

Sorting permutations using a pop stack with a bypass

Lapo Cioni^a, Luca Ferrari^{b,*}, Rebecca Smith^c

^a Dipartimento di Statistica, Informatica e Applicazioni, Università degli Studi di Firenze, Firenze, Italy

^b Dipartimento di Matematica e Informatica "U. Dini", Università degli Studi di Firenze, Firenze, Italy

^c Department of Mathematics, SUNY Brockport, Brockport, NY, USA

ARTICLE INFO

Article history:

Received 24 June 2025

Received in revised form 31 October 2025

Accepted 21 December 2025

Available online xxxx

Keywords:

Sorting algorithm

Permutation

Pop stack

Enumeration

Permutation pattern

Lattice path

ABSTRACT

We introduce a new sorting device for permutations which makes use of a pop stack augmented with a bypass operation. This results in a sorting machine, which is more powerful than the usual Popstacksort algorithm and seems to have never been investigated previously.

In the present paper, we give a characterization of sortable permutations in terms of forbidden patterns and reinterpret the resulting enumerating sequence using a class of restricted Motzkin paths. Moreover, we describe an algorithm to compute the set of all preimages of a given permutation, thanks to which we characterize permutations having a small number of preimages. Finally, we provide a full description of the preimages of principal classes of permutations, and we discuss the device consisting of two pop stacks in parallel, again with a bypass operation.

© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

Sorting devices for permutations constitute a flourishing research area in contemporary combinatorics. Starting from the stacksorting procedure introduced by Knuth [18], dozens of articles (many of which are very recent) have explored the subject. For example, machines making use of various types of containers have been studied, as well as networks of the corresponding devices. Typical containers that are considered in this context include stacks, queues, and dequeues, as well as their “pop” versions. In particular, a *pop stack* is a stack whose push and pop operations are similar to the usual ones for stacks. The distinction is that a pop operation extracts *all* the elements from the pop stack, rather than just the element on the top. The sorting power of a pop stack was originally studied by Avis and Newborn [4], who also considered pop stacks in parallel. Concerning pop stacks in series, we mention that the right-greedy version of pop stacks in series was introduced by Pudwell and Smith [21] and was successively studied also by Claesson and Guðmundsson [14].

When dealing with a classical stack, the algorithm *Stacksort* introduced by Knuth uses only two operations, namely the push operation (which pushes the current element of the input into the stack) and the pop operation (which pops the stack by moving the top of the stack into the output). Recall that permutations sortable using *Stacksort* are characterized as those avoiding the pattern 231. In this case, a bypass operation, which sends the current element of the input directly into the output, is clearly redundant, since it can be simulated by means of a push followed by a pop. This is clearly false if we replace the stack with a queue: without a bypass operation, the resulting device would be able to sort only the identity permutations, whereas adding a bypass gives rise to a much more interesting device. In the latter case, an optimal

* Corresponding author.

E-mail addresses: lapo.cioni@unifi.it (L. Cioni), luca.ferrari@unifi.it (L. Ferrari), rsmith@brockport.edu (R. Smith).

sorting algorithm, called *Queuesort*, is described in [19] and further studied in [12], and sortable permutations can be characterized as those avoiding the pattern 321.

When we move to the “pop” versions, again in the case of a pop queue a bypass operation is necessary in order to have nontrivial algorithms [13]. On the other hand, in the case of a pop stack, having or not having a bypass operation gives rise to two different (and nontrivial) sorting devices (unlike what happens for a classical stack). Curiously, there is no evidence in the literature of papers investigating the sorting properties of a pop stack with a bypass. This paper aims to fill this gap. Clearly, adding a bypass to a pop stack slightly increases the sorting power of the device and poses interesting combinatorial questions on its properties, which we analyze in some detail. More specifically, the content of our paper is the following.

In Section 3 we characterize the set of sortable permutations in terms of two forbidden patterns; the enumeration of the resulting class of pattern avoiding permutations was already known (odd-indexed Fibonacci numbers, sequence A001519 in [23]); however, in Section 4 we give an independent proof of this enumerative result by describing a bijective link with a restricted class of Motzkin paths. In Section 5 we describe an algorithm to compute the preimage of a given permutation and use it to characterize and enumerate permutations having 0, 1, or 2 preimages. Section 6 contains a complete description of the preimages of principal classes of permutations, by determining in which cases we obtain classes (and in such cases by determining the basis of the resulting classes). Section 7 is devoted to the characterization and (partly) enumeration of the sets of sortable permutations for the compositions of our sorting algorithm with other classical sorting algorithms (the characterization being given in terms of forbidden patterns). Finally, in Section 8 we consider the device consisting of two pop stacks in parallel with a bypass, and we determine the basis of the associated set of sortable permutations, and in Section 9 we give some hints for further work.

2. Preliminaries on permutations and sorting algorithms

Let $[1, n] = \{1, \dots, n\}$. A permutation π of size n is a bijection from $[1, n]$ to $[1, n]$. It can be represented in linear form as $\pi = \pi_1 \cdots \pi_n$, where $\pi_i = \pi(i)$, for all $1 \leq i \leq n$. The *identity permutation* of size n is $12 \cdots n$, and is denoted id_n . We denote by S_n the set of all permutations of size n and by $S = \bigcup_{n \in \mathbb{N}} S_n$ the set of all permutations.

For a given permutation π , we say that two elements are *adjacent* when their positions are consecutive integers, whereas we say that they are *consecutive* when their values are consecutive integers. For instance, if $\pi = 3645712$, the elements 1 and 7 are adjacent, the elements 5 and 6 are consecutive, and the elements 4 and 5 are both adjacent and consecutive. We warn the reader that we will strictly adhere to this terminology regarding the terms “consecutive” and “adjacent” throughout the paper.

Given $\pi = \pi_1 \cdots \pi_n \in S_n$, an element π_i is called a *left-to-right maximum* when it is larger than all the elements to its left (that is, $\pi_i = \max\{\pi_j \mid j \leq i\}$). We denote by $LTR(\pi)$ the set of left-to-right maxima of π . Therefore, the set of left-to-right maxima of the above permutation is $LTR(3645712) = \{3, 6, 7\}$. Moreover, the *reverse* of π is the permutation $\pi^r = \pi_n \cdots \pi_1$.

Given $\pi \in S_n$ and $\tau = \tau_1 \cdots \tau_m \in S_m$, the *direct sum* of π and τ is the permutation $\pi \oplus \tau$ obtained by concatenating π with the sequence $(\tau_1 + n) \cdots (\tau_m + n)$ (obtained from τ by adding n to each of its elements). For instance, the direct sum of the permutations 3142 and 42315 is 314286759.

We note that any sequence of n distinct integers can be regarded as a permutation by appropriately rescaling its elements to $[1, n]$. Indeed, two sequences are said to be *order-isomorphic* if their elements are in the same relative order. This notion is useful in the definition of a pattern that we give next.

Let π, ρ be two permutations of size n and k , respectively. Then ρ is a *pattern* of π (or π contains the pattern ρ) when there exists a subsequence $\pi_{i_1} \cdots \pi_{i_m}$ of π , with $i_1 \leq \cdots \leq i_m$, which is order-isomorphic to ρ . Such a subsequence is also called an *occurrence* of the pattern ρ . Conversely, if π does not contain ρ , then we say that π *avoids* ρ . The set of all permutations avoiding a pattern ρ is denoted $Av(\rho)$, and similarly, for $T \subseteq S$, we write $Av(T)$ for the set of permutations avoiding *each* pattern of T . We also use the notations $Av_n(\rho)$ and $Av_n(T)$ to denote the sets of permutations of size n avoiding ρ and T , respectively. As an example, the permutation 35142 contains the patterns 213 and 321 (among others), but avoids the pattern 123. In particular, an occurrence of the pattern 21 is called an *inversion*.

The notion of pattern defines a partial order $\leq$ on S , by declaring $\rho \leq \pi$ whenever ρ is a pattern of π . The resulting poset is called the *permutation pattern poset*, and its down-sets¹ are usually called *classes*. It is not difficult to see that any nonempty class of the permutation pattern poset can be defined in terms of the avoidance of a set of patterns. More formally, given nonempty $\mathcal{D} \subseteq S$, $\mathcal{D}$ is a class if and only if there exists an antichain $T \subseteq S$ such that $\mathcal{D} = Av(T)$. In this case, we say that T is the *basis* of the class $\mathcal{D}$.

There are also several variations of the above notion of pattern. One that is relevant for our paper (specifically in Proposition 7.1) is that of a *barred pattern*. A *barred pattern* is a permutation in which some of the entries are barred. For a permutation π to avoid the barred pattern ρ means that every set of entries of π which form a copy of the nonbarred entries of ρ can be extended to form a copy of all entries of ρ . For example, the permutation 462351 avoids the barred pattern 35241, as it has two occurrences of 3241 (namely 4251 and 4351), which can both be extended to an occurrence of 35241.

¹ A down-set of a poset P is a subset of P which is closed downwards; dually, an up-set is a subset of P which is closed upward.

To conclude this section on preliminaries, we briefly recall a few sorting algorithms for permutations that will be used in our paper. We will not give formal definitions of the algorithms, instead settling for an informal description of each of them.

The classical *Stacksort* algorithm was first introduced by Knuth [18], and tries to sort a permutation by making use of a stack as follows: the entries of the input permutation are processed from left to right and, when the current entry is smaller than the top of the stack (or the stack is empty), then such entry is pushed into the stack; otherwise the stack is popped, and the top of the stack is moved in the next available position of the output (i.e. to the immediate right of the elements already in the output). When all entries for the input permutations have been processed, the content of the stack is poured into the output. As is well known, the set of sortable permutations for *Stacksort* is the set of 231-avoiding permutations, which are counted by Catalan numbers. A similar sorting algorithm is *Queuesort* [19], where the stack is replaced with a queue and a bypass operation is also allowed. In this case, the algorithm proceeds by inserting the current entry into the queue if it is larger than the element at the back of the queue; otherwise, the current entry is compared with the element in front of the queue, and the smaller one is output. The set of sortable permutations using *Queuesort* is known to be the set of 321-avoiding permutations, which is again counted by Catalan numbers. Finally, we mention the very classical sorting algorithm *Bubblesort*, whose description can be found in most standard algorithm textbooks. In fact, our version of *Bubblesort* consists of a single pass of the algorithm reported in textbooks. In terms of *Stacksort* and *Queuesort*, *Bubblesort* can be interpreted as a special case of any of the two in which the container has depth 1. Permutations sortable through *Bubblesort* are precisely those that avoid the two patterns 231 and 321, and are counted by the sequence of the powers of two [1].

3. An optimal algorithm to sort permutations using a pop stack with a bypass

A *pop stack* is a container in which elements can be stacked on top of each other, on which two operations are defined: the *PUSH* operation, which inserts an element into the stack, and the *POP* operation, which extracts *all the elements* from the stack. The difference between a pop stack and a (classical) stack lies therefore in the way the elements are removed from the stack.

When using a stack or a pop stack to sort a permutation, the elements of the permutations are usually processed from left to right, and either the current element of the permutation is pushed into the stack, or the topmost element (or all the content) of the stack is popped into the output, again from left to right. We now introduce a new kind of pop stack, by allowing one more operation: this is the *BYPASS* operation, which takes the current element of the input permutation and moves it directly into the output, in the next available position. More formally, given a permutation $\pi = \pi_1\pi_2 \cdots \pi_n$, a pop stack with bypass has the following allowed operations:

PUSH: insert the current element of the input into the pop stack, on top of all the other elements (if there are any);

POP: remove all the elements in the pop stack, from top to bottom, sending them into the output;

BYPASS: output the current element of the input.

Our goal is to use a pop stack with bypass in order to sort permutations. It is not too difficult to realize that not every permutation can be sorted.

Proposition 3.1. *Let π be a permutation such that $231 \leq \pi$ or $4213 \leq \pi$. Then π cannot be sorted using a pop stack with bypass.*

Proof. Suppose that π contains the pattern 231, and let bca be an occurrence of 231 in π . We analyze all possible configurations that can occur during the sorting process. In order to obtain the identity permutation, neither b nor c can reach the output before a , therefore they must enter the pop stack and wait until a is the current element of the input. However, this implies that b and c are inserted into the pop stack in this order, so they will necessarily appear in the output in reverse order, which means that the output will be unsorted.

Now suppose that π contains the pattern 4213, and let $dbac$ be an occurrence of 4213 in π . Using a similar argument as before, we observe that, in order to sort the output, both d and b must enter the pop stack and wait for a to be the current element of the input. Since the *POP* operation removes all the elements currently in the pop stack, b and d will reach the output before or after c (that is, the output will contain either the subsequence bdc or the subsequence cbd). This implies that the output will not be the identity permutation. $\square$

We will now prove the converse of the above proposition, by introducing an optimal sorting algorithm, that we call PSB (an acronym for “PopStacksort with Bypass”), by which we can sort all permutations in the class $\text{Av}(231, 4213)$.

The algorithm PSB maintains elements in the pop stack only when they are consecutive in value (and increasing from top to bottom), which ensures at least that a *POP* operation is not directly responsible for the possible failure of the sorting procedure. An example of how PSB works is shown in Fig. 1, where the input permutation (on the right of the pop stack) is not sortable.

To prove that PSB actually sorts every permutation in $\text{Av}(231, 4213)$ we need some preliminary lemmas. In what follows, we will denote by $\text{PSB}(\pi)$ the output of PSB on the permutation π .

```

1  $S := \emptyset$ ;
2  $i := 1$ ;
3 while  $i \leq n$  do
4   if  $S = \emptyset$  or  $\pi_i = \text{TOP}(S) - 1$  then
5     PUSH;
6   else if  $\pi_i < \text{TOP}(S) - 1$  then
7     BYPASS;
8   else
9     POP;
10    PUSH;
11     $i := i + 1$ ;
12 POP;

```

Algorithm 1: PSB (S is the pop stack; $\text{TOP}(S)$ is the current top element of the pop stack; $\pi = \pi_1 \dots \pi_n$ is the input permutation).

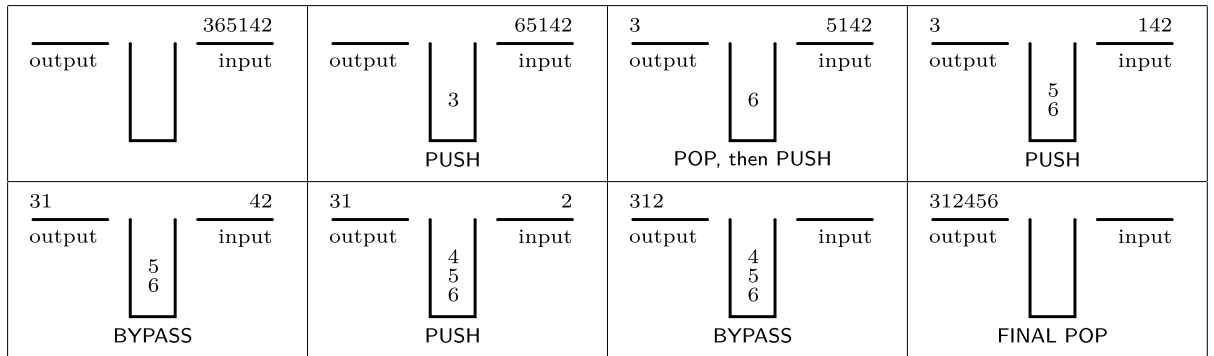


Fig. 1. Using PSB to sort 365142.

Lemma 3.2. Let a, b be two positive integers such that $a < b$, and let π be a permutation. If the pair (a, b) is a noninversion in π , then it remains a noninversion in $\text{PSB}(\pi)$ (that is, PSB does not create new inversions).

Proof. Let $a < b$, and suppose that a appears to the left of b in π . Consider the configuration in which a is the current element of the input, and suppose that a enters the pop stack (otherwise there is nothing to prove). If a is still in the pop stack when b is processed, then b is certainly larger than the top of the pop stack, and so a POP operation is performed, which is enough to ensure that a is also to the left of b in $\text{PSB}(\pi)$. $\square$

We also propose a (rather technical) generalization of the previous lemma, which is not actually needed in the proof of the next Proposition 3.5, but which instead will be extremely useful in Section 6.

Lemma 3.3. Given a permutation π , suppose that $\text{PSB}(\pi)$ contains the subsequence $m\beta$, where $m \in \mathbf{N}$ is larger than each element of the sequence β . Then $m\beta$ is also a subsequence of π .

Proof. The lemma is obvious when β is empty, so suppose that β contains at least one element. Thanks to the previous lemma, π contains the subsequence $m\tilde{\beta}$, for some rearrangement of the elements of β . Since m reaches the output before the elements of β , there must be an element $M > m$ before $\tilde{\beta}$ in π . As a consequence, we observe that, when an element of $\tilde{\beta}$ is the current element of the input, the top of the pop stack is certainly strictly larger than m (which is in turn larger than all elements of $\tilde{\beta}$), and therefore the output will also contain the sequence $\tilde{\beta}$. This means that actually $\tilde{\beta} = \beta$. $\square$

Lemma 3.4. Suppose that a is an element of a permutation π that enters the pop stack during the execution of PSB. Then there exists a (necessarily unique) nonnegative integer $k \geq 0$ such that π contains the decreasing subsequence of (not necessarily adjacent) consecutive elements $a+k, a+k-1, \dots, a$ and $a+k$ is the rightmost left-to-right maximum of π (weakly) to the left of a . In particular, when $k = 0$, we see that the left-to-right maxima of π are precisely the elements that are at the bottom of the pop stack at some point during the sorting procedure.

Proof. We argue by induction on the position of the element a in π .

If a is the first element of π , then a is a left-to-right maximum and enters the pop stack.

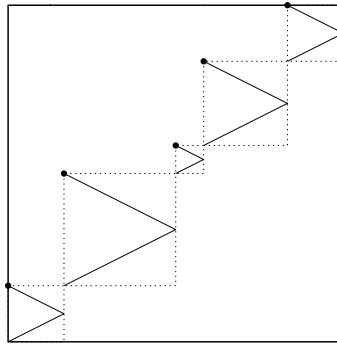


Fig. 2. The bullets indicate that each summand begins with its maximum.

Now suppose that a is the i -th element of π and that a enters the pop stack. This may happen in two distinct ways. The first case is when a is larger than the current top of the pop stack (and so it induces a POP operation). In such a case, we see that a is larger than every element inside the pop stack (since the elements in the pop stack are consecutive), and so in particular it is larger than the current element lying at the bottom of the pop stack, which (by the induction hypothesis) is a left-to-right maximum of π . Therefore, a is a left-to-right maximum of π . The second case occurs when a is pushed into the pop stack on top of a nonempty sequence of elements. In such a case, the elements in the pop stack (from top to bottom) are $a + 1, a + 2, \dots, a + k$ (for some $k \geq 1$). This means that π contains the subsequence $a + k, \dots, a + 1, a$, and that $a + k$ is a left-to-right maximum (by the induction hypothesis). Moreover, there cannot be elements larger than $a + k$ between $a + k$ and a in π , otherwise any such element would have induced a POP operation, and so $a + k$ could not be inside the pop stack when a is processed. This means that $a + k$ is the rightmost left-to-right maximum to the left of a . $\square$

Proposition 3.5. *Let $\pi \in \text{Av}(231, 4213)$. Then $\text{PSB}(\pi)$ is an identity permutation.*

Proof. Let π be a permutation such that $\text{PSB}(\pi)$ is not an identity permutation. This means that there exist two elements a, b such that $a < b$ and where b appears to the left of a in $\text{PSB}(\pi)$. We want to prove that π contains 231 or 4213.

By Lemma 3.2, the element b must occur to the left of a in π as well. We now focus on the configuration when b is sent to the output. This may happen by means of either a POP operation or a BYPASS operation.

In the former case, the element b is sent to the output from the pop stack; therefore, the current element of the input, say c , must be larger than $\text{TOP}(S)$. Since the content of the pop stack consists of consecutive elements by construction, then $b < c$, and the elements b, c, a form an occurrence of the pattern 231 in π .

In the latter case, let $c = \text{TOP}(S)$. This implies $b < c - 1$. The element $c - 1$ must appear somewhere in π . If it appears to the left of c , then $(c - 1)cb$ forms an occurrence of the pattern 231 in π . Clearly $c - 1$ cannot occur between c and b , otherwise it would be in the pop stack by Lemma 3.4. If $c - 1$ is between b and a , then $b(c - 1)a$ forms an occurrence of the pattern 231 in π . Finally, if $c - 1$ appears to the right of a in π , then $cba(c - 1)$ forms an occurrence of the pattern 4213 in π .

Therefore, if $\pi \in \text{Av}(231, 4213)$, then $\text{PSB}(\pi)$ is an identity permutation. $\square$

The class of permutations sortable using a pop stack with a bypass is, of course, a superclass of those sortable using a classical pop stack, which is $\text{Av}(231, 312)$, as shown in [4]. From a geometric point of view, the structure of permutations in the class $\text{Av}(231, 4213)$ is depicted in Fig. 2. More formally, each such permutation can be expressed in a unique way as the direct sum of permutations avoiding 231 and 213 starting with their maximum. Moreover, in each direct summand, the entries of the decreasing sequence are those entering the pop stack, whereas the entries of the increasing sequence are those that bypass the pop stack.

4. A bijective link with a class of restricted Motzkin paths

The coefficients $(|\text{Av}_n(231, 4213)|)_n$ are the odd-indexed Fibonacci numbers, that is, the sequence A001519 in [23], as shown in [3]. For our purposes, Fibonacci numbers are precisely sequence A000045 in [23], and so they are defined by the usual recurrence $F_n = F_{n-1} + F_{n-2}$ (for $n \geq 2$), with initial conditions $F_0 = 0$ and $F_1 = 1$.

It is possible to give a bijective proof of this enumerative result by providing a link with a class of restricted Motzkin paths, whose enumeration can be easily carried out using standard techniques.

To be more precise, there is a very nice bijection between the permutations sortable by PSB and a subset of ternary words that are counted by the odd-indexed Fibonacci numbers. The bijection is to take the operation(s) that each entry of each sortable permutation utilizes in PSB and convert it to a letter of the word.

For each permutation π , the *sorting word* of π will be defined entry-wise using the operation of PSB used when the entry is the next in the input to be acted on as follows:

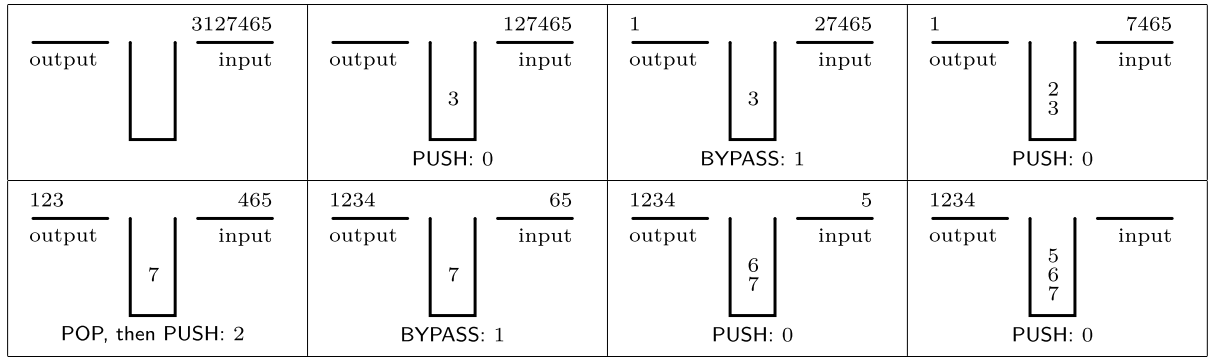


Fig. 3. Sorting 3127465 using PSB.

PUSH: 0
 BYPASS: 1
 POP, then PUSH: 2

Note that since each entry of the permutation will either be pushed into the pop stack or bypass the pop stack exactly once, the sorting word will have the same size as the sortable permutation.

For example, the permutation $\pi = 3127465$ has sorting word 0102100 as shown in Fig. 3.

Let W be the set of ternary words on $\{0, 1, 2\}$ such that each word:

1. begins with 0,
2. ends with 0 or 2, and
3. avoid the consecutive pattern 12.

Proposition 4.1. *The sorting words of permutations sortable by PSB are exactly the ternary words of W .*

Proof. To see that these conditions are necessary is a straightforward application of the algorithm PSB which prioritizes pushing entries into the pop stack over bypassing when appropriate for sorting. The first entry of the permutation must be pushed into the pop stack, so every sorting word begins with 0. The last entry cannot bypass the stack as if this would allow for sorting, the last entry must be one less than the current top entry of the pop stack or the pop stack must be empty, so the sorting word cannot end with a 1. Finally, the consecutive pattern 12 would indicate a bypass followed by a pop, which again means that the entry that bypassed the stack could have been pushed to the top of the pop stack and then popped with the rest of those entries for the same output. This proves that the set of sorting words of sortable permutations is contained in W .

Conversely, given $w \in W$, we now show that w is the sorting word of a sortable permutation by explicitly constructing the unique sortable permutation π having w as sorting word. To do so, we start by observing that w can be factorized into subwords as $w = w_1 w_2 \cdots w_k$ such that, for all $i \geq 2$, each w_i starts with 2 and (if it has size strictly greater than 1) the remaining letters are 0's and 1's, with a 0 at the end; the factor w_1 has exactly the same form, except that it starts with 0. The form of this factorization depends on the defining properties of W , in particular on the fact that there are no consecutive occurrences of 12. Denote with ℓ_i the size of w_i , for each i .

The permutation π that we are going to construct can be factorized in terms of its left-to-right maxima as $\pi = \alpha_1 \alpha_2 \cdots \alpha_k$ where, for each i , α_i starts with a left-to-right maximum. Since every left-to-right maximum (except for the first one) pops the stack, the output of PSB on π is $\mathbf{PSB}(\pi) = \mathbf{PSB}(\alpha_1) \mathbf{PSB}(\alpha_2) \cdots \mathbf{PSB}(\alpha_k)$. In view of Lemma 3.4, the set containing the initial letter 0 and each letter 2 in a sorting word correspond to the left-to-right maxima of the permutation associated with that sorting word. Thus for w to be the sorting word of π , then α_i need to have size ℓ_i , for each i . Moreover, since we require π to be sortable, each α_i must contain all the integers from $\ell_{i-1} + 1$ to ℓ_i .

The above discussion allows us to finally describe a procedure to construct π starting from w . We successively replace the letters of w with positive integers, starting from 1 and increasingly, as follows. For i running from 1 to k , scan w_i from left to right and replace each 1 with an integer (increasingly from left to right); then replace with integers (increasingly from right to left) the remaining letters of w_i . As a consequence of the above discussion, the resulting permutation π is sortable and has w as its sorting word.

For instance, starting from $w = \underbrace{0110}_{w_1} \underbrace{210}_{w_2} \underbrace{2}_{w_3} \underbrace{2010}_{w_4} \underbrace{2}_{w_5} \in W$, one obtains the permutation $\pi = \underbrace{4\ 1\ 2\ 3}_{\alpha_1} \underbrace{7\ 5\ 6}_{\alpha_2} \underbrace{8}_{\alpha_3}$
 $\underbrace{12\ 11\ 9\ 10}_{\alpha_4} \underbrace{13}_{\alpha_5}$. $\square$

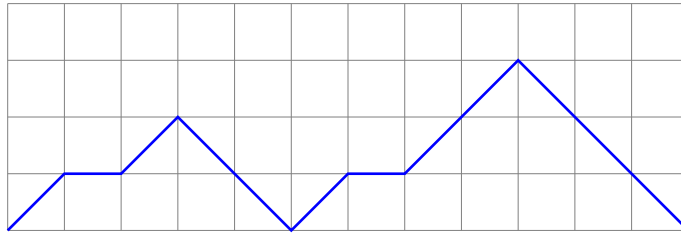


Fig. 4. The Motzkin path $UHUDDUHUUDDD$ corresponding to $\pi = 3127465$.

The permutations sortable by PSB can similarly be placed in bijection with a restricted class of Motzkin paths. Recall that a *Motzkin path* is a lattice path from $(0, 0)$ to $(0, n)$ that never goes below the x -axis which takes steps of the following types:

- Up (NE): $(1, 1)$
- Down (SE): $(1, -1)$
- Horizontal (E): $(1, 0)$

For each permutation π , the *sorting path* of π will be defined entry-wise. Specifically, for each entry π_i , use the operation of PSB utilized when π_i is the next in the input to be acted on as follows:

PUSH: Up step

BYPASS: Horizontal step

POP, then PUSH: Down steps until reaching the x -axis, followed by an Up step

Completion: Down steps until the x -axis is reached.

Note that a Motzkin path corresponding to a sorting word of size n will have exactly n total Up and Horizontal steps combined.

Returning to our previous example where $\pi = 3127465$, we can see that π has sorting path $UHUDDUHUUDDD$ shown in Fig. 4 based on its sorting sequence from PSB as was shown in Fig. 3.

Using the same methods as in Proposition 4.1, the permutations sortable by PSB can be shown to be in bijection with the set of restricted Motzkin paths M defined below.

Let M be the subset of Motzkin paths where each path $p \in M$ satisfies the following added restrictions.

1. p begins with an Up step,
2. p ends with a Down step,
3. p never has a Horizontal step immediately preceded or followed by a Down step,
4. when p has a Down step, that step is immediately followed by as many Down steps as necessary to reach the x -axis.

Proposition 4.2. *The sorting paths of permutations sortable by PSB are exactly the Motzkin paths of M .*

For enumeration purposes, recalling that $\text{Av}_n(231, 4213)$ is the set of permutations sortable by PSB of size n , denote by

W_n the corresponding set of sorting words of W size n , and

M_n the corresponding set of sorting paths of M with exactly n total Up and Horizontal steps combined.

The enumeration of the class of permutations $\text{Av}(231, 4213)$ was originally given by Atkinson [3], but given the new context that defines these same permutations as those sortable by PSB, we can use either the bijection from Proposition 4.1 or Proposition 4.2 to reconstruct this result using recursion.

Theorem 4.3. (Atkinson [3]) *The number of permutations sortable by PSB of size n is given by the odd indexed Fibonacci number F_{2n-1} where we define $F_{-1} = 1$ for consistency.*

Proof. The only Motzkin path with exactly $n = 0$ total Up and Horizontal steps is the empty path, so $|M_0| = 1 = F_{-1}$. Furthermore, the only path in M_1 is UD , so $|M_1| = 1 = F_1$. For $n \geq 2$, each Motzkin path in M_n can begin one of three ways UD , UpD , $UHH \cdots HqD$ where p, q begin with a U step and the last D shown in each prefix is the first return of the corresponding path to the x -axis, i.e. the D shown corresponds to the pop of the entry corresponding to the first U . Thus, p, q are themselves paths in M .

Removing the UD in the first case creates a bijective map between these paths of M_n and all paths of M_{n-1} . Similarly, if a path begins UpD where p begins with a U step and the last D shown in each prefix is the first return of the corresponding

path to the x -axis, removing the first U and the last D of this prefix gives a bijection between these paths of M_n and all paths of M_{n-1} . Finally, in the case of the paths that begin $UHH \cdots HqD$, if we remove the first U , the initial sequence of consecutive H steps and the last D of this prefix (so that the new prefix is q), these paths are in bijection with the Motzkin paths of $\bigcup_{i=1}^{n-2} M_i$ as q is nonempty.

By induction and some Fibonacci identities,

$$\begin{aligned} |M_n| &= 2|M_{n-1}| + \sum_{i=1}^{n-2} |M_i| \\ &= F_{2(n-1)-1} + \sum_{i=1}^{n-1} F_{2i-1} \\ &= F_{2n-3} + F_{2n-2} \\ &= F_{2n-1}. \quad \square \end{aligned}$$

Alternatively, we can see that the sorting words have the same recursive behavior or look at the regular expression for the set of sorting words. We give a proof sketch using the latter here.

Proof. The set of sorting words for sortable permutations is a regular language. In fact, it is generated by the regular expression $0(0 + 2 + 1^+0)^*$ (where, in particular, the notation 1^+ is a shortcut for 11^*). A standard application of the classical Schützenberger methodology [11] gives immediately the generating function $F(x)$ of the sequence that counts the sorting words of sortable permutations with respect to the length, which is

$$F(x) = x \cdot \frac{1}{1 - \left(2x + x \cdot \frac{x}{1-x}\right)} = \frac{x(1-x)}{1-3x+x^2}.$$

It is then easy to see that $F(x)$ is the generating function of odd-indexed Fibonacci numbers (with the correct initial conditions). $\square$

5. Preimages under PSB

In this section we investigate the preimages of a generic permutation σ under PSB. More precisely, since the algorithm PSB induces a map **PSB** from the set of all permutations to itself, we may ask to determine the set of permutations **PSB**⁻¹(σ) of all permutations whose output under PSB is σ . This kind of investigation is somehow classical for *Stacksort* [7,15], and has been recently carried out also for *Queuesort* [12] and *Bubblesort* [8].

We start by providing an alternative description of PSB that focuses on the left-to-right maxima of the input permutation. To this aim, we will use the notion of shuffle, which we define next.

Let ρ, σ be two sequences of distinct integers. We say that a sequence τ is a *shuffle* of ρ and σ if τ contains both ρ and σ as subsequences and contains no elements other than those of ρ and σ . We denote the set of all possible shuffles of ρ and σ with $\rho \sqcup \sigma$. As an example, if $\rho = 31$ and $\sigma = 24$, then the set of all shuffles of ρ and σ is $\rho \sqcup \sigma = \{3124, 3214, 3241, 2314, 2341, 2431\}$.

We can now give an alternative description of PSB.

Let $\pi \in S_n$, and let $m_1, \dots, m_k$ be its left-to-right maxima. For every $i = 1, \dots, k$, suppose that $m_i(m_i - 1) \cdots (m_i - j_i)$ is the longest decreasing subsequence of consecutive elements starting with m_i and entirely to the left of m_{i+1} (if it exists). Then we can write π as $\pi = m_1 A_1 \cdots m_k A_k$, with $A_i \in ((m_i - 1) \cdots (m_i - j_i)) \sqcup P_i$, where the P_i 's collect all the remaining elements of π . Using Lemma 3.4 and the description of PSB, we obtain **PSB**(π) = $P_1(m_1 - j_1) \cdots m_1 P_2(m_2 - j_2) \cdots m_2 \cdots P_k(m_k - j_k) \cdots m_k$.

For example, for the permutation $\pi = 635247198$, we see that $m_1 = 6$, $m_2 = 7$, $m_3 = 9$, $j_1 = 2$, $j_2 = 0$, and $j_3 = 1$. Moreover, $P_1 = 32$ and $P_2 = 1$. Furthermore, P_3 is empty, hence we obtain **PSB**(π) = 324561789. Notice that 3524 is one of the possible shuffles of 54 and 32, but there exist others, such as 3542. We remark that any shuffle of 54 and 32 (in the input permutation π) will give the same output. For instance, setting $\sigma = 635427198$, we get **PSB**(σ) = 324561789 = **PSB**(π).

The description above allows us to understand the relation between the left-to-right maxima of a permutation and those of its preimages.

Proposition 5.1. *Let π, σ be permutations such that **PSB**(π) = σ . If m is a left-to-right maximum of π , then m is a left-to-right maximum of σ as well.*

Proof. The proof is an immediate consequence of the description of PSB given above. In fact, setting $\pi = m_1 A_1 \cdots m_k A_k$ with the same notation as above, we see that, for every $i = 1, \dots, k$, the element m_i is greater than all the elements of A_i .

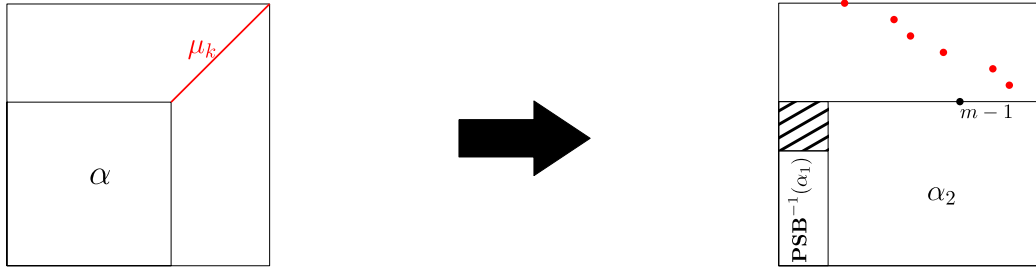


Fig. 5. Computing one preimage of $\sigma = \alpha \mu_k$ when m is the first (i.e. smallest) element of μ_k .

Since $\sigma = \mathbf{PSB}(\pi) = P_1(m_1 - j_1) \cdots m_1 P_2(m_2 - j_2) \cdots m_2 \cdots P_k(m_k - j_k) \cdots m_k$, then m_i is clearly a left-to-right maximum also of σ . $\square$

Since sequences of consecutive left-to-right maxima play an important role in the algorithm, we provide a description of the output permutation that conveniently highlights such sequences. For a given permutation σ , we write $\sigma = \mu_1 B_1 \cdots \mu_{k-1} B_{k-1} \mu_k$, where the μ_i 's are the maximal sequences of consecutive left-to-right maxima of σ , and the B_i 's (possibly empty) collect all remaining elements. For each i , we set $\mu_i = m_{i,1} \cdots m_{i,\ell_i}$.

Since PSB always outputs a permutation ending with its maximum, we will be interested exclusively in such permutations.

We now introduce a recursive algorithm that generates all preimages of a given permutation σ . We start by giving an informal description of the algorithm, in order to better understand how it works. Notice that our algorithm is actually defined on sequences of distinct integers (not necessarily permutations): this allows us to recursively execute it on subsequences of elements of a permutation.

Suppose that $\sigma = \sigma_1 \sigma_2 \cdots \sigma_n = \alpha \mu_k$, where μ_k is the maximum suffix of consecutive left-to-right maxima of σ (and α is the remaining prefix). For each entry m in μ_k , construct permutations as follows. First, remove the suffix of left-to-right maxima starting with m . Then reinsert the removed elements into the (remaining prefix of the) permutation in all possible ways, according to the following rules:

- the elements are reinserted in decreasing order;
- the maximum (i.e. n) is inserted to the immediate right of one of the remaining left-to-right maxima of σ (i.e. a left-to-right maximum to the left of m in σ) or at the beginning of σ ;
- the minimum (i.e. m) is inserted somewhere to the right of $m - 1$.

At this point, consider the prefix of all elements strictly before n and (recursively) compute all its possible preimages. Fig. 5 illustrates how the algorithm works for a specific choice of m . In the figure, the permutation $\sigma = \alpha \mu_k$ is decomposed as $\sigma = \alpha_1 \alpha_2 \mu_k$, where α_1 is the prefix of α ending with the left-to-right maximum to the right of which n is reinserted.

We illustrate the above algorithm by computing the preimages of the permutation 3154267. In this case the maximum suffix of consecutive left-to-right maxima is 67. So we have two families of preimages.

- If we remove 6 and 7, we have three possible positions where to reinsert 7 (at the beginning, right after 3 or right after 5) and, for each of them, three possible positions where to reinsert 6 (right after 5, 4 or 2). If we reinsert 7 at the beginning, we thus get the three preimages 7315642, 7315462 and 7315426. If we reinsert 7 after 3, we get the three preimages 3715642, 3715462 and 3715426. If we reinsert 7 after 5, after having computed the unique preimage of 315, which is 351, we get the three preimages 3517642, 3517462 and 3517426.
- If we remove only 7, we need to (recursively) compute all the preimages of the prefix 315426, which turns out to be just one, and append 7. In this way, we obtain the further preimage 3516427.

Proposition 5.2. *The above algorithm computes all the preimages of a given permutation.*

Proof. (Sketch.) Let $\sigma \in S_n$. It is easy to realize that all the permutations produced by the above described algorithm are indeed preimages of σ under PSB.

Conversely, let $\pi \in \mathbf{PSB}^{-1}(\sigma)$. If we decompose π as $\pi = LnR$, we observe that the elements in L are processed by PSB independently from the rest of π , hence $\mathbf{PSB}(L)$ is a prefix of σ . When n is pushed into the pop stack (clearly at the bottom of it), each of the remaining elements of π (i.e., the elements of R) is either pushed into the pop stack or bypasses it. After all elements of π are processed, the elements in the pop stack are poured into the output. From the definition of PSB it then follows that the elements appearing in the pop stack after n is processed constitute the maximal decreasing subsequence of π made of consecutive elements starting with n (call it ρ). Notice that the elements of ρ need not be adjacent in π . Notice that all the remaining elements of R appear in σ in the same order as they appear in π . In particular,

as a consequence of the maximality of ρ , the maximum of the elements of R bypassing the pop stack necessarily appears to the left of the minimum of ρ in R . Collecting all the above information tells us that π must have precisely the form prescribed by our algorithm for preimages. $\square$

In the rest of the present section we focus on the sets of permutations having a fixed number of preimages through **PSB**.

Given $n \geq 0$, let $C_n^{(k)} = \{\sigma \in S_n \mid |\mathbf{PSB}^{-1}(\sigma)| = k\}$, and $c_n^{(k)} = |C_n^{(k)}|$.

Proposition 5.3.

$$C_n^{(0)} = \{\sigma = \sigma_1 \cdots \sigma_n \mid \sigma_n \neq n\}$$

and

$$c_n^{(0)} = (n-1)(n-1)!.$$

Proof. We have already observed that any permutation in the image of **PSB** ends with its maximum. On the other hand, if a permutation σ ends with its maximum n , then by Proposition 5.2 it has at least one preimage under **PSB**. For instance, using the same notation as in the above algorithm to compute preimages, if $\sigma = \alpha\mu_k$, with $\mu_k = \mu_{k,1}\mu_{k,2} \cdots \mu_{k,\ell_k} = \mu_{k,1}\beta \neq \emptyset$ and $\beta \neq \emptyset$, then $\rho = \beta^r\alpha\mu_{k,1}$ is a preimage of σ (the case $\beta = \emptyset$ requires a little bit more care, but can still be solved using our algorithm). Therefore, a permutation has no preimages if and only if it does not end with its maximum.

The number of permutations with no preimages is thus $(n-1)(n-1)!$, since to construct such a permutation we have to choose its final element among the $n-1$ possible values, while the other elements can be arranged in all possible ways. $\square$

Proposition 5.4. For $n \geq 3$, the set $C_n^{(1)}$ consists of all permutations of size n ending with n whose left-to-right maxima are consecutive and nonadjacent. More formally,

$$C_n^{(1)} = \{\pi = \pi_1 \cdots \pi_n \mid \pi_n = n, \text{ there exists } k \geq 0 \text{ such that } LTR(\pi) = \{n-k, \dots, n\}$$

and for every $\pi_i, \pi_j \in LTR(\pi)$, $\pi_i \neq \pi_j$, we have $|j-i| > 1\}$.

Moreover

$$c_n^{(1)} = \sum_{k=2}^{\lceil \frac{n}{2} \rceil} (n-k)! \binom{n-k-1}{k-2}.$$

Proof. Let $\pi = \pi_1 \cdots \pi_n$ be a permutation such that $\pi_n = n$ and the set of its left-to-right maxima is a subsequence of consecutive and nonadjacent entries of π . Then, using our notation, we have $\pi = (n-k)B_k(n-k+1)B_{k-1} \cdots (n-1)B_1n$ (modulo a slight change in the indexing), and we know by Proposition 5.2 that the only preimage of π is $(n-k)(n-k+1)B_k(n-k+2)B_{k-1} \cdots nB_1$.

On the other hand, we now show that any permutation which has not the form described in the statement of the proposition cannot have exactly one preimage. Clearly, if π does not end with n then it has no preimages. So let $\pi = \pi_1 \cdots \pi_n$ be a permutation with $\pi_n = n$ and such that $LTR(\pi)$ is not an interval. Let $l, r \in LTR(\pi)$ such that $l+1, l+2, \dots, r-1 \notin LTR(\pi)$ and $r+1, r+2, \dots, n-1, n \in LTR(\pi)$ (in other words, l and r identify the rightmost gap in $LTR(\pi)$). Thus we can express π as $\pi = \pi_1 L l m r R_1(r+1)R_2(r+2) \cdots R_{n-r}n$, for some sequences $L, R_1, \dots, R_{n-r}$ (where, of course, the R_i 's do not contain left-to-right maxima and Ll is empty in case $\pi_1 = l$). Then two distinct preimages of π are $\pi_1 r L l m (r+1)R_1(r+2)R_2(r+3) \cdots nR_{n-r}$ and $r\pi_1 L l m (r+1)R_1(r+2)R_2(r+3) \cdots nR_{n-r}$.

The only remaining case is when $LTR(\pi) = \{n-k, \dots, n\}$ for some k , but some left-to-right maxima are adjacent. Let i be the maximum integer such that $\pi_i = n-j \in LTR(\pi)$ and $\pi_{i+1} = n-j+1 \in LTR(\pi)$. Then π can be written as $\pi = \pi_1 L(n-j)(n-j+1)R_{j-1} \cdots (n-1)R_1n$ for some sequences $L, R_{j-1}, \dots, R_1$ (and the R_i 's as above and, again, $L(n-j)$ is empty in case $\pi_1 = n-j$). Then two distinct preimages of π are $\pi_1(n-j+1)L(n-j)(n-j+2)R_{j-1} \cdots nR_1$ and $(n-j+1)\pi_1 L(n-j)(n-j+2)R_{j-1} \cdots nR_1$.

We can now prove that $c_n^{(1)} = \sum_{k=2}^{\lceil \frac{n}{2} \rceil} (n-k)! \binom{n-k-1}{k-2}$. To obtain a permutation with exactly one preimage, we need to choose k left-to-right maxima, which are forced to be the k largest elements of the permutation, as they must be consecutive (warning: this is not the same k as in the definition of $C_n^{(1)}$!). Since we require any two left-to-right maxima to be nonadjacent, then necessarily $k \leq \lceil \frac{n}{2} \rceil$. Moreover, any permutation with at least two elements and ending with its maximum has at least two left-to-right maxima, so $k \geq 2$. Once k has been chosen, we need to choose the k nonadjacent positions to place the left-to-right maxima (obviously in increasing order). Since the first and the last element of the permutations are necessarily left-to-right maxima (the last element in particular is n), the remaining $n-2$ elements consist of $k-2$ nonadjacent left-to-right maxima and $(n-2)-(k-2) = n-k$ non-left-to-right maxima. The possible ways to arrange these

elements can be counted by placing the $n - k$ non-left-to-right maxima, then choosing $k - 2$ inner spaces between them (among the $n - k - 1$ available), which can be done in $\binom{n-k-1}{k-2}$ ways. Finally, the remaining elements of the permutation are smaller than all the left-to-right maxima and can be arranged in all possible ways, which are $(n - k)!$.

Summing up, we obtain $\sum_{k=2}^{\lceil \frac{n}{2} \rceil} (n - k)! \binom{n-k-1}{k-2}$ distinct permutations. $\square$

The first terms of the sequence $c_n^{(1)}$ (starting from $n = 1$) are 1, 0, 1, 2, 8, 36, 198, ..., and are now recorded as sequence A374162 in [23].

Proposition 5.5. *For $n \geq 4$, the set $C_n^{(2)}$ consists of all permutations of size n ending with n whose left-to-right maxima are consecutive and nonadjacent except for the first one, which is required to be nonconsecutive with the second one, and can possibly be adjacent to the second one. More formally,*

$$C_n^{(2)} = \{\pi = \pi_1 \cdots \pi_n \mid \pi_n = n, \text{ there exists } k \geq 0 \text{ such that } LTR(\pi) = \{\pi_1, n - k, \dots, n\}, \\ \text{with } \pi_1 \neq n - k - 1, \text{ and for every } \pi_i, \pi_j \in LTR(\pi), \pi_i \neq \pi_j, i, j \neq 1, \text{ we have } |j - i| > 1\}.$$

Moreover

$$c_n^{(2)} = \sum_{k=3}^n \sum_{j=1}^{n-k} \frac{n - k - j + 1}{j} (n - k)! \binom{n - j - k}{k - 3}. \quad (1)$$

Proof. Let $\pi = \pi_1 \cdots \pi_n$ be a permutation such that $\pi_n = n$, there exists $k \geq 0$ such that $LTR(\pi) = \{\pi_1, n - k, \dots, n\}$, with $\pi_1 \neq n - k - 1$, and π has no adjacent left-to-right maxima, except possibly for the first two. Thus we can write π as $\pi = \pi_1 A_0(n - k) A_k(n - k + 1) A_{k-1} \cdots (n - 1) A_1 n$, and we know from Proposition 5.2 that the only two preimages of π are $\pi_1(n - k) A_0(n - k + 1) A_k(n - k + 2) A_{k-1} \cdots n A_1$ and $(n - k) \pi_1 A_0(n - k + 1) A_k(n - k + 2) A_{k-1} \cdots n A_1$.

On the other hand, we now show that any permutation having a different form cannot have exactly two preimages. We can assume that π is not as in the previous two propositions, otherwise it would have less than two preimages. Similarly to what we did in the previous proposition, let $\pi = \pi_1 \cdots \pi_n$ be a permutation with $\pi_n = n$ such that there exist $l, r \in LTR(\pi)$ that identify the rightmost gap in $LTR(\pi)$ excluding the gap identified by the first and the second left-to-right maxima. So we can write π as $\pi_1 l l M r R_1(r + 1) R_2(r + 2) \cdots R_{n-r} n$, for some sequences $L, R_1, \dots, R_{n-r}$. Notice that L may be empty, but certainly $\pi_1 \neq l$. Then π has at least three different preimages, namely $\pi_1 r l l M(r + 1) R_1(r + 2) R_2(r + 3) \cdots n R_{n-r}$, $r \pi_1 l l M(r + 1) R_1(r + 2) R_2(r + 3) \cdots n R_{n-r}$ and $\alpha r M(r + 1) R_1(r + 2) R_2(r + 3) \cdots n R_{n-r}$, where α is a preimage of the sequence $\pi_1 l l$ (this sequence has at least one preimage because it ends with its maximum).

The only remaining case occurs when $LTR(\pi) = \{\pi_1, n - k, \dots, n\}$ for some k (with $\pi_1 \neq n - k - 1$), and there is at least one pair of adjacent left-to-right maxima not involving π_1 . Let $i > 1$ be the maximum index such that $\pi_i = n - j$ and $\pi_{i+1} = n - j + 1$ are both left-to-right maxima. Then π can be written as $\pi = \pi_1 L(n - j)(n - j + 1) R_{j-1} \cdots (n - 1) R_1 n$ for some sequences $L, R_{j-1}, \dots, R_1$, with L possibly empty. Then π has at least three distinct preimages, namely $\pi_1(n - j + 1) L(n - j)(n - j + 2) R_{j-1} \cdots n R_1$, $(n - j + 1) \pi_1 L(n - j)(n - j + 2) R_{j-1} \cdots n R_1$ and $\alpha(n - j + 1)(n - j + 2) R_{j-1} \cdots n R_1$, where α is a preimage of the sequence $\pi_1 L(n - j)$ (this sequence has at least one preimage because it ends with its maximum).

We are now left to prove the enumerative formula. We choose the number k of left-to-right maxima of $\pi \in C_n^{(2)}$, and we observe that $k \geq 3$ (just as a consequence of the characterization of $C_n^{(2)}$ we have proved right above). We express π as $\pi = L(n - k + 2) R_1 \cdots (n - 1) R_{k-2} n$, where L is the longest prefix of π containing no left-to-right maxima other than π_1 . Denote with j the length of L , and observe that $1 \leq j \leq n - k$ (this is due to the fact that L does not contain the largest $k - 1$ left-to-right maxima and not even $n - k + 1$). Since $\pi_1 \leq n - k$ for the non-consecutivity condition of π_1 and $n - k + 2$, the elements of L are all smaller than $n - k + 1$. Therefore, we can choose the j elements of L in $\binom{n-k}{j}$ different ways. The maximum of the chosen elements is π_1 , while the remaining $j - 1$ elements can be permuted in all possible ways to obtain L . For the remaining part we reason in the same way as Proposition 5.4 to obtain that we have $(n - j - k + 1)! \binom{n-j-k}{k-3}$ different ways to choose a sequence with $n - j$ elements of which $k - 1$ are non-adjacent left-to-right maxima. Summing everything together, we obtain that the number of permutations with exactly two preimages is given by (1). $\square$

6. Preimages of classes

The investigation of the preimages of a sorting algorithm is not limited to preimages of a single permutation. A common task in this framework is to study the preimages of sets of permutations, typically classes of permutations (i.e. sets of permutations defined in terms of the avoidance of certain patterns). Clear evidence for this approach exists in the literature, for instance in the cases of *Stacksort* [16] (and references therein), *Queuesort* [19] and *Bubblesort* [1]. When the preimages of permutation classes turn out to be classes themselves, the best way to describe them is to find an explicit basis. In case this is not possible, some kind of algorithmic procedure to determine the elements of the preimage is a notable alternative.

In the present section, we consider the preimages of principal classes under **PSB**, and we completely determine the permutations ρ for which $\mathbf{PSB}^{-1}(\text{Av}(\rho))$ is a class. Moreover, in all cases, we explicitly describe the basis of the resulting class. Our results will be useful in the next section, where we will compose **PSB** with other sorting algorithms.

Our first result concerns the principal classes whose basis is a permutation that begins with its maximum.

Proposition 6.1. *Given $n \geq 2$, let $\rho \in S_n$ be such that $\rho = n\alpha$, for some $\alpha \in S_{n-1}$. Then $\mathbf{PSB}^{-1}(\text{Av}(\rho)) = \text{Av}(B)$, where $B = \{n(n+1)\alpha\} \cup \{(n+2)n\tau \mid \tau \in (n+1) \sqcup \alpha, \tau \neq (n+1)\alpha\}$.*

Proof. Let $\rho = n\alpha$, with $\alpha \in S_{n-1}$. We start by proving that if a permutation σ contains the pattern ρ , then for every permutation π such that $\mathbf{PSB}(\pi) = \sigma$, π contains the pattern $n(n+1)\alpha$ or the pattern $(n+2)n\tau$, for some $\tau \in (n+1) \sqcup \alpha$, $\tau \neq (n+1)\alpha$.

Let π be a permutation such that $\mathbf{PSB}(\pi) = \sigma \geq \rho$. Then σ contains an occurrence $m\beta$ of ρ , in which m is larger than every element of the sequence β . By Lemma 3.3, this implies that π contains the subsequence $m\beta$ as well. We know that, during the sorting process, m is sent to the output before the first element of β becomes the current element of the input (notice that β is not empty, because $n \geq 2$). There are two possible cases that lead to such a situation.

If m enters the pop stack, then there must exist an element M larger than m between m and β in π (otherwise the pop stack would not be popped). Thus, π contains the sequence $mM\beta$, which is an occurrence of the pattern $n(n+1)\alpha$.

On the other hand, suppose that m does not enter the pop stack. Then, when m is the current element of the input, there must be a left-to-right maximum N of π , $N > m+1$, that appears before m in π . Also, π cannot contain the subsequence $N, N-1, \dots, m+2, m+1, m$, otherwise m would be pushed into the pop stack. This implies that either there exists an element M , with $m < M < N$, which is not between N and m in π , or there exist two elements M_1, M_2 , with $m < M_1 < M_2 < N$, such that π contains the subsequence NM_1M_2m .

In the first case, we obtain that either π contains the subsequence $MN\beta$, which is an occurrence of the pattern $n(n+1)\alpha$ in π , or M is somewhere between two elements of β or to the right of β in π . In the latter case, the elements N, m, β and M form an occurrence of the pattern $(n+2)n\tau$, for some $\tau \in (n+1) \sqcup \alpha$, $\tau \neq (n+1)\alpha$.

In the second case, $M_1M_2\beta$ is an occurrence of $n(n+1)\alpha$ in π .

To conclude the proof, we show that, for every permutation π containing one of the patterns in the statement of the proposition, $\mathbf{PSB}(\pi)$ contains ρ .

Let π be a permutation that contains an occurrence $MN\beta$ of the pattern $n(n+1)\alpha$. Then $\mathbf{PSB}(\pi)$ contains the subsequence $M\beta$, since M cannot be in the pop stack after N has been processed by the algorithm (since $N > M$). Therefore, $\mathbf{PSB}(\pi)$ contains an occurrence $M\beta$ of the pattern $n\alpha$.

On the other hand, suppose that π contains an occurrence $NM\psi$ of a pattern $(n+2)n\tau$, for some $\psi \in M_1 \sqcup \beta$, $\psi \neq M_1\alpha$ (with $M < M_1 < N$). Note that M is not part of a subsequence of consecutive elements of π that begins with a left-to-right maxima, therefore, by Lemma 3.4, M does not enter the pop stack and $\mathbf{PSB}(\pi)$ contains an occurrence $M\beta$ of ρ . $\square$

Notice that, for $n = 2$ and $\alpha = 1$, the previous proposition gives $\mathbf{PSB}^{-1}(\text{Av}(21)) = \text{Av}(231, 4213)$, which agrees with our characterization of sortable permutations provided in Section 3.

We next determine the preimage of a principal class whose basis is a permutation ending with its maximum and beginning with its second maximum.

Proposition 6.2. *Given $n \geq 3$, let $\rho \in S_n$, with $\rho = (n-1)\alpha n$, for some $\alpha \in S_{n-2}$. Then $\mathbf{PSB}^{-1}(\text{Av}(\rho)) = \text{Av}(B)$, where $B = \{(n-1)n\alpha\} \cup \{(n+1)(n-1)\tau \mid \tau \in n \sqcup \alpha, \tau \neq n\alpha\}$.*

Proof. Let $\rho = (n-1)\alpha n \in S_n$. We start by proving that if a permutation σ contains the pattern ρ , then every permutation π such that $\mathbf{PSB}(\pi) = \sigma$ contains the pattern $(n-1)n\alpha$ or the pattern $(n+1)(n-1)\tau$, for some $\tau \in n \sqcup \alpha$, $\tau \neq n\alpha$.

Let π be a permutation such that $\mathbf{PSB}(\pi) = \sigma \geq \rho$. Then σ contains an occurrence $m\beta M$ of ρ , with $m < M$ and $m > b$, for every element b of the sequence β . By Lemma 3.3, this implies that π contains the subsequence $m\beta$ as well, however we do not have any information on the position of M in π , so we examine all possible cases.

If M is between m and β , then $mM\beta$ forms an occurrence of $(n-1)n\alpha$ in π .

If M appears before m in π , then M is necessarily pushed into the pop stack, while m is forced to bypass. Indeed, M and m appears in the output in reverse order, so necessarily M cannot bypass; moreover, if m were pushed into the pop stack while M is still inside, then β (which is not empty, since $n \geq 3$) would not appear in the output between M and m . This means that M and m cannot be inside the pop stack together, and M remains inside the pop stack until all the elements of β are bypassed, so also m is forced to bypass. As a consequence, by Lemma 3.4, m is not part of a sequence of consecutive elements of π starting with a left-to-right maximum. Specifically, either there exists an element M_1 , with $m < M_1 < M$, which is not between M and m in π , or there exist two elements M_1, M_2 , with $m < M_1 < M_2 < M$, such that π contains the subsequence MM_1M_2n . In the former case, either π contains the subsequence $M_1M\beta$, which is an occurrence of the pattern $(n-1)n\alpha$, or M_1 is somewhere between two elements of β or to the right of β in π . Thus the elements M, m, β and M_1 form an occurrence of the pattern $(n+1)(n-1)\tau$, for some $\tau \in n \sqcup \alpha$, $\tau \neq n\alpha$. In the latter case, $M_1M_2\beta$ is an occurrence of $(n-1)n\alpha$ in π .

Finally, we consider the case in which M appears after the first element of β in π . Then we consider m and β and argue in the same way as the proof of Proposition 6.1, by distinguishing two cases, depending on whether m enters the pop stack or not. In the former case we obtain an occurrence of $(n-1)n\alpha$ in π , whereas in the latter case we obtain either the pattern $(n-1)n\alpha$ or a sequence of elements that, together with M , forms an occurrence of a pattern $(n+1)(n-1)\tau$, for some $\tau \in n \sqcup \alpha$, $\tau \neq n\alpha$.

To conclude the proof, we show that, for every permutation π containing one of the patterns in the statement of the proposition, $\mathbf{PSB}(\pi)$ contains ρ .

Let π be a permutation containing an occurrence $mM\beta$ of the pattern $(n-1)n\alpha$. Then $\mathbf{PSB}(\pi)$ contains the subsequence $m\beta$, since m cannot be in the pop stack after M has been processed by the algorithm (because $M > m$). Since $\mathbf{PSB}$ puts the maximum N of π at the end of the output permutation, we get that $\mathbf{PSB}(\pi)$ contains an occurrence $m\beta N$ of the pattern $(n-1)\alpha n$. Notice that N might coincide with M , but it cannot be any of the elements of β , nor m .

On the other hand, suppose that π contains an occurrence $M_1 m \psi$ of a pattern $(n+1)(n-1)\tau$, for some $\psi \in M \sqcup \alpha$, $\psi \neq M\alpha$. Note that m is not part of a subsequence of consecutive elements of π starting with a left-to-right maximum, since ψ contains the element M , with $m < M < M_1$. Therefore, by Lemma 3.4, m is not pushed into the pop stack, hence $\mathbf{PSB}(\pi)$ contains the subsequence $m\beta$. Since $\mathbf{PSB}$ puts the maximum N of π at the end of the output permutation, we have that $\mathbf{PSB}(\pi)$ contains an occurrence $m\beta N$ of the pattern $(n-1)\alpha n$. Notice that N might coincide with M_1 , but it cannot be any of the elements of β , nor m or M . $\square$

We observe that for $n = 3$ and $\alpha = 1$, the previous proposition gives $\mathbf{PSB}^{-1}(\text{Av}(213)) = \text{Av}(231, 4213) = \mathbf{PSB}^{-1}(\text{Av}(21))$. This is not surprising, since $\mathbf{PSB}(\pi)$ ends with its maximum for all π . Therefore, if $\mathbf{PSB}(\pi)$ contains an inversion, then it also contains the pattern 213.

The patterns ρ of the form $n\alpha$ and $(n-1)\alpha n$ are the only ones for which $\mathbf{PSB}^{-1}(\text{Av}(\rho))$ is a class. The next three propositions are devoted to show that this is indeed the case. In all cases, in order to prove that $\mathbf{PSB}^{-1}(\text{Av}(\rho))$ is not a permutation class, we exhibit two permutations σ, π such that $\sigma \leq \pi$ and $\rho \leq \mathbf{PSB}(\sigma)$, but $\rho \not\leq \mathbf{PSB}(\pi)$.

Proposition 6.3. *Let $\rho = \alpha n \beta \in S_n$, for some nonempty sequences α, β . Then $\mathbf{PSB}^{-1}(\text{Av}(\rho))$ is not a permutation class.*

Proof. Let $\sigma = (n+1)\alpha(n+2)\beta n$ and $\pi = (n+1)(n+3)\alpha(n+2)\beta n$. Then $\sigma \leq \pi$, because we obtain σ by removing $n+3$ from π . Also, $\mathbf{PSB}(\sigma) = \alpha(n+1)\beta n(n+2)$, which contains the occurrence $\alpha(n+1)\beta$ of $\alpha n \beta$. On the other hand, $\mathbf{PSB}(\pi) = (n+1)\alpha \beta n(n+2)(n+3)$, and we now show that such a permutation does not contain ρ . Indeed, since α and β are not empty, ρ neither begins nor ends at its maximum. Thus $n+3$ cannot be part of an occurrence of ρ in $\mathbf{PSB}(\pi)$. For the same reason, neither $n+2$ nor $n+1$ can. Therefore, an occurrence of ρ in $\mathbf{PSB}(\pi)$ should be an occurrence of ρ in $\alpha \beta n$, which is impossible since ρ and $\alpha \beta n$ have the same size and are different. Therefore $\rho \not\leq \mathbf{PSB}(\pi)$. $\square$

Proposition 6.4. *Let $\rho = \alpha(n-1)\beta n \in S_n$, for some nonempty sequences α, β . Then $\mathbf{PSB}^{-1}(\text{Av}(\rho))$ is not a permutation class.*

Proof. Let $\sigma = n\alpha(n+1)\beta(n-1)$ and $\pi = n(n+2)\alpha(n+1)\beta(n-1)$. Then $\sigma \leq \pi$, because we obtain σ by removing $n+2$ from π . Also, $\mathbf{PSB}(\sigma) = \alpha n \beta(n-1)(n+1)$, which contains the occurrence $\alpha n \beta(n+1)$ of ρ . On the other hand, $\mathbf{PSB}(\pi) = n\alpha \beta(n-1)(n+1)(n+2)$, and we now show that such a permutation does not contain ρ . Indeed, if we remove both $n+1$ and $n+2$ from $\mathbf{PSB}(\pi)$, the resulting permutation (of size n) is different from ρ , therefore, to have an occurrence of ρ in $\mathbf{PSB}(\pi)$, at least one of $n+1$ and $n+2$ must be involved. Actually, we cannot choose both, since the maximum and the second maximum are not adjacent in ρ . Therefore $\mathbf{PSB}(\pi)$ contains an occurrence of ρ if and only if $\pi' = n\alpha \beta(n-1)$ contains an occurrence of $\rho' = \alpha(n-1)\beta$. Since α and β are not empty, ρ' neither begins nor ends with its maximum $n-1$. Hence n and $n-1$ cannot be part of an occurrence of ρ' in π' , but ρ' has size $n-1$ and π' has size n . Therefore, there are no occurrences of ρ' in π' , that is, $\rho \not\leq \mathbf{PSB}(\pi)$. $\square$

Proposition 6.5. *Let $\rho = \alpha(n-1)n \in S_n$, with $\alpha \in S_{n-2}$ and $n \geq 4$. Then $\mathbf{PSB}^{-1}(\text{Av}(\rho))$ is not a permutation class.*

Proof. Let $\sigma = (n-1)(n+1)n\alpha$ and $\pi = (n-1)(n+1)(n+2)n\alpha$. Then $\sigma \leq \pi$ because we obtain σ by removing $n+2$ from π . Also, $\mathbf{PSB}(\sigma) = (n-1)\alpha n(n+1)$, which contains the occurrence $\alpha n(n+1)$ of ρ . On the other hand, $\mathbf{PSB}(\pi) = (n-1)(n+1)n\alpha(n+2)$, and we now show that such a permutation does not contain ρ . Indeed, since α has at least two elements and $(n-1)(n+1)n(n+2)$ cannot be an occurrence of ρ , then a possible occurrence of ρ in $\mathbf{PSB}(\pi)$ should contain at least one element of α . Moreover, since ρ ends with its maximum, $n+2$ should belong to such an occurrence. Finally, since ρ has n elements, at least one element amongst $n-1, n+1$ and n should be selected to form our occurrence. However, this is impossible, because the three latter elements are all greater than each element of α , and ρ ends with its two largest elements. Therefore, $\rho \not\leq \mathbf{PSB}(\pi)$. $\square$

We remark that, if $\rho = 123$, we cannot use the same argument as in the above proof to conclude that $\mathbf{PSB}^{-1}(\text{Av}(\rho))$ is not a permutation class. This is however true, by choosing the two permutations $\sigma = 321$ and $\pi = 3421$.

7. Compositions with other sorting algorithms

If we execute a sorting algorithm $\mathbf{X}$ on a permutation and then execute another sorting algorithm $\mathbf{Y}$ on the output permutation, we get a new sorting algorithm, whose associated map is the *composition* of the maps associated with $\mathbf{X}$ and $\mathbf{Y}$.

In this section, we investigate the permutations that can be sorted using a composition of PSB with another sorting algorithm. Specifically, we will consider the algorithms *Queuesort*, *Stacksort* and *Bubblesort*, whose associated maps will be denoted $\mathbf{Q}$, $\mathbf{S}$ and $\mathbf{B}$, respectively. In several cases we obtain matches with sequences in [23], and we also propose some conjectures for possible further work.

We begin by considering compositions in which PSB is performed first.

Proposition 7.1. *Let $\pi \in S_n$. Then $\mathbf{S} \circ \mathbf{PSB}(\pi) = id_n$ if and only if $\pi \in \text{Av}(2341, 25314, 52314, 45231, 42531, 3\bar{5}241)$.*

Proof. The algorithm *Stacksort* sorts a permutation if and only if it belongs to the permutation class $\text{Av}(231)$. Therefore, π is not sortable by the composition $\mathbf{S} \circ \mathbf{PSB}$ if and only if $\mathbf{PSB}(\pi)$ contains the pattern 231.

We start by supposing that π is a permutation which is not sortable by $\mathbf{S} \circ \mathbf{PSB}$. Then there exist three elements a, b, c , with $a < b < c$, such that $\mathbf{PSB}(\pi)$ contains such elements in the order b, c, a . By Lemma 3.2, we have that π contains the elements a, b, c , either in the order b, c, a or in the order c, b, a .

Suppose that π contains the elements b, c, a in this order. We know that, when we apply the algorithm PSB, these elements remain in the same order. Suppose that c is pushed into the pop stack during the sorting process. Then c must reach the output before a is the current element of the input, otherwise a would bypass, thus reaching the output before c . Therefore there must be an element d between c and a in π that forces the pop stack to be emptied. Since this happens only when the current element of the input is larger than the content of the pop stack, we have $d > c$, hence π contains the pattern 2341 (given by the elements b, c, d, a). On the other hand, suppose that c is not pushed into the pop stack during the sorting process. Then it is not a left-to-right maximum, and there exists at least one element larger than c and to the left of c in π . Let e be the rightmost of such elements. Now consider the instant in which c is the current element of the input. By construction, e is in the pop stack. Since c does not enter the pop stack by hypothesis, then π does not contain the decreasing sequence $e, e-1, \dots, c+1, c$. Therefore, either there exists an element d , with $c < d < e$, which is not between e and c in π , or there exist two elements d_1, d_2 , with $c < d_1 < d_2 < e$, that appear in π in the order e, d_1, d_2, c . The first case gives one of the patterns 2341, 25314, 42531, 45231 or 52314, depending on the positions of d and e with respect to b, c , and a . The second case gives one of the patterns 2341, 45231 or 42531, depending on the positions of d_1 and d_2 with respect to b, c , and a .

On the other hand, suppose that π contains the elements c, b, a in this order, and during the sorting process b reaches the output before c . For this to happen, c must be pushed into the pop stack and stay there until b becomes the current element. Moreover, the pop stack must be emptied before a becomes the current element. This can occur only if there is an element d greater than c between b and a in π . Note that the elements c, b, d, a form an occurrence of the pattern 3241 in π . Since c must remain in the pop stack until b is the current element of the input, there cannot be any element larger than c between c and b in π . This implies that π must avoid the patterns $3\bar{5}241$ and $3\bar{4}251$. However, we can ignore the pattern $3\bar{4}251$, since an occurrence of the pattern 34251 would contain an occurrence of the pattern 2341, which is prohibited.

In order to prove the converse statement, we need to show that, if a permutation π contains one of the patterns specified in the statement, then $\mathbf{S} \circ \mathbf{PSB}(\pi)$ is not the identity permutation. Equivalently, this amounts to prove that $\mathbf{PSB}(\pi)$ contains 231. This can be done by means of a (somehow lengthy but) rather easy case-by-case analysis. $\square$

Notice that the set of permutations such that $\mathbf{S} \circ \mathbf{PSB}(\pi) = id_n$ is not a permutation class (this is due to the presence of a barred pattern among those to be avoided). This is consistent with Proposition 6.3, which asserts that $\mathbf{PSB}^{-1}(\text{Av}(231))$ is not a class. For this reason, the previous proposition needed an *ad-hoc* proof. This is not the case for *Queuesort*, which can be more easily dealt with using the tools developed in the previous section.

Proposition 7.2. *Let $\pi \in S_n$. Then $\mathbf{Q} \circ \mathbf{PSB}(\pi) = id_n$ if and only if $\pi \in \text{Av}(3421, 53241, 53214)$.*

Proof. The algorithm *Queuesort* sorts a permutation if and only if it belongs to the permutation class $\text{Av}(321)$. Therefore, the set of permutations that can be sorted by the above composition is $\mathbf{PSB}^{-1}(\text{Av}(321))$. We can thus invoke Proposition 6.1 to obtain $\mathbf{PSB}^{-1}(\text{Av}(321)) = \text{Av}(3421, 53241, 53214)$, which concludes the proof. $\square$

The first terms of the sequence enumerating the above class are 1, 2, 6, 23, 101, 480, 2400, 12434, 66142, 359112, 1981904, 11085198, ... This is sequence A218225 in [23], as it is confirmed by computing a combinatorial specification for the class using the Combinatorial Exploration Framework developed in [2,6]. This sequence is also conjectured by Burstein to enumerate different classes of pattern avoiding permutations, see [23]. In particular, one of these conjectures has been recently

proved in [10], where the authors also connect it with another conjecture on the enumeration of pattern-restricted inversion sequences [17]. Despite our efforts, we have not been able to relate our class of permutations with any of the above mentioned classes of permutations or inversion sequences.

Finally, we consider the composition of **PSB** with **B**. Since the set of permutations sortable using Bubblesort is characterized by the avoidance of the two patterns 231 and 321, we can use the previous propositions to obtain the set of sortable permutations.

Proposition 7.3. *Let $\pi \in S_n$. Then $\mathbf{B} \circ \mathbf{PSB}(\pi) = id_n$ if and only if $\pi \in \text{Av}(2341, 3421, 3241, 25314, 52314, 53214)$.*

Proof. The algorithm Bubblesort sorts a permutation if and only if it belongs to the permutation class $\text{Av}(231, 321)$. Therefore, π is not sortable by the composition of $\mathbf{B} \circ \mathbf{PSB}$ if and only if $\mathbf{PSB}(\pi)$ contains the pattern 231 or the pattern 321. Since such patterns are those considered in Proposition 7.1 and Proposition 7.2, we can intersect the corresponding preimages, that is,

$$\begin{aligned} \mathbf{PSB}^{-1}(\text{Av}(231, 321)) &= \mathbf{PSB}^{-1}(\text{Av}(231)) \cap \mathbf{PSB}^{-1}(\text{Av}(321)) \\ &= \text{Av}(2341, 25314, 52314, 45231, 42531, 3\bar{5}241) \cap \text{Av}(3421, 53241, 53214) \\ &= \text{Av}(2341, 25314, 52314, 45231, 42531, 3\bar{5}241, 3421, 53241, 53214). \end{aligned}$$

Notice that a permutation avoids the patterns 3421 and $3\bar{5}241$ if and only if it avoids the patterns 3421 and 3241. Also, the avoidance of the patterns 45231, 42531 and 53241 is implied by the avoidance of the patterns 3421 and 3241. Therefore

$$\begin{aligned} \mathbf{PSB}^{-1}(\text{Av}(231, 321)) &= \text{Av}(2341, 25314, 52314, 45231, 42531, 3\bar{5}241, 3421, 53241, 53214) \\ &= \text{Av}(2341, 3421, 3241, 25314, 52314, 53214). \quad \square \end{aligned}$$

The inverse of the above class (i.e. the class whose basis is the set of inverses of the above permutations) has a regular insertion encoding [24], thus it is possible to automatically deduce its rational generating function [2,6], which is

$$\frac{(3x-1)(x^2+2x-1)(x-1)^2}{5x^5+6x^4-24x^3+22x^2-8x+1}.$$

The associated coefficients start 1, 2, 6, 21, 76, 273, 970, 3422, 12027, 42194, 147901, 518206, ... and do not appear in [23].

We now consider compositions of **PSB** with the same algorithms as above, but this time we choose to perform **PSB** last. In our first result, to find the basis of the preimage we exploit some properties of Queuesort, for which we refer the reader to [19].

Proposition 7.4. *Let $\pi \in S_n$. Then $\mathbf{PSB} \circ \mathbf{Q}(\pi) = id_n$ if and only if $\pi \in \text{Av}(4231, 2431, 54213)$.*

Proof. The algorithm **PSB** sorts a permutation if and only if it belongs to the permutation class $\text{Av}(231, 4213)$. Therefore, π is not sortable by the composition of $\mathbf{PSB} \circ \mathbf{Q}$ if and only if $\mathbf{Q}(\pi)$ contains the pattern 231 or the pattern 4213.

Suppose that $\mathbf{Q}(\pi)$ contains the pattern 231, and let bca be one of its occurrences. Since Queuesort does not create new inversions, π contains either the sequence bca or the sequence cba .

Assume that π contains the sequence bca . If c is not a left-to-right maximum of π , then there exists an element d greater than c to the left of c in π . Therefore, π contains the pattern 4231 or the pattern 2431. If c is a left-to-right maximum, then it enters the queue during the sorting process. Since c must be popped before a is the current element, there must be an element d before a in π , such that d is larger than c but smaller than the last element e of the queue. The elements b, e, d, a form the pattern 2431 in π .

On the other hand, assume that π contains the sequence cba . Then c must be a left-to-right maximum, otherwise c and b would also form an inversion in the output. Therefore, c enters the queue and must exit before a is the current element of the input. Therefore, there must be an element d before a in π , such that d is greater than c but smaller than the last element e of the queue. The elements c, e, d, a form the pattern 2431 in π . This solves the case in which $\mathbf{Q}(\pi)$ contains the pattern 231.

Now suppose that $\mathbf{Q}(\pi)$ contains the pattern 4213, and let $dbac$ be one of its occurrences. Since Queuesort does not create new inversions, π contains the sequence $dcba$, the sequence $dbca$ or the sequence $dbac$ (and of course only one case can occur). However, notice that the first two cases cannot occur, as in each of them c is not a left-to-right maximum of π , so the inversion between c and a would remain in the output. Hence π contains the sequence $dbac$. If d is not a left-to-right maximum of π , then there exists an element e greater than d that forms the pattern 54213 together with d, b, a and c . Otherwise, d is a left-to-right maximum of π and enters the queue during the sorting of π with Queuesort. Since d goes to the output before b is considered, there exists an element e between d and b in π such that $e > d$ and e is smaller than the last element f of the queue. The elements d, f, e and b form the pattern 2431 in π . This concludes the proof. $\square$

The first terms of the associated enumerating sequence are 1, 2, 6, 22, 89, 380, 1679, 7602, 35072, 164266, 779022, 3733444, ..., and match sequence A049123 in [23], as it can be confirmed again by using the Combinatorial Exploration Framework [2,6]. To the best of our knowledge, no combinatorial interpretation for this sequence was previously known.

Proposition 7.5. *Let $\pi \in S_n$. Then $\mathbf{PSB} \circ \mathbf{B}(\pi) = \text{id}_n$ if and only if $\pi \in \text{Av}(2341, 2431, 3241, 4231, 45213, 54213)$.*

Proof. The algorithm PSB sorts a permutation if and only if it belongs to the permutation class $\text{Av}(231, 4213)$. We can use the results of [1] to find the preimages $\mathbf{B}^{-1}(\text{Av}(231))$ and $\mathbf{B}^{-1}(\text{Av}(4213))$, and then intersect the two sets. They are both permutation classes, specifically $\mathbf{B}^{-1}(\text{Av}(231)) = \text{Av}(2431, 4231, 2341, 3241)$ and $\mathbf{B}^{-1}(\text{Av}(4213)) = \text{Av}(45213, 54213)$, which gives the thesis. $\square$

Also in this case, as it happened for Proposition 7.3, the (inverse of the) above class has a regular insertion encoding, so we can compute the associated generating function, which is

$$\frac{1 - 3x}{1 - 4x + 2x^2}.$$

The sequence of the coefficients of the above generating function is (essentially) A006012 in [23]. This is a very interesting sequence that satisfies a simple linear recurrence relation, with plenty of combinatorial interpretations. The present one can thus be added to the list.

8. Pop stacks in parallel with a bypass

As with other sorting devices, we can connect multiple pop stacks with a bypass option in different ways to create more robust sorting machines. In this section, we consider one of the most natural such options by putting pop stacks in parallel where for each entry of the input, there is still an option to bypass the pop stacks and go directly to the output. Pop stacks in parallel were originally studied by Atkinson and Sack [5] who determined a recurrence relation for the enumeration of the sortable permutations in the case of two pop stacks in parallel; they also proved that the set of forbidden patterns that characterize the sortable permutations for any number k of pop stacks in parallel is finite. Atkinson and Sack [5] also conjectured that the generating function for the enumeration of sortable permutations for any value k is rational, which was proved by the last author and Vatter [22].

As before, we begin with the necessary condition that sortable permutations must avoid a certain class of permutations.

Proposition 8.1. *Let π be a permutation that contains one of the following permutations:*

$$2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153.$$

Then π cannot be sorted by two pop stacks in parallel with a bypass operation.

Proof. With each of the patterns 2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153, consider what must be done to sort a permutation that contains them. Recall that, when sorting a permutation, the entries of a pop stack must be in increasing order from top to bottom and also be such that each entry π_i is exactly one less than the entry immediately below π_i in the pop stack.

Consider an attempt to sort any of the patterns by two pop stacks in parallel with a bypass. Notice that, for any of the patterns 2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153, at least the first three entries must enter a pop stack before any can be popped, as there is a later entry in the pattern that is smaller than all three. However, we see that the second entry, say π_{α_2} , of each pattern must enter a different pop stack from that entered by the first entry, say π_{α_1} , as either $\pi_{\alpha_2} > \pi_{\alpha_1}$ (2341, 25314, 45213, 45231) or there is both a later entry, call it π_{α_k} , such that $\pi_{\alpha_2} < \pi_{\alpha_k} < \pi_{\alpha_1}$ and so π_{α_2} cannot be one less than the current top element of the pop stack π_{α_1} entered (42513, 42531, 52314, 642135, 642153). Finally, the third entry, say π_{α_3} of each pattern, can now not enter either of the pop stacks for the same reasons that π_{α_2} could not enter the same pop stack as π_{α_1} .

Hence, any algorithm will not be able to sort a permutation containing any of the patterns 2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153. $\square$

Below we give Algorithm 2, which we will denote as PSBP. Like PSB, this algorithm chooses to push an entry into a pop stack instead of bypassing the stacks if both moves are an option to sort a permutation.

We can show PSBP is optimal in the sense that it sorts all permutations sortable by the machine consisting of two pop stacks in parallel with a bypass option.

Proposition 8.2. *The set of permutations sortable by PSBP is the class*

$$\text{Av}(2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153).$$

```

1   $S_1, S_2, O := \emptyset;$ 
2   $i := 1;$ 
3  while  $i \leq n$  do
4    if  $\pi_i = \text{TOP}(S_1) - 1$  then
5       $\text{PUSH}_1;$ 
6    else if  $\pi_i = \text{TOP}(S_2) - 1$  then
7       $\text{PUSH}_2;$ 
8    else if  $S_1 = \emptyset$  then
9       $\text{PUSH}_1;$ 
10   else if  $S_2 = \emptyset$  then
11      $\text{PUSH}_2;$ 
12   else if  $\pi_i < \max(\text{TOP}(S_1) - 1, \text{TOP}(S_2) - 1)$  then
13      $\text{BYPASS};$ 
14   else
15     if  $\text{TOP}(S_1) < \text{TOP}(S_2)$  then
16        $\text{POP}_1;$ 
17        $\text{PUSH}_1;$ 
18     else
19        $\text{POP}_2;$ 
20        $\text{PUSH}_2;$ 
21 if  $S_1 = \emptyset$  then
22    $\text{POP}_2;$ 
23 else if  $S_2 = \emptyset$  then
24    $\text{POP}_1;$ 
25 else if  $\text{TOP}(S_1) < \text{TOP}(S_2)$  then
26    $\text{POP}_1;$ 
27    $\text{POP}_2;$ 
28 else
29    $\text{POP}_2;$ 
30    $\text{POP}_1;$ 

```

Algorithm 2: PSBP (S_j is pop stack j ; O is the output; $\text{TOP}(S_j)$ is the current top element of the pop stack S_j ; operations PUSH_j , POP_j , BYPASS are “insert π_i into S_j ”, “pour the whole content of S_j into the output”, “bypass the pop stacks”, respectively; $\pi = \pi_1 \cdots \pi_n$ is the input permutation).

Proof. Consider a permutation π that cannot be sorted by PSBP. Look specifically at the critical stage at which PSBP fails in the sense that an entry π_k is output before $\pi_k - 1$ has been output (we may assume that this is the first failure and thus π_k is the smallest entry to enter the output prematurely). Consider whether π_k enters the output from a pop stack or π_k passes to the output directly from the input via the bypass operation.

First suppose π_k is popped from one of the pop stacks before $\pi_k - 1$ is output. Notice that PSBP operates in such a manner that the entries of each pop stack are always increasing and in consecutive order from top to bottom. Thus π_k must be the top entry of one of the pop stacks. Say $\pi_j > \pi_k$ is the top entry of the other pop stack. Suppose the next entry from the input is π_i . The stage immediately before π_k is popped to the output is shown on in the left picture of Fig. 6. Based on the algorithm, $\pi_i > \pi_j$ or $\pi_k < \pi_i < \pi_j - 1$. Also, $\pi_k - 1$ must appear after π_i in π .

In the instance $\pi_i > \pi_j$, consider whether π_k or π_j appeared earlier in π . If $k < j < i$, then π contains a pattern $\pi_k \pi_j \pi_i (\pi_k - 1)$. That is, π contains 2341. Otherwise if $j < k < i$, then the entry $\pi_j - 1$ must appear before π_j in π or after π_k (and thus after π_i as well) for $\pi_j - 1$ to not be directly above π_j in the pop stack. Therefore, π contains a pattern $(\pi_j - 1) \pi_j \pi_i (\pi_k - 1)$ in the former case or one of the patterns $\pi_j \pi_k \pi_i (\pi_j - 1) (\pi_k - 1)$ or $\pi_j \pi_k \pi_i (\pi_k - 1) (\pi_j - 1)$ in the latter case. That is, π contains 2341, 42531, or 42513.

In the other instance where π_k is popped to the output, we have $\pi_k < \pi_i < \pi_j - 1$. It must be the case that $\pi_j - 1$ appears after π_i as $\pi_k < \pi_i < \pi_j - 1$, so $\pi_j - 1$ cannot be in the same pop stack as π_k , as the entries there are in consecutive order and $\pi_j - 1$ is too small to be in the pop stack with π_j on top. If $k < j < i$, then π contains a pattern $\pi_k \pi_j \pi_i (\pi_j - 1) (\pi_k - 1)$ which contains 2341 or a pattern $\pi_k \pi_j \pi_i (\pi_k - 1) (\pi_j - 1)$ (which is 25314). If instead $j < k < i$, then π contains $\pi_j \pi_k \pi_i (\pi_j - 1) (\pi_k - 1)$ which contains 2341 or a pattern $\pi_j \pi_k \pi_i (\pi_k - 1) (\pi_j - 1)$ (which is 52314).

The last possibility is that π_k is output before $\pi_k - 1$ through a bypass operation. Due to the requirements of PSBP, this means $\pi_k - 1$ appears after π_k in π and also both stacks have top entries $\pi_j, \pi_m > \pi_k$. The stage immediately before π_k is bypassed to the output is shown on in the right picture of Fig. 6. Without loss of generality, suppose $\pi_j < \pi_m$. For π_k not to have been pushed on top of π_j , there must be another entry $\pi_j - 1$ that appears after π_k in π . In the case where $j < m$, this means π contains the pattern $\pi_j \pi_m \pi_k (\pi_j - 1) (\pi_k - 1)$ or $\pi_j \pi_m \pi_k (\pi_k - 1) (\pi_j - 1)$, that is π contains 45231 or 45213. Otherwise, if $m < j$, for π_j to have not been pushed on top of π_m , we must have had the entry $\pi_m - 1$ appear before π_m or after π_k . That is, π contains one of the following patterns:



Fig. 6. The stage immediately prior to which PSBP first fails to sort a permutation π . The left picture shows when π_k will enter the output via a pop stack and the right picture shows when π_k will enter the output via the bypass operation.

$(\pi_m - 1)\pi_m\pi_j\pi_k(\pi_j - 1)(\pi_k - 1)$, i.e. contains 45231 (without π_j),
 $(\pi_m - 1)\pi_m\pi_j\pi_k(\pi_k - 1)(\pi_j - 1)$, i.e. contains 45213 (without π_j),
 $\pi_m\pi_j\pi_k(\pi_m - 1)(\pi_j - 1)(\pi_k - 1)$, i.e. contains 42531 (without π_m),
 $\pi_m\pi_j\pi_k(\pi_m - 1)(\pi_k - 1)(\pi_j - 1)$, i.e. contains 42513 (without π_m),
 $\pi_m\pi_j\pi_k(\pi_j - 1)(\pi_m - 1)(\pi_k - 1)$, i.e. contains 2341 (without π_m, π_j),
 $\pi_m\pi_j\pi_k(\pi_k - 1)(\pi_m - 1)(\pi_j - 1)$, i.e. contains 642153,
 $\pi_m\pi_j\pi_k(\pi_j - 1)(\pi_k - 1)(\pi_m - 1)$, i.e. contains 52314 (without π_j),
 $\pi_m\pi_j\pi_k(\pi_k - 1)(\pi_j - 1)(\pi_m - 1)$, i.e. contains 642135. $\square$

Once again, the inverse of the above class has a regular insertion encoding [24], thus it is possible to automatically deduce its rational generating function [2,6], which is

$$\frac{(1-x)(1-2x)(1-4x)}{1-8x+20x^2-18x^3+3x^4}. \quad (2)$$

The sequence of coefficients has now been recorded as A374165 in [23].

In fact, we can show that any machine made up of k pop stacks in parallel with a bypass option will sort a class of permutations characterized by the avoidance of a finite set of patterns. This is an extension of work of Atkinson and Sack [5, Theorem 2] showing that a machine made up of k pop stacks in parallel has sortable permutations being exactly a permutation class with a finite basis. We can even utilize their constructive technique which is not dissimilar to specific case shown in Proposition 8.2. As there is a small error in the original published construction, we include the full proof of Theorem 8.3 here. We also note that the basis given following Atkinson and Sack [5, Theorem 2] for the $k = 2$ case is actually the complement of the correct basis. Indeed, the set of permutations sortable by two pop stacks in parallel (without a bypass) is $\text{Av}(2341, 3412, 25314, 42531, 52314, 53124, 53142)$.

Before stating Theorem 8.3, we will describe an algorithm that will sort a sortable permutation π by k pop stacks in parallel with a bypass where we do not concern ourselves with the image of any non-sortable permutation. At each stage, the process (in priority order) is as follows:

1. If the next entry of the input π_j is the next needed entry for the output, π_j goes directly to the output via the bypass operation.
2. If the top entry of any pop stack S_i is the next needed entry for the output, pop S_i .
3. If the next entry of the input π_j is exactly one less than the top entry t_i of pop stack S_i , push π_j on top of t_i .
4. If the next entry of the input π_j is not exactly one less than the top entry of any of the pop stacks, push π_j into an empty pop stack.
5. If none of these moves are possible, π is not sortable.

With this set of legal moves in mind, we can now prove that the basis for the class of permutations sortable by k pop stacks in parallel with a bypass is finite.

Theorem 8.3. *There is a finite set of permutations B_k such that a permutation π is sortable by k pop stacks in parallel with a bypass if and only if $\pi \in \text{Av}(B_k)$.*

Proof. Consider a permutation π that cannot be sorted by k pop stacks in parallel with a bypass option by any algorithm. Specifically, there must be a point of failure in any process where no legal move can be made. At this stage,

1. every pop stack S_i contains a nonempty consecutive sequence of entries increasing from top to bottom with top entry t_i and bottom entry b_i ,
2. the last entry of the output is c (where $c = 0$ if the output is currently empty),
3. the current input entry is π_j

4. $t_i > c + 1$ for all i ,
5. For each i , we have either $\pi_j > t_i$ or $\pi_j < t_i - 1$, and
6. $\pi_j > c + 1$.

Notice that each entry b_i , if $b_i \neq n$, must have entered the stacks prior to $b_i + 1$, as otherwise b_i would have been placed on top of $b_i + 1$ instead of being put at the bottom of an empty stack. Further, the entry $c + 1$ is not in the stacks, so must appear after π_j in π . Finally, π_j cannot enter any stack, so if $\pi_j \neq n$, then $\pi_j + 1$ must appear after π_j .

However, conversely, suppose we have any set of (not necessarily distinct) entries $\{b_1, b_2, \dots, b_k, b_1 + 1, b_2 + 1, \dots, b_k + 1, \pi_j, \pi_j + 1, c + 1\}$ (where we eliminate the largest entry) such that

1. b_i appears before $b_i + 1$,
2. π_j appears after b_i for all i
3. π_j appears before $\pi_j + 1$, and
4. $c + 1$ appears after π_j .

Then each b_i needs to be pushed into a different pop stack, none of the pop stacks can be popped, and π_j has no legal move (push or bypass). Thus we have a finite set of permutations that can minimally fulfill the constraints forcing a failure. Therefore the basis of permutations that can be sorted by a machine consisting of k pop stacks in parallel with a bypass is finite. $\square$

It is a straightforward extension of the proof of [22, Theorem 1] to show that the generating function for the enumeration of permutations sortable by k pop stacks in parallel with a bypass is rational for any k , so we include the following theorem without further justification.

Theorem 8.4. *For any positive integer k , the set of permutations sortable by k pop stacks in parallel with a bypass has a rational generating function.*

9. Further work

9.1. An alternate algorithm and sorting words

While PSB is an ideal algorithm to handle permutations when sorting with pop stacks with a bypass, the priority of push over bypass can be a weakness when trying to modify to sort words. When stack sorting, the patterns that must be avoided to be sortable are the same for permutations and words. However, the same is not true for pop stack sorting where, in addition to avoiding 213 and 312, words must also avoid 1010, as shown for Cayley words by Cerbai [9] and later expanded to all words in [20]. The same additional restriction of avoiding 1010 for words applies to PSB.

If we have a regular word with the same number of copies of each letter, for example Stirling permutations, or if we want to check the count of each letter prior to sorting, we can remove the need to avoid 1010 by prioritizing the bypass operation more. It can be shown using techniques similar to those given in Section 3 that Algorithm 3, which we call PSBW(k), gives the procedure for k regular words with the smallest letter 0. This family of algorithms could also be expanded to keep track of individual repetitions to properly prioritize the bypass when sorting other types of words.

9.2. Conjecture on the enumeration of simple permutations sortable by two pop stacks in series

We have already noted that the terms of the generating function for the sequence corresponding to the permutations of size n sortable by $k = 2$ pop stacks in parallel now appear in [23] as A374165. However, there is also a potential nice closed form for the simple sortable permutations that we boldly put here as a conjecture:

Conjecture. *Let a_n be the number of simple permutations of size n that can be sorted by a machine consisting of two pop stacks in parallel where entries are allowed to bypass the pop stacks. Then $a_0 = a_1 = 1$, $a_2 = 2$, $a_n = F_{2n-5} - 1$ if $n \geq 3$ is odd, and $a_n = F_{2n-5}$ if $n > 3$ is even (where F_n is the n -th Fibonacci number).*

Authors contribution

All authors contributed equally to this work.

Funding

L.F. is member of the INdAM research group GNCS; he has been partially supported by the 2024 INdAM-GNCS project CUP_E53C23001670001.

```

1  S := ∅;
2  i := 1;
3  j := 0;
4  while i ≤ kn do
5      if  $\pi_i \leq \lfloor \frac{j}{k} \rfloor$  then
6          BYPASS;
7          j := j + 1;
8      else if  $\pi_i \leq \text{TOP}(S)$  then
9          PUSH;
10     else
11         POP;
12         j := i - 1;
13         if  $\pi_i \leq \lfloor \frac{j}{k} \rfloor$  then
14             BYPASS;
15             j := j + 1;
16         else
17             PUSH;
18     i := i + 1;
19 POP;

```

Algorithm 3: PSBW(k) (S is the pop stack; $\text{TOP}(S)$ is the current top element of the pop stack; $\pi = \pi_1 \cdots \pi_{kn}$ is the input k -regular word (for a permutation, $k = 1$); j is the number of entries in the output; $\text{TOP}(S)$ at the top of pop stack.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this article.

Acknowledgements

The authors would like to warmly thank Jay Pantone and Henning Ulfarsson, for their valuable assistance in computing some of the generating functions in Section 7 and for testing some matches with sequences in [23] using their Combinatorial Exploration Framework [2,6].

Data availability

No data was used for the research described in the article.

References

- [1] M.H. Albert, M.D. Atkinson, M. Bouvel, A. Claesson, M. Dukes, On the inverse image of pattern classes under bubble sort, *J. Comb.* 2 (2011) 231–243.
- [2] M.H. Albert, C. Bean, A. Claesson, E. Nadeau, J. Pantone, H. Ulfarsson, Combinatorial exploration: an algorithmic framework for enumeration, *Mem. Am. Math. Soc.* (2026), in press.
- [3] M.D. Atkinson, Restricted permutations, *Discrete Math.* 195 (1999) 27–38.
- [4] D. Avis, M. Newborn, On pop-stacks in series, *Util. Math.* 19 (1981) 129–140.
- [5] M.D. Atkinson, J.-R. Sack, Pop-stacks in parallel, *Inf. Process. Lett.* 70 (1999) 63–67.
- [6] C. Bean, J.S. Eliasson, T.K. Magnusson, E. Nadeau, J. Pantone, H. Ulfarsson, Tilings: combinatorial exploration for permutation classes, at, <https://github.com/PermutaTriangle/Tilings/>.
- [7] M. Bousquet-Melou, Sorted and/or sortable permutations, *Discrete Math.* 225 (2000) 25–50.
- [8] M. Bouvel, L. Cioni, L. Ferrari, Preimages under the Bubblesort operator, *Electron. J. Comb.* 29 (2022), #P4.32.
- [9] G. Cerbai, Sorting Cayley permutations with pattern-avoiding machines, *Australas. J. Comb.* 80 (2021) 322–341.
- [10] S. Chern, S. Fu, Z. Lin, Burstein's permutation conjecture, Hong and Li's inversion sequence conjecture and restricted Eulerian distributions, *Proc. Edinb. Math. Soc.* (2) 66 (2023) 1179–1201.
- [11] N. Chomsky, M.-P. Schützenberger, The algebraic theory of context-free languages, in: P. Braffort, D. Hirschberg (Eds.), *Computer Programming and Formal Systems*, North-Holland, Amsterdam, 1963, pp. 118–161.
- [12] L. Cioni, L. Ferrari, Preimages under the Queuesort algorithm, *Discrete Math.* 344 (2021) 112561.
- [13] L. Cioni, L. Ferrari, Sorting with a popqueue, *RAIRO Theor. Inform. Appl.* 58 (2024) 13.
- [14] A. Claesson, B. Guðmundsson, Enumerating permutations sortable by k passes through a pop-stack, *Adv. Appl. Math.* 108 (2019) 79–96.
- [15] C. Defant, Postorder preimages, *Discrete Math. Theor. Comput. Sci.* 19 (2017) 3.
- [16] C. Defant, Stack-sorting preimages of permutation classes, *Sémin. Lothar. Comb.* 82 (2020) B82b.
- [17] L. Hong, R. Li, Length-four pattern avoidance in inversion sequences, *Electron. J. Comb.* 29 (2022), #P4.37.
- [18] D.E. Knuth, *The Art of Computer Programming*, Vol. 1, Addison-Wesley, Boston, 1968.
- [19] H. Magnusson, Sorting operators and their preimages, Master's thesis, Reykjavik University, 2013.
- [20] T. Mansour, H. Skogman, R. Smith, Sorting inversion sequences, *Discrete Math. Theor. Comput. Sci.* 27 (1) (2025), #1.

- [21] L.K. Pudwell, R. Smith, Two-stack-sorting with pop stacks, *Australas. J. Comb.* 74 (2019) 179–195.
- [22] R. Smith, V. Vatter, The enumeration of permutations sortable by pop stacks in parallel, *Inf. Process. Lett.* 109 (2009) 626–629.
- [23] N.J.A. Sloane, The online encyclopedia of integer sequences, at oeis.org.
- [24] V. Vatter, Finding regular insertion encodings for permutation classes, *J. Symb. Comput.* 47 (2012) 259–265.