Combinatoric properties of sorting algorithms

Lapo Cioni

November 22, 2022

A permutation of length n is a function $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$, and can be written as the concatenation of its images, $\pi(1) \cdots \pi(n)$. Denote with S_n the set of permutations of length n, and with S the set of all permutations.

Knuth [12] introduced the algorithm `Stacksort`, which attempts to sort a permutation using a stack. His work inspired some further investigations on algorithms that sort permutations using addictional structures, for example Tarjan [17] considered networks of stacks and Pratt [16] studied a deque. Initially, this was interesting from the point of view of computer science, since the considered external structures were storage devices with no functional power, and it was thus natural to ask what permutations they could compute. Shortly thereafter, combinatorialists noticed how the tools that they were developing to address these problems were maybe even more interesting than the devices themselves. One particular concept became central in the study of permutations (in general, not just from the point of view of sorting): the *pattern* of a permutation.

A permutation $\rho \in S_k$ is a *pattern* of $\pi \in S_n$ if and only if there exists a subsequence of π of length k such that its elements are order isomorphic to those of ρ. If ρ is not a pattern of π, we say that π *avoids* ρ. Given $B \subseteq S$, $Av(B)$ is the set of permutations avoiding each pattern $\rho \in B$, and is called *class*. Finally, $Av_n(B)$ is the set of permutations of length n in $Av(B)$.

Subsequently, many other algorithms were introduced and analyzed, such as `Queuesort`, which sorts using a queue, and `Bubblesort`, which sorts without external structures, by just reading the permutation from left to right and swapping adjacent elements which are in decreasing order. Of particular interest is the sorting device composed by two stacks in parallel, for which an optimal sorting algorithm (and a characterization of sortable permutations) is still not know. Specifically, there exists an algorithm that decides, in polynomial time with respect to the length of the input, if a permutation can be sorted using the allowed operations [15]. The algorithm iteratively constructs a graph that encodes all the sorting processes satisfying a specific property, and uses it to determine if such a sorting process exists. Still, an algorithm that sorts a permutation without addictional structures (such as graphs), just by looking at the elements, is yet to be found.

This research project fits into the frame of investigating properties of permutations arising from sorting algorithms. In what follows, we introduce the properties we want to analyze, recounting some of the successes that have been reached for some sorting algorithms.

Preimages For any sorting algorithm `Sort`, we define a function f_{Sort} that maps every permutation π to its output under `Sort`. Thus, for $\sigma \in S$, we can consider the set $f^{-1}_{Sort}(\sigma)$ of the preimages of σ under the function associated to the algorithm. Sometimes, for brevity, the set is just referred to as the "preimages of σ under `Sort`".
The set $f^{-1}_{Sort}(1 \cdots n)$ is of particular interest, since it consists of all the permutations of length n that are sorted to the identity $1 \cdots n$ by the algorithm, and interesting information on its characterization and its cardinality are given in terms of pattern avoidance. For example, the

permutations sorted by `Stacksort` are those that avoid the pattern 231, and the number of such permutations of length n is the n-th Catalan number (sequence A000108 of [14]).

Clearly, the general problem is to characterize the preimages of any permutation. This has been done, with various degrees of success, for several sorting algorithms, such as `Stacksort` [3, 7], `Bubblesort` [4] and `Queuesort` [5].

Finally, it is also possible to study the preimages of a class instead than a single permutation, see `Stacksort` [8, 13, 18], `Bubblesort` [1] and `Queuesort`[13].

Composition Given a permutation π, we can apply a sorting algorithm to π, and a second sorting algorithm to the output (notice that the second sorting algorithm may be the same as the first), thus *composing* the two. The resulting procedure is a new sorting algorithm, therefore the same kind of questions as above can be posed (which permutations are sortable, how many there are, what are the preimages of a permutation or a class, etc).

For example, consider the procedure that applies `Stacksort` twice. Julian West [19] proved that this procedure sorts every permutation avoiding the classical pattern 2341 and the barred pattern $3\bar{5}241$, and conjectured a formula for their enumeration, which was first proved by Zeilberger [20], and subsequently purely combinatorial proofs have been found, such as [6]. Specifically, the sortable permutations of length n are counted by sequence A000139 of [14]. This sequence counts, among other things, the number of fighting fish with $n+1$ free upper edges [10].

The algorithms that we aim to explore are `Pancakesort` and those arising from $\mathscr{C}$-machines or shuffle methods.

Suppose that we have a stack of n pancakes, each with a different diameter. We want to rearrange them so that the diameters decrease bottom to top. We have a spatula, which can be inserted in any position to flip all of the pancakes above it. More formally, we have a permutation $\pi = \pi_1 \cdots \pi_n$, and we can reverse the first k elements, for $k = 1, \cdots, n$, to obtain $\pi_k \pi_{k-1} \cdots \pi_1 \pi_{k+1} \pi_{k+2} \cdots \pi_n$. The goal is to apply this operation several times to obtain $1 \cdots n$.

This kind of operations arise from bioinformatics, specifically from genome rearrangement problems. The DNA is a sequence of nucleotides, which can be viewed as a permutation. It can undergo some mutations, that change it by rearranging the sequence of nucleotides in some specific ways. Usually the problems studied involve the distance, in terms of number of mutations, betweeen any two sequences. One of the possible mutations is prefix-reversal, studied in [11] in the context of finding the minimal number of steps to obtain sequence $1 \cdots n$.

We consider a particular algorithm that allows the prefix-reversal operations, studied by Magnusson [13]: let $\pi = A_1 k A_2 (k+1)(k+2) \cdots n$, where A_1 and A_2 are sequences of elements, with A_2 nonempty. Reverse $A_1 k$, obtaining $k A_1^r A_2 (k+1)(k+2) \cdots n$, where A_1^r denotes the reverse of sequence A_1. Then reverse $k A_1^r A_2$, obtaining $A_2^r A_1 k (k+1) \ldots n$.

This algorithm is called `Pancakesort`. Magnusson found that the sortable permutations are those avoiding the patterns 132, 312 and 3241. Also, for any pattern $\rho \in S$, he described the preimages of the class $Av(\rho)$.

Question 1. Given a permutation σ, how many (and which) permutations are preimages of σ under `Pancakesort`? That is, what is the cardinality of $f^{-1}_{Pancakesort}(\sigma)$, and how is the set characterized?

Question 2. Given a nonnegative integer k, how many (and which) permutations have precisely k preimages under `Pancakesort`?

A $\mathscr{C}$-machine is a device that generalizes stacks, introduced by Albert et al. [2]. It is used as a container of elements of a permutation, arranged in some order with some restrictions.

Specifically, every $\mathscr{C}$-machine is equipped with a class $\mathscr{C}$ (that is, a set of the form $Av(B)$, with B a set of patterns), which dictates the way in which the elements may be arranged inside the machine. This are the allowed operations:

- remove the next entry from the input and immediately append it to the end of the output (bypassing the machine);
- remove the next entry from the input and place it anywhere in the container in such a way that the partial permutation in the container is in the same relative order as a permutation in the class;
- remove the leftmost entry from the container and append it to the end of the output.

In their paper, the authors used this device to generate permutations starting from the input permutations $1 \cdots n$, $n \geq 0$, but this device could also be used for sorting. If we swap input and ouput we obtain a device, which we call $\mathscr{C}'$-machine, with the following allowed operations:

- remove the next entry from the input and immediately append it to the end of the output (bypassing the machine);
- remove the next entry from the input and place it in the rightmost position in the container, provided that the resulting partial permutation in the container is in the same relative order as a permutation in the class $\mathscr{C}$;
- remove any entry from the container and append it to the end of the output.

Basically, this device reverse the way in which a $\mathscr{C}$-machine works, allowing to insert element only in a specific position (the rightmost one), but remove them from anywhere. Notice that, for $\mathscr{C} = Av(\rho)$, $\rho \in S_2$, the $\mathscr{C}'$-machine works either as a queue or a stack for the purpose of sorting a permutation.

Question 3. Given a $\mathscr{C}'$-machine, with $\mathscr{C} = Av(\rho)$, $\rho \in S_3$, which (and how many) permutations can be sorted? What algorithm could sort those sortable permutations?

Question 4. What would be the preimages of a permutation σ under the algorithm in Question 3?

The last device that we want to consider, called *shuffle queue*, was introduced by Stoyan Dimitrov [9]. It consists of a container in which the elements are inserted in the rightmost position and taken out from the leftmost position, just as a queue. The difference is that another operation is also possible: a shuffle.
A *shuffle* of a sequence λ of length k consists of applying a permutation $\sigma \in S_k$ to λ. For example, if we shuffle the sequence 2651 with the permutation 4132 we obtain 1256, since the fourth element goes to the first position, the first element goes to the second position, and so on.
Dimitrov considered three different variations of a shuffle queue. The first one, called $\mathbb{Q}_\Sigma$ allow the following operations:

push remove the next entry from the input and place it in the rightmost position in the container;

pop remove the leftmost entry from the container and append it to the end of the output;

shu shuffle the content of the container with a permutation $\sigma \in \Sigma$.

We can modify $\mathbb{Q}_\Sigma$ by forcing the container to be emptied after every *shu* operation. That is, if *shu* is executed, then *pop* must be executed until the container is empty. This is denoted $\mathbb{Q}'_\Sigma$.

The second variation instead forces the shuffle queue to be emptied with every *pop* operation. That is, if *pop* is executed, then *pop* must be executed until the container is empty. This is denoted $\mathbb{Q}^{pop}_\Sigma$.

Finally, the author focused on two specific sets Σ of allowed shuffles: cuts and reverses. The former consists of all permutations of the form $k(k+1)\cdots n1\cdots(k-1)$, which corresponds to taking a prefix of the sequence and putting it at the end. The latter consists only of the permutations $n\cdots 1$, which reverse the content of the shuffle queue. Sortable permutations were studied, describing an algorithm that sorts them, but there is still nothing about their preimages.

Question 5. Given a permutation σ, which (and how many) are its preimages under one of the sorting algorithms that use a shuffle queue?

The broader scope of this research, other than populate the lists of results about sorting algorithms, is to develop tools for analizing sorting problems. That may result fruitful for unsolved sorting problems, such as the problem of charactering permutations sortable using two stack in series, and sorting them without external structures.

Time frame	**Objectives**
Months 1-4	Questions 1 and 2
Months 5-12	Write article for `Pancakesort` preimages, Questions 3 and 4, and PP2024
Months 13-18	Write article on $\mathscr{C}$-machines images and preimages Question 5
Months 19-24	Write article on shuffle queues preimages, PP2025 and FPSAC2025

References

[1] M. H. Albert, M. D. Atkinson, M. Bouvel, A. Claesson, M. Dukes. *On the inverse image of pattern classes under bubble sort.* Journal of Combinatorics, 2(2):231–243, 2011.

[2] M. H. Albert, C. Homberger, J. Pantone, N. Shar, V. Vatter, *Generating permutations with restricted containers.* Journal of Combinatorial Theory, Series A 157 (2018): 205-232.

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

[4] M. Bouvel, L. Cioni, L. Ferrari. *Preimages under the bubblesort operator.* Electronic Journal of Combinatorics, Volume 29, Issue 4 (2022), P4.32.

[5] L. Cioni, L. Ferrari. *Preimages under the Queuesort algorithm.* Discrete Mathematics 344.11 (2021): 112561.

[6] R. Cori, J. Benjamin, S Gilles, *Description trees for some families of planar maps.* Proceedings of the 9th conference on formal power series and algebraic combinatorics, 1997.

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

[8] C. Defant, *Stack-sorting preimages of permutation classes*, arXiv preprint arXiv:1809.03123 (2018).

[9] S. Dimitrov, *Sorting by shuffling methods and a queue.* Extended Abstracts EuroComb 2021. Birkhäuser, Cham, 2021. 201-207.

[10] E. Duchi, V. Guerrini, S. Rinaldi, G. Schaeffer, *Fighting fish.* J. Phys. A, 50.2 (2017): 024002.

[11] W. H. Gates, C. H. Papadimitriou, *Bounds for sorting by prefix reversal.* Discrete mathematics 27.1 (1979): 47-57.

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

[13] H. Magnússon, *Sorting operators and their preimages*, Ph.D. thesis, 2013.

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

[15] A. Pierrot, D. Rossin, *2-stack sorting is polynomial.* Theory of Computing Systems 60.3 (2017): 552-579.

[16] V. R. Pratt, *Computing permutations with double-ended queues. Parallel stacks and parallel queues* Proc. Fifth Annual ACM Symposium on Theory of Computing (Austin, Tex., 1973), pp. 268–277.

[17] R. Tarjan, *Sorting using networks of queues and stacks.* Journal of the ACM, 19 (2): 341–346 (1972).

[18] H. Úlfarsson, A. Claesson, *Sorting and preimages of pattern classes.* Discrete Mathematics & Theoretical Computer Science Proceedings, AR, 595–606 (2012).

[19] J. West, *Permutations with forbidden subsequences, and, stack sortable permutations.* PhD thesis, Massachusetts Institute of Technology, 1990.

[20] D. Zeilberger, *A proof of Julian West's conjecture that the number of two-stacksortable permutations of length n is 2 (3n)!/((n+ 1)!(2n+ 1)!).* Discrete Mathematics 102.1 (1992): 85-93.