Sorting algorithms and permutation patterns

Computer and Information Sciences departmental seminar

Henning Arnór Úlfarsson

School of Computer Science, Reykjavík University

October 19, 2011

Table of Contents

Permutations

A permutation is a bijection $\pi\colon \{1,\dots,n\} \to \{1,\dots,n\}$ for some n.
We use one-line notation for permutations.

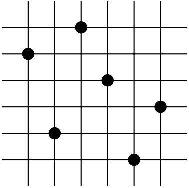
$$\pi = 526413$$

is the permutation that sends

$$\begin{aligned} 1 &\mapsto 5 \\ 2 &\mapsto 2 \\ 3 &\mapsto 6 \\ 4 &\mapsto 4 \\ 5 &\mapsto 1 \\ 6 &\mapsto 3 \end{aligned}$$

Drawing permutations

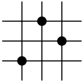
We can draw the graph of a permutation by placing dots on a grid.

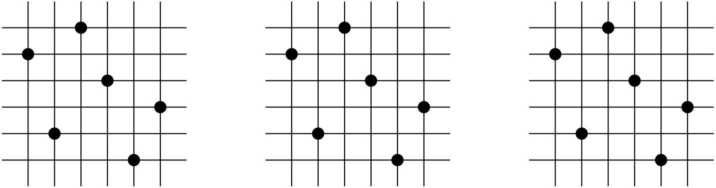
$526413 =$ 

Classical patterns

Patterns are permutations inside other permutations . . .

Example

The pattern  occurs in the permutation 526413.

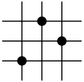


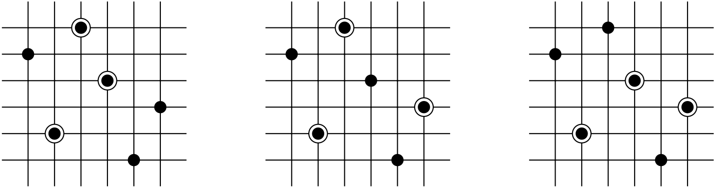
The same permutation avoids the pattern .

Classical patterns

Patterns are permutations inside other permutations . . .

Example

The pattern  occurs in the permutation 526413.

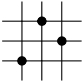


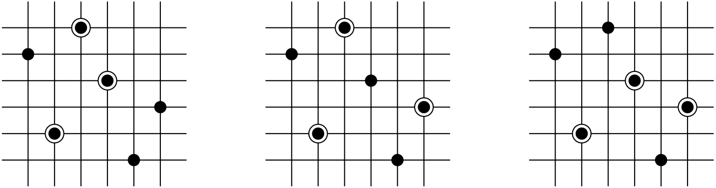
The same permutation avoids the pattern .

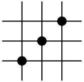
Classical patterns

Patterns are permutations inside other permutations . . .

Example

The pattern  occurs in the permutation 526413.



The same permutation avoids the pattern  .

Mesh patterns

Classical patterns form the base of a hierarchy of generalizations

mesh

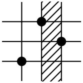
bivincular

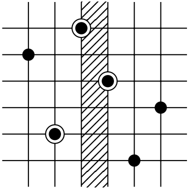
vincular

classical

Mesh patterns

Vincular patterns (2000) may require positions to be adjacent

The pattern  occurs in the permutation 526413



This is an occurrence

mesh

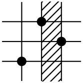
bivincular

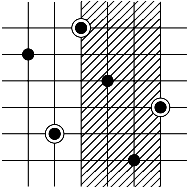
vincular

classical

Mesh patterns

Vincular patterns (2000) may require positions to be adjacent

The pattern  occurs in the permutation 526413



This is not an occurrence

mesh

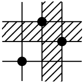
bivincular

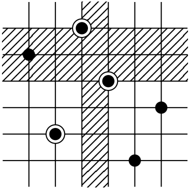
vincular

classical

Mesh patterns

Bivincular patterns (2009) may also require values to be adjacent

The pattern  is avoided by the permutation 526413



This is not an occurrence

mesh

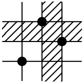
bivincular

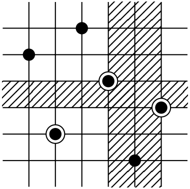
vincular

classical

Mesh patterns

Bivincular patterns (2009) may also require values to be adjacent

The pattern  is avoided by the permutation 526413



Neither is this one

mesh

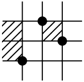
bivincular

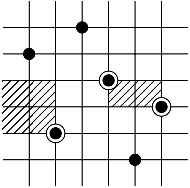
vincular

classical

Mesh patterns

Mesh patterns (2010) may forbid certain boxes from containing elements

The pattern  occurs in the permutation 526413



This is an occurrence

mesh

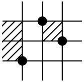
bivincular

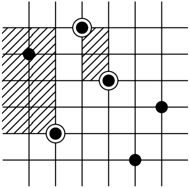
vincular

classical

Mesh patterns

Mesh patterns (2010) may forbid certain boxes from containing elements

The pattern  occurs in the permutation 526413



This is not an occurrence

mesh

bivincular

vincular

classical

Simsun permutations

A permutation is simsun if any restriction of it to $\{1, \ldots, k\} \subseteq \{1, \ldots, n\}$ has no double descents.

Example

The permutation 4536712 is not simsun: If we restrict to $\{1, \ldots, 5\}$ we have $4\underline{531}2$.

A permutation is simsun if and only if it avoids

Simsun permutations

A permutation is simsun if any restriction of it to $\{1,\ldots,k\} \subseteq \{1,\ldots,n\}$ has no double descents.

Example

The permutation 4536712 is not simsun: If we restrict to $\{1,\ldots,5\}$ we have $4\underline{531}2$.

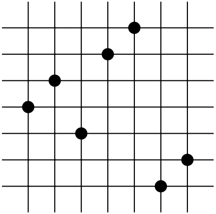
A permutation is simsun if and only if it avoids

Simsun permutations

A permutation is simsun if any restriction of it to $\{1, \ldots, k\} \subseteq \{1, \ldots, n\}$ has no double descents.

Example

The permutation 4536712 is not simsun: If we restrict to $\{1, \ldots, 5\}$ we have 45312.



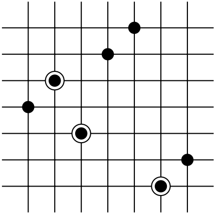
A permutation is simsun if and only if it avoids

Simsun permutations

A permutation is simsun if any restriction of it to $\{1, \ldots, k\} \subseteq \{1, \ldots, n\}$ has no double descents.

Example

The permutation 4536712 is not simsun: If we restrict to $\{1, \ldots, 5\}$ we have $4\underline{531}2$.



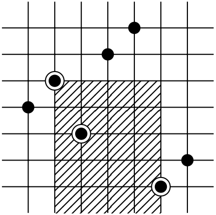
A permutation is simsun if and only if it avoids

Simsun permutations

A permutation is simsun if any restriction of it to $\{1, \dots, k\} \subseteq \{1, \dots, n\}$ has no double descents.

Example

The permutation 4536712 is not simsun: If we restrict to $\{1, \dots, 5\}$ we have $4\underline{531}2$.



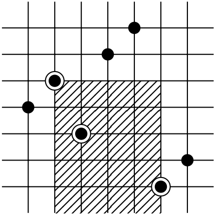
A permutation is simsun if and only if it avoids

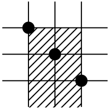
Simsun permutations

A permutation is simsun if any restriction of it to $\{1, \ldots, k\} \subseteq \{1, \ldots, n\}$ has no double descents.

Example

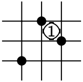
The permutation 4536712 is not simsun: If we restrict to $\{1, \ldots, 5\}$ we have 4$\underline{531}$2.



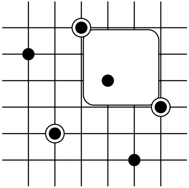
A permutation is simsun if and only if it avoids 

Marked mesh patterns

Marked mesh pattern (2010) restrict the number of elements in a region

The pattern 

occurs in the permutation 526413



This is an occurrence

marked mesh

mesh

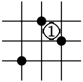
bivincular

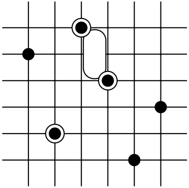
vincular

classical

Marked mesh patterns

Marked mesh pattern (2010) restrict the number of elements in a region

The pattern  occurs in the permutation 526413



This is not an occurrence

marked mesh

mesh

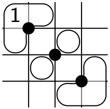
bivincular

vincular

classical

Freely braided permutations

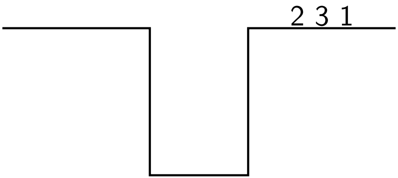
Green and Losonczy, 2002, defined freely braided permutations as those permutations avoiding the classical patterns 3421, 4231, 4312, 4321. Equivalently, these are the permutations avoiding



We now look at sorting and introduce a new type of pattern

Sorting with one stack

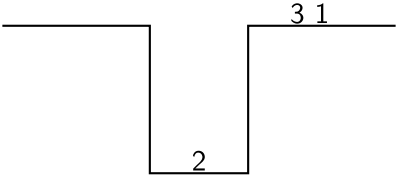
If we try to sort the permutation 231 with one stack . . .



. . . we fail. So 231 is not sortable in one pass.

Sorting with one stack

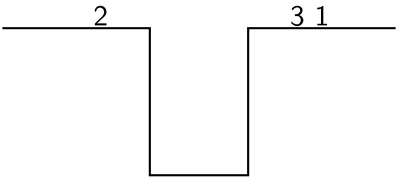
If we try to sort the permutation 231 with one stack . . .



. . . we fail. So 231 is not sortable in one pass.

Sorting with one stack

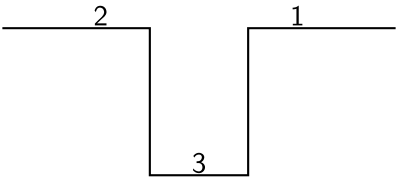
If we try to sort the permutation 231 with one stack . . .



. . . we fail. So 231 is not sortable in one pass.

Sorting with one stack

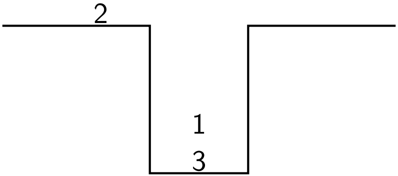
If we try to sort the permutation 231 with one stack ...



... we fail. So 231 is not sortable in one pass.

Sorting with one stack

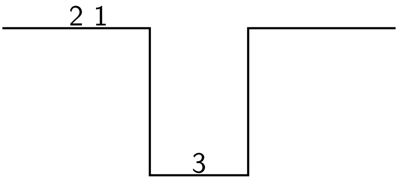
If we try to sort the permutation 231 with one stack . . .



. . . we fail. So 231 is not sortable in one pass.

Sorting with one stack

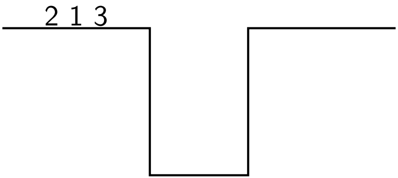
If we try to sort the permutation 231 with one stack ...



... we fail. So 231 is not sortable in one pass.

Sorting with one stack

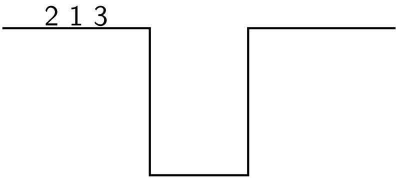
If we try to sort the permutation 231 with one stack . . .



. . . we fail. So 231 is not sortable in one pass.

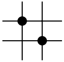
Sorting with one stack

If we try to sort the permutation 231 with one stack . . .



. . . we fail. So 231 is not sortable in one pass.

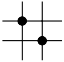
Characterizing permutations sortable in one pass

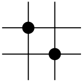
We failed because the outcome contained the pattern . The points making up that pattern must have come from a pattern

in the original permutation . . . and something pushed the big element out of the stack before the small element could get on top and out of the stack

This result is due to Knuth (1968): A permutation is sortable in one pass if and only if it avoids 231.

Characterizing permutations sortable in one pass

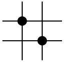
We failed because the outcome contained the pattern . The points making up that pattern must have come from a pattern

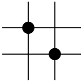


in the original permutation . . . and something pushed the big element out of the stack before the small element could get on top and out of the stack

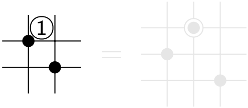
This result is due to Knuth (1968): A permutation is sortable in one pass if and only if it avoids 231.

Characterizing permutations sortable in one pass

We failed because the outcome contained the pattern . The points making up that pattern must have come from a pattern

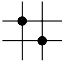


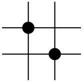
in the original permutation . . . and something pushed the big element out of the stack before the small element could get on top and out of the stack



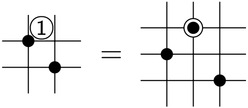
This result is due to Knuth (1968): A permutation is sortable in one pass if and only if it avoids 231.

Characterizing permutations sortable in one pass

We failed because the outcome contained the pattern . The points making up that pattern must have come from a pattern

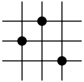


in the original permutation ... and something pushed the big element out of the stack before the small element could get on top and out of the stack



This result is due to Knuth (1968): A permutation is sortable in one pass if and only if it avoids 231.

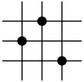
Characterizing permutations sortable in two passes

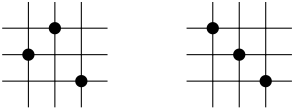
A permutation is not sorted in two passes if we pass it once through and the outcome contains the pattern . The points making up that pattern can come from either of the patterns

in the original permutation . . . and something pushed the big elements out of the stack before the small element could get on top and out of the stack

This reproves a theorem of West.

Characterizing permutations sortable in two passes

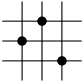
A permutation is not sorted in two passes if we pass it once through and the outcome contains the pattern . The points making up that pattern can come from either of the patterns

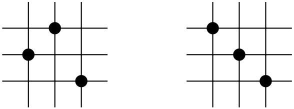


in the original permutation ... and something pushed the big elements out of the stack before the small element could get on top and out of the stack

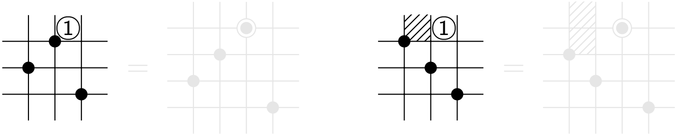
This reproves a theorem of West.

Characterizing permutations sortable in two passes

A permutation is not sorted in two passes if we pass it once through and the outcome contains the pattern  . The points making up that pattern can come from either of the patterns

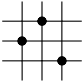


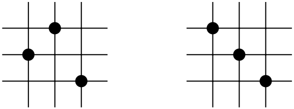
in the original permutation ... and something pushed the big elements out of the stack before the small element could get on top and out of the stack



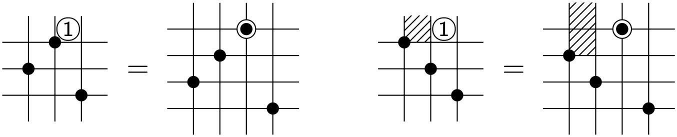
This reproves a theorem of West.

Characterizing permutations sortable in two passes

A permutation is not sorted in two passes if we pass it once through and the outcome contains the pattern . The points making up that pattern can come from either of the patterns

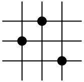


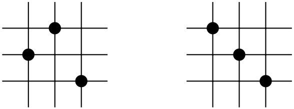
in the original permutation . . . and something pushed the big elements out of the stack before the small element could get on top and out of the stack



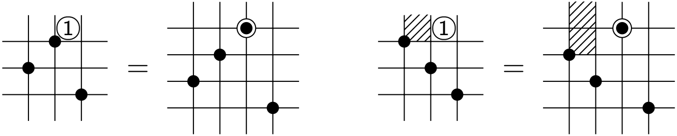
This reproves a theorem of West.

Characterizing permutations sortable in two passes

A permutation is not sorted in two passes if we pass it once through and the outcome contains the pattern . The points making up that pattern can come from either of the patterns



in the original permutation . . . and something pushed the big elements out of the stack before the small element could get on top and out of the stack

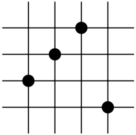


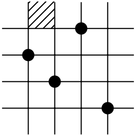
This reproves a theorem of West.

Characterizing permutations sortable in two passes

Theorem (West 1990)

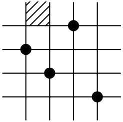
A permutation is sorted in two passes if and only if it avoids





Characterizing permutations sortable in three passes

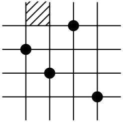
A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern



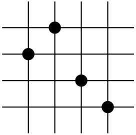
The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern

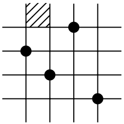


The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

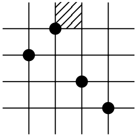


Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern

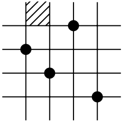


The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

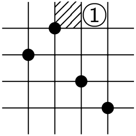


Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern

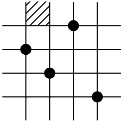


The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

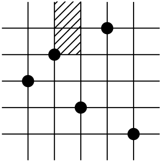


Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern

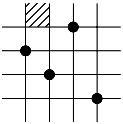


The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

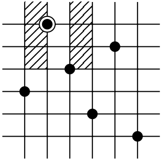


Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern

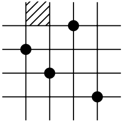


The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

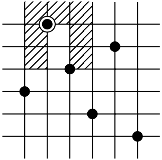


Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern

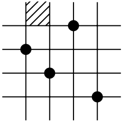


The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

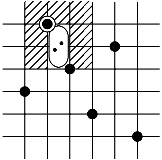


Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern

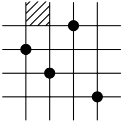


The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is

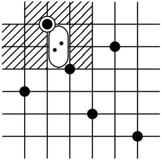


Characterizing permutations sortable in three passes

A permutation is not sorted in three passes if we pass it once through a stack and the outcome contains either of the patterns on the last slide. Say it is the pattern



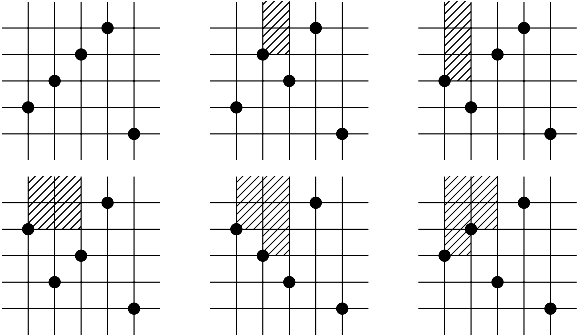
The points making up the underlying 3241 pattern can come from many patterns in the original permutation. One example is



Characterizing permutations sortable in three passes

Theorem (Ú. 2011)

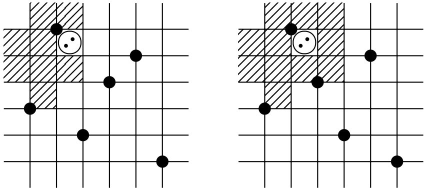
A permutation π is sortable in three passes if and only if it avoids the following decorated patterns



Characterizing permutations sortable in three passes

Theorem (Ú. 2011)

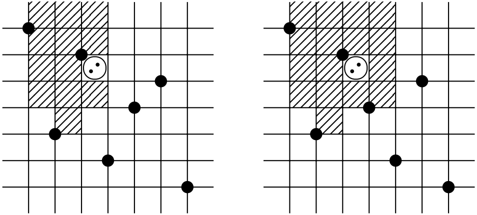
A permutation π is sortable in three passes if and only if it avoids the following decorated patterns



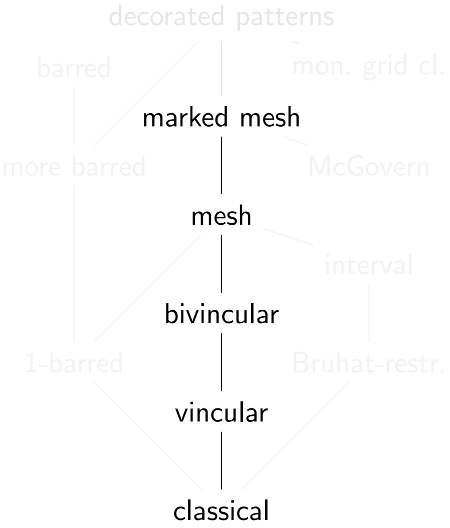
Characterizing permutations sortable in three passes

Theorem (Ú. 2011)

A permutation π is sortable in three passes if and only if it avoids the following decorated patterns

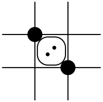


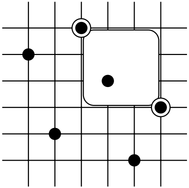
Decorated patterns



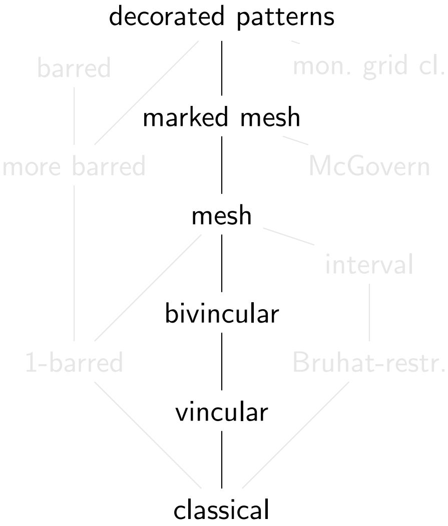
Decorated patterns

Decorated patterns (2011) allow even finer control over the elements in a region

The pattern  occurs in the permutation 526413.

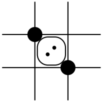


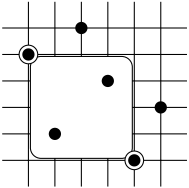
This is an occurrence



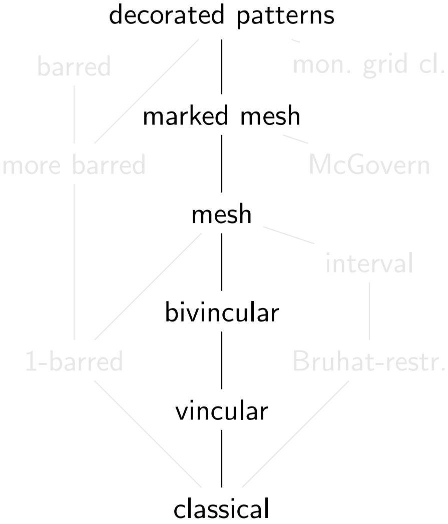
Decorated patterns

Decorated patterns (2011) allow even finer control over the elements in a region

The pattern  occurs in the permutation 526413.

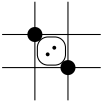


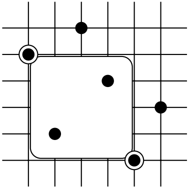
This is not an occurrence



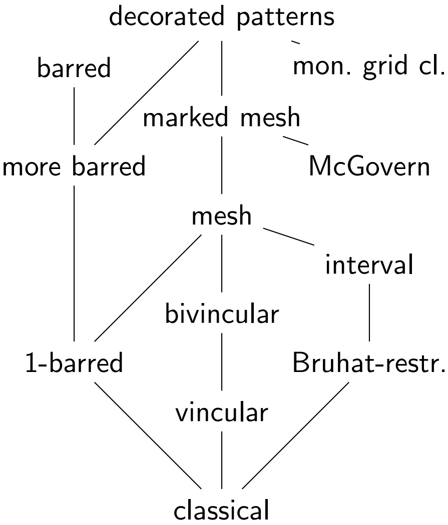
Decorated patterns

Decorated patterns (2011) allow even finer control over the elements in a region

The pattern  occurs in the permutation 526413.



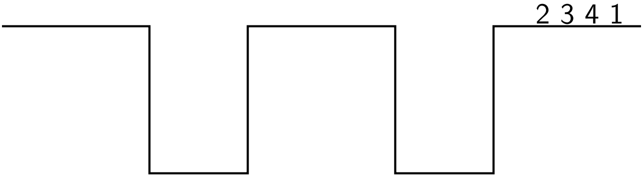
This is not an occurrence



We end with some open problems.

Sorting with two stacks

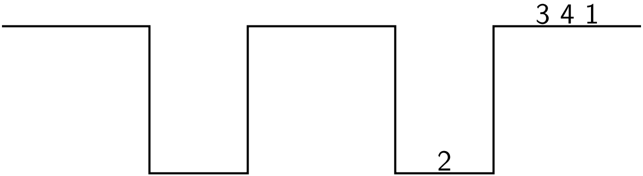
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all.
We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

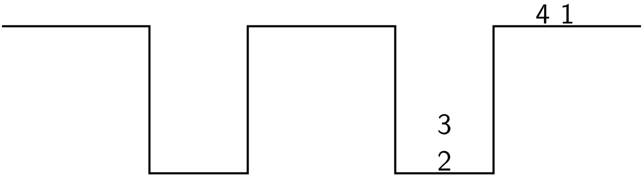
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all.
We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

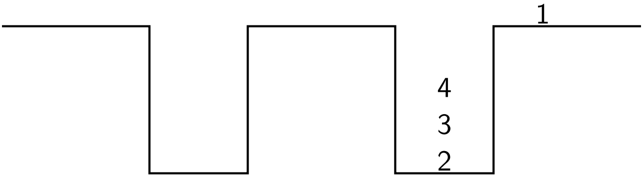
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all.
We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

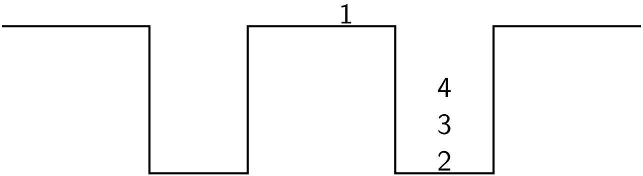
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all.
We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

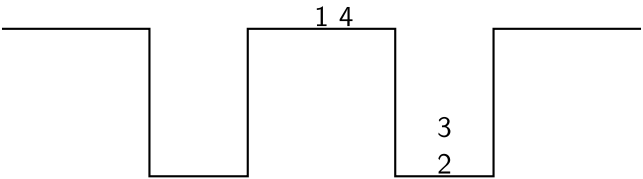
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all. We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

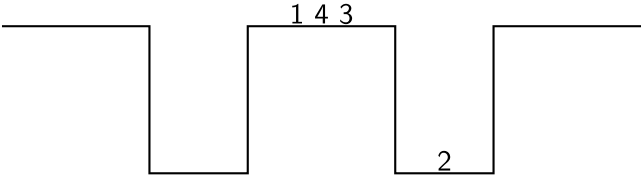
We can sort the permutation 2341 with two stacks ...



... by being smart. This is possible for some permutations – not for all. We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

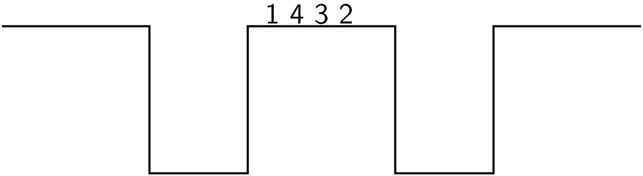
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all. We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

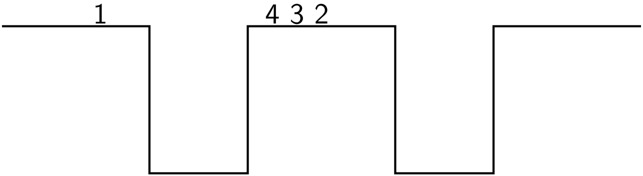
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all.
We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

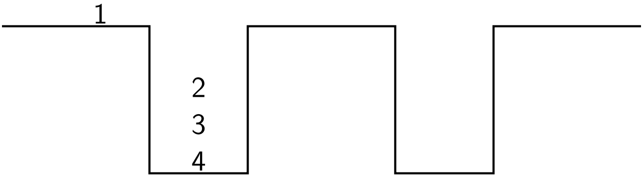
We can sort the permutation 2341 with two stacks ...



... by being smart. This is possible for some permutations – not for all. We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

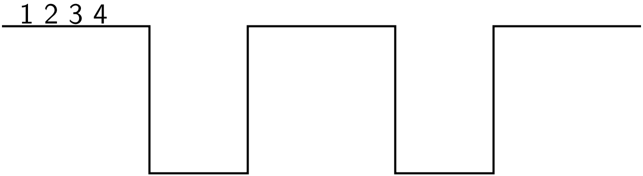
We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all.
We don't know how to characterize them by pattern avoidance.

Sorting with two stacks

We can sort the permutation 2341 with two stacks . . .



. . . by being smart. This is possible for some permutations – not for all. We don't know how to characterize them by pattern avoidance.

Open Problem 1

Characterize permutations sortable with two stacks: wide-open and considered very hard.

Open Problem 2

Count permutations sortable with three passes through a single stack: wide-open and considered very hard. Maybe easier now because we have the patterns describing these permutations.

Open Problem 3

Can a computer do what we did? Can it figure out the patterns describing permutations sortable in four passes through a single stack?

Open Problem 4

Applications to other sorting operations. The method used above can easily be extended for the bubble-sort operator. How about others?

For more information

Describing West-3-stack-sortable permutations with permutation patterns
http://arxiv.org/abs/1110.1219

Thank you for listening!
Any questions?