FPSAC 2014, Chicago, USA DMTCS proc. **AT**, 2014, 679–690

A new generation tree for permutations, preserving the number of fixed points

Philippe Duchon* and Romaric Duvignau†

Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France
CNRS, LaBRI, UMR 5800, F-33400 Talence, France

Abstract. We describe a new uniform generation tree for permutations with the specific property that, for most permutations, all of their descendants in the generation tree have the same number of fixed points. Our tree is optimal for the number of permutations having this property. We then use this tree to describe a new random generation algorithm for derangements, using an expected $n + O(1)$ calls to a random number generator. Another application is a combinatorial algorithm for exact sampling from the Poisson distribution with parameter 1.

Résumé. Nous décrivons un nouvel arbre de génération uniforme pour les permutations, ayant la propriété que, pour la majorité des permutations, tous leurs descendants ont le même nombre de points fixes; notre arbre est optimal en ce sens. Grâce à cet arbre, nous obtenons un nouvel algorithme de génération aléatoire uniforme de dérangements, algorithme qui nécessite, en moyenne, $n + O(1)$ appels à un générateur de nombres aléatoires; ainsi qu'une méthode combinatoire de simulation exacte de la loi de Poisson de paramètre 1.

Keywords: permutations, derangements, fixed points, generation tree, random generation

1 Introduction

1.1 Fixed points in random permutations

Throughout this paper, the phrase "random permutation" means a uniformly distributed random permutation on the set $[n] = \{1, \ldots, n\}$ of integers less than equal to n; the size n will either be mentioned explicitly, or should be obvious from the context. For any permutation $\sigma \in \mathcal{S}_n$, $\text{fp}(\sigma)$ will denote the number of its fixed points, *i.e.* $\text{fp}(\sigma) = \#\{i \leq n : i = \sigma(i)\}$. The permutations that have no fixed points are named *derangements*. The set of all permutations of size n is denoted by $\mathcal{S}_n$ and that of derangements by $\mathcal{D}_n$.

It is a well known fact that, in a random permutation of size n, each element has probability exactly $1/n$ of being a fixed point. Thus, by linearity of expectation, the expected number of fixed points in a random permutation is exactly 1, no matter the size. It is only slightly less well known that the number of

*Email: `Philippe.Duchon@labri.fr`
†Email: `Romaric.Duvignau@labri.fr`

fixed points in a random permutation of size n follows a distribution that, as n goes to infinity, converges to the Poisson distribution with parameter 1.

A probabilistic consequence is that this convergence in distribution can be realized as almost sure convergence; that is, one can construct, in a single probability space, a sequence $(S_n)_{n\geq 1}$ of permutation-valued random variables, with S_n uniform on the set of permutations of size n, such that, with probability 1, $F = \lim_n \mathrm{fp}(S_n)$ exists and is Poisson distributed. In a sense, the construction of our generation tree is a combinatorial instance of such a realization.

1.2 Uniform generation trees for permutations

A uniform generation tree for permutations can be seen as the description, for each nonnegative integer n, of a bijection ϕ_n between $\mathcal{S}_n \times [n+1]$ and $\mathcal{S}_{n+1}$. The countable set of permutations of all sizes can then be seen as the nodes of an infinite rooted tree, with the unique permutation of size 0 as the root, and where each permutation $\sigma \in \mathcal{S}_n$ has exactly $n+1$ children, obtained by applying ϕ_n to the pairs (σ, i) for $i \in [n+1]$. The $n!$ nodes at distance n from the root are then all permutations of size n. Such a generation tree can also be seen as the description of an algorithm for the random generation of uniform permutations: from a random permutation of size n and an independent uniform integer in the range $[n+1]$, ϕ_n obviously gives us a random permutation of size $n+1$. Thus, starting from the root and repeating n times the simple algorithm of moving to a uniformly chosen child of the current node yields a uniform permutation of size n.

Consistently with the tree terminology, $\phi_n(\sigma, j)$ will be called the *j-th child of* σ, and σ will be called the *parent* of each of its children. All permutations in the subtree rooted at σ are collectively called the *descendants* of σ.

Many simple combinatorial descriptions of such uniform generation trees can be given. We briefly describe two of them.

- Last value insertion: from a permutation $\sigma \in \mathcal{S}_n$ and an integer $j \in [n+1]$, we obtain a new permutation $\sigma' \in \mathcal{S}_{n+1}$ as follows: set $\sigma'(n+1) = j$, and for $1 \leq i \leq n$, $\sigma'(i) = \sigma(i)$ if $\sigma(i) < j$, and $\sigma'(i) = 1 + \sigma(i)$ if $\sigma(i) \geq j$. In other words, j gives the value of $\sigma'(n+1)$, and all previous value at least as large as j are shifted up by 1.

- Cycle insertion: this generation scheme is best described by its action on the cycles of the permutation. From a permutation $\sigma \in \mathcal{S}_n$ and an integer $j \in [n+1]$, we obtain a new permutation $\sigma'' \in \mathcal{S}_{n+1}$ as follows: if $j = n+1$, simply set $\sigma''(n+1) = n+1$ and, for $1 \leq i \leq n$, $\sigma''(i) = \sigma(i)$; otherwise, set $\sigma''(j) = n+1$, $\sigma''(n+1) = \sigma(j)$, and $\sigma''(i) = \sigma(i)$ for all other values of i. In other words, $n+1$ is inserted "right after j" in its pre-existing cycle – or is added as a new fixed point, if $j = n+1$.

Because last value insertion often shifts many values by 1, it can dramatically change the number of fixed points – that is, $\mathrm{fp}(\sigma)$ and $\mathrm{fp}(\phi_n(\sigma, j))$ can be very different. On the other hand, cycle insertion can only change the number of fixed points by ± 1: it increases by 1 if $n+1$ is inserted as a new fixed point, and decreases by 1 if j was previously a fixed point. As one descends in the tree according to the random choices of children, the probability of changing the number of fixed points becomes arbitrarily small (it is exactly $2/n$ on the n-th step). One can easily check, though, that on an infinite random descent into the tree, this change in the number of fixed points will, with probability 1, happen infinitely many times.

The specific generation tree we are interested in this paper has the following properties:

1. For each $n \geq 2$, all but exactly 2^{n-1} permutations in $\mathcal{S}_n$ have the same number of fixed points as their parent.
2. If a permutation has the same number of fixed points as its parent, then all its children (and, by immediate induction, all its descendants) also have this property.
3. If a permutation σ does not have the same number of fixed points as its parent, then the difference is ± 1. Furthermore, exactly one of its children has exactly one more fixed points than σ (and, possibly, one or more children can have one fewer fixed points).

As a consequence, the situation with the evolution of the number of fixed points in an infinite random descent through the tree is quite different from the "cycle insertion" generation tree: with probability 1, the descent will, at some (random) level N in the tree, reach a permutation whose number of fixed points is the same as its parent's; and, once this happens, this number of fixed points will remain identical forever. We even know the probability distribution for N: for any N, the probability that N is larger than n is exactly $2^{n-1}/n!$.

We will call *special* permutations the 2^{n-1} permutations of size n whose parent has a different number of fixed points. Others are simply *non-special*.

1.3 Outline of the paper

The rest of the paper is organized as follows. In Section 2, we describe our generation tree and prove its properties. In Section 3, we use the tree to describe a new random generation algorithm for derangements, which we analyze. A second application of our generation tree is given in Section 4, with a combinatorial algorithm for sampling from the Poisson distribution with unit parameter. We finish the paper with a conclusion outlining further possible applications.

2 The generation tree

In this section, we describe our new generation tree and prove its properties. The construction makes use of a combinatorial bijection on derangements due to Rakotondrajao (2007).

First note that any permutation σ of size n can be seen as a pair (S, δ) where S is its set of k fixed points and δ a derangement of $[n-k]$. Such a derangement is easily constructed by *normalizing* the derangement $\tilde{\delta}$ of the non-fixed points of σ. In the rest of this section, a permutation is always represented by such a pair and a derangement is noted using the usual decomposition into cycles.

Let us first define precisely the *normalization* process: for any integer n, set of integers S of size $k \leq n$, and $i \in [n+1] \backslash S$, we define $\pi_S(i) = \#\{j \leq i : j \notin S\}$. It is straightforward to invert the normalization: $\pi_S^{-1}(i)$ is the i-th element not in S. The normalized version δ of the original derangement $\tilde{\delta}$, is obtained as follows $\delta(i) = \pi_S(\tilde{\delta}(\pi_S^{-1}(i)))$.

For any finite set of positive integers S, we note $\gamma_n(S)$ the largest nonnegative integer less than or equal to n which is not in S, *i.e.*, $\max\limits_{0 \leq i \leq n} i | i \notin S$.

If S is the set of fixed points of a permutation σ of size n, then $\gamma_n(S)$ is the largest non-fixed point of σ, or 0 if σ is the identity.

As an example, for $n = 7$ and $\sigma = 5432167$ (in word form), the set of fixed points is $S = \{3, 6, 7\}$, π_S is defined on $\{1, 2, 4, 5\}$, the unnormalized derangement has cycle decomposition $\tilde{\delta} = (1\ 5)(2\ 4)$, and the normalized derangement is $\delta = (1\ 4)(2\ 3)$. In this case, the largest nonfixed point is $\gamma_7(S) = 5$.

We define the *critical derangement* Δ_n, for even n, to be $(1\ 2)(3\ 4)\dots(n-1\ n)$, and in particular Δ_0 is the empty derangement. A non-critical derangement of size n, is any derangement of size n if n is odd, or any derangement of size n different from Δ_n if n is even.

2.1 A bijection for derangements

Our generation tree will need a bijection τ_n for derangements of size n. For that purpose, we use directly the bijection over derangements defined in Rakotondrajao (2007) that proves combinatorially the well known recurrence $d_n = nd_{n-1} + (-1)^n$, where d_n stands for $|\mathcal{D}_n|$. We recall briefly the bijection and refer the interested reader to the original paper for further results.

A bijective map τ_n is defined from $\mathcal{D}_{n-1} \times [n] \setminus \{(\Delta_{n-1}, n)\}$ to $\mathcal{D}_n$ for odd n, and from $\mathcal{D}_{n-1} \times [n]$ to $\mathcal{D}_n \setminus \{\Delta_n\}$ for even n. For $(\delta, i) \in \mathcal{D}_{n-1} \times [n]$ (with the condition that, if n is odd, $(\delta, i) \neq (\Delta_{n-1}, n)$), $\delta' = \tau_n(\delta, i)$ is constructed as follows:

- If $i < n$, by inserting n into the cycle between i and $\delta(i)$, *i.e.*, $\delta'(i) = n$ and $\delta'(n) = \delta(i)$. This is cycle insertion as described in the introduction.
- If $i = n$, let p be the largest integer such that $\delta|_{[p-1]} = \Delta_{p-1}$ (possibly $p = 1$). Three cases must be distinguished:
  - If $\delta(p) = p + 1$ (implying $\delta(p + 1) \neq p$), then δ' is obtained by removing p from its cycle, and adding the short cycle $(p\ n)$ instead, *i.e.*, $\delta'(\delta^{-1}(p)) = p + 1$, $\delta'(p) = n$ and $\delta'(n) = p$.
  - If $\delta(p) \neq p + 1$ and p is in a cycle of size larger than 2, then[(i)] remove $\delta^{-1}(p)$ from the cycle, and create the short cycle $(\delta^{-1}(p)\ \ n)$, *i.e.*, $\delta'(\delta^{-2}(p)) = p$, $\delta'(\delta^{-1}(p)) = n$ and $\delta'(n) = \delta^{-1}(p)$.
  - If $\delta(p) \neq p+1$ and p is in a cycle of size 2, remove the cycle $(p\ \delta(p))$ from δ, then insert p just before $p + 1$ and finally add the short cycle $(\delta(p)\ n)$, *i.e.*, $\delta'(\delta^{-1}(p + 1)) = p$, $\delta'(p) = p + 1$, $\delta'(\delta(p)) = n$ and $\delta'(n) = \delta(p)$.

It is proved in Rakotondrajao (2007) that τ_n is a bijection. Thus, for a non-critical derangement δ of size n, $\tau_n^{-1}(\delta)$ is well-defined as a pair (δ', i), where δ' is a derangement of size $n - 1$, $i \in [n]$ and the pair (Δ_{n-1}, n) cannot be produced whenever n is odd.

2.2 A new generation tree for permutations

A permutation $\sigma = (S, \delta)$ is said to be *special* if δ is a critical derangement. A (n, k)-permutation is a permutation of size n having k fixed points.

We are now ready to define our generation tree for all permutations (the beginning of the tree is depicted in Figure 1). The root of the tree is the empty permutation $(\emptyset, \Delta_0)$, each node at height n has $n+1$ children, and the i-th child $\sigma' = (S', \delta')$ of a (n, k)-permutation $\sigma = (S, \delta)$ is obtained by the following process:

1. If σ is special and $i = n + 1$:
$$\sigma' = (S \cup \{n + 1\}, \Delta_{n-k}).$$

(i) There appears to be a typo in the original article regarding this case; the proof of the bijection corresponds to what we describe here.

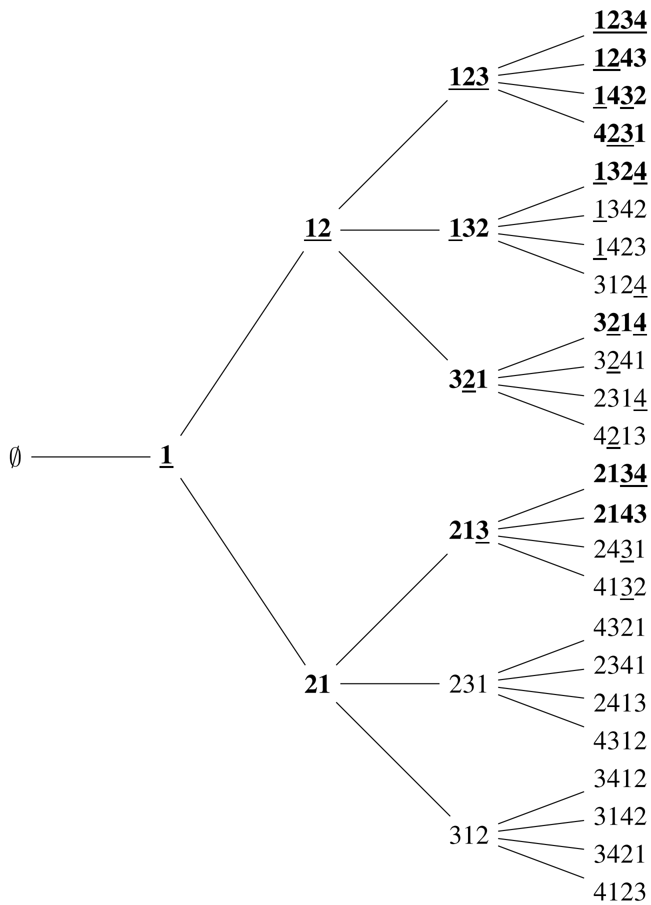

Fig. 1: Our complete generation tree up to level 4. Each node at level n has $n+1$ children, depicted in order from bottom to top. Special permutations are drawn in bold. Fixed points are underlined.

2. If σ is special and $\gamma_n(S) < i \leq n$:

$$\sigma' = (S \setminus \{i\}, \Delta_{n-k+2}).$$

3. If $i \in S$; σ is special and $i \leq \gamma_n(S)$ or σ is not special:

$$\sigma' = (S \setminus \{i\} \cup \{n+1\}, \tau_{n+1-k}(\delta, \pi_{S\setminus\{i\}}(i))).$$

4. If $i \notin S$; σ is special and $i \neq n+1$ or σ is not special:

$$\sigma' = (S, \tau_{n+1-k}(\delta, \pi_S(i))).$$

We first check that σ' is well defined:

- Whenever $\gamma_n(S) < i \leq n$, we have $i \in S$.
- For $i \in S$, $i < n+1$ and $\pi_{S\setminus\{i\}}$ is an integer in the range $[n+1-k]$.
- For $i \notin S$, if σ is not special then i may be $n+1$, $\pi_{S\setminus\{i\}}(i)$ is an integer in the range $[n+1-k]$ and δ is any non-critical derangements.
- When σ is special, or equivalently δ is critical, either rules 3 or 4 cannot be applied with $i = n+1-k$ (the only pair where τ is not defined), as:
  - Rule 3: Since $i \in S$ and $i \leq \gamma_n(S)$ implies $\gamma_n(S) \neq 0$ and $\pi_S(\gamma_n(S)) = n-k$, thus $\pi_{S\setminus\{i\}}(i) \leq n-k$.
  - Rule 4: Since $i \neq n+1$, $\pi_S(i) \in [n-k]$.

Applying rules 1 and 2 is straightforward when the permutation is represented as a set of fixed points and a normalized derangement. Applying rules 3 and 4 in the case where σ is special is also straightforward: the normalized derangement is modified by inserting $n+1-k$ right after the appropriate value (either $\pi_{S\setminus\{i\}}(i)$ or $\pi_S(i)$) in its cycle decomposition. If σ is non-special and $\pi_{S\setminus\{i\}}(i) \neq n+1-k$ (resp. $\pi_S(i)$), the application of rules 3 and 4 yields the same construction.

The only remaining cases, where the construction uses the second case of Rakotondrajao's bijection, are non-special permutations with $i > \gamma_n(S)$, i.e. i is a fixed point greater that the greatest non-fixed point of σ (rule 3) or $i = n+1$ (rule 4). For instance if $\sigma = 5432167 = (\{3,6,7\}, (1\ 4)(2\ 3))$, then its 6-th child is $(\{3,7,8\}, (1\ 2\ 3)(4\ 5))$ via rule 3, and its 8-th child is $(\{3,6,7\}, (1\ 2\ 3)(4\ 5))$ via rule 4.

Notice that the children of a special permutation σ are also special when they are produced by rules 1 and 2, and that they are the last children of σ; they are non-special when produced by rules 3 and 4. The children of a non-special permutation are always non-special since the bijection τ cannot produce critical derangements. The number of fixed points of the children of a permutation σ is $\text{fp}(\sigma)$, except if σ is special and rule 1 (then they have $\text{fp}(\sigma)+1$) or rule 2 ($\text{fp}(\sigma)-1$) has been applied.

Theorem 2.1 *The above description defines a generation tree.*

Proof: We need to prove that for any permutation $\sigma' = (S', \delta')$ of size $n+1$, there exists at least one pair (σ, i) such that σ' is the i-th child of σ in the tree. This is sufficient, since it implies that all $(n+1)!$ permutations of size $n+1$ appear among the $(n+1)!$ nodes of level $n+1$ in the tree – each must appear exactly once.

On the one hand, if σ' is special, there are two cases:

- $n+1 \in S'$: σ' is the $n+1$-th child of $\sigma = (S' \setminus \{n+1\}, \delta')$. σ is indeed special and rules 1 gives directly σ'.
- $n+1 \notin S'$: σ' is the i-th child of $\sigma = (S' \cup \{i\}, \Delta_{n-1-|S'|})$ with $i = \gamma_n(S')$. Note $\gamma_n(S') > \gamma_n(S' \cup \{i\})$, hence $\gamma_n(S' \cup \{i\}) < i \leq n$ and since σ is special rules 2 produces σ'.

On the other hand, if σ' is not special, then δ' is non-critical. Let $(\delta, i') = \tau^{-1}(\delta')$. There are again two cases depending whether $n+1$ is in S' or not:

- $n+1 \in S'$: σ' is the i-th child of $\sigma = (S, \delta)$ with $i = \pi_{S'}^{-1}(i')$ and $S = S' \setminus \{n+1\} \cup \{i\}$. Obviously $\pi_{S'}(i) = \pi_{S\setminus\{i\}}(i)$, thus $i' = \pi_S(i)$ and thus rules 3 produces σ'.

- $n+1 \notin S'$: σ' is the i-th child of (S', δ) with $i = \pi_{S'}^{-1}(i')$. By definition of $\pi_{S'}^{-1}$ we have $i \notin S'$, hence this case corresponds to rule 4.

□

2.3 Optimality

As stated in the introduction, the generation tree described above makes only little changes on the number of fixed points of a permutation: all children of a node have at most one more or one fewer fixed points than their parent. The same property holds for the "cycle insertion" generation tree. However, in such a tree, exactly $2(n-1)!$ of the $n!$ permutations of size n have a different number of fixed points than their parent.

In our tree, only special permutations have a different number of fixed points than their parent. By their description, special permutations of size n are in bijection with subsets of $[n]$ whose size has the same parity as n (otherwise, the corresponding derangement cannot be critical). Thus, there are exactly 2^{n-1} special permutations of size n (there is a simple involution on subsets of $[n]$ that changes the parity of the size: just remove n if present, or add it if absent).

Thus, in our tree, only 2^{n-1} nodes (out of $n!$) at level n have a different number of fixed points than their parent. This number is optimal for every n.

Proposition 2.2 *In any uniform generation tree for permutations, at least 2^{n-1} of the permutations of size n have a different number of fixed points than their parent.*

Proof: Let d_n^k stand for the number of (n,k)-permutations; these are the *rencontre numbers*, which are related to the derangement numbers by $d_n^k = \binom{n}{k} d_{n-k}$. From the recurrence for derangement numbers, we immediately get $d_n^k = n \cdot d_{n-1}^k + \binom{n}{k} \cdot (-1)^{n-k}$.

When $n-k$ is odd, $d_n^k < n \cdot d_{n-1}^k$, so that $(n-1,k)$-permutations collectively have enough children to accommodate all (n,k)-permutations. But when $n-k$ is even, $d_n^k > n d_{n-1}^k$, thus at least $\binom{n}{k}$ (n,k)-permutations must have a parent that is *not* an $(n-1,k)$-permutation.

Summing over all values of k with $n-k$ even, we get the required lower bound of 2^{n-1} special permutations.

□

3 Uniform random generation of derangements

In this section, we show how our generation tree can be used in random generation.

The fact that non-special permutations have the same number of fixed points as each of their children, and by induction as any of their descendants, allows us to design algorithms that make use of this property while sampling a uniform permutation.

An instance of such designed algorithm is the task of sampling uniformly a derangement in an efficient manner, both in time/space complexity and in *randomness* complexity. The randomness cost can be estimated by two different parameters, and we analyze our sampling algorithm for the two measures: the expected number of random bits used by the algorithm, and the expected number of calls to a unit-cost `Random`(k) primitive that returns a uniform number in $[k]$.

A basic rejection algorithm to sample uniformly a derangement of size n, is to generate uniform permutations of size n, until one is obtained that has no fixed points. Since $d_n/n!$ converges (quickly) to $1/e$ in the large n limit, this simple *rejection method* needs to generate, in expectation, $e + O(1/n!)$ permutations of size n. Thus its expected time and random number complexity is $en + O(1/(n-1)!)$ by using any standard method for generating permutations, such as the Fisher-Yates shuffle, also known as Knuth shuffle (see Fisher and Yates (1938); Durstenfeld (1964); Knuth (1997)). This method can be improved by an *anticipated rejection* mechanism, *i.e.* the algorithm restarts as soon as the current generated permutation is certain to contain some fixed points. Approximate calculations indicate that the expected time and random number complexity should be $(e-1)n + o(n)$ in this case.

In the literature, Martínez et al. (2008) propose an equivalent of the Fisher-Yates shuffle for derangements, resulting in a bounded time algorithm with an expected random number complexity of $2n + o(n)$. In Ardnt (2009), experimental studies are performed comparing this method and the anticipated rejection method, resulting in comparable results; an extension to permutations having only cycles of length at least $m \geq 3$ is also given.

In Martínez et al. (2008), the numbers d_n are used by their algorithm in order to maintain the uniform distribution while sampling a derangement. We can reuse this idea with Rakotondrajao's bijection to get another sampling algorithm. From a uniform derangement δ of odd size n, we sample a uniform derangement δ' of size $n+2$ in the following way:

- With probability $(n+1)/d_{n+2}$, we sample a uniform integer i in $[n]$ and output $\delta' = \tau_{n+2}(\Delta_{n+1}, i)$.
- With complementary probability $1-(n+1)/d_{n+2}$, we sample two uniform integers i and j respectively in $[n+1]$ and $[n+2]$ and output $\delta' = \tau_{n+2}(\tau_{n+1}(\delta, i), j)$.

This algorithm (suitably extended to also cover even sizes) has worst time complexity linear in n (with suitable implementation choices) and always uses at most $(3/2)n$ calls to `Random()` (increasing size by 2 requires at most 3 random picks). We describe hereafter a rejection algorithm, based on our generation tree, that needs only $n+O(1)$ calls to `Random()`, in expectation. This is similar to the algorithm sketched at the end of Désarménien (1984), which is based on Lehmer encoding and a specific bijection.

We propose the following natural algorithm UNIFORMDERANGEMENT(n) in order to sample uniformly a derangement of size n:

1. Perform a random descent in the tree until reaching a non-special permutation σ or level n, whichever comes first.
2. If the reached permutation is a derangement, continue the descent until reaching level n and return the derangement; otherwise, repeat Step 1.

Proposition 3.1 UNIFORMDERANGEMENT*(n) returns a uniform derangement of size* n.

Proof: The algorithm works as an anticipated rejection method: it is equivalent to performing a uniform sampling of a permutation of size n, and as soon as the generated permutation is known to have fixed points, the algorithm aborts and retries; otherwise it returns the sampled permutation. Thus the returned permutations are uniform derangements of size n. □

Proposition 3.2 UNIFORMDERANGEMENT*(n) uses in expectation* $n + O(1)$ *calls to* `Random()`.

Proof: Let $A = A_n$ be the random variable describing the number of calls to `Random()` used by the algorithm on input n. We separate the cost into two contributions corresponding to the two phases of the algorithm, such that $A = C_1 + C_2$, where C_1 counts the calls to `Random()` made (including rejections) until a derangement is obtained that either is non-special, or has size n; and C_2 counts only calls made by step 2 of the algorithm.

We have $\mathbb{E}(C_1) = \mathbb{E}(C)/p$, where C is the number of calls during one trial of the first phase, and p the probability of success for each trial. We know $pd_n/n! \geq 1/e - 1/n!$, so that $1/p = e + O(1/n!)$.

We now compute an upper bound on $\mathbb{E}(C)$. For at least m calls (for $m \geq 1$) to be made before rejection in a single trial, the size m permutation reached must be special, which happens with probability $2^{m-1}/m!$, so that we have $\mathbb{P}(C \geq m) = 2^{m-1}/m!$. Taking sums, we get

$$\mathbb{E}(C) = \sum_{m=1}^{n-1} \mathbb{P}(C \geq m) \leq \sum_{m=1}^{\infty} \frac{2^{m-1}}{m!} = -\frac{1}{2} + \frac{1}{2} \sum_{m=0}^{\infty} \frac{2^m}{m!} = \frac{e^2 - 1}{2}.$$

Now C_2 is at most $n - 3$, since the smallest non-special permutations are of size 3. This carries over to $\mathbb{E}(C_2) \leq n - 3$. By linearity of expectation, we get the expected total number of calls $\mathbb{E}(A) = \mathbb{E}(C_1) + \mathbb{E}(C_2) \leq n - 3 + e(e^2 - 1)/2 + O(1/n!)$. □

To estimate the number of random bits used by the same algorithm, we assume that the primitive `Random(k)` is optimal in the sense described in Knuth and Yao (1976). This implies that, for any $k > 2$, the expected random bit cost of the call `Random(k)` is at most $2 + \log_2(k)$. This is bounded above by $2 + \log_2(n)$, since k is never larger than n. Multiplying by the estimation for the expected number of calls, we get

Proposition 3.3 UNIFORMDERANGEMENT*(n) uses in expectation* $n \log_2(n) + o(n \log(n))$ *random bits.*

We make no effort at a tighter upper bound, since the same lower bound is valid: the expected number of random bits cannot be less than $\log_2(d_n)$, which is $n \log_2(n) + O(n)$.

We might improve the sampling algorithm a little bit by rejecting the current permutation if its size plus number of fixed points exceed n (since then it is impossible for the number of fixed points to reach 0 before the permutation size exceeds n). However, the above analysis shows that this is extremely unlikely to happen for values of n larger than a few units, so such an optimization would not significantly change the overall performance of the algorithm.

4 Sampling a Poisson random variate

A second consequence of our generation tree is a combinatorial algorithm to sample from the Poisson distribution with parameter 1, *i.e.* obtaining a random variate that takes each integer value $k \geq 0$ with probability $1/(ek!)$.

Several sampling methods already exist for the Poisson distribution (see Devroye (1986) for most of them). These algorithms are efficient but use rational and irrational numbers. By contrast, our algorithm is combinatorial in nature and only uses small integer numbers.

Our algorithm is pretty simple: perform a random descent in the generation tree (by which we mean, start from the root, and at each level pick a uniform random child of the current node), until a non-special permutation is reached; then output its number of fixed points.

Algorithm POISSON

$n \leftarrow 1$, $g \leftarrow 0$, $k \leftarrow 1$
loop
    $i \leftarrow$ **Random**$(n+1)$
    if $i = n+1$ **then**
        $k \leftarrow k+1$
    else if $i > g$ **then**
        $k \leftarrow k-1$, $g \leftarrow n+1$
    else
        return k
    $n \leftarrow n+1$
end loop

Since, in the description of Subsection 2.2, the permutations only remain special when rules 1 and 2 are used, we only need to keep track of three parameters: the current size n, the number of fixed points k, and $\gamma_n(S)$ (g in the algorithm).

We start by proving that the algorithm is correct: its output is effectively Poisson distributed. Let ν denote the distribution of the output, and μ the Poisson distribution with parameter 1. We will prove that $d_{\mathrm{TV}}(\mu, \nu) = 0$, where d_{TV} is the *total variation distance*

$$d_{\mathrm{TV}}(\mu, \nu) = \sum_{k:\mu(k)>\nu(k)} \mu(k) - \nu(k),$$

for which we only need the property (apart from it effectively being a distance on the set of probability measures on integers) that $d_{\mathrm{TV}}(\mu, \nu)$ is the minimum value (taken over all possible joint distributions) of $\mathbb{P}(X \neq Y)$ where X and Y are random variables of respective distributions μ and ν.

By using our generation tree, we can describe three different random processes. Process 1 is our Poisson sampling algorithm. Process 2, parametrized by some integer n, performs a random descent in the tree until level n, and outputs the number of fixed points in the final permutation. Process 3 works similarly, but instead of always outputting the number of fixed points, it outputs the special value `Failure` if the final permutation is special. We can assume that the three processes use the same random choices for their descents.

The probability that processes 2 and 3 output different results is exactly the probability of reaching a special permutation, that is, $2^{n-1}/n!$. The same is true of the probability that processes 1 and 3 output different results. Thus, the probability that processes 1 and 2 output different results is at most $2^n/n!$. Thus, $2^n/n!$ is an upper bound for the total variation distance between the distribution of the results of our algorithm and that of process 2. Let μ_n denote the distribution of the output of process 2, that is, the number of fixed points in a random permutation of size n: we thus have

$$d_{\mathrm{TV}}(\nu, \mu_n) \leq \frac{2^n}{n!}.$$

Now, the total variation distance between the number of fixed points in a random permutation of size n, and the Poisson distribution with parameter 1, is less than the total variation distance between the numbers of fixed points in random permutations of sizes n and $n+1$; this can be seen easily by noticing that $d_n^k/n! < 1/(k!e) < d_{n+1}^k/(n+1)!$ for odd $n-k$, and $d_n^k/n! > 1/(k!e) > d_{n+1}^k/(n+1)!$ for even $n-k$. Thus, if μ_n denotes the distribution of the number of fixed points of a random permutation of size n, we have

$$d_{\mathrm{TV}}(\mu, \mu_n) \leq d_{\mathrm{TV}}(\mu_n, \mu_{n+1}) = \frac{2^n}{(n+1)!}.$$

By combining the two previous inequalities, we have, for any n, $d_{\mathrm{TV}}(\mu, \nu) \leq (n+2)2^n/(n+1)!$, thus $d_{\mathrm{TV}}(\mu, \nu) = 0$, thus $\mu = \nu$.

We now turn to a short analysis of the random bit complexity of our algorithm.

Proposition 4.1 *Our* POISSON *algorithm uses, in expectation, between* 6.89 *and* 6.9 *random bits, assuming the* `Random` *primitive is optimal in terms of random bits used.*

Proof: Let C be the number of random bits used by the algorithm and C_n the number of random bits (if any) consumed at level n, so that $C = \sum_{n=2}^{\infty} C_n$.

We have $\mathbb{E}(C_n) = \frac{2^{n-2}}{(n-1)!}\mathbb{E}(U_n)$, where U_n is the number of bits used by an optimal uniform sampler `Random`(n) in the sense of Knuth and Yao (1976); the $2^{n-2}/(n-1)!$ factor is simply the probability of reaching level n of the tree. The inequality $\log_2(n) \leq \mathbb{E}(U_n) \leq \log_2(n) + 2$ is valid for all n, but using it blindly to bound the complexity of our algorithm gives an upper bound of 9.59, significantly more bits than we want.

Instead, we compute exactly $D_k = \sum_{n=2}^{k} \frac{2^{n-2}}{(n-1)!}\mathbb{E}(U_n)$ for some constant k. The lower and upper bounds for $\mathbb{E}(C)$ then become $D_k + \sum_{n \geq k} \frac{2^{n-1}\log_2(n+1)}{n!}$ and $D_k + \sum_{n \geq k} \frac{2^{n-1}(2+\log_2(n+1))}{n!}$, respectively. For fixed n, $\mathbb{E}(U_n)$ can be simply obtained from the binary expansion of $1/n$, as described in Knuth and Yao (1976).

To obtain the estimates given in the proposition, we computed the above bounds for $k = 10$. Values of $\mathbb{E}(U_n)$ for n from 2 to 10 are $1, 8/3, 2, 18/5, 11/3, 24/7, 3, 14/3, 23/5$, yielding $D_{10} = \frac{97771}{14175} \simeq 6.897$. This is a lower bound for the expected random bit complexity.

For the upper bound, we need an upper bound for $M = \sum_{n \geq 10} \frac{2^{n-1}(2+\log_2(n+1))}{n!}$. We only use the inequality $2 + \log_2(n+1) \leq n$ (which is certainly valid for $n \geq 10$); we get

$$M \leq \sum_{n \geq 9} \frac{2^n}{n!} = e^2 - \sum_{n=0}^{8} \frac{2^n}{n!} = e^2 - \frac{2327}{315} \simeq 0.00175.$$

Indeed, both the lower and upper bound are between 6.89 and 6.9. □

Our Poisson algorithm, efficient though it may be, is not optimal; again applying the results in Knuth and Yao (1976), an optimal algorithm would use, in expectation, fewer than $2 + E$ bits, where $E = 1 + e^{-1}\sum_{k \geq 0} \log_2(k!)/k! \simeq 1.89$ is the binary entropy of the Poisson distribution; note, however, that such an algorithm would need the binary expansions of all individual probabilities of the Poisson distribution, $p_k = 1/(k!e)$.

5 Conclusion

We presented a new uniform generation tree for permutations, with the property of preserving as much as possible the number of fixed points between a permutation and its parent. This generation tree gives us a new and efficient algorithm for the uniform random generation of derangements, as well as a new method for sampling from the Poisson distribution with parameter 1. Both algorithms use only small integer numbers.

We believe other applications to random generation can be devised for the generation tree, such as sampling permutations having some predefined condition on their number of fixed points: *e.g.* an even number, a fixed number. We expect our tree to make many rejection-based algorithms for this kind of task particularly efficient.

Acknowledgements

The OEIS Foundation Inc (2013) was very helpful in identifying the *rencontre numbers* in some apparently unrelated work, and thus lead to the tree construction that is the focus of the present paper.

References

J. Ardnt. *Generating Random Permutations*. PhD thesis, Australian National University, 2009.

L. Devroye. *Non-Uniform Random Variate Generation*. Springer Verlag, 1986.

R. Durstenfeld. Algorithm 235: Random permutation. *Commun. ACM*, 7(7):420–, July 1964. ISSN 0001-0782.

J. Désarménien. Une autre interprétation du nombre de dérangements. In *Actes 8e Sém. Lothar. Combin.*, pages 11–16. IRMA, Strasbourg, 1984.

R. A. Fisher and F. Yates. *Statistical Tables for Biological, Agricultural and Medical Research*. Edinburgh: Oliver and Boyd, 1938.

D. E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. ISBN 0-201-89684-2.

D. E. Knuth and A. C. Yao. The complexity of nonuniform random number generation. In *Proceedings of Symposium on Algorithms and Complexity*, pages 357–428. Academic Press, New York, 1976.

C. Martínez, A. Panholzer, and H. Prodinger. Generating random derangements. In *I. Munro, R. Sedgewick, W. Szpankowski, and D. Wagner, editors, Proc. of the 10th ACM-SIAM Workshop on Algorithm Engineering and Experiments (ALENEX) and the 5th ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 234–240. SIAM, 2008.

F. Rakotondrajao. k-fixed-points-permutations. In *INTEGERS: Electronic Journal of Combinatorial Number Theory*, volume 7, 2007. A36.

The OEIS Foundation Inc. The on-line encyclopedia of integer sequences. http://oeis.org, 2013.