Korean J. Math. **24** (2016), No. 3, pp. 469–487
http://dx.doi.org/10.11568/kjm.2016.24.3.469

GENERALIZATION OF THE SCHENSTED ALGORITHM FOR RIM HOOK TABLEAUX

Jaejin Lee

Abstract. In [6] Schensted constructed the Schensted algorithm, which gives a bijection between permutations and pairs of standard tableaux of the same shape. Stanton and White [8] gave analog of the Schensted algorithm for rim hook tableaux. In this paper we give a generalization of Stanton and White's Schensted algorithm for rim hook tableaux. If k is a fixed positive integer, it shows a one-to-one correspondence between all generalized hook permutations $\mathcal{H}$ of size k and all pairs (P, Q), where P and Q are semistandard k-rim hook tableaux and k-rim hook tableaux of the same shape, respectively.

1. Introduction

In [6] Schensted constructed the Schensted algorithm, which gives a bijection between permutations π and pairs (P, Q) of standard tableaux of the same shape(see also [2]). After Knuth generalized it to semistandard tableaux in [3], various analogs of the Schensted algorithm came: versions for rim hook tableaux ([8,9]), shifted tableaux ([5,10]), shifted rim hook tableaux [4], oscillating tableaux [1], and skew tableaux [7].

Let k be a fixed positive integer. Stanton and White [8] gave the Schensted algorithm for k-rim hook tableaux, which identifies hook permutations of size k with pairs of k-rim hook tableaux of the same shape. In this paper we give a generalization of Stanton and White's Schensted

Received July 4, 2016. Revised September 1, 2016. Accepted September 1, 2016.
2010 Mathematics Subject Classification: 05E10.
Key words and phrases: partition, hook, rim hook, generalized hook permutation, rim hook tableau, semistandard rim hook tableau, Schensted algorithm.

algorithm for k-rim hook tableaux. It shows a one-to-one correspondence between all generalized hook permutations $\mathcal{H}$ of size k and all pairs (P, Q), where P and Q are semistandard k-rim hook tableaux and k-rim hook tableaux of the same shape, respectively. In particular, if all the hooks of $\mathcal{H}$ were of size 1 then this algorithm reduces to the Schensted algorithm for semistandard tableaux given by Knuth [3], and if $\mathcal{H}$ were a hook permutation of size k then this algorithm becomes the Schensted algorithm for rim hook tableaux given by Stanton and White [8].

In Section 2 we provide the definitions used in this paper. Section 3 describes the "bumping" algorithm which is the basic building block of the subsequent algorithms. It is an analog to Schensted "bumping." In Section 4 the "insertion" and "deletion" algorithms are given. In Section 5 we give the "encode" and "decode" algorithms and state the theorems which follow from these algorithms.

2. Preliminaries

We use the standard notation $\mathbb{N}$ for the set of all positive integers.

Definition 2.1. A *partition* λ of a nonnegative integer n is a sequence of nonnegative integers $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_\ell)$ such that

(1) $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\ell > 0$,
(2) $\sum_{i=1}^{\ell} \lambda_i = n$.

We write $\lambda \vdash n$, or $|\lambda| = n$. We say each term λ_i is a *part* of λ. The number of nonzero parts is called the *length* of λ and is written $\ell = \ell(\lambda)$.

We sometimes abbreviate the partition λ with $1^{j_1}2^{j_2}3^{j_3}\ldots$, where j_i is the number of parts of size i. Sizes which do not appear are omitted and if $j_i = 1$, then it is not written. Thus a partition $(5, 3, 2, 2, 2, 1) \vdash 15$ can be written $12^3 35$.

Definition 2.2. Let $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ be a partition. The *Ferrers diagram (shape)* D_λ of λ is the array of cells or boxes arranged in rows and columns, λ_1 in the first row, λ_2 in the second row, etc., with each row left-justified. That is,

$$D_\lambda = \{(i, j) \in \mathbb{N}^2 \mid 1 \leq i \leq \ell(\lambda), 1 \leq j \leq \lambda_i\},$$

where we regard the elements of D_λ as a collection of boxes in the plane with matrix-style coordinates. Sometimes we identify a partition with its diagram, so that $x \in \lambda$ should be interpreted as $x \in D_\lambda$. See Figure 2.1 for a Ferrers diagram of $\lambda = (6,5,5,3,2) \vdash 21$.

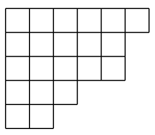


Figure 2.1

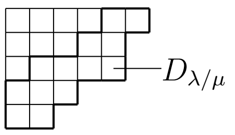


Figure 2.2

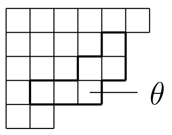


Figure 2.3

Definition 2.3. Let λ, μ be partitions with $D_\mu \subseteq D_\lambda$. A *skew shape* $D_{\lambda/\mu}$ is defined as the set-theoretic difference $D_\lambda \setminus D_\mu$. Figure 2.2 shows the skew shape $D_{\lambda/\mu}$ when $\lambda = (6,5,5,3,2)$ and $\mu = (4,3,1)$.

Definition 2.4. A *tableau (skew tableau)* is the shape λ (skew shape λ/μ), each cell of which contains a positive integer. We say that the tableau (skew tableau) T of shape λ (skew shape λ/μ) has *content*$(T) = \rho = (\rho_1, \rho_2, \cdots, \rho_m)$ if T contains ρ_1 1's, ρ_2 2's, $\cdots$, ρ_m m's.

A *hook* is a shape $\lambda = 1^i j$ and a *hook tableau* is a tableau of shape $\lambda = 1^i j$ all of whose entries are the same.

Definition 2.5. A skew shape θ is called a *rim hook* if θ is connected and contains no 2×2 block of cells. If θ is a rim hook, then its *head* is the upper rightmost cell in θ and its *tail* is the lower leftmost cell in θ. See Figure 2.3.

Definition 2.6. Suppose λ is a shape.

(1) The *outer rim* of λ is the set of cells in λ with no cells in λ immediately below and to the right.
(2) An *outer rim hook* of λ is a contiguous set of cells in the outer rim whose removal from λ leaves a shape.
(3) The *outside border* of λ is the collection of cells not in λ but immediately below and to the right of the cells in λ; or in the first row and to the right of λ; or in the first column and below λ.
(4) We call a contiguous set of cells in the outside border of λ a *snake*.

Figure 2.4 shows the outer rim and the outside border of λ. Here the set of cells marked with x forms an outer rim hook of λ.

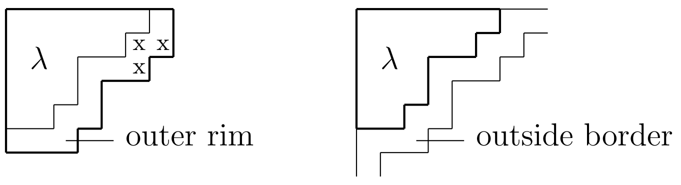


Figure 2.4

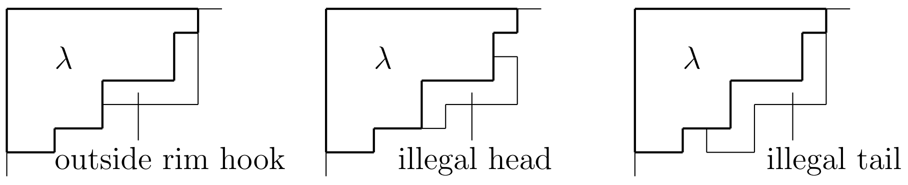


Figure 2.5

Suppose σ is a snake outside λ. Its *head* is the upper rightmost cell in σ and its *tail* is the lower leftmost cell. Either σ is an *outside rim hook* of λ, i.e., $\lambda \cup \sigma$ is a shape, or σ has an *illegal head*, an *illegal tail*, or both. See Figure 2.5. Similarly we define the *inner rim*, *inner rim hook*, *inside border* and *inside rim hook* of a skew shape α.

If σ is a snake or hook or rim hook, let $|\sigma|$ denote the number of cells in σ. We define a *k-hook* (*k-rim hook*) to be hook (rim hook) α with $|\alpha| = k$.

Let k be a fixed positive integer. From now on, unless we explicitly specify to contrary, all hooks and rim hooks in the tableau are k-hooks and k-rim hooks, respectively.

Definition 2.7. A *k-rim hook tableau P* of shape λ and content $\rho = (\rho_1, \cdots, \rho_m)$ with $\rho_i = k$ or 0 for each i and $\rho_m \neq 0$ is a tableau of shape λ such that

(1) If $\rho_1 = k$ and $\rho_2 = \cdots = \rho_m = 0$, a k-rim hook with all 1's and shape λ is a k-rim hook tableau.
(2) the set of τ containing m is a rim hook outside $\lambda - \tau$, and
(3) $\lambda - \tau$ is also a k-rim hook tableau.

We simliarly define a *skew k-rim hook tableau.*

Definition 2.8. A *semistandard k-rim hook tableau P* of shape λ and content $\rho_{(k)} = (\rho_1 k, \rho_2 k \ldots, \rho_m k)$ is defined recursively as follows. If $\rho_1 = 1$ and $\rho_2 = \cdots = \rho_m = 0$, a k-rim hook with all 1's and shape λ is a semistandard k-rim hook tableau. Let $\rho_m \neq 0$. If there is a k-rim hook

τ containing the m's in P such that the removal of τ from P leaves a semistandard k-rim hook tableau, then P is a semistandard k-rim hook tableau. We define a *skew semistandard k-rim hook tableau* in a similar way.

Figure 2.6 shows examples of a 4-rim hook tableau P and a semistandard 4-rim hook tableau Q of shape $(8, 7, 5, 4, 4, 3, 1)$, respectively. Here $\text{content}(P) = (4, 4, 4, 4, 4, 4, 0, 4, 4)$ and $\text{content}(Q) = (12, 4, 4, 4, 0, 8)$.

$$P = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 1&1&1&3&8&8&9&9\\ \hline 1&2&3&3&8&9&9\\ \cline{1-7} 2&2&3&5&8\\ \cline{1-5} 2&4&4&5\\ \cline{1-4} 4&4&5&5\\ \cline{1-4} 6&6&6\\ \cline{1-3} 6\\ \cline{1-1}\end{array} \qquad Q = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 1&1&1&1&6&6&6&6\\ \hline 1&1&1&1&6&6&6\\ \cline{1-7} 1&1&1&3&6\\ \cline{1-5} 1&2&2&3\\ \cline{1-4} 2&2&3&3\\ \cline{1-4} 4&4&4\\ \cline{1-3} 4\\ \cline{1-1}\end{array}$$

Figure 2.6

If P is a k-rim hook tableau, we write $\kappa_P\langle r\rangle$ (or just $\kappa\langle r\rangle$) for a rim hook of P containing r.

Definition 2.9. Let λ be a shape and σ be a k-snake outside λ. *Slideup* (λ, σ) is the k-snake outside λ whose tail is adjacent to the head of σ. Note that neither σ nor Slideup (λ, σ) need not be a rim hook outside λ. In fact, σ will have a legal head on λ if and only if Slideup (λ, σ) will have an illegal tail on λ. Similarly we define *Slidedown* (λ, σ). See Figure 2.7 for Slideup (λ, σ) and Slidedown (λ, σ).

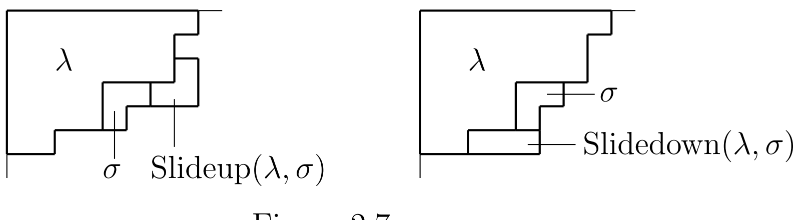


Figure 2.7

Before we describe the Schensted algorithm, we need the Bump algorithm. It will fix a snake outside λ with an illegal head or tail so that the resulting new set is an outside rim hook of a certain shape $\hat{\lambda}$. Let τ be a rim hook of skew shape and $x = (i, j) \in \tau$.

Algorithm Bump (Input: τ, x, direction; Output: $\hat{x}$)

```
begin
        if direction is outward then
            $\hat{x} \leftarrow (i+1, j+1)$
        else (* direction is inward *) then
            $\hat{x} \leftarrow (i-1, j-1)$
end.
```

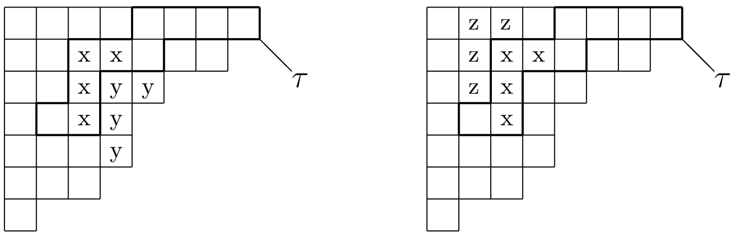


Figure 2.8

Definition 2.10. Let us write Bump(τ, x, direction) to mean the resulting cell $\hat{x}$ in Bump(τ, x, direction; $\hat{x}$). If τ is a rim hook of skew shape and $\psi \subseteq \tau$, then we write

$$\mathrm{Bump}_\tau(\psi,\ \text{direction}) = \{\mathrm{Bump}\ (\tau, x,\ \text{direction}) \mid x \in \psi\}.$$

If ψ is the set of cells marked with x's in Figure 2.8, the set of cells marked with y's shows $\mathrm{Bump}_\tau(\psi,\ \text{out})$ while the set of cells indicated with z's shows $\mathrm{Bump}_\tau(\psi,\ \text{in})$.

Suppose H is a hook tableau of size k. Thus H consists of k cells with a fixed entry. Let cont(H) denote the content of H.

Definition 2.11. The array $\mathcal{H} = \begin{pmatrix} H(1) & H(2) & \dots & H(m) \\ H_1 & H_2 & \dots & H_m \end{pmatrix}$ of k-hook tableaux is called a *generalized hook permutation* of length m if the following conditions hold:

(1) $H(i)$ has the same shape as H_i for $1 \leq i \leq m$,
(2) $\mathrm{cont}(H(i)) = i^k$ for $1 \leq i \leq m$.

Let $\mathcal{H}^{\mathrm{t}}$ and $\mathcal{H}_{\mathrm{b}}$ stand for the top and bottom row of $\mathcal{H}$, respectively, and define

$$\begin{aligned}\mathrm{cont}(\mathcal{H}^{\mathrm{t}}) &= \mathrm{cont}(H(1)) \cup \mathrm{cont}(H(2)) \cup \cdots \cup \mathrm{cont}(H(m)),\\ \mathrm{cont}(\mathcal{H}_{\mathrm{b}}) &= \mathrm{cont}(H_1) \cup \mathrm{cont}(H_2) \cup \cdots \cup \mathrm{cont}(H_m).\end{aligned}$$

Note that $\mathrm{cont}(\mathcal{H}^{\mathrm{t}}) = (k, k, \cdots, k) = 1^k 2^k \cdots m^k$. Figure 2.9 shows a generalized hook permutation of lengh 7. Here $\mathrm{cont}(\mathcal{H}_{\mathrm{b}}) = (4, 4, 8, 4, 8)$ and $\mathrm{cont}(\mathcal{H}^{\mathrm{t}}) = (4, 4, 4, 4, 4, 4, 4)$.

$\mathcal{H} =$

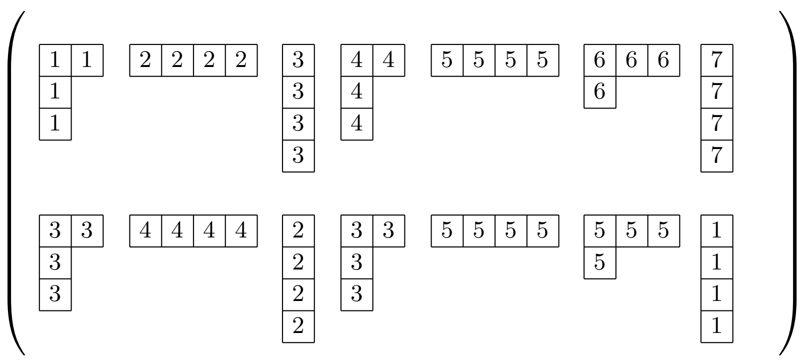


Figure 2.9

3. Bumping algorithms

Now we describe a procedure called the "bumping algorithm" which is the basic building block of the subsequent algorithms. The bumping algorithm shows us how an area within a shape, called the *bumping hook,* changes two tableaux, one a semistandard rim hook tableau and the other a skew semistandard rim hook tableau. The result of this procedure is a new bumping hook and two new semistandard tableaux.

We describe two such algorithms, *BumpOut* and *BumpIn.* The movement of BumpOut is outward while the movement of BumpIn is inward. However, we will analyze only BumpOut in detail because these two algorithms are "mirror images"of one another.

The input to this algorithm is a pair of tableaux (T, S), whose overlap is called the *bumping hook*, satisfying Conditions C_1–C_4. The result is a new pair of tableaux $(\hat{T}, \hat{S})$ which also satisfy these four conditions:

C_1: T is a semistandard rim hook tableau of shape λ with entries $\leq i$.

C$_2$: S is a skew semistandard rim hook tableau of shape α with entries $\geq i$. We assume j is the smallest entry in S.

C$_3$: $\sigma = \lambda \cap \alpha$ is an outer rim hook of λ.

C$_4$: $\sigma = \lambda \cap \alpha$ an inner rim hook of α and $|\sigma| = k$.

We call σ the *bumping hook*. It is convenient, specially for the termination, to assume from now on that any (skew)semistandard rim hook tableau has ∞ in every cell in its outside border and 0 in every cell in its left and top borders.

Associated with the pair (T, S) the entry of the bumping hook σ is a value j which appears in S and is smallest in S. Let τ be the rightmost k-rim hook containing the j's in S such that the removal of τ from S leaves a skew semistandard k-rim hook tableau. Since $|\sigma| = |\tau| = k$, we have the following three basic cases:

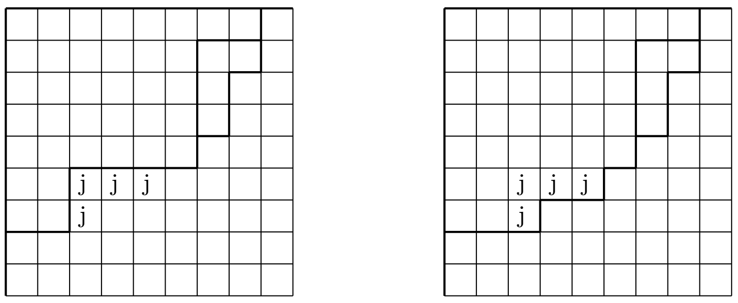
Case (1) σ and τ are disjoint ($\sigma \cap \tau = \emptyset$).
Case (2) σ and τ overlap ($\sigma \cap \tau \neq \emptyset$ and $\sigma \neq \tau$).
Case (3) σ and τ coincide ($\sigma = \tau$).

Algorithm BumpOut (Input: T,S; Output: $\hat{T}$,$\hat{S}$)

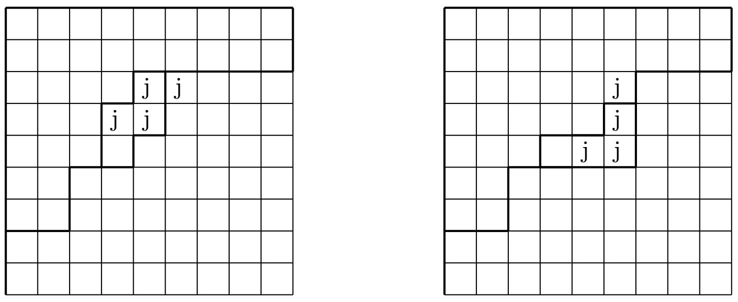
```
begin
  if $\sigma \cap \tau = \emptyset$ then
    $\hat{S} \leftarrow S - \tau(j)$
    $\hat{T} \leftarrow T \cup \tau(j)$
  else if $\sigma \cap \tau \neq \emptyset$ and $\sigma \neq \tau$ then
    $\tau' \leftarrow \text{Bump}_{\tau}(\tau \cap \sigma, \text{out}) \cup (\tau - (\tau \cap \sigma))$
    $\hat{S} \leftarrow S - \tau(j)$
    $\hat{T} \leftarrow T \cup \tau'(j)$
  else $(* \ \sigma = \tau \ *)$
    $\tau' \leftarrow \sigma$
    repeat
      $\tau' \leftarrow \text{Slidedown}(\lambda - \sigma, \tau')$
    until $\tau'$ is legal on $\lambda$.
    $\hat{S} \leftarrow S - \tau(j)$
    $\hat{T} \leftarrow T \cup \tau'(j)$
end.
```

In the above $\tau(i)$ denotes k-rim hook containing the i's. See Figures 3.1. In each figure, the boundaries of λ and α will be indicated in heavy line, so that it clearly shows σ enclosed in heavy outline. T, S are given in the left figures. Right figures show the resulting $\hat{T}, \hat{S}$ from BumpOut. Verification that in each case Conditions C_1–C_4 are maintained is easily accomplished by careful analysis of the various cases. Since the basic idea for this verification is similar to White's in [9], details are omitted.

Case (1) $\sigma \cap \tau = \emptyset$



Case (2) $\sigma \cap \tau \neq \emptyset$ and $\sigma \neq \tau$



Case (3) $\sigma = \tau$

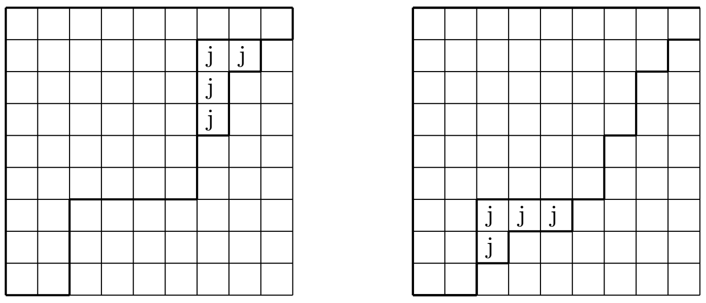


Figure 3.1

We now describe the BumpIn algorithm. This algorithm can be obtained from BumpOut by reversing the construction in BumpOut. But Algorithm BumpIn differs significantly from BumpOut in Case (3). This case provides for the only circumstances under which a hook can be bumped out of the tableau. This occurs when τ' cannot be constructed because Slideup encounters cells above the first row. See Figure 3.2.

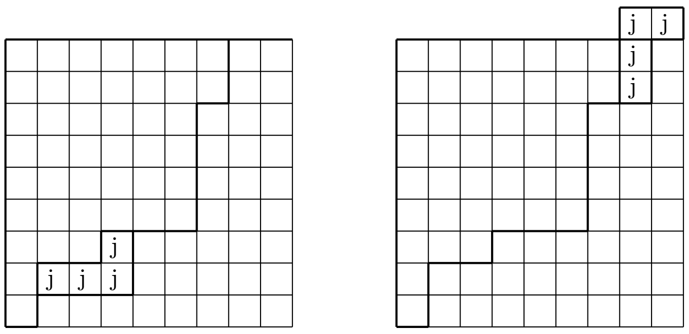


Figure 3.2

This special case will stop the Delete algorithm in Section 4 and a hook of j's will be removed from T.

BumpIn has two additional outputs: *timetostop*, which indicates when the special circumstances described above happen, and j, the value in τ at the time of this occurrence.

Algorithm BumpIn (Input:T,S; Output:$\hat{T}$,$\hat{S}$, j,timetostop)

```
begin
   if $\sigma \cap \tau = \emptyset$ then
      $\hat{S} \leftarrow S \cup \tau(j)$
      $\hat{T} \leftarrow T - \tau(j)$
   else if $\sigma \cap \tau \neq \emptyset$ and $\sigma \neq \tau$ then
      $\tau' \leftarrow \text{Bump}_{\tau}(\tau \cap \sigma, \text{in}) \cup (\tau - (\tau \cap \sigma))$
      $\hat{S} \leftarrow S \cup \tau'(j)$
      $\hat{T} \leftarrow T - \tau(j)$
   else $(* \ \sigma = \tau \ *)$
      $\tau' \leftarrow \sigma$
      repeat
           $\tau' \leftarrow \text{Slideup}(\alpha - \sigma, \tau')$
      until $\tau'$ is legal on $\alpha$ or $\tau'$ encounters cells above 1st row
```

```
    if $\tau'$ encounters cells above 1st row then
        timetostop $\leftarrow$ true
    else (* $\tau'$ is legal on $\alpha$ *)
    $\hat{S} \leftarrow S \cup \tau'(j)$
    $\hat{T} \leftarrow T - \tau(j)$
end.
```

Since every construction in BumpOut is inverted in BumpIn, we have the following crucial lemma:

LEMMA 3.1. *BumpOut and BumpIn are inverse algorithms. That is, the procedure:*

```
begin
    BumpOut $(T, S; \hat{T}, \hat{S})$
    BumpIn $(\hat{T}, \hat{S}; \hat{\hat{T}}, \hat{\hat{S}}, j,$ timetostop,)
end.
```

yields $T = \hat{\hat{T}}, S = \hat{\hat{S}}$ *and timetostop = false; and the procedure:*

```
begin
    BumpIn $(T, S; \hat{T}, \hat{S}, j,$ timetostop)
    if not timetostop then
      BumpOut $(\hat{T}, \hat{S}; \hat{\hat{T}}, \hat{\hat{S}})$
end.
```

also yields $T = \hat{\hat{T}}, S = \hat{\hat{S}}$.

4. Schensted insertion and deletion algorithms

Using the Bumping algorithms in Section 3 we now describe insertion and deletion algorithms which are rim hook analogs of the ordinary Schensted insertion and deletion algorithms for identifying permutations with pairs of standard tableaux.

Algorithm Insert has as input a semistandard k-rim hook tableau, and a hook tableau of size k. The hook tableau must first be positioned so that we can apply Algorithm BumpOut in Section 3. Suppose λ is a shape and τ is a hook of size k.

Algorithm Position (Input: λ, τ; Output: $\hat{\tau}$)

```
begin
```

```
    $\hat\tau \leftarrow \tau$
    repeat
        $\hat\tau \leftarrow$ Slideup$(\emptyset, \hat\tau)$
    until $\lambda \cap \hat\tau = \emptyset$
    repeat
        $\hat\tau \leftarrow$ Slidedown$(\lambda, \hat\tau)$
    until $\hat\tau$ is legal on $\lambda$
end.
```

If $\hat\tau$ has an illegal tail on λ, then Slidedown$(\lambda, \hat\tau)$ has a legal head on λ. Thus both loops in the above algorithm must terminate and the resulting $\hat\tau$ is an outside rim hook of λ. See Figure 4.1.

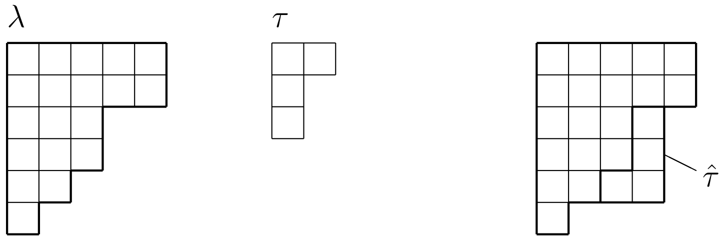


Figure 4.1

Let T be a semistandard rim hook tableau. We denote by T_j the semistandard rim hook tableau obtained from T by removing all the rim hooks whose entry is larger than j. Similarly, we denote by T^j the skew semistandard rim hook tableau obtained from T by removing all the rim hooks whose entry is smaller than or equal to j. See Figure 4.2.

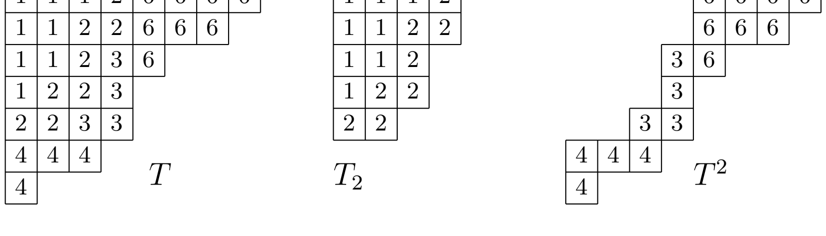


Figure 4.2

Now suppose T has shape μ and content $\rho_{(k)} = (\rho_1 k, \rho_2 k, \cdots, \rho_m k)$. Let σ be a hook of size k with j's in the cells. The output from Algorithm Insert will be another semistandard rim hook tableau $\hat{T}$ of content $\hat\rho_{(k)} =$

$(\rho_1 k, \rho_2 k, \cdots, (\rho_j+1)k, \cdots, \rho_m k)$ and shape $\hat{\mu}$ such that $\hat{\sigma} = \hat{\mu} - \mu$ is an outside rim hook of μ.

Algorithm Insert (Input: T, σ, j; Output: $\hat{T}, \hat{\sigma}$)

```
begin
        Position $(\lambda, \sigma; \sigma_1)$
        $A \leftarrow T_j \cup \sigma_1(j)$
        $B \leftarrow T^j$
        while $B$ contains finite entries do
            BumpOut $(A, B; \hat{A}, \hat{B})$
            $\sigma_{\hat{A}\hat{B}} \leftarrow$ bumping hook of $\hat{A}$ and $\hat{B}$
            $A \leftarrow \hat{A}$
            $B \leftarrow \hat{B}$
        $\hat{T} \leftarrow \hat{A}$
        $\hat{\sigma} \leftarrow \sigma_{\hat{A}\hat{B}}$
end.
```

At the end, $\hat{A} = \hat{T}$, and $\hat{B}$ contains infinite entries only. By the result of Section 3, $\sigma_{\hat{A}\hat{B}}$ is the intersection of $\hat{T}$ and $\hat{B}$, and $\sigma_{\hat{A}\hat{B}}$ is an outside rim hook of μ.

Figure 4.3, Figure 4.4 and Figure 4.5 give an example of the Insert algorithm. A semistandard 4-rim hook tableau T and a hook σ of size 4 (with 3's in the cells) are given in Figure 4.3. Then Figure 4.4 (a)–(e) describe A and B (with cells in σ_{AB} indicated in heavy outline) at each pass through the main loop. Figure 4.5 shows the new semistandard rim hook tableau $\hat{T}$ and rim hook $\hat{\sigma}$ obtained from Algorithm Insert using T and σ in Figure 4.3.

$$T = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 1&1&1&2&3&3&3&4\\ \hline 1&2&2&2&3&4&4&4\\ \hline 2&2&5&5&7&7&7&7\\ \hline 2&5&5&6\\ \cline{1-4} 2&6&6&6\\ \cline{1-4} 6&6&6&6\\ \cline{1-4}\end{array} \qquad \sigma = \begin{array}{|c|c|}\hline 3&3\\ \hline 3\\ \cline{1-1} 3\\ \cline{1-1}\end{array}$$

Figure 4.3

We now describe Algorithm Delete which reverses the Insert algorithm. In this algorithm we use the BumpIn algorithm in the previous section.

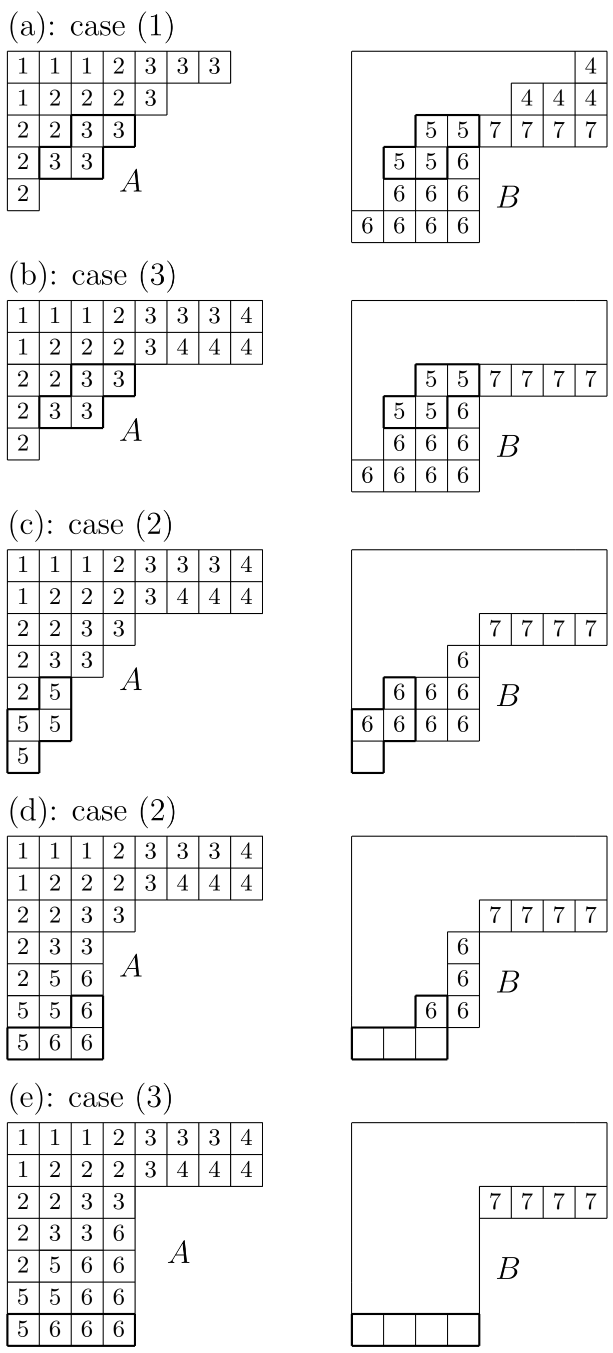


Figure 4.4

$$\hat{T} = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 1&1&1&2&3&3&3&4\\\hline 1&2&2&2&3&4&4&4\\\hline 2&2&3&3&7&7&7&7\\\hline 2&3&3&6\\\cline{1-4} 2&5&6&6\\\cline{1-4} 5&5&6&6\\\cline{1-4} 5&6&6&6\\\cline{1-4}\end{array}$$

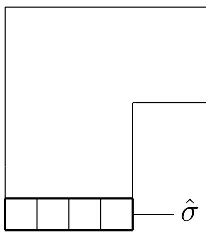


Figure 4.5

First, we need Algorithm Hook to reverse the Position algorithm described earlier. Suppose λ is a shape and τ ($|\tau| = k$) is an outside rim hook of λ. The output from the Hook algorithm will be a hook $\hat{\tau}$ of the size k.

```
*Algorithm Hook (Input: $\lambda, \tau$; Output: $\hat{\tau}$)*
**begin**
    $\hat{\tau} \leftarrow \tau$
    **repeat**
        $\hat{\tau} \leftarrow$ Slideup$(\lambda, \hat{\tau})$
    **until** $\hat{\tau}$ is contained in the first row
    **repeat**
        $\hat{\tau} \leftarrow$ Slidedown$(\emptyset, \hat{\tau})$
    **until** $\hat{\tau}$ intersects the first column
**end**.
```

Certainly we have the following lemma.

Lemma 4.1. *Position and Hook are inverses of one another. That is, the procedure:*

```
**begin**
        Position $(\lambda, \tau; \hat{\tau})$
        Hook $(\lambda, \hat{\tau}\ ; \hat{\hat{\tau}})$
**end**.
```

yields $\tau = \hat{\hat{\tau}}$; and the procedure:

```
**begin**
        Hook $(\lambda, \tau; \hat{\tau})$
        Position $(\lambda, \hat{\tau}; \hat{\hat{\tau}})$
**end**.
```

also yields $\tau = \hat{\hat{\tau}}$ under the special circumstances that Hook will be used.

The Delete algorithm has as input a semistandard rim hook tableau T of shape μ and content $\rho_{(k)} = (\rho_1 k, \rho_2 k, \cdots, \rho_m k)$ and an outer rim hook σ ($|\sigma| = k$) of μ.

Algorithm Delete will produce the following: a semistandard rim hook tableau $\hat{T}$ of shape $\hat{\mu}$ and content $\hat{\rho}_{(k)}$, a value j, and a hook $\hat{\sigma}$ such that

(a) $\hat{\rho}_{(k)} = (\rho_1 k, \rho_2 k, \cdots, (\rho_j - 1)k, \cdots, \rho_m k)$,
(b) $|\hat{\sigma}| = |\sigma| = k$,
(c) $\hat{\mu} = \mu - \sigma$.

```
*Algorithm Delete (Input: $T, \sigma$; Output: $\hat{T}, \hat{\sigma}, j$)*
**begin**
        $A \leftarrow T$
        $B \leftarrow \sigma(\infty)$
        **repeat**
            BumpIn ($A, B$; $\hat{A}, \hat{B}, j$, timetostop)
            $\sigma_{\hat{A}\hat{B}} \leftarrow$ bumping hook of $\hat{A}$ and $\hat{B}$
            $A \leftarrow \hat{A}$
            $B \leftarrow \hat{B}$
        **until** timetostop
        $\hat{T} \leftarrow \hat{A} \cup \hat{B}$
        $\lambda \leftarrow$ shape of $\hat{A}$
        Hook ($\lambda - \sigma_{\hat{A}\hat{B}}, \sigma_{\hat{A}\hat{B}}; \hat{\sigma}$)
**end**.
```

Figure 4.6 shows an example for Algorithm Delete. The input T and σ (with cells of σ indicated in heavy outline) are given in Figure 4.6–(a). Then Figure 4.6–(b) shows the output $\hat{T}$ and $\hat{\sigma}$ obtained from the Delete algorithm.

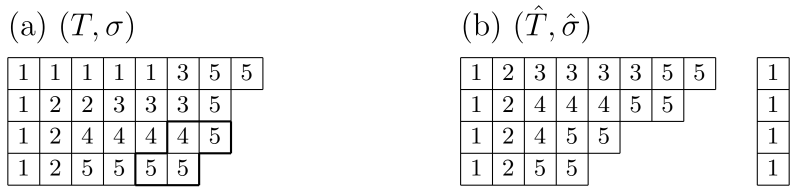


Figure 4.6

From Lemma 3.1 and Lemma 4.1 we have the following theorem.

Theorem 4.2. *Insert and Delete are inverses of one other. That is, the procedure:*

```
begin
        Insert $(T, \sigma, j; \hat{T}, \hat{\sigma})$
        Delete $(\hat{T}, \hat{\sigma}; \hat{\hat{T}}, \hat{\hat{\sigma}}, j)$
end.
```

yields $T = \hat{\hat{T}}, \sigma = \hat{\hat{\sigma}}$; and the procedure:

```
begin
        Delete $(T, \sigma; \hat{T}, \hat{\sigma}, j)$
        Insert $(\hat{T}, \hat{\sigma}, j; \hat{\hat{T}}, \hat{\hat{\sigma}})$
end.
```

will yield $T = \hat{\hat{T}}, \sigma = \hat{\hat{\sigma}}$.

5. Schensted encoding and decoding algorithms

We now describe the Shensted algorithm which assigns to a generalized hook permutation $\mathcal{H}$ of shape $(\tau^{(1)}, \tau^{(2)}, \cdots, \tau^{(m)})$ a pair (P, Q) of tableaux, where P is a semistandard rim hook tableau of content $\mathcal{H}_b$ and Q is a rim hook tableau of content $\mathcal{H}^t$ and the same shape as P.

Algorithm Encode (Input: $\mathcal{H}$; Output: P, Q)

```
begin
    $P, Q \leftarrow \emptyset$
    for $i \leftarrow 1$ to $m$
        $j \leftarrow$ content of $H_i$
        Insert $(P, \tau^{(i)}, j; \hat{P}, \sigma)$
        $\hat{Q} \leftarrow Q \cup \sigma(i)$
        $Q \leftarrow \hat{Q}$
        $P \leftarrow \hat{P}$
end.
```

Figure 5.1 shows the final semistandard 4-rim hook tableau P and 4-rim hook tableau Q obtained from the Encode algorithm for a generalized hook permutation $\mathcal{H}$ given in Figure 2.9 of the Section 2.

Now we construct Algorithm Decode which is the inverse of Encode. Suppose P is a semistandard rim hook tableau of shape λ and content $\rho_{(k)} = (\rho_1 k, \rho_2 k, \cdots, \rho_m k)$ and Q is a rim hook tableau of shape λ and

$$P = \begin{array}{cccccccc} 1 & 3 & 3 & 3 & 3 & 3 & 5 & 5 \\ 1 & 3 & 4 & 4 & 4 & 5 & 5 & \\ 1 & 3 & 4 & 5 & 5 & & & \\ 1 & 3 & 5 & 5 & & & & \\ 2 & & & & & & & \\ 2 & & & & & & & \\ 2 & & & & & & & \\ 2 & & & & & & & \end{array} \qquad Q = \begin{array}{cccccccc} 1 & 1 & 2 & 2 & 4 & 4 & 5 & 5 \\ 1 & 2 & 2 & 4 & 4 & 5 & 5 & \\ 1 & 3 & 3 & 6 & 6 & & & \\ 3 & 3 & 6 & 6 & & & & \\ 7 & & & & & & & \\ 7 & & & & & & & \\ 7 & & & & & & & \\ 7 & & & & & & & \end{array}$$

Figure 5.1

content $(k, k, \cdots, k)$. Algorithm Decode has as input a pair (P, Q). The result is a generalized hook permutation $\mathcal{H}$.

Algorithm Decode (Input: P, Q; Output:$\mathcal{H}$)

```
begin
    $\sigma \leftarrow \kappa_Q\langle m\rangle$
    Delete $(P, \sigma;\ \tilde{P}, \tilde{\sigma}, j)$
        $\tilde{Q} \leftarrow Q - \kappa_Q\langle m\rangle(m)$
        Decode $(\tilde{P}, \tilde{Q}; \hat{\tilde{P}}, \hat{\tilde{Q}},\ \mathcal{H}_1 = \begin{pmatrix} H(1) & H(2) & \dots & H(m-1) \\ H_1 & H_2 & \dots & H_{m-1} \end{pmatrix})$
        $H_m \leftarrow \tilde{\sigma}(j)$
        $H(m) \leftarrow \tilde{\sigma}(m)$
end.
```

Clearly we have the following theorem.

Theorem 5.1. *Encode and Decode are inverses of one another. That is, the procedure:*

```
begin
        Decode $(P, Q; \hat{P}, \hat{Q}, \mathcal{H})$
        Encode $(\mathcal{H}; \hat{\hat{P}}, \hat{\hat{Q}})$
end.
```

yields $P = \hat{\hat{P}}$ and $Q = \hat{\hat{Q}}$, and the procedure:

```
begin
        Encode $(\mathcal{H}; P, Q)$
        Decode $(P, Q; \hat{\mathcal{H}})$
end.
```

yields $\mathcal{H} = \hat{\mathcal{H}}$.

The algorithms and theorems of Section 3, 4 and 5 yield the following theorem:

Theorem 5.2. *Algorithms Encode and Decode construct a bijection between all generalized hook permutations $\mathcal{H}$ and all pairs (P, Q), where P is a semistandard k-rim hook tableau of shape λ and Q is a k-rim hook tableau of shape λ. Furthermore* $\mathrm{cont}(P) = \mathrm{cont}(\mathcal{H}_b)$ *and* $\mathrm{cont}(Q) = \mathrm{cont}(\mathcal{H}^t) = (k, k, \cdots, k) = 1^k 2^k \cdots m^k$.

References

[1] A. Berele, *A Schensted-type correspondence for the symplectic group,* J. Combin. Theory Ser. A **43** (1986), 320–328.

[2] D. E. Knuth, *Sorting and Searching;* The Art of Computer Programming, **Vol. 3** (1973), Addison-Wesley, Mass.

[3] D. E. Knuth, *Permutations, matrices, and generalized Young tableaux,* Pacific J. Math. **34** (1970), 709–727.

[4] J. Lee, *A Schensted algorithm for shifted rim hook tableaux,* J. Korean Math. Soc. **31** (1994), 179–203.

[5] B. E. Sagan, *Shifted tableaux, Schur Q-functions and a conjecture of R. Stanley,* J. Combin. Theory Ser. A **45** (1987), 62–103.

[6] C. Schensted, *Longest increasing and decreasing subsequences,* Canad. J. Math. **13** (1961), 179–191.

[7] B. Sagan and R. Stanley, *Robinson-Schensted algorithms for skew tableaux,* J. Combin. Theory Ser. A **55** (1990), 161–193.

[8] D. W. Stanton and D. E. White, *A Schensted algorithm for rim hook tableaux,* J. Combin. Theory Ser. A **40** (1985), 211–247.

[9] D. E. White, *A bijection proving orthogonality of the characters of S_n,* Advances in Math. **50** (1983), 160–186.

[10] D. R. Worley, A Theory of Shifted Young Tableaux, Ph. D. thesis (1984), M.I.T.

Jaejin Lee
Department of Mathematics
Hallym University
Chunchon 24252, Korea
E-mail: `jjlee@hallym.ac.kr`