Depth for classical Coxeter groups

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

AMS Eastern Sectional Meeting,
March 7, 2015

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.

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

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

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.

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 excedence to its correct location.

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 excedence to its correct location.

For $w = 2537146$, we have

$$253\mathbf{7}14\mathbf{6} \to 253\mathbf{6}1\mathbf{4}7 \to 2\mathbf{5}34\mathbf{1}67 \to \mathbf{21}34567 \to 1234567$$

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 excedence 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 it is equal to total displacement, which is the sum of the sizes of excedences. One optimal algorithm is to always swap the rightmost excedence with the leftmost sub-excedence to its right.

Total displacement

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

For $w = 2537146$, we have

$$25\mathbf{71}46 \to 2531\mathbf{74}6 \to 25314\mathbf{76} \to 2\mathbf{5}3\mathbf{1}467 \to 213\mathbf{54}67$$
$$\to \mathbf{21}34567 \to 1234567$$

Total displacement

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

For $w = 2537146$, we have

$$253\mathbf{71}46 \to 2531\mathbf{74}6 \to 25314\mathbf{76} \to 2\mathbf{531}467 \to 213\mathbf{54}67 \\ \to \mathbf{21}34567 \to 1234567$$

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

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.

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.

Hence

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

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 $d(w) = \ell(w)$ are the 321 avoiding permutations. (Petersen–Tenner)

Cost Coincidences

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

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

Cost Coincidences

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

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

The permutations for which $d(w) = (a(w) + \ell(w))/2$ is not characterized by (mesh) 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 $\{\pm 1, \dots, \pm n\}$ 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

Shuffling: swap a pair of entries at positions i and j

Double unsigning: swap a pair of entries at positions i and j and change both signs

Single unsigning: change the sign of the entry at position i

Machine d costs (by the Petersen–Tenner average rule) the $j-i$ for a shuffling move, $i+j-1$ for a double unsigning, and i for a single unsigning. (Intuition: There is a neutral chaotic exorcist at the far left that changes signs, so unsigning moves need to swing the letters through that spot.)

Length for B_n

The cost for machine ℓ is the total count of the following:

- Positions $i < j$ with $w(i) > w(j)$
- Positions $i < j$ with $w(i) + w(j) < 0$
- Positions i with $w(i) < 0$

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

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

Oddness of a signed permutation

We can have a sum $\oplus$ of signed permutations and sum decompositions defined by ignoring the signs. For example, $2\bar{4}3\bar{1}7\bar{5}6 = 2\bar{4}3\bar{1} \oplus 3\bar{1}2$ is the sum decomposition.

Oddness of a signed permutation

We can have a sum $\oplus$ of signed permutations and sum decompositions defined by ignoring the signs. For example, $2\bar{4}3\bar{1}7\bar{5}6 = 2\bar{4}3\bar{1} \oplus 3\bar{1}2$ is the sum decomposition.

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.

The negative identity $\bar{1} \cdots \bar{n}$ is the oddest element, with oddness n.

Depth for a signed permutation

We have the following formula for depth for B_n

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

If we write all the entries of the signed permutations, including the ones at negative positions, the two left terms are half the sum of sizes of excedences. Single unsigning moves are slightly expensive, and $o(w)$ counts how many times they need to be used.

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:

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 $x > i$.

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.

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 example

For $w = 2\bar{4}3\bar{1}7\bar{5}6$, the formula gives
$d(w) = (1+2) + (4+1+5-3/2) + 1/2 = 12$

$$2\bar{4}3\bar{1}\mathbf{7}\bar{\mathbf{5}}6 \xrightarrow{1} 2\bar{4}3\bar{1}\bar{5}\mathbf{7}\mathbf{6} \xrightarrow{1} 2\bar{4}3\bar{1}\bar{\mathbf{5}}67 \xrightarrow{5} \mathbf{2}\bar{\mathbf{4}}3\bar{1}567 \xrightarrow{1} \bar{\mathbf{4}}23\bar{\mathbf{1}}567 \xrightarrow{4} 1234567$$

Sketch of proof of formula

We show that our algorithm achieves our formula for $d(w)$, 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 d by the right amount, and that no move can reduce our conjectured formula by more.

An important property

It turns out each step of our algorithm reduces *length* by $\ell(t)$.

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

The group D_n

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 d go down by 1.

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.

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.

If $w = \bar{2}1345\bar{5}\bar{7}\bar{8}6$, then the type D decomposition is $w = \bar{2}134\bar{5} \oplus \bar{2}\bar{3}1$, while the type B decomposition has $w = \bar{2}1 \oplus 1 \oplus 1 \oplus \bar{1} \oplus \bar{2}\bar{3}1$.

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.

If $w = \bar{2}134\bar{5}\bar{7}\bar{8}6$, then the type D decomposition is $w = \bar{2}134\bar{5} \oplus \bar{2}\bar{3}1$, while the type B decomposition has $w = \bar{2}1 \oplus 1 \oplus 1 \oplus \bar{1} \oplus \bar{2}\bar{3}1$.

Define $o(w)$ as the number of type B blocks minus the number of type D blocks (so $o(w) = 3$).

Depth for an even signed permutation

We have the following formula for depth for D_n

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

If we write all the entries of the signed permutations, including the ones at negative positions, the two left terms are half the sum of sizes of excedences. 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.

Minimizing over products

We can rephrase the definition of $\ell(w)$ and $a(w)$ as

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

and

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

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

Reduced products

We can rephrase the definition of $d(w)$ as

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

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

Reduced products

We can rephrase the definition of $d(w)$ as

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

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

The "important property" means that it is always possible (for classical groups) to restrict to **reduced factorizations**, meaning factorizations $w = t_1 \dots t_k$ where

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

Reduced reflection length

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

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

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

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

where we take the minimum over the restricted set of products $w = t_1 \ldots t_k$ with

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

Since depth can always be given by a reduced factorization, we have

$$d(w) = \frac{a'(w) + \ell(w)}{2}.$$

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 $s_i s_j s_i$.

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.

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 $1\overline{2}$, $\overline{1}\overline{2}$, $\overline{2}\overline{1}$, $\overline{3}21$, $\overline{3}2\overline{1}$, and 321

In D_n (and any simply-laced group), short-braid-avoiding is equivalent to being fully commutative, which is characterized by Billey-Postnikov avoiding 321. (This is avoiding 321 as a permutation of $\{\pm 1, \ldots, \pm n\}$, not allowing the simultaneous use of opposite entries.)

Achieving the lower bound

The elements for which $a(w) = d(w)$ (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 $d(w) = (a(w) + \ell(w))/2$ 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 (even for B_n or D_n)?
- 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!