Depth for classical Coxeter groups

Eli Bagno (Jerusalem College of Technology)
joint work with Riccardo Biagioli (Lyon 1), Alex Woo (U. Idaho), Mordechai Novick (Jerusalem College of Tech.)

Hit Combinatorics Seminar,
May 3, 2015

Exceedances in S_n

Definition

Let $w \in S_n$.
$i \in [n]$ is an *exceedance* of w if $w(i) > i$.

$\mathrm{exc}(w)=$ number of exceedances in w.

Definition

For an exceedance i of w, the **exceedance size** of i is $w(i) - i$.

Example

$$w = \begin{pmatrix} \textcircled{1} & \textcircled{2} & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix}$$

The exceedances $1, 2$ are circled. The exceedance sizes are $2 - 1 = 1$ and $4 - 2 = 2$.

Descents in S_n

Definition

Let $w \in S_n$.
$i \in \{1, \ldots, n-1\}$ is called a **descent** of w if $w(i) > w(i+1)$.

$\mathrm{des}(w)$ is the number of descents in w.

Example

$$w = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix}$$

$des(w) = 2$, 2 and 3 are descents.

Sorting by transpositions

One can imagine various "machines" that can sort permutations (to the identity) by swapping pairs of entries.

Machine ℓ: Can only swap adjacent entries, and every move costs 1.

Machine **a**: Can swap arbitrary pairs of entries, and every move costs 1.

Machine **d**: Can swap arbitrary pairs of entries, and a move costs the distance between the entries.

Question: Can we look at a permutation and easily tell the minimum cost to sort it?

Inversions

For Machine ℓ, the answer is called the **length** of the permutation, and it is equal to the number of inversions. One optimal algorithm is to always swap the rightmost descent.

For $w = 2537146$, we have

$$2537146 \to 2531746 \to 2531476 \to 2531467 \to 2513467$$
$$\to 2153467 \to 2135467 \to 2134567 \to 1234567$$

So $\ell(w) = 8$, and we have $1 + 3 + 1 + 3 = 8$ inversions.

Inversions

For Machine ℓ, the answer is called the **length** of the permutation, and it is equal to the number of inversions. One optimal algorithm is to always swap the rightmost descent.

For $w = 2537146$, we have

$$253\mathbf{71}46 \to 2531\mathbf{74}6 \to 25314\mathbf{76} \to 25\mathbf{31}467 \to 2\mathbf{51}3467$$
$$\to 21\mathbf{53}467 \to 213\mathbf{54}67 \to \mathbf{21}34567 \to 1234567$$

So $\ell(w) = 8$, and we have $1 + 3 + 1 + 3 = 8$ inversions.

Inversions

For Machine ℓ, the answer is called the **length** of the permutation, and it is equal to the number of inversions. One optimal algorithm is to always swap the rightmost descent.

For $w = 2537146$, we have

$$253\mathbf{71}46 \to 2531\mathbf{74}6 \to 25314\mathbf{76} \to 25\mathbf{31}467 \to 2\mathbf{51}3467$$
$$\to 21\mathbf{53}467 \to 213\mathbf{54}67 \to \mathbf{21}34567 \to 1234567$$

So $\ell(w) = 8$, and we have $1 + 3 + 1 + 3 = 8$ inversions.

Cycles

For Machine **a**, the answer is called the **absolute length** or **reflection length**, and it is equal to n minus the number of cycles. One optimal algorithm is to always swap the rightmost exceedance to its correct location.

For $w = 2537146$, we have

$$253\mathbf{7}14\mathbf{6} \to 253\mathbf{6}14\mathbf{7} \to 2\mathbf{5}34167 \to \mathbf{21}34567 \to 1234567$$

So $a(w) = 4$. We have $n = 7$ and 3 cycles, since $w = (125)(476)(3)$.

Cycles

For Machine **a**, the answer is called the **absolute length** or **reflection length**, and it is equal to n minus the number of cycles. One optimal algorithm is to always swap the rightmost exceedance to its correct location.

For $w = 2537146$, we have

$$253\mathbf{7}14\mathbf{6} \to 253\mathbf{6}147 \to 2\mathbf{5}34167 \to \mathbf{21}34567 \to 1234567$$

So $a(w) = 4$. We have $n = 7$ and 3 cycles, since $w = (125)(476)(3)$.

Cycles

For Machine **a**, the answer is called the **absolute length** or **reflection length**, and it is equal to n minus the number of cycles. One optimal algorithm is to always swap the rightmost exceedance to its correct location.

For $w = 2537146$, we have

$$2537146 \to 2536147 \to 2534167 \to 2134567 \to 1234567$$

So $a(w) = 4$. We have $n = 7$ and 3 cycles, since $w = (125)(476)(3)$.

Cycles

For Machine **a**, the answer is called the **absolute length** or **reflection length**, and it is equal to n minus the number of cycles. One optimal algorithm is to always swap the rightmost exceedance to its correct location.

For $w = 2537146$, we have

$$253\mathbf{7}14\mathbf{6} \to 253\mathbf{6}14\mathbf{7} \to 2\mathbf{5}3\mathbf{4}167 \to \mathbf{2}1\mathbf{3}4567 \to 1234567$$

So $a(w) = 4$. We have $n = 7$ and 3 cycles, since $w = (125)(476)(3)$.

Cycles

For Machine **a**, the answer is called the **absolute length** or **reflection length**, and it is equal to n minus the number of cycles. One optimal algorithm is to always swap the rightmost exceedance to its correct location.

For $w = 2537146$, we have

$$253\mathbf{7}14\mathbf{6} \rightarrow 253\mathbf{6}14\mathbf{7} \rightarrow 2\mathbf{5}34\mathbf{1}67 \rightarrow \mathbf{21}34567 \rightarrow 1234567$$

So $a(w) = 4$. We have $n = 7$ and 3 cycles, since $w = (125)(476)(3)$.

Cycles

For Machine **a**, the answer is called the **absolute length** or **reflection length**, and it is equal to n minus the number of cycles. One optimal algorithm is to always swap the rightmost exceedance to its correct location.

For $w = 2537146$, we have

$$253\mathbf{7}14\mathbf{6} \to 253\mathbf{6}147 \to 2\mathbf{5}34167 \to \mathbf{21}34567 \to 1234567$$

So $a(w) = 4$. We have $n = 7$ and 3 cycles, since $w = (125)(476)(3)$.

Cycles

For Machine **a**, the answer is called the **absolute length** or **reflection length**, and it is equal to n minus the number of cycles. One optimal algorithm is to always swap the rightmost exceedance to its correct location.

For $w = 2537146$, we have

$$253\mathbf{7}14\mathbf{6} \to 253\mathbf{6}14\mathbf{7} \to 2\mathbf{5}34\mathbf{1}67 \to \mathbf{21}34567 \to 1234567$$

So $a(w) = 4$. We have $n = 7$ and 3 cycles, since $w = (125)(476)(3)$.

Total displacement

For Machine **d**, the answer is called the **depth**, and Petersen–Tenner showed that it is equal to total displacement, which is the **sum of the sizes of exceedances**. One optimal algorithm is to always swap the rightmost exceedance with the leftmost non-exceedance to its right.

For $w = 2537146$, we have

$$25\mathbf{3}\mathbf{7}\mathbf{1}46 \to 2531\mathbf{7}4\mathbf{6} \to 25314\mathbf{7}\mathbf{6} \to 2\mathbf{5}3\mathbf{1}467 \to 213\mathbf{5}\mathbf{4}67 \to \mathbf{2}\mathbf{1}34567 \to 1234567$$

So $dp(w) = 7$, and the sum of sizes of exceedances is $1 + 3 + 0 + 3 + 0 + 0 + 0 = 7$.

Total displacement

For Machine **d**, the answer is called the **depth**, and Petersen–Tenner showed that it is equal to total displacement, which is the **sum of the sizes of exceedances**. One optimal algorithm is to always swap the rightmost exceedance with the leftmost non-exceedance to its right.

For $w = 2537146$, we have

$$25\mathbf{7}\mathbf{1}46 \to 2531\mathbf{7}4\mathbf{6} \to 25314\mathbf{7}\mathbf{6} \to 2\mathbf{5}3\mathbf{1}467 \to 213\mathbf{5}\mathbf{4}67$$
$$\to \mathbf{2}\mathbf{1}34567 \to 1234567$$

So $dp(w) = 7$, and the sum of sizes of exceedances is $1 + 3 + 0 + 3 + 0 + 0 + 0 = 7$.

Total displacement

For Machine **d**, the answer is called the **depth**, and Petersen–Tenner showed that it is equal to total displacement, which is the **sum of the sizes of exceedances**. One optimal algorithm is to always swap the rightmost exceedance with the leftmost non-exceedance to its right.

For $w = 2537146$, we have

$$253\mathbf{7}\mathbf{1}46 \to 2531\mathbf{7}4\mathbf{6} \to 25314\mathbf{7}\mathbf{6} \to 2\mathbf{5}3\mathbf{1}467 \to 213\mathbf{5}\mathbf{4}67 \to \mathbf{2}\mathbf{1}34567 \to 1234567$$

So $dp(w) = 7$, and the sum of sizes of exceedances is $1 + 3 + 0 + 3 + 0 + 0 + 0 = 7$.

From 'machine theory' back to group theory

Let (W, S) be a Coxeter system where W is a Coxeter group and S is its simple reflections set.
Let $T = \{wsw^{-1} \mid w \in W, s \in S\}$ be the set of reflections of W.
Using the language of group theory the definition of $\ell(w)$ and $a(w)$ will be

$$\ell(w) = \min_{w=s_1\cdots s_k} k$$

and

$$\mathbf{a}(w) = \min_{w=t_1\cdots t_k} k.$$

where we take the minima over all ways of writing w as a product of simple reflections s_i or reflections t_i respectively.

Depth in terms of roots

Definition

Let $\Phi = \Pi \cup -\Pi$ denote the roots system of W.
Let $\Delta = \{\alpha_1, \dots, \}$ denote the set of simple roots.

The **depth** of a positive root $\beta \in \Pi$ is

$$dp(\beta) = min\{k \mid s_1 \cdots s_k(\beta) \in -\Pi, s_i \in S\}.$$

For any group element $w \in W$, $dp(w)$ is the minimum sum of depths in a factorization of w into reflections.

Explicitly:

$dp(w) = min\{\sum_i dp(\beta_i) \mid w = t_{\beta_1} \cdots t_{\beta_k}, \beta_i \in T\}$,
where t_{β_i} is the reflection corresponding to the root β_i.

Comparing the machines

Petersen and Tenner observed that the cost for Machine **d** of swapping a **single** pair is exactly the average of the cost of Machine **a** (which is 1) and the cost of simulating that swap with Machine ℓ.

Explicitly:

For each reflection t

$$dp(t) = \frac{\ell(t)+1}{2} = \frac{\ell(t)+a(t)}{2}.$$

Comparing the machines

Petersen and Tenner observed that the cost for Machine **d** of swapping a **single** pair is exactly the average of the cost of Machine **a** (which is 1) and the cost of simulating that swap with Machine ℓ.

Explicitly:

For each reflection t

$$dp(t) = \frac{\ell(t)+1}{2} = \frac{\ell(t)+a(t)}{2}.$$

Comparing the machines

Hence

$$\frac{a(w) + \ell(w)}{2} \leq dp(w) \leq \ell(w).$$

The first inequality is not an equality because the most efficient method for Machine **d**, when simulated by Machine ℓ, might not be most efficient for Machine ℓ (or for Machine **a**).

Cost Coincidences

The permutations for which $dp(w) = \ell(w)$ are the 321 avoiding permutations. (Petersen–Tenner)

The permutations for which $dp(w) = a(w)$ (and hence $a(w) = \ell(w)$) are the 321 and 3412 avoiding permutations. (Tenner)

The permutations for which $dp(w) = \frac{a(w)+\ell(w)}{2}$ are not characterized by pattern avoidance (BiSC came up with nothing reasonable), and this seems like a hard problem.

Cost Coincidences

The permutations for which $dp(w) = \ell(w)$ are the 321 avoiding permutations. (Petersen–Tenner)

The permutations for which $dp(w) = a(w)$ (and hence $a(w) = \ell(w)$) are the 321 and 3412 avoiding permutations. (Tenner)

The permutations for which $dp(w) = \frac{a(w)+\ell(w)}{2}$ are not characterized by pattern avoidance (BiSC came up with nothing reasonable), and this seems like a hard problem.

Cost Coincidences

The permutations for which $dp(w) = \ell(w)$ are the 321 avoiding permutations. (Petersen–Tenner)

The permutations for which $dp(w) = a(w)$ (and hence $a(w) = \ell(w)$) are the 321 and 3412 avoiding permutations. (Tenner)

The permutations for which $dp(w) = \frac{a(w)+\ell(w)}{2}$ are not characterized by pattern avoidance (BiSC came up with nothing reasonable), and this seems like a hard problem.

The group B_n

A **signed permutation** is a permutation w on the set $\{\pm 1, \ldots, \pm n\}$ with the property that $w(-i) = -w(i)$ for all i.

It suffices to specify $w(i)$ for $i > 0$, so we can think of a signed permutation as a permutation with the additional property that some of the entries can be possessed by negativity.

For example, we might have $w = 2\bar{4}3\bar{1}7\bar{5}6$. (To save space, we draw the negative signs on top of the numbers.)

Machines for B_n

Machine ℓ can swap two adjacent entries or change the sign of the leftmost entry.

Machine **a** can do:

Shuffling: swap a pair of entries at positions i and j . We denote this by t_{ij}.

Double unsigning: swap a pair of entries at positions i and j and change both signs. We denote this by $t_{\bar{i}j}$.

Single unsigning: change the sign of the entry at position i. We denote this by t_{ii}.

Machine **d** costs (by the Petersen–Tenner average rule):

- $j - i$ for a shuffling move.
- $i + j - 1$ for a double unsigning.
- i for a single unsigning.

Length for B_n-Brenti's formula

The cost for machine ℓ, which is the length of w, is the total count of the following:

- $Inv(w)$: Positions $i < j$ with $w(i) > w(j)$.
- $NSP(w)$: Positions $i < j$ with $w(i) + w(j) < 0$.
- $NEG(w)$: Positions i with $w(i) < 0$.

Example

For $w = 2\bar{4}3\bar{1}7\bar{5}6$, we have $\ell(w) = (3+1+2+1+2) + (2+3+1+1) + 3 = 19$, with sorting algorithm

$$2\bar{4}3\bar{1}\mathbf{7}\bar{\mathbf{5}}6 \to 2\bar{4}3\bar{1}\bar{5}\mathbf{76} \to 2\bar{4}3\bar{\mathbf{1}}\bar{\mathbf{5}}67 \to 2\bar{4}\mathbf{3}\bar{\mathbf{5}}\bar{1}67 \to \cdots \to \bar{\mathbf{5}}\bar{4}\bar{1}2367$$
$$\to 5\bar{4}\bar{1}2367 \to \cdots \to \bar{4}\bar{1}23567 \to 4\bar{1}23567 \to \cdots \to 1234567$$

Decomposition of permutations

Definition

Let $u \in B_k, v \in B_{n-k}$. Define the *direct sum* of u and v by:

$$(u \oplus v)(i) := \begin{cases} u(i) & i \in \{1, \dots, k\}; \\ v(i-k) + \mathrm{sgn}(v(i-k))k & i \in \{k+1, \dots, k+l\}. \end{cases}$$

Definition

A permutation $w \in B_n$ will be called *decomposable* if it can be expressed as a nontrivial direct sum of permutations and *indecomposable* otherwise.

Example

Let $w = w^1 \oplus w^2 \oplus w^3 = [4, \bar{3}, 1, 2, 7, 5, \bar{6}, 9, \bar{8}]$, where

$$w^1 = [4, \bar{3}, 1, 2],$$
$$w^2 = [3, 1, \bar{2}],$$
$$w^3 = [2, \bar{1}],$$

while

$$[\bar{8}, 1, 9, 3, 5, 2, \bar{6}, 4, 7]$$

is indecomposable.

Oddness of a signed permutation

Definition

Given a signed permutation w, define the **oddness** of w to be the number of blocks in the sum decomposition with an odd number of signed elements.

Example

- In the previous example,

$$w = [4, \bar{3}, 1, \bar{2}, 7, 5, \bar{6}, 9, \bar{8}]$$

  we have $o(w) = 3$.
- The negative identity $w = [\bar{1}, \cdots, \bar{n}]$ is the oddest element, with oddness $o(w) = n$.

Depth for a signed permutation

Theorem

We have the following formula for depth for B_n

$$d(w) = \left(\sum_{w(i)>i} (w(i) - i) \right) + \left(\sum_{w(i)<0} \left(|w(i)| - \frac{1}{2} \right) \right) + \frac{o(w)}{2}.$$

Depth in complete notation

If we write the permutation w in complete notation like in this example:

$$\begin{pmatrix} \bar{3} & \bar{2} & \bar{1} & 1 & 2 & 3 \\ 2 & \bar{1} & 3 & \bar{3} & 1 & \bar{2} \end{pmatrix}$$

and define $cexc(w)$ to be the sum of exceedances as a permutation of $\{\pm 1, \ldots, \pm n\}$ with the convention that each $i > 0$ with $w(i) < 0$ contributes $|w(i)| - (-i) - 1 = |w(i)| + i - 1$ then we have:

Theorem

$$d(w) = \frac{cexc(w) + o(w)}{2}.$$

Algorithm for signed permutations

To sort a signed permutation w using the minimum depth, we do the following to each block in the sum decomposition:

1. If possible apply a shuffling move to positions i and j, where $x = w(i)$ is the largest positive entry in w with $x > i$, and $y = w(j)$ is the smallest entry in w with $i < j \leq x$. Repeat this step until there is no positive entry $x = w(i)$ with $x > i$.
2. If there are at least two negative entries, apply a double unsigning move at positions i and j, where $x = w(i)$ and $y = w(j)$ are the two negative entries of largest absolute value in w, and go back to Step 1.
3. If there is one negative entry, apply a single unsigning move the negative entry, and go back to Step 1.

Algorithm for signed permutations

To sort a signed permutation w using the minimum depth, we do the following to each block in the sum decomposition:

1. If possible apply a shuffling move to positions i and j, where $x = w(i)$ is the largest positive entry in w with $x > i$, and $y = w(j)$ is the smallest entry in w with $i < j \leq x$. Repeat this step until there is no positive entry $x = w(i)$ with $x > i$.
2. If there are at least two negative entries, apply a double unsigning move at positions i and j, where $x = w(i)$ and $y = w(j)$ are the two negative entries of largest absolute value in w, and go back to Step 1.
3. If there is one negative entry, apply a single unsigning move the negative entry, and go back to Step 1.

Algorithm for signed permutations

To sort a signed permutation w using the minimum depth, we do the following to each block in the sum decomposition:

1. If possible apply a shuffling move to positions i and j, where $x = w(i)$ is the largest positive entry in w with $x > i$, and $y = w(j)$ is the smallest entry in w with $i < j \leq x$. Repeat this step until there is no positive entry $x = w(i)$ with $x > i$.
2. If there are at least two negative entries, apply a double unsigning move at positions i and j, where $x = w(i)$ and $y = w(j)$ are the two negative entries of largest absolute value in w, and go back to Step 1.
3. If there is one negative entry, apply a single unsigning move the negative entry, and go back to Step 1.

Algorithm for signed permutations

To sort a signed permutation w using the minimum depth, we do the following to each block in the sum decomposition:

1. If possible apply a shuffling move to positions i and j, where $x = w(i)$ is the largest positive entry in w with $x > i$, and $y = w(j)$ is the smallest entry in w with $i < j \leq x$. Repeat this step until there is no positive entry $x = w(i)$ with $x > i$.
2. If there are at least two negative entries, apply a double unsigning move at positions i and j, where $x = w(i)$ and $y = w(j)$ are the two negative entries of largest absolute value in w, and go back to Step 1.
3. If there is one negative entry, apply a single unsigning move the negative entry, and go back to Step 1.

Example

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, 9, \bar{4}, \bar{1}] \in B_9.$$

Our first step will be to shuffle entry 9 to position 9:

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \mathbf{9}, \bar{4}, \bar{1}] \stackrel{t_{78}}{\to} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \mathbf{9}, \bar{1}] \stackrel{t_{89}}{\to} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \bar{1}, 9].$$

Now apply Step 1, consecutively to entries 8, and 7:

$$[\bar{6}, \bar{3}, \bar{2}, \mathbf{8}, 7, 5, \bar{4}, \bar{1}, 9] \stackrel{t_{47}}{\to} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, 7, 5, \mathbf{8}, \bar{1}, 9] \stackrel{t_{78}}{\to} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \mathbf{7}, 5, \bar{1}, 8, 9]$$

$$\stackrel{t_{57}}{\to} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \bar{1}, 5, 7, 8, 9].$$

Example

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, 9, \bar{4}, \bar{1}] \in B_9.$$

Our first step will be to shuffle entry 9 to position 9:

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \mathbf{9}, \bar{4}, \bar{1}] \stackrel{t_{78}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \mathbf{9}, \bar{1}] \stackrel{t_{89}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \bar{1}, 9].$$

Now apply Step 1, consecutively to entries 8, and 7:

$$[\bar{6}, \bar{3}, \bar{2}, \mathbf{8}, 7, 5, \bar{4}, \bar{1}, 9] \stackrel{t_{47}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, 7, 5, \mathbf{8}, \bar{1}, 9] \stackrel{t_{78}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \mathbf{7}, 5, \bar{1}, 8, 9]$$

$$\stackrel{t_{57}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \bar{1}, 5, 7, 8, 9].$$

Example

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, 9, \bar{4}, \bar{1}] \in B_9.$$

Our first step will be to shuffle entry 9 to position 9:

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \mathbf{9}, \bar{4}, \bar{1}] \xrightarrow{t_{78}} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \mathbf{9}, \bar{1}] \xrightarrow{t_{89}} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \bar{1}, 9].$$

Now apply Step 1, consecutively to entries 8, and 7:

$$[\bar{6}, \bar{3}, \bar{2}, \mathbf{8}, 7, 5, \bar{4}, \bar{1}, 9] \xrightarrow{t_{47}} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, 7, 5, \mathbf{8}, \bar{1}, 9] \xrightarrow{t_{78}} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \mathbf{7}, 5, \bar{1}, 8, 9]$$

$$\xrightarrow{t_{57}} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \bar{1}, 5, 7, 8, 9].$$

Example

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, 9, \bar{4}, \bar{1}] \in B_9.$$

Our first step will be to shuffle entry 9 to position 9:

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \mathbf{9}, \bar{4}, \bar{1}] \stackrel{t_{78}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \mathbf{9}, \bar{1}] \stackrel{t_{89}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \bar{1}, 9].$$

Now apply Step 1, consecutively to entries 8, and 7:

$$[\bar{6}, \bar{3}, \bar{2}, \mathbf{8}, 7, 5, \bar{4}, \bar{1}, 9] \stackrel{t_{47}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, 7, 5, \mathbf{8}, \bar{1}, 9] \stackrel{t_{78}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \mathbf{7}, 5, \bar{1}, 8, 9]$$

$$\stackrel{t_{57}}{\rightarrow} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \bar{1}, 5, 7, 8, 9].$$

Example

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, 9, \bar{4}, \bar{1}] \in B_9.$$

Our first step will be to shuffle entry 9 to position 9:

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \mathbf{9}, \bar{4}, \bar{1}] \stackrel{t_{78}}{\to} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \mathbf{9}, \bar{1}] \stackrel{t_{89}}{\to} [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, \bar{4}, \bar{1}, 9].$$

Now apply Step 1, consecutively to entries 8, and 7:

$$[\bar{6}, \bar{3}, \bar{2}, \mathbf{8}, 7, 5, \bar{4}, \bar{1}, 9] \stackrel{t_{47}}{\to} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, 7, 5, \mathbf{8}, \bar{1}, 9] \stackrel{t_{78}}{\to} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \mathbf{7}, 5, \bar{1}, 8, 9]$$

$$\stackrel{t_{57}}{\to} [\bar{6}, \bar{3}, \bar{2}, \bar{4}, \bar{1}, 5, 7, 8, 9].$$

Example, Cont.

Now, none of the positive entries is located to the left of its natural position, so we proceed with Step 2 to unsign the two largest negative digits in absolute order (6 and 4).

$$[\bar{\mathbf{6}}, \bar{3}, \bar{2}, \bar{\mathbf{4}}, \bar{1}, 5, 7, 8, 9] \xrightarrow{t_{\bar{1}4}} [4, \bar{3}, \bar{2}, 6, \bar{1}, 5, 7, 8, 9].$$

Example, Cont.

We apply again Step 1 to push 6 and then 4 forward to their natural positions:

$$[4,\bar{3},\bar{2},\mathbf{6},\bar{1},5,7,8,9] \xrightarrow{t_{45}} [4,\bar{3},\bar{2},\bar{1},\mathbf{6},5,7,8,9] \xrightarrow{t_{56}} [\mathbf{4},\bar{3},\bar{2},\bar{1},5,6,7,8,9]$$

$$\xrightarrow{t_{12}} [\bar{3},\mathbf{4},\bar{2},\bar{1},5,6,7,8,9] \xrightarrow{t_{23}} [\bar{3},\bar{2},\mathbf{4},\bar{1},5,6,7,8,9] \xrightarrow{t_{34}} [\bar{3},\bar{2},\bar{1},4,5,6,7,8,9]$$

Example,Cont.

We unsign now the couple 3 and 2:

$$[\bar{3},\bar{2},\bar{1},4,5,6,7,8,9] \xrightarrow{t_{\bar{1}2}} [2,3,\bar{1},4,5,6,7,8,9].$$

Exapmle, Cont.

Now again Step 1, to move 3 and 2 to their natural positions:

$$[2, \mathbf{3}, \bar{1}, 4, 5, 6, 7, 8, 9] \xrightarrow{t_{23}} [\mathbf{2}, \bar{1}, 3, 4, 5, 6, 7, 8, 9] \xrightarrow{t_{12}} [\bar{1}, 2, 3, 4, 5, 6, 7, 8, 9].$$

Example, Cont.

Finally we unsign 1:

$$[\bar{\mathbf{1}},2,3,4,5,6,7,8,9] \xrightarrow{t_{\bar{1}1}} [1,2,3,4,5,6,7,8,9],$$

and we are done.

We obtained $w = t_{\bar{1}1}t_{12}t_{23}t_{\bar{1}2}t_{34}t_{23}t_{12}t_{56}t_{45}t_{\bar{1}4}t_{57}t_{78}t_{47}t_{89}t_{78}$. The sum of depths of the reflection in the decomposition is 22, and corresponds to

$d(w) = (8-4)+(7-5)+(9-7)-(-6-3-2-4-1)-\lfloor \frac{5}{2} \rfloor$, since w is indecomposable.

Example, Cont.

Finally we unsign 1:

$$[\bar{\mathbf{1}}, 2, 3, 4, 5, 6, 7, 8, 9] \xrightarrow{t_{\bar{1}1}} [1, 2, 3, 4, 5, 6, 7, 8, 9],$$

and we are done.

We obtained $w = t_{\bar{1}1} t_{12} t_{23} t_{\bar{1}2} t_{34} t_{23} t_{12} t_{56} t_{45} t_{\bar{1}4} t_{57} t_{78} t_{47} t_{89} t_{78}$. The sum of depths of the reflection in the decomposition is 22, and corresponds to

$d(w) = (8-4) + (7-5) + (9-7) - (-6-3-2-4-1) - \lfloor \frac{5}{2} \rfloor$,

since w is indecomposable.

The bill

$$w = [\bar{6}, \bar{3}, \bar{2}, 8, 7, 5, 9, \bar{4}, \bar{1}] \in B_9.$$

- In w, 9 is two places away from its natural position, so $9 - w^{-1}(9) = 2$. This is the cost we paid when moving 9 to its place.
- 8 and 7 contribute 4 and 2, respectively.
- The treatment of the pair 6 and 4, from the unsigning process to the arrival at their natural positions costs $6 + 4 - 1 = 9$.
- The treatment of 2 and 3 costs $2 + 3 - 1 = 4$.
- The unsigning of 1 costs 1.

$$d(w) = 2 + 4 + 2 + 9 + 4 + 1 = 22.$$

Sketch of proof of the formula

We show that our algorithm achieves our formula for $dp(w)$, and that any other sequence of reflections costs more.

It suffices by induction to show that a single step of our algorithm reduces our conjectured formula for the depth by the right amount, and that no move can reduce our conjectured formula by more.

Explicitly, for each $w \in B_n$ and $t \in T$ which is being used in the course of the algorithm one has:

$$d(w) - d(wt) \leq dp(t).$$

The maximum depth

We can use the algorithm described above to obtain the maximum value of the depth on B_n.

Theorem

For each $\pi \in B_n$ we have $dp(\pi) \leq \binom{n+1}{2}$ with equality if and only if $\pi = [\bar{1}, \bar{2}, \cdots, \bar{n}]$.

Reduced products

Let $\pi \in B_n$ be indecomposable. Use the algorithm to write π as a product of reflections

$$\pi = t_1 \cdots t_u.$$

For each $k \in \{1, \ldots, u\}$ write t_k as a product of simple reflections in the following way:

- For $t_k = r_{ij}$:

$$t_k = s_{j-1} \cdots s_i s_{i+1} \cdots s_{j-1}.$$

- For $t_k = r_{\bar{i},j}$:

$$t_k = s_{i-1} \cdots s_1 s_{j-1} \cdots s_2 s_0 s_1 s_0 s_2 \cdots s_{j-1} s_1 s_2 \cdots s_{i-1}.$$

Theorem

The decomposition of π described above is reduced.

Let $S(\pi)$ be the multiset of all simple reflections appearing in this decomposition of π.

Proof:

The proof is based on a correspondence which will be formed between the multiset $S(\pi)$ and the multiset $Inv(\pi) \cup Neg(\pi) \cup Nsp(\pi)$ where :

$$Inv(\pi) = \{(\pi(i), \pi(j) \mid 1 \leq i < j \leq n, \pi(i) > \pi(j)\}.$$

$$Neg(\pi) = \{(i, -i) \mid -i = \pi(k), k > 0\}.$$

$$Nsp(\pi) = \{(\pi(i), \pi(j) \mid \pi(i) + \pi(j) < 0\}.$$

Theorem

The decomposition of π described above is reduced.

Let $S(\pi)$ be the multiset of all simple reflections appearing in this decomposition of π.

Proof:

The proof is based on a correspondence which will be formed between the multiset $S(\pi)$ and the multiset $Inv(\pi) \cup Neg(\pi) \cup Nsp(\pi)$ where :

$$Inv(\pi) = \{(\pi(i), \pi(j) \mid 1 \leq i < j \leq n, \pi(i) > \pi(j)\}.$$

$$Neg(\pi) = \{(i, -i) \mid -i = \pi(k), k > 0\}.$$

$$Nsp(\pi) = \{(\pi(i), \pi(j) \mid \pi(i) + \pi(j) < 0\}.$$

Shuffling moves

Each simple reflection in the decomposition of t_{ij}, used in Step 1 reverses either an inversion from $Inv(w)$ containing the larger element a in position i or one containing the smaller element b in position j (or containing both elements).

$$[\bar{6},\bar{3},\bar{2},\mathbf{8},7,5,\bar{\mathbf{4}},\bar{1},9] \xrightarrow{t_{47}} [\bar{6},\bar{3},\bar{2},\bar{\mathbf{4}},7,5,\mathbf{8},\bar{1},9].$$

Consequently, when we count toward the depth, we consider only the inversions of the form (a,k) for each k which is located between a and b.

When we count toward the length, we add also the inversions of the form (k,b) for each k which is located between a and b.

Shuffling moves

Each simple reflection in the decomposition of t_{ij}, used in Step 1 reverses either an inversion from $Inv(w)$ containing the larger element a in position i or one containing the smaller element b in position j (or containing both elements).

$$[\bar{6},\bar{3},\bar{2},\mathbf{8},7,5,\bar{\mathbf{4}},\bar{1},9] \xrightarrow{t_{47}} [\bar{6},\bar{3},\bar{2},\bar{\mathbf{4}},7,5,\mathbf{8},\bar{1},9].$$

Consequently, when we count toward the depth, we consider only the inversions of the form (a,k) for each k which is located between a and b.

When we count toward the length, we add also the inversions of the form (k,b) for each k which is located between a and b.

Shuffling moves

Each simple reflection in the decomposition of t_{ij}, used in Step 1 reverses either an inversion from $Inv(w)$ containing the larger element a in position i or one containing the smaller element b in position j (or containing both elements).

$$[\bar{6}, \bar{3}, \bar{2}, \mathbf{8}, 7, 5, \bar{\mathbf{4}}, \bar{1}, 9] \xrightarrow{t_{47}} [\bar{6}, \bar{3}, \bar{2}, \bar{\mathbf{4}}, 7, 5, \mathbf{8}, \bar{1}, 9].$$

Consequently, when we count toward the depth, we consider only the inversions of the form (a, k) for each k which is located between a and b.
When we count toward the length, we add also the inversions of the form (k, b) for each k which is located between a and b.

Example

$$[\bar{6},\bar{3},\bar{2},\mathbf{8},7,5,\bar{\mathbf{4}},\bar{1},9] \xrightarrow{t_{47}} [\bar{6},\bar{3},\bar{2},\bar{\mathbf{4}},7,5,\mathbf{8},\bar{1},9].$$

$$t_{47} = s_6 s_5 s_4 s_5 s_6.$$

The inverted pairs:

$$(8,7),(8,5),(8,\bar{4}),(7,\bar{4}),(5,\bar{4})$$

Only the black ones are counted toward the depth,

Unsigning moves

Let a and b be negative elements unsigned together in a move of the form $t_{\bar{i},j}$.
The corresponding reduced expression:

$$t_{\bar{i},j} = s_{i-1} \cdots s_1 s_{j-1} \cdots s_2 s_0 s_1 s_0 s_2 \cdots s_{j-1} s_1 s_2 \cdots s_{i-1}$$

Writing this as $ws_0s_1s_0w^{-1}$, each of w and w^{-1} represents one direction in which a and b are moved during the reflection (to the beginning of the permutation before unsigning, and back to their original positions after unsigning).
The reflections which form w plus both appearances of s_0 will count toward the depth and the rest will not.

Unsigning moves

Let a and b be negative elements unsigned together in a move of the form $t_{\bar{i},j}$.
The corresponding reduced expression:

$$t_{\bar{i},j} = s_{i-1} \cdots s_1 s_{j-1} \cdots s_2 s_0 s_1 s_0 s_2 \cdots s_{j-1} s_1 s_2 \cdots s_{i-1}$$

Writing this as $ws_0s_1s_0w^{-1}$, each of w and w^{-1} represents one direction in which a and b are moved during the reflection (to the beginning of the permutation before unsigning, and back to their original positions after unsigning).
The reflections which form w plus both appearances of s_0 will count toward the depth and the rest will not.

Unsigning moves

Let a and b be negative elements unsigned together in a move of the form $t_{\bar{i},j}$.
The corresponding reduced expression:

$$t_{\bar{i},j} = s_{i-1} \cdots s_1 s_{j-1} \cdots s_2 s_0 s_1 s_0 s_2 \cdots s_{j-1} s_1 s_2 \cdots s_{i-1}$$

Writing this as $ws_0 s_1 s_0 w^{-1}$, each of w and w^{-1} represents one direction in which a and b are moved during the reflection (to the beginning of the permutation before unsigning, and back to their original positions after unsigning).
The reflections which form w plus both appearances of s_0 will count toward the depth and the rest will not.

Example

$$w = [\bar{2}, \bar{\mathbf{5}}, \bar{1}, \bar{\mathbf{3}}, 4] \xrightarrow{t_{\bar{2}4}} [\bar{2}, \mathbf{3}, \bar{1}, \mathbf{5}, 4]$$

$$t_{\bar{2}4} = s_1 s_3 s_2 s_0 w_1 s_0 s_2 s_3 s_1.$$

The inversions of w involved here:

$$Inv(w) : s_1 : (\bar{2}, \bar{5}), s_3 : (\bar{3}, \bar{1}), s_2 : (\bar{2}, \bar{3}),$$

$$Neg(w) : s_0 : (5, \bar{5}),$$

$$Nsp(w) : s_1 : (\bar{5}, \bar{3}),$$

$$Neg(w) : s_0 : (3, \bar{3}),$$

$$Nsp(w) : s_2 : (\bar{2}, \bar{5}), s_3 : (\bar{5}, \bar{1}), s_1 : (\bar{2}, \bar{3}).$$

Every simple reflection in this decomposition reverses an inversion of π.

- The reflections which bring a and b to the beginning of the permutation: $s_{i-1} \cdots s_1 s_j s_{j-1} \cdots s_2$ reverse inversions of $Inv(\pi)$.
- The reflections which return a and b to their (reversed) places: $s_2 \cdots s_{j-1} s_1 s_2 \cdots s_{i-1}$ reverse inversions of $Nsp(\pi)$.
- The two instances of s_0 in the middle reverse inversions of $Neg(\pi)$,
- The s_1 between those two reverses the reflection (a, b) which belongs to $Inv(\pi)$ if $a > b$ or to $Nsp(\pi)$ otherwise.

Two Corollaries

Corollary

Each step of our algorithm reduces length *by* $\ell(t)$*.*

This means that simulating our algorithm (and hence one optimal use of Machine d) using Machine ℓ produces an optimal sort using Machine ℓ.

Corollary

Let $w \in B_n$ *with the decomposition* $w = t_1 \cdots t_u$*, given by the algorithm. If there is some* $k \in \{1, \ldots, u\}$ *such that* t_k *is not a simple reflection then* $\ell(\pi) > dp(\pi)$*.*

Depth reduced elements

Definition

Let W be a Coxeter group.

An Element $w \in W$ is called **depth reduced** if there exists an expression $w = t_1 \cdots t_k$ realizing the depth of w such that

$$\ell(w) = \sum_{i=1}^{k} \ell(t_i).$$

Such a decomposition for w is called a **reduced factorization**

By the algorithm described above, in B_n every element is depth reduced.

Reduced reflection length

Define the **reduced reflection length** $a'(w)$ as

$$a'(w) = \min_{w=t_1\cdots t_k} k.$$

where we take the minimum over the restricted set of products $w = t_1 \ldots t_k$ with $\ell(w) = \sum_{i=1}^{k} \ell(t_i)$.

Reduced reflection length

Define the **reduced reflection length** $a'(w)$ as

$$a'(w) = \min_{w=t_1\cdots t_k} k.$$

where we take the minimum over the restricted set of products $w = t_1 \dots t_k$ with $\ell(w) = \sum_{i=1}^{k} \ell(t_i)$.

Depth as average of reduced reflection length and the length

Recall that

$$dp(w) = \min_{t_1 \cdots t_k} \sum_{i=1}^{k} \frac{\ell(t_i)+1}{2}$$

.

Since depth can always be given by a reduced factorization, we may run only over reduced products so

$$dp(w) = \min_{t_1 \cdots t_k} \frac{\sum_{i=1}^{k} (\ell(t_i)+1)}{2} = \frac{a'(w)+\ell(w)}{2}.$$

where

Comparing length and depth

An element in a Coxeter group is **short-braid-avoiding** if no reduced decomposition (product of simple reflections realizing w) has a consecutive subexpression $s_i s_j s_i$.

It is easy to show that $d(w) = \ell(w)$ if and only if the depth of w is realized by a reduced factorization and w is short-braid-avoiding.

Since the depth is always realized by a reduced factorization in S_n, B_n, and D_n, this shows that $d(w) = \ell(w)$ in those groups if and only if w is short-braid-avoiding.

Comparing length and depth

An element in a Coxeter group is **short-braid-avoiding** if no reduced decomposition (product of simple reflections realizing w) has a consecutive subexpression $s_i s_j s_i$.

It is easy to show that $d(w) = \ell(w)$ if and only if the depth of w is realized by a reduced factorization and w is short-braid-avoiding.

Since the depth is always realized by a reduced factorization in S_n, B_n, and D_n, this shows that $d(w) = \ell(w)$ in those groups if and only if w is short-braid-avoiding.

Comparing length and depth

An element in a Coxeter group is **short-braid-avoiding** if no reduced decomposition (product of simple reflections realizing w) has a consecutive subexpression $s_i s_j s_i$.

It is easy to show that $d(w) = \ell(w)$ if and only if the depth of w is realized by a reduced factorization and w is short-braid-avoiding.

Since the depth is always realized by a reduced factorization in S_n, B_n, and D_n, this shows that $d(w) = \ell(w)$ in those groups if and only if w is short-braid-avoiding.

Short-braid-avoidance in B_n and D_n

For permutations, this reproves the Petersen–Tenner theorem that $d(w) = \ell(w)$ if and only if w avoids 321.

In B_n, short-braid-avoiding is equivalent to Stembridge's notion of **fully commutative top-and-bottom**, which is characterized by avoiding $1\bar{2}$, $\bar{1}\bar{2}$, $\bar{2}\bar{1}$, $\bar{3}21$, $\bar{3}2\bar{1}$, and 321

Short-braid-avoidance in B_n and D_n

For permutations, this reproves the Petersen–Tenner theorem that $d(w) = \ell(w)$ if and only if w avoids 321.

In B_n, short-braid-avoiding is equivalent to Stembridge's notion of **fully commutative top-and-bottom**, which is characterized by avoiding $1\bar{2}$, $\bar{1}\bar{2}$, $\bar{2}\bar{1}$, $\bar{3}21$, $\bar{3}2\bar{1}$, and 321

The group D_n

Definition

The group of signed permutations has an index 2 subgroup consisting of signed permutations with an even number of negative entries.

The double unsigning move swapping the leftmost entries is now a move for Machine ℓ, single unsigning moves are banned, and costs for double unsigning moves for Machine $\mathbf{d}$ go down by 1.

Explicitly:

Machine **d** costs:

- $j - i$ for a shuffling move t_{ij}
- $i + j - 2$ for double unsigning $t_{\bar{i}j}$.

Sum decomposition for D_n

For D_n, we need to distinguish between two types of sum decomposition. A **type D decomposition** requires that each block have an even number of negative entries, while a **type B decomposition** does not.

Example

$$w = [\bar{2}, 1, 3, 4, \bar{5}, \bar{7}, \bar{8}].$$

The type D decomposition is

$$w = [\bar{2}, 1, 3, 4, \bar{5}] \oplus [\bar{2}, \bar{3}, 1].$$

while the type B decomposition has

$$w = [\bar{2}, 1] \oplus [1] \oplus [1] \oplus [\bar{1}] \oplus [\bar{2}, \bar{3}, 1].$$

Definition

Define $r(w)$ as the number of type B blocks minus the number of type D blocks

Example

$$w = [\bar{2}, 1, 3, 4, \bar{5}] \oplus [\bar{2}, \bar{3}, 1].$$

$$w = [\bar{2}, 1] \oplus [1] \oplus [1] \oplus [\bar{1}] \oplus [\bar{2}, \bar{3}, 1].$$

so $r(w) = 3$.

Depth for an even signed permutation

We have the following formula for depth for D_n

$$dp(w) = \left(\sum_{w(i)>i} (w(i) - i) \right) + \left(\sum_{w(i)<0} |w(i)| - 1 \right) + r(w).$$

The last term counts the "wasted" moves that are needed to join type B blocks so that we can perform the needed double unsigning moves.

Depth in complete notation

If we write the permutation w in complete notation like in this example:

$$\begin{pmatrix} \bar{3} & \bar{2} & \bar{1} & 1 & 2 & 3 \\ 2 & \bar{1} & 3 & \bar{3} & 1 & \bar{2} \end{pmatrix}$$

and define $cexc(w)$ to be the sum of exceedances as a permutation of $\{\pm 1, \ldots, \pm n\}$ with the convention that each $i > 0$ with $w(i) < 0$ contributes $|w(i)| - (-i) - \mathbf{2} = |w(i)| + i - \mathbf{2}$ then we have:

Theorem

$$d(w) = \frac{cexc(w)}{2} + r(w).$$

The algorithm for D_n

Let $w \in D_n$ be D indecomposable.

1. Apply shuffling moves only to the entries of the rightmost B block.
2. After the previous step, the first entry in the rightmost block must be negative.
   If there is than one type B block, apply a depth 1 shuffling move to switch the first entry in the rightmost block and the entry to its left.
   This joins the two rightmost type B blocks into one type B block. The rightmost type B block is now bigger, and we go back to step 1.
3. Apply a double unsigning move, then further shuffling and double unsigining moves as in type B algorithm.

The algorithm for D_n

Let $w \in D_n$ be D indecomposable.

1. Apply shuffling moves only to the entries of the rightmost B block.
2. After the previous step, the first entry in the rightmost block must be negative.
   If there is than one type B block, apply a depth 1 shuffling move to switch the first entry in the rightmost block and the entry to its left.
   This joins the two rightmost type B blocks into one type B block. The rightmost type B block is now bigger, and we go back to step 1.
3. Apply a double unsigning move, then further shuffling and double unsigining moves as in type B algorithm.

The algorithm for D_n

Let $w \in D_n$ be D indecomposable.

1. Apply shuffling moves only to the entries of the rightmost B block.
2. After the previous step, the first entry in the rightmost block must be negative.
   If there is than one type B block, apply a depth 1 shuffling move to switch the first entry in the rightmost block and the entry to its left.
   This joins the two rightmost type B blocks into one type B block. The rightmost type B block is now bigger, and we go back to step 1.
3. Apply a double unsigning move, then further shuffling and double unsigining moves as in type B algorithm.

The algorithm for D_n

Let $w \in D_n$ be D indecomposable.

1. Apply shuffling moves only to the entries of the rightmost B block.
2. After the previous step, the first entry in the rightmost block must be negative.
   If there is than one type B block, apply a depth 1 shuffling move to switch the first entry in the rightmost block and the entry to its left.
   This joins the two rightmost type B blocks into one type B block. The rightmost type B block is now bigger, and we go back to step 1.
3. Apply a double unsigning move, then further shuffling and double unsigining moves as in type B algorithm.

Example

$$[3,\bar{1},2,6,\bar{5},4] \xrightarrow{t_{45}} [3,\bar{1},2,\bar{5},6,4] \xrightarrow{t_{56}} [3,\bar{1},2,\bar{5},4,6]$$

Unite two blocks:

$$\xrightarrow{t_{34}} [3,\bar{1},\bar{5},2,4,6]$$

Shuffle inside united block:

$$\xrightarrow{t_{13}} [\bar{5},\bar{1},3,2,4,6]$$

Then double unsign:

$$\xrightarrow{t_{\bar{1}2}} [1,5,3,2,4,6]$$

and shuffle toward the end:

$$\xrightarrow{t_{24}} [1,2,3,5,4,6] \xrightarrow{t_{45}} [1,2,3,4,5,6]$$

Achieving the lower bound

The elements for which $a(w) = dp(w)$ (and hence both are equal to $\ell(w)$) are the **boolean elements**, where no reduced decomposition has any simple reflection more than once. These are characterized by avoiding 10 patterns for B_n and 20 for D_n (Tenner).

The more general question of when $d(w) = \frac{a(w)+\ell(w)}{2}$ seems hard and is not characterized by pattern avoidance.

Achieving the lower bound

The elements for which $a(w) = dp(w)$ (and hence both are equal to $\ell(w)$) are the **boolean elements**, where no reduced decomposition has any simple reflection more than once. These are characterized by avoiding 10 patterns for B_n and 20 for D_n (Tenner).

The more general question of when $d(w) = \frac{a(w)+\ell(w)}{2}$ seems hard and is not characterized by pattern avoidance.

Problems

- Is depth realized by a reduced factorization into transpositions for all elements in all Coxeter groups?
- Can depth be realized by a product of $a(w)$ reflections for B_n (For S_n the answer is positive, for D_n its negative).
- Find the generating function for depth in B_n or D_n (See Guay-Paquet–Petersen for S_n)
- Characterize depth for affine Coxeter groups

Thank you

Thank you for your attention!