

Exhaustive Generation of Pattern-Avoiding Permutations

Hung Hoang

(joint work with Liz Hartung, Torsten Mütze, and Aaron Williams)

Introduction

Steinhaus-Johnson-Trotter algorithm

[Johnson 63], [Trotter 62], [Williams 13]

greedily generates all permutations of length n :

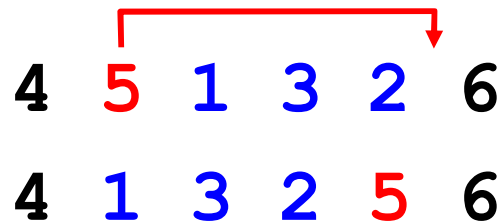
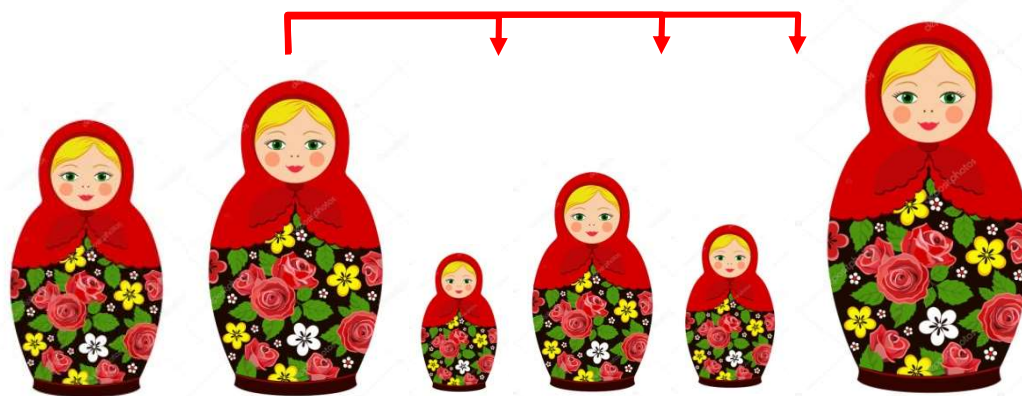
- start with the identity permutation $12 \dots n$,
- transpose the largest possible symbol with an adjacent symbol to create a new permutation

This work: generalization of the above algorithm to generate pattern-avoiding permutations

1234
1243
1423
4123
4132
1432
1342
1324
3124
3142
3412
4312
4321
3421
3241
3214
2314
2341
2431
4231
4213
2413
2143
2134

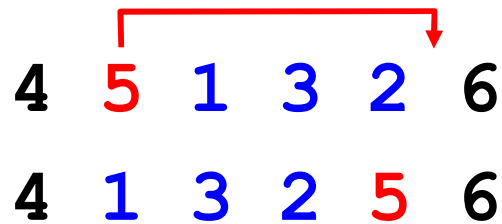
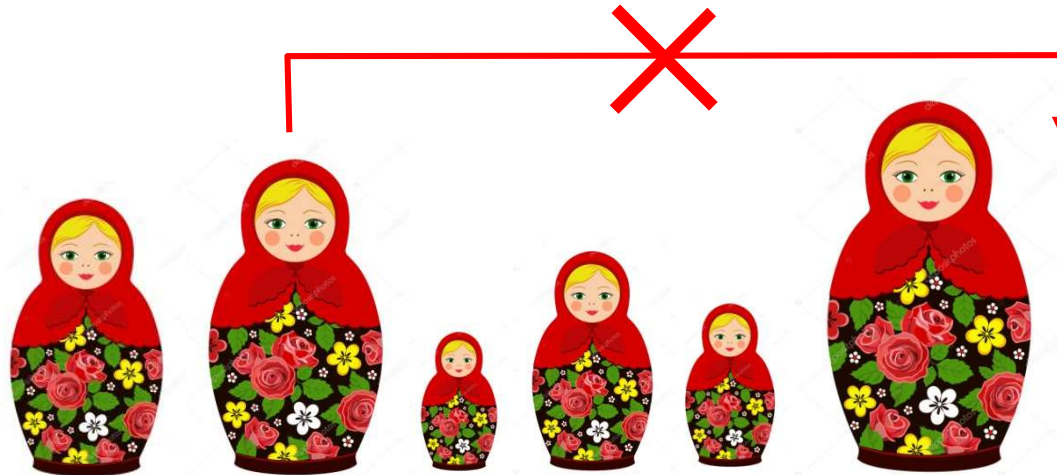
Jumps

- **Jump** := move an entry in the permutation across some neighboring smaller entries (right or left)



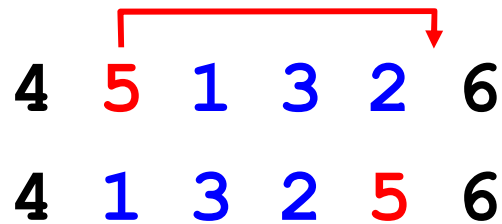
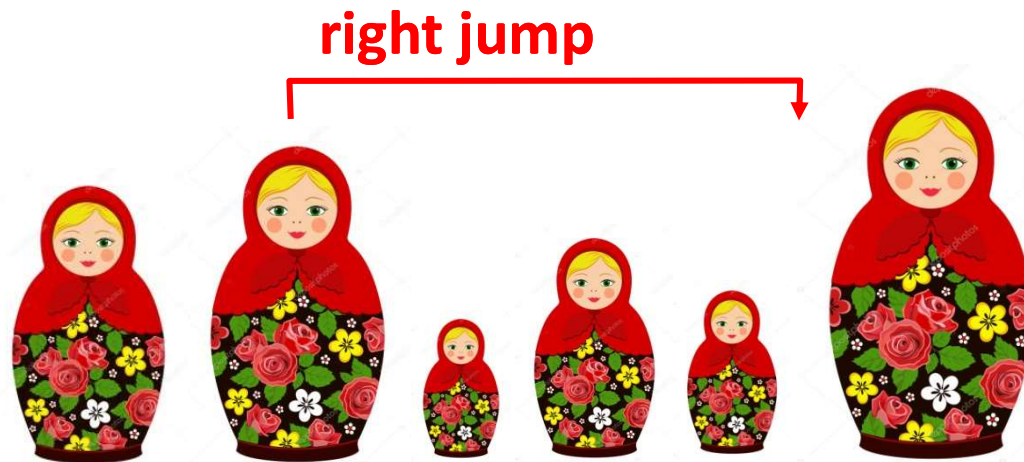
Jumps

- **Jump** := move an entry in the permutation across some neighboring smaller entries (right or left)



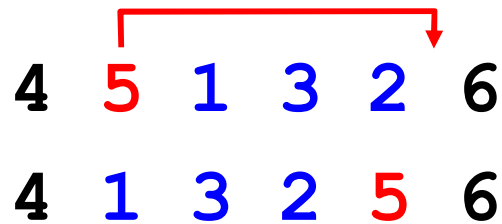
Jumps

- **Jump** := move an entry in the permutation across some neighboring smaller entries (right or left)



Jumps

- **Jump** := move an entry in the permutation across some neighboring smaller entries (right or left)



left jump



Algorithm J

Algorithm J

generates a set of permutations $F_n \subseteq S_n$

- start with the identity permutation $12 \dots n \in F_n$
- in the current permutation, perform a **minimal jump** of the **largest possible value**, so that a previously unvisited permutation from F_n is created.

- Example: $F_4 = \{1234, 1423, 4123, 4213, 2143\}$

 **1****2**3**4** ✓ minimal jumps

 **1****2****3****4** ✗ not minimal

 **1****2****3****4** ✓

Algorithm J

Algorithm J

generates a set of permutations $F_n \subseteq S_n$

- start with the identity permutation $12 \dots n \in F_n$
- in the current permutation, perform a **minimal jump** of the **largest possible value**, so that a previously unvisited permutation from F_n is created.

- Example: $F_4 = \{1234, 1423, 4123, 4213, 2143\}$

1234

1423

4123

4213

2143



Algorithm J

Algorithm J

generates a set of permutations $F_n \subseteq S_n$

- start with the identity permutation $12 \dots n \in F_n$
- in the current permutation, perform a **minimal jump** of the **largest possible value**, so that a previously unvisited permutation from F_n is created.

Stop, if no jump is possible or jump direction is ambiguous.

- Example: $F_4 = \{1234, 1423, 4123, 4213, 2143\}$

1234
1423
4123
4213
2143



4213
2143



no jump
possible

1423



direction
ambiguous

Pattern-avoiding permutations

Theorem:

If $L_n, M_n \subseteq S_n$ can be generated by Algorithm J then $L_n \cap M_n$ and $L_n \cup M_n$ can be generated by Algorithm J.

- F, G : propositional formulas ($\wedge, \vee$, and patterns as variables)
- $S_n(F \wedge G) = S_n(F) \cap S_n(G)$ (avoid F **and** G)
- $S_n(F \vee G) = S_n(F) \cup S_n(G)$ (avoid F **or** G)
- A pattern τ is called **tame**, if $S_n(\tau)$ can be generated by Algorithm J.

Corollary:

If the variables of F are tame patterns, $S_n(F)$ can be generated by Algorithm J.

Sufficient conditions

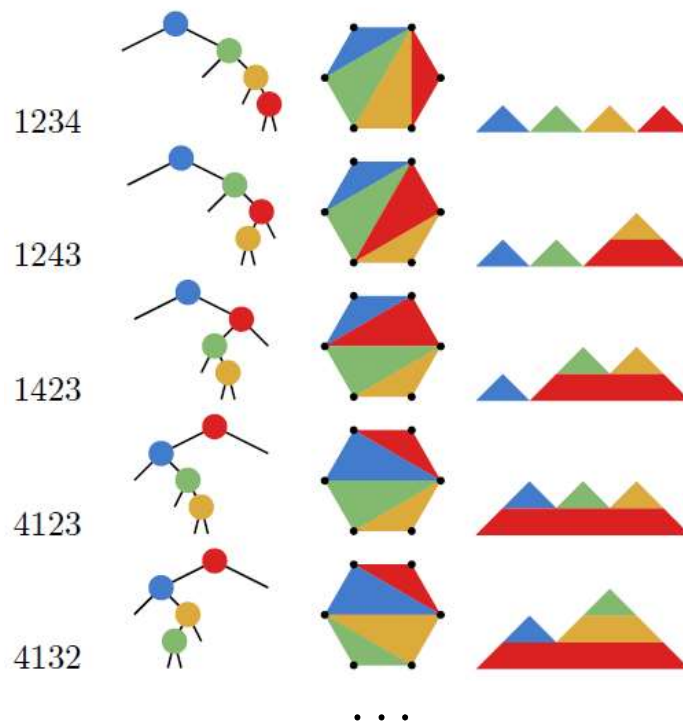
Theorem:

If a classical pattern $\tau \in S_k$ does not have the largest value k at the leftmost or rightmost position, then it is tame.

231

Catalan families

- binary trees by **rotations**
- triangulations by **edge flips**
- Dyck paths by **hill flips**



Sufficient conditions

Theorem:

If a classical pattern $\tau \in S_k$ does not have the largest value k at the leftmost or rightmost position, then it is tame.

231

- Catalan families
- binary trees by **rotations**
 - triangulations by **edge flips**
 - Dyck paths by **hill flips**

231 $\wedge$ 132

permutations

without peaks

Binary strings by **bitflips** (BRGC)

1234		000
4123		001
4312		011
3124		010
3214		110
4321		111
4213		101
2134		100

Sufficient conditions

Theorem:

If for a vincular pattern $\tau \in S_k$, k is part of the vincular pair and not at the leftmost or rightmost position, then τ is tame.

231

Set partitions by **element exchanges**

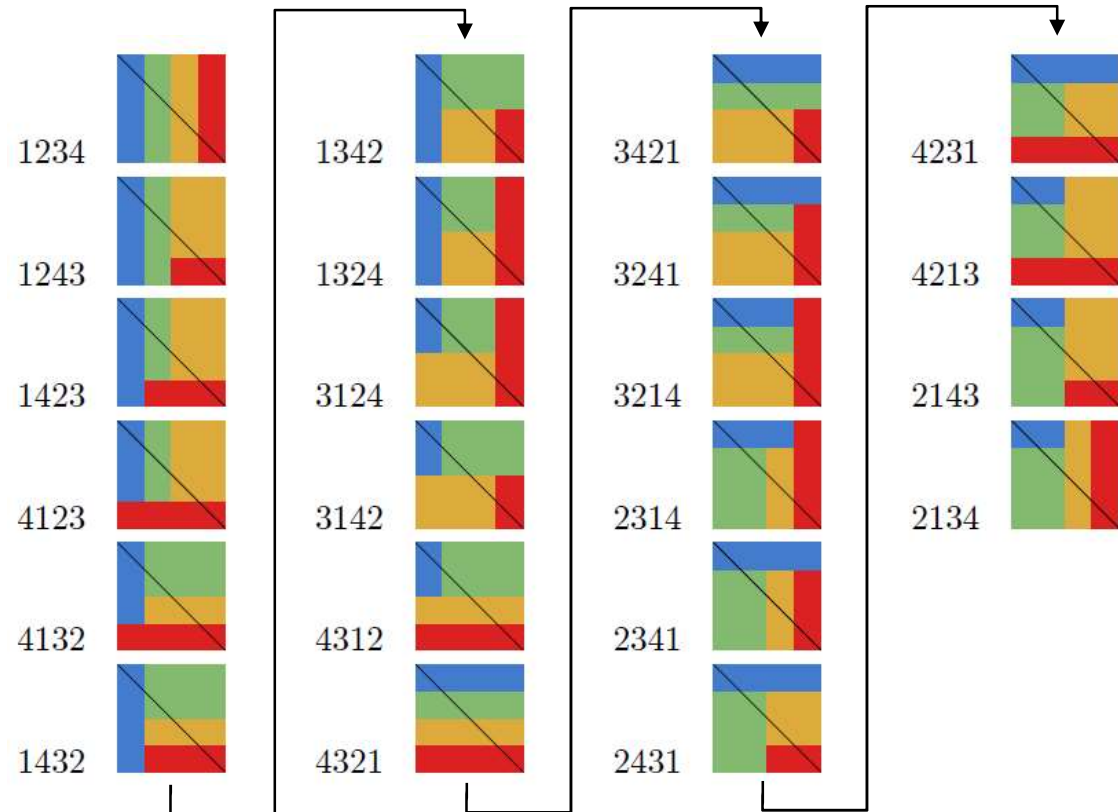
1234	1 2 3 4
1243	1 2 34
1423	1 24 3
4123	14 2 3
4132	14 23
1432	1 234
1324	1 23 4
...	

Sufficient conditions

Theorem:

If for a vincular pattern $\tau \in S_k$, k is part of the vincular pair and not at the leftmost or rightmost position, then τ is tame.

2413 $\wedge$ 3142
Baxter permutations



Sufficient conditions

Similar conditions for:

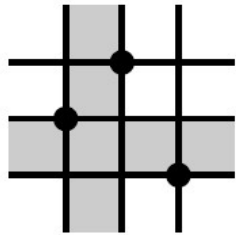
- bivincular patterns [*Bousquet-Mélou, Claesson, Dukes, and Kitaev 10*]
- barred patterns [*West 90*]
- Bruhat restricted patterns [*Woo and Yong 06*]
- partially ordered patterns [*Kitaev 06*]
- geometric grid classes [*Albert, Atkinson, Bouvel, Ruškuc, Vatter 2013*]
- ...

Sufficient conditions

All the previous patterns are special mesh patterns [Brändén and Claesson 11], for which we establish a general sufficient condition.

Theorem:

If a mesh pattern satisfies [...], it is tame.



$(2 + 2)$ -free posets

vexillary, separable,
skew-merged, X-shaped,
clumped, 2-stack sortable
permutations...

Counted patterns

- Generalization: impose upper bounds for the number of occurrences of each pattern

$$F = ((\tau_1, c_1) \wedge (\tau_2, c_2)) \vee (\tau_3, c_3)$$

|

pattern τ_1 appears at most c_1 times

If $\tau_1, \tau_2, \dots, \tau_k$ are tame, $S_n(F)$ can be generated by Algorithm J.

Final remarks

- **Application to enumeration:** Algorithm J provides a method to count the number of permutations that avoid a given pattern.
- **Efficiency:** constant time algorithms for several important patterns (231, 231, Baxter, etc.)
- **General framework:** Algorithm J generates many other combinatorial objects, by encoding them as permutations (e.g., lattice congruences of weak order on S_n)
- **Cyclic property** (first and last permutation related by jump): general condition established