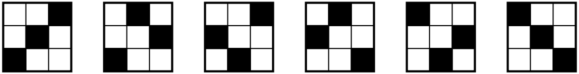
Sorting with C-machines: Enumerative and Analytic Aspects

Jay Pantone

Dartmouth College

Hanover, NH



BIRS Workshop in Analytic and Probabilistic Combinatorics

October 26, 2016

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.

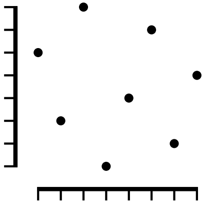
Permutations

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 $\Longleftrightarrow$



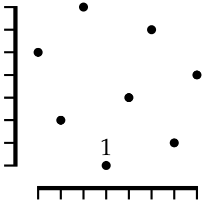
Permutations

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 $\Longleftrightarrow$



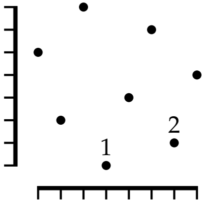
PERMUTATIONS

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

- *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 $\Longleftrightarrow$



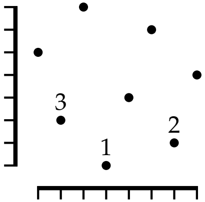
Permutations

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 $\iff$



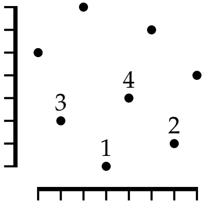
Permutations

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 $\iff$



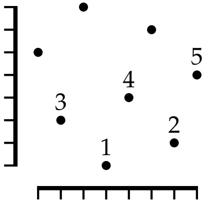
Permutations

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 $\iff$

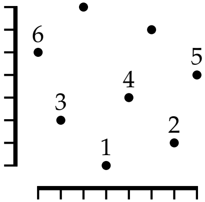


Permutations

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

- *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 $\iff$ 

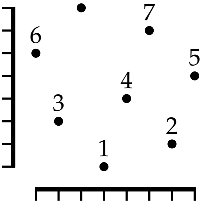
Permutations

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 $\iff$



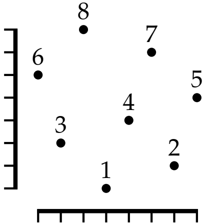
Permutations

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 $\Longleftrightarrow$



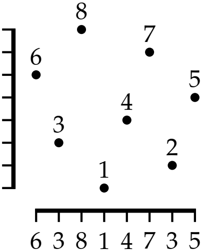
Permutations

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 $\Longleftrightarrow$



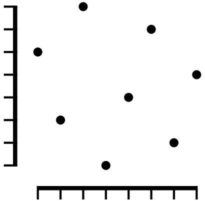
Pattern Containment

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

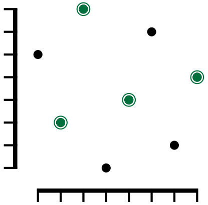
$$1423 \leqslant 63814725 \qquad \Longleftrightarrow$$



Pattern Containment

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

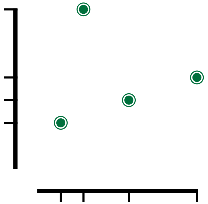
$$1423 \leqslant 63814725 \qquad \Longleftrightarrow$$



Pattern Containment

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

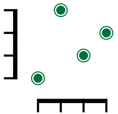
$$1423 \leqslant 63814725 \qquad \Longleftrightarrow$$



Pattern Containment

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

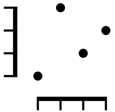
$$1423 \leqslant 63814725 \qquad \Longleftrightarrow$$



Pattern Containment

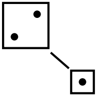
A permutation π contains a permutation σ (denoted $\sigma \leqslant \pi$) if you can delete dots in the picture of π and shrink the picture to get the picture of σ.

$$1423 \leqslant 63814725 \qquad \Longleftrightarrow$$

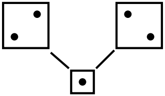


Permutation Poset

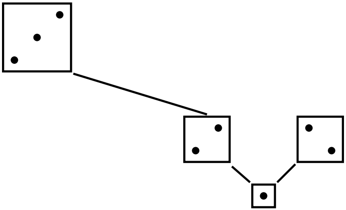
Permutation Poset



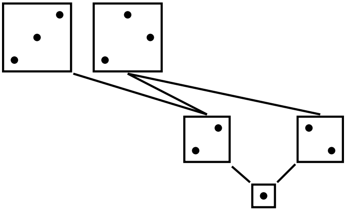
Permutation Poset



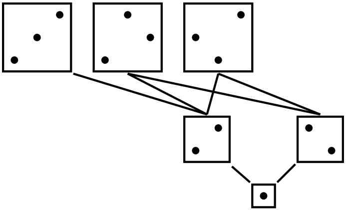
Permutation Poset



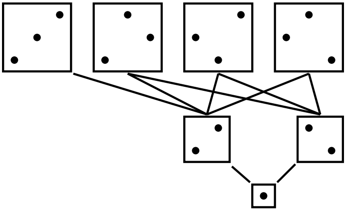
Permutation Poset



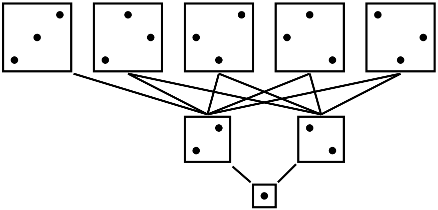
Permutation Poset



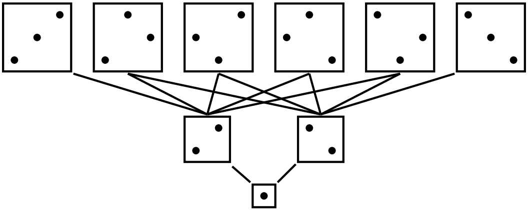
Permutation Poset



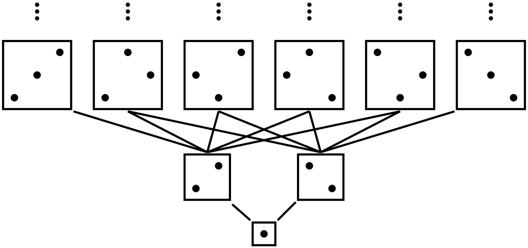
Permutation Poset



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 $\sigma \leqslant \pi$, 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 $\sigma \leqslant \pi$, 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*.

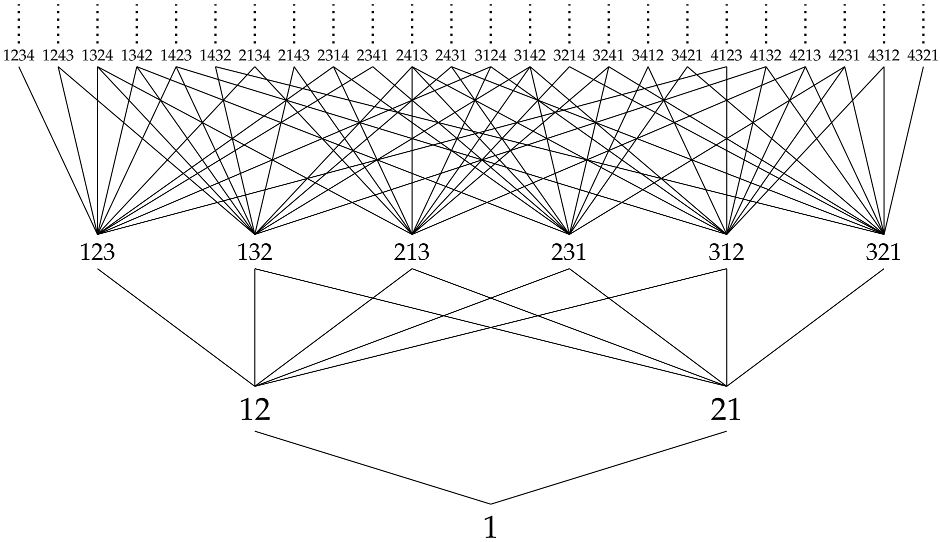
Permutation Poset

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

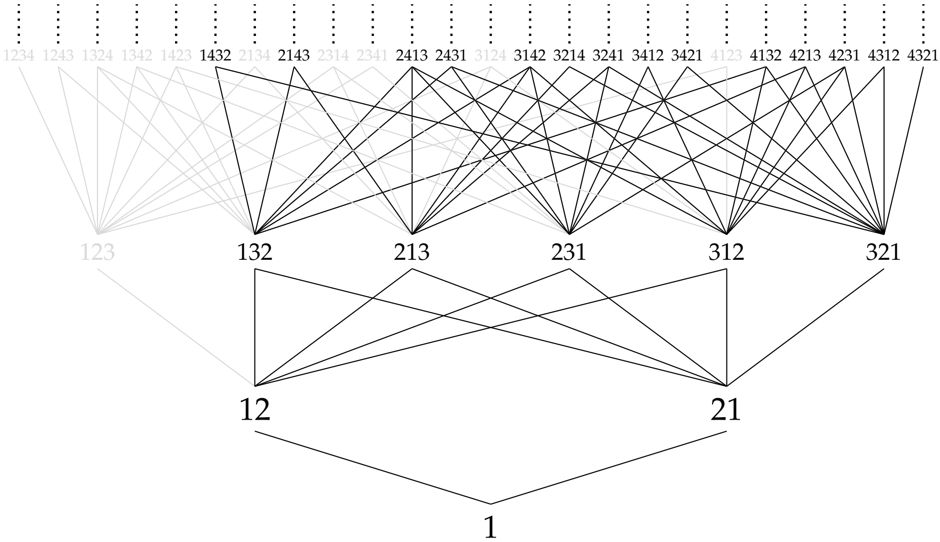
A class can be specified by the set of minimal permutations not in the class, called its *basis*.

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

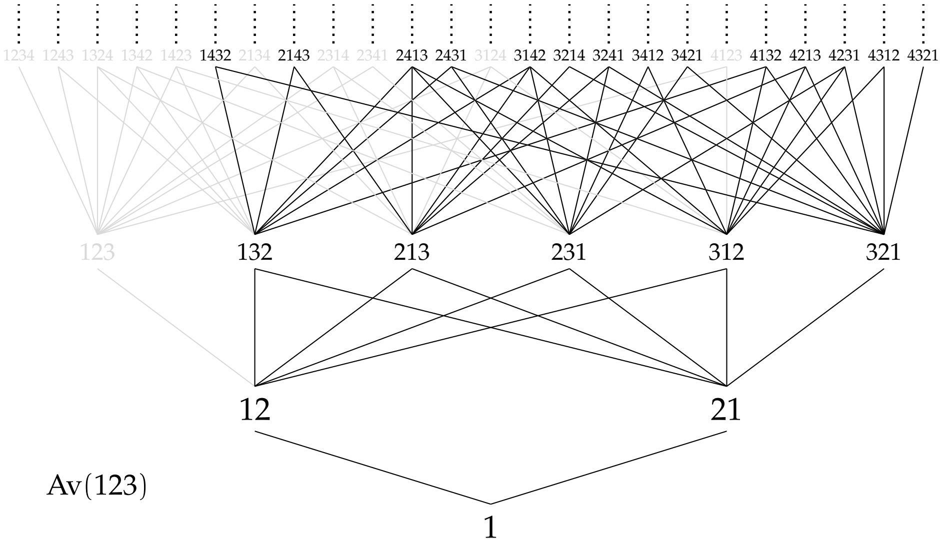
Permutation Poset



Permutation Poset

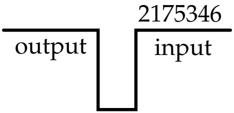


Permutation Poset



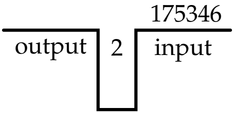
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



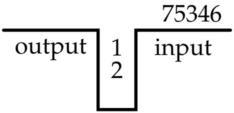
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



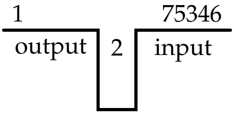
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



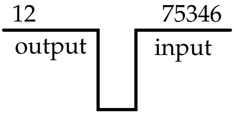
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



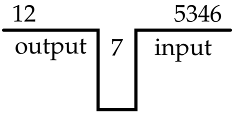
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



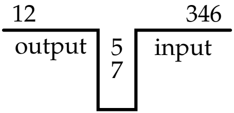
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



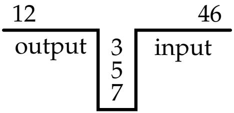
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



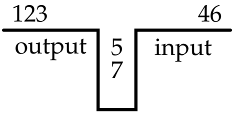
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



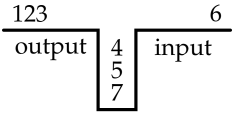
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



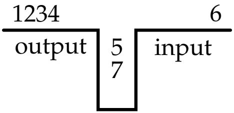
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



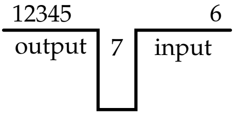
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



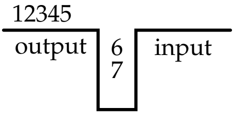
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



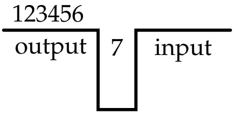
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



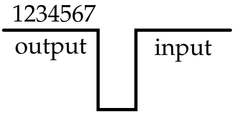
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



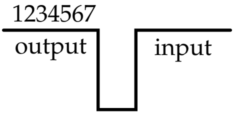
Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



Sorting with a Stack

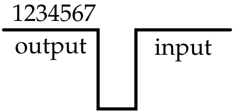
The permutation 2175346 can be sorted by a stack.



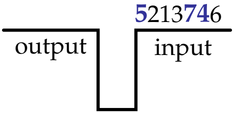
Not all permutations can be sorted by a stack.

Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

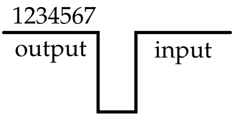


Not all permutations can be sorted by a stack.

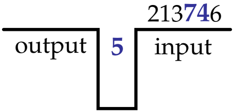


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

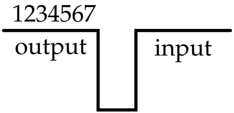


Not all permutations can be sorted by a stack.

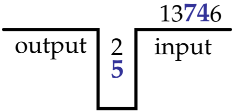


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

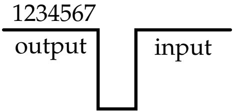


Not all permutations can be sorted by a stack.

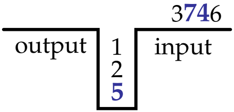


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

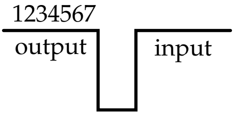


Not all permutations can be sorted by a stack.

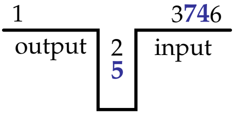


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

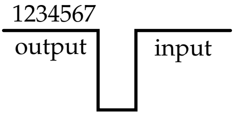


Not all permutations can be sorted by a stack.

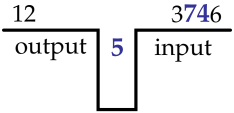


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

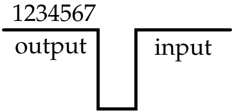


Not all permutations can be sorted by a stack.

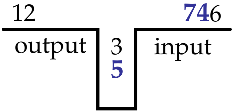


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

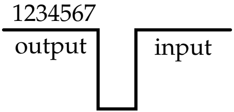


Not all permutations can be sorted by a stack.

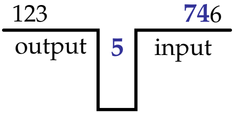


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

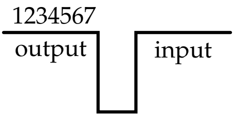


Not all permutations can be sorted by a stack.

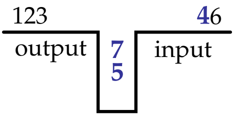


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

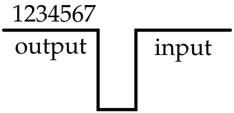


Not all permutations can be sorted by a stack.

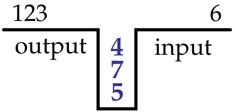


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

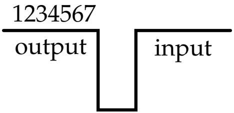


Not all permutations can be sorted by a stack.

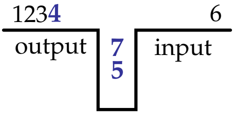


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

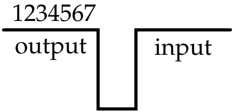


Not all permutations can be sorted by a stack.

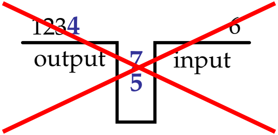


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.

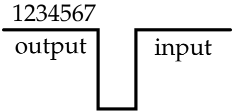


Not all permutations can be sorted by a stack.

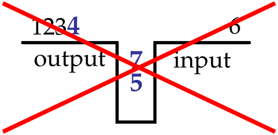


Sorting with a Stack

The permutation 2175346 can be sorted by a stack.



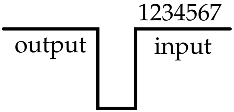
Not all permutations can be sorted by a stack.



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

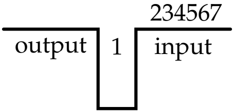
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

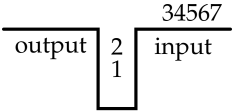
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

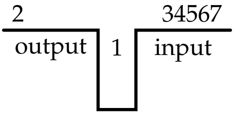
The permutation 2156473 can be <u>generated</u> by a stack.



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

Generating with a Stack

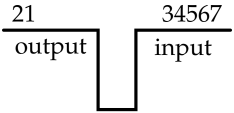
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

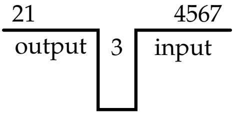
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

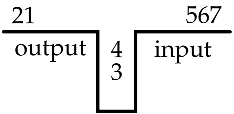
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

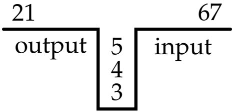
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

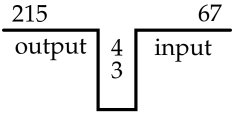
The permutation 2156473 can be <u>generated</u> by a stack.



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

Generating with a Stack

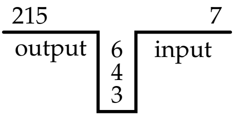
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

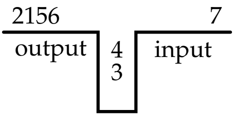
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

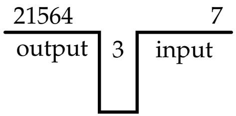
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

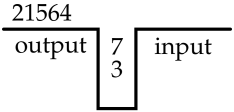
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

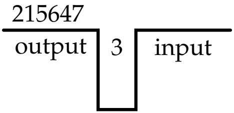
The permutation 2156473 can be <u>generated</u> by a stack.



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

GENERATING WITH A STACK

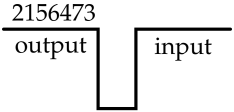
The permutation 2156473 can be generated by a stack.



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

Generating with a Stack

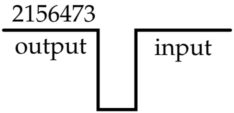
The permutation 2156473 can be generated by a stack.



(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.

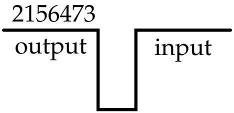


(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

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.



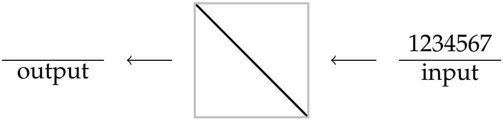
(Notice the stack always holds a decreasing subpermutation when read top to bottom.)

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

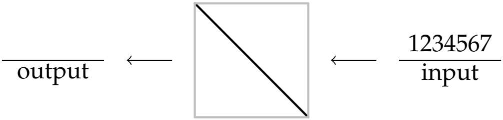
The stack can be thought of as a container that always holds a decreasing permutation, where at any time we can:



Generalizing Stacks

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

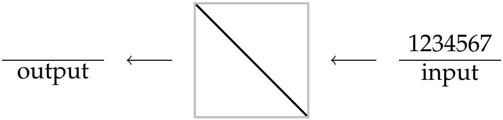
- push a new maximum entry into the container such that the container holds a decreasing permutation



Generalizing Stacks

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

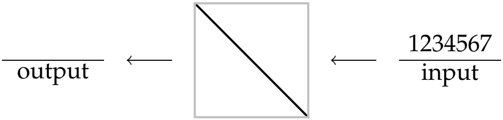
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

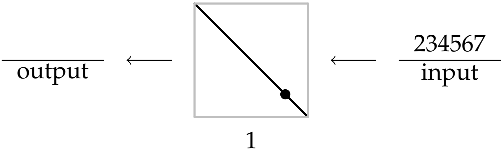
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

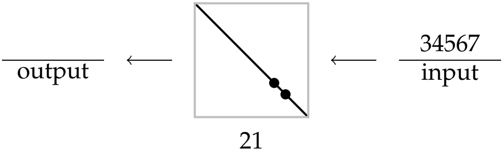
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

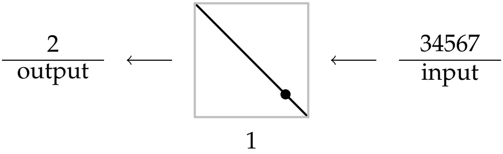
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

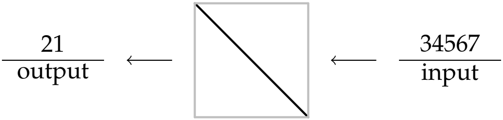
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

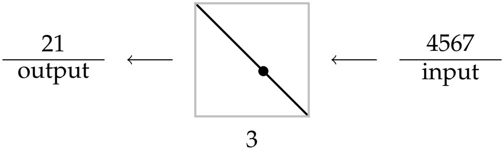
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

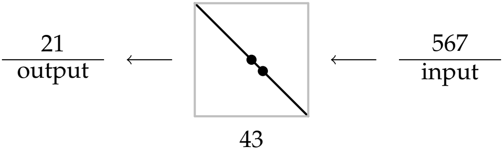
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

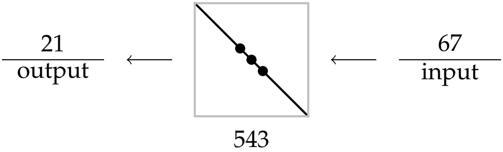
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

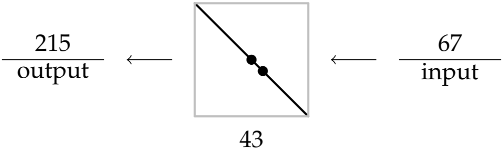
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

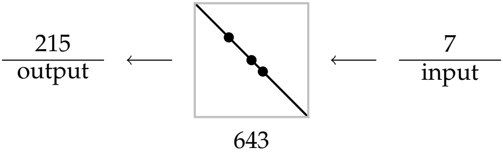
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

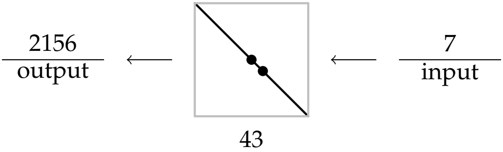
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

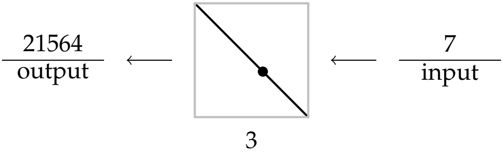
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

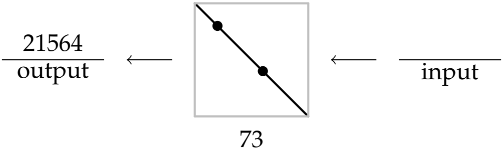
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

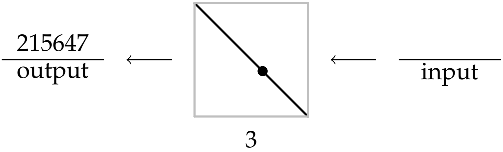
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

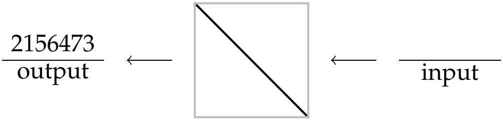
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



Generalizing Stacks

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

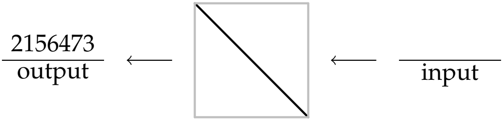
- push a new maximum entry into the container such that the container holds a decreasing permutation
- pop the leftmost entry out of the container.



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

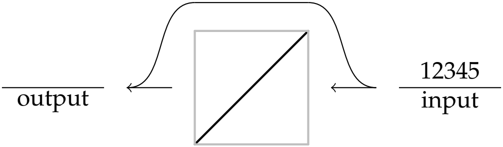
We allow a third operation: an entry can bypass the container and move straight from the input to the output.

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.

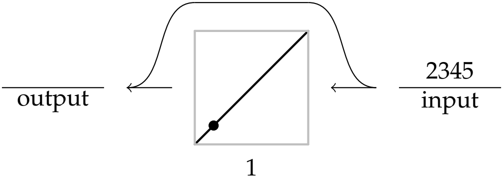
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

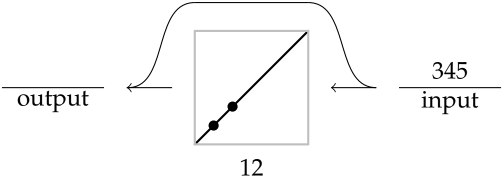
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

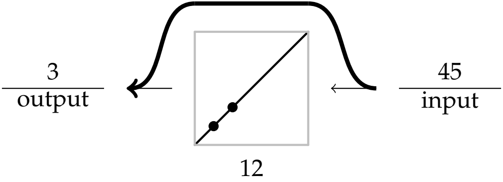
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

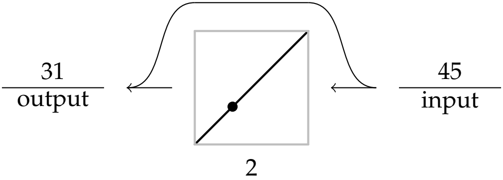
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

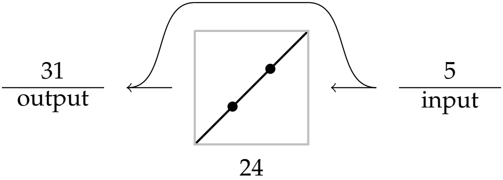
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

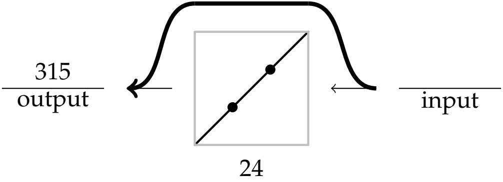
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

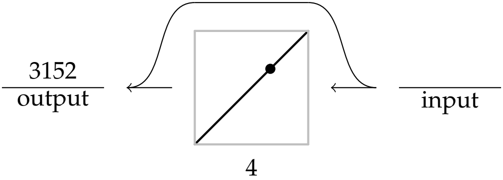
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

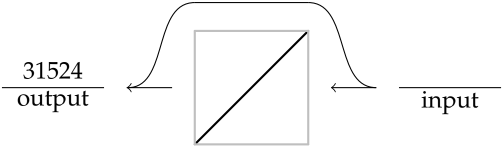
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

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

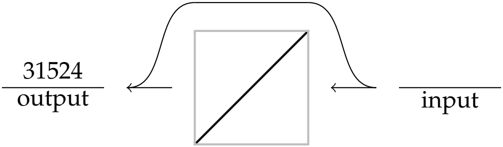
In the Av(12)-machine, we didn't need the bypass because we could just push and then immediately pop.



Generalizing Stacks

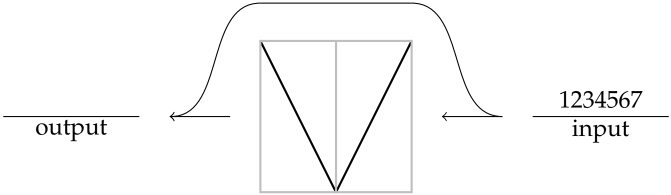
We allow a third operation: an entry can bypass the container and move straight from the input to the output.

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



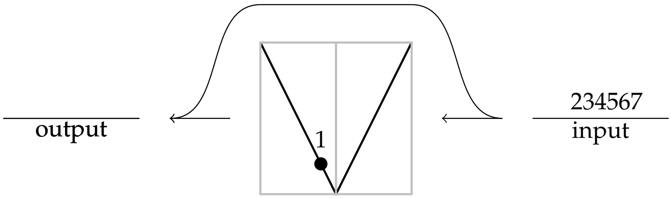
The Av(21)-machine generates the class Av(321).

Example: The Av$(231, 132)$-Machine



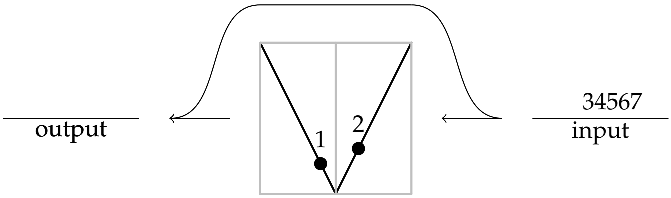
Whenever the machine is nonempty, there are always two places to push a new entry.

EXAMPLE: THE Av(231, 132)-MACHINE



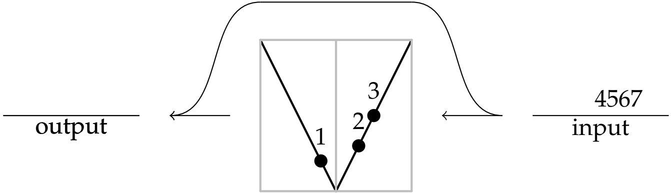
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



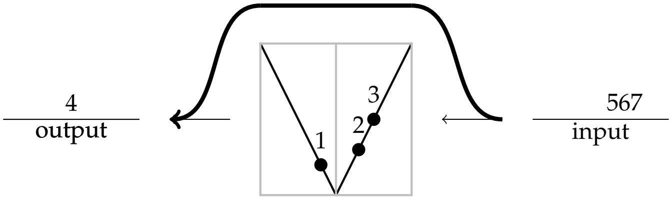
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



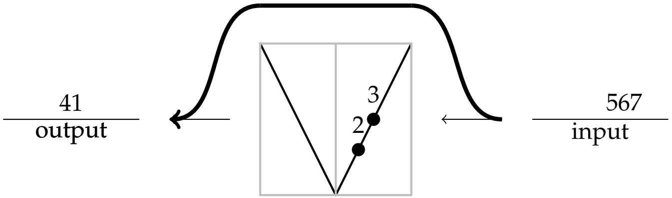
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



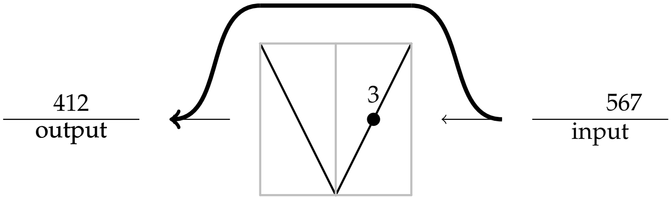
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



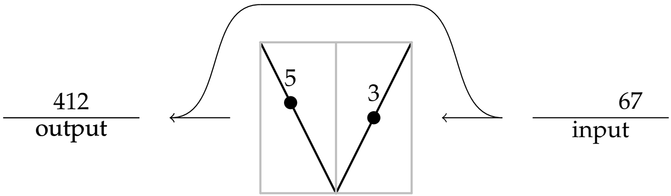
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



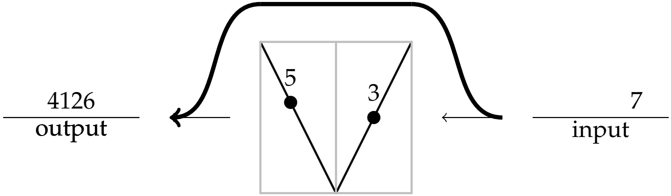
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



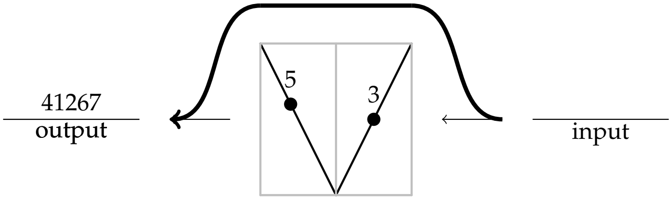
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



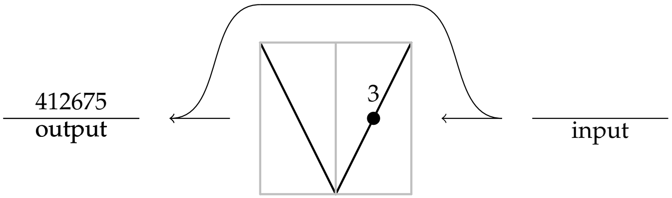
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



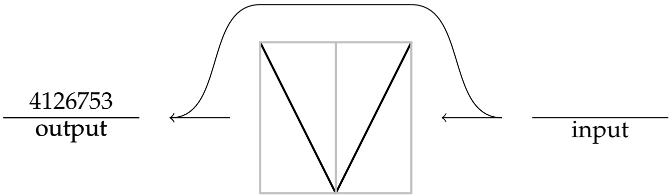
Whenever the machine is nonempty, there are always two places to push a new entry.

Example: The Av(231, 132)-Machine



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

EXAMPLE: THE Av(231, 132)-MACHINE



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

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 $\text{Av}(12)$-machine generates the class $\text{Av}(312)$.
- The $\text{Av}(21)$-machine generates the class $\text{Av}(321)$.

Theorem. The $\text{Av}(B)$-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

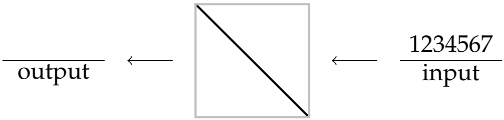
The operation sequence of the Av(12)-machine makes it easy to find the generating function for Av(312).

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).

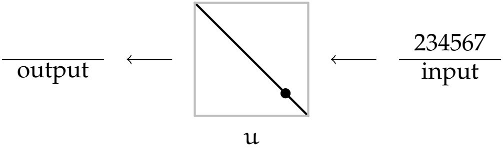
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

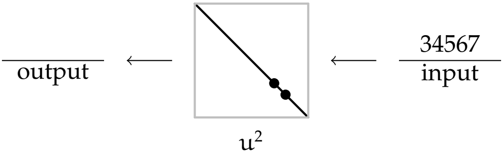
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

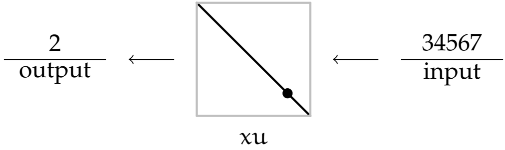
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

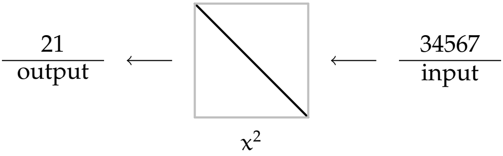
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

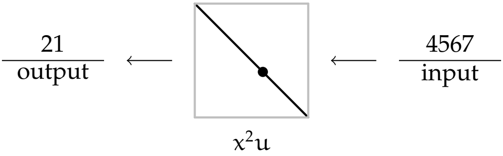
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

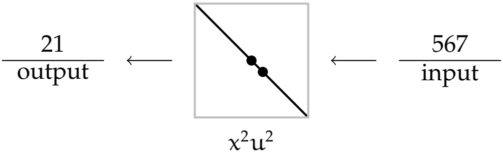
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

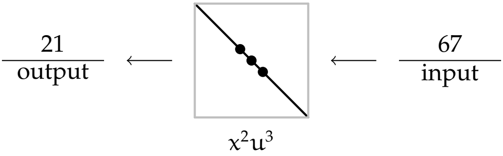
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

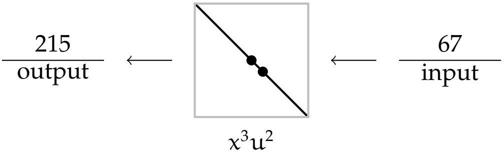
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

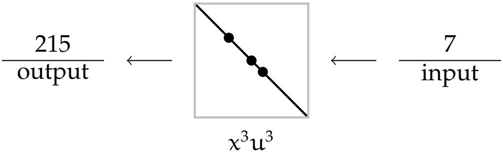
Let $f(x,u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

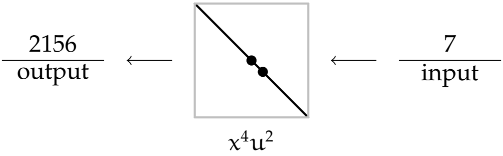
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

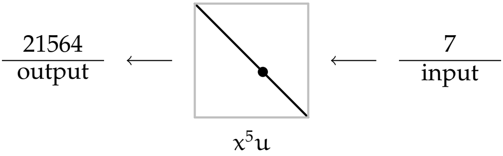
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

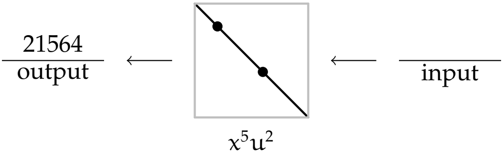
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

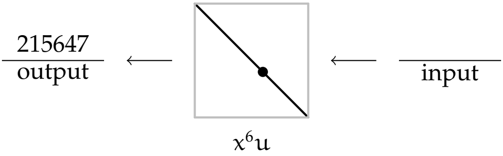
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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).

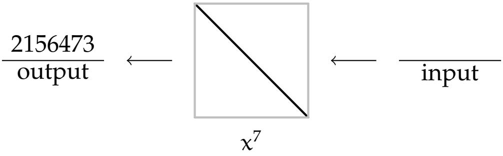
Let $f(x, u)$ be the generating function for valid states of the Av(12)-machine where u tracks the number of entries in the machine and x tracks the number that have been output so far.



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.

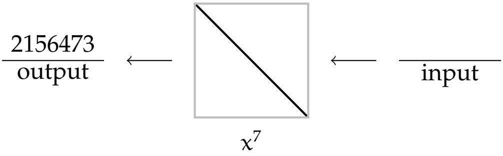


States with no entries in the machine are those with no u term.

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:

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

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:

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

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:

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

The kernel method shows that $f(x,0) = \dfrac{1-\sqrt{1-4x}}{2x}$.

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:

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

The kernel method shows that $f(x,0) = \dfrac{1-\sqrt{1-4x}}{2x}$.

(There are some uniqueness issues I'm sweeping under the rug.)

2×4 Classes

There are 56 essentially different classes of permutations that avoid two patterns of length 4.

2×4 Classes

There are 56 essentially different classes of permutations that avoid two patterns of length 4.

Enumerations are known for all but three:

- $\mathrm{Av}(4123, 4231)$
- $\mathrm{Av}(4123, 4312)$
- $\mathrm{Av}(4231, 4321)$

2×4 Classes

There are 56 essentially different classes of permutations that avoid two patterns of length 4.

Enumerations are known for all but three:

- $\mathrm{Av}(4123, 4231)$
- $\mathrm{Av}(4123, 4312)$
- $\mathrm{Av}(4231, 4321)$

These three can all be modeled with $\mathcal{C}$-machines.

$\mathrm{Av}(4123, 4231)$

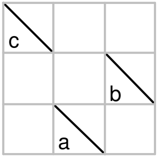
$\mathrm{Av}(4123, 4231)$ is generated by the $\mathrm{Av}(123, 231)$-machine.

The class $\mathrm{Av}(123, 231)$ is a geometric grid class.

Av(4123, 4231)

Av(4123, 4231) is generated by the Av(123, 231)-machine.

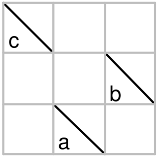
The class Av(123, 231) is a geometric grid class.



$\mathrm{Av}(4123, 4231)$

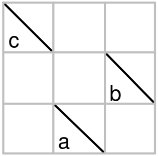
$\mathrm{Av}(4123, 4231)$ is generated by the $\mathrm{Av}(123, 231)$-machine.

The class $\mathrm{Av}(123, 231)$ is a geometric grid class.



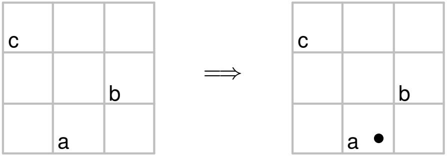
The states of the machine can be represented by 4-tuples (a, b, c, P) where a, b, and c are the number of a, b, c entries, and P is a boolean representing whether we can or can't pop.

Av(4123, 4231)



We can describe the state transitions:

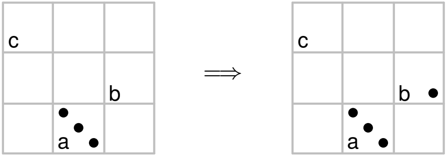
Av(4123, 4231)



We can describe the state transitions:

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

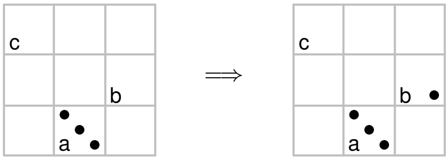
$\mathrm{Av}(4123, 4231)$



We can describe the state transitions:

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

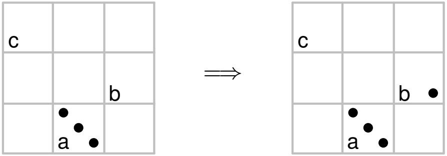
$\mathrm{Av}(4123, 4231)$



We can describe the state transitions:

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

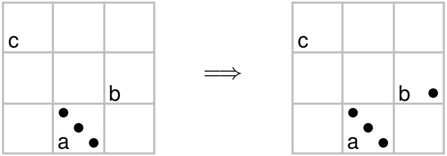
Av(4123, 4231)



We can describe the state transitions:

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

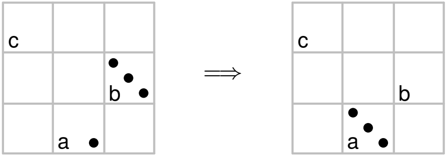
Av(4123, 4231)



We can describe the state transitions:

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

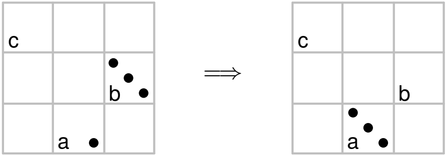
Av(4123, 4231)



We can describe the state transitions:

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

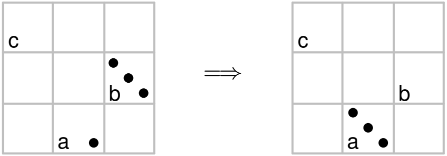
$\mathrm{Av}(4123, 4231)$



We can describe the state transitions:

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

$\mathrm{Av}(4123, 4231)$



We can describe the state transitions:

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

Av(4123, 4231)

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

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

(C is the generating function for the Catalan numbers.)

Av(4123, 4231)

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

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

(C is the generating function for the Catalan numbers.)

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

Av(4123, 4312)

$$
\begin{aligned}
A(a,x) &= 1 + \frac{x}{a}(A(a,x) - A(0,x)) + aA(a,x) + xB(a,0,a,x) \\
B(a,b,c,x) &= \frac{cx}{(1-c)(1-x)}\left(\frac{A(a,x)-A(b,x)}{a-b}\right) + \frac{cx}{(1-c)(1-x)}B(a,b,c,x) \\
&\quad + \frac{x}{b(1-x)}(B(a,b,c,x) - B(a,0,c,x))
\end{aligned}
$$

Av(4231, 4321)

$$
\begin{aligned}
A_p &= 1 + x(A_p(a,x) + A_n(a,x)) + \frac{x}{a}(A_p(a,x) - A_p(0,x)) + \frac{x}{a}B_p(a,a,0,x) \\
A_n &= a(A_p(a,x) + A_n(a,x)) \\
B_p &= x(B_p(a,b,c,x) + B_n(a,b,c,x)) + \frac{x}{c}(B_p(a,b,c,x) - B_p(a,b,0,x)) \\
B_n &= a(B_p(a,b,c,x) + B_n(a,b,c,x)) + \frac{b^2}{c-b}((A_p(c,x) - A_p(b,x)) \\
&\qquad + (A_n(c,x) - A_n(b,x)))\frac{b^2}{c-b}((B_p(c,c,c,x) - B_p(b,c,c,x)) \\
&\qquad + +(B_n(c,c,c,x) - B_n(b,c,c,x)))
\end{aligned}
$$

Initial terms

Although we don't know how to solve these functional equations, they can be used to obtain many initial terms of each sequence. (Between 600 and 5000 terms. Probably more are possible.)

Initial terms

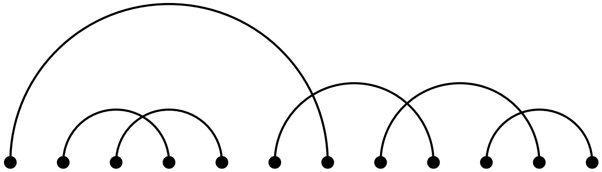
Although we don't know how to solve these functional equations, they can be used to obtain many initial terms of each sequence. (Between 600 and 5000 terms. Probably more are possible.)

We can use two tools to analyze these terms:

- Automated Guessing
- The Method of Differential Approximants

Quick Detour: 2-Colorable Matchings

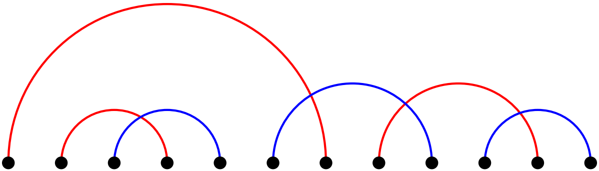
To illustrate the power of these two methods, consider the class of 2-*colorable matchings*.



I've computed 1500 terms of this sequence, but the generating function is still unknown.

Quick Detour: 2-Colorable Matchings

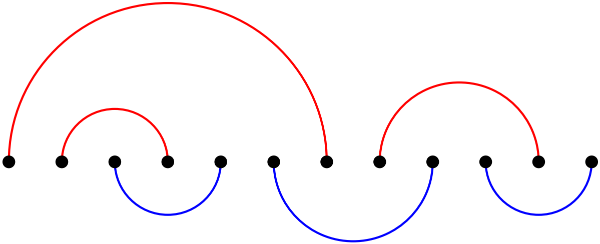
To illustrate the power of these two methods, consider the class of 2-*colorable matchings*.



I've computed 1500 terms of this sequence, but the generating function is still unknown.

Quick Detour: 2-Colorable Matchings

To illustrate the power of these two methods, consider the class of 2-*colorable matchings*.



I've computed 1500 terms of this sequence, but the generating function is still unknown.

Quick Detour: 2-Colorable Matchings

Using a few hundred terms, automated guessing says the generating function $F(z)$ satisfies the *algebraic differential equation*

$$\begin{aligned}0 = {} & (-8+12z)\,F(z)^7 + (-12z+20)\,F(z)^6 + (3z-18)\,F(z)^5 - F(z)^3 - 8z^3F'(z)^3 + 7F(z)^4 \\ & - 12z^2F(z)\,F'(z)^2 + 6z\,(3z-10)\,F(z)^4F'(z) + z^2F(z)^3F''(z)/2 + 6z^3\,(4z-3)\,F(z)^2F'(z)^3 \\ & - 8z^2\,(8z-3)\,F(z)^4F'(z)^2 + 32zF(z)^3F'(z) + 4z^2\,(4z-1)\,F(z)^6F''(z) \\ & - 12z\,(5z-4)\,F(z)^5F'(z) - 2z^2\,(4z-3)\,F(z)^5F''(z) + 16z\,(3z-1)\,F(z)^6F'(z) \\ & - 6zF(z)^2F'(z) + 42z^2F(z)^2F'(z)^2 + 12z^2\,(3z-4)\,F(z)^3F'(z)^2 - 8z^3\,(4z-1)\,F(z)^3F'(z)^3 \\ & + 15z^3F(z)\,F'(z)^3 - 3z^2F(z)^4F''(z)\end{aligned}$$

Quick Detour: 2-Colorable Matchings

The method of differential approximants predicts that $\mathsf{F}(z)$ has singularities at:

Quick Detour: 2-Colorable Matchings

The method of differential approximants predicts that $\mathsf{F}(z)$ has singularities at:

1. $z = 0.07490791222594518 \pm 10^{-17}$

Quick Detour: 2-Colorable Matchings

The method of differential approximants predicts that $\mathsf{F}(z)$ has singularities at:

1. $z = 0.07490791222594518 \pm 10^{-17}$

   Growth rate $\approx 13.349724618992102$

Quick Detour: 2-Colorable Matchings

The method of differential approximants predicts that $\mathsf{F}(z)$ has singularities at:

1. $z = 0.0749079122259451\,8 \pm 10^{-17}$

   $$\text{Growth rate} \approx 13.349724618992102$$

2. $z = -2.88202477916 \pm 10^{-11}$

QUICK DETOUR: 2-COLORABLE MATCHINGS

The method of differential approximants predicts that $\mathsf{F}(z)$ has singularities at:

1. $z = 0.074907912225945 18 \pm 10^{-17}$

   Growth rate $\approx 13.349724618992102$

2. $z = -2.88202477916 \pm 10^{-11}$

$$\left(= -\frac{256}{9\pi^2} = -2.8820247791598\ldots\right).$$

Back to the Three Unknown Permutation Classes

Applying the method of automated guessing to our three sequences:

Back to the Three Unknown Permutation Classes

Applying the method of automated guessing to our three sequences:

- Av$(4123, 4231)$, 1000 terms, no results

Back to the Three Unknown Permutation Classes

Applying the method of automated guessing to our three sequences:

- $\text{Av}(4123, 4231)$, 1000 terms, no results
- $\text{Av}(4123, 4312)$, 1000 terms, no results

Back to the Three Unknown Permutation Classes

Applying the method of automated guessing to our three sequences:

- $\mathrm{Av}(4123, 4231)$, 1000 terms, no results
- $\mathrm{Av}(4123, 4312)$, 1000 terms, no results
- $\mathrm{Av}(4231, 4321)$, 600 terms, no results

Back to the Three Unknown Permutation Classes

Applying the method of automated guessing to our three sequences:

- $\text{Av}(4123, 4231)$, 1000 terms, no results
- $\text{Av}(4123, 4312)$, 1000 terms, no results
- $\text{Av}(4231, 4321)$, 600 terms, no results

Conjecture. The generating functions for these three permutation classes are non-differentially-algebraic.

Av(4123,4231)

Dominant singularity at

$$x = 0.20092861430290850630066749465511761874842136315274588937508575345652139 \pm 10^{-71}$$

Av(4123,4231)

Dominant singularity at

$$x = 0.20092861430290850630066749465511761874842136315274588937508575345652139 \pm 10^{-71}$$

Other singularities:

- $x = 0.2061466445892927115969855884 0 \pm 10^{-29}$
- $x = 0.20724531832263 \pm 10^{-14}$
- $x = 0.24870945696 \pm 0.00390217832i \pm (1 + i)10^{-11}$

Av(4123,4312)

Dominant singularity at

$$x = 0.1715728752538099023966225515806038428606562492461038536466405240185\ldots$$
$$\ldots 3504307578592229922493134471685452997230753817540595015032 \pm 10^{-125} = 3-2\sqrt{2}$$

Av(4123,4312)

Dominant singularity at

$$x = 0.17157287525380990239662255158060384286065624924610385364664052401 85\ldots$$
$$\ldots 3504307578592229922493134471685452997230753817540595015032 \pm 10^{-125} = 3-2\sqrt{2}$$

Other singularities:

- $x = 0.1597726221747308831 \pm 10^{-19}$
- $x = 0.2439516153417787 \pm 0.1200274949895230i \pm (1+i)10^{-16}$
- $x = 0.643104131 \pm 10^{-9}$

Av(4231,4321)

Dominant singularity at

$$x = 0.1697075539292771109936127228338072222506560152943900573 3\ldots$$
$$\ldots 31923504338040028420722677 40380 \pm 10^{-93}$$

Av(4231,4321)

Dominant singularity at

$$x = 0.16970755392927711099361272283380722225065601529439005733\ldots$$
$$\ldots 319235043380400284207226774038 0 \pm 10^{-93}$$

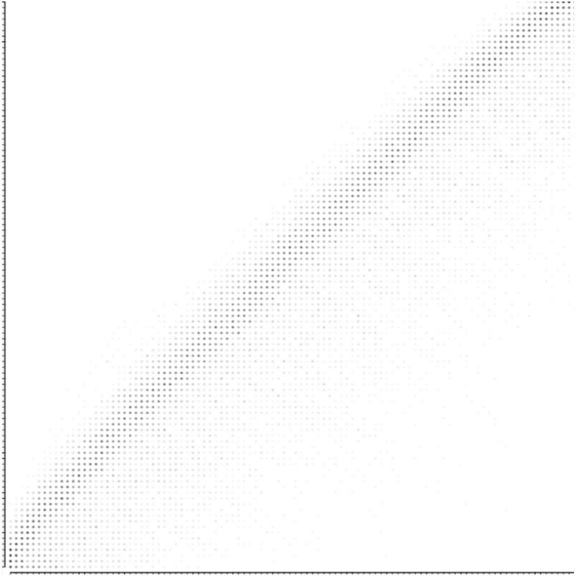
Other singularities:

- $x = .1715728752538099023966225515806038 42\ldots \pm 10^{-70}$
- $x = 0.167216154\ldots \pm 0.004184699\ldots i \pm (1+i)10^{-37}$
- $x = 0.16666666666666666666666666666666666666 \pm 10^{-37}$
- $x = 0.14589803375031545 \pm 10^{-17}$
- $x = 0.241270585132 \pm 0.024119881572i \pm (1+i)10^{-12}$
- $\ldots$

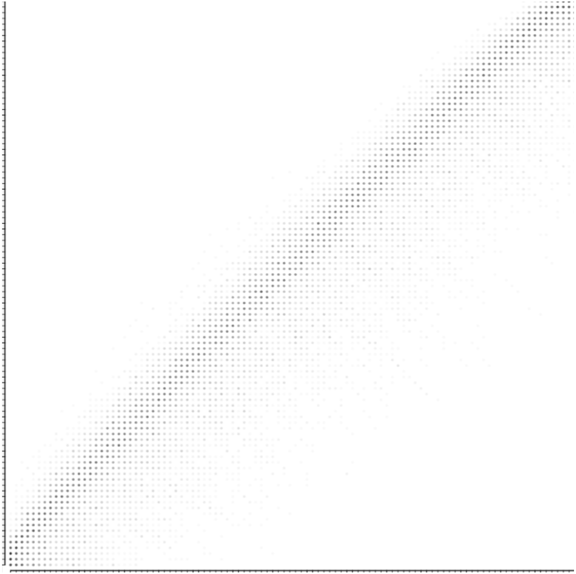
One More Epsilon of Information

Random Sampling!

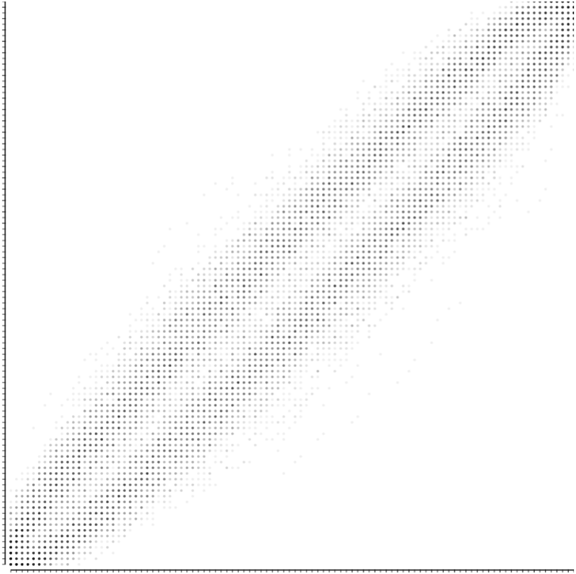
$\mathrm{Av}(4123, 4231)$



$\mathrm{Av}(4123, 4312)$



Av(4231, 4321)



Thanks!