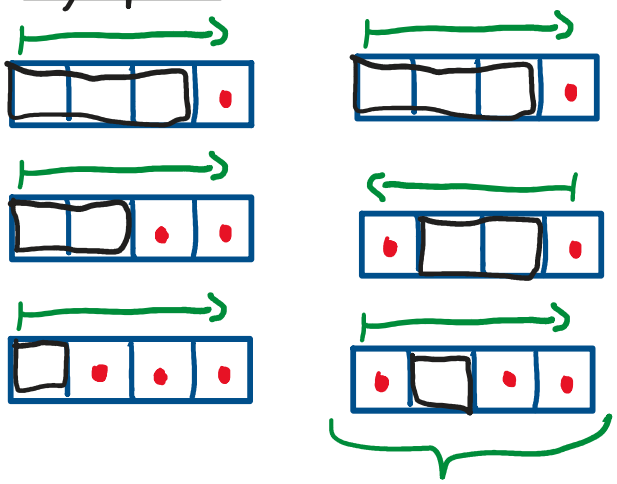
9.hodina 6.11. 2023

pondělí 6. listopadu 2023 11:58

- omezení – pomalý posun hodnot proti směru průchodu (max 1 pozice s 1 průchodem)
  => vylepšení – změna směru průchodu

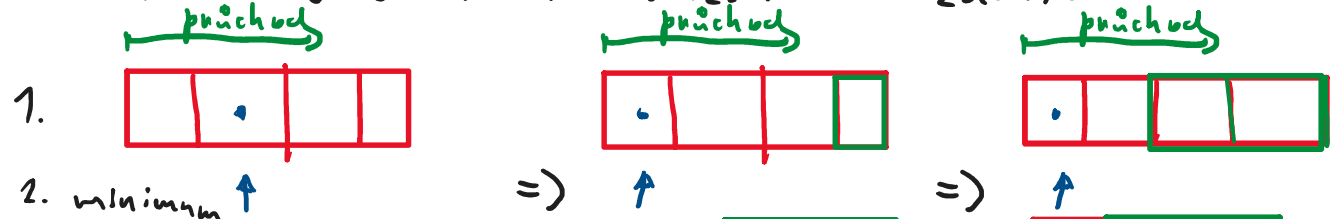


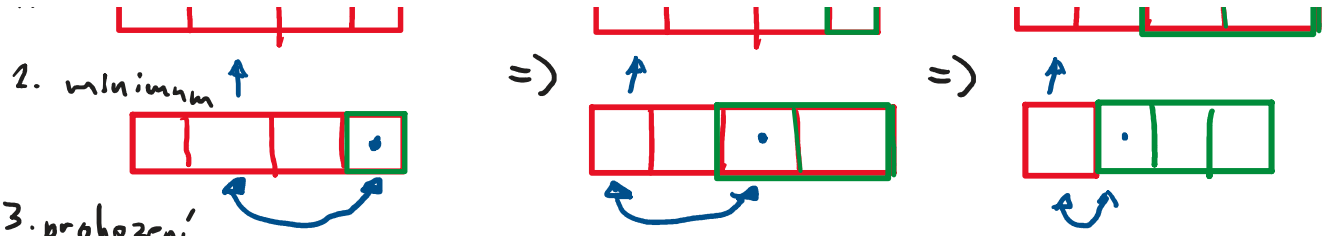
Shake sort

- hodnoty na konci průchodu jsou již seřazené
  => nemusíte porovnávat
  - neseřazená část se zmenšuje
- nedojde-li k prohození v průchodu
  => hodnoty jsou seřazené → $\Omega(n)$

Selection Sort (řazení výběrem)

- sekvence hodnot rozdělena na 2 části:
  1. neseřazená (Na začátku všechny hodnoty)
  2. seřazená (Na začátku prázdná)
- při průchodu neseřazené části nalezeno minimum (maximum)
  => výběr a potom hodnota prohozena s poslední hodnotou neseřazené části => zařazení na začátek





- počet prohození $O(n)$
- délka průchodu (průměr) $\frac{n}{2}$

- nevýhoda - nestabilní (viz stabilní třídění 5. hodina)
- výhoda - menší počet prohození

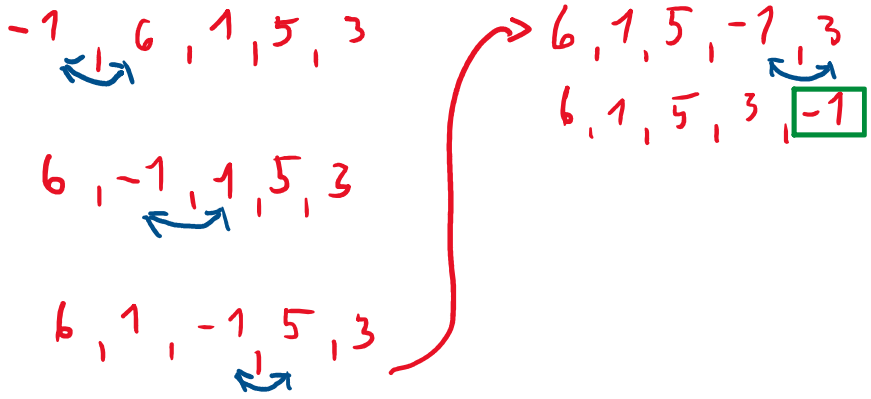
Insertion sort (třídění vkládáním)

10.hodina 13.11. 2023

pondělí 13. listopadu 2023 11:11

- průchod – první prvek z neseřazené části probublávání seřazenou částí až na správnou pozici
- setřídění – opakovat n-1 krát průchod
- porovnání ~ rychlejší než bubble a selection
  - stále v $O(n^2)$
  - rychlejší než quick sort (pro $n \leq 16$)
- příklad: insertion (sestupně), nejmenší na konec

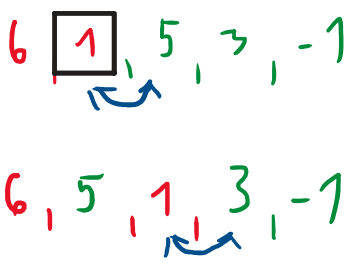
1. průchod →



2. průchod

3. průchod

4. průchod



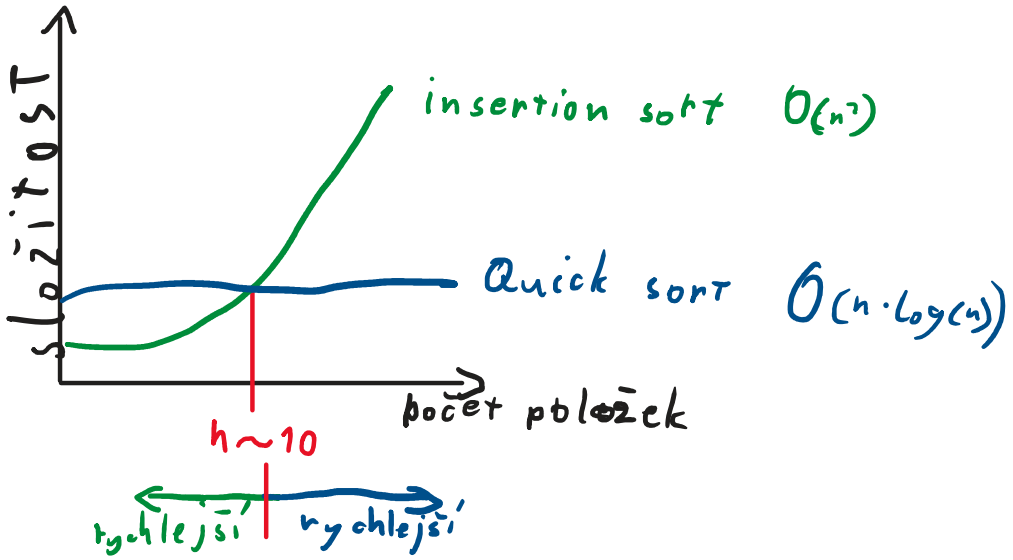
6, 5, 3, 1, -1

5. průchod

6, 5, 3, 1, -1

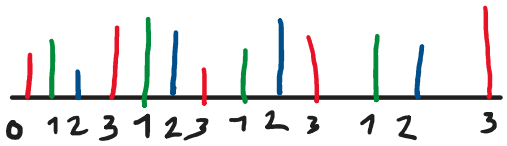
11.hodina 20.11. 2023

pondělí 20. listopadu 2023 10:53



Shell sort ("skořápkové" řazení vkládáním)

- princip: - funguje jako insertion sort (průchod)
  - porovnávané položky jsou na začátku s určitým krokem
  - krok => $\frac{n}{2}$ > krok > 1 => rozestup



- rozestup se každým průchodem zmenšuje
- při velkém kroku položky přeskočí velkou vzdálenost

- výhoda - nejrychlejší z kvadratických ř. algoritmů
  $\Theta(n^{\frac{3}{2}}) = \Theta(n^{1,5})$
  - paměťová složitost $O(1)$
- nevýhoda - nestabilní

Rychlé řadící algoritmy

counting sort (řazení počítáním)

- Lze implementovat jako stabilní
- vhodný pro menší počet hodnot - M
  => paměťová náročnost $O(M)$ - pomocné pole pro četnost
- příklad:
  1) neseřazené hodnoty
     1, 2, 5, 4, 2, 1, 3, 3, 4, 5, 1, 2, 3
  2) zjistit četnosti hodnot - 1 průchod pole (scan) => $O(n)$
     => uložit četnosti do pom. pole

hodnota	1	2	3	4	5
četnost	3	3	3	2	2

=> pro každou položku musíme vyhledat četnost

3) vygenerování seřazené hodnoty z tabulky
   => $O(n)$ až $O(n \cdot \log n)$

111 222 333 44 55
3 3 3 2 2

- výhoda: - řazené položky nemusí být všechny uložené v paměti
  => lze je načítat (streamy)
  - stačí uchovat četnosti položek
    => po načtení (scanu) generujeme ústním

- stačí uchovat četnosti položek
  => po načtení (scanu) generujeme výstup
- nevýhoda: - paměťová náročnost (max $O(n)$)
- použití: - musí být stabilní pro použití u RADIX SORTU