Generating restricted classes of involutions, Bell and Stirling permutations

Maddalena PONETI
Dipartimento di Scienze Matematiche e Informatiche "R. Magari"
Università di Siena, Pian dei Mantellini 44, Siena - Italy
poneti@unisi.it

Vincent VAJNOVSZKI
LE2I, Université de Bourgogne
BP 47870, 21078 Dijon Cedex, France
vvajnov@u-bourgogne.fr

July 8, 2008

Abstract

We present a recursive generating algorithm for unrestricted permutations which is based on both the decomposition of a permutation as a product of transpositions and as a union of disjoint cycles. It generates permutations at each recursive step and slight modifications of it produce generating algorithms for Bell permutations and involutions. Further refinements yield algorithms for these classes of permutations subject to additional restrictions: a given number of cycles or/and fixed points. As particular cases, we obtain generating algorithms for permutations counted by the Stirling numbers of the first and second kind, even permutations, fixed-point-free involutions and derangements. All these algorithms run in constant amortized time.

1 Introduction and motivation

There is a great deal of literature on the exhaustive generation of permutations, beginning with the campanologists' historical works [19, 21] followed by more systematical approaches [11, 13, 20, 22]; see [18] for a survey or the seminal book of D. Knuth [14]. More recently, a great interest was shown in the generation of particular classes of permutations: involutions [25], derangements [3], with fixed number of cycles [2] or inversions [7, 24], with forbidden patterns [6].

A recursive generating algorithm is given in [4] where Catalan objects are generated at each recursive step (not only terminal ones) and in [23] particular classes of permutations are generated based on their representation as product of transpositions. The present work is motivated by these papers. More precisely, here we give a new generating algorithm for unrestricted permutations which is based on both the decomposition of a permutation as a product of transpositions and as a union of disjoint cycles. As in [4] our algorithm generates objects at each recursive step and we show that a slight modification of it produces similar algorithms for Bell permutations and involutions. Further refinements yield algorithms for these classes of permutations subject to additional restrictions: a given number of cycles or/and fixed points. As particular cases, we obtain generating algorithms for permutations counted by the Stirling numbers of the first and second kind, even permutations, fixed-point-free involutions and derangements. All these algorithms run in constant amortized time. This is the first paper presenting an algorithm where its versions produce a large number of classes of restricted permutations and it is an extended form of the preliminary conference version of [8].

2 Preliminaries

A length-n permutation is a bijection from the set $[1, n] = \{1, 2, \ldots, n\}$ onto itself. The more common representation of a permutation π is the one-line notation: $\pi(1)\,\pi(2)\,\ldots\,\pi(n)$. Two alternative powerful ways to represent a permutation are the standard decomposition and cycle representation, both defined below. In the following we denote by S_n the set of all $n!$ length-n permutations.

2.1 Standard decomposition

We denote by $\langle \ell, j\rangle$ the transposition of the element in position ℓ and the element in position j, that is the permutation π of appropriate length with $\pi(i) = i$ for all i, except $\pi(\ell) = j$ and $\pi(j) = \ell$; clearly $\langle \ell, j\rangle = \langle j, \ell\rangle$. For instance the permutation $4\,2\,3\,1 \in S_4$ is the transposition $\langle 1, 4\rangle$ and the product of a permutation with a transposition is the usual product of two permutations, e.g., if $\pi = 2\,1\,3\,4 \in S_4$ then $\pi \cdot \langle 1, 4\rangle = 4\,1\,3\,2$.

Lemma 1. *Any permutation $\pi \in S_n$ can uniquely be written*

$$\pi = \prod_{i=1}^{n} \langle p_i, i\rangle = \langle p_1, 1\rangle \cdot \langle p_2, 2\rangle \cdot \langle p_2, 3\rangle \cdot \ldots \cdot \langle p_n, n\rangle \text{ with } p_i \in [1, i]. \tag{1}$$

Proof. In spite of this result being quite intuitive and folkloric we give below a constructive proof because this construction will be used later. For any $\pi \in S_n$ we construct iteratively the n-sequence $(p_1, p_2, \ldots, p_n) \in [1, 1] \times [1, 2] \times \ldots \times [1, n]$ which satisfies relation (1): Run through the entries of π from right to left, setting $p_i = \pi^{-1}(i)$ and replacing π by $\pi \cdot \langle \pi^{-1}(i), i\rangle$. In particular, when i is a fixed point in the current permutation (i.e., $\pi(i) = i$), then $p_i = i$ and $\langle \pi^{-1}(i), i\rangle$ is the identity.

Intuitively, what we do is to construct π^{-1} by sorting π using selection sort: for i running from n down to 1 we move i (which is in one of the positions $1, 2, \ldots, i$) into position i by exchanging it with the element that is in position i. Since the permutation we use to sort π is the inverse of the permutation given in the right side of (1), the original permutation π equals $\prod_{i=1}^{n} \langle p_i, i\rangle$ and this construction is an injective mapping from S_n to $[1, 1] \times [1, 2] \times \ldots \times [1, n]$. Finally, cardinality arguments show that this construction yields a bijection from S_n onto $[1, 1] \times [1, 2] \times \ldots \times [1, n]$. $\square$

For example, the decomposition of $\pi = 4\,1\,3\,2$ given by relation (1) is $\langle 1, 1\rangle \cdot \langle 1, 2\rangle \cdot \langle 3, 3\rangle \cdot \langle 1, 4\rangle$. If we relax the condition $p_i \in [1, i]$, then the decomposition of a permutation as product of transpositions is not necessarily unique, for instance the permutation π above can be written as $\langle 4, 1\rangle \cdot \langle 4, 2\rangle \cdot \langle 3, 3\rangle \cdot \langle 4, 4\rangle$. For a permutation $\pi \in S_n$, its decomposition given by (1) is called its *standard decomposition.*

Recall that given a group G with a generating set $U \subset G$, the *directed Cayley graph* is constructed as follows: the vertex set is G and there is a directed edge from a to b if there exists $u \in U$ with $b = a \cdot u$. If the generating set is such that $u \in U$ implies that $u^{-1} \in U$, then the Cayley graph is called *undirected*. See for example [10] for more details about Cayley graphs. Actually, the decomposition in relation (1) gives the path from $1\,2\,\ldots\,n \in S_n$ to π in a spanning tree of the undirected Cayley graph of the permutations group S_n with generating set $\{\langle \ell, j\rangle\}_{1\le \ell<j\le n}$. For $n = 4$ such a tree is depicted in Figure 1.

2.2 Cycle representation

A *cycle* C in a permutation $\pi \in S_n$ is a sequence $C = (a_0 a_1 \ldots a_{j-1})$ such that $\pi(a_i) = a_{(i+1) \bmod j}$ for all i, $0 \le i \le j-1$. Obviously, the cycle $(a_i a_{i+1} \ldots a_{j-1} a_0 a_1 \ldots a_{i-1})$ is equivalent to the cycle C and we choose to represent cycles with their smallest element last. Any permutation is the union of disjoint cycles and the *cycle representation* of a permutation is obtained by imposing the condition that the cycles are written in increasing order of their smallest element (that is, their last element); for example, the cycle representation of $4\,2\,5\,1\,7\,6\,3 \in S_7$ is $(4\,1)(2)(5\,7\,3)(6)$. It is worth mentioning that if π' is the permutation in S_n obtained from $\pi \in S_n$ by erasing the parentheses in the cycle representation of π, then π can be uniquely recovered from π' and the transformation $\pi \hookrightarrow \pi'$ is a bijection from S_n onto itself. This mapping is essentially the *transformation fondamentale* of [9, Proposition 1.3.1], see also [17, p. 17].

The standard decomposition and cycle representation are intimately related. For $\pi \in S_n$ with its standard decomposition given by relation (1) let j be a position such that $p_i = i$ for all $i \ge j$. It follows that j is a fixed point of π and so the rightmost *deranged* point in π (that is, the largest i with $\pi(i) \ne i$) equals the largest i with $p_i \ne i$. This makes consistent the following definition: For $\pi \in S_n$, $D(\pi) = \max_i\{\pi(i) \ne i\} = \max_i\{p_i \ne i\}$, and by convention if π has no deranged points (that is, π is the identity permutation), then $D(\pi) = 1$. We note that it might happen that $p_i = i$ but $\pi_i \ne i$ when $i < D(\pi)$.

Lemma 2. *Let $\pi \in S_n$ and j be a fixed point for π. For any $\ell \ne j$, $1 \le \ell \le n$, we have*

1. *$\pi \cdot \langle \ell, j\rangle$ is the permutation obtained from π by inserting j into the cycle containing ℓ between ℓ and $\pi(\ell)$.*
2. *if π has k cycles, then $\pi \cdot \langle \ell, j\rangle$ has $k-1$ cycles.*

Proof. 1. If ℓ is also a fixed point for π the statement is obvious; otherwise it results directly from the form of π and $\pi \cdot \langle \ell, j\rangle$ given below.

$$\pi = \begin{pmatrix} \cdots & \ell & \cdots & j & \cdots \\ \cdots & \pi(\ell) & \cdots & j & \cdots \end{pmatrix}$$

$$\pi \cdot \langle \ell, j\rangle = \begin{pmatrix} \cdots & \ell & \cdots & j & \cdots \\ \cdots & j & \cdots & \pi(\ell) & \cdots \end{pmatrix}.$$

2. Multiplying π by $\langle \ell, j\rangle$ results in the merging of two cycles: the cycle containing ℓ and the length-one cycle containing j. $\square$

Observe that, with the notations above, the number of fixed points of $\pi \cdot \langle \ell, j\rangle$ equals those of π, minus two if ℓ is a fixed point of π or minus one otherwise.

We denote by $S_{n,k}$ the set of permutations in S_n with exactly k cycles and its cardinality is the signless Stirling number of the first kind [17, p. 18]. The next corollary is a particular case of the previous lemma.

Corollary 1. *Let τ be a permutation in $S_{n,k}$ such that $D(\tau) < n$. For any j, $D(\tau) < j \leq n$, and any ℓ, $1 \leq \ell < j$, the permutation $\pi = \tau \cdot \langle \ell, j \rangle$ is in $S_{n,k-1}$.*

The lemma below shows that each length-n permutation with k cycles, other than the identity one, can be obtained uniquely from a permutation with $k+1$ cycles by the above transformation and it is the core of our generating algorithms.

Lemma 3. *For each $\pi \in S_{n,k}$, $k \neq n$, there exists a unique triplet (τ, ℓ, j) such that $\pi = \tau \cdot \langle \ell, j \rangle$ where:*

- *$\tau \in S_{n,k+1}$ with $D(\tau) < D(\pi)$,*
- *$j > D(\tau)$,*
- *$\ell < j$.*

Proof. Firstly, since $k \neq n$, π is not the identity permutation, and so $D(\pi) > 1$. Let $\prod_{i=1}^{n} \langle p_i, i \rangle$ be the standard decomposition of π, and define: $j = D(\pi)$, $\ell = p_j$ and $\tau = \prod_{i=1}^{n} \langle t_i, i \rangle$ with $t_i = p_i$ for all i except that $t_j = j$. Then, $\tau = \pi \cdot \langle \ell, j \rangle$ and the triplet (τ, ℓ, j) satisfies the statement of the lemma and the uniqueness results from the unique standard decomposition of π. $\square$

The generating tree induced by the recursive construction of $S_n = \cup_{k=1}^{n} S_{n,k}$, given by Lemma 3 for $n = 4$, is presented in Figure 1: $1\,2 \ldots n$ is the root and if $\tau \in S_{n,k+1}$ is the parent of $\pi \in S_{n,k}$, then the transposition $\langle \ell, j \rangle$ which precedes π is such that $\pi = \tau \cdot \langle \ell, j \rangle$.

2.3 Bell permutations

Definition 1. *The set B_n of length-n* Bell permutations *is the set of permutations in S_n, where each cycle is a decreasing sequence of integers (assuming that cycles are represented with their smallest element last).*

B_n is in bijection with the set of all partitions of $[1, n]$: each cycle in $\pi \in B_n$ represents a block of the partition. For instance the partition corresponding to $4\,2\,7\,1\,3\,6\,5 \in S_7$ is $\{4, 1\}\{2\}\{7, 5, 3\}\{6\}$; thus, B_n is counted by the nth Bell number (sequence `A000110` in [16]). See also [12] for an alternative definition of Bell permutations in terms of pattern avoidance.

In the standard decomposition $\prod_{i=1}^{n} \langle p_i, i \rangle$ of $\pi \in S_n$ we say that two transpositions $\langle p_u, u \rangle$ and $\langle p_v, v \rangle$, $u < v$, *meet* if $p_v \in \{p_u, u\}$ and *meet at left* if $p_v = p_u$. Although we do not use this fact below, it is interesting to note that a short proof reveals that π is a Bell permutation iff any two transpositions in the standard decomposition of π that meet must meet at left.

We say that i is a *tail* of $\pi \in S_n$ if i is minimal in its cycle. In particular if $\pi \in B_n$, then $\pi(i)$ is the largest element of that cycle. We denote by $\mathrm{Tail}(\pi)$ the set of tails of $\pi \in S_n$ and by $B_{n,k}$ the set of Bell permutations with k cycles, and $B_{n,k}$ is counted by the Stirling number of the second kind [17, pp. 33]. Below are the 'Bell' counterparts of Corollary 1 and Lemma 3.

Lemma 4. *Let τ be a permutation in $B_{n,k}$ and $D(\tau) < n$. For any j, $D(\tau) < j \leq n$, and any $\ell \in \mathrm{Tail}(\tau)$, $1 \leq \ell < j$, the permutation $\pi = \tau \cdot \langle \ell, j \rangle$ is in $B_{n,k-1}$.*

Proof. By Lemma 2, j is inserted after ℓ in the cycle containing ℓ, and π has $k - 1$ cycles. $\square$

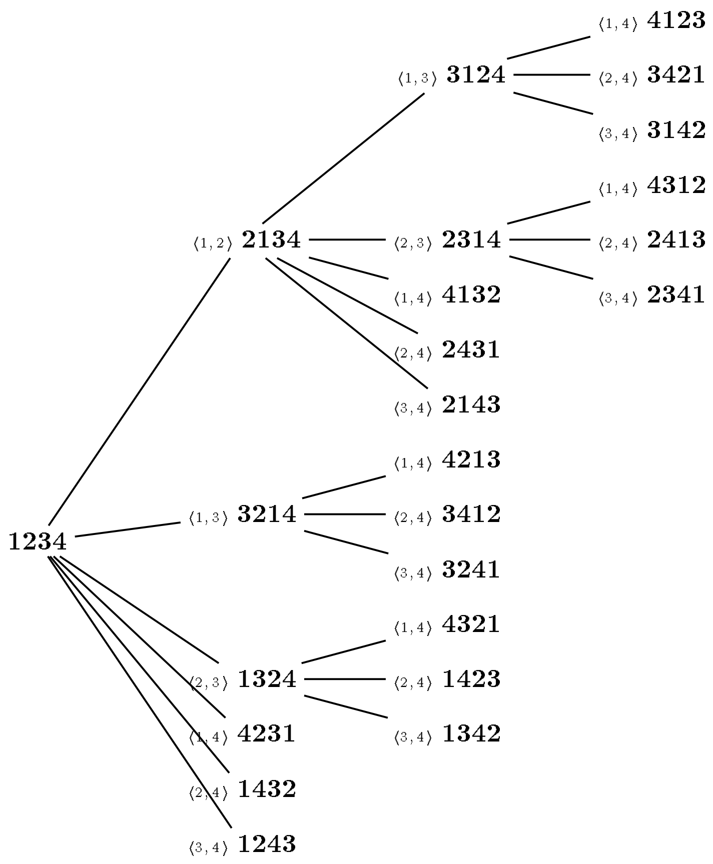

Figure 1: The tree induced by the recursive construction of S_4 given by Lemma 3. It corresponds to the generating tree induced for $n = 4$ by the call `gen_N()`, `gen_P({1},2)` and the call `gen_A({1},2)` with $X = T \cup \{j\}$. Each permutation is preceded by the transposition which transforms its parent in this permutation. The root, at level zero, is the identity permutation and at level d there are all the permutations with $4 - d$ cycles.

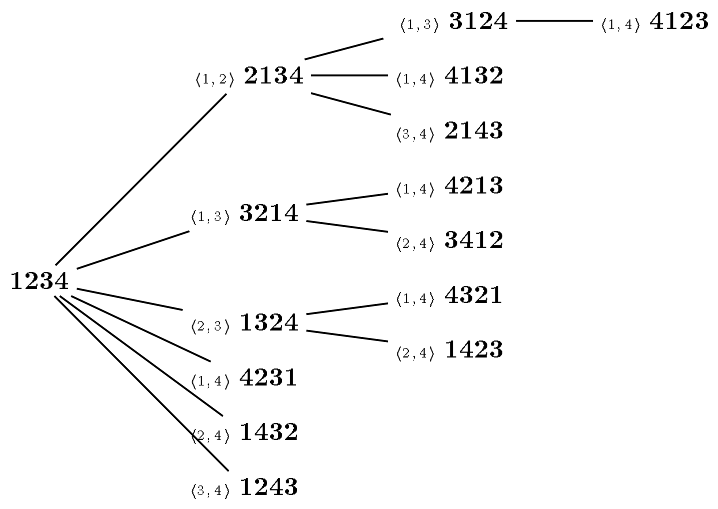

Figure 2: The tree induced by the recursive construction of B_4 given by Lemma 5. It is the restriction of the tree in Figure 1 to Bell permutations and corresponds to the generating tree induced by the call of `gen_A({1},2)`, for $n = 4$ and with $X = T$.

Lemma 5. *For each $\pi \in B_{n,k}$, $k \neq n$, there exists a unique triplet (τ, ℓ, j) such that $\pi = \tau \cdot \langle \ell, j\rangle$ where:*

- *$\tau \in B_{n,k+1}$ with $D(\tau) < D(\pi)$,*
- *$j > D(\tau)$,*
- *$\ell \in \mathrm{Tail}(\tau)$ and $\ell < j$.*

In addition, $\mathrm{Tail}(\pi) = \mathrm{Tail}(\tau) \setminus \{j\}$.

Proof. As in the proof of Lemma 3, if $\prod_{i=1}^{n}\langle p_i, i\rangle$ is the standard decomposition of π, then $j = D(\pi)$, $\ell = p_j$ and $\tau = \pi \cdot \langle \ell, j\rangle$. It is easy to check that $\tau \in B_{n,k+1}$, $D(\tau) < D(\pi)$ and $\mathrm{Tail}(\pi) = \mathrm{Tail}(\tau) \setminus \{j\}$. The uniqueness is a consequence of Lemma 3. $\square$

The generating tree induced by the recursive construction of $B_n = \cup_{k=1}^{n} B_{n,k}$, given by Lemma 5 for $n = 4$, is presented in Figure 2.

2.4 Involutions

A permutation $\pi \in S_n$ is an *involution* if $\pi \cdot \pi$ is $1\,2\,3\,\ldots\,n$, the identity in S_n; or equivalently, any cycle in π has length at most two. We denote by I_n (*resp.* $I_{n,k}$) the set of length-n involutions (*resp.* length-n involutions with k cycles); clearly $k \geq \lceil \frac{n}{2} \rceil$, $I_n \subset B_n$ and $I_{n,k} \subset B_{n,k}$. For $\pi \in S_n$ we denote by $\mathrm{Fix}(\pi)$ the set of fixed points of π and so $\mathrm{Fix}(\pi) \subseteq \mathrm{Tail}(\pi)$. Below are the 'involution' counterparts of Corollary 1 and Lemma 3. Their proofs are similar to those of Lemmata 4 and 5 and can be easily recovered by the reader.

Lemma 6. *Let $\tau \in I_{n,k}$ and $D(\tau) < n$. For any j, $D(\tau) < j \leq n$, and $\ell \in \mathrm{Fix}(\tau)$, $1 \leq \ell < j$, the permutation $\pi = \tau \cdot \langle \ell, j\rangle$ is in $I_{n,k-1}$.*

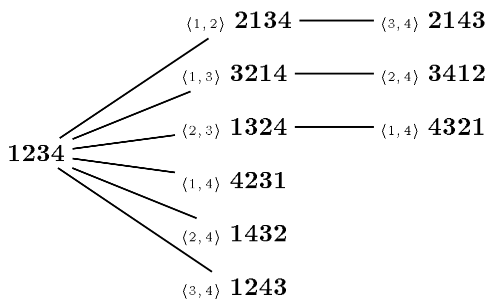

Figure 3: The tree induced by the recursive construction of I_4 given by Lemma 7. It is the restriction of the tree in Figure 2 (and the one in Figure 1) to involutions and corresponds to the generating tree induced by the call of `gen_A({1},2)`, for $n = 4$ and with $X = T \setminus \{\ell\}$.

Lemma 7. *For each $\pi \in I_{n,k}$, $k \neq n$, there exists a unique triplet (τ, ℓ, j) such that $\pi = \tau \cdot \langle \ell, j \rangle$ where:*

- *$\tau \in I_{n,k+1}$ with $D(\tau) < D(\pi)$,*
- *$j > D(\tau)$,*
- *$\ell \in \mathrm{Fix}(\tau)$ and $\ell < j$.*

In addition, $\mathrm{Fix}(\pi) = \mathrm{Fix}(\tau) \setminus \{\ell, j\}$.

The generating tree induced by the recursive construction of $I_n = \cup_{k=\lceil \frac{n}{2} \rceil}^{n} I_{n,k}$, given by Lemma 7 for $n = 4$, is presented in Figure 3.

3 Generating algorithms

The next remark gives the interpretation of Lemmata 3, 5 and 7 in terms of generating trees (or ECO operator), see for instance [1] and the references therein.

Remark 1. *Let $\tau \in S_n$ with $D(\tau) < n$ and $1 < k \leq n$.*

- *If $\tau \in S_{n,k}$, then define $S_\tau = \{\tau \cdot \langle \ell, j \rangle \mid j > D(\tau), \ell < j\}$. By Lemma 3 it follows that the family $\{S_\tau\}_{\tau \in S_{n,k}}$ is a partition of $S_{n,k-1}$.*
- *If $\tau \in B_{n,k}$, then define $B_\tau = \{\tau \cdot \langle \ell, j \rangle \mid j > D(\tau), \ell \in \mathrm{Tail}(\tau), \ell < j\}$. By Lemma 5 it follows that the family $\{B_\tau\}_{\tau \in B_{n,k}}$ is a partition of $B_{n,k-1}$.*
- *If $\tau \in I_{n,k}$, then define $I_\tau = \{\tau \cdot \langle \ell, j \rangle \mid j > D(\tau), \ell \in \mathrm{Fix}(\tau), \ell < j\}$. By Lemma 7 it follows that the family $\{I_\tau\}_{\tau \in I_{n,k}}$ is a partition of $I_{n,k-1}$.*

Alternatively, S_τ (resp. B_τ, I_τ) is the set of successors of τ in the generating tree induced by Lemma 3 (resp. 5, 7); see Figure 1 (resp. 2, 3).

As an application of Lemma 3 and with the first point of the remark above we obtain the naive algorithm, `gen_N` in Figure 4 (a), for generating the set S_n; n and π are a global variables and initially $\pi = 1\,2 \ldots n$. If in a particular call the current permutation is π, then it produces

S_π, and gen_N is recursively called for each permutation in S_π. Note that, after each recursive call to gen_N, the current permutation π is reset to its initial value before this call. See Figure 1 for the generating tree induced by the call gen_N() with $n = 4$ and Table 1 for the list produced by this call. Generally, gen_N() produces S_n by covering the generating tree in pre-order, that is, visiting the root then visiting recursively the sub-trees.

This algorithm has two disadvantages: (i) to compute $D(\pi)$ requires generally a linear time in n, and (ii) as can be seen in the last two points of Remark 1, for some classes of permutations, ℓ does not cover an interval of integers. To eliminated these disadvantages we add to the generating procedure two additional parameters: q and T, where $q = D(\pi) + 1$ and T is the set of allowed values of ℓ for a given j. The procedure so obtained is gen_P in Figure 4 (b). In this case the main call becomes gen_P($\{1\}$,2) which corresponds to the initial permutation $\pi = 1\,2 \dots n$ and gen_P(T,q) produces several calls of gen_P($T \cup \{q, q+1, \dots j\}$,$j+1$), one for each ℓ, with $j \in [q, n]$. Note that after an iteration on j is completed, j is added to T. Also the call of this procedure with $q = n+1$ simply prints the current permutation and does not produce recursive calls.

This algorithm remains inefficient: sending a variable length set from a call to the next recursive call requires linear time and space. A last improvement is obtained by implementing the set T as a global variable represented by a linked list which is reset to its original value at the end of each call. In all the following algorithms, including Gen_P with the above final improvement but excluding gen_N, each call performs a constant number of operations on T: the addition of the largest element of T (the statement $T := T \cup \{j\}$ or the call of gen_P with $T \cup \{j\}$ as first parameter); the deletion of several of its largest elements (added iteratively by the statement $T := T \cup \{j\}$ at the end of the procedure); or the deletion of a given element (not necessarily the last one).

For the sake of conciseness, we choose to present all the algorithms with sets transmitted as parameters and having in mind that always a global linked list representation is possible for these sets and all the operations on this list are efficient; that is, done in constant time. Below we will use the following 'CAT principles', which are slight modifications of those in [15], in order to show that our algorithms produce classes of permutations in constant amortized time.

We call a recursive generating algorithm *amortized-recursive* if the total amount of computation in each call is proportional to the number of direct calls produced by this call. In other words, if in an algorithm each iteration of its loops (if any) produces a new recursive call, then it is amortized-recursive. In this case the total amount of computation of the algorithm is proportional to the total number of recursive calls, and we have:

Lemma 8. *(First CAT principle) A recursive generating algorithm runs in constant amortized time if it is amortized-recursive and each call (not only the terminal ones) produces a new object.*

In a recursive procedure we define the *degree* of a particular call of the procedure to be the number of 'immediate' calls that result.

Lemma 9. *(Second CAT principle) A recursive generating algorithm runs in constant amortized time if: (1) it is amortized-recursive, (2) each terminal call (degree-zero call) produces a new object, and (3) the number of degree-one calls is in $O(p)$, with p being the number of generated objects (or equivalently, the number of terminal calls).*

Proof. Let p be as in the statement of the lemma and denote by r the number of degree-one calls. If $r = 0$, then $p \geq \frac{\textit{the number of recursive calls}}{2}$ and the statement holds. Otherwise, $p \geq \frac{(\textit{the number of recursive calls}) - r}{2}$ and when $r \in O(p)$, again the statement holds. $\square$

```
procedure gen_N()
local $j, \ell$;
print($\pi$);
for $j := D(\pi)+1$ to $n$ do
    for $\ell := 1$ to $j-1$ do
        $\pi := \pi \cdot \langle \ell, j \rangle$;
        gen_N();
        $\pi := \pi \cdot \langle \ell, j \rangle$;
    end do
end do
end procedure.
```

(a)

```
procedure gen_P($T$,$q$)
local $j, \ell$;
print($\pi$);
for $j := q$ to $n$ do
    for $\ell \in T$ do
        $\pi := \pi \cdot \langle \ell, j \rangle$;
        gen_P($T \cup \{j\}$,$j+1$);
        $\pi := \pi \cdot \langle \ell, j \rangle$;
    end do
    $T := T \cup \{j\}$;
end do
end procedure.
```

(b)

Figure 4: (a) Algorithm producing the set S_n, and (b) its version where $q = D(\pi)+1$ and the set $T = [1, j-1]$ are parameters of the procedure. The call gen_P({1},2) produces S_n and in both cases n and π are global variables with π initialized by $1\,2\,\ldots\,n$.

If in gen_P we replace the first parameter of its recursive call by a *generic* parameter X, then we obtain the algorithm gen_A in Figure 5 (a), and for $X = T \cup \{j\}$ we retrieve gen_P. The next theorem says that, according to different instances of X, gen_A generates the sets S_n, B_n and I_n.

Theorem 1. *The algorithm* gen_A *produces in constant amortized time:*

1. *Unrestricted permutations if $X = T \cup \{j\}$;*
2. *Bell permutations if $X = T$;*
3. *Involutions if $X = T \setminus \{\ell\}$.*

Proof. (1) As mentioned above, when $X = T \cup \{j\}$, gen_P and gen_A coincide, and when the transposition $\langle \ell, j \rangle$ is applied, then $T = [1, j-1]$ and $\ell \in T$.
(2) If $X = T$, when the transposition $\langle \ell, j \rangle$ is applied to the current permutation, then $\ell \in T = [1, j-1] \cap \mathrm{Tail}(\pi)$ and by Lemma 5, gen_A produces Bell permutations.
(3) If $X = T \setminus \{\ell\}$, when the transposition $\langle \ell, j \rangle$ is applied to the current permutation, then $\ell \in T = [1, j-1] \cap \mathrm{Fix}(\pi)$ and, by Lemma 7, gen_A produces involutions.
In any case, gen_A satisfies the first CAT principle, disregarding the operations on the set T. Finally, the implementation of T (as mentioned before Lemma 8) by a global variable represented by a linked list yields CAT algorithms. $\square$

See Figures 1, 2 and 3 for the generating trees induced by the calls of gen_A({1},2) with $n = 4$ and corresponding to different instances of X. In Table 1 are the lists produced by these calls.

Permutations with a given number of cycles

The *level* of a particular call of a recursive algorithm is defined as follows: the main call has the level zero, and a recursive call at level d produces 'immediate' calls at level $d+1$. By the second

```
procedure gen_A($T$,$q$)
local $j,\ell$;
print($\pi$);
for $j := q$ to $n$ do
    for $\ell \in T$ do
        $\pi := \pi \cdot \langle \ell, j \rangle$;
        gen_A($X$,$j+1$);
        $\pi := \pi \cdot \langle \ell, j \rangle$;
    end do
    $T := T \cup \{j\}$;
end do
end procedure.
```

(a)

```
procedure gen_K($T$,$q$,$d$)
local $j,\ell$;
if $d = n-k$
then print($\pi$);
else for $j := q$ to $k+d+1$ do
        for $\ell \in T$ do
            $\pi := \pi \cdot \langle \ell, j \rangle$;
            gen_K($X$,$j+1$,$d+1$);
            $\pi := \pi \cdot \langle \ell, j \rangle$;
        end do
        $T := T \cup \{j\}$;
    end do
end if
end procedure.
```

(b)

Figure 5: (a) The generalization of gen_P; it produces the sets S_n, B_n and I_n according to different instances of X: S_n (for $X = T \cup \{j\}$), B_n (for $X = T$), I_n (for $X = T \setminus \{\ell\}$). (b) Algorithm producing $S_{n,k}$, $B_{n,k}$ ($k \neq 1$) and $I_{n,k}$ according to whether $X = T \cup \{j\}$, $X = T$, $X = T \setminus \{\ell\}$, respectively. d is the level of the call and k, the number of cycles, is a global variable.

point of Lemma 2, the permutations printed by gen_A at level d have $n-d$ cycles, and if we impose the condition that permutations are printed only at level $d = n-k$, then the sets $S_{n,k}$, $B_{n,k}$ and $I_{n,k}$ are obtained. However, not every permutation at level less than $n-k$ produces eventually a permutation at level $n-k$, and thus with k cycles. To ensure that the level $n-k$ is reached it is enough to impose the condition that $j \leq k+d+1$ on each call at level d. The algorithm thus obtained is gen_K in Figure 5 (b). The main call is gen_K({1},2,0), d is the level of the call and k, the number of cycles, is a global variable. The next theorem shows that gen_K is efficient.

Theorem 2. *The algorithm* gen_K *produces in constant amortized time:*

1. *Permutations with k cycles (counted by the signless Stirling number of the first kind) if $X = T \cup \{j\}$;*
2. *Bell permutations with k cycles, $k \neq 1$, (counted by the Stirling number of the second kind) if $X = T$;*
3. *Involutions with k cycles, $k \geq \lceil \frac{n}{2} \rceil$, (and so with $n-2k$ fixed points) if $X = T \setminus \{\ell\}$. In particular,* gen_K *produces fixed-point-free involutions when $2k = n$.*

Proof. (1) If $k \neq 1$, then each call of gen_K has degree at least two; and when $k = 1$ there is a single call of degree one, namely the call at level zero (the main call) which produces the permutation $2\,1\,3\ldots n$. We refer the reader to Figure 1 where the permutations with k cycles are at level $4-k$.
(2) Suppose that $k \neq 1$ and that π is a Bell permutations obtained at level $d < n-k$. It is easy to check that $\mathrm{Tail}(\pi) \cap [1, k+d]$ has at least two elements and so π has at least two successors. We refer the reader to Figure 2 where the Bell permutations with k cycles are at level $4-k$.

S_4	B_4	I_4	S_4	B_4	I_4	S_4	B_4	I_4
1234	✓	✓	2413			3241		
2134	✓	✓	2341			1324	✓	✓
3124	✓		4132	✓		4321	✓	✓
4123	✓		2431			1423	✓	
3421			2143	✓	✓	1342		
3142			3214	✓	✓	4231	✓	✓
2314			4213	✓		1432	✓	✓
4312			3412	✓	✓	1243	✓	✓

Table 1: The lists for the sets S_4, B_4 and I_4 (read from top to bottom and from left to right). They are generated by the call `gen_A({1},2)` for $n = 4$ and with $X = T \cup \{j\}$, $X = T$ and $X = T \setminus \{\ell\}$, respectively, and are obtained by covering in pre-order the generating trees in Figures 1, 2 and 3.

(3) Suppose that $k \geq \lceil \frac{n}{2} \rceil$ and that π is an involution at level $d < n - k$. In this case $k - d \geq 1$. Indeed, $d < n - k$ combined with $n \leq 2k$ implies that $d < k$; so $k - d \geq 1$. It is easy to check that $k - d = 1$ occurs only when n is even and $k = d + 1 = \frac{n}{2}$.

On the other hand, π has $n - d$ cycles and thus at least $n - 2d$ fixed points, and $n - k - d$ of them are $k + d + 1, k + d + 2, \ldots, n$. Thus, the number of $\ell \in \text{Fix}(\pi)$, $\ell < j = k + d + 1$, is at least $(n - 2d) - (n - k - d) = k - d \geq 1$, and each call has degree at least two, except for n even, $k = \frac{n}{2}$ and $d = \frac{n}{2} - 1$, which corresponds to the involutions on the last but one level in the generating tree for even n and $k = \frac{n}{2}$. We refer the reader to Figure 3 where the involutions with k cycles are at level $4 - k$.

Neglecting the operations on the set T, in each of the three cases `gen_K` satisfies the second CAT principle. As above, the implementation of T by a linked list yields CAT algorithms.

□

Note that `gen_K` does not generate efficiently the singleton set $B_{n,1} = \{n\,1\,2\ldots(n-1)\}$.

Corollary 2. *If the algorithm* `gen_K` *prints permutations at each level $d \leq n - k$ (not only at level $d = n - k$), then it produces the same three classes of permutations but with* at least *k cycles.*

In a permutation $\pi \in S_n$ a couple (i, j) is an inversion if $i < j$ but $\pi(i) > \pi(j)$. A permutation is called *even* (resp. *odd*) if it has an even (resp. odd) number of inversions. The set of even permutations forms a subgroup of S_n denoted by A_n, called the alternating group, and its cardinality is $\frac{n!}{2}$.

Corollary 3. *If n is even, then the permutations produced by* `gen_A` *at even levels are even and at odd levels are odd. Conversely, if n is odd, then the permutations produced at even level are odd and those produced at odd level are even.*

Proof. The permutation $\pi \in S_n$ is even iff the number of even-length cycles in π is even; see [5, pp. 77].

Permutations with a given number of fixed points

Before the first call to gen_P, the initial permutation is $1\,2\,\ldots\,n$ and it has the maximal number of fixed points, and as mentioned after Lemma 2, the number of fixed points of $\pi \cdot \langle \ell, j \rangle$ in gen_P decreases by two or by one, according to whether ℓ is a fixed point of π or not. If we impose on gen_P the condition that only length-n permutations with f fixed points are printed ($f \in \{0, 1, \ldots, n-2, n\}$), then the resulting algorithm is inefficient: not every permutation with more than f fixed points eventually produces a permutation with f fixed points. Note that permutations with f fixed points, $f < n$, are obtained at each level $d \geq 1$, with $n - 2d \leq f \leq n - d - 1$.

Here we show how to modify the algorithm gen_P in order to obtain a CAT generating algorithm for permutations with exactly f fixed points. The algorithm thus obtained is gen_F in Figure 6 and as in procedure gen_K we introduce a third parameter, e. If π is the permutation corresponding to a given call of gen_F(T,q,e), then q and T have the same meaning as in gen_P and the number of fixed point of π is $f + e$ (e is the fixed points 'excess'); so permutation are printed when $e = 0$ and the main call, corresponding to $1\,2\,\ldots\,n$ is gen_F($\{1\}$,2,$n-f$). In particular, when $f = 0$ the algorithm produces derangements. The upper bound $n - \lceil \frac{e}{2} \rceil + 1$ in the loop on j ensures that the current call eventually makes calls with $e = 0$ and thus produces permutations with f fixed points, and when $j = n - \lfloor \frac{e}{2} \rfloor + 1$, then ℓ must be a fixed point of the current permutation. By simple calculation, it can be shown that each call to gen_F has degree at least two, except possibly at most p of them, with p being the number of generated permutations. By a careful linked list implementation of the involved sets, the algorithm thus obtained runs in constant average time. This is formally stated in Theorem 3; see also Figure 7.

Theorem 3. *The algorithm* gen_F *produces in constant amortized time permutations with a given number of fixed points.*

Combining this result with the previous cases (Bell permutations and/or permutations with a given number of cycles) we obtain:

Corollary 4.

- *If* gen_F *prints permutations at a given level d (*resp. *at each level $\leq d$), with $d \geq 1$ and $n - 2d \leq f \leq n - d - 1$, then it produces permutations with f fixed point and with $n - d$ cycles (*resp. *with at least $n - d$ cycles).*

- *Changing the first parameter of* gen_F *in its inner call from $T \cup \{j\}$ to T the obtained algorithm produces Bell permutations with a given number fixed points. In this case if* gen_F *prints permutations at a given level d (*resp. *at each level $\leq d$), with $d \geq 1$ and $n - 2d \leq f \leq n - d - 1$, then it produces Bell permutations with f fixed point and with $n - d$ cycles (*resp. *with at least $n - d$ cycles).*

4 Final remarks

The generating algorithms presented in this paper can easily be modified in order to produce other classes of permutations: permutations with at least a given number of fixed points or Bell permutations with a bounded cycle lengths. Can the techniques presented here be applied to generate other classes of objects? Also, our algorithms produce permutations by covering

```
procedure gen_F($T$,$q$,$e$)
local $j,\ell$,$a$,$U$;
if $e=0$ then print($\pi$);
else for $j:=q$ to $n-\lceil\frac{e}{2}\rceil+1$ do
          if $j=n-\lfloor\frac{e}{2}\rfloor+1$
          then $U:=T\cap\mathrm{Fix}(\pi)$;
          else if $e=1$
               then $U:=T\setminus\mathrm{Fix}(\pi)$;
               else $U:=T$;
               endif
          endif
          for $\ell\in U$ do
               if $\ell\in\mathrm{Fix}(\pi)$
               then $a:=e-2$;
               else $a:=e-1$;
               endif
               $\pi:=\pi\cdot\langle\ell,j\rangle$;
               gen($T\cup\{j\}$,$j+1$,$a$);
               $\pi:=\pi\cdot\langle\ell,j\rangle$;
          end do
          $T:=T\cup\{j\}$;
     end do
end procedure.
```

Figure 6: Algorithm producing the length-n permutations with f fixed points. The main call is gen_F($\{1\}$,2,$n-f$), n and π are global variables and initially $\pi=1\,2\,\ldots\,n$.

a generating tree and every two consecutive permutations on a branch of this tree differ by a transposition. Does an order exist to cover the whole tree so that two consecutive permutations differ by a transposition, that is, permutations are listed in Gray code?

Acknowledgment. Special thanks are due to an anonymous referee for a very careful reading of a not so careful manuscript.

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

[3] J-L. Baril, V. Vajnovszki, Gray Code for Derangements, *Discrete Applied Mathematics*, 2004, **140**(1-3), 207–221,

[4] A. Bernini, I. Fanti, E. Grazzini, An exhaustive generation algorithm for Catalan objects and others, *PU.M.A.*, 2006, **17**(1-2), 39-53.

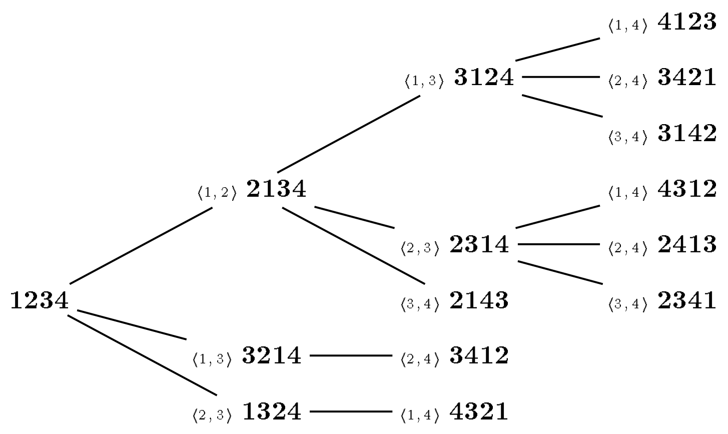

Figure 7: The tree induced by the call of `gen_F({1},2,4)` for $n = 4$. It produces, on its last levels, length-4 permutations with zero fixed points, i.e., derangements.

[5] M. Bóna, *Combinatorics of Permutations*, Chapman and Hall/CRC, Boca Raton, Florida, USA, 2004.

[6] 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.

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

[8] I. Fanti, M. Poneti, V. Vajnovszki, Generating involutions, Bell and Stirling permutations, *CGCS'07*, Luminy, France, May 2007.

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

[10] J. Gallian, D. Witte, A survey: Hamiltonian cycles in Cayley graphs, *Discrete Mathematics*, 1984, **51** 293–304.

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

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

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

[14] D. Knuth, *The Art of Computer Programming, Volume 4, Generating all Tuples and Permutations*, Fascicle 2, Addison-Wesley, 2005.

[15] Frank Ruskey, Combinatorial Generation, book in preparation.

[16] N.J.A. Sloan, *Encyclopedia of Integer Sequences.*

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

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

[19] F. Stedman, *Campanologia*, 1677, reprinted by Christopher Groom, 1900.

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

[21] W.H. Thompson, A note on Grandsire triples, London, 1886. (Reprinted in Grandsire, J.X. Snowdon, London, 1905.)

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

[23] V. Vajnovszki, Generating involutions, derangements, and relatives by ECO, *GASCom'06*, Dijon, France, Sept. 2006.

[24] T. Walsh, Loop-free sequencing of bounded integer compositions, *J. Combin. Math. Combin. Comput.*, 2000, 33, 323–345.

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