Efficient generation of some greedy Gray codes

Vincent Vajnovszki
joint work with Nathanaël Hassler and Dennis Wong
Laboratoire d'Informatique de Bourgogne & Macao Polytechnic University

2024 KMS Annual Meeting
October 24–26th
Suwon, Republic of Korea

Overview

- The greedy Gray code algorithm
- Restricted classes of binary words
- Efficient exhaustive generation

- The greedy Gray code algorithm
- Restricted classes of binary words
- Efficient generation

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000
001
011
010
110
111
101
100

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000
001
011
010
110
111
101
100

0011

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a ‘small change’.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000
001
011
010
110
111
101
100

0011

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000
001
011
010
110
111
101
100

0011
1001

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000
001
011
010
110
111
101
100

0011
1001

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000
001
011
010
110
111
101
100

0011
1001
0101

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a ‘small change’.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000	
001	0011
011	1001
010	0101
110	
111	
101	
100	

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a ‘small change’.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000
001
011
010
110
111
101
100

0011
1001
0101
0110

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000	
001	0011
011	1001
010	0101
110	0110
111	
101	
100	

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000	
001	0011
011	1001
010	0101
110	0110
111	1010
101	
100	

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a ‘small change’.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000	
001	0011
011	1001
010	0101
110	0110
111	1010
101	
100	

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000	
001	0011
011	1001
010	0101
110	0110
111	1010
101	1100
100	

Gray codes

A **Gray code** for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only by a 'small change'.

[Torsten Mütze, Combinatorial Gray codes – an updated survey, 2023]

000	
001	0011
011	1001
010	0101
110	0110
111	1010
101	1100
100	

Homogeneous transposition

0100110000101
0100111000001

Homogeneous transposition

Homogeneous transposition

0100110000101 0100111000001	0100110000101 0101110000001
Homogeneous transposition	Non homogeneous transposition

Homogeneous transposition

Homogeneous transposition	Non homogeneous transposition
0100110000101 0100111000001	0100110000101 0101110000001

Let S be a set of same length and same weight binary words.

Definition
A *homogeneous Gray code* for S is a list containing every word of S, such that two consecutive words differ by a homogeneous transposition.

The greedy Gray code algorithm

S : set of same length and same weight binary words

Algorithm

1. Initialize the list $\mathcal{L}$ with a particular word in S.
2. For the last word in $\mathcal{L}$, homogeneously transposes the leftmost possible 1 with the leftmost possible 0, such that the obtained word is in S but not in $\mathcal{L}$.
3. If at point 2. a new word is obtained, then append it to the list $\mathcal{L}$ and return to point 2.

This definition is a specialisation of that introduced in [Aaron Williams, The greedy Gray code algorithm, 2013]

The greedy Gray code algorithm

S : set of same length and same weight binary words

Algorithm

1. Initialize the list $\mathcal{L}$ with a particular word in S.
2. For the last word in $\mathcal{L}$, homogeneously transposes the leftmost possible 1 with the leftmost possible 0, such that the obtained word is in S but not in $\mathcal{L}$.
3. If at point 2. a new word is obtained, then append it to the list $\mathcal{L}$ and return to point 2.

This definition is a specialisation of that introduced in [Aaron Williams, The greedy Gray code algorithm, 2013]

This algorithm is not suitable for efficiently generating Gray codes since it may need to "remember" an exponential number of objects

Example

1001

0101

0011

Example

	0011
1001	1001
0101	0101
0011	0110
	1010
	1100

Example

	0011
1001	1001
0101	0101
0011	0110
	1010
	1100

Example

	0011
1001	1001
0101	0101
0011	0110
	1010
	1100

Questions

- Which classes of binary words this algorithm generates ?

- Which first words generate the whole class ?

- The greedy Gray code algorithm
- Restricted classes of binary words
- Efficient generation

Fibonacci words

Definition
Let $F_n(k)$ be the set of length n and weight k binary words that do not have two consecutive 1's.

$$|F_n(k)| = \binom{n-k+1}{k}.$$

Example:
$F_5(2) = [00101, 01001, 01010, 10001, 10010, 10100]$.

Generalised Dyck prefixes

Definition

Let $C_n(p, k)$ be the set of length n and weight k binary words with the property that any prefix contains at least p times as many 0's as 1's.

$$|C_n(p, k)| = \binom{n}{k} - p\binom{n}{k-1}.$$

Generalised Dyck prefixes

Definition

Let $C_n(p,k)$ be the set of length n and weight k binary words with the property that any prefix contains at least p times as many 0's as 1's.

$$|C_n(p,k)| = \binom{n}{k} - p\binom{n}{k-1}.$$

Example:

- $C_n(0,k)$ is the set of length n binary words of weight k

$$|C_n(0,k)| = \binom{n}{k}$$

Generalised Dyck prefixes

Definition

Let $C_n(p,k)$ be the set of length n and weight k binary words with the property that any prefix contains at least p times as many 0's as 1's.

$$|C_n(p,k)| = \binom{n}{k} - p\binom{n}{k-1}.$$

Example:

- $C_n(0,k)$ is the set of length n binary words of weight k

$$|C_n(0,k)| = \binom{n}{k}$$

- $C_{2n}(1,n)$ is the set of length $2n$ Dyck words

$$|C_n(1,k)| = \frac{1}{n+1}\binom{2n}{n}$$

Generalised Dyck prefixes

Definition

Let $C_n(p,k)$ be the set of length n and weight k binary words with the property that any prefix contains at least p times as many 0's as 1's.

$$|C_n(p,k)| = \binom{n}{k} - p\binom{n}{k-1}.$$

Example:

- $C_n(0,k)$ is the set of length n binary words of weight k

$$|C_n(0,k)| = \binom{n}{k}$$

- $C_{2n}(1,n)$ is the set of length $2n$ Dyck words

$$|C_n(1,k)| = \frac{1}{n+1}\binom{2n}{n}$$

- $C_{3n}(2,n)$ is in bijection with size $3n$ ternary trees.

The set $C_8(1,4)$ of Dyck words of length 8

01010101	00101011
00110101	00110011
00101101	01010011
01001101	01000111
00011101	00100111
00011011	00010111
01001011	00001111

Generators

Definition
For $\alpha \in S$, we denote by $\mathcal{S}(\alpha)$ the list obtained by applying the greedy algorithm for S, starting with α.

Generators

Definition

For $\alpha \in S$, we denote by $\mathcal{S}(\alpha)$ the list obtained by applying the greedy algorithm for S, starting with α. In addition, if $\mathcal{S}(\alpha)$ contains each binary word in S, then α is called a *generator*.

Proposition

The lexicographic smallest possible binary word and the lexicographic largest possible binary word are generators for $F_n(k)$ and for $C_n(p,k)$.

- The greedy Gray code algorithm
- Restricted classes of binary words
- Efficient exhaustive generation

Generation algorithms

Exhaustive generation algorithms are developed in computer science for verification purposes, in statistical physics for computer experimentation, or in bio-informatics to assess statistical significance of weak signals.

An exhaustive generation algorithms is optimal if it runs in constant average time (it is a CAT algorithm).

Recursive tail partitioned lists

The tail of a binary word is its unique suffix of the form $011 \cdots 1$

Recursive tail partitioned lists

The tail of a binary word is its unique suffix of the form $011\cdots1$

Definition

$\mathcal{L}$ is a *recursive tail partitioned* list if it has the form

$$\mathcal{L} = \mathcal{L}_1 \cdot 01^u, \mathcal{L}_2 \cdot 01^{u+1}, \mathcal{L}_3 \cdot 01^{u+2}, \cdots, \mathcal{L}_{\ell+1} \cdot 01^{u+\ell}$$

or the form

$$\mathcal{L} = \mathcal{L}_1 \cdot 01^{u+\ell}, \mathcal{L}_2 \cdot 01^{u+\ell-1}, \mathcal{L}_3 \cdot 01^{u+\ell-2}, \cdots, \mathcal{L}_{\ell+1} \cdot 01^{u}$$

for some $u, \ell \geq 0$, and each list $\mathcal{L}_i$, is in turn recursive tail partitioned.

Recursive tail partitioned lists

Theorem

If $\mathcal{L}$ is a list of same length and same weight binary words and it is

- a homogeneous Gray code, and
- suffix partitioned,

then $\mathcal{L}$ is recursive tail partitioned.

The list $\mathcal{D}_1(01010101)$ obtained by the greedy algorithm with $p = 1$, which is an homogeneous Gray code for $C_8(1, 4)$.

The list $\mathcal{D}_1(01010101)$ obtained by the greedy algorithm with $p = 1$, which is an homogeneous Gray code for $C_8(1, 4)$.

01010101	00101011
00110101	00110011
00101101	01010011
01001101	01000111
00011101	00100111
00011011	00010111
01001011	00001111

The list $\mathcal{D}_1(01010101)$ obtained by the greedy algorithm with $p = 1$, which is an homogeneous Gray code for $C_8(1, 4)$.

01010101	00101011
00110101	00110011
00101101	01010011
01001101	01000111
00011101	00100111
00011011	00010111
01001011	00001111

The list $\mathcal{D}_1(01010101)$ obtained by the greedy algorithm with $p = 1$, which is an homogeneous Gray code for $C_8(1, 4)$.

01010101	00101011
00110101	00110011
00101101	01010011
01001101	01000111
00011101	00100111
00011011	00010111
01001011	00001111

The list $\mathcal{D}_1(01010101)$ obtained by the greedy algorithm with $p = 1$, which is an homogeneous Gray code for $C_8(1, 4)$.

01010101	00101011
00110101	00110011
00101101	01010011
01001101	01000111
00011101	00100111
00011011	00010111
01001011	00001111

The list $\mathcal{D}_1(01010101)$ obtained by the greedy algorithm with $p = 1$, which is an homogeneous Gray code for $C_8(1, 4)$.

01010101	00101011
00110101	00110011
00101101	01010011
01001101	01000111
00011101	00100111
00011011	00010111
01001011	00001111

The list $\mathcal{D}_1(01010101)$ obtained by the greedy algorithm with $p = 1$, which is an homogeneous Gray code for $C_8(1, 4)$.

01010101	00101011
00110101	00110011
00101101	01010011
01001101	01000111
00011101	00100111
00011011	00010111
01001011	00001111

CAT generation for a homogeneous Gray code for $C_n(p,k)$

procedure pref(m, j)
  if $m = (p+1)j$ **then**
    if $p = 0$ **then return**
    end if
    $m \leftarrow m - 1$; $j \leftarrow j - 1$
  end if
  if $S_j < m$ **then** *# Increasing tail*
    for $i = 0$ **to** $j - 1$ **do** *# i is the number of 1's in the tail*
      pref$(m - i - 1, j - i)$
      $S_{j-i} \leftarrow m - i$
      print(S)
  end if
  if $S_j = m$ **then** *# Decreasing tail*
    for $i = j - 1$ **downto** 0 **do** *# i is the number of 1's in the tail*
      $S_{j-i} \leftarrow \max(S_{j-i-1} + 1, (p+1)(j-i))$
      print(S)
      pref$(m - i - 1, j - i)$
  end if
end procedure

- the main call is pref(n, k), it generates $C_n(p, k)$
- S_i is the position of the ith 1 in the word
- table S and the parameter p are global
- S is initialized as $S_i \leftarrow (p+1)i$ for $1 \leq i \leq k$

Algorithm analysis

With a classical complexity analysis, we can obtain the following result

Proposition

The call pref(n, k) generates the homogeneous greedy Gray code for $C_n(p, k)$ efficiently.

See [Frank Ruskey, Combinatorial Generation Book].

Biblio

A. Bultena and F. Ruskey.
An Eades-McKay algorithm for well-formed parentheses strings.
Information Processing Letters, 68 :255–259, 1998.

T. Mütze.
Combinatorial Gray codes – an updated survey.
Electronic Journal of Combinatorics, (Dynamic Survey DS26), 2023.

F. Ruskey.
Combinatorial Generation.
Book in preparation.

A. Williams.
The greedy Gray code algorithm.
In *Algorithms and Data Structures*, page 525–536, Berlin, Heidelberg, 2013.

D. Wong and V. Vajnovszki.
Greedy Gray codes for Dyck words and ballot sequences.
In *COCOON 2023*, 2023.

Thank you !