Gray codes — from old to new order relations

Vincent VAJNOVSZKI

Université de Bourgogne
Le2i, UMR-CNRS 5158

18 June

From Gray codes to order relations and vice versa

From Gray codes to order relations and vice versa

- combinations
- Fibonacci words
- Lucas words
- Lyndon words, necklaces

- Algorithmic and graph theoretic issues

- FRANK GRAY, Pulse code communication, U. S. Patent 2632058 (1953)

F. Gray described in a patent that was awarded in 1953 a scheme for listing n-bit words so that successive words differ in just one bit. It was designed for analog to digital conversion

0 0 0 0	**1** 1 0 0
0 0 0 1	**1** 1 0 1
0 0 1 1	**1** 1 1 1
0 0 1 0	**1** 1 1 0
0 1 1 0	**1** 0 1 0
0 1 1 1	**1** 0 1 1
0 1 0 1	**1** 0 0 1
0 1 0 0	**1** 0 0 0

$$\mathcal{B}_n = \begin{cases} \epsilon & \text{if} \quad n = 0, \\ 0 \cdot \mathcal{B}_{n-1} \circ 1 \cdot \overline{\mathcal{B}}_{n-1} & \text{if} \quad n > 0 \end{cases}$$

b	b in binary	g=BRGC of b
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

- H.S. WILF, Combinatorial algorithms: An update, *CBNS, SIAM*, 55 (1989)

  'A *combinatorial* (or *generalized*) *Gray code* is an ordered list of combinatorial objects such that successive objects differ by a small pre-specified amount'

- T. WALSH, Generating Gray codes in $O(1)$ worst-case time per word. Proceeding of 4th Conference on Discrete Mathematics and Theoretical Computer Science, *LNCS*, (2003)

  'A *Gray code* is an infinite set of word-lists with unbounded word-length such that the Hamming distance between any two successive words in any list is bounded independently of the word-length'

- C. SAVAGE, A Survey of Combinatorial Gray Codes, *SIAM Rev.* **39** 4 (1997)

> ‘Since bijections are known between most members of the Catalan family, a Gray code for one member of the family gives implicitly a listing scheme for every other member of the family. However, the resulting list may not look like Gray codes, since bijections need not preserve minimal changes between elements’

Now it has many applications as:

- interconnection network
- addressing microprocessors
- hashing algorithms
- distributed systems
- detecting-correcting codes
- solving puzzles (Tower of Hanoi, Brain, Chinese-Rings, Spin-Out)
- circuit testing
- analog-digital converters and signal encoding
- data compression
- graphics and image processing
- Hamiltonian circuits in hypercubes
- Cayley graphs of Coxeter groups
- continuous space-filling curves
- classification of Venn diagrams
- and by applying the binary Gray code, a variety of problems have been solved and the complexities of the solutions to other problems have been improved

However, the code's central idea existed centuries before !

• In the 17th-century England, bell ringers had the delightful custom of ringing all the bells in all possible permutations, in a Gray code manner. In spite of these early applications, the code has only been widely know since the advent of computers

The *Cambridge Forty-Eight:*

1	2	2	2	2	2	2	2	...
2	1	1	1	5	5	3	3	...
3	3	3	5	1	3	5	1	...
4	4	5	3	3	1	1	5	...
5	5	4	4	4	4	4	4	...

• In 1872, Louis Gros, a French law clerk, published a brochure presenting this code as a solution to the “Chinese-Rings” puzzle

0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 1 0 0
1 1 0 1
1 1 1 1

• According to Heath, the code was used by Émile Baudot for telegraphy in the 1870's

Binary Gray codes

- H.S. WILF, Combinatorial algorithms: An update, *CBNS, SIAM*, **55** (1989)
- M.W. BUNDER, K.P. TOGNETTI, G.E. WHEELER, On binary reflected Gray codes and functions, *Discrete Math.* **308** (2008)

$$unrank : \{0,1\}^n \to \{0,1\}^n$$

$unrank : \{0,1\}^n \to \{0,1\}^n$

Theorem Let $g_n g_{n-1} \dots g_0$ be the b-th sequence of length n in BRGC, $0 \leq b \leq 2^n - 1$, $b = b_n b_{n-1} \dots b_0$

- $g_i = (b + 2^i)_{i+1}$
- $g_i = \lfloor b/(2^{i+1}) + \frac{1}{2} \rfloor \mod 2$
- $g_i = (b_{i+1} + b_i) \mod 2$

Algorithm:

```
short binaryToGray(short b)
{ return (b>>1)^ b; }
```

Algorithm:

g is *b* in binary, with the first of any sequence of 0s or 1s becoming a 1 and every other digit a 0;

$$rank : \{0,1\}^n \longrightarrow \{0,1\}^n$$

$rank : \{0,1\}^n \to \{0,1\}^n$

Theorem Let $g_n g_{n-1} \dots g_0$ be the b-th sequence of length n in BRGC, $0 \leq b \leq 2^n - 1$, $b = b_n b_{n-1} \dots b_0$

- $b_i = \left(\sum_{j=i}^{n} g_i \right) \mod 2$
- $b_i = \begin{cases} g_n & \text{if} \quad i = n \\ (g_i + b_{i+1}) \mod 2 & \text{elsewhere} \end{cases}$

b	b in binary	g=BRGC of b	
0	0	0	0
1	1	1	1
2	1 0	1 1	3
3	1 1	1 0	2
4	1 0 0	1 1 0	6
5	1 0 1	1 1 1	7
6	1 1 0	1 0 1	5
7	1 1 1	1 0 0	4
8	1 0 0 0	1 1 0 0	12
9	1 0 0 1	1 1 0 1	13
10	1 0 1 0	1 1 1 1	15
11	1 0 1 1	1 1 1 0	14
12	1 1 0 0	1 0 1 0	10
13	1 1 0 1	1 0 1 1	11
14	1 1 1 0	1 0 0 1	9
15	1 1 1 1	1 0 0 0	8

Restricted binary words and order relations

Definition The set of length n binary words with exactly m occurrences of 1 is denoted by $C_{n,m}$ and these words code m-combinations of an n-element set

1 1 1 0 0	1 2 3
0 1 1 0 1	2 3 5

Restricted binary words and order relations

Definition The set of length n binary words with exactly m occurrences of 1 is denoted by $C_{n,m}$ and these words code m-combinations of an n-element set

1 1 1 0 0	1 2 3
0 1 1 0 1	2 3 5

1 1 1 0 0	1 2 3
1 1 0 0 1	1 2 5

- A. NIJENHUIS, H.S. WILF. *Combinatorial Algorithms for Computers and Calculators*. Academic Press, 1978

Theorem The list defined by

$$\mathcal{C}_{n,m} = \begin{cases} \epsilon & \text{if} \quad n = 0 \\ 0^n & \text{if} \quad m = 0 \\ 1^n & \text{if} \quad n = m \\ 0 \cdot \mathcal{C}_{n-1,m} \circ 1 \cdot \overline{\mathcal{C}}_{n-1,m-1} & \text{if} \quad n > m > 0 \end{cases}$$

is a 2-Gray code for $C_{n,m}$ (and so, an optimal Gray code).

• A. NIJENHUIS, H.S. WILF. *Combinatorial Algorithms for Computers and Calculators*. Academic Press, 1978

Theorem The list defined by

$$\mathcal{C}_{n,m} = \begin{cases} \epsilon & \text{if} \quad n = 0 \\ 0^n & \text{if} \quad m = 0 \\ 1^n & \text{if} \quad n = m \\ 0 \cdot \mathcal{C}_{n-1,m} \circ 1 \cdot \overline{\mathcal{C}}_{n-1,m-1} & \text{if} \quad n > m > 0 \end{cases}$$

is a 2-Gray code for $C_{n,m}$ (and so, an optimal Gray code).

$$\mathcal{B}_n = \begin{cases} \epsilon & \text{if} \quad n = 0, \\ 0 \cdot \mathcal{B}_{n-1} \circ 1 \cdot \overline{\mathcal{B}}_{n-1} & \text{if} \quad n > 0 \end{cases}$$

$\mathcal{B}_4$	$\mathcal{C}_{4,2}$
0 0 0 0	
0 0 0 1	
0 0 1 1	0 0 1 1
0 0 1 0	
0 1 1 0	0 1 1 0
0 1 1 1	
0 1 0 1	0 1 0 1
0 1 0 0	
1 1 0 0	1 1 0 0
1 1 0 1	
1 1 1 1	
1 1 1 0	
1 0 1 0	1 0 1 0
1 0 1 1	
1 0 0 1	1 0 0 1
1 0 0 0	

Definition The lexicographical order on $\{0,1\}^n$ is defined as: $x = x_1 x_2 \dots x_n$ is less than $y = y_1 y_2 \dots y_n$ iff $x_k = 0$ and $y_k = 1$, where k is the leftmost position with $x_k \neq y_k$

Definition The lexicographical order on $\{0,1\}^n$ is defined as: $x = x_1 x_2 \dots x_n$ is less than $y = y_1 y_2 \dots y_n$ iff $x_k = 0$ and $y_k = 1$, where k is the leftmost position with $x_k \neq y_k$

Definition x is less than y in*reflected order*, denoted by $x < y$, if $\sum_{j=1}^{k} x_j$ is even (and $\sum_{j=1}^{k} y_j$ is odd), where k is the leftmost position with $x_k \neq y_k$

Theorem The reflected order on $\{0,1\}^n$ induces the BRGC

Theorem The reflected order on $\{0,1\}^n$ induces the BRGC

0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0

Theorem The *reflected order* $<$ induced a 2-Gray code on the set $C_{n,m}$

Theorem The *reflected order* $<$ induced a 2-Gray code on the set $C_{n,m}$

Theorem The *reflected Gray code order* $<$ induced a 4-Gray code (not minimal) on the set D_{2n}, of length $2n$ Dyck words

Definition We say that x is less than y in *dual reflected order*, denoted by $x \prec y$, if $x_1 x_2 \ldots x_k$, the length k prefix of x contains an odd number of 0's, where k is the leftmost position with $x_k \neq y_k$

$$x \prec y \text{ iff } x^c > y^c$$

Definition We say that x is less than y in *dual reflected order*, denoted by $x \prec y$, if $x_1 x_2 \dots x_k$, the length k prefix of x contains an odd number of 0's, where k is the leftmost position with $x_k \neq y_k$

$$x \prec y \text{ iff } x^c > y^c$$

Theorem The *dual reflected order* $\prec$ induced a 1-Gray code on the set $\{0,1\}^n$ and a 2-Gray code on the set $C_{n,m}$

(Generalized) Fibonacci words

- V. V., A Loopless Generation of Bitstrings without p consecutive ones, *Discrete Math. and Theoretical Computer Science*, Springer, (2001)

Definition $F_n^{(p)}$ denotes the set of length n binary words such that there are no p consecutive 1s

$$card(F_n^{(p)}) = f_{n+p}^{(p)}$$

where $f_n^{(p)}$ is the pth *order* nth *Fibonacci number*

$$f_n^{(p)} = \begin{cases} 0 & \text{if} \quad 0 \leq n < p-1, \\ 1 & \text{if} \quad n = p-1, \\ \sum_{j=n-p}^{n-1} f_j^{(p)} & \text{if} \quad n \geq p, \end{cases}$$

$F_5^{(2)}$
0 0 0 0 0
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
0 0 1 0 1
0 1 0 0 0
0 1 0 0 1
0 1 0 1 0
1 0 0 0 1
1 0 0 0 0
1 0 0 1 0
1 0 1 0 0
1 0 1 0 1

$F_5^{(2)}$	$F_4^{(3)}$
0 0 0 0 0	0 0 0 0
0 0 0 0 1	0 0 0 1
0 0 0 1 0	0 0 1 0
0 0 1 0 0	0 0 1 1
0 0 1 0 1	0 1 0 0
0 1 0 0 0	0 1 0 1
0 1 0 0 1	0 1 1 0
0 1 0 1 0	1 0 0 0
1 0 0 0 1	1 0 0 1
1 0 0 0 0	1 0 1 0
1 0 0 1 0	1 0 1 1
1 0 1 0 0	1 1 0 0
1 0 1 0 1	1 1 0 1

Theorem The dual reflected order $\prec$ induced a 1-Gray code on the set $F_n^{(p)}$

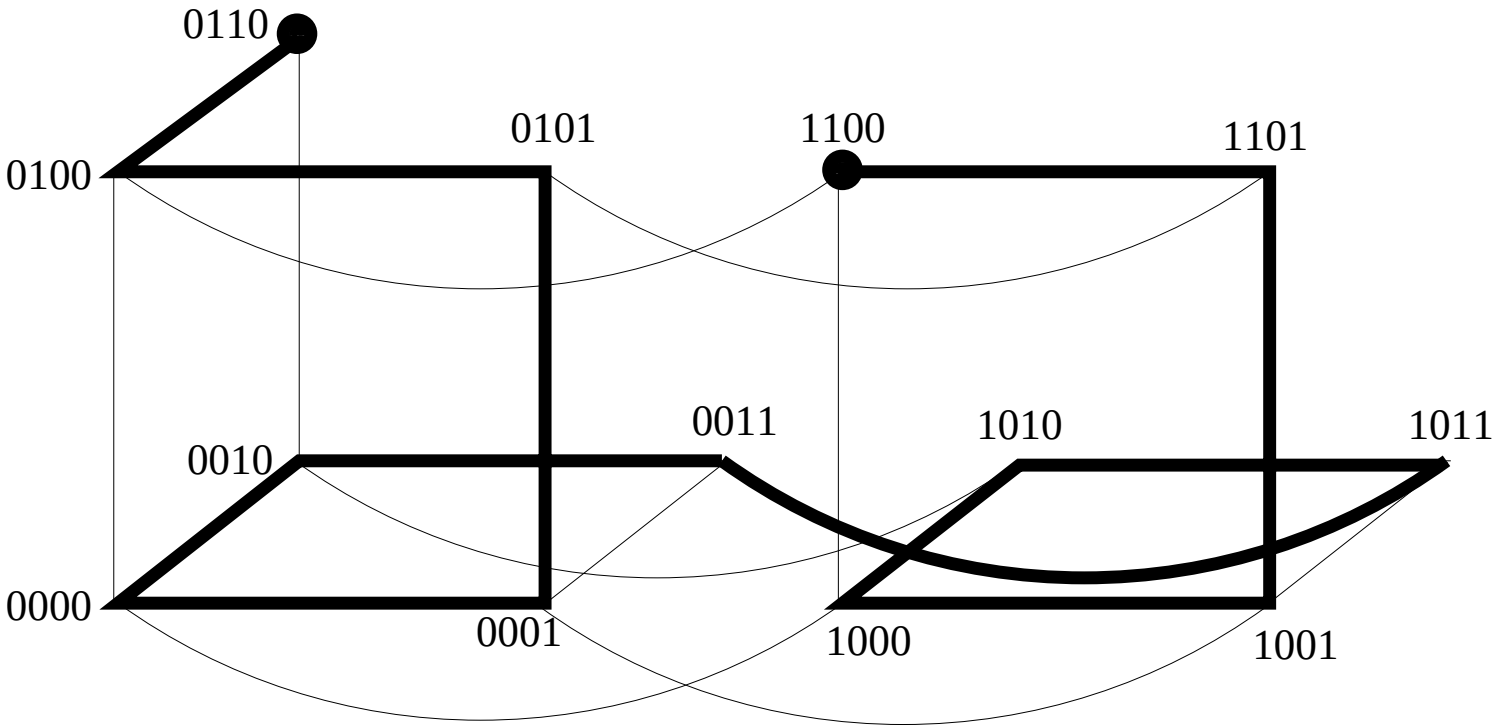
Theorem The dual reflected order $\prec$ induced a 1-Gray code on the set $F_n^{(p)}$

Theorem The dual reflected order $\prec$ on the set $F_n^{(p)}$ defines the list

$$\mathcal{F}_n^{(p)} = \begin{cases} \mathcal{C}_n & \text{if} \quad 0 \leq n < p \\ 0 \cdot \overline{\mathcal{F}}_{n-1}^{(p)} \circ 10 \cdot \overline{\mathcal{F}}_{n-2}^{(p)} \circ \cdots \circ 1^{p-1}0 \cdot \overline{\mathcal{F}}_{n-p}^{(p)} & \text{if} \quad n \geq p \end{cases}$$

$\mathcal{F}_4^{(3)}$

0 1 1 0
0 1 0 0
0 1 0 1
0 0 0 1
0 0 0 0
0 0 1 0
0 0 1 1
1 0 1 1
1 0 1 0
1 0 0 0
1 0 0 1
1 1 0 1
1 1 0 0


0110
0100
0101
1100
1101
0011
1010
1011
0010
0000
0001
1000
1001

$Q(F_4^{(3)})$

(Generalized) Lucas words

- V. V. AND JEAN-LUC BARIL, Minimal change list for Lucas strings and some graph theoretic consequences, *TCS*, 346 (2005)

Definition $L_n^{(p)}$ is the set of length-n binary strings such that there are no 1^p factors if strings are regarded circularly, i.e., the last entry of a string is followed by the first one

The set $L_4^{(2)}$
0 0 0 0
0 0 0 1
0 0 1 0
0 1 0 0
0 1 0 1
1 0 0 0
1 0 1 0

The set $L_4^{(2)}$
0 0 0 0
0 0 0 1
0 0 1 0
0 1 0 0
0 1 0 1
1 0 0 0
1 0 1 0

The set $L_4^{(3)}$	weight
0 0 0 0	even
0 0 0 1	odd
0 0 1 0	odd
0 0 1 1	even
0 1 0 0	odd
0 1 0 1	even
0 1 1 0	even
1 0 0 0	odd
1 0 0 1	even
1 0 1 0	even
1 1 0 0	even

Theorem Let $\lambda_p(z)$ the generating function for the parity difference integer sequence.

$$\begin{aligned}\lambda_p(z) &= \frac{p+1}{1-(-z)^{p+1}} - \frac{1}{1+z} \\ &= (p+1)\cdot\sum_{k\geq 0}(-z)^{k(p+1)} - \sum_{n\geq 0}(-z)^n\end{aligned}$$

Corollary The parity difference integer sequence corresponding to Lucas strings satisfies

$$\lambda_{n,p} = \begin{cases} (-1)^{n+1} & \text{if} \quad (p+1) \not| n, \\ (-1)^n \cdot p & \text{if} \quad (p+1)|n \end{cases}$$

Theorem If $(p+1) \not| n$ then $\prec$ induces a 1-Gray code on $L_n^{(p)}$

Theorem If $(p+1) \not| n$ then $\prec$ induces a 1-Gray code on $L_n^{(p)}$

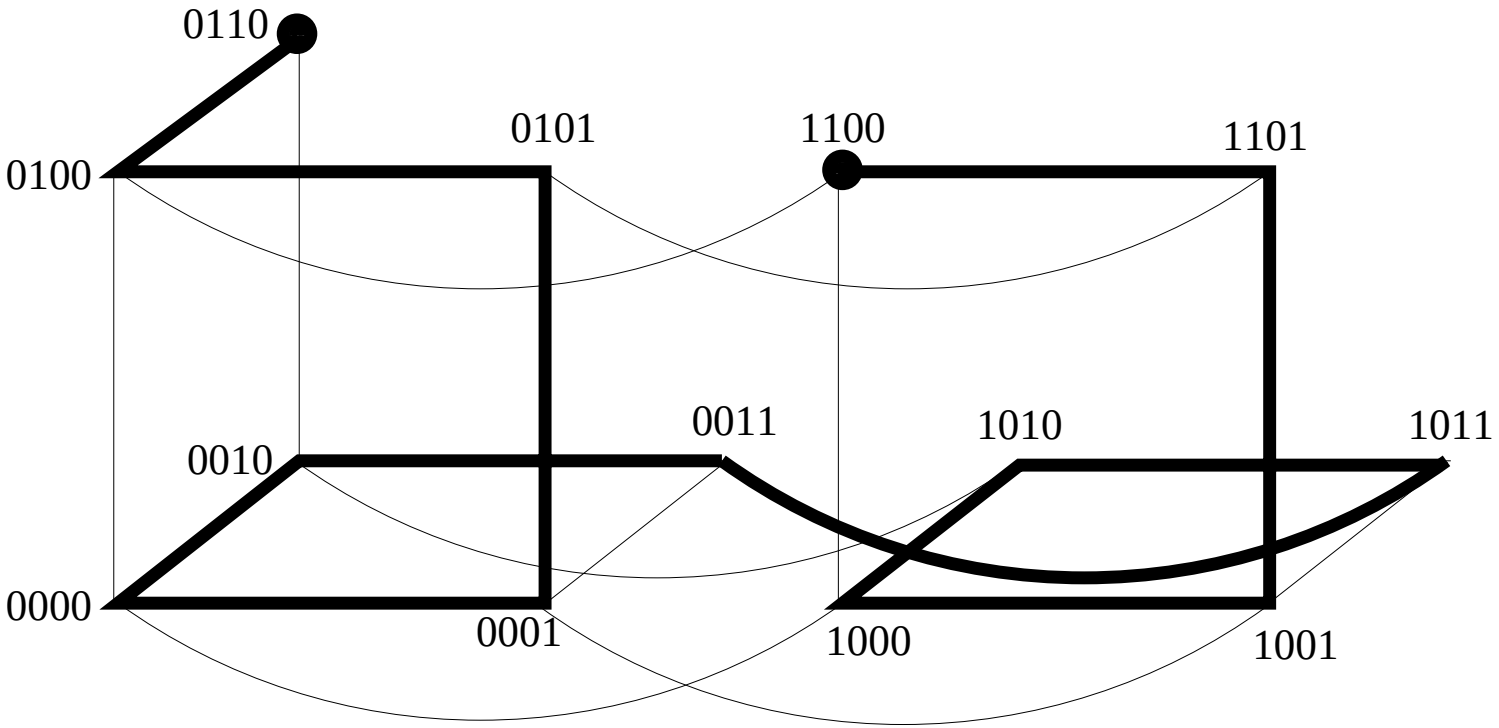
The list $\mathcal{L}_4^{(2)}$
0 1 0 0
0 1 0 1
0 0 0 1
0 0 0 0
0 0 1 0
1 0 1 0
1 0 0 0

Theorem If $(p+1)|n$ then $\prec$ induces a 2-Gray code on $L_n^{(p)}$ there are exactly $p-1$ words x with $d(x, succ(x)) = 2$

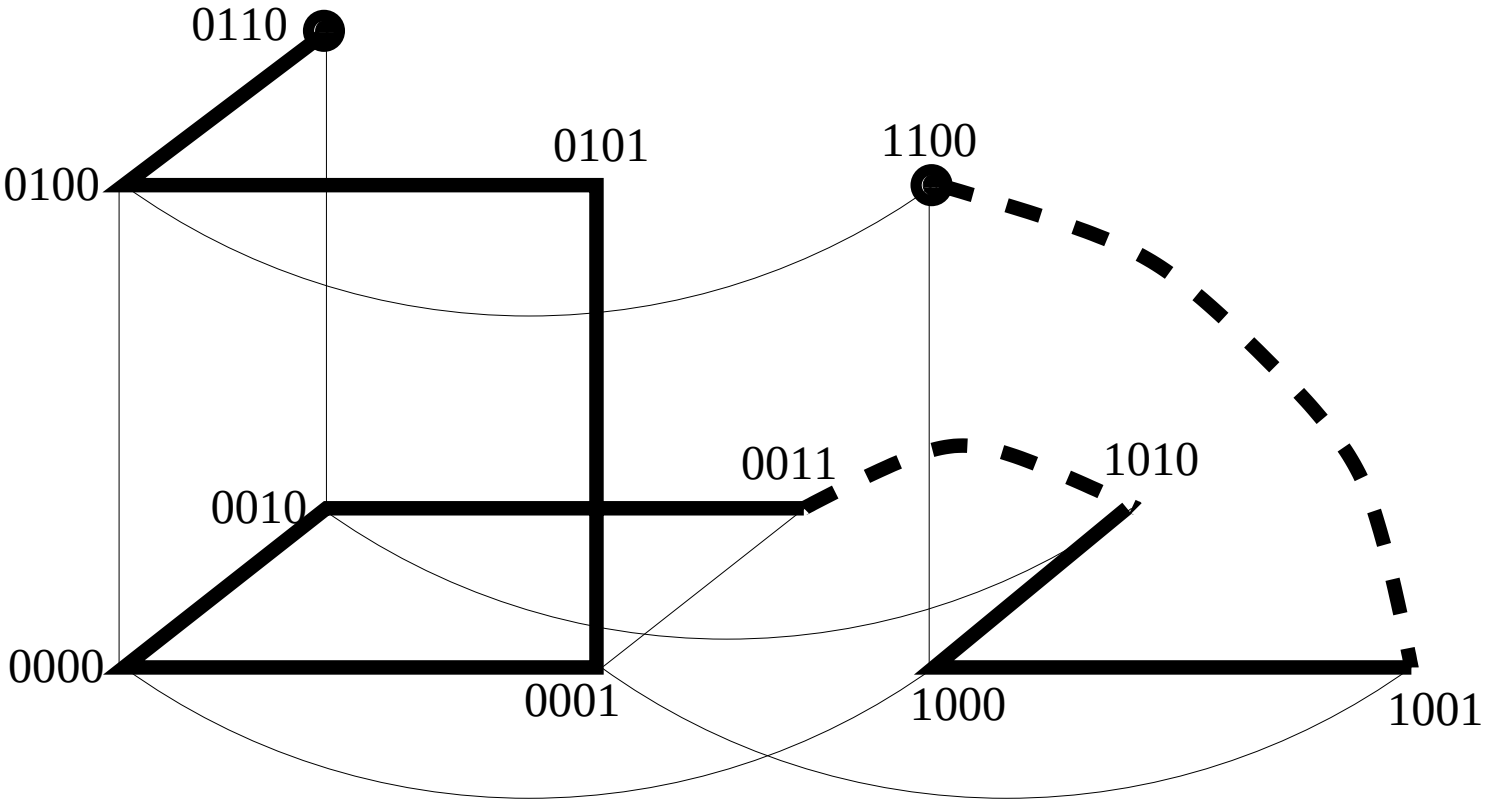
Theorem If $(p+1)|n$ then $\prec$ induces a 2-Gray code on $L_n^{(p)}$ there are exactly $p-1$ words x with $d(x, succ(x)) = 2$

The list $\mathcal{L}_4^{(3)}$
0 1 1 0
0 1 0 0
0 1 0 1
0 0 0 1
0 0 0 0
0 0 1 0
0 0 1 1
1 0 1 **0**
1 0 0 0
1 0 0 1
1 **1** 0 **0**

Lemma Let $succ(x)$ denote the successor of $x \in L_n^{(p)}$ in the list $\mathcal{L}_n^{(p)}$ and $s(x)$ the successor of $x \in F_n^{(p)}$ in the list $\mathcal{F}_n^{(p)}$. Then $succ(x)$ is either $s(x)$, $s^2(x) = s(s(x))$ or $s^3(x) = s(s(s(x)))$


0110
0100
0101
1100
1101
0011
1010
1011
0010
0000
0001
1000
1001

$Q(F_4^{(3)})$



$Q(L_4^{(3)})$

Corollary If $(p+1)|n$ then

1. The minimal number of paths covering $Q(L_n^{(p)})$ is p
2. The length of the maximal path in $Q(L_n^{(p)})$ is $card(L_n^{(p)}) - (p+1)$

When a graph does not have a Hamiltonian path it may be desirable to visit each vertex but not necessarily once, such that the Hamming distance between two successive vertices is one

A graph is in the class $\mathcal{H}(s,t)$ if it has a path that visits every vertex at least s times and at most t times, and such a path is called $\mathcal{H}(s,t)$-path

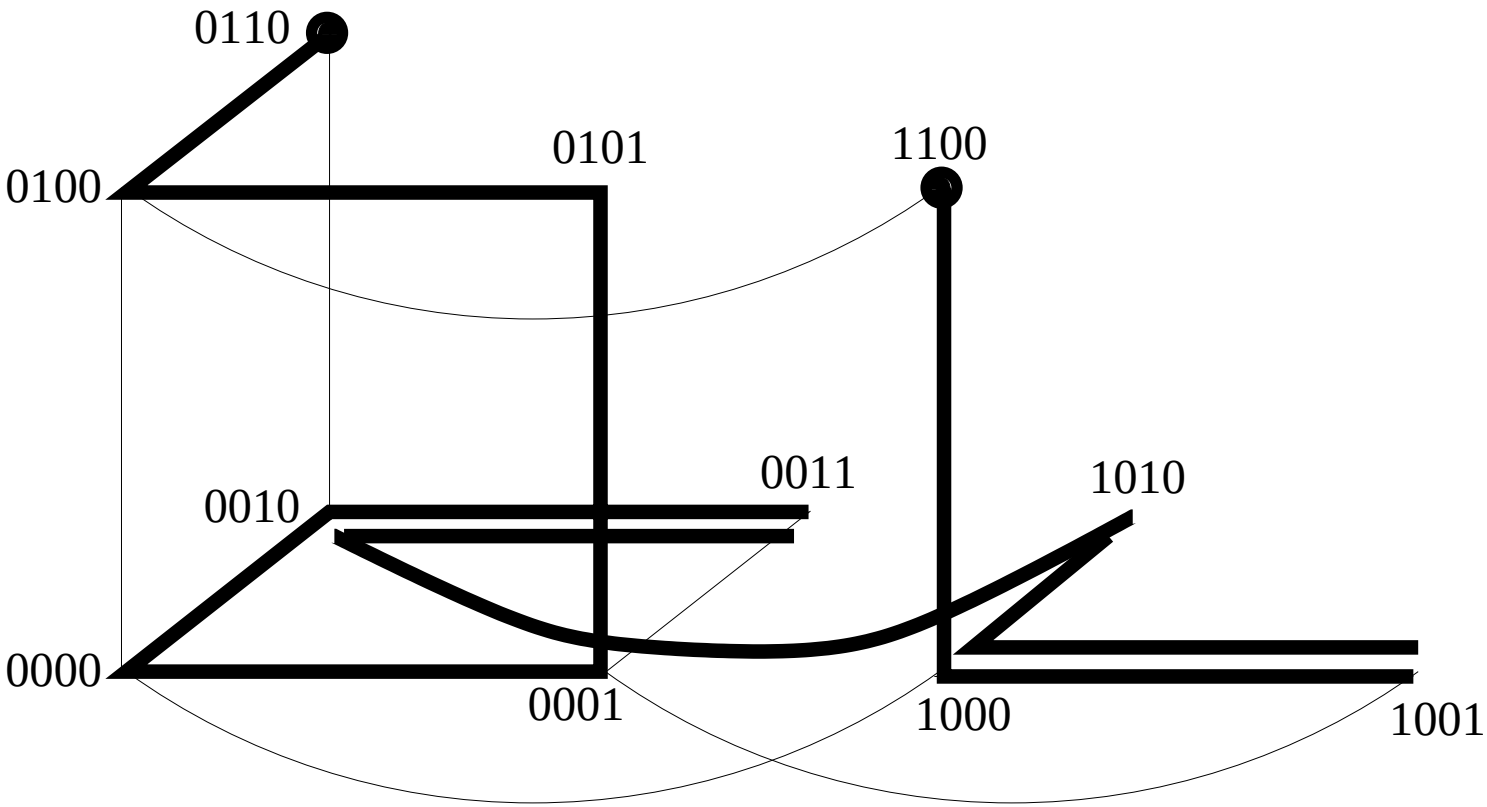
Thus a graph is in $\mathcal{H}(1,1)$ exactly if it is Hamiltonian. In this context, we have

When a graph does not have a Hamiltonian path it may be desirable to visit each vertex but not necessarily once, such that the Hamming distance between two successive vertices is one

A graph is in the class $\mathcal{H}(s,t)$ if it has a path that visits every vertex at least s times and at most t times, and such a path is called $\mathcal{H}(s,t)$-path

Thus a graph is in $\mathcal{H}(1,1)$ exactly if it is Hamiltonian. In this context, we have

Corollary If $(p+1)|n$ then $Q(L_n^{(p)})$ is in $\mathcal{H}(1,2)$



$Q(L_4^{(3)})$

Lyndon words

- V. V., Gray code order for Lyndon words, *DMTCS*, **9**(2), 2007
- V. V., More restrictive Gray codes for necklaces and Lyndon words, *IPL*, **106**(3), 2008

Lyndon words

- V. V., Gray code order for Lyndon words, *DMTCS*, **9**(2), 2007
- V. V., More restrictive Gray codes for necklaces and Lyndon words, *IPL*, **106**(3), 2008

The conjugacy class of a string $x = uv$ is the set of all strings vu

- A *necklace* is a binary string which is minimal with respect to lexicographic order in its conjugacy class
- A *Lyndon word* is an aperiodic necklace

- A *pre-necklace* is a binary string which is the prefix of a some necklace, or equivalently, of some Lyndon word
- An *unlabeled necklace* is a word which is lexicographically minimal under both rotation and permutation of alphabet symbols
- An *unlabeled Lyndon word* is an aperiodic unlabeled necklace

- A *pre-necklace* is a binary string which is the prefix of a some necklace, or equivalently, of some Lyndon word
- An *unlabeled necklace* is a word which is lexicographically minimal under both rotation and permutation of alphabet symbols
- An *unlabeled Lyndon word* is an aperiodic unlabeled necklace

$$\begin{array}{ccccc} UL_n & \subset & UN_n & & \\ \cap & & \cap & & \\ L_n & \subset & N_n & \subset & PN_n \end{array}$$

PN_5	N_5	L_5
0 1 1 1 1	0 1 1 1 1	0 1 1 1 1
0 1 1 1 0		
0 1 1 0 1		
0 1 0 1 0		
0 1 0 1 1	0 1 0 1 1	0 1 0 1 1
0 0 0 1 1	0 0 0 1 1	0 0 0 1 1
0 0 0 1 0		
0 0 0 0 0	0 0 0 0 0	
0 0 0 0 1	0 0 0 0 1	0 0 0 0 1
0 0 1 0 1	0 0 1 0 1	0 0 1 0 1
0 0 1 0 0		
0 0 1 1 0		
0 0 1 1 1	0 0 1 1 1	0 0 1 1 1
1 1 1 1 1	1 1 1 1 1	

Theorem The $\prec$ order yields a 3-Gray code on the sets of pre-necklaces, necklaces and Lyndon words of length n

Theorem The $\prec$ order yields a 3-Gray code on the sets of pre-necklaces, necklaces and Lyndon words of length n

Definition Let $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$ be words in $\{0,1\}^n$ and k the rightmost position in which x and x differ. We say that x is less than y in $<^*$ order if $\sum_{j=k}^{n} x_j$ is even

0 0 0 0
1 0 0 0
1 1 0 0
0 1 0 0
0 1 1 0
1 1 1 0
1 0 1 0
0 0 1 0
0 0 1 1
1 0 1 1
1 1 1 1
0 1 1 1
0 1 0 1
1 1 0 1
1 0 0 1
0 0 0 1

Theorem

1. $<^*$ order induces a cyclic 2-Gray code on N_n and L_n
2. $<^*$ order induces a cyclic 3-Gray code on UN_n and UL_n

Theorem

1. $<^*$ order induces a cyclic 2-Gray code on N_n and L_n
2. $<^*$ order induces a cyclic 3-Gray code on UN_n and UL_n

N_6	L_6	UN_6	UL_6
0 0 0 0 0 0		✓	
0 0 0 0 1 1	✓	✓	✓
0 1 1 0 1 1			
0 0 1 0 1 1	✓	✓	✓
0 0 1 1 1 1	✓		
1 1 1 1 1 1			
0 1 1 1 1 1	✓		
0 1 0 1 1 1	✓		
0 0 0 1 1 1	✓	✓	✓
0 0 0 1 0 1	✓	✓	✓
0 1 0 1 0 1		✓	
0 0 1 1 0 1	✓		
0 0 1 0 0 1		✓	
0 0 0 0 0 1	✓	✓	✓

Question 1 Can the Gray codes presented here be efficiently implemented? That is, can N_n, L_n, UN_n and UL_n be efficiently listed in $<^*$ order?

Question 2 Can the previous results be extended to k-ary alphabet?

Loopless generation

When an application requires an exhaustive examination of all the objects in a combinatorial class, Gray codes can be used to speed up the task

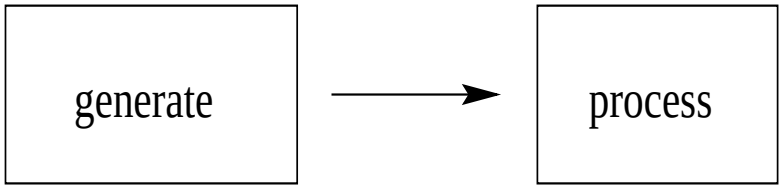
Indeed, with a Gray code scheme, it is possible to list m objects, each of size n, in time $O(m+n)$ rather than $O(m \times n)$, by listing the first object and thereafter listing only the (constant size) change between successive objects

Moreover, often Gray code definitions involve elegant recursive constructions which provide new insights into the structure of the combinatorial class

If the number of changes made in transforming one word to the next on the list is $O(1)$, can the transformation be made in $O(1)$ time even in the worst case?

If the number of changes made in transforming one word to the next on the list is $O(1)$, can the transformation be made in $O(1)$ time even in the worst case?

A generating algorithm is *loopless* (or *loop-free*) if after the initial object is generated, each succeeding object may be obtained by at most a fixed number of operations. A constant delay between outputs is particularly desirable in applications where the output of one computation serves as input to another.



A loopless generating algorithm produces a Gray code list. The converse is not true: Gray codes can be generated by algorithms that are not loopless

0	0 0 0 0	$\emptyset$
1	0 0 0 **1**	$\{4\}$
2	0 0 **1** 1	$\{3,4\}$
3	0 0 1 **0**	$\{3\}$
4	0 **1** 1 0	$\{2,3\}$
5	0 1 1 **1**	$\{2,3,4\}$
6	0 1 **0** 1	$\{2,4\}$
7	0 1 0 **0**	$\{2\}$
8	**1** 1 0 0	$\{1,2\}$
9	1 1 0 **1**	$\{1,2,4\}$
10	1 1 **1** 1	$\{1,2,3,4\}$
11	1 1 1 **0**	$\{1,2,3\}$
12	1 **0** 1 0	$\{1,3\}$
13	1 0 1 **1**	$\{1,3,4\}$
14	1 0 **0** 1	$\{1,3\}$
15	1 0 0 **0**	$\{1\}$

```
while (true)
     if rank is even
     then b_n := 1 − b_n;
     else k:=the index of last 1 bit in b;
          if k > 1
          then b_{k−1} := 1 − b_{k−1};
          else exit;
          end if
     end if
     rank := rank + 1;
end while
```

Essentially, there are two techniques to design loopless generating algorithms:

- Ruskey's method by *'finished and unfinished lists'*
- and Ehrlich's *'e'* array
- G. EHRLICH, Loopless algorithms for generating permutations, combinations, and other combinatorial objects, J. ACM, **20** (1973)

Essentially, there are two techniques to design loopless generating algorithms:

- Ruskey's method by *'finished and unfinished lists'*
- and Ehrlich's *'e'* array

- G. EHRLICH, Loopless algorithms for generating permutations, combinations, and other combinatorial objects, J. ACM, **20** (1973)

- which was very greatly generalized by T.R. Walsh

- T. WALSH, Loop-free sequencing of bounded integer compositions, J. Combin. Math. Combin. Comput., **33**,(2000)

- T. WALSH, Generating Gray codes in $O(1)$ worst-case time per word. Proceeding of 4th Conference on Discrete Mathematics and Theoretical Computer Science, *LNCS* (2003)

A word-list where all the words with the same prefix are consecutive is *genlex*.

- T. WALSH, Generating Gray codes in $O(1)$ worst-case time per word. Proceeding of 4th Conference on Discrete Mathematics and Theoretical Computer Science, *LNCS* (2003)

A word-list where all the words with the same prefix are consecutive is *genlex*.
Walsh gives a very general method to design a loopless generating algorithms for a *genlex* word-list if it satisfies:
For any length k prefix, let L be the sub-list of words with this prefix

- L sub-list contains only one word, or
- in L the $(k+1)$-th letter assumes at least two values

A word-list which does not obey to Walsh's criteria

$\vdots$
c b c d
a b c d
a b c e
d b c e

- D. KNUTH, The Art of Computer Programming, Volume 4
- F. RUSKEY, *Combinatorial generation*. Book in preparation
- A. BERNINI, E. GRAZZINI, E. PERGOLA, R. PINZANI, A general exhaustive generation for Gray structures, *Acta Informatica*, 2007, **44**(5)
- R.W. DORAN, The Gray code, *Research Report*