31-1-3 Řazení kořenek

Tato úloha se zabývá netradičním řadícím algoritmem, kde se soustředí ne na rychlost, ale na počet posunutí v řetězci. Kvůli tomuto omezení, nemůžeme použít obvyklé postupy jako Quicksort nebo Mergesort.

Při řazení takovým to způsobem, musíme maximalizovat efektivitu jednoho posunutí. Pokud chceme posouvat co nejmenší počet *lahviček*, neměli bychom posouvat ty, co už jsou na správném místě. Z celého řetězce tak musíme vybrat nejdelší vzestupnou řadu čísel (každá *lahvička* má přiřazené svoje místo, vyjádřeno číslem; dvě lahvičky nemůžou být na stejném místě). Nejjednodušší způsob je použít LIS (Longest increasing subsequence), buď pouze pomocí Dynamic Programming (*O(n2)),* nebo přidat Binary search a snížit tak *O* na *n log n*.

Výsledný počet kroků potom bude *délka původního řetězce – LIS řetězec.* Nyní již stačí pouze porovnat původní a LIS řetězec a přesunout neseřazený prvek tak, aby byl na správném místě vůči seřazenému řetězci a označíme ho za seřazený. Takto postupujeme, dokud řetězec nebude seřazený.

Příklad takového řešení:

* Je dán řetězec: [9, 8, 1, 4, 2, 6, 3, 7, 5, 0] (10 prvků)
* Nejdelší seřazený řetězec: [1, 2, 3, 5] (4 prvky)
  + Potřebný počet kroků k seřazení řetězce: 10–4 = 6

1. Prvek 9 posuneme za prvek 5
2. Prvek 8 posuneme za prvek 5
3. Prvek 4 posuneme za prvek 3
4. Prvek 6 posuneme za prvek 5
5. Prvek 7 posuneme za prvek 6
6. Prvek 0 posuneme na začátek

* Výsledný řetězec: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]