Section 3.4 is devoted to computational aspects of Coxeter groups via the titular “normal form”, which is the lexicographically smallest word that represents a given group element \((s_1 < s_2 < \cdots < s_n)\). In this document, we will write \(w_\ast\) for \(w_{[n-1]}\) and \(\ast w\) for \([n-1]w\), when \(n\) is clear from context, and we write \([w]\) for the normal form of \(w\).

The entire point of the section is the statement of Theorem 3.4.7. Using the rather less cluttered notation described above, we may write it as

**Theorem (3.4.7).** Let \((W, S)\) be a Coxeter group of rank \(n\), with \(v \in W\) and \(s \in S\). Then there exists \(j \in [n - 1]\) such that

\[
[v s] = \begin{cases} 
[v_s] s_n [s_n^\ast v s] & \text{if } \ast v s \neq e \text{ is distinguished}, \\
[v_s] s_j [s_j^\ast v] & \text{if } \ast v s \text{ is not distinguished}, \\
[v_s] s & \text{if } \ast v = s = s_n, \\
(s) & \text{if } v = e.
\end{cases}
\]

As the authors remark in the text, this theorem does not quite have all the pieces to create an algorithm: missing in particular are how to check whether an arbitrary group element \(w s\) is in the right-quotient \([n-1]W\), and if not, the explicit computation of \(j\).

It is worth noting that the first problem can be solved in a somewhat inefficient way, which are nevertheless not horrible if we could compute lengths in constant time (i.e. an oracle for \(\ell\)). Namely, we can use the naïve algorithm of multiplying by all possible \(t \in J\) and checking if any of them decrease the length. Notice that in type A we do have such an oracle, since \(\ell(\sigma)\) is the number of inversions.

The latter problem has no such recourse: the naïve algorithm is of course to try all the \(s'\) to see which one works, but to see whether “it works”, you have to solve the word problem in \(W\). But if we could do that, then there really wouldn’t be any reason for all the normal form business in the first place. However, we can easily do it in type A because we have an alternate “normal form”: one-line notation. In light of this fact, the footnote on page 78 suggests a possible resolution—the numbers game as a “one-line notation” for general \(W\)—and using it in this particular capacity would sidestep both issues that the authors mention.

(I haven’t read section 4.3; perhaps this is too inefficient as a subroutine in this algorithm?)

Besides Theorem 3.4.7, there are two other plot points: The first is Theorem 3.4.8, which is called the “second major result of the section”. I respectfully disagree: it is a rather technical point which basically says that the Exchange Property respects the normal form, with slight reservations if \(\ell(ws) > \ell(w)\).
The second is the normal form forest, which provides a very efficient (and, hence, rather cryptic) graphical representation of all the (normal forms of) group elements. Not much time is devoted to computational aspects here, but it seems that if you are given this forest you do not improve the efficiency of Theorem 3.4.7 much, and you essentially need Theorem 3.4.7 to compute the forest.

Missteps to avoid (which I did not):

- Proposition 3.4.2 really is consistent with Corollary 2.4.6; the reason the indexing is wrong is because we are now using right-cosets.

- Similarly, although $W^J = (JW)^{-1}$ as sets, don’t forget that $w^J \neq (Jw)^{-1}$.

- Proposition 3.4.2 does not imply that it is algorithmically efficient to choose the vertices so that the edges with high $m(s, t)$ are last. In fact, it implies the opposite, which accounts for the seemingly strange ordering of the vertices in Example 3.4.4.

- When computing $\tau_k$ in the normal form forest, you can prune a branch from the tree immediately if the first letter changes to something other than $k$ using braid moves: there is no need to worry about reducing the result.

- Theorem 3.4.7 is almost entirely unusable as a recursive algorithm (i.e. starting the word and peeling off $s_i$ from the right), even for type A. Instead, do the problem iteratively, starting with the identity and adding on $s_i$ successively to the right. In this way you don’t have to do any work to compute $*v$ and $v*$ since these can be read immediately from $[v]$, which you compute in the previous step.

A priori, the first case of the theorem might cause some hangups in the iterative structure, since

$$[vs] = [v*]s_n[s_n*v] = [v*]s_n[(s_n*v)s_j][(s_n*v)*]$$

which makes it look like you must re-compute $(s_n*v)_*$ and $(s_n*v)_*$ from scratch; not doing that was supposed to be the advantage over recursion. But in fact, the word $\tilde{V}$, which is $[*v]$ with the first $s_n$ removed, is actually the normal form for $s_n*v$, and hence you can still read $(s_n*v)_*$ and $(s_n*v)_*$ from the normal form of $v$.

Notice that $\tilde{V}$ is an expression for $s_n*v$, and it has $\ell(s_n*v) = \ell(*v) - 1$ letters, so it is reduced. Moreover, since $s_n[s_n*v]$ is a reduced expression for $*v$, we must have $[s_n*v] = \tilde{V}$ or else we contradict the minimality of $[*v]$.

Somewhat more concretely: what the first case says is that the piece of the word between the first $s_n$ and the first instance $s_n$ such that the tail of the word is not distinguished (and this piece might not exist, if the two $s_n$’s are the same), is left entirely unaffected by right-multiplication by $s$. 

2
For our examples, we write $j$ instead of $s_j$. In type A, we write a bar over one-line notation to avoid confusion.

**Example.** Let $W = S_6$ and $w = 4212 = 321546$. We aim to compute $[w]$, and as suggested above, we do this using Theorem 3.4.7 iteratively. Clearly $[4] = 4$ using the last case.

In the sequel we omit arguments of the following type: We will compute $[42]$ in $S_6$, so $v = [4]v = e$, and note that $(4)_*2 = 42$ so we may take $j = 2$ and apply the second case to compute $[42]$ instead in $W_4 = S_5$. We note that $(4)_*2 = 42$ is not distinguished since $42 = 24$, so the second case again applies: $[42] = [(4)_*2][(4)] = [2][4]$, and so $[42] = 24$.

Next, $[421] = 214$. We start by replacing $v = 42$ by $[v] = 24$, and then proceed as before: $[421] = [21]4$. Since the new bracket is being computed in $S_3$, we have $v = 2 = *v$ so $*v[2] = 21$ is distinguished and the first case applies: $[21] = [(2)_*2][2(2)]1 = [e]2[1] = 21$, as desired.

So far we have not done any calculations to show whether $*v[2]$ was distinguished because things were very straightforward; for our last step we will be more careful.

We claim $[4212] = 1214$. As before $[4212] = [212]4$, and we claim that $*v[2] = 212$ is not distinguished (as an element of $S_3$). To see this, we apply Lemma 2.4.7 in the second equality below:

$$321 = 212 
otin [2] \setminus (S_3) = \left( S_3^{(2)} \right)^{-1} = \{ x \in S_3 : x_i < x_j \text{ for all } i < j \text{ unless } x_i = 3 \}.$$

Therefore the second case applies. Since $v = 21$ is already in normal form (as guaranteed by iteration), we easily read off $v_* = 21$ and $*v = e$ to conclude $[21] = [e][2][2](2)1 = [e]2[1] = 21$, as desired.

(We could attempt to be less *ad hoc* and make the argument that $j$ must be 1, since it cannot be 2, because 221 is not reduced. But this is still *ad hoc* since the same would not conclude that, say, $[323] = 232$: we would need to exclude $j = 1$ as well.)

**Example.** Let $W = S_4$ and $w = w_0 = 4321$. A lucky guess shows that $w = 121321$, and we omit the calculations to show that $[w] = 121321$ as well, except perhaps the last line will show the idea for larger computations:

$$[12132 \cdot 1] = [121]3[3(32)1] = 1213[21] = 121321.$$

In general, we see that for $W = S_n$, the normal form of the longest element will be

$$1 \ 21 \ 321 \ 4321 \ \cdots \ n \ldots 21$$
**Example:** Let $W = H_3$, with $(s_1 s_2)^5 = (s_2 s_3)^3 = e$. We construct the normal form forest:

![Diagram](image)

The first two trees are trivial to construct (and suggest the construction for all dihedral groups). In the third we show some false branches:

- **3213** is not permitted since $3213 = 3231 = 2321$,
- **321232** is not permitted since $321232 = 321323 = 323123 = 232123$,
- **3212132121** is $3212321212 = 3231231212 = 2321231212$

Other potential false branches were much more easily eliminated by using braid moves more directly: no two adjacent labels have the same number, the pattern 31 (that is, 3 below 1) can never occur since $31 = 13$, similarly 323 and 21212 are forbidden. These seven restrictions account for all possible false branches except for the ultimate case (32121321232) which is similar but longer.