Computer Science Journal of Moldova, vol.**11**, no.2(32), 2003

Gray Code for Cayley Permutations

J.-L. Baril

Abstract

A length-n Cayley permutation p of a total ordered set S is a length-n sequence of elements from S, subject to the condition that if an element x appears in p then all elements $y < x$ also appear in p. In this paper, we give a Gray code list for the set of length-n Cayley permutations. Two successive permutations in this list differ at most in two positions.

Keywords : Weak-order, Gray Code, Permutations, Combinations.

1 Introduction and definitions

A Gray code for a class of combinatorial objects is an *ordered* list for the objects of the class such that two successive objects differ in a '*small prespecified way*', see for example Carla Savage [Sav89].

Let $S = \{a_1, \ldots, a_n\}$ be a total ordered set with $a_1 < a_2 < \ldots < a_n$. A length-n Cayley permutation (C-permutation) p of S is a length-n sequence $p = p_1\ p_2\ \ldots p_n$ of elements from S satisfying the following property: if for any i, a_i appears in p, then all elements a_j, $j < i$, also appear in p. In fact, S may contain more than n elements, but in any C-permutation of length-n only the first n elements of S can appear. For example, if S is the set of natural numbers, then there are thirteen C-permutations of length three: 000, 001, 011, 012, 010, 021, 101, 102, 100, 201, 110, 120, 210. Without any loss of generality we will consider only the C-permutations of the set of natural numbers.

Cayley permutations have many interesting combinatorial interpretations; some of the more natural ones are the weak-orders. Recall that a weak-order is a relation $\leq$ that is transitive (if $x \leq y$ and $y \leq$

z then $x \leq z$) and complete ($x \leq y$ or $y \leq x$ always holds). We can write $x \equiv y$ if $x \leq y$ and $y \leq x$, and we note $x < y$ if $x \leq y$ and $y \not\leq x$. There exists a one-to-one map between C-permutations of length-n and the weak-orders on n elements. Indeed, a C-permutation $p = p_1\ p_2\ \ldots\ p_n$ of length-n can represent the weak-order on the set $\{1, 2, \ldots, n\}$ defined as follows: j is preceded by exactly p_j signs $<$. For example, the thirteen weak-orders on three elements $\{1, 2, 3\}$ are: $1 \equiv 2 \equiv 3$, $1 \equiv 2 < 3$, $1 < 2 \equiv 3$, $1 < 2 < 3$, $1 \equiv 3 < 2$, $1 < 3 < 2$, $2 < 1 \equiv 3$, $2 < 1 < 3$, $2 \equiv 3 < 1$, $2 < 3 < 1$, $3 < 1 \equiv 2$, $3 < 1 < 2$, $3 < 2 < 1$.

We may also regard C-permutations as certain classes of trees called Cayley trees [Cay91], as multipartite compositions, or as the different ways in which n different things can be distributed into an unknown number of different parcels without blank lot [Gro62].

If we note W_n the set of all C-permutations of length-n, for $w_n = card(W_n)$, this gives [Gro62]:

$$w_n = \sum_{i=1}^{\infty} 2^{-(i+1)} \times i^n,$$

or recursively

$$w_n = \sum_{i=0}^{n-1} \binom{n}{i} \times w_i, \text{ for } n \geq 1 \text{ with } w_0 = 1 \tag{1}$$

where $\binom{n}{i}$ represent the cardinality of the set $C_{n,i}$ of all i-combinations of $[n] = \{0, 1, \ldots, n-1\}$. Moreover $\frac{w_n}{n!}$ is the coefficient of x^n in the series of $(2 - e^x)^{-1}$ [Cay91]. M. More and A.S. Fraenkel [FM84] gave lexicographic generating and ranking algorithms for C-permutations.

Various studies have been made on Gray codes and generation algorithms for permutations and their restrictions (with given ups and downs [vBR92], [Kor01], involutions [Wal01], and derangements [BVar]) or their generalizations (multiset permutations [Vaj]). In this paper, we give a Gray code list for the set of C-permutations of length-n verifying that two successive elements in the list differ at most in two positions. The aim of this article is twofold. One is to propose the first Gray code for Cayley permutations and thus provide new insights into the combinatorics of particular classes of permutations. The other is to show how the shuffle constructor enables us to obtain new Gray codes and generating algorithms from similar results for simpler objects.

For a set L of length-n sequences we denote by $\mathcal{L}$ an ordered list of all sequences in L. We note $first(\mathcal{L})$ and $last(\mathcal{L})$ the first and the last element of the list $\mathcal{L}$ respectively. The *rank* of an element of $\mathcal{L}$ is the number of elements which precede it, and so the rank of $first(\mathcal{L})$ is 0. $\overline{\mathcal{L}}$ is the list obtained by reversing $\mathcal{L}$, and more generally $\mathcal{L}^{(i)}$ is the list $\mathcal{L}$ if i is even and $\overline{\mathcal{L}}$ if i is odd ; if $\mathcal{L}_1$, ..., $\mathcal{L}_n$ are n ($n > 1$) lists then $\bigcirc_{i=1}^{n}\mathcal{L}_i = \mathcal{L}_1 \circ \ldots \circ \mathcal{L}_n$ is their concatenation.

2 The Gray code

Our construction of a Gray code for the set W_n of length-n C-permutations is based on the combinatorial proof of the relation above (1). Indeed, if we assume that W_0 contains only the empty word λ then each element u of W_n ($n \geq 1$) can be recursively constructed from a length-i C-permutation v ($0 \leq i \leq n-1$) and an i-combination c of n objects. In addition, v and c are unique.

More formally, let $0 \leq i \leq n-1$ and

• $c = c_1\ c_2\ \ldots\ c_n$ be the binary representation of an i-combination of n objects,

• $v = v_1\ v_2\ \ldots\ v_i$ be a C-permutation of length-i.

We define a C-permutation of length-n, $u = u_1\ u_2 \ldots u_n$, denoted by $\sqcup(c, v)$, where each u_k, $1 \leq k \leq n$, is defined as:

$$u_k = \begin{cases} 0 & \text{if } c_k = 0 \\ v_j + 1 & \text{if } c_k = 1 \text{ is the } j\text{th } 1 \text{ in } c. \end{cases}$$

Note that the number of 1s in c equals the length of v, and in particular if there are no 1s in c then $\sqcup(c, \lambda) = 0\ 0 \ldots 0 = c$.
For example, if $i = 5$, $n = 8$, $c = 0\ \mathbf{1}\ \mathbf{1}\ 0\ 0\ \mathbf{1}\ \mathbf{1}\ 0\ \mathbf{1}\ 0$ and $v = 1\ 3\ 0\ 2\ 1$ then $\sqcup(c, v) = 0\ \mathbf{2}\ \mathbf{4}\ 0\ 0\ \mathbf{1}\ \mathbf{3}\ 0\ \mathbf{2}\ 0$.

We call $\sqcup\!\sqcup(c,v)$ the *shuffle* of c by v, and c is the *trajectory* of v in $\sqcup\!\sqcup(c,v)$; see also [Vaj] where Vajnovszki gives a more general definition for the shuffle operator over combinatorial objects.

If $C_{n,i}$ denotes the set of all i-combinations of n objects in binary sequence representation, we have:

Theorem 1 *For $n \geq 1$,*

(*1*) $\sqcup\!\sqcup : C_{n,i} \times W_i \hookrightarrow W_n$ *is a one-to-one map for* $0 \leq i \leq n-1$

(*2*) $\bigcup_{i=0}^{n-1} C_{n,i} \times W_i$ *is isomorphic to* W_n.

Proof : (1) Let $c, c' \in C_{n,i}$; $v, v' \in W_i$; and $u = \sqcup\!\sqcup(c,v)$, $u' = \sqcup\!\sqcup(c',v')$. If $u = u'$ then $u_k = 0$ iff $u'_k = 0$ and so $c = c'$. By a simple translation we have $v = v'$.

(2) it is sufficient to prove that each C-permutation of length-n can be uniquely constructed by the shuffle operation from an appropriate combination c and a C-permutation of length smaller than n.

Let $u \in W_n$, and we construct the binary sequence $c = c_1 \ \dots c_n$ as:

$$c_k = \begin{cases} 0 & \text{if } u_k = 0 \\ 1 & \text{otherwise} \end{cases} \quad \forall 1 \leq k \leq n.$$

Let i be the weight of c (the number of ones in c), and we define $v = v_1 \ \dots v_i \in W_i$ as: for $1 \leq k \leq i$, $v_k = u_j - 1$ if u_j is the kth non zero element in u. We can see that $u = \sqcup\!\sqcup(c,v)$. The uniqueness of the decomposition is obtained by the first point of the present theorem.

$\square$

Now, we extend the shuffle operation to lists of C-permutations and lists of combinations. If $\mathcal{L} = \ell_1, \ell_2, \ell_3, \dots$ is a list of length-i C-permutations ($i \geq 1$) and $c \in C_{n,i}$ then $\sqcup\!\sqcup(c,\mathcal{L})$ is the ordered list $\sqcup\!\sqcup(c,\ell_1) \circ \sqcup\!\sqcup(c,\ell_2) \circ \sqcup\!\sqcup(c,\ell_3) \circ \dots = \bigcirc_{\ell \in \mathcal{L}} \sqcup\!\sqcup(c,\mathcal{L})$ and obviously we have: $\sqcup\!\sqcup(c,\overline{\mathcal{L}}) = \bigcirc_{\ell \in \overline{\mathcal{L}}} \sqcup\!\sqcup(c,\ell) = \overline{\bigcirc_{\ell \in \mathcal{L}} \sqcup\!\sqcup(c,\ell)}$.

Moreover, if $\mathcal{C} = c_1, c_2, \dots$ is a list of combinations in $C_{n,i}$ then $\sqcup\!\sqcup(\mathcal{C},\mathcal{L})$ is the list $\sqcup\!\sqcup(c_1,\mathcal{L}) \circ \sqcup\!\sqcup(c_2,\overline{\mathcal{L}}) \circ \sqcup\!\sqcup(c_3,\mathcal{L}) \dots$. More formally:

$$\sqcup(\mathcal{C},\mathcal{L}) = \bigcirc_{c\in\mathcal{C}} \sqcup(c,\mathcal{L}^{(s)}) = \bigcirc_{c\in\mathcal{C}} \bigcirc_{\ell\in\mathcal{L}^{(s)}} \sqcup(c,\ell) \qquad (2)$$

where s is the rank of c in $\mathcal{C}$.

Lemma 1

$$\sqcup(\mathcal{C},\mathcal{L}) = \begin{cases} \overline{\sqcup(\overline{\mathcal{C}},\overline{\mathcal{L}})} \text{ if } card(\mathcal{C}) \text{ is odd} \\ \\ \overline{\sqcup(\overline{\mathcal{C}},\mathcal{L})} \text{ if } card(\mathcal{C}) \text{ is even} \end{cases}$$

Proof : Suppose that $card(\mathcal{C})$ is odd. Then:

$$\begin{aligned} \overline{\sqcup(\overline{\mathcal{C}},\overline{\mathcal{L}})} &= \overline{\bigcirc_{c\in\overline{\mathcal{C}}} \bigcirc_{\ell\in\overline{\mathcal{L}}^{(s)}} \sqcup(c,\ell)} \\ &= \bigcirc_{c\in\mathcal{C}} \bigcirc_{\ell\in\mathcal{L}^{(s)}} \sqcup(c,\ell), \end{aligned}$$

where s is the rank of c in $\overline{\mathcal{C}}$. If we denote by r the rank of c in $\mathcal{C}$ then $r = card(\mathcal{C}) - s + 1$. Since $card(\mathcal{C})$ is odd, $r = s \bmod 2$ and

$$\begin{aligned} \overline{\sqcup(\overline{\mathcal{C}},\overline{\mathcal{L}})} &= \bigcirc_{c\in\mathcal{C}} \bigcirc_{\ell\in\mathcal{L}^{(r)}} \sqcup(c,\ell) \\ &= \sqcup(\mathcal{C},\mathcal{L}). \end{aligned}$$

When $card(\mathcal{C})$ is even the proof is similar. $\square$

Now, let d be the Hamming distance on length-n sequences. A 2-Gray code for a set S is a list $\mathcal{S}$ for this set where any two successive sequences s and s' verify $d(s,s') \leq 2$. When S is a 2-Gray code for a set of combinations in binary sequence representation, we say that $\mathcal{S}$ is homogeneous if $\mathcal{S}$ is a 2-Gray code, all pairs of successive sequences of $\mathcal{S}$ differ by a transposition of two bits, and bits between those transposed are 0s.

Lemma 2 *Let $\mathcal{C}$ be a homogeneous Gray code for a set $C \subset C_{n,k}$ of combinations in binary sequence representation and $\mathcal{L}$ a 2-Gray code for a set $L \subset W_k$ of Cayley permutations. Then the list $\sqcup(\mathcal{C},\mathcal{L})$ is a 2-Gray code.*

Proof : Let u, u' be two successive elements of the list $\sqcup(\mathcal{C}, \mathcal{L})$. There are as a whole two cases:

1) u and u' are defined from the same combination $c \in \mathcal{C}$ and two successive elements v and v' of $\mathcal{L}$ that are $u = \sqcup(c, v)$, $u' = \sqcup(c, v')$. Since $\mathcal{L}$ is a 2-Gray code ($d(v, v') \leq 2$) and u and u' are defined along the same trajectory c, $d(u, u') \leq 2$.

2) The two successive elements u and u' are defined by two combinations c and c' and only one C-permutation v that is $u = \sqcup(c, v)$, $u' = \sqcup(c', v)$. Since c and c' are successive in $\mathcal{C}$ and $\mathcal{C}$ is homogeneous, c and c' differ in exactly two positions, say i and j, and $c_k = c'_k = 0$ for $i < k < j$. Thus $u_k = u'_k = 0$ for $i < k < j$, $u_i = u'_j$, and $u_j = u'_i$. $\square$

In order to define our Gray code for C-permutations, we need a homogeneous Gray code for the set $C_{n,k}$ of combinations in binary sequence representation.

Various Gray codes are given for combinations, but one of them, as defined by Ruskey, is more interesting for our purpose. It's crucial that the code is homogeneous which is the case when the Gray code is two-close. More precisely, we use the following Gray code, which is a slight variation of Ruskey's Gray code, where each binary sequence is reversed (see Vajnovszki and Walsh [VW02]):

$$\mathcal{C}_{n,k} =$$

$$= \begin{cases} 1^n & \text{if } k = n \\ 0^n & \text{if } k = 0 \\ 0.1^{n-1} \circ 1\ \mathcal{C}_{n-1,k-2} & \text{if } k = n-1 \\ 0\ \overline{\mathcal{C}_{n-1,1}} \circ 1\ 0^{n-1} & \text{if } k = 1 \\ 0\ \overline{\mathcal{C}_{n-1,k}} \circ 1\ 0\ \mathcal{C}_{n-2,k-1} \circ 1\ 1\ \mathcal{C}_{n-2,k-2} & \text{if } 1 < k < n-1\ . \end{cases} \tag{3}$$

Property 1 *[Rus93] The list $\mathcal{C}_{n,k}$ defined by (3) satisfies the properties:*

(1) $\mathcal{C}_{n,k}$ is a list of all k-combinations of $[n]$

(2) $first(\mathcal{C}_{n,k}) = 0\ 1^k\ 0^{n-k-1}$

(3) $last(\mathcal{C}_{n,k}) = 1^k\ 0^{n-k}$

(4) The list $\mathcal{C}_{n,k}$ is a two-close 2-Gray code.

Let $\mathcal{W}_n$ be the list for the set W_n and $\phi_{n,k}$ the sequence defined by $\phi_{n,n-1} = 0$ and for $1 \le k \le n-1$

$$\phi_{n,n-k-1} = \phi_{n,n-k} + \binom{n}{n-k}.$$

For $k \ge 0$, we denote by $\mathcal{V}_k$ the list $\mathcal{W}_k^{(\phi_{n,k})}$ and define recursively the list of all C-permutations by $\mathcal{W}_0 = \lambda$ and for $n \ge 1$

$$\begin{aligned} \mathcal{W}_n &= \sqcup(\mathcal{C}_{n,n-1}^{(0)}, \mathcal{V}_{n-1}) \circ \sqcup(\mathcal{C}_{n,n-2}^{(1)}, \mathcal{V}_{n-2}) \circ \ldots \\ &\ldots \circ \sqcup(\mathcal{C}_{n,1}^{(n-2)}, \mathcal{V}_1) \circ \sqcup(\mathcal{C}_{n,0}^{(n-1)}, \mathcal{V}_0) \\ &= \bigcirc_{k=0}^{n-1} \sqcup (\mathcal{C}_{n,n-1-k}^{(k)}, \mathcal{V}_{n-1-k}) = \\ &= \bigcirc_{k=0}^{n-1} \sqcup (\mathcal{C}_{n,n-1-k}^{(k)}, \mathcal{W}_{n-1-k}^{(\phi_{n,n-1-k})}) \end{aligned} \tag{4}$$

In table 1 we give the list for the set W_4:

<u>Table 1</u>: The list
$\mathcal{W}_4 = \sqcup(\mathcal{C}_{4,3}^{(0)}, \mathcal{W}_3^{(0)}) \circ \sqcup(\mathcal{C}_{4,2}^{(1)}, \mathcal{W}_2^{(4)}) \circ \sqcup(\mathcal{C}_{4,1}^{(2)}, \mathcal{W}_1^{(10)}) \circ \sqcup(\mathcal{C}_{4,0}^{(3)}, \mathcal{W}_0^{(14)})$.

The first fifty two elements form the sublist $\sqcup(\mathcal{C}_{4,3}^{(0)}, \mathcal{W}_3^{(0)})$; the sublist $\sqcup(\mathcal{C}_{4,2}^{(1)}, \mathcal{W}_2^{(4)})$ is in **boldface**; the sublist $\sqcup(\mathcal{C}_{4,1}^{(2)}, \mathcal{W}_1^{(10)})$ is in *italic* . The last element is the single element list $\sqcup(\mathcal{C}_{4,0}^{(3)}, \mathcal{W}_0^{(14)})$.

1. 0123	14. 1011	27. 1203	40. 1110	53. **1200**	66. **0201**
2. 0132	15. 1021	28. 1302	41. 1210	54. **2100**	67. **0101**
3. 0122	16. 1012	29. 1202	42. 1120	55. **1100**	68. **0110**
4. 0212	17. 2011	30. 2102	43. 2110	56. **1010**	69. **0210**
5. 0312	18. 2021	31. 3102	44. 2210	57. **2010**	70. **0120**
6. 0213	19. 3021	32. 2103	45. 3210	58. **1020**	71. *0100*
7. 0231	20. 2031	33. 2301	46. 2310	59. **1002**	72. *0001*
8. 0321	21. 2013	34. 3201	47. 2130	60. **2001**	73. *0010*
9. 0221	22. 3012	35. 2201	48. 3120	61. **1001**	74. *1000*
10. 0211	23. 2012	36. 2101	49. 2120	62. **0011**	75. **0000**
11. 0112	24. 1022	37. 1102	50. 1220	63. **0021**	
12. 0121	25. 1032	38. 1201	51. 1320	64. **0012**	
13. 0111	26. 1023	39. 1101	52. 1230	65. **0102**	

Property 2 *For $n \geq 1$*

(1) $first(\mathcal{W}_n) = 0\ 1\ 2\ \ldots\ n-1$

(2) $last(\mathcal{W}_n) = 0\ 0\ \ldots\ 0$

(3) Two successive elements of the list $\mathcal{W}_n$ differ in at most two positions.

Proof : By Definition 2 of the list $\mathcal{W}_n$, we have $first(\mathcal{W}_1) = last(\mathcal{W}_1) = 0$ and for $n \geq 1$, $last(\mathcal{W}_n) = 0\ 0 \ldots 0$. The recurrence on $n \geq 1$ completes the proof:

$$\begin{aligned} first(\mathcal{W}_n) &= first(\sqcup(\mathcal{C}_{n,n-1}, \mathcal{W}_{n-1})) \\ &= first(\sqcup(\mathcal{C}_{n,n-1}, first(\mathcal{W}_{n-1})) \\ &= \sqcup(0\ 1\ 1 \ldots 1, 0\ 1\ 2 \ldots n-2) \\ &= 0\ 1\ 2\ 3 \ldots n-1. \end{aligned}$$

For the third point of the property, Lemma 2 proves that each block $\sqcup(\mathcal{C}^{(k)}_{n,n-1-k}, \mathcal{W}^{(\phi_{n,n-1-k})}_{n-1-k})$ is a 2-Gray code.

In order to prove this, we must verify that the last element of the list $\sqcup(\mathcal{C}^{(k+1)}_{n,n-k}, \mathcal{W}^{(\phi_{n,n-k})}_{n-k})$ and the first element of the list $\sqcup(\mathcal{C}^{(k)}_{n,n-1-k}, \mathcal{W}^{(\phi_{n,n-1-k})}_{n-1-k})$ differ in at most two positions.

There are four cases :

• k is even and $\phi_{n,n-1-k}$ is even. Then we have:

$$first(\sqcup(\mathcal{C}_{n,n-1-k}^{(k)}, \mathcal{W}_{n-1-k}^{(\phi_{n,n-1-k})})) = \sqcup(first(\mathcal{C}_{n,n-1-k}), first(\mathcal{W}_{n-1-k}))$$
$$= 0\ 1\ 2 \ldots (n-1-k)\ 0^k$$

and since $\phi_{n,n-k} = \phi_{n,n-1-k} - \binom{n}{n-k}$ then

$$\begin{aligned} last(\sqcup(\mathcal{C}_{n,n-k}^{(k+1)}, \mathcal{W}_{n-k}^{(\phi_{n,n-k})})) &= \sqcup(last(\overline{\mathcal{C}_{n,n-k}}), last(\overline{\mathcal{W}_{n-k}})) \\ &= 0\ 1\ 2\ \ldots (n-1-k)\ (n-k)\ 0^{k-1}. \end{aligned} \tag{5}$$

• k is odd and $\phi_{n,n-1-k}$ is even:

$$first(\sqcup(\mathcal{C}_{n,n-1-k}^{(k)}, \mathcal{W}_{n-1-k}^{(\phi_{n,n-1-k})})) = \sqcup(first(\overline{\mathcal{C}_{n,n-1-k}}), first(\mathcal{W}_{n-1-k}))$$
$$= 1\ 2 \ldots (n-1-k)\ 0^{k+1}$$

and

$$\begin{aligned} last(\sqcup(\mathcal{C}_{n,n-k}^{(k+1)}, \mathcal{W}_{n-k}^{(\phi_{n,n-k})})) &= \sqcup(last(\mathcal{C}_{n,n-k}), last(\overline{\mathcal{W}_{n-k}})) \\ &= 1\ 2\ \ldots (n-1-k)\ (n-k)\ 0^k. \end{aligned} \tag{6}$$

• k is even and $\phi_{n,n-1-k}$ is odd:

$$first(\sqcup(\mathcal{C}_{n,n-1-k}^{(k)}, \mathcal{W}_{n-1-k}^{(\phi_{n,n-1-k})})) = \sqcup(first(\mathcal{C}_{n,n-1-k}), first(\overline{\mathcal{W}_{n-1-k}}))$$
$$= 0\ 1^{n-1-k}\ 0^k$$

and

$$\begin{aligned} last(\sqcup(\mathcal{C}_{n,n-k}^{(k+1)}, \mathcal{W}_{n-k}^{(\phi_{n,n-k})})) &= \sqcup(last(\overline{\mathcal{C}_{n,n-k}}), last(\mathcal{W}_{n-k})) \\ &= 0\ 1^{n-k}\ 0^{k-1}. \end{aligned} \tag{7}$$

• k is odd and $\phi_{n,n-1-k}$ is odd:

$$first(\sqcup(\mathcal{C}_{n,n-1-k}^{(k)}, \mathcal{W}_{n-1-k}^{(\phi_{n,n-1-k})})) = \sqcup(first(\overline{\mathcal{C}_{n,n-1-k}}), first(\overline{\mathcal{W}_{n-1-k}}))$$

$$= 1^{n-1-k}\ 0^{k+1}$$

and

$$\begin{array}{ll} last(\sqcup(\mathcal{C}^{(k+1)}_{n,n-k}, \mathcal{W}^{(\phi_{n,n-k})}_{n-k})) & = \sqcup(last(\mathcal{C}_{n,n-k}), last(\mathcal{W}_{n-k})) \\ & = 1^{n-k}\ 0^k. \end{array} \quad . \tag{8}$$

□

3 Algorithmic considerations

In this part we explain how the recursive definition (4) can be implemented into efficient generating algorithms. Such algorithms already exist for combinations [EM][RP90], so we will just give the main difficulties inimplementing ours.

Successive C-permutations are stored in an array w, and a loop statement generates the sublists $\mathcal{L}_k = \sqcup(\mathcal{C}^{(k)}_{n,n-1-k}, \mathcal{W}^{(\phi_{n,n-1-k})}_{n-1-k})$ for $0 \le k \le n-1$. For each list $\mathcal{L}_k$, we use an algorithm from Vajnovszki and Walsh [VW02] to generate the two-close list $\mathcal{C}^{(k)}_{n,n-1-k}$ such that there is a constant number of computations between successive combinations. So, for each $(n-1-k)$-combination c of length-n we produce recursively the list $\mathcal{W}_{n-1-k}$ or $\overline{\mathcal{W}_{n-1-k}}$ according to the parity of $\phi_{n,n-1-k}$ and to the rank of c in $\mathcal{C}^{(k)}_{n,n-1-k}$.

In order to store the different combinations c for each level of recursivity, we use $(n-1)$ global arrays T_i $(1 \le i \le n-1)$. More precisely, if $c = c_1\ c_2 \dots c_k$ is a combination in integer sequence representation, the array T_i is defined by: $T_i[0] = c_1$, for $1 \le j \le k-1$ $T_i[c_j] = c_{j+1}$ and $T_i[c_k] = n+1$. In fact, $T_i[c_j]$ is the position of the $(j+1)$th entry greater than $(i-1)$ in the current C-permutation w. Notice that T_i corresponds to a combination on the trajectory defined by T_{i-1}. The interest of this representation of combinations is that between two consecutive combinations on the level i we need to change at most one value in the array T_i, and we don't modify the others. Likewise, between two lists $\mathcal{L}_k$, we need to modify only one entry on one array T_i. This structure allows us to assure that this algorithm transforms an object into its successor in a constant amortized time.

More precisely, this can be implemented as follows:
- *initialize a global array* w *by* $first(\mathcal{W}_n) = 0\ 1\ 2 \ldots\ n-1$
- *initialize* $(n-1)$ *global arrays* $T_1, T_2, \ldots, T_n$ according to $first(\mathcal{W}_n)$
- *For* k *from* $n-1$ *downto* 0
    - *while* $T_{n-k} \neq last(\mathcal{C}^{(k)}_{n,n-1-k})$ *do*
        - $T_{n-k} = succ(T_{n-k})$
        - *call resursively the algorithm on the trajectory* T_{n-k}
    - *modify the array* T_{n-k} *according to the relations* (5) (6) (7) (8)

The time complexity of this algorithm is proportional to the total number of recursive calls. The number of calls of degree 1 is at most w_n, and the others calls have a degree of at least 2. So, we have the following inequality:

$$\frac{number\ of\ recursive\ calls}{number\ of\ generated\ objects} \leq 3,$$

which proves that the complexity is linear in the number of generated words.

4 Concluding remarks

Our Gray code verifies that two successive elements differ in at most two positions i and j $(i < j)$ without more restrictions on these indices. Are there a similar Gray code $\mathcal{W}_n$ and constant c (independent of n) such that i and j verify the $|i-j| < c$? Moreover, we do not find a Gray code verifying that two successive elements of the list differ in only one position. However, if we denote by $even(W_n)$ the number of C-permutations $w = w_1 \ldots w_n \in W_n$ verifying $\sum_{i=1}^{n} c_i$ is even, and $odd(W_n) = w_n - even(w_n)$, we verify easily that for $n \geq 3 :\ even(W_n) < odd(W_n) - 2$. This inequality proves that there is no Gray code such that two successive elements differ in only one position and the entry on this position differs by one.

References

[BVar] J.L. Baril and V. Vajnovszki. Gray code for derangements. *Discrete Applied Math.*, (to appear).

[Cay91] A. Cayley. On the analytical forms called trees. In *Collected Mathematical Papers*, volume 4. Cambridge University Press, 1891.

[EM] G. Eades and B. McKay. An algorithm for generating subsets of fixed size with a strong minimal change property. *I.P.L.*, 19:131–133.

[FM84] A.S. Fraenkel and M. Mor. Cayley permutations. *Discrete Mathematics*, 48:101–112, 1984.

[Gro62] O.A. Gross. Preferential arrangements. *Amer. Math.*, Monthly 69, 1962.

[Kor01] J.F. Korsh. Loopless generation of up-down permutations. *Discrete Math.*, 240:97–122, 2001.

[RP90] F. Ruskey and A. Proskurowski. Generating binary trees by transpositions. *J. Algorithms*, 11:68–84, 1990.

[Rus93] F. Ruskey. Simple combinatorial Gray codes constructed by reversing sublists. *L.N.C.S*, 762:201–208, 1993.

[Sav89] C.S. Savage. Gray code sequences of partitions. *J. Algorithms*, (10):577–595, 1989.

[Vaj] V. Vajnovszki. Generating multiset permutations. *To appear T.C.S.*

[vBR92] D. Roelants van Baronaigien and F. Ruskey. Generating permutations with given ups and downs. *Discrete Appl. Math.*, 1 (36), 1992.

[VW02] V. Vajnovski and T. Walsh. A loopless two-close Gray code algorithm for listing k-ary Dick words. *Submited*, 2002.

[Wal01] T. Walsh. Gray codes for involutions. *J. Combin. Math. Combin. Comput.*, 36:95–118, 2001.

J.-L. Baril,
Received June 26, 2003

LE2I, UMR-CNRS 5158, Université de Bourgogne
BP 47870, 21078 Dijon cedex, France
E–mail: $barjl@u-bourgogne.fr$