

Enumerating permutations sortable by k passes through a pop-stack

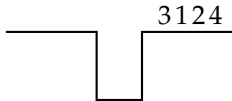
Anders Claesson

Bjarki Ágúst Guðmundsson

University of Iceland

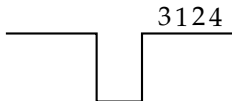
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



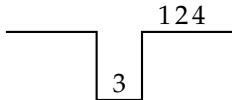
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



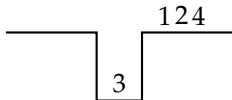
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



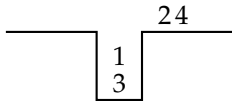
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



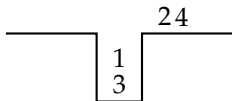
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



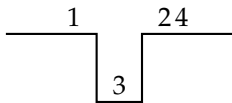
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



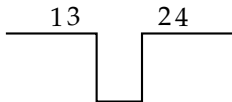
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



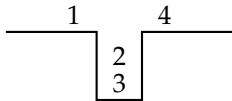
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



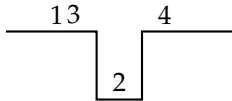
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



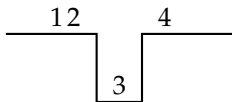
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



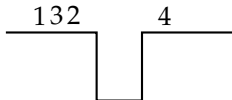
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



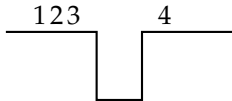
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



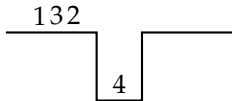
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



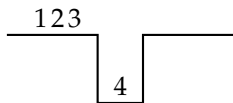
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



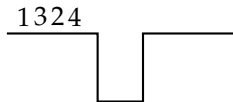
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



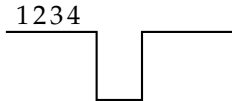
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



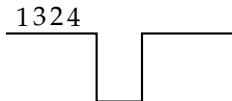
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



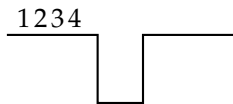
Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



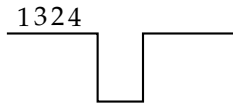
Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.



Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.



Theorem (Pudwell & Smith, 2017)

A permutation π is sortable by *two passes through a pop-stack* if and only if π avoids 2341, 3412, 3421, 4123, 4231, 4312, $4\bar{1}352$ and $413\bar{5}2$.

Theorem (Knuth, 1968)

A permutation π is sortable by a *stack* if and only if π avoids 231.

$$\frac{1 - \sqrt{1 - 4x}}{2x}$$

Theorem (Avis & Newborn, 1981)

A permutation π is sortable by a *pop-stack* if and only if π avoids 231 and 321.

$$\frac{x - 1}{2x - 1}$$

Theorem (Pudwell & Smith, 2017)

A permutation π is sortable by *two passes through a pop-stack* if and only if π avoids 2341, 3412, 3421, 4123, 4231, 4312, $4\bar{1}352$ and $413\bar{5}2$.

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

Stacks vs pop-stacks

A **stack** is a LIFO data structure with two operations:

- ▶ **Push**: Add an element to the top of the stack.
- ▶ **Pop**: Remove the top-most element from the stack.

A **pop-stack** is a LIFO data structure with two operations:

- ▶ **Push**: Add an element to the top of the stack.
- ▶ **Pop**: Remove **all** elements from the stack.

We'll insist that elements on the stack are increasing when read from top to bottom and sort greedily w.r.t. the push operation.

How many permutations are sortable by k passes through a pop-stack?

$$k = 1 \qquad \frac{x - 1}{2x - 1}$$

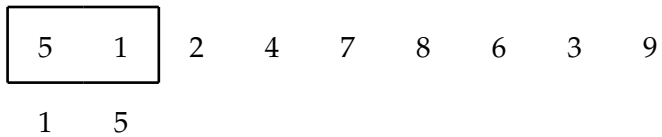
$$k = 2 \qquad \frac{x^3 + x^2 + x - 1}{2x^3 + x^2 + 2x - 1}$$

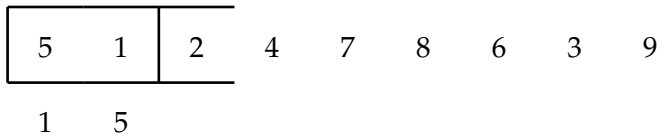
$$k \geq 3 \qquad \text{Rational?}$$

5 1 2 4 7 8 6 3 9

5 1 2 4 7 8 6 3 9

5 1 2 4 7 8 6 3 9





5	1	2
---	---	---

4 7 8 6 3 9

1 5 2

5	1	2	4	7	8	6	3	9
1	5	2						

5	1	2	4
---	---	---	---

7 8 6 3 9

1 5 2 4

5	1	2	4	7	8	6	3	9
1	5	2	4					

5	1	2	4	7
---	---	---	---	---

8 6 3 9

1 5 2 4 7

5	1	2	4	7	8	6	3	9
1	5	2	4	7				

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1 5 2 4 7

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1 5 2 4 7

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1 5 2 4 7 3 6 8

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1 5 2 4 7 3 6 8

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1 5 2 4 7 3 6 8 9

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1	5	2	4	7	3	6	8	9
---	---	---	---	---	---	---	---	---

1 2 5

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1	5	2	4	7	3	6	8	9
---	---	---	---	---	---	---	---	---

1 2 5

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1	5	2	4	7	3	6	8	9
---	---	---	---	---	---	---	---	---

1 2 5 4

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1 2 5 4

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1 2 5 4

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1	5	2	4	7	3	6	8	9
---	---	---	---	---	---	---	---	---

1 2 5 4 3 7

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1 2 5 4 3 7

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1	5	2	4	7	3	6	8	9
---	---	---	---	---	---	---	---	---

1 2 5 4 3 7 6

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1	5	2	4	7	3	6	8	9
---	---	---	---	---	---	---	---	---

1 2 5 4 3 7 6

5	1	2	4	7	8	6	3	9
---	---	---	---	---	---	---	---	---

1	5	2	4	7	3	6	8	9
---	---	---	---	---	---	---	---	---

1 2 5 4 3 7 6 8

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1 2 5 4 3 7 6 8

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9

1 2 5 4 3 7 6 8 9

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9
1	2	3	4	5				

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2 3 4 5

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2 3 4 5

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9
1	2	3	4	5	6	7		

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2 3 4 5 6 7

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2 3 4 5 6 7 8

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2 3 4 5 6 7 8

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9

1 2 3 4 5 6 7 8 9

5	1	2	4	7	8	6	3	9
1	5	2	4	7	3	6	8	9
1	2	5	4	3	7	6	8	9
1	2	3	4	5	6	7	8	9

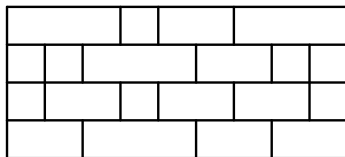
- ▶ A **sorting trace** of length 9 and order 3.
- ▶ Numbers within a block are in decreasing order.
- ▶ Adjacent numbers in different blocks form an ascent.
- ▶ Each perm is the “blockwise reversal” of the one above.
- ▶ The last permutation is the identity.

7	5	2	4	9	1	8	6	3
2	5	7	4	1	9	3	6	8
2	5	1	4	7	3	9	6	8
2	1	5	4	3	7	6	9	8
1	2	3	4	5	6	7	8	9

A sorting trace

addaadadda
 aaaddadaaa
 aadaadadaa
 adaddadada

The same sorting plan



Its **sorting plan**

0, 9, 10, 5, 5, 10, 5, 10, 9, 0

... and its encoding

7	5	2	4	9	1	8	6	3
2	5	7	4	1	9	3	6	8
2	5	1	4	7	3	9	6	8
2	1	5	4	3	7	6	9	8
1	2	3	4	5	6	7	8	9

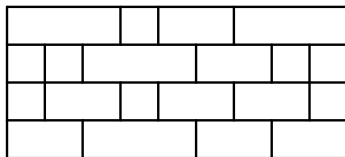
A sorting trace

```

0110010110
0001101000
0010010100
0101101010

```

The same sorting plan

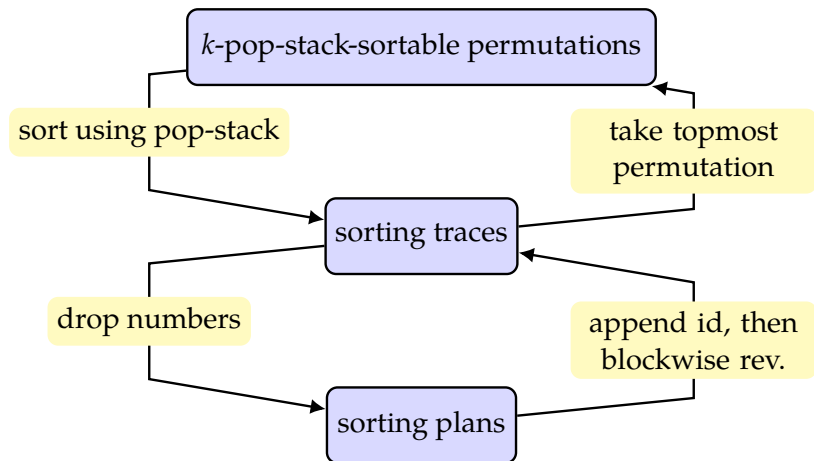


Its **sorting plan**

0, 9, 10, 5, 5, 10, 5, 10, 9, 0

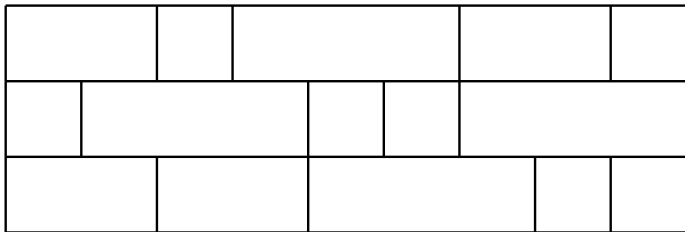
... and its encoding

Basic bijections

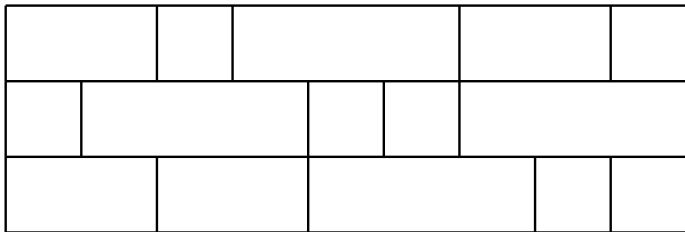


Bijection between k -pop-stack-sortable permutations of $[n]$ and sorting plans of length n and order k

An **operation array** of length 9 and order 3:

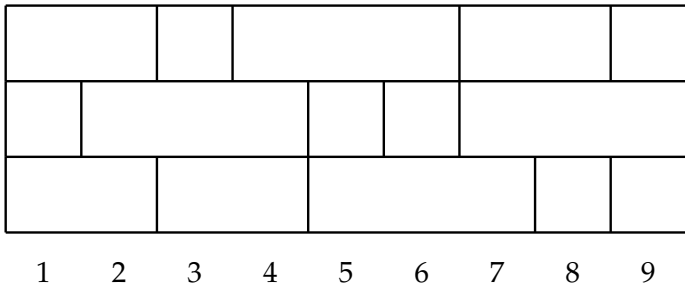


An **operation array** of length 9 and order 3:



- ▶ Assume there is a sorting trace with this operation array.

An **operation array** of length 9 and order 3:



- ▶ Assume there is a sorting trace with this operation array.
- ▶ The last permutation must be the identity.

An **operation array** of length 9 and order 3:

2	1	4	3	7	6	5	8	9
1	2	3	4	5	6	7	8	9

- ▶ Assume there is a sorting trace with this operation array.
- ▶ The last permutation must be the identity.
- ▶ Each perm is the “blockwise reversal” of the perm above.

An **operation array** of length 9 and order 3:

2	3	4	1	7	6	9	8	5
2	1	4	3	7	6	5	8	9
1	2	3	4	5	6	7	8	9

- ▶ Assume there is a sorting trace with this operation array.
- ▶ The last permutation must be the identity.
- ▶ Each perm is the “blockwise reversal” of the perm above.

An **operation array** of length 9 and order 3:

3	2	4	6	7	1	8	9	5
2	3	4	1	7	6	9	8	5
2	1	4	3	7	6	5	8	9
1	2	3	4	5	6	7	8	9

- ▶ Assume there is a sorting trace with this operation array.
- ▶ The last permutation must be the identity.
- ▶ Each perm is the “blockwise reversal” of the perm above.

An **operation array** of length 9 and order 3:

3	2	4	6	7	1	8	9	5
2	3	4	1	7	6	9	8	5
2	1	4	3	7	6	5	8	9
1	2	3	4	5	6	7	8	9

- ▶ Assume there is a sorting trace with this operation array.
- ▶ The last permutation must be the identity.
- ▶ Each perm is the “blockwise reversal” of the perm above.

An **operation array** of length 9 and order 3:

3	2	4	6	7	1	8	9	5
2	3	4	1	7	6	9	8	5
2	1	4	3	7	6	5	8	9
1	2	3	4	5	6	7	8	9

- ▶ Assume there is a sorting trace with this operation array.
- ▶ The last permutation must be the identity.
- ▶ Each perm is the “blockwise reversal” of the perm above.

An **operation array** of length 9 and order 3:

3	2	4	6	7	1	8	9	5
2	3	4	1	7	6	9	8	5
2	1	4	3	7	6	5	8	9
1	2	3	4	5	6	7	8	9

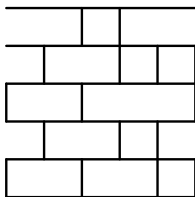
- ▶ Assume there is a sorting trace with this operation array.
- ▶ The last permutation must be the identity.
- ▶ Each perm is the “blockwise reversal” of the perm above.
- ▶ This **semitrace** is not a (proper) trace, so the operation array is *not* a sorting plan!

7	3	5	1	6	8	2	4
5	3	7	1	4	2	8	6
3	5	1	7	4	2	6	8
3	1	5	2	4	7	6	8
1	3	2	5	4	6	7	8
1	2	3	4	5	6	7	8

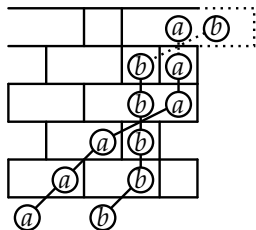
An example semitrace

7	3	5	1	6	8	②	④
5	3	7	1	④	②	8	6
3	5	1	7	④	②	6	8
3	1	5	②	④	7	6	8
1	3	②	5	④	6	7	8
1	②	3	④	5	6	7	8

The progress of 2 and 4



A forbidden segment

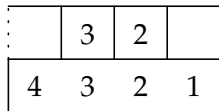
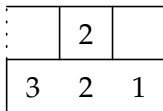
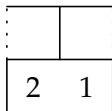


Following a and b

Lemma

In a sorting plan, each but the first row has blocks of size ≤ 3 .

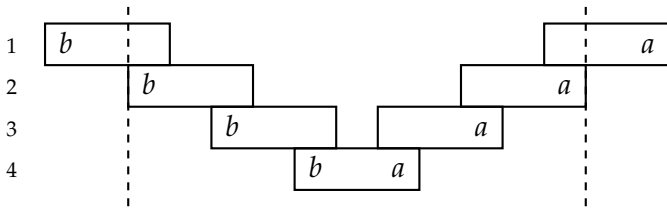
Proof



Lemma

If T is a forbidden segment in some semitrace of order k , then $|T| \leq 4k - 5$.

Proof



Corollary

For a fixed k , there are finitely many forbidden segments of order k , and they can be listed.

- ▶ For $k = 1$ there are no forbidden segments.
- ▶ For $k = 2$ the forbidden segments are

1 \emptyset

2 $\left\{ \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \end{array} , \begin{array}{|c|} \hline \square \\ \hline \square \\ \hline \square \\ \hline \square \\ \hline \end{array} \right\}$

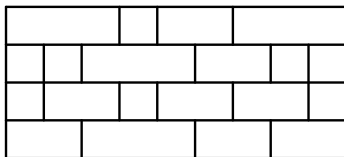
3 $\left\{ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \right\} \setminus \left\{ \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \end{array} \right\}$

7	5	2	4	9	1	8	6	3
2	5	7	4	1	9	3	6	8
2	5	1	4	7	3	9	6	8
2	1	5	4	3	7	6	9	8
1	2	3	4	5	6	7	8	9

A sorting trace

addaadadda
 aaaddadaaa
 aadaadadaa
 adaddadada

The same sorting plan



Its **sorting plan**

0, 9, 10, 5, 5, 10, 5, 10, 9, 0

... and its encoding

7	5	2	4	9	1	8	6	3
2	5	7	4	1	9	3	6	8
2	5	1	4	7	3	9	6	8
2	1	5	4	3	7	6	9	8
1	2	3	4	5	6	7	8	9

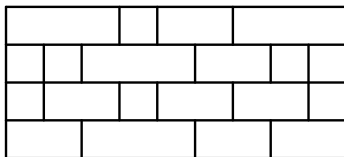
A sorting trace

```

0110010110
0001101000
0010010100
0101101010

```

The same sorting plan



Its **sorting plan**

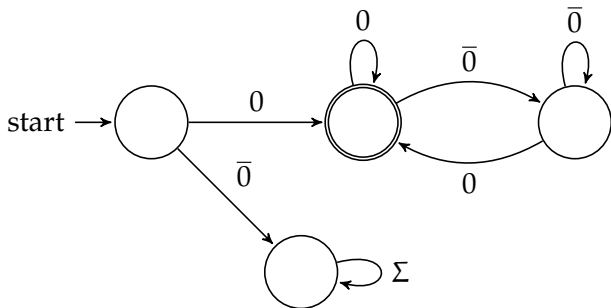
0, 9, 10, 5, 5, 10, 5, 10, 9, 0

... and its encoding

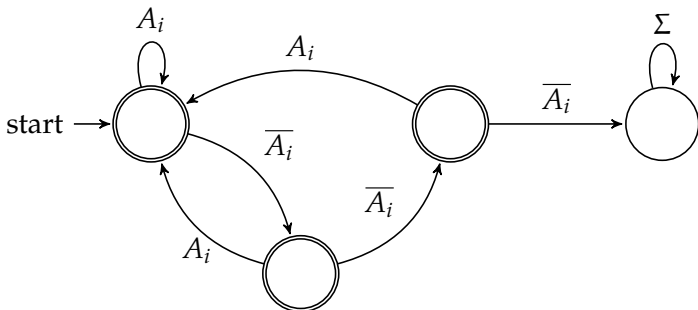
- ▶ Encode an operation array of length n and order k as a sequence of n integers, each in the range

$$\Gamma = \{0, 1, \dots, 2^k - 1\}.$$

- ▶ Thus operation arrays can be seen as strings in Γ^* .
- ▶ A DFA for the language of operation arrays, W , is

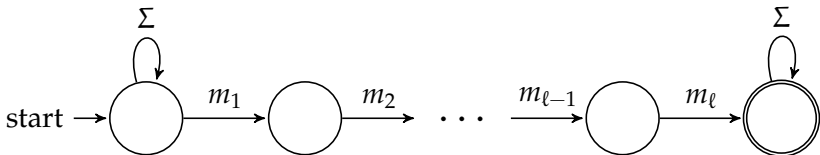


- ▶ Let A_i be the set of symbols in Σ that represent a column that has a bar in the i th row
- ▶ The following DFA, R_i , recognizes the operation arrays that have blocks of size at most 3 in row i :



- ▶ Operation arrays that have blocks of size at most 3 in all but the first row is thus recognized by the DFA $W \cap R_2 \cap \dots \cap R_k$.

- ▶ Recall that sorting plans are characterized by avoiding forbidden segments.
- ▶ Let us encode a segment M in the same manner as we encode operation arrays, resulting in the sequence m_1, \dots, m_ℓ .
- ▶ An operation array A contains the segment M if and only if the encoding of A contains $m_1 \cdots m_\ell$ as a factor.
- ▶ The following NFA, Q_M , recognizes strings over Σ that contain the encoding of M as a factor:



Let \mathcal{F} be the set of forbidden segments. Then

$$S = W \cap \bigcap_{i=2}^k R_i \cap \bigcap_{F \in \mathcal{F}} \overline{Q_F}$$

recognizes the set of sorting plans.

Theorem

The language $S = \{ w \in \Sigma^ : w \text{ is a sorting plan} \}$ is regular.*

Main result

- ▶ Let $p_k(n)$ be the number of k -pop-stack-sortable permutations of $[n]$.
- ▶ Let $P_k(x) = \sum_{n \geq 0} p_k(n)x^n$.

Theorem

The generating function $P_k(x)$ is rational.

Data

This has been implemented. Running on a big cluster we got:

k	1	2	3	4	5	6
degree	1	3	10	25	71	213
growth rate	2.0000	2.6590	3.4465	4.2706	5.1166	5.9669
vertices	4	5	12	32	99	339
edges	8	11	34	120	477	2010

All the generating functions, source code, and text files defining the DFAs can be found on GitHub:

github.com/SuprDewd/popstacks

Generating functions

1 $(x - 1) / (2x - 1)$

2 $(x^3 + x^2 + x - 1) / (2x^3 + x^2 + 2x - 1)$

3 $(2x^{10} + 4x^9 + 2x^8 + 5x^7 + 11x^6 + 8x^5 + 6x^4 + 6x^3 + 2x^2 + x - 1) / (4x^{10} + 8x^9 + 4x^8 + 10x^7 + 22x^6 + 16x^5 + 8x^4 + 6x^3 + 2x^2 + 2x - 1)$

4 $(64x^{25} + 448x^{24} + 1184x^{23} + 1784x^{22} + 2028x^{21} + 1948x^{20} + 1080x^{19} + 104x^{18} - 180x^{17} + 540x^{16} + 1156x^{15} + 696x^{14} + 252x^{13} + 238x^{12} + 188x^{11} + 502x^{10} + 806x^9 + 544x^8 + 263x^7 + 185x^6 + 99x^5 + 33x^4 + 13x^3 + 3x^2 + x - 1) / (128x^{25} + 896x^{24} + 2368x^{23} + 3568x^{22} + 3928x^{21} + 3064x^{20} + 176x^{19} - 2304x^{18} - 2664x^{17} - 1580x^{16} - 352x^{15} - 576x^{14} - 1104x^{13} - 760x^{12} - 138x^{11} + 686x^{10} + 1238x^9 + 869x^8 + 382x^7 + 210x^6 + 102x^5 + 27x^4 + 12x^3 + 3x^2 + 2x - 1)$

5,6 too large to display