ECO-based Gray codes generation for particular classes of words

Vincent VAJNOVSZKI

Université de Bourgogne
Le2i, UMR-CNRS 5158

June 25-27, 2012

Previous work

- F. Ruskey, in *ISAAC Conference, LNCS*, 1993
- S. Bacchelli, E. Barcucci, E. Grazzini, E. Pergola, *Acta Informatica*, 2004
- A. Bernini, E. Grazzini, E. Pergola, R. Pinzani, *Acta Informatica*, 2007
- V. Vajnovszki, *JMIT*, Mons, Belgique, 2008
- F. Ruskey, Joe Sawada, Aaron Williams, in *Journal of Combinatorial Theory, Series A* 2012

Eco-System

$$\vartheta : \mathcal{O}_n \to 2^{\mathcal{O}_{n+1}}$$

If the operator ϑ satisfies :

1. if $x_1, x_2 \in \mathcal{O}$, and $x_1 \neq x_2$, then $\vartheta(x_1) \cap \vartheta(x_2) = \varnothing$,
2. for each $y \in \mathcal{O}_n$, $n \geq 1$, there exists a unique $x \in \mathcal{O}_{n-1}$ such that $y \in \vartheta(x)$,

then $\{\vartheta(x)\}_{x \in \mathcal{O}_{n-1}}$, $n \geq 1$, is a partition of $\mathcal{O}_n$ and ϑ is called an ECO operator

ECO (or succession) rules= formal system consisting of

- a root $e_0 \in \Sigma$
- a set of *productions* of the form

$$\{(k) \rightsquigarrow (e_1(k))(e_2(k)) \cdots (e_{|k|}(k))\}_{k \in \Sigma}$$

which explain how to derive, for any given $k \in \Sigma$, its $|k|$ successors, $(e_1(k)), (e_2(k)), \ldots, (e_{|k|}(k))$

Definition

p-ary Dyck words which are binary words with:

- exactly $p-1$ times as many 0's as 1's
- satisfying the *p*-th order suffix property: any suffix has at least $p-1$ times as many 0's as 1's

D^p_{np} = set of p-ary Dyck words of length np, and $D^2_{2n} = D_{2n}$

Definition

If the suffix property condition is dropped, then the obtained word is called Grand Dyck word / p-ary Grand Dyck words

GD_{2n}= set of length $2n$ Grand Dyck words

GD^p_{np} = length np, p-ary Grand Dyck words

The set of length n binary words with exactly m occurrences of 0 is denoted by $C_{n,m}$

$$D^p_{np} \subset GD^p_{np} = C_{np,n(p-1)}$$

Definition

If the suffix property condition is dropped, then the obtained word is called Grand Dyck word / p-ary Grand Dyck words

GD_{2n}= set of length $2n$ Grand Dyck words

GD^p_{np} = length np, p-ary Grand Dyck words

The set of length n binary words with exactly m occurrences of 0 is denoted by $C_{n,m}$

$$D^p_{np} \subset GD^p_{np} = C_{np,n(p-1)}$$

Definition

A Motzkin word is a word over the alphabet $\{0, 1, a\}$ which after erasing each occurrence of a gives a Dyck word; and we denote M_n the set of length n Motzkin words

M_n = set of length n Motzkin words

Definition

A Schröder word is a Motzkin word in which each length maximal factor of the form $aa\ldots a$ has even length

S_{2n} = the set of length $2n$ Schröder words

Definition

A Motzkin word is a word over the alphabet $\{0, 1, a\}$ which after erasing each occurrence of a gives a Dyck word; and we denote M_n the set of length n Motzkin words

M_n = set of length n Motzkin words

Definition

A Schröder word is a Motzkin word in which each length maximal factor of the form $aa \ldots a$ has even length

S_{2n} = the set of length $2n$ Schröder words

Definition (T. Walsh)

A Gray code is an infinite collection of word-lists, one list for words with same length, such that the number of positions in which two consecutive words in each list differ is bounded (independently of the word-length)

Definition

A Gray code has distance $\delta \geq 1$ if consecutive words differ in at most δ positions

Definition (T. Walsh)

A Gray code is an infinite collection of word-lists, one list for words with same length, such that the number of positions in which two consecutive words in each list differ is bounded (independently of the word-length)

Definition

A Gray code has distance $\delta \geq 1$ if consecutive words differ in at most δ positions

- A Gray code is circular if the last and the first word in the list differ in the same way
- A Gray code is called homogeneous if the 1 and the 0 that exchange positions are separated only by 0's
- (k) or $(\overline{k})$ labeled node in the generating tree corresponds to a word with k successors
- the successors of a $(\overline{k})$ labeled node are the same as for (k), but in reverse order

- A Gray code is circular if the last and the first word in the list differ in the same way
- A Gray code is called homogeneous if the 1 and the 0 that exchange positions are separated only by 0's

- (k) or $(\bar{k})$ labeled node in the generating tree corresponds to a word with k successors
- the successors of a $(\bar{k})$ labeled node are the same as for (k), but in reverse order

Definition (F. Ruskey)

An algorithm for generating a list of words is called CAT (as Constant Average Time) if the number of operations necessary to transform each word into its successor in the list, is constant in average

Dyck words

The succession rule for p-ary Dyck words

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(p+1)\cdots(p+k-1) \end{cases}$$

ith return rule?

110010111000

ith return rule?

110010111000

ith return rule?

110010111000 $\longrightarrow$ 1110010111000**0**

ith return rule?

110010111000 $\longrightarrow$ 11100101110000

ith return rule?

$$
\begin{array}{lcl}
110010111000 & \longrightarrow & 1110010111100000 \\
 & \longrightarrow & 1100110111100000
\end{array}
$$

ith return rule?

110010111000 $\longrightarrow$ 11100101110000

$\longrightarrow$ 11001101110000

*i*th return rule?

$$\begin{array}{lcl} 110010111000 & \longrightarrow & 11100101110000 \\ & \longrightarrow & 11001101110000 \\ & \longrightarrow & 11001011110000 \end{array}$$

ith return rule?

$$
\begin{array}{lcl}
110010111000 & \longrightarrow & 11100101110000 \\
 & \longrightarrow & 11001101110000 \\
 & \longrightarrow & 11001011110000
\end{array}
$$

ith return rule?

$$
\begin{array}{lll}
110010111000 & \longrightarrow & 1110010111000 \\
 & \longrightarrow & 1100110111000 \\
 & \longrightarrow & 1100101111000 \\
 & \longrightarrow & 1100101110010
\end{array}
$$

Last descent rule!

110010111000

Last descent rule!

110010111000

Last descent rule!

110010111000 $\longrightarrow$ 1100101111 0000

Last descent rule!

110010111000 $\longrightarrow$ 1100101111 0000

$\longrightarrow$ 1100101110 1000

Last descent rule!

$$
\begin{array}{lcl}
110010111000 & \longrightarrow & 1100101111{0000} \\
 & \longrightarrow & 1100101110{1000} \\
 & \longrightarrow & 1100101110{0100}
\end{array}
$$

Last descent rule!

$$\begin{array}{lll} 110010111000 & \longrightarrow & 1100101111 0000 \\ & \longrightarrow & 11001011101000 \\ & \longrightarrow & 11001011100100 \\ & \longrightarrow & 11001011100010 \end{array}$$

Let $d = d_1 d_2 \dots d_{np}$ be a p-ary Dyck word of length np, and ℓ be its length-maximal 0's suffix.

$$d = d_1 d_2 \dots d_{np-\ell} 0^{\ell}$$

- $d_{np-\ell} = 1$
- the suffix 0^{ℓ} is called the last descent of d

For each u, $0 \le u \le \ell$

$$d_1 d_2 \dots d_{np-\ell} 0^{u} 1 0^{\ell-u} 0^{p-1}$$

is a p-ary Dyck word of length $(n+1)p$ called a successor of d

last descent rule

Let $d = d_1 d_2 \dots d_{np}$ be a p-ary Dyck word of length np, and ℓ be its length-maximal 0's suffix.

$$d = d_1 d_2 \dots d_{np-\ell} 0^\ell$$

- $d_{np-\ell} = 1$
- the suffix 0^ℓ is called the last descent of d

For each u, $0 \le u \le \ell$

$$d_1 d_2 \dots d_{np-\ell} 0^u 1 0^{\ell-u} 0^{p-1}$$

is a p-ary Dyck word of length $(n+1)p$ called a successor of d

last descent rule

Let $d = d_1 d_2 \ldots d_{np}$ be a p-ary Dyck word of length np, and ℓ be its length-maximal 0's suffix.

$$d = d_1 d_2 \ldots d_{np-\ell} 0^\ell$$

- $d_{np-\ell} = 1$
- the suffix 0^ℓ is called the last descent of d

For each u, $0 \le u \le \ell$

$$d_1 d_2 \ldots d_{np-\ell} 0^u 1 0^{\ell-u} 0^{p-1}$$

is a p-ary Dyck word of length $(n+1)p$ called a successor of d

last descent rule

Let $d = d_1 d_2 \dots d_{np}$ be a p-ary Dyck word of length np, and ℓ be its length-maximal 0's suffix.

$$d = d_1 d_2 \dots d_{np-\ell} 0^\ell$$

- $d_{np-\ell} = 1$
- the suffix 0^ℓ is called the last descent of d

For each u, $0 \le u \le \ell$

$$d_1 d_2 \dots d_{np-\ell} 0^u 1 0^{\ell-u} 0^{p-1}$$

is a p-ary Dyck word of length $(n+1)p$ called a successor of d

last descent rule

Let $d = d_1 d_2 \ldots d_{np}$ be a p-ary Dyck word of length np, and ℓ be its length-maximal 0's suffix.

$$d = d_1 d_2 \ldots d_{np-\ell} 0^\ell$$

- $d_{np-\ell} = 1$
- the suffix 0^ℓ is called the last descent of d

For each u, $0 \leq u \leq \ell$

$$d_1 d_2 \ldots d_{np-\ell} 0^u 1 0^{\ell-u} 0^{p-1}$$

is a p-ary Dyck word of length $(n+1)p$ called a successor of d

last descent rule

- (k) corresponds to a Dyck word with its last descent of length $k-1$. Its (ordered list of) successors are obtained by:
  - inserting from right to left a 1 into its last descent
  - adding a 0^{p-1} suffix
- $(\overline{k})$ labeled node are the same but in reverse order

- (k) corresponds to a Dyck word with its last descent of length $k-1$. Its (ordered list of) successors are obtained by:
  - inserting from right to left a 1 into its last descent
  - adding a 0^{p-1} suffix
- $(\bar{k})$ labeled node are the same but in reverse order

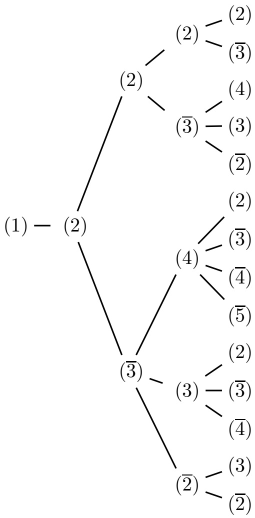
- (k) corresponds to a Dyck word with its last descent of length $k-1$. Its (ordered list of) successors are obtained by:
  - inserting from right to left a 1 into its last descent
  - adding a 0^{p-1} suffix
- $(\overline{k})$ labeled node are the same but in reverse order

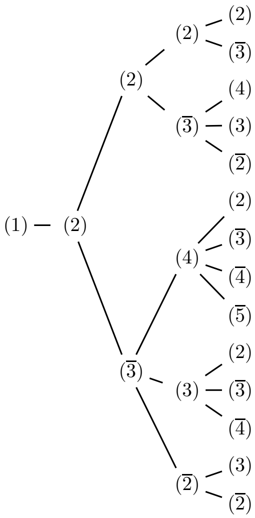
Theorem

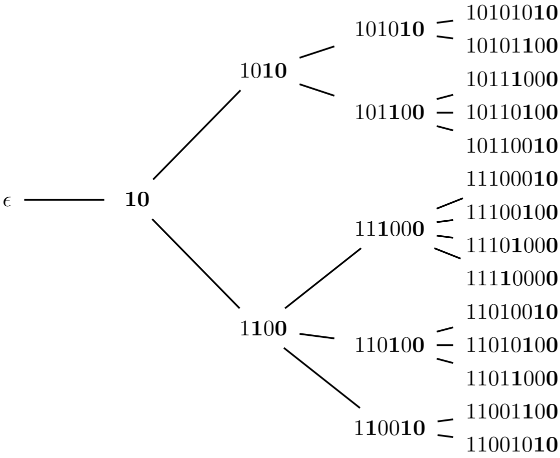
The succession rule

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1}) \cdots (\overline{p+k-1}) \end{cases}$$

gives a circular Gray code for p-ary Dyck words, where the root of the generating tree is the empty word ϵ


(1)
(2)
(2)
$(\overline{3})$
(2)
$(\overline{3})$
(4)
(3)
(2)
$(\overline{3})$
(4)
(3)
$(\overline{2})$
(2)
$(\overline{3})$
$(\overline{4})$
$(\overline{5})$
(2)
$(\overline{3})$
$(\overline{4})$
(3)
$(\overline{2})$


(1)
(2)
(2)
$(\overline{3})$
(2)
$(\overline{3})$
(4)
(3)
$(\overline{3})$
(2)
$(\overline{2})$
(4)
(3)
$(\overline{2})$
(2)
$(\overline{3})$
$(\overline{4})$
$(\overline{5})$
(2)
$(\overline{3})$
$(\overline{4})$
(3)
$(\overline{2})$


ϵ
10
1010
1100
101010
101100
111000
110100
110010
10101010
10101100
10111000
10110100
10110010
11100010
11100100
11101000
11110000
11010010
11010100
11011000
11001100
11001010

Proposition

For $p \geq 2$, the first and last word given by this succession rule are

- $(10^{p-1})^n$, $n \geq 1$, *and*
- $110^{2p-2}(10^{p-1})^{n-2}$, $n \geq 2$

Remark

For $p = 2$, the Gray code induced on D_{2n} by this succession rule is the reverse of Ruskey-Proskurowski's Gray code (1990)

Proposition

For $p \geq 2$, the first and last word given by this succession rule are

- $(10^{p-1})^n$, $n \geq 1$, *and*
- $110^{2p-2}(10^{p-1})^{n-2}$, $n \geq 2$

Remark

For $p = 2$, the Gray code induced on D_{2n} by this succession rule is the reverse of Ruskey-Proskurowski's Gray code (1990)

More restrictive Gray codes

Theorem

The succession rule

$$\left\{ \begin{array}{l} (1)_a \\ (k_a) \rightsquigarrow (p_a)\, (\overline{p+1})_b\, (\overline{p+2})_b\, \cdots (\overline{p+k-1})_b \\ \\ \end{array} \right.$$

More restrictive Gray codes

Theorem

The succession rule

$$\begin{cases} (1)_a \\ (k_a) \rightsquigarrow (p_a)\ (\overline{p+1})_b\ (\overline{p+2})_b\ \cdots (\overline{p+k-1})_b \\ (k_b) \rightsquigarrow (\overline{p+k-1})_a\ (p_a)\ (\overline{p+1})_b\ (\overline{p+2})_b\ \cdots\ (\overline{p+k-2})_b \end{cases}$$

gives a homogeneous Gray code for p-ary Dyck words, where the root of the generating tree is the empty word ϵ

Proposition

The first and last length np word given by this succession rule are

- $(10^{p-1})^n$
- $1^n 0^{(p-1)n}$

Remark

The Gray code induced on D_{np}^{p} by the succession rule is Eades-McKay-Bultena-Ruskey's Gray code (1998)

Proposition

The first and last length np word given by this succession rule are

- $(10^{p-1})^n$
- $1^n 0^{(p-1)n}$

Remark

The Gray code induced on D^p_{np} by the succession rule is Eades-McKay-Bultena-Ruskey's Gray code (1998)

ith descent rule

*i*th descent rule

01011000

Combinations and Grand Dyck words

*i*th descent rule

01011000

Combinations and Grand Dyck words

*i*th descent rule

$01011000 \longrightarrow 010111000$

Combinations and Grand Dyck words

*i*th descent rule

$$01011000 \longrightarrow 010111000$$
$$\longrightarrow 010110100$$

Combinations and Grand Dyck words

*i*th descent rule

$$\begin{array}{lcl} 01011000 & \longrightarrow & 010111000 \\ & \longrightarrow & 010110100 \\ & \longrightarrow & 010110010 \end{array}$$

Combinations and Grand Dyck words

*i*th descent rule

$$
\begin{array}{lcl}
01011000 & \longrightarrow & 010111000 \\
 & \longrightarrow & 010110100 \\
 & \longrightarrow & 010110010 \\
 & \longrightarrow & 010110001
\end{array}
$$

Let $d = d_1 d_2 \ldots d_n$ be a binary word in $C_{n,m}$, and ℓ be its length-maximal 0's suffix

$$d = d_1 d_2 \ldots d_{n-\ell} 0^{\ell}$$

- $d_{n-\ell} = 1$
- the suffix 0^{ℓ} is called the last descent of d

For each u, $0 \le u \le \ell$,

$$d_1 d_2 \ldots d_{n-\ell} 0^{u} 1 0^{\ell-u}$$

is a length $(n+1)$ binary word in $C_{n+1,m}$, called a successor of d.

last descent rule

Let $d = d_1 d_2 \dots d_n$ be a binary word in $C_{n,m}$, and ℓ be its length-maximal 0's suffix

$$d = d_1 d_2 \dots d_{n-\ell} 0^{\ell}$$

- $d_{n-\ell} = 1$
- the suffix 0^{ℓ} is called the last descent of d

For each u, $0 \leq u \leq \ell$,

$$d_1 d_2 \dots d_{n-\ell} 0^{u} 1 0^{\ell-u}$$

is a length $(n+1)$ binary word in $C_{n+1,m}$, called a successor of d.

last descent rule

Let $d = d_1 d_2 \ldots d_n$ be a binary word in $C_{n,m}$, and ℓ be its length-maximal 0's suffix

$$d = d_1 d_2 \ldots d_{n-\ell} 0^\ell$$

- $d_{n-\ell} = 1$
- the suffix 0^ℓ is called the last descent of d

For each u, $0 \leq u \leq \ell$,

$$d_1 d_2 \ldots d_{n-\ell} 0^u 1 0^{\ell-u}$$

is a length $(n+1)$ binary word in $C_{n+1,m}$, called a successor of d.

last descent rule

Let $d = d_1 d_2 \dots d_n$ be a binary word in $C_{n,m}$, and ℓ be its length-maximal 0's suffix

$$d = d_1 d_2 \dots d_{n-\ell} 0^\ell$$

- $d_{n-\ell} = 1$
- the suffix 0^ℓ is called the last descent of d

For each u, $0 \leq u \leq \ell$,

$$d_1 d_2 \dots d_{n-\ell} 0^u 1 0^{\ell-u}$$

is a length $(n+1)$ binary word in $C_{n+1,m}$, called a successor of d.

last descent rule

Let $d = d_1 d_2 \ldots d_n$ be a binary word in $C_{n,m}$, and ℓ be its length-maximal 0's suffix

$$d = d_1 d_2 \ldots d_{n-\ell} 0^\ell$$

- $d_{n-\ell} = 1$
- the suffix 0^ℓ is called the last descent of d

For each u, $0 \leq u \leq \ell$,

$$d_1 d_2 \ldots d_{n-\ell} 0^u 1 0^{\ell-u}$$

is a length $(n+1)$ binary word in $C_{n+1,m}$, called a successor of d.

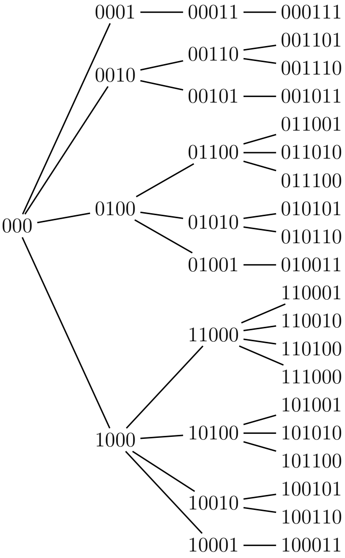
last descent rule

Theorem

For a fixed $m \geq 1$, the succession rule

$$\begin{cases} (m+1) \\ (k) \rightsquigarrow (1)(\overline{2}) \cdots (\overline{k}) \end{cases}$$

gives a (circular) Gray code for $C_{n,m}$, $n \geq m$ where the size zero object is 0^m


000
0001
0010
0100
1000
00011
00110
00101
01100
01010
01001
11000
10100
10010
10001
000111
001101
001110
001011
011001
011010
011100
010101
010110
010011
110001
110010
110100
111000
101001
101010
101100
100101
100110
100011

In particular, for a given $p \geq 2$, when $n = \frac{m}{p-1} \cdot p$, (that is, at level $n - m$ in the generating tree) the succession this rule yields a Gray code for $GD_n^p = C_{n,m}$.

Remark

The Gray code induced on $C_{n,m}$ by the succession rule (9) is the revolving door Gray code Liu-Tang/Nijenhuis-Wilf

In particular, for a given $p \geq 2$, when $n = \frac{m}{p-1} \cdot p$, (that is, at level $n - m$ in the generating tree) the succession this rule yields a Gray code for $GD_n^p = C_{n,m}$.

Remark

The Gray code induced on $C_{n,m}$ by the succession rule (9) is the revolving door Gray code Liu-Tang/Nijenhuis-Wilf

Last descent rule?

Last descent rule?

$1a1a1aa000$

Last descent rule?

$1a1a1aa000$

Motzkin words

Last descent rule?

$$1a1a1aa000 \longrightarrow 1a1a1aa000a$$

Motzkin words

Last descent rule?

$$1a1a1aa000 \longrightarrow 1a1a1aa000a$$
$$\longrightarrow 1a1a1aa00a0$$

Last descent rule?

$$\begin{array}{lcl} 1a1a1aa\color{blue}{000} & \longrightarrow & 1a1a1aa\color{blue}{000}\color{red}{a} \\ & \longrightarrow & 1a1a1aa\color{blue}{00}\color{red}{a}\color{blue}{0} \\ & \longrightarrow & 1a1a1aa\color{blue}{0}\color{red}{a}\color{blue}{00} \end{array}$$

Last descent rule?

$$
\begin{array}{lll}
1a1a1aa000 & \longrightarrow & 1a1a1aa000a \\
 & \longrightarrow & 1a1a1aa00a0 \\
 & \longrightarrow & 1a1a1aa0a00 \\
 & \longrightarrow & 1a1a1aaa000
\end{array}
$$

Last descent rule?

$$
\begin{array}{lcl}
1a1a1aa000 & \longrightarrow & 1a1a1aa000a \\
 & \longrightarrow & 1a1a1aa00a0 \\
 & \longrightarrow & 1a1a1aa0a00 \\
 & \longrightarrow & 1a1a1aaa000 \\
 & \longrightarrow & 1a1a1aa00010
\end{array}
$$

Last descent rule?

$$
\begin{array}{lcl}
1a1a1aa000 & \longrightarrow & 1a1a1aa000a \\
 & \longrightarrow & 1a1a1aa00a0 \\
 & \longrightarrow & 1a1a1aa0a00 \\
 & \longrightarrow & 1a1a1aaa000 \\
 & \longrightarrow & 1a1a1aa00010 \\
 & \longrightarrow & 1a1a1aa00100
\end{array}
$$

Last descent rule?

$$
\begin{array}{lll}
1a1a1aa000 & \longrightarrow & 1a1a1aa000a \\
 & \longrightarrow & 1a1a1aa00a0 \\
 & \longrightarrow & 1a1a1aa0a00 \\
 & \longrightarrow & 1a1a1aaa000 \\
 & \longrightarrow & 1a1a1aa00010 \\
 & \longrightarrow & 1a1a1aa00100 \\
 & \longrightarrow & 1a1a1aa01000
\end{array}
$$

Last descent rule?

$$
\begin{array}{lcl}
1a1a1aa000 & \longrightarrow & 1a1a1aa000a \\
 & \longrightarrow & 1a1a1aa00a0 \\
 & \longrightarrow & 1a1a1aa0a00 \\
 & \longrightarrow & 1a1a1aaa000 \\
 & \longrightarrow & 1a1a1aa00010 \\
 & \longrightarrow & 1a1a1aa00100 \\
 & \longrightarrow & 1a1a1aa01000 \\
 & \longrightarrow & 1a1a1aa10000
\end{array}
$$

Last descent rule?

$$
\begin{array}{lll}
1a1a1aa000 & \longrightarrow & 1a1a1aa000a \\
 & \longrightarrow & 1a1a1aa00a0 \\
 & \longrightarrow & 1a1a1aa0a00 \\
 & \longrightarrow & 1a1a1aaa000 \\
 & \longrightarrow & 1a1a1aa00010 \\
 & \longrightarrow & 1a1a1aa00100 \\
 & \longrightarrow & 1a1a1aa01000 \\
 & \longrightarrow & 1a1a1aa10000
\end{array}
$$

Last ascent-free rule!

Let $w \in M_n$ be a length-n Motzkin word, s its length-maximal suffix which does not contain the letter 1, and

$$k = |s|_a + 1 = \text{the number of successors of } w$$

Last ascent-free rule!

Let $w \in M_n$ be a length-n Motzkin word, s its length-maximal suffix which does not contain the letter 1, and

$$k = |s|_a + 1 = \text{the number of successors of } w$$

- The word $w\textcolor{red}{a}$ is a Motzkin word of length $n+1$ with $k+1$ successors;

Last ascent-free rule!

Let $w \in M_n$ be a length-n Motzkin word, s its length-maximal suffix which does not contain the letter 1, and

$$k = |s|_a + 1 \text{=the number of successors of } w$$

- The word wa is a Motzkin word of length $n+1$ with $k+1$ successors;
- If $|s|_a > 0$, then for a given j, $|s|_a \geq j > 0$ let $w'as''$ be the factorization of w where s'' is the suffix of w with exactly $j-1$ occurrences of a. The word $v = w'1s''0$ is a Motzkin word of length $n+1$; it has j successors.

$11a01a0a0$

$11a01a0a0$

$$11a01a0a0 \longrightarrow 11a01a0a0a$$

$$
\begin{array}{lcl}
11a01a0a0 & \longrightarrow & 11a01a0a0a \\
 & \longrightarrow & 11a0110a00
\end{array}
$$

$$
\begin{array}{lll}
11a01a0a0 & \longrightarrow & 11a01a0a0a \\
 & \longrightarrow & 11a0110a00 \\
 & \longrightarrow & 11a01a0100
\end{array}
$$

Theorem

The succession rule

$$\begin{cases} (1) \\ (k) \rightsquigarrow (\overline{1})(\overline{2})\cdots(\overline{k-1})(k+1) \end{cases}$$

gives a Gray code for M_n with distance 4

Schröder words

Let $w \in S_{2n}$, and k the length of its length-maximal 0's suffix, plus one; $w = w'0^{k-1}$

w has $2k$ successors and the following gives the successors of w (an ECO operator for the set of Schröder words)

- $waa \in S_{2n+2}$; it has with 2 successors
- $w'100^{k-1} \in S_{2n+2}$; it has $2k + 2$ successors

In addition, if $k > 1$, then for any j, $1 \leq j \leq k - 1$ we have

- $w'0^{k-1-j}aa0^j \in S_{2n+2}$; it has $2j + 2$ successors
- $w'0^{k-j}100^{j-1} \in S_{2n+2}$; it has $2j + 2$ successors

Schröder words

Let $w \in S_{2n}$, and k the length of its length-maximal 0's suffix, plus one; $w = w'0^{k-1}$
w has $2k$ successors and the following gives the successors of w (an ECO operator for the set of Schröder words)

- $waa \in S_{2n+2}$; it has with 2 successors
- $w'100^{k-1} \in S_{2n+2}$; it has $2k+2$ successors

In addition, if $k > 1$, then for any j, $1 \leq j \leq k-1$ we have

- $w'0^{k-1-j}aa0^j \in S_{2n+2}$; it has $2j+2$ successors
- $w'0^{k-j}100^{j-1} \in S_{2n+2}$; it has $2j+2$ successors

Schröder words

Let $w \in S_{2n}$, and k the length of its length-maximal 0's suffix, plus one; $w = w'0^{k-1}$
w has $2k$ successors and the following gives the successors of w (an ECO operator for the set of Schröder words)

- $waa \in S_{2n+2}$; it has with 2 successors
- $w'100^{k-1} \in S_{2n+2}$; it has $2k+2$ successors

In addition, if $k > 1$, then for any j, $1 \leq j \leq k-1$ we have

- $w'0^{k-1-j}aa0^j \in S_{2n+2}$; it has $2j+2$ successors
- $w'0^{k-j}100^{j-1} \in S_{2n+2}$; it has $2j+2$ successors

Let $w \in S_{2n}$, and k the length of its length-maximal 0's suffix, plus one; $w = w'0^{k-1}$
w has $2k$ successors and the following gives the successors of w (an ECO operator for the set of Schröder words)

- $waa \in S_{2n+2}$; it has with 2 successors
- $w'100^{k-1} \in S_{2n+2}$; it has $2k + 2$ successors

In addition, if $k > 1$, then for any j, $1 \leq j \leq k-1$ we have

- $w'0^{k-1-j}aa0^j \in S_{2n+2}$; it has $2j + 2$ successors
- $w'0^{k-j}100^{j-1} \in S_{2n+2}$; it has $2j + 2$ successors

Schröder words

Let $w \in S_{2n}$, and k the length of its length-maximal 0's suffix, plus one; $w = w'0^{k-1}$

w has $2k$ successors and the following gives the successors of w (an ECO operator for the set of Schröder words)

- $waa \in S_{2n+2}$; it has with 2 successors
- $w'100^{k-1} \in S_{2n+2}$; it has $2k+2$ successors

In addition, if $k > 1$, then for any j, $1 \leq j \leq k-1$ we have

- $w'0^{k-1-j}aa0^{j} \in S_{2n+2}$; it has $2j+2$ successors
- $w'0^{k-j}100^{j-1} \in S_{2n+2}$; it has $2j+2$ successors

Schröder words

Let $w \in S_{2n}$, and k the length of its length-maximal 0's suffix, plus one; $w = w'0^{k-1}$
w has $2k$ successors and the following gives the successors of w (an ECO operator for the set of Schröder words)

- $waa \in S_{2n+2}$; it has with 2 successors
- $w'100^{k-1} \in S_{2n+2}$; it has $2k + 2$ successors

In addition, if $k > 1$, then for any j, $1 \leq j \leq k - 1$ we have

- $w'0^{k-1-j}aa0^j \in S_{2n+2}$; it has $2j + 2$ successors
- $w'0^{k-j}100^{j-1} \in S_{2n+2}$; it has $2j + 2$ successors

last descent rule

$11aa1aa000$

last descent rule

$11aa1aa000$

last descent rule

$11aa1aa000 \longrightarrow 11aa1aa000aa$

last descent rule

$$
\begin{array}{lll}
11aa1aa000 & \longrightarrow & 11aa1aa000aa \\
 & \longrightarrow & 11aa1aa00aa0
\end{array}
$$

last descent rule

$$
\begin{aligned}
11aa1aa000 &\longrightarrow 11aa1aa000aa \\
&\longrightarrow 11aa1aa00aa0 \\
&\longrightarrow 11aa1aa0aa00
\end{aligned}
$$

last descent rule

$$
\begin{array}{lll}
11aa1aa000 & \longrightarrow & 11aa1aa000aa \\
 & \longrightarrow & 11aa1aa00aa0 \\
 & \longrightarrow & 11aa1aa0aa00 \\
 & \longrightarrow & 11aa1aaaa000
\end{array}
$$

last descent rule

$$\begin{array}{lll}
11aa1aa000 & \longrightarrow & 11aa1aa000aa \\
 & \longrightarrow & 11aa1aa00aa0 \\
 & \longrightarrow & 11aa1aa0aa00 \\
 & \longrightarrow & 11aa1aaaa000 \\
 & \longrightarrow & 11aa1aa00010
\end{array}$$

last descent rule

$$
\begin{array}{lcl}
11aa1aa000 & \longrightarrow & 11aa1aa000aa \\
 & \longrightarrow & 11aa1aa00aa0 \\
 & \longrightarrow & 11aa1aa0aa00 \\
 & \longrightarrow & 11aa1aaaa000 \\
 & \longrightarrow & 11aa1aa00010 \\
 & \longrightarrow & 11aa1aa00100
\end{array}
$$

last descent rule

$$
\begin{array}{lll}
11aa1aa000 & \longrightarrow & 11aa1aa000aa \\
 & \longrightarrow & 11aa1aa00aa0 \\
 & \longrightarrow & 11aa1aa0aa00 \\
 & \longrightarrow & 11aa1aaaa000 \\
 & \longrightarrow & 11aa1aa00010 \\
 & \longrightarrow & 11aa1aa00100 \\
 & \longrightarrow & 11aa1aa01000
\end{array}
$$

$$
\begin{array}{lcl}
11aa1aa000 & \longrightarrow & 11aa1aa000aa \\
 & \longrightarrow & 11aa1aa00aa0 \\
 & \longrightarrow & 11aa1aa0aa00 \\
 & \longrightarrow & 11aa1aaaa000 \\
 & \longrightarrow & 11aa1aa00010 \\
 & \longrightarrow & 11aa1aa00100 \\
 & \longrightarrow & 11aa1aa01000 \\
 & \longrightarrow & 11aa1aa10000
\end{array}
$$

last descent rule

(8)

$$
\begin{array}{lll}
11aa1aa000 & \longrightarrow & 11aa1aa000aa \\
 & \longrightarrow & 11aa1aa00aa0 \\
 & \longrightarrow & 11aa1aa0aa00 \\
 & \longrightarrow & 11aa1aaaa000 \\
 & \longrightarrow & 11aa1aa00010 \\
 & \longrightarrow & 11aa1aa00100 \\
 & \longrightarrow & 11aa1aa01000 \\
 & \longrightarrow & 11aa1aa10000
\end{array}
$$

last descent rule

$$
\begin{array}{lcll}
\overset{(8)}{11aa1aa000} & \longrightarrow & 11aa1aa000aa & (2) \\
 & \longrightarrow & 11aa1aa00aa0 & \\
 & \longrightarrow & 11aa1aa0aa00 & \\
 & \longrightarrow & 11aa1aaaa000 & \\
 & \longrightarrow & 11aa1aa00010 & \\
 & \longrightarrow & 11aa1aa00100 & \\
 & \longrightarrow & 11aa1aa01000 & \\
 & \longrightarrow & 11aa1aa10000 &
\end{array}
$$

last descent rule

$$
\begin{array}{cccc}
(8) & & & \\
11a a 1 a a{\color{blue}000} & \longrightarrow & 11aa1aa{\color{blue}000}{\color{red}aa} & (2) \\
 & \longrightarrow & 11aa1aa{\color{blue}00}{\color{red}aa}{\color{blue}0} & (4) \\
 & \longrightarrow & 11aa1aa{\color{blue}0}{\color{red}aa}{\color{blue}00} & \\
 & \longrightarrow & 11aa1aa{\color{red}aa}{\color{blue}000} & \\
 & \longrightarrow & 11aa1aa{\color{blue}000}{\color{red}10} & \\
 & \longrightarrow & 11aa1aa{\color{blue}00}{\color{red}1}{\color{blue}00} & \\
 & \longrightarrow & 11aa1aa{\color{blue}0}{\color{red}1}{\color{blue}000} & \\
 & \longrightarrow & 11aa1aa{\color{red}1}{\color{blue}0000} & \\
\end{array}
$$

last descent rule

$$\begin{array}{cccc}
(8) & & & \\
11aa1aa000 & \longrightarrow & 11aa1aa000aa & (2) \\
 & \longrightarrow & 11aa1aa00aa0 & (4) \\
 & \longrightarrow & 11aa1aa0aa00 & (6) \\
 & \longrightarrow & 11aa1aaaa000 & \\
 & \longrightarrow & 11aa1aa00010 & \\
 & \longrightarrow & 11aa1aa00100 & \\
 & \longrightarrow & 11aa1aa01000 & \\
 & \longrightarrow & 11aa1aa10000 &
\end{array}$$

last descent rule

$$\begin{array}{cccc}
(8) & & & \\
11aa1aa000 & \longrightarrow & 11aa1aa000aa & (2) \\
 & \longrightarrow & 11aa1aa00aa0 & (4) \\
 & \longrightarrow & 11aa1aa0aa00 & (6) \\
 & \longrightarrow & 11aa1aaaa000 & (8) \\
 & \longrightarrow & 11aa1aa00010 & \\
 & \longrightarrow & 11aa1aa00100 & \\
 & \longrightarrow & 11aa1aa01000 & \\
 & \longrightarrow & 11aa1aa10000 &
\end{array}$$

last descent rule

$$\begin{array}{cccc}
\overset{(8)}{11aa1aa000} & \longrightarrow & 11aa1aa000aa & (2) \\
 & \longrightarrow & 11aa1aa00aa0 & (4) \\
 & \longrightarrow & 11aa1aa0aa00 & (6) \\
 & \longrightarrow & 11aa1aaaa000 & (8) \\
 & \longrightarrow & 11aa1aa00010 & (4) \\
 & \longrightarrow & 11aa1aa00100 & \\
 & \longrightarrow & 11aa1aa01000 & \\
 & \longrightarrow & 11aa1aa10000 &
\end{array}$$

last descent rule

$$\begin{array}{cccc}
(8) & & & \\
11aa1aa000 & \longrightarrow & 11aa1aa000aa & (2) \\
 & \longrightarrow & 11aa1aa00aa0 & (4) \\
 & \longrightarrow & 11aa1aa0aa00 & (6) \\
 & \longrightarrow & 11aa1aaaa000 & (8) \\
 & \longrightarrow & 11aa1aa00010 & (4) \\
 & \longrightarrow & 11aa1aa00100 & (6) \\
 & \longrightarrow & 11aa1aa01000 & \\
 & \longrightarrow & 11aa1aa10000 &
\end{array}$$

last descent rule

$$
\begin{array}{lcll}
 & & (8) & \\
11aa1aa000 & \longrightarrow & 11aa1aa000aa & (2) \\
 & \longrightarrow & 11aa1aa00aa0 & (4) \\
 & \longrightarrow & 11aa1aa0aa00 & (6) \\
 & \longrightarrow & 11aa1aaaa000 & (8) \\
 & \longrightarrow & 11aa1aa00010 & (4) \\
 & \longrightarrow & 11aa1aa00100 & (6) \\
 & \longrightarrow & 11aa1aa01000 & (8) \\
 & \longrightarrow & 11aa1aa10000 &
\end{array}
$$

last descent rule

$$\begin{array}{ccll}
\overset{(8)}{11aa1aa000} & \longrightarrow & 11aa1aa000aa & (2) \\
 & \longrightarrow & 11aa1aa00aa0 & (4) \\
 & \longrightarrow & 11aa1aa0aa00 & (6) \\
 & \longrightarrow & 11aa1aaaa000 & (8) \\
 & \longrightarrow & 11aa1aa00010 & (4) \\
 & \longrightarrow & 11aa1aa00100 & (6) \\
 & \longrightarrow & 11aa1aa01000 & (8) \\
 & \longrightarrow & 11aa1aa10000 & (10)
\end{array}$$

Theorem

The succession rule

$$\left\{\begin{array}{l}(2)\\(2k) \rightsquigarrow (2)(\overline{4})(\overline{4})(\overline{6})(\overline{6})\cdots(\overline{2k})(\overline{2k})(\overline{2k+2})\end{array}\right.$$

gives a Gray code for S_n with distance 5

Theorem

The succession rule

$$\left\{ \begin{array}{l} (2) \\ (2k) \rightsquigarrow (2)(\overline{4})(\overline{4})(\overline{6})(\overline{6}) \cdots (\overline{2k})(\overline{2k})(\overline{2k+2}) \end{array} \right.$$

gives a Gray code for S_n with distance 5

- $(\bar{j})$: $\ldots aa \ldots$
- $(\bar{j})$: $\ldots 10 \ldots$

Algorithmic implementation

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1})\cdots(\overline{p+k-1}) \end{cases}$$

`gen_Dyck_up` generates D_{np}^{p}

- words are stored in the global array d
- initialized by 0^{np}
- main call is: `gen_Dyck_up(0,1)`

  0 is the size
  1 is the degree
- `gen_Dyck_down`

  executes the statements of `gen_Dyck_up` in reverse order
  replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa

Algorithmic implementation

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1})\cdots(\overline{p+k-1}) \end{cases}$$

`gen_Dyck_up` generates D^p_{np}

- words are stored in the global array d
- initialized by 0^{np}
- main call is: `gen_Dyck_up(0,1)`

  0 is the size
  1 is the degree
- `gen_Dyck_down`

  executes the statements of `gen_Dyck_up` in reverse order
  replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa

Algorithmic implementation

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1})\cdots(\overline{p+k-1}) \end{cases}$$

`gen_Dyck_up` generates D_{np}^{p}

- words are stored in the global array d
- initialized by 0^{np}
- main call is: `gen_Dyck_up(0,1)`

  0 is the size

  1 is the degree
- `gen_Dyck_down`

  executes the statements of `gen_Dyck_up` in reverse order

  replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa

Algorithmic implementation

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1})\cdots(\overline{p+k-1}) \end{cases}$$

`gen_Dyck_up` generates D^p_{np}

- words are stored in the global array d
- initialized by 0^{np}
- main call is: `gen_Dyck_up(0,1)`

  0 is the size
  1 is the degree
- `gen_Dyck_down`

  executes the statements of `gen_Dyck_up` in reverse order
  replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa

Algorithmic implementation

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1})\cdots(\overline{p+k-1}) \end{cases}$$

`gen_Dyck_up` generates D^p_{np}

- words are stored in the global array d
- initialized by 0^{np}
- main call is: `gen_Dyck_up(0,1)`

  0 is the size

  1 is the degree
- `gen_Dyck_down`

  executes the statements of `gen_Dyck_up` in reverse order

  replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa

Algorithmic implementation

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1})\cdots(\overline{p+k-1}) \end{cases}$$

`gen_Dyck_up` generates D_{np}^{p}

- words are stored in the global array d
- initialized by 0^{np}
- main call is: `gen_Dyck_up(0,1)`

  0 is the size
  1 is the degree
- `gen_Dyck_down`

  executes the statements of `gen_Dyck_up` in reverse order
  replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa

Algorithmic implementation

$$\begin{cases} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1})\cdots(\overline{p+k-1}) \end{cases}$$

`gen_Dyck_up` generates D_{np}^{p}

- words are stored in the global array d
- initialized by 0^{np}
- main call is: `gen_Dyck_up(0,1)`

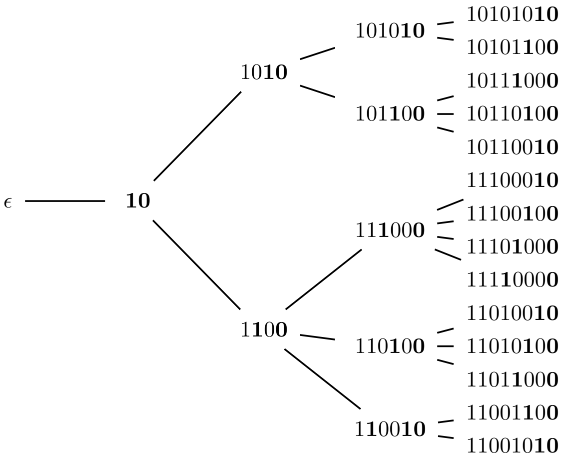
  0 is the size

  1 is the degree
- `gen_Dyck_down`

  executes the statements of `gen_Dyck_up` in reverse order

  replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa

```
procedure gen_Dyck_up(size, k)
local i;
if size = n·p then Print(d);
else d[size + 1] := 1;
     gen_Dyck_up(size + p, p);
     d[size + 1] := 0;
     for i from p + 1 to k + p − 1 do
          d[size + p + 1 − i] := 1;
          gen_Dyck_down(size + p, i);
          d[size + p + 1 − i] := 0;
     end do
end if
end procedure.
```


ϵ
10
1010
1100
101010
101100
111000
110100
110010
10101010
10101100
10111000
10110100
10110010
11100010
11100100
11101000
11110000
11010010
11010100
11011000
11001100
11001010

Proposition

Procedure `gen_Dyck_up` *generates p-ary Dyck words of length np in constant average time.*

- the total amount of computation in each call is proportional with the number of direct calls produced by this call
- each non-terminal call, except the root, produces at least two recursive calls (i.e., there is no call of degree one, except to the main call)
- each terminal call (degree-zero call) produces a new permutation

Proposition

Procedure `gen_Dyck_up` *generates p-ary Dyck words of length np in constant average time.*

- the total amount of computation in each call is proportional with the number of direct calls produced by this call
- each non-terminal call, except the root, produces at least two recursive calls (i.e., there is no call of degree one, except to the main call)
- each terminal call (degree-zero call) produces a new permutation

Proposition

Procedure `gen_Dyck_up` *generates p-ary Dyck words of length np in constant average time.*

- the total amount of computation in each call is proportional with the number of direct calls produced by this call
- each non-terminal call, except the root, produces at least two recursive calls (i.e., there is no call of degree one, except to the main call)
- each terminal call (degree-zero call) produces a new permutation

Proposition

Procedure `gen_Dyck_up` *generates p-ary Dyck words of length np in constant average time.*

- the total amount of computation in each call is proportional with the number of direct calls produced by this call
- each non-terminal call, except the root, produces at least two recursive calls (i.e., there is no call of degree one, except to the main call)
- each terminal call (degree-zero call) produces a new permutation

Remark

Similar algorithms can be designed for

- *homogeneous Gray code for p-ary Dyck words*
- *Motzkin words*
- *Schröder words*

$C_{n,m}$

$$\begin{cases} (m+1) \\ (k) \rightsquigarrow (1)(\overline{2})\cdots(\overline{k}) \end{cases}$$

The implementation of this succession rule does not give a CAT algorithm.

Remark (numerical evidences)

For an integer $p \geq 2$ and for $n = \frac{mp}{p-1}$ we have

$$\frac{total\ amount\ of\ computation}{number\ of\ generated\ words} \leq 2$$

and this rule yields a CAT algorithm for the set of p-ary Grand Dyck words

$C_{n,m}$

$$\begin{cases} (m+1) \\ (k) \rightsquigarrow (1)(\overline{2})\cdots(\overline{k}) \end{cases}$$

The implementation of this succession rule does not give a CAT algorithm.

Remark (numerical evidences)

For an integer $p \geq 2$ and for $n = \frac{mp}{p-1}$ we have

$$\frac{total\ amount\ of\ computation}{number\ of\ generated\ words} \leq 2$$

and this rule yields a CAT algorithm for the set of p-ary Grand Dyck words

$C_{n,m}$

$$\begin{cases} (m+1) \\ (k) \rightsquigarrow (1)(\overline{2})\cdots(\overline{k}) \end{cases}$$

The implementation of this succession rule does not give a CAT algorithm.

Remark (numerical evidences)

For an integer $p \geq 2$ and for $n = \frac{mp}{p-1}$ we have

$$\frac{\textit{total amount of computation}}{\textit{number of generated words}} \leq 2$$

and this rule yields a CAT algorithm for the set of p-ary Grand Dyck words

$C_{n,m}$

$$\begin{cases} (m+1) \\ (k) \rightsquigarrow (1)(\overline{2})\cdots(\overline{k}) \end{cases}$$

The implementation of this succession rule does not give a CAT algorithm.

Remark (numerical evidences)

For an integer $p \geq 2$ and for $n = \frac{mp}{p-1}$ we have

$$\frac{\textit{total amount of computation}}{\textit{number of generated words}} \leq 2$$

and this rule yields a CAT algorithm for the set of p-ary Grand Dyck words