Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA

CURRICULUM IN MATEMATICA
CICLO XXXV

Sede amministrativa Università degli Studi di Firenze
Coordinatore Prof. Matteo Focardi

Preimages of sorting algorithms

Settore Scientifico Disciplinare INF/01

Dottorando:
Lapo Cioni

Tutore
Prof. Luca Ferrari

Coordinatore
Prof. Matteo Focardi

Anni 2019/2023

Contents

Chapter 1

Introduction

A *pattern* of a permutation is a subsequence of the permutation, up to a rescaling of its values. We say that a permutation *contains* a pattern if it contains a subsequence which is order-isomorphic to it. If a permutation does not contain a pattern, it *avoids* that pattern. The study of sorting algorithms by means of patterns (and the idea of pattern itself) comes from Knuth [20], who introduced the algorithm `Stacksort`. For a detailed survey and bibliographic pointers on the topic of patterns, sorting algorithms and their derivations we refer the reader to [5, 13] and to the introduction of [28].

Essentially, a sorting algorithm is a procedure that takes an input sequence and, by suitably applying certain allowed operations, outputs a weakly increasing sequence whenever possible. `Stacksort` uses an auxiliary container called *stack*, in which the elements can be stacked on top of each other, and retrieved in order from top to bottom. Knuth found that a permutation π can be sorted using `Stacksort` if and only if it avoids the pattern 231.

Knuth's work inspired some further investigations using different containers, for example queues, deques [23] and networks of queues and stacks [24]. Especially remarkable is the work of West [29], which concerns the iteration of `Stacksort`. To frame the importance of his work, we need to note that we are considering *deterministic* algorithms, that is, algorithms that perform operations determined only by their input, so that the same input will always produce the same output. Therefore, an algorithm can be seen as a map from the set of possible inputs to the set of possible outputs. This map can be composed with itself any number of times, and the resulting process is equivalent to the iteration of the corresponding algorithm. From a computational standpoint, this process is interesting because, for many algorithms, it sorts any permutation. Therefore, it can be used as a "proper" sorting algorithm, provided that the average number of iterations needed is small enough. From a combinatorial point of view, composing the map with itself gives useful information on the starting algorithm, and also helps to develop tools for its study. For example, West considered the machine obtained by applying `Stacksort` twice, and found that the sortable permutations are those avoiding the pattern 2341 and the barred pattern $3\bar{5}241$, the latter being a generalization of the classical concept of patterns. Clearly, applying `Stacksort` twice is just the first step in understanding how its iteration works. In fact, West defined the *sorting trees* of `Stacksort`, a family of trees T_n where every node is labeled with a permutation. Each tree is rooted at the identity permutation $1 \cdots n$, and two permutations have a parent-child relation if and only if the parent is the image of the child under `Stacksort`.

From the preimage tree we can retrieve several informations about the algorithm, such as the average height (which is the average number of iteration needed to sort a permutation of a given length), the number (and hopefully the characterization) of permutations sorted with a given number of iterations, and the number of preimages of any permutation. Since every

sorting algorithm has an associated map, the concept of "preimage of a permutation under the map" is applicable. Therefore, the preimages of a permutation π are the permutations that, after one sorting, give π as the output. In the preimage tree, they are the children of π. Clearly, computing and characterizing the preimages of a permutation is essential in order to obtain the sorting trees. For `Stacksort`, various results have been achieved, for example in [7, 10].

Note that the preimages under a sorting algorithm can be studied not just for single permutations, but for entire permutation classes. A permutation class is the set of all permutations avoiding a prescribed set of patterns. Thus, the preimage of a class is the set of permutations whose output avoids certain patterns. For example, Av(21) denotes the set of permutations avoiding the pattern 21, therefore the preimages of Av(21) under `Stacksort` are those belonging to Av(231), since they are the sortable ones. Besides `Stacksort` [11, 21, 27], preimages of permutation classes have been studied for many sorting algorithms, such as `Bubblesort` [11, 21, 27], a sorting algorithm widely used in computer science, and `Queuesort` [3], a sorting algorithm that uses a queue instead of a stack. On the other hand, there is not much on the preimages of single permutations. Filling this gap is the main focus of this thesis. Specifically, we describe the preimages of any permutation under `Queuesort`, `Bubblesort` and a new algorithm that uses a popqueue (a modified queue), which we called `Cons`.

The thesis is organized as follows. Chapter 2 introduces the notations and the preliminary notions. Chapter 3 focuses on the preimages of `Queuesort`, and is joint work with Luca Ferrari. We describe a procedure that, for any permutation π, computes the preimages of π under `Queuesort`. Additionally, we prove some enumerative results concerning the number of preimages of any permutation of a specific form. Finally, we compute the number of permutations with a fixed number k of preimages, for small values of k. Chapter 4 is joint work with Luca Ferrari and Mathilde Bouvel, and focuses on `Bubblesort`, answering the same questions of the previous chapter, and more. We consider the *preimage tree* of `Bubblesort`, and we give an exhaustive description of its shape and the heights of its nodes. In Chapter 5, which is joint work with Luca Ferrari, we define two new sorting algorithms, `Cons` and `Min`, both of which use a popqueue to sort permutations. The two algorithms look similar, but their differencies arise when they are applied twice. Finally, we find the preimages of any permutation under `Cons`. In Chapter 6 we consider a container formed by two queues in series, with several possibilities for the allowed operations. For each case, we describe the permutations that can be sorted. Finally, in Chapter 7, which is joint work with Mathilde Bouvel and Benjamin Izart, we study the interval poset of permutations, a partially ordered set studied in [26]. After giving the basic definitions, we describe a correspondence between interval posets and decomposition trees, which is then used to answer all the open questions of the original paper. Chapter 8 contains some hints for further work.

Chapter 2

Preliminary notions

2.1 Permutations

A permutation π of length (or size) n is a bijection from $\{1, \ldots, n\}$ to $\{1, \ldots, n\}$, and its length is denoted by $|\pi|$. A common way to represent a permutation is the linear notation, in which π is written as the sequence $\pi_1 \cdots \pi_n$ of its images, where $\pi_i = \pi(i)$. This we will mostly use throughout all the thesis.

Another representation that will be useful in some situation is the plot, where a permutation π is drawn as a graph with dots at coordinates (i, π_i), for every $i = 1, \ldots, n$. We denote with S_n the set of permutations of size n, and with S the set of permutations of any size. We say that a permutation $\pi = \pi_1 \cdots \pi_n \in S_n$ *contains the pattern* $\rho = \rho_1 \cdots \rho_k \in S_k$ (and write $\rho \leq \pi$) if and only if π contains a subsequence which is order isomorphic to ρ, that is, the elements of ρ are in the same relative order of some subsequence of π. If π does not contain ρ we say that π *avoids* ρ, and write $\rho \not\leq \pi$. Given a set B of patterns, we denote with $\mathrm{Av}(B)$ the set of permutations avoiding every element of B, and with $\mathrm{Av}_n(B)$ the permutations of size n in $\mathrm{Av}_n(B)$.

An element π_i of π is called a left-to-right maximum if $\pi_i > \pi_j$ for every $j < i$, that is, if π_i is greater than every element on its left when π is written in linear notation. We decompose π in two different ways to highlight its left-to-right maxima:

- LTR-*max decomposition*: let $\mu_1, \ldots, \mu_k$ be the left-to-right maxima of π, then we can write $\pi = \mu_1 A_1 \cdots \mu_k A_k$, where the A_i's are (possibly empty) blocks of elements which are not left-to-right maxima. If A_k is empty, we may omit it;
- LTR-*max block decomposition*: by merging the adjacent left-to-right maxima of π into blocks, we obtain $\pi = M_1 P_1 \cdots M_{k-1} P_{k-1} M_k$, where the M_j's are nonempty (except maybe for M_k) blocks containing the adjacent left-to-right maxima of π and the P_i's are nonempty blocks containing all the remaining elements. Notice that, in this notation, the number of left-to-right maxima of π is $|M_1| + \cdots + |M_k|$, and not k.

Finally, for a permutation π, we denote with $\mathrm{LTR}_p(\pi)$ (resp. $\mathrm{LTR}_v(\pi)$) the set of positions (resp. values) of the left-to-right maxima of π.

2.2 Sorting procedures

The process of sorting a permutation consists of following a sequence of instruction whose final goal is to obtain an increasing permutation. The increasing permutation (also called *identity permutation*) of size n is denoted id_n. Whenever the size of the increasing permutation is not known, but clear from the context, we just write id.

The sorting process may not always end with an increasing permutation, since there may be restrictions in the allowed operations that make it impossible to happen. We are interested specifically in sorting procedures which do not sort every permutation and, in what follows, we will detail some of those that we will study.

2.3 Queuesort

A queue is a container in which elements are arranged in the same order in which they entered. We allow for a bypass of the queue, i.e. an element can either enter the queue or be sent directly into the output. Notice that this is equivalent to considering two queues in parallel, which is the classical way in which queue-sorting was studied.

Given a permutation $\pi = \pi_1\pi_2\cdots\pi_n$, the algorithm `Queuesort` attempts to sort π by using a queue in the following way: scan π from left to right and, letting π_i denote the current element,

- if the queue is empty or π_i is larger than the last element of the queue, then π_i is inserted to the back of the queue;
- otherwise, compare π_i with the first element of the queue, then output the smaller one.

When all the elements of π have been processed, pour the content of the queue into the output.

We will give an alternate description of the algorithm in Chapter 3.

There is a natural function $\mathbf{Q}$ associated with `Queuesort`, which maps an input permutation π into the permutation $\mathbf{Q}(\pi)$ that is obtained by performing `Queuesort` on π.

Tarjan [24] proved that a permutation is sortable if and only if it avoids the pattern 321. In terms of its geometric structure, a 321-avoiding permutation is a permutation that can be expressed as the shuffle of two increasing subpermutations. In other words $\mathbf{Q}^{-1}(id_n) = \mathrm{Av}_n(321)$. Moreover, as it is well-known, $|\mathbf{Q}^{-1}(id_n)| = |\mathrm{Av}_n(321)| = C_n$, where $C_n = \frac{1}{n+1}\binom{2n}{n}$ is the n-th Catalan number.

2.4 Bubblesort

The `Bubblesort` operator, denoted $\mathbf{B}$, corresponds to applying one pass of the classical bubble sorting procedure to a permutation. Specifically, $\mathbf{B}(\pi)$ is obtained from π scanning its elements from left to right, each time exchanging an element with the one sitting to its right whenever the latter is smaller. Thereby (see also Lemma 2.4.1), the left-to-right maxima of the permutation "bubble up" to the right, until they are blocked by the next left-to-right maximum.

For example, for $\pi = \mathbf{4}21\mathbf{63}\mathbf{7}\mathbf{8}5$, the left-to-right maxima are $4, 6, 7$ and 8 (shown in bold) and $\mathbf{B}(\pi) = 21436758$.

Notice that we focus on the `Bubblesort` operator $\mathbf{B}$ instead than on the entire bubble sorting procedure. This is because the classical procedure applies $\mathbf{B}$ repeatedly until the input is sorted, so every permutation would be a preimage of the identity permutation. On the other hand, $\mathbf{B}$ does not sort every permutation, and shares some features with other "imperfect" sorting algorithms such as `Queuesort` and `Stacksort`, as demonstrated in [3]. For example, the set of sortable permutation for $\mathbf{B}$ is a permutation class, namely $Av(231, 321)$.

Actually, `Queuesort` and `Bubblesort` have more in common than it may appear, since the latter can be seen as an instance of the former, with some restrictions. Consider a queue of length one. That is, a queue who can hold only one element at a time. If we modify the `Queuesort` algorithm so that it can only add an element into the queue whenever it is empty, then we have a sorting algorithm that is substantially the same as the `Bubblesort` operator, in which the element inside the queue corresponds to the one that is being swapped by `Bubblesort`.

Therefore $\mathbf{B}$ is the function associated to a device consisting of a queue of length one with a bypass.

The algorithm `Bubblesort` may be described in several other ways, and we give two below. The reader needing an explanation of the equivalence with the above definition can find it in [3, Lemma 1 and the equation displayed just above it]. The first parallels the recursive definition of the stacksorting operator $\mathbf{S}$: decomposing π into $\pi = \pi_L n \pi_R$ with n the maximal value occurring in π, we have $\mathbf{B}(\pi) = \mathbf{B}(\pi_L)\pi_R n$. The second focuses on the left-to-right maxima, and we record it in a lemma for future reference.

Lemma 2.4.1. *Let π be a permutation, and decompose π using the* LTR*-max decomposition. That is, $\pi = \mu_1 A_1 \mu_2 A_2 \dots \mu_k A_k$, where the μ_i's are all the left-to-right maxima of π, and the A_i are possibly empty sequences of integers). Then $\mathbf{B}(\pi) = A_1 \mu_1 A_2 \mu_2 \dots A_k \mu_k$.*

From this description, we have the following corollary.

Corollary 2.4.2. *For any π, the set of left-to-right maxima of π is included in the set of left-to-right maxima of $\mathbf{B}(\pi)$.*

Chapter 3

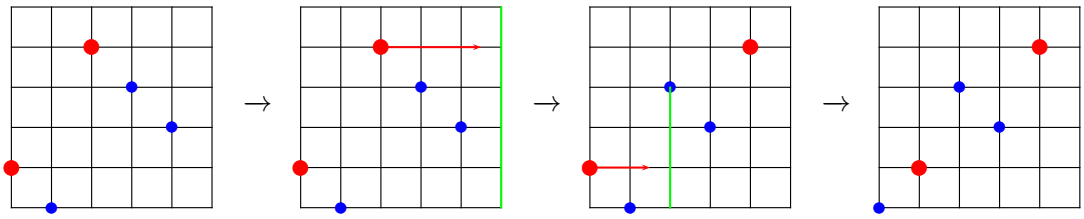
Queuesort preimages

In order to study preimages of the map $\mathbf{Q}$, the first tool we develop is an effective description of the behavior of `Queuesort` directly on the input permutation.

Let π be a permutation of length n, and denote with $\mu_1, \mu_2, \ldots, \mu_k$ its left-to-right maxima, listed from left to right. Thus, in particular, $\mu_k = n$. Then $\mathbf{Q}(\pi)$ is obtained from π by moving its left-to-right maxima to the right according to the following instruction:

- for i running from k down to 1, repeatedly swap μ_i with the element on its right, until such an element is larger than μ_i.

For instance, if $\pi = 21543$, then there are two left-to-right maxima, namely 2 and 5; according to the above instructions, π is thus modified along the following steps: $\mathbf{2}1\mathbf{5}43 \rightsquigarrow \mathbf{2}1435 \rightsquigarrow 12435$, and so $\mathbf{Q}(21543) = 12435$.



The above alternative description of `Queuesort` is, as a matter of fact, a different algorithm which is however equivalent to `Queuesort` (meaning that, starting from a given input, it returns exactly the same output). The proof of this fact is easy, and it relies on the fact that the elements of the input permutations that enter the queue are precisely its left-to-right maxima. In the rest of the chapter, with some abuse of terminology, whenever we will declare that we perform the algorithm `Queuesort`, we will in fact perform our equivalent algorithm.

An immediate consequence of this alternative description of `Queuesort` is our first result, characterizing permutations having nonempty preimage.

Proposition 3.0.1. *Given $\pi = \pi_1\pi_2\cdots\pi_n$, we have that $\mathbf{Q}^{-1}(\pi) \neq \emptyset$ if and only if $\pi_n = n$.*

Proof. Suppose first that $\mathbf{Q}^{-1}(\pi) \neq \emptyset$, and let $\sigma \in \mathbf{Q}^{-1}(\pi)$. From the above description of `Queuesort`, it follows immediately that the last element of $\mathbf{Q}(\sigma) = \pi$ is n. Conversely, suppose that $\pi_n = n$, and define $\sigma = n\pi_1\cdots\pi_{n-1}$. It is easy to check that $\mathbf{Q}(\sigma) = \pi$, and so $\sigma \in \mathbf{Q}^{-1}(\pi) \neq \emptyset$. □

3.1 A recursive characterization of preimages

The key ingredient to state our recursive characterization of preimages is the LTR-max block decomposition of a permutation, defined in Section 2.1. Since, by Proposition 3.0.1, we know that all the permutations that have preimages under $\mathbf{Q}$ ends with their maximum, we write them as $M_1P_1M_2P_2\cdots M_{k-1}P_{k-1}M_k$, with nonempty M_k.

Moreover, for the rest of this chapter $m_i = |M_i|$ denotes the length of M_i, and analogously $p_i = |P_i|$ denotes the length of P_i, for all i. Sometimes we will also need to refer to the last element of M_i, which will be denoted μ_i. Also, in some cases we will use N and R in place of M and P, respectively.

In order to determine the preimages of a given permutation π, we can exploit the description of `Queuesort` illustrated in the previous section, observing that every preimage of π can be obtained by moving suitable elements of π to the left, until they reach a suitable position. The next proposition is recorded for further reference, and says that the only elements of π that can be moved are left-to-right maxima of π.

Proposition 3.1.1. *Suppose that $\pi = \mathbf{Q}(\sigma)$. Then each left-to-right maximum of σ is a left-to-right maximum of π as well.*

Proof. Let μ be a left-to-right maximum of σ. When performing `Queuesort` on σ, the left-to-right maxima of π which are greater than μ are on its right, and will be moved to the right, thus not interfering with μ. Then μ is moved to the right, and it stop before the first greater element, preserving its status of left-to-right maximum. Finally, since the elements on its left are smaller, the operations that `Queuestort` performs on them still mantain μ as a left-to-right maximum. □

Conversely, the next result gives a necessary condition for a left-to-right maximum of π to be movable in order to get preimages.

Proposition 3.1.2. *Suppose that $\pi = \mathbf{Q}(\sigma)$, with $\pi, \sigma \in S_n$. Let π_i be a left-to-right maximum of π, for some $i < n$. If π_{i+1} is not a left-to-right maximum of π, then π_i is not a left-to-right maximum of σ.*

Proof. Suppose, by contradiction, that π_i is a left-to-right maximum of σ. According to our description of `Queuesort`, consider the instant when π_i is moved to the right. When π_i reaches its final position, it is clear that all the elements $\pi_{i+1}, \pi_{i+2}, \ldots, \pi_n$ are already in their final positions in π (since, when a left-to-right maximum is moved to its final position, no further elements can overtake it). This means that, when π_i reaches its final position, it is immediately followed by π_{i+1}. However, by hypothesis, π_{i+1} is not a left-to-right maximum of π, so necessarily $\pi_i > \pi_{i+1}$, hence π_i should overtake π_{i+1}, which gives a contradiction. □

As a consequence of the above proposition, whenever π_i is not followed by a left-to-right maximum in π, we cannot move it to the left in order to get a preimage of π.

We are now ready to start our description of all the preimages of a given permutation π. Our approach consists of splitting the set of preimages into two disjoint subsets, each of which can be described in a recursive fashion. For the rest of the section, we will assume that $\pi = M_1P_1M_2P_2\cdots M_{k-1}P_{k-1}M_k \in S_n$ is not the identity permutation (whose preimages are well understood) and has at least one preimage (therefore M_k is nonempty).

Proposition 3.1.3. *Suppose that $|M_k| \geq 2$. Denote with π' the permutation obtained from π by removing n, and let σ' be any preimage of π'. Let $\sigma' = N_1R_1\cdots N_{s-1}R_{s-1}N_s$ be the* LTR-*max block decomposition of σ'. Then every permutation obtained from σ' by inserting n in any position to the right of N_{s-1} is a preimage of π.*

Proof. Let σ be obtained from σ' by inserting n somewhere to the right of N_{s-1}. Our goal is to show that $\mathbf{Q}(\sigma) = \pi$. Observe that, when we perform `Queuesort` on σ', the elements of N_s are not moved to the right. Therefore, when we perform `Queuesort` on σ, first of all n is moved to the rightmost position. After that, since n was to the right of N_{s-1}, the algorithm performs the same operations it would perform on σ'. As a consequence, $\mathbf{Q}(\sigma) = \mathbf{Q}(\sigma')n = \pi' n = \pi$, as desired. □

Proposition 3.1.4. *Let M'_{k-1} (respectively, M'_k) be the (possibly empty) sequence obtained by removing the last element μ_{k-1} (respectively, n) from M_{k-1} (respectively M_k). Denote with σ any preimage of the permutation $\rho = M_1 P_1 \cdots M_{k-2} P_{k-2} M'_{k-1} n$. Then the permutation τ obtained by concatenating σ with $\mu_{k-1} P_{k-1} M'_k$ is a preimage of π.*

Proof. When performing `Queuesort` on τ, first of all n is moved to the rightmost position. After that, observe that the element μ_{k-1} is larger than all the elements on its left. Hence, all the elements of σ will not overtake it during the execution of the algorithm. Therefore, at the end, the output permutation is obtained by concatenating $\mathbf{Q}(\sigma)$, $\mu_{k-1} P_{k-1} M'_k$ and n, i.e. $\mathbf{Q}(\tau) = \pi$, as desired. □

The previous two propositions told us how to find some preimages of π. The next two propositions aim at showing that these are the only preimages of π.

Proposition 3.1.5. *Suppose that $|M_k| \geq 2$. Denote with π' the permutation obtained from π by removing n. Let $\sigma = N_1 R_1 \cdots N_{s-1} R_{s-1} N_s$ be any preimage of π such that there exists an element of M_k different from n which belongs neither to R_{s-1} nor to N_s. Denote with σ' the permutation obtained from σ by removing n. Then σ' is a preimage of π'.*

Proof. Looking at the position of n in σ, we can distinguish two cases.

If $n \in N_s$, then $\sigma = \sigma' n$, and clearly $\mathbf{Q}(\sigma' n) = \mathbf{Q}(\sigma) = \pi = \pi' n$, hence $\mathbf{Q}(\sigma') = \pi'$, as desired.

On the other hand, suppose that $n \notin N_s$. This means of course that $N_s = \emptyset$, and n is the rightmost element of N_{s-1}. Let $D = \{\gamma \in M_k \,|\, \gamma \neq n, \gamma \notin N_s, \gamma \notin R_{s-1}\}$, and set $\alpha = \max D$. Notice that α indeed exists, since our hypothesis implies that $D \neq \emptyset$.

When we perform `Queuesort` on σ, first of all the element n is moved to the rightmost position. After that, we now want to show that the next element to be moved is α.

First, we notice that α is in fact a left-to-right maximum of σ (hence it is moved by `Queuesort`). Indeed, if this were not the case, α would be to the left of N_{s-1} in σ, and so it could never overtake the elements of N_{s-1} and R_s, contrary to the fact that $\alpha \in M_k$. Our next aim is to prove that there cannot exist elements between α and n in σ that can be moved by `Queuesort`. By contradiction, suppose that β is such an element. Then necessarily β is a left-to-right maximum of σ, and it belongs neither to R_{s-1} (since R_{s-1} does not contain left-to-right maxima) nor to N_s (because elements in N_s are already at the end of the permutation and so they are not moved by `Queuesort`). Moreover, $\beta \neq n$ (by our assumption) and $\beta > \alpha$ (since β is a left-to-right maximum to the right of α in σ). Finally, $\beta \in M_k$, since $\alpha \in M_k$ is a left-to-right maximum of π, hence β ($> \alpha$) cannot be in a position to the left of M_k in π. We can thus conclude that $\beta \in D$, but this gives a contradiction, since $\max D = \alpha < \beta \in D$.

To conclude the proof, we now want to show that $\mathbf{Q}(\sigma')n = \mathbf{Q}(\sigma)$ (since obviously $\mathbf{Q}(\sigma) = \pi = \pi' n$, hence $\mathbf{Q}(\sigma') = \pi'$). A moment's thought should convince one that, to this aim, it is enough to prove that there exists no element after n in σ which is larger than α and is the first element of a descent. For a contradiction, suppose that ω is such an element. Then α cannot overtake ω (when `Queuesort` is performed on σ), and this implies that $\alpha \notin M_k$ (since there is at least one element following α in π which is smaller than α), which is false. □

Proposition 3.1.6. *Let M'_{k-1} (respectively, M'_k) be the (possibly empty) sequence obtained by removing the last element μ_{k-1} (respectively, n) from M_{k-1} (respectively M_k). Denote with $\sigma = N_1R_1 \cdots N_{s-1}R_{s-1}N_s$ a preimage of π such that all the elements of M_k other than n belong to $R_{s-1}N_s$. Then σ is the concatenation of a preimage of $M_1P_1 \cdots M_{k-2}P_{k-2}M'_{k-1}n$ and $\mu_{k-1}P_{k-1}M'_k$.*

Proof. We start by showing that n is the last element of N_{s-1} (and so $N_s = \emptyset$). Since $P_{k-1} \neq \emptyset$, the element immediately after μ_{k-1} in π is not a left-to-right maximum of π. Thus, by Proposition 3.1.2, μ_{k-1} is not a left-to-right maximum of σ. Denote with ν any left-to-right maximum of σ preceding μ_{k-1} such that $\nu > \mu_{k-1}$. As a consequence of Proposition 3.1.1, all the left-to-right maxima of σ larger than μ_{k-1} must belong to M_k. However, since ν precedes μ_{k-1} (and μ_{k-1} is not a left-to-right maximum of σ), ν cannot belong to $R_{s-1}N_s$. Therefore we can conclude that $\nu = n$, hence n is the last element of N_{s-1} and $\sigma = N_1R_1 \cdots N_{s-1}R_{s-1}$.

Now decompose σ as $\sigma = \rho\tau$, where the first element of τ is μ_{k-1}. Performing `Queuesort` on σ, we have first that n is moved to the last position. After that, we observe that all the left-to-right maxima of σ that are moved are smaller than μ_{k-1}. In fact, all left-to-right maxima of σ larger than μ_{k-1} are also left-to-right maxima of π (by Proposition 3.1.1), therefore they are to the right of μ_{k-1} in σ, i.e. they belong to τ. Thus we have that $M_1P_1M_2P_2 \cdots M_{k-1}P_{k-1}M_k = \pi = \mathbf{Q}(\sigma) = \theta\tau n$, for a suitable θ, hence $\tau = \mu_{k-1}P_{k-1}M'_k$.

Finally, since all the left-to-right maxima of σ (other than n) which are moved by `Queuesort` are smaller than μ_{k-1}, they cannot overtake μ_{k-1}, and so $\pi = \mathbf{Q}(\sigma) = \mathbf{Q}(\rho)'\tau n$ (where, as usual, $\mathbf{Q}(\rho)'$ denotes the permutation obtained from $\mathbf{Q}(\rho)$ by removing n). Thus $\mathbf{Q}(\rho) = \mathbf{Q}(\rho)'n = M_1P_1 \cdots M_{k-2}P_{k-2}M'_{k-1}n$, which concludes the proof. □

The last four propositions allow us to state the announced recursive description of all preimages of a given permutation.

Theorem 3.1.7. *Let $\pi = M_1P_1 \cdots M_{k-1}P_{k-1}M_k \in S_n$, with $M_k \neq \emptyset$, and suppose that π is different from the identity permutation. A permutation $\sigma \in S_n$ is a preimage of π if and only if exactly one of the following holds:*

- *$\sigma = \tau\mu_{k-1}P_{k-1}M'_k$, where, with a little abuse of notation, $\tau \in \mathbf{Q}^{-1}(M_1P_1 \cdots M_{k-2}P_{k-2}M'_{k-1}n)$ denotes a preimage of $M_1P_1 \cdots M_{k-2}P_{k-2}M'_{k-1}n$ and M'_{k-1}, M'_k are defined as in Proposition 3.1.6;*
- *if π' is defined (as in Proposition 3.1.5) by removing n from π and $\sigma' = N_1R_1 \cdots N_{s-1}R_{s-1}N_s$ is a preimage of π', then σ is obtained by inserting n in one of the positions to the right of N_{s-1}.*

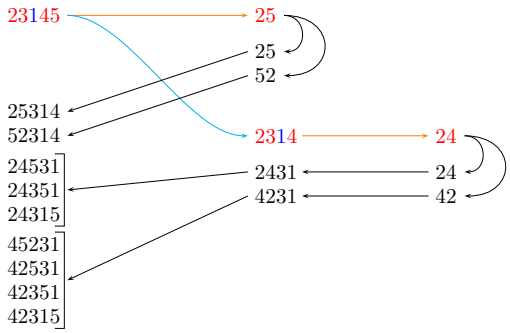
Proof. Notice that, for a given π, exactly one of the above two cases holds: indeed, in the second case there is at least one element of M_k other than n which is moved, whereas this does not happen in the first case. In particular, in the second case a preimage exists only if $|M_k| \geq 2$ (otherwise no preimage of π' can exist). Thus the theorem is a consequence of Propositions 3.1.3 to 3.1.6. □

Thanks to the above theorem, we can also describe a recursive algorithm to determine all preimages of a given permutation $\pi = M_1P_1 \cdots M_{k-1}P_{k-1}M_k$:

- if π is the identity permutation, then the preimages of π are precisely the 321-avoiding permutations (of the same length);
- otherwise

– compute all the preimages of $M_1P_1\cdots M_{k-2}P_{k-2}M'_{k-1}n$ and concatenate them with $\mu_{k-1}P_{k-1}M'_k$;
– if $|M_k| \geq 2$, then compute all the preimages $N_1R_1\cdots N_{s-1}R_{s-1}N_s$ of π' and insert n in each of the positions to the right of N_{s-1}.

Example. To illustrate the above theorem, we now compute all the preimages of the permutation 23145. The various steps of the procedure are depicted in the figure below:



3.2 Enumerative results

The main goal of the present section is to count how many preimages under `Queuesort` a given permutation has. We begin by proving some preliminary facts, which are however relevant enough to deserve a proper presentation.

Our first achievement is an important feature of the `Queuesort` algorithm: the number of preimages of a permutation depends only on the positions of its left-to-right maxima, and not on their values. The proof of this property will stem from a more general fact, that reveals an interesting feature of `Queuesort`.

Recall that, given $\pi \in S_n$, $\mathrm{LTR}_p(\pi)$ denotes the set of the positions of the left-to-right maxima of π, i.e. $\mathrm{LTR}_p(\pi) = \{i \leq n \,|\, \pi_i \text{ is a left-to-right maximum of } \pi\}$.

Lemma 3.2.1. *Let $\pi, \sigma \in S_n$, with σ obtained from π by moving some of its left-to-right maxima to the left, from positions $i_1, \ldots, i_h$ to positions $i'_1, \ldots, i'_h$, where $i_1 < \cdots < i_h$, $i'_1 < \cdots < i'_h$, $i'_j < i_j$ for every $j = 1, \ldots, h$. Then, for every $j = 1, \ldots, h$, the position of $\sigma_{i'_j}$ in $\mathbf{Q}(\sigma)$ is at least i_j.*

Proof. We start by observing that all the elements of π that are moved to the left must remain left-to-right maxima also in σ. We now perform `Queuesort` on σ and describe what happens to the left-to-right maxima in positions $i'_1, \ldots, i'_h$. The element $\sigma_{i'_h}$ is the first one which is considered. Since all the elements between positions i'_h and i_h are smaller that $\sigma_{i'_h}$ (by construction of σ), its final position in $\mathbf{Q}(\sigma)$ will be at least i_h. Moving on, suppose that all the elements $\sigma_{i'_{j+1}}, \ldots, \sigma_{i'_h}$ have already been processed, and their final positions are at least $i_{j+1}, \ldots, i_h$, respectively. When `Queuesort` considers $\sigma_{i'_j}$, the only elements between positions i'_j and i_j that could possibly be greater than $\sigma_{i'_j}$ are the previously moved left-to-right maxima, whose positions are however strictly greater than i_j (since $i_j < i_{j+1}$). Therefore, also the final position of $\sigma_{i'_j}$ in $\mathbf{Q}(\sigma)$ is at least i_j. Repeating the argument for all the moved left-to-right maxima gives the thesis. □

Lemma 3.2.2. *Let $\pi = \pi_1 \cdots \pi_n = M_1 P_1 \cdots M_{k-1} P_{k-1} M_k \in S_n$ and let σ be a preimage of π. As usual, let μ_i be the last element of M_i. Then, for all $i = 1, \ldots, k-1$, there exists a left-to-right maximum of π which is to the left of μ_i in σ and to the right of μ_i in π.*

Proof. By Proposition 3.1.2, μ_i is not a left-to-right maximum of σ. Thus there must be a left-to-right maximum of σ to the left of μ_i which is greater than it. Such an element cannot be to the left of μ_i in π, since μ_i is a left-to-right maximum of π, and this gives the thesis. □

Corollary 3.2.3. *Using the notations of the above lemma, the largest element of σ moved by* `Queuesort` *belongs to M_k.*

Theorem 3.2.4. *Let $\rho, \pi \in S_n$. If $\mathrm{LTR}_p(\rho) \subseteq \mathrm{LTR}_p(\pi)$, then $|\,\mathbf{Q}^{-1}(\rho)| \leq |\,\mathbf{Q}^{-1}(\pi)|$.*

Proof. Define a map $f : \mathbf{Q}^{-1}(\rho) \to \mathbf{Q}^{-1}(\pi)$ as follows. Given $\tau \in \mathbf{Q}^{-1}(\rho)$, suppose that, performing `Queuesort` on τ, the left-to-right maxima of τ in positions $i'_1 < \cdots < i'_h$ are moved to the right to positions $i_1 < \cdots < i_h$, respectively. Define $f(\tau) = \sigma$ as the permutation obtained from π by moving the elements in positions $i_1, \ldots, i_h$ to the left to positions $i'_1, \ldots, i'_h$, respectively.

We now show that f is well defined, i.e. that indeed $\sigma \in \mathbf{Q}^{-1}(\pi)$. Since $\mathrm{LTR}_p(\rho) \subseteq \mathrm{LTR}_p(\pi)$, $\pi_{i_1}, \ldots, \pi_{i_h}$ are left-to-right maxima of π, so, by Lemma 3.2.1, performing `Queuesort` on σ moves them at least to their original positions (possibly to the right of them). First of all, we prove that no further elements of σ are moved by `Queuesort`. By contradiction, let $\sigma_{a'}$ be a left-to-right maximum of σ that is moved by `Queuesort`, with $a' \neq i_j$ for every j. Let a be such that $\pi_a = \sigma_{a'}$. Since, by construction of σ, the set of elements preceding π_a in π is a subset of the elements preceding $\sigma_{a'}$ in σ, we have that π_a is a left-to-right maximum of π. If we now look at ρ, we have two possibilities concerning the index a: either $a \notin \mathrm{LTR}_p(\rho)$ or $a \in \mathrm{LTR}_p(\rho)$.

If $a \notin \mathrm{LTR}_p(\rho)$, set $b = \max\{c < a \mid \rho_c \text{ is a left-to-right maximum of } \rho\}$. By Lemma 3.2.2 applied to ρ, there exists a $\bar{t}$ such that $i'_{\bar{t}} < b < i_{\bar{t}}$. In particular, by definition of b we have $i'_{\bar{t}} < b < a < i_{\bar{t}}$, and so $\pi_a < \pi_{i_{\bar{t}}}$ (because $i_{\bar{t}} \in \mathrm{LTR}_p(\rho) \subseteq \mathrm{LTR}_p(\pi)$). Since in σ the element π_a is to the right of the element $\pi_{i_{\bar{t}}}$, that is in position $i'_{\bar{t}}$ (as a consequence of the way σ is obtained from π), we have that $\sigma_{a'} = \pi_a$ is not a left-to-right maximum of σ, which is a contradiction.

If instead $a \in \mathrm{LTR}_p(\rho)$, we can assume that there exists no $\bar{t}$ such that $i'_{\bar{t}} < a < i_{\bar{t}}$, because otherwise the same argument of the previous case would lead to the contradiction that $\sigma_{a'}$ is not a left-to-right maximum of σ. We now consider ρ. In particular, denoting with M_j the block of left-to-right maxima containing ρ_a, let b be the index of the rightmost element of M_j. Applying Lemma 3.2.2, we find that there exists $\bar{u}$ such that $i'_{\bar{u}} < b < i_{\bar{u}}$. By the above assumption, we have $a < i'_{\bar{u}}$. This implies that, when we apply `Queuesort` to σ, the element in position $i'_{\bar{u}}$ will move at least to position $i_{\bar{u}}$, and the element $\sigma_{a'} = \pi_a$ does not change position because all the elements between π_a and π_b (included) do not move. This is however in contrast with the hypothesis that $\sigma_{a'}$ is moved by `Queuesort`.

We have thus shown that the only elements in σ moved by `Queuesort` are those in positions $i'_1, \ldots, i'_h$. It remains to prove that they are moved to their original positions $i_1, \ldots, i_h$. We first observe that $\sigma_{i'_h}$ goes back to position i_h because, applying Corollary 3.2.3 to ρ, we obtain that (when applying `Queuesort`) $\tau_{i'_h}$ goes back to the last block of left-to-right maxima of ρ, and so $\sigma_{i'_h}$ goes back to the last block of left-to-right maxima of π as well, and this means that it cannot move further. Now suppose by contradiction that there exists some $j < h$ such that the element $\sigma_{i'_j}$ is moved by `Queuesort` to a position strictly to the right of i_j. Let $\bar{j}$ be the maximum of such j's. Since $\rho = \mathbf{Q}(\tau)$, ρ has a left-to-right maximum in position $i_{\bar{j}} + 1$, so the same is true for π. Now observe that, in σ, `Queuesort` moves $\sigma_{i'_{\bar{j}}}$ to a position strictly to the right of $i_{\bar{j}}$, so at that point of the execution of `Queuesort` the element $\pi_{i_{\bar{j}}+1}$ is not in position

$i_{\bar{j}} + 1$. Thus it must be to the right of that position (it cannot be to the left by Lemma 3.2.1), which contradicts the maximality of $\bar{j}$.

We have thus shown that the map f is well defined. To conclude, we observe that it is also injective, because each preimage of ρ is uniquely determined by the i_j's and i'_j's. □

Corollary 3.2.5. *If two permutations π and ρ have their left-to-right maxima in the same positions, then they have the same number of preimages.*

Proof. By hypothesis we have $\mathrm{LTR}_p(\pi) = \mathrm{LTR}_p(\rho)$, so the previous theorem implies that $|\mathbf{Q}^{-1}(\pi)| = |\mathbf{Q}^{-1}(\rho)|$. □

Another relevant consequence of Theorem 3.2.4 is the following proposition, which reveals to be a useful tool in several circumstances.

Proposition 3.2.6. *Let $\pi = M_1 P_1 \cdots M_{k-1} P_{k-1} M_k \in S_n$, with $|M_i| = |\{\mu_i\}| = 1$ for a given $i \neq 1, n$. Let $\rho = N_1 R_1 \cdots N_{i-1} R'_i N_{i+1} \cdots N_{k-1} R_{k-1} N_k \in S_n$, with $|M_j| = |N_j|$, $|P_j| = |R_j|$, for all $j \neq i$, $|P_{i-1} M_i P_i| = |R'_i|$ and such that R'_i does not contain any left-to-right maximum of ρ. Then $|\,\mathbf{Q}^{-1}(\rho)| = |\,\mathbf{Q}^{-1}(\pi)|$.*

Proof. We start by observing that $\mathrm{LTR}_p(\rho) \subset \mathrm{LTR}_p(\pi)$, so, by Theorem 3.2.4, we have $|\,\mathbf{Q}^{-1}(\rho)| \leq |\,\mathbf{Q}^{-1}(\pi)|$.

We now show that $|\,\mathbf{Q}^{-1}(\rho)| \geq |\,\mathbf{Q}^{-1}(\pi)|$. In fact, due to Proposition 3.1.2, μ_i cannot be moved when looking for a preimage of π. So, given a preimage of π, we can construct a preimage of ρ as described in the proof of Theorem 3.2.4 and show that it is indeed a preimage of ρ using similar arguments. Therefore ρ has at least as many preimages as π, thus giving the desired inequality. □

In other words, the previous proposition says that, in some sense, the presence of isolated left-to-right maxima does not affect the number of preimages.

We now provide some results concerning permutations with a given number of preimages. We already know (Proposition 3.0.1) that $\pi \in S_n$ has no preimages if and only if its last element is different from n. Therefore, setting $Q_n^{(k)} = \{\pi \in S_n \,|\, |\,\mathbf{Q}^{-1}(\pi)| = k\}$ and $q_n^{(k)} = |Q_n^{(k)}|$, we have that $Q_n^{(0)} = \{\pi \in S_n \,|\, \pi_n \neq 0\}$ and $q_n^{(0)} = (n-1)! \cdot (n-1)$. The next propositions deal with $Q_n^{(1)}$ and $Q_n^{(2)}$.

Proposition 3.2.7. *For all n, we have $Q_n^{(1)} = \{\pi \in S_n \,|\, \pi_n = n$ and π does not have two adjacent left-to-right maxima$\}$. As a consequence,*

$$q_n^{(1)} = (n-1)! \cdot \sum_{i=0}^{n-1} \frac{(-1)^i}{i!},$$

that is the $(n-1)$-th derangement number (sequence A000166 in [22]).

Proof. Suppose first that $\pi \in S_n$ does not have any two adjacent left-to-right maxima and its last element is n. Then certainly $|\,\mathbf{Q}^{-1}(\pi)| \neq 0$. Using repeatedly Proposition 3.2.6, we can assert that $|\,\mathbf{Q}^{-1}(\pi)| = |\,\mathbf{Q}^{-1}(\rho)|$, where ρ is any permutation of length n whose only left-to-right maxima are in positions 1 and n. Thus, in particular, $|\,\mathbf{Q}^{-1}(\pi)| = |\,\mathbf{Q}^{-1}((n-1)(n-2)\cdots 21n)|$, and it is clear (by a direct computation, or by invoking Theorem 3.1.7) that the last quantity is equal to 1.

On the other hand, suppose that the last element of π is n (otherwise, of course, π has no preimages), but there exists i such that π_i, π_{i+1} are both left-to-right maxima. We could

now invoke Theorem 3.1.7 to find two distinct preimages of π. However we prefer to explicitly describe two such preimages, since it is so simple. One preimage of π can be obtained by moving n at the beginning of the permutation; in other words, $\pi_n\pi_1\pi_2\cdots\pi_{n-1}$ is a preimage of π (this is trivial to verify). Another preimage of π can be obtained by placing n between π_i and π_{i+1} and replacing $\pi_1\cdots\pi_i$ with any of its preimages τ (notice that the set of preimages of $\pi_1\cdots\pi_i$ is indeed not empty, since π_i is a left-to-right maximum); in other words, $\tau n\pi_{i+1}\cdots\pi_{n-1}$ is a preimage of π (this is also quite easy to realize, and so left to the reader). The two above described preimages are indeed distinct, since the former starts with n whereas the latter does not.

To conclude the proof, we recall the so called Foata's fundamental bijection [17], which maps a permutation σ written in one-line notation to the permutation in cycle notation obtained by inserting a left parenthesis in σ preceding every left-to-right maximum, then a right parenthesis where appropriate. Applying such a map to $Q_n^{(1)}$ returns the set of permutations of length n whose only fixed point is n, which is clearly equinumerous with the set of derangements of length $n-1$, as desired. □

Proposition 3.2.8. *For all n, we have $Q_n^{(2)} = \{\pi \in S_n \,|\, \pi_n = n$ and π does not have two adjacent left-to-right maxima except for the first two elements$\}$. As a consequence, $q_n^{(2)}$ satisfies the recurrence relation*

$$q_{n+1}^{(2)} = (n-1)q_n^{(2)} + (n-1)q_{n-1}^{(2)}, \qquad n \geq 3,$$
$$q_0^{(2)} = q_1^{(2)} = q_3^{(2)} = 0, \quad q_2^{(2)} = 1.$$

Proof. By an argument completely analogous to that of the previous proposition, we can show that any permutation ending with n and whose only adjacent left-to-right maxima are the first two elements has exactly two preimages, that are obtained by moving n into the first and into the second position, respectively.

On the other hand, if $\pi \in S_n$ does not have adjacent left-to-right maxima, we know by the previous proposition that $|\,\mathbf{Q}^{-1}(\pi)| = 1$. Moreover, if π contains two adjacent left-to-right maxima $\pi_i = \alpha$ and π_{i+1}, with $i \geq 2$, then π has at least three preimages, obtained as follows: either move n into the first position, or move both α and n into the first two positions, or replace α with n and move α into the first position.

We will now prove the recurrence relation. We can immediately see that the initial conditions hold. First observe that, if we take a permutation in $Q_{n+1}^{(2)}$, then we can remove $n+1$, which is the last element, to obtain a permutation of length n without any two adjacent left-to-right maxima except for the first two elements, and such that its last element is not n. We can now use Foata's fundamental correspondence [17] to get a bijection with the set A_n of permutation $\pi \in S_n$ with exactly one fixed point, say $\pi_i = i$, such that $i < \max\alpha$ whenever α is a cycle of π with at least two elements. For instance, $\pi = (321)(4)(65)$ is not in A_6 since $4 > \max(321)$. Thus, set $a_n = |A_n|$, the recurrence for $q_n^{(2)}$ is equivalent to the following, which we will now prove:

$$a_n = (n-1)a_{n-1} + (n-1)a_{n-2}, \qquad n \geq 4.$$

Given $\rho \in A_n$, its maximum n necessarily belongs to a cycle with at least two elements. If such a cycle has exactly two elements, say n and α, with $1 \leq \alpha \leq n-1$, then we can remove them to obtain a permutation of length $n-2$ belonging to A_{n-2}, thus obtaining the second summand of the r.h.s. of the recurrence equation, i.e. $(n-1)a_{n-2}$. If instead the cycle contains at least three elements, we have two distinct cases. If it contains at least one other element larger than the fixed point, then we can remove n to obtain a permutation in A_{n-1}. Observe that, to invert this construction, given a permutation in A_{n-1}, there are $n-2$ possible positions

to insert n in order to get a permutation of A_n (since n can be placed before every element except for the fixed point). Finally, if all the elements of the cycle containing n are smaller than the fixed point, except for n, then we can proceed as follows. Let α be the element following n in its cycle (i.e. $\alpha = \rho_n$ in the one-line notation of ρ) and let β be the fixed point. Remove n from ρ, and switch α with β. By construction, we obtain a permutation of length $n-1$ with exactly one fixed point, which is smaller than the maximum of each cycle, and from which we can revert back to the starting permutation uniquely. The last two cases together account for the first summand of the right-hand side of the recurrence relation, i.e. $(n-1)a_{n-1}$. □

Sequence $q_n^{(2)}$ starts $0, 0, 1, 0, 2, 6, 32, 190\ldots$ and is essentially A055596 in [22]. We thus deduce, for $n \geq 2$, the closed formula $q_n^{(2)} = (n-1)! - 2q_n^{(1)}$ (recall that $q_n^{(1)}$ equals the $(n-1)$-th derangement number), as well as the exponential generating function

$$\sum_{n\geq 0} q_n^{(2)} \frac{x^n}{n!} = \int \frac{(2 - x - 2e^{-x})}{1-x} dx.$$

We have thus seen that there exist permutations having 0, 1 or 2 preimages. We now show (Propositions 3.2.10 and 3.2.11) that there exist permutations having any number of preimages, except for 3. Before that, we need a preliminary result, which is of interest in its own.

Lemma 3.2.9. *Let $\pi = M_1 P_1 M_2$, with $|M_2| = 1$. Then $|\mathbf{Q}^{-1}(\pi)| = C_{m_1}$, the m_1-th Catalan number.*

Proof. By hypothesis $\pi = M_1' \mu_1 P_1 n$, where μ_1 is the last element of M_1. Applying Theorem 3.1.7, we obtain that all the preimages of π are of the form $\tau \mu_1 P_1$, with $\tau \in \mathbf{Q}^{-1}(M_1' n)$. However, the permutation $M_1' n$ is increasing, hence $|\mathbf{Q}^{-1}(M_1' n)| = C_{m_1}$, which gives the thesis. □

Proposition 3.2.10. *Given $n \geq 2$, let $\pi = n(n-1)(n-2)\cdots 21(n+2)(n+3)(n+1)(n+4) \in S_{n+4}$. Then $|\mathbf{Q}^{-1}(\pi)| = n+2$.*

Proof. We will repeatedly use Theorem 3.1.7 to compute preimages. First of all, observe that only the first case of such a theorem applies to π, since the second-to-last element of π is not $n+3$. Hence, the preimages of π are precisely those permutations of the form $\sigma(n+3)(n+1)$, where σ is any preimage of $n\cdots 1(n+2)(n+4)$. Therefore our problem reduces to the enumeration of such σ. Preimages of $n\cdots 1(n+2)(n+4)$ can again be computed using Theorem 3.1.7. In particular, the first case of that theorem gives rise to a single preimage, which is $(n+4)n\cdots 1(n+2)$. Looking at the second case, we first have to compute the set of preimages of $n\cdots 1(n+2)$, which is easily seen to consist of the single permutation $\tau = (n+2)n\cdots 1$ of length $n+1$ (notice that this holds since we are supposing that $n \geq 2$, so that τ is not an increasing permutation); then we have to suitably insert $n+4$ into τ, and this can be done in $n+1$ different ways, namely by inserting $n+4$ to the right of each element of τ. Thus we have found that the total number of preimages of π is precisely $n+2$, as desired. □

The previous proposition can be easily extended to the case $n = 0$, since it is immediate to check that $|\mathbf{Q}^{-1}(2314)| = 2$. However, it does not hold when $n = 1$, as $|\mathbf{Q}^{-1}(13425)| = 5$. The next proposition shows that this is no accident.

Proposition 3.2.11. *There exists no permutation π such that $|\mathbf{Q}^{-1}(\pi)| = 3$.*

Proof. As usual, suppose that π is decomposed as $\pi = M_1 P_1 \cdots M_{k-1} P_{k-1} M_k$. If $k = 1$, then $|\mathbf{Q}^{-1}(\pi)|$ is a Catalan number, which is certainly different from 3. If $k \geq 2$, we distinguish two cases, depending on the cardinality of M_2.

- If $|M_2| > 1$, denote with α and β the two largest elements of M_2, with $\alpha < \beta$, so that $M_2 = M_2''\alpha\beta$. Then the following four permutations are all preimages of π, and are all distinct (this is a consequence of Theorem 3.1.7, or it can be checked by applying to each of them our equivalent version of `Queuesort`):
  - $\beta M_1 P_1 M_2'' \alpha P_2 \cdots M_k$;
  - $\alpha\beta M_1 P_1 M_2'' P_2 \cdots M_k$;
  - $\alpha M_1 \beta P_1 M_2'' P_2 \cdots M_k$;
  - $\alpha M_1 \beta P_1 M_2'' \beta P_2 \cdots M_k$.
- If $|M_2| = 1$, then Proposition 3.2.6 tells us that the number of preimages of π is the same as the number of preimages of any permutation $\sigma = N_1 R_1 \cdots N_{k-2} R_{k-2} N_{k-1}$ such that $|N_1| = |M_1|$, $|R_1| = |P_1 M_2 P_2|$ and $|N_i| = |M_{i+1}|$, $|R_i| = |P_{i+1}|$, for all $i \geq 2$. We can iterate this argument until either the second maximal sequence of consecutive left-to-right maxima of the resulting permutation has cardinality ≥ 2 or we obtain a permutation σ of the form $\sigma = N_1 R_1 N_2$, with $|N_2| = 1$. In the former case (which occurs when there exists $i \geq 2$ such that $|M_i| \geq 2$), we are in the situation described in the previous item point. In the latter case (which occurs when $|M_i| = 1$, for all $i \geq 2$), we can apply Lemma 3.2.9, thus obtaining again that $|\mathbf{Q}^{-1}(\pi)|$ is a Catalan number. □

The last part of our paper is devoted to find an expression for the number of preimages of a generic permutation π of the form $\pi = M_1 P_1 M_2$. We have already proved (Lemma 3.2.9) that, when $|M_2| = 1$, we get a Catalan number. In order to find a general formula, we need some preliminary work.

For any $n \geq 1$ and $2 \leq i \leq n$, define $G_{n,i} = \{\pi \in \mathrm{Av}_n(321) \mid \pi_i$ is not a left-to-right maximum and, for every $j < i$, π_j is a left-to-right maximum$\}$, and $g_{n,i} = |G_{n,i}|$. Also define $G_{n,n+1} = \{id_n\}$, $g_{n,n+1} = 1$.

Proposition 3.2.12. *For every $n \geq 2$, the following recurrence holds:*

$$g_{n,i} = \begin{cases} C_{n-1}, & \text{if } i = 2, \\ n-1, & \text{if } i = n, \\ g_{n-1,i-1} + g_{n,i+1} & \text{if } 3 \leq i \leq n-1. \end{cases} \tag{3.1}$$

Proof. We observe that $G_{n,2} = \{\pi \in \mathrm{Av}_n(321) \mid \pi_2 = 1\}$, which is in bijection with $\mathrm{Av}_{n-1}(321)$ (by the removal of π_2). As a consequence, $g_{n,2} = C_{n-1}$.
Moreover, given $\pi \in G_{n,n}$, the first $n-1$ elements are in increasing order, and the last element can be chosen arbitrarily, except that it cannot be n. Thus $g_{n,n} = |G_{n,n}| = n-1$.
To prove the recurrence relation, we describe a bijection f from $G_{n,i}$ to $G_{n-1,i-1} \cup G_{n,i+1}$ (where $\cup$ denotes disjoint union). Given $\pi \in G_{n,i}$, we define $f(\pi)$ as follows:

$$f(\pi) = \begin{cases} \pi_2 \cdots \pi_n & \in G_{n-1,i-1}, \quad \text{if } \pi_1 = 1, \\ \pi_1 \cdots \pi_{i-1}\pi_{i+1}\pi_i\pi_{i+2}\cdots\pi_n & \in G_{n,i+1}, \quad \text{if } \pi_1 \neq 1 \text{ and } \pi_{i+1} > \pi_{i-1}, \\ \pi_i\pi_1 \cdots \pi_{i-1}\pi_{i+1}\cdots\pi_n & \in G_{n,i+1}, \quad \text{if } \pi_1 \neq 1 \text{ and } \pi_{i+1} < \pi_{i-1}. \end{cases}$$

It is easy to check that f is indeed a bijection by explicitly constructing its inverse. □

The above recurrence relation also extends to the case $i = n$ (and also $i = n+1$, by setting all undefined values to be 0).

Corollary 3.2.13. *For every $n \geq 1$, $2 \leq i \leq n+1$, $g_{n,i} = \binom{2n-i+1}{n} \frac{i-1}{2n-i+1}$.*

Proof. Up to suitably rescaling the indices, the sequence $g_{n,i}$ satisfies the same initial conditions and recurrence relation as sequence A033184 in [22]. More specifically, $g_{n,i} = A033184(n, i-1)$. This gives the desired closed formula. □

If we represent sequence A033184 as a triangle, we obtain one of the so-called Catalan triangles. Its entries are sometimes called ballot numbers [1].

Before stating and proving the main result of this section, we need to introduce one more statistic on 321-avoiding permutations. Namely, we define $b_{n,i} = |\{\pi \in \mathrm{Av}_n(321) \mid \pi_i = n\}|$. Using the bijection that associates every permutation with its group-theoretic inverse, we can see that $b_{n,i} = |\{\pi \in \mathrm{Av}_n(321) \mid \pi_n = i\}|$.

The next proposition shows that the $b_{n,i}$'s are another version of the ballot numbers.

Proposition 3.2.14. *The numbers $b_{n,i}$ correspond to sequence* A009766 *of [22], if we properly shift them. Namely, the element $b_{n,i}$ is the element of indices $n-1, i-1$ of sequence* A009766*.*

Proof. We will prove the statement by showing that the $b_{n,i}$'s satisfy the same recurrence relation (and initial conditions) as sequence A009766, namely $b_{n+1,i} = \sum_{j=1}^{i} b_{n,j}$ (and $b_{1,1} = 1$, which is plainly true). Let $\pi \in \mathrm{Av}_{n+1}(321)$ such that $\pi_{n+1} = i$. By removing i (and rescaling), we get a permutation $\pi' \in \mathrm{Av}_n(321)$. If $\pi'_n = j \neq n$, then necessarily $j < i$ (since π avoids 321), and each such π' gives uniquely a permutation in $\mathrm{Av}_{n+1}(321)$ ending with i (by appending i to π'). On the other hand, if $j = n$, then $\pi' = M_1 P_1 \cdots M_{k-1} P_{k-1} M_k$ is such that M_k is not empty, and the last element of P_{k-1} is smaller than i, because otherwise π would contain the pattern 321. Thus, denoting with α the number of such permutations, we have

$$b_{n+1,i} = \sum_{j=1}^{i-1} b_{n,j} + \alpha.$$

To determine α we observe that, removing M_k from π', if π' is not the identity permutation, we obtain a bijection with the set of permutations avoiding 321 of any length $m < n$ whose last element j is smaller than i and different from m. Hence, using induction (on n) and repeatedly applying the recurrence relation for the $b_{n,i}$'s recalled at the beginning of this proof, we get that

$$\alpha = 1 + \sum_{m=2}^{n-1} \sum_{j=1}^{\min(m-1,i-1)} b_{m,j} = b_{n,i}.$$

Indeed, the inner sum for a fixed m is the same as the left-hand side of the recurrence relation, except for the element $b_{m,m}$, which is obtained by the inner sum for $m-1$ (for $m = 2$, we observe that $b_{2,2} = 1$). Summing up, we get:

$$b_{n+1,i} = \sum_{j=1}^{i-1} b_{n,j} + b_{n,i} = \sum_{j=1}^{i} b_{n,j},$$

as desired. □

Remark 3.2.15. *Corollary 3.2.13 and Proposition 3.2.14 imply that $g_{n,i} = b_{n,n+2-i}$ for all $n \geq 1$, $2 \leq i \leq n+1$.*

Since we will frequently use two different recursions for $b_{n,i}$ (both mentioned in [22]), we record them here for ease of further reference:

$$b_{n+1,i} = \sum_{j=1}^{i} b_{n,j}, \quad n, i \geq 1, \tag{3.2}$$

$$b_{n,i+1} = b_{n,i} + b_{n-1,i+1}, \quad n \geq 2, i \geq 1. \tag{3.3}$$

Our last preliminary result concerns the enumeration of another subfamily of 321-avoiding permutations. In the statement we will also make use of the *multinomial coefficient* $\left(\!\binom{u}{v}\!\right)$, which counts the number of multisets of cardinality v over a set of cardinality u.

Lemma 3.2.16. *The number of* 321*-avoiding permutations of length $n+k$ whose k largest elements are left-to-right maxima is*

$$\sum_{i=0}^{n-1} \left(\!\binom{n-i+1}{k}\!\right) b_{n,i+1}.$$

Proof. Let $\pi' \in \mathrm{Av}_n(321)$. Thus, if $\pi' = M_1 P_1 \cdots M_t$, we have that $P_1 \cdots P_{t-1}$ is an increasing sequence. How many ways do we have to insert $n+1, \ldots, n+k$ into π' in order to obtain a permutation $\pi \in \mathrm{Av}_{n+k}(321)$ such that the k largest elements are left-to-right maxima? We can distinguish two cases.
If π' is the identity permutation of length n, then we just have to insert the k elements $n+1, \cdots, n+k$ in increasing order in any of the $n+1$ possible positions, and we can do that in $\left(\!\binom{n+1}{k}\!\right)$ ways.
If π' is not the identity permutation of length n, let i be the position of the last element of M_{t-1}, with $1 \leq i \leq n-1$. Then we can insert the k elements $n+1, \ldots, n+k$ in any of the $n-i+1$ positions following i in increasing order. Indeed, inserting an element in a previous position would form an occurrence of the pattern 321. As before, this can be done in $\left(\!\binom{n-i+1}{k}\!\right)$ ways. Moreover, the permutations $\pi' \in \mathrm{Av}_n(321)$ such that the last element of M_{t-1} is in position i are $\sum_{j=i+1}^{n} b_{j,i}$. In fact, by removing M_t from π' we obtain a permutation of length j, with $i < j \leq n$, that has its maximum in position i, and the number of these permutations is $b_{j,i}$ by definition. Summing up, the number of 321-avoiding permutations of length $n+k$ whose k largest elements are left-to-right maxima is

$$\left(\!\binom{n+1}{k}\!\right) + \sum_{i=1}^{n-1} \left(\!\binom{n-i+1}{k}\!\right) \sum_{m=i+1}^{n} b_{m,i}. \tag{3.4}$$

Observe that

$$\sum_{m=i+1}^{n} b_{m,i} = b_{n,i+1}. \tag{3.5}$$

Indeed, by iteration of recurrence (3.3), we obtain $b_{n,i+1} = b_{n,i} + b_{n-1,i+1} = b_{n,i} + b_{n-1,i} + b_{n-2,i+1} = \cdots = \sum_{m=i+1}^{n} b_{m,i}$. Plugging (3.5) into expression (3.4), we thus obtain

$$\left(\!\binom{n+1}{k}\!\right) + \sum_{i=1}^{n-1} \left(\!\binom{n-i+1}{k}\!\right) b_{n,i+1} = \sum_{i=0}^{n-1} \left(\!\binom{n-i+1}{k}\!\right) b_{n,i+1},$$

which is the thesis. □

We are now ready to state our main result concerning the enumeration of preimages of permutations of the form $\pi = M_1 P_1 M_2$.

Theorem 3.2.17. *Let $\pi = M_1 P_1 M_2 \in S_n$, with $M_2 \neq \emptyset$. Then*

$$|\mathbf{Q}^{-1}(\pi)| = \sum_{i=1}^{m_2} \sum_{j=0}^{i-1} \binom{i-1}{j} \left(\sum_{l=0}^{m_1-2} \left(\!\binom{m_1-l}{j+1}\!\right) b_{m_1-1,l+1} \right) \left(\sum_{k=2}^{m_2-i+1} g_{m_2-i,k} \left(\!\binom{k}{p_1+i-j-1}\!\right) \right), \tag{3.6}$$

where all the summations are set to be 1 whenever the set of indices is empty.

Proof. Our goal is to describe how to obtain a generic preimage σ from π in such a way that we are able to count them. Let μ_1 be the last element of M_1. By Lemma 3.2.2, there is at least one element of M_2 which is to the left of μ_1 in σ. Let β be the rightmost of such elements, and suppose that β is the i-th element of M_2. Clearly $1 \leq i \leq m_2$. Let j be the number of elements of M_2 smaller than β and to the left of μ_1 in σ, with $0 \leq j \leq i-1$. They can be chosen in $\binom{i-1}{j}$ different ways. In other words, σ can be written as $\sigma = L\mu_1 R$, where L is a suitable permutation of the elements of M_1 other that μ_1 and the $j+1$ elements of M_2 mentioned above, and R is a suitable permutations of the remaining elements (that is, the elements of P_1 and the remaining elements of M_2). We now wish to characterize (and count) the permutations L and R giving rise to a preimage of π.

Concerning L, this has to be a permutation whose $j+1$ aforementioned elements are left-to-right maxima and such that $\mathbf{Q}(L)$ is an increasing sequence. This means that L is a 321-avoiding permutation of length m_1+j whose largest $j+1$ elements are left-to-right maxima. By Lemma 3.2.16, the number of such permutations is

$$\sum_{l=0}^{m_1-2} \left(\!\binom{m_1-l}{j+1}\!\right) b_{m_1-1,l+1}.$$

Concerning R, we can construct it as follows. Start by taking a permutation ρ of the $m_2 - i$ elements of M_2 to the right of β; notice that ρ has to be a 321-avoiding permutation, since applying `Queuesort` to it returns an increasing permutation. Next insert the remaining $p_1+i-j-1$ elements into suitable positions and preserving the order they have in π. Specifically, such elements cannot be inserted to the right of the leftmost non-left-to-right maximum of ρ, whereas any other position is allowed. This is due to the fact that, applying `Queuesort` to any permutation, the relative order of the non left-to-right maxima is preserved. Therefore, denoting with k the position of the leftmost non left-to-right maximum of ρ, we have $2 \leq k \leq m_2 - i$, and there are k allowed positions. Notice that, if ρ is the identity permutation, then the above argument cannot apply; in such a case, the total number of allowed positions is $m_2 - i + 1$. Summing up, and recalling the definition of $g_{n,i}$, the total number of possible permutations R is:

$$\left(\!\binom{m_2-i+1}{p_1+i-j-1}\!\right) + \sum_{k=2}^{m_2-i} g_{m_2-i,k} \left(\!\binom{k}{p_1+i-j-1}\!\right) = \sum_{k=2}^{m_2-i+1} g_{m_2-i,k} \left(\!\binom{k}{p_1+i-j-1}\!\right).$$

Combining the contributions coming from the above arguments, we get the thesis. □

Formula (3.6), in its full generality, is rather involved, and it is not easy to apply it to effectively compute the number of preimages. However, the next lemma will allow us to simplify it.

Lemma 3.2.18. *For every $n \geq 1$, $1 \leq i \leq n$,*

$$b_{n,i} = \sum_{h=1}^{i-1} \binom{n-h}{n-i} b_{i-1,h}. \tag{3.7}$$

Proof. To prove (3.7) we exploit a well-known combinatorial interpretation of the ballot number $b_{n,i}$ in terms of lattice paths starting from $(0,0)$, ending at $(n-1, i-1)$, remaining below the line $y = x$ and using north steps $N = (0,1)$ and east steps $E = (1,0)$. Specifically, each such path can be decomposed into its longest prefix that uses $i-2$ E steps, followed by the remaining suffix. It is clear that the number of such prefixes ending at point $(i-2, h-1)$ is

$b_{i-1,h}$, with $1 \leq h \leq i-1$. Moreover, due to the specific kind of decomposition we have chosen, the remaining suffix can be any sequence of $n-i+1$ E steps and $i-h$ N steps starting with E: there are $\binom{n-h}{n-i}$ such sequences. Summing over h gives the desired formula. □

Theorem 3.2.19. *Let $\pi = M_1 P_1 M_2 \in S_n$, with $M_2 \neq \emptyset$. Then*

$$|\mathbf{Q}^{-1}(\pi)| = \sum_{i=1}^{m_2} \sum_{j=0}^{i-1} \binom{i-1}{j} b_{m_1+j+1,m_1} \cdot b_{m_2+p_1-j,m_2-i+1}. \tag{3.8}$$

Proof. Looking at formula (3.6), we start by observing that

$$\sum_{l=0}^{m_1-2} \left(\!\binom{m_1-l}{j+1}\!\right) b_{m_1-1,l+1} = \sum_{l=0}^{m_1-2} \binom{m_1+j-l}{j+1} b_{m_1-1,l+1} = b_{m_1+j+1,m_1}.$$

Indeed, in the first equality we just express the multinomial coefficient as a binomial, and the second equality comes from Lemma 3.2.18.

Moreover, the summation in (3.6) involving the $g_{n,i}$'s can be treated analogously by means of Lemma 3.2.18, just recalling Remark 3.2.15 and suitably modifying the index of summation (namely replacing k with $h = m_2 - i + 2 - k$). □

Yet another way to express formula (3.6) comes from expanding the ballot numbers of the previous theorem in terms of Catalan numbers.

Corollary 3.2.20. *For $\pi = M_1 P_1 M_2 \in S_n$, the quantity $|\mathbf{Q}^{-1}(\pi)|$ can be expressed as a linear combination of Catalan numbers. More precisely, for any fixed $m_2 = |M_2|$, we have that $|\mathbf{Q}^{-1}(\pi)|$ is a linear combination of the Catalan numbers $C_{m_1}, C_{m_1+1}, \dots C_{m_1+m_2-1}$ with polynomial coefficients in p_1, i.e.:*

$$|\mathbf{Q}^{-1}(\pi)| = \sum_{t=0}^{m_2-1} \omega_{m_2,t}(p_1) C_{m_1+t},$$

where $\omega_{m_2,t}(p_1)$ is a polynomial in p_1 of degree $m_2 - t - 1$, for all t.

Proof. Indeed, if we write $b_{m_1+j+1,m_1} = b_{m_1+j+1,(m_1+j+1)-(j+1)}$, by induction on the difference $j+1$ of the indices, using recurrence (3.3), we get

$$b_{m_1+j+1,m_1} = \sum_{h=1}^{\lfloor \frac{j+1}{2} \rfloor + 1} (-1)^{h-1} \binom{j+2-h}{h-1} C_{m_1+j+1-h}. \tag{3.9}$$

Replacing the ballot number b_{m_1+j+1,m_1} with the above linear combination of Catalan numbers into (3.8), we can indeed express $|\mathbf{Q}^{-1}(\pi)|$ as a linear combination of Catalan numbers. The largest Catalan number occurring in such a linear combination is obtained when h takes its minimum value 1 and j takes its maximum value $m_2 - 1$, which corresponds to $C_{m_1+m_2-1}$. Similarly, the smallest Catalan number is C_{m_1}, corresponding to the minimum value of the difference $j-h$, which is -1. Moreover, by using the closed form for $g_{n,i}$ found in Corollary 3.2.13 and Remark 3.2.15, we observe that $b_{m_2+p_1-j,m_2-i+1}$ can be written as

$$b_{m_2+p_1-j,m_2-i+1} = \binom{2m_2+p_1-i-j}{m_2-i} \frac{p_1-j+i}{2m_2+p_1-i-j}.$$

For any fixed m_2, this is a polynomial of degree $m_2 - i$ in p_1. To conclude the proof, we now determine the degree of the polynomial multiplying C_{m_1+t} in the above mentioned linear

combination, for any t in the range $[0, m_2 - 1]$. Looking at (3.9), the Catalan number C_{m_1+t} (for fixed t) shows up for all $j \geq t$ in (3.8), and so also for all $i \geq t+1$. Notice that, for $i = t+1$, the coefficient of C_{m_1+t} has degree $m_2 - t - 1$ in p_1 (since, in this case, C_{m_1+t} is obtained only when $j = t$ and $h = 1$ in (3.9)), whereas for $i > t + 1$ the resulting polynomials have lesser degree. □

Effective enumerative results can be obtained for small values of the parameter m_2 in formula (3.8). For instance, when $m_2 = 1$, we find the same result stated in Lemma 3.2.9. We are also able to get simple closed formulas when $m_2 = 2$ and $m_2 = 3$.

Corollary 3.2.21. *For $\pi = M_1 P_1 M_2 \in S_n$, we get:*

- $|\mathbf{Q}^{-1}(\pi)| = C_{m_1+1} + (p_1 + 1)C_{m_1}$, *when* $|M_2| = 2$;
- $|\mathbf{Q}^{-1}(\pi)| = C_{m_1+2} + (p_1 + 1)C_{m_1+1} + \frac{1}{2}(p_1 + 1)(p_1 + 4)C_{m_1}$, *when* $|M_2| = 3$.

The above corollary, together with some further calculations, seems to suggest that $\omega_{m_2,t}(p_1) = \omega_{m_2+1,t+1}(p_1)$, for all m_2, t. This could clearly simplify the computations needed to determine $|\mathbf{Q}^{-1}(\pi)|$ when m_2 increases.

Chapter 4

Bubblesort preimages

4.1 Computing the preimages

We begin this chapter presenting a procedure to compute the set $\mathbf{B}^{-1}(\sigma)$ of all preimages of any given permutation σ. This procedure is simple, and can also be implicitly found (together with some of the consequences it bears) in [12]. In this article the authors, using a different approach, obtain some of the results we also get in the present section, such as Corollary 4.1.4.

First, as an immediate consequence of Lemma 2.4.1, a permutation is in the image of $\mathbf{B}$ if and only if it ends with its maximum, therefore we have that $\mathbf{B}^{-1}(\sigma)$ is empty for any σ not ending with its maximum. This trivial case being solved, we now focus on the interesting case where σ does end with its maximum.

Let $\sigma = \sigma_1\sigma_2\dots\sigma_n$ be a permutation of size n which ends with its maximum. Define P as a set which contains only σ. For each i from n down to 2, do the following: for each $\pi \in P$,

- if π_{i-1} is not a left-to-right maximum of π, then replace π in the set P by the permutation $\pi_1\dots\pi_i\pi_{i-1}\dots\pi_n$ (that is to say, we swap π_i and π_{i-1});
- if π_{i-1} is a left-to-right maximum of π, then π stays in the set P, and in addition we add in P the permutation $\pi_1\dots\pi_i\pi_{i-1}\dots\pi_n$ (where π_i and π_{i-1} are swapped).

Example 4.1.1. The table below shows the evolution of the set P of the above procedure, for $\sigma = 325146$.

$i = \dots$	initialization	6	5	4	3	2
P contains	325146	325164	325614	325614	352614	352614, 532614
				326514	362514	362514, 632514

We may note that, when starting any step i, for any $\pi \in P$, π_i is always a left-to-right maximum of π, and π_{i-1} is a left-to-right maximum of π if and only if σ_{i-1} is a left-to-right maximum of σ. Indeed, all steps until step i (excluded) of the above procedure leave the prefixes of length $i-1$ unchanged.

It is useful to have a different (although equivalent) presentation of this procedure, which we now give. Starting from σ, where we see the rightmost element σ_n as distinguished, we move the distinguished element to the left until it becomes the leftmost, according to the following rules.

- If the element immediately to the left of the distinguished one is not a left-to-right maximum of the current sequence, then the distinguished element is forced to move to the left (*i.e.* is swapped with its left neighbor). The distinguished element remains the same.

- If the element immediately to the left of the distinguished one is a left-to-right maximum of the current sequence, then the distinguished element may either move to the left or stay in place. In the first case, the distinguished element remains the same. In the second case, the distinguished element becomes the left neighbor of the previously distinguished element.

It is easy to see that the set P computed by the original procedure consists of all possible results of applying this alternative procedure.

Some remarks (all easily observed) about this alternative procedure are useful. First, the index i (between n and 2) of a given step of the original procedure always corresponds to the position of the distinguished element in the evolving sequence. Second, the distinguished element is always a left-to-right maximum of σ, and also of the evolving sequence. Third, for any sequence π produced, the elements which were at some point distinguished in the sequence are exactly the left-to-right maxima of π.

Theorem 4.1.2. *Let σ be any permutation ending with its maximum, and P be the set produced by the procedure above. Then P is the set of preimages of σ under $\mathbf{B}$, that is to say, $P = \mathbf{B}^{-1}(\sigma)$.*

Proof. Assume first that a sequence π has been produced by the above procedure. It means that π has been produced from σ by considering some left-to-right maxima of σ, from the right to the left, and moving these left-to-right maxima to the left, until they reach the position of the next left-to-right maximum which moves to the left. This is exactly undoing the action of $\mathbf{B}$. More precisely, let $\pi = \mu_1 A_1 \cdots \mu_k A_k$ be the LTR-max decomposition of π (defined in Section 2.1, where the μ_i are the left-to-right maxima of π. By construction of π, A_i is not empty if and only if μ_i has been moved by our procedure. Applying $\mathbf{B}$ to π yields $A_1\mu_1 \cdots A_k\mu_k$, thus exactly the elements that were moved by our procedure will be moved to the right by `Bubblesort`. Moreover, $\mathbf{B}$ moves μ_i to the right until it reaches the position immediately before μ_{i+1}, and we claim that this is μ_i's original position in σ: indeed, our procedure only moves the μ_j's, so it must have started moving μ_i to the left immediately after considering μ_{i+1}. This proves that $\mathbf{B}(\pi) = \sigma$, and therefore $P \subseteq \mathbf{B}^{-1}(\sigma)$.

For the converse inclusion, we proceed by induction on the size of σ. The statement is obvious for size 1. So, let us consider $\pi \in \mathbf{B}^{-1}(\sigma)$, for σ of size greater than 1. We decompose π around its maximal element as $\pi = LnR$. Then $\mathbf{B}(\pi) = \mathbf{B}(L)Rn$, so that $\sigma = \mathbf{B}(L)Rn$. Starting from $\sigma = \mathbf{B}(L)Rn$, the above procedure is always allowed to move n towards the left, and may decide when reaching $\mathbf{B}(L)nR$ to distinguish the last element of $\mathbf{B}(L)$ instead of n. Indeed, $\mathbf{B}(L)$ necessarily ends with its maximum, so that the last element of $\mathbf{B}(L)$ is a left-to-right maximum. Since $\mathbf{B}(L)$ is a sequence shorter than σ ending with its largest value, we may apply the induction hypothesis to it, and deduce that L has been produced by the above procedure applied to $\mathbf{B}(L)$. Combining these two facts, it follows that our procedure applied to $\sigma = \mathbf{B}(L)Rn$ can produce $LnR = \pi$, therefore showing that $\mathbf{B}^{-1}(\sigma) \subseteq P$. □

Theorem 4.1.2 has several consequences. First, we can refine Corollary 2.4.2 and describe $\mathbf{B}^{-1}(\sigma)$ exactly from the left-to-right maxima of σ.

Corollary 4.1.3. *Let σ be a permutation of size n ending with its maximum (*i.e.*, $\sigma_n = n$). Let k be the number of left-to-right maxima of σ (including n).*

There is a bijective correspondence between the preimages of σ under $\mathbf{B}$ and the subsets of the $k-1$ left-to-right maxima of σ different from n.

More precisely, this correspondence works as follows. For any set $S = \{s_1 < \cdots < s_j\}$ of $j \leq k-1$ left-to-right maxima of σ different from n, writing $\sigma = B_0 s_1 B_1 s_2 B_2 \dots s_j B_j n$ (for the B_i possibly empty sequences of integers, which contain the $k-j$ left-to-right maxima not in S

and the elements of σ which are not left-to-right maxima), the corresponding preimage of σ is $s_1 B_0 s_2 B_1 \dots s_j B_{j-1} n B_j$.

Proof. From the alternative description of the procedure computing $\mathbf{B}^{-1}(\sigma)$, we have seen that the elements which are distinguished at some point are exactly the left-to-right maxima of the preimage produced. In addition, by definition of this procedure, the distinguished elements form a subset containing n of the set of left-to-right maxima of σ. This proves the claimed bijective correspondence.

To describe precisely the preimage corresponding to a subset S, it is enough to note that every distinguished element moves to the left until a new distinguished element is chosen, leaving all other elements unchanged. □

This allows to count the preimages of any given permutation, in total or by the number of their left-to-right maxima.

Corollary 4.1.4. *Let σ be a permutation of size n ending with its maximum, and with k left-to-right maxima.*

The cardinality of $\mathbf{B}^{-1}(\sigma)$ is 2^{k-1}, and for any $1 \leq j \leq k$, the number of preimages of σ with j left-to right maxima is $\binom{k-1}{j-1}$.

Proof. The cardinality of $\mathbf{B}^{-1}(\sigma)$ follows immediately from Corollary 4.1.3. By Corollary 4.1.3, a preimage of σ with j left-to-right maxima corresponds bijectively to a subset containing $j-1$ elements of the set of left-to-right maxima of σ different from n. We have $\binom{k-1}{j-1}$ different ways to select these subsets, thus proving the lemma. □

Second, we can characterize the permutations having a given number of preimages.

Corollary 4.1.5. *For any $k \geq 1$, the permutations having exactly 2^{k-1} preimages under $\mathbf{B}$ are those ending with their maximum and having k left-to-right maxima in total.*

In particular, there are $\left[{n-1 \atop k-1}\right]$ permutations of size n having 2^{k-1} preimages under $\mathbf{B}$, where $\left[{n \atop k}\right]$ are the (unsigned) Stirling numbers of the first kind.

Proof. The first statement follows immediately from Corollary 4.1.3. The second follows from the well-known fact that Stirling numbers of the first kind enumerate permutations according to their size and number of cycles, using the classical Foata bijection which maps permutations of size n with k cycles to permutations of size n with k left-to-right maxima. □

4.2 The trees of iterated preimages

Remember that, for any n, S_n denote the set of permutations of size n and $id_n = 12 \dots n$ the identity permutation of size n. We start by defining $T(\pi)$ for any permutation π, and $T_n = T(id_n)$.

Definition 4.2.1. Let T_n be the tree whose nodes are the permutations of S_n such that:

- T_n has root id_n;
- for every $\sigma, \tau \in S_n$, τ is a child of σ if and only if $\mathbf{B}(\tau) = \sigma$ and $\sigma \neq \tau$. Note that the situation $\mathbf{B}(\tau) = \sigma$ and $\sigma = \tau$ occurs only when $\sigma = \tau = id_n$.

Also, given a permutation $\pi \in S_n$, we define the *tree of its preimages* $T(\pi)$ as the subtree of T_n with root π.

For example, Fig. 4.1 shows the tree T_4.

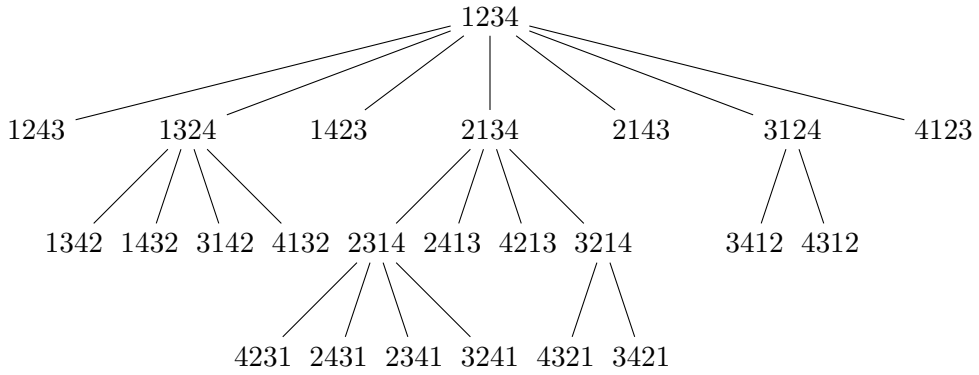


Figure 4.1: The tree T_4.

4.2.1 Isomorphisms between subtrees

For any permutation π, $T(\pi)$ describes all possible preimages of π under repeated applications of $\mathbf{B}^{-1}$. From Section 4.1, we can see that the "shape" of this tree depends on π only through the location of its left-to-right maxima. More precisely, the following lemma holds.

Lemma 4.2.2. *Let π and τ be two permutations of the same size. If π and τ have their left-to-right maxima in the same positions, then $T(\pi)$ and $T(\tau)$ are isomorphic.*

Proof. Let π, τ be permutations of the same size n which have their left-to-right maxima in the same positions, and let h and k be the depth of $T(\pi)$ and $T(\tau)$, respectively (the *depth* of a tree being defined as the maximum depth of its nodes). Without loss of generality, we can assume that $h \geq k$. The proof is by induction on h. If $h = 0$, then $h = k = 0$, and so $T(\pi)$ and $T(\tau)$ both consist of a single node, and our claim trivially holds.

Now suppose that $h > 0$. Unless $\pi = id_n$, by definition, the nodes of $T(\pi)$ at depth 1 are the preimages of π under $\mathbf{B}$. By Corollary 4.1.3, these preimages are in bijection with all possible subsets of left-to-right maxima of π which do not contain n. Instead of identifying a subset of left-to-right maxima of π by the *values* of the left-to-right maxima it contains, we can identify it by the *positions* of the left-to-right maxima it contains. Since π and τ have their left-to-right maxima in the same positions, it follows from Corollary 4.1.3 that there is a bijection between the preimages of π under $\mathbf{B}$ and the preimages of τ under $\mathbf{B}$. In addition, for every $\sigma \in \mathbf{B}^{-1}(\pi)$, the corresponding $\rho \in \mathbf{B}^{-1}(\tau)$ has its left-to-right maxima in the same positions as those of σ. Since $T(\sigma)$ and $T(\rho)$ have depth at most $h - 1$ and $k - 1$ respectively, we can apply the inductive hypothesis and obtain that $T(\sigma)$ and $T(\rho)$ are isomorphic. Summing up, we have that the children of π and τ are in a bijective correspondence, and the trees rooted at two children paired together by this bijection are isomorphic. Therefore $T(\pi)$ and $T(\tau)$ are isomorphic.

Finally, if π is the identity permutation of size n, and τ (of the same size) has its left-to-right maxima in the same positions as π, then necessarily $\tau = id_n$ as well, and in this case our claim trivially holds. □

As the next proposition shows, all possible shapes of the trees $T(\pi)$ can be found starting at depth 1 in T_n. To establish this proposition, we rely on the LTR-max block decomposition of π (defined in Section 2.1), which will be also essential in the description of $T(\pi)$ in the next subsection. We recall it below both for convenience and to fix the notation for the indices and the length of the blocks.

Definition 4.2.3. Given π, we decompose it as $\pi = M_1 P_1 \cdots M_{\ell-1} P_{\ell-1} M_\ell$, where the M_i's are all the maximal sequences of consecutive left-to-right maxima of π (called *blocks*), and the

P_i's collect all the remaining elements. In particular, all the P_i's are nonempty, and M_i is nonempty for all i except possibly for $i = \ell$. Moreover, $m_i = |M_i|$ denotes the length of M_i, and analogously $p_i = |P_i|$ denotes the length of P_i, for all i.

Notice that $m_1 + \cdots + m_\ell = k$ is the total number of left-to-right maxima of π.

Proposition 4.2.4. *For every permutation $\pi \in S_n$, $\pi \neq id_n$, there exists a child τ of id_n in T_n such that $T(\pi)$ and $T(\tau)$ are isomorphic.*

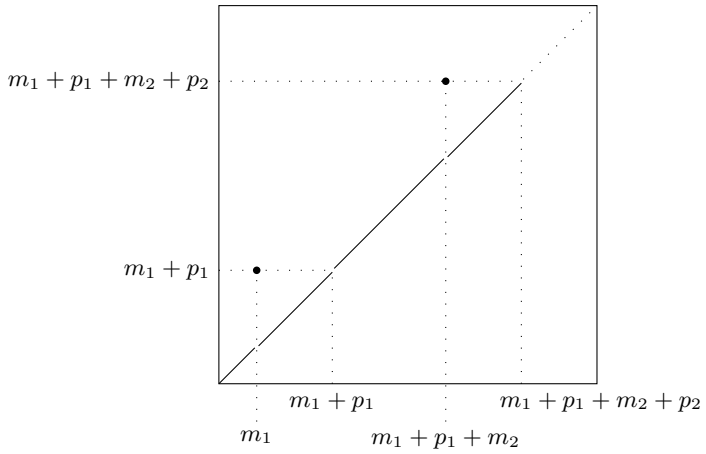


Figure 4.2: The permutation τ described in the proof of Proposition 4.2.4.

Proof. Let $\pi = M_1 P_1 \cdots M_{\ell-1} P_{\ell-1} M_\ell \in S_n$. Define τ as the permutation in which the elements $m_1+p_1, m_1+p_1+m_2+p_2, \ldots, m_1+p_1+\cdots+m_{\ell-1}+p_{\ell-1}$, are in the positions $m_1, m_1+p_1+m_2, \ldots, m_1+p_1+m_2+\cdots+m_{\ell-2}+p_{\ell-2}+m_{\ell-1}$, respectively, while all the other elements are in increasing order. We can see an example of this construction in Fig. 4.2. Therefore, τ and π have their left-to-right maxima in the same positions, thus by Lemma 4.2.2 the trees $T(\pi)$ and $T(\tau)$ are isomorphic.

We are left with showing that τ is a child of id_n in T_n. Since $p_1 \neq 0$, then $\tau \neq id_n$, so we only need to check that $\mathbf{B}(\tau) = id_n$. Observe that the elements $m_1 + p_1 + \cdots + m_i + p_i$ are the last left-to-right maxima of their blocks in τ, for every $i = 1, \ldots, \ell - 1$, and all the elements before the positions $m_1 + p_1 + \cdots + m_i + p_i$ are smaller than or equal to $m_1 + p_1 + \cdots + m_i + p_i$. Therefore $\mathbf{B}(\tau) = id_n$, because the $m_1 + p_1 + \cdots + m_i + p_i$'s are the only elements moved by `Bubblesort`, and they are moved to their correct position. □

4.2.2 The skeleton of the tree of preimages

Here we describe how the "shape" of any tree $T(\pi)$ is completely determined by a small piece of information about π, which we encapsulate in its *label*.

Definition 4.2.5. The *label* of a permutation σ is the pair (k, m_ℓ), where k and m_ℓ are defined as in Definition 4.2.3.[1] The *skeleton* of a tree $T(\pi)$ is obtained from $T(\pi)$ by replacing each permutation at a node with its label. Fig. 4.3 shows the skeleton of the tree $T(2134)$, and can be compared with the subtree $T(2134)$ of $T(1234)$ in Fig. 4.1.

[1] In particular, by definition of k and m_ℓ, the first component of a label is always at least as large as the second, with equality only in the case of the identity permutations.

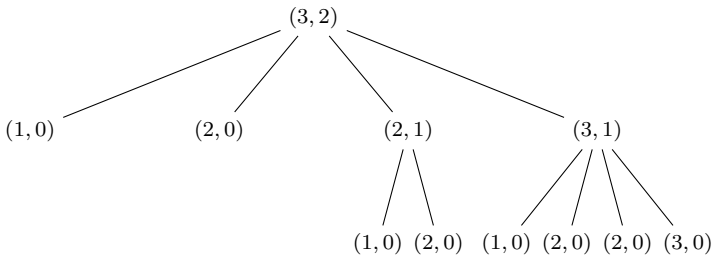


Figure 4.3: The skeleton of the tree $T(2134)$.

Remark 4.2.6. Since they have their left-to-right maxima in the same positions, the permutations π and τ of Proposition 4.2.4 have the same label. (It can also be observed that the trees $T(\pi)$ and $T(\tau)$ have the same skeleton. This follows by recursively applying Corollary 4.1.3, as in the proof of Lemma 4.2.2.)

Given a permutation π, we can determine the skeleton of $T(\pi)$ using only the pair (k, m_ℓ). Specifically, it is the tree with root labeled by (k, m_ℓ), and whose children (and recursively, descendants) are obtained as described in the next proposition.

Proposition 4.2.7. *Let $\pi \in S_n$ with label (k, m_ℓ). Let T be the skeleton of $T(\pi)$. Then the root of T has label (k, m_ℓ) and its children have the following labels:*

- *for every $h = 0, \dots, m_\ell - 2$:*
  - *for every $i = 1, \dots, k - 1 - h$, there are $\binom{k-2-h}{i-1}$ children with label $(k - i, h)$;*
- *if $\pi \neq id_n$, we also have the case corresponding to $h = m_\ell - 1$:*
  - *for every $i = 0, \dots, k - m_\ell$, there are $\binom{k-m_\ell}{i}$ children with label $(k - i, m_\ell - 1) = (k - i, h)$.*

Proof. We want to find the number of preimages of π with any given label. If $m_\ell = 0$, then π does not end with its maximum, hence it has no preimage. Thus the root of T has no children, and our claim vacuously holds.

Suppose that $m_\ell > 0$. This means that $\pi = \pi_1 \cdots \pi_{n-m_\ell}(n - m_\ell + 1) \cdots n$. We can apply the procedure described in Section 4.1 to find the preimages of π. From this procedure we can see that, if $\pi \neq id_n$, then its preimages can only have labels (k', h) with $k' \leq k$ and $h < m_\ell$, which corresponds to the labels listed in the above statement.

If instead $\pi = id_n$, then it has label (n, n) and its preimages can only have labels (k', h), with $k' \leq n$ and $h \leq n$, $h \neq n - 1$. Indeed, we cannot obtain a preimage of id_n with $h = n - 1$, because that would mean that only the element 1 is not part of the last sequence of left-to-right maxima, which is impossible. Instead we can have $h = n$, but only by leaving id_n unchanged.

To obtain a permutation with label $(k - i, h)$ with $h < m_\ell - 1$, referring to the procedure of Section 4.1, we are forced to leave unchanged (i.e. not to swap) all the elements from n down to $n - h + 1$, then to swap $n - h$ with $n - h - 1$. We are allowed to do so, because they are all left-to-right maxima, since $h < m_\ell - 1$.

After these steps, we obtain $\pi'(n-h-1)(n-h+1)(n-h+2)\cdots n$, with[2] $\pi' = \pi_1 \cdots \pi_{n-h-2}(n-h)$. Note that π' has $k - h - 1$ left-to-right maxima. Moreover, there is a bijection between the preimages of π' and the preimages of π ending with the suffix $(n-h-1)(n-h+1)(n-h+2)\cdots n$,

[2]Note that π' is not a permutation, but just a sequence of distinct integers. However, as we have seen in Lemma 2.4.1, it still makes sense to consider **B** on such sequences.

which consists of just appending the suffix $(n-h-1)(n-h+1)(n-h+2)$. Under this bijection, if a preimage of π' has $k-i-h$ left-to-right maxima (for some i such that $1 \leq i \leq k-1-h$), then the corresponding preimage of π has label $(k-i, h)$.

From Corollary 4.1.4, the number of preimages of π' with $k-i-h$ left-to-right maxima is $\binom{k-h-2}{k-i-h-1} = \binom{k-h-2}{i-1}$, for every $h = 0, \ldots, m_\ell - 2$ and $i = 1, \ldots, k-1-h$. Exploiting the above bijection, this prove the first item of our proposition.

Consider now the case $h = m_\ell - 1$, $\pi \neq id_n$. Then, applying the procedure of Section 4.1, we are forced to leave unchanged all the elements from n down to $n - m_\ell + 2$, then swap $n - m_\ell + 1$ with π_{n-m_ℓ}. Note that π_{n-m_ℓ} is an element of π which is not a left-to-right maximum, so if we define $\pi' = \pi_1 \cdots \pi_{n-m_\ell-1}(n - m_\ell + 1)$, we have that π' has $k - m_\ell + 1 = k - h$ left-to-right maxima. An argument analogous to the one we have used for the case $h < m_\ell - 1$ shows that the number of preimages of π with label $(k-i, m_\ell - 1)$ are $\binom{k-m_\ell}{k-i-m_\ell} = \binom{k-m_\ell}{i}$.

Finally, note that, if $\pi = id_n$, then there is an additional preimage, which is id_n, with label (n, n). However it does not correspond to a child of id_n, because Definition 4.2.1 prevents a permutation from being a child of itself. □

Corollary 4.2.8. *Let π be a permutation of size n with label (k, m_ℓ) such that $\pi \neq id_n$. Then $T(\pi)$ has depth m_ℓ. In addition, for every $n \geq 1$, T_n has depth $n-1$.*

Proof. We prove the statement by induction on m_ℓ. If $m_\ell = 0$, then $T(\pi)$ consists only of the root, and so has depth 0, as required. If $m_\ell \geq 1$, then by Proposition 4.2.7 the root of $T(\pi)$ has children whose labels are of the form (k', h), for every $0 \leq h \leq m_\ell - 1$ and some k'. By induction hypothesis, the subtree rooted at each child with label (k', h) has depth h. It follows that $T(\pi)$ has depth $1 + (m_\ell - 1) = m_\ell$, since the maximum value of h is $m_\ell - 1$.

We now consider T_n. If $n = 1$ then T_n consists of a single node and the statement is true. Otherwise, if $n > 1$, then (again by Proposition 4.2.7) the root of T_n has children with labels (k', h) for every $0 \leq h \leq n-2$ and some k'. Since these children are not the identity permutation, we can apply the first part of this corollary to them. We obtain that each child with label (h, k') is the root of a subtree of depth h. Since the maximum value of h is $n-2$, T_n has depth $1 + (n-2) = n-1$. □

Corollary 4.2.9. *For any given node $\pi \neq id_n$ in T_n, either half of its children are leaves or all of its children are leaves.*

Proof. Let (k, m_ℓ) be the label of $\pi \neq id_n$. If $m_\ell = 0$, the statement is vacuously true, because π is a leaf. Otherwise, if $m_\ell = 1$, then by Proposition 4.2.7 we have that all of its children have label $(k', 0)$ for some k', and so they are all leaves.

Finally, if $m_\ell > 1$, then (again by Proposition 4.2.7) the number of children of π which are leaves, that is with label $(k', 0)$ for some k', is

$$\sum_{i=1}^{k-1} \binom{k-2}{i-1} = \sum_{j=0}^{k-2} \binom{k-2}{j} = 2^{k-2}.$$

By Corollary 4.1.4, π has 2^{k-1} preimages, or equivalently it has 2^{k-1} children (since $\pi \neq id_n$). This proves our statement. □

Notice that, for $\pi = id_n$ (with label (n, n)), it is still true that it has 2^{n-2} children which are leaves and 2^{n-1} preimages, but the total number of children is now $2^{n-1} - 1$ (id_n being a preimage of itself, but not one of its children).

4.2.3 The inverse problem: deciding if a tree is isomorphic to $T(\pi)$ for some π

We now consider the following problem: given a (rooted unlabeled) tree T, does there exist a permutation π such that T coincides with the (unlabeled) skeleton of $T(\pi)$? This problem can be easily solved thanks to our previous results on the labels of the nodes of $T(\pi)$. If T consists of just a leaf, then of course T is isomorphic to $T(\pi)$ for some π (just take $\pi = 1$ or any permutation of size at least 2 not ending with its maximum). So, assume that T has depth at least 1.

The first step is to determine the label of a candidate π. By Corollary 4.1.4, we can immediately say that, if the root of T has neither 2^{k-1} nor $2^{k-1} - 1$ children, for some $k > 0$, then T cannot be the (unlabeled) skeleton of any $T(\pi)$.

If the root of T has $2^{k-1} - 1$ children, then the only candidate permutation is $\pi = id_k$. In particular, by Corollary 4.2.8, it is necessary that T has depth $k - 1$. We can then use Proposition 4.2.7 to check if the number of children of every node of T matches with the numbers given in that proposition.

Otherwise, suppose that the number of children of the root of T is 2^{k-1} for some k. Let m_ℓ be the depth of T. By Corollary 4.1.4 and Corollary 4.2.8, we know that a permutation needs to have label (k, m_ℓ) for $T(\pi)$ to have 2^{k-1} children of its root and depth m_ℓ. Therefore we can use Proposition 4.2.7 to check if the number of children of every node of T matches with the numbers given in Proposition 4.2.7 for a permutation $\pi \neq id_n$ with label (k, m_ℓ).

The next proposition summarizes the above discussion.

Proposition 4.2.10. *Let T be a (rooted unlabeled) tree, let i be the number of children of the root of T, and m_ℓ be the depth of T. Then*

- *if $i = 2^{m_\ell} - 1$, then T may only coincide with the (unlabeled) skeleton of $T_{m_\ell+1}$;*
- *if there exists a positive integer k such that $i = 2^{k-1}$, then T may only coincide with the (unlabeled) skeleton of a permutation π with label (k, m_ℓ);*
- *in all the other cases, T does not coincide with the unlabeled skeleton of any permutation.*

4.3 Heights of nodes and leaves in T_n

4.3.1 Nodes

Recall that the height of a node in a rooted tree is the number of edges on the path connecting that node to the root. The height of a node of the tree T_n corresponds to the number of passes of `Bubblesort` needed to sort the permutation at this node. Therefore, we can refer to [3, Prop. 17] to find information on the number of nodes of T_n.

Proposition 4.3.1 ([3])**.** *The set of permutations of size n sorted by at most k passes of `Bubblesort` is the set $\mathrm{Av}_n(\Gamma_{k+2})$, where Γ_k is the set of all permutations of size k whose final element is 1*[3]*. As a consequence, setting $\varphi_n^{(k)} = |\mathrm{Av}_n(\Gamma_k)|$, the number of nodes at height at most k in T_n is given by $\varphi_n^{(k+2)} = (k+1)^{n-k-1}(k+1)!$*

We can thus immediately deduce the number of nodes at a given height in T_n.

[3] We warn the reader that we have made a slight change of notation with respect to [3] here; more specifically, our set Γ_k is Γ_{k-2} in [3].

n \ k	0	1	2	3	4	5
1	1					
2	1	1				
3	1	3	2			
4	1	7	10	6		
5	1	15	38	42	24	
6	1	31	130	222	216	120

Table 4.1: Number of nodes in T_n having height k.

Corollary 4.3.2. *The number $f_n^{(k)}$ of nodes at height k in T_n is given by*

$$f_n^{(k)} = \varphi_n^{(k+2)} - \varphi_n^{(k+1)} = (k+1)^{n-k-1}(k+1)! - k^{n-k}k! = k! \cdot ((k+1)^{n-k} - k^{n-k}).$$

The first lines of the infinite triangular matrix of the coefficients $f_n^{(k)}$ are given in Table 4.1. This is sequence A056151 in [22].

We notice that the elements on the diagonal of Table 4.1 are the factorial numbers, more specifically $f_n^{(n-1)} = (n-1)!$. Indeed, the set of permutations of size n needing the maximum number of passes of `Bubblesort` to be sorted (that is, $n-1$ passes) is the set of permutations of size n ending with 1, whose cardinality is clearly $(n-1)!$.

From the expression of $\varphi_n^{(k)}$ in Proposition 4.3.1, we can derive the asymptotic behavior of the average height of a node in T_n. This analysis is described in [16, Theorem 7.14] and follows easily from the asymptotic behavior of the Ramanujan P-function (see [16, Table 4.11] or [20, p. 119-120]), which we state in Lemma 4.3.3 below. We then reproduce the analysis of [16, Theorem 7.14], as a preparation for Proposition 4.3.9 below.

Lemma 4.3.3 ([16])**.** *The Ramanujan P-function, defined by $P(n) = \sum_{k=0}^{n-1} \frac{k!k^{n-k}}{n!}$, behaves asymptotically as $\sqrt{\frac{\pi n}{2}} + O(1)$.*

Proposition 4.3.4 ([16])**.** *The average height of a node in T_n is asymptotically equal to $n - \sqrt{\frac{\pi n}{2}} + O(1)$.*

Proof. The average height of a node in T_n is given by

$$H_n := \frac{1}{n!} \sum_{k=1}^{n-1} \text{number of nodes of height at least } k \text{ in } T_n,$$

each node at height k contributing indeed exactly k times to this sum. Writing the number of nodes of height at least k in T_n as the difference of $n!$ (the total number of nodes) and the number of nodes of height at most $k-1$ in T_n, we then compute

$$H_n = \frac{1}{n!} \sum_{k=1}^{n-1} (n! - \varphi_n^{(k+1)}) = \frac{1}{n!} \sum_{k=1}^{n-1} (n! - k^{n-k}k!) = (n-1) - \sum_{k=0}^{n-1} \frac{k^{n-k}k!}{n!} = (n-1) - P(n),$$

proving our claim. □

Recall the (obvious) fact that T_n contains $n!$ nodes. With Proposition 4.3.1 and Corollary 4.3.2, we have refined this counting according to the height of the nodes in T_n. We now

address the analogous problems in $T(\pi)$ for $\pi \neq id_n$. More precisely, given a permutation π (of size n) having label (k, m_ℓ), we determine an expression for the number of nodes of $T(\pi)$ (which does not depend on n). This expression is a summation formula in which each summand counts nodes in $T(\pi)$ of a prescribed height.

Lemma 4.3.5. *Let π and τ be two permutations having labels (k, m_ℓ) and $(k, m_\ell - 1)$, respectively, with $1 \leq m_\ell \leq k-1$. Then the tree obtained by removing the leaves at height m_ℓ in $T(\pi)$ is isomorphic to $T(\tau)$.*

Proof. Remember that, by Corollary 4.2.8, $T(\pi)$ has height m_ℓ and $T(\tau)$ has height $m_\ell - 1$. The proof is by induction on m_ℓ.

If $m_\ell = 1$, then $T(\tau)$ consists of the single node τ, while $T(\pi)$ has height 1, therefore the statement is true.

Now let $m_\ell \geq 2$, and suppose that the statement is true for $m_\ell - 1$. We will show that there is a bijective correspondence between the children of τ and the children of π such that the subtree rooted at a child of τ is isomorphic to the subtree rooted at the corresponding child of π, after removing the leaves at height m_ℓ (if any).

Proposition 4.2.7 allows us to determine the labels of the children of τ and π in $T(\tau)$ and $T(\pi)$, respectively. Specifically, τ and π have the same number of children with labels $(k-i, h)$, for every $h = 0, \ldots, m_\ell - 3$ and every $i = 0, \ldots, k-1-h$. Regarding the remaining children, we have that the number of children of τ labeled $(k-i, m_\ell - 2)$ is equal to the sum of the number of children of π labeled $(k-i, m_\ell - 2)$ and $(k-i, m_\ell - 1)$, for every $i = 0, \ldots, k - m_\ell + 1$. This induces the announced bijective correspondence between the children of τ in $T(\tau)$ and those of π in $T(\pi)$.

The children of τ and π with the same labels give isomorphic subtrees by Proposition 4.2.7. In addition, if this label is $(k-i, h)$ for some $h \leq m_\ell - 2$ (and some suitable i), then the subtrees contain no leaf at height m_ℓ in $T(\tau)$ or $T(\pi)$ (again by Corollary 4.2.8), ensuring our claim restricted to such children of π and τ.

Therefore, we are left with considering a child of π in $T(\pi)$ with label $(k-i, m_\ell - 1)$, to which corresponds a child of τ in $T(\tau)$ of label $(k-i, m_\ell - 2)$. We can apply the inductive hypothesis to such children of π and τ, thus obtaining that each subtree of $T(\tau)$ rooted at a child of τ with label $(k-i, m_\ell - 2)$ is isomorphic to a subtree of $T(\pi)$ rooted at a child of π with label $(k-i, m_\ell - 1)$ after removing the leaves at height $m_\ell - 1$ (in the subtree, *i.e.* at height m_ℓ in $T(\pi)$). This concludes the proof. □

Proposition 4.3.6. *For a permutation π having label (k, m_ℓ), different from an identity permutation, the number of nodes of the tree $T(\pi)$ of its preimages under $\mathbf{B}$ is*

$$N(k, m_\ell) = \sum_{j=0}^{m_\ell} j!(j+1)^{k-j}. \tag{4.1}$$

Moreover, each summand in Eq. (4.1) records the contribution of each level of $T(\pi)$. In other words, denoting with $N_j(k, m_\ell)$ the number of nodes at height j in $T(\pi)$, we have that $N_j(k, m_\ell) = j!(j+1)^{k-j}$.

Proof. In order to prove Eq. (4.1) we proceed by induction on m_ℓ. If $m_\ell = 0$, then π has no children, hence $N(k, 0) = 1$, which is consistent with Eq. (4.1).

Now suppose that Eq. (4.1) holds when the cardinality of the longest suffix of left-to-right maxima of π is strictly smaller than m_ℓ. Recalling Proposition 4.2.7, we have the following recursive expression for the number of nodes of $T(\pi)$:

$$
\begin{aligned}
N(k,m_\ell) &= 1+\sum_{h=0}^{m_\ell-2}\sum_{i=1}^{k-1-h}\binom{k-2-h}{i-1}N(k-i,h)+\sum_{i=0}^{k-m_\ell}\binom{k-m_\ell}{i}N(k-i,m_\ell-1)\\
&= 1+\sum_{h=0}^{m_\ell-2}\sum_{i=1}^{k-1-h}\binom{k-2-h}{i-1}\sum_{j=0}^{h}j!(j+1)^{k-i-j}+\sum_{i=0}^{k-m_\ell}\binom{k-m_\ell}{i}\sum_{j=0}^{m_\ell-1}j!(j+1)^{k-i-j}\\
&= 1+\sum_{h=0}^{m_\ell-2}\sum_{j=0}^{h}j!(j+1)^{k-j-1}\sum_{i=0}^{k-2-h}\binom{k-2-h}{i}(j+1)^{-i}\\
&\quad+\sum_{j=0}^{m_\ell-1}j!(j+1)^{k-j}\sum_{i=0}^{k-m_\ell}\binom{k-m_\ell}{i}(j+1)^{-i}\\
&= 1+\sum_{h=0}^{m_\ell-2}\sum_{j=0}^{h}j!(j+1)^{k-j-1}\left(1+\frac{1}{j+1}\right)^{k-2-h}+\sum_{j=0}^{m_\ell-1}j!(j+1)^{k-j}\left(1+\frac{1}{j+1}\right)^{k-m_\ell}\\
&= 1+\sum_{h=0}^{m_\ell-2}\sum_{j=0}^{h}j!(j+1)^{h-j+1}(j+2)^{k-2-h}+\sum_{j=0}^{m_\ell-1}j!(j+1)^{m_\ell-j}(j+2)^{k-m_\ell}.
\end{aligned}
$$

We then exchange the order of the two sums in the middle term of the last expression, use the geometric sum formula and we get:

$$
\begin{aligned}
N(k,m_\ell) &= 1+\sum_{j=0}^{m_\ell-2}j!(j+1)^{1-j}(j+2)^{k-2}\sum_{h=j}^{m_\ell-2}(j+1)^{h}(j+2)^{-h}+\sum_{j=0}^{m_\ell-1}j!(j+1)^{m_\ell-j}(j+2)^{k-m_\ell}\\
&= 1+\sum_{j=0}^{m_\ell-2}j!(j+1)(j+2)^{k-1-j}-\sum_{j=0}^{m_\ell-2}j!(j+1)^{m_\ell-j}(j+2)^{k-m_\ell}\\
&\quad+\sum_{j=0}^{m_\ell-1}j!(j+1)^{m_\ell-j}(j+2)^{k-m_\ell}\\
&= 1+\sum_{j=1}^{m_\ell-1}j!(j+1)^{k-j}+m_\ell!(m_\ell+1)^{k-m_\ell}=\sum_{j=0}^{m_\ell}j!(j+1)^{k-j},
\end{aligned}
$$

which gives Eq. (4.1).

Concerning the evaluation of $N_j(k,m_\ell)$, Lemma 4.3.5 implies that $N_j(k,m_\ell)=N_j(k,m_\ell-1)$, for all $j\le m_\ell-1$. By a repeated application of the lemma, we get that $N_j(k,m_\ell)=N_j(k,j)=N(k,j)-N(k,j-1)=j!(j+1)^{k-j}$, as desired. □

4.3.2 Leaves

In the tree T_n the leaves represent permutations that cannot be obtained as output of `Bubblesort`, *i.e.*, which do not belong to the image of $\mathbf{B}$. We saw just after Lemma 2.4.1 that these permutations are those not ending with their maximum, so that the total number of leaves in T_n is given by $(n-1)\cdot(n-1)!$.

Our next result is a closed formula for the number of leaves at height k in T_n, for any $k\le n-1$. To this aim, we make use of the so-called *ECO method*, illustrated in [4] and further developed and employed by many authors (see for instance [14]). We will not give a detailed description of this method here, since our application is simple enough to be outlined directly.

Recall that leaves in T_n correspond to permutations whose last element is not the maximum. Thus, denoting with $\mathrm{Av}_n^*(\Gamma_k)$ the set of permutations of size n avoiding Γ_k and such that their

last element is different from n, we are interested in the coefficients $\gamma_n^{(k)} = |\mathrm{Av}_n^*(\Gamma_k)|$, since $\gamma_n^{(k+2)}$ gives the number of leaves at height at most k in T_n.

Proposition 4.3.7. *For all n, k, we have*

$$\gamma_n^{(k)} = \begin{cases} (n-1)(n-1)! & n < k, \\ (k-2)(k-1)^{n-k}(k-1)! & n \geq k. \end{cases}$$

Proof. We consider the following general procedure to generate all permutations of size n. Given any permutation of size $n-1$, construct n different permutations of size n by adding a new rightmost element k, for any choice of k between 1 and n, and suitably rescaling the other elements (namely, all elements of the starting permutation which are greater than or equal to k are increased by 1, whereas all the remaining elements are left untouched). It is immediate to realize that, starting from the set of all permutations of size $n-1$, the above procedure generates exactly once every permutation of size n.

We now adapt the above construction to our setting. Every permutation of $\mathrm{Av}_n^*(\Gamma_k)$ can be obtained from a permutation of $\mathrm{Av}_{n-1}(\Gamma_k)$ by adding a suitable rightmost element. More specifically, we cannot add n (because we require that our permutation does not end with its maximum); moreover, if $n \geq k$, we cannot add any element between 1 and $n-k+1$ as well (otherwise we would create one of the forbidden patterns belonging to Γ_k). On the other hand, any of the remaining elements is allowed and generates a valid permutation. This means that every permutation in $\mathrm{Av}_{n-1}(\Gamma_k)$ generates $k-2$ distinct permutations of $\mathrm{Av}_n^*(\Gamma_k)$ and every permutation in $\mathrm{Av}_n^*(\Gamma_k)$ is obtained in this way exactly once. We thus deduce that, when $n \geq k$,

$$\gamma_n^{(k)} = (k-2)\varphi_{n-1}^{(k)} = (k-2)(k-1)^{n-k}(k-1)!,$$

whereas for $n < k$ we have that $\gamma_n^{(k)} = (n-1)(n-1)!$, which concludes the proof. □

Corollary 4.3.8. *The number $g_n^{(k)}$ of leaves of T_n at height k is given by*

$$g_n^{(k)} = k!(k(k+1)^{n-k-1} - (k-1)k^{n-k-1}).$$

Proof. Just observe that $g_n^{(k)} = \gamma_n^{(k+2)} - \gamma_n^{(k+1)}$ and that the maximum height of a node of T_n is $n-1$, so we are only interested in the case $n \geq k+1$ of the previous proposition. □

As in the case of nodes, Proposition 4.3.7 allows us to derive the asymptotic behavior of the average height of a leaf in T_n.

Proposition 4.3.9. *The average height of a leaf in T_n is asymptotically equal to $n - \sqrt{\frac{\pi n}{2}} + O(1)$.*

Proof. As in the proof of Proposition 4.3.7, we have that the average height of a leaf in T_n is

$$\begin{aligned} G_n &= \frac{1}{(n-1)(n-1)!} \sum_{k=1}^{n-1} \text{number of leaves of height at least } k \text{ in } T_n \\ &= \frac{1}{(n-1)(n-1)!} \sum_{k=1}^{n-1} \left((n-1)(n-1)! - \gamma_n^{(k+1)}\right) = (n-1) - \sum_{k=1}^{n-1} \frac{(k-1)k^{n-k-1}k!}{(n-1)(n-1)!} \\ &= (n-1) - \frac{n}{n-1} \sum_{k=1}^{n-1} \frac{k^{n-k}k!}{n!} + \frac{1}{n-1} \sum_{k=1}^{n-1} \frac{k^{n-1-k}k!}{(n-1)!} \\ &= (n-1) - \frac{nP(n)}{n-1} + \frac{P(n-1)+1}{n-1}, \end{aligned}$$

and the asymptotic behavior of the Ramanujan P-function yields the announced result. □

In the same manner as we have done for the nodes, we now address the analogous problem of counting the leaves in $T(\pi)$, for $\pi \neq id_n$. More precisely, given a permutation π (of size n) having label (k, m_ℓ), we determine an expression for the number of leaves of $T(\pi)$ (which does not depend on n but only on the label (k, m_ℓ)). This expression is a summation formula in which each summand counts the leaves of a prescribed height in $T(\pi)$.

Proposition 4.3.10. *For a permutation π having label (k, m_ℓ), different from an identity permutation, the number of leaves of the tree $T(\pi)$ of its preimages under* $\mathbf{B}$ *is*

$$L(k, m_\ell) = \sum_{j=1}^{m_\ell - 1} j!j(j+1)^{k-j-1} + m_\ell!(m_\ell + 1)^{k-m_\ell}. \tag{4.2}$$

Moreover, each summand in Eq. (4.2) *records the contribution of each level of $T(\pi)$. In other words, denoting with $L_j(k, m_\ell)$ the number of leaves at height j in $T(\pi)$, we have that $L_j(k, m_\ell) = j!j(j+1)^{k-j-1}$ for $j < m_\ell$, and $L_{m_\ell}(k, m_\ell) = m_\ell!(m_\ell + 1)^{k-m_\ell}$.*

Proof. The proof of Eq. (4.2) is by induction, following the exact same steps as the proof of Proposition 4.3.6. The recursive equation for the number of leaves in $T(\pi)$, which is needed in the inductive step of the proof, is again obtained from Proposition 4.2.7. It actually differs from the one for nodes in the proof of Proposition 4.3.6 only by the initial term 1 (accounting for the root node); namely for $m_\ell \geq 1$, we have

$$L(k, m_\ell) = \sum_{h=0}^{m_\ell - 2} \sum_{i=1}^{k-1-h} \binom{k-2-h}{i-1} L(k-i, h) + \sum_{i=0}^{k-m_\ell} \binom{k-m_\ell}{i} L(k-i, m_\ell - 1),$$

and for $m_\ell = 0$ it holds that $L(k, 0) = 1$. From there, the same steps of computations as in the proof of Proposition 4.3.6 (followed by additional elementary simplifications) yield, for $m_\ell \geq 1$:

$$L(k, m_\ell) = \sum_{j=1}^{m_\ell - 1} j!j(j+1)^{k-j-1} + m_\ell!(m_\ell + 1)^{k-m_\ell},$$

as claimed.

We now move to the claimed expression for $L_j(k, m_\ell)$. We shall first establish it for $j = m_\ell$, then for $j = m_\ell - 1$, and then for smaller j iterating the argument.

We first note that all the nodes of $T(\pi)$ at height m_ℓ are leaves (since m_ℓ is the height of this tree). Using Proposition 4.3.6, we therefore have $L_{m_\ell}(k, m_\ell) = N_{m_\ell}(k, m_\ell) = m_\ell!(m_\ell + 1)^{k-m_\ell}$. As a consequence, the total number of leaves having height at most $m_\ell - 1$ in $T(\pi)$ is $\sum_{j=1}^{m_\ell - 1} j!j(j+1)^{k-j-1}$.

Next, we claim that the number of leaves having height at most $m_\ell - 2$ in $T(\pi)$ is $\sum_{j=1}^{m_\ell - 2} j!j(j+1)^{k-j-1}$. From this claim, the announced formula $L_{m_\ell - 1}(k, m_\ell) = (m_\ell - 1)!(m_\ell - 1)m_\ell^{k-m_\ell}$ immediately follows by taking the difference.

To prove our claim, we use Lemma 4.3.5. This lemma indeed implies that $L_j(k, m_\ell) = L_j(k, m_\ell - 1)$, for all $j \leq m_\ell - 2$. This shows that the number of leaves having height at most $m_\ell - 2$ in $T(\pi)$ is the same as the number of leaves having height at most $m_\ell - 2$ in $T(\sigma)$ for σ a permutation with label $(k, m_\ell - 1)$. The latter is equal to $L(k, m_\ell - 1) - L_{m_\ell - 1}(k, m_\ell - 1)$, hence equal to $\sum_{j=1}^{m_\ell - 2} j!j(j+1)^{k-j-1}$ as established earlier, thus proving our claim.

We are now left with showing that $L_h(k, m_\ell) = h!h(h+1)^{k-h-1}$ for $h \leq m_\ell - 2$. We proceed iteratively, for decreasing values of h. At each step, the reasoning is similar to the above case for $h = m_\ell - 1$. We first use Lemma 4.3.5 (several times, as in the proof of Proposition 4.3.6) to

argue that the number of leaves having height at most $h-1$ in $T(\pi)$ is the same as the number of leaves having height at most $h-1$ in $T(\sigma)$ for σ a permutation with label (k,h). This number is $\sum_{j=1}^{h-1} j!j(j+1)^{k-j-1}$. Then, $L_h(k,m_\ell)$ is the difference between $L(k,m_\ell)-\sum_{h+1\leq j\leq m_\ell} L_j(k,m_\ell)$ and the above quantity. The result follows from the formulas previously established for $L_j(k,m_\ell)$ for $j \geq h+1$. □

Remark 4.3.11. *Combining Propositions 4.3.6 and 4.3.10 tells us that, for π a permutation of label (k,m_ℓ), at height $j < m_\ell$ in $T(\pi)$, the ratio between the number of leaves and the number of nodes is $\frac{j}{j+1}$ (equivalently, the ratio between the number of internal nodes and the number of nodes is $\frac{1}{j+1}$).*

Chapter 5

Sorting with a popqueue

A popqueue is a container in which we can insert and extract elements in the same way as a queue, except that the extraction removes all elements in the popqueue, instead of only one. That is, the allowed operations are the following:

e: *enqueue*, insert the current element to the back of the popqueue;

p: *pop*, remove all the elements currently in the popqueue, from the front to the back, and send them into the output;

b: *bypass*, move the current element of the input permutation into the output.

This device can be used to sort permutations, and in fact we want to describe an algorithm that, using the allowed operations, sorts every sortable permutation. The following proposition describes the sortable permutations in terms of permutation patterns.

Proposition 5.0.1. *If a permutation $\pi \in S_n$ contains an occurrence of the patterns* 321 *or* 2413*, then it cannot be sorted using a popqueue.*

Proof. Let π be a permutation containing an occurence cba of the pattern 321, and suppose by contradiction that it can be sorted using the previous operations. Then c must enter the popqueue, because otherwise it would be output before a. For the same reason b have to enter the popqueue. But then c would exit the popqueue before b, which is a contradiction. Otherwise, if π contains an occurrence $bdac$ of the pattern 2413, then the elements b and d would have to enter the popqueue, but then they could only be output together, so the element c could not be output in between b and d. □

Clearly, this is a more restrictive condition than when sorting with a queue (see Section 2.3 and Chapter 3), since each popqueue operation can be mimicked by a standard queue.

5.1 Two optimal sorting algorithms

We now provide two different sorting algorithms which, although rather similar, have a sensibly different behavior when applied twice. To fix the notation for the algorithms, Q denotes the popqueue, $Front(Q)$ denotes the current first element of the popqueue and $Back(Q)$ denotes the current last element of the popqueue.

Algorithm 5.1.1. *Min*
input: a permutation $\pi = \pi_1 \cdots \pi_n$
output: a permutation $\mathbf{Min}(\pi)$
for $i = 1, \ldots, n$ *do:*

- *if $Front(Q)$ is the minimal non-output element (i.e. if $Front(Q)$ is smaller than all the unprocessed elements $\pi_i, \dots, \pi_n$), then empty the popqueue and enqueue π_i;*
- *else compare π_i, $Back(Q)$ and $Front(Q)$;*
  - *if $Back(Q) < \pi_i$, enqueue π;*
  - *otherwise, if $Front(Q) > \pi_i$, then output π_i;*
  - *else, empty the popqueue and then enqueue π_i;*

Finally, empty the popqueue.

The reason for the name `Min` comes from the fact that, whenever the first element of the popqueue is the next one to output (i.e., it is the minimum element not already in the output), the algorithm pours the whole content of the popqueue into the output. If this is not the case, the current element of the input is enqueued, provided that it is larger than the element in the back of the popqueue, otherwise the smallest between the current element and the element in the front of the popqueue is output (of course, in the latter case the whole content of the popqueue reaches the output).

`Min` is an optimal sorting algorithm, as it is shown in the next proposition.

Proposition 5.1.2. *Let $\pi \in S_n$. Then $\mathbf{Min}(\pi) \neq id_n$ if and only if $321 \leq \pi$ or $2413 \leq \pi$.*

Proof. By Proposition 5.0.1, we know that a permutation containing 321 or 2413 is not sortable.

On the other hand, suppose that $\mathbf{Min}(\pi) \neq id_n$, and consider the first configuration in which an incorrect element reaches the output (here "incorrect" means that it is not the minimum element not already in the output). If such an element comes to the output after executing a bypass operation, then we call it b and we observe that it must be smaller than the front element c of the popqueue. Moreover, there must be an element $a < b$ not yet in the output, otherwise b would be the correct element to output. Since (the content of) the popqueue is increasing by construction, then a must be in the input, hence π contains an occurrence cba of the pattern 321.

Otherwise, if the first incorrect element goes through the popqueue and reaches the output after executing a pop operation, then we have two possibilities. In the first case, the first element b of the popqueue is the smallest element not yet in the output, but the sorting procedure fails because there are elements d inside the popqueue and c in the input such that $d > c$.

In such a case, the last element which has reached the output is $b-1$, and this necessarily happened after a bypass operation when b and d were already in the popqueue (otherwise b would be already in the output). Therefore the elements b, d, $b-1$, c form an occurrence of the pattern 2413 in π. In the second case, if the first element of the popqueue is not the minimum element a not yet in the output, then a must still be in the input, and the last element c of the popqueue must be larger than the current element b of the input (which is in turn larger than a). Therefore π contains an occurrence cba of the pattern 321. □

The second algorithm we describe is `Cons`.

Algorithm 5.1.3. *`Cons`*
input: a permutation $\pi = \pi_1 \cdots \pi_n$
output: a permutation $\mathbf{Cons}(\pi)$
for $i = 1, \dots, n$ do:

- *if $\pi_i = Back(Q) + 1$, then enqueue π_i;*
- *else, compare π_i and $Front(Q)$;*

– *if* $Front(Q) > \pi_i$*, then output* π_i*;*
– *else, empty the popqueue and then enqueue* π_i*;*

Finally, empty the popqueue.

The name `Cons` comes from the first instruction, that forces the content of the popqueue to consists of consecutive elements during the whole execution.

As it happened for `Min`, also `Cons` is an optimal sorting algorithm.

Proposition 5.1.4. *Let* $\pi \in S_n$*. Then* $\mathbf{Cons}(\pi) \neq id_n$ *if and only if* $321 \leq \pi$ *or* $2413 \leq \pi$*.*

Proof. We already know, by Proposition 5.0.1, that a permutation containing either the pattern 321 or the pattern 2413 is not sortable. On the other hand, suppose that $\mathbf{Cons}(\pi) \neq id_n$, and consider the first configuration in which an "incorrect" element reaches the output. If the incorrect element b comes directly from the input, by performing a bypass operation, then an argument completely analogous to that used in Proposition 5.1.2 shows that π contains the pattern 321.

Otherwise, if the first incorrect element goes to the output after a pop operation, then let b be the last element of the popqueue, $d > b+1$ be the current element of the input and a be the smallest element not yet in the output. Then $b+1$ and a must still be in the input, because Q consists of consecutive elements and no incorrect element reached the output yet. So either π contains the occurrence $bda(b+1)$ of the pattern 2413, or π contains the occurrence $d(b+1)a$ of the pattern 321. □

We thus have two different optimal sorting algorithms, which follow distinct heuristics to sort π. Although the set of sortable permutations is the same, the output of `Min` and `Cons` is different in general. For instance, $\mathbf{Min}(2413) = 1243$ and $\mathbf{Cons}(2413) = 2134$. We remark that both heuristics are somehow reasonable in the framework of permutation sorting. First of all, neither `Min` nor `Cons` create new inversions. Moreover, if the first element of the popqueue is the minimum element among those not yet in the output, then it is reasonable to pop the popqueue immediately, as `Min` does, because the same thing would certainly be done before performing any bypass operation. On the other hand, if we were to allow non-consecutive elements in the popqueue, then the output would certainly not be the identity permutation, so it is reasonable to impose that the elements inside the popqueue are consecutive, as `Cons` does. Both ideas give us optimal algorithms, whose outputs differs only for nonsortable permutations, although the order in which the single operations are executed may be different even for sortable permutations. One could be tempted to see what happens when using both euristhics, by designing an algorithm that only allows consecutive elements inside the popqueue, and (at the same time) pops the popqueue whenever its front element is the next one to be output. However, the resulting algorithm would actually be equivalent to `Cons`, because the output would be the same, although operations would be performed in a different order in general, by prioritizing pop operations over enqueue ones.

As a consequence of the results of this section, we thus have that the set of sortable permutations of length n is $\mathrm{Av}_n(321, 2413)$. The enumeration of such a class is known [31]: this is the sequence of even-indexed Fibonacci numbers (sequence A001519 in [22]), whose first terms are $1, 1, 2, 5, 13, 34, \ldots$.

We close this section by proving some properties of `Cons` that will be useful later.

Lemma 5.1.5. *Let* $\pi = \pi_1 \cdots \pi_n \in S_n$*. When performing* `Cons` *on* π*,* π_i *enters the popqueue if and only if it is a left-to-right maximum. Moreover, the relative order of the non left-to-right maxima of* π *is preserved in* $\mathbf{Cons}(\pi)$*.*

Proof. The proof is by induction over the number of steps during the execution of `Cons`. The first element π_1 is always a left-to-right maximum, and is in fact enqueued.

Now suppose by induction that, at some point, all the elements currently enqueued are left-to-right maxima, and also all the left-to-right maxima in the output have been enqueued at some point, while the other elements have not. We will prove that this remains true after `Cons` has executed its next instruction. Let π_i be the current element of the input. If $\pi_i = Back(Q) + 1$, then π_i is greater than $Back(Q)$ and thus it is greater than all the previous left-to-right maxima, because they all entered the popqueue in increasing order by inductive hypothesis. Therefore π_i is a left-to-right maxima as well, and is in fact enqueued by `Cons`. If instead π_i is smaller than $Front(Q)$, then it is not a left-to-right maximum by definition, and it bypasses the popqueue. Finally, if $\pi_i > Back(Q) + 1$, then π_i is a left-to-right maximum by the same argument as for the case $\pi_i = Back(Q)$, and it is in fact enqueued (after having popped the popqueue).

Finally, if a and b are not left-to-right maxima of π, they do not enter the popqueue, hence they keep their relative order in the output. □

This lemma does not hold for `Min`. Indeed, although every left-to-right maximum does enter the popqueue, some non left-to-right maxima may also enter it (for example, the element 4 in 25143 enters the popqueue). Notice that `Queuesort` has the same properties described in the above lemma.

5.2 Two passes through a popqueue

As we have already observed, `Min` and `Cons` are both optimal sorting algorithms, even if they exploit different strategies. In particular, the images of *unsortable* permutations are different, and this is clearly relevant when we run each of the two algorithms multiple times.

In this section we investigate permutations which are sortable by running each of the previous algorithms twice. This has been done for `Stacksort` by West [29, 30], who found that the set of sortable permutations is not a class, nevertheless it can be described in terms of the avoidance of a pattern and a barred pattern.

We start by defining the sets $Sort_M^{(k)}$ and $Sort_C^{(k)}$ of permutations sortable by k applications of **Min** and **Cons**, respectively, that is $Sort_M^{(k)} = \{\pi \in S \mid \mathbf{Min}^k(\pi)) = id\}$ and $Sort_C^{(k)} = \{\pi \in S \mid \mathbf{Cons}^k(\pi)) = id\}$. In the previous section we have shown that $Sort_M^{(1)} = Sort_C^{(1)} = \mathrm{Av}(321, 2413)$.

It is interesting to notice that $Sort_M^{(2)}$ and $Sort_C^{(2)}$ are instead unrelated from the point of view of set containment. Indeed, consider the permutations 2431 and 35214. We have $\mathbf{Min}^2(2431) = 1243$ and $\mathbf{Cons}^2(2431) = 1234$, hence $2431 \in Sort_C^{(2)} \setminus Sort_M^{(2)}$. On the other hand, $\mathbf{Min}^2(35214) = 12345$ and $\mathbf{Cons}^2(35214) = 21345$, hence $35214 \in Sort_M^{(2)} \setminus Sort_C^{(2)}$. This shows that $\mathbf{Min}^2$ is capable of sorting permutations that $\mathbf{Cons}^2$ is unable to sort, and vice versa.

What we show next is that $Sort_M^{(2)}$ and $Sort_C^{(2)}$ are not just distinct sets; they also have different features from the point of view of pattern containment. More precisely, $Sort_C^{(2)}$ is a permutation class, whereas $Sort_M^{(2)}$ is not.

Proposition 5.2.1. *The set $Sort_M^{(2)}$ is not a permutation class.*

Proof. Given the permutation 241653, we have that $\mathbf{Min}^2(241653) = 123456$, however $\mathbf{Min}^2(2431) = 1243$, and clearly $2413 \leq 241653$. □

Proposition 5.2.2. *Let $\pi \in S_n$ such that π contains one of the following patterns:*

- 4321*;*
- 35241*;*
- 35214*;*
- 52413*;*
- 25413*;*
- 246153*;*
- 246135*;*
- 426153*;*
- 426135*.*

Then $\mathbf{Cons}^2(\pi) \neq id_n$*.*

Proof. By using the facts that

- the relative order of non left-to-right maxima of π is preserved in $\mathbf{Cons}(\pi)$ (see Lemma 5.1.5), and
- non-inversions in π remain non-inversions in $\mathbf{Cons}(\pi)$,

as well as the definition of `Cons`, which requires the elements in the popqueue to be consecutive at all times, it is not difficult to show that, if π contains one of the patterns 4321, 35241, 35214, 52413, 25413, 246153 or 246135, then $\mathbf{Cons}(\pi)$ contains the pattern 321 or the pattern 2413, therefore $\pi \notin Sort_C^{(2)}$.

We now consider the remaining patterns. Suppose that $dbfaec$ is an occurrence of the pattern 426153 in π. We start by observing that d precedes a in $\mathbf{Cons}(\pi)$ since, when f is processed, d must either be already in the output or exit the popqueue (because e is still in the input and $d < e < f$). Now, recalling the above facts, if d precedes b in $\mathbf{Cons}(\pi)$ then dba is an occurrence of 321 in $\mathbf{Cons}(\pi)$, otherwise $bdac$ is an occurrence of 2413 in $\mathbf{Cons}(\pi)$. Therefore in both cases $\pi \notin Sort_C^{(2)}$. An occurrence of the pattern 426135 can be dealt with in a similar way. □

Proposition 5.2.3. *Let $\pi \in S_n$ such that $\mathbf{Cons}^2(\pi) \neq id_n$. Then π contains one of the following patterns:*

- 4321*;*
- 35241*;*
- 35214*;*
- 52413*;*
- 25413*;*
- 246153*;*
- 246135*;*
- 426153*;*
- 426135*.*

Proof. Since $\mathbf{Cons}^2(\pi) \neq id_n$, we know that $321 \leq \mathbf{Cons}(\pi)$ or $2413 \leq \mathbf{Cons}(\pi)$. We consider the two cases separately.

If $321 \leq \mathbf{Cons}(\pi)$, then $321 \leq \pi$, since `Cons` does not produce new inversions. Let c, b, a be elements forming an occurrence of 321 in π. If c is not a left-to-right maximum, then of course π contains an occurrence of 4321. Otherwise, the left-to-right maximum c enters the popqueue, but it has to reach the output before b. This happens precisely when the current element e

of the input is larger than $Back(Q)+1$ (and so also $e > c$). Now observe that the element $d = Back(Q)+1$ cannot appear before c in π (since c is a left-to-right maximum), nor between c and e in π (otherwise, when e is the current element of the input, the last element of the popqueue could not be $d-1$). Therefore, all possible ways in which the elements a, b, c, d, e can occur in π are *cedba*, *cebda* e *cebad*, which form occurrences of the patterns 4321 (ignoring c), 35241 and 35214, respectively.

Otherwise, suppose that $\mathbf{Cons}(\pi)$ contains the pattern 2413, and the elements b, d, a, c form such an occurrence. Since `Cons` does not create new inversions, in the permutation π the element d appears before a and c, which are then not left-to-right maxima. Therefore π contains the elements d, a, c precisely in this order. Since b has to precede a in π, we are thus left with two possible configurations, which are *dbac* and *bdac*.

If π contains the subword *dbac*, then necessarily d is a left-to-right maximum of π (otherwise b could not reach the output before d), and so in particular d enters the popqueue at some point. Moreover, we need a to reach the output after both d and b. In order to understand how this can be possible, we focus on the instant in which d (and so the entire content of the popqueue) is popped. In such a situation, b has to be already in the output, d is in the popqueue, and the current element of the input, call it f, is larger than $Back(Q)+1$ (and so also than d). Now observe that $e = f-1$ is necessarily in the input at this moment: in fact it cannot be inside the popqueue (since the content of the popqueue is increasing) and it cannot be already in the output, otherwise it would appear before d in π and so d would not be a left-to-right maximum. Summing up, we have that π must contain one among the subwords $dbfeac$, $dbfaec$, $dbface$, corresponding to the patterns 25413 (ignoring d), 426153, 426135.

If instead π contains the subword *bdac*, then d may or may not be a left-to-right maximum. In case it is not, then there is an element $e > d$ preceding d in π, hence π contains either *bedac* or *ebdac*, corresponding to patterns 25413 or 52413. On the other hand, if d is a left-to-right maximum of π, we can use an argument completely analogous to the one of the previous case, this time getting that π has to contain one among the patterns 25413, 246153, 246135. □

The first terms of the sequence counting $Sort_C^{(2)}$ with respect to the length are $1, 1, 2, 6, 23, 99, 445, 2029, 9292, 42608, 195445, \ldots$, and do not appear in [22]. We also observe that analogous numbers can also be computed for $Sort_M^{(2)}$, and the first values are $1, 1, 2, 6, 22, 89, 379, 1660, 7380, 33113, 149059, \ldots$. From these data, it appears reasonable to conjecture that, for any $n \geq 3$, the number of permutations of length n sortable by $\mathbf{Min}^2$ is smaller than the number of permutations of length n sortable by $\mathbf{Cons}^2$.

5.3 Preimages

In this section we will study the preimages of a generic permutation under `Cons`. Note that we might also study the preimages under `Min`, and that they are (in general) different. We choose `Cons` since it has the property that all left-to-right maxima, and no other element, enter the popqueue during the sorting process. This property is shared with `Queuesort`, and is crucial to describe a simple procedure to find all the preimage of a generic permutation. For brevity, in the sequel we just call "preimages" the preimages under `Cons`.

We start by giving a different description of `Cons`, which highlights how it behaves on the left-to-right maxima of a permutation. We say that two elements of a permutation π are *adjacent* if their positions differ by one, and *consecutive* if their values differ by one.

Let $\pi = \pi_1 \cdots \pi_n \in S_n$. Mark π_1, and repeat the following steps until there are no marked elements:

- if there are no elements to the right of the (necessarily unique) block of adjacent marked elements, then unmark all marked elements;
- otherwise, compare the rightmost element μ of the block of adjacent marked elements with the element α to its right:
  - if $\mu > \alpha$, then swap α with the entire block of marked elements;
  - if $\mu = \alpha - 1$, then mark α;
  - if $\mu < \alpha - 1$, then mark α, and unmark all other elements of π.

As an example, we illustrate how this procedure operates on the permutation $\pi = 3241687$. The marked elements are indicated in **bold**.

$$\mathbf{3}241687 \to \mathbf{23}41687 \to 2\mathbf{34}1687 \to 21\mathbf{34}687 \to 2134\mathbf{6}87 \to 2134\mathbf{87} \to 2134\mathbf{678} \to 2134678$$

Notice that, since an element is marked if and only if it is greater than the previous marked element, then this procedure marks precisely the left-to-right maxima of the permutation. During the execution they are moved to the right until they reach the next left-to-right maximum; when this happens, they are glued together and continue moving to the right if and only if they are consecutive. This means that, at any point during the execution of this procedure, the marked elements are both consecutive and adjacent.

Notice that our marking mimics precisely what happens in the popqueue during the execution of `Cons`: the block of marked elements are the elements in the popqueue, the elements on its left are in the output and those on its right are still in the input.

From now on, when we refer to the execution of `Cons` on a permutation, we consider this alternative description.

Proposition 5.3.1. *Let σ be a permutation and set $\pi = \mathbf{Cons}(\sigma)$. Then the last element of π is n. Also, denoting with $\mathrm{LTR}_v(\tau)$ the set of (values of) the left-to-right maxima of a permutation τ, we have $\mathrm{LTR}_v(\sigma) \subseteq \mathrm{LTR}_v(\pi)$.*

Proof. During the execution of `Cons` on σ, n is going to be marked, because it is a left-to-right maximum, and it reaches the end of the permutation, since there are no greater elements than can block it.

Now, let μ be a left-to-right maximum of σ and suppose, by contradiction, that μ is not a left-to-right maximum of π. Thanks to the above considerations, μ is marked during the execution of `Cons` and moves to the right. Since it is not a left-to-right maximum of the output π, at some point it must be swapped with an element α greater than itself. Therefore `Cons` must reach a configuration where μ is part of a block of consecutive marked left-to-right maxima $\mu_1 \cdots \mu_k$, with μ_k greater than the element α on its right and $\mu = \mu_j$ for some j. This is impossible, since $\mu_j < \alpha < \mu_k$ and the elements $\mu_1, \ldots, \mu_k$ are consecutive, so α cannot be to the right of μ_k. Therefore μ is a left-to-right maximum of π. □

In view of the previous proposition, we can look for the preimages of a given permutation π by looking at all the subsets of $\mathrm{LTR}_v(\pi)$ and, for each of them, listing all the possible preimages with the prescribed set of left-to-right maxima. Notice that it is possible for a permutation to have no preimage (as well as many preimages) for a given subset. For example, there are no preimages of 213 whose left-to-right maxima are both 2 and 3, while 3421 and 3214 are both preimages of 2134 whose left-to-right maxima are 3 and 4. To describe preimages we introduce the notion of *mix* of two sequences.

Definition 5.3.2. Let $L = l_1 \cdots l_p$ and $A = a_1 \cdots a_r$ be two sequences of positive integers, so that the l_i's and a_j's are all different. We say that a sequence $m_1 \cdots m_{p+r}$ is a *mix* of L and A if it contains both L and A as subsequences, and $m_1 = l_1$. Define $Mix(L, A)$ as the set of all the mixes of L and A.

Essentially, the mix of two sequences L and A is a special shuffle of the two sequences, in which the first element belongs to L.

For example, the mix of 245 and 13 is the set $\{24513, 24153, 24135, 21453, 21435, 21345\}$.

We are now ready to describe the set $\mathbf{Cons}^{-1}(\pi)$ of the permutations whose output under `Cons` is π.

Proposition 5.3.3. *Let $\pi \in S_n$ be a permutation ending with n. Let $B \subseteq \mathrm{LTR}_v(\pi)$ such that $n \in B$.*

If B contains two consecutive integers that are not adjacent in π, then there are no preimages of π whose set of left-to-right maxima is B.

Otherwise, we can write π as $\pi = A_1 L_1 A_2 L_2 \cdots A_k L_k$, where the blocks L_i are maximal sequences of consecutive elements of B. The blocks A_i contain the remaining elements of π, and may be empty. Then, all the preimages of π whose set of left-to-right maxima is B are those of the form $\rho = \rho_1 \rho_2 \cdots \rho_k$, with $\rho_i \in Mix(L_i, A_i)$, for every $i = 1, \ldots, k$.

Proof. Let π be a permutation ending with n and $B \subseteq \mathrm{LTR}_v(\pi)$ such that $n \in B$.

Suppose that $\mu, \mu + 1 \in B$ are not adjacent in π. Suppose that there exists a preimage σ of π such that $\mathrm{LTR}_v(\sigma) = B$. We show that $\mathbf{Cons}(\sigma) \neq \pi$, thus obtaining a contradiction. During the execution of `Cons`, μ is marked and moves to the right. Since $\mu + 1$ is also a left-to-right maximum of σ, μ will reach it and $\mu + 1$ will be marked while maintaining μ marked. Therefore both μ and $\mu + 1$ will be unmarked at the same time, and will be adjacent in the output. Since μ and $\mu + 1$ are not adjacent in π, $\mathbf{Cons}(\sigma) \neq \pi$, which is a contradiction. Thus π has no preimages whose set of left-to-right maxima is B.

On the other hand, if B contains no consecutive elements which are not adjacent in π, let ρ, L_i and A_j be as in the statement of the proposition, and let λ_i be the first element of L_i, for every $i = 1, \ldots, k$. We want to prove that ρ is a preimage of π. We start by proving that $\mathbf{Cons}(\rho_i) = A_i L_i$. This is in fact true, because the elements of L_i are left-to-right maxima of π, and therefore greater than the elements of A_i. By definition of mix, the first element of ρ_i is λ_i, therefore the elements of A_i are not left-to-right maxima of ρ_i. Finally, the elements of L_i are consecutive, therefore `Cons` will move all of them together to the end of the string, thus obtaining $A_i L_i$. Now we just need to notice that the elements of different L_i are not consecutive to obtain that, during the execution of `Cons` on ρ, all the elements of each L_i will be unmarked before marking the elements of L_{i+1}, hence $\mathbf{Cons}(\rho) = A_1 L_1 \cdots A_k L_k = \pi$.

To conclude, we need to prove that every preimage σ of π such that $\mathrm{LTR}_v(\sigma) = B$ is of the form $\rho_1 \cdots \rho_k$, for some mixes ρ_i of L_i and A_i. First of all, the elements of σ that are not left-to-right maxima must be in the same relative order as they are in π, since `Cons` does not change their relative positions. So, reading such elements in σ from left to right, we get the string $A_1 \cdots A_k$. The same argument can be used for the left-to-right maxima of σ, thus getting that σ contains the subsequence $L_1 \cdots L_k$. The first left-to-right maximum is λ_1, and it must be the first element of σ. Furthermore, all the elements of A_1 must be to the left of λ_2 in σ. Indeed, since all the elements of L_1 are consecutive, and λ_2 is strictly larger than the last element of L_1 plus 1, then during the execution of `Cons` the elements of L_1 are marked and move to the right, but they will be unmarked as soon as λ_2 is reached. Therefore the elements of L_1 cannot overcome λ_2 and all the elements of A_1 must be to their right, otherwise $\mathbf{Cons}(\sigma)$ would not be the permutation $\pi = A_1 L_1 \cdots A_k L_k$. Finally, all the elements of A_2 must be to the right of λ_2, because otherwise the last element of L_1 would overcome them and the image of σ would

not have all the elements of A_2 to the right of L_1. Therefore σ can be written as $\rho_1 S$, where ρ_1 is a mix of L_1 and A_1.

Iterating this argument, we obtain that a mix ρ_2 of L_2 and A_2 must follow ρ_1 in σ, and so on, till we obtain $\sigma = \rho_1 \cdots \rho_k$. □

By Proposition 5.3.1, we know that the left-to-right maxima of any preimage of π must be a subset of $\mathrm{LTR}_v(\pi)$ containing n. Therefore the previous proposition describes all the preimages of a generic permutation.

For example, given $\pi = 3245617$, if we select the subset $B = \{4, 5, 7\}$, we have that the corresponding preimages are 4532761, 4352761, 4325761. If we look at all the subsets of $\mathrm{LTR}_v(\pi) = \{3, 4, 5, 6, 7\}$ (containing 7 and without consecutive elements not adjacent in π), we obtain the following permutations, which are all the preimages of π:

$\{7\}$: 7324561,

$\{3, 7\}$: 3724561,

$\{4, 7\}$: 4327561,

$\{5, 7\}$: 5324761,

$\{3, 5, 7\}$: 3524761,

$\{4, 5, 7\}$: 4532761, 4352761, 4325761.

5.3.1 Enumerative results

The correspondence between subsets of $\mathrm{LTR}_v(\pi)$ and preimages of π would be particularly simple for permutations with no consecutive left-to-right maxima. Indeed, in that case there would be only one preimage for every legal subset. Unfortunately this never happens, because a permutation has preimages if and only if it ends with its maximum n, hence $n-1$ is always a left-to-right maximum. Nonetheless we still have a simple case, for which we are able to count preimages.

Proposition 5.3.4. *Let $\pi \in S_n$ be a permutation ending with n, such that the only consecutive elements in $\mathrm{LTR}_v(\pi)$ are $n-1$ and n. Suppose that $n-1$ and n are not adjacent in π, and let $k = |\mathrm{LTR}_v(\pi)|$. Then $|\mathbf{Cons}^{-1}(\pi)| = 2^{k-2}$ and there is a bijection between $\mathbf{Cons}^{-1}(\pi)$ and the subsets of $\mathrm{LTR}_v(\pi)$ containing n but not $n-1$.*

Proof. Referring to Proposition 5.3.3, let B be a subset of $\mathrm{LTR}_v(\pi)$ containing n. If $n-1 \in B$, then there are no preimages of π whose left-to-right maxima are the elements of B, since $n-1$ and n are not adjacent in π. Otherwise, if $n-1 \notin B$, then we express $\pi = A_1 L_1 \cdots A_k L_k$ as in Proposition 5.3.3, noting that the blocks L_i consist of just one element (call it λ_i). Therefore there exists precisely one mix between L_i and A_i, for every i, which provides exactly one preimage. Thus, we have a correspondence mapping each subset of $\mathrm{LTR}_v(\pi)$ containing n and not containing $n-1$ into a preimage of π. Since the preimages are all different, we have a bijection between preimages of π and subsets of $\mathrm{LTR}_v(\pi)$ with the required properties. As a consequence, the number of preimages of π is the number of subsets of $\mathrm{LTR}_v(\pi)$ of size k, containing n and not containing $n-1$, which is 2^{k-2}. □

Notice that the case described in the previous proposition gives a result analogous to Corollary 4.1.4, which concerns the enumeration of the preimages of a permutation under `Bubblesort`. Specifically, the number of preimages under `Bubblesort` of a permutation π ending with n and

having k left-to-right maxima is 2^{k-1}. In fact, there is exactly one preimage for every subset of $\mathrm{LTR}_v(\pi)$ containing n, because (in the case of `Bubblesort`) the left-to-right maximum $n-1$ does not give troubles and can be selected. On the other hand, if there are consecutive elements in $\mathrm{LTR}_v(\pi)$ other than $n-1$ and n, the analogy fails and there does not seem to be any link between the two situations.

To conclude, we provide some results concerning permutations with a given number of preimages. Define $c_n^{(k)}$ as the number of permutations of length n which have exactly k preimages under `Cons`. We already noticed that a permutation has preimages if and only if it ends with its maximum, therefore $c_n^{(0)} = (n-1)(n-1)!$, for every $n \geq 1$, and $c_0^{(0)} = 0$. We record this result for future reference.

Proposition 5.3.5. *Let $\pi = \pi_1 \cdots \pi_n \in S_n$. Then π has no preimages under `Cons` if and only if $\pi_n \neq n$, for every $n \geq 1$.*

Therefore $c_n^{(0)} = (n-1)(n-1)!$, for every $n \geq 1$, and $c_0^{(0)} = 0$.

The next propositions deal with permutations having exactly 1, 2 or 3 preimages.

Proposition 5.3.6. *Let $\pi = \pi_1 \cdots \pi_n \in S_n$. Then π has exactly one preimage under `Cons` if and only if $\pi_n = n$ and $\pi_1 = n-1$, for every $n \geq 3$.*

Therefore $c_n^{(1)} = (n-2)!$ for every $n \geq 3$, and $c_0^{(1)} = 1$, $c_1^{(1)} = 1$, $c_2^{(1)} = 0$.

Proof. If $\pi_n = n$ and $\pi_1 = n-1$ then π has one preimage by Proposition 5.3.4.

On the other hand, suppose that $\pi_n \neq n$ or $\pi_1 \neq n-1$. If π does not end with n, then it has no preimages. Otherwise, we have that $\pi_n = n$ and $\pi_1 \neq n-1$ is a left-to-right maximum of π. Writing $\pi = \pi_1 M n$, we have that both $n\pi_1 M$ and $\pi_1 n M$ are preimages of π, so π has more than one preimage.

The formula for $c_n^{(1)}$ follows immediately, since there are no restrictions on the terms $\pi_2, \ldots, \pi_{n-1}$. □

The following lemma helps dealing with the case in which $n-1$ and n are adjacent in π.

Lemma 5.3.7. *Let π be a permutation of length $n > 2$ such that $\pi_{n-1} = n-1$ and $\pi_n = n$. Then π has at least four preimages.*

Proof. Since $n > 2$, then π_1, $n-1$ and n are all distinct left-to-right maxima of π. Writing π as $\pi_1 M (n-1) n$, we have that $(n-1)\pi_1 M n$, $n\pi_1 M (n-1)$, $(n-1)n\pi_1 M$ and $\pi_1 n M (n-1)$ are all distinct preimages of π. □

Proposition 5.3.8. *Let $\pi = \pi_1 \cdots \pi_n \in S_n$. Then π has exactly two preimages under `Cons` if and only if $\pi_n = n$, $\pi_1 \neq n-1 \neq \pi_{n-1}$ and $\mathrm{LTR}_v(\pi) = \{\pi_1, n-1, n\}$, for every $n \geq 4$.*

Therefore $c_n^{(2)} = (n-2)! \sum_{j=1}^{n-3} \frac{1}{j}$ for every $n \geq 4$, and $c_0^{(2)} = 0$, $c_1^{(2)} = 0$, $c_2^{(2)} = 1$, $c_3^{(2)} = 0$.

Proof. If π is of the form described in the statement, then we can use Proposition 5.3.4 to count its preimages, which are precisely two.

On the other hand, if $\pi_n \neq n$ or $\pi_1 = n-1$, then we can use the same argument as in the proof of Proposition 5.3.6 to see that π has at most one preimage. If $\pi_{n-1} = n-1$, then, by Lemma 5.3.7, we have that π has more than two preimages. The only remaining case is that of π having more than three left-to-right maxima. Suppose that α is a left-to-right maximum, with $\pi_1 < \alpha < n-1$. Writing $\pi = \pi_1 M_1 \alpha M_2 (n-1) M_3 n$, the following three permutations are preimages of π: $n\pi_1 M_1 \alpha M_2 (n-1) M_3$, $\pi_1 n M_1 \alpha M_2 (n-1) M_3$ and $\alpha \pi_1 M_1 n M_2 (n-1) M_3$.

To find a closed formula for $c_n^{(2)}$ we recall the so called Foata's fundamental bijection [17], which maps a permutation σ written in one-line notation to the permutation in cycle notation

obtained by inserting a left parenthesis in σ preceding every left-to-right maximum, then a right parenthesis where appropriate. Applying such a map to the set of permutations of length $n \geq 4$ having exactly two preimages, returns the set of permutations of length n in which n is a fixed point and consisting of two further cycles, the one containing $n-1$ having size at least two. If the cycle not containing $n-1$ has size j, with $1 \leq j \leq n-3$, then we have $\binom{n-2}{j}$ different ways to choose the elements in that cycle, and $(j-1)!$ different ways to arrange the elements into the cycle. The remaining $n-1-j$ elements (including $n-1$) can be arranged in $(n-2-j)!$ different ways. Summing up, we obtain that the number of permutations of length n with exactly two preimages is

$$\sum_{j=1}^{n-3} \binom{n-2}{j} (j-1)!(n-2-j)! = \sum_{j=1}^{n-3} \frac{(n-2)!}{j!(n-2-j)!}(j-1)!(n-2-j)! = (n-2)! \sum_{j=1}^{n-3} \frac{1}{j},$$

which concludes the proof. □

The quantities $H_n = \sum_{j=1}^{n} \frac{1}{j}$ are commonly known as *harmonic numbers*, so the above formula for $c_n^{(2)}$ can be written as $c_n^{(2)} = (n-2)!H_{n-3}$.

Proposition 5.3.9. *Let $\pi = \pi_1 \cdots \pi_n \in S_n$ with $n \geq 4$. Then π has exactly three preimages under* `Cons` *if and only if $\pi_n = n$, $\mathrm{LTR}_v(\pi) = \{k, k+1, n-1, n\}$, with $1 \leq k \leq n-3$, and both k, $k+1$ and $n-1$, n are not adjacent.*

Proof. If π is as in the statement above, then we can use Proposition 5.3.3 to count its preimages, which are precisely three, one for each of the subsets $\{k, n\}$, $\{k+1, n\}$, $\{n\}$.

On the other hand, we can argue as in the previous propositions to observe that, if a permutation has less than four left-to-right maxima or $n-1$ and n are adjacent, then it cannot have three preimages. If k and $k+1$ are adjacent, then π has at least four preimages, since each of the four subsets $\{k, n\}$, $\{k+1, n\}$, $\{n\}$, $\{k, k+1, n\}$ corresponds to at least one preimage. The same holds when the four left-to-right maxima are $k, h, n-1, n$, with $h > k+1$.

Finally, if π has more than four left-to-right maxima, suppose that $\{\alpha, \beta, \gamma, n-1, n\} \subseteq \mathrm{LTR}_v(\pi)$. Then there is a least a preimage for each of the subsets $\{\alpha, n\}$, $\{\beta, n\}$, $\{\gamma, n\}$, $\{n\}$, so π has more than three preimages. □

Looking at the previous results, we may wonder whether every number of preimages is allowed. More formally, does there exist any permutation π such that $|\mathbf{Cons}^{-1}(\pi)| = k$, for every k?

Notice that `Stacksort` [9], `Queuesort` (Proposition 3.2.11) and `Bubblesort` (Corollary 4.1.4) have some prohibited cardinality. Surprisingly, this is not the case for `Cons`.

Proposition 5.3.10. *For any positive integer k, there exists a permutation $\pi \in S$ such that $|\mathbf{Cons}^{-1}(\pi)| = k$.*

Proof. Let $k \geq 5$ and let $\rho \in S_{k-4}$. Then, by Proposition 5.3.3, $\pi = (k-3)\rho(k-2)(k-1)$ is a permutation of length $k-1$ with k preimages. Additionally, $\pi = 3152647$ has four preimages.

Finally, Propositions 5.3.5, 5.3.6, 5.3.8 and 5.3.9 deal with the cases $k = 0, 1, 2, 3$, thus concluding the proof. □

Chapter 6

Sorting with two queues in series

The `Queuesort` algorithm, which we presented in Section 2.3, uses a queue to sort a permutation. Clearly, it is possible to use more than one queue to (attempt to) sort a permutation, and this has been done in [24], where the author considers a network of queues. In this chapter we consider two queues, and study all the possible ways in which we can arrange them "in series" to sort a permutation.

6.1 Queues in series

Let Q_1 and Q_2 be two queues, and call $Front(Q_i)$ and $Back(Q_i)$ the first and last elements of Q_i, respectively. As detailed in Fig. 6.1, Q_1 and Q_2 are *queues in series* when the following operations are avalaible:

(e_1): move the current element of the input into Q_1;

(e_2): move $Front(Q_1)$ in Q_2;

(o_0): output the current element of the input;

(o_1): output $Front(Q_1)$;

(o_2): output $Front(Q_2)$;

(b): move the current element of the input into Q_2.

We want to characterize sortable permutations, and describe an optimal sorting algorithm. Observe that, if at any point during the sorting process Q_2 contains elements which are not in increasing order, then the output cannot be the identity permutation.

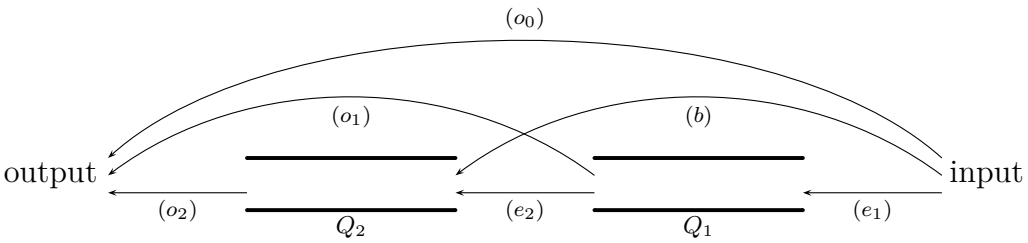


Figure 6.1: Two queues in series.

We begin considering only some subsets of the allowed operation, both because they are interesting by itself, and because one of them will be useful to understand the complete case. Namely, we exclude all the bypasses of queues, but only one at a time:

- (o_0), which bypasses both queues;
- (o_1), which bypasses Q_2;
- (b), which bypasses Q_1.

6.1.1 No (o_0)

Suppose that we are allowed to use all the previous operations, except (o_0). This means that every element of the permutation has to pass through Q_1 or Q_2 (or both) during the sorting process.

Proposition 6.1.1. *If a permutation contains the pattern* 4321*, then it is not sortable using the allowed operations (excluding* (o_0)*).*

Proof. Let π be a permutation that contains an occurrence *dcba* of the pattern 4321, and try to sort it. Suppose that, during the sorting process, d does not enter Q_1. Then it must enter Q_2, and wait there until elements c, b and a are sent into the output. Therefore c, b and a cannot enter Q_2, hence they must all pass through Q_1. Since the order of the elements passing through a queue does not change, c cannot be output after b and a, so the permutation cannot be sorted in this way.

On the other hand, suppose that d enters Q_1, and c bypass Q_1 and enters Q_2. Then we can reason as in the previous case and say that neither b nor a can enter Q_2, but they also cannot change order using only Q_1. This again renders impossible to sort them.

The last case is for both d and c to enter Q_1, waiting until b bypasses Q_1 to enter Q_2. Again, b prevents a from entering Q_2, but c prevents a from entering Q_1. Finally, c cannot move from Q_1 into Q_2, since this would force d to enter Q_2 before c, and they would not be sorted in the output. □

Actually, all the permutations avoiding the pattern 4321 are sortable without using (o_0). We present an algorithm that sorts every such permutation, followed by the proof of its optimality. *Input* denotes the current first element of the input.

Algorithm 6.1.2. $\mathcal{ALG}_{o_0}$*:*

Repeat the following steps until the input permutation π *is sorted, or the algorithm stops:*

1. *Execute* (o_1) *or* (o_2) *if this outputs the minimal non-output element (which is the next element that should be sent into the output);*
2. *else, execute* (e_1)*, unless this would cause the pattern* 321 *to appear in* Q_1*;*
3. *else compare* $Back(Q_2)$*,* $Front(Q_1)$ *and* $Input$*, and send into* Q_2 *(either executing* (b) *or* (e_2)*) the smallest element bigger than* $Back(Q_2)$*, if such element exists;*
4. *else, the algorithm stops.*

We define the algorithm so that it stops whenever it reaches a configuration which it cannot sort. In this way, we can recover non sortable patterns by analysing halting configurations.

The following remark is an immediate consequence of the description of $\mathcal{ALG}_{o_0}$.

Remark 6.1.3. *At every step of the sorting process with the algorithm $\mathcal{ALG}_{o_0}$, the content of Q_1 avoids the pattern* 321*.*

The following propositions describe a configuration in which the permutation is sorted, and one where it is not.

Proposition 6.1.4. *If, during the sorting process, we reach a configuration in which the input is empty and $Back(Q_2) < Front(Q_1)$, then the permutation is sorted by $\mathcal{ALG}_{o_0}$.*

Proof. Suppose that, during the sorting process, we reach a configuration in which the input is empty and $Back(Q_2) < Front(Q_1)$, and let σ be the permutation obtained by concatenating the contents of Q_2 and Q_1. Then σ avoids the pattern 321, because it is composed by an increasing sequence (in Q_2) followed by a sequence that avoids the pattern 321 (in Q_1, by Remark 6.1.3), in which the first element of the second sequence is greater than any element of the first sequence. This means that σ is sorted by `Queuesort`. In particular, during the sorting of σ with `Queuesort`, a configuration in which the content of the queue is the same as the content of Q_2 would be reached. Since $\mathcal{ALG}_{o_0}$ performs the same instructions as `Queuesort` when the input is empty, it will also sort the permutation in the same way.

□

Proposition 6.1.5. *Let π be a permutation, and suppose that during the sorting of π we reach a configuration in which $Back(Q_2) > Input$, and such that we cannot execute (e_1) (because it would form the pattern* 321 *in Q_1). Then π contains the pattern* 4321 *and is not sortable.*

Proof. Consider the first time in which the sorting process reach a configuration as described above, and let a be the first element of the input, $b = Back(Q_2)$ and Λ the configuration (that is, the "photography" of the state of input, output and the queues). Now consider the step in which b has moved into Q_2. If it entered Q_2 by operation (e_2), then call d and c the two elements in Q_1 that formed the pattern 321 with b. Necessarily, π contains the elements d, c, b, a in this order, so it contains the pattern 4321.

On the other hand, suppose by contradiction that b passes through Q_1, and consider the step in which it enters Q_2 by (e_2), which must happen before Λ. Let a' be the first element of the input, and $\alpha = Back(Q_2)$. Then (e_1) cannot be executed since, otherwise, the algorithm would have given priority to it. Moreover $\alpha < a'$, because otherwise Λ would not be the first configuration in which both (e_1) cannot be executed and $Back(Q_2) > Input$. Finally, $\alpha < b < a'$ since otherwise (e_2) would have been performed. However, if we follow the instructions of the algorithm, we can see that another element (either a' ot the current $Front(Q_1)$) is also put into Q_2 right after b. Then configuration Λ can only occur right after the moving of b into Q_2, but this is impossible, because $b = Back(Q_2)$ is smaller than $a' = Front(Q_1)$.

Therefore π contains the pattern 4321 and, by Proposition 6.1.1, is not sortable. □

We can now prove that $\mathcal{ALG}_{o_0}$ sorts every sortable permutation.

Theorem 6.1.6. *$\mathcal{ALG}_{o_0}(\pi) = id$ if and only if* $4321 \not\leq \pi$.

Proof. If π contains the pattern 4321 then, by Proposition 6.1.1, it cannot be sorted.

On the other hand, let π be a permutation that is not sorted by $\mathcal{ALG}_{o_0}$. Consider the configuration in which the algorithm stops, denoting with b the first element of the input (if any) and $\alpha = Back(Q_2)$. We know that we cannot execute (e_1), and that $\alpha > b$, because otherwise instruction (3) would be performed (either (b) or (e_2)). Then, by Proposition 6.1.5, we have $4321 \leq \pi$.

If the input is empty in Λ, the reasoning is longer. Let Λ be the current (halting) configuration, and let a be the next element that should be output. Since Q_2 is increasing from front to back, and the input is empty, then a is in Q_1 in Λ.

Tracing back the operations, we can return to the last configuration Λ' in which operation (e_2) has been executed. In Λ', let $\delta = Back(Q_2)$ and $\gamma = Front(Q_1)$. Since (e_2) is performed, then $\delta < \gamma$. Hence the input cannot be empty (in Λ'), because otherwise, by Proposition 6.1.4, the permutation would be sorted by $\mathcal{ALG}_{o_0}$. Let a' be the first element of the input. We have two possibilities: either $a' < \delta$, or $a' > \gamma$. In the first case we have that $4321 \leq \pi$ by Proposition 6.1.5, as we want.

We now show that the second case is impossible. Namely, we prove that if $\delta < \gamma < a'$ in configuration Λ', then a cannot be in any queue, nor in the input. Clearly a cannot be in Q_2, since Λ' appears before Λ and a is in Q_1 in Λ. Suppose by contradiction that a is in the input. In Λ', (e_2) is performed, therefore instruction (2) failed and there exist two elements d and c in Q_1 that form the pattern 321 together with $a' = Input$. In particular, (e_2) is performed for the last time, therefore d and c remain in Q_1 for the rest of the execution of the algorithm, preventing a to enter Q_1. Since a is in Q_1 in configuration Λ, this is impossible and a cannot be in the input in configuration Λ'.

Suppose now that a is already in Q_1 in configuration Λ'. Recall that $Front(Q_1) = \gamma$, and suppose by contradiction that $\gamma > a$. Then, by Remark 6.1.3, the elements between γ and a in Q_1 can either be greater than γ or smaller than a. Since (e_2) is performed for the last time in Λ', they cannot be greater than γ, and must hence be smaller than a. However, any element smaller than a is in the output in configuration Λ, therefore a is the first element of Q_1 in Λ, which is a contradiction with the fact that the algorithm stops.

Finally, suppose that a is in Q_1 in configuration Λ', and $\gamma < a$. Since a is the smallest element not in the output in Λ, then γ, δ and all the other elements situated in Q_1 in configuration Λ' are sent into the output at some point before configuration Λ. However, Λ' is the last configuration in which (e_2) is performed, so all the elements in Q_2 in configuration Λ come from the input, by instruction (3). This is impossible, because after that instruction $Back(Q_1) < Front(Q_2)$, so the instruction that empties the input must have been performed while $Back(Q_2) < Front(Q_1)$, which implies that the permutation is sorted, by Proposition 6.1.4. This is a contradiction, therefore $a' < \delta$ and, as we already explained, $4321 < \pi$.

□

6.1.2 No (o_1)

Suppose that we can use all the operations, except (o_1). This case gives the same result as the previous one. Indeed, we will prove that every permutation containing the pattern 4321 is not sortable, and we will then describe an algorithm that sorts every 4321-avoiding permutation.

For starters, notice that if a permutation enters any queue, then it must pass through Q_2.

Proposition 6.1.7. *If a permutation contains the pattern* 4321*, then it is not sortable in any way using the allowed operations (excluding* (o_1)*).*

Proof. Let π be a permutation containing an occurrence *dcba* of the pattern 4321, and try to sort it. Suppose that, during the sorting process, d does not enter Q_1. Since there are smaller elements after it, it cannot go into the output, and must thus enter Q_2. However, c must also enter Q_1 or Q_2, but then it cannot be output before d, which would cause the output not to be sorted.

On the other hand, suppose that d enters Q_1. Then c cannot enter Q_1, otherwise it could not reach the output before d. Therefore c must enter Q_2. However, this implies that b cannot

reach the output before c, unless by operation (o_0), which would cause b to reach the output before a. In any case, π cannot be sorted using the allowed permutations. □

We now describe a sorting algorithm that utilizes the allowed operations.

Algorithm 6.1.8. $\mathcal{ALG}_{o_1}$

Repeat the following steps until the input permutation π is sorted or the algorithm stops:

1. *Execute (o_0) or (o_2) if this outputs the minimal non-output element (which is the next element that should be sent into the output);*
2. *else, if $Back(Q_1) < Input$ and $Back(Q_2) < Input$, then execute (e_1);*
3. *else, if $Back(Q_2) < Input < Front(Q_1)$ then execute (b)[1];*
4. *else, if $Back(Q_2) < Front(Q_1)$, execute (e_2);*
5. *else, the algorithm stops.*

The following remark is an immediate consequence of the description of $\mathcal{ALG}_{o_0}$.

Remark 6.1.9. *At every step of the sorting process with the algorithm $\mathcal{ALG}_{o_1}$, the content of both Q_1 and Q_2 is in increasing order.*

Actually, the algorithm has a stronger property. Indeed, the concatenation of the contents of Q_2 and Q_1 is in increasing order, and the following proposition is the missing piece for proving that.

Proposition 6.1.10. *At every step of the sorting process with the algorithm $\mathcal{ALG}_{o_1}$, either one of the queues is empty or $Back(Q_2) < Front(Q_1)$.*

Proof. The thesis is a consequence of the restrictions on the instructions of $\mathcal{ALG}_{o_1}$. Indeed, the queues are empty at the beginning of the sorting process, and if either one of the queues is empty or $Back(Q_2) < Front(Q_1)$, then this remains true after any of the instruction is performed. □

When put together, Remark 6.1.9 and Proposition 6.1.10 imply that the concatenation of the contents of Q_1 and Q_2 is an increasing sequence at all times. We encapsulate this fact in a lemma, to use it in the proof of next proposition.

Lemma 6.1.11. *During the execution of algorithm $\mathcal{ALG}_{o_1}$, the concatenation of the contents of Q_1 and Q_2 is an increasing sequence at all times.*

Proposition 6.1.12. *Let π be a permutation and suppose that, during the sorting of π using $\mathcal{ALG}_{o_1}$, we reach a configuration in which $Back(Q_2) > Input$, and neither of the two elements can be output. Then π contains the pattern* 4321.

Proof. Let Λ be the configuration in which, during the sorting of π with $\mathcal{ALG}_{o_1}$, $Back(Q_2) > Input$ for the first time. In Λ, let $c = Back(Q_2)$, $b = Input$ and a the smallest element not yet sent into the output (so the next element to be output). By Lemma 6.1.11, a is in the input, and is different from b.

Suppose that c was sent in Q_1 by operation (b) (instruction (3)), then there exists an element $d > c$ that prevented c from entering Q_1. Thus the elements d, c, b, a forms an occurrence of 4321 in π.

[1] if Q_2 is empty, we just want $Input < Front(Q_1)$.

On the other hand, suppose by contradiction that c entered Q_2 by operation (e_2) (instruction 4), and call Λ' the configuration in which $c = Front(Q_1)$ and (e_2) is performed. Let γ be the first element of the input in configuration Λ', $\delta = Back(Q_2)$ and $\omega = Back(Q_1)$. Necessarily, $\delta < \gamma$, since Λ is the first configuration in which $Back(Q_2) > Input$, and $\delta < c < \omega$ by Lemma 6.1.11. Finally, $\delta < c < \gamma < \omega$ since (e_2) is performed instead of (e_1) or (b). This is enough to reach a contradiction, since the only possible operation to execute after (e_2) are either (b) or (e_2), both of which would change the last element of Q_2 from c to something else, and would make impossible to reach configuration Λ.

This implies that c entered Q_2 by operation (b), and π contains the pattern 4321, concluding the proof. □

We can then prove that $\mathcal{ALG}_{o_1}$ sorts all permutations in Av(4321).

Theorem 6.1.13. *$\mathcal{ALG}_{o_1}(\pi) = id$ if and only if $4321 \not\leq \pi$.*

Proof. If π contains the pattern 4321 then, by Proposition 6.1.7, it cannot be sorted.

On the other hand, let π be a permutation for which $\mathcal{ALG}_{o_1}$ stops. As a consequence of Lemma 6.1.11 and the fact that instruction (4) cannot be performed, we have that Q_1 is empty. Since instruction (2) cannot be performed, then $Back(Q_2) > Input$, therefore π contains the pattern 4321 by Proposition 6.1.12. □

6.1.3 No (b)

Suppose that we can use all the operations, except (b). Notice that every element that is not immediately sent to the output by (o_0) must enter Q_1. Even in this case only 4321-avoiding permutations are sortable.

Proposition 6.1.14. *If a permutation contains the pattern 4321, then it is not sortable in any way using the allowed operations.*

Proof. Let π be a permutation containing an occurrence $dcba$ of the pattern 4321. Then d, c and b Must enter Q_1. The elements passing through Q_1 must be sorted using only a queue (namely, Q_2) with a bypass. Since the pattern 321 is not sortable using just a queue, then d, c and b cannot be sorted. □

The proof of the proposition highlights an interesting characteristic of our structure: Q_1 essentially acts as a container which serves as an input for `Queuesort`, while operation (o_0) allows to immediately output elements bypassing the queue sorting procedure.

Algorithm 6.1.15. *$\mathcal{ALG}_b$:*

Repeat the following steps until the input permutation π is sorted or the algorithm stops:

1. *execute (o_0) if this outputs the minimal non-output element (which is the next element that should be sent into the output);*
2. *else, execute a step of the `Queuesort` algorithm using Q_2 as the queue and $Front(Q_1)$ as the current element of the input, unless this would output an element which is not the next element to be outputted;*
3. *else, execute (e_1), unless this would cause the the pattern 321 to appear in the concatenation of the contents of Q_2 and Q_1;*
4. *else, the algorithm stops.*

The algorithm heavily relies on `Queuesort`, and makes sure that the elements fed into it (that is, into Q_1) avoid the pattern 321, since otherwise they could not be sorted.

Theorem 6.1.16. *$\mathcal{ALG}_b(\pi) = id$ if and only if $4321 \not\leq \pi$.*

Proof. If π contains the pattern 4321 then, by Proposition 6.1.14, it cannot be sorted.

On the other hand, let π be a permutation for which $\mathcal{ALG}_{o_1}$ stops. Consider the configuration in which the algorithm stops, and let a the next element that should be sent into the output. Since the sequence of elements inside Q_1 and Q_2 avoids the pattern 321, the element a must still be in the input, otherwise it would be correctly sent into the output by `Queuesort`.

Let b be the first element of the input. The algorithm stops, therefore instruction (3) fails and there exists two elements, d and c, that forms an occurrence of the pattern 321 together with b. Since a is in the output and is different from b (otherwise a would be sent into the output by (o_0)), the elements d, c, b, a form an occurrence of the pattern 4321 in π. □

6.2 Sorting with every operation

Finally, suppose that we have no forbidden operation. This is the complete *sorting with two queues in series* problem. The characterization of permutations sorted without using operation (o_0) allows us to see the problem as sorting with $\mathcal{ALG}_{o_0}$ with a bypass. First of all, we prove that all permutations containing the pattern 54321 cannot be sorted.

Proposition 6.2.1. *Let π be a permutation containing the pattern 54321. Then it is not sortable using two queues in series.*

Proof. Let π be a permutation containing an occurrence *edcba* of the pattern 54321. Then the elements e, d, c, b cannot be sent to the output by (o_0). Therefore they have to be sorted using the two queues with all the other operations. By Proposition 6.1.1, this is impossible. □

Next, we define an algorithm that sorts every 54321 avoiding permutation. To do so, we make use of the notion of *right-to-left minima*, which are elements smaller that everything after them. That is, π_i is a right-to-left minima of π if and only if $\pi_i < \pi_j$ for every $j > i$.

This elements are used to determine when (and if) to output something using o_0. Since there is nothing bigger after them, they can be sent into the output by (o_0). The only concern is that, when a right-to-left minima is the current element of the input, there might still be some element inside Q_1 or Q_2 smaller than it. Therefore algorithm $\mathcal{ALG}_{o_0}$ is performed until the right-to-left minima is the next element to be sent into the output.

Algorithm 6.2.2. *$\mathcal{ALG}_{2Q}$:*

Repeat the following steps until the input permutation π is sorted or the algorithm stops:

1. *if the current element of the input is not a right-to-left minima, then execute a step of algorithm $\mathcal{ALG}_{o_0}$, using it as the current input, and Q_1, Q_2 as the queues;*
2. *else, if the current element of the input is the next element to be sent into the output, execute (o_0);*
3. *else, execute a step of algorithm $\mathcal{ALG}_{o_0}$, using an empty input and Q_1, Q_2 as the queues;*
4. *else, the algorithm stops.*

Notice that the algorithm never "shows" the right-to-left minima to $\mathcal{ALG}_{o_0}$, because they are not meant to enter any queue.

Theorem 6.2.3. *$\mathcal{ALG}_{2Q}(\pi) = id$ if and only if $54321 \not\leq \pi$.*

Proof. If π contains the pattern 54321, then it is not sortable by Proposition 6.2.1.

On the other hand, let π be a permutation for which $\mathcal{ALG}_{2Q}$ stops, and consider the configuration in which that happens.

Since the algorithm stops, then the subalgorithm $\mathcal{ALG}_{o_0}$ cannot perform any operation. By Theorem 6.1.6, this implies that π contains an occurrence *edcb* of the pattern 4321. Since the subalgorithm never considers any right-to-left minima of π, neither of the elements forming the pattern can be one. Specifically, b is not a right-to-left minima of π. Therefore there exists an element a smaller than b which appears after b in π. Therefore the elements e, d, c, b, a form an occurrence of the pattern 54321 in π. □

Chapter 7

Interval posets of permutations

In this chapter we focus on something different: the intervals of a permutation. In particular, we focus on the partially ordered set (*poset*) of the intervals of a permutation, defined in [26].

Intervals in permutations can be defined in several ways (focusing on the indices – a.k.a. positions – in $\sigma = \sigma_1\sigma_2\dots\sigma_n$ or on the values). Here, we follow the definition of [26], and define an *interval* of a permutation $\sigma = \sigma_1\sigma_2\dots\sigma_n$ as an interval $[j, j+h]$ of values (for some $1 \leq j \leq n-1$ and some $0 \leq h \leq n-j$) which is the image by σ of an interval $[i, i+h]$ of positions (for some $1 \leq i \leq n-h$). Namely, $[j, j+h]$ is an interval of σ when there exists an i satisfying $\{\sigma_i, \dots, \sigma_{i+h}\} = [j, j+h]$. The singletons $\{1\}, \{2\}, \dots, \{n\}$ and the complete interval $[1, n]$ are always intervals of σ, and are called *trivial*. The empty set is also an interval of σ, although most often we do not include it in the set of intervals of σ. Non-trivial and non-empty intervals are called *proper*. For example, the proper intervals of $\sigma = 4\,5\,6\,7\,9\,3\,1\,2\,8$ are $[4,5], [4,6], [4,7], [5,6], [5,7], [6,7], [1,2]$ and $[1,3]$.

The inclusion relation naturally equips the set of intervals (proper ones and trivial ones) with a poset structure: the elements of this poset are the intervals, and the relation is the set inclusion. We can consider two versions of this poset: a first one in which the empty interval is an element (hence, the only minimal element in the poset), and a second one which excludes the empty interval (the minimal elements being then the singletons $\{1\}$ through $\{n\}$).

While posets are essentially "unordered" objects, we follow [26] and consider particular plane embeddings of these posets.

Definition 7.0.1. Let σ be a permutation. The (original) *interval poset* of σ, denoted $P(\sigma)$, is the plane embedding of the poset of the non-empty intervals of σ where the minimal elements appear in the order $\{1\}, \{2\}, \dots, \{n\}$ from left to right.

We denote $P_\bullet(\sigma)$ the poset of the possibly empty intervals of σ with the same plane embedding: $P_\bullet(\sigma)$ is just $P(\sigma)$ with a new minimum smaller than all minimal elements of $P(\sigma)$.

The left part of Fig. 7.1 shows the interval poset of our running example. Clearly, in this figure as well as in general, every element of the poset represents the interval which consists of the set of values below it in the poset.

Deviating from [26], we also wish to consider a second embedding of the interval poset of a permutation, illustrated on the right part of Fig. 7.1. We believe that this second embedding is also very natural, maybe even more natural than the first one once compared with the decomposition trees of permutations, as we shall see later.

Definition 7.0.2. Let σ be a permutation. The modified interval poset of σ, denoted $\tilde{P}(\sigma)$, is the plane embedding of the poset of the non-empty intervals of σ where the minimal elements appear in the order $\{\sigma_1\}, \{\sigma_2\}, \dots, \{\sigma_n\}$ from left to right.

We also define $\tilde{P}_\bullet(\sigma)$ adding a new minimum, as in Definition 7.0.1.

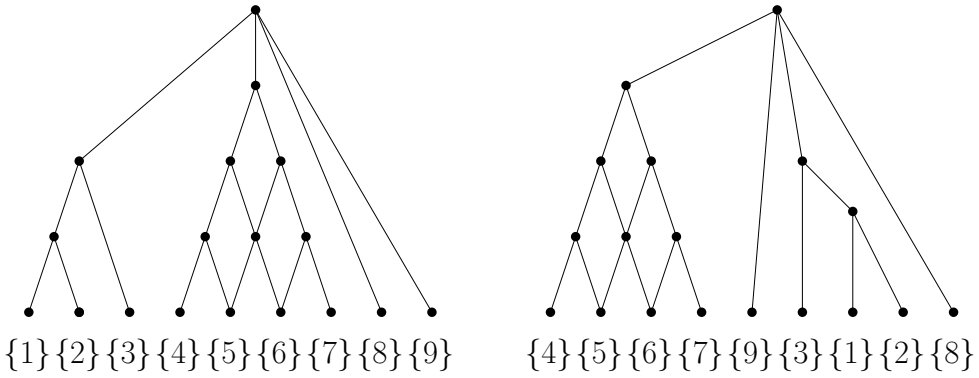


Figure 7.1: From left to right: The original interval poset $P(\sigma)$, and the modified interval poset $\tilde{P}(\sigma)$, for $\sigma = 4\,5\,6\,7\,9\,3\,1\,2\,8$.

Although these two embeddings are different, they can be connected by the following remark. It is illustrated comparing the left parts of Figs. 7.1 and 7.2.

Proposition 7.0.3. *For any permutation σ, $P(\sigma) = \tilde{P}(\sigma^{-1})$ (and consequently, we also have $P_\bullet(\sigma) = \tilde{P}_\bullet(\sigma^{-1})$).*

Proof. First, observe that there is a bijective correspondence between the intervals of σ and those of σ^{-1}. Namely, every interval $[j, j+h]$ of σ such that $\{\sigma_i, \dots, \sigma_{i+h}\} = [j, j+h]$ corresponds to the interval $[i, i+h]$ of σ^{-1} – indeed, $[i, i+h] = \{\sigma_j^{-1}, \dots, \sigma_{j+h}^{-1}\}$. Next observe that the inclusion relation is preserved by this correspondence: for I and J two intervals of σ, and I' and J' the corresponding intervals of σ^{-1}, it holds that $I \subseteq J$ if and only if $I' \subseteq J'$. Therefore, the nonplane poset structures of $P(\sigma)$ and $\tilde{P}(\sigma^{-1})$ are identical.

We are left with proving that the plane embeddings of $P(\sigma)$ and $\tilde{P}(\sigma^{-1})$ are the same. For this, we identify the minimal elements of $P(\sigma)$ and $\tilde{P}(\sigma^{-1})$ by a pair (i, j) where i is the position of this element and j its value. Therefore, in $P(\sigma)$, the i-th minimal element in the left-to-right order has value i hence corresponds to the pair (σ_i^{-1}, i). On the other hand, in $\tilde{P}(\sigma^{-1})$, the i-th minimal element in the left-to-right order is at position i in σ^{-1}, hence also corresponds to the pair (σ_i^{-1}, i), concluding the proof. □

In this chapter we focus on statements regarding the posets $\tilde{P}(\sigma)$, but Proposition 7.0.3 then of course allows to interpret them on the original posets $P(\sigma)$ by considering the inverse permutation.

7.0.1 Substitution decomposition and decomposition trees

While interval posets of permutations have been defined and studied only recently, the inclusion relations among the intervals of permutations have been the subject of many studies, in the algorithmic and in the combinatorial literature. In both cases, the set of intervals is represented by means of a tree, which is called *strong interval tree* or *(substitution) decomposition tree* depending on the context. For historical references, we refer to the introduction of [28, Section 3.2], and to [8, Section 3], which in addition explains the equivalence between the two approaches.

Below, we review the definition of these trees, following essentially the combinatorial approach introduced in [2]. This is also the approach presented in the survey [28, Section 3.2]. We need to recall some terminology.

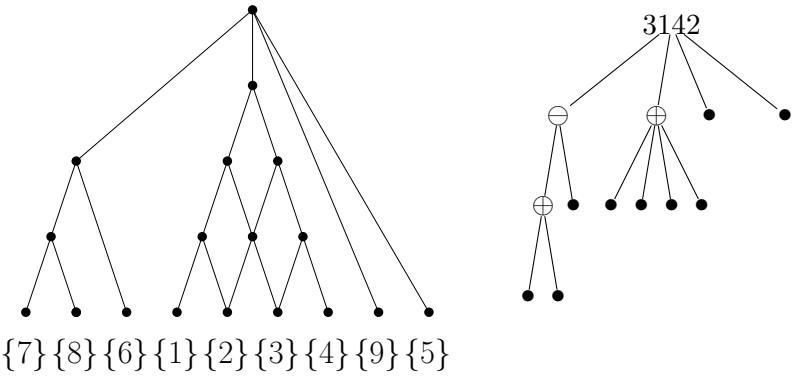


Figure 7.2: From left to right: The interval poset $\tilde{P}(\sigma^{-1})$, and the decomposition tree $T(\sigma^{-1})$, for $\sigma = 4\,5\,6\,7\,9\,3\,1\,2\,8$, *i.e.*, $\sigma^{-1} = 7\,8\,6\,1\,2\,3\,4\,9\,5$. The substitution decomposition of σ^{-1} is indeed $\sigma^{-1} = 3142[\ominus[\oplus[1,1],1],\oplus[1,1,1,1],1,1]$.

A permutation is *simple* if it is of size at least 4 and its only intervals are the trivial ones. For example, there are two simple permutations of size 4 (namely, 2413 and 3142) and a simple permutation of size 7 is 5247316.

Given π a permutation of size k and k permutations $\alpha_1, \ldots, \alpha_k$, the *inflation* of π by $\alpha_1, \ldots, \alpha_k$ (a.k.a. *substitution* of $\alpha_1, \ldots, \alpha_k$ in π), denoted $\pi[\alpha_1, \ldots, \alpha_k]$, is the permutation obtained from π by replacing each element π_i by an interval I_i, such that all elements in I_i are larger than all elements in I_j whenever $\pi_i > \pi_j$, and such that the elements of I_i form a subsequence order-isomorphic to α_i. For instance $312[12, 231, 4321] = 89\,231\,7654$. For any $k \geq 2$, we write inflations in $\pi = 12\ldots k$ (resp. $\pi = k\ldots 21$) as inflations in $\oplus$ (resp. $\ominus$), the value of k being simply determined by the number of components in the inflation. For instance, $\oplus[1, 3412, 21, 12]$ means $1234[1, 3412, 21, 12] = 145237689$. Finally, we say that a permutation σ is $\oplus$- (resp. $\ominus$-)indecomposable when there does not exist any k and α_i such that $\sigma = \oplus[\alpha_1, \ldots, \alpha_k]$ (resp. $\sigma = \ominus[\alpha_1, \ldots, \alpha_k]$).

Theorem 7.0.4. *[2] Every permutation σ of size at least 2 can be uniquely decomposed as $\pi[\alpha_1, \ldots, \alpha_k]$ with one of the following satisfied:*

- *π is simple;*
- *$\pi = 12\ldots k$ for some $k \geq 2$ and all α_i are $\oplus$-indecomposable;*
- *$\pi = k\ldots 21$ for some $k \geq 2$ and all α_i are $\ominus$-indecomposable.*

The description of σ as $\pi[\alpha_1, \ldots, \alpha_k]$ satisfying the above is called the substitution decomposition *of σ.*

Remark 7.0.5. *In the first item, we note our (somewhat unusual) convention that simple permutations are of size at least 4. In the second item, the statement of [2] is rather $\pi = \oplus[\alpha, \beta]$ with only α $\oplus$-indecomposable. It is easily seen to be equivalent to our second item above, decomposing recursively inside β until reaching a second component in $\oplus$ which is itself $\oplus$-indecomposable. A similar remark obviously applies to the third statement.*

Applying the substitution decomposition recursively inside the α_i of Theorem 7.0.4 until we reach permutations of size 1, we can represent every permutation by a tree, called its *decomposition tree*, which we denote $T(\sigma)$. See an example on the right side of Fig. 7.2.

Definition 7.0.6. A *decomposition tree* of size n is a rooted tree with n leaves and in which every internal vertex v satisfies the following:

- either v is labeled by a simple permutation, whose size is equal to the number of children of v,
- or v is labeled by $\oplus$ or $\ominus$ and has at least 2 children.

Decomposition trees are *plane*, meaning that the children of every internal vertex are ordered from left to right.

In addition, decomposition trees are required to not contain any $\oplus - \oplus$ edge nor any $\ominus - \ominus$ edge.

The following theorem follows immediately from Theorem 7.0.4 (the absence of $\oplus - \oplus$ and $\ominus - \ominus$ edges echoing the conditions in the second and third items of Theorem 7.0.4).

Theorem 7.0.7. *The correspondence between permutations and decomposition trees is a size-preserving bijection. Under this correspondence, σ_i corresponds to the i-th leaf of $T(\sigma)$ in the left-to-right order.*

We do not discuss here in details the relation between the intervals of a permutation and its decomposition tree. We point out to the interested readers that the nodes of $T(\sigma)$ are the so-called *strong intervals* of σ. Those are defined as the intervals of σ which do not overlap any other interval of σ, two intervals I and J being overlapping when $I \cap J$ is neither empty nor equal to I or J. Details can be found in [8, Section 3] where references are also given.

However, the relation between interval posets and decomposition trees – which we present in the next section – will hopefully clarify the link between decomposition trees and intervals of permutations, even without going back to these references.

7.1 Computing the poset from the decomposition tree

We start by a few definitions (illustrated by Fig. 7.3), and some easy facts from [26].

The *dual claw poset of size k* is the poset with $k+1$ elements, one being larger than all others, which are incomparable among them. Note that it has k minimal elements. It was observed in [26, Proposition 4.3] that it is the interval poset of all simple permutations of size k (and only those).

The *argyle poset of size k* is the interval poset of the permutation $12\dots k$. It has k minimal elements. It was observed in [26, Proposition 4.4] that it is the interval poset of exactly two permutations: $12\dots k$ and $k\dots 21$.

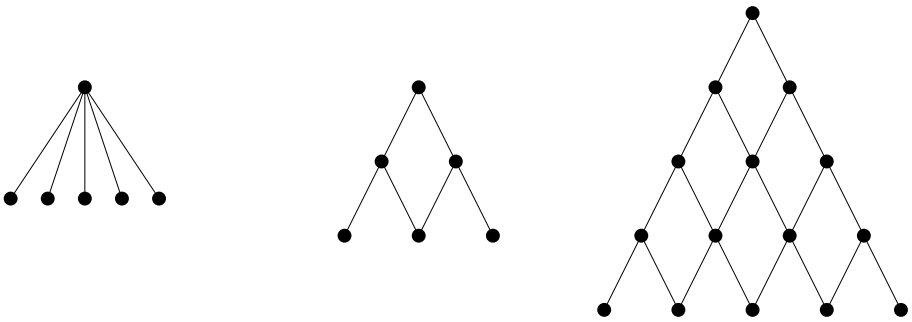


Figure 7.3: From left to right: the dual claw poset of size 5, and the argyle posets of size 3 and 5.

Now, consider the following procedure, which takes as input a permutation σ (or equivalently, its decomposition tree $T(\sigma)$), and returns a plane embedding of a poset (which we denote $Q(\sigma)$ for the moment).

1. If $\sigma = 1$ (*i.e.* $T(\sigma) = \bullet$), we set $Q(\sigma)$ to be the poset containing only one element.
2. Otherwise, we consider the substitution decomposition $\sigma = \pi[\alpha_1, \dots, \alpha_k]$ of σ.
3. If π is simple, we let R be the dual claw of size k. Otherwise (*i.e.* if π is $\oplus$ or $\ominus$) we denote by R the argyle poset of size k.
4. We let the minimal elements of R be $p_1, \dots, p_k$, in this order from left to right.
5. The poset $Q(\sigma)$ is obtained by taking R and replacing, for each $i \in [1, k]$, p_i by the recursively-obtained poset $Q(\alpha_i)$.

We can observe that the auxiliary poset R in this procedure actually is $R = \tilde{P}(\pi)$.

Proposition 7.1.1. *For every permutation σ, $Q(\sigma) = \tilde{P}(\sigma)$.*

In other words, the interval poset $\tilde{P}(\sigma)$ of any permutation σ can be obtained from $T(\sigma)$ by replacing any internal node labeled by a simple permutation by a single element, and any internal node labeled by $\oplus$ or $\ominus$ having k children by several elements arranged in an argyle poset structure with k minimal elements. This is illustrated comparing the two pictures of Fig. 7.2.

We note that it was observed in [26, Section 2] that the substitution decomposition lays the ground work for interval posets. Specifically, Proposition 7.1.1 can actually be seen as a rephrasing of [26, Theorem 4.8]. Nevertheless, for completeness, we provide a proof of Proposition 7.1.1 below.

Similarly, it was observed in [26, Section 6] that the "separation trees" of separable permutations bear some resemblance with interval posets, although they are not the same. Actually, these separation trees are a restricted version of the decomposition trees in the special case of separable permutations, so our procedure explains precisely this resemblance and allows to go from one representation to the other.

Proof of Proposition 7.1.1. The proof is by induction on the depth (counted in number of edges) of $T(\sigma)$. The statement is clear when $T(\sigma)$ is of depth 0, *i.e.*, $T(\sigma) = \bullet$. So, let us assume that $T(\sigma)$ is of depth at least 1, and let $\sigma = \pi[\alpha_1, \dots, \alpha_k]$ be the substitution decomposition of σ. By induction hypothesis, for each i, $Q(\alpha_i) = \tilde{P}(\alpha_i)$.

If π is simple, then every interval of σ is either $[1, |\sigma|]$ or included in some α_i, implying that $Q(\sigma) = \tilde{P}(\sigma)$.

If π is $\oplus$ or $\ominus$, the intervals of σ are either included in some α_i or consist of a union of α_i's for a set of consecutive indices i. Such intervals being represented exactly by the elements of an argyle poset, this implies that $Q(\sigma) = \tilde{P}(\sigma)$ also in this case. □

Remark 7.1.2. *With the above procedure and Proposition 7.1.1, it is natural to refer to the elements of $\tilde{P}_\bullet(\sigma)$ as elements of smaller interval posets. More precisely, denoting $\pi[\alpha_1, \dots, \alpha_k]$ the substitution decomposition of σ, we can see $\tilde{P}_\bullet(\sigma)$ as the poset obtained by identifying the minimal elements of $\tilde{P}(\pi)$ with the maxima of the $\tilde{P}(\alpha_i)$'s, and then adding a minimum $\emptyset$. Therefore we will refer to the elements of $\tilde{P}_\bullet(\sigma)$ different from $\emptyset$ using the corresponding elements of $\tilde{P}(\pi)$ or $\tilde{P}(\alpha_i)$.*

While the material presented in this section is rather simple and not new, it allows to answer one of the open questions of [26], namely Question 7.3. This question is interested in a description of the shared properties of the *interval generators* of an interval poset. We prefer to call *realizers* these interval generators of [26], in line with the usual terminology in the algorithmic literature.

Given an interval poset P, a *realizer* of P is a permutation σ such that $\tilde{P}(\sigma) = P$.

Given a decomposition tree T, we define its *skeleton* $sk(T)$ as the plane tree with the same set of nodes and the same genealogy, but where internal nodes have different labels. An internal node of $sk(T)$ is labeled *prime* (resp. *linear*) when its label in T is a simple permutation (resp. $\oplus$ or $\ominus$).

Proposition 7.1.3. *Let P be an interval poset, and σ be any permutation such that $\tilde{P}(\sigma) = P$. Then a permutation τ is a realizer of P if and only if $sk(T(\tau)) = sk(T(\sigma))$.*

Proof. It follows immediately from Proposition 7.1.1 and the description of the procedure computing $Q(\sigma)$. □

For example, this allows to compute the set of realizers of the poset (without labels on the minimal elements) displayed in Fig. 7.2, left. This set is obtained by considering the skeleton of the tree shown in Fig. 7.2, right, and by considering all possible labels for its prime root (here 2413 or 3142), all possible labels for the second child of the root (here $\oplus$ or $\ominus$), and all possible labels on the left branch starting at the first child of the root (here $\oplus - \ominus$ or $\ominus - \oplus$, keeping in mind that decomposition trees do not contain any $\oplus - \oplus$ nor $\ominus - \ominus$ edge). This yields the following set of eight realizers: $\{342678915, 324678915, 342987615, 324987615, 786123495, 768123495, 786123495, 768123495\}$.

We note that this idea was already present in [26, Theorem 5.1], but only with the purpose of computing the *number* of realizers of an interval poset. Of course, the point of view of decomposition trees allows to provide an alternative proof of [26, Theorem 5.1], although we do not provide details here, since they are very close to the proof of Theorem 4.1 in [26].

Some of the results of [26] can be seen as consequences (or special cases) of Proposition 7.1.3.

Corollary 7.1.4. *The following claims hold.*

- *For every $k \geq 4$, the dual claw poset of size k is the poset of all simple permutations of size k, and only those (see [26, Proposition 4.3]).*
- *For every $k \geq 2$, the argyle poset of size k is the poset of exactly two permutations: $12\dots k$ and $k\dots 21$. (see [26, Proposition 4.4]).*
- *For every permutation $\sigma = \sigma_1\sigma_2\dots\sigma_n$, denoting σ^R its* reverse *$\sigma_n\dots\sigma_2\sigma_1$, we have $P(\sigma^R) = P(\sigma)$ (see [26, Lemma 2.5]).*

Proof. For the first item, we simply observe that the simple permutations σ of size k are exactly those such that $sk(T(\sigma))$ consists of a prime node at the root with only k leaves pending under it.

Similarly, for the second item, we observe that $sk(T(\sigma))$ consists of a linear node at the root with only k leaves pending under it if and only if $\sigma = 12\dots k$ or $k\dots 21$.

For the third item, we first recall that for all σ, $P(\sigma^R) = \tilde{P}((\sigma^R)^{-1})$ by Proposition 7.0.3. Second, we observe that $(\sigma^R)^{-1} = (\sigma^{-1})^C$ where for any permutation $\pi = \pi_1\pi_2\dots\pi_n$, π^C denotes the *complement* $(n+1-\pi_1)(n+1-\pi_2)\dots(n+1-\pi_n)$ of π. So, our claim is equivalent to showing that $\tilde{P}(\pi) = \tilde{P}(\pi^C)$ for all π. And this holds since $T(\pi^C)$ is obtained from $T(\pi)$ by complementing each simple permutation at a node, and by changing each $\oplus$ (resp. $\ominus$) into a $\ominus$ (resp. $\oplus$), noting that these operations do not affect the prime or linear labels of the nodes of $sk(T(\pi))$. □

In addition, [26, Theorem 4.8] states that interval posets are the posets which can be constructed starting from the 1-element poset, and recursively replacing minimal elements with dual claw posets, argyle posets or binary tree posets (defined in [26, Definition 4.2]). Our

Proposition 7.1.1 states that interval posets are those that can be constructed starting from the 1-element poset, and recursively replacing minimal elements with dual claw posets or argyle posets. Since binary tree posets can straightforwardly be obtained from the 1-element poset recursively replacing minimal elements with argyle posets with two minimal elements, Proposition 7.1.1 therefore implies [26, Theorem 4.8]. It also allows to identify more clearly which "building blocks" are needed, namely dual claw posets and argyle posets, without explicitly needing binary tree posets.

7.2 Alternative proofs of known structural results

In this section, we review several structural properties of the interval posets, which were already proved in [26]. We believe that the approach through decomposition trees allows to provide more straightforward proofs of these statements.

We briefly recall some classical terminology regarding properties of posets. More details can be found in [25]. Let P be a generic poset, whose partial order is denoted by $\leq$.

For a and b be two elements of P, we say that a *covers* b when $b < a$ and there is no element c of P such that $b < c < a$. The *Hasse diagram* of P is a drawing of the graph whose vertices are the elements of P, and whose edges are the covering relations in P, with a being drawn higher than b whenever $b < a$. We say that P is *planar* when its Hasse diagram can be drawn in such a way that no two edges cross.

For any two elements a and b in P, their *meet* (resp. *join*) denoted $a \vee b$ (resp. $a \wedge b$) is the smallest element c such that $a \leq c$ and $b \leq c$. (resp. the largest element c such that $c \leq a$ and $c \leq a$), if such an element exists. We say that P is a *lattice* when, for any two elements a and b of P, both $a \vee b$ and $a \wedge b$ exist. We also say that P is *modular* when, for any two elements a and b of P, they both cover $a \wedge b$ if and only if $a \vee b$ covers them both.

Theorem 7.2.1. *([26, Theorem 3.2]) For every permutation σ, the posets $\tilde{P}(\sigma)$ and $\tilde{P}_\bullet(\sigma)$ are planar.*

Proof. The proof heavily relies on Proposition 7.1.1 and the computation of $\tilde{P}(\sigma)$ from the decomposition tree $T(\sigma)$ of σ which is the core of Section 7.1. Clearly, for any permutation σ, the tree $T(\sigma)$ is planar, and the transformations performed to obtain $\tilde{P}(\sigma)$ from it maintain the planar property. This shows that $\tilde{P}(\sigma)$ is planar (in addition with a planar drawing where all minimal elements can be placed on a horizontal line at the bottom of the picture). Since $\tilde{P}_\bullet(\sigma)$ is obtained by adding to $\tilde{P}(\sigma)$ a new minimum smaller than all minimal elements of $P(\sigma)$, the above ensures that $\tilde{P}_\bullet(\sigma)$ is also planar. □

Theorem 7.2.2. *([26, Theorem 3.3]) For every permutation σ, the poset $\tilde{P}_\bullet(\sigma)$ is a lattice.*

Proof. First, we note the following fact: if I and J are two elements of some poset $\tilde{P}_\bullet(\sigma)$ such that $I \subseteq J$ we have $I \wedge J = I$ and $I \vee J = J$. We shall use this fact repeatedly, particularly when one of I or J is $\emptyset$.

Now, we prove the statement by structural induction on the substitution decomposition of σ.

If $\sigma = 1$, our claim follows immediately from the above fact (since the only two elements of $\tilde{P}_\bullet(\sigma)$ are $\{1\}$ and $\emptyset$).

If σ is a simple permutation, then $\tilde{P}_\bullet(\sigma)$ is a dual claw poset with an added minimum $\emptyset$. Clearly, any pair of elements have a meet and a join in this poset.

If σ is increasing or decreasing then $\tilde{P}_\bullet(\sigma)$ is an argyle poset with an added minimum. The elements of this poset correspond to the intervals $[a, b]$ for $1 \leq a \leq b \leq |\sigma|$, with the addition of $\emptyset$. Obviously, for all such intervals, $[a, b] \vee [a', b'] = [\min(a, a'), \max(b, b')]$ and

$[a,b] \wedge [a',b'] = [\max(a,a'), \min(b,b')]$ with the convention that $[x,y] = \emptyset$ whenever $x > y$. Using also the fact observed at the beginning of the proof in the case that one of the considered element is the emptyset, it follows that for increasing or decreasing permutations σ, $\tilde{P}_\bullet(\sigma)$ is a lattice.

Otherwise, we consider the substitution decomposition $\pi[\alpha_1, ..., \alpha_k]$ of σ, for which it holds that $\pi \neq \sigma$. Note that in this case we can apply the induction hypothesis to π as well as to each α_i.

Let I and J be two elements of $\tilde{P}_\bullet(\sigma)$. If I or J is $\emptyset$, then $I \vee J$ and $I \wedge J$ exist from the fact noted earlier. Therefore, let us assume that $I \neq \emptyset$ and $J \neq \emptyset$.

If I and J are elements of $\tilde{P}(\pi)$, we have $I \wedge J$ and $I \vee J$ in $\tilde{P}_\bullet(\pi)$ through the induction hypothesis. Then, $I \vee J$ is unchanged in $\tilde{P}_\bullet(\sigma)$ and $I \wedge J$ also stays unchanged, unless it is the minimal element ($\emptyset$) of $\tilde{P}_\bullet(\pi)$. In this case, since there is no relation between the $\tilde{P}(\alpha_i)$, we have $I \wedge J$ is the minimal element ($\emptyset$) of $\tilde{P}_\bullet(\sigma)$.

If I and J are elements of the same subposet $\tilde{P}(\alpha_i)$, we have $I \wedge J$ and $I \vee J$ in $\tilde{P}_\bullet(\sigma)$ through the induction hypothesis similarly to the previous case.

Otherwise, I and J belong to different subposets $\tilde{P}(\alpha_i)$, and we define I_π and J_π the smallest elements of $\tilde{P}(\pi)$ that contain respectively I and J. By transitivity, we have $I \vee J = I_\pi \vee J_\pi$ which we know exists due to the cases considered earlier. As for $I \wedge J$, $I \cap J$ is empty, and thus $I \wedge J$ is the minimal element ($\emptyset$) of $\tilde{P}_\bullet(\sigma)$.

This concludes our inductive proof that for any permutation σ, $\tilde{P}_\bullet(\sigma)$ is a lattice. □

Theorem 7.2.3. *([26, Theorem 3.5]) For every permutation σ, the poset $\tilde{P}_\bullet(\sigma)$ is modular if and only if σ is a simple permutation or 1 or 12 or 21.*

While the proof of this theorem in [26] is based on a characterization of modularity by sublattice avoidance, our proof relies only on the definition of a modular lattice.

Proof. First, let σ be a permutation whose decomposition tree T has depth at least 2. Denote by $\pi[\alpha_1, ..., \alpha_k]$ the substitution decomposition of σ. Let I be an interval (of size 1) of σ corresponding to a leaf of T at maximal depth. Let i be the index such that I lies in $\tilde{P}(\alpha_i)$. Of course, I is a minimal element of $\tilde{P}(\sigma)$. Let J be another minimal element of $\tilde{P}(\sigma)$ which lies in $\tilde{P}(\alpha_j)$ for some $j \neq i$.

Then, in $\tilde{P}_\bullet(\sigma)$, $I \wedge J = \emptyset$, which they cover. Defining p_i and p_j as the maximal elements of $\tilde{P}(\alpha_i)$ and $\tilde{P}(\alpha_j)$ respectively, we have $I \vee J = p_i \vee p_j$ (as in the proof of Theorem 7.2.2). It is possible that $p_i \vee p_j$ covers p_i and p_j, and it is possible that $p_j = J$. However, because I corresponds to a leaf of depth at least 2 in T, it holds that $p_i \neq I$. Therefore $I \vee J$ does not cover I, and $\tilde{P}_\bullet(\sigma)$ cannot be modular in this case.

Now, assume that $\sigma = 12 \dots k$ or $k \dots 21$ for some $k \geq 3$. Consequently, $\tilde{P}(\sigma)$ is an argyle poset with $k \geq 3$ minimal elements. Taking $I = \{1\}$ and $J = \{k\}$, we see that they both cover their joint $\emptyset$ in $\tilde{P}_\bullet(\sigma)$. However, their meet is $[1,k]$ which does not cover them due to the argyle structure itself.

We are left with the cases $\sigma = 1, 12, 21$ or σ is simple. In such cases, denoting $n = |\sigma|$, observe that all elements in $\tilde{P}_\bullet(\sigma)$ are either $\emptyset$, $[1,n]$, or cover the former while being covered by the latter. Therefore, $\tilde{P}_\bullet(\sigma)$ is modular, concluding the proof. □

Finally, in [26, Theorem 3.11] it is shown that $\tilde{P}_\bullet(\sigma)$ is distributive if and only if σ is 1 or 12 or 21. For this particular statement, decomposition trees do not allow for a proof substantially different from [26], so we leave this property outside of the present work.

7.3 Enumerative properties of interval posets

Proposition 7.1.3 allows us to identify interval posets with trees in a certain family. Specifically, the following holds.

Corollary 7.3.1. *Interval posets with n minimal elements are in bijection with trees of the form $sk(T)$ for T a decomposition tree with n leaves.*

This perspective on interval posets is very useful to derive enumeration results on interval posets, using classical tools from tree enumeration.

7.3.1 The number of realizers of a given interval poset

Section 5 of [26] is interested in computing the number of realizers of a given interval poset. The statement proved in [26] can be rephrased in terms of decomposition trees, and we state it here for completeness. The proof is straightforward from the results of our Section 7.1, and this is essentially how the statement is proved in [26] (although the language is a little bit different). Therefore, the statement is given without proof here.

Theorem 7.3.2. *([26, Theorem 5.1]) Let P be an interval poset. Denote by $rl(P)$ the number of realizers of P, that is to say the number of permutations σ such that $P = \tilde{P}(\sigma)$.*

Let σ be one permutation such that $P = \tilde{P}(\sigma)$, and let T be the skeleton of the decomposition tree of σ. Then

$$rl(P) = \prod_{v \text{ non-leaf vertex of } T} rl(v)^{\varepsilon_v},$$

where the $rl(v)$ are given by

$$rl(v) = \begin{cases} 2 \text{ if } v \text{ is linear,} \\ \text{number of simple permutations of size } k \text{ if } v \text{ is prime with } k \text{ children,} \end{cases}$$

and the exponents ε_v are given by

$$\varepsilon_v = \begin{cases} 1 \text{ if } v \text{ is the root of } T, \\ 1 \text{ if } v \text{ is prime,} \\ 1 \text{ if } v \text{ is linear with a prime parent,} \\ 0 \text{ if } v \text{ is linear with a linear parent.} \end{cases}$$

Equivalently, ε_v is 0 if v is linear with a linear parent, 1 otherwise.

Note that the last case in the definition of ε_v echoes the fact that there are no edges between two linear nodes with the same $\oplus$ or $\ominus$ labels in decomposition trees, leaving therefore no choice for the label of a linear node whose parent is also linear.

7.3.2 Interval posets with exactly two realizers

In Question 7.2 of [26], B. Tenner asked the following: how many interval posets have exactly two realizers? We solve this question, and give a precise description of these interval posets. To state our results, we need some terminology.

A permutation is *separable* if it avoids the patterns 2413 and 3142. Separable permutations enjoy several other characterizations, including some which are more adapted for our purpose. Specifically, a permutation is separable if and only if it can be obtained from permutations of

size 1 by repeated applications of the operations $\oplus$ and $\ominus$. This is essentially how separable permutations where first considered in [6], by means of their "separating trees". With the point of view of substitution decomposition used throughout this paper, the characterization of [6] is then equivalent to saying that a permutation is separable if and only if its decomposition tree contains only $\oplus$ and $\ominus$ nodes.

Separable permutations made an appearance in [26], where Theorem 6.2 states that an interval poset $P(\sigma)$ is binary if and only if σ is separable (an interval poset being by definition binary when it does not contain any dual claw with more than two minimal elements). Since separable permutations are stable by taking the inverse, it follows from Proposition 7.0.3 that also $\tilde{P}(\sigma)$ is binary if and only if σ is separable.

The answer to Question 7.2 of [26] actually also involves separable permutations.

Theorem 7.3.3. *An interval poset $P = \tilde{P}(\sigma)$ is such that P has exactly two realizers if and only if σ is a separable permutation of size at least 2, or $\sigma = 2413$ or 3142.*

As a consequence, the number a_n of interval posets with exactly two realizers is given by the sequence $a_1 = 0$, $a_2 = s_2 = 1$, $a_3 = s_3 = 3$, $a_4 = s_4 + 1 = 12$ and $a_n = s_n$ for all $n \geq 5$, with s_n the n-th little Schröder number (see [22, sequence A001003]).

Proof. Consider an interval poset $P = \tilde{P}(\sigma)$. We also denote by T the decomposition tree of σ.

Of course, if $\sigma = 1$, then 1 is the only realizer of P.

Since there are more than two simple permutations of any size at least 5, if T contains a prime node with at least 5 children, then P has more than two realizers.

If T contains a prime node v with 4 children, then we have two choices for the simple permutation labeling v, namely 2413 and 3142. This shows that P has two realizers when $\sigma = 2413$ or 3142. However, if T contains some other internal node $u \neq v$, then we can also choose (among at least two possibilities) the label of u (may u be prime or linear). Therefore, in such cases, P has more than two realizers.

We are left with the case where $\sigma \neq 1$ is separable, corresponding to $T \neq \bullet$ containing only linear nodes. The skeleton of T together with the $\oplus$ or $\ominus$ label of the root determines T (and hence σ) completely, because $\oplus$ and $\ominus$ labels are required to alternate in decomposition trees. Therefore P has exactly two realizers: σ and the permutation whose decomposition tree is T with all $\oplus$ changed into $\ominus$ and conversely (which is none other than the complement of π – see the proof of Corollary 7.1.4, third item).

We now turn to the proof of the second part of our statement. The fact that $a_1 = 0$ follows from the particular case of $\sigma = 1$ above. Otherwise, except when $n = 4$, a_n is half the number of separable permutations of size n. The number of separable permutations of size n is the n-th large Schröder number, whose half is called little Schröder number. In the particular case $n = 4$, we need to add 1 to s_n, to account for the dual claw poset with 4 minimal elements, whose two realizers are 2413 and 3142. □

7.3.3 Counting interval posets

Although the question of counting interval posets with n minimal elements was neither studied nor posed in [26], we believe it is natural. We answer this question completely in this subsection.

Let $\mathcal{P}$ be the family of rooted plane trees, where internal nodes carry a type which is either prime or linear, in which the size is defined as the number of leaves, and in which the number of children of any linear (resp. prime) node is at least 2 (resp. at least 4). Let $\mathcal{P}_n$ be the set of trees of size n in $\mathcal{P}$. Clearly, $\mathcal{P}_n$ is the set of skeletons of decomposition trees of permutations of size n, and from Corollary 7.3.1 we can identify $\mathcal{P}_n$ with the set of interval posets with n minimal elements.

We now use the approach of symbolic combinatorics (see [15, Part A] for example) to obtain the enumeration of trees in $\mathcal{P}$, or equivalently of interval posets.

By definition of $\mathcal{P}$, it follows that $\mathcal{P}$ satisfies the following combinatorial specification, where $\bullet$ denotes a leaf, $\uplus$ is the disjoint union, and $Seq_{\geq k}$ is the sequence operator restricted to sequences of at least k components:

$$\mathcal{P} = \bullet \uplus Seq_{\geq 2}(\mathcal{P}) \uplus Seq_{\geq 4}(\mathcal{P}).$$

Indeed, the first term corresponds to the tree consisting of a single leaf, the second to trees with a linear root, and the third to trees with a prime root.

Denoting p_n the cardinality of $\mathcal{P}_n$, we let $P(z) = \sum_{n\geq 0} p_n z^n$ be the ordinary generating function of $\mathcal{P}$. The combinatorial specification above indicates that $P(z)$ satisfies the following equation:

$$P(z) = z + \frac{P(z)^2}{1-P(z)} + \frac{P(z)^4}{1-P(z)}. \tag{7.1}$$

Equivalently, this can be rewritten as

$$P(z) = z\phi(P(z)) \text{ with } \phi(u) = \frac{1}{1-u\left(\frac{1+u^2}{1-u}\right)}. \tag{7.2}$$

From there, the Lagrange inversion formula (see, *e.g.* [15, Theorem A.2]) can be applied to obtain an explicit formula for p_n.

Theorem 7.3.4. *The number p_n of interval posets with n minimal elements is*

$$p_n = \begin{cases} 1 & \text{if } n = 1, \\ \frac{1}{n}\displaystyle\sum_{i=1}^{n-1} \sum_{k=0}^{\min\{i, \frac{n-i-1}{2}\}} \binom{n+i-1}{i}\binom{i}{k}\binom{n-2k-2}{i-1} & \text{if } n > 1. \end{cases}$$

The first terms of this sequence (starting from p_1) are $1, 1, 3, 12, 52, 240, 1160, 5795, 29681$. We contributed this sequence to the OEIS [22], where it is now sequence A348479.

Proof. Applying the Lagrange inversion theorem to Eq. (7.2), we have $p_n = \frac{1}{n}[u^{n-1}]\phi(u)^n$. In our computation of $\phi(u)^n$, we make use of the following identity, valid for any $n \geq 1$:

$$\left(\frac{1}{1-z}\right)^n = \sum_{i\geq 0} \binom{n+i-1}{i} z^i. \tag{7.3}$$

We derive

$$\begin{aligned}
\phi(u)^n &= \left(\frac{1}{1-u\left(\frac{1+u^2}{1-u}\right)}\right)^n = \sum_{i\geq 0}\binom{n+i-1}{i}u^i\left(\frac{1+u^2}{1-u}\right)^i \\
&= \sum_{i\geq 0}\binom{n+i-1}{i}u^i(1+u^2)^i\left(\frac{1}{1-u}\right)^i \\
&= \sum_{i\geq 0}\binom{n+i-1}{i}u^i\sum_{k=0}^{i}\binom{i}{k}u^{2k}\left(\frac{1}{1-u}\right)^i \\
&= 1 + \sum_{i>0}\binom{n+i-1}{i}u^i\sum_{k=0}^{i}\binom{i}{k}u^{2k}\sum_{j\geq 0}\binom{i+j-1}{j}u^j \qquad (7.4) \\
&= 1 + \sum_{i>0}\sum_{k=0}^{i}\sum_{j\geq 0}\binom{n+i-1}{i}\binom{i}{k}\binom{i+j-1}{j}u^{i+2k+j}. \qquad (7.5)
\end{aligned}$$

The reason why we isolate the term for $i = 0$ in Eq. (7.4) is in order to apply Eq. (7.3) with a positive power of $\frac{1}{1-u}$.

We now want to compute $[u^{n-1}]\phi(u)^n$. Since $p_1 = 1$ is obvious, we can assume $n > 1$. The exponent of u in Eq. (7.5) is $i + 2k + j$, so we want $n - 1 = i + 2k + j$, *i.e.*, $j = n - i - 2k - 1$. Since $j \geq 0$, i cannot be greater than $n - 1$, while k cannot be greater than $\frac{n-i-1}{2}$. Since k is also at most i, we have $k \leq \min\{i, \frac{n-i-1}{2}\}$. Therefore

$$p_n = \frac{1}{n}[u^{n-1}]\phi(u)^n = \frac{1}{n}\sum_{i=1}^{n-1}\sum_{k=0}^{\min\{i,\frac{n-i-1}{2}\}}\binom{n+i-1}{i}\binom{i}{k}\binom{n-2k-2}{n-i-2k-1}.$$

To conclude the proof we just note that $\binom{n-2k-2}{n-i-2k-1} = \binom{n-2k-2}{i-1}$. □

From Eq. (7.1), applying the methods of analytic combinatorics (see [15, Part B]), we can also derive the asymptotic behavior of p_n.

Theorem 7.3.5. *Let Λ be the function defined by $\Lambda(u) = \frac{u^2+u^4}{1-u}$.*

The radius of convergence ρ of the generating function $P(z)$ of interval posets is given by $\rho = \tau - \Lambda(\tau)$, where τ is the unique solution of $\Lambda'(u) = 1$ such that $\tau \in (0, 1)$.

The behavior of $P(z)$ near ρ is given by

$$P(z) = \tau - \sqrt{\frac{2\rho}{\Lambda''(\tau)}}\sqrt{1 - \frac{z}{\rho}} + \mathcal{O}\left(1 - \frac{z}{\rho}\right).$$

Numerically, we have $\tau \approx 0.2708$, $\rho \approx 0.1629$, $\sqrt{\frac{2\rho}{\Lambda''(\tau)}} \approx 0.2206$.

As a consequence, the number p_n of interval posets with n minimal elements satisfies, as $n \to \infty$,

$$p_n \sim \sqrt{\frac{\rho}{2\pi\Lambda''(\tau)}}\frac{\rho^{-n}}{n^{3/2}}.$$

Numerically, we have $\sqrt{\frac{\rho}{2\pi\Lambda''(\tau)}} \approx 0.0622$, $\rho^{-1} \approx 6.1403$.

Proof. To prove this theorem we just need to prove that Λ satisfies the hypothesis of [8, Theorem 1], which is an adaptation of [15, Proposition IV.5 and Theorem VI.6] to the setting where trees are counted by the number of *leaves* (as opposed to the more classical counting by the number of *nodes*). Specifically, we can immediately see from their definitions that Λ is analytic at 0, has non-negative Taylor coefficients, and has radius of convergence 1, and that $P(z)$ is aperiodic. Finally, since $\lim_{u\to 1}\Lambda'(u) = +\infty > 1$, the result follows immediately from [8, Theorem 1]. □

We note that the classical theorems [15, Proposition IV.5 and Theorem VI.6] could also have been applied to obtain Theorem 7.3.5, starting from Eq. (7.2) instead of Eq. (7.1). (And we can check that both approaches indeed yield the same results.)

Remark 7.3.6. *Remember from Definition 7.0.1 that interval posets are defined as* plane embeddings *of some posets. It would also make sense to consider* non-plane *versions of these posets, defining a non-plane interval poset with n minimal elements simply as the poset of the non-empty intervals of a permutation of size n. Following the same approach to put in correspondence interval posets with trees, it follows that the family $\mathcal{Q}$ of non-plane interval posets satisfies the following combinatorial specification, where $MSet_{\geq k}$ is the multi-set operator restricted to multi-sets of at least k components:*

$$\mathcal{Q} = \bullet \uplus MSet_{\geq 2}(\mathcal{Q}) \uplus MSet_{\geq 4}(\mathcal{Q}).$$

While this specification can be translated on generating functions, the resulting equation involves Pólya operators, making its resolution much harder, even if just numerically. Nevertheless, from this equation, it can be proved that the sequence (q_n) enumerating non-plane interval posets behaves asymptotically like $\alpha n^{-3/2}\beta^n$, following the approach of [15, Section VII.5] or [19]. Iterating the equation for the generating function, we computed the first 400 terms of the sequence (q_n), which allowed to find loose numerical estimates for α and β as $\tilde{\alpha} = 0.1964$ and $\tilde{\beta} = 3.7545$.

7.3.4 Counting tree interval posets

A *tree interval poset* is an interval poset which is a tree. Of course, this definition applies to interval posets $P(\sigma)$ (or $\tilde{P}(\sigma)$) since intervals posets $P_\bullet(\sigma)$ (or $\tilde{P}_\bullet(\sigma)$) are never trees.

Put in our language, [26, Theorem 6.1] characterizes the tree interval posets as the $\tilde{P}(\sigma)$ such that the substitution decomposition of σ does not involve any $\oplus$ or $\ominus$ with at least three components. This result can be recovered from the procedure described in Section 7.1, where we can see that an interval poset $\tilde{P}(\sigma)$ is a tree if and only if the decomposition tree of σ has no linear node with more than two children. Indeed, every internal node of the tree is substituted with a dual claw or an argyle poset, and the resulting poset is a tree if and only if the substituted posets are themselves trees. Now, among the posets which can be substituted, the only ones that are not trees are the argyle posets with more that two minimal elements. Consequently, for $\tilde{P}(\sigma)$ to be a tree, the decomposition tree of σ must be free of linear nodes with more than two children.

In [26, Question 7.1], B. Tenner also asks how many tree interval posets have n minimal elements. We solve this question using the same techniques as the previous subsection, giving a closed formula for the number of tree interval posets with n minimal elements and its asymptotic behavior.

Let $\mathcal{T}$ be the family of rooted plane trees, where internal nodes carry a type which is either prime or linear, in which the size is defined as the number of leaves, and in which the number of children of any linear node is exactly 2, while the number of children of any prime node is at least 4. Let $\mathcal{T}_n$ be the set of trees of size n in $\mathcal{T}$. Clearly, $\mathcal{T}_n$ is the set of skeletons of decomposition trees of permutations of size n whose interval poset is a tree, and we can identify $\mathcal{T}_n$ with the set of tree interval posets with n minimal elements.

We now enumerate trees in $\mathcal{T}$ using the approach of symbolic combinatorics in the same fashion as we did to enumerate trees in $\mathcal{P}$.

Like in the previous subsection, we derive that $\mathcal{T}$ satisfies the following combinatorial specification, where $\bullet$ denotes a leaf, $\uplus$ is the disjoint union, $\times$ is the Cartesian product, and $Seq_{\geq k}$ is the sequence operator restricted to sequences of at least k components:

$$\mathcal{T} = \bullet \uplus (\mathcal{T} \times \mathcal{T}) \uplus Seq_{\geq 4}(\mathcal{T}).$$

Denoting t_n the cardinality of $\mathcal{T}_n$, we let $T(z) = \sum_{n\geq 0} t_n z^n$ be the ordinary generating function of $\mathcal{T}$. The combinatorial specification above indicates that $T(z)$ satisfies the following equation:

$$T(z) = z + T(z)^2 + \frac{T(z)^4}{1 - T(z)}. \tag{7.6}$$

Equivalently, this can be rewritten as

$$T(z) = z\psi(T(z)) \text{ with } \psi(u) = \frac{1}{1 - u\left(1 + \frac{u^2}{1-u}\right)}. \tag{7.7}$$

From there, we apply again the Lagrange inversion formula to obtain an explicit formula for t_n.

Theorem 7.3.7. *The number t_n of tree interval posets with n minimal elements is*

$$t_n = \begin{cases} 1 & \text{if } n = 1, \\ \frac{1}{n}\left[\sum_{i=1}^{n-3} \sum_{k=1}^{\min\{i, \frac{n-i-1}{2}\}} \binom{n+i-1}{i}\binom{i}{k}\binom{n-i-k-2}{k-1} + \binom{2n-2}{n-1}\right] & \text{if } n > 1. \end{cases}$$

The first terms of this sequence (starting from t_1) are $1, 1, 2, 6, 21, 78, 301, 1198, 4888$. This is sequence A054515 in the OEIS [22].

Proof. The proof follows the same steps as that of Theorem 7.3.4. Applying the Lagrange inversion theorem to Eq. (7.7), we have $t_n = \frac{1}{n}[u^{n-1}]\psi(u)^n$. In our computation of $\psi(u)^n$, we make use again of the identity expressed in Eq. (7.3), valid for any $n \geq 1$.

We derive

$$\begin{aligned} \psi(u)^n &= \left(\frac{1}{1 - u\left(1 + \frac{u^2}{1-u}\right)}\right)^n = \sum_{i\geq 0}\binom{n+i-1}{i}u^i\left(1 + \frac{u^2}{1-u}\right)^i \\ &= \sum_{i\geq 0}\binom{n+i-1}{i}u^i\left(1 + \sum_{k=1}^{i}\binom{i}{k}\left(\frac{u^2}{1-u}\right)^k\right) \qquad (7.8) \\ &= \sum_{i\geq 0}\binom{n+i-1}{i}u^i\left(1 + \sum_{k=1}^{i}\binom{i}{k}u^{2k}\sum_{j\geq 0}\binom{k+j-1}{j}u^j\right) \\ &= \sum_{i\geq 1}\sum_{k=1}^{i}\sum_{j\geq 0}\binom{n+i-1}{i}\binom{i}{k}\binom{k+j-1}{j}u^{i+2k+j} + \sum_{i\geq 0}\binom{n+i-1}{i}u^i. \qquad (7.9) \end{aligned}$$

Note that we isolated the term for $k = 0$ in Eq. (7.8), in order to apply Eq. (7.3) with a positive power of $\frac{1}{1-u}$.

We now want to compute $[u^{n-1}]\psi(u)^n$. Since $t_1 = 1$ is obvious, we can assume $n > 1$. The exponent of u in the first term of Eq. (7.9) is $i + 2k + j$, so we want $n - 1 = i + 2k + j$, *i.e.*, $j = n - i - 2k - 1$. Since $j \geq 0$ and $k \geq 1$, i cannot be greater than $n - 3$, while k cannot be greater than $\frac{n-i-1}{2}$. Since k is also at most i, we have $k \leq \min\{i, \frac{n-i-1}{2}\}$. On the other hand, the coefficient of u^{n-1} is the second term of Eq. (7.9) is $\binom{2n-2}{n-1}$. Therefore $t_n = \frac{1}{n}[u^{n-1}]\psi(u)^n$ gives

$$t_n = \frac{1}{n}\left[\sum_{i=1}^{n-3}\sum_{k=1}^{\min\{i, \frac{n-i-1}{2}\}}\binom{n+i-1}{i}\binom{i}{k}\binom{n-i-k-2}{n-i-2k-1} + \binom{2n-2}{n-1}\right].$$

To conclude the proof we just note that $\binom{n-i-k-2}{n-i-2k-1} = \binom{n-i-k-2}{k-1}$. □

As in the previous subsection, we can obtain the asymptotic behavior of t_n using analytic combinatorics, either from Eq. (7.6) (which is the version we present) or from Eq. (7.7). The following can be proved exactly like Theorem 7.3.5.

Theorem 7.3.8. *Let Λ be the function defined by $\Lambda(u) = u^2 + \frac{u^4}{1-u}$.*

The radius of convergence ρ of the generating function $T(z)$ of tree interval posets is given by $\rho = \tau - \Lambda(\tau)$, where τ is the unique solution of $\Lambda'(u) = 1$ such that $\tau \in (0, 1)$.

The behavior of $T(z)$ near ρ is given by

$$T(z) = \tau - \sqrt{\frac{2\rho}{\Lambda''(\tau)}}\sqrt{1-\frac{z}{\rho}} + \mathcal{O}\left(1-\frac{z}{\rho}\right).$$

Numerically, we have $\tau \approx 0.3501$, $\rho \approx 0.2044$, $\sqrt{\frac{2\rho}{\Lambda''(\tau)}} \approx 0.2808$.

As a consequence, the number t_n of tree interval posets with n minimal elements satisfies, as $n \to \infty$,

$$t_n \sim \sqrt{\frac{\rho}{2\pi\Lambda''(\tau)}}\frac{\rho^{-n}}{n^{3/2}}.$$

Numerically, we have $\sqrt{\frac{\rho}{2\pi\Lambda''(\tau)}} \approx 0.0792$, $\rho^{-1} \approx 4.8920$.

Remark 7.3.9. *Like in Remark 7.3.6, we can find an equation for the generating function of non-plane interval posets which are trees. And similarly, we can deduce from this equation that the asymptotic behavior of the sequence enumerating these objects is of the form $\alpha n^{-3/2}\beta^n$. Here, the loose numerical estimates for α and β which we obtain in the same fashion as in Remark 7.3.6 are $\tilde{\alpha} = 0.2597$ and $\tilde{\beta} = 2.9784$.*

7.4 The Möbius function on interval posets

In this section we will calculate the Möbius function on interval posets $\tilde{P}_\bullet(\sigma)$. We first recall some basic concepts. We refer the reader to [25] or [18] for details.

Definition 7.4.1. Let $(P, \leq)$ be a partially ordered set. If P has a maximum element M, then the elements covered by M are called *coatoms*.

Definition 7.4.2. Let $(P, \leq)$ be a partially ordered set and let $a, b \in P$. The *interval* $[a, b]$ of P is the set $[a, b] = \{x \in P \mid a \leq x \leq b\}$. If every interval of P is finite, then P is said to be *locally finite*.

In this paper we use the term interval to denote both the intervals of a permutation and the intervals of a poset. To avoid ambiguity, we will specify every time that we refer to the interval of a poset P by writing *interval of P*.

Definition 7.4.3. Let $(P, \leq)$ be a partially ordered set which is locally finite, and let $a, b \in P$. The *Möbius function* between a and b is recursively defined as

$$\mu_P(a,b) = \begin{cases} 1 & \text{if } a = b, \\ -\displaystyle\sum_{x:a<x\leq b} \mu_P(x,b) & \text{if } a < b, \\ 0 & \text{otherwise.} \end{cases}$$

Whenever P is clear from the context, we write just μ instead of μ_P.

Here we used the "top-down" definition of the Möbius function, but we point out that the classical definition is the (obviously equivalent) "bottom-up" definition, given by $\mu(a, b) = -\sum_{x:a\leq x<b} \mu(a, x)$, for $a < b$. For our purpose, the top-down definition is more convenient, because in $\tilde{P}_\bullet(\sigma)$ it is simpler to start from the top and recursively compute the Möbius function with the elements below.

The following lemma is an immediate consequence of [18, Lemma 10.4], and describes a simple case where the Möbius function is 0. This special case arises often in $\tilde{P}_\bullet(\sigma)$. We also present a brief, self-contained proof of the lemma.

Lemma 7.4.4 ([18]). *Let $[a,b]$ be an interval of P. If there exists an element $x \in [a,b]$, $x \neq a,b$, which is comparable with every element in the interval $[a,b]$ of P, then $\mu(a,b) = 0$.*

Proof. Let $x \in [a,b]$ be an element satisfying the properties of the statement. By definition of the Möbius function, we have $\sum_{y:x\leq y\leq b} \mu(y,b) = 0$. Consider an element c covered by x. Since x is comparable with every element of $[a,b]$, there is no other element of $[a,b]$ which covers c. This implies that $\mu(c,b) = -\sum_{y:c<y\leq b} \mu(y,b) = -\sum_{y:x\leq y\leq b} \mu(y,b) = 0$. Reasoning by induction, we can see that for every element $z \neq x$ in the interval $[a,x]$ of P it holds that $\mu(z,b) = 0$. Indeed,

$$\mu(z,b) = \sum_{y:z<y<x} \mu(y,b) + \sum_{y:x\leq y\leq b} \mu(y,b) = \sum_{y:z<y<x} 0 + 0 = 0.$$

In particular, $\mu(a,b) = 0$. □

Theorem 7.4.5. *Let σ be a permutation of size n whose substitution decomposition is $\pi[\alpha_1, ..., \alpha_k]$. For any $I \in \tilde{P}_\bullet(\sigma)$, it holds that*

$$\mu(I,[1,n]) = \begin{cases} 1 & \text{if } I = [1,n], \\ -1 & \text{if } I \text{ is covered by } [1,n] \text{ (i.e., } I \text{ is a coatom)}, \\ k-1 & \text{if } I = \emptyset \text{ and } \pi \text{ is either simple or } 12 \text{ or } 21, \\ 1 & \text{if } \pi \text{ is } 12\dots k \text{ or } k\dots 21 \text{ for some } k \geq 3 \\ & \quad \text{and } I \text{ is covered by the two coatoms of } \tilde{P}_\bullet(\sigma), \\ 0 & \text{otherwise.} \end{cases}$$

Proof. If $I = [1,n]$ then $\mu(I,[1,n]) = 1$, while if I is a coatom then $\mu(I,[1,n]) = -1$, by definition. Suppose now that I is neither $[1,n]$ nor a coatom.

We now distinguish cases according to the substitution decomposition $\pi[\alpha_1, \dots, \alpha_k]$ of σ.

If π is simple, then $\tilde{P}_\bullet(\sigma)$ is obtained by identifying the k minimal elements of the dual claw poset $\tilde{P}(\pi)$ with the maxima of $\tilde{P}(\alpha_i)$, for $1 \leq i \leq k$, and adding the minimum $\emptyset$. This is also true when $\pi = 12$ or 21, because the argyle poset with two minimal elements is equal to the dual claw poset with two minimal elements. Consider an element I of $\tilde{P}_\bullet(\sigma)$ such that I is neither $\emptyset$, nor a coatom, nor $[1,n]$. Let i be the index such that $\tilde{P}(\alpha_i)$ contains I. Note that I is not the maximum of $\tilde{P}(\alpha_i)$, since it is not a coatom. Then, $\mu(I,[1,n]) = 0$ by Lemma 7.4.4, where the element x of the lemma is the maximum of $\tilde{P}(\alpha_i)$. Finally, we consider the case $I = \emptyset$. From the above results and the definition of μ, we have

$$\begin{aligned} \mu(\emptyset,[1,n]) &= -\sum_{J\in\tilde{P}(\sigma)} \mu(J,[1,n]) = -\mu([1,n],[1,n]) - \sum_{J \text{ coatom}} \mu(J,[1,n]) \\ &= -1 - \sum_{J \text{ coatom}} (-1) = -1 + k. \end{aligned}$$

We are left with the case where π is $12\dots k$ or $k\dots 21$ for some $k \geq 3$. In this case, $\tilde{P}_\bullet(\sigma)$ is obtained by identifying the k minimal elements of the argyle poset $\tilde{P}(\pi)$ with the maxima of $\tilde{P}(\alpha_i)$, for $1 \leq i \leq k$, and adding the minimum $\emptyset$. We can easily compute the Möbius function for every element $I \in \tilde{P}(\pi)$. We have seen that $\mu(I,[1,n]) = 1$ (resp. -1) if I is $[1,n]$ (resp. a coatom). It follows that $\mu(I,[1,n]) = 1$ if I is the element covered by both coatoms, and that $\mu(I,[1,n]) = 0$ for all the others $I \in \tilde{P}(\pi)$ – see Fig. 7.4.

We now consider an element $I \in \tilde{P}(\alpha_i)$ for some i. If I is the maximum of $\tilde{P}(\alpha_i)$, then it is also a minimal element of $\tilde{P}(\pi)$, and hence $\mu(I,[1,n]) = 0$ as we have seen. Otherwise, we apply

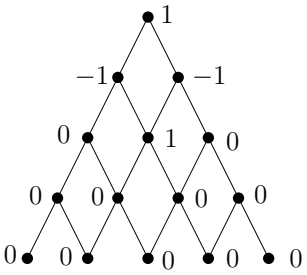


Figure 7.4: The Möbius function between the maximum and any element in an argyle poset.

again Lemma 7.4.4 (with x the maximum of $\tilde{P}(\alpha_i)$), and we obtain $\mu(I,[1,n])=0$. Finally, if $I=\emptyset$, then we have

$$\begin{aligned}\mu(\emptyset,[1,n]) &= -\sum_{J\in\tilde{P}(\sigma)}\mu(J,[1,n]) = -\mu([1,n],[1,n]) - \sum_{J \text{ coatom}}\mu(J,[1,n]) - \mu(\bar{I},[1,n]) = \\ &= -1+2-1=0,\end{aligned}$$

concluding the proof of the theorem. □

It may seem that Theorem 7.4.5 only allows to compute the Möbius function on intervals of $\tilde{P}_\bullet(\sigma)$ whose largest element is the maximum of $\tilde{P}_\bullet(\sigma)$. The following remark shows that Theorem 7.4.5 is actually easily extended to all intervals of $\tilde{P}_\bullet(\sigma)$.

Remark 7.4.6. *Let σ be a permutation and J be an element of $\tilde{P}_\bullet(\sigma)$. Define $j=|J|$. Let $\mathcal{J}$ be the subposet of $\tilde{P}_\bullet(\sigma)$ consisting of the elements in the interval $[\emptyset,J]$ of $\tilde{P}_\bullet(\sigma)$. There exists a permutation τ (of size j) such that $\tilde{P}_\bullet(\tau)$ is isomorphic to $\mathcal{J}$.*

Proof. Let $\hat{\tau}$ be the subsequence of σ composed by the elements of J. Note that the values occurring in $\hat{\tau}$ form an interval of integers (J being an interval of σ). We then define τ as the permutation obtained by rescaling $\hat{\tau}$ to the set $\{1,\dots,j\}$. Since the relative order among the elements remains unchanged, the intervals of τ correspond to the subsets of J that are intervals of σ. Therefore the poset $\tilde{P}_\bullet(\tau)$ is isomorphic to the poset $\mathcal{J}$. □

As a consequence, for any $I,J\in\tilde{P}_\bullet(\sigma)$, we can compute $\mu_{\tilde{P}_\bullet(\sigma)}(I,J)$ using the Möbius function on $\tilde{P}_\bullet(\tau)$. More precisely, letting I' be the interval obtained rescaling I by the same value that we used to rescale J into $[1,j]$, we have $\mu_{\tilde{P}_\bullet(\sigma)}(I,J)=\mu_{\tilde{P}_\bullet(\tau)}(I',[1,j])$.

For example, let $\sigma=456793128$, whose interval poset $\tilde{P}_\bullet(\sigma)$ is represented in Fig. 7.1, right. If we want to calculate $\mu_{\tilde{P}_\bullet(\sigma)}(\{5\},[4,7])$, we consider $\tau=1234$ (corresponding to $\tau'=4567$ rescaled by 3) and compute $\mu_{\tilde{P}_\bullet(\sigma)}(\{5\},[4,7])=\mu_{\tilde{P}_\bullet(\tau)}(\{2\},[1,4])=0$.

Chapter 8

Further work

In this chapter we briefly discuss some possible improvements and future work related to the results of this thesis.

In Chapter 4 we extensively studied the preimage tree of `Bubblesort`, finding results on the shape of the tree, ways to generate it, and other properties. It would be interesting to do the same for `Queuesort` and `Cons`, as it would also give some insight on the iterates of the two algorithms. For a given permutation π, what can be said about the preimage tree of π under `Queuesort`? Is it possible to find the average height of the tree, maybe even just for the identity permutation? What about the preimage tree of π under `Cons`?

In Chapter 5 we introduced the algorithms `Cons` and `Min`, and studied the preimages of `Cons`. What can be said about the preimages of `Min`? Notice that, although our explicit focus was on the preimages of a single permutation, the results of Section 5.2 can be seen from the perspective of permutation classes. Recall that the permutations sorted by one iteration of `Cons` (or `Min`) are those avoiding the patterns 321 and 2413. Therefore a permutation is sorted by two iterations of `Cons` (or `Min`) if and only if it belongs to the preimage of the class $\mathrm{Av}(321, 2413)$. Thus, Section 5.2 describes the preimages of $\mathrm{Av}(321, 2413)$ under `Cons` and `Min`. What are the preimages of the class $\mathrm{Av}(B)$ under `Cons` or `Min`, for a given set of patterns B? We believe that, when B contains a single pattern ρ, then the approach used by Magnusson [21] on `Queuesort` can be replicated for `Cons` with little effort. The algorithm `Min` might reveal to be more difficult to study, though. Finally, other algorithms can be defined, as suggested by Vince Vatter. Let d be a positive integer. Then we impose that the popqueue must be increasing and may never contain adjacent entries that differ by more than d. In this way, `Cons` and `Min` corresponds to $d = 1$ and $d = \infty$, respectively. What can we say about other values of d?

`Bubblesort` can be seen as a modification of `Queuesort`, where the queue has maximum capacity 1 (that is, it can contain at most one element at any time). Consider a queue with maximum capacity k, and call k-`Queuesort` the algorithm that sorts using that queue. What can we say about k-`Queuesort` and its preimages? If we consider a popqueue instead of a queue, can we describe an optimal sorting algorithm? The tools developed in this thesis could be helpful for tackling such problems. Specifically, the way in which a sorting algorithm behaves on the left-to-right maxima of an input permutation can be obtained using the same approach used for `Queuesort`, `Bubblesort` and `Cons`.

Acknowledgements

First of all, I would like to thank the reviewers of this thesis, Anders Claesson and Vince Vatter, for agreeing to read the thesis, for finding some mistakes and also for prompting further research directions.

La restante parte dei ringraziamente sarà in italiano, principalmente perché non tutte le persone a cui sono rivolti leggono fluentemente l'inglese.

Questa tesi raccoglie il lavoro svolto durante il dottorato, il quale è iniziato a novembre 2019. Dopo pochi mesi dall'inizio, in Italia è stato registrato il primo caso di Covid-19, che ha comportato un (relativamente lungo) lockdown seguito da anni di attenzioni ed incertezze. Personalmente non l'ho vissuta bene, sia dal punto di vista lavorativo che personale. Questo per dire che quando ringrazierò le persone che ho avuto vicino e senza le quali non sarei arrivato fino a questo punto, non le sto ringraziando solo perché senza di loro non avrei scritto questa tesi, le sto ringraziando perché ci sono state *proprio* in questi anni.

Prima di iniziare a ringraziare gente, però, ho altre due cose da dire. La prima è che comunque sono contento, quasi orgoglioso, dei risultati contenuti in questa tesi. Nonostante la premessa, quel che è venuto fuori lavorativamente non è poi così male. La seconda cosa è per coloro che ricordano i ringraziamenti della mia tesi magistrale: il "proprio" della frase precedente è evidenziato solo per far capire l'intonazione, stavolta (purtroppo) non ci sono motivi ulteriori per enfatizzare del testo.

Ah, già, un ultima cosa. Dato che per molte persone non ho una cosa singola per cui ringraziare, ho deciso di scrivere la prima cosa che mi verrà in mente, oppure quella più buffa.

Ringrazio Arianna, Ilaria, Lucia e Miller, ovvero la mia famiglia. Tra i vari motivi, vi ringrazio per la pazienza che avete, sia nei miei confronti che tra voi.

Ringrazio Vanessa, perché mi è sempre vicina, anche ora che è fisicamente lontana. Buona fortuna con il tuo dottorato, amore.

Ringrazio Tommaso, perché sì.

Ringrazio Pietro perché non mi ha (ancora) chiesto indietro il controller che mi aveva prestato. Un chiaro indice della nostra profonda e duratura amicizia. Grazie davvero. \[T]/

Un grazie a Camilla, Ettore e Giada, i miei colleghi in T1, che spero di non aver distratto troppo con le mie chiacchiere. In ogni caso, grazie per non aver acceso eccessivamente il riscaldamento quando ero in ufficio.

Ringrazio Daniele, Silvia, Tommaso e Vanessa, che ancora non hanno comprato una calcolatrice.

Ringrazio Luca, che ha acconsentito ad andare a mangiare una pizza a Chicago.

Ringrazio Mathilde per avermi prestato una bici quando sono stato a Nancy e per avermi chiesto di giocare a Pokemon Go quando eravamo a Valparaiso.

Ringrazio Emanuele, Filippo, Franceschino, Francescone e Janira. Nonostante l'impossibilità a vedersi siamo comunque riusciti a concludere la campagna, scoprendo che nonostante anche i draghi mangino carne umana, sono un po' troppo scemi per essere davvero pericolosi. Per Janira, che non ha partecipato, benvenuta e buona fortuna con le prossime avventure (sì, ce ne

saranno, siamo solo pessimi a programmare le serate).

Ora vorrei ringraziare un gruppo di persone che ho conosciuto poco dopo l'inizio del dottorato, per una serie di circostanze accidentali. Effettivamente, conoscerci è stata una coincidenza, della quale sono particolarmente felice. Aldo, Alessio, Francesco, Giuse, Noe, Soul, grazie per i pomeriggi e le serate passate a chiacchierare ed ammazzare il tempo (sappiate che ho seriamente pensato di scrivere "mandare ko Dialga", ma ho immaginato che sarebbe risultata troppo strana per essere capita immediatamente, anche da voi). In particolare, grazie Noemi per aver scritto il messaggio che ha permesso a tutti quanti di conoscerci.

Ringrazio tutte le persone di Sgabuzzini, passate, presenti e future. Siete state (ed avete contribuito a creare) un punto di ritrovo fondamentale per studenti e studentesse, tra cui io. Grazie per aver riempito le mie giornate per anni.

Bibliography

[1] M. Aigner, *Enumeration via ballot numbers*, Discrete Mathematics 308.12 (2008): 2544–2563.

[2] M. H. Albert, M. D. Atkinson, *Simple permutations and pattern restricted permutations*, Discrete Mathematics 300.1–3 (2005): 1–15.

[3] M. H. Albert, M. D. Atkinson, M. Bouvel, A. Claesson, M. Dukes, *On the inverse image of pattern classes under bubble sort*, Journal of Combinatorics 2.2 (2011): 231-243.

[4] E. Barcucci, A. Del Lungo, E. Pergola, R. Pinzani, *ECO: a methodology for the enumeration of combinatorial objects*, Journal of Difference Equations and Applications 5.4–5 (1999): 435–490.

[5] M. Bóna, *A survey of stack-sorting disciplines*, The Electronic Journal of Combinatorics 9.2 (2003): A1.

[6] P. Bose, J. Buss, A. Lubiw, *Pattern matching for permutations*, Information Processing Letters 65.5 (1998): 277–283.

[7] M. Bousquet-Mélou, *Sorted and/or sortable permutations*, Discrete Mathematics 225.1-3 (2000): 25-50.

[8] M. Bouvel, M. Mishna, C. Nicaud, *Some families of trees arising in permutation analysis*, The Electronic Journal of Combinatorics 27.2 (2020): P2.20.

[9] C. Defant, *Fertility numbers*, Journal of Combinatorics 11.3 (2020): 527–548.

[10] C. Defant, *Preimages Under the Stack-Sorting Algorithm*, Graphs and Combinatorics 33 (2017), 103–122.

[11] C. Defant, *Stack-sorting preimages of permutation classes*, Séminaire Lotharingien de Combinatoire 82 (2020): B82b.

[12] C. Defant, J. Propp, *Quantifying noninvertibility in discrete dynamical systems*, Electronic Journal of Combinatorics 27.3 (2020): P3.51.

[13] L. Ferrari, *Sorting with stacks and queues: some recent developments*, Keynote address at the on-line conference *Permutation Patterns 2021*. Available at https://www.youtube.com/watch?v=cTT9t5gddmE.

[14] L. Ferrari, E. Pergola, R. Pinzani, S. Rinaldi, *Some applications arising from the interactions between the theory of Catalan-like numbers and the ECO method*, Ars Combinatoria, 99 (2011): 109–128.

[15] P. Flajolet, R. Sedgewick, *Analytic Combinatorics*, Cambridge University Press, 2008.

[16] P. Flajolet and R. Sedgewick, *An Introduction to the Analysis of Algorithms*, Addison–Wesley, second edition, 2013.

[17] D. Foata, M. P. Schützenberger, *Théorie géométrique des polynômes Eulériens*, Lecture Notes in Mathemathics 138 (1970) Springer, Berlin.

[18] C. Godsil, *An introduction to the moebius function*, available at arXiv:1803.06664 (2018).

[19] F. Harary, R. W. Robinson, A. J. Schwenk, *Twenty-step algorithm for determining the asymptotic number of trees of various species*, Journal of the Australian Mathematical Society 20.4 (1975): 483–503.

[20] D. E. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, Addison–Wesley, third edition, 1997.

[21] H. Magnússon, *Sorting operators and their preimages*, PhD thesis, 2013.

[22] OEIS Foundation Inc. (2022), *The On-Line Encyclopedia of Integer Sequences*, published electronically at http://oeis.org

[23] V. R. Pratt, *Computing permutations with double-ended queues. Parallel stacks and parallel queues*, Proceedings of the fifth annual ACM symposium on Theory of computing (1973), 268–277.

[24] R. Tarjan, *Sorting using networks of queues and stacks*, Journal of the ACM (JACM) 19.2 (1972): 341-346.

[25] R. P. Stanley, *Enumerative Combinatorics: Volume 1.* Cambridge University Press, USA, second edition, 2011.

[26] B. E. Tenner, *Interval posets for permutations*, Order (2022), 1-14.

[27] H. Úlfarsson, A. Claesson, *Sorting and preimages of pattern classes*, Discrete Mathematics & Theoretical Computer Science (2012): 595–606.

[28] V. Vatter, *Permutation Classes*, Chapter 12 of The Handbook of Enumerative Combinatorics, pages 753–834. Chapman and Hall/CRC Press, 2015.

[29] J. West, *Permutations with forbidden subsequences, and, stack sortable permutations*, PhD thesis, Massachusetts Institute of Technology, 1990.

[30] J. West, *Sorting twice through a stack*, Theoretical Computer Science 117.1–2 (1993): 303–313.

[31] J. West, *Generating trees and forbidden subsequences*, Discrete Mathematics 157.1–3 (1996): 363–374.