Sorting with stacks and queues: some recent developments

L. Ferrari

Dipartimento di Matematica e Informatica “U. Dini", Universitá degli Studi di Firenze,
Viale Morgagni 65, 50134 Firenze, Italy
luca.ferrari@unifi.it

Permutation Patterns Virtual Workshop 2021, 14-15 June 2021.

In collaboration with: Giulio Cerbai, Lapo Cioni, Anders Claesson, Einar Steingrímsson

Stack sorting (and relatives...)

General framework:

INPUT - a permutation π;

MACHINE - a network of devices (may be stacks, queues, etc...), connected in series or in parallel (or in some more fancy way...);

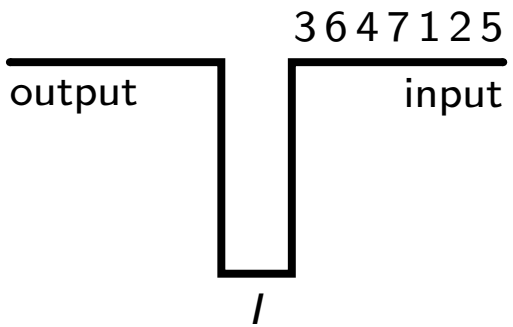
OUTPUT - another permutation $f(\pi)$, which is hopefully the identity, otherwise somehow "closer" to the identity than the original permutation π.

Main questions

- Characterize and enumerate sortable permutations.
- Design optimal sorting algorithms.
- Investigate properties of the associated map.

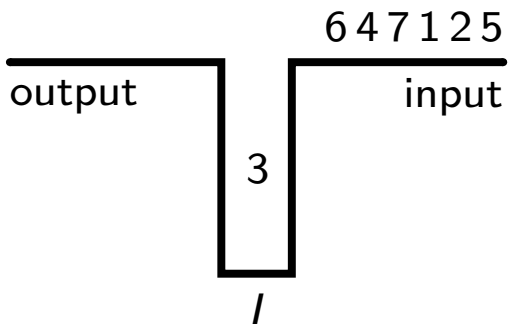
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



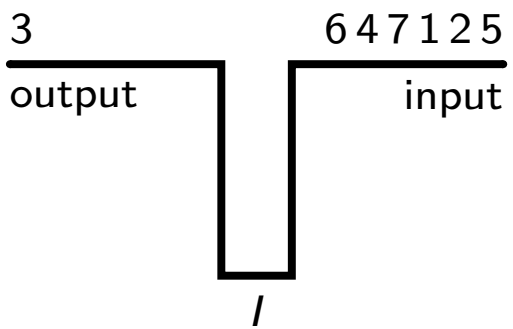
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



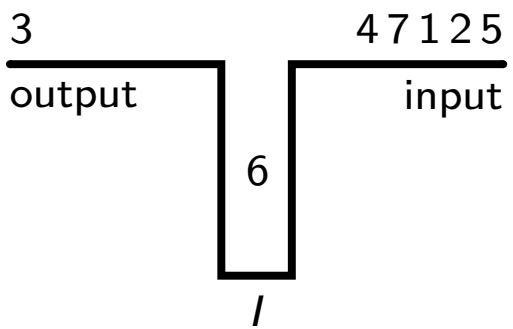
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



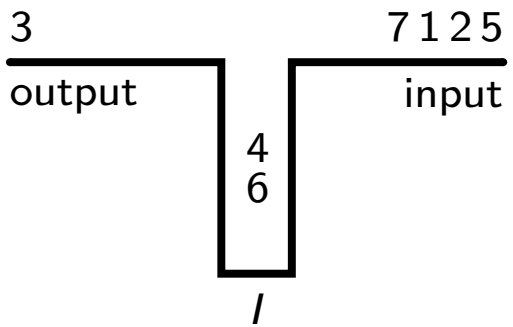
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



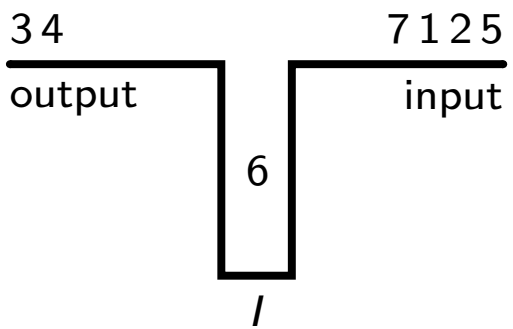
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



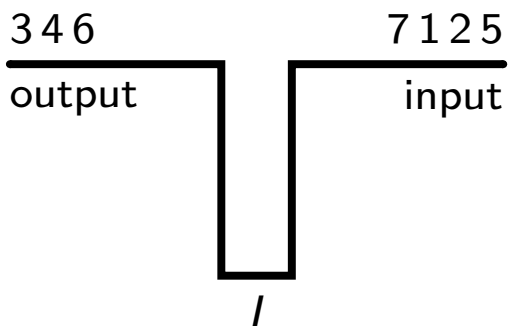
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



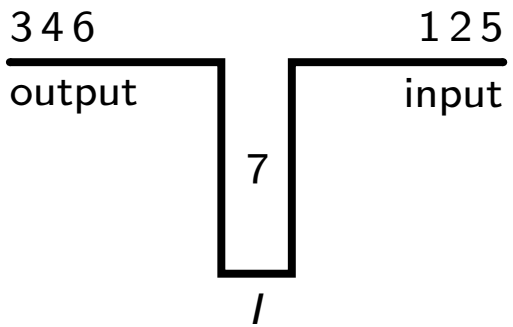
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



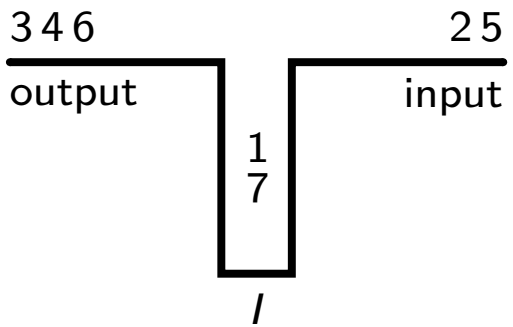
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



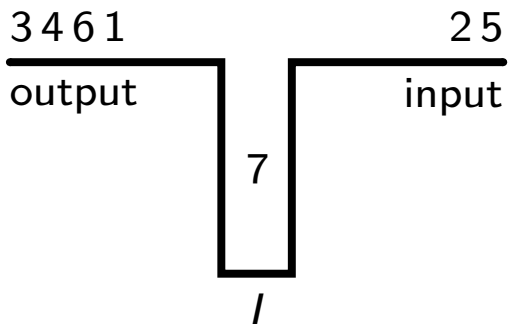
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



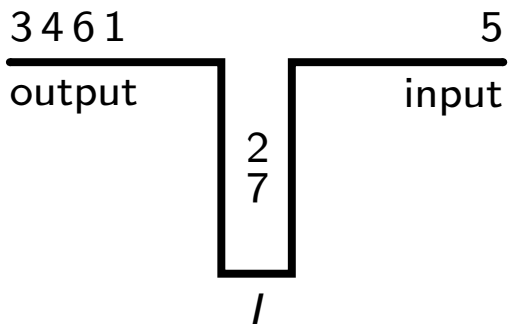
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



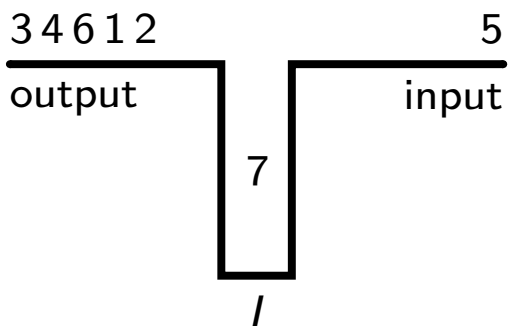
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



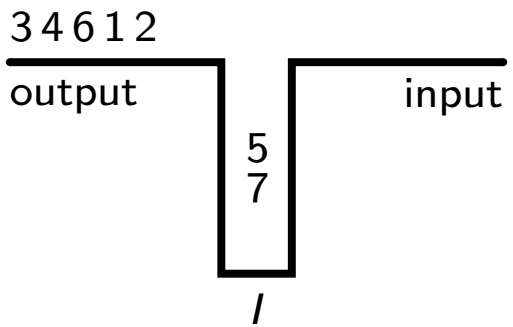
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



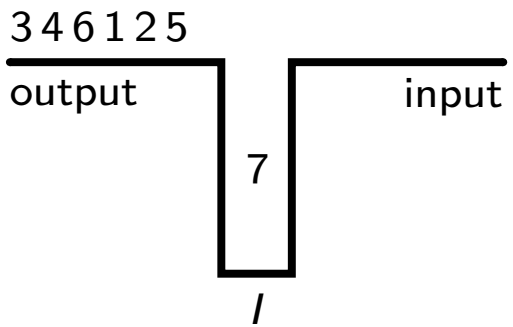
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



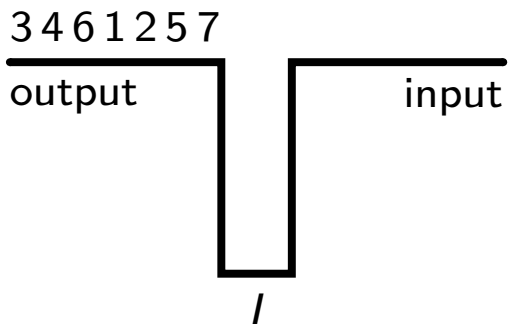
Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



Stacksort: where it all started

Sorting permutations using a stack [Knuth, 1968]:



What do we know about sorting with one stack?

We know (almost) everything:

- sortable permutations: $Av(231)$;
- enumeration of sortable permutations: C_n (Catalan numbers);
- optimal algorithm (Stacksort) [Knuth, West];
- what about preimages of the associated Stacksort map?

These are very neat and beautiful results! Several researches on complex networks of sorting devices immediately begun.

From stacksort to networks

- [Even and Itai, 1971]: graph encoding of sorting with parallel queues and stacks;
- [Tarjan, 1972]: characterizazion of permutations sortable by queues in parallel; some partial information on permutations sortable by stacks in parallel (only in 2015, [Albert and Bousquet-Melou] gave more precise results for sorting with two stacks in parallel);
- [Pratt, 1973]: deque-sortable permutations;
- [Atkinson and Sack, 1999]: characterizzation of permutations sortable by popstacks in parallel (later, in 2009, [Smith and Vatter] showed that the generating function of sortable permutations is rational).

Very difficult problems even with very small networks!

Two stacks in series

Still open in its full generality.

- No characterization of sortable permutations is known; but we know [Murphy, 2002] that they are a class with infinite basis, even if we don't know the elements of the basis.
- No enumeration of sortable permutations is known; but we know [Elvey Price and Guttmann, 2017] some information on the asymptotic behavior of the associated generating function.
- No optimal sorting algorithm is known; but we know [Pierrot and Rossin, 2014] that deciding if a permutation is sortable is polynomial.

Possible restrictions:

- fix the algorithm;
- set constraints on the contents of the stacks.

Two stacks in series

Several special cases have been considered.

- 2-stacksort [West, 1990]: right-greedy algorithm ($\rightsquigarrow$ increasing stacks):
  - characterization in terms of barred patterns [West],
  - exact enumeration [Zeilberger, 1992],
  - a lot of interesting bijective combinatorics
    - nonseparable planar maps and left ternary trees [Cori, Jacquard and Schaeffer, 1997; Del Lungo, Del Ristoro and Penaud ,2000],
    - generalized Tamari intervals [Fang and Preville-Ratelle, 2017; Preville-Ratelle and Viennot, 2017],
    - fighting fish [Duchi, Guerrini, Rinaldi and Schaeffer, 2017; Fang, 2018].
    - extension to 3 stacks: mesh (decorated) patterns [Ulfarsson, 2011; Claesson and Ulfarsson, 2012], lower/upper bounds [Bona, 2020; Defant, 2020].

Two stacks in series

Several special cases have been considered.

- Increasing stacks [Atkinson, Murphy and Ruškuc, 2002]:
  - characterization in terms of patterns (infinite basis),
  - exact enumeration (same counting sequence as 1342-avoiding permutations, first example of an (extremely!) unbalanced Wilf equivalence),
  - optimal sorting algorithm (left-greedy, as opposed to the right-greedy algorithm of West).

Two stacks in series

Several special cases have been considered.

- Popstacks [Avis and Newborn, 1981]:
  - characterization in terms of patterns: separable permutations (for any number of popstacks in series),
  - exact enumeration (Schröder numbers, [Shapiro and Stephens, 1991]),
  - optimal sorting algorithm.
- Popstacks, with right-greedy algorithm [Pudwell and Smith, 2017]:
  - characterization in terms of barred patterns,
  - exact enumeration (rational generating function),
  - links with other combinatorial structures (polyominoes),
  - generalization to k popstacks ([Claesson and Guðmundsson, 2018]: rational generating function).

Two stacks in series

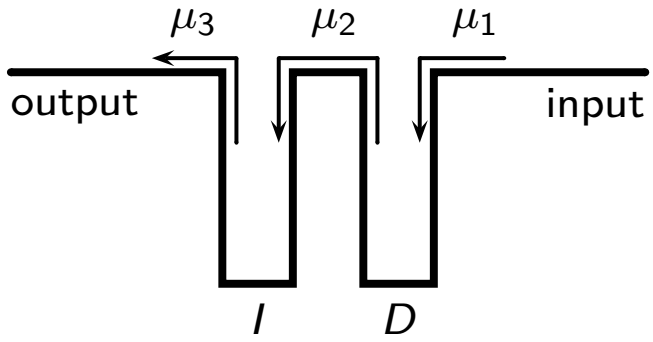
Several special cases have been considered.

- A stack and a popstack in series (with or without a queue between them) [Smith and Vatter, 2014]:
  - characterization in terms of three patterns in the PS case,
  - characterization in terms of divided patterns in the PQS case (but no information on the basis),
  - exact enumeration in the PS case (algebraic generating function).

Rebecca Smith's *DI* machine (2014)

A decreasing stack followed by an increasing stack.
Operations:

- μ_1: from input to D;
- μ_2: from D to I;
- μ_3: from I to output.



Optimal sorting algorithm: when the current element in the input is π_i:

- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D) < \pi_i < TOP(I)$;
- μ_2: if $TOP(D) < TOP(I)$;
- μ_3: otherwise.

Characterization and exact enumeration of sortable permutations:

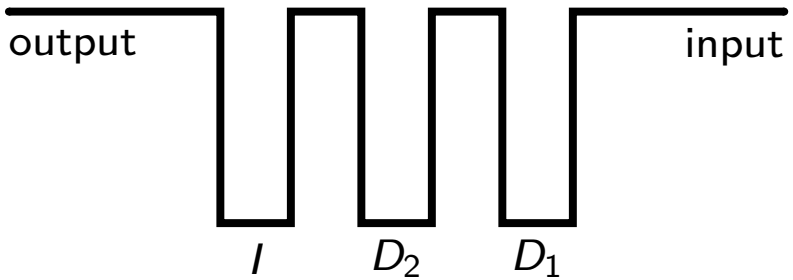
- π sortable iff $\pi \in Av(3241, 3142)$;
- $|Av_{n+1}(3241, 3142)|$ is the n-th Schröder number.

Comparison with Stacksort

Sortable permutations	Stacksort	DI machine
Characterization	$AV(231)$	$Av(3241, 3142)$
Enumeration	Catalan numbers	Schröder numbers
Optimal sorting algorithm	yes	yes

What about a D^2I machine?

A D^2I machine: comparison with two stacks in series



Sortable permutations	two stacks in series	D^2I machine
Characterization	infinite basis	infinite basis
Enumeration	???	???
Optimal sorting algorithm	???	YES!!!

A D^2I machine: infinite basis

Theorem (Cerbai, Cioni and F., 2020)

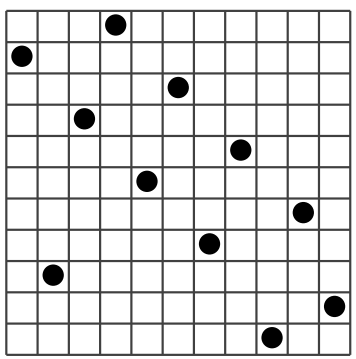
For $j \geq 0$, define the permutation:

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

where:

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

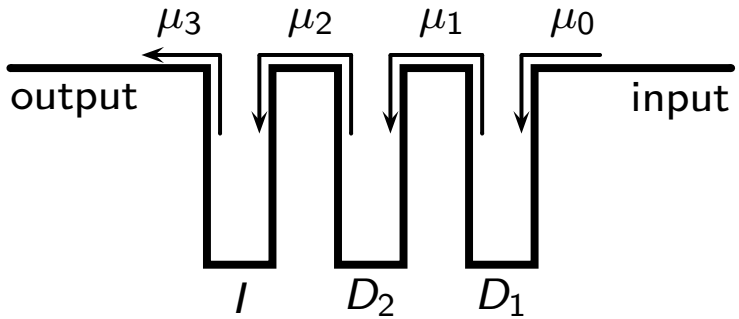
Then the set of permutations $\{\alpha_j\}_{j\geq 0}$ constitutes an infinite antichain and is a subset of the basis of the class of sortable permutations.



A D^2I machine: optimal sorting algorithm

Operations:

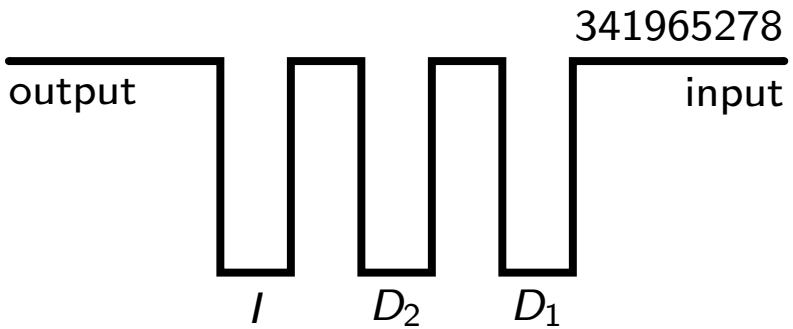
- μ_0: from input to D_1;
- μ_1: from D_1 to D_2;
- μ_2: from D_2 to I;
- μ_3: from I to output.



Optimal sorting algorithm [Cerbai, Cioni and F., 2020]: when the current element in the input is π_i:

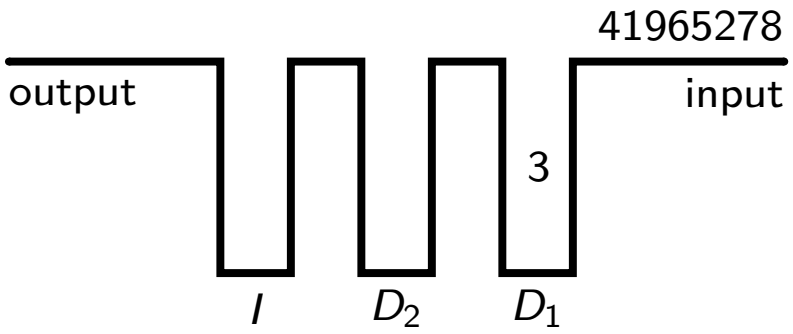
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



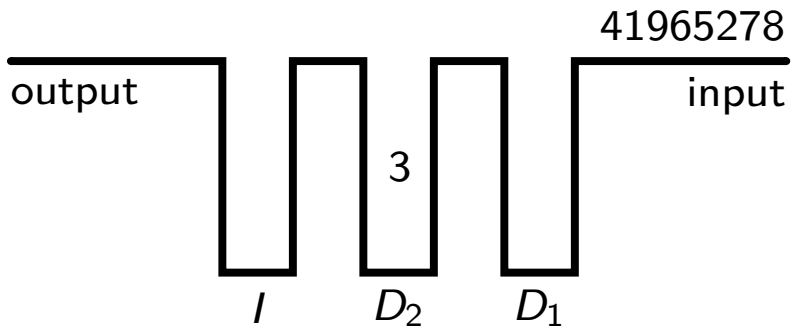
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



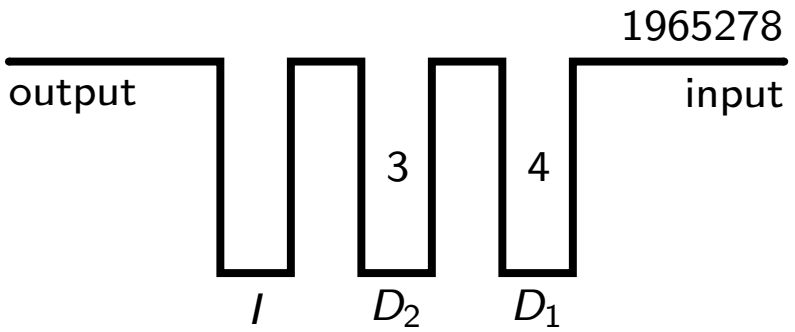
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



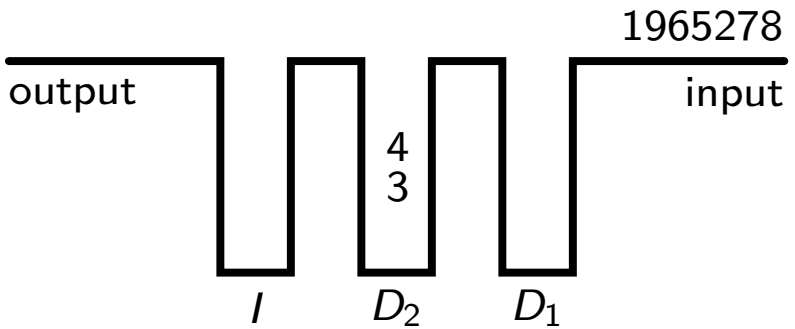
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



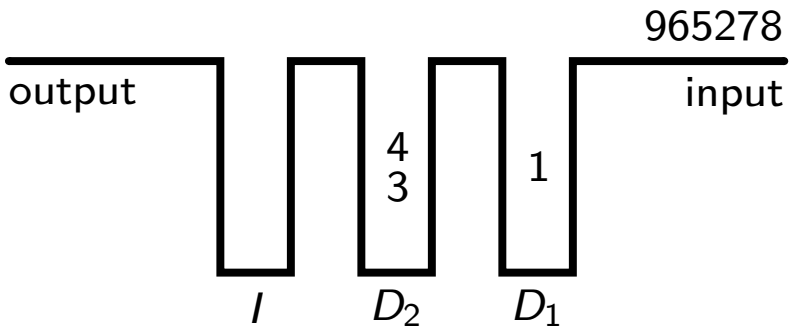
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



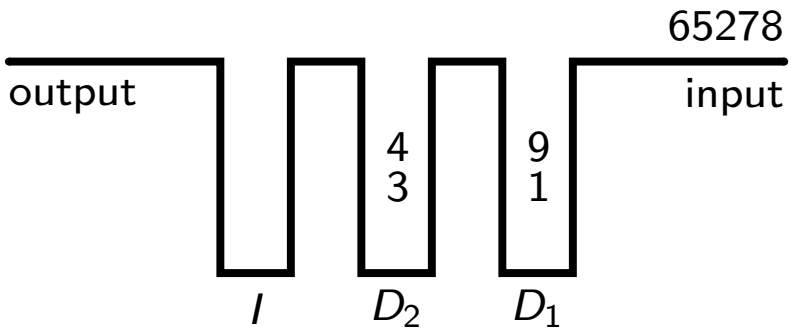
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



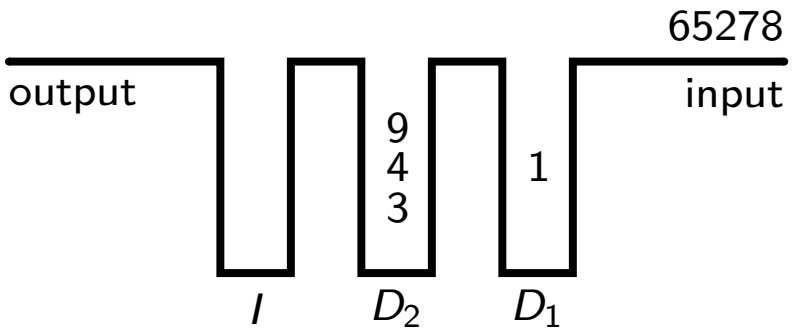
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



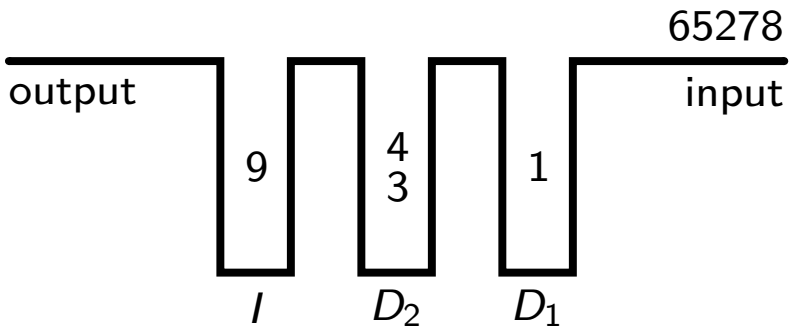
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



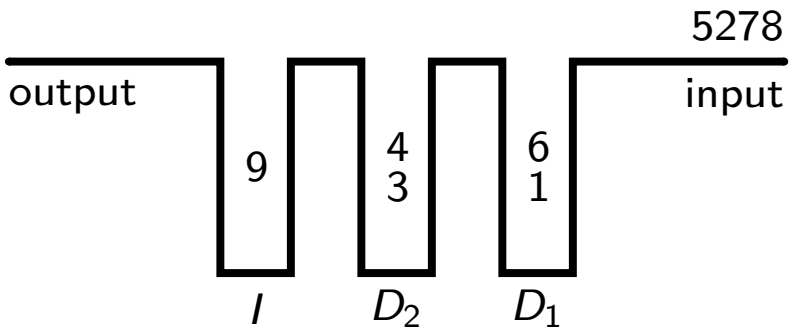
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



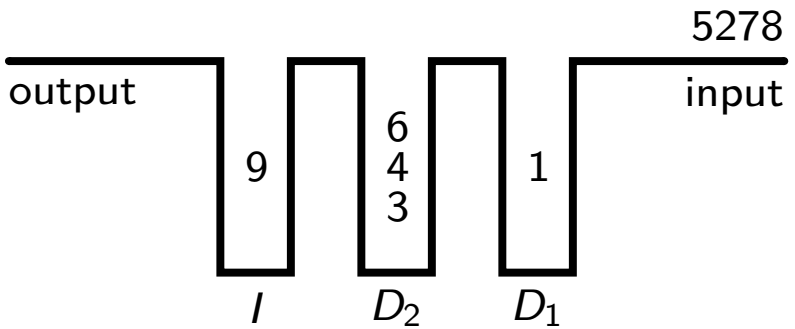
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



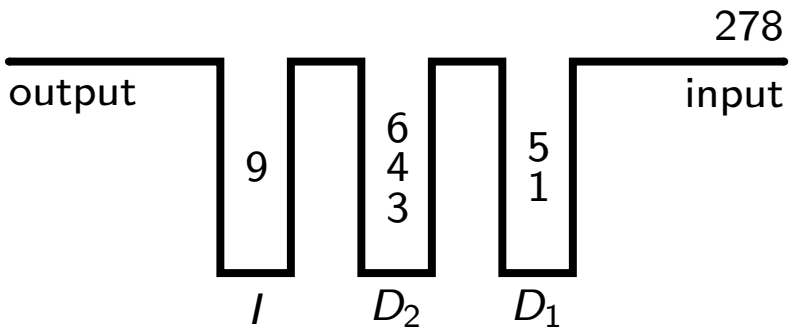
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



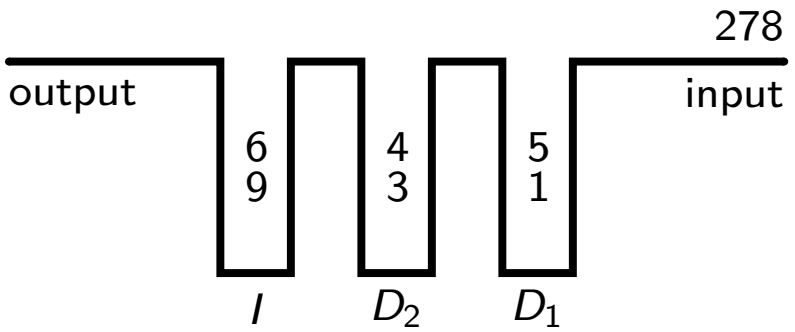
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



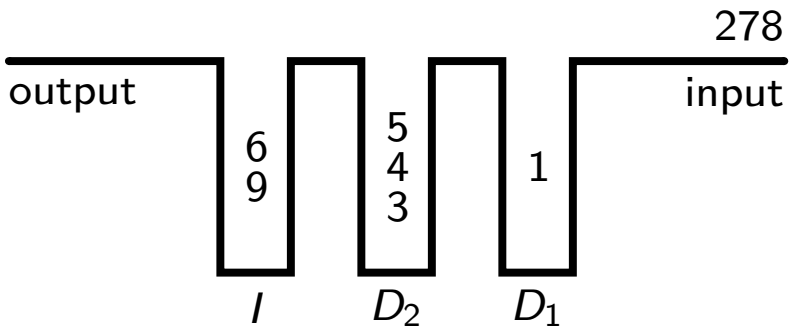
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



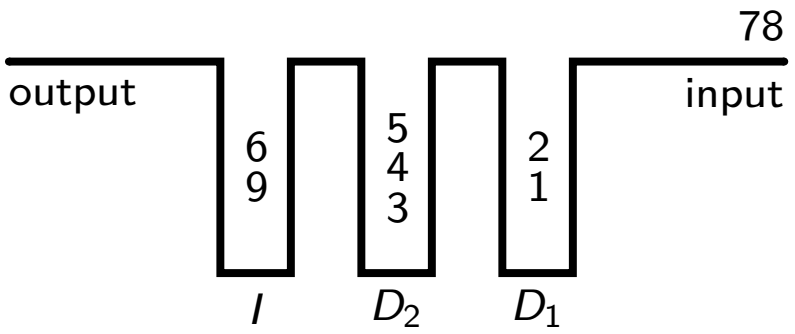
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



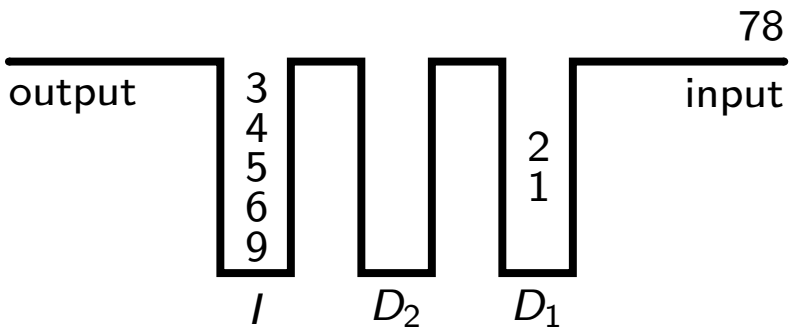
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



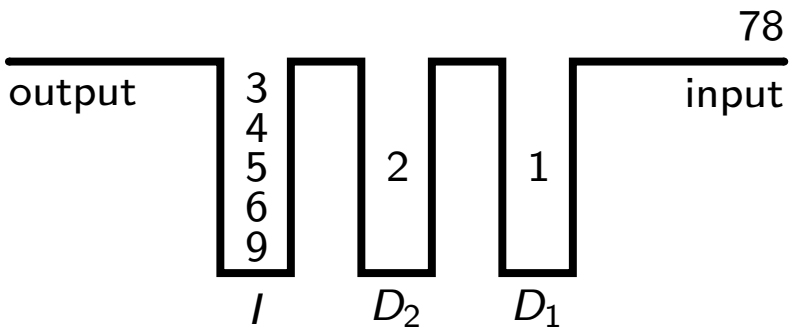
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



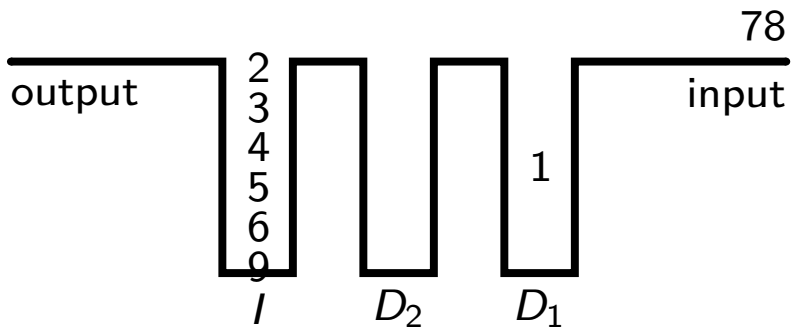
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



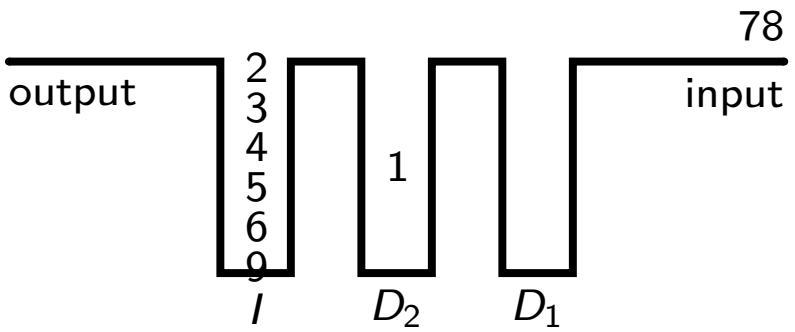
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



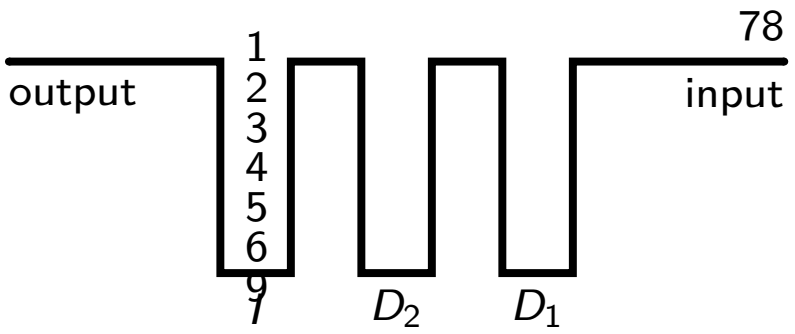
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



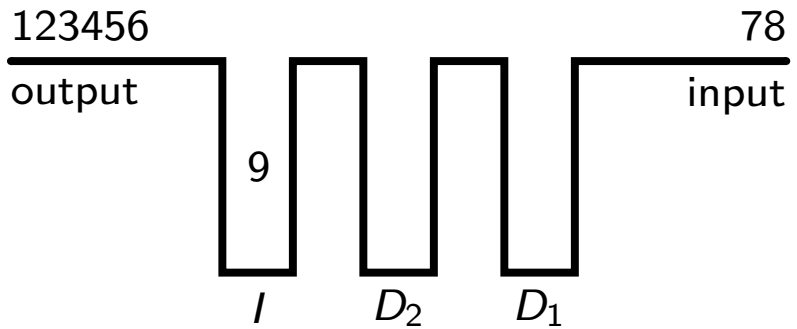
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



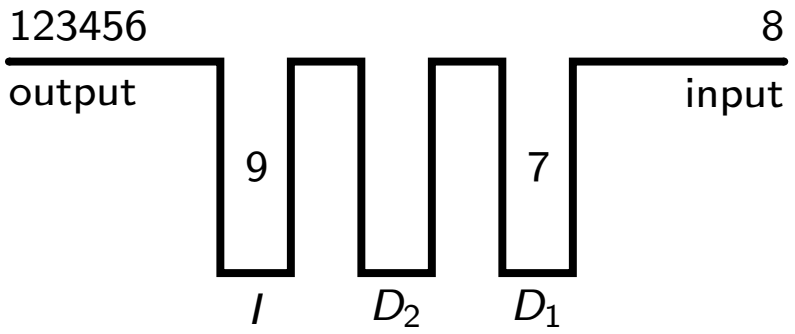
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



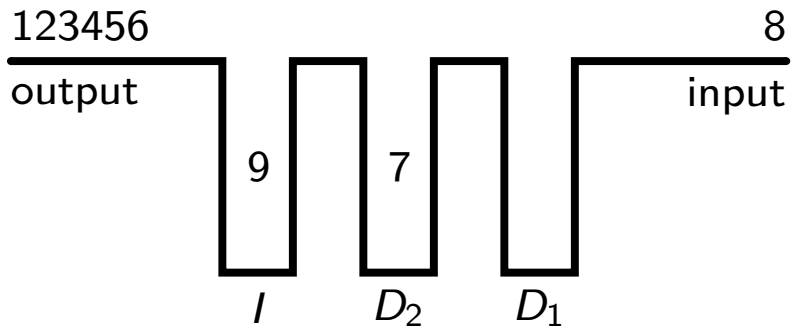
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



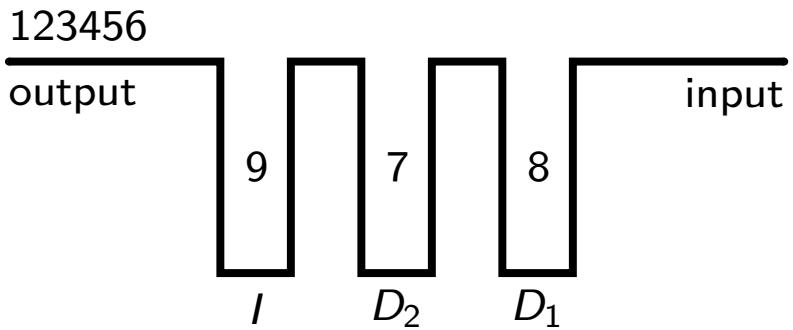
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



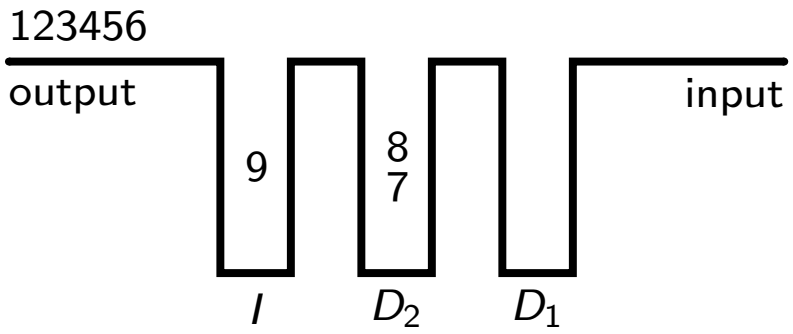
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



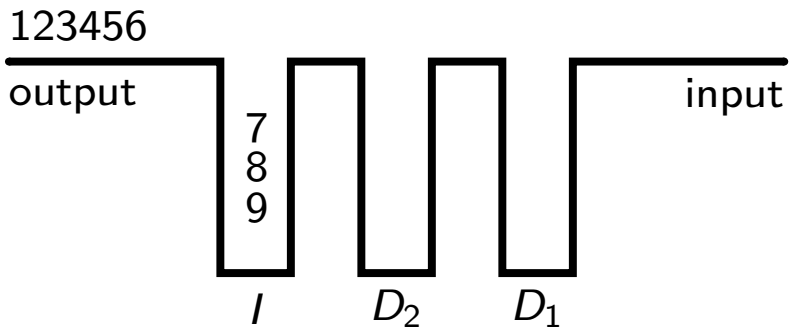
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



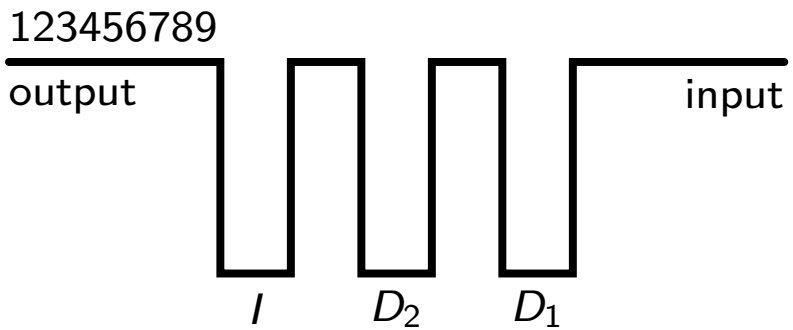
- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm



- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

A D^2I machine: optimal sorting algorithm

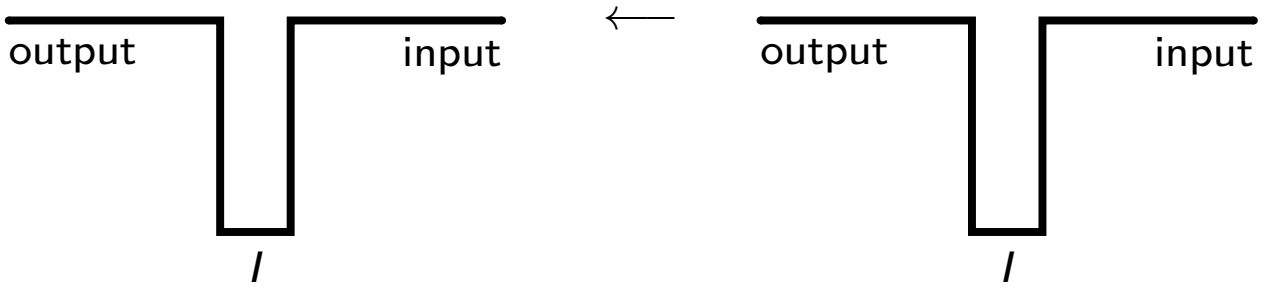


- μ_3: if $TOP(I)$ is the next element to output;
- μ_1: if $TOP(D_2) < TOP(D_1) < TOP(I)$;
- μ_0: if $TOP(D_1) < \pi_i < TOP(I)$ and the sequence of elements from π_i to the first element larger than $Top(D_2)$ is increasing;
- μ_2: if $TOP(D_2) < TOP(I)$;
- μ_3: otherwise.

Can we generalize to D^kI?

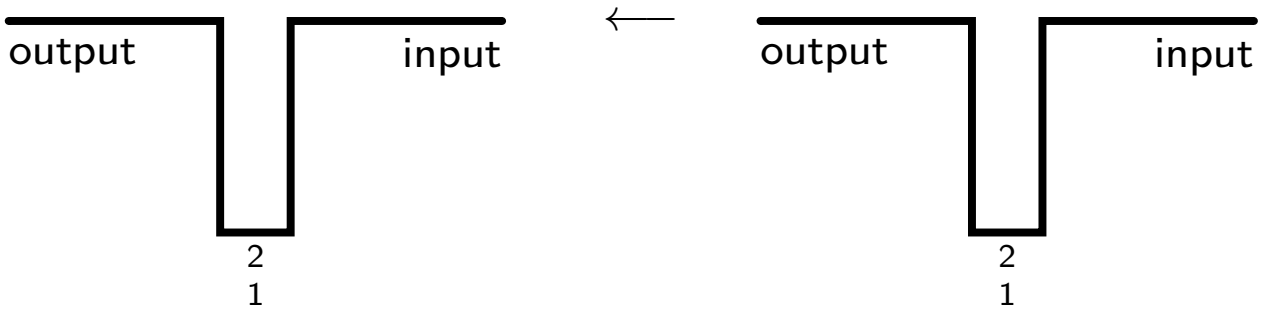
Two passes from an increasing stack

West-2-stack sorting:

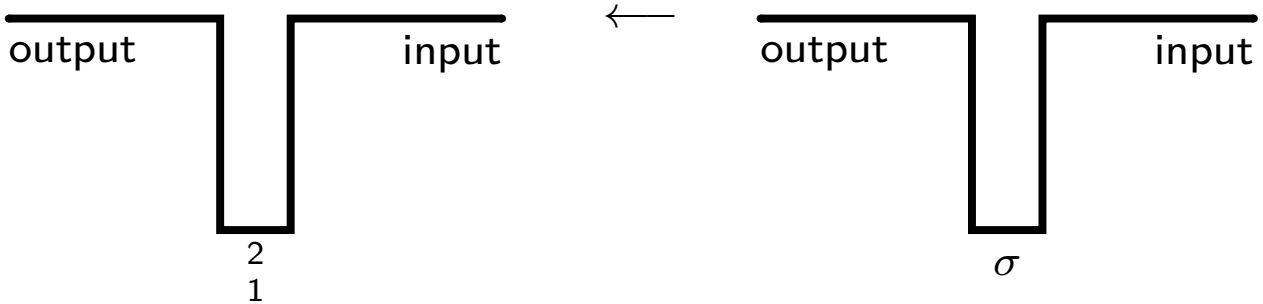


Two passes from pattern-restricted stacks

West-2-stack sorting:



σ-stack sorting [Cerbai, Claesson and F., 2020]:



Why pattern-restricted stacks?

Motivations:

- better understanding of the general “two stacks in series" sorting problem;
- development of a family of sorting devices that may help in tackling the general “two stacks in series" sorting problem;
- nice enumerative and bijective combinatorics!

Main questions:

- characterization of σ-sortable permutations;
- enumeration of σ-sortable permutations.

$|\sigma| = 2$

$\sigma = 21$: West-2-stack sorting machine.

- characterization in terms of barred patterns $\rightsquigarrow$ not a class!
- nice enumeration formula: $\frac{2}{(n+1)(2n+1)}\binom{3n}{n}$.

$\sigma = 12$: Rebecca Smith's device (a decreasing stack, followed by an increasing stack), but with a right-greedy algorithm.

- characterizazion in terms of a single classical pattern: $\mathrm{Av}(213)$ $\rightsquigarrow$ is a class!
- (obviously) nice enumeration formula: C_n.

Additional interesting question: for what σ do we get classes?

Classes vs. non-classes

When $|\sigma| = 3$, we get that 321-sortable permutations are a class, whose enumeration sequence is 2^{n-1}. In all the remaining cases, σ-sortable permutations are not a class:

σ	σ-sortable permutation	non-σ-sortable pattern
123	4132	132
132	2413	132
213	4132	132
231	361425	1324
312	3142	132

Looking at more data, we find a surprising conjecture for the number of σ whose associated σ-sortable permutations are not a class:

- σ of length 3 $\rightsquigarrow$ 5 non-class;
- σ of length 4 $\rightsquigarrow$ 14 non-classes;
- σ of length 5 $\rightsquigarrow$ 42 non-classes;
- σ of length 6 $\rightsquigarrow$ 132 non-class;

These are the Catalan numbers!

Classes vs. non-classes

Given a permutation σ, denote with $\hat{\sigma}$ the permutation obtained from σ by swapping its first two elements.

Theorem (Cerbai, Claesson and F., 2020)

$\mathrm{Sort}(\sigma)$ *is not a class iff* $\hat{\sigma} \in \mathrm{Av}(231)$.

What about classes?

Theorem (Cerbai, Claesson and F., 2020)

If $\hat{\sigma}$ *contains 231, then* $\mathrm{Sort}(\sigma) = \mathrm{Av}(132, \sigma^r)$.

Open problem: classify σ-machines in terms of the number of permutations they sort (*Wilf-equivalence* for σ-machines).

Patterns of length 3

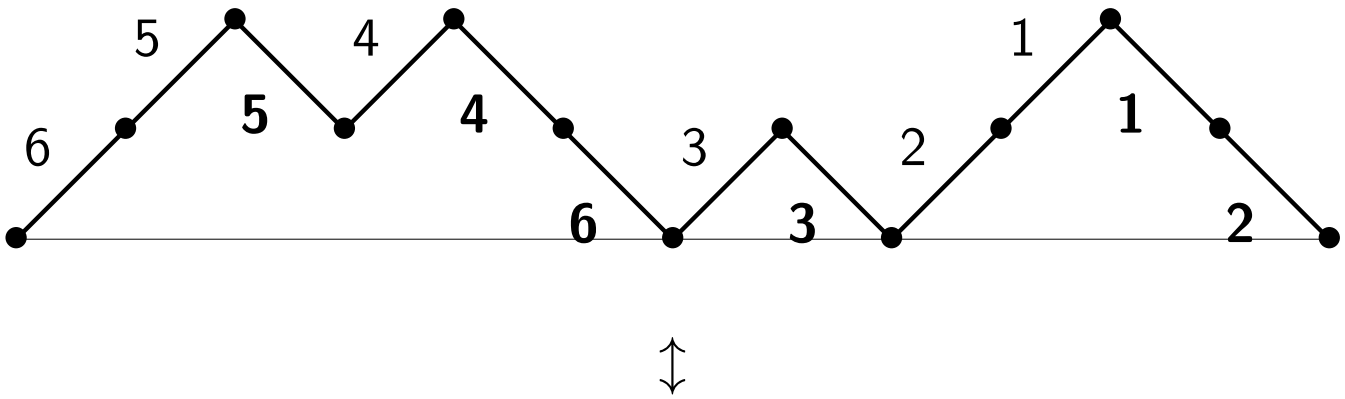
Three patterns have been solved:

- $\sigma = 321$: 2^{n-1} $(n \geq 1)$, sequence A011782 in OEIS $\rightsquigarrow$ Dyck paths of height at most 2 (generalization to $k(k-1)\cdots 21$ and Dyck paths of height at most $k-1$) [Cerbai, Claesson and F., 2020];
- $\sigma = 123$: $1 + \sum_{k=1}^{n-1}(n-k)C_k$ $(n \geq 1)$, sequence A294790 in OEIS $\rightsquigarrow$ Schröder paths avoiding *UHD* [Cerbai, Claesson and F., 2020];
- $\sigma = 132$: $\sum_{k=0}^{n-1}\binom{n-1}{k}C_k$ $(n \geq 1)$, sequence A007317 in OEIS $\rightsquigarrow$ restricted growth functions avoiding 12231 [Cerbai, Claesson, F. and Steingrímsson, 2020].

Three patterns are still open:

- $\sigma = 213$: $1, 1, 2, 5, 16, 62, 273, 1307, 6626, 35010, 190862, 1066317, \ldots$, not in OEIS;
- $\sigma = 231$: $1, 1, 2, 6, 23, 102, 496, 2569, 13934, 78295, 452439, 2674769, \ldots$, not in OEIS;
- $\sigma = 312$: $1, 1, 2, 5, 15, 52, 201, 843, 3764, 17659, 86245, 435492, \ldots$, sequence A202062 in OEIS $\rightsquigarrow$ ascent sequences avoiding 201 (for which however no formula is known).

$\sigma = 321$



5 4 6 3 1 2

$$|\mathrm{Sort}_n(321)| = |\mathrm{Av}_n(132, 123)|$$
$$= |\{\text{Dyck paths of semilength } n \text{ and height} \leq 2\}| = 2^{n-1}$$

$\sigma = 123$

$\pi = 5\,6\,7\,4\,8\,9\,1\,3\,2$ $\qquad\qquad$ $|\pi| = n$

$\sigma = 123$

$$\pi = \underbrace{5\,6}_{L}\ \underbrace{7\,4\,8\,9\,1\,3\,2}_{w} \qquad |\pi| = n$$

L: initial sequence of consecutive ascents deprived of its last element

$$|L| = r$$

$\sigma = 123$

$$\pi = \underbrace{5\ 6}_{L}\ \underbrace{7\ 4\ \not{8}\ \not{9}\ 1\ 3\ 2}_{w} \qquad |\pi| = n$$

$$|L| = r$$

$$|\{\text{elements of } w \text{ larger than its first element}\}| = s$$

$\rho =$ permutation obtained from w by removing green elements

$\sigma = 123$

$$\pi = \underbrace{5\ 6}_{L}\ \underbrace{7\ 4\ 8\ 9\ 1\ 3\ 2}_{w} \qquad |\pi| = n$$

$$|L| = r$$

$$|\{\text{elements of } w \text{ larger than its first element}\}| = s$$

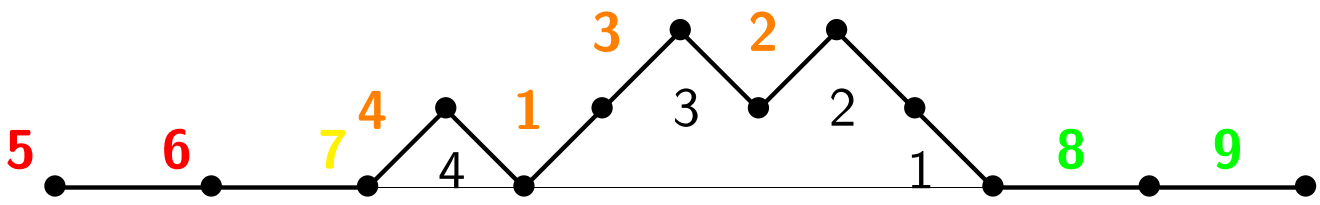
τ = permutation obtained from ρ by removing the first (=maximum) element

$$|\tau| = n - r - s - 1 \qquad \tau \in \mathrm{Av}(213)$$

$\sigma = 123$

$$\pi = \underbrace{5\,6}_{L}\ \underbrace{7\,4\,8\,9\,1\,3\,2}_{w}$$

$$\updownarrow$$



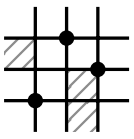
$$|\mathrm{Sort}_n(123)| = |\{\text{Schröder paths of semilength } n-1 \text{ avoiding } UHD\}|$$

$$\sum_{n\geq 0} |\mathrm{Sort}_n(123)| x^n = \frac{1}{1-x} + \frac{1}{1-x}\left(x(C(x)-1)\right)\frac{1}{1-x} = \frac{1-2x+xC(x)}{(1-x)^2}$$

$|\mathrm{Sort}_n(123)| =$ subtract n from partial sums of partial sums of Catalan numbers

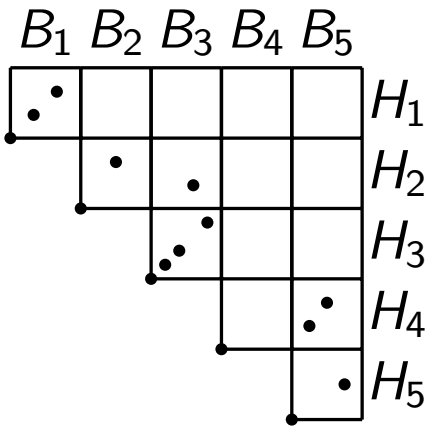
$\sigma = 132$

Theorem

$\mu =$ 

$$\mathrm{Sort}(132) = \mathrm{Av}(2314, \mu)$$

Grid decomposition



Restricted growth function: blocks are horizontal strips H_i

111223332345445

Corollary

$$|\mathrm{Sort}_n(132)| = |\mathcal{R}_n(12231)|$$
$$= \sum_{k=0}^{n-1} \binom{n-1}{k} C_k$$

(sequence A007317 in the Encyclopedia). Last equality is due to [Jelínek and Mansour, 2008].

Further work on σ-sorting

- Γ-sortable permutations, where Γ is any set of patterns; when $|\Gamma| = 2$, several cases have been solved [Baril, Cerbai, Khalil and Vajnovszki, 2021+];
- characterize sorted permutations and investigate fertility [Berlow, 2021+; Defant and Zheng, 2021];
- σ-sortable words of various types: unrestricted words [Defant and Kravitz, 2020], Cayley permutations [Cerbai, 2021+], (modified) ascent sequences [Cerbai and Claesson, 2021+].

Much more on all this in Giulio Cerbai's PhD thesis (2021)!

Queuesort: characterization and enumeration of sortable permutations

What about replacing the stack with a queue in Stacksort?

Not a big deal: the only sortable permutations are the increasing ones!

However, if we allow the bypass of the queue, we obtain an interesting sorting algorithm, called *Queuesort*.

- Sortable permutations are 321-avoiding permutations; more generally, permutations which are sortable by k queues in parallel (again with a bypass) are those avoiding $(k+2)(k+1)k\cdots 21$.
- Sortable permutations are counted by Catalan numbers.

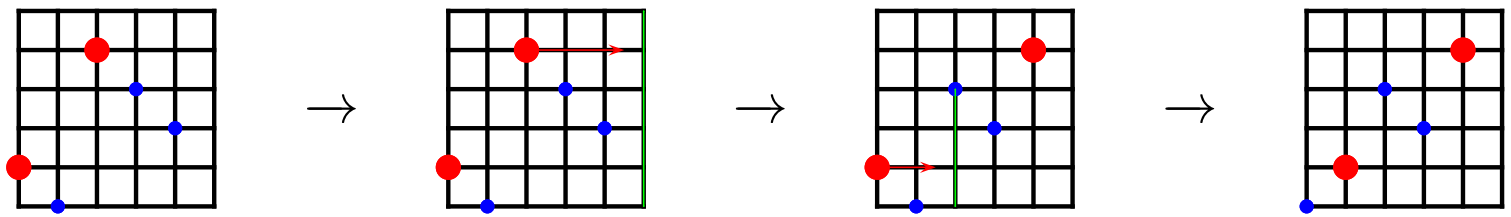
Preimages under Queuesort

The investigation of preimages of the Stacksort operator begun with [Bousquet-Mélou, 2000] and recently has received a lot of attention ([Defant, 2017; Defant, Engen and Miller, 2020], to cite just a few). It is a rather difficult problem, so results are difficult to obtain and there is still much that remains unknown.

Curiously, it seems that nothing had been done done for the (somehow simpler) Queuesort operator. Hopefully, more (and neater) results should be obtained! The next slides will describe what we know on the combinatorics of preimages under Queuesort [Cioni and F., 2021+].

The Queuesort map

An alternative (but equivalent) description of Queuesort:



Corollary

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

A recursive characterization of preimages

LTR-max decomposition of π:

$$\pi = M_1 P_1 M_2 P_2 M_3 \cdots M_{k-1} P_{k-1} M_k$$

where the M_i's are all the maximal sequences of contiguous LTR maxima of π.

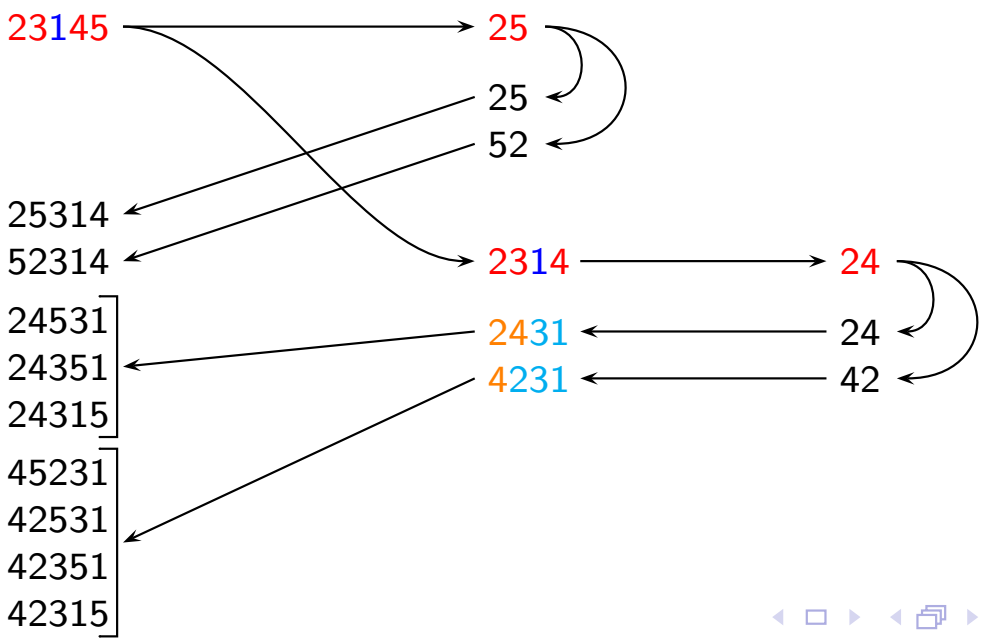
$$351268974 \rightsquigarrow \underbrace{35}_{M_1}\underbrace{12}_{P_1}\underbrace{689}_{M_2}\underbrace{74}_{P_2}\underbrace{}_{M_3}$$

For any string α, denote with α' the string obtained from α by removing the last element.
Denote with μ_i the last (=largest) element of M_i.

A recursive characterization of preimages

$\pi = M_1 P_1 \cdots M_{k-1} P_{k-1} M_k \in S_n$:

- if π is the identity permutation, then the preimages of π are precisely the 321-avoiding permutations (of the same length);
- otherwise
  - compute all the preimages of $M_1 P_1 \cdots M_{k-2} P_{k-2} M'_{k-1} n$ and concatenate them with $\mu_{k-1} P_{k-1} M'_k$;
  - if $|M_k| \geq 2$, then compute all the preimages $N_1 R_1 \cdots N_{s-1} R_{s-1} N_s$ of π' and insert n in each of the positions to the right of N_{s-1}.



How many permutations with a given number of preimages?

$$Q_n^{(k)} = \{\pi \in S_n \,|\, |q^{-1}(\pi)| = k\}$$

$$q_n^{(k)} = |Q_n^{(k)}|$$

We already know:

$$Q_n^{(0)} = \{\pi \in S_n \,|\, \pi_n \neq n\}$$

$$q_n^{(0)} = (n-1)! \cdot (n-1)$$

What happens for other values of k?

How many permutations with a given number of preimages?

- $Q_n^{(1)} = \{\pi \in S_n \mid \pi_n = n \text{ and } \pi \text{ does not have two adjacent LTR maxima}\}$

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

- $Q_n^{(2)} = \{\pi \in S_n \mid \pi_n = n \text{ and the only adjacent LTR maxima are the first two elements}\}$

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

$$q_n^{(2)} = (n-1)! - 2q_n^{(1)}$$

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

How many permutations with a given number of preimages?

What about larger values of k?

We know that there exists permutations having any number of preimages...

Proposition

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

... with only one exception!

Proposition

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

How many preimages for a given permutation?

Theorem

For $\pi = M_1 P_1 M_2 \in S_n$, $M_2 \neq \emptyset$:

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

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

where the b's are (some version of) the ballot numbers and $\omega_{m_2,t}(p_1)$ is a polynomial in p_1 of degree $m_2 - t - 1$, for all t.

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

Further work on preimages of sorting operators

- Iterates of the Queuesort map
- Further properties of the sets $Q_n^{(k)}$ (pattern avoidance,...)
- Pattern avoiding queues in series
- Preimages of maps associated with other sorting algorithms: C-machines (thanks Vince for the suggestion!), Bubblesort (work in progress with Lapo and Mathilde),...

Thank you for listening...

and see you in person next year!!!