Sorting permutations using a pop stack with a bypass

Rebecca Smith SUNY Brockport

This talk is based on joint work with Lapo Cioni and Luca Ferrari

When sorting permutations, various stack-based structures have been extensively studied since Knuth's introduction of the stacksorting procedure [6]. We present a sorting device which augments a pop stack with a bypass operation. A pop stack is a last-in, first-out structure, but unlike a standard stack, a pop operation removes all entries at once rather than just the top entry. Our device adds a bypass operation, which allows entries to move directly from input to output, creating a sorting machine more powerful than the classical Popstacksort algorithm introduced by Avis and Newborn [3].

We first provide a characterization of sortable permutations in terms of forbidden patterns:

Theorem 1. *A permutation π can be sorted using a pop stack with a bypass if and only if $\pi \in Av(231, 4213)$, i.e., π avoids the patterns 231 and 4213.*

The enumeration of this class was previously established by Atkinson [1]:

Theorem 2. *The number of permutations of length n that can be sorted using a pop stack with bypass is given by the odd-indexed Fibonacci number F_{2n-1}.*

We establish a new combinatorial proof of this result by developing a bijection between the sortable permutations and a class of restricted Motzkin paths. Specifically, we map each sortable permutation to a unique Motzkin path satisfying the following conditions:

- The path begins with an Up step and ends with a Down step.
- The path never has a Horizontal step immediately preceded or followed by a Down step.
- When the path has a Down step, that step is immediately followed by as many Down steps as necessary to reach the x-axis.

This bijection allows us to establish the enumeration through standard techniques for counting restricted Motzkin paths.

We further investigate the structure of the preimages under our sorting algorithm. For any permutation σ, the set $\mathrm{PSB}^{-1}(\sigma)$ consists of all permutations that sort to σ using our device. We develop an algorithm to compute all preimages of a given permutation, which allows us to prove the following results:

Proposition 3. *The permutations having exactly 0, 1, and 2 preimages under the pop stack with bypass sorting algorithm can be characterized as follows:*

- *Permutations with 0 preimages are exactly those not ending with their maximum.*
- *Permutations with 1 preimage are those ending with their maximum where all left-to-right maxima are consecutive and non-adjacent.*
- *Permutations with 2 preimages have a specific structure where all left-to-right maxima except possibly the first one are consecutive and non-adjacent.*

We determine precisely when the preimage of a principal class under our sorting operation forms a permutation class itself:

Proposition 4. *Let ρ be a permutation. The set $PSB^{-1}(Av(\rho))$ is a permutation class if and only if ρ has one of the following forms:*

- $\rho = n\alpha$ *for some* $\alpha \in S_{n-1}$.
- $\rho = (n-1)\alpha n$ *for some* $\alpha \in S_{n-2}$.

In all other cases, $PSB^{-1}(Av(\rho))$ is not a permutation class.

For the cases where the preimage is a class, we are able to provide explicit bases (i.e., minimal sets of forbidden patterns).

We then extend our analysis to consider a device consisting of two pop stacks in parallel with a bypass. Building on work by Atkinson and Sack [4], who studied pop stacks in parallel (without bypass) and proved that the set of sortable permutations for any number k of pop stacks in parallel is characterized by a finite set of forbidden patterns, we prove:

Theorem 5. *The set of permutations sortable by two pop stacks in parallel with a bypass is precisely the permutation class*

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

More generally, we prove:

Theorem 6. *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 $\pi \in Av(B_k)$.*

Following the approach originally applied to classical pop stacks in [7], we determine that the generating function for the permutations sortable by two pop stacks in parallel with a bypass is:

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

This generating function was computed using the regular insertion encoding approach developed by Vatter [8] and the automation packages by Albert et al. [2] and Bean et al. [5]. We extend this result to prove:

Theorem 7. *For any positive integer k, the set of permutations sortable by k pop stacks in parallel with a bypass has a rational generating function.*

This work opens several directions for future research, including the extension to words (rather than permutations). We also make the following conjecture based on some initial numerical results.

Conjecture 8. *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 with a bypass. 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).*

References

[1] M. D. Atkinson, Restricted permutations, *Discrete Math.*, 195 (1999) 27–38.

[2] M. H. Albert, C. Bean, A. Claesson, E. Nadeau, J. Pantone, and H. Ulfarsson, Combinatorial Exploration: An algorithmic framework for enumeration, *Memoirs of the American Mathematical Society* (to appear).

[3] D. Avis and M. Newborn, On pop-stacks in series, *Utilitas Math.*, 19 (1981) 129–140.

[4] M. D. Atkinson and J.-R. Sack, Pop-stacks in parallel, *Inform. Process. Lett.* 70 (1999), 63–67.

[5] C. Bean, J. S. Eliasson, T. K. Magnusson, E. Nadeau, J. Pantone, and H. Ulfarsson, Tilings: combinatorial exploration for permutation classes, at https://github.com/PermutaTriangle/Tilings/.

[6] D. E. Knuth, *The Art of Computer Programming, Vol. 1*, Boston, Addison-Wesley, 1968.

[7] R. Smith and V. Vatter, The enumeration of permutations sortable by pop stacks in parallel, *Inform. Process. Lett.* 109 (2009), 626–629.

[8] V. Vatter, Finding regular insertion encodings for permutation classes, *J. Symbolic Comput.* 47 (2012) 259–265.