Finding small patterns in permutations in linear time

Sylvain Guillemot   Dániel Marx

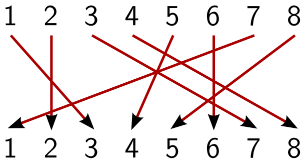
Institute for Computer Science and Control
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

SODA 2014
January 5, 2014
Portland, OR

Permutations

Different interpretations:

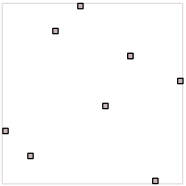
- Bijective mapping $\sigma : [n] \to [n]$:



- Ordering of $[n]$:

3 2 7 8 4 6 1 5

- n points in the plane "in general position":



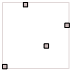
Subpermutations

There is a natural way of defining the meaning of σ being a subpermutation of π (or a "permutation pattern" in π).

Example:

σ

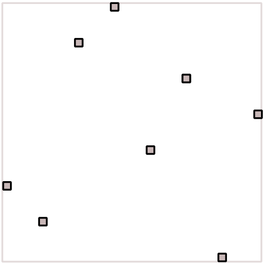
1 4 2 3



is a subpermutation of

π

3 2 7 8 4 6 1 5



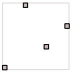
Subpermutations

There is a natural way of defining the meaning of σ being a subpermutation of π (or a "permutation pattern" in π).

Example:

σ

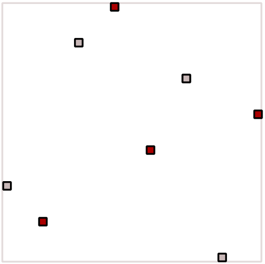
1 4 2 3



is a subpermutation of

π

3 2 7 8 4 6 1 5



Excluding subpermutations

There are $n!$ permutations of length n, but their number is much smaller if we exclude a fixed permutation:

Theorem [Marcus and Tardos 2004]

For every fixed permutation σ, the number of permutations of length n avoiding σ is at most c^n for some constant c depending on σ.

Example:

The number of permutations of length n avoiding 231 is exactly the Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n} < 4^n$.

Finding patterns in permutations

> PERMUTATION PATTERN
> **Input:** Two permutations σ and π.
> **Decide:** Is σ a subpattern of π?

- NP-hard in general [Bose, Buss, and Lubiw 1998].
- Can be solved in time $n^{\ell+O(1)}$ by brute force, where $\ell = |\sigma|$ and $n = |\pi|$.
- Can be solved in time $n^{0.47\ell+o(\ell)}$ [Ahal and Rabinovich 2008].

Finding patterns in permutations

PERMUTATION PATTERN
Input: Two permutations σ and π.
Decide: Is σ a subpattern of π?

- NP-hard in general [Bose, Buss, and Lubiw 1998].
- Can be solved in time $n^{\ell+O(1)}$ by brute force, where $\ell = |\sigma|$ and $n = |\pi|$.
- Can be solved in time $n^{0.47\ell+o(\ell)}$ [Ahal and Rabinovich 2008].

Main result:

Theorem

PERMUTATION PATTERN can be solved in time $2^{O(\ell^2 \log \ell)} \cdot n$, where $\ell = |\sigma|$ and $n = |\pi|$.

Decompositions

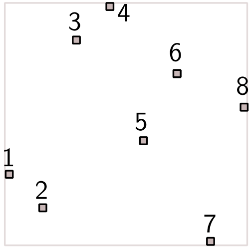
We define the notion of d-**wide decomposition** and solve the problem using the following win/win strategy:
($\ell = |\sigma|$, $n = |\pi|$)

1. There is an algorithm that either
   - finds σ in π or
   - finds a $2^{O(\ell \log \ell)}$-wide decomposition of π.
2. There is an algorithm that, given σ and a d-wide decomposition of π, decides if σ is a subpattern of π in time $(d\ell)^{O(\ell)} \cdot n$.

These two algorithms together give a $2^{O(\ell^2 \log \ell)} \cdot n$ time algorithm for Permutation Pattern.

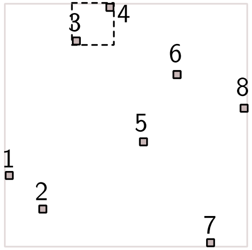
Decompositions

Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



Decompositions

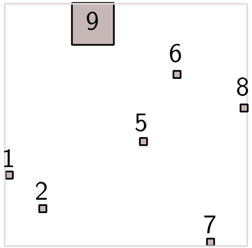
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3, 4)$

Decompositions

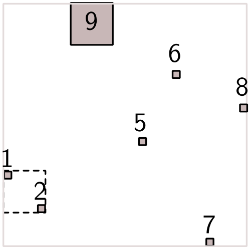
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3, 4) \rightarrow 9$

Decompositions

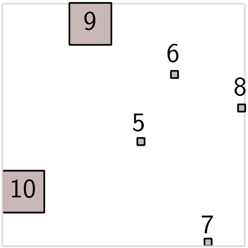
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3, 4) \rightarrow 9$
$(1, 2)$

Decompositions

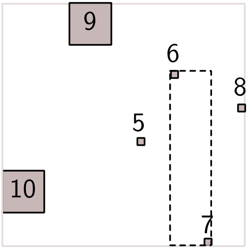
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \to 9$
$(1,2) \to 10$

Decompositions

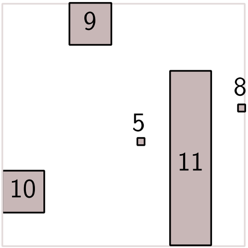
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \to 9$
$(1,2) \to 10$
$(6,7)$

Decompositions

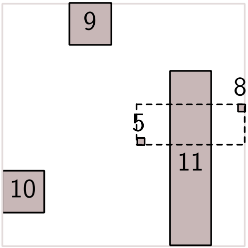
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3, 4) \rightarrow 9$
$(1, 2) \rightarrow 10$
$(6, 7) \rightarrow 11$

Decompositions

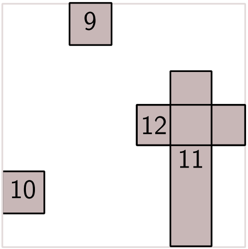
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3, 4) \rightarrow 9$
$(1, 2) \rightarrow 10$
$(6, 7) \rightarrow 11$
$(5, 8)$

Decompositions

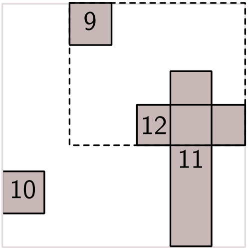
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \to 9$
$(1,2) \to 10$
$(6,7) \to 11$
$(5,8) \to 12$

Decompositions

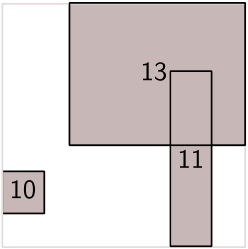
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \to 9$ $\quad (9,12)$
$(1,2) \to 10$
$(6,7) \to 11$
$(5,8) \to 12$

Decompositions

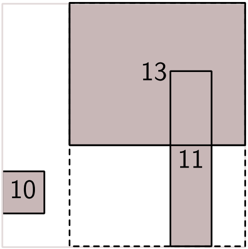
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \rightarrow 9$ $\quad (9,12) \rightarrow 13$
$(1,2) \rightarrow 10$
$(6,7) \rightarrow 11$
$(5,8) \rightarrow 12$

Decompositions

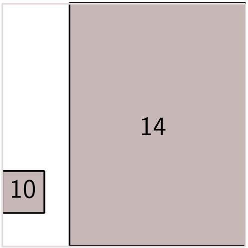
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \to 9$
$(1,2) \to 10$
$(6,7) \to 11$
$(5,8) \to 12$
$(9,12) \to 13$
$(13,11)$

Decompositions

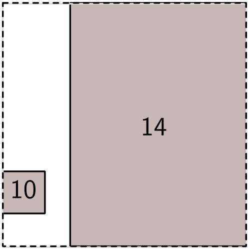
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \rightarrow 9$ $\quad (9,12) \rightarrow 13$
$(1,2) \rightarrow 10$ $\quad (13,11) \rightarrow 14$
$(6,7) \rightarrow 11$
$(5,8) \rightarrow 12$

Decompositions

Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \to 9$

$(1,2) \to 10$

$(6,7) \to 11$

$(5,8) \to 12$

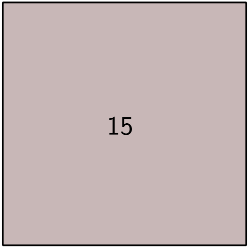
$(9,12) \to 13$

$(13,11) \to 14$

$(10,14)$

Decompositions

Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



$(3,4) \to 9$

$(1,2) \to 10$

$(6,7) \to 11$

$(5,8) \to 12$

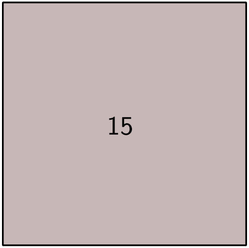
$(9,12) \to 13$

$(13,11) \to 14$

$(10,14) \to 15$

Decompositions

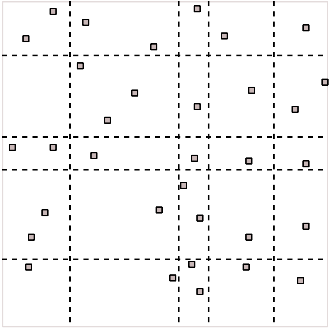
Starting with a set of points (degenerate rectangles), the decomposition is a sequence of merges, where a merge consists of replacing two rectangles with their bounding box:



- R_1 sees R_2 horizontally (resp., vertically) if there is a horizontal (resp., vertical) line intersecting both.
- A rectangle family is d-**wide** if every rectangle sees less than d other rectangles horizontally and less than d other rectangles vertically.
- The decomposition is d-**wide** if it is d-wide in every step.

Grids

$r \times r$-grid: partitioning the rows and the columns into r classes such that every cell contains a point.



Observation: If a point set has an $r \times r$-grid, then it contains every permutation of length r.

Fact

If π has an $r \times r$-grid, then every decomposition of π is $\Omega(r)$-wide.

Finding decompositions

Large grids imply large width:

Fact

If π contains an $r \times r$-grid, then every decomposition of π is $\Omega(r)$-wide.

Large width implies large grids:

Theorem

There is an $O(n)$ time algorithm that finds either an $r \times r$ grid in π or a $2^{O(r \log r)}$-wide decomposition of π.

Finding decompositions

Large grids imply large width:

Fact

If π contains an $r \times r$-grid, then every decomposition of π is $\Omega(r)$-wide.

Large width implies large grids:

Theorem

There is an $O(n)$ time algorithm that finds either an $r \times r$ grid in π or a $2^{O(r \log r)}$-wide decomposition of π.

The algorithm relies on the following previous result:

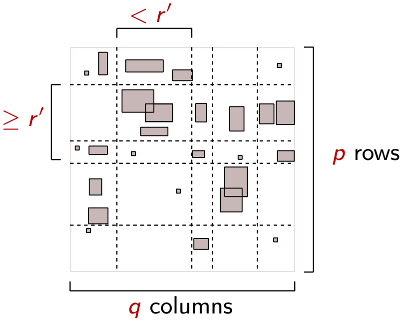
Theorem (essentially [Marcus and Tardos 2004])

If M is a point set in $[p] \times [q]$ with $|M| > r^4 \binom{r^2}{r}(p+q)$, then we can find an $r \times r$-grid in M.

Finding decompositions

We maintain a partition of rows and columns that is compatible with every current rectangle and satisfies that

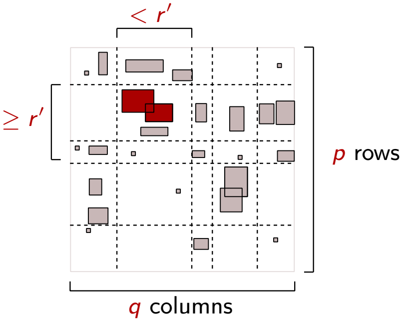
- every row/column contains $< r'$ rectangles and
- every two adjacent rows/columns contain $\geq r'$ rectangles.



Finding decompositions

We maintain a partition of rows and columns that is compatible with every current rectangle and satisfies that

- every row/column contains $< r'$ rectangles and
- every two adjacent rows/columns contain $\geq r'$ rectangles.

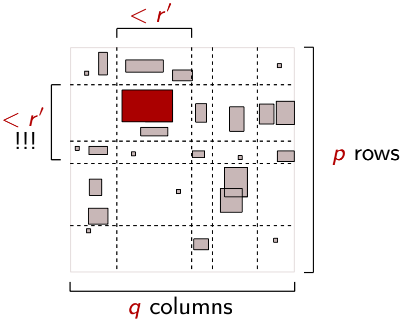


Case 1: If a cell contains two rectangles, merge them. If two adjacent rows/columns contain $< r'$ rectangles, then merge them.

Finding decompositions

We maintain a partition of rows and columns that is compatible with every current rectangle and satisfies that

- every row/column contains $< r'$ rectangles and
- every two adjacent rows/columns contain $\geq r'$ rectangles.

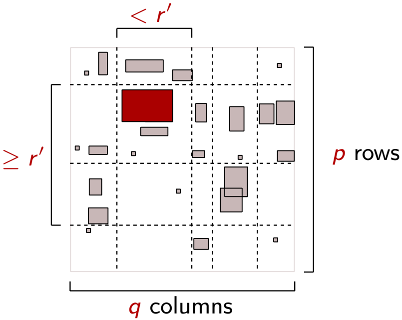


Case 1: If a cell contains two rectangles, merge them. If two adjacent rows/columns contain $< r'$ rectangles, then merge them.

Finding decompositions

We maintain a partition of rows and columns that is compatible with every current rectangle and satisfies that

- every row/column contains $< r'$ rectangles and
- every two adjacent rows/columns contain $\geq r'$ rectangles.

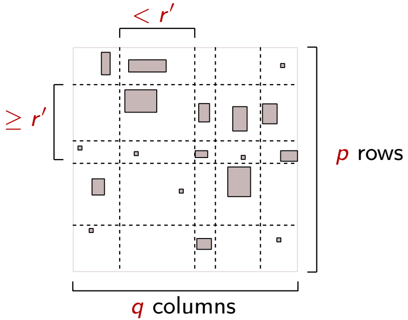


Case 1: If a cell contains two rectangles, merge them. If two adjacent rows/columns contain $< r'$ rectangles, then merge them.

Finding decompositions

We maintain a partition of rows and columns that is compatible with every current rectangle and satisfies that

- every row/column contains $< r'$ rectangles and
- every two adjacent rows/columns contain $\geq r'$ rectangles.



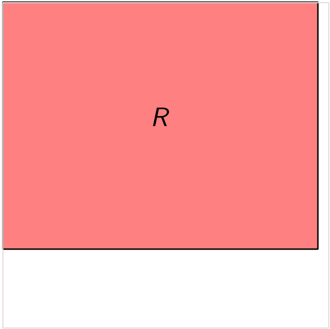
Case 2: Every cell contains at most one rectangle
$\Rightarrow$ There are $\Omega((p+q)r')$ nonempty cells
$\Rightarrow$ Result of [Marcus and Tardos 2004] implies that there is an $r \times r$-grid (if r' is sufficiently large).

Finding patterns

We would like to design a dynamic programming algorithm to solve PERMUTATION PATTERN using a given decomposition.

Problem:

The decomposition does not break the problem into independent subproblems.

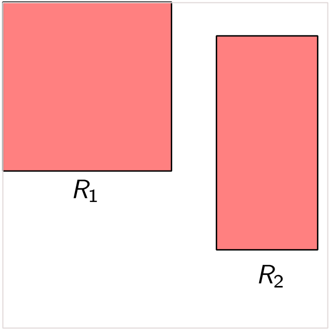


Finding patterns

We would like to design a dynamic programming algorithm to solve PERMUTATION PATTERN using a given decomposition.

Problem:

The decomposition does not break the problem into independent subproblems.

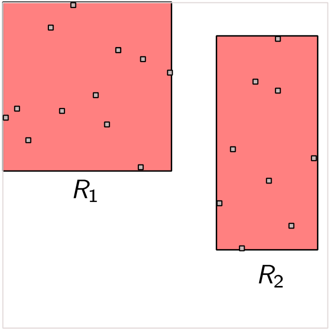


Finding patterns

We would like to design a dynamic programming algorithm to solve PERMUTATION PATTERN using a given decomposition.

Problem:

The decomposition does not break the problem into independent subproblems.



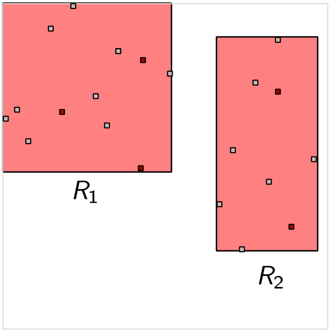
The interaction between the points in R_1 and R_2 can be arbitrary!

Finding patterns

We would like to design a dynamic programming algorithm to solve PERMUTATION PATTERN using a given decomposition.

Problem:

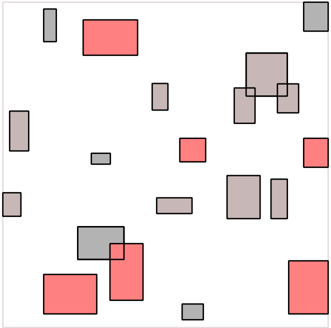
The decomposition does not break the problem into independent subproblems.



The interaction between the points in R_1 and R_2 can be arbitrary!

Connected components

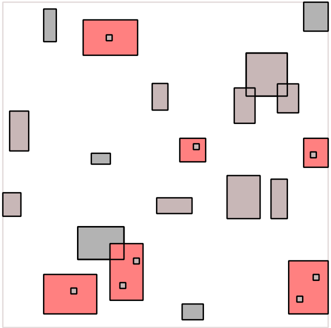
Visibility graph: two rectangles are adjacent if they see each other horizontally or vertically.



Observation: The degree of the visibility graph is less than $2d$ if the rectangle family is d-wide. Therefore, there are $d^{O(\ell)} \cdot n$ sets K of size $\leq \ell$ that are connected in the visibility graph.

Connected components

Visibility graph: two rectangles are adjacent if they see each other horizontally or vertically.

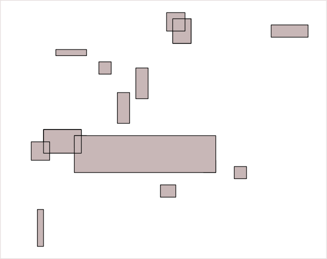


Subproblems defined by

- step i of the decomposition,
- a connected set K of size $\leq \ell$ in the visibility graph,
- a subpermutation σ' if σ, and
- a mapping of σ' into the rectangles of K.

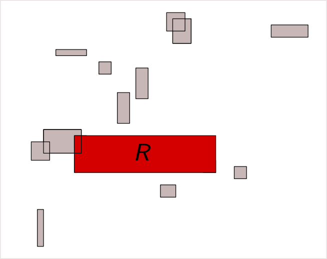
Connected components

How to solve a subproblem at step i using the subproblems at step $i-1$?



Connected components

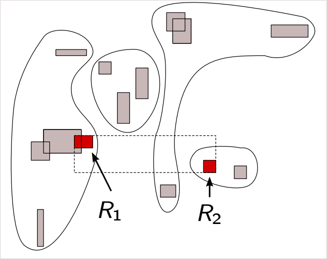
How to solve a subproblem at step i using the subproblems at step $i-1$?



- If a rectangle R of K was created at step i by merging R_1 and R_2, then we have to distribute the points assigned to R in every possible way between R_1 and R_2.
- In step $i-1$, set K falls apart into some number of connected sets, but the interaction between them is completely understood.

Connected components

How to solve a subproblem at step i using the subproblems at step $i-1$?



- If a rectangle R of K was created at step i by merging R_1 and R_2, then we have to distribute the points assigned to R in every possible way between R_1 and R_2.
- In step $i-1$, set K falls apart into some number of connected sets, but the interaction between them is completely understood.

Algorithm summary

Theorem

PERMUTATION PATTERN can be solved in time $2^{O(\ell^2 \log \ell)} \cdot n$, where $\ell = |\sigma|$ and $n = |\pi|$.

Win/win strategy:

1. There is an algorithm that either
   - finds σ in π or
   - finds a $2^{O(\ell \log \ell)}$-wide decomposition of π.
2. There is an algorithm that, given σ and a d-wide decomposition of π, decides if σ is a subpattern of π in time $(d\ell)^{O(\ell)} \cdot n$.

Hardness

PARTITIONED PERMUTATION PATTERN: for every $i \in \sigma$, a subset $S_i \subseteq \pi$ is given where it can be mapped.

Theorem

PARTITIONED PERMUTATION PATTERN is W[1]-hard parameterized by $|\sigma|$.

3-DIMENSIONAL PERMUTATION PATTERN: natural generalization to 3-dimensional points in general position.

Theorem

3-DIMENSIONAL PERMUTATION PATTERN is W[1]-hard parameterized by $|\sigma|$.

Conclusions

- Finding patterns in permutations is fixed-parameter tractable.
- Algorithm is based on a novel width measure for permutations.
- Win/win situation similar to certain algorithms based on minors and bidimensionality.
- But our decomposition is strictly speaking not a decomposition.