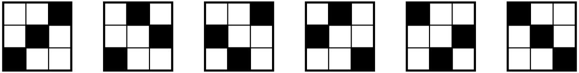
Sorting with $\mathcal{C}$-Machines

(joint work with Michael Albert, Cheyne Homberger, Nathaniel Shar, and Vince Vatter)

Jay Pantone
Dartmouth College
Hanover, NH



University of Melbourne, Victoria

July 27, 2015

PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

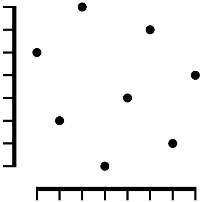
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\Longleftrightarrow$



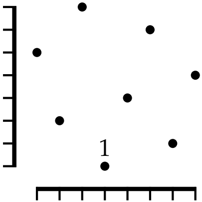
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\Longleftrightarrow$



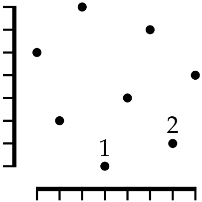
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\Longleftrightarrow$



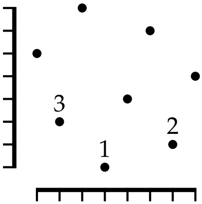
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\iff$



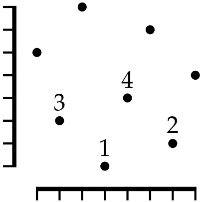
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\iff$



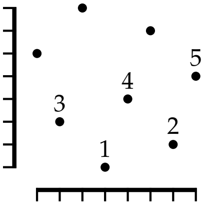
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\iff$



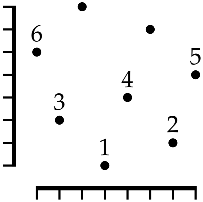
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\Longleftrightarrow$



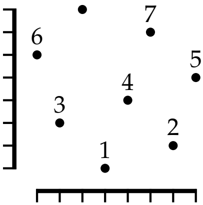
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\Longleftrightarrow$



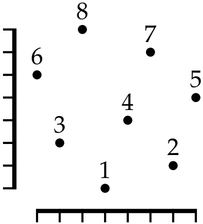
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\iff$



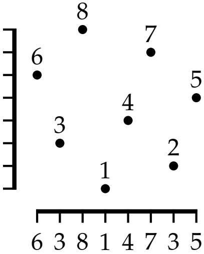
PERMUTATIONS

A *permutation* of length n is viewed as an ordering of the integers $\{1, \ldots, n\}$.

- *Example:* 63814725 is a permutation of length eight.

Geometric interpretation: A permutation can be viewed as points in the plane, no two of which lie on the same horizontal or vertical line.

- *Example:* 63814725 $\iff$



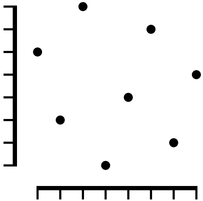
PATTERN CONTAINMENT

A permutation π contains a permutation σ (denoted $\sigma \leq \pi$) if you can delete dots in the picture of π and shrink the picture to get the picture of σ.

PATTERN CONTAINMENT

A permutation π contains a permutation σ (denoted $\sigma \leq \pi$) if you can delete dots in the picture of π and shrink the picture to get the picture of σ.

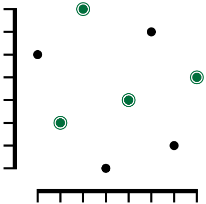
$$1423 \leq 63814725 \qquad \Longleftrightarrow$$



PATTERN CONTAINMENT

A permutation π contains a permutation σ (denoted $\sigma \leq \pi$) if you can delete dots in the picture of π and shrink the picture to get the picture of σ.

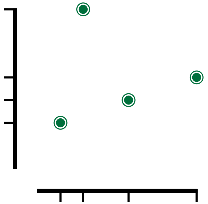
$$1423 \leq 63814725 \quad \Longleftrightarrow$$



PATTERN CONTAINMENT

A permutation π contains a permutation σ (denoted $\sigma \leq \pi$) if you can delete dots in the picture of π and shrink the picture to get the picture of σ.

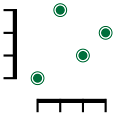
$1423 \leq 63814725 \quad \Longleftrightarrow$



PATTERN CONTAINMENT

A permutation π contains a permutation σ (denoted $\sigma \leq \pi$) if you can delete dots in the picture of π and shrink the picture to get the picture of σ.

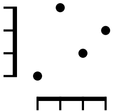
$$1423 \leq 63814725 \qquad \Longleftrightarrow$$



PATTERN CONTAINMENT

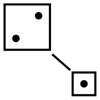
A permutation π contains a permutation σ (denoted $\sigma \leq \pi$) if you can delete dots in the picture of π and shrink the picture to get the picture of σ.

$$1423 \leq 63814725 \quad \Longleftrightarrow$$

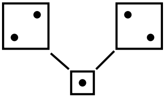


Permutation Poset

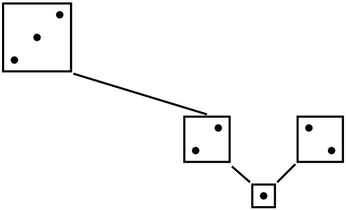
Permutation Poset



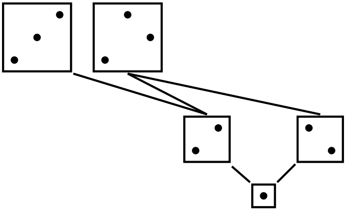
PERMUTATION POSET



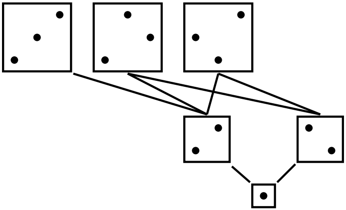
PERMUTATION POSET



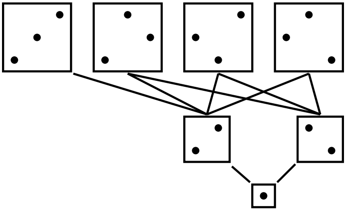
Permutation Poset



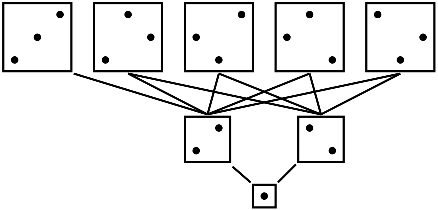
PERMUTATION POSET



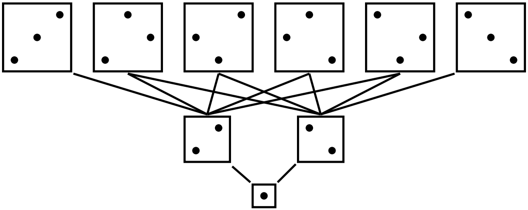
PERMUTATION POSET



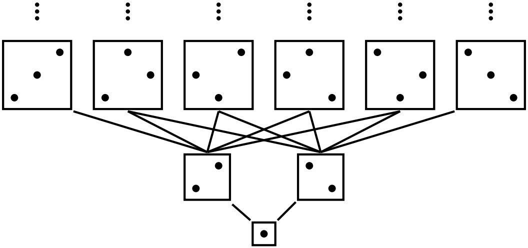
PERMUTATION POSET



PERMUTATION POSET



PERMUTATION POSET



PERMUTATION POSET

A *permutation class* is a downset in the permutation poset. In other words, if π is in the class $\mathcal{C}$ and $\sigma \leq \pi$, then we must have $\sigma \in \mathcal{C}$.

PERMUTATION POSET

A *permutation class* is a downset in the permutation poset. In other words, if π is in the class $\mathcal{C}$ and $\sigma \leq \pi$, then we must have $\sigma \in \mathcal{C}$.

A class can be specified by the set of minimal permutations not in the class, called its *basis*.

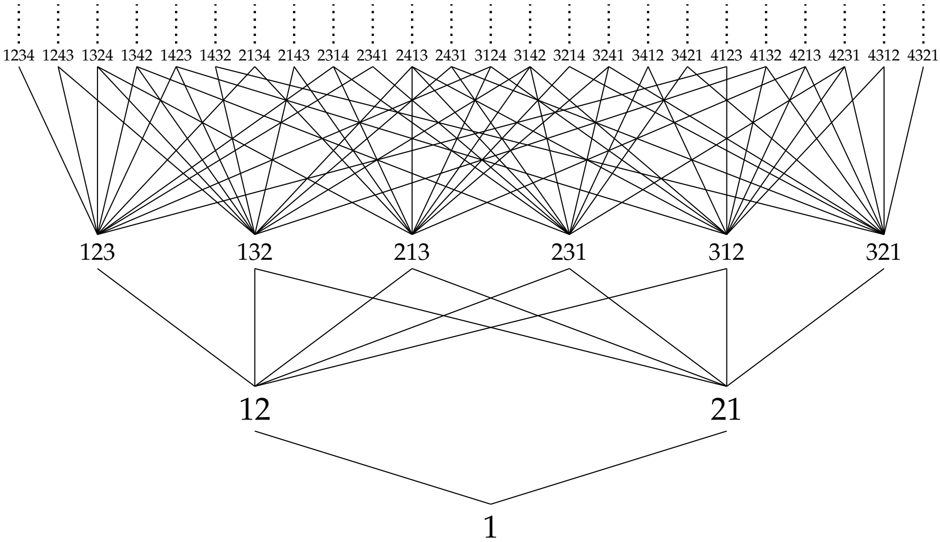
PERMUTATION POSET

A *permutation class* is a downset in the permutation poset. In other words, if π is in the class $\mathcal{C}$ and $\sigma \leq \pi$, then we must have $\sigma \in \mathcal{C}$.

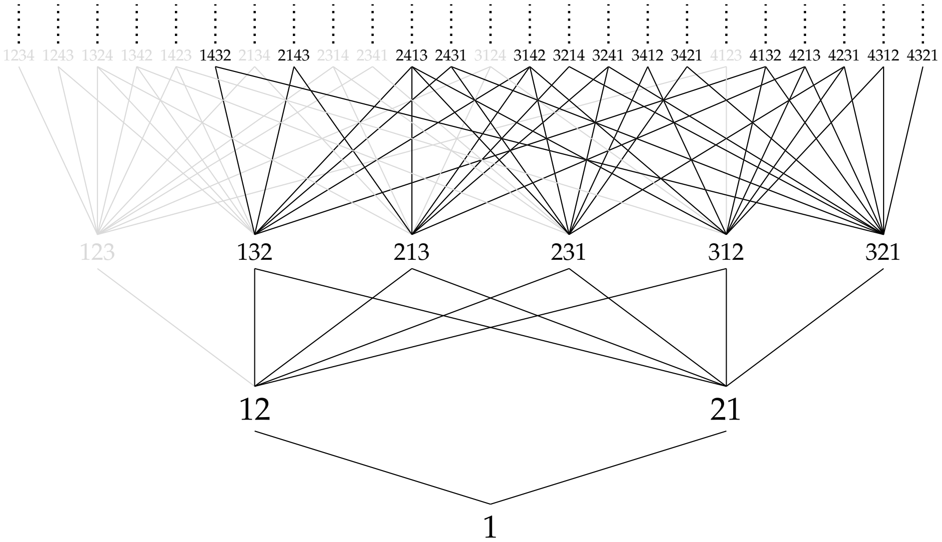
A class can be specified by the set of minimal permutations not in the class, called its *basis*.

The class with basis B is denoted $\mathrm{Av}(B)$.

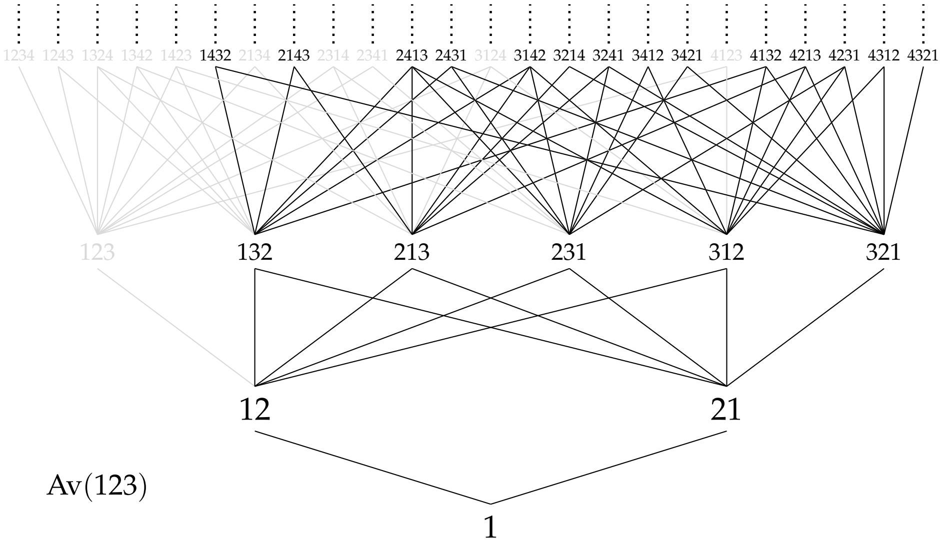
PERMUTATION POSET



Permutation Poset



PERMUTATION POSET



Generating Functions

Let $\mathcal{C}_n$ be the set of permutations in $\mathcal{C}$ of length n.

GENERATING FUNCTIONS

Let $\mathcal{C}_n$ be the set of permutations in $\mathcal{C}$ of length n.

The generating function for the class $\mathcal{C}$ is

$$f_{\mathcal{C}}(x) = \sum_{\pi \in \mathcal{C}} x^{|\pi|} = \sum_{n \geq 1} |\mathcal{C}_n| x^n.$$

GENERATING FUNCTIONS

Let $\mathcal{C}_n$ be the set of permutations in $\mathcal{C}$ of length n.

The generating function for the class $\mathcal{C}$ is

$$f_{\mathcal{C}}(x) = \sum_{\pi \in \mathcal{C}} x^{|\pi|} = \sum_{n \geq 1} |\mathcal{C}_n| x^n.$$

We sometimes choose to include constant term 1 in $f(x)$.

GENERATING FUNCTIONS

Let $\mathcal{C}_n$ be the set of permutations in $\mathcal{C}$ of length n.

The generating function for the class $\mathcal{C}$ is

$$f_{\mathcal{C}}(x) = \sum_{\pi \in \mathcal{C}} x^{|\pi|} = \sum_{n \geq 1} |\mathcal{C}_n| x^n.$$

We sometimes choose to include constant term 1 in $f(x)$.

- *Example:*

$$f_{\text{Av}(123)}(x) = \frac{1 - \sqrt{1 - 4x}}{2x}$$

GENERATING FUNCTIONS

Let $\mathcal{C}_n$ be the set of permutations in $\mathcal{C}$ of length n.

The generating function for the class $\mathcal{C}$ is

$$f_{\mathcal{C}}(x) = \sum_{\pi \in \mathcal{C}} x^{|\pi|} = \sum_{n \geq 1} |\mathcal{C}_n| x^n.$$

We sometimes choose to include constant term 1 in $f(x)$.

- *Example:*

$$\begin{aligned} f_{\text{Av}(123)}(x) &= \frac{1 - \sqrt{1 - 4x}}{2x} \\ &= 1 + x + 2x^2 + 5x^3 + 14x^4 + \cdots \end{aligned}$$

Sorting with a Stack

A stack is a container that holds entries of a permutation. There are two operations:

Sorting with a Stack

A stack is a container that holds entries of a permutation. There are two operations:

- *push* the next input symbol on top of the stack, which shifts all symbols in the stack down,

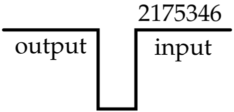
SORTING WITH A STACK

A stack is a container that holds entries of a permutation. There are two operations:

- *push* the next input symbol on top of the stack, which shifts all symbols in the stack down,
- *pop* the top symbol on the stack to the output.

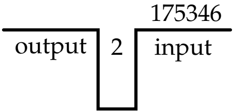
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



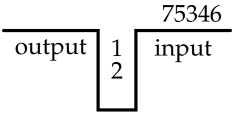
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



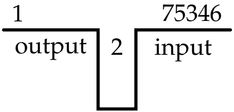
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



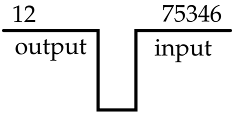
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



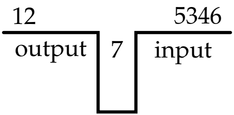
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



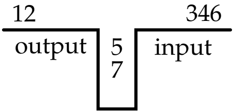
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



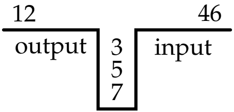
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



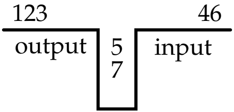
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



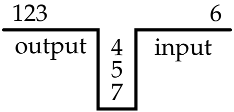
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



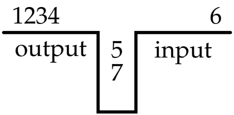
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



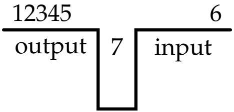
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



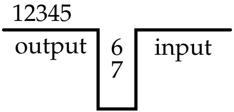
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



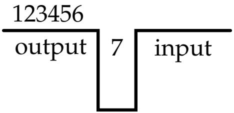
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



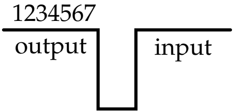
SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



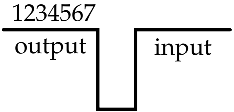
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



SORTING WITH A STACK

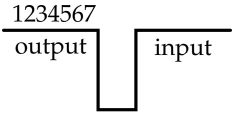
The permutation 2175346 can be sorted by a stack.



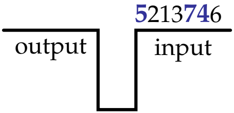
Not all permutations can be sorted by a stack.

Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

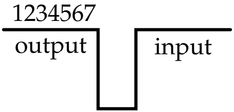


Not all permutations can be sorted by a stack.

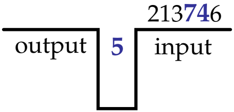


SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.

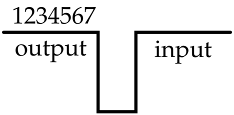


Not all permutations can be sorted by a stack.

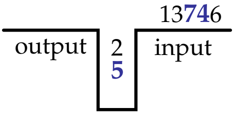


SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.

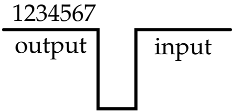


Not all permutations can be sorted by a stack.

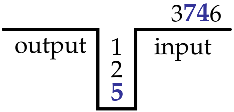


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

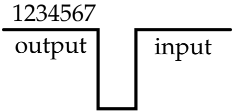


Not all permutations can be sorted by a stack.

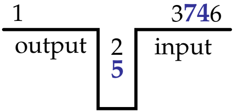


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

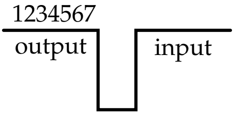


Not all permutations can be sorted by a stack.

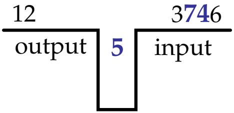


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

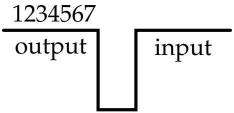


Not all permutations can be sorted by a stack.

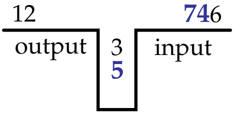


SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.

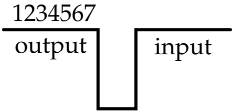


Not all permutations can be sorted by a stack.

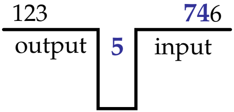


SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.

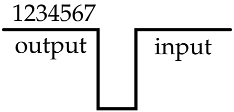


Not all permutations can be sorted by a stack.

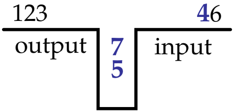


SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.

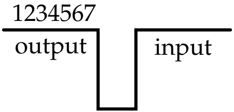


Not all permutations can be sorted by a stack.

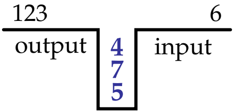


SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.

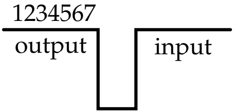


Not all permutations can be sorted by a stack.

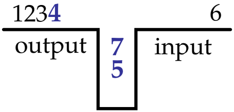


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

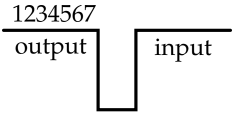


Not all permutations can be sorted by a stack.

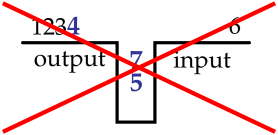


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

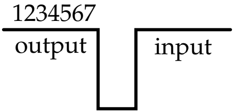


Not all permutations can be sorted by a stack.

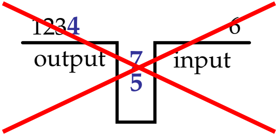


SORTING WITH A STACK

The permutation 2175346 can be sorted by a stack.



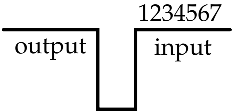
Not all permutations can be sorted by a stack.



Permutations containing 231 cannot be sorted by a stack. In fact, the permutations that are stack-sortable are exactly those in Av(231).

GENERATING WITH A STACK

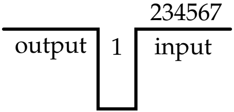
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

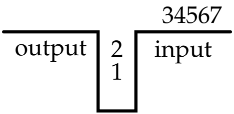
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

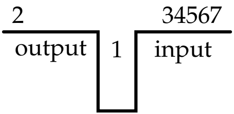
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

Generating with a Stack

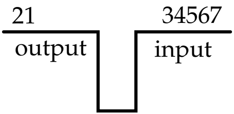
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

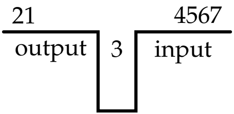
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

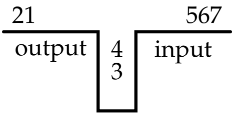
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

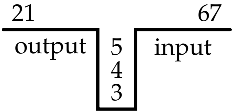
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

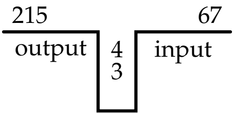
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

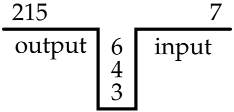
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

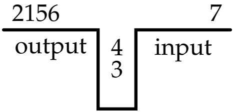
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

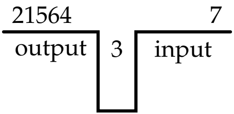
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

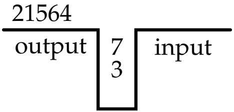
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

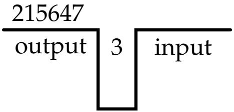
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

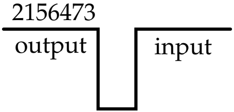
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

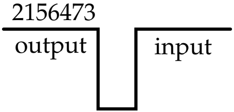
The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

GENERATING WITH A STACK

The permutation 2156473 can be generated by a stack.

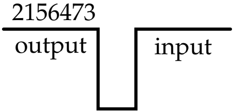


(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

The permutations that can be *generated* by a stack are exactly the inverses of those that can be *sorted* by a stack.

Generating with a Stack

The permutation 2156473 can be generated by a stack.



(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

The permutations that can be *generated* by a stack are exactly the inverses of those that can be *sorted* by a stack.

A stack can generate the permutations in $\mathrm{Av}(231^{-1}) = \mathrm{Av}(312)$.

GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

- push a new maximum entry into the container such that the container holds a decreasing permutation

GENERALIZING STACKS

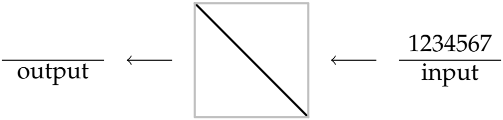
The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.

GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

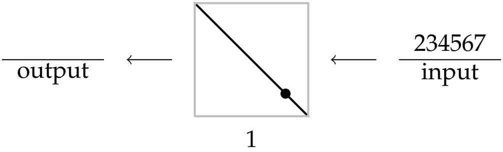
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

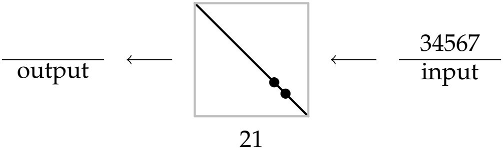
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

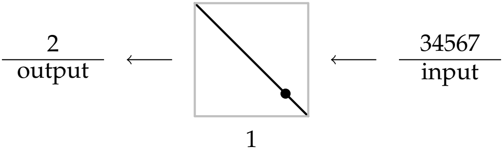
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

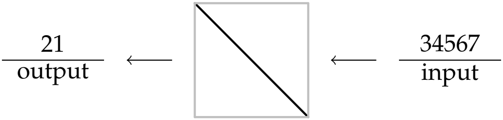
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

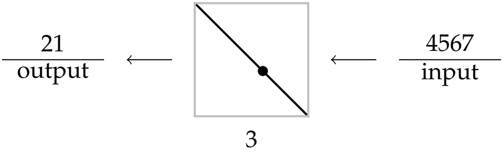
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

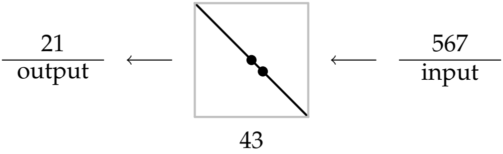
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

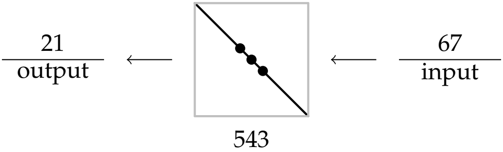
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

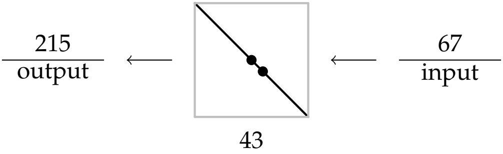
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

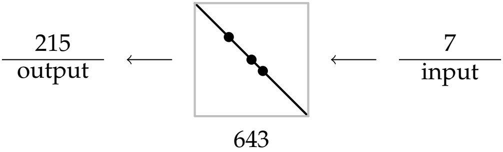
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

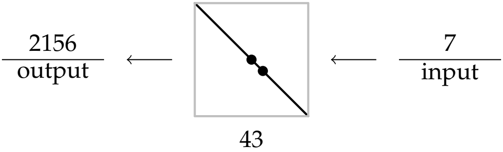
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

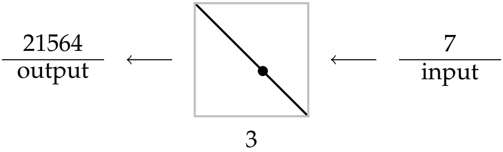
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

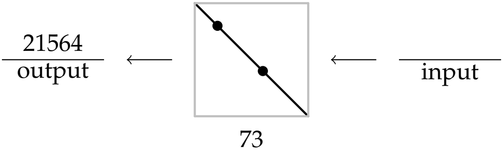
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

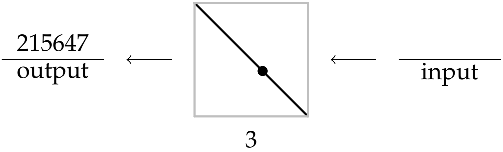
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

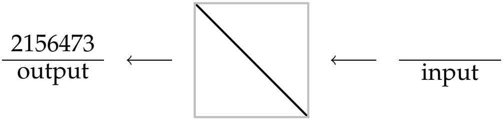
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

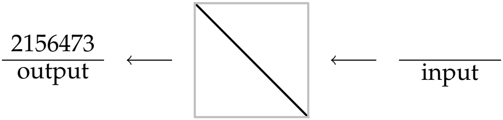
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



GENERALIZING STACKS

The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:

- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Since the container holds permutations that avoid the pattern 12, we call this the Av(12)-machine.

Generalizing Stacks

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

GENERALIZING STACKS

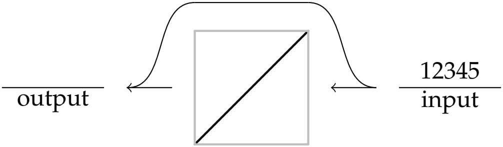
We allow a third operation: an entry can bypass the container and move straight from the input to the output.

In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.

GENERALIZING STACKS

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

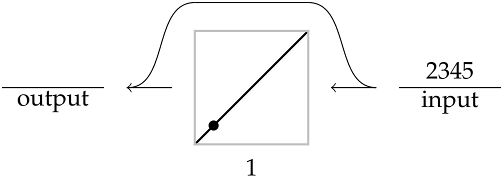
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

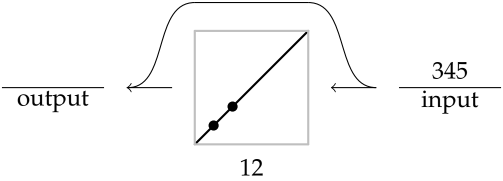
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



GENERALIZING STACKS

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

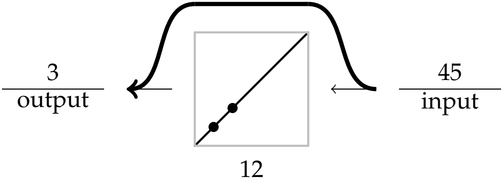
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



GENERALIZING STACKS

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

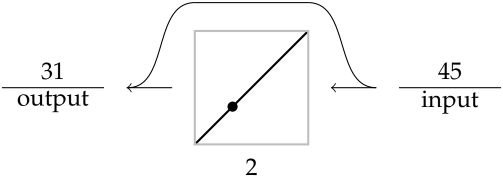
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



GENERALIZING STACKS

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

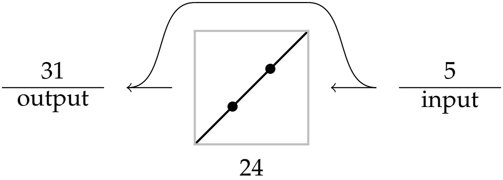
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



GENERALIZING STACKS

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

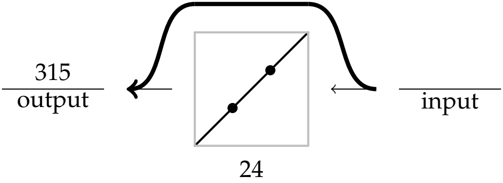
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

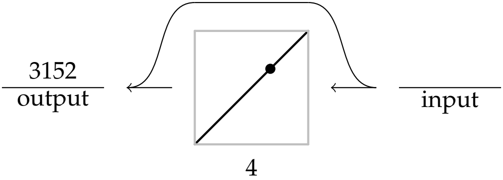
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



GENERALIZING STACKS

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

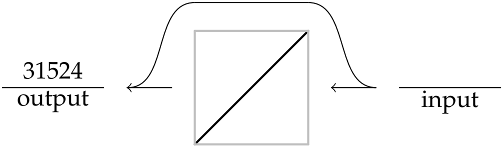
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

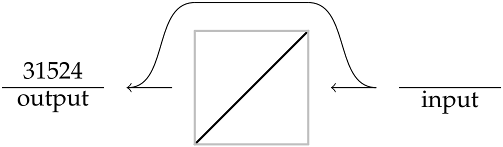
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

We allow a third operation: an entry can bypass the container and move straight from the input to the output.

In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



The Av(21)-machine generates the class Av(321).

The Basis Theorem

- The Av(12)-machine generates the class Av(312).

The Basis Theorem

- The Av(12)-machine generates the class Av(312).
- The Av(21)-machine generates the class Av(321).

THE BASIS THEOREM

- The Av(12)-machine generates the class Av(312).
- The Av(21)-machine generates the class Av(321).

Theorem. The Av(B)-machine generates the class

$$\text{Av}(\{{}^{+}\beta : \beta \in B\}),$$

where ${}^{+}\beta$ is formed by adding a new maximum in front of β.

THE BASIS THEOREM

- The Av(12)-machine generates the class Av(312).
- The Av(21)-machine generates the class Av(321).

Theorem. The Av(B)-machine generates the class

$$\mathrm{Av}(\{{}^{+}\beta : \beta \in B\}),$$

where ${}^{+}\beta$ is formed by adding a new maximum in front of β.

Example: The Av(123, 4132)-machine generates the class Av(4123, 54132).

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

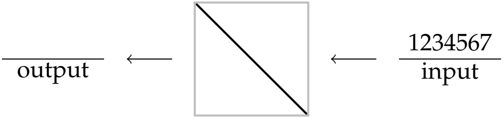
The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

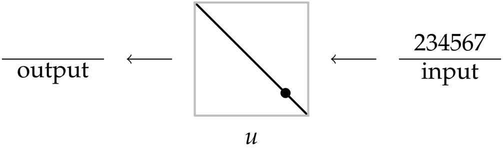
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

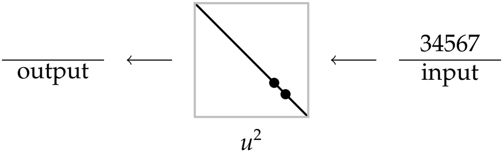
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



Enumerating Av(312) via the Av(12)-Machine

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

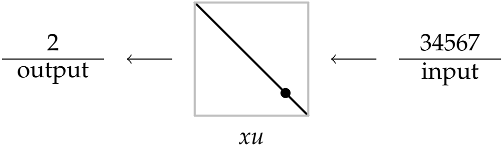
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

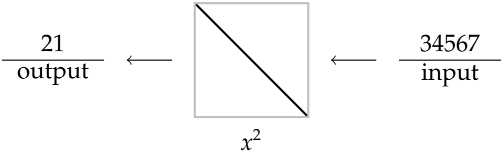
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



Enumerating Av(312) via the Av(12)-Machine

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

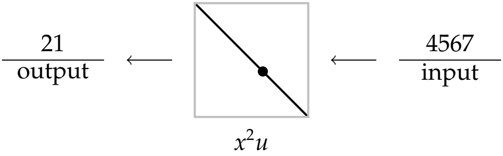
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

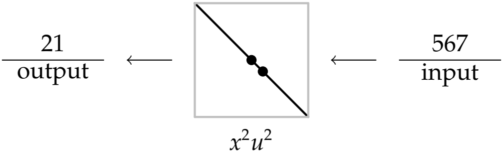
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

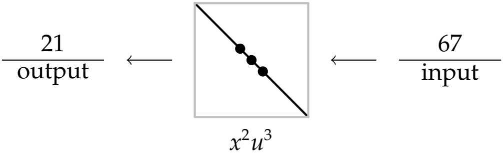
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

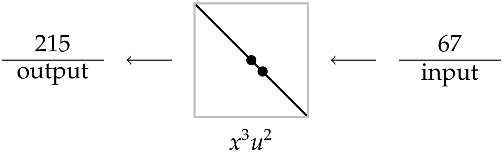
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



Enumerating Av(312) via the Av(12)-Machine

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

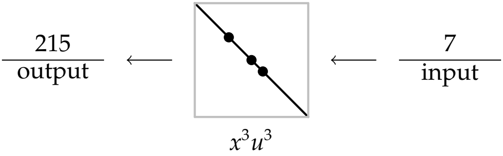
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

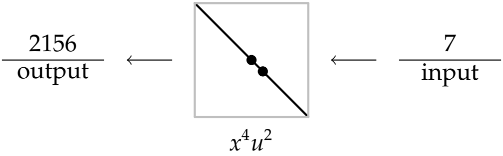
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

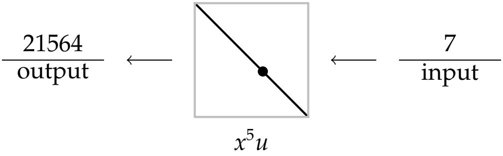
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

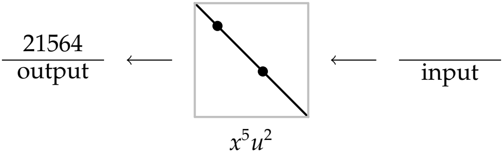
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

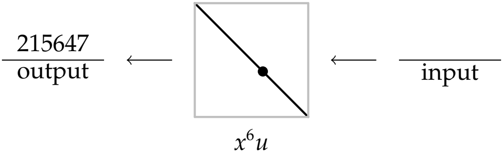
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



ENUMERATING Av(312) VIA THE Av(12)-MACHINE

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

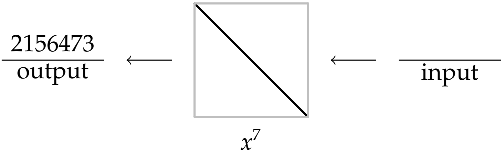
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



Enumerating Av(312) via the Av(12)-Machine

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.

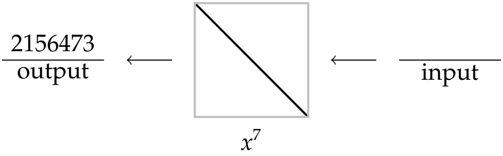


States with no entries in the machine are those with no u term.

Enumerating Av(312) via the Av(12)-Machine

The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



States with no entries in the machine are those with no u term.

Hence the generating function for these states is $f(x,0)$.

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

Enumerating Av(312) via the Av(12)-Machine

Every machine state arises either:

- as the start state

Enumerating Av(312) via the Av(12)-Machine

Every machine state arises either:

- as the start state
- from pushing an entry into a previous state

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

- as the start state
- from pushing an entry into a previous state
- from popping an entry from a *nonempty* state

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

- as the start state
- from pushing an entry into a previous state
- from popping an entry from a *nonempty* state

This translates to a functional equation:

$$f(x,u) = 1 + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

- as the start state
- from pushing an entry into a previous state
- from popping an entry from a *nonempty* state

This translates to a functional equation:

$$\boldsymbol{f(x,u)} = 1 + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

- **as the start state**
- from pushing an entry into a previous state
- from popping an entry from a *nonempty* state

This translates to a functional equation:

$$f(x,u) = \mathbf{1} + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

- as the start state
- **from pushing an entry into a previous state**
- from popping an entry from a *nonempty* state

This translates to a functional equation:

$$f(x,u) = 1 + \boldsymbol{uf(x,u)} + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

- as the start state
- from pushing an entry into a previous state
- **from popping an entry from a *nonempty* state**

This translates to a functional equation:

$$f(x,u) = 1 + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

Every machine state arises either:

- as the start state
- from pushing an entry into a previous state
- from popping an entry from a *nonempty* state

This translates to a functional equation:

$$f(x,u) = 1 + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

We want to solve for $f(x,0)$.

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

We can solve for $f(x, 0)$ using the kernel method.

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

We can solve for $f(x, 0)$ using the kernel method.

$$f(x, u) = 1 + uf(x, u) + \frac{x}{u}\left(f(x, u) - f(x, 0)\right)$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

We can solve for $f(x, 0)$ using the kernel method.

$$f(x,u) = 1 + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

$$\underbrace{\left(1 + u - \frac{x}{u}\right)}_{\text{kernel}} f(x,u) = 1 - \frac{x}{u} f(x,0).$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

We can solve for $f(x,0)$ using the kernel method.

$$f(x,u) = 1 + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

$$\underbrace{\left(1 + u - \frac{x}{u}\right)}_{\text{kernel}} f(x,u) = 1 - \frac{x}{u} f(x,0).$$

Substitute

$$u = \frac{1 \pm \sqrt{1-4x}}{2},$$

to get

$$0 = \left[1 - \frac{x}{u} f(x,0)\right]_{u=\frac{1\pm\sqrt{1-4x}}{2}}.$$

ENUMERATING Av(312) VIA THE Av(12)-MACHINE

We can solve for $f(x,0)$ using the kernel method.

$$f(x,u) = 1 + uf(x,u) + \frac{x}{u}\left(f(x,u) - f(x,0)\right)$$

$$\underbrace{\left(1 + u - \frac{x}{u}\right)}_{\text{kernel}} f(x,u) = 1 - \frac{x}{u} f(x,0).$$

Substitute

$$u = \frac{1 \pm \sqrt{1-4x}}{2},$$

to get

$$0 = \left[1 - \frac{x}{u} f(x,0)\right]_{u = \frac{1 \pm \sqrt{1-4x}}{2}}.$$

Solving,

$$f(x,0) = \frac{1 \pm \sqrt{1-4x}}{2x}.$$

ENUMERATING Av(321) VIA THE Av(21)-MACHINE

When considering the Av(21)-machine, we have to allow by-passes.

ENUMERATING Av(321) VIA THE Av(21)-MACHINE

When considering the Av(21)-machine, we have to allow bypasses.

If we're not careful, we'll overcount: for example, the permutation 1 could be generated by

Enumerating Av(321) via the Av(21)-Machine

When considering the Av(21)-machine, we have to allow by-passes.

If we're not careful, we'll overcount: for example, the permutation 1 could be generated by

- a push then a pop, or

ENUMERATING Av(321) VIA THE Av(21)-MACHINE

When considering the Av(21)-machine, we have to allow bypasses.

If we're not careful, we'll overcount: for example, the permutation 1 could be generated by

- a push then a pop, or
- a bypass.

Enumerating Av(321) via the Av(21)-Machine

When considering the Av(21)-machine, we have to allow bypasses.

If we're not careful, we'll overcount: for example, the permutation 1 could be generated by

- a push then a pop, or
- a bypass.

To ensure uniqueness of generation sequences, we require that

ENUMERATING Av(321) VIA THE Av(21)-MACHINE

When considering the Av(21)-machine, we have to allow bypasses.

If we're not careful, we'll overcount: for example, the permutation 1 could be generated by

- a push then a pop, or
- a bypass.

To ensure uniqueness of generation sequences, we require that

- a bypass is always preferred when possible,

ENUMERATING Av(321) VIA THE Av(21)-MACHINE

When considering the Av(21)-machine, we have to allow bypasses.

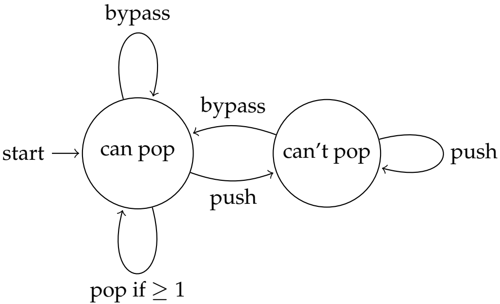
If we're not careful, we'll overcount: for example, the permutation 1 could be generated by

- a push then a pop, or
- a bypass.

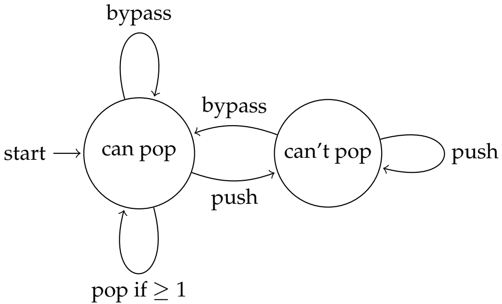
To ensure uniqueness of generation sequences, we require that

- a bypass is always preferred when possible,
- an entry should be popped as soon as possible.

ENUMERATING Av(321) VIA THE Av(21)-MACHINE



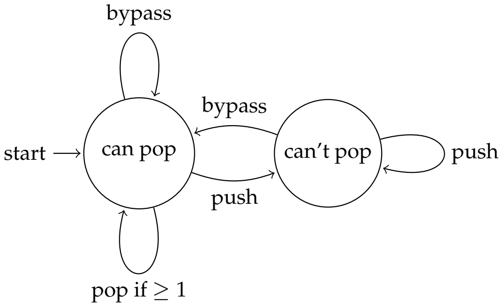
ENUMERATING Av(321) VIA THE Av(21)-MACHINE



$$\text{[can pop]}: \quad f(x,u) = 1 + x(f(x,u) + g(x,u)) + \frac{x}{u}(f(x,u) - f(x,0)),$$

$$\text{[can't pop]}: \quad g(x,u) = u(f(x,u) + g(x,u)).$$

ENUMERATING Av(321) VIA THE Av(21)-MACHINE



$$\text{[can pop]}: \quad f(x,u) = 1 + x(f(x,u) + g(x,u)) + \frac{x}{u}(f(x,u) - f(x,0)),$$

$$\text{[can't pop]}: \quad g(x,u) = u(f(x,u) + g(x,u)).$$

Solve $g(x,u)$ in terms of $f(x,u)$ → substitute into $f(x,u)$ → kernel method → same answer!

ENUMERATING THE SCHRÖDER CLASSES

A *2×4 class* is a permutation class with two basis elements of length 4.

ENUMERATING THE SCHRÖDER CLASSES

A *2×4 class* is a permutation class with two basis elements of length 4.

- *Example:* $\mathrm{Av}(1324, 2413)$ is a $2{\times}4$ class.

ENUMERATING THE SCHRÖDER CLASSES

A *2×4 class* is a permutation class with two basis elements of length 4.

- *Example:* Av(1324, 2413) is a 2×4 class.

Up to symmetries of the permutation containment order there are 56 essentially different 2×4 classes.

ENUMERATING THE SCHRÖDER CLASSES

A *2×4 class* is a permutation class with two basis elements of length 4.

- *Example:* $\text{Av}(1324, 2413)$ is a $2{\times}4$ class.

Up to symmetries of the permutation containment order there are 56 essentially different $2{\times}4$ classes.

Some of these 56 $2{\times}4$ classes have the same enumeration despite being non-isomorphic. There are precisely 38 different enumerations for the $2{\times}4$ classes.

ENUMERATING THE SCHRÖDER CLASSES

A *2×4 class* is a permutation class with two basis elements of length 4.

- *Example:* $\mathrm{Av}(1324, 2413)$ is a 2×4 class.

Up to symmetries of the permutation containment order there are 56 essentially different 2×4 classes.

Some of these 56 2×4 classes have the same enumeration despite being non-isomorphic. There are precisely 38 different enumerations for the 2×4 classes.

Of these 38 classes, the exact enumerations are known for 29 of them. (More on this later!)

ENUMERATING THE SCHRÖDER CLASSES

There are ten 2×4 classes that are enumerated by the Schröder numbers: $1, 2, 6, 22, 90, \ldots$.

ENUMERATING THE SCHRÖDER CLASSES

There are ten 2×4 classes that are enumerated by the Schröder numbers: $1, 2, 6, 22, 90, \ldots$.

Six of them are of the form $\mathrm{Av}({}^{+}\beta_1, {}^{+}\beta_2)$ and so they are generated by $\mathcal{C}$-machines.

Enumerating the Schröder Classes

There are ten 2×4 classes that are enumerated by the Schröder numbers: $1, 2, 6, 22, 90, \ldots$.

Six of them are of the form $\mathrm{Av}({}^{+}\beta_1, {}^{+}\beta_2)$ and so they are generated by $\mathcal{C}$-machines.

- *Example:* $\mathrm{Av}(4231, 4132)$ is enumerated by the Schröder numbers, and is generated by the $\mathrm{Av}(231, 132)$ machine.

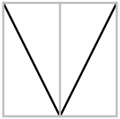
ENUMERATING THE SCHRÖDER CLASSES

There are ten 2×4 classes that are enumerated by the Schröder numbers: $1, 2, 6, 22, 90, \ldots$.

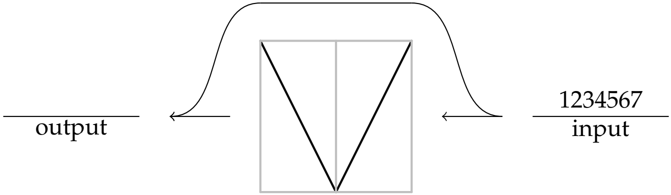
Six of them are of the form $\mathrm{Av}({}^{+}\beta_1, {}^{+}\beta_2)$ and so they are generated by $\mathcal{C}$-machines.

- *Example:* $\mathrm{Av}(4231, 4132)$ is enumerated by the Schröder numbers, and is generated by the $\mathrm{Av}(231, 132)$ machine.

Every permutation in $\mathrm{Av}(231, 132)$ is a descending segment followed by an ascending segment, and so we say that the class has the shape

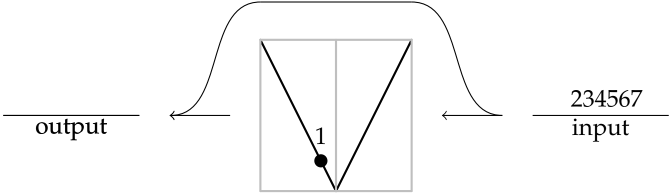


ENUMERATING THE SCHRÖDER CLASSES



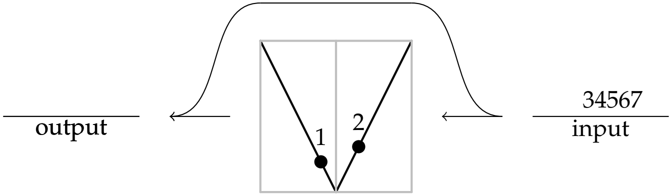
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



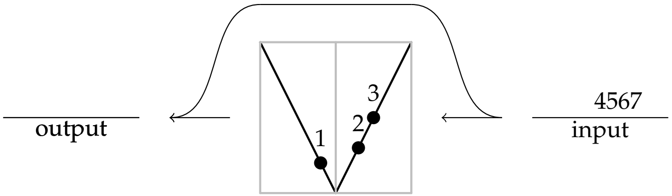
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



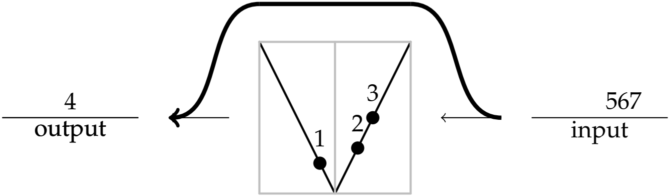
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



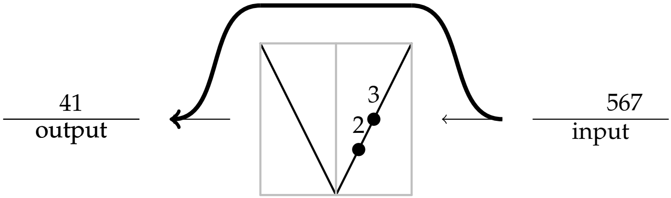
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



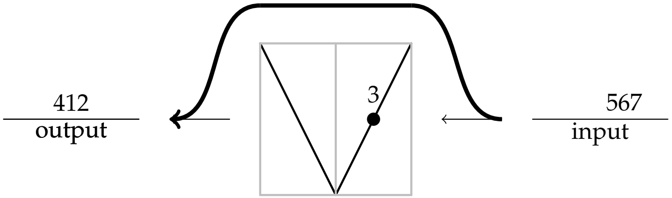
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



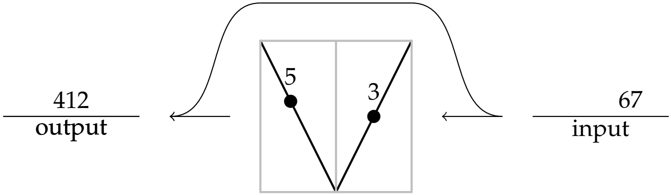
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



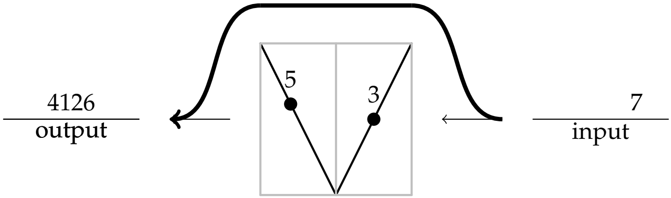
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



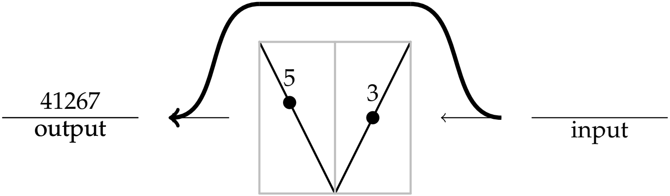
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



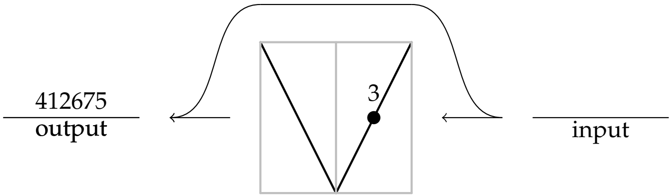
Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



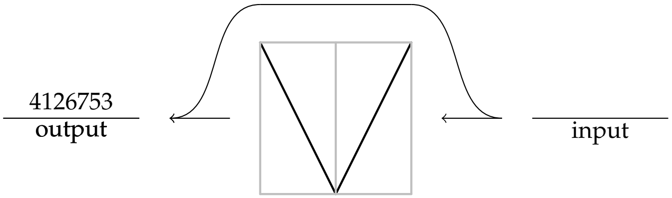
Whenever the machine is nonempty, there are always two places to push a new entry.

Enumerating the Schröder Classes



Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES



Whenever the machine is nonempty, there are always two places to push a new entry.

ENUMERATING THE SCHRÖDER CLASSES

A small adjustment to the earlier functional equations gives:

$$f(x,u) = 1 + x(f(x,u) + g(x,u)) + \frac{x}{u}(f(x,u) - f(x,0)),$$
$$g(x,u) = 2u((f(x,u) - f(x,0)) + g(x,u)) + uf(x,0).$$

ENUMERATING THE SCHRÖDER CLASSES

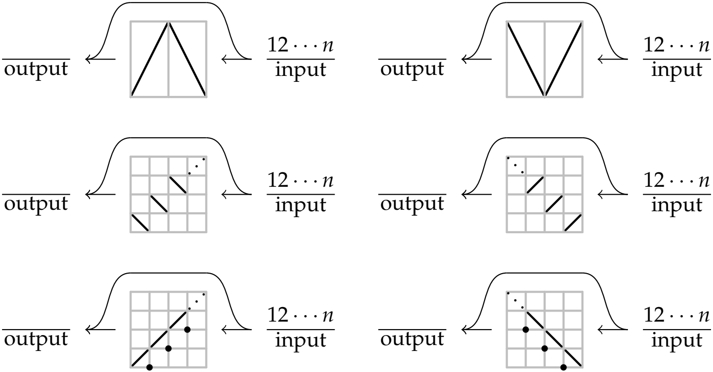
A small adjustment to the earlier functional equations gives:

$$f(x,u) = 1 + x(f(x,u) + g(x,u)) + \frac{x}{u}(f(x,u) - f(x,0)),$$
$$g(x,u) = 2u((f(x,u) - f(x,0)) + g(x,u)) + uf(x,0).$$

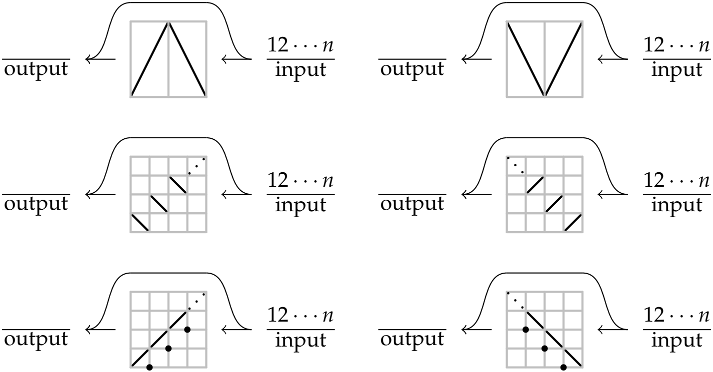
An application of the kernel method gives the generating function for the Schröder numbers.

$$f(x,0) = \frac{3 - x - \sqrt{1 - 6x + x^2}}{2}$$

ENUMERATING THE SCHRÖDER CLASSES

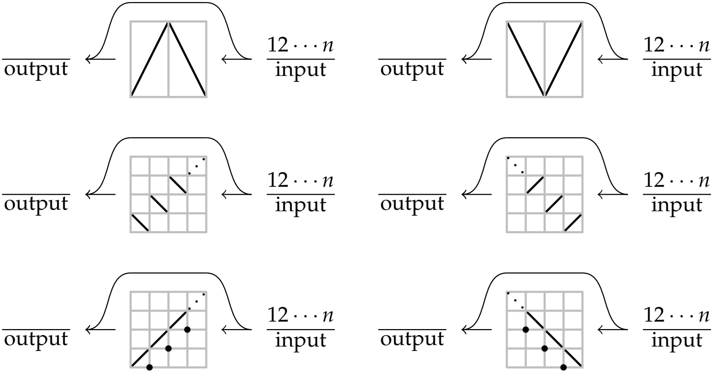


ENUMERATING THE SCHRÖDER CLASSES



All are enumerated by the Schröder numbers.

ENUMERATING THE SCHRÖDER CLASSES



All are enumerated by the Schröder numbers.

Even better, the push-pop-bypasses sequences automatically give a bijection between any pair of these classes.

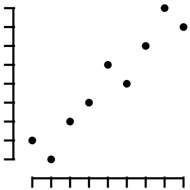
THE FIBONACCI MACHINES

There are exactly two permutation classes whose enumeration is given by the Fibonacci numbers.

THE FIBONACCI MACHINES

There are exactly two permutation classes whose enumeration is given by the Fibonacci numbers.

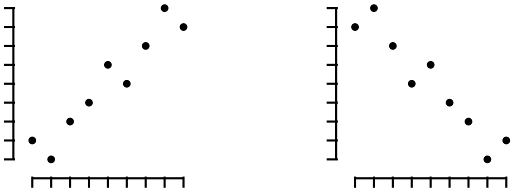
- $\mathcal{F}_{\oplus} = \mathrm{Av}(231, 312, 321)$: 1 and 21 patterns stacked $\nearrow$



THE FIBONACCI MACHINES

There are exactly two permutation classes whose enumeration is given by the Fibonacci numbers.

- $\mathcal{F}_\oplus = \text{Av}(231, 312, 321)$: 1 and 21 patterns stacked $\nearrow$
- $\mathcal{F}_\ominus = \text{Av}(123, 132, 213)$: 1 and 12 patterns stacked $\nwarrow$



THE FIBONACCI MACHINES

By the basis theorem, the $\mathcal{F}_\oplus$-machine generates the class

$$\text{Av}(4231, 4312, 4321)$$

and the $\mathcal{F}_\ominus$-machine generates the class

$$\text{Av}(4123, 4132, 4213).$$

THE FIBONACCI MACHINES

By the basis theorem, the $\mathcal{F}_{\oplus}$-machine generates the class

$$\mathrm{Av}(4231, 4312, 4321)$$

and the $\mathcal{F}_{\ominus}$-machine generates the class

$$\mathrm{Av}(4123, 4132, 4213).$$

These two enumerations are new, and they demonstrate the power of the $\mathcal{C}$-machine model.

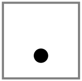
ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

To enumerate the class generated by the $\mathcal{F}_{\oplus}$-machine, we keep track of whether the upper-rightmost layer consist of a single entry or two entries.

ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

To enumerate the class generated by the $\mathcal{F}_{\oplus}$-machine, we keep track of whether the upper-rightmost layer consist of a single entry or two entries.

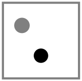
Single entry $\longrightarrow$ two possible push locations.



ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

To enumerate the class generated by the $\mathcal{F}_{\oplus}$-machine, we keep track of whether the upper-rightmost layer consist of a single entry or two entries.

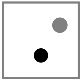
Single entry $\longrightarrow$ two possible push locations.



ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

To enumerate the class generated by the $\mathcal{F}_{\oplus}$-machine, we keep track of whether the upper-rightmost layer consist of a single entry or two entries.

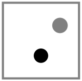
Single entry $\longrightarrow$ two possible push locations.



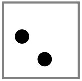
ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

To enumerate the class generated by the $\mathcal{F}_{\oplus}$-machine, we keep track of whether the upper-rightmost layer consist of a single entry or two entries.

Single entry $\longrightarrow$ two possible push locations.



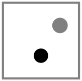
Two entries $\longrightarrow$ one possible push location.



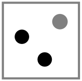
ENUMERATING THE $\mathcal{F}_\oplus$-MACHINE

To enumerate the class generated by the $\mathcal{F}_\oplus$-machine, we keep track of whether the upper-rightmost layer consist of a single entry or two entries.

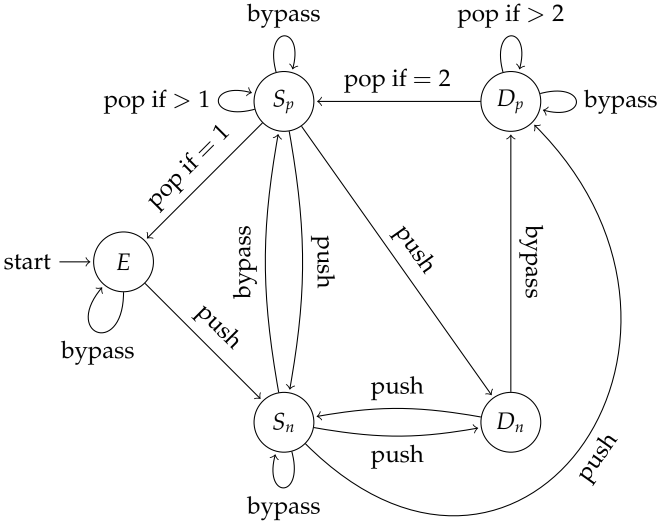
Single entry $\longrightarrow$ two possible push locations.



Two entries $\longrightarrow$ one possible push location.



ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE



ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

The automaton turns into functional equations:

$$
\begin{aligned}
E &= 1 + xE + x\left(S_p\big|_{u=0}\right)\\
S_p &= x(S_n + S_p) + \frac{x}{u}\left(S_p - S_p\big|_{u=0}\right) + x\left(D_p\big|_{u=0}\right)\\
S_n &= E + u(S_n + S_p) + u^2(D_n + D_p)\\
D_p &= x(D_n + D_p) + \frac{x}{u}\left(D_p - D_p\big|_{u=0}\right)\\
D_n &= S_n + S_p
\end{aligned}
$$

Enumerating the $\mathcal{F}_{\oplus}$-Machine

The automaton turns into functional equations:

$$
\begin{aligned}
E &= 1 + xE + x\left(S_p\big|_{u=0}\right)\\
S_p &= x(S_n + S_p) + \frac{x}{u}\left(S_p - S_p\big|_{u=0}\right) + x\left(D_p\big|_{u=0}\right)\\
S_n &= E + u(S_n + S_p) + u^2(D_n + D_p)\\
D_p &= x(D_n + D_p) + \frac{x}{u}\left(D_p - D_p\big|_{u=0}\right)\\
D_n &= S_n + S_p
\end{aligned}
$$

- Get lots of terms of $S_p\big|_{u=0}$ and $D_p\big|_{u=0}$.

ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

The automaton turns into functional equations:

$$\begin{aligned}
E &= 1 + xE + x\left(S_p\big|_{u=0}\right)\\
S_p &= x(S_n + S_p) + \frac{x}{u}\left(S_p - S_p\big|_{u=0}\right) + x\left(D_p\big|_{u=0}\right)\\
S_n &= E + u(S_n + S_p) + u^2(D_n + D_p)\\
D_p &= x(D_n + D_p) + \frac{x}{u}\left(D_p - D_p\big|_{u=0}\right)\\
D_n &= S_n + S_p
\end{aligned}$$

- Get lots of terms of $S_p\big|_{u=0}$ and $D_p\big|_{u=0}$.
- Guess their generating functions.

Enumerating the $\mathcal{F}_{\oplus}$-Machine

The automaton turns into functional equations:

$$\begin{aligned}
E &= 1 + xE + x\left(S_p\big|_{u=0}\right)\\
S_p &= x(S_n + S_p) + \frac{x}{u}\left(S_p - S_p\big|_{u=0}\right) + x\left(D_p\big|_{u=0}\right)\\
S_n &= E + u(S_n + S_p) + u^2(D_n + D_p)\\
D_p &= x(D_n + D_p) + \frac{x}{u}\left(D_p - D_p\big|_{u=0}\right)\\
D_n &= S_n + S_p
\end{aligned}$$

- Get lots of terms of $S_p\big|_{u=0}$ and $D_p\big|_{u=0}$.
- Guess their generating functions.
- Plug in these guesses, then solve the system.

ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

The automaton turns into functional equations:

$$E = 1 + xE + x\left(S_p\big|_{u=0}\right)$$
$$S_p = x(S_n + S_p) + \frac{x}{u}\left(S_p - S_p\big|_{u=0}\right) + x\left(D_p\big|_{u=0}\right)$$
$$S_n = E + u(S_n + S_p) + u^2(D_n + D_p)$$
$$D_p = x(D_n + D_p) + \frac{x}{u}\left(D_p - D_p\big|_{u=0}\right)$$
$$D_n = S_n + S_p$$

- Get lots of terms of $S_p\big|_{u=0}$ and $D_p\big|_{u=0}$.
- Guess their generating functions.
- Plug in these guesses, then solve the system.
- Verify the guesses.

ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

The automaton turns into functional equations:

$$
\begin{aligned}
E &= 1 + xE + x\left(S_p\big|_{u=0}\right)\\
S_p &= x(S_n + S_p) + \frac{x}{u}\left(S_p - S_p\big|_{u=0}\right) + x\left(D_p\big|_{u=0}\right)\\
S_n &= E + u(S_n + S_p) + u^2(D_n + D_p)\\
D_p &= x(D_n + D_p) + \frac{x}{u}\left(D_p - D_p\big|_{u=0}\right)\\
D_n &= S_n + S_p
\end{aligned}
$$

- Get lots of terms of $S_p\big|_{u=0}$ and $D_p\big|_{u=0}$.
- Guess their generating functions.
- Plug in these guesses, then solve the system.
- Verify the guesses.
- The solution is now rigorous.

ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

We find that the class generated by the $\mathcal{F}_{\oplus}$-machine has an algebraic generating function $E(x)$ that satisfies the minimal polynomial:

$$\begin{aligned}(2x^2+8x-1)E(x)^4+(x^3+4x^2-46x+5)E(x)^3\\ +(3x^3-21x^2+94x-9)E(x)^2\\ +(x^3+12x^2-82x+7)E(x)\\ +3x^2+26x-2=0\end{aligned}$$

ENUMERATING THE $\mathcal{F}_{\oplus}$-MACHINE

We find that the class generated by the $\mathcal{F}_{\oplus}$-machine has an algebraic generating function $E(x)$ that satisfies the minimal polynomial:

$$
\begin{aligned}
(2x^2+8x-1)E(x)^4+(x^3+4x^2-46x+5)E(x)^3 \\
+(3x^3-21x^2+94x-9)E(x)^2 \\
+(x^3+12x^2-82x+7)E(x) \\
+3x^2+26x-2=0
\end{aligned}
$$

and exponential growth rate

$$\approx 5.162$$

Enumerating the $\mathcal{F}_{\ominus}$-Machine

The same approach fails for the $\mathcal{F}_{\ominus}$-machine because now we're pushing and popping on the same layer.

ENUMERATING THE $\mathcal{F}_{\ominus}$-MACHINE

The same approach fails for the $\mathcal{F}_{\ominus}$-machine because now we're pushing and popping on the same layer.

Instead we construct a context-free grammar to represent the allowed push-pop-bypass sequences.

ENUMERATING THE $\mathcal{F}_{\ominus}$-MACHINE

$$
\begin{array}{lcccccccc}
S & \longrightarrow & \epsilon & | & xS & | & (+w)W_n(-w)S & & \\
W_p & \longrightarrow & \epsilon & | & xW_p & | & (+w)W_n(-w)W_p & | & (+r)R_n(-r)W_p \\
W_n & \longrightarrow & & & xW_p & | & (+w)W_n(-w)W_p & | & (+r)R_n(-r)W_p \\
R_p & \longrightarrow & \epsilon & | & xR_p & | & (+w)W_n(-w)R_p & & \\
R_n & \longrightarrow & & & xR_p & | & (+w)W_n(-w)R_p & &
\end{array}
$$

ENUMERATING THE $\mathcal{F}_{\ominus}$-MACHINE

$$
\begin{array}{lcccccccc}
S & \longrightarrow & \epsilon & | & xS & | & (+w)W_n(-w)S & & \\
W_p & \longrightarrow & \epsilon & | & xW_p & | & (+w)W_n(-w)W_p & | & (+r)R_n(-r)W_p \\
W_n & \longrightarrow & & & xW_p & | & (+w)W_n(-w)W_p & | & (+r)R_n(-r)W_p \\
R_p & \longrightarrow & \epsilon & | & xR_p & | & (+w)W_n(-w)R_p & & \\
R_n & \longrightarrow & & & xR_p & | & (+w)W_n(-w)R_p & &
\end{array}
$$

W states: leftmost block is a single entry
R states: leftmost block is two entries

ENUMERATING THE $\mathcal{F}_{\ominus}$-MACHINE

$$\begin{array}{ccccccccc}
S & \longrightarrow & \epsilon & \mid & xS & \mid & (+w)W_n(-w)S & & \\
W_p & \longrightarrow & \epsilon & \mid & xW_p & \mid & (+w)W_n(-w)W_p & \mid & (+r)R_n(-r)W_p \\
W_n & \longrightarrow & & & xW_p & \mid & (+w)W_n(-w)W_p & \mid & (+r)R_n(-r)W_p \\
R_p & \longrightarrow & \epsilon & \mid & xR_p & \mid & (+w)W_n(-w)R_p & & \\
R_n & \longrightarrow & & & xR_p & \mid & (+w)W_n(-w)R_p & &
\end{array}$$

W states: leftmost block is a single entry
R states: leftmost block is two entries

This is easily solved to find that the algebraic generating function $s(x)$ for Av(4123, 4132, 4213) has minimal polynomial

$$1 + (x-1)s(x) - xs(x)^2 + xs(x)^3,$$

Enumerating the $\mathcal{F}_{\ominus}$-Machine

$$\begin{array}{lllllllll}
S & \longrightarrow & \epsilon & | & xS & | & (+w)W_n(-w)S & & \\
W_p & \longrightarrow & \epsilon & | & xW_p & | & (+w)W_n(-w)W_p & | & (+r)R_n(-r)W_p \\
W_n & \longrightarrow & & & xW_p & | & (+w)W_n(-w)W_p & | & (+r)R_n(-r)W_p \\
R_p & \longrightarrow & \epsilon & | & xR_p & | & (+w)W_n(-w)R_p & & \\
R_n & \longrightarrow & & & xR_p & | & (+w)W_n(-w)R_p & &
\end{array}$$

W states: leftmost block is a single entry
R states: leftmost block is two entries

This is easily solved to find that the algebraic generating function $s(x)$ for $\text{Av}(4123, 4132, 4213)$ has minimal polynomial

$$1 + (x-1)s(x) - xs(x)^2 + xs(x)^3,$$

and growth rate

$$\frac{67240 + (779\sqrt{57} - 1927)r^{1/3} - (19\sqrt{57} - 457)r^{2/3}}{40344} \approx 5.219,$$

where $r = 1502 + 342\sqrt{57}$.

More Exotic $\mathcal{C}$-Machines

We have found four simple machines that generate classes whose enumerations are unknown.

MORE EXOTIC $\mathcal{C}$-MACHINES

We have found four simple machines that generate classes whose enumerations are unknown.

For each of these machines, we have been able to set up functional equations (that we can't solve), as well as generate many terms in the counting sequence.

MORE EXOTIC $\mathcal{C}$-MACHINES

We have found four simple machines that generate classes whose enumerations are unknown.

For each of these machines, we have been able to set up functional equations (that we can't solve), as well as generate many terms in the counting sequence.

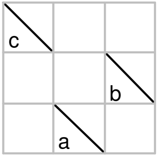
We have tried to guess algebraic, D-finite, and D-algebraic generating functions based on these terms, but have found none.

THE Av(123, 231)-MACHINE

The class Av(123, 231) is a geometric grid class.

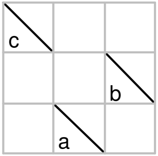
THE Av(123, 231)-MACHINE

The class Av(123, 231) is a geometric grid class.



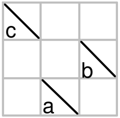
THE Av(123, 231)-MACHINE

The class Av(123, 231) is a geometric grid class.

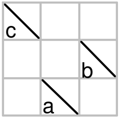


The states of the machine can be represented by 4-tuples (a, b, c, P) where a, b, and c are the number of a, b, c entries, and P is a boolean representing whether we can or can't pop.

THE Av(123, 231)-MACHINE

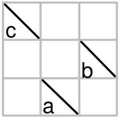


THE Av(123, 231)-MACHINE



We can describe the state transitions:

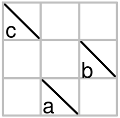
THE Av(123, 231)-MACHINE



We can describe the state transitions:

- $(0, 0, 0, T) \longrightarrow \{(1, 0, 0, F), (0, 0, 0, T)\}$

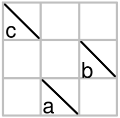
THE Av(123, 231)-MACHINE



We can describe the state transitions:

- $(0, 0, 0, T) \longrightarrow \{(1, 0, 0, F), (0, 0, 0, T)\}$
- $(a, 0, 0, F) \longrightarrow \{(a + 1, 0, 0, F), (a, 1, 0, F), (a, 0, 0, T)\}$

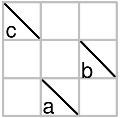
THE Av(123, 231)-MACHINE



We can describe the state transitions:

- $(0, 0, 0, T) \longrightarrow \{(1, 0, 0, F), (0, 0, 0, T)\}$
- $(a, 0, 0, F) \longrightarrow \{(a + 1, 0, 0, F), (a, 1, 0, F), (a, 0, 0, T)\}$
- $(a, 0, 0, T) \longrightarrow \{(a + 1, 0, 0, F), (a, 1, 0, F), (a, 0, 0, T), (a - 1, 0, 0, T)\}$

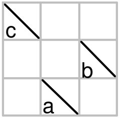
THE Av(123, 231)-MACHINE



We can describe the state transitions:

- $(0,0,0,T) \longrightarrow \{(1,0,0,F),(0,0,0,T)\}$
- $(a,0,0,F) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T)\}$
- $(a,0,0,T) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T),(a-1,0,0,T)\}$
- $(a,b,0,F) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T)\}$

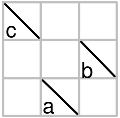
The Av(123, 231)-Machine



We can describe the state transitions:

- $(0,0,0,T) \longrightarrow \{(1,0,0,F),(0,0,0,T)\}$
- $(a,0,0,F) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T)\}$
- $(a,0,0,T) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T),(a-1,0,0,T)\}$
- $(a,b,0,F) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T)\}$
- $(a,b,0,T) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T),(a-1,b,0,T)\}, \quad (a \geq 2)$

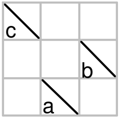
THE Av(123, 231)-MACHINE



We can describe the state transitions:

- $(0,0,0,T) \longrightarrow \{(1,0,0,F),(0,0,0,T)\}$
- $(a,0,0,F) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T)\}$
- $(a,0,0,T) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T),(a-1,0,0,T)\}$
- $(a,b,0,F) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T)\}$
- $(a,b,0,T) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T),(a-1,b,0,T)\}, \quad (a \geq 2)$
- $(1,b,0,T) \longrightarrow \{(1,b+1,0,F),(1,b,1,F),(1,b,0,T),(b,0,0,T)\}$

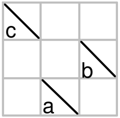
THE Av(123, 231)-MACHINE



We can describe the state transitions:

- $(0,0,0,T) \longrightarrow \{(1,0,0,F),(0,0,0,T)\}$
- $(a,0,0,F) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T)\}$
- $(a,0,0,T) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T),(a-1,0,0,T)\}$
- $(a,b,0,F) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T)\}$
- $(a,b,0,T) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T),(a-1,b,0,T)\}, \quad (a \geq 2)$
- $(1,b,0,T) \longrightarrow \{(1,b+1,0,F),(1,b,1,F),(1,b,0,T),(b,0,0,T)\}$
- $(a,b,c,F) \longrightarrow \{(a,b,c+1,F),(a,b,c,T)\}$

THE Av(123, 231)-MACHINE



We can describe the state transitions:

- $(0,0,0,T) \longrightarrow \{(1,0,0,F),(0,0,0,T)\}$
- $(a,0,0,F) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T)\}$
- $(a,0,0,T) \longrightarrow \{(a+1,0,0,F),(a,1,0,F),(a,0,0,T),(a-1,0,0,T)\}$
- $(a,b,0,F) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T)\}$
- $(a,b,0,T) \longrightarrow \{(a,b+1,0,F),(a,b,1,F),(a,b,0,T),(a-1,b,0,T)\}, \quad (a \geq 2)$
- $(1,b,0,T) \longrightarrow \{(1,b+1,0,F),(1,b,1,F),(1,b,0,T),(b,0,0,T)\}$
- $(a,b,c,F) \longrightarrow \{(a,b,c+1,F),(a,b,c,T)\}$
- $(a,b,c,T) \longrightarrow \{(a,b,c+1,F),(a,b,c,T),(a,b,c-1,T)\}$

THE Av(123, 231)-MACHINE

With a little work, we can translate the state transitions into a set of functional equations:

$$\begin{aligned} A(a,x) &= 1 + \frac{x}{a}(A(a,x) - A(0,x)) + aA(a,x) + xB(0,a,x), \\ B(a,b,x) &= \frac{1}{a}(A(a,x) - A(0,x))\frac{bC}{1-b} + B(a,b,x)\frac{bC}{1-b} \\ &\qquad + \frac{x}{a}(1+C)(B(a,b,x) - B(0,b,x)). \end{aligned}$$

THE Av(123, 231)-MACHINE

With a little work, we can translate the state transitions into a set of functional equations:

$$\begin{aligned} A(a,x) &= 1 + \frac{x}{a}(A(a,x) - A(0,x)) + aA(a,x) + xB(0,a,x), \\ B(a,b,x) &= \frac{1}{a}(A(a,x) - A(0,x))\frac{bC}{1-b} + B(a,b,x)\frac{bC}{1-b} \\ &\quad + \frac{x}{a}(1+C)(B(a,b,x) - B(0,b,x)). \end{aligned}$$

The generating function for the class is $A(0,x)$, but we have no idea how to solve these equations.

THE Av(123, 231)-MACHINE

Using the state transitions, dynamic programming, and Amazon cloud computing, we have computed the first 1000 terms of Av(4123, 4231).

THE Av(123, 231)-MACHINE

Using the state transitions, dynamic programming, and Amazon cloud computing, we have computed the first 1000 terms of Av(4123, 4231).

Despite all of this information, we have been unable to determine the nature of the generating function.

THE Av(123, 231)-MACHINE

Some common forms of generating functions are:

THE $\mathrm{Av}(123, 231)$-MACHINE

Some common forms of generating functions are:

- rational: $f(x) = \frac{p(x)}{q(x)}$ for polynomials $p(x)$ and $q(x)$

THE Av(123, 231)-MACHINE

Some common forms of generating functions are:

- rational: $f(x) = \frac{p(x)}{q(x)}$ for polynomials $p(x)$ and $q(x)$
- algebraic: there exists a bivariate polynomial $F(x, y)$ such that $F(x, f) = 0$

THE Av(123, 231)-MACHINE

Some common forms of generating functions are:

- rational: $f(x) = \frac{p(x)}{q(x)}$ for polynomials $p(x)$ and $q(x)$
- algebraic: there exists a bivariate polynomial $F(x, y)$ such that $F(x, f) = 0$
- D-finite: $f(x)$ is the solution to a linear ODE with polynomial coefficients

THE Av(123, 231)-MACHINE

Some common forms of generating functions are:

- rational: $f(x) = \frac{p(x)}{q(x)}$ for polynomials $p(x)$ and $q(x)$
- algebraic: there exists a bivariate polynomial $F(x, y)$ such that $F(x, f) = 0$
- D-finite: $f(x)$ is the solution to a linear ODE with polynomial coefficients
- D-algebraic: there exists a polynomial $F(x, y_0, y_1, \cdots, y_n)$ such that $F(x, y, y', \cdots, y^{(n)}) = 0$

THE Av(123, 231)-MACHINE

Some common forms of generating functions are:

- rational: $f(x) = \frac{p(x)}{q(x)}$ for polynomials $p(x)$ and $q(x)$
- algebraic: there exists a bivariate polynomial $F(x, y)$ such that $F(x, f) = 0$
- D-finite: $f(x)$ is the solution to a linear ODE with polynomial coefficients
- D-algebraic: there exists a polynomial $F(x, y_0, y_1, \cdots, y_n)$ such that $F(x, y, y', \cdots, y^{(n)}) = 0$

For example, the exponential generating function $B(x)$ for the Bell numbers is D-algebraic because

$$B(x)B'(x) - B(x)B''(x) + B'(x)^2 = 0.$$

THE Av(123, 231)-MACHINE

We wrote an algorithm to conjecture a generating function given initial terms.

THE Av(123, 231)-MACHINE

We wrote an algorithm to conjecture a generating function given initial terms.

▶ *Example:*

```
> guessade(L, egf);

ADE found! (0.157 seconds)

                                                 / 2       \
               /d       \                        |d         |    /d       \2
        F(x) |-- F(x)| - F(x) |--- F(x)| + |-- F(x)|
               \dx      /                        |  2       |    \dx      /
                                                 \dx       /
```

THE Av(123, 231)-MACHINE

We wrote an algorithm to conjecture a generating function given initial terms.

- *Example:*

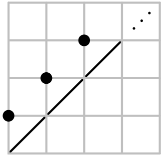
```
> guessade(L, egf);

ADE found! (0.157 seconds)

                                              / 2       \
              /d        \                     |d        |   /d       \2
         F(x) |-- F(x)| - F(x) |--- F(x)| + |-- F(x)|
              \dx       /                     |  2      |   \dx      /
                                              \dx       /
```

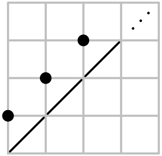
Despite having 1000 initial terms of Av(4123, 4231), we can't guess any generating function!

THREE MORE EXOTIC MACHINES

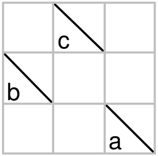


600 terms
Av(4231, 4321)

THREE MORE EXOTIC MACHINES

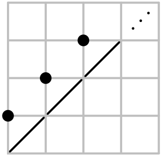


600 terms
Av(4231, 4321)

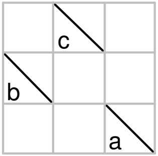


1000 terms
Av(4123, 4312)

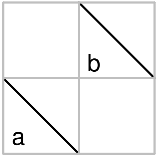
THREE MORE EXOTIC MACHINES



600 terms
$\text{Av}(4231, 4321)$



1000 terms
$\text{Av}(4123, 4312)$



5000 terms
$\text{Av}(4123, 4231, 4312)$

UNSOLVED MYSTERIES

Conway and Guttmann analyzed 36 terms of the counting sequence of Av(4231), which is also suspected to have a non-D-finite generating function and found that the sequence exhibited strange behavior.

UNSOLVED MYSTERIES

Conway and Guttmann analyzed 36 terms of the counting sequence of Av(4231), which is also suspected to have a non-D-finite generating function and found that the sequence exhibited strange behavior.

We are in the process of exploring the asymptotic behavior of these four exotic classes in a similar way.

Thanks for coming! Any questions?