Young Tableaux

October 31, 2001

1 Introduction

A *Young Tableau of shape* $(n_1, n_2, \ldots, n_m)$, where $n_1 \geq n_2 \geq n_3 \geq \cdots \geq n_m > 0$ is an arrangement of $n_1 + n_2 + \cdots + n_m$ distinct positive integers in an array of m left-justified rows, with n_i elements in row i, such that integers in each row are increasing from left to right and integers in each column are increasing from top to bottom. For example,

$$\begin{array}{lllll} 1 & 10 & 12 & 14 & 22 \\ 11 & 13 & 15 & 20 & 27 \\ 17 & 21 & 23 & & \\ 19 & 28 & & & \\ 24 & & & & \end{array}$$

is a Young Tableau of shape $(5, 5, 3, 2, 1)$. We will be interested in questions such as: how many distinct Young tableaux can we make with n elements? how many distinct Young tableaus are there with a given shape $(n_1, n_2, \ldots, n_m)$? how can we generate all Young tableaux with n elements or all Young tableaux of a given shape? how can we generate a random Young tableau? Answering these questions will lead us to the famous Robinson-Schensted correspondence between permutations and pairs of tableaux and the connection between involutions and tableaux.

Let us study an example carefully before we proceed further. Suppose we want to generate all Young tableaux with 4 elements. The possible shapes of Young tableaux with 4 elements are simply the integers partitions of 4.

```
In[1]:= Partitions[4]
Out[1]= {{4}, {3, 1}, {2, 2}, {2, 1, 1}, {1, 1, 1, 1}}
```

Combinatorica contains a function called `Tableaux` that takes as input a shape and returns all Young tableaux of that shape. Below we use `Tableaux` to generate all Young tableux with 4 elements. We see from this example that there is one Young tableaux of shape (4), three of shape $(3, 1)$, two of shape $(2, 2)$, three of shape $(2, 1, 1)$, and one of shape $(1, 1, 1, 1)$. Thus a total of 10 Young tableaux can be made from 4 elements.

```
In[2]:= Map[Tableaux, Partitions[4]] // ColumnForm
Out[2]= {{{1, 2, 3, 4}}}
        {{{1, 3, 4}, {2}}, {{1, 2, 4}, {3}}, {{1, 2, 3}, {4}}}
        {{{1, 3}, {2, 4}}, {{1, 2}, {3, 4}}}
        {{{1, 4}, {2}, {3}}, {{1, 3}, {2}, {4}}, {{1, 2}, {3}, {4}}}
        {{{1}, {2}, {3}, {4}}}
```

Combinatorica contains a function called `NumberOfTableaux` that returns the number of Young tableaux of a given shape.

```
In[9]:= Map[NumberOfTableaux, Partitions[4]]
Out[9]= {1, 3, 2, 3, 1}
```

2 Insertion and Deletion

Let P be a Young tableaux and let x be a positive integer not in P. Then x can be inserted into P and the algorithm to do this is illustrated in the following example. Suppose we want to insert 2 into the following tableaux:

$$\begin{array}{llll} 1 & 3 & 4 & 6 \\ 7 & 8 & & \\ 12 & & & \\ 13 & & & \end{array}$$

2 is inserted into the first row and there it displaces element 3, "bumping" it off to a subsequent row. This gives us the tableau

$$\begin{array}{llll} 1 & 2 & 4 & 6 \\ 7 & 8 & & \\ 12 & & & \\ 13 & & & \end{array}$$

and we want to insert 3 into the second row of this tableau. Inserting 3 into row 2 bumps off the 7 and gives us the tableau

$$\begin{array}{llll} 1 & 2 & 4 & 6 \\ 3 & 8 & & \\ 12 & & & \\ 13 & & & \end{array}$$

and we want to insert 7 into row 3. Inserting 7 into row 3 bumps off 12 and we get the tableau

$$\begin{array}{llll} 1 & 2 & 4 & 6 \\ 3 & 8 & & \\ 7 & & & \\ 13 & & & \end{array}$$

with 12 waiting to be inserted into row 4. Inserting 12 into row 4 bumps off the 13 and gives the tableau

$$\begin{array}{llll} 1 & 2 & 4 & 6 \\ 3 & 8 & & \\ 7 & & & \\ 12 & & & \end{array}$$

with 13 waiting to be inserted into row 5. The result of inserting 13 is the tableau

$$\begin{array}{llll} 1 & 2 & 4 & 6 \\ 3 & 8 & & \\ 7 & & & \\ 12 & & & \\ 13 & & & \end{array}$$

Combinatorica has a function called `InsertIntoTableau` that uses this "bumping" algorithm to insert into a tableau.

```
In[4]:= InsertIntoTableau[2, {{1, 3, 4, 6}, {7, 8}, {12}, {13}}]
Out[4]= {{1, 2, 4, 6}, {3, 8}, {7}, {12}, {13}}
```

Let P be a tableau and x be a positive integer not in P. We use P_{ij} to denote the element in row i and column j. For convenience we assume that the tableau is bordered on the left and on the top by 0's and on the bottom and on the right by ∞. The algorithm for insertion is described below.

1. Set $i \leftarrow 1$ and set $x_i \leftarrow x$.
2. Find a position j such $P_{i(j-1)} < x_i < P_{ij}$.
3. Set $x_{i+1} \leftarrow P_{ij}$, $r_i \leftarrow j$, and $P_{ij} \leftarrow x_i$.
4. If $x_{i+1} < \infty$ then increment i and return to Step (2).
5. Set $s \leftarrow i$ and $t \leftarrow j$ and terminate the algorithm.

Here is a quick, informal argument to show that the algorithm is correct, that is, it returns a tableau which contains all the elements in P along with x. The claim is that after each execution of Step (3), P is a tableau. It is clear, because of the j that is found in Step (2), that the row into which x_i in inserted continues to be in increasing order. Also, x_i replaces a larger element and so after insertion, x_i is smaller than the element below it. We only need to make sure that x_i is larger than the element above it. For $i = 1$, there is no element above x_i and so this is trivially true. When $i > 1$, note that x_i is an element that previously occupied row $(i - 1)$. In particular, x_i lived in row $(i - 1)$ and column r_{i-1}. Let $c = r_{i-1}$ and we have $P_{(i-1)(c-1)} < x_i$ and $x_i < P_{ic}$. So in row i, x_i will get inserted in or to the left of column c. Since everything in row $(i - 1)$ that is in or the left of column c is less than x_i, we have that after insertion, x_i is indeed larger than the element above it.

From the proof sketch above it is clear that the "bumping sequence" satisfies

$$x = x_1 < x_2 < \cdots < x_s < x_{s+1} = \infty$$

and the sequence of column indices where the insertions take place satisfies

$$r_1 \geq r_2 \geq \cdots \geq r_s = t.$$

Also, (s, t), the position where the last insertion takes place satisfies

$$P_{st} \neq \infty \qquad \text{and} \qquad P_{(s+1)t} = P_{s(t+1)} = \infty. \tag{1}$$

Given a tableau P and a position (s, t) such that equations in (1) are satisfied, it is possible to reverse each step of the the insertion operation and get the original tableau back. In the beginning of this section we presented an example in which we inserted the element 2 into the tableau

$$\begin{array}{llll} 1 & 3 & 4 & 6 \\ 7 & 8 & & \\ 12 & & & \\ 13 & & & \end{array}$$

and obtained the tableau

$$\begin{array}{llll} 1 & 2 & 4 & 6 \\ 3 & 8 & & \\ 7 & & & \\ 12 & & & \\ 13 & & & \end{array}$$

In this example, the position $(s, t) = (5, 1)$ and we can rewind the insertion operation by taking the element in this position, namely 13, and insert it into the previous row, replacing the largest element smaller than it. The element 12 that gets dislodged from row 4 then gets inserted into row 3 dislodging element 3 which in turn gets inserted into row 1 dislodging element 2. At this point the deletion algorithm stops and we have the original tableau. *Combinatorica* contains a function called `DeleteFromTableau` that is the implementation of the deletion algorithm. The following example illustrates that `InsertIntoTableau` and `DeleteFromTableau` can be thought of as inverses of each other.

```
In[3]:= InsertIntoTableau[2, {{1, 3, 4, 6}, {7, 8}, {12}, {13}}]
Out[3]= {{1, 2, 4, 6}, {3, 8}, {7}, {12}, {13}}

In[4]:= DeleteFromTableau[%, 5]
Out[4]= {{1, 3, 4, 6}, {7, 8}, {12}, {13}}
```

3 Robinson-Schensted-Knuth Correspondence

Starting with a permutation $p = (p_1, p_2, \ldots, p_n)$ we can construct a tableau by inserting elements $p_1, p_2, \ldots, p_n$ in that order starting with an empty tableau. There is a *Combinatorica* function called `ConstructTableau` to does this.

```
In[6]:= ConstructTableau[{3, 1, 5, 2, 4}]
Out[6]= {{1, 2, 4}, {3, 5}}
```

This is equivalent to inserting into a tableau repeatedly, as the following example shows.

```
In[10]:= InsertIntoTableau[3, {}]
Out[10]= {{3}}

In[11]:= InsertIntoTableau[1, %]
Out[11]= {{1}, {3}}

In[12]:= InsertIntoTableau[5, %]
Out[12]= {{1, 5}, {3}}

In[13]:= InsertIntoTableau[2, %]
Out[13]= {{1, 2}, {3, 5}}

In[14]:= InsertIntoTableau[4, %]
Out[14]= {{1, 2, 4}, {3, 5}}
```

It is tempting to think that this defines a bijection from n-permutations to tableaux containing the elements $1, 2, \ldots, n$. However, this is not the case and many different n-permutations can map onto the same tableau. The following experiment shows that there are 5 5-permutations that map on to the tableau $\{\{1, 2, 4\}, \{3, 5\}\}$.

```
In[19]:= Length[Select[Map[ ConstructTableau, Permutations[5] ],
                              (#=={{1, 2, 4}, {3, 5}})&
                 ]
         ]
Out[19]= 5
```

One of these is $(3, 1, 5, 2, 4)$. Another is $(1, 3, 5, 2, 4)$, as verified in the following example:

```
In[20]:= ConstructTableau[{1, 3, 5, 2, 4}]
Out[20]= {{1, 2, 4}, {3, 5}}
```

Can you find the other 3? One way of finding the permutations that map on to a particular tableau is to find all sequences of deletions from the tableau that lead to the empty tableau. The reverse of each of these sequences is a permutation that maps on to the tableau.

The Robinson-Schensted-Knuth correspondence is a series of results, by Robinson in 1930's, by Schensted in 1960's, and by Knuth in 1970 that showed a bijection between n-permutations and pairs of Young tableaux. The see this bijection write a permutation $p = (p_1, p_2, \ldots, p_n)$ in its two line notation as

$$p = \begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ p_1 & p_2 & p_3 & \ldots & p_n \end{pmatrix}.$$

Start with a pair (P, Q) of empty tableaux and for each $i = 1, 2, \ldots, n$ insert p_i into P and set $Q_{st} \leftarrow i$ where (s, t) is the newly filled position in P. To see this construction in action let us start with the permutation $(3, 1, 5, 2, 4)$ again. After 3 and 1 have been processed we have the pair of tableau

$$\begin{matrix} 1 \\ 3 \end{matrix} \qquad\qquad \begin{matrix} 1 \\ 2 \end{matrix}$$

After the 5 is processed we have

$$\begin{matrix} 1 & 5 \\ 3 & \end{matrix} \qquad\qquad \begin{matrix} 1 & 3 \\ 2 & \end{matrix}$$

and after the 2 is processed we have

$$\begin{matrix} 1 & 2 \\ 3 & 5 \end{matrix} \qquad\qquad \begin{matrix} 1 & 3 \\ 2 & 4 \end{matrix}$$

and after the 4 is processed we have the final pair

$$\begin{matrix} 1 & 2 & 4 \\ 3 & 5 & \end{matrix} \qquad\qquad \begin{matrix} 1 & 3 & 5 \\ 2 & 4 & \end{matrix}$$

So at the end of the process we have a tableau P *and* a tableau Q that informs us of the order in which insertions were made into P. This allows us to reverse the process unambiguously. In the above example, the largest element in Q, namely 5, is in row 1 and so the first deletion from P is in row 1 resulting in the deletion of 4. The resulting pair of tableaux is

$$\begin{matrix} 1 & 2 \\ 3 & 5 \end{matrix} \qquad\qquad \begin{matrix} 1 & 3 \\ 2 & 4 \end{matrix}.$$

The largest element in Q, namely 4, is in row 2 and that tells us that the next deletion from P is in row 2. We can continue in this manner till (P, Q) become the empty pair of tableaux and in the process we would have recovered the permutation that we started with.

Unfortunately, *Combinatorica* does not contain a function that implements the RSK correspondence directly, though it is not hard to implement such a function. First we implement a function called `NewInsertIntoTableau` that inserts an element into a tableau and returns a pair in which the first element is the position of the newly created cell and the second element is the new tableau. This function is a slight modification of the function `InsertIntoTableau` that *Combinatorica* contains.

```
NewInsertIntoTableau[e_Integer, {}] := {{1, 1}, {{e}}}
NewInsertIntoTableau[e_Integer, t1_?TableauQ] :=
  Module[{item = e, row = 0, col, t = t1},
         While[row < Length[t],
               row++;
               If[Last[t[[row]]] <= item,
                  AppendTo[t[[row]], item];
                  Return[{{row, Length[ t[[row]] ]}, t}]
               ];
               col = Ceiling[BinarySearch[t[[row]], item]];
               {item, t[[row, col]]} = {t[[row, col]], item};
         ];
         {{Length[t] + 1, 1}, Append[t, {item}]}]
```

The following example shows NewInsertIntoTableau in action. The insertion of 10 into the given tableau creates a new cell in position $(2,3)$ and this position is also returned by the function.

```
In[3]:= NewInsertIntoTableau[10, {{1, 2, 11}, {7, 8}}]
Out[3]= {{2, 3}, {{1, 2, 10}, {7, 8, 11}}}
```

Using this function it is easy to write a function ConstructTableauxPair that implements the RSK-correspondence.

```
ConstructTableauxPair[p_?PermutationQ] :=
  Module[{t, P = {}, Q = Table[Infinity, {Length[p]}, {Length[p]}]},
         Do[t = NewInsertIntoTableau[p[[i]], P];
            P = t[[2]];
            Q[[ t[[1, 1]], t[[1, 2]] ]] = i,
            {i, Length[p]}
         ];
         {P, Select[Map[Cases[#, _Integer] &, Q], (# != {}) &]}
  ]
```

Here are the 6 tableaux pairs corresponding to the 6 3-permutations.

```
In[6]:= Map[ ConstructTableauxPair, Permutations[3] ] // ColumnForm
Out[6]= {{{1, 2, 3}}, {{1, 2, 3}}}
        {{{1, 2}, {3}}, {{1, 2}, {3}}}
        {{{1, 3}, {2}}, {{1, 3}, {2}}}
        {{{1, 3}, {2}}, {{1, 2}, {3}}}
        {{{1, 2}, {3}}, {{1, 3}, {2}}}
        {{{1}, {2}, {3}}, {{1}, {2}, {3}}}
```

In the above example, for 4 of the 6 permutations the corresponding tableaux pair (P, Q) satisfies $P = Q$. It turns out that the permutations for which this is true are familiar to us. In the following example, those 4-permutations are selected for which the two corresponding tableaux are identical. On examining the cycle structure of these permutations we see that they are all involutions!

```
In[7]:= Select[Permutations[4], (t = ConstructTableauxPair[#]; t[[1]] == t[[2]])&]
Out[7]= {{1, 2, 3, 4}, {1, 2, 4, 3}, {1, 3, 2, 4}, {1, 4, 3, 2},
         {2, 1, 3, 4}, {2, 1, 4, 3}, {3, 2, 1, 4}, {3, 4, 1, 2}, {4, 2, 3, 1},
         {4, 3, 2, 1}}

In[10]:= Map[ToCycles, %] // ColumnForm
Out[10]= {{1}, {2}, {3}, {4}}
         {{1}, {2}, {4, 3}}
         {{1}, {3, 2}, {4}}
         {{1}, {4, 2}, {3}}
         {{2, 1}, {3}, {4}}
         {{2, 1}, {4, 3}}
         {{3, 1}, {2}, {4}}
         {{3, 1}, {4, 2}}
         {{4, 1}, {2}, {3}}
         {{4, 1}, {3, 2}}
```

In fact, we will soon be able to prove that there is a one-to-one correspondence between n-involutions and tableaux with elements $1, 2, \ldots, n$.

The stepping stone to this result is a remarkable property of the Robinson-Schensted-Knuth (RSK) correspondence: a permutation p corresponds to a tableaux pair (P, Q) if and only if the permutation p^{-1} corresponds to the tableaux pair (Q, P). The following example illustrates this property.

```
In[11]:= ConstructTableauxPair[p = RandomPermutation[10] ]
Out[11]= {{{1, 3, 5, 10}, {2, 4, 6}, {7, 9}, {8}},
          {{1, 3, 4, 5}, {2, 7, 10}, {6, 9}, {8}}}

In[12]:= ConstructTableauxPair[ InversePermutation[p] ]
Out[12]= {{{1, 3, 4, 5}, {2, 7, 10}, {6, 9}, {8}},
          {{1, 3, 5, 10}, {2, 4, 6}, {7, 9}, {8}}}
```

We need to make a couple of remarks in preparation for proving this remarkable property.

- The RSK-correspondence can be slightly generalized to *two line arrays*

  $$\begin{pmatrix} q_1 & q_2 & q_3 & \cdots & q_n \\ p_1 & p_2 & p_3 & \cdots & p_n \end{pmatrix}$$

  where $q_1 < q_2 < \cdots < q_n$ and the p_i's are distinct. These correspond to tableaux pairs (P, Q) where P contains the elements $\{p_1, p_2, \ldots, p_n\}$, Q contains the elements $\{q_1, q_2, \ldots, q_n\}$, and P and Q have identical shape.

- The construction of a tableaux pair (P, Q) from a two line array by repeated insertion can be viewed as constructing the first rows of P and Q and then recursively constructing the rest of P and the rest of Q from a smaller two line array. To be specific consider the two line array

  $$\begin{pmatrix} 1 & 3 & 5 & 6 & 8 \\ 7 & 2 & 9 & 5 & 3 \end{pmatrix}$$

  and focus on what happens to the first rows of P and Q as the two line array is processed. The specific actions that take place are:

  - Insert 7, $Q_{11} \leftarrow 1$
  - Insert 2, bump 7
  - Insert 9, $Q_{12} \leftarrow 5$
  - Insert 5, bump 9 and
  - Insert 3, bump 5.

  Thus the first row of P is 23 and the first row of Q is 15. Furthermore the remaining rows of P and Q are the tableaux corresponding to the "bumped" two-line array

  $$\begin{pmatrix} 3 & 6 & 8 \\ 7 & 9 & 5 \end{pmatrix}$$

  This remark is important because in the following inductive proof we will prove that a certain property holds for the first rows of P and Q and then use the inductive hypothesis for the rest of tableaux.

We are now ready to prove that a permutation p corresponds to a tableaux pair (P, Q) if and only if p^{-1} corresponds to (Q, P). We start with a definition. A column (q_i, p_i) is said to be in *class-t* with respect to the two-line array M defined as

$$\begin{pmatrix} q_1 & q_2 & q_3 & \cdots & q_n \\ p_1 & p_2 & p_3 & \cdots & p_n \end{pmatrix}, \qquad q_1 < q_2 < \cdots < q_n \text{ and } p_1, p_2, \ldots, p_n \text{ distinct}$$

if $p_i = P_{1t}$ after the insertion algorithm has been applied successively to $p_1, p_2, \ldots, p_i$ starting with an empty tableau P. Remember that every p_i when first inserted into P is inserted into row 1. So the above definition is saying that if an element p_i is first inserted into column t then (p_i, q_i) is in class-t.

What are the pairs in class-1? If a pair (q_i, p_i) is in class-1, then it must be the case that p_i lived in the top left corner of the tableau when it first arrived. This means that p_i is the smallest among $\{p_1, p_2, \ldots, p_i\}$. In other words p_i is a left-to-right minimum. More precisely, a pair (q_i, p_i) is in class-1 if and only if p_i is a left-to-right minimum in $(p_1, p_2, \ldots, p_n)$. Continuing in this manner, it is easy to see that the pairs in class-2 correspond to the left-to-right minima of the two-line array obtained after the pairs in class-1 have been deleted. `TableauClasses` is a *Combinatorica* function that partitions the elements of permutation p into classes according to their initial columns during Young tableaux construction.

```
In[1]:= TableauClasses[{5, 1, 3, 4, 2}]
Out[1]= {{1, 5}, {2, 3}, {4}}
```

For any fixed t, the elements of class t can be labeled

$$(q_{i_1}, p_{i_1}), (q_{i_2}, p_{i_2}), \ldots, (q_{i_k}, p_{i_k})$$

in such a way that

$$q_{i_1} < q_{i_2} < \cdots < q_{i_k}$$

and

$$p_{i_1} > p_{i_2} > \cdots > p_{i_k}.$$

This is because the tableau position P_{1t} takes on the decreasing sequence of values $p_{i_1}, p_{i_2}, \ldots, p_{i_k}$ as the insertion algorithm proceeds. At the end of the construction we have

$$P_{1t} = p_{i_k} \qquad\qquad Q_{1t} = q_{i_1}.$$

The "inverse" of M, denoted M^{-1}, is the two-line array obtained by exchanging the two rows of M and then sorting according to the first row. Now we make the observation that (q_i, p_i) is in class-t with respect to M if and only if (p_i, q_i) is in class-t with respect to M^{-1}. This observation immediately follows from an alternate characterization of elements in class-t. A pair (q_i, p_i) is in class-t with respect to M if and only if t is the maximum number of indices $i_1, i_2, \ldots, i_t$ such that

$$q_{i_1} < q_{i_2} < \cdots < q_{i_t} = q_i$$

and

$$p_{i_1} < p_{i_2} < \cdots < p_{i_t} = p_i.$$

Because of the symmetry of the inequalities it follows that (q_i, p_i) is in class-t with respect to M if and only if (p_i, q_i) is in class-t with respect to M^{-1}. Below we provide an example to illustrate this alternative characterization of class-t, but leave the proof as an exercise. The permutation in the example below has 4 classes. The pair $(14, 15)$ is in class-4 because

$$(1, 8) < (3, 12) < (5, 14) < (14, 15).$$

Here the $<$ sign above stands for both coordinates being smaller. Similarly, this characterization can be verified for other pairs.

```
In[1]:= p = RandomPermutation[15]
Out[1]= {8, 4, 12, 10, 14, 13, 7, 6, 2, 1, 5, 3, 9, 15, 11}

In[6]:= TableauClasses[p] /. i_Integer :> {Position[p, i][[1,1]], i} // ColumnForm
Out[6]= {{10, 1}, {9, 2}, {2, 4}, {1, 8}}
        {{12, 3}, {11, 5}, {8, 6}, {7, 7}, {4, 10}, {3, 12}}
        {{13, 9}, {6, 13}, {5, 14}}
        {{15, 11}, {14, 15}}
```

The fact that (q_i, p_i) is in class-t with respect to M if and only if (p_i, q_i) is in class-t with respect to M^{-1} is very useful for our next step. Let

$$\{(q_{i_1}, p_{i_1}), (q_{i_2}, p_{i_2}), \ldots, (q_{i_k}, p_{i_k})\}$$

be the elements in class-t with respect to M such that

$$q_{i_1} < q_{i_2} < \cdots < q_{i_k}$$

and

$$p_{i_1} > p_{i_2} > \cdots > p_{i_k}.$$

Then class-t with respect to M^{-1} is

$$\{(p_{i_1}, q_{i_1}), (p_{i_2}, q_{i_2}), \ldots, (p_{i_k}, q_{i_k})\}$$

such that

$$p_{i_k} < \cdots < p_{i_2} < p_{i_1}$$

and

$$q_{i_k} > \cdots > q_{i_2} > q_{i_1}.$$

Now suppose that the RSK-construction is carried out on M^{-1} and the resulting tableaux pair is (P^{-1}, Q^{-1}). After M^{-1} is processed $P^{-1}_{1t} = q_{i_1}$ and $Q^{-1}_{1t} = p_{i_k}$. Now that $P_{1t} = Q^{-1}_{1t}$ and $Q_{1t} = P^{-1}_{1t}$ and this means that the first rows of P and and Q^{-1} are identical and the first rows of Q and P^{-1} are identical. Now suppose that the remaining rows of P and Q are constructed from the "bumped" two-line array R, obtained from M by deleting q_i's and p_i's that are in the first rows. Similarly denote by R^{-1} the "bumped" two-line array obtained from M^{-1}. It is easy to see that R^{-1} is the "inverse" of R and by using induction we see that the rest of the rows of P are identical to the rest of the rows of Q^{-1} and the rest of the rows of Q are identical to the rest of the rows of P^{-1}. Thus we have shown that a permutation p corresponds to a tableaux pair (P, Q) if and only if p^{-1} corresponds to (Q, P).

From this we immediately get a bijection between n-involutions and tableaux with elements $1, 2, \ldots, n$ because for any involution p, $p = p^{-1}$ and for the corresponding tableaux $(P, Q) = (Q, P)$ implying that $P = Q$. So every involution corresponds to a single tableaux P. In the opposite direction, given a tableau P we can start with the pair (P, P) and obtain the involution corresponding to (P, P).