list $= \emptyset$

while π ist nicht identische Permutation do

  if π hat absteigenden Strip then

    $k :=$ kleinste Element in absteigendem Strip von π

    Bestimme Pos. i von k und i' von $k-1$ in π

    $\rho := (i'+1, i)$-Reversal

    if $\pi\rho$ hat keinen absteigenden Strip then

      $l :=$ größte Element in absteigendem Strip in π

      Bestimme Pos. j von l und j' von $l+1$ in π

      $\rho := (j, j'-1)$-Reversal

  else {π hat keinen abst. Strip}

  (*) $\rho :=$ Reversal, das an den ersten zwei BPs von $ext(\pi)$ „schneidet"

  $\pi := \pi\rho$

  list $:=$ list $\cup\, \rho$.

<u>Satz:</u> Vorheriger Algorithmus berechnet 2-Approxim. für das MinSR-Problem.

Beweis: Wir zeigen, dass der Alg. im Schnitt pro Reversal 1 BP beseitigt.

Hat π absteigenden Strip $\rightarrow$ ausgeführter Reversal eliminiert 1 BP. Hat π danach keinen abst. Strip mehr $\rightarrow$ es wurden 2 BP durch ein Reversal beseitigt.
Einzige Ausnahme ist Reversal nach (*); dieser Fall kann aber nur auftr.:

- zu Beginn der Berechnung,
- nach der Beseitigung von 2 BP durch ein Reversal

$\square$

Da jede Permutation höchstens linear in n viele Breakpoints hat, braucht der Alg. Laufzeit in $O(n^2)$.

Sortieren gerichteter Permutationen

Def.: Gerichtete Permutation

$$\pi = (\pi_1, \ldots, \pi_n)$$

$$\pi_i \in \{\vec{1}, \vec{2}, \dots, \vec{n}, \overleftarrow{1}, \overleftarrow{2}, \dots, \overleftarrow{n}\}$$

wobei kein $\{\pi_i, \pi_j\} = \{\vec{k}, \overleftarrow{k}\}$

<u>Def:</u> (i,j)-Reversal ρ

$$\pi\rho(i,j) = (\pi_1, \pi_2, \dots, \pi_{i-1}, \overline{\pi_j}, \overline{\pi_{j-1}}, \dots, \overline{\pi_i}, \pi_{j+1}, \dots, \pi_n)$$

mit $\overline{\vec{k}} = \overleftarrow{k}$ und

$\overline{\overleftarrow{k}} = \vec{k}$.

<u>Satz:</u> Das zugehörige Optimierungsproblem „Sortieren mit minimaler Anzahl Reversals" ist in Poly-Zeit lösbar.

Alg. geht auf Hannenhalli und Pevzner 1995 zurück.