# Fast Algorithms for Permutation Pattern Detection

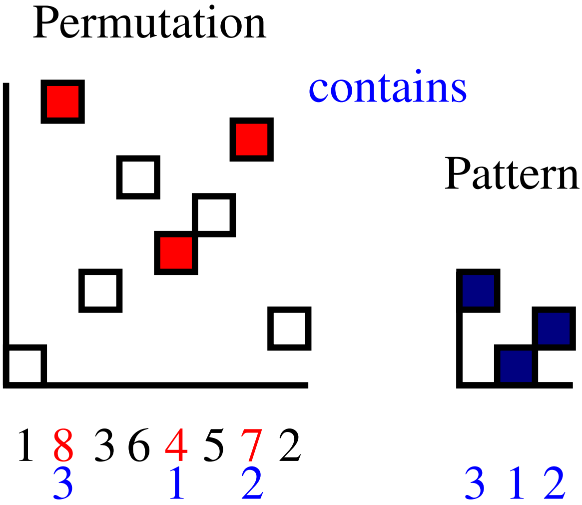
William Kuszmaul

Stanford University

Duluth REU
Supervised by Joe Gallian

June 26, 2017

# Patterns with no Adjacency Constraints



# Classic Research Problem

Pick small set of patterns $\Pi$.
**The Question:**

$$\text{How many permutations of size } n \text{ avoid all } \pi \in \Pi?$$

**Notation:** Set of such permutations is called $\mathrm{AV}_n(\Pi)$.
For single patterns $\pi$, we say $\mathrm{AV}_n(\pi)$ instead of $\mathrm{AV}_n(\{\pi\})$.

**Example:**

$$\Pi = \{2413, 3142\}.$$
$$|\mathrm{AV}_n(\{2413, 3142\})| \text{ is the } n\text{-th Schröder number.}$$

# My Research: Can we treat pattern avoidance as an experimental science?

**Example Experiment:**
For each $\Pi \subseteq S_4$

1. Compute $|\mathrm{AV}_1(\Pi)|, \ldots, |\mathrm{AV}_{16}(\Pi)|$
2. Search for sequence in OEIS

**My Research:** Can we build fast and practical algorithms for permutation pattern avoidance?

# My Research: Can we treat pattern avoidance as an experimental science?

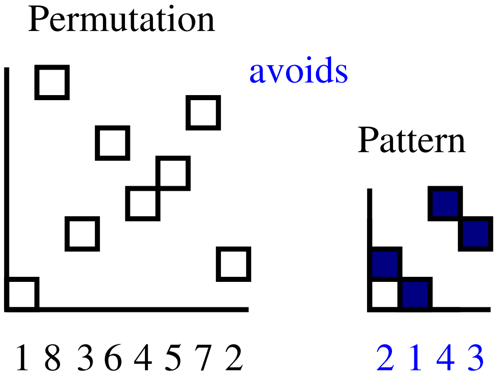
**Example Experiment:**

For each $\Pi \subseteq S_4$ $\longleftarrow$ **Over two million subsets**

1. Compute $|\mathrm{AV}_1(\Pi)|, \ldots, |\mathrm{AV}_{16}(\Pi)|$ $\longleftarrow$ **The hard part!**
2. Search for sequence in OEIS

**My Research:** Can we build fast and practical algorithms for permutation pattern avoidance?

# Detecting Patterns is NP-Hard (BBL '98)




Best Algorithms for permutation size $n$ and pattern size $k$:

- $O(1.79^n \cdot nk)$ time (Bruner, Lackner, 2012)
- $2^{O(k^2 \log k)} \cdot n$ time (Guillemot, Marx, 2014)

# My Idea: Amortize avoidance-detection cost

**The Insight:**

Can circumvent NP-hardness issue by asking

*which* permutations contain a pattern,
instead of
*if* a permutation contains a pattern.

**My Algorithm Can:**

- Do avoidance detection in linear time using information about smaller permutations.
- For a set of patterns $\Pi \subseteq S_k$, compute sequence

$$|\mathrm{AV}_1(\Pi)|, |\mathrm{AV}_2(\Pi)|, \ldots, |\mathrm{AV}_n(\Pi)|$$

  in $O(|\mathrm{AV}_{\leq n-1}(\Pi)| \cdot k)$ time and $O(n^k)$ space.
- Compute $A_{16}(\Pi)$ for every $\Pi \subseteq S_4$ on my laptop in 3 hrs and 15 min.

**Part 1:** An Experiment on Millions of Sets

Examining $|\mathrm{AV}_1(\Pi)|, \ldots, |\mathrm{AV}_{16}(\Pi)|$ for $\Pi \subseteq S_4$.

# OEIS analysis for $\Pi \subseteq S_4$ with $|\Pi| > 4$

| Sequences Ignored | OEIS Matches | Distinct Sequences |
|---|---|---|
| None | 1,412,002 | 1,386 |
| Constant ones | 585,999 | 1,096 |
| Polynomial of degree $\leq 3$ | 32,019 | 446 |
| Polynomial of degree $\leq 3$, or solvable using standard techniques, or already known | 289 | 32 |

# Some Interesting Sequences

1. **A228180 The number of single edges on the boundary of ordered trees with n edges.**
   **Generating function is $(x \cdot C + 2x^3 \cdot C^4)/(1-x)$ where $C$ is the generating function for the Catalan numbers.**
   Appears 11 times. Example match:
   $\{2413\ 4132\ 1432\ 1342\ 1324\}$

2. **A071721 $\frac{6n}{(n+1)(n+2)}\binom{2n}{n}$.**
   Appears 6 times. Example match:
   $\{2431\ 4132\ 1432\ 1342\ 1324\ 1423\}$

# Some Interesting Sequences

3. **A071717 Expansion of $(1 + x^2C)C^2$, where $C$ is the generating function for Catalan numbers.**
   Appears 7 times. Example match:
   {2431 3142 4132 1432 1342 1324 1423}

4. **A071726 Expansion of $(1 + x^3C)C$, where $C$ is the generating function for Catalan numbers.**
   Appears 6 times. Example match:
   {2431 2413 3142 4132 1432 1342 1324 1423}

5. **A071742 Expansion of $(1 + x^4C)C$, where $C$ is the generating function for Catalan numbers.**
   **(Now proven by Struct algorithm!)**
   Appears 3 times. Example match:
   {2431 2143 3142 4132 1432 1342 1324 1423 1243}

# Some Interesting Sequences

6. **A000778** $C(n) + C(n+1) - 1$**, where** $C(n)$ **is the** $n$**-th Catalan number.**
   Appears 24 times. Example match:
   {2431 3142 4132 1432 1342 1324}

7. **A109262 A Catalan transform of the Fibonacci numbers.**
   Appears 4 times. Example match:
   {2413 4132 1432 1342 1423}

8. **A119370 G.f. satisfies** $A(x) = 1 + xA(x)^2 + x^2(A(x)^2 - A(x))$**.**
   Appears 3 times. Example match:
   {2413 3142 1432 1342 1423}

# Some Interesting Sequences

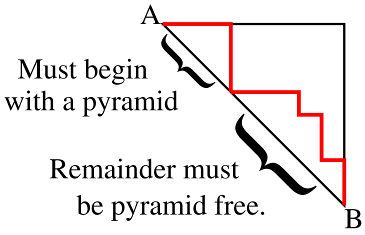
9. **A124671 Row sums of a triangle generated from Eulerian numbers.**
   **G.f. equals $x(1-3x+3x^2)/((1-2x)(x-1)^4)$.**
   Appears 4 times. Example match:
   {2341 2134 3412 3124 1342 1324 4123 1243}

10. **A035929 Number of $n \times n$ Catalan paths starting with an $m$-pyramid for some $m$, and followed by a pyramid free path.**
    Appears 14 times. Example match:
    {2143 3142 1432 1342 1324}

# What does A035929 count?

**Description:** Number of $n \times n$ Catalan paths starting with an $m$-pyramid for some $m$, and followed by a pyramid free path.

**Example for $n = 8$:**

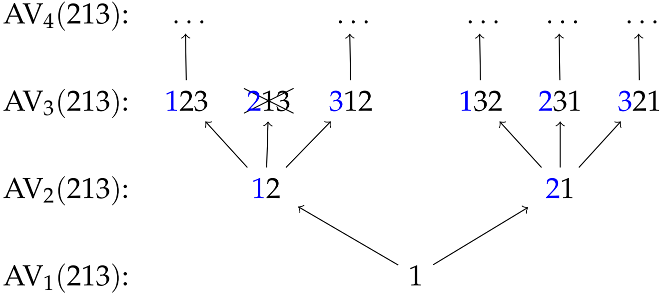



A Catalan path goes from A to B without ever going below the diagonal.

# **Part 2:** The Algorithm

Building $\mathrm{AV}_1(\pi), \ldots, \mathrm{AV}_n(\pi)$ for single pattern $\pi \in S_k$.

# Building $\mathrm{AV}_n(\pi)$ layer by layer




**Strategy:** Build each $\mathrm{AV}_n(\pi)$ out of $\mathrm{AV}_{n-1}(\pi)$.

**Runtime:** $O(|\mathrm{AV}_{\leq n-1}(\pi)| \cdot n \cdot \text{time to check single permutation})$

**The Problem:** Detecting patterns in a single perm is NP-hard!

# Pattern Detection by Induction

Does 25143 avoid 123?

# Pattern Detection by Induction

Does 25143 avoid 123?

| | Permutation in $S_4$ | Avoids 123? |
|---|---:|---:|
| Remove first letter: | 25143 | yes |
| | 4132 | |

# Pattern Detection by Induction

Does 25143 avoid 123?

| | Permutation in $S_4$ | Avoids 123? |
|---|---|---|
| Remove first letter: | 25143 | yes |
| | 4132 | |
| Remove second letter: | 25143 | yes |
| | 2 143 | |

# Pattern Detection by Induction

Does 25143 avoid 123?

| | Permutation in $S_4$ | Avoids 123? |
|---|---|---|
| Remove first letter: | 25143 | yes |
| | 4132 | |
| Remove second letter: | 25143 | yes |
| | 2 143 | |
| Remove third letter: | 25143 | yes |
| | 14 32 | |

# Pattern Detection by Induction

Does 25143 avoid 123?

| | Permutation in $S_4$ | Avoids 123? |
|---|---|---|
| Remove first letter: | 25143 | yes |
| | 4132 | |
| Remove second letter: | 25143 | yes |
| | 2 143 | |
| Remove third letter: | 25143 | yes |
| | 14 32 | |
| Remove fourth letter: | 25143 | yes |
| | 241 3 | |

# Pattern Detection by Induction

Does 25143 avoid 123?

| | Permutation in $S_4$ | Avoids 123? |
|---|---|---|
| Remove first letter: | 25143 | yes |
| | 4132 | |
| Remove second letter: | 25143 | yes |
| | 2 143 | |
| Remove third letter: | 25143 | yes |
| | 14 32 | |
| Remove fourth letter: | 25143 | yes |
| | 241 3 | |

All four tests pass $\longrightarrow$ 25143 avoids 123

# DETECTING PATTERN AVOIDANCE IN TIME $O(k)$.

Let $w$ be a permutation.

**Defn:** $w \downarrow_i$ is the reduction of $w$ without its $i$-th letter.

**Example:** $23514 \downarrow_2 = \mathrm{red}(2\ \ 514) = 2413$.

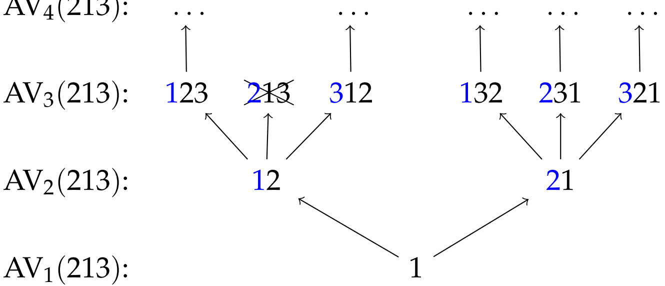
**Theorem:** If $w \downarrow_1,\ w \downarrow_2,\ \ldots, w \downarrow_{k+1}$ avoid $\pi$, then so does $w$.

Does $w$ avoid $\pi$?

| | Permutation in $S_{n-1}$ | Avoids $\pi$? |
|---|---|---|
| Remove 1-st letter: | $w \downarrow_1$ | yes |
| Remove 2-nd letter: | $w \downarrow_2$ | yes |
| $\vdots$ | $\vdots$ | $\vdots$ |
| Remove $(k+1)$-th letter: | $w \downarrow_{k+1}$ | yes |

All $k+1$ tests pass $\longrightarrow$ $w$ avoids $\pi$.

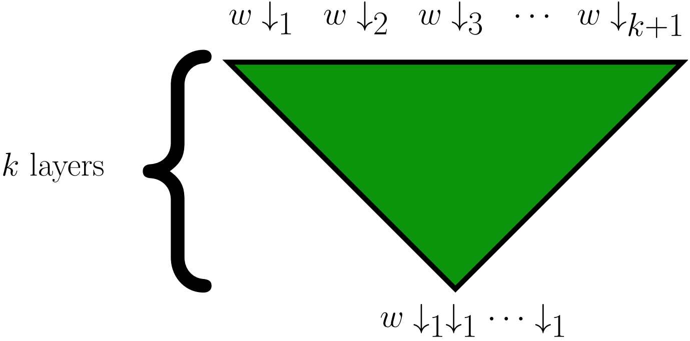
# A FAST ALGORITHM FOR BUILDING $AV_n(\pi)$




**Strategy:** Build each $AV_n(\pi)$ using information about $AV_{n-1}(\pi)$.
**Runtime:** $O(|AV_{\leq n-1}(\pi)| \cdot n \cdot k)$

**The New Problem:** Storing all of $AV_{n-1}(\pi)$ is impractical.

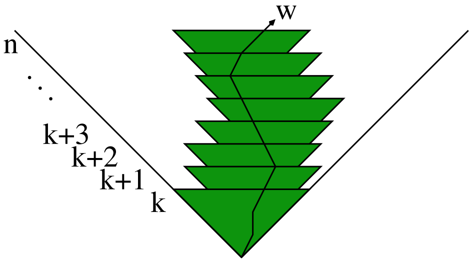
# How much do we actually have to store?




**Observation:** $w$ and $w\downarrow_1, \ldots, w\downarrow_{k+1}$ are order-isomorphic in their final $n-k-1$ letters.

**Algorithmic Consequence:** Can detect whether $w$ contains $\pi$ using only the subtree rooted at $w\downarrow_1\downarrow_1\cdots\downarrow_1$.

# SPACE-EFFICIENT COMPUTATION OF $|A_n(\pi)|$




**The Idea:** Instead of visiting avoiders in BFS order, visit avoiders in DFS of $k$-level BFS's.
**Space Usage:** $O(n \cdot \text{Max size of } k\text{-level BFS}) = O(n^{k+1})$.

# Thanks for listening!

**Link to Paper:** (Published in *Mathematics of Computation*)
`arxiv.org/abs/1509.08216`

**Link to Software and Data:**
`github.com/williamkuszmaul/patternavoidance`

**Contact Information:**
`kuszmaul@cs.stanford.edu`