Permutree sorting

Vincent Pilaud, Viviane Pons, Daniel Tamayo Jiménez

Séminaire Lotharingien de Combinatoire **85B** (2021)
Article #31, 12 pp.

Proceedings of the 33rd Conference on Formal Power Series and Algebraic Combinatorics (Ramat Gan)

Permutree sorting

Vincent Pilaud[*1], Viviane Pons[†2], and Daniel Tamayo Jiménez[‡2]

[1]*CNRS & LIX, École Polytechnique, Palaiseau, France*
[2]*Université Paris-Saclay, CNRS, Laboratoire de recherche en informatique, Orsay, France*

Abstract. Generalizing stack sorting and c-sorting for permutations, we define the permutree sorting algorithm. Given two disjoint subsets U and D of $\{2, \dots, n-1\}$, the (U, D)-permutree sorting tries to sort a permutation $\pi \in \mathfrak{S}_n$ and fails if and only if there are $1 \leq i < j < k \leq n$ such that π contains the subword jki with $j \in U$ or kij with $j \in D$. This algorithm is seen as a way to explore an automaton which either rejects all reduced words of π, or accepts those reduced words for π whose prefixes are all (U, D)-permutree sortable.

Keywords: stack sorting, automata, permutrees, weak order

1 Introduction

The motivation of this paper is the classical family of *stack-sortable* permutations introduced by D. Knuth in his textbook [3, Sect. 2.2.1] and characterized by the following equivalent conditions for a permutation $\pi \in \mathfrak{S}_n$:

(i) π is sent to the identity by the stack sorting S defined by $S(\tau n \rho) := S(\tau)S(\rho)n$.
(ii) π avoids the pattern 231 (*i.e.* there is no $p < q < r$ such that $\pi_r < \pi_p < \pi_q$).
(iii) π is minimal (minimal number of inversions) among all linear extensions of a binary tree on n nodes (seen as a poset, labeled in inorder, oriented towards its root).
(iv) For $i < j < k$, the inversion set $\mathrm{inv}(\pi) := \{(\pi_p, \pi_q) \mid p < q \text{ and } \pi_p > \pi_q\}$ of π contains the inversion (k, j) as soon as it contains the inversion (k, i).
(v) π admits a reduced word of the form $\pi = c_{I_1} \cdots c_{I_p}$ with nested $I_1 \supseteq \cdots \supseteq I_p$, where $c_{\{i_1 < \cdots < i_j\}} := s_{i_j} \cdots s_{i_1}$ is a product of the simple transpositions $s_i := (i \;\; i+1)$.

In his work on lattice congruences [7, 8, 6], N. Reading defined counterparts to conditions (iii), (iv), and (v) above, parametrized by the choice of a Coxeter element c in a finite Coxeter group W: the minimality in c-Cambrian classes, the c-alignment, and the c-sortability. In the situation of the symmetric group $\mathfrak{S}_n$, we can think of a Coxeter element on $\mathfrak{S}_n$ as an orientation of an $(n-1)$-path, or equivalently as a partition of $\{2, \dots, n-1\}$ into two subsets U and D. The Cambrian analogues of the conditions (ii), (iii), (iv) and (v) above are the following equivalent conditions for a permutation $\pi \in \mathfrak{S}_n$:

[*]vincent.pilaud@lix.polytechnique.fr. Supp. by ANR CAPPS 17 CE40 0018 and CHARMS 19 CE40 0017.
[†]viviane.pons@lri.fr
[‡]daniel.tamayo-jimenez@lri.fr

(ii′) For $i < j < k$, the permutation π avoids the subword jki if $j \in U$ and kij if $j \in D$.
(iii′) π is minimal among all linear extensions of a c-Cambrian tree on n nodes [1].
(iv′) For $i < j < k$, if $\mathrm{inv}(\pi)$ contains (k, i), it contains (k, j) if $j \in U$ and (j, i) if $j \in D$.
(v′) π admits a reduced word of the form $\pi = c_{I_1} \cdots c_{I_p}$ with nested $I_1 \supseteq \ldots \supseteq I_p$, where $c_I := c_{i_1} \cdots c_{i_{|I|}}$ is the subword of $c := c_1 \cdots c_{n-1}$ indexed by $I := \{i_1 < \ldots < i_j\}$.

These Cambrian combinatorics motivated the introduction of permutree combinatorics [4]. Permutrees generalize and interpolate between permutations, binary trees, and binary sequences, and explain the combinatorial, geometric, and algebraic similarities between them. The data is now given by two subsets U and D of $\{2, \ldots, n-1\}$ that are not anymore required to form a partition of $\{2, \ldots, n-1\}$ (they may intersect and may not cover all the set). It was proved in [4] that the conditions (ii′), (iii′), and (iv′) are still equivalent for a permutation $\pi \in \mathfrak{S}_n$. We call a permutation *(U, D)-permutree minimal* when satisfying these conditions, *i.e.* when it is minimal (minimal number of inversions) in its (U, D)-permutree class. The objective of this paper is to discuss characterizations of the permutree minimal permutations in terms of their reduced words. In other words, we aim at a condition playing the role of condition (v′) and equivalent to conditions (ii′), (iii′), and (iv′) for arbitrary subsets U and D of $\{2, \ldots, n-1\}$.

We first focus on the case where $U = \varnothing$ and $D = \{j\}$ for some $j \in \{2, \ldots, n-1\}$, or the opposite. To recognize the permutree minimal permutations in terms of their reduced words, we use two automata $\mathbb{U}(j)$ and $\mathbb{D}(j)$ defined inductively in Section 2 (we assume the reader familiar with basic automata theory, see *e.g.* [2]). More precisely, a permutation contains no subword jki (resp. kij) with $i < j < k$ if and only if it admits a reduced word accepted by the automaton $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$). We moreover show in Section 3 that the sets of reduced words for a permutation π accepted by the automaton $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$) have a rich combinatorial structure. This leads on the one hand to a simple algorithm to construct a reduced word for π accepted by $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$), and to a tree structure on the permutations containing no subword jki (resp. kij) with $i < j < k$.

We then study the situation of arbitrary subsets U and D of $\{2, \ldots, n-1\}$. In general, the reduced words accepted by the automata $\mathbb{U}(j)$ for each $j \in U$ and by $\mathbb{D}(j)$ for each $j \in D$ are distinct. We prove however in Section 4 that there is a reduced word simultaneously accepted by all these automata when U and D are disjoint. It implies that given any permutation π avoiding jki if $j \in U$ and kij if $j \in D$, we can sort π while preserving these avoiding conditions. We call *(U, D)-permutree sorting* such a procedure, as they generalize the classical stack sorting.

When the subsets U and D partition $\{2, \ldots, n-1\}$, we actually show in Section 5 that the reduced word simultaneously accepted by the automata $\mathbb{U}(j)$ for $j \in U$ and $\mathbb{D}(j)$ for $j \in D$ is the c-sorting word of π as defined in [8]. This yields in particular an alternative proof that condition (v′) characterizes the Cambrian minimal permutations.

Finally, we discuss possible extensions to arbitrary finite Coxeter groups in Section 6.

All proofs and details omitted here for space reasons can be found in [5].

2 Automata for reduced words

The *symmetric group* $\mathfrak{S}_n$ is generated by *simple transpositions* $s_i := (i \;\; i+1)$ for $i \in [n-1]$. Each permutation π decomposes as products of transpositions of the form $\pi = s_{i_1} \cdots s_{i_k}$. The minimal number of transpositions in such a decomposition is the *length* $\ell(\pi)$ of π and the decompositions of length $\ell(\pi)$ are the *reduced words* for π.

For a fixed $j \in \{2, \dots, n-1\}$, we say that a permutation π *avoids* jki (resp. kij) if for any $i < j < k$, the word jki (resp. kij) does not appear as a subword of the one-line notation of π. We insist on the fact that while the value j is fixed, i and k take all possible values such that $1 \leq i < j < k \leq n$. This convenient notion here should not be mixed up with the notion of pattern avoidance where j is not fixed. For instance, the permutation 42135 avoids $2ki$, $3ki$, and $4ki$ (and therefore the pattern 231), but contains $ki3$ (and therefore the pattern 312) because its one-line notation contains 423.

For $j \in \{2, \dots, n-1\}$, we define inductively two *automata* $\mathbb{U}(j)$ and $\mathbb{D}(j)$ as shown in Figure 1. The induction stops at $\mathbb{U}(n)$ and $\mathbb{D}(1)$, which are defined by deleting the transitions s_n and s_0 respectively in Figure 1. For instance, Figure 2 shows the complete automata $\mathbb{U}(2)$ and $\mathbb{D}(2)$ for $n = 4$. We call a state *healthy*, *ill*, or *dead* depending on whether it belongs to the top, middle, or bottom row of the automata. Each state has $n-1$ possible transitions, one for each s_i for $i \in [n-1]$, but we only explicitly indicate the ones between different states. The automata $\mathbb{U}(j)$ and $\mathbb{D}(j)$ take as entry a reduced word $s_{i_1} \cdots s_{i_\ell}$ for a permutation of $\mathfrak{S}_n$ and read it from left to right. We start at the initial state (marked with "start"), and at step t we follow the transition marked by the letter s_{i_t} if any, or stay in the current state otherwise. After ℓ steps, the reduced word $s_{i_1} \cdots s_{i_\ell}$ is accepted if the current state is accepting (doubly circled, healthy or ill states), and rejected otherwise (dead states). The main tool of this paper is the following statement.

Theorem 2.1. *Fix $j \in \{2, \dots, n-1\}$. The following conditions are equivalent for $\pi \in \mathfrak{S}_n$:*

- *π admits a reduced word accepted by the automaton $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$),*
- *π avoids jki (resp. kij).*

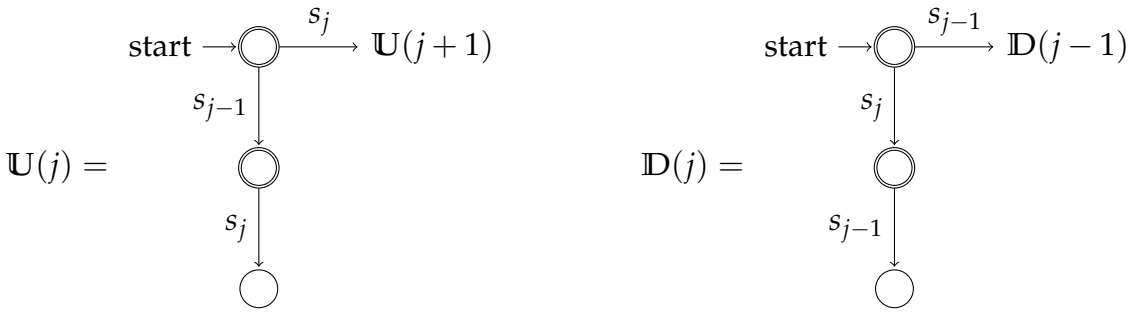


Figure 1: Recursive definition of the automata $\mathbb{U}(j)$ and $\mathbb{D}(j)$.

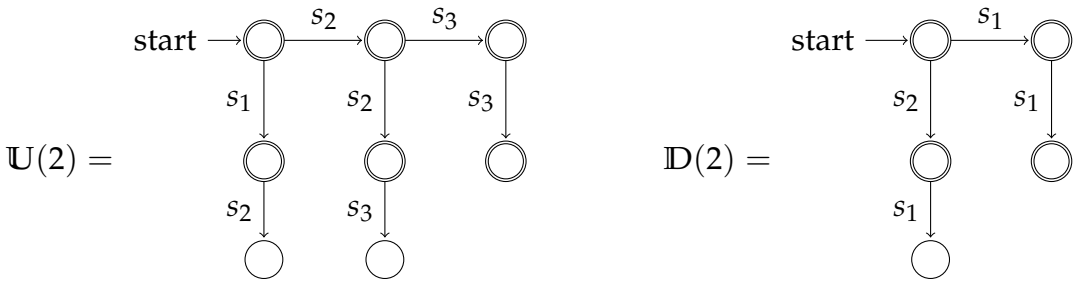


Figure 2: The complete automata $\mathbb{U}(2)$ and $\mathbb{D}(2)$ for $n = 4$.

For example, the permutation $\pi = 3412$ avoids $2ki$ and $s_2 \cdot s_1 \cdot s_3 \cdot s_2$ is a reduced expression of π accepted by $\mathbb{U}(2)$. On the other hand, 3241 contains the subword 241 and all its reduced expressions (such as $s_2 \cdot s_1 \cdot s_2 \cdot s_3$) are rejected by $\mathbb{U}(2)$.

3 Structure of accepted reduced words

3.1 The set of accepted reduced words

Observe that a given permutation π may admit both accepted and rejected reduced words. For instance, the transposition $(j-1 \;\; j+1)$ has reduced words $s_j \cdot s_{j-1} \cdot s_j$ accepted by $\mathbb{U}(j)$ and $s_{j-1} \cdot s_j \cdot s_{j-1}$ rejected by $\mathbb{U}(j)$. However, the set of accepted reduced words satisfies the following properties.

Proposition 3.1 (Who can do more can do less!)**.** *The set of reduced words accepted by $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$) is closed by taking prefix.*

Proposition 3.2 (When health goes, everything goes!)**.** *If π admits an accepted reduced expression, then π admits an accepted reduced expression starting with any descent that remains in the healthy states. In other words, let $\ell \in [n-1]$ distinct from $j-1$ (resp. j). If $\pi \in \mathfrak{S}_n$ avoids jki (resp. kij) and reverses ℓ and $\ell+1$, it admits a reduced word starting with s_ℓ and accepted by $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$).*

Proposition 3.3 (All roads lead to Rome!)**.** *Given a permutation $\pi \in \mathfrak{S}_n$, all the reduced words for π accepted by $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$) end at the same state.*

3.2 Finding accepted reduced words

Proposition 3.2 has a strong algorithmic consequence. Imagine we want to test whether a permutation $\pi \in \mathfrak{S}_n$ is minimal in its permutree class for $U = \{j\}$ and $D = \varnothing$. Of course, the quickest way is to check for all $i < j < k$ whether π contains the subword jki. But since this interpretation will be lost beyond type A, let us impose the use of reduced

words for π to make this test. While it would be *a priori* necessary to check all reduced words on the automaton $\mathbb{U}(j)$, Proposition 3.2 enables us to construct without loss of generality a reduced word for π and we will just need to check if this one is accepted by $\mathbb{U}(j)$. Somewhat dually, one can also construct a reduced word accepted by $\mathbb{U}(j)$ that is a reduced word for π if and only if π avoids jki. This is done in the following algorithm, that we call *$(\{j\},\varnothing)$-permutree sorting*.

Algorithm 1: $(\{j\},\varnothing)$-permutree sorting

Input : a permutation $\pi \in \mathfrak{S}_n$ and an integer $j \in [n]$
Output: a reduced word accepted by $\mathbb{U}(j)$, candidate reduced word for π
$w := \varepsilon$
repeat
  if $\exists\ \ell \neq j-1$ *such that* ℓ *and* $\ell+1$ *are reversed in* π **then**
    $\pi := s_\ell \cdot \pi, \quad w := w \cdot s_\ell$
    if $\ell = j$ **then** $j := j+1$
if $j-1$ *and* j *are reversed in* π **then**
  $\pi := s_{j-1} \cdot \pi, \quad w := w \cdot s_{j-1}$
  $w := w \cdot w' \cdot w''$ where w' sorts $\pi_{[j]}$ and w'' sorts $\pi_{[n]\setminus[j]}$
return w

Corollary 3.4. *For any permutation π and $j \in \{2,\dots,n-1\}$, Algorithm 1 returns a reduced word w accepted by $\mathbb{U}(j)$ such that w is a reduced word for π if and only if π avoids jki.*

Example 3.5. *Let us present the $(\{2\},\varnothing)$-permutree sorting algorithm in action for the permutations $\pi_1 := 3421$ and $\pi_2 := 4231$. The steps of the algorithm are presented in the table below. Each row contains the states of the permutation π and of the word w and the current values of j and ℓ in use at each step. Notice that for $\pi_1 := 3421$ the algorithm ends with the identity, which coincides with the fact that $\pi_1 := 3421$ avoids $2ki$. In contrast, for $\pi_2 := 4231$ the algorithm ends with the permutation 1243, which coincides with the fact that $\pi_2 := 4231$ contains $2ki$.*

π_1	w_1	j_1	ℓ_1
3421	ε	2	2
2431	s_2	3	1
1432	$s_2 \cdot s_1$	3	3
1342	$s_2 \cdot s_1 \cdot s_3$	4	2
1243	$s_2 \cdot s_1 \cdot s_3 \cdot s_2$	4	3
1234	$s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_3$	4	

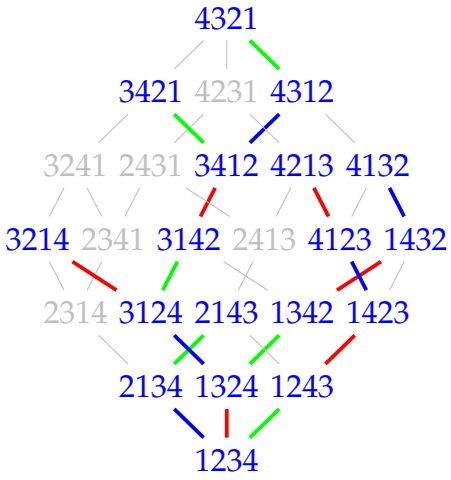
π_2	w_2	j_2	ℓ_2
4231	ε	2	3
3241	s_3	2	2
2341	$s_3 \cdot s_2$	3	1
1342	$s_3 \cdot s_2 \cdot s_1$	3	2
1243	$s_3 \cdot s_2 \cdot s_1 \cdot s_2$	3	

3.3 Generating trees on accepted reduced words

Propositions 3.1 and 3.3 also have a relevant consequence, which is more combinatorial this time. Namely, they naturally define generating trees for the $(\{j\},\varnothing)$-permutree minimal permutations, following certain special reduced words for them. To construct these trees, pick an arbitrary priority order $\prec$ on $\{s_1,\dots,s_{n-1}\}$. For a $(\{j\},\varnothing)$-permutree minimal permutation $\pi\in\mathfrak{S}_n$, denote by $\pi(\{j\},\varnothing,\prec)$ the $\prec$-lexicographic minimal reduced word for π that is accepted by $\mathbb{U}(j)$. Denote by $\mathcal{R}(n,\{j\},\varnothing,\prec)$ the set of reduced words of the form $\pi(\{j\},\varnothing,\prec)$ for all $(\{j\},\varnothing)$-permutree minimal permutations $\pi\in\mathfrak{S}_n$. The following statement is an analogue of Proposition 3.1.

Proposition 3.6. *The set $\mathcal{R}(n,\{j\},\varnothing,\prec)$ is closed by prefix.*

This yields a natural generating tree for $\mathcal{R}(n,\{j\},\varnothing,\prec)$ where the parent of a reduced word w is obtained by deleting its last letter. Replacing each reduced word by the corresponding permutation, this provides a generating tree for the $(\{j\},\varnothing)$-permutree minimal permutations of $\mathfrak{S}_n$. We have represented this tree on the right for $j=2$ with the priority order $s_1\prec s_2\prec s_3$. It is natural to draw these trees on top of the Hasse diagram of the right weak order on permutations. The edges of the trees corresponding to the right multiplications by s_1, s_2 and s_3 are colored by blue, red, and green respectively.



4 Intersection of automata

4.1 A common reduced word

We now consider arbitrary subsets U and D of $\{2,\dots,n-1\}$. We already know from [4] and Theorem 2.1 that the following conditions are equivalent for $\pi\in\mathfrak{S}_n$:

(i) the permutation π is minimal in its (U,D)-permutree class,
(ii) for $i<j<k$, the permutation π avoids the subword jki if $j\in U$ and kij if $j\in D$,
(iii) for each $j\in U$ (each $j\in D$), there is a reduced word for π accepted by $\mathbb{U}(j)$ (resp. by $\mathbb{D}(j)$).

A natural question is whether there is a reduced word simultaneously accepted by all these automata. We start with an example showing that this is not always the case.

Example 4.1. *For $j\in\{2,\dots,n-1\}$, let $U=\{j\}=D$ and $\pi=s_{j-1}\cdot s_j\cdot s_{j-1}=s_j\cdot s_{j-1}\cdot s_j$. Then, the word $s_{j-1}\cdot s_j\cdot s_{j-1}$ is accepted by $\mathbb{D}(j)$ but not by $\mathbb{U}(j)$, while the word $s_j\cdot s_{j-1}\cdot s_j$ is accepted by $\mathbb{U}(j)$ but not by $\mathbb{D}(j)$.*

This example clearly extends to all subsets U and D of $\{2, \dots, n-1\}$ with a non-empty intersection. In contrast, this situation cannot occur when U and D are disjoint.

Theorem 4.2. *Consider two disjoint subsets U and D of $\{2, \dots, n-1\}$. The following conditions are equivalent for $\pi \in \mathfrak{S}_n$:*

- *π admits a reduced word accepted by all automata $\mathbb{U}(j)$ for $j \in U$ and $\mathbb{D}(j)$ for $j \in D$,*
- *π avoids jki for all $j \in U$ and kij for all $j \in D$.*

4.2 Intersection of automata

Theorem 4.2 can be rephrased in terms of intersection of automata. Recall that the intersection of some automata $\mathbb{A}_1, \dots, \mathbb{A}_p$ is the automaton $\mathbb{A} = \bigcap_{i \in [p]} \mathbb{A}_i$ such that a word is accepted by $\mathbb{A}$ if and only if it is accepted by all $\mathbb{A}_1, \dots, \mathbb{A}_p$. A state of the automaton $\mathbb{A}$ is p-tuple formed by states of the automata $\mathbb{A}_1, \dots, \mathbb{A}_p$, and a transition t simultaneously changes all entries of the p-tuple corresponding to states modified by t. See [2, p. 59–60] for details. We denote by $\mathbb{P}(U, D)$ the intersection of the automata $\mathbb{U}(j)$ for $j \in U$ and $\mathbb{D}(j)$ for $j \in D$. We thus obtain the following statement.

Corollary 4.3. *When U and D are disjoint, the following conditions are equivalent for $\pi \in \mathfrak{S}_n$:*

- *π admits a reduced word accepted by the automaton $\mathbb{P}(U, D)$,*
- *π contains no subword jki if $j \in U$ and kij if $j \in D$ with $i < j < k$.*

We say that a state of $\mathbb{P}(U, D)$ is *healthy* (resp. *ill*, resp. *dead*) when the corresponding states in $\mathbb{U}(j)$ for $j \in U$ and $\mathbb{D}(j)$ for $j \in D$ are all healthy (resp. contain at least one ill state, but no dead one, resp. contains at least one dead state).

4.3 The set of accepted reduced words of $\mathbb{P}(U, D)$

Applying the principles of Section 3.1 to each automaton $\mathbb{U}(j)$ for $j \in U$ and $\mathbb{D}(j)$ for $j \in D$, we derive similar principles for the automaton $\mathbb{P}(U, D)$.

Proposition 4.4. *The set of reduced words accepted by $\mathbb{P}(U, D)$ is closed by prefix.*

Proposition 4.5. *If a permutation π avoids jki for $j \in U$ and kij for $j \in D$, and admits a reduced word starting with s_ℓ such that the transition s_ℓ leads to an healthy state of $\mathbb{P}(U, D)$, then it admits a reduced word starting with s_ℓ and accepted by $\mathbb{P}(U, D)$.*

Proposition 4.6. *Given a permutation $\pi \in \mathfrak{S}_n$, all the reduced words for π accepted by $\mathbb{P}(U, D)$ end at the same state.*

4.4 Permutree sorting

We now generalize Algorithm 1 to the following *(U, D)-permutree sorting algorithm*. As in Algorithm 1, the algorithm will read the automaton $\mathbb{P}(U, D)$ without actually constructing it. To virtually follow the edges of the automaton $\mathbb{P}(U, D)$, we use the following two operations on our sets U and D:

- $\mathrm{moveU}(U, \ell)$ is U if $\ell \notin U$ and $(U \smallsetminus \{\ell\}) \cup \{\ell+1\}$ if $\ell \in U$,
- $\mathrm{moveD}(D, \ell)$ is D if $\ell+1 \notin D$ and $(D \smallsetminus \{\ell+1\}) \cup \{\ell\}$ if $\ell+1 \in D$.

In contrast to Algorithm 1, we have opted here for a recursive style.

Algorithm 2: (U, D)-permutree sorting

Function $\mathrm{permutreeSort}(\pi, U, D)$
  Input : a permutation $\pi \in \mathfrak{S}_n$ and two disjoint subsets U and D of $[n]$
  Output: a reduced word accepted by $\mathbb{P}(U, D)$, candidate reduced word for π
  if $\exists\, \ell \in [n-1]$ *such that* ℓ *and* $\ell+1$ *are reversed in* π, *and* $\ell+1 \notin U$ *and* $\ell \notin D$ **then**
    return $s_\ell \cdot \mathrm{permutreeSort}(s_\ell \cdot \pi,\ \mathrm{moveU}(U, \ell),\ \mathrm{moveD}(D, \ell))$
  if $\exists\, \ell \in [n-1]$ *such that* ℓ *and* $\ell+1$ *are reversed in* π,
  and $(\ell+1 \notin U$ *or* $\pi([\ell+1]) = [\ell+1])$ *and* $(\ell \notin D$ *or* $\pi([\ell-1]) = [\ell-1])$ **then**
    return $s_\ell \cdot \mathrm{permutreeSort}(s_\ell \cdot \pi,\ \mathrm{moveU}(U \smallsetminus \{\ell+1\}, \ell),\ \mathrm{moveD}(D \smallsetminus \{\ell\}, \ell))$
  return ε

Note that in Algorithm 2, we could ignore n in the list U (resp. 1 in the list D) since $\mathbb{U}(n)$ (resp. $\mathbb{D}(1)$) accepts all reduced words. We have decided not to do it to be coherent with our recursive definition of $\mathbb{U}(j)$ and $\mathbb{D}(j)$.

Corollary 4.7. *For any permutation π and any disjoint subsets U and D of $\{2, \dots, n-1\}$, Algorithm 2 returns a reduced word w accepted by $\mathbb{P}(U, D)$ such that w is a reduced word for π if and only if π avoids jki for $j \in U$ and kij for $j \in D$.*

Example 4.8. *The tables on the right present the $(\{3\}, \{2\})$-permutree sorting algorithm in action for the permutations $\pi_1 := 3214$, $\pi_2 := 1324$ and $\pi_3 := 1342$. Each row in these tables contains the states of the permutation π and of the word w, the current values of j and ℓ in use at each step, and the values of k for which we have to check that $\pi([k]) = [k]$, crossed in red when it fails. These tables show that π_1 and π_2 are $(\{3\}, \{2\})$-permutree sortable while π_3 is not (as it contains $3ki$).*

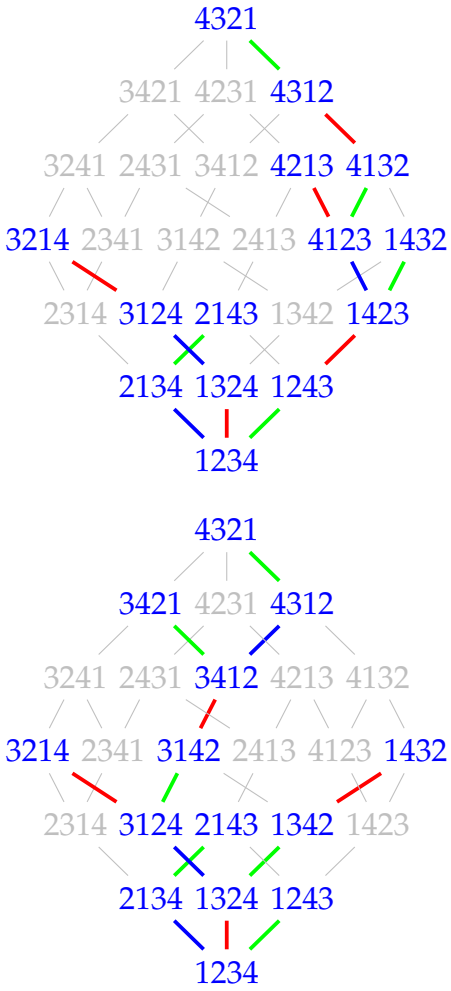
π_1	w_1	U_1	D_1	ℓ_1	k_1
3214	ε	$\{3\}$	$\{2\}$	1	.
3124	s_1	$\{3\}$	$\{1\}$	2	3
2134	$s_1 \cdot s_2$	$\varnothing$	$\{1\}$	1	0
1234	$s_1 \cdot s_2 \cdot s_1$				

π_2	w_2	U_2	D_2	ℓ_2	k_2
1324	ε	$\{3\}$	$\{2\}$	2	1, 3
1234	s_2				

π_3	w_3	U_3	D_3	ℓ_3	k_3
1342	ε	$\{3\}$	$\{2\}$	2	1, ~~3~~

4.5 Generating trees

As in Section 3.3, we can define natural generating trees for the (U, D)-permutree minimal permutations. Namely, fix an arbitrary priority order $\prec$ on $\{s_1, \dots, s_{n-1}\}$. For an (U, D)-permutree minimal permutation π, we denote by $\pi(U, D, \prec)$ the $\prec$-lexicographic minimal reduced word for π that is accepted by $\mathbb{P}(U, D)$. We denote by $\mathcal{R}(n, U, D, \prec)$ the set of reduced words of the form $\pi(U, D, \prec)$ for all (U, D)-permutree minimal permutations $\pi \in \mathfrak{S}_n$. As in Proposition 3.6, $\mathcal{R}(n, U, D, \prec)$ is closed by prefix. This yields a natural generating tree on $\mathcal{R}(n, U, D, \prec)$ where the parent of a reduced word w is obtained by deleting its last letter. Replacing each reduced word by the corresponding permutation, this provides a generating tree for the (U, D)-permutree minimal permutations of $\mathfrak{S}_n$. We have represented this tree on the right for $(U, D) = (\{2, 3\}, \varnothing)$ (top) and $(\{2\}, \{3\})$ (bottom) and the priority order $s_1 \prec s_2 \prec s_3$. As in Section 3.3, these trees are drawn on top of the Hasse diagram of the right weak order on permutations and the edges are colored according to the simple transpositions.



5 Permutree sorting versus Coxeter sorting

In this section, we discuss the particular case when U and D partition $\{2, \dots, n-1\}$. In that situation, we connect the (U, D)-permutree sorting with the c-sorting of [9].

We consider a *Coxeter element* c of $\mathfrak{S}_n$, *i.e.* the product of all simple transpositions $\{s_1, \dots, s_{n-1}\}$ in an arbitrary order. For a permutation $\pi \in \mathfrak{S}_n$, the *c-sorting word* $\pi(c)$ is the lexicographically smallest reduced word for π in the infinite word $c^\infty = c \cdot c \cdot c \cdot c \cdots$. We let $I_1, \dots, I_p$ denote the subsets of $[n-1]$ such that $\pi(c) = c_{I_1} \cdot c_{I_2} \cdots c_{I_p}$ where c_I is the subword of c obtained by keeping only the letters s_i for $i \in I$. The permutation π is *c-sortable* if $I_1 \supseteq I_2 \supseteq \cdots \supseteq I_p$.

A Coxeter element c of $\mathfrak{S}_n$ defines a partition $\{2, \dots, n-1\} = U_c \sqcup D_c$, where U_c (resp. D_c) consists of the elements $j \in \{2, \dots, n-1\}$ such that s_j appears before (resp. after) s_{j-1} in c. For instance, when $c = s_2 \cdot s_5 \cdot s_4 \cdot s_3 \cdot s_1 \cdot s_6$, we obtain $U_c = \{2, 4, 5\}$ and $D_c = \{3, 6\}$. Said differently, $j \in U$ (resp. $j \in D$) if c is accepted by $\mathbb{U}(j)$ but not by $\mathbb{D}(j)$ (resp. by $\mathbb{D}(j)$ but not by $\mathbb{U}(j)$). The c-sorting of [9] and the (U_c, D_c)-permutree sorting are connected as follows.

Theorem 5.1. *The following are equivalent for any Coxeter element c and any permutation π:*
(i) *π is c-sortable,*
(ii) *the c-sorting word $\pi(c)$ is accepted by the automaton $\mathbb{P}(U_c, D_c)$,*
(iii) *there exists a reduced word for π accepted by the automaton $\mathbb{P}(U_c, D_c)$,*
(iv) *for each $j \in \{2, \dots, n-1\}$, there exists a reduced word for π that is accepted by the automaton $\mathbb{U}(j)$ if $j \in U_c$ and $\mathbb{D}(j)$ if $j \in D_c$,*
(v) *π avoids jki for $j \in U_c$ and kij for $j \in D_c$.*

We conclude this section with three negative observations and warnings about the connection between c-sorting and (U_c, D_c)-permutree sorting.

Remark 5.2. *Even if a permutation π avoids jki (resp. kij) for a given j, there might be no Coxeter element c for which π is c-sortable and $j \in U_c$ (resp. $j \in D_c$). For instance,* 41325 *avoids* $2ki$ *and* $ki4$*, but contains* 352 *and* 413*, so it is not c-sortable for any Coxeter element c.*

Remark 5.3. *When a permutation π is not c-sortable, there might exist $j \in U_c$ (resp. $j \in D_c$) for which the c-sorting word $\pi(c)$ is not accepted by $\mathbb{U}(j)$ (resp. $\mathbb{D}(j)$) even if π avoids jki (resp. kij). For instance, consider $c = s_2 \cdot s_1 \cdot s_3$ and $\pi = 4213 = s_3 \cdot s_1 \cdot s_2 \cdot s_1 = s_3 \cdot s_2 \cdot s_1 \cdot s_2 = s_1 \cdot s_3 \cdot s_2 \cdot s_1$. Then $2 \in U_c$, and the c-sorting word $\pi(c) = s_1 \cdot s_3 \cdot s_2 \cdot s_1$ is rejected by $\mathbb{U}(2)$ while π contains no $2ki$ (and indeed $s_3 \cdot s_2 \cdot s_1 \cdot s_2$ is accepted by $\mathbb{U}(2)$).*

Remark 5.4. *Given a Coxeter element c, the word c^∞ which is used to compute $\pi(c)$ is a sorting network. This means that we decide* beforehand *a list of transpositions to apply if appropriate. On the other hand, the permutree sorting given in Algorithm 2 is* not *a sorting network. Indeed, the order on transpositions depends on the permutation and more specifically on the state of the automaton we are at. A natural question then occurs: can we replace the permutree sorting algorithm by a sorting network? Or said differently, when U and D are disjoint but do not cover $\{2, \dots, n-1\}$, can we find a word $\tilde{c}$ which plays the role of c^∞ in the sense that looking at $\pi(\tilde{c})$ would be enough to check whether π is accepted by $\mathbb{P}(U, D)$? The answer is negative in general. A counter-example is found for $n = 5$, $U = \{2\}$, and $D = \{4\}$. In this case one can check through computer exploration that no reduced word $\tilde{c}$ of the maximal permutation* 54321 *can be used as a sorting network.*

6 Extensions to finite Coxeter groups

In this section we present possible extensions of our results to other finite Coxeter groups. Remember that the c-sorting algorithm defined by Reading [7] works in all types and defines the Cambrian lattices as certain lattice quotients of the weak order. On the other hand, there is no definition of permutrees [4] beyond type A. The present work can be used as a first step towards defining and studying such objects.

A Coxeter group is generated by a family $(s_i)_{i \in [n]}$ such that $s_i^2 = 1$ and $(s_i s_j)^{m_{i,j}} = 1$ for certain coefficients $m_{i,j}$ with $2 \leq m_{i,j} \leq \infty$ when $i \neq j$. In particular, this defines the

Coxeter graph whose vertices are the s_i and edges are the pairs (s_i, s_j) with $m_{i,j} > 2$. A Coxeter element c of a finite Coxeter group W defines an orientation of the Coxeter graph of W: we orient the edge from s_i to s_j if s_i appears before s_j in c and reciprocally. Each orientation defines a certain lattice congruence of the weak order: it is the smallest lattice congruence such that $s_j \equiv s_j s_i \equiv \cdots \equiv s_j s_i s_j \ldots$ where the last element is of length $m_{i,j} - 1$. The c-sortable elements are the weak order minimal elements of the congruence classes. See [6] for more details.

By allowing each edge of the Coxeter graph to be either non-oriented, right, left or with both orientations, we define some new lattice congruences which we call the *permutree congruences*. In type A, the Coxeter graph is a line (s_j is connected to s_{j-1} and s_{j+1}) and the orientation is then entirely determined by our two sets U and D: $j \in U$ (resp. $j \in D$) means we orient the edge (s_j, s_{j-1}) from s_j to s_{j-1} (resp. s_{j-1} to s_j). The case where $U = \varnothing$ and $D = \{j\}$ or the opposite corresponds to choosing one edge and orient it. Then the automaton $\mathbb{D}(j)$ is actually constructed recursively by following a certain path in the graph from the oriented edge (in type A, this means decreasing j). We conjecture that this principle can be somehow generalized to other types and give two examples in type B and D.

6.1 Type B

The Coxeter graph of type B is similar to the one of type A with the exception that $m_{n-1,n} = 4$ (whereas $m_{n-1,n} = 3$ in type A). We claim that the automata $\mathbb{D}(j)$ and $\mathbb{U}(j)$ can be recursively constructed following Figure 3. The idea is to "unfold" the edge (s_{n-1}, s_n) and use the type A construction on a linear graph $s_1, \ldots, s_n, s_{n-1}, s_n, \ldots, s_1$. Notice that $\mathbb{D}(j)$ has not been altered while $\mathbb{U}(j)$ is virtually as before except for when $j = n$ where it is now connected to $\mathbb{D}(n)$ as if it was "bouncing back". These automata capture completely the sorting process of type B permutations. It can also be checked for small sizes that they indeed select the right elements in the lattice. In particular, the case where $U \sqcup D = \{2, \ldots, n\}$ gives $\binom{2n}{n}$ permutree sortable permutations, *i.e.* the B-Catalan numbers as expected from the Cambrian case.

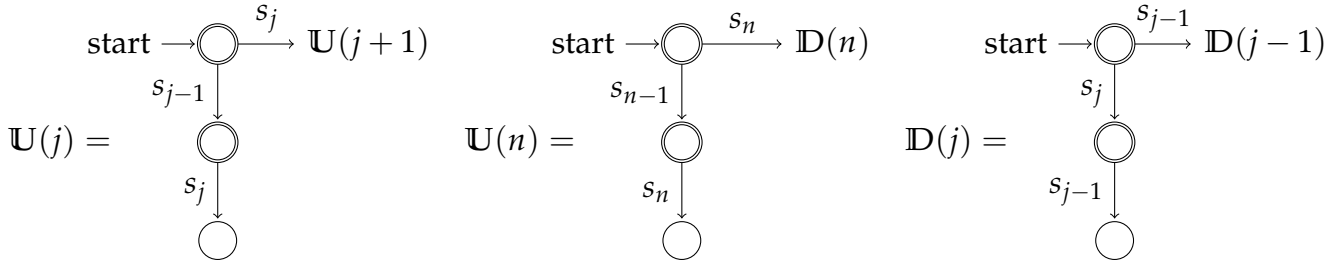


Figure 3: Recursive definition of the automata $\mathbb{U}(j)$ and $\mathbb{D}(j)$ in type B.

6.2 Type D

Type D is an example of a simply-laced case, meaning that $m_{i,j} \leq 3$ for all i and j. In this case, we see a "branching" phenomenon and it appears that a single oriented edge leads to multiple automata which need to be intersected and follow a more technical construction than for type A. As of yet, we know no general construction but we show in Figure 4 a working example (checked by computer exploration) for the case D_4.

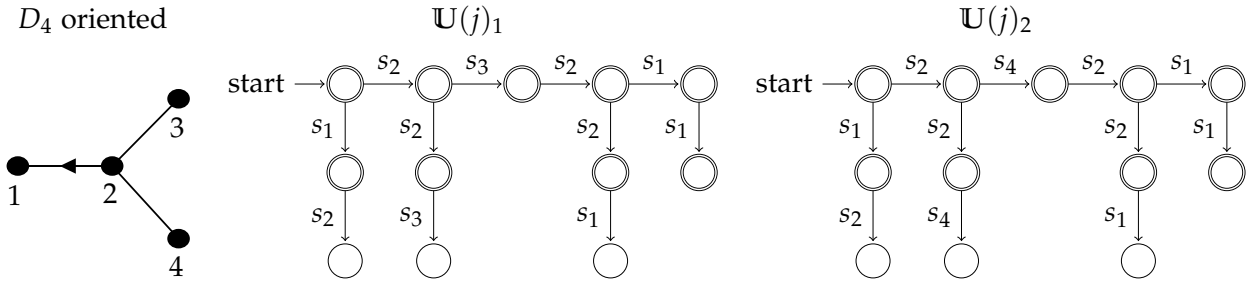


Figure 4: A partial orientation of D_4 and the corresponding automata that form $\mathbb{U}(2)$.

References

[1] G. Chatel and V. Pilaud. "Cambrian Hopf Algebras". *Adv. Math.* **311** (2017), pp. 598–633.

[2] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Co., Reading, Mass., 1979, pp. x+418.

[3] D. E. Knuth. *The art of computer programming. Vol. 1: Fundamental algorithms*. Second printing. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont, 1969, pp. xxi+634.

[4] V. Pilaud and V. Pons. "Permutrees". *Algebraic Combinatorics* **1**.2 (2018), pp. 173–224.

[5] V. Pilaud, V. Pons, and D. T. Jiménez. "Permutree sorting". 2020. arXiv:2007.07802.

[6] N. Reading. "Lattice congruences of the weak order". *Order* **21**.4 (2004), pp. 315–344.

[7] N. Reading. "Cambrian lattices". *Adv. Math.* **205**.2 (2006), pp. 313–353.

[8] N. Reading. "Clusters, Coxeter-sortable elements and noncrossing partitions". *Trans. Amer. Math. Soc.* **359**.12 (2007), pp. 5931–5958.

[9] N. Reading. "Sortable elements and Cambrian lattices". *Algebra Universalis* **56**.3-4 (2007), pp. 411–437.