Fast Algorithms for Permutation Pattern Detection

William Kuszmaul

Stanford University

Duluth REU
Supervised by Joe Gallian

June 26, 2017
Patterns with no Adjacency Constraints

Permutation contains Pattern

1 3 6 2
1 2 5

contains

1 2 3 3

Pattern

3 1 2
**Classic Research Problem**

Pick small set of patterns $\Pi$.

**The Question:**

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

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

**Example:**

$$\Pi = \{2413, 3142\}.$$  
$|AV_n(\{2413, 3142\})|$ is the $n$-th Schröder number.
**My Research: Can we treat pattern avoidance as an experimental science?**

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

1. Compute $|AV_1(\Pi)|, \ldots, |AV_{16}(\Pi)|$
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 $\Pi \subseteq S_4$ \[\text{Over two million subsets}\]

1. Compute $|AV_1(\Pi)|, \ldots, |AV_{16}(\Pi)|$ \[\text{The hard part!}\]
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)

Permutation

avoids

Pattern

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

\textit{which} permutations contain a pattern,

instead of

\textit{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

$$|AV_1(\Pi)|, |AV_2(\Pi)|, \ldots, |AV_n(\Pi)|$$

in $O(|AV_{\leq n-1}(\Pi)| \cdot k)$ 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 $|AV_1(\Pi)|, \ldots, |AV_{16}(\Pi)|$ for $\Pi \subseteq S_4$. 
### OEIS Analysis for $\Pi \subseteq S_4$ with $|\Pi| > 4$

<table>
<thead>
<tr>
<th>Sequences Ignored</th>
<th>OEIS Matches</th>
<th>Distinct Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>1,412,002</td>
<td>1,386</td>
</tr>
<tr>
<td>Constant ones</td>
<td>585,999</td>
<td>1,096</td>
</tr>
<tr>
<td>Polynomial of degree $\leq 3$</td>
<td>32,019</td>
<td>446</td>
</tr>
<tr>
<td>Polynomial of degree $\leq 3$, or solvable using standard techniques, or already known</td>
<td>289</td>
<td>32</td>
</tr>
</tbody>
</table>
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\}
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.

A Catalan path must begin with a pyramid.

Remainder must be pyramid free.
Part 2: The Algorithm

Building $AV_1(\pi), \ldots, AV_n(\pi)$ for single pattern $\pi \in S_k$. 
**Building AV\(_n(\pi)\) Layer by Layer**

AV\(_4(213)\): \[\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \]

AV\(_3(213)\): \[\begin{array}{c}
123 \\
\uparrow \\
213 \\
\uparrow \\
312 \\
\uparrow \\
132 \\
\uparrow \\
231 \\
\uparrow \\
321 \\
\end{array}\]

AV\(_2(213)\):
\[\begin{array}{c}
12 \\
\uparrow \\
21 \\
\end{array}\]

AV\(_1(213)\):
\[\begin{array}{c}
1 \\
\end{array}\]

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

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

**The Problem:** Detecting patterns in a single perm is NP-hard!
Does 25143 avoid 123?
## Pattern Detection by Induction

Does 25143 avoid 123?

<table>
<thead>
<tr>
<th>Remove first letter:</th>
<th>Permutation in $S_4$</th>
<th>Avoids 123?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>4132</td>
<td></td>
</tr>
</tbody>
</table>
**Pattern Detection by Induction**

Does 25143 avoid 123?

<table>
<thead>
<tr>
<th>Remove first letter:</th>
<th>Permutation in $S_4$</th>
<th>Avoids 123?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>4132</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Remove second letter:</th>
<th>Permutation in $S_4$</th>
<th>Avoids 123?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>2 143</td>
<td></td>
</tr>
</tbody>
</table>
**Pattern Detection by Induction**

Does 25143 avoid 123?

<table>
<thead>
<tr>
<th>Remove first letter:</th>
<th>Permutation in $S_4$</th>
<th>Avoids 123?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>4132</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Remove second letter:</th>
<th>Permutation in $S_4$</th>
<th>Avoids 123?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>2 143</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Remove third letter:</th>
<th>Permutation in $S_4$</th>
<th>Avoids 123?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>14 32</td>
<td></td>
</tr>
</tbody>
</table>
**Pattern Detection by Induction**

Does 25143 avoid 123?

<table>
<thead>
<tr>
<th>Remove first letter:</th>
<th>25143</th>
<th>4132</th>
<th>Avoids 123?</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove second letter:</td>
<td>25143</td>
<td>2 143</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Remove third letter:</td>
<td>25143</td>
<td>14 32</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Remove fourth letter:</td>
<td>25143</td>
<td>241 3</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>
## Pattern Detection by Induction

Does 25143 avoid 123?

<table>
<thead>
<tr>
<th>Remove first letter:</th>
<th>Permutation in $S_4$</th>
<th>Avoids 123?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>4132</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Remove second letter:</th>
<th>25143</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove third letter:</td>
<td>25143</td>
<td>yes</td>
</tr>
<tr>
<td>Remove fourth letter:</td>
<td>25143</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2413</th>
<th></th>
</tr>
</thead>
</table>

All four tests pass → 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:** $23514 \downarrow_2 = \text{red}(2 \ 514) = 2413$.

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

Does $w$ avoid $\pi$?

<table>
<thead>
<tr>
<th>Remove $i$-th letter:</th>
<th>Permutation in $S_{n-1}$</th>
<th>Avoids $\pi$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove 1-st letter:</td>
<td>$w \downarrow_1$</td>
<td>yes</td>
</tr>
<tr>
<td>Remove 2-nd letter:</td>
<td>$w \downarrow_2$</td>
<td>yes</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>Remove $(k + 1)$-th letter:</td>
<td>$w \downarrow_{k+1}$</td>
<td>yes</td>
</tr>
</tbody>
</table>

All $k + 1$ tests pass $\rightarrow w$ avoids $\pi$. 
A FAST ALGORITHM FOR BUILDING $AV_n(\pi)$

$AV_4(213): \ldots \uparrow \ldots \uparrow \ldots \uparrow \ldots$

$AV_3(213): 123 \quad 213 \quad 312 \quad 132 \quad 231 \quad 321$

$AV_2(213): 12 \quad 21$

$AV_1(213): 1$

**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?**

\[
\begin{align*}
&\quad w \downarrow_1 \quad w \downarrow_2 \quad w \downarrow_3 \quad \cdots \quad w \downarrow_{k+1} \\
&\quad \{ \quad w \downarrow_1 \downarrow_1 \quad \cdots \quad \downarrow_1 \\
&\quad \downarrow_1 \downarrow_1 \quad \cdots \quad \downarrow_1 \\
&\quad k \text{ layers}
\end{align*}
\]

**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 \(w \downarrow_1 \downarrow_1 \cdots \downarrow_1\).
The Idea: Instead of visiting avoiders in BFS order, visit avoiders in DFS of $k$-level BFS’s.

Space Usage: $O(n \cdot \text{Max size of } k\text{-level BFS}) = O(n^{k+1})$. 

Space-efficient computation of $|A_n(\pi)|$
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