Stack sorting with increasing and decreasing stacks

Stack sorting with increasing and decreasing stacks

Giulio Cerbai, Luca Ferrari (Dipartimento di Matematica e Informatica, University of Firenze, Italy)

The problem of sorting a permutation using a stack was first introduced by Knuth [4] in the 1960s; in its classical formulation, the aim is to sort a permutation using a first-in/last-out device. As it is well known, in this case the set of sortable permutations is a class, whose basis consists of the single element 231, and whose enumeration is given by Catalan numbers.

More generally (see [8]), one can consider a network of sorting devices, each of which is represented as a node in a directed graph; when there is an arc from node S to node T the machine is allowed to pop an element from S and push it into T; if we mark two distinct vertices as the input and the output machine, then the sorting problem consists of looking for a sequence of operations that allows us to move a permutation from the input to the output machine, thus obtaining the identity permutation.

In this framework, some of the typical problems are the following:

- characterize the permutations that can be sorted by a given network;
- enumerate sortable permutations with respect to their length;
- if the network is too complex, find a specific algorithm that sorts "many" input permutations and characterize such permutations.

Concerning the last stated problem, note that, for a given network of devices, although the set of sortable permutations forms a class in general, this is not anymore true if one choose a specific sorting strategy; this approach leads in general to more complicated characterizations which involve other kinds of patterns (as it happens, for instance, for the 2-West stack-sortable permutations [9]).

Although it's very hard to obtain interesting results for large networks, a lot of work has been done for some particular, small networks (see [2] for a dated survey, or [3] for a more recent one); in this work we restrict our attention to the case of stacks in series, with the restriction that the elements are maintained inside each stack either in increasing or in decreasing order. Our starting point is [6], where Rebecca Smith proves that the permutations sorted by a decreasing stack followed by an increasing one form the class $Av(3241, 3142)$.

Many decreasing stacks followed by an increasing one. Generalizing the approach of [6], we will consider here a sorting device made by k decreasing stacks in series, denoted by $D_1, \dots, D_k$, followed by an increasing stack I. Recall that "decreasing" (resp., "increasing") stack means that the elements inside the stack have to be in decreasing (resp., increasing) order from top to bottom. When $k = 0$, we just have a single increasing stack, so we obtain the usual Stacksort procedure. When $k = 1$, we obtain exactly the DI machine described in [6]. In the sequel we denote our machine with D^kI.

The possible operations the D^kI machine can perform are the following:

- d_0: push the next element of the input permutation into the first decreasing stack D_1;
- d_i, for $i = 1, \ldots, k-1$: pop an element from D_i and push it into the next decreasing stack D_{i+1};
- d_k: pop an element from D_k and push it into the increasing stack I;
- d_{k+1}: pop an element from the increasing stack I and output it (by placing it on the right of the list of elements that have already been output).

Notice that each operation can be performed only if it does not violate the restrictions of the stacks; in this case, we call it a *legal* operation. For the special case of the operation d_{k+1}, we will assume that d_{k+1} is legal either if we are pushing into the output the smallest among the elements not already in the output or if all the other operations are not legal.

For any given k, we are now interested in characterizing the set

$$B(k) = \{\pi \in S \mid \text{ there is a sequence of legal operations } d_{i_1}, \ldots, d_{i_s} \text{ that sorts } \pi\}.$$

If $\pi \in B(k)$, we say that π is k-sortable. Using a standard argument it is easy to show that $B(k)$ is a class, for every k.

The natural way to describe the class $B(k)$ is to understand its basis. Here we show that, when $k = 2$, the basis of $B(k)$ is infinite, by explicitly finding an infinite antichain of permutations which are not 2-sortable and are minimal with respect to the pattern ordering. The construction of the infinite antichain described in the next theorem can be easily adapted to every $k \geq 2$. An extremely useful tool to find such an antichain has been the software *PermLab* [1], developed by Michael Albert.

Theorem 1. *For $j \geq 0$, define the permutation:*

$$\alpha_j = 2j+4, 3, a_1, b_1, a_2, b_2, \ldots, a_j, b_j, 1, 5, 2$$

where:

$$\begin{cases} A_j = (a_1, \ldots, a_j) = (2j+2, 2j, 2j-2, \ldots, 6, 4), \\ B_j = (b_1, \ldots, b_j) = (2j+5, 2j+3, 2j+1, \ldots, 9, 7). \end{cases}$$

Then the set of permutations $\{\alpha_j\}_{j\geq 0}$ constitutes an infinite antichain each of whose elements is not 2-sortable. Moreover, α_j is minimal with respect to such a property, i.e. if we remove any element of α_j we obtain a 2-sortable permutation. As a consequence, the basis of $B(2)$ is infinite, since it contains the infinite antichain $\{\alpha_j\}_{j\geq 0}$.

A left-greedy algorithm. Instead of making an unrestricted use of the D^kI machine, we may define a specific algorithm, by choosing the priority of each operation. Depending on our choices, we obtain different sorting procedures: each of them determines a different set of sortable permutations, which is interesting to understand.

Our first proposal is a left-greedy procedure, where, at each step, we perform the legal operation d_j having maximum index j. Setting $B_{lg}(k) = \{\pi : \pi \text{ is sorted by the left-greedy procedure}\}$, it turns out that $B_{lg}(k)$ is in fact a class, and we are able to completely characterize it.

Proposition 2. *For every $k \geq 1$, $B_{lg}(k) = Av(231)$.*

Thus, using this left greedy algorithm, we obtain a procedure that sorts precisely the same permutations as Stacksort does. Thus, in a sense, adding any number of decreasing stacks before an increasing one does not improve the sorting power of the machine, if we always perform the leftmost legal operation. This does not mean, however, that this "left-greedy D^kI machine" is equivalent to Stacksort. Indeed, taking for instance $k = 1$ and the input permutation 2341, the left-greedy D^kI machine returns 2134 as output, whereas Stacksort returns 2314. This is of course due to the fact that the elements of the input permutation exit the (last) decreasing stack in a different order in the two procedures. We can therefore define the map $\phi_k : S_n \to S_n$, for $k \geq 1$, that associates to an input permutation π of length n the output of the last stack D_k in the left-greedy algorithm. As a consequence of the last proposition, for each π and $k \geq 1$, $\pi \in Av(231)$ if and only if $\phi_k(\pi) \in Av(231)$. In order to have a better understanding of the left-greedy D^kI machine, it would be interesting to explore more deeply the properties of the maps ϕ_k.

Example. We report the values of $\phi_k(\pi)$ when $\pi = 36257418$ and $k = 1, 2, 3, 4, 5$. It is easy to observe that, for sufficiently large values of k, the succession $\{\phi_k(\pi)\}_{n\in\mathbb{N}}$ eventually becomes constant. However, we do not know precisely when this happens.

$$\begin{aligned} k = 1: &\quad 36275418, \\ k = 2: &\quad 37652841, \\ k = 3: &\quad 37652841, \\ k = 4: &\quad 38765241, \\ k = 5: &\quad 38765241. \end{aligned}$$

An almost left-greedy algorithm. There is a better way to design an almost left-greedy algorithm, which is able to sort more permutations. The idea is to give the increasing stack a privileged role, using it only when it is strictly necessary. Formally, at each step we choose to perform the first legal operation according to the following priority rule:

$$d_{k+1} > d_{k-1} > d_{k-2} > \cdots > d_1 > d_0 > d_k,$$

where $d > d'$ means that the priority of operation d is higher than the priority of operation d'.

In analogy with the previous case, define $B_{alg}(k)$ as the set of permutations sorted by the partial left-greedy algorithm with k decreasing stacks (from now on it will be called the almost left-greedy D^kI machine). We notice immediately that the permutation 231, which is not sorted by the left-greedy D^kI machine, is instead sortable by the almost left-greedy DI machine: the sequence of the operations performed by the algorithm in this case is $d_0, d_0, d_1, d_1, d_0, d_1, d_2, d_2, d_2$. Unfortunately, in this case $B_{alg}(k)$ is not in general a permutation class, except for the case $k = 1$, for which it is quite easy to prove that the almost left-greedy strategy is equivalent to the (optimal) sorting strategy defined in [6], so that $B_{alg}(1) = Av(3241, 3142)$. As an example, for $k = 2$, the permutation 631425 can be sorted, whereas its subpermutation 52314 cannot.

The fact that $B_{alg}(k)$ is not a downset in general makes the analysis of the almost left-greedy machine more difficult. However, when $k = 2$, we are able to obtain a partial characterization of $B_{alg}(2)$ in terms of *barred patterns*.

Theorem 3. *1. Let π be an almost left-greedy D^2I sortable permutation; then:*

- *π avoids 3214;*

- *π avoids the following barred patterns, each of which is obtained by suitably adding barred elements to the pattern* 52314*:*
  - $63\bar{1}425$*;*
  - $7\bar{2}\bar{1}4536$, $7\bar{3}\bar{1}4526$*;*
  - $\bar{7}\bar{2}8\bar{1}4536$, $\bar{7}\bar{3}8\bar{1}4526$*;*
  - $\bar{8}\bar{2}7\bar{1}4536$, $\bar{8}\bar{3}7\bar{1}4526$.

2. *Let π be a permutation that is not almost left-greedy D^2I sortable. Then one of the following cases holds:*

   - *π contains* 3214*;*
   - *π contains one of the barred patterns listed above;*
   - *π contains an occurrence of* 52314 *that extends to* 82714536 *(resp.,* 83714526*) which in turn is part of one of the following patterns:*
     - 9 3 1 8 2 5 6 4 7 *(resp.,* 9 4 1 8 2 5 6 3 7*);*
     - 10 2 1 4 9 3 6 7 5 8 *(resp.,* 10 2 1 5 9 3 6 7 4 8*);*
     - 10 3 1 4 9 2 6 7 5 8 *(resp.,* 10 3 1 5 9 2 6 7 4 8*);*
     - 10 2 11 1 4 9 3 6 7 5 8 *(resp.,* 10 2 11 1 5 9 3 6 7 4 8*);*
     - 10 3 11 1 4 9 2 6 7 5 8 *(resp.,* 10 3 11 1 5 9 2 6 7 4 8*);*
     - 11 2 10 1 4 9 3 6 7 5 8 *(resp.,* 10 2 11 1 5 9 3 6 7 4 8*);*
     - 11 3 10 1 4 9 2 6 7 5 8 *(resp.,* 10 3 11 1 5 9 2 6 7 4 8*);*

The above theorem fails to characterize $B_{alg}(2)$, due the long "bad" permutations listed in (2). Unfortunately the construction used to obtain these "bad" patterns can be repeated to generate, starting from 52314, a sequence of permutations of increasing lengths whose sortability depends on how many times we iterate the construction. To be more precise, define the permutations $\gamma_m \in S_{3m+2}$ as follows:

$$\gamma_m = 3m+2,\ \underbrace{2\ 3m+1\ 1}_{231},\ \underbrace{4\ 3m\ 3}_{231}\ldots\underbrace{2m-2\ 2m+3}_{231}2n-3\ \underbrace{2m\ 2m+1\ 2m-1}_{231}2m+2.$$

In other words, starting from $\gamma_1 = 52314$, γ_{i+1} is obtained from γ_i by inserting a new maximum in the first position and putting the old maximum between 2 and 1, then suitably rescaling the remaining elements. We can prove that:

1. $\gamma_i \leq \gamma_{i+1}$, for each $i \geq 1$;
2. $\gamma_i \in B_{alg}(2)$ if and only if i is even.

For example, we have:

- $\gamma_1 = 52314 \notin B_{alg}(2)$;
- $\gamma_2 = 82714536 \in B_{alg}(2)$;
- $\gamma_3 = 11\ 2\ 10\ 14936758 \notin B_{alg}(2)$;

  $\vdots$

The existence of an infinite sequence of permutations with this property suggests that it would be quite difficult to obtain simple characterization of $B_{alg}(2)$; it is also conceivable that it should be possible to adapt the above construction to greater values of k, thus obtaining similar (negative) results.

References

[1] M. Albert, "PermLab: Software for Permutation Patterns", at http://www.cs.otago.ac.nz/staffpriv/malbert/permlab.php.

[2] M. Bona, "A survey of stack sorting disciplines", Electron. J. Combin., 9(2) (2002-2003) A1.

[3] S. Kitaev, "Patterns in permutations and words", Monographs in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg, 2011.

[4] D. E. Knuth, "The art of computer programming", volume 1, Fundamental Algorithms, Addison-Wesley, Reading, Massachusetts, 1973.

[5] D. Kremer, "Permutations with forbidden subsequences and a generalized Schröder number", Discrete Math., 218 (2000) 121–130.

[6] R. Smith, "Two stacks in series: a decreasing stack followed by an increasing stack", Ann. Comb., 18 (2014) 359-363.

[7] R. Smith, "Comparing algorithms for sorting with t stacks in series", Ann. Comb., 8 (2004) 113–121.

[8] R. E. Tarjan, "Sorting using networks of queues and stacks", Journal of the ACM, 19 (1972) 341–346.

[9] J. West, "Permutations with forbidden subsequences and Stack sortable permutations", PhD-thesis, Massachusetts Institute of Technology, 1990.

[10] J. West, "Sorting twice through a stack", Theoret. Comput. Sci., 117 (1993) 303–313.