On the complexity of multiplication in the Iwahori-Hecke algebra of the symmetric group

ON THE COMPLEXITY OF MULTIPLICATION IN THE IWAHORI–HECKE ALGEBRA OF THE SYMMETRIC GROUP

ALICE C. NIEMEYER, GÖTZ PFEIFFER, AND CHERYL E. PRAEGER

ABSTRACT. We present new efficient data structures for elements of Coxeter groups of type A_m and their associated Iwahori–Hecke algebras $H(A_m)$. Usually, elements of $H(A_m)$ are represented as simple coefficient list of length $M = (m+1)!$ with respect to the standard basis, indexed by the elements of the Coxeter group. In the new data structure, elements of $H(A_m)$ are represented as nested coefficient lists. While the cost of addition is the same in both data structures, the new data structure leads to a huge improvement in the cost of multiplication in $H(A_m)$.

1. INTRODUCTION

Let Z be a commutative ring with one. The Iwahori-Hecke algebra H of a finite Coxeter group W over the ring Z relative to a parameter $q \in Z$ is a Z-free Z-algebra with a basis in bijection with the elements of the group W, and a multiplication that is determined by deforming the multiplication in W; see Section 3 for details. For a general introduction to finite Coxeter groups and their Iwahori–Hecke algebras we refer the reader to the textbook by Geck and Pfeiffer [4], for the particular case of the symmetric group we refer to Mathas' book [5]. They have applications in various branches of Mathematics, such as knot theory, or the representation theory of groups of Lie type (see [2] and its references).

We study the complexity, or cost, of multiplication in the Iwahori-Hecke algebra $H = H(A_m)$ of type A_m, so that W is the symmetric group Sym_{m+1}. Thus H has dimension $M = (m+1)!$. The standard basis for H is indexed by the elements of W and, for example in the CHEVIE package [3] for GAP3, general elements of H are represented as *simple coefficient lists* of length M in this basis.

We introduce new data structures. First we represent elements of W in a *tower data-structure* described in Section 2, in which the set of the transpositions $(i, i+1)$ for $1 \leqslant i \leqslant m$ is replaced by a certain set of roughly $m^2/2$ cycles. Then, using this, in Section 3 we represent each element of H as a *nested coefficient list*. At the top level, this is a list of length m with entries (indexed by certain of these cycles) being nested coefficient lists representing elements in the Iwahori-Hecke algebra $H(A_{m-1})$. We also develop tools for multiplying two such lists.

Using either of these data-structures for H, multiplying elements h, h' of H necessarily requires us to consider each of the M^2 pairs of coefficients of h, h'. We compare the 'worst-case cost' (complexity) of multiplying two elements of H using simple coefficient lists and nested coefficient lists. The following theorem is proved in Section 4.

2010 *Mathematics Subject Classification.* 20C08; 20F55, 20B40.

Key words and phrases. Iwahori–Hecke algebra, finite Coxeter group, symmetric group, complexity.

Theorem 1.1. *Let* Z, H, W, M *and* m *be as above.*

(a) *The cost of multiplying two elements in* H *each represented as a coefficient list over* Z, *using* (4.4), *is at most* $\frac{m^2+m+4}{2}M^2$ *operations in* Z.

(b) *The cost of multiplying two elements in* H, *each represented as a nested coefficient list over* $H(A_{m-1})$, *is at most* $(1+e)M^2$ *operations in* Z, *where* $e = \exp(1)$.

1.1. **The standard data structure in CHEVIE.** The Coxeter group $W = \mathsf{Sym}_{m+1}$ acting on $\{1,\dots,m+1\}$ has a set S of distinguished generating involutions, namely the transpositions $(i,i{+}1)$ of adjacent points, $i = 1,\dots,m$. Each element $w \in W$ can be written as a product $w = r_1 \cdots r_k$ of k Coxeter generators $r_i \in S$, and k is called the *length* of w, if it is minimal with this property. In that case this expression of w as word in the generators is called a *reduced expression*. A general element $h \in H$ is a linear combination of basis elements T_w, $w \in W$, where T_w is regarded as a product $T = T_{r_1} \cdots T_{r_k}$, corresponding to a reduced expression of $w = r_1 \cdots r_k$.

The standard strategy in existing implementations of Hecke algebra arithmetic, such as the GAP3 package CHEVIE [3], regards an element H as a linear combination of basis elements T_w, $w \in W$, and reduces the task of multiplying two elements of H to the case where the second element is T_s for some $s \in S$. The underlying data structure resembles the simple coefficient list, that we consider here.

1.2. **Nested coefficient lists.** An element of $W = \mathsf{Sym}_{m+1}$ can alternatively be decomposed in terms of coset representatives along the chain $\mathsf{Sym}_1 < \mathsf{Sym}_2 < \cdots < \mathsf{Sym}_{m+1}$ of 'parabolic' subgroups of W. Here Sym_j is the symmetric group on $\{1,\dots,j\}$ fixing each point i with $j < i \leqslant m+1$, and we choose a set X_j of coset representatives of Sym_j in Sym_{j+1} to be a set of cycles $a(j,l)$ for $0 \leqslant l \leqslant j$ (see Equation 2.3). Each $w \in W$ can then be written uniquely as $w = x_1 \cdots x_m$ where each $x_j = a(j,a_j) \in X_j$ for some a_j. It is sufficient to record the list of integers a_j for $1 \leqslant j \leqslant m$ and this is the *tower* of w, see Definition 2.1. Various properties such as the Coxeter length or the descent set of w can be extracted easily from the tower. Furthermore, it is possible to find the tower of w^{-1} from the tower of w (see Proposition 2.4) and the tower of a product of two permutations from their towers (see Proposition 2.8).

We exploit the tower data structure to represent elements of H as nested coefficient lists, with the 'nesting' corresponding to the Iwahori-Hecke subalgebras $H(A_j)$ for the subgroups $W(A_j) = \mathsf{Sym}_{j+1}$. In this case the task of multiplying two elements of H is reduced to the case where the second element is T_x with $x \in \bigcup X_j$. Proposition 3.2 provides a formula for the efficient evaluation of such a product.

It is certainly desirable to develop similar techniques for other types of Coxeter groups. However, new ideas are needed in order to generalize this construction from type A to other types.

2. The tower data-structure for the symmetric group

For $m = 1,2,\dots$, we denote the Coxeter group of type A_m by $W(A_m)$. Here we identify the group $W(A_m)$ with the symmetric group Sym_{m+1} on the $m+1$ points $\{1,\dots,m+1\}$. Denote $s_i = (i,i{+}1)$ and $S = \{s_i : i = 1,\dots,m\}$. The elements of S are called the

simple reflections of W and S is a Coxeter generating set of W. Thus every element can be written as a product of simple reflections, usually in many different ways. For $w \in W(A_m)$, denote the *length* of w by $\ell(w)$, that is $\ell(w)$ is the minimal number k such that $w = s_{i_k} \cdots s_{i_k}$ for simple reflections $s_{i_j} \in S$. We inductively define a normal form for elements of W as follows. We define, for $0 \leqslant k \leqslant m$, elements $a(m,k)$, where $a(m,0) = 1$ and for $k \geqslant 1$,

$$a(m,k) = s_m s_{m-1} \cdots s_{m-k+1} \in W(A_m). \tag{2.1}$$

For each k, the length

$$\ell(a(m,k)) = k. \tag{2.2}$$

Note that for $k \geqslant 1$, $a(m,k)$ is the $(k+1)$-cycle

$$a(m,k) = (m{-}k{+}1, m{-}k{+}2, \ldots, m{+}1) \in \mathsf{Sym}_{m+1} \tag{2.3}$$

and the set

$$X_m = \{a(m,k) : 0 \leqslant k \leqslant m\} \tag{2.4}$$

is the set of minimal length right coset representatives of $W(A_{m-1})$ in $W(A_m)$. This means that for each element $w \in W(A_m)$ there are unique elements $u \in W(A_{m-1})$ and $x_m = a(m, a_m) \in X_m$ such that $w = u \cdot a(m, a_m)$, where the explicit multiplication dot indicates that this decomposition of w is *reduced*, i.e., $\ell(w) = \ell(u) + \ell(x_m) = \ell(u) + a_m$.

Repeated application of the above for $j = 1, \ldots, m$ yields integers $a_j \in \{0, \ldots, j\}$ such that w is the reduced product

$$w = x_1 \cdot x_2 \cdot \ldots \cdot x_m \tag{2.5}$$

of the coset representatives $x_j = a(j, a_j) \in X_j$ with the property that

$$\begin{aligned} \ell(w) &= \ell(x_1) + \ell(x_2) + \cdots + \ell(x_m) \\ &= a_1 + a_2 + \cdots + a_m. \end{aligned} \tag{2.6}$$

Definition 2.1. For $w \in W(A_m)$ as in Equation 2.6 we call the sequence

$$\tau(w) = (a_1, a_2, \ldots, a_m)$$

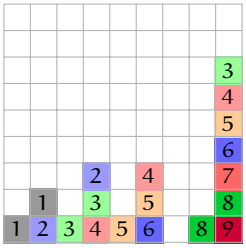
the *tower of* w.

The sequence $\tau(w)$ of m non-negative integers determines the terms x_i in the representation of w in Equation 2.5 and thus yields a decomposition of $w \in W(A_m)$ as a product of coset representatives along the chain of subgroups $W(A_0) < W(A_1) < \cdots < W(A_m)$.

Example 2.2. As an example of our data structure we consider for $m = 9$ the permutation $w = (1,8,10,3)(2,4,6,7,5) \in \mathsf{Sym}_{10}$. Then

$$\begin{aligned} w &= s_1 \cdot s_2 s_1 \cdot s_3 \cdot s_4 s_3 s_2 \cdot s_5 \cdot s_6 s_5 s_4 \cdot s_8 \cdot s_9 s_8 s_7 s_6 s_5 s_4 s_3 \\ &= a(1,1) \cdot a(2,2) \cdot a(3,1) \cdot a(4,3) \cdot a(5,1) \cdot a(6,3) \cdot a(7,0) \cdot a(8,1) \cdot a(9,7). \end{aligned} \tag{2.7}$$

As a tower, w can be expressed as $\tau(w) = (1,2,1,3,1,3,0,1,7)$. We illustrate the tower of w with the following *tower diagram*.



The tower diagram for w is constructed by writing the subscripts of the letters of the i-th factor $a(i, a_i)$ of w into the i-th column of the diagram from the bottom up. Thus the tower diagram consists of columns of lengths $\tau(w) = (1, 2, 1, 3, 1, 3, 0, 1, 7)$, that is the i-th column has height a_i. Observe that the position and the height of a column in the diagram uniquely determine the actual entries of the column.

It is sometimes convenient to assume $a_0 = 0$ and $a_j = 0$ for $j > m$, allowing us to view a permutation in S_m as an element of S_j for any $j \geqslant m$. It can also be convenient to omit trailing zeros from $\tau(w)$. From this representation of a permutation w one can read off quickly

- the length of w as $\ell(w) = a_1 + a_2 + \cdots + a_m$, by (2.6);
- a reduced expression for w as concatenation of the words $a(1, a_1)$, $a(2, a_2)$, $\ldots$, $a(m, a_m)$, by (2.5);
- the image of a point i under the permutation w, or the permutation w as product of the cycles $a(1, a_1)$, $a(2, a_2)$, $\ldots$, $a(m, a_m)$, using (2.3).

We can also determine the *descent set* $\mathcal{D}(w) = \{s \in S : \ell(sw) < \ell(w)\}$ of w from its tower $\tau(w)$, as shown in Lemma 2.7 below.

Furthermore, we can compute

- the tower $\tau(ww')$ of a product of $w, w' \in W(A_m)$ from the towers $\tau(w)$ and $\tau(w')$ (see Lemma 2.6);
- the tower $\tau(w^{-1})$ of the inverse of $w \in W(A_m)$ from the tower $\tau(w)$ (see Lemma 2.3).

From the permutation one can of course determine other properties, like the order of w, its cycle structure and thus its conjugacy class in $W(A_m)$, etc.

In the next three subsections we consider the practicability of this data structure for the purpose of extensive computations with elements in $W(A_{m-1})$. We will see how to multiply two towers and how to invert a tower.

Inverses. The tower $\tau(w^{-1})$ of the inverse of the permutation w can be computed from the tower $\tau(w)$ on the basis of the following observation.

Lemma 2.3. *Suppose* $w = x_1 \cdots x_m \in W(A_m)$ *with* $x_i = a(i, a_i)$ *and* $a_m \neq 0$. *Set* $a_0 = 0$ *and let* k *be the largest index* $i < m$ *with* $a_i = 0$. *Moreover, let*

$$w' = x_1 \cdots x_{k-1} \cdot x'_k \cdots x'_{m-1} \in W(A_{m-1})$$

with $x'_i = a(i, a_{i+1} - 1)$, *for* $i \geqslant k$. *Then* $w^{-1} = (w')^{-1} \cdot a(m, m - k)$.

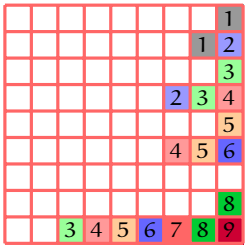


FIGURE 1. Transpose of tower diagram for w in Example 2.2

Proof. By the hypothesis $a(k, a_k) = a(k, 0) = 1$ and $a_i > 0$ for all $i > k$. Therefore we can write

$$\begin{aligned} w &= a(1, a_1) \cdots a(k-1, a_{k-1}) \cdot 1 \cdot a(k+1, a_{k+1}) \cdots a(m, a_m) \\ &= a(1, a_1) \cdots a(k-1, a_{k-1}) \cdot s_{k+1} a(k, a_{k+1} - 1) \cdots s_m a(m-1, a_m - 1) \\ &= s_{k+1} \cdots s_m \, x_1 \cdots x_{k-1} \cdot x'_k \cdots x'_{m-1} \\ &= a(m, m-k)^{-1} w', \end{aligned}$$

as required. □

Note that since $w' \in W(A_{m-1})$ its inverse $(w')^{-1}$ can now be computed recursively, by applying Lemma 2.3 to the smaller group $W(A_{m-1})$.

Proposition 2.4. *Let* $\tau(w) = (a_1, \ldots, a_m)$. *Let* $k = \max\{i \mid 0 \leqslant i < m$ *and* $a_i = 0\}$. *Then* $\tau(w^{-1}) = (a'_1, \ldots, a'_m)$, *where* $a'_m = a(m, m{-}k)$ *and, recursively,* $(a'_1, \ldots, a'_{m-1}) = \tau((w')^{-1})$, *where* $w' = a(m, m-k)\, w$ *is a word of length* $\ell(w') = \ell(w) - (m{-}k)$.

In practice, this process amounts to transposing and straightening the tower diagram of w, as illustrated by the following example.

Example 2.5. We continue Example 2.2 to find the tower diagram of the inverse of the element w given in Equation 2.7. The inverse w^{-1} of w is

$$\begin{aligned} &s_3 s_4 s_5 s_6 s_7 s_8 s_9 \cdot s_8 \cdot s_4 s_5 s_6 \cdot s_5 \cdot s_2 s_3 s_4 \cdot s_3 \cdot s_1 s_2 \cdot s_1 \\ &\quad = \; a(9,7)^{-1} a(8,1)^{-1} a(7,0)^{-1} a(6,3)^{-1} a(5,1)^{-1} a(4,3)^{-1} a(3,1)^{-1} a(2,2)^{-1} a(1,1)^{-1}. \end{aligned}$$

Observe that w^{-1} can be represented by the transpose of the tower diagram of w as in Figure 1, that is to say, $a(9,7)^{-1}$ is the bottom row of Figure 1, and similarly for the other rows. The word for w^{-1} can be read off from Figure 1, if we read row by row from left to right and bottom to top.

We find the tower diagram for w^{-1} by applying Lemma 2.3 repeatedly. In the first step, applying Lemma 2.3, we have $m = 9$ and the value of k is the largest integer with $a(k, 0)^{-1}$ occurring in the above expression, namely $k = 7$. (It would be 0 if all terms $a(i, a_i)^{-1}$ had $a_i > 0$.) Note that $k = 7$ is the largest integer missing in the right most column of Figure 1. We replace w^{-1} by $(w')^{-1} a(9, 9-7)$, where $w' = a(1,1) \cdots a(6,3) \cdot x'_7 \cdot x'_8$, where $x'_7 = a(7, a_8 - 1) = a(7, 0) = 1$ and $x'_8 = a(8, a_9 - 1) = a(8, 6) = s_8 s_7 s_6 s_5 s_4 s_3$. Thus $w^{-1} = (s_1 \cdot s_2 s_1 \cdot s_3 \cdot s_4 s_3 s_2 \cdot s_5 \cdot s_6 s_5 s_4 \cdot s_8 s_7 s_6 s_5 s_4 s_3)^{-1} \cdot s_9 s_8$. This expression is represented by the left most diagram of Figure 2. Note that the 8×8 subgrid with thick gridlines represents the transpose of the tower diagram of w'.

Applying Lemma 2.3 recursively to w', we find the following expressions for w^{-1}: each line represents one application of Lemma 2.3, where the s_i in the terms $x_i = a(i, a_i)$ for $i \geqslant k$ have been underlined for emphasis.

$$\begin{aligned}
w^{-1} &= \left(s_1 \cdot s_2 s_1 \cdot s_3 \cdot s_4 s_3 s_2 \cdot s_5 \cdot s_6 s_5 s_4 \cdot \underline{s_8} s_7 s_6 s_5 s_4 s_3\right)^{-1} \cdot s_9 s_8 \\
&= \left(\underline{s_1} \cdot \underline{s_2} s_1 \cdot \underline{s_3} \cdot \underline{s_4} s_3 s_2 \cdot \underline{s_5} \cdot \underline{s_6} s_5 s_4 \cdot \underline{s_7} s_6 s_5 s_4 s_3\right)^{-1} \cdot s_8 \cdot s_9 s_8 \\
&= \left(s_1 \cdot s_3 s_2 \cdot \underline{s_5} s_4 \cdot \underline{s_6} s_5 s_4 s_3\right)^{-1} \cdot s_7 s_6 s_5 s_4 s_3 s_2 s_1 \cdot s_8 \cdot s_9 s_8 \\
&= \left(s_1 \cdot \underline{s_3} s_2 \cdot \underline{s_4} \cdot \underline{s_5} s_4 s_3\right)^{-1} \cdot s_6 s_5 \cdot s_7 s_6 s_5 s_4 s_3 s_2 s_1 \cdot s_8 \cdot s_9 s_8 \\
&= \left(s_1 \cdot s_2 \cdot \underline{s_4} s_3\right)^{-1} \cdot s_5 s_4 s_3 \cdot s_6 s_5 \cdot s_7 s_6 s_5 s_4 s_3 s_2 s_1 \cdot s_8 \cdot s_9 s_8 \\
&= s_3 s_2 s_1 \cdot s_4 \cdot s_5 s_4 s_3 \cdot s_6 s_5 \cdot s_7 s_6 s_5 s_4 s_3 s_2 s_1 \cdot s_8 \cdot s_9 s_8.
\end{aligned}$$

This sequence of applications of Lemma 2.3 is also illustrated by the sequence of diagrams in Figure 2. Note that in each diagram, the value of k used in Lemma 2.3 is given below. Also, the subdiagram with thick gridlines represents the transpose of the tower diagram of the element to which Lemma 2.3 is applied. Those columns with thin gridlines represent the completed columns of the tower diagram of w^{-1}. Thus the tower of w^{-1} is $\tau(w^{-1}) = (0, 0, 3, 1, 3, 2, 7, 1, 2)$.

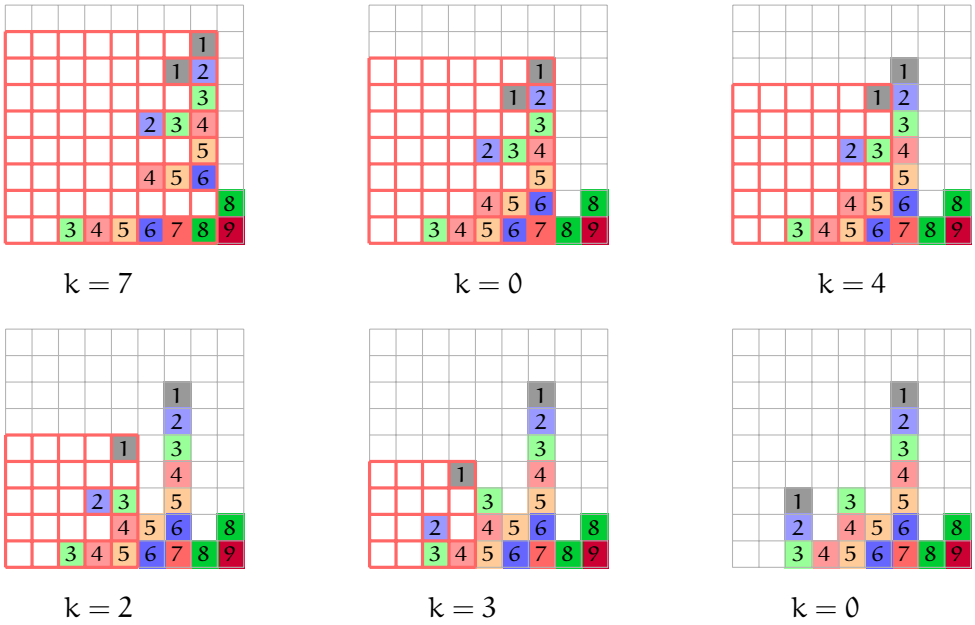


FIGURE 2. Recursive applications of Lemma 2.3 for Example 2.5

Products. Multiplication of towers can be based on the following multiplicative property of the coset representatives $a(m, k)$. It allows a product $a(m, k)\, a(j, l)$ with $j \leqslant m$ to be rewritten as a product of at most two coset representatives in increasing order.

Proposition 2.6. *For* $1 \leqslant j \leqslant m$, $0 \leqslant k \leqslant m$ *and* $0 \leqslant l \leqslant j$,

$$a(m,k)\,a(j,l) = \begin{cases} a(j,l)\cdot a(m,k), & k < m-j, \\ a(m,k+l), & k = m-j, \\ a(j-1,l-1)\cdot a(m,k-1), & m-j < k \leqslant m-j+l, \\ a(j-1,l)\cdot a(m,k), & k > m-j+l. \end{cases}$$

If we write the factors as permutations

$$a(m,k) = (m-k+1, m-k+2, \ldots, m+1), \qquad a(j,l) = (j-l+1, j-l+2, \ldots, j+1),$$

using equation (2.3), with support

$$M = \{m-k+1, m-k+2, \ldots, m+1\}, \qquad J = \{j-l+1, j-l+2, \ldots, j+1\},$$

respectively, and denote $J-1 = \{x-1 : x \in J\}$, then the four cases of the lemma correspond to the cases

- $J \cap M = \varnothing$, the *pass* case;
- $J \cap M \neq \varnothing$ and $J-1 \cap M = \varnothing$, the *join* case;
- $|J \cap M| > 1$ and $J-1 \nsubseteq M$, the *cancel* case;
- $J \subseteq M$ and $J-1 \subseteq M$, the *shift* case.

Note that, by equation (2.2), the length of the product is

$$\ell(a(m,k)\,a(j,l)) = \begin{cases} k+l-2, & \text{if } m-j < k \leqslant m-j+l, \\ k+l, & \text{otherwise,} \end{cases} \tag{2.8}$$

where $k+l$ is the sum of the lengths of the factors. Hence with the exception of the *cancel* case, all products $a(m,k)\,a(j,l)$ are reduced.

Proof. Recall from equation (2.1) that

$$a(m,k) = s_m s_{m-1} \cdots s_{m-k+1}, \qquad a(j,l) = s_j s_{j-1} \cdots s_{j-l+1}.$$

In the *pass* case, i.e., if $j < m-k$, all generators s_i occurring in $a(j,l)$ commute with those in $a(m,k)$ and hence $a(m,k)a(j,l) = a(j,l)\cdot a(m,k)$.

In the *join* case, i.e., if $j = m-k$,

$$a(m,k)\,a(j,l) = s_m s_{m-1} \cdots s_{j+1} \cdot s_j s_{j-1} \cdots s_{j-l+1} = a(m,k+l).$$

In the *cancel* case, i.e., if $j > m-k$ and $j-l \leqslant m-k$, the simple reflection s_{m-k+1} occurs as a factor in

$$a(j,l) = s_j s_{j-1} \cdots s_{m-k+2} \cdot s_{m-k+1} \cdot s_{m-k} s_{m-k-1} \cdots s_{j-l+1},$$

and

$$\begin{aligned} a(m,k)\,s_i &= s_{i-1}\,a(m,k), && \text{for } i \in \{j, j-1, \ldots, m-k+2\}, \\ a(m,k)\,s_i &= a(m,k-1), && \text{for } i = m-k+1, \text{ and} \\ a(m,k-1)\,s_i &= s_i\,a(m,k-1), && \text{for } i \in \{m-k, m-k-1, \ldots, j-l+1\}. \end{aligned}$$

It follows that

$$\begin{aligned} a(m,k)\,a(j,l) &= s_{j-1}s_{j-2}\cdots s_{m-k+1}\cdot s_{m-k}s_{m-k-1}\cdots s_{j-l+1}\,a(m,k-1)\\ &= a(j-1,l-1)\cdot a(m,k-1), \end{aligned}$$

as desired.

Finally, in the *shift* case, i.e., if $j > m-k+l$, we have $a(m,k)s_i = s_{i-1}a(m,k)$ for all factors s_i of $a(j,l) = s_j s_{j-1}\cdots s_{j-l+1}$. Hence

$$a(m,k)a(j,l) = s_{j-1}s_{j-2}\cdots s_{j-l}a(m,k) = a(j-1,l)\cdot a(m,k),$$

as desired. □

In particular, for $0 \leqslant k,l \leqslant m = j$, it follows that

$$a(m,k)\,a(m,l) = \begin{cases} a(m,l), & k=0,\\ a(m{-}1,l{-}1)\cdot a(m,k{-}1), & 0<k\leqslant l,\\ a(m{-}1,l)\cdot a(m,k), & k>l. \end{cases} \tag{2.9}$$

As an immediate application, we compute the descent set of a permutation w.

Lemma 2.7. *Let $w \in W(A_m)$ with tower $\tau(w) = (a_1,\dots,a_m)$. Then*

$$\mathcal{D}(w) = \{s_i : a_i > a_{i-1}\}.$$

In particular, s_i is the smallest descent of $w \neq 1$ if i is the smallest index with $a_i > 0$.

Proof. Write $w = w'a(i-1,a_{i-1})a(i,a_i)w''$ and set $k = a_{i-1}$. Let $1 \leqslant i \leqslant m$. Note that $s_i a(j,a_j) = a(j,a_j)s_i$ for $j < i-1$, and that $s_i a(i-1,k) = a(i,k+1)$. Hence

$$s_i w = w'\,s_i\,a(i{-}1,a_{i-1})\,a(i,a_i)\,w'' = w'\,a(i,a_{i-1}{+}1)\,a(i,a_i)\,w''.$$

By equation (2.9),

$$a(i,a_{i-1}{+}1)\,a(i,a_i) = \begin{cases} a(i{-}1,a_i{-}1)\,a(i,a_{i-1}), & a_{i-1} < a_i\\ a(i{-}1,a_i)\,a(i,a_{i-1}{+}1), & a_{i-1} \geqslant a_i. \end{cases}$$

It follows that $\ell(s_i w) < \ell(w)$ if and only if $a_{i-1} < a_i$, as claimed. □

Proposition 2.6 implies that for the tower $\tau(w) = (a_1,\dots,a_m)$, there is a *multiplication function*

$$\mu_m\colon (j,k,l) \mapsto (j',k',l'),$$

which produces integers j',k',l' from j,k,l such that $a(m,k)\,a(j,l) = a(j',l')\,a(m,k')$. More precisely, by Proposition 2.6

$$\mu_m(j,k,l) = \begin{cases} (j,k,l), & \text{if } j < m-k,\\ (0,k+l,0), & \text{if } j = m-k,\\ (j-1,k-1,l-1), & \text{if } m-k < j \leqslant m-k+l,\\ (j-1,k,l), & \text{if } j > m-k+l. \end{cases} \tag{2.10}$$

This yields the following formula for multiplying a tower $\tau(w) = (a_1,\dots,a_m)$ with a coset representative $a(j,l)$. We denote the resulting tower $\tau(w\,a(j,l))$ by $(a_1,\dots,a_m)\star a(j,l)$.

Proposition 2.8. *Let* $(a_1, \dots, a_m)$ *be a tower and let* $0 < l \leqslant j \leqslant m$. *Then*

$$(a_1, \dots, a_m) \star a(j, l) = (a'_1, \dots, a'_m),$$

where a'_m *is determined by* $(j', a'_m, l') = \mu_m(j, a_m, l)$ *and, recursively,*

$$(a'_1, \dots, a'_{m-1}) = \begin{cases} (a_1, \dots, a_{m-1}), & \text{if } l' = 0, \\ (a_1, \dots, a_{m-1}) \star a(j', l'), & \text{if } l' > 0. \end{cases}$$

Proof. If $w = a(1, a_1) \cdots a(m, a_m)$ then, by the definition of μ_m,

$$\begin{aligned} w\, a(j, l) &= a(1, a_1) \cdots a(m{-}1, a_{m-1})\, a(m, a_m)\, a(j, l) \\ &= a(1, a_1) \cdots a(m{-}1, a_{m-1})\, a(j', l')\, a(m, a'_m), \end{aligned}$$

where $(j', a'_m, l') = \mu_m(j, a_m, l)$. Now, either $l' = 0$, whence $a(j', l') = 1$, or $l' > 0$, and

$$\tau(a(1, a_1) \cdots a(m{-}1, a_{m-1})\, a(j', l')) = (a_1, \dots, a_{m-1}) \star a(j', l'),$$

by induction on m. In any case, this yields $(a_1, \dots, a_m) \star a(j, l) = (a'_1, \dots, a'_m)$, as desired. □

The task of multiplying two towers $\tau(w)$ and $\tau(w')$ (that is to compute the tower $\tau(ww')$ from the towers $\tau(w)$ and $\tau(w')$) is thus reduced to incorporating the parts of $\tau(w')$ into $\tau(w)$, one at a time:

$$(a_1, \dots, a_m) \star (a'_1, \dots, a'_m) = \Big(\dots \big((a_1, \dots, a_m) \star a(1, a'_1) \big) \star \dots \Big) \star a(m, a'_m). \tag{2.11}$$

3. Computing in the Iwahori–Hecke algebra

In this section we use the tower data structure to develop an algorithm for the multiplication in the Iwahori–Hecke algebra of the symmetric group $W = W(A_m) = \mathsf{Sym}_{m+1}$.

Let Z be a commutative ring with one, and let $q \in Z$. The Iwahori–Hecke algebra $H = H(A_m)$ of W is the Z-free Z-algebra (with one, denoted 1_H) with basis $\{T_w \mid w \in W\}$ and multiplication defined by

$$T_w T_s = \begin{cases} T_{ws}, & \ell(ws) > \ell(w), \\ (q-1)T_w + qT_{ws}, & \ell(ws) < \ell(w), \end{cases} \tag{3.1}$$

for $w \in W$, $s \in S$.

We first translate Lemma 2.6 into the context of Hecke algebras.

Lemma 3.1. *For* $m \geqslant j \geqslant 1$ *and* $k, l \geqslant 1$,

$$T_{a(m,k)}\, T_{a(j,l)} = \begin{cases} T_{a(j,l)}\, T_{a(m,k)}, & k < m - j, \\ T_{a(m,k+l)}, & k = m - j, \\ (q-1)\, T_{a(j-1,j-m+k-1)}\, T_{a(m,m-j+l)} & \\ \qquad + q\, T_{a(j-1,l-1)}\, T_{a(m,k-1)}, & m - j < k \leqslant m - j + l, \\ T_{a(j-1,l)}\, T_{a(m,k)}, & k > m - j + l. \end{cases}$$

Proof. In three of the four cases, the product $a(m,k)\,a(j,l)$ is reduced, whence we get with Lemma 2.6 that

$$T_{a(m,k)}\,T_{a(j,l)} = T_{a(m,k)\,a(j,l)} = T_{a(j',l')\,a(m,k')} = T_{a(j',l')}\,T_{a(m,k')},$$

where $(j',k',l') = \mu_m(j,k,l)$. In the *cancel* case, i.e., if $m-j < k \leqslant m-j+1$, mimicking the proof of Lemma 2.6, we get

$$\begin{aligned} T_{a(m,k)}\,T_{a(j,l)} &= T_{a(m,k)}\,(T_jT_{j-1}\cdots T_{m-k+2})\,T_{m-k+1}\,(T_{m-k}T_{m-k-1}\cdots T_{j-l+1}) \\ &= (T_{j-1}T_{j-2}\cdots T_{m-k+1})\,T_{a(m,k-1)}\,T_{m-k+1}^2\,(T_{m-k}T_{m-k-1}\cdots T_{j-l+1}) \\ &= (q-1)\,T_{a(j-1,j-m+k-1)}\,T_{a(m,m-j+l)} + q\,T_{a(j-1,l-1)}\,T_{a(m,k-1)}, \end{aligned}$$

as desired. □

A general element $h \in H$ is a linear combination $h = \sum_{w\in W} z_w T_w$ with coefficients $z_w \in Z$. Moreover, it follows from Equation 2.5 and Lemma 3.1 that we may also write

$$h = \sum_{k=0}^{m} h_k T_{a(m,k)} \tag{3.2}$$

with coefficients $h_k \in H(A_{m-1})$, each of which in turn is a combination of the $T_{a(m-1,k)}$ with coefficients in $H(A_{m-2})$, and so on. (Note that $H(A_0) = Z$.)

Thus we can represent an element $h \in H$ as a *nested coefficient list* $h = (h_0,\dots,h_m)$ with $h_l \in H(A_{m-1})$, $l = 0,\dots,m$. It is obvious how to add two such elements.

Let $j \leqslant m$. Assuming we can multiply $h \in H(A_m)$ by $g_l \in H(A_{j-1})$, $l = 0,\dots,j$, we can multiply h by $g = \sum_{l=0}^{j} g_l T_{a(j,l)} \in H(A_j)$, and obtain

$$h\,g = \sum_{l=0}^{j} (h\,g_l) T_{a(j,l)}, \tag{3.3}$$

where $h\,g_l \in H(A_m)$ is a $H(A_{m-1})$-combination of $T_{a(m,k)}$. The products $(h\,g_l)T_{a(j,l)}$ in turn can then be evaluated using the following Proposition 3.2 (where $h\,g_l$ is replaced by h). This proposition provides a recursive formula for the computation of a product of the form $hT_{a(j,l)}$, where $h \in H(A_m)$ and $0 \leqslant l \leqslant j \leqslant m$.

Proposition 3.2. *Let* $h = (h_0,\dots,h_m) \in H(A_m)$ *and let* j,l *be integers such that* $1 \leqslant j \leqslant m$ *and* $0 \leqslant l \leqslant j$. *Then*

$$(h_0,\dots,h_m)T_{a(j,l)} = (h'_0,\dots,h'_m),$$

where h'_k *for* $0 \leqslant k \leqslant m$ *is given in the following table:*

Line	h'_k	*Conditions*
(a)	$h_k T_{a(j,l)}$	$k < m-j$
(b)	$q h_{k+1} T_{a(j-1,l-1)}$	$m-j \leqslant k < m-j+l$
(c)	$h_{m-j} + (q-1)\sum_{i=1}^{l} h_{m-j+i} T_{a(j-1,i-1)}$	$k = m-j+l$
(d)	$h_k T_{a(j-1,l)}$	$k > m-j+l$

Proof. Note that $h = (h_0,\dots,h_m) = \sum_{k=0}^{m} h_k T_{a(m,k)}$ and we are computing $h' = (h'_0,\dots,h'_m) = \sum_{k=0}^{m} h'_k T_{a(m,k)}$ such that $hT_{a(j,l)} = h'$. If $l = 0$ then $T_{a(j,l)} = 1_H$ and hence $h' = h$ so $h'_k = h_k$ as in Case (a), (c) or (d), according as $k < m-j$, $k = m-j$, or $k > m-j$, respectively. Assume now that $l > 0$.

Suppose first that $k < m - j$ then $T_{a(m,k)}T_{a(j,l)} = T_{a(j,l)}T_{a(m,k)}$ and we see below that this is the only contribution to h'_k. In particular, $h'_k = h_k T_{a(j,l)}$, and Line (a) holds.

We defer the case $k = m - j$ and consider next $k \in [m - j + 1, m - j + l]$. Then

$$\begin{aligned}(3.4)\quad &\sum_{k=m-j+1}^{m-j+l} h_k T_{a(m,k)} T_{a(j,l)} \\ &= \sum_{k=m-j+1}^{m-j+l} (q-1) h_k T_{a(j-1,j-m+k-1)} T_{a(m,m-j+l)} + q h_k T_{a(j-1,l-1)} T_{a(m,k-1)} \\ &= (q-1)\left(\sum_{i=1}^{l} h_k T_{a(j-1,i-1)}\right) T_{a(m,m-j+l)} + \sum_{k=m-j}^{m-j+l-1} q h_{k+1} T_{a(j-1,l-1)} T_{a(m,k)}.\end{aligned}$$

If $m - j + 1 \leqslant k < m - j + l$ this yields a contribution of $q h_{k+1} T_{a(j-1,l-1)}$ to h'_k. We again see below this is the only contribution to h'_k and Line (b) holds.

We find two contributions to h'_{m-j+l}. The first contribution comes from the coefficient of $T_{a(m,m-j+l)}$ in Equation (3.4), namely $(q-1)\sum_{i=1}^{l} h_{m-j+i} T_{a(j-1,i-1)}$. The second contribution comes from considering $k = m - j$, where we find

$$h_k T_{a(m,k)} T_{a(j,l)} = h_k T_{a(m,k+l)} = h_{m-j} T_{a(m,m-j+l)},$$

that is a contribution of h_{m-j}. We find no other contributions below, and hence the sum of these two yields Line (c).

Finally, let $m - j + l < k \leqslant m$. Then $T_{a(m,k)}T_{a(j,l)} = T_{a(j-1,l)}T_{a(m,k)}$ and hence $h'_k = h_k T_{a(j-1,l)}$ and Line (d) holds. □

Example 3.3. We illustrate the process with the computation of the product hg of an element $h \in H(A_2)$ and an element $g \in H(A_1)$. We have

$$h = h_0 T_{a(2,0)} + h_1 T_{a(2,1)} + h_2 T_{a(2,2)},$$

where $h_0, h_1, h_2 \in H(A_1)$ and, for $i \in \{0, 1, 2\}$,

$$h_i = h_{i0} T_{a(1,0)} + h_{i1} T_{a(1,1)},$$

with $h_{i0}, h_{i1} \in H(A_0) = Z$. All in all, $h \in H(A_2)$ is represented by 3 elements $h_i \in H(A_1)$, or by $3! = 6$ scalars $h_{ij} \in H(A_0) = Z$. We write

$$h = (h_0, h_1, h_2) = ((h_{00}, h_{01}), (h_{10}, h_{11}), (h_{20}, h_{21})).$$

Note that multiplying h with a scalar $z \in Z$, or adding any two elements of $H(A_2)$, requires 6 operations in Z. In a similar way,

$$g = g_0 T_{a(1,0)} + g_1 T_{a(1,1)} = (g_0, g_1) \in H(A_1),$$

where $g_i \in H(A_0)$ for $i = 0, 1$. Recall that $T_{a(j,0)} = 1_H$, for $j \geqslant 0$, so that such factors in a product can simply be ignored. Then computing

$$hg = h(g_0, g_1) = h(g_0 T_{a(1,0)} + g_1 T_{a(1,1)}) = hg_0 + (hg_1) T_{a(1,1)},$$

requires $3 \cdot 6 = 18$ operations in Z (6 each for multiplying $h \in H(A_2)$ with the scalars g_i, and another 6 for adding two elements in $H(A_2)$), plus the number of operations needed

to determine $h'T_{a(1,1)}$, where $h' = hg_1$. We get

$$h'T_{a(1,1)} = (h'_0, h'_1, h'_2)T_{a(1,1)} = (h''_0, h''_1, h''_2),$$

where h''_k, $k \in \{0, 1, 2\}$, are determined by Proposition 3.2 with $m = 2$ and $j = l = 1$, as follows. For $k = 2$, case (c) yields

$$h''_2 = h'_1 + (q-1)h'_2,$$

using 4 operations in Z, as each $h'_i \in H(A_1)$ is represented by 2 scalars. For $k = 1$, case (b) yields

$$h''_1 = qh'_2,$$

using 2 operations in Z. For $k = 0$, case (a) yields

$$h''_0 = h'_0 T_{a(1,1)} = (h'_{00}, h'_{01})T_{a(1,1)} = (h''_{00}, h''_{01}),$$

where h''_{00} and h''_{01} are determined by a further application of Proposition 3.2 with $m = 1$ and $j = l = 1$, as follows. Case (c) yields

$$h''_{01} = h'_{00} + (q-1)h'_{01},$$

using 2 operations in Z. Case (b) yields

$$h''_{00} = qh'_{01},$$

using 1 further operation in Z.

In total, the computation of hg in terms of nested coefficient lists needs 27 operations in Z. Note that computing the same product hg in terms of simple coefficient lists needs the same number of operations in Z, as $T_{a(1,1)} = T_{s_1}$.

We now determine the complexity of the arithmetical operations in $H = H(A_m)$ in general.

4. Complexity

We compare the complexity of the arithmetical operations in $H = H(A_m)$, using different data structures to represent elements of H, once as simple coefficient list $(z_w)_{w\in W}$ over Z, once as nested coefficient list $(h_0, \dots, h_m)$ over $H(A_{m-1})$. In each case we cost addition and multiplication in Z as 1 unit. Let

$$M = |W(A_m)| = \dim_Z H(A_m) = (m+1)!$$

The sum of two elements $h, h' \in H(A_m)$ obviously needs M operations in Z, in either representation.

4.1. **Simple coefficient lists.** We now determine the cost of multiplying two elements in $H(A_m)$ where each is represented as a simple coefficient list $(z_w)_{w\in W}$ over Z. Consider first the product of a single basis element T_w, $w \in W$, with a generator T_s, $s \in S$. The relations of H allow us to compute their product in H based on the formula (3.1).

Now suppose that

$$h = \sum_{w\in W} z_w T_w \in H(A_m),$$

with coefficients $z_w \in \mathsf{Z}$. We represent h as its list of coefficients $(z_w)_{w\in W}$. Then

$$h\mathsf{T}_s = \sum_{w\in W} z'_w \mathsf{T}_w \in \mathsf{H}(A_m), \tag{4.1}$$

where

$$z'_w = \begin{cases} qz_{ws}, & \ell(ws) > \ell(w), \\ z_{ws} + (q-1)z_w, & \ell(ws) < \ell(w). \end{cases}$$

On the basis of this, we can define an operation of T_s, $s \in S$, on coefficient lists as

$$(z_w)_{w\in W} \star \mathsf{T}_s = (z'_w)_{w\in W}. \tag{4.2}$$

Then $(z_w)_{w\in W} \star \mathsf{T}_s$ is the coefficient list of $h\mathsf{T}_s$.

For $b \in \mathsf{Z}$, we set $b(z_w)_{w\in W} = (bz_w)_{w\in W}$.

For an element $v = r_1 r_2 \cdots r_l \in W$ of length $\ell(v) = l$, with $r_1, \dots, r_l \in S$, we note that $h\mathsf{T}_v = (\dots((h\mathsf{T}_{r_1})\mathsf{T}_{r_2})\cdots\mathsf{T}_{r_l})$ and set

$$(z_w)_{w\in W} \star \mathsf{T}_v = (\dots(((z_w)_{w\in W} \star \mathsf{T}_{r_1}) \star \mathsf{T}_{r_2}) \star \cdots \star \mathsf{T}_{r_l}). \tag{4.3}$$

Then $(z_w)_{w\in W} \star \mathsf{T}_v$ is the coefficient list of $h\mathsf{T}_v$.

Finally, if

$$h' = \sum_{v\in W} b_v \mathsf{T}_v \in \mathsf{H}(A_m),$$

with coefficients $b_v \in \mathsf{Z}$ we can compute the product hh' recursively as

$$hh' = \sum_{v\in W} hb_v \mathsf{T}_v \in \mathsf{H}(A_m),$$

The coefficient list of hh' is

$$\sum_{v\in W} b_v (z_w)_{w\in W} \star \mathsf{T}_v, \tag{4.4}$$

where addition of coefficient lists is the usual vector addition.

Proof of Theorem 1.1(a). In order to determine the cost of a product, we proceed as follows. We first consider the cost of computing $(z_w)_{w\in W} \star \mathsf{T}_s = (z'_w)_{w\in W}$ using (4.1).

Computing z'_w, $w \in W$, needs at most 2 operations in Z. Hence, computing the coefficients z'_w requires a total of $2M$ operations. Consequently, computing the coefficient list $b_v(z_w)_{w\in W} \star \mathsf{T}_v$ for $v \in W$ and a coefficient $b_v \in \mathsf{Z}$ using (4.3) requires $(2\ell(v)+1)M$ operations. Finally, using the fact that

$$\sum_{v\in W} \ell(v) = \frac{M}{2}\binom{m+1}{2},$$

computing the coefficient list of the product hh' for $h' = \sum_{v\in W(A_m)} b_v \mathsf{T}_v$ requires

$$\sum_{v\in W(A_m)} (2\ell(v)+1)M = \Big(M + 2\sum_{v\in W(A_m)} \ell(v)\Big)M = (1 + \binom{m+1}{2})M^2$$

operations to produce M coefficient lists $b_v(z_w)_{w\in W}\star T_v$, which require at most a further M vector additions at a cost of M^2, yielding a total of

$$(2+\binom{m+1}{2})M^2 = \frac{m^2+m+4}{2}M^2.$$

operations. □

4.2. **Nested coefficient lists.** Now we consider multiplying two elements in $H(A_m)$ given as nested coefficient lists. The proof of Theorem 1.1(b) requires a definition and two lemmas.

Definition 4.1. For $0 \leqslant l \leqslant m$, denote by $c(m,l)$ the maximum over all $h \in H(A_m)$ and j, $l \leqslant j \leqslant m$, of the cost of computing a product $hT_{a(j,l)}$. Moreover, denote by $C(m,l)$ the maximum cost of computing a product hg over all $h \in H(A_m)$ and $g \in H(A_l)$.

For $l > m$, we set $c(m,l) = 0$, as the construction of $hT_{a(j,l)} \in H(A_j)$ from the element $h \in H(A_m)$ does not require any operations in Z.

Lemma 4.2. *Let* $m \geqslant l \geqslant 0$ *and* $M = (m+1)!$. *Then*

(i) $c(m,l) \leqslant \frac{3}{2}lM$;

(ii) $\sum_{i=1}^{l} c(m,i) \leqslant \frac{3}{4}l(l+1)M$.

Proof. (ii) is an obvious consequence of (i).

For (i), we first show that $c(m,l)$ satisfies the following recursion.

$$c(m,l) \leqslant (m-l)\,c(m-1,l) + (2l+1)\,m! + 2\sum_{i=1}^{l-1} c(m-1,i). \tag{4.5}$$

Clearly, $c(m,0) = 0$ since $T_{a(j,0)} = 1_H$. For $m \geqslant l > 0$, we can determine $c(m,l)$ recursively as follows. Let $h = \sum_{k=0}^{m} h_k T_{a(m,k)}$ for elements $h_k \in H(A_{m-1})$ and suppose that $hT_{a(j,l)} = \sum_{k=0}^{m} h'_k T_{a(m,k)}$ for elements $h'_k \in H(A_{m-1})$. We consider each of the cases of Proposition 3.2.

Cases (a) and (d) only occur for $l < m$. Each value of k which arises in these cases contributes at most $c(m-1,l)$ to the cost of computing h'_k. Thus the total cost arising from case (a) is at most $(m-j)\,c(m-1,l)$, and the total cost arising from case (d) is at most $(j-l)\,c(m-1,l)$. Therefore the total cost arising from cases (a) and (d) together is at most $(m-l)\,c(m-1,l)$.

In case (c), the ith summand contributes a cost of $c(m-1,i-1)$, giving a total contribution of at most $\sum_{i=1}^{l} c(m-1,i-1)$ towards the cost of computing the element $\sum_{i=1}^{l} h_{m-j+i}T_{a(j-1,i-1)}$ in $H(A_{m-1})$. Forming the sum requires $l-1$ additions at cost of at most $(l-1)m!$ operations. Multiplying this quantity by $q-1$ requires a further $m!$ operations, and adding h_{m-j} costs an additional $m!$ operations. Thus the total cost for case (c) is at most $(l+1)\,m! + \sum_{i=1}^{l} c(m-1,i-1)$.

The computation required for case (b) can make use of the computation done for case (c), if we record the elements $h_{m-j+i}T_{a(j-1,i-1)}$. For $1 \leqslant i \leqslant l$, set $k = i-1+m-j$ so that

$m-j+i=k+1$ and $m-j\leqslant k<m-j+l$. Thus in the course of the computations for case (c) we have already obtained the terms $h_{k+1}T_{a(j-1,k-m+j)}$ for $m-j\leqslant k<m-j+l$. For each of these values of k, we have, using the second case of Lemma 3.1:

$$h_{k+1}T_{a(j-1,l-1)}=(h_{k+1}T_{a(j-1,k+j-m)})T_{a(m-k-1,l-1-k+m-j)},$$

which is one of the terms needed for the computation in case (b). The cost of multiplying $h_{k+1}T_{a(j-1,k+j-m)}$ by $T_{a(m-k-1,l-1-k+m-j)}$ is $c(m-1,l-1-k+m-j)$. Thus the total cost of computing the terms $h_{k+1}T_{a(j-1,l-1)}$ for $m-j\leqslant k<m-j+l$ is at most

$$\sum_{k=m-j}^{m-j+l-1}c(m-1,l-1-k+m-j)=\sum_{i=1}^{l}c(m-1,l-i)=\sum_{i=1}^{l}c(m-1,i-1).$$

Multiplying each of the l terms by q requires another $l\,m!$ operations. Thus the total cost for case (b) is at most $l\,m!+\sum_{i=1}^{l}c(m-1,i-1)$.

Adding the four components, we find that the total cost $c(m,l)$ satisfies

$$c(m,l)\leqslant(m-l)\,c(m-1,l)+(2l+1)\,m!+2\sum_{i=1}^{l}c(m-1,i-1),$$

if $l\leqslant m$.

Using the fact that $c(m-1,0)=0$, we have $\sum_{i=1}^{l}c(m-1,i-1)=\sum_{i=1}^{l-1}c(m-1,i)$, and hence the recursion (4.5) holds.

For $l>0$, we now prove by induction on m that the statement

$$(*)\qquad c(m,l)\leqslant\tfrac{3}{2}l(m+1)!\text{ for all }l\text{ satisfying }0<l\leqslant m$$

holds for all $m\geqslant 0$.

For $m=0$ there is nothing to prove.

Assume now that $k\geqslant 1$ and that assertion $(*)$ holds for $m=k-1$. Then we have $c(k-1,l)\leqslant\frac{3}{2}l\,k!$ for $l<k$, and $c(k-1,l)=0$ for $l=k$. Moreover, using the recursion (4.5) and part (ii) for $m=k-1$,

$$\begin{aligned}c(k,l)&=(2l+1)k!+(k-l)\,c(k-1,l)+2\sum_{i=1}^{l-1}c(k-1,i)\\&\leqslant((2l+1)+\tfrac{3}{2}(k-l)\,l+\tfrac{3}{2}l(l-1))k!\\&=((2l+1)+\tfrac{3}{2}(k-1)\,l)k!\\&\leqslant(3l+\tfrac{3}{2}(k-1)\,l)k!,\end{aligned}$$

as $l\geqslant 1$, and we find that $c(k,l)\leqslant\frac{3}{2}(k+1)lk!=\frac{3}{2}l(k+1)!=\frac{3}{2}l(k+1)!$, as desired.

Hence $c(m,l)\leqslant\frac{3}{2}l(m+1)!=\frac{3}{2}lM$ for all $m\geqslant l\geqslant 0$. □

Lemma 4.3. *For $0\leqslant l\leqslant m$, we have $C(m,l)\leqslant(1+e)(l+1)!\,M$.*

Proof. If $l=0$ then $g\in H(A_0)=Z$ is a scalar, and multiplying h by a scalar costs $C(m,0)=M$ operations in the worst case. It is easy to see that $C(m,1)\leqslant 3M+3\leqslant 2!\cdot 3\cdot M$ and that $C(m,2)\leqslant\frac{31}{2}M+9\leqslant 3!\cdot 3\cdot M$. Hence the claim is true for $l=0,1,2$.

If $l > 2$ then $g = \sum_{k=0}^{l} g_k T_{a(l,k)} \in H(A_l)$ (for some $g_k \in H(A_{l-1})$) and, using Lemma 4.2, the cost of computing $hg = \sum_{k=0}^{l} (hg_k) T_{a(l,k)}$, where, for each k, we first compute hg_k and then multiply the result by $T_{a(l,k)}$ and then add the $l+1$ terms, satisfies

$$\begin{aligned} C(m,l) &\leqslant (l+1)C(m,l-1) + \sum_{k=0}^{l} c(m,k) + lM \\ &\leqslant (l+1)(C(m,l-1) + (\tfrac{3}{4} + \tfrac{1}{l+1})lM) \\ &\leqslant (l+1)(C(m,l-1) + lM). \end{aligned} \tag{4.6}$$

Define a function f by $f(0) = 1$ and $f(l) = 1 + \sum_{j=0}^{l-1} \frac{1}{j!}$ for $l \geqslant 1$.

We show by induction on l that $C(m,l) \leqslant f(l)\,(l+1)!\,M$ for $l > 1$. For $l = 2$ this follows from the above bound on $C(m,2)$.

Now suppose that $l > 2$ and that the assertion holds for $l-1$. Then, by (4.6) and the inductive hypothesis,

$$\begin{aligned} C(m,l) &\leqslant (l+1)(C(m,l-1) + lM) \\ &\leqslant (l+1)(f(l-1)\,l!\,M + lM) \\ &= (f(l-1) + \tfrac{1}{(l-1)!})\,(l+1)!\,M \\ &= f(l)\,(l+1)!\,M. \end{aligned}$$

Since $f(l) = 1 + \sum_{j=0}^{l-1} \frac{1}{j!} \leqslant 1 + \sum_{j=0}^{\infty} \frac{1}{j!} = 1 + e$, the result follows. □

The proof of Theorem 1.1(b) now follows as $C(m,m) \leqslant (1+e)M^2$.

Remark 4.4. The complexity analysis shows that computing in Iwahori-Hecke algebras of type A with nested coefficient lists is more efficient than with simple coefficient lists. Experiments with a prototype implementation in GAP4 [1] suggest that the speed-up achieved in practice is even more dramatic than the complexity analysis predicts. This may be due to the fact that certain costs arising with simple coefficient lists have not been taken into account. For example, when computing the coefficients z'_w in Equation (4.1), additional cost can arise from computing the product ws in W, from comparing the lengths of w and ws, and from locating the coefficient z_{ws} in a simple coefficient list.

4.3. **Acknowledgements.** The first and third authors acknowledge the support of the Australian Research Council Discovery Project DP140100416. The second author thanks the School of Mathematics and Statistics at the University of Western Australia for their hospitality during a visit to Perth. We thank the anonymous referees for their useful comments and suggestions.

References

[1] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.8.4*, 2016.
[2] Meinolf Geck, PyCox*: computing with (finite) Coxeter groups and Iwahori-Hecke algebras*, LMS J. Comput. Math. **15** (2012), 231–256. MR 2988815
[3] Meinolf Geck, Gerhard Hiß, Frank Lübeck, Gunter Malle, and Götz Pfeiffer, CHEVIE — *A system for computing and processing generic character tables*, Appl. Algebra Engrg. Comm. Comput. **7** (1996), 175–210. MR 99m:20017

[4] Meinolf Geck and Götz Pfeiffer, *Characters of finite Coxeter groups and Iwahori-Hecke algebras*, London Mathematical Society Monographs. New Series, vol. 21, Oxford University Press, New York, 2000. MR 2002k:20017
[5] Andrew Mathas, *Iwahori-Hecke algebras and Schur algebras of the symmetric group*, University Lecture Series, vol. 15, American Mathematical Society, Providence, RI, 1999. MR 1711316 (2001g:20006)

ACN: Lehrstuhl B für Mathematik, RWTH Aachen, Pontdriesch 10-16, 52062 Aachen, Germany

E-mail address: alice.niemeyer@mathb.rwth-aachen.de

CEP: Centre for the Mathematics of Symmetry and Computation, University of Western Australia, 35 Stirling Highway, Crawley, WA 6009, Australia

E-mail address: cheryl.praeger@uwa.edu.au

G.P.: School of Mathematics, Statistics and Applied Mathematics, National University of Ireland, Galway, University Road, Galway, Ireland

E-mail address: goetz.pfeiffer@nuigalway.ie