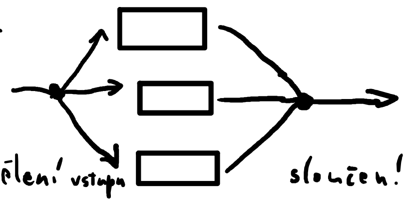
5.hodina 9.10. 2023

pondělí 9. října 2023 11:59

- paralelizace:
    - sériový – pro zpracování úlohy musí být dostupný výsledek z předchozí úlohy => 
    - paralelní – části úlohy lze zpracovat souběžně a neblokují se



- příklad – sample sort – rozdělení do vzorků (sample) setřídění vzorků paralelně (souběžně) → vzorků (merge)
- výpočetní složitost – nejhorší (O) – nejpomalejší
    - průměrná (θ)
    - nejlepší (Ω) – nejrychlejší
- složitost prohození položek => pro in-place algoritmy
    - umístění v paměti – blízko u sebe (lokalita)
        - anebo náhodně (vzdáleně)
        - => záleží na typu paměti (RAM, Sekvenční)
    - velikost položky – malé ~B (rychlé)
        - větší ~kB, MB => pomalejší rychlost MB/s
        - ↳ Lze pracovat s adresou-ukazatele (4B, 8B) (nekopírují se přímo položky ale jen adresy)
- stabilita – při opakovaném použití udrží předchozí relativní řazení
    - příklad: radixové řazení – podle řádu

máme čísla 0 až 999 => 000 až 999

↳ 3 řády

nejdříve řadíme podle jednotek, potom podle desítek...

↳ řazení podle jednotek zůstane zachováno – stabilita

39	039	333	019	019
137	137	023	023	023
125	125	125	125	039
333	333	137	333	125
23	023	039	137	137
19	019	019	039	333

=> řazení podle více klíčů - nutnost (např. příjmení, jméno)

- použitá metoda - jedna nebo kombinace více metod

6.hodina 16.10. 2023

pondělí 16. října 2023 12:13

0	1	2	3
1	2	3	4

- vkládání (10, 0) => [10 | 2 | 3 | 4] O(n)
- záměna (swap) (0, 3) => [4 | 2 | 3 | 1] O(1)
- výběr : minimum.pole = 1 – O(n)

  maximum.pole = 4

- slučování (merge) pole 1 [1 | 2 | 3] pole 2 [4 | 5 | 6]
  - sloučit (pole 1, pole 2) => [1 | 2 | 3 | 4 | 5 | 6] O(n)
- dělení (dividing) pole 1 [1 | 2 | 3 | 4 | 5]

  část (pole 1, 2, 4) => [2 | 3 | 4] O(n) anebo O(1)

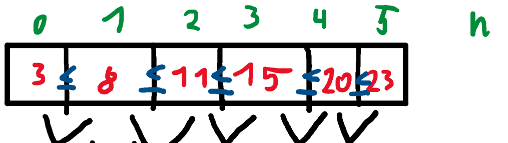
- počítání (counting) O(n)

7.hodina 23.10. 2023

pondělí 6. listopadu 2023 15:54

Zjištění seřazenosti pole nebo seznamu

- vzestupné a sestupné řazení => Ano / ne
- podmínka – vzestupné – Dvě po sobě následující hodnoty (jakékoliv)
  - => aktuální hodnota je $</=$ následující
  - pro i od 0 do n-1
  - sestupné => větší nebo rovno následující $i >/= i+1$
- lze paralelizovat – souběžný běh
  - $i \in (0,4)$

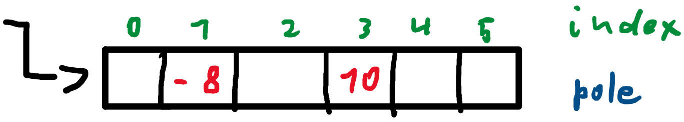


(n-1) dvojic => $O(n)$

$\Omega(1)$

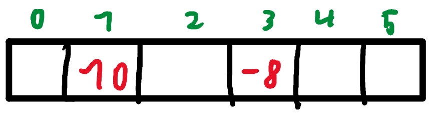
Prohození dvou hodnot na místech v paměti (swap)

- počáteční stav



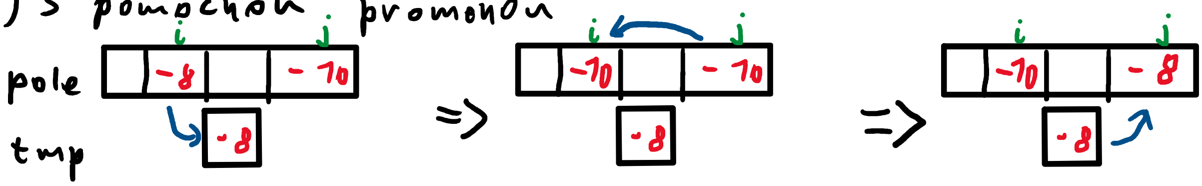
prohodit (pole[1], pole[3])

- koncový stav



- procedura prohodit musí pracovat s místy kde jsou hodnoty uloženy (reference) => prohazujeme hodnoty na místech

- 3 různé způsoby – Techniky
  1) s pomocnou proměnnou



tmp = pole[i] pole[i] = pole[j] pole[j] = tmp

  2) aritmetický výpočet
     - [illegible] pro čísla (celá)

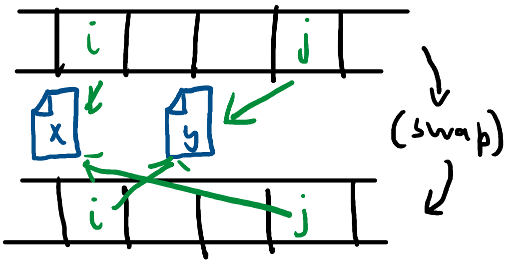
2) aritmetický výpočet

- nevýhoda pro čísla (celá)
- výhoda bez pomocní proměnné

- využívá případné přetečení (overflow), podtečení datového typu

3) bitový výpočet XOR

- pokud jsou položky datově velké (~>1MB) lze zaměňovat adresy položek místo fyzického přenosu dat – práce s ukazately

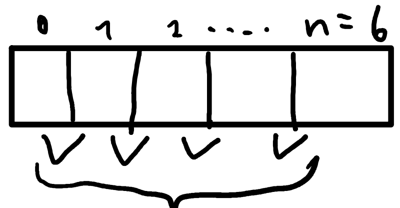


- velikost ukazatele 4B nebo 8B (64bit)

Pomalé řadící algoritmy – $O(n^2)$

Buble sort (řazení probublávání)

- vždy se porovnávají sousední hodnoty v poli, např. pole[j] a pole[j+1]
  - podle podmínky a požadavku na řazení (výsledku) se hodnoty prohodí nebo neprohodí (dvojice se přeskočí)
- algoritmus pracuje v průchodech – postupně projde všechny dvojice



n-1 dvojic => $O(n)$

=> průchod opakujeme n-krát – hodnoty postupně probublávají na své správné místo podle velikosti a řazení

=> (n-krát průchod) · $O(n)$ => $O(n^2)$

- algoritmus musí provést alespoň 1 průchod – zjistí zda jsou nebo nejsou hodnoty seřazeny => $\Omega(n)$

- příklad:

1	8	-5	2	7	14

=> sestupně $\boxed{\text{prohození}}$

průchody/dx	1	2	3	4	5
1	$\boxed{8, 1}$, -5, 2, 7, 14	8, 1, -5, 2, 7, 14	8, 1, $\boxed{2, -5}$, 7, 14	8, 1, 2, $\boxed{7, -5}$, 14	
2					
3					
4					
5					