Discrete Mathematics and Theoretical Computer Science DMTCS vol. **12:1**, 2010, 109–122

Generating involutions, derangements, and relatives by ECO

Vincent Vajnovszki

LE2I – UMR CNRS, Université de Bourgogne, B.P. 47 870, 21078 DIJON-Cedex France.
Email: vvajnov@u-bourgogne.fr

received August 14, 2007, *revised June 16, 2009, March 4, 2010*, *accepted February 25, 2010*.

We show how the ECO method can be applied to exhaustively generate some classes of permutations. A previous work initiating this technique and motivating our research was published in Acta Informatica, 2004, by S. Bacchelli, E. Barcucci, E. Grazzini and E. Pergola.

Keywords: ECO method, exhaustive generation, Bell permutations, involutions, derangements.

1 Introduction

There is a significant literature on the exhaustive generation of permutations, see for instance (11; 15; 17; 24; 25) and (21) for a survey. More recently, great interest was given to the generation of particular classes of permutations, see for example (3; 4; 5; 6; 9; 10; 18; 27) and the references therein.

In this paper we show how the ECO method can be applied to exhaustively generate unrestricted permutations, Bell permutations, involutions, fixed point free involutions and derangements. The obtained algorithms are new, run in constant amortized time and are easy to implement and to understand. Actually there already exist, more or less *ad hoc*, efficient generating algorithms for these classes of permutations. For example, unrestricted permutations are generated looplessly in (11), Bell permutations by a CAT algorithm in (18), involutions looplessly in (27), derangements are generated in Gray code order by a CAT algorithm in (6). Here we emphasize a general unified ECO method based approach.

This note is the extended form of its preliminary conference version (26).

ECO operators and rules

We just restate the definition of an *ECO operator* as it is formalized in (2; 12). Let $\mathcal{O}$ be a class of combinatorial objects and $\mathcal{O}_n$ the subclass of objects of size n. An operator ϑ on the class $\mathcal{O}$ is a family of functions (one for each n) $\vartheta : \mathcal{O}_n \to 2^{\mathcal{O}_{n+1}} \cup 2^{\mathcal{O}_{n+2}}$ with $2^{\mathcal{O}_n}$ being the power set of $\mathcal{O}_n$. If the operator ϑ satisfies the following conditions:

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 $x \in \mathcal{O}_{n-1} \cup \mathcal{O}_{n-2}$ such that $y \in \vartheta(x)$,

then $\{\vartheta(x) \cap \mathcal{O}_n\}_{x \in \mathcal{O}_{n-1} \cup \mathcal{O}_{n-2}}$ is a partition of $\mathcal{O}_n$, $n \geq 1$, and ϑ is called an ECO operator.

In addition, if

3. for each $x \in \mathcal{O}_n$ and $y \in \vartheta(x)$ one can compute y from x, and conversely, in constant time and constant size extra-space,

then we call ϑ *uniform*. So, uniformity has an algorithmic meaning.

An *ECO (or succession) rule* corresponding to an ECO operator is a formal system consisting of a root e_0 and a set of *productions* of the form

$$\begin{array}{ccl} (k) & \stackrel{1}{\rightsquigarrow} & (e_1(k))(e_2(k))\ldots((e_j)(k)) \\ & \stackrel{2}{\rightsquigarrow} & (e_{j+1}(k))(e_{j+2}(k))\ldots((e_k)(k)) \end{array}$$

where e_0 and each $e_i(k)$, $1 \leq i \leq k$, are integers. The right side of these productions are sequences of parenthesed integers.

An ECO rule explains how, to an object $x \in \mathcal{O}_n$ with $\mathrm{card}(\vartheta(x)) = k$, the operator ϑ associates its k successors $y_1, y_2, \ldots, y_j \in \mathcal{O}_{n+1}$, $y_{j+1}, y_{j+2}, \ldots, y_k \in \mathcal{O}_{n+2}$ with $\mathrm{card}(\vartheta(y_i)) = e_i(k)$. Note that in productions above the size of objects does not occur explicitly, and when each object of size n has only successors of size $n+1$, then these productions can be expressed simply

$$(k) \quad \rightsquigarrow \quad (e_1(k))(e_2(k))\ldots((e_k)(k)).$$

Permutations

Let $\mathfrak{S}_n$ denote the set of all length-n permutations and for a $\pi \in \mathfrak{S}_n$ we denote by $\overline{\pi}$ the permutation in $\mathfrak{S}_{n+1}$ such that $\overline{\pi}(i) = \pi(i)$ for all i, $1 \leq i \leq n$, and $\overline{\pi}(n+1) = n+1$. $\langle j, k\rangle$ is the transposition of i and j, that is the permutation π of appropriate length with $\pi(i) = i$ for all i, except $\pi(j) = k$ and $\pi(k) = j$. For two permutations $\pi, \sigma \in \mathfrak{S}_n$, $\pi \cdot \sigma$ denotes their product, i.e., the permutation in $\mathfrak{S}_n$ with $(\pi \cdot \sigma)(i) = \pi(\sigma(i))$ for all i. For example, if $\pi, \sigma \in \mathfrak{S}_4$ with $\pi = 2\,3\,1\,4$ and $\sigma = 3\,4\,1\,2 \in \mathfrak{S}_4$, then $\pi \cdot \sigma = 1\,4\,2\,3$.

Fact 1 stated below gives a recursive construction for $\mathfrak{S}_n$. It is based on the ECO operator

$$\vartheta : \mathfrak{S}_n \to 2^{\mathfrak{S}_{n+1}},$$

$$\pi \stackrel{\vartheta}{\longmapsto} \{\overline{\pi} \cdot \langle i, n+1\rangle \,|\, 1 \leq i \leq n\} \cup \{\overline{\pi}\}.$$

The first three levels of this recursive construction are depicted in Figure 1(a). For instance, $\vartheta(1) = \{2\,1, 1\,2\} = \mathfrak{S}_2$ and $\vartheta(2\,1) = \{3\,1\,2, 2\,3\,1, 2\,1\,3\} \subset \mathfrak{S}_3$. The ECO rule corresponding to this ECO operator is (see the first row of Table 1).

$$\begin{array}{l} root : (2) \\ rule : (k) \quad \rightsquigarrow \quad (k+1)^k, \end{array}$$

which means that:

- the root permutation, i.e. the length-1 permutation, has two successors, and

- if a permutation has k successors, then each of them has in turn $k+1$ successors, that is, each $e_i(k)$ equals $k+1$.

Fact 1 $\mathfrak{S}_1 = \{1\}$. *If* $\pi \in \mathfrak{S}_t$, $t \geq 1$, *then*

- *for any* $i \in \{1, 2, \ldots, t\}$ *the permutation* $\overline{\pi} \cdot \langle i, t+1 \rangle$ *is in* $\mathfrak{S}_{t+1}$,
- *the permutation* $\overline{\pi}$ *is in* $\mathfrak{S}_{t+1}$.

Moreover, any permutation in $\mathfrak{S}_t$, $t \geq 2$, *can be uniquely obtained from a permutation in* $\mathfrak{S}_{t-1}$ *by one of the two transformations above.*

The algorithmic implementation of this construction is given by the recursive procedure in Algorithm 1. Note that, if in this algorithm the last recursive call of `gen_Perm(`$t+1$`)` is removed, then it produces 1-cycles, that is, permutations consisting of a single cycle. In this case, the first levels of the obtained generating tree are presented in Figure 1(b), and this tree is the restriction to 1–cycles of the generating tree for unrestricted permutations in Figure 1(a).

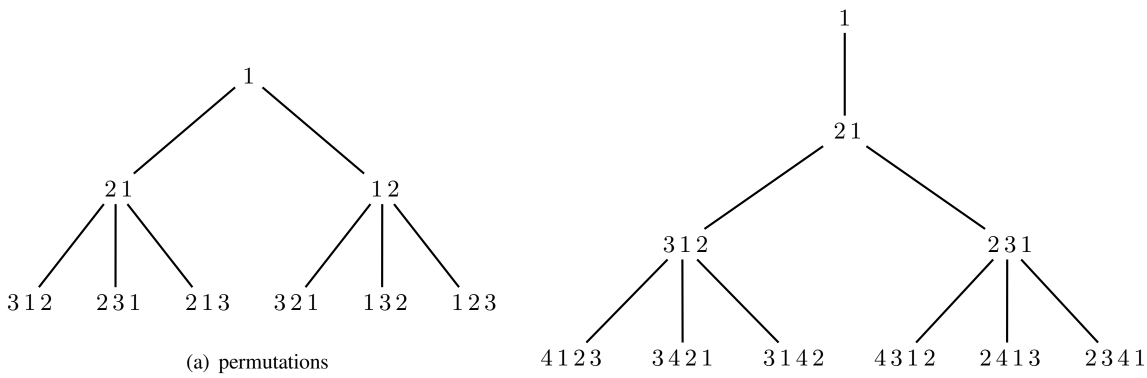
Algorithm 1 Algorithm producing exhaustively length-n permutations. n and π are both global variables and initially π is $1\,2\,3\,\ldots\,n \in \mathfrak{S}_n$. The main call is `gen_Perm(1)` and the statement $\pi := \pi \cdot \langle i, t+1 \rangle$ after each recursive call restores the value of π before this call.

```
procedure gen_Perm(t)
local i;
if t = n then Print(π);
else for i from 1 to t do
          π := π · ⟨i, t+1⟩;
          gen_Perm(t+1);
          π := π · ⟨i, t+1⟩;
     end do
     gen_Perm(t+1);
end if
end procedure.
```

In the remainder of this paper we will give similar ECO-based recursive constructions together with their algorithmic implementations for Bell permutations, involutions, pure involutions and derangements. As for unrestricted permutations above, the computation tree for each obtained algorithm coincides with the induced tree of the corresponding ECO operator.

2 Bell permutations

A cycle C in a permutation $\pi \in \mathfrak{S}_n$ is a sequence $C = (x_0 x_1 \ldots x_{j-1})$ such that $x_{i+1} = \pi(x_i)$ for all i, $0 \leq i \leq j-1$, where the subscript i is taken modulo j. Obviously, the cycle $(x_i x_{i+1} \ldots x_{j-1} x_0 x_1 \ldots x_{i-1})$ is equivalent to the cycle C and we choose to represent the cycles with their smallest element last. Any permutation is a product of disjoint cycles and the *standard cycle representation* of a permutation is obtained by imposing that cycles be written in increasing order of their smallest element (that is, their last



(a) permutations

(b) 1-cycles

Fig. 1: The first levels of the generating tree for permutations and 1–cycles.

element). For example, the standard cycle representation of $4\,2\,5\,1\,7\,6\,3 \in \mathfrak{S}_7$ is $(4\,1)(2)(\,5\,7\,3)(6)$. As an aside independent of the rest of this article, let π' be the permutation in $\mathfrak{S}_n$ obtained after erasing the parentheses in the standard cycle representation of $\pi \in \mathfrak{S}_n$. π can uniquely be recovered from π' and the transformation $\pi \mapsto \pi'$ is a bijection from $\mathfrak{S}_n$ to itself. This map is essentially the *transformation fondamentale* of (14, Proposition 1.3.1), see also (23, pp. 17).

We now consider the set $\mathfrak{B}_n$ of permutations in $\mathfrak{S}_n$ where each cycle is a decreasing sequence of integers. $\mathfrak{B}_n$ is in bijection with the set of all partitions of $\{1, 2, \ldots n\}$: each cycle in $\pi \in \mathfrak{B}_n$ represents a block of the partition. For instance the partition corresponding to $4\,2\,7\,1\,3\,6\,5 \in \mathfrak{B}_7$ is $\{4, 1\}\{2\}\{7, 5, 3\}\{6\}$. Thus, $\mathfrak{B}_n$ is counted by the Bell numbers (sequence A000110 in (22)) and we call it the set of *Bell permutations*. See also (16) for an alternative definition of Bell permutations in terms of pattern avoidance.

For $\pi \in \mathfrak{S}_n$, a *tail* in $\pi \in \mathfrak{S}_n$ is an integer i such that $\pi(i)$ is minimal in its cycle, and let T_π denote the set of tails of π.

Lemma 1 $\mathfrak{B}_1 = \{1\}$ *and the set of tails of the permutation* 1 *is* $T_1 = \{1\}$. *If* $\pi \in \mathfrak{B}_t$, $t \geq 1$, *and its set of tails is* T_π, *then*

(i) for any $i \in T_\pi$, *the permutation* $\overline{\pi} \cdot \langle i, t+1 \rangle$ *is in* $\mathfrak{B}_{t+1}$ *and its set of tails is* T_π,

(ii) the permutation $\overline{\pi}$ *is in* $\mathfrak{B}_{t+1}$ *and its set of tails is* $T_\pi \cup \{t+1\}$.

Moreover, any permutation in $\mathfrak{B}_t$, $t \geq 2$, *can be uniquely obtained from a permutation in* $\mathfrak{B}_{t-1}$ *by one of the two transformations above.*

Proof: The proof is based on the following observations. If $\pi \in \mathfrak{B}_t$, $t \geq 1$, and i is a tail of π, then $\pi(i)$ is the maximal element in the cycle of i and $\overline{\pi} \cdot \langle i, t+1 \rangle$ is the permutation obtained from π by inserting $t+1$ between i and $\pi(i)$ in this cycle. So, $\overline{\pi} \cdot \langle i, t+1 \rangle \in \mathfrak{B}_{t+1}$ and the set of tails of $\overline{\pi} \cdot \langle i, t+1 \rangle$ equals that of π. On the other hand, $\overline{\pi}$ is a length $t+1$ Bell permutation where $t+1$ is a new single element cycle. $\square$

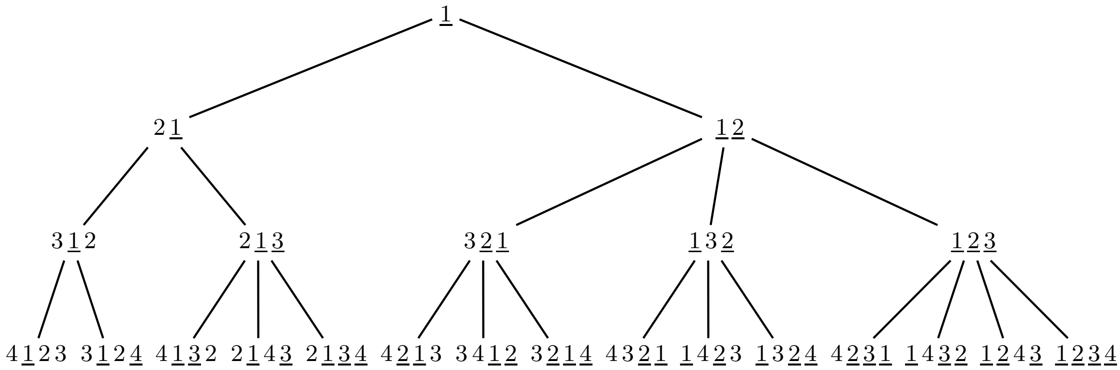


Fig. 2: The first four levels of the generating tree for Bell permutations. It is simply the restriction to Bell permutations of the generating tree for unrestricted permutations in Figure 1(a). The tails of each permutation are underlined.

The previous recursive construction is embodied in the generating procedure in Algorithm 2 and the induced generating tree is the restriction to Bell permutations of the generating tree for unrestricted permutations induced by Fact 1. The first levels of this tree are depicted in Figure 2.

Now we show how the construction in Lemma 1 is related to the ECO rule for Bell permutations in Table 1. Suppose that π is a Bell permutation with c cycles (and so, $c = \operatorname{card}(T_\pi)$). π produces $c+1$ permutations: c by point (i) and one by point (ii). Let consider now a permutation with k successors, and so with $k-1$ cycles. It produces

- by point (i): $k-1$ Bell permutations with $k-1$ cycles each, and so with k successors;
- by point (ii): one Bell permutation with k cycles, and so with $k+1$ successors.

Algorithm 2 Procedure generating length-n Bell permutations. Initially $\pi = 1\,2\,3\,\ldots\,n \in \mathfrak{S}_n$, $T = \{1\}$ and the main call is `gen_Bell(1)`.

```
procedure gen_Bell(t)
local i;
if t = n then Print(π);
else for i ∈ T do
          π := π · ⟨i, t+1⟩; gen_Bell(t+1); π := π · ⟨i, t+1⟩;
     end do
     T := T ∪ {t+1}; gen_Bell(t+1); T := T \ {t+1};
end if
end procedure.
```

3 Involutions

An involution is a permutation π where $\pi \cdot \pi$ equals the identity permutation, and let $\mathfrak{I}_n \subseteq \mathfrak{S}_n$ denote the set of length-n involutions. Here we give two recursive constructions for $\mathfrak{I}_n$: *by fixed points* and *by recurrence*.

3.1 Involution by fixed points

Lemma 2 $\mathfrak{I}_1 = \{1\}$ *and the set of fixed points of the permutation* 1 *is* $F_1 = \{1\}$. *If* $\pi \in \mathfrak{I}_t$, $t \geq 1$, *and* F_π *is the set of its fixed points, then*

(i) for any $i \in F_\pi$, *the permutation* $\overline{\pi} \cdot \langle i, t+1 \rangle$ *is in* $\mathfrak{I}_{t+1}$ *and its set of fixed points is* $F_\pi \setminus \{i\}$,

(ii) the permutation $\overline{\pi}$ *is in* $\mathfrak{I}_{t+1}$ *and its set of fixed points is* $F_\pi \cup \{t+1\}$.

Moreover, any permutation in $\mathfrak{I}_t$, $t \geq 2$, *can be uniquely obtained from a permutation in* $\mathfrak{I}_{t-1}$ *by one of the two transformations above.*

Proof: Clearly, every permutation obtained from a $\pi \in \mathfrak{I}_t$, $t \geq 1$, by each of the two transformations above is an involution. Conversely, for any $\tau \in \mathfrak{I}_n$, $n \geq 2$, there exists a unique $\pi \in \mathfrak{I}_{n-1}$ such that τ is obtained from π by one of these transformations. Indeed, if n is not a fixed point of τ, then let $i = \tau(n)$ and π be the involution defined by $\pi(j) = \tau(j)$ for all j, $1 \leq j \leq n-1$, except $\pi(i) = i$; and so $\tau = \overline{\pi} \cdot \langle i, n \rangle$. If n is a fixed point of τ, then π is simply defined by $\pi(j) = \tau(j)$ for all j, $1 \leq j \leq n-1$, and in this case $\tau = \overline{\pi}$. $\square$

The recursive construction in the previous lemma is implemented in Algorithm 3 and the first levels of the induced generating tree are depicted in Figure 3. Now we show how this lemma mirrors the first ECO rule for involutions in Table 1.

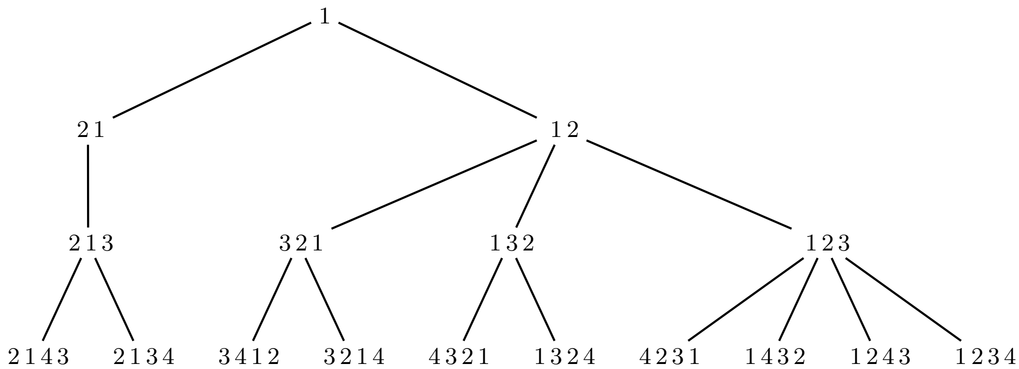


Fig. 3: The first four levels of the generating tree for involutions by fixed points.

Let π be an involution with f fixed points. Then it produces $f+1$ involutions: f by point (i) and one by point (ii). Conversely, if π has k successors, then it has $f = k-1$ fixed points and produces

- by point (i): $k-1$ involutions with $f-1$ fixed points each, and so with $f=k-1$ successors
- by point (ii): one involutions with $f+1$ fixed points, and so with $f+2=k+1$ successors.

Algorithm 3 Procedure generating length-n involutions by fixed points. Initially π is $1\,2\,3\,\ldots\,n \in \mathfrak{S}_n$, $F=\{1\}$ and the main call is `gen_Inv(1)`.

```
procedure gen_Inv(t)
local i;
if t = n then Print(π);
else for i ∈ F do
          π := π · ⟨i, t+1⟩;  F := F \ {i};
          gen_Inv(t+1);
          F := F ∪ {i};  π := π · ⟨i, t+1⟩;
     end do
     F := F ∪ {t+1};  gen_Inv(t+1);  F := F \ {t+1};
end if
end procedure.
```

3.2 Involution by recurrence

Here we describe an alternative definition for $\mathfrak{I}_n$ involving a *jumping* succession rule, that is, the successors of an length-n involution are involutions of length $n+1$ and length $n+2$. For $\pi \in \mathfrak{S}_n$, $\overline{\overline{\pi}}$ denotes the permutation $\overline{(\overline{\pi})} \in \mathfrak{S}_{n+2}$.

Lemma 3 *Let define $\mathfrak{I}_0 = \{\epsilon\}$, with ϵ being the 'length-zero' permutation. If $\pi \in \mathfrak{I}_t$, $t \geq 0$, then*

(i) $\overline{\pi}$ *is in* $\mathfrak{I}_{t+1}$,

(ii) $\overline{\overline{\pi}} \cdot \langle t+1, t+2\rangle$ *is in* $\mathfrak{I}_{t+2}$,

(iii) for all $i \in \{1, 2, \ldots t\}$, $\overline{\overline{\pi}} \cdot \sigma$ *is in* $\mathfrak{I}_{t+2}$ *where*

- $\sigma = \langle i, t+2\rangle$ *if i is a fixed point of π, and*
- $\sigma = \langle i, t+1\rangle \cdot \langle \pi(i), t+2\rangle \cdot \langle t+1, t+2\rangle$ *elsewhere.*

Moreover, any permutation in $\mathfrak{I}_t$, $t \geq 1$, can be uniquely obtained from a permutation in $\mathfrak{I}_{t-1}$ or in $\mathfrak{I}_{t-2}$ by one of these transformations.

Proof: It is easy to check that every permutation obtained from a $\pi \in \mathfrak{I}_t$, $t \geq 0$, by each of the three transformations above is an involution. Now we show that for each $\tau \in \mathfrak{I}_n$, $n \geq 1$, there exists a unique involution π of length $n-1$ or $n-2$ such that τ is obtained from π by one of these transformations. An involution $\tau \in \mathfrak{I}_n$ can be in one of the three following cases:

(1) n is fixed point of τ,

(2) $n \geq 2$, and $\tau(n) = n-1$ (and so $\tau(n-1) = n$), or

(3) $n \geq 2$, n is not a fixed point of τ and $\tau(n) \neq n-1$.

In case (1) $\pi \in \mathfrak{I}_{n-1}$ is defined by $\pi(j) = \tau(j)$ for all j, $1 \leq j \leq n-1$, and $\tau = \overline{\pi}$. Similarly, in case (2) $\pi \in \mathfrak{I}_{n-2}$ is defined by $\pi(j) = \tau(j)$ for all j, $1 \leq j \leq n-2$, and $\tau = \overline{\overline{\pi}} \cdot \langle n-1, n\rangle$.

Let now $\tau \in \mathfrak{I}_n$ as in case (3). When $n-1$ is a fixed point of τ let $i = \tau^{-1}(n) = \tau(n)$ and define the length-$(n-2)$ permutation π by $\pi(j) = \tau(j)$ for all j, $1 \leq j \leq n-2$, except $\pi(i) = i$; in this case π is an involution, i is a fixed point of π and $\tau = \overline{\overline{\pi}} \cdot \langle i, n\rangle$. So τ is obtained from π by the first transformation of the last point of the present lemma.

When $n-1$ is not a fixed point of τ, then define the length-$(n-2)$ permutation π as

$$\pi(j) = \begin{cases} \tau(j) & \text{if} \quad 1 \leq j \leq n-2, j \neq i, j \neq k \\ i & \text{if} \quad j = k \\ k & \text{if} \quad j = i \end{cases}$$

where $i = \tau(n-1)$ and $k = \tau(n)$. Clearly, π is an involution, i is not a fixed point of π and $\tau = \overline{\overline{\pi}} \cdot \langle i, n-1\rangle \cdot \langle \pi(i), n\rangle \cdot \langle n-1, n\rangle$; and τ is obtained from π by the second transformation of the last point of the this lemma. $\square$

Below we show that the lemma above is related to the second ECO rule for involutions in Table 1. Its algorithmic counterpart is given by Algorithm 4 and in Figure 4 are depicted the first levels of the induced generating tree.

Algorithm 4 Procedure generating length-n involutions by recurrence. Initially $\pi = 1\,2\,3\,\ldots\,n$ and the main call is `gen_Inv_R(0)`.

```
procedure gen_Inv_R(t)
local i,σ;
if t = n then Print(π);
else gen_Inv_R(t+1);
     if t ≤ n−2 then
          π := π · ⟨t+1,t+2⟩; gen_Inv_R(t+2); π := π · ⟨t+1,t+2⟩;
          for i from 1 to t do
               if π[i] = i then σ := ⟨i,t+2⟩;
               else σ := ⟨i,t+1⟩ · ⟨π[i],t+2⟩ · ⟨t+1,t+2⟩;
               end if
               π := π · σ; gen_Inv_R(t+2); π := π · σ^{−1};
          end do
     end if
end if
end procedure.
```

If π is an involution of length t, then it has $t+2$ successors: one of length $t+1$ by point (i), and $t+1$ of length $t+2$ by point (ii) and (iii). Conversely, if π has k successors (and so length $t = k-2$), then it produces

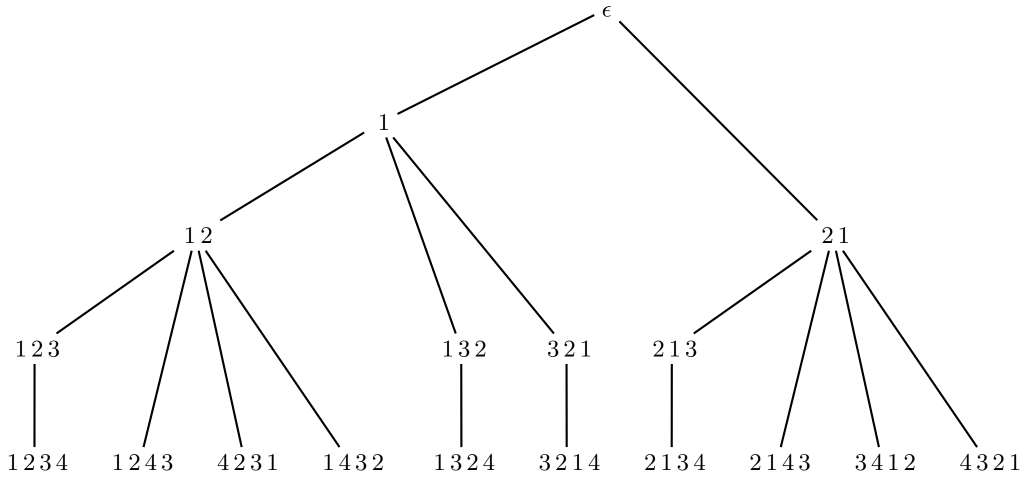


Fig. 4: The first five levels of the generating tree for involutions by recurrence.

- by point (i): one involution of length $t+1$, and so with $t+3 = k+1$ successors
- by point (ii) and (iii): $k-1$ involutions of length $t+2$, and so with $t+4 = k+2$ successors each.

3.3 Pure involutions

A *pure involution* (or *fixed point free involution*) is an involution with no fixed point, that is, with no entries i with $\pi(i) = i$. So, a pure involution is a permutation which is both an involution and a derangement and let $\mathfrak{P}_n$ denote the set of length-n pure involutions. Clearly $\mathfrak{P}_n = \varnothing$ for any odd n. The next lemma is the restriction of Lemma 3 to involutions where fixed points are disallowed. Its algorithmic counterpart is the generating procedure given in Algorithm 5 and its corresponding ECO rule is given in Table 1.

Algorithm 5 Procedure generating length-n pure involution. Initially $\pi = 1\,2\,3\,\ldots\,n$ and the main call is `gen_P_Inv(0)`.

```
procedure gen_P_Inv(t)
local i,σ;
if t = n then Print(π);
else π := π · ⟨t+1, t+2⟩; gen_P_Inv(t+2); π := π · ⟨t+1, t+2⟩;
     for i from 1 to t do
          σ := ⟨i, t+1⟩ · ⟨π[i], t+2⟩ · ⟨t+1, t+2⟩;
          π := π · σ; gen_P_Inv(t+2); π := π · σ^{-1};
     end do
end if
end procedure.
```

Lemma 4 *Let $\mathfrak{P}_0 = \{\epsilon\}$. If $\pi \in \mathfrak{P}_t$, $t \geq 0$, then*

- $\overline{\overline{\pi}} \cdot \langle t+1, t+2\rangle$ *is in* $\mathfrak{P}_{t+2}$;
- *for all* $i \in \{1, 2, \dots t\}$, $\overline{\overline{\pi}} \cdot \sigma$ *is in* $\mathfrak{P}_{t+2}$ *where* $\sigma = \langle i, t+1\rangle \cdot \langle \pi(i), t+2\rangle \cdot \langle t+1, t+2\rangle$.

Moreover, any permutation in $\mathfrak{P}_t$, with t even, $t \geq 2$, can be uniquely obtained from a permutation in $\mathfrak{P}_{t-2}$ by one of the transformations above.

4 Derangements

A *derangement* is a permutation with no fixed points; let $\mathfrak{D}_n$ denote the set of derangements of length n. The lemma below gives a recursive definition for $\mathfrak{D}_n$; see Table 1 for its corresponding ECO rule and Figure 5 for the induced generating tree. Its algorithmic version is given by Algorithm 6.

Algorithm 6 Procedure generating the set of all length-n derangements. π is initialized by $1\,2\,3\,\dots\,n$, and the main call is `gen_Der(0)`.

```
procedure gen_Der(t)
local i,σ;
if t = n then Print(π);
else for i from 1 to t do
          π := π · ⟨i,t+1⟩; gen_Der(t+1); π := π · ⟨i,t+1⟩;
     end do
     if t ≤ n−2 then
          π := π · ⟨t+1,t+2⟩; gen_Der(t+2); π := π · ⟨t+1,t+2⟩;
          for i from 1 to t do
               σ := ⟨i,t+1⟩ · ⟨π[i],t+2⟩ · ⟨t+1,t+2⟩;
               π := π · σ; gen_Der(t+2); π := π · σ^{-1};
          end do
     end if
end if
end procedure.
```

Lemma 5 $\mathfrak{D}_0 = \{\epsilon\}$. *If $\pi \in \mathfrak{D}_t$, $t \geq 0$, then*

(i) for all $i \in \{1, 2, \dots t\}$, $\overline{\pi} \cdot \langle i, t+1\rangle$ *is in* $\mathfrak{D}_{t+1}$,

(ii) $\overline{\overline{\pi}} \cdot \langle t+1, t+2\rangle$ *is in* $\mathfrak{D}_{t+2}$,

(iii) for all $i \in \{1, 2, \dots t\}$, $\overline{\overline{\pi}} \cdot \langle i, t+1\rangle \cdot \langle \pi(i), t+2\rangle \cdot \langle t+1, t+2\rangle$ *is in* $\mathfrak{D}_{t+2}$.

Moreover, any permutation in $\mathfrak{D}_n$, $n \geq 2$, can be uniquely obtained from a permutation in $\mathfrak{D}_{n-1}$ or $\mathfrak{D}_{n-2}$ by one of the three transformations above.

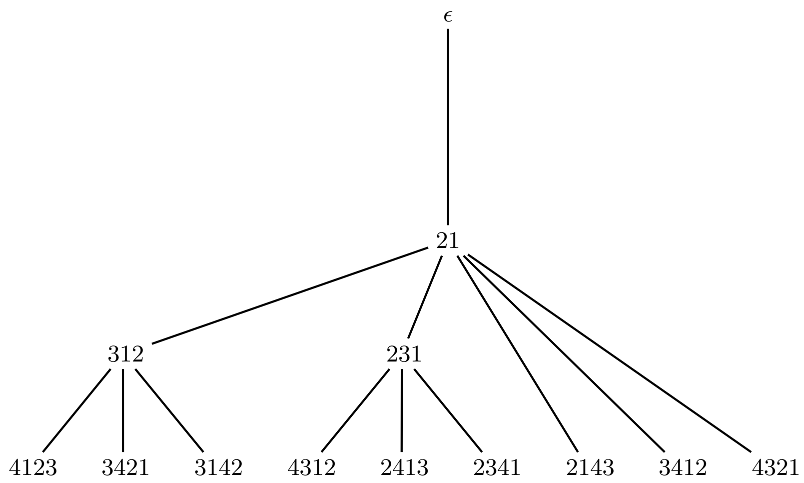


Fig. 5: The first five levels (the second level is empty) of the generating tree for derangements.

Proof: It is routine to check that for a $\pi \in \mathfrak{D}_t$, $t \geq 0$, the permutations obtained from π by applying each of the three transformations above is a derangement.

Conversely, for every $\tau \in \mathfrak{D}_n$, $n \geq 2$, we define a permutation π of length $n-1$ or $n-2$ such that τ is uniquely obtained from π by one of these transformations. For a $\tau \in \mathfrak{D}_n$ one of the following cases occurs:

(1) n belongs to a cycle of τ of length at least three, or

(2) n belongs to a length-2 cycle of τ and $\tau(n) = n-1$ (and so, $\tau(n-1) = n$), or

(3) n belongs to a length-2 cycle of τ and $\tau(n) \neq n-1$.

In case (1) , let $i = \tau^{-1}(n)$ and define π as the length-$(n-1)$ permutation with $\pi(j) = \tau(j)$ for all j, $1 \leq j \leq n-1$, except $\pi(i) = \tau(n) \neq i$. Clearly $\pi \in \mathfrak{D}_{n-1}$, $\tau = \overline{\pi} \cdot \langle i, n \rangle$ and n belongs to a cycle of τ of length at least three. In this case τ is uniquely obtained from a $\pi \in \mathfrak{D}_{n-1}$ and a $i \in \{1, 2, \ldots, n-1\}$.

In case (2) the length-$(n-2)$ permutation π defined by $\pi(j) = \tau(j)$ for all j, $1 \leq j \leq n-2$ is in $\mathfrak{D}_{n-2}$ and $\overline{\overline{\pi}} \cdot \langle n-1, n \rangle = \tau$. In this case τ is uniquely obtained from a $\pi \in \mathfrak{D}_{n-2}$.

Finally, let τ be as in case (3) above and define the length-$(n-2)$ permutation π by

$$\pi(j) = \begin{cases} \tau(j) & \text{if} \quad 1 \leq j \leq n-2, j \neq i, k \\ i & \text{if} \quad j = k \\ \tau(n-1) & \text{if} \quad j = i \end{cases}$$

where $i = \tau(n) = \tau^{-1}(n)$ and $k = \tau^{-1}(n-1)$. In other words, π is the permutation obtained from τ by deleting the length-2 cycle containing n and then replacing $(n-1)$ in its cycle by $i = \tau(n)$, thus $\pi \in \mathfrak{D}_{n-2}$. For example $\pi = 2\,1$ corresponds to $\tau = 4\,3\,2\,1$ and to $\tau = 3\,4\,1\,2$. π and τ are related by $\tau = \overline{\overline{\pi}} \cdot \langle \pi^{-1}(i), n-1 \rangle \cdot \langle i, n \rangle \cdot \langle n-1, n \rangle$ and so any permutation in $\mathfrak{D}_n$ satisfying case (3) above is uniquely obtained from a $\pi \in \mathfrak{D}_{n-2}$ and an $i \in \{1, 2, \ldots, n-2\}$. If for a given π as in the previous

object	ECO rule	k
Permutations	$root : (2)$ $rule : (k) \rightsquigarrow (k+1)^k$	$k-1=$ the length
Bell permutations	(2) $(k) \rightsquigarrow (k)^{k-1}(k+1)$	$k-1=\sharp$ of cycles
Involutions	(2) $(k) \rightsquigarrow (k-1)^{k-1}(k+1)$	$k-1=\sharp$ of fixed points
	(2) $(k) \overset{1}{\rightsquigarrow} (k+1)$ $\overset{2}{\rightsquigarrow} (k+2)^{k-1}$	$k-2=$ the length
Pure Involutions	(1) $(k) \rightsquigarrow (k+2)^k$	$k-1=$ the length
Derangements	(1) $(k) \overset{1}{\rightsquigarrow} (k+2)^{\frac{k-1}{2}}$ $\overset{2}{\rightsquigarrow} (k+4)^{\frac{k+1}{2}}$	$\frac{k-1}{2}=$ the length
Bessel permutations	(2) $(2) \rightsquigarrow (2)(3)$ $(k) \rightsquigarrow (k-1)(k)^{k-2}(k+1),$ if $k>2$	$k-1=$ the length

Tab. 1: ECO systems for some classes of restricted permutations together with the meaning of the parameter k.

expression for τ, i covers $\{1, 2, \ldots, n-2\}$ so does $\pi^{-1}(i)$, and replacing $\pi^{-1}(i)$ by i this expression becomes $\tau = \overline{\overline{\overline{\pi}}} \cdot \langle i, n-1\rangle \cdot \langle \pi(i), n\rangle \cdot \langle n-1, n\rangle$, which completes the proof. $\square$

Now we show that the construction in lemma above is related to the ECO rule for derangements in Table 1. Let π be a derangement of length t. It produces $2t+1$ successors: t derangements of length $t+1$ by point (i), and $t+1$ derangements of length $t+2$ by point (ii) and (iii). Then, t is related to k, the number of successors of π, by $k = 2t+1$. Suppose now that π has k successors. Its length is $t = \frac{k-1}{2}$ and it produces

- by point (i): $t = \frac{k-1}{2}$ derangements, each of them of length $t+1$ and so with $2(t+1)+1 = k+2$ successors;
- by point (ii) and (iii): $t+1 = \frac{k+1}{2}$ derangements, each of them of length $t+2$ and so with $2(t+2)+1 = k+4$ successors.

5 Complexity

If a generating algorithm produces combinatorial objects so that only a constant amount of computation is done between successive objects, in an amortized sense, then one says that it runs in *constant amortized time* (*CAT*). Now we explain why the generating algorithms presented here are CAT.

A recursive generating algorithm satisfying the following properties is a CAT algorithm, see for instance (20): The amount of computation in each call is proportional to the number of recursive calls produced by it, and each call

1. is a terminal call and produces a combinatorial object, or
2. produces at least two recursive calls, or
3. produces only one recursive call which in turn has either property 1 or 2.

All the algorithms presented in this paper are CAT. Indeed, if sets (of tails or of fixed points) are represented by linked lists, then in each generating procedure the amount of computation in each call is proportional to the number of recursive calls produced. In addition, except for the initial call of the procedure generating derangements (that is, `gen_Der(0)`), all the algorithms satisfy the three CAT properties and that is a direct consequence of the shape and uniformity of the corresponding ECO rules and operators.

6 Concluding remarks

The techniques presented here were anticipated in (1; 7). Nevertheless, the previous approaches used strings of integers for encoding the objects and it was necessary to have another algorithm to retrieve the object. Also, the application of our method to other classes of permutations such as Bessel permutations (13; 8) (see the last row of Table 1 for their ECO system), or generally, to other classes of combinatorial objects, remains an interesting problem.

Acknowledgements

The author wish to thank the anonymous referees, whose remarks and suggestions greatly improved the overall quality of the paper.

References

[1] S. Bacchelli, E. Barcucci, E. Grazzini, E. Pergola. Exhaustive generation of combinatorial objects by ECO, *Acta Informatica*, 2004, 40(8), 585–602.

[2] E. Barcucci, A. Del Lungo, E. Pergola, R. Pinzani. ECO: a methodology for the enumeration of combinatorial objects, *Journal of Difference Equations and Applications*, 1999, 5, 1–56.

[3] J.-L. Baril. Gray code for permutations with a fixed number of cycles, *Discrete Mathematics*, 2007, 307, 1559–1571.

[4] J.-L. Baril. More restrictive Gray codes for some classes of pattern avoiding permutations, *Information Processing Letters*, 2009, 109, 799–804.

[5] J.-L. Baril, D. Phan-Thuan. ECO-generation for p-generalized Fibonacci and Lucas permutations, *Journal of Pure Mathematics and Applications*, 2006, 17 (1-2), 19–37.

[6] J.-L. Baril, V. Vajnovszki, Gray code for derangements, *Discrete Applied Mathematics*, 2004, 140, 207–221.

[7] A. Bernini, E. Grazzini, E. Pergola, R. Pinzani. A general exhaustive generation for Gray structures, *Acta Informatica*, 2007, 44(5), 361–376.

[8] A. Claesson. Generalized pattern avoidance, *Europ. J. Combinatorics*, 2001, 22, 961–971.

[9] W.M.B. Dukes, M. F. Flanagan, T. Mansour, V. Vajnovszki. Combinatorial Gray codes for classes of pattern avoiding permutations, *Theoretical Computer Science*, 2008, 396, 35–49 (`arXiv:0704.2048v1`).

[10] S. Effer, F. Ruskey. A CAT algorithm for listing permutations with a given number of inversions, *Information Processing Letters*, 2003, 86(2), 107–112.

[11] G. Ehrlich. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations, *J. Assoc. Comput. Mach.*, 1973, 20, 500–513.

[12] L. Ferrari, E. Pergola, R. Pinzani, S. Rinaldi. Jumping succession rules and their generating functions, *Discrete Mathematics*, 2003, 271, 29–50.

[13] P. Flajolet, R. Schott. Non-overlapping partitions, continued fractions, Bessel functions and a divergent series, *Europ. J. Combinatorics*, 1990, 11, 421–432.

[14] D. Foata and M.-P. Schützenberger. Théorie géometrique des polynômes Euleriens, *Lectures Notes in Math.* 138, Springer, Berlin, 1970.

[15] S.M. Johnson. Generation of permutations by adjacent transpositions, *Math. Comp.*, 1963, 17, 282–285.

[16] G. Labelle, P. Leroux, E. Pergola, R. Pinzani. Stirling numbers interpolation using permutations with forbidden subsequences, *Discrete Mathematics*, 2002, 246(1-3), 177–195.

[17] D.H. Lehmer. Permutations by adjacent interchanges, *Amer. Math. Montly*, 1965, 72, 36–46.

[18] M. Poneti, V. Vajnovszki. Generating restricted classes of involutions, Bell and Stirling permutations, *Europ. J. Combinatorics*, 2010, 31, 553–564.

[19] F. Ruskey, D. Roelants van Baronaigien. Fast recursive algorithm for generating combinatorial objects, *Congressus Numeratium*, 1984, 41, 53–62.

[20] F. Ruskey. *Combinatorial generation*. Book in preparation.

[21] R. Sedgewick. Permutation generation methods, *Comput. Surveys.* 1977, 9(2), 137–164.

[22] N. Sloane. The on-line encyclopedia of integer sequences, `http://www.research.att.com/~njas/sequences`.

[23] R. Stanley. *Enumerative Combinatorics*, Cambridge University Press, Cambridge, England, 1997.

[24] H. Steinhaus. *One hundred problems in elementary mathematics*, Dover Publications 1979 (In Polish 1958).

[25] H.F. Trotter. Perm (Algorithm 115), *Comm. ACM*, 1962, 5(8), 434–435.

[26] V. Vajnovszki. Generating involutions, derangements, and relatives by ECO, *GASCom'07*, Dijon–France, Sept. 2006.

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