_i \in [2..\infty]$ $(i \in [1..d])$, A is a cellular $(n, d, \mathbf{a}, \mathbf{b})$-perfect array, then algorithm* Colour$(N, d, \mathbf{a}, k, \mathbf{k}, A, S)$ *produces a cellular $(kN, d, \mathbf{a}, \mathbf{c})$-perfect array P, where $\mathbf{c} = \langle k_1 a_1, k_2 a_2, \ldots, k_d a_d \rangle$.*

Proof The input array A is N-ary, therefore R is also N-ary. The colouring array C contains the elements of $[0..N(k-1)]$, so elements of P are in $[0..kN - 1]$.

The number of dimensions of S equals to the number of dimensions of P that is, d.

Since A is cellular and c_i is a multiple of b_i $(i \in [1..d])$, P is cellular.

All that has to be shown is that the patterns in P are different.

Let's consider two elements of P as heads of two windows and their contents – patterns p and q. If these heads have different cellindex, then the considered patterns are different due to the periodicity of R. E.g. in Table **??** $P[11, 9]$ has cellindex 8, the pattern headed by $P[9, 11]$ has cellindex 2, therefore they are different (see parity of the elements).

If two heads have identical cellindex but different blockindex, then the indexing lemma can be applied. ■

Proof of the main theorem. Lemma 18 implies that the first call of Colour in line 10 of Growing results a doubly symmetric perfect output S_1. In every iteration step (in lines 14–16 of Growing) the zeroth block of S_i is the same as S_{i-1}, since the zeroth cell of the colouring array is filled up with zeros.

Thus S_1 is transformed into a doubly symmetric perfect output S_r having the required prefixes $S_1,\ S_2,\ \ldots, S_{r-1}$. ■

29.8. d-complexity of one-dimensional arrays

In this section the complexity measure ***d-complexity*** [31, 39] is studied. This measure is intended to express the total quantity of information included in a word counting the different d-subsequences of the investigated word. The background of this complexity measure lies in biology and chemistry. Some natural words have a winding structure [] and some bends can be cut forming new words. The distance parameter d is the bound for the length of bends, which can be cut.

In this paper a new complexity measure, d-complexity is studied. This measure is intended to express the total quantity of information included in a sequence counting the different d-subsequences (belonging to the given set of sequences) of the investigated sequence. The background of the new complexity measure lies in biology and chemistry. Some natural sequences, as amino-acid sequences in proteins or nucleotid sequences in DNA-moleculas have a winding structure [13, 16] and some bends can be cut forming new sequences. The distance parameter d is the bound for the length of bends.

We use the basic concepts and notations of the theory of algorithms [11], formal languages [52] and graphs [5].

29.8.1. Definitions

Let n and q be positive integers, $\mathcal{A} = \{a_1, a_2, \ldots, a_q\}$ be an alphabet, $\mathcal{A}^n$ the set of n-length words, $\mathcal{A}^*$ the set of finite words and $\mathcal{A}^+$ the set of finite nonempty words over $\mathcal{A}$. The length of a word w is denoted by $\lambda(w)$.

Let $m_1,\ m_2,\ \ldots,\ m_q$ be nonnegative integers. Then the multialphabet $\mathcal{M} = (\mathcal{A}, m_1, m_2, \ldots, m_q)$ is the multiset $\{a_1^{m_1}, a_2^{m_2}, \ldots, a_p^{m_q}\}$, $\mathcal{M}^n$ the set of words $w \in \mathcal{A}^n$ and containing $a_i \in \mathcal{A}$ at most m_i times $(i = 1, 2, \ldots, q)$, $\mathcal{M}^+$ the set of words

$w \in \mathcal{M}^n$ containing $a_i \in \mathcal{A}^+$ at most m_i times ($i = 1, 2, \ldots, q$). Alphabets can be considered as special multialphabets with infinite multiplicities of the letters.

Nonempty sets of words — e.g. $\mathcal{A}^n$, $\mathcal{A}^+$, $\mathcal{M}^n$, $\mathcal{M}^+$ — are called languages. Set of i-length elements of a language L is denoted by $L(i)$.

Definition 1. Let d, r and s be positive integers, u and w words such that $u = u_1u_2 \ldots u_r$ and $w = w_1w_2 \ldots w_s$. If $r \geq 2$, then u is a ***d*-subword** of w ($u \subset_d w$), iff there exists a sequence $i_1,\ i_2,\ \ldots,\ i_r$ with $1 \leq i_1, i_r \leq s$, $1 \leq i_{j+1} - i_j \leq d$ ($j = 1, \ldots, r-1$) such that $u_j = w_{i_j}$ ($j = 1, 2, \ldots, r$). If $r = 1$, then u is a d-subword of w, iff there exists a sequence i with $1 \leq i \leq s$ such that $u_1 = w_i$. If for given u, w and d there exist several such sequences (or indeces), then the sequence belonging to u, w and d is the lexicographically minimal of them. The differences $i_{j+1} - i_j$ are called jumps. 1-length words contain a jump of size zero. ■

According to this definition only nonempty words can be d-subwords.

Now we classify the d-subwords of a given word w according to their length, position of their last letter and the length of their longest jump.

Definition 2. Let d and n be positive integers, $\mathcal{M}$ a multialphabet, $L \subseteq \mathcal{M}^+$ a multilanguage, $w = (w_1w_2 \ldots w_n) \in \mathcal{M}^n$ a word, $D(w, L, i, d)$ denote the i-length subwords of w belonging to L, $P(L, w, i, d)$ denote the d-subwords of w for which the first appearance ends with w_i, $U(w, L, i, d)$ the set of subwords of w for which the largest jump in its first occurence in w equals to i (everywhere $0 \leq i \leq n$). The ***d*-complexity function** $C(w, L, i, d)$ of w is

$$C(w, L, i, d) = |D(w, L, i, d)| \quad (i = 1, 2, \ldots, n), \tag{29.6}$$

the ***d*-characteristic function** $H_d(w, L, i, d)$ of w is

$$H(w, L, i, d) = |P(w, L, i, d)| \quad (i = 1, 2, \ldots, n). \tag{29.7}$$

the ***djump function*** $J(w, L, i, d)$ of w is

$$J(w, L, i, d) = |U(w, L, i, d)| \quad (i = 0, 1, \ldots, n-1), \tag{29.8}$$

the ***total d-complexity*** $T(w, L, d)$ of w is

$$T(w, L, d) = \sum_{i=1}^{n} C(w, L, i, d) = \sum_{i=1}^{n} H(w, L, i, d) = \sum_{i=0}^{n-1} J(w, L, i, d). \quad ■ \tag{29.9}$$

The special case $d = 1$, $L = \mathcal{A}^+$ of $C(w, L, i, d)$ was introduced by K. Heinz [22] in 1977 and studied later by many authors (see e.g. the papers [1, 18, 54]. The special case $L = \mathcal{A}^+$ of $C(w, L, i, d)$ was introduced in 1984 [24] and studied later e.g. in [32, 39].

Example 1. Let $\mathcal{A} = \{A, E, L, T\}$ be an alphabet, $L = \mathcal{A}^+$, $w = (ELTE)$. Classification according to the length of the subwords results $D(w, L, 1, 1) = D(w, L, 1, 2) = D(w, L, 1, 3) = \{E, L, T\}$, $D(w, L, 2, 1) = \{EL, LT, TE\}$, $D(w, L, 2, 2) = \{EL, ET, LT, LE, TE\}$, $D(w, L, 2, 3) = \{EL, ET, EE, LT, LE, TE\}$, $D(w, L, 3, 1) = \{ELT,\ LTE\}, D(w, L, 3, 2) = D(w, L, 3, 3) = \{ELT,\ ELE, ETE, LTE\}$, $D(w, L, 4, 1) = D(w, L, 4, 2) =$

$D(w, L, 4, 3) = \{ELTE\}$ and so $T(w, L, 1) = 3 + 3 + 2 + 1 = 9$, $T(w, L, 2) = 3 + 5 + 4 + 1 = 13$, $T(w, L, 3) = 3 + 6 + 4 + 1 = 14$.

Classification according to the last letter gives $P(w, L, 1, 2) = \{E\}$, $P(w, L, 2, 2) = \{EL, L\}$, $P(w, L, 3, 2) = \{ELT, ET, LT, T\}$ and $P(w, L, 4, 2) = \{ELTE, ELE, ETE, LE, LTE, TE\}$ and so $T(w, L, 2) = 1 + 2 + 4 + 6 = 13$.

Classification according to the largest jumps results $U(w, \mathcal{A}^+, 3, 3) = \{EE\}$, $U(w, \mathcal{A}^+, 2, 3) = \{ELE, ET, ETE, LE\}$, $U(w, \mathcal{A}^+, 1, 3) = \{E, EL, ELT, ELTE, L, LT, LTE, T, TE\}$ and so $T(w, L, 3) = 1 + 4 + 9 = 14$.

Let $\mathcal{M} = (\mathcal{A}, 1, 1, \ldots, 1)$, $N = \mathcal{M}^+$. The multisubwords can contain at most one E and so $T(w, N, 3) = 3+5+2+0 < T(w, L, 3) = 14$. Decreasing of some complexity values is natural since bounded multiplicities exclude some subwords. ■

Definition 3. Let L be a finite language over $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and $w \in \mathcal{A}^+$ be a word. w is a **d-covering word** of L iff w contains all elements of L as a d-subword. The length of the shortest d-covering word of L is denoted by $\gamma(L, d)$. Such words are called **minimal**. If $L = \mathcal{A}^n$, then we use $g(q, n, d)$ instead of $\gamma(L, d)$. ■

The special case $d = 1$ is well-known and widely studied as superstring problem [56].

Definition 4. Let d, q and n be positive integers, $\mathcal{A} = \{0, 1, \ldots, q-1\}$ be an alphabet and L be a language over $\mathcal{A}$. The maximum of the d-complexities of the elements of L is denoted by $\mu(L, d)$. Words having complexity $\mu(L, d)$ are called **maximal**. If $L = \mathcal{A}^n$, then we use $f(q, n, d)$ instead of $\mu(L, d)$. Minimal and maximal words are called **extremal**. ■

Definition 5. Let $\mathcal{F}(\mathcal{M}) = \{L_1, L_2, \ldots\}$ be a sequence of languages over some alphabet multialphabet $\mathcal{M}$ with the property $L_i \subseteq \mathcal{M}^i$ $(i = 1, 2, \ldots)$. Such sequences of languages are called **family**. ■

Definition 6. Let d, n and q be positive integers, $\mathcal{A} = \{0, 1, \ldots, q-1\}$ an alphabet, $\mathcal{F} = \{L_1, L_2, \ldots\}$ a family of languages and $w = (x_1 x_2 \ldots)$ be a word over $\mathcal{A}$. w is a **d-superminimal word of the family $\mathcal{F}$** iff the word $(x_1 x_2 \ldots x_{\gamma(L_k, d)})$ is a d-covering word of L_n for $n = 1, 2, \ldots$. ■

Definition 7. Let q be a positive integer, $\mathcal{A} = (0, 1, \ldots, q-1)$ be an alphabet, d be a positive integer and $w = (x_1\ x_2\ \ldots)$ be a word over $\mathcal{A}$. w is **d-supermaximal word** of the alphabet $\mathcal{A}$ iff the word $x_1 x_2 \ldots x_n$ is a d-maximal word for $n = 1, 2, \ldots$. ■

Definition 8. The words consisting of identical letters are called **homogeneous**, the words consisting of different letters are called **rainbow** and the words consisting of copies of a given rainbow word (the last copy can be only a prefix) are called **cyclical**. ■

29.8.2. Bounds

In this section we formulate some simple properties of the investigated complexity measures.

Lemmas 3.1, 3.2, 3.3 and 3.4 characterize the monotonity properties of the functions C and H.

Lemmas 3.5, 3.6, 3.7 and 3.8 give some bounds of the functions C, H and J.

Lemma 29.5 *If d is a positive integer, L is a language and w is a word, then the d-complexity function $C(w, L, i, d)$ is a monotone nondecreasing function of the distance d, that is*

$$C(w, L, i, d+1) \geq C(w, L, i, d) \quad (i = 1, 2, \ldots, \lambda(w)) \tag{29.10}$$

and in some cases the equality holds.

Proof $D(w, L, i, d) \subseteq D(w, L, i, d+1)$ implies the relation. E.g. if $\lambda(w) \geq 3$ and the first, second and last letters of a word w are identical, then $D(w, L, 2, \lambda(w) - 2) = D(w, L, 2, \lambda(w) - 1)$ and generally if $d \geq \lambda(w) - 1$, then in (5) equality holds for any i. ■

Lemma 29.6 *If d and q are positive integers and $\mathcal{A} = \{0, 1, \ldots, q-1\}$, then $C(w, \mathcal{A}^+, i, d)$ is a trapezoidal function of i, that is there exist a positive integer j $(1 \leq j \leq \lambda(w))$ and a nonnegative integer m such that*

$$C(w, \mathcal{A}^+, i, d) < C(w, \mathcal{A}^+, i+1, d) \;\; (i = 1, 2, \ldots, j-1), \tag{29.11}$$

$$C(w, \mathcal{A}^+, j, d) = C(w, \mathcal{A}^+, j+p, d) \;\; (p = 0, 1, \ldots, m) \tag{29.12}$$

and

$$C(w, \mathcal{A}^+, i, d) > C(w, \mathcal{A}^+, i+1, d) \;\; (i = j+m, j+m+1, \ldots, \lambda(w) - 1). \tag{29.13}$$

Proof Let ■

Special case $d = 1$ of this assertion was recently proved by de Luca [Lu]. Since only words belonging to L are counted as d-subwords, the function C is not always trapezoidal. E.g. if $\mathcal{A} = \{0, 1, \ldots, q-1\}$ an alphabet, m is a nonnegative integer, $L = \mathcal{A}^1 \cup \mathcal{A}^3 \cup \ \ldots \cup \mathcal{A}^{2m+1}$ and w a covering word of $\mathcal{A}^{2m+1}$, then the function $C(w, L, i, \infty)$ has k local maxima, since now $C(w, L, i, \infty)$ equals to q^i for $i = 1,\ 3,\ \ldots,\ 2m+1$ and equals zero otherwise.

Lemma 29.7 *If L is a language, then the d-characteristic function $H(w, L, i, d)$ is a nondecreasing function d that is*

$$H(w, L, i, d) \leq H(w, L, i, d+1) \quad (i = 1,\ 2,\ \ldots,\ \lambda(w)) \tag{29.14}$$

and if $L = \mathcal{A}^+$ for some finite alphabet $\mathcal{A}$, then the d-characteristic function H is an increasing function of i that is

$$H(w, \mathcal{A}^+, i, d) \leq H(w, \mathcal{A}^+, i+1, d) \;\; (i = 1,\ 2,\ \ldots,\ \lambda(w) - 1). \tag{29.15}$$

Lemma 29.8 *If d is a positive integer, $\mathcal{A}$ is an alphabet and $L = \mathcal{A}^+$ is a language, then the cumulated values of the functions C and H are increasing functions of i that is*

$$\sum_{j=1}^{i} C(w, \mathcal{A}^+, j, d) < \sum_{j=1}^{i+1} C(w, \mathcal{A}^+, j, d) \;\; (i = 1,\ 2,\ \ldots,\ \lambda(w) - 1), \tag{29.16}$$

$$\sum_{j=1}^{i} H(w, \mathcal{A}^+, j, d) < \sum_{j=1}^{i+1} H(w, \mathcal{A}^+, j, d) \quad (i = 1,\ 2,\ \ldots,\ \lambda(w) - 1). \tag{29.17}$$

Proof The corresponding D and P are nonempty sets. ■

E.g. if $L(\lambda(w))$ is empty, then the functions C and H are only nondecreasing ones.

Lemma 29.9 *Let $\mathcal{A} = \{0, 1, \ldots, q-1\}$, $w \in \mathcal{A}^+$ and $L \subseteq \mathcal{A}^+$, then*

$$0 \le C(w, L, i, d) \le \min \left(L(i), \binom{n}{i} \right) \quad (i = 1, 2, \ldots, n) \tag{29.18}$$

and if $w \in \mathcal{A}^n$, $L = \mathcal{A}^n$ and $d \ge n$, then

$$1 \le C(w, \mathcal{A}^+, i, d) \le \min \left(q^i, \binom{n}{i} \right) \quad (i = 1, 2, \ldots, n). \tag{29.19}$$

The lower bounds are sharp. The upper bound is sharp iff $q \ge n - 1$.

Lemma 29.10 *Let $\mathcal{A} = \{0, 1, \ldots, q-1\}$, $w \in \mathcal{A}^+$ and $L \subseteq \mathcal{A}^+$, then*

$$0 \le H(w, L, i, d) \le 2^{i-1} \quad (i = 1, 2, \ldots, n) \tag{29.20}$$

and if $w \in \mathcal{A}^n$, $L = \mathcal{A}^n$ and $d \ge n$, then

$$1 \le H(w, \mathcal{A}^+, i, d) \le 2^{i-1} \quad (i = 1, 2, \ldots, n). \tag{29.21}$$

The lower bounds are sharp. The upper bound is sharp iff $q \ge n - 1$.

Lemma 29.11 *Let $\mathcal{A} = \{0, 1, \ldots, q-1\}$, $w \in \mathcal{A}^+$ and $L \subseteq \mathcal{A}^+$, then*

$$0 \le C(w, L, i, d) \le \min |(L(i)|, n - i - 1) \quad (i = 1, 2, \ldots, n) \tag{29.22}$$

Lemma 29.12 *If q is a positive integer and $\mathcal{A} = \{0, 1, \ldots, q-1\}$, $w \in \mathcal{A}^n$, then*

$$1 \le C(w, \mathcal{A}^+, i, 1) \le \min(q^i, n - i - 1) \quad (i = 1, 2, \ldots, n). \tag{29.23}$$

The lower bound is sharp if and only if w is homogeneous. The upper bound is sharp.

The first proof of Lemma 29.12 is due to Heinz [22]. Shallit [54] has proved that in the case $q = 2$ the upper bound can be achieved. M.-C. Anisiu [1] proposed an algorithm constructing for any positive n an n-length binary word w with the following property: if $2^k + k - 1 \le n \le 2^{k+1} + k$, then w either contains all k-length binary words as subword or all of its $(k+1)$-length subwords are different.

The jump function characterizes the decreasing of the complexity due to the decreasing of d.

Lemma 29.13 *Let q, n and d be positive integers, $\mathcal{A} = \{0, 1, \ldots, q-1\}$, $w \in \mathcal{A}^n$, $L = \mathcal{A}^+$. Then*

$$J(w, L, i, \infty) \begin{cases} \leq (n-i)2^{n-i-1} & \textit{if } n/2 \leq i \leq n-1, \\ \leq (n-i)2^{n-i-1} - 1, & \textit{if } n/3 \leq i < n/2 \textit{ and } n = 2i+1 \\ \leq (n-i)2^{n-i-1} - 7, & \textit{if } n/3 \leq i \leq n/2 \textit{ and } n = 2i+2, \\ \leq (n-i)2^{n-i-1} - 14, & \textit{if } n/3 \leq i \leq n/2 \textit{ and } n = 2i+3. \end{cases} \tag{29.24}$$

The bounds are sharp iff w is a rainbow word.

Proof If $n/2 \leq i \leq n-1$, then at most one jump of length i is possible. We have $n-i$ positions to start the jump and can choose or not the remaining $(n-i-1)$ letters.

If $n = 2i+1$, then we have either two jumps of length $i(w_1 - w_i, w_i - w_{2i+1})$ or one $(w_j - w_{j+i})$. Due to the second jump we loss a further subword comparing with the previous case.

If $n = 2i+2$, then we have place also for one or two i-length jumps. The exponential part of the formula counts 2 subwords containing jumps of length i and $i+1$ and counts twice 5 subwords containing two jumps of length i so we loss 7 subwords.

If $n = 2i+3$, then also one or two jumps of length at most i are permitted. We loss 2 subwords containing jumps of length i and $i+2$, 6 subwords containing jumps of length i and $i+1$ and all together 6 subwords containing two jumps both of length i. ■

29.8.3. Recurrence relations

The following recurrence properties of the complexity functions H and J are useful in the later analysis. In this section always $A = \{0, 1, \ldots, q-1\}$, $L = \mathcal{A}^n$, $w = w_1 w_2 \ldots w_n$ and $d \geq 1$, therefore the parameters w, L and d are omitted.

Lemma 29.14 *If $1 \leq i \leq n$, then*

$$H(i) \geq \delta(i) + \sum_{j=z}^{i-1} H(j), \tag{29.25}$$

where $z = \min(0, i-d, k)$, $k = \max(0,\ p | w_p = w_i$ and $1 \leq p \leq i-1)$ and

$$\delta(i) = \begin{cases} 1, & \textit{if} \quad \exists j \quad (1 \leq j < i) \textit{ with } w_j = w_i \\ 0, & \textit{if} \quad \nexists j \quad (1 \leq j < i) \textit{ with } \quad w_j = w_i. \end{cases} \tag{29.26}$$

Corollary 29.15 *If $w = (0\ 1 \ldots\ )$ is a crossbow word and $i \geq 1$, then*

$$H(i) = \begin{cases} 1 + \sum_{j=1}^{i-1} H(j), & \textit{if } i > d, \\ i, & \textit{if } 1 \leq i \leq d. \end{cases} \tag{29.27}$$

Corollary 29.16 *If $w = (0\ 1 \ldots\ q-1)^*$ is an infinite cyclical word and $d \geq q$, then*

$$H(i) = \begin{cases} \sum_{j=1}^{i-1} H(j), & \textit{if } q > d, \\ 2^i, & \textit{if } 1 \leq i \leq d. \end{cases} \tag{29.28}$$

The next three lemmas contain useful results of the theory of linear recurrence equations. The first one [38] gives the solution of the d-order inhomogeneous recurrence relation, the second one [31] characterizes the roots of the coresponding characteristic equations and the third one [**?**] sums the values of the solution function.

Lemma 29.17 *If d is a positive integer, then the solution of the inhomogeneous linear recurrence equation*

$$f(n) = \begin{cases} a + \sum_{i=1}^{d} f(n-d), & \textit{if } n > d, \\ b(n), & \textit{if } 1 \leq n \leq d \end{cases} \tag{29.29}$$

is

$$f(n) = k_1 n + k_2 + \sum_{i=1}^{d} c_{i,d} r_{i,d}^i, \tag{29.30}$$

where the constants $k_1, k_2, c_{1,d}, c_{2,d}, \ldots, c_{d,d}$ can be computed using the given initial values $b(n)$ and the values $r_{i,d}$ can be estimated using the next lemma.

Lemma 29.18 *If $d \geq 2$, then let $r_{1,d},\ r_{2,d},\ \ldots, r_{d,d}$ be the roots of the d-order algebraic equation*

$$z^d - z^{d-1} - \ldots - z - 1 = 0 \tag{29.31}$$

in decreaising order of their absolute values. Then

$$2 - \frac{1}{2^{d-1}} < r_{1,d} < \frac{1}{2^d} \tag{29.32}$$

and

$$|r_{i,d}| < 1 \quad (i = 2,\ 3,\ \ldots,\ d). \tag{29.33}$$

Lemma 29.19 *If*

$$f(n) = k_1 + k_2 n + \sum_{i=1}^{d} c_{i,d} r_{i,d}^i, \tag{29.34}$$

then

$$\sum_{i=1}^{n} f(i) = \frac{c_{1,d} r_{1,d}}{r_{1,d} - 1} r_{1,d}^n + k_1 n + \frac{k_2 n(n+1)}{2} + \sum_{i=1}^{d} \frac{c_{i,d} r_{i,d}}{1 - r_{i,d}} + \sum_{j=2}^{d} \frac{c_{j,d} r_{j,d}}{r_{j,d} - 1} r_{j,d}^n = \Theta(r_{1,d}^n) + O(n^2). \tag{29.35}$$

29.8.4. Algorithms

The algorithms are defined using the pseudocode of [11].

At first we present an algoritm d-COMPLEXITY estimating the d-complexity of words.

Definition of algorithm d-COMPLEXITY

Input: q the size of the alphabet,
d the distance parameter,
n the length of the investigated word,
$w[1], w[2], \ldots, w[n]$ the letters of the investigated word.

Output: $I[n]$ the d-complexity of the investigated word.

Working variables: $U[0],\ U[1],\ \ldots,\ U[q-1]$ position of the last occurence of letters $0,\ 1,\ \ldots,\ q-1$.
z starting index of the summing.

```
01 for $i \leftarrow 0$ to $q-1$

02     $U[i] \leftarrow 0$
03 $U[w[1]] \leftarrow 1$
04 $I[0] \leftarrow 0$
05 for $i \leftarrow 1$ to $n$
06     $H[i] \leftarrow 0$
07     $z \leftarrow \max(0, U[w[i]],\ i+1-d)$
08     for $j \leftarrow z$ to $i$
09         $H[i] \leftarrow H[i] + H[j]$
10         if $U[w[i] = 0$ then $H[i] \leftarrow H[i] + 1$
11     $I[i] \leftarrow I[i-1] + H[i]$
```

The d-complexity of the investigated word is greater or equal to the output of this algorithm. If all letters of w are different or if w is a 1-subword of the word $(0\ 1\ \ldots\ q-1)^*$ or $d \geq n-1$, then the output $I[n]$ of the algorithm is the exact complexity of w.

The running time of this algorithm is $O(qn)$.

Definition of the algorithm MARTIN

Input: q the size of the alphabet,
n the length of the words to be covered.

Output: $w_1,\ w_2, \ldots\ w_{q^n+n-1}$: the letters of the Martin word $M(q,n)$.

Working variables: $C[0], C[1], \ldots, C[q^n-1]$ counters of the vertices of de Bruijn graph.
k the index of the actual vertex of de Bruijn graph.

```
01 for $i \leftarrow 0$ to $q^n - 1$
02     $C[i] \leftarrow 0$
03 for $i \leftarrow 1$ to $n-1$
04     $w[i] \leftarrow 0$
05 for $i \leftarrow n$ to $q^n + n - 1$
06     $k \leftarrow 0$
```

```
07        for $j \leftarrow 1$ to $n$
08             $k \leftarrow kn + w[i-n+j]$
09        $w[i] \leftarrow n - 1 - C[k]$
10        $C[k] \leftarrow C[k] + 1$
```

The running time of $MARTIN$ is $\Theta(nq^n)$, memory requirement is $\Theta(q^n)$.

Definition of algorithm Super
Input: q the size of the alphabet,
n the length of the word to be generated.

Definition of algorithm MaxSub
Input: n the length of the word to be generated.
Output: $m_1,\ m_2, \dots\ m_n$: the letters of a maximal word $M(q,n)$.

Working variables: $C[0], C[1], \dots, C[q^n - 1]$ counters of the vertices of de Bruijn graph $B(q,n)$.

```
01 $k \leftarrow \lfloor \log_q n \rfloor$
02 if $n \le q^k + k - 1$ then $k \leftarrow k - 1$
03 CALL $MARTIN(q,k)$    $q$ ▷    $MARTIN$ produces a Martin-word
   $M(q,k) = m_1 m_2 m_{q^k+k-1}$
04 if $n = q^k + k - 1$ then RETURN
05 Add 0 to the Martin-word and and according to the new word draw a path in
   the de Bruijn graph $B(q,k+1)$
06 Construct cycles in $B(q,k+1)$ using the remaining edges following Martin
   principle
07 Order these cycles in decreasing order of length. Let $|C_1| \ge |C_2| \ge \dots |C_m|$.
08 Transform $M(q,k)$ at first into a cyclical de Bruijn word, then back into a linear
   de Bruijn word $L$ ending at the starting vertex of $C_1$.
09 Add $C_1$ to $L$ receiving the letters $m_{q^k+k+2} m_{q^k+k+3} \dots m_{q^k+k+1+|C_1|}$
10 if the length of the new word is greater or equal to $n$, then RETURN
11 for $i \leftarrow 2$ to $m$
12      Insert $C_i$ in the constructed word
13      if the length of the new word is greater or equal to $n$, thenthen RETURN
```

29.8.5. Construction and complexity of extremal words

Due to the finiteness of the corresponding sets d-extremal words of languages exist in all cases. Similar assertion does not hold for maximal and minimal words of families of languages.

Theorem 6.1 and 6.3 contain general bounds of d-complexity, resp. 1-complexity. Theorem 6.4 characterizes the maximal d-complexity for medium distance and implies the surprising corollary that binary alphabet is sufficient to get the same order of maximal subword complexity what can be achieved using an infinite alphabet.

Theorem 6.6 and 6.7 give the maximal complexity for different values of the parameters. Theorem 6.8 contains the condition of the existence of 1-superminimal words. Theorem 6.9 gives the complexity of cyclical words and together with Theorem 6.6 has the corollary that $f(q,n,d)$ and $f(d,n,q)$ have the same order of growth. Theorem 6.10 gives the that the n-length words can be covered using only letters needed to cover the words $(0^n)\ (1)^n\ \ldots\ (q-1)^n)$ and so illustrates the decreasing of the length of minimal d-covering words due to the permitted jumps in the construction.

Theorem 29.20 *If d, n and q are positive integers, $\mathcal{A} = \{0, 1, \ldots, q-1\}$ is an alphabet and $w \in \mathcal{A}^n$, then*

$$n \leq T(w, \mathcal{A}^+, d) \leq 2^n - 1. \tag{29.36}$$

The lower bound is tight iff w is homogeneous. The upper bound is sharp iff w is a rainbow word and $d \geq n-1$.

Proof Since $w_1 w_2 \ldots w_i \in D(w, \mathcal{A}^+, i, d)$ holds for $i = 1, 2, \ldots, n$, the lower bound is correct and for $w = 0^n$ the equality holds. The upper bound can be proved using equation 29.19 and inequality $\min(q^i, \binom{n}{i}) \leq \binom{n}{2}$ or directly exploiting that an n-element set has $2^n - 1$ nonempty subsets. If $q \geq n$ and $d \geq n-1$, then $w = 0\ 1\ \ldots\ n-1$ belongs to $\mathcal{A}^+$ and contains $2^n - 1$ different elements of $\mathcal{A}^n$ as a d-subword so the upper bound is also tight. From the other side if $n < k$, then w contains repeated letters and so $C(w, \mathcal{A}^+, 1, d) < n$, and if $d < n-1$, then the first and last letters of w do not form d-subsequence and so $C(w, \mathcal{A}^+, 2, d) < \binom{n}{2}$. ■

Corollary 29.21 *If d, n and q are positive integers, then*

$$f(1, n, d) = n \tag{29.37}$$

and if $d \geq n-1$, then

$$f(n, n, d) = f(n, n, n-1) = 2^n - 1. \tag{29.38}$$

Theorem 29.22 *If q and n are positive integers, $\mathcal{A} = \{0, 1, \ldots, q-1\}$ is an alphabet and $w \in \mathcal{A}^{\lambda(w)}$, then*

$$n \leq T(w, \mathcal{A}^+, 1) \leq \frac{n(n+1)}{2}. \tag{29.39}$$

The lower bound is sharp if w is homogeneous. The upper bound is sharp iff w is a rainbow word.

Proof We have $n+1$ places for the 2 delimiters determining a 1-subword of w. If $q \geq n$, then all 1-subwords of the word $u = 0\ 1\ \ldots\ \lambda(w) - 1$ are different. From the other side if $q < n$, then w has to contain repeated letters not representing different 1-subwords. ■

Theorem 29.23 *If n and q are positive integers and $1 < q < n-1$, then*

$$f(q,n,1) = \frac{n(n+1)}{2} - nk + \frac{k^2-k}{2} + \frac{q^{k+1}-q}{q-1}, \tag{29.40}$$

where k is the unique integer satisfying $q^k + k - 1 \le n < q^{k+1} + k$ and so $k - 1 < \log_q n < k+1$.

Proof Summing the inequalities 29.23 for $i = 1, 2, \ldots, n$ results

$$T(w, \mathcal{A}^+, 1) \le \sum_{i=1}^{n} \min(q^i, n-i+1). \tag{29.41}$$

In the sum for small values of i the exponential part, for large values of i the second part gives the minimum. Therefore concrete calculations result the formula of the theorem as a bound. Algorithm *MaxSub* produces a suitable word for any possible pair of $q - n$ achieving this bound. ■

In the binary case Shallit [54] has proved the existence of words with such complexity. Using alphabet containing $q \ge 3$ letters and algorithm *SUPER* we constructed [31] infinite words $w = w_1w_2\ldots$ with $T(w_1w_2\ldots w_n) = f(q,n,1)$.

Corollary 29.24 *If $q \ge 2$, then*

$$f(\infty,n,1) = \Theta(f(q,n,1)) = \frac{n(n+1)}{2} + n\lg n + O(qn). \tag{29.42}$$

Theorem 29.25 *If d, q and n are positive integers, $1 < d < n-1$ and $q \ge n$, then*

$$f(q,n,d) = k_1(q,d)n + k_2(q,d) + \sum_{i=1}^{d} c_{i,d} r_{i,d}^n. \tag{29.43}$$

If $d \le 4$, then computation of the constants in Theorem **??** requires only technical steps, otherwise there are algebraic difficulties. Z. Kása [?] has found another recurrence relation and gave explicite formulas for $d \ge n/2$. Using the jump function we reprove his result and extend it for some smaller values of d.

Theorem 29.26 *If d, q and n are positive integers and $q \ge n-1$, then*

$$f(q,n,d) = \begin{cases} c_{1,2}r_{1,2}^n + c_{2,2}r_{2,2}^n - n - 3, & \textit{if } d = 2 \\ c_{1,3}r_{1,3}^n + c_{2,3}r_{2,2}^n + c_{3,3}r_{3,3}^n - n/2 - 3/2, & \textit{if } d = 3 \\ c_{1,4}r_{1,4}^n + c_{2,4}r_{2,4}^n + c_{3,4}r_{3,4}^n + c_{4,4}r_{1,4}^n - n/3 - 1, & \textit{if } d = 4, \\ 2^n - (n-2-d)2^{n-1-d} - 2 & \textit{if } \frac{5}{2} \le \frac{n}{2} \le d \le n-2, \\ 2^n - (n-2-d)2^{n-d-1} - 1, & \textit{if } 3 \le n = 2d+1, \\ 2^n - (n-2-d)2^{n-d-1} + 5, & \textit{if } 4 \le n = 2d+2, \\ 2^n - (n-2-d)2^{n-d-1} + 14, & \textit{if } 5 \le n = 2d+3. \end{cases}$$

where $c_{1,2} = (3/2 + 0,7\sqrt{5})$, $c_{2,2} = (3/2 - 0,7\sqrt{5})$, $r_{1,2} = (1+\sqrt{5})/2$, $r_{1,2} =$

n/i	0	1	2	3	4	5	6	7	8	9	2^n-1
1	1										1
2	2	1									3
3	3	3	1								7
4	4	6	4	1							15
5	5	10	12	4	1						31
6	6	15	25	12	4	1					63
7	7	21	**51**	31	12	4	1				127
8	8	28	**97**	73	32	12	4	1			255
9	9	36	**176**	185	79	32	12	4	1		511
10	10	45	**??**	**??**	191	90	32	12	4	1	1023

Figure 29.7 Values of jumping function of rainbow words of length 1, 2, ..., 10.

$(1+\sqrt{5})/2$ *and so*

$$f(n-1,n,2) \approx 3,065247.1,618034^n - 0,065247(-0,618034)^n - n - 3,$$
$$f(n-1,n,3) \approx 1,614776.1,839287^n - n/2 - 3/2 +$$
$$0,737353^n.[0,061034\cos(2,176234(n+1)) - 0,052411\sin(2,176234(n+1))].$$

Proof The formulas for $d=2$ and $d=3$ appeared in [?], the case $d=4$ is a special case of Theorem 29.25.

If $n/2 \le d \le n-2$, then according to Lemma 29.13 the sum of the jumping values $J(w,L,i,\infty)$ of a rainbow word w for $i=n-1,\ n-2,\ \dots,\ d+1$ results our formula.

The last 3 formulas also can be derived using Lemma 29.13. ■

Table 1 shows the values of the jump function for rainbow words. The first and second columns are computed using the formulas $J(1)=n$ and $J(2)=n(n-1)/2$, the last columns using the formulas for $f(q,n,d)$ and the medium values (printed by bold digits) using the recurrence relation. These data support the conjecture that the jump function is trapezoidal.

Theorem 29.27 *If q is a positive integer, $\mathcal{A}=\{0,1,\dots,q-1\}$ is an alphabet, $L_n=\mathcal{A}^n$ for $n=1,2,\dots$ and $\mathcal{F}=\{L_1,L_2,\dots\}$, then $\mathcal{F}$ has a 1-superminimal word if and only if $n\ne 2$.*

The next theorem shows that the maximal d-complexity is exponential even over a binary alphabet and $d=2$.

Theorem 29.28 *Let $q\ge 2$ and $d\ge 2$ be positive integers and $w=w_1w_2\dots=0\ 1\ \dots q-1\ 0\ 1\dots q-1\dots$ be an infinite cyclical word over the alphabet $\mathcal{A}=\{0,1,\dots,q-1\}$ and $p_n=w_1w_2\dots$. Then*

$$T(p_n,\mathcal{A}^+,d)=\Theta(r_{1,d}^n)+O(n), \tag{29.44}$$

If $q = 2$, then

$$T(p_n, \mathcal{A}^+, d) = \frac{5+2\sqrt{5}}{2}\left(\frac{1+\sqrt{5}}{2}\right)^n + \frac{5-2\sqrt{5}}{2}\left(\frac{1-\sqrt{5}}{2}\right)^n - 2, \qquad (29.45)$$

and so

$$T(p_n, \mathcal{A}^+, 2) = \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right). \qquad (29.46)$$

Proof In this case for the d-characteristic function we get a Fibonacci-type recursion $H(w, \mathcal{A}^+,\ i+2, d) = H(w, \mathcal{A}^+, i, d) + H(w, \mathcal{A}^+, i+1, d)$ with initial values $H(w, \mathcal{A}^+, 1, d) = 1$ and $H(w, \mathcal{A}^+, 2, d) = 2$. The sum of the corresponding characteristic values gives the total complexity. ■

Corollary 29.29 *If q, n and d are positive integers, $q \geq 2$ and $d \geq 2$, then*

$$\Theta(f(q,n,d)) = \Theta(f(d,n,q)) = \Theta(r_{1,d}^n). \qquad (29.47)$$

Theorem 29.30 *If q, n and d are positive integers, $d \geq q$, $\mathcal{A} = \{0, 1, , \ldots, q-1)$ and $L = \mathcal{A}^n$, then*

$$g(q,n,d) = qn \qquad (29.48)$$

and $w = (0\ 1\ \ldots\ q-1)^n$ is a minimal q-covering word of L.

Corollary 29.31 *If n and q are positive integers, $\mathcal{A} = \{0, 1, \ldots, q-1\}$, $\mathcal{M} = (\mathcal{A}, n, n, \ldots, n)$ and $L = \mathcal{M}^n$, then $\gamma(L, n) = qn$ and $w = (1\ 2\ \ldots\ q)^n$ is a minimal q-covering word of L.*

Exercises

29.8-1

29.9. Finite two-dimensional arrays with maximal complexity

The section starts with definitions, then bounds and several further properties of the complexity functions are given.

29.9.1. Definitions

Let $q \geq 2$ be a positive integer and $X = \{0, 1, \ldots, q-1\}$ an alphabet. Let $X^{M\times N}$ denote the set of q-ary $M \times N$ arrays (M, $N \geq 1$ positive integers), and $X^{**} = \cup_{M,N\geq 1} X^{M\times N}$ the set of finite q-ary two-dimensional arrays.

Let m and n be positive integers with $1 \leq m \leq M$ and $1 \leq n \leq N$. A q-ary $m \times n$ array $\mathcal{B} = [b_{ij}]_{m\times n}$ is a subarray of the q-ary $M \times N$ array $\mathcal{A} = [a_{kl}]_{M\times N}$

if there exist indices r, s such that $r+m-1 \leq M$, $s+n-1 \leq N$ and $\mathcal{B} = [a_{uv}]_{r \leq u \leq r+m-1, s \leq v \leq s+n-1}$. $\square$

According to this definition only nonempty arrays can be (m, n)-subarrays.

We remark that we are dealing with *aperiodic* q-ary $M \times N$ arrays (written on a planar surface, with all the subarrays situated completely within the borders of the array). Another point of view is to consider the given array wrapped round on itself (written on a torus), hence a *periodic* array. Existence results for periodic and aperiodic arrays which contain every rectangular subarray of given sizes precisely once are given by Paterson [46], respectively Mitchell [45].

Notions of complexity similar to those for words can be introduced for arrays.

Let $\mathcal{A} \in X^{M \times N}$ be a q-ary array and m, n positive integers with $1 \leq m \leq M$ and $1 \leq n \leq N$. Let $D_{\mathcal{A}}(m,n)$ denote the set of different $m \times n$ subarrays of $\mathcal{A}$. The subarray complexity function, or, simply, the *complexity function* $C_{\mathcal{A}} : \{1, 2, \ldots, M\} \times \{1, 2, \ldots, N\} \to \mathbb{N}$ of $\mathcal{A}$ is

$$C_{\mathcal{A}}(m,n) = |D_{\mathcal{A}}(m,n)|, \quad m = 1, 2, \ldots, M, \ n = 1, 2, \ldots, N, \tag{29.49}$$

and the total complexity function $T_{\mathcal{A}}$ of $\mathcal{A}$ is

$$T_{\mathcal{A}} = \sum_{m=1}^{M} \sum_{n=1}^{N} C_{\mathcal{A}}(m,n). \quad \square \tag{29.50}$$

The one-dimensional complexity and total complexity functions were introduced by M. Heinz [22] in 1977, and studied later by many authors (see e.g. recent papers [1, ?, 6, 3, 8, 18, 14, 54]).

Example 29.1 Let $X = \{0, 1, 2, 3, 4, 5\}$ be an alphabet and

$$\mathcal{A}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathcal{A}_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad \mathcal{A}_3 = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix}.$$

Then $T_{\mathcal{A}_1} = 6$, $T_{\mathcal{A}_2} = 15$, and $T_{\mathcal{A}_3} = 18$. $\square$

The q-ary $M \times N$ array $\mathcal{A}$ is (q, m, n)-**extremal** if

$$C_{\mathcal{A}}(m,n) = \max_{\mathcal{B} \in X^{M \times N}} C_{\mathcal{B}}(m,n). \quad \square \tag{29.51}$$

The q-ary $M \times N$ array $\mathcal{A}$ is (q, m, n)-**perfect** if it contains each of the q^{mn} possible $m \times n$ q-ary arrays as a subarray exactly once. $\square$

The arrays consisting of identical letters are called **homogeneous** arrays, the arrays consisting of different letters are called **rainbow** arrays. $\square$

We mention that a q-ary $M \times N$ rainbow array exists if and only if $q \geq MN$. It is obvious that (q, m, n)-extremal arrays always exist in $X^{M \times N}$ for arbitrary values of M, N, while (q, m, n)-perfect arrays can exist only for M, N satisfying $q^{mn} = (M-m+1)(N-n+1)$.

The function $H_{q,M,N} : \{1, 2, \ldots, M\} \times \{1, 2, \ldots, N\} \to \mathbb{N}$ given by

$$H_{q,M,N}(m,n) = \min\{q^{mn}, (M-m+1)(N-n+1)\} \tag{29.52}$$

is called **maximal complexity function**. $\square$

The q-ary $M \times N$ array $\mathcal{A}$ **is** (q,m,n)**-maximal** if

$$C_{\mathcal{A}}(m,n) = H_{q,M,N}(m,n); \tag{29.53}$$

it is **maximal** if (29.53) holds for all $m = 1,2,\ldots,M$, $n = 1,2,\ldots,N$. $\square$

29.9.2. Bounds

We present the natural bounds of the complexity function for q-ary arrays $\mathcal{A} \in X^{M\times N}$, as well as those of the total complexity function.

Claim 29.32 *For each q-ary $M \times N$ array $\mathcal{A}$ we have*

$$\begin{aligned} &1 \le C_{\mathcal{A}}(m,n) \le \min\{q^{mn}, (M-m+1)(N-n+1)\}, \\ &m = 1,2,\ldots,M, \ n = 1,2,\ldots,N. \end{aligned} \tag{29.54}$$

The lower bound is sharp for homogeneous arrays and the upper bound is sharp for rainbow arrays. The total complexity of $\mathcal{A}$ satisfies the inequality

$$MN \le T_{\mathcal{A}} \le \sum_{i=1}^{M}\sum_{j=1}^{N} H_{q,M,N}(i,j). \tag{29.55}$$

Proof From the definition of the subarray it follows that $C_{\mathcal{A}}(m,n) \ge 1$, $m = 1,2,\ldots,M$, $n = 1,2,\ldots,N$; for a homogeneous array the equality holds.

It is obvious that the complexity $C_{\mathcal{A}}(m,n)$ cannot exceed the total number of subarrays over X, that is q^{mn}; it also cannot exceed the total number of subarrays of dimension $m \times n$ of the given array (possible not all different), namely $(M-m+1)(N-n+1)$. It follows that $1 \le C_{\mathcal{A}}(m,n) \le \min\{q^{mn}, (M-m+1)(N-n+1)\}$, $m = 1,2,\ldots,M$, $n = 1,2,\ldots,N$. For a rainbow array $\mathcal{R}$ we have $C_{\mathcal{R}}(m,n) = (M-m+1)(N-n+1) = \min\{q^{mn}, (M-m+1)(N-n+1)\}$.

By summing up the inequalities (29.54) we obtain (29.55). ■

Remark 29.33 *In terms of the maximal complexity functions, inequality (29.54) may be reformulated as*

$$1 \le C_{\mathcal{A}}(m,n) \le H_{q,M,N}(m,n), \qquad m = 1,2,\ldots,M, \ n = 1,2,\ldots,N.$$

It follows that every (q,m,n)- perfect array, as well as any rainbow array, is (q,m,n)- maximal.

The values of the complexity and total complexity for homogeneous and rainbow arrays can be easily computed.

Claim 29.34 *If $\mathcal{H}$ is a homogeneous $M \times N$ array and $\mathcal{R}$ is an $M \times N$ rainbow* *array, then*

$$\begin{array}{l} C_{\mathcal{H}}(m,n) = 1, \quad C_{\mathcal{R}}(m,n) = (M-m+1)(N-n+1), \\ m = 1,2,\ldots,M, \;\; n = 1,2,\ldots,N \end{array}$$

and

$$T_{\mathcal{H}} = MN, \quad T_{\mathcal{R}} = \frac{M(M+1)N(N+1)}{4}.$$

Proof The complexity functions $C_{\mathcal{H}}$ and $C_{\mathcal{R}}$ were given in the proof of Proposition 29.32. Easy calculations give the formulas for $T_{\mathcal{H}}$ and $T_{\mathcal{R}}$. ■

The shape of the complexity function for words was proved in [14, 40, 36, 3] to be *trapezoidal*, i. e. it has an ascending part, possibly a horizontal one, and the last part is a descending line . The main feature is that after becoming constant, the complexity function of an arbitrary word cannot increase again. The question for arrays is: for a fixed m_0, is $C_{\mathcal{A}}(m_0,\cdot)$ still trapezoidal? For $m_0 = 1$, the answer is positive, as a consequence of the mentioned result for words; nevertheless, this is not true for all the values $m_0 = 1,2,\ldots,M$. The array $\mathcal{A}$ in the following example has the property that $C_{\mathcal{A}}(2,\cdot)$ increases again after becoming a constant.

Example 29.2 For the array $\mathcal{A} \in \{0,1\}^{3\times 19}$ given by

$$\mathcal{A} = \left(\begin{array}{ccccccccccccccccccc} 1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1 \\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1&0&1&1&0&0&1&0&1&0&0&0&0 \end{array} \right)$$

one has $C_{\mathcal{A}}(2,4) = C_{\mathcal{A}}(2,5) = 21$, $C_{\mathcal{A}}(2,6) = C_{\mathcal{A}}(2,7) = 22$ and $C_{\mathcal{A}}(2,8) = 21$.

29.9.3. Properties of the maximal complexity function

We shall describe some properties of the function $H_{q,M,N}$ related to the shape of its graph, namely its monotonicity and its maximum.

For $M = 1$ (or $N = 1$) the arrays are in fact finite sequences (words). It vas shown in [2, 3, 14] that for a given N we have

$$H_{q,1,N}(1,n) = \begin{cases} q^n, \;\; n \le k \\ N-n+1, \;\; k+1 \le n \le N, \end{cases}$$

where k is the only natural number for which $q^k + k \le N < q^{k+1} + k + 1$. The maximum of $H_{q,1,N}$ is equal to $N - k$ and is attained at the unique point $k+1$ for $q^k + k < N \le q^{k+1} + k + 1$, and at both k and $k+1$ for $N = q^k + k$, hence $H_{q,1,N}$ is trapezoidal.

In the remaining part of this section we shall consider proper arrays (with $M, N \ge 2$).

Remark 29.35 *If both sizes of the array are smaller than the cardinal of the alphabet X ($M, N \le q$), we have*

$$(M-m+1)\,(N-n+1) \le q^2 \le q^{mn} \textit{ for } mn \ne 1,$$

hence

$$H_{q,M,N}(m,n) = \begin{cases} \min\{q, MN\}, & m = n = 1 \\ (M-m+1)(N-n+1), & \textit{otherwise.} \end{cases}$$

The maximum will be given by

$$H_{\max} = \max\{\min\{q, MN\}, N(M-1), M(N-1)\}$$

and will be attained at one of the points $(1,1)$, $(1,2)$ *or* $(2,1)$. *If* $q < MN$, *we have* $H_{\max} = \max\{q, N(M-1), M(N-1)\}$; *if* $q \geq MN$, $H_{\max} = MN = h(1,1)$.

In what follows we shall consider $\max\{M,N\} > q$.

Claim 29.36 *Let* $m_0 \in \{1,...,M\}$ *be fixed; the function* $H_{q,M,N}(m_0,\cdot)$ *is trapezoidal, the horizontal part containing at most two points; the last part is a descending line and the maximum of* $H_{q,M,N}(m_0,\cdot)$ *is attained at the first point* d_{m_0} *situated on the descending line, or on* $d_{m_0} - 1$.

Proof The values of $H_{q,M,N}(m_0,n), n \in \{1,...,N\}$ are given by the minimum of the values of an increasing exponential and of a descending line. At the beginning, if $(M-m_0+1)N > q^{m_0}$, $H_{q,M,N}(m_0,\cdot)$ will be situated on the exponential, and surely it will end on the descending line. Therefore $H_{q,M,N}(m_0,\cdot)$ will have a trapezoidal shape, with a horizontal part with at most two points.

There will be a point $d_{m_0} \leq N$ which is the least value of n for which $H_{q,M,N}(m_0,n)$ is on the descending line, i.e. if $d_{m_0} > 1$

$$\begin{aligned} (M-m_0+1)(N-d_{m_0}+1) &\leq q^{m_0 d_{m_0}} \\ (M-m_0+1)(N-d_{m_0}+2) &\geq q^{m_0(d_{m_0}-1)}. \end{aligned}$$

The maximal value of $H_{q,M,N}(m_0,\cdot)$ will be given by

$$\mu_{m_0} = \max\left\{q^{m_0(d_{m_0}-1)}, (M-m_0+1)(N-d_{m_0}+1)\right\}.$$

The maximum of $H_{q,M,N}$ over $\{1,...,M\} \times \{1,...,N\}$ will be then $H_{\max} = \max\{\mu_m : m \in \{1,...,M\}\}$. ■

Remark 29.37 *The maximum of* $H_{q,M,N}$ *can be attained at a unique point (for example* $H_{2,4,5}(2,2) = 12$*) or at several points (*$H_{2,4,2}(1,2) = H_{2,4,2}(2,1) = H_{2,4,2}(3,1) = 4$*).*

29.9.4. On the existence of maximal arrays

In [3] it was proved, using the results in [7, 12, ?, 54] that there exist finite words with maximal complexity, of any given length; it follows that there are $M \times 1$ and $1 \times N$ maximal arrays for all positive integers M and N. More than that, in [?] the number of the words with maximal complexity is presented. Nevertheless, if both M and N are ≥ 2, the situation differs, as the following proposition shows.

Claim 29.38 *There are sizes M, $N \geq 2$ for which there are no arrays with maximal complexity.*

Proof For $M = N = 4$ calculations show that the total complexity $T_\mathcal{A}$ of any 4×4 array is ≤ 69, while $\sum_{i=1}^{4} \sum_{j=1}^{4} H_{2,4,4}(i,j) = 70$. It follows that for each 4×4 array $\mathcal{A}$ there exists at least one pair (m,n) for which $C_\mathcal{A}(m,n) < H_{2,4,4}(m,n)$. ■

Open question Find the pairs M, N for which there exist maximal arrays in X^{**}.

The result in Proposition 29.38 prevents us from obtaining a q-ary array with maximal complexity for any M and $N \geq 2$. A further question is: given M, N and $m \leq M$, $n \leq N$, is there an $M \times N$ array $\mathcal{A}_{m,n}$ which is (q,m,n)-maximal?

A partial answer is given in [?]: in the binary case, if $(M-m+1)(N-n+1) = 2^{Martin1934}$, there exists a $M \times N$ array which is $(2,m,n)$-maximal (in fact it is $(2,m,n)$-perfect).

Exercises

29.9-1

Problems

29-1 How many

Chapter Notes

Cyclic sequences in which every possible sequence of a fixed length occurs exactly once have been studied for more than a hundred years. The first proof of the existence of $(2,1,a,2^a)$-perfect sequences was published by Flye-Sainte [19] in 1894. The problem was extended to arrays by Fan, Fan, Ma, and Siu in 1985 [17].

One dimensional perfect arrays are often called de Bruijn or Good sequences, since the papers of De Bruijn [7] and Good [20] make these sequences popular. Two dimensional perfect arrays were called perfect maps by Reed and Stewart in 1962 [51] and by Paterson in 1996 [47], or de Bruijn tori by Hurlbert and Isaak and Mitchell in 1993 and later [25, 26, 29].

The even De Bruijn sequences were introduced by A. Iványi and Z. Tóth in 1988 [35, 37]. They proposed an algorithm constructing even sequences for arbitrary alphabet size. Later Hurlbert and Isaak [26] provided an universal algorithm which constructs an infinite sequence whose prefixes are even for the corresponding alphabet size.

The concept of *growing sequences* was introduced by G. Hurlbert and G. Isaak [26].

The necessary conditions (29.2) and (29.3) were formulated at first in papers of Cock in 1988 [10], and in 1994 of Hurlbert and Isaak [26].

For Section **??**:

For Section 29.4:

For Section 29.5:

[4] [9] [10]
[11] [12]
[7] [15] [19] [20] [23]
[25] [26] [27] [28]
[29] [30]
[31] [32] [33] [35] [37]
[41] [43] [45] [46]
[47] [48] [49] [50]
[51] [55]

For Section **??**:

Bibliography

[1] M.-C. Anisiu. Finite words with special complexity properties. *Pure Mathematics and Applications*, 13:31–37, 2002. 1468, 1471, 1480

[2] M.-C. Anisiu, Z. Blázsik, Z. Kása. Maximal complexity of finite words. *Pure Mathematics and Applications*, 13:39–48, 2002. 1482

[3] M.-C. Anisiu, J. Cassaigne. Properties of complexity function for finite words. *Revue Anal. Numér. Theor. Approx.*, 33(2):123–139, 2004. 1480, 1482, 1483

[4] B. Arazi. Position recovery using binary sequences. *Electronic Letters*, 20:61–62, 1984. 1485

[5] C. Berge. *Graphs and Hypergraphs* (2nd revised edition). North-Holland, 1976. MR0384579 (52 #5453). 1467

[6] Z. Blázsik, Z. Kása. Dominating sets in de bruijn graphs. *Pure Mathematics and Applications*, 13(1–2):79–85, 2002. 1480

[7] N. G. Bruijn. A combinatorial problem. *Nederlandse Akademie van Wetenschappen. Proceedings*, 49:758–764, 1946. 1456, 1483, 1484, 1485

[8] A. Carpi, A. de Luca. Words and special factors. *Theoretical Computer Science*, 254:145–182, 2001. 1480

[9] F. Chung, R. L. Graham, P. Diaconis. Universal cycles for combinatorial structures. *Discrete Mathematics*, 110(1–3):43–59, 1992. 1456, 1485

[10] J. C. Cock. Toroidal tilings from de Bruijn cyclic sequences. *Discrete Mathematics*, 70(2):209–210, 1988. 1456, 1457, 1485

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. *Introduction to Algorithms* 3rd edition. The MIT Press/McGraw-Hill, 2009. MR2572804 (2010j:68001). 1467, 1474, 1485

[12] L. J. Cummings, D. Weidemann. Embedded De Bruijn sequences. *Congressus Numer.*, 53:155–160, 1986. 1483, 1485

[13] J. Dassow, V. Mitrana, A. salomaaa. Operations and languages generating devices suggested by genome evolution. *Theoretical Computer Science*, 270(1–2):701–708, 2002. MR 1871091 (2003g:68068). 1467

[14] A. de Luca. On the combinatorics of finite words. *Theoretical Computer Science*, 218(1):13–39, 1999. MR1687752 (2000g:68123). 1480, 1482

[15] J. Dénes, A. D. Keedwell. Frequency allocation for a mobile radio telephone system. *IEEE Transactions on Communication*, 36:765–767, 1988. 1453, 1485

[16] W. Ebeling, R. Feistel. *Physik der Selbstorganisation und Evolution*. Akademie Verlag, Berlin, 1982. Zbl 0747.92020. 1467

[17] C. T. Fan, S. M. Fan, S. M. Ma, M. K. Siu. On de Bruijn arrays. *Ars Combinatoria*, 19A:205–213, 1985. 1453, 1456, 1484

[18] S. Ferenczi, Z. Kása. Complexity for finite factors of infinite sequences. *Theoretical Computer Science*, 218:177–195, 1999. MR1687792 (2000d:68121). 1468, 1480

[19] T. M. Flye-Sainte. Solution of problem 58. *Intermediare des Mathematiciens*, 1:107–110, 1894. 1453, 1484, 1485

[20] I. Good. Normally recurring decimals. *Australasian Journal of Combinatorics*, 21:167–169, 1946. 1484, 1485

[21] M. Harvit. Spectrometric imager. *Applied Optics*, 10:1415–1421, 1971. 1453

[22] M. Heinz. Zur teilwortkomplexität für wörter und folgen über einem endlichen alphabet. *Elektronische Informationverarbeitung und Kibernety*, 13:27–38, 1977. 1468, 1471, 1480

[23] M. Horváth, A. Iványi. Growing perfect cubes. *Discrete Mathematics*, 308:4378–4388, 2008. 1456, 1465, 1466, 1485

[24] L. Hunyadvári, A. Iványi. On some complexity measures of words. In I. Peák (Ed.), *Automata, Languages and Mathematical Systems*, 7–16 pages. Karl Marx University, Budapest, 1984. Bbl559.68065. 1468

[25] G. Hurlbert, G. Isaak. On the de Bruijn torus problem. *Combinatorial Theory Series A*, 164(1):50–62, 1993. 1484, 1485

[26] G. Hurlbert, G. Isaak. A meshing technique for de bruijn tori. *Contemporary Mathematics*, 164(1):50–62, 1994. 1456, 1457, 1458, 1484, 1485

[27] G. Hurlbert, G. Isaak. New constructions for De Bruijn tori. *Designs, Codes and Cryptography*, 1:47–56, 1995. 1485

[28] G. Hurlbert, G. Isaak. On higher dimensional perfect factors. *Ars Combinatoria*, 45:229–239, 1997. 1485

[29] G. Hurlbert, C. J. Mitchell, K. G. Paterson. On the existence of the Bruijn tori with two by two window property. *Combinatorial Theory Series A*, 76(2):213–230, 1996. 1484, 1485

[30] G. Isaak. Construction for higher dimensional perfect multifactors. *Aequationes Mathematicae*, 178:153–160, 2002. 1485

[31] A. Iványi. On the d-complexity of words. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Computarorica*, 8:69–90, 1987. 1453, 1467, 1473, 1477, 1485

[32] A. Iványi. Construction of infinite De Bruijn arrays. *Discrete Applied Mathhematics*, 22:289–293, 1988/89. 1468, 1485

[33] A. Iványi. Construction of three-dimensional perfect matrices. *Ars Combinatoria*, 29C:33–40, 1990. 1465, 1466, 1485

[34] A. Iványi, J. Madarász. Perfect hypercubes. *Electronic Notes on Discrete Mathematics*, 2011. (to appear). 1465, 1466

[35] A. Iványi, Z. Tóth. Existence of De Bruijn words. In I. Peák (Ed.), *Second Conference on Automata, Languages and Programming Systems* (Salgótarján, 1988), 165–172 pages. Karl Marx University of Economics, 1988. 1465, 1466, 1484, 1485

[36] S. Jaeger, R. Lima, B. Mossé. Symbolic analysis of finite words, I: The complexity function. *Bulletin of Brazil Society of Mathematics. New Series*, 457–477 pages, 2013. 1482

[37] D. E. Knuth. *The Art of Computer Programming, Volume 4A. Combinatorial Algorithms*. Addison-Wesley, 2011. 1455, 1456, 1465, 1466, 1484, 1485

[38] D. E. Knuth, R. Graham, O. Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, 1997 (second edition). 1473

[39] Z. Kása. On the d-complexity of strings. *Pure Mathematics and Applications*, 9:119–128, 1998. 1467, 1468

[40] F. Levé, P. Séébold. Proof of a conjecture on word complexity. *Bulletin of Belgian Mathematical Society Simon Stevin*, 8:277–291, 2001. 1482

[41] L. Lovász. *Combinatorial Problems and Exercises* (2. edtion). AMS Chelsea Publishing, 2007. First edition: Academic Press, 1979. MR0537284 (80m:05001). 1456, 1485

[42] S. L. Ma. A note on binary arrays with certain window property. *IEEE Transactions on Information Theory*, 30:774–775, 1984. 1453

[43] M. H. Martin. A problem in arrangements. *Bulletin of American Mathematical Society*, 40:859–864, 1934. 1456, 1485

[44] C. J. Mitchell, K. G. Paterson. Decoding perfect maps. *Designs, Codes and Cryptography*, 4:11–30, 1994. 1453

[45] C. J. Mitchell. Aperiodic and semi-periodic perfect maps. *IEEE Transactions on Information Theory*, 41(1):88–95, 1995. MR1366746 (97b:94018). 1480, 1485

[46] K. G. Paterson. Perfect maps. *IEEE Transactions on Information Theory*, 40(3):743–753, 1994. 1456, 1480, 1485

[47] K. G. Paterson. New classes of perfect maps. i. *Combinatorial Theory Series A*, 73(2):302–334, 1996. 1456, 1484, 1485

[48] K. G. Paterson. New classes of perfect maps. ii. *Combinatorial Theory Series A*, 73(2):335–345, 1996. 1456, 1485

[49] E. M. Petriou, J. Basran. On the position measurement of automated guided vehicle using pseudorandom encoding. *IEEE Transactions on Instrumentation and Measurement*, 38:799–803, 1989. 1485

[50] E. M. Petriou, J. Basran, F. Groen. Automated guided vehicle position recovery. *IEEE Transactions on Instrumentation and Measurement*, 39:254–258, 1990. 1485

[51] I. S. Reed, R. M. Stewart. Note on the existence of perfect maps. *IRE Transactions on Information Theory*, 8:10–12, 1962. 1453, 1484, 1485

[52] G. Rozenberg, A. Salomaa (Eds.). *Handbook of Formal Languages. Vol. 1.* Springer-Verlag, Berlin, 1997. MR98d681. 1467

[53] C. Savage. A survey of combinatorial gray codes. *SIAM Review*, 39:605–629, 1997. 1456

[54] J. O. Shallit. On the maximum number of distinct factors in a binary string. *Graphs and Combinatorics*, 9:197–200, 1993. 1468, 1471, 1477, 1480, 1483

[55] N. Vörös. On the complexity of symbol sequences. In *Conference of Young Programmers and Mathematicians* (ed. A. Iványi), Eötvös Loránd University, Budapest, 43–50 pages, 1984. 1485

[56] E. Yang, Z. The shortest common superstring problem: average case analysis for both exact and approximate matching. *IEEE Transactions on Information Theory*, 45(6):1867–1886, 1999. 1469

This bibliography is made by HBibTEX. First key of the sorting is the name of the authors (first author, second author etc.), second key is the year of publication, third key is the title of the document.

Underlying shows that the electronic version of the bibliography on the homepage of the book contains a link to the corresponding address.

Subject Index

This index uses the following conventions. Numbers are alphabetised as if spelled out; for example, "2-3-4-tree" is indexed as if were "two-three-four-tree". When an entry refers to a place other than the main text, the page number is followed by a tag: *exe* for exercise, *exa* for example, *fig* for figure, *pr* for problem and *fn* for footnote.

The numbers of pages containing a definition are printed in *italic* font, e.g.

time complexity, *583*.

Name Index

This index uses the following conventions. If we know the full name of a cited person, then we print it. If the cited person is not living, and we know the correct data, then we print also the year of her/his birth and death.