Finally, we present a successor algorithm for the revolving door ordering, in Algorithm 2.13. In this algorithm, the successor of the last k-subset is the first one. In other words, we think of the list $A^{n,k}$ as being ordered cyclicly, and therefore we define

$$\text{successor}(\{1,\ldots,k-1,n\}) = \{1,\ldots,k\}.$$

Note that this is also a minimal change.

Algorithm 2.13 begins by defining t_{k+1} to be $n+1$. This means that we do not have to handle the situation $j = k$ as a special case.

Algorithm 2.13: KSUBSETREVDOORSUCCESSOR ($\overrightarrow{T}, k, n$)

$t_{k+1} \leftarrow n+1$
$j \leftarrow 1$
while $(j \leq k)$ **and** $(t_j = j)$
  do $j \leftarrow j+1$
if $k \not\equiv j \bmod 2$

$$\textbf{then} \begin{cases} \textbf{if } j = 1 \\ \quad \textbf{then } t_1 \leftarrow t_1 - 1 \\ \quad \textbf{else } \begin{cases} t_{j-1} \leftarrow j \\ t_{j-2} \leftarrow j-1 \end{cases} \end{cases}$$

$$\textbf{else} \begin{cases} \textbf{if } t_{j+1} \neq t_j + 1 \\ \quad \textbf{then } \begin{cases} t_{j-1} \leftarrow t_j \\ t_j \leftarrow t_j + 1 \end{cases} \\ \quad \textbf{else } \begin{cases} t_{j+1} \leftarrow t_j \\ t_j \leftarrow j \end{cases} \end{cases}$$

return ($\overrightarrow{T}$)

2.4 Permutations

2.4.1 Lexicographic ordering

We now look at the generation of all $n!$ permutations of the set $\{1,\ldots,n\}$. A *permutation* is a bijection from a set to itself. One way to represent a permutation $\pi : \{1,\ldots,n\} \rightarrow \{1,\ldots,n\}$ is by listing its values, as follows:

$$[\pi[1],\ldots,\pi[n]].$$

We call this the *list representation* of the permutation π. Saying that π is a permutation is equivalent to saying that each element in $\{1,\ldots,n\}$ occurs exactly once in this list.

First, we will look at the lexicographic ordering of permutations. The lexicographic ordering is defined in terms of the list representation. As an example, when $n = 3$, the lexicographic ordering of the six permutations of $\{1, 2, 3\}$ is as follows:

$$[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1].$$

We begin by describing an algorithm for generating permutations in lexicographic order. This generation algorithm depends on a successor algorithm that finds the permutation that immediately follows a given permutation (in lexicographic order). In Algorithm 2.14, π is a permutation of $\{1, \ldots, n\}$ given in list representation.

Algorithm 2.14 has four steps. In the first **while** loop, we find i such that

$$\pi[i] < \pi[i+1] > \pi[i+2] > \cdots > \pi[n].$$

Note that by setting $\pi[0]$ to 0, we ensure that the **while** loop terminates with $0 \leq i \leq n-1$. If $i = 0$, then

$$\pi = [n, n-1, \ldots, 1]$$

is the last permutation lexicographically and has no successor. Otherwise, we proceed to the second **while** loop, where we find the integer j such that $\pi[j] > \pi[i]$ and $\pi[k] < \pi[i]$ for $j < k \leq n$ (i.e., j is the position of the last element among $\pi[i+1], \ldots, \pi[n]$ that is greater than $\pi[i]$). The third step is to interchange $\pi[i]$ and $\pi[j]$, and the fourth step is to reverse the sublist

$$[\pi[i+1], \ldots, \pi[n]].$$

Algorithm 2.14: PERMLEXSUCCESSOR (n, π)

$\pi[0] \leftarrow 0$
$i \leftarrow n - 1$
while $\pi[i+1] < \pi[i]$
  do $i \leftarrow i - 1$
if $i = 0$
  then return ("undefined")
$j \leftarrow n$
while $\pi[j] < \pi[i]$
  do $j \leftarrow j - 1$
$t \leftarrow \pi[j]$
$\pi[j] \leftarrow \pi[i]$
$\pi[i] \leftarrow t$
for $h \leftarrow i + 1$ **to** n
  do $\rho[h] \leftarrow \pi[h]$
for $h \leftarrow i + 1$ **to** n
  do $\pi[h] \leftarrow \rho[n + i + 1 - h]$
return (π)

As an example, suppose that $n = 7$ and

$$\pi = [3, 6, 2, 7, 5, 4, 1].$$

Then, after the first **while** loop, we have $i = 3$, since

$$2 < 7 > 5 > 4 > 1.$$

After the second **while** loop, we have $j = 6$ since $4 > 2$ and $1 < 2$. In the third step, we interchange π_3 and π_6, producing

$$[3, 6, 4, 7, 5, 2, 1].$$

Finally, we reverse the sublist

$$[7, 5, 2, 1],$$

producing the permutation

$$[3, 6, 4, 1, 2, 5, 7],$$

which is the successor of π.

It is now easy to generate all $n!$ permutations of $\{1, \ldots, n\}$. We can begin with the permutation $[1, 2, \ldots, n]$ (which is the first permutation lexicographically) and invoke Algorithm 2.14 a total of $n! - 1$ times.

We next turn to ranking and unranking permutations in lexicographic order. In the lexicographic ordering of permutations of $\{1, \ldots, n\}$, we first have the $(n-1)!$ permutations that begin with a "1", followed by the $(n-1)!$ permutations that begin with a "2", etc. Hence, if π is a permutation of $\{1, \ldots, n\}$, it is clear that

$$(\pi[1] - 1)\,(n-1)! \leq \mathsf{rank}(\pi) \leq \pi[1]\,(n-1)! - 1.$$

Let r' denote the rank of π within the group of $(n-1)!$ permutations that begin with $\pi[i]$. Then r' is the rank of $[\pi[2], \ldots, \pi[n]]$ when it is considered as a permutation of $\{1, \ldots, n\} \backslash \{\pi[1]\}$. If we decrease every element of $[\pi[2], \ldots, \pi[n]]$ that is greater than $\pi[1]$ by one, then we obtain a permutation π' of $\{1, \ldots, n-1\}$ that also has rank r'.

This observation leads to a recursive formula for lexicographic rank of permutations of $\{1, \ldots, n\}$. For $n > 1$, we have

$$\mathsf{rank}(\pi, n) = (\pi[1] - 1)\,(n-1)! + \mathsf{rank}(\pi', n-1),$$

where

$$\pi'[i] = \begin{cases} \pi[i+1] - 1 & \text{if } \pi[i+1] > \pi[1] \\ \pi[i+1] & \text{if } \pi[i+1] < \pi[1]. \end{cases}$$

Initial conditions for this recurrence relation are given by

$$\mathsf{rank}([1], 1) = 0.$$

We work out a small example to illustrate:

$$\begin{aligned}
\mathsf{rank}([2,4,1,3],4) &= 6 + \mathsf{rank}([3,1,2],3) \\
&= 6 + 4 + \mathsf{rank}([1,2],2) \\
&= 6 + 4 + 0 + \mathsf{rank}([1],1) \\
&= 6 + 4 + 0 + 0 \\
&= 10.
\end{aligned}$$

It is easy to convert this recursive formula into a non-recursive algorithm, which we present as Algorithm 2.15.

Algorithm 2.15: PERMLEXRANK (n, π)

```
r ← 0
ρ ← π
for j ← 1 to n
    do { r ← r + (ρ[j] − 1)(n − j)!
         for i ← j + 1 to n
             do { if ρ[i] > ρ[j]
                    then ρ[i] ← ρ[i] − 1
return (r)
```

Now suppose we want to unrank the integer r, where $0 \leq r \leq n! - 1$. Unranking can be done fairly easily if we first determine the *factorial representation* of r, by expressing r in the form

$$r = \sum_{i=1}^{n-1} (d_i \cdot i!),$$

where $0 \leq d_i \leq i$ for $i = 1, \ldots, n-1$. (We leave it as an exercise to prove that any non-negative integer r such that $0 \leq r \leq n! - 1$ has a unique factorial representation of this form.)

Suppose that $\pi = \mathsf{unrank}(r)$ in the lexicographic ordering. It is easy to see that

$$\pi[1] = d_{n-1} + 1.$$

Thus the first element of π is determined immediately from the factorial representation of r. Now, denote

$$r' = r - d_{n-1} \cdot (n-1)!,$$

and suppose that $\pi' = \mathsf{unrank}(r')$, where π' is a permutation of $\{1, \ldots, n-1\}$. (This could be done recursively, for example.) Suppose we increment by one all elements of π' that are greater than d_{n-1}. Finally, define

$$\pi[i] = \pi'[i+1]$$

for $2 \le i \le n$. Then it will be the case that $\pi = \mathsf{unrank}(r)$.

As an example, suppose that $n = 4$ and $r = 10$. The factorial representation of r is

$$1 \cdot 3! + 2 \cdot 2! + 0 \cdot 1!.$$

Hence, $\pi[1] = d_3 + 1 = 2$. Now, compute $r' = r - 6 = 4$. It can be verified that $\pi' = \mathsf{unrank}(4) = [3, 1, 2]$. Then we increment the first and third elements by one, so $\pi' = [4, 1, 3]$. Hence, we obtain

$$\mathsf{unrank}(10) = [2, 4, 1, 3].$$

Algorithm 2.16 is a non-recursive implementation of this unranking algorithm. In this algorithm, we use a function mod which performs modular reduction according to the following rule:

$$\mathsf{mod}(x, m) = r \Leftrightarrow x \equiv r \bmod m \text{ and } 0 \le r \le m - 1.$$

Algorithm 2.16: PERMLEXUNRANK (n, r)

$$
\begin{array}{l}
\pi[n] \leftarrow 1 \\
\textbf{for } j \leftarrow 1 \textbf{ to } n - 1 \\
\quad \textbf{do} \begin{cases}
d \leftarrow \frac{\mathsf{mod}(r,(j+1)!)}{j!} \\
r \leftarrow r - d \cdot j! \\
\pi[n - j] \leftarrow d + 1 \\
\textbf{for } i \leftarrow n - j + 1 \textbf{ to } n \\
\quad \textbf{do} \begin{cases} \textbf{if } \pi[i] > d \\ \quad \textbf{then } \pi[i] \leftarrow \pi[i] + 1 \end{cases}
\end{cases} \\
\textbf{return } (\pi)
\end{array}
$$

We illustrate Algorithm 2.16 by recomputing $\mathsf{unrank}(10)$. Initially, we set

$$\pi[4] = 1.$$

When $j = 1$, we compute

$$d = \frac{\mathsf{mod}(10, 2)}{1} = 0,$$

$$\pi[3] = 1 \text{ and } \pi[4] = 2.$$

When $j = 2$, we have

$$d = \frac{\mathsf{mod}(10, 6)}{2} = 2,$$

$$r = 10 - 2 \cdot 2 = 6,$$

and

$$\pi[2] = 3.$$

Finally, when $j = 3$, we have

$$d = \frac{\mathsf{mod}(6, 24)}{6} = 1,$$

$$r = 6 - 1 \cdot 6 = 0,$$

$$\pi[1] = 2, \pi[2] = 4 \text{ and } \pi[4] = 3.$$

Hence, we obtain

$$\mathsf{unrank}(10) = [2, 4, 1, 3],$$

as before.

2.4.2 Minimal change ordering

First we need to give some thought as to what a minimal change would be in the context of permutations. It is certainly the case that any two distinct permutations π and π' of $\{1, \ldots, n\}$ must differ in at least two positions. Further, if π and π' differ in exactly two positions, then one can be obtained from the other by a single *transposition* (i.e., by exchanging the elements in the two given positions). It may even happen that the two positions are adjacent; so, we in fact transpose two adjacent elements in order to transform π into π'. This is equivalent to saying that there exists an integer i, $1 \leq i \leq n - 1$, such that

$$\pi'[j] = \begin{cases} \pi[j+1] & \text{if } j = i \\ \pi[j-1] & \text{if } j = i + 1 \\ \pi[j] & \text{if } j \neq i, i + 1. \end{cases}$$

This is in fact the definition we will take for a minimal change for permutations.

The *Trotter-Johnson algorithm* is a nice example of a minimal change algorithm for generating the $n!$ permutations. It can be most easily described recursively. Suppose we have a listing of the $(n - 1)!$ permutations of $\{1, \ldots, n - 1\}$ in minimal change order, say

$$\mathsf{T}^{n-1} = [\pi_0, \pi_1, \ldots, \pi_{(n-1)!-1}].$$

Form a new list by repeating each permutation in the list T^{n-1} n times. Now insert the element n into each of the n copies of each permutation π_i, as follows. If i is even, then we first insert element n after the element in position $n - 1$, then after the element in position $n - 2$, etc., and finally preceding the element in position 1. If i is odd, then we proceed in the opposite order, inserting element n into the n copies of π_i from the beginning to the end of π.

We illustrate the procedure for $n = 1, 2, 3$ and 4. We begin with $n = 1$, where we have

$$\mathsf{T}^1 = [1].$$

Next, we obtain

$$\mathsf{T}^2 = [[1, 2], [2, 1]].$$