Sorting monoids and algebras on Coxeter groups

Florent Hivert[1] Anne Schilling[2] Nicolas M. Thiéry[2,3]

[1]LITIS/LIFAR, Université Rouen, France

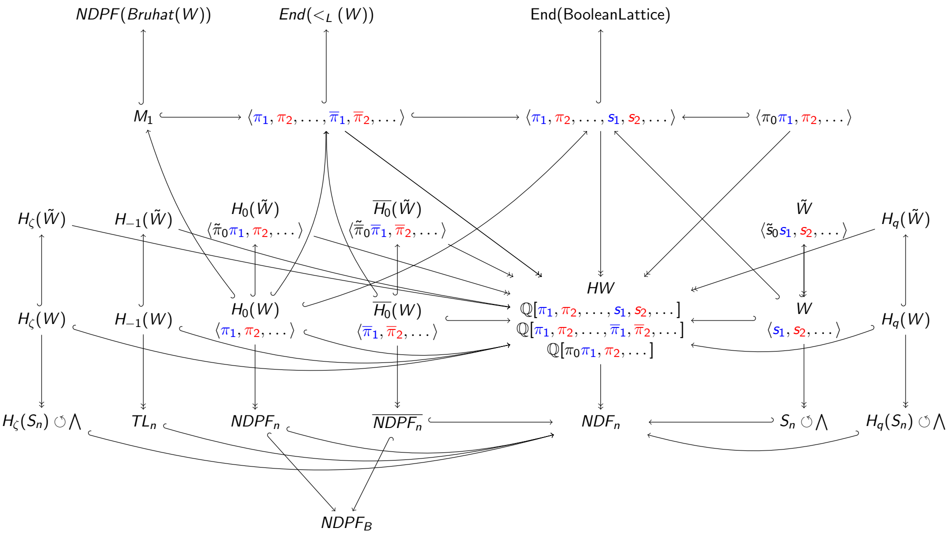
[2]University of California at Davis, USA

[3]Laboratoire de Mathématiques d'Orsay, Université Paris Sud, France

Davis, June 5th of 2009

arXiv:0711.1561v1 [math.RT]
arXiv:0804.3781v1 [math.RT]
+ research in progress ...

The Big Picture



Bubble (anti) sort algorithm

1234

Bubble (anti) sort algorithm

1234

Bubble (anti) sort algorithm

1243

Bubble (anti) sort algorithm

1423

Bubble (anti) sort algorithm

4123

Bubble (anti) sort algorithm

4123

Bubble (anti) sort algorithm

4132

Bubble (anti) sort algorithm

4312

Bubble (anti) sort algorithm

4312

Bubble (anti) sort algorithm

4321

Bubble (anti) sort algorithm

4321

Bubble (anti) sort algorithm

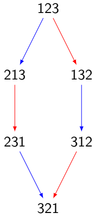
4321

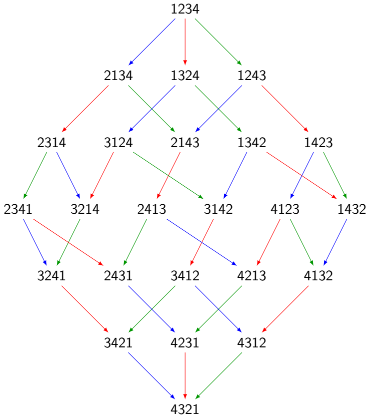
Underlying combinatorics: right permutohedron

Bubble (anti) sort algorithm

4321

Underlying combinatorics: right permutohedron

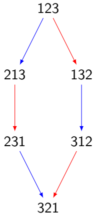


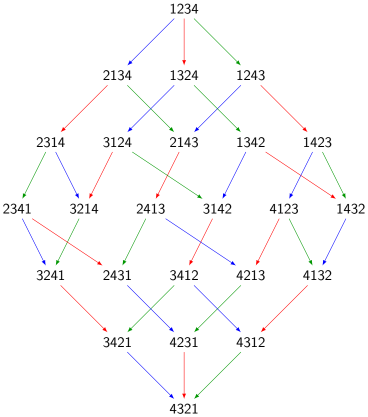


Bubble (anti) sort algorithm

4321

Underlying combinatorics: right permutohedron



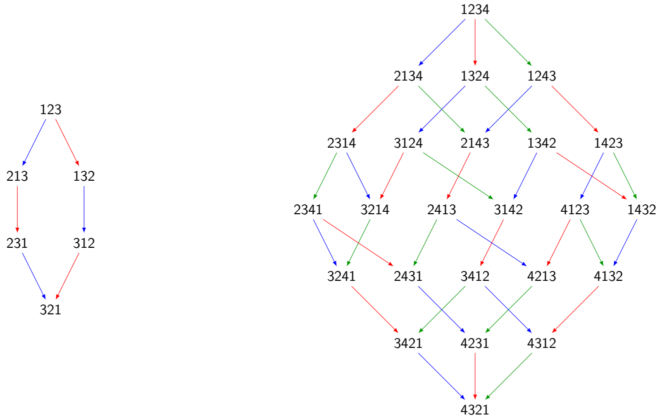


Elementary transpositions: $s_1, s_2, s_3, \ldots$

Bubble (anti) sort algorithm

4321

Underlying combinatorics: right permutohedron

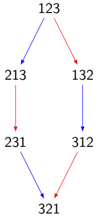


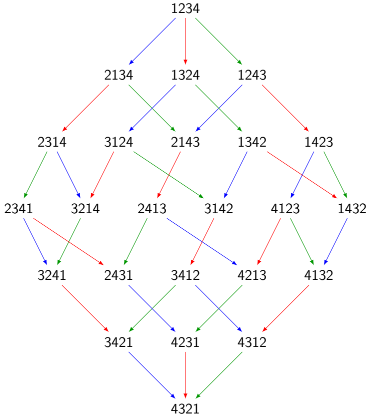
Elementary transpositions: $s_1, s_2, s_3, \ldots$

Elementary bubble antisort operators: $\pi_1, \pi_2, \pi_3, \ldots$

Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!





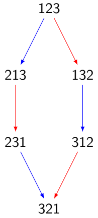
Proposition (HT'08)

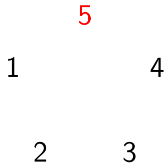
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

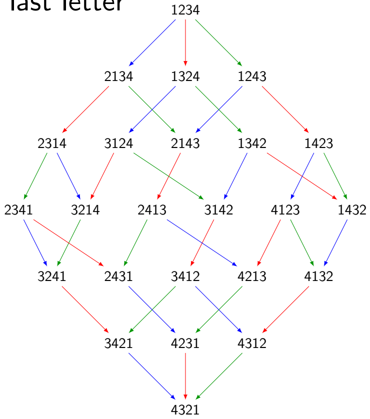
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







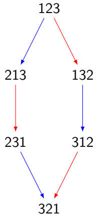
Proposition (HT'08)

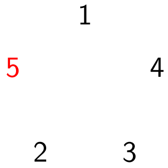
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

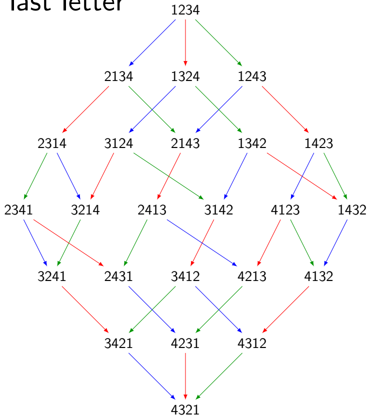
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321   There is no return!

New operator π_0: acts between first and last letter







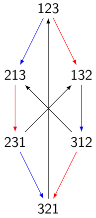
Proposition (HT'08)

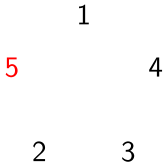
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

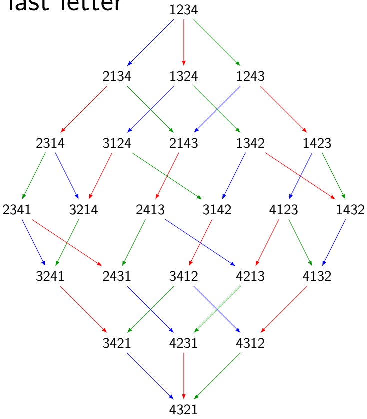
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







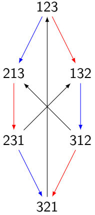
Proposition (HT'08)

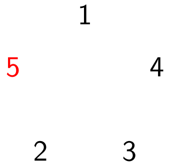
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

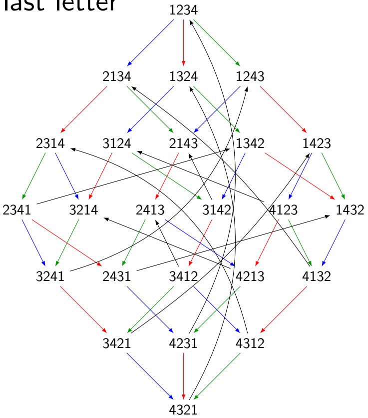
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







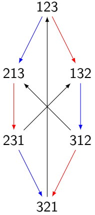
Proposition (HT'08)

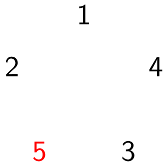
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

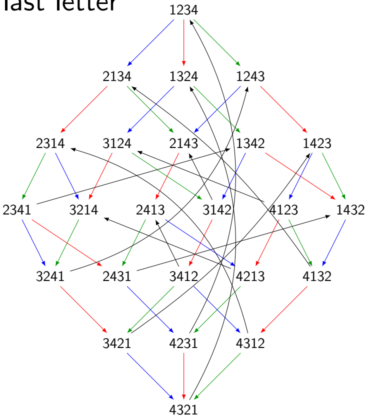
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







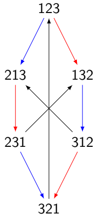
Proposition (HT'08)

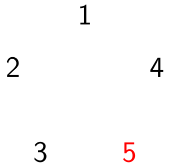
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

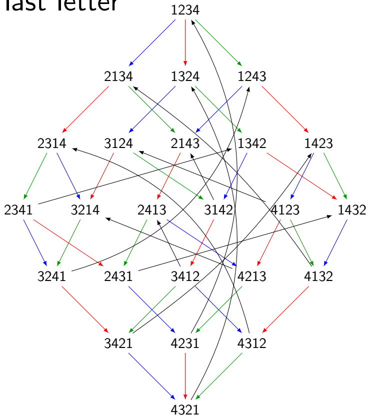
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 **There is no return!**

New operator π_0: acts between first and last letter







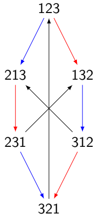
Proposition (HT'08)

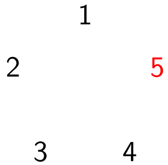
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

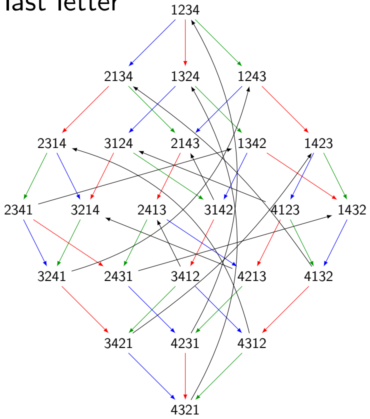
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







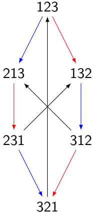
Proposition (HT'08)

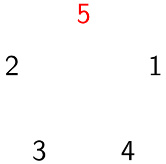
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

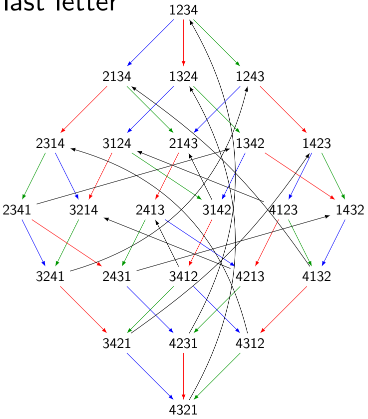
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







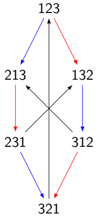
Proposition (HT'08)

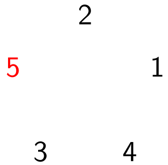
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

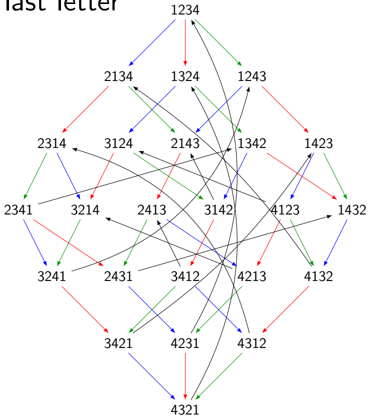
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







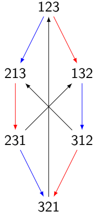
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

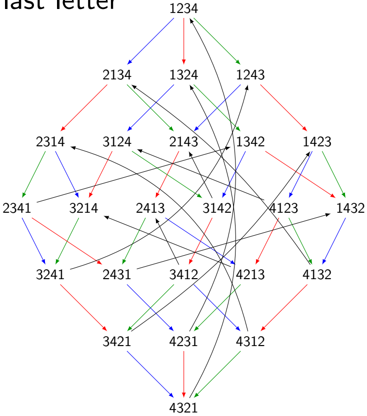
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter



2
3 1
5 4



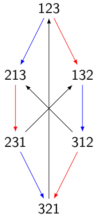
Proposition (HT'08)

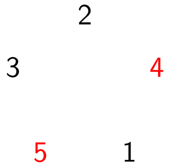
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

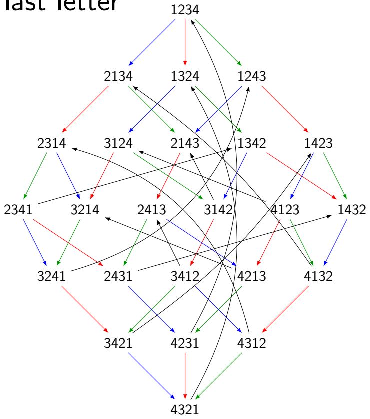
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







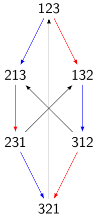
Proposition (HT'08)

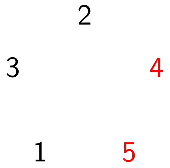
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

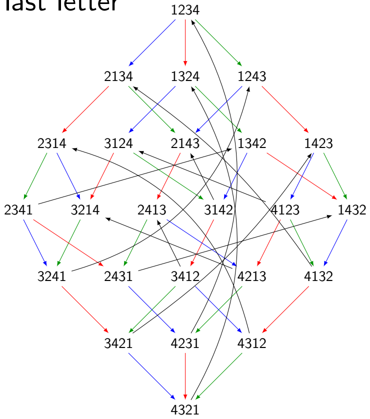
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







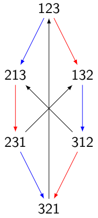
Proposition (HT'08)

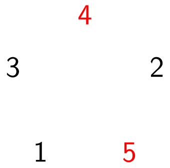
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

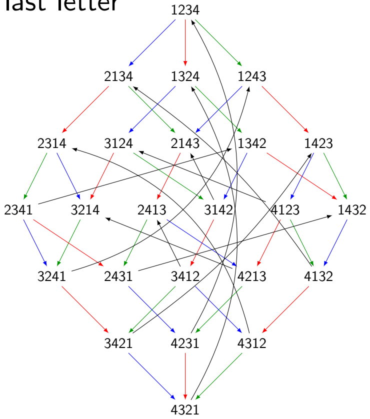
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 — There is no return!

New operator π_0: acts between first and last letter







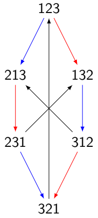
Proposition (HT'08)

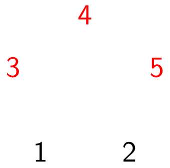
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

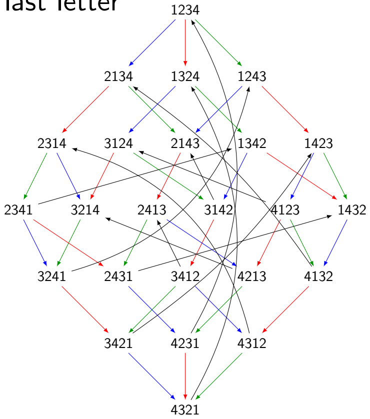
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







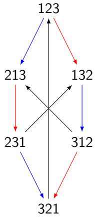
Proposition (HT'08)

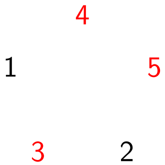
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

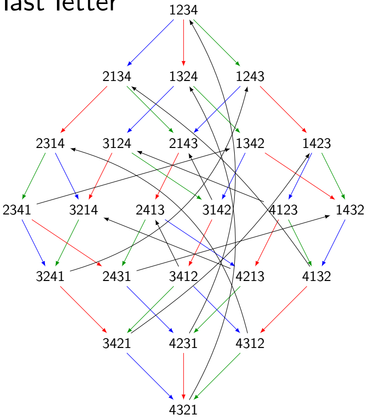
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







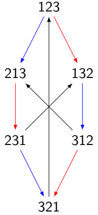
Proposition (HT'08)

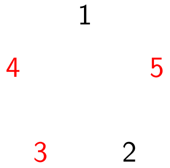
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

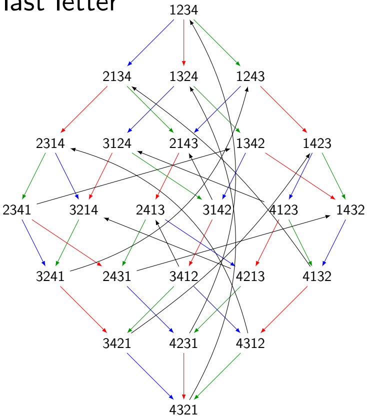
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







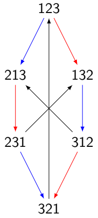
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

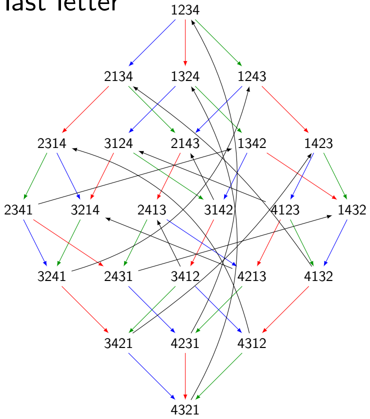
New operator π_0: acts between first and last letter



5

4 1

3 2



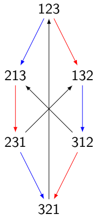
Proposition (HT'08)

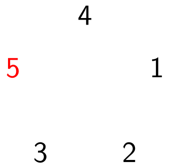
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

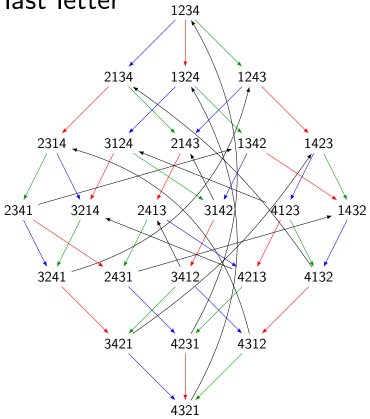
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321   There is no return!

New operator π_0: acts between first and last letter







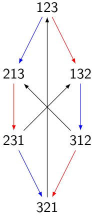
Proposition (HT'08)

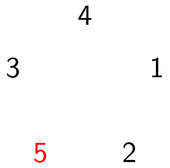
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

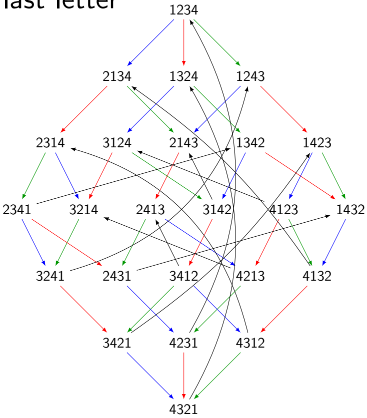
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







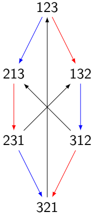
Proposition (HT'08)

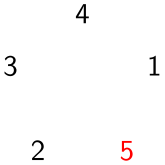
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

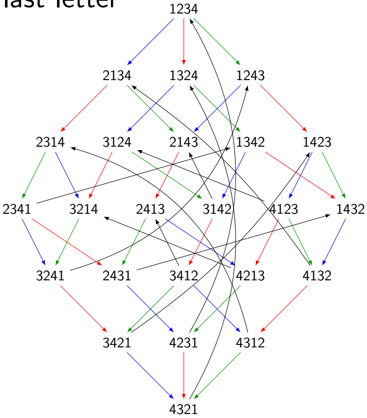
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







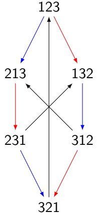
Proposition (HT'08)

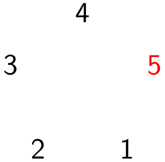
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

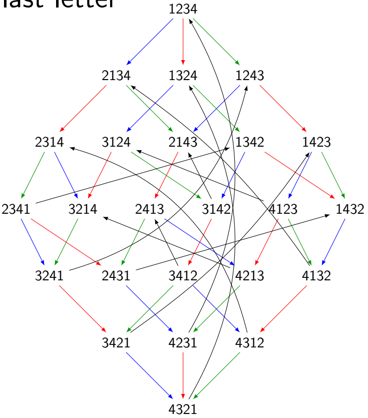
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







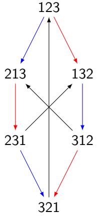
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321   There is no return!

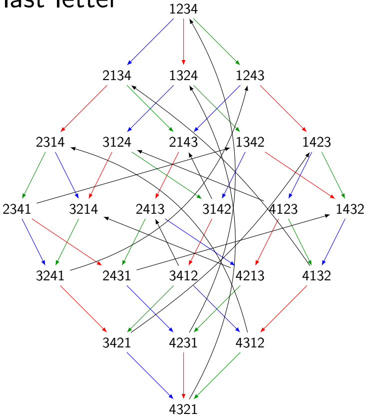
New operator π_0: acts between first and last letter



5

3   4

2   1



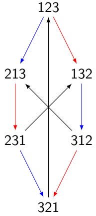
Proposition (HT'08)

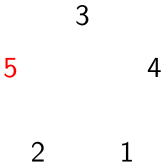
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

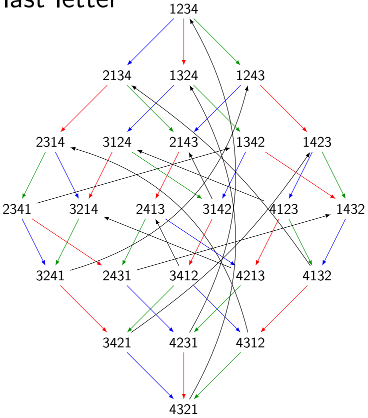
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







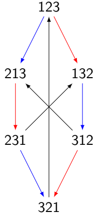
Proposition (HT'08)

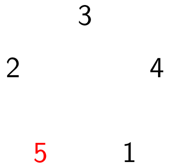
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

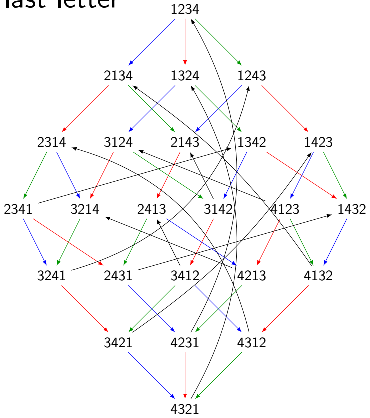
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







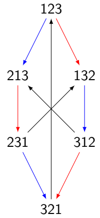
Proposition (HT'08)

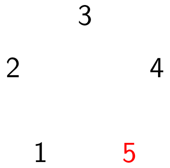
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

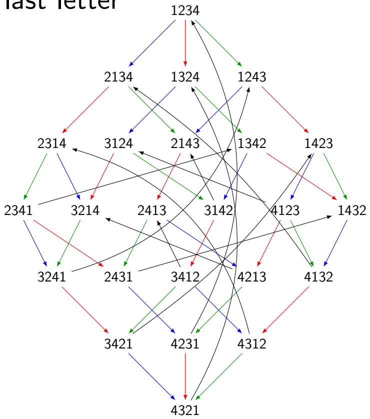
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







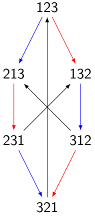
Proposition (HT'08)

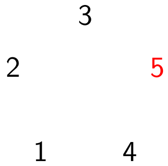
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

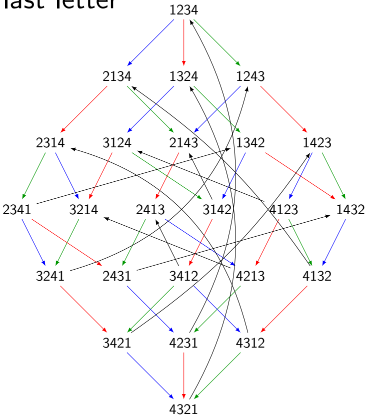
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







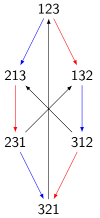
Proposition (HT'08)

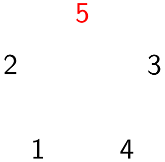
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

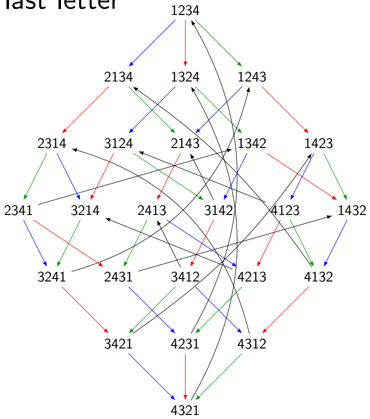
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







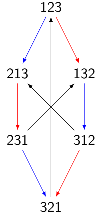
Proposition (HT'08)

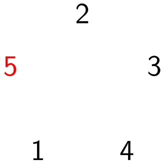
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

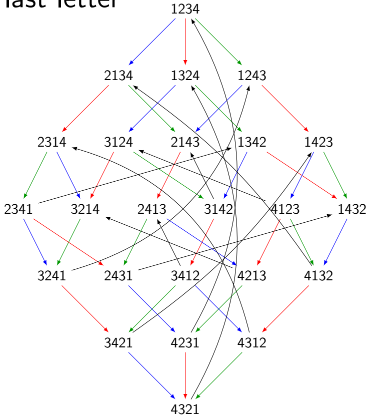
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321   There is no return!

New operator π_0: acts between first and last letter







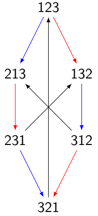
Proposition (HT'08)

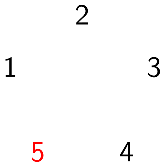
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

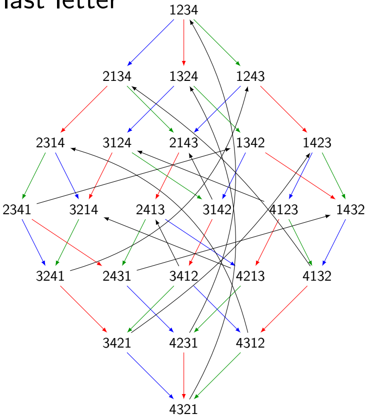
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







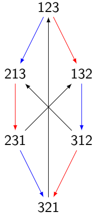
Proposition (HT'08)

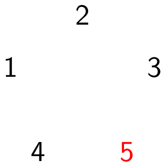
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

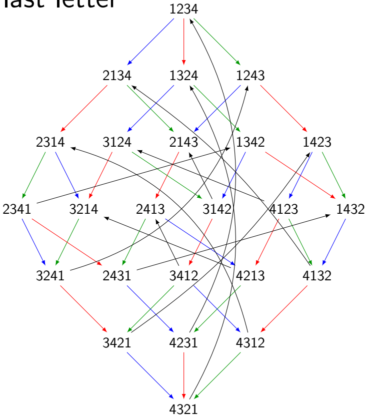
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







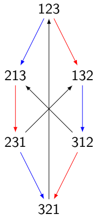
Proposition (HT'08)

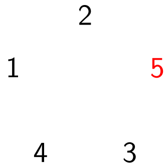
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

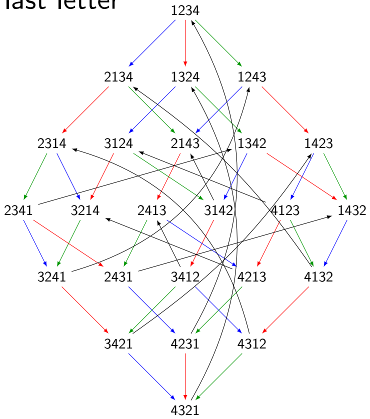
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







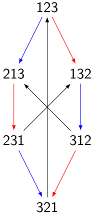
Proposition (HT'08)

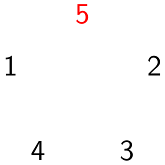
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

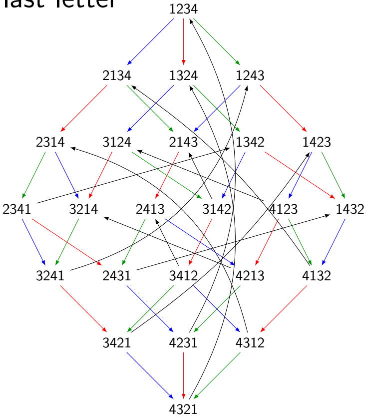
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321   There is no return!

New operator π_0: acts between first and last letter







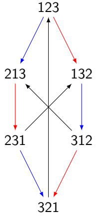
Proposition (HT'08)

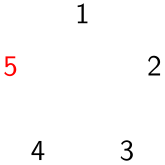
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

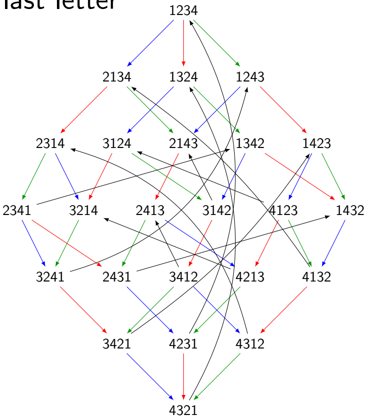
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







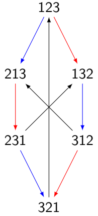
Proposition (HT'08)

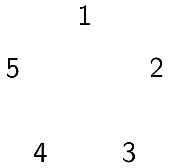
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

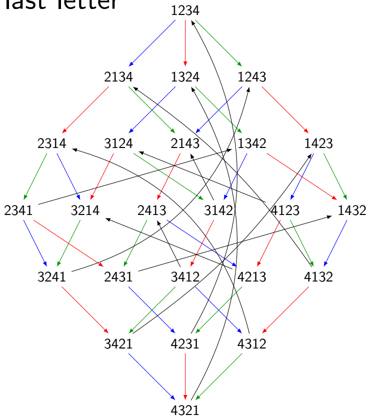
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







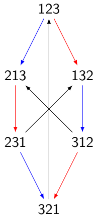
Proposition (HT'08)

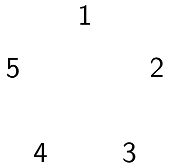
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

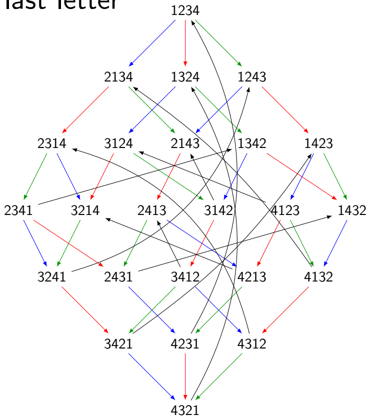
Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

Coxeter groups

Definition (Coxeter group W)

Generators : $(s_i)_{i \in S}$ (simple reflections)

Relations: $s_i^2 = 1$ and $\underbrace{s_i s_j \cdots}_{m_{i,j}} = \underbrace{s_j s_i \cdots}_{m_{i,j}}$, for $i \neq j$

Group algebra: $\mathbb{C}[W]$

Example (Type A_n: symmetric group $\mathfrak{S}_{n+1}$)

Generators: $(s_i)_{i=1,\ldots,n}$ (elementary transpositions)

Relations:

$$s_i^2 = 1 \qquad \text{for all } 1 \leq i \leq n,$$

$$s_i s_j = s_j s_i \qquad \text{for all } |i-j| > 1,$$

$$s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1} \qquad \text{for all } 1 \leq i \leq n-1.$$

Coxeter groups

Definition (Coxeter group W)

Generators : $(s_i)_{i\in S}$ (simple reflections)
Relations: $s_i^2 = 1$ and $\underbrace{s_i s_j \cdots}_{m_{i,j}} = \underbrace{s_j s_i \cdots}_{m_{i,j}}$, for $i \neq j$

Group algebra: $\mathbb{C}[W]$

Example (Type A_n: symmetric group $\mathfrak{S}_{n+1}$)

Generators: $(s_i)_{i=1,\dots,n}$ (elementary transpositions)
Relations:

$$\begin{aligned} s_i^2 &= 1 && \text{for all } 1 \leq i \leq n, \\ s_i s_j &= s_j s_i && \text{for all } |i-j| > 1, \\ s_i s_{i+1} s_i &= s_{i+1} s_i s_{i+1} && \text{for all } 1 \leq i \leq n-1. \end{aligned}$$

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$
- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{1234}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{1234}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$21\underline{34}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$21\underline{34}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{3}1\underline{4}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

  $2\underline{34}1$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{34}1$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 0, 1 — 2 — 3 ⇒ 4 $\quad 1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{3}2\underline{4}1$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{34}21$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{34}21$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

4321

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

4321

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$432\underline{1}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

  $43\underline{1}2$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$4\underline{1}32$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{1}432$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{1}43\textcolor{red}{2}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{1}43\underline{2}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{1}4\underline{2}3$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{1}\underline{2}43$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{12}43$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{12}4\underline{3}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{123}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{123}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$21\underline{3}4$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$21\underline{3}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$2\underline{3}14$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{3}14$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}214$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}214$

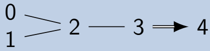
- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

  $\underline{3}241$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{3}24\underline{1}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{3}2\underline{1}4$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}\underline{1}24$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{31}24$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1324

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1324

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1342

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$134\underline{2}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$13\underline{2}4$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$1\underline{2}34$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

  $\underline{2}134$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}134$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}314$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{2}341$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}34\underline{1}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{2}3\underline{1}4$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}\underline{1}34$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{21}34$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1234

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: $0, 1 > 2 — 3 \Rightarrow 4$ $\quad 1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1234

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1234

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

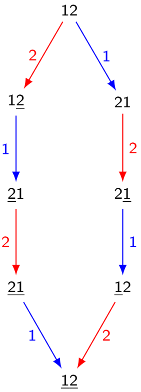
- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

1234

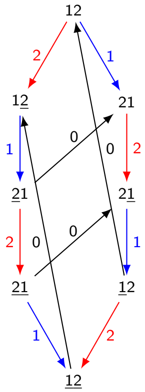
- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Geometric argument (I)



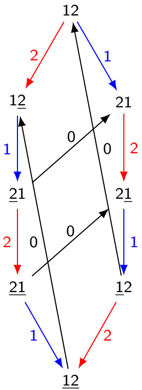
π_0, π_1, π_2 on C_2

Geometric argument (I)

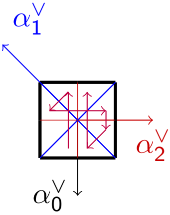


π_0, π_1, π_2 on C_2

Geometric argument (I)

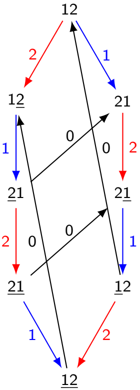


π_0, π_1, π_2 on C_2

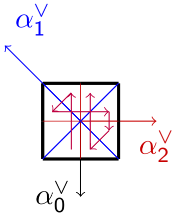


Quotient at level 0
(Steinberg torus)

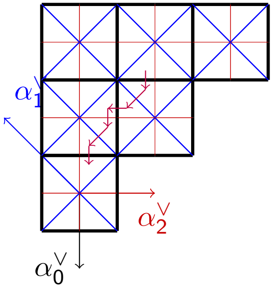
Geometric argument (I)



π_0, π_1, π_2 on C_2

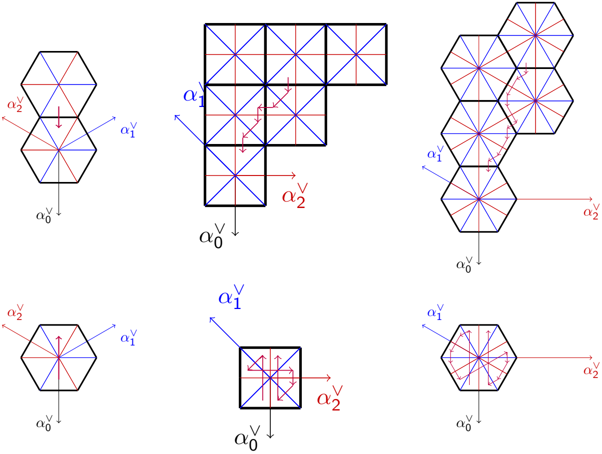


Quotient at level 0
(Steinberg torus)



Alcove picture at level 1

Type free geometric argument (II)



$1 \overset{0}{—} 2$ $\quad 0 \xrightarrow{2} 1 \xleftarrow{2} 2$ $\quad 0 — 1 \xrightarrow{3} 2$

$\widetilde{A}_2 = A_2^{(1)}$ $\quad \widetilde{C}_2 = C_2^{(1)}$ $\quad \widetilde{G}_2 = G_2^{(1)}$

0-(Iwahori)-Hecke algebras (or monoids)

Definition (0-Hecke algebra $H(W)(0)$)

Generators : $(\pi_i)_{i\in S}$

Relations: $\pi_i^2 = \pi_i$ and $\underbrace{\pi_i\pi_j\cdots}_{m_{i,j}} = \underbrace{\pi_j\pi_i\cdots}_{m_{i,j}}$ for $i \neq j$

Basis: $(\pi_w)_{w\in W}$

Example (Type A_n)

Generators: $(\pi_i)_{i=1,\dots,n}$

Relations:

$$\pi_i^2 = \pi_i \qquad \text{for all } 1 \le i \le n,$$

$$\pi_i\pi_j = \pi_j\pi_i \qquad \text{for all } |i-j| > 1,$$

$$\pi_i\pi_{i+1}\pi_i = \pi_{i+1}\pi_i\pi_{i+1} \qquad \text{for all } 1 \le i \le n-1.$$

0-(Iwahori)-Hecke algebras (or monoids)

Definition (0-Hecke algebra $H(W)(0)$)

Generators : $(\pi_i)_{i\in S}$
Relations: $\pi_i^2 = \pi_i$ and $\underbrace{\pi_i\pi_j\cdots}_{m_{i,j}} = \underbrace{\pi_j\pi_i\cdots}_{m_{i,j}}$ for $i \neq j$

Basis: $(\pi_w)_{w\in W}$

Example (Type A_n)

Generators: $(\pi_i)_{i=1,\dots,n}$
Relations:

$$\begin{aligned} \pi_i^2 &= \pi_i && \text{for all } 1 \leq i \leq n, \\ \pi_i\pi_j &= \pi_j\pi_i && \text{for all } |i-j| > 1, \\ \pi_i\pi_{i+1}\pi_i &= \pi_{i+1}\pi_i\pi_{i+1} && \text{for all } 1 \leq i \leq n-1. \end{aligned}$$

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i\in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and

$$\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}, \text{ for } i \neq j$$

Basis: $(T_w)_{w\in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Motivation: Macdonald polynomials, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, and ...

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i \in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w \in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Motivation: Macdonald polynomials, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, and ...

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i\in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w\in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Motivation: Macdonald polynomials, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, and ...

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i \in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w \in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Motivation: Macdonald polynomials, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, and . . .

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i \in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w \in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Motivation: Macdonald polynomials, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, and ...

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i \in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w \in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Motivation: Macdonald polynomials, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, and . . .

Motivation

Geometric questions:

Fl_n variety of complete flags in $\mathbb{C}^n$

$$\emptyset = V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_n = \mathbb{C}^n$$

with $\dim V_i = i$

Cohomology ring (Borel)

$$H^*(\mathrm{Fl}_n, \mathbb{Z}) \cong \mathbb{Z}[x_1, \ldots, x_n]/I_n$$

with I_n ideal generated by symmetric functions without constant term

Schubert classes $\leftrightarrow$ Schubert polynomials $\mathfrak{S}_w$, $w \in S_n$

Motivation

Geometric questions:

Fl_n variety of complete flags in $\mathbb{C}^n$

$$\emptyset = V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_n = \mathbb{C}^n$$

with $\dim V_i = i$

Cohomology ring (Borel)

$$H^*(\mathrm{Fl}_n, \mathbb{Z}) \cong \mathbb{Z}[x_1, \ldots, x_n]/I_n$$

with I_n ideal generated by symmetric functions without constant term

Schubert classes $\leftrightarrow$ Schubert polynomials $\mathfrak{S}_w$, $w \in S_n$

Motivation

Geometric questions:

Fl_n variety of complete flags in $\mathbb{C}^n$

$$\emptyset = V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_n = \mathbb{C}^n$$

with $\dim V_i = i$

Cohomology ring (Borel)

$$H^*(\mathrm{Fl}_n, \mathbb{Z}) \cong \mathbb{Z}[x_1, \dots, x_n]/I_n$$

with I_n ideal generated by symmetric functions without constant term

Schubert classes $\leftrightarrow$ Schubert polynomials $\mathfrak{S}_w$, $w \in S_n$

Algebraic definition of Schubert polynomials

Definition (Schubert polynomials, Lascoux-Schützenberger)

$$\mathfrak{S}_w = \partial_{w^{-1}w_0}\left(x_1^{n-1}x_2^{n-2}\cdots x_{n-1}\right)$$

where ∂_i is the divided difference operator

$$\partial_i f = \frac{1-s_i}{x_i - x_{i+1}} f \quad \text{for } f \in \mathbb{Z}[x_1,\ldots,x_n]$$

Relations

$$\partial_i^2 = 0$$
$$\partial_i\partial_{i+1}\partial_i = \partial_{i+1}\partial_i\partial_{i+1}$$
$$\partial_i\partial_j = \partial_j\partial_i \qquad \text{for } |i-j|>1$$

Algebraic definition of Schubert polynomials

Definition (Schubert polynomials, Lascoux-Schützenberger)

$$\mathfrak{S}_w = \partial_{w^{-1}w_0}(x_1^{n-1}x_2^{n-2}\cdots x_{n-1})$$

where ∂_i is the divided difference operator

$$\partial_i f = \frac{1-s_i}{x_i - x_{i+1}} f \quad \text{for } f \in \mathbb{Z}[x_1, \ldots, x_n]$$

Relations

$$\partial_i^2 = 0$$
$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$
$$\partial_i \partial_j = \partial_j \partial_i \qquad \text{for } |i-j| > 1$$

Combinatorial formulation of Schubert polynomials

Theorem (Fomin, Kirillov)

Coefficient of $x_1^{a_1} \cdots x_n^{a_n}$ in $\mathfrak{S}_w(x_1, \ldots, x_n)$:
number of factorizations of w in the nilCoxeter algebra into factors of length $a_1, \ldots, a_n$ that are strictly decreasing

Replace nilCoxeter algebra by the 0-Hecke algebra
($T_i^2 = 0$ is replaced by $\pi_i^2 = \pi_i$)
$\rightsquigarrow$ Grothendieck polynomials

Combinatorial formulation of Schubert polynomials

Theorem (Fomin, Kirillov)

Coefficient of $x_1^{a_1} \cdots x_n^{a_n}$ in $\mathfrak{S}_w(x_1, \ldots, x_n)$:
number of factorizations of w in the nilCoxeter algebra into factors of length $a_1, \ldots, a_n$ that are strictly decreasing

Replace nilCoxeter algebra by the 0-Hecke algebra
($T_i^2 = 0$ is replaced by $\pi_i^2 = \pi_i$)
$\rightsquigarrow$ Grothendieck polynomials

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \dots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \dots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \dots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \dots, s_1, s_2, \dots]$:
- $\mathbb{Q}[\pi_1, \pi_2, \dots, \overline{\pi}_1, \overline{\pi}_2, \dots]$:
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \dots]$:

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \ldots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \ldots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \ldots, s_1, s_2, \ldots]$:
- $\mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$:
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \ldots]$:

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \ldots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \ldots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \ldots, s_1, s_2, \ldots]$:
- $\mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$:
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \ldots]$:

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \ldots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \ldots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \ldots, s_1, s_2, \ldots]$:
- $\mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$:
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \ldots]$:

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \ldots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \ldots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \ldots, s_1, s_2, \ldots]$:
- $\mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$:
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \ldots]$:

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \ldots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \ldots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \ldots, s_1, s_2, \ldots]$:
- $\mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$:
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \ldots]$:

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \ldots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \ldots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \ldots, s_1, s_2, \ldots]$:
- $\mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$:
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \ldots]$:

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting algebras

- Transpositions: $\mathbb{Q}[s_1, s_2, \ldots]$: algebra of the symmetric group
- Bubble antisort: $\mathbb{Q}[\pi_1, \pi_2, \ldots]$: 0-Hecke algebra
- Bubble sort: $\mathbb{Q}[\overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem

Variants:

- $\mathbb{Q}[\pi_1, \pi_2, \ldots, s_1, s_2, \ldots]$: Hecke group algebra
- $\mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$: idem!
- $\mathbb{Q}[\pi_0, \pi_1, \pi_2, \ldots]$: idem!

Theorem (HT 05, HT 06, HST 09)

Characterization, representation theory
Connection with the affine Hecke algebra
Combinatorics: descents

Sorting monoids

- Transpositions: $\langle s_1, s_2, \ldots \rangle$: symmetric group
- Bubble antisort: $\langle \pi_1, \pi_2, \ldots \rangle$: 0-Hecke monoid
- Bubble sort: $\langle \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: idem

Variants:

- $\langle \pi_1, \pi_2, \ldots, s_1, s_2, \ldots \rangle$: ?
- $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: ?
- $\langle \pi_0, \pi_1, \pi_2, \ldots \rangle$: ?

Theorem (HST 09, work in progress)

Representation theory for $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
Combinatorics: left/right/Bruhat order, tilings of n-gons by rombis

Sorting monoids

- Transpositions: $\langle s_1, s_2, \ldots \rangle$: symmetric group
- Bubble antisort: $\langle \pi_1, \pi_2, \ldots \rangle$: 0-Hecke monoid
- Bubble sort: $\langle \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: idem

Variants:

- $\langle \pi_1, \pi_2, \ldots, s_1, s_2, \ldots \rangle$: ?
- $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: ?
- $\langle \pi_0, \pi_1, \pi_2, \ldots \rangle$: ?

Theorem (HST 09, work in progress)

Representation theory for $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
Combinatorics: left/right/Bruhat order, tilings of n-gons by rombis

Sorting monoids

- Transpositions: $\langle s_1, s_2, \ldots \rangle$: symmetric group
- Bubble antisort: $\langle \pi_1, \pi_2, \ldots \rangle$: 0-Hecke monoid
- Bubble sort: $\langle \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: idem

Variants:

- $\langle \pi_1, \pi_2, \ldots, s_1, s_2, \ldots \rangle$: ?
- $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: ?
- $\langle \pi_0, \pi_1, \pi_2, \ldots \rangle$: ?

Theorem (HST 09, work in progress)

Representation theory for $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
Combinatorics: left/right/Bruhat order, tilings of n-gons by rombis

Sorting monoids

- Transpositions: $\langle s_1, s_2, \ldots \rangle$: symmetric group
- Bubble antisort: $\langle \pi_1, \pi_2, \ldots \rangle$: 0-Hecke monoid
- Bubble sort: $\langle \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: idem

Variants:

- $\langle \pi_1, \pi_2, \ldots, s_1, s_2, \ldots \rangle$: ?
- $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: ?
- $\langle \pi_0, \pi_1, \pi_2, \ldots \rangle$: ?

Theorem (HST 09, work in progress)

Representation theory for $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
Combinatorics: left/right/Bruhat order, tilings of n-gons by rombis

Sorting monoids

- Transpositions: $\langle s_1, s_2, \ldots \rangle$: symmetric group
- Bubble antisort: $\langle \pi_1, \pi_2, \ldots \rangle$: 0-Hecke monoid
- Bubble sort: $\langle \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: idem

Variants:

- $\langle \pi_1, \pi_2, \ldots, s_1, s_2, \ldots \rangle$: ?
- $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$: ?
- $\langle \pi_0, \pi_1, \pi_2, \ldots \rangle$: ?

Theorem (HST 09, work in progress)

Representation theory for $\langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
Combinatorics: left/right/Bruhat order, tilings of n-gons by rombis

Sorting monoids

- Transpositions: $\langle s_1, s_2, \dots \rangle$: symmetric group
- Bubble antisort: $\langle \pi_1, \pi_2, \dots \rangle$: 0-Hecke monoid
- Bubble sort: $\langle \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$: idem

Variants:

- $\langle \pi_1, \pi_2, \dots, s_1, s_2, \dots \rangle$: ?
- $\langle \pi_1, \pi_2, \dots, \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$: ?
- $\langle \pi_0, \pi_1, \pi_2, \dots \rangle$: ?

Theorem (HST 09, work in progress)

Representation theory for $\langle \pi_1, \pi_2, \dots, \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$
Combinatorics: left/right/Bruhat order, tilings of n-gons by rombis

Sorting monoids

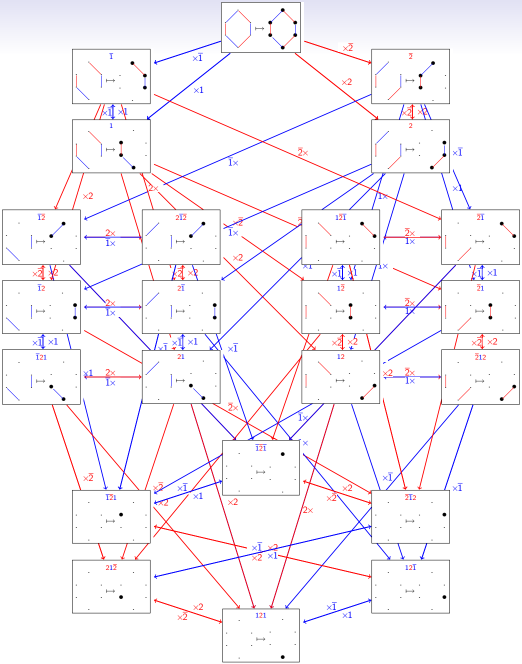
- Transpositions: $\langle s_1, s_2, \dots \rangle$: symmetric group
- Bubble antisort: $\langle \pi_1, \pi_2, \dots \rangle$: 0-Hecke monoid
- Bubble sort: $\langle \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$: idem

Variants:

- $\langle \pi_1, \pi_2, \dots, s_1, s_2, \dots \rangle$: ?
- $\langle \pi_1, \pi_2, \dots, \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$: ?
- $\langle \pi_0, \pi_1, \pi_2, \dots \rangle$: ?

Theorem (HST 09, work in progress)

Representation theory for $\langle \pi_1, \pi_2, \dots, \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$
Combinatorics: left/right/Bruhat order, tilings of n-gons by rombis



Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide
- Sage-Combinat: `combinat.sagemath.org`

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide
- Sage-Combinat: `combinat.sagemath.org`

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide
- Sage-Combinat: `combinat.sagemath.org`

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide
- Sage-Combinat: `combinat.sagemath.org`

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide
- Sage-Combinat: `combinat.sagemath.org`

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide
- Sage-Combinat: `combinat.sagemath.org`