SOURCE CODE 1. Rang (also Nummer) einer Permutation der $\mathfrak{S}_n$ in der Johnson–Trotter–Reihenfolge.

```
def rank_johnson_trotter(pi,n):
    """Berechne die Nummer einer Permutationen von {1,2,...,n}
    in der Auflistung mit dem Johnson-Trotter-Algorithmus."""
    # Startwert für den Rückgabewert "rank"
    rank = 0
    # Länge des Permutationswortes sollte natürlich n sein
    if n != len(pi):
        print('Oops!')
    # Startwert für Rückgabewert "Liste b"
    # list_b = [0]*n
    # Inverse Permutation:
    invpi = invert_pi(pi)

    for i in range(1,n+1):
        # Wenn wir im Permutationswort pi alle Elemente > i wegstreichen,
        # an welcher Stelle steht dann i - Das ist äquivalent mit:
        # Wieviele Elemente j KLEINER i stehen VOR i im Permutationswort pi
        # Wir beantworten diese Frage mithilfe der INVERSEN Permutation:
        # Für wieviele Elemente j KLEINER i gilt pi^{-1}(j) < pi^{-1}(i)
        moves = count_less_than_last(invpi[:i])
        # Damit bestimmen wir nun "rekursiv" die gewünschte Nummer:
        if rank % 2 == 1:
            remainder = moves
        else:
            remainder = i-1-moves
        # list_b[i-1] = remainder
        rank = i*rank + remainder
    return rank # , list_b
```