for $2 \leq i \leq n$. 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:

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

Algorithm 2.16: PERMLEXUNRANK (n, r)

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

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

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