Třídění a vyhledávání

Přednáška 7

rev 2019.2

Motivace

- velké soubory dat
  - ➢ častý požadavek je získat určitá data
  - ➢ jak je obtížné vyhledat určitá data
    - v nesetříděném souboru dat?
    - v setříděném souboru dat?
  - ➢ dokážete stanovit obtížnost přístupu?
- pro vyhledávání je lepší mít data seřazená
- různé metody řazení
- rozdílný přístup pro
  - řazení neseřazeného pole
  - a pro přidávání dalších prvků do seřazených dat

Motivace

- kolik operací je potřeba pro seřazení čtyř prvků?
  - počítáme nejhorší variantu
  - 6 porovnání (N.(N-1)/2)
  - 0-6 přesunů (výměn) hodnot
- průměrný čas přístupu je cca N/2.
- u setříděného pole je to půlení intervalu
  - tj maximálně log2 N
  - pro 4 prvky = max 2 pokusy, ale pro polovinu prvků to bude rychlejší
- N mohou být ve skutečnosti tisíce, miliony položek

Motivace

- pro vložení do seřazeného pole
  - bude záležet na směru řazení
  - v našem případě až 4 porovnání
  - k tomu až N výměn

  - čím méně porovnání (prvek je více vlevo)
  - tím více přesunů

  ➢počet operací je „konstantní“

Složitost algoritmů

- slouží například k
  - hodnocení
  - porovnání vhodnosti algoritmů
- rozlišujeme
  - časovou
  - paměťovou
- někdy lze určit přesně
  - přístup k x-tému prvku v poli nebo lineárním seznamu
- někdy pouze statisticky
  - střední doba
  - řazení dat může trvat pro různé datové sady různě
  - Často se však uvádí nejhorší možný výsledek složitosti

Složitost algoritmů

- kolik kroků je potřeba k získání x-tého prvku v poli?
  - 1 kroků
- kolik kroků je potřeba pro získání x-tého prvku v lineárním seznamu?
  - N kroků
- dosažení cíle pro půlení intervalu
  - log2 N
  - log2 8 = 3

Časová složitost

- Big O notation
  - O – Order, řád funkce
- může se udávat v počtu kroků
  - záleží na počtu dat (základních + pracovních)
- může se lišit pro různé operace
  - seřazení
  - vyhledání
  - zjištění hodnoty
- pro složitý vzorec
  - bere se nejvýrazněji rostoucí část součtu
  - například:
    - pro 10N^2+3N+50
    - můžeme uvést O(N^2)

Časová složitost

- O(1)
  - konstantní složitost
  - vyhledání prvku v poli pomocí indexu
- O(N)
  - lineární složitost
  - vyhledání prvku v lineárním seznamu
  - suma N čísel v poli
- O(N^2)
  - kvadratická
  - vyhledání maxima ve čtvercové matici
- O(log2 N)
  - logaritmická
  - vyhledání pozice v seřazeném poli půlením intervalu

Vyhledávání

- náročnost
  - podle polohy hledané položky
- uhodněte číslo na které myslím

- normální vyhledávání
  - až N
- binární vyhledávání
  - využívá půlení intervalu
  - složitost log2 N

Vyhledávání

- realizace
  - polem
  - lineárním seznamem
  - binárním stromem
- výhody realizací
  - pro vyhledávání
  - pro vkládání

Vyhledávání v poli

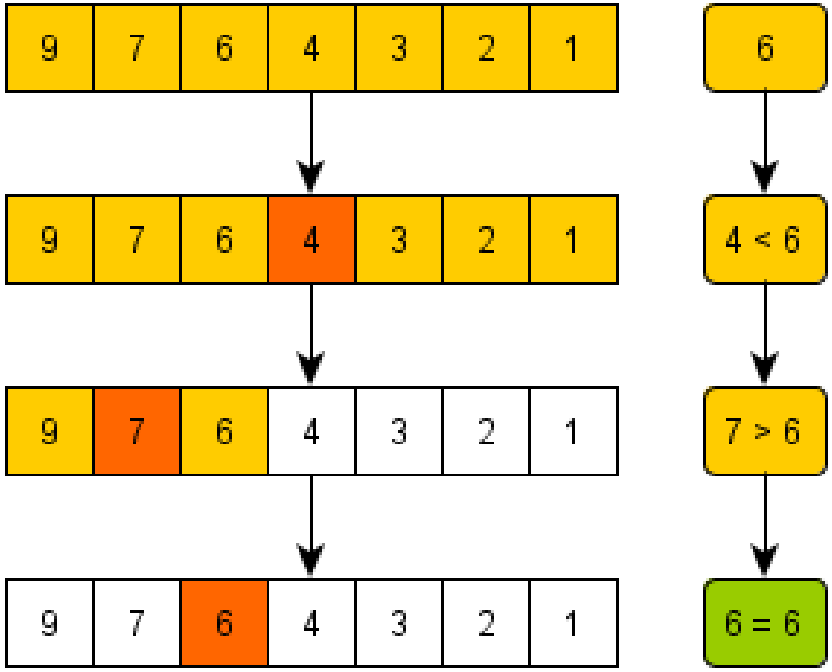
- hledání extrému
  - maximum / minimum
  - v setříděném poli na kraji
  - v nesetříděném musíme projít celé pole
- více stejných hodnot v poli
  - která z nich je ta požadovaná?
- nejhorší případ
  - hledáme hodnotu, která v poli není
- co když není přesná hodnota?
  - hledat menší
  - hledat větší
  - zahlásit chybu?

Vyhledávání sekvenční

- hledání pozice hodnoty v nesetříděném poli dat
  - postupně projít všechny prvky
- vhodný pro malé vzorky dat
  - maximální O(n)
  - průměrná O(n/2)
- algoritmus má dvě porovnání
  - konce pole
  - nalezení hodnoty
  - zjednodušení?
    - Na konec pole se dá hledaná hodnota
    - zarážka (sequential search sentinel)

Vyhledávání binární

- vyžaduje setříděné pole
- nejhorší O(log2 n)
- vyhledávání pomocí metody půlení intervalu



Vyhledávání interpolační

- vyžaduje setříděné pole
- podobné binárnímu
- pomáháme si znalostí
  - odhadem pozice
  - na základě znalosti rozložení hodnot
- O(log (log n))

- hledání jmen
  - určíme si četnost zastoupení jednotlivých písmen
  - z rozložení určíme počátek, krok

  - pomocná tabulka – pozice prvních jmen dle písmena
- hledání výšek osob

Řazení

- data
  - jednoduchá (int, float…)
  - složitá (záznamy v databázi, vektor, komplexní číslo)
- směr řazení
  - sestupně
  - vzestupně
- stanovení porovnávací funkce
  - vrací
    - `-1` – první je před druhým
    - `0` – parametry stejné/nezáleží na pořadí
    - `+1` – první je za druhým

Řazení

- dvojce hodnota – klíč
  - řazení pomocí klíčů
  - stabilní řazení
    - zachovává pořadí položek se stejným klíčem
    - hodnota 0 porovnávací fce
      - ➢ nechat prvky jak jsou
- prostor pro řazení
  - stejný
  - pomocný

➢paměťová náročnost

Řazení

- testy pro ověření kvality a času řazení
  - správně seřazená data
  - opačně seřazená data
  - téměř seřazená data správně
  - téměř seřazená data opačně
  - náhodná data
- přirozený algoritmus
  - vyšší rychlost pro částečně seřazená data

Bubble sort

- řazení probubláním
- princip
  - porovnáváme sousedy a jejich případná výměna
- složitost O(n^2)
- u téměř seřazeného pole se blížíme k O(n)
- velké hodnoty zleva probublají rychle
  - maximum se na konec dostane v jednom cyklu
- malé hodnoty se doleva posunují pomalu
  - v jednom cyklu posun maximálně o jednu pozici

Bubble sort


5 1 12 -5 16
Nesetříděná řada
5 1 12 -5 16
5 > 1
Výměna prvků
1 5 12 -5 16
5 < 12
Pořadí zůstává
1 5 12 -5 16
12 > -5
Výměna prvků
1 5 -5 12 16
12 < 16
Pořadí zůstává
1 5 -5 12 16
1 < 5
Pořadí zůstává
1 5 -5 12 16
5 > -5
Výměna prvků
1 -5 5 12 16
5 < 12
Pořadí zůstává
1 -5 5 12 16
1 > -5
Výměna prvků
-5 1 5 12 16
1 < 5
Pořadí zůstává
-5 1 5 12 16
-5 < 1
Pořadí zůstává
-5 1 5 12 16
Setříděná řada
4 průchody

Shaker sort

- vychází z bubble sortu
- princip
  - netřídí jen jedním směrem
  - směr průchodu se po každém cyklu mění
  - obousměrný bubble sort
- složitost O(n^2)
- z praktického hlediska neefektivní
  - odstraňuje některé nedostatky bubble sortu
  - nedojde k výraznému zlepšení

Comb sort

- vychází z bubble sortu
- princip
  - zavádí se snižující přírůstek
  - volí se různá vzdálenost mezi porovnávanými prvky
  - začne se velkou vzdálenosti (půl intervalu)
  - postupně se vzdálenost zkracuje (faktor 1.3) až k jedné
  - v poslední fázi degraduje na klasický bubble sort
- složitost: O(n) až O(n^2)

Select sort

- řazení výběrem
- pole rozdělíme
  - seřazená část
  - neseřazená část
- princip
  - vybíráme extrém z neseřazené části
  - vyměníme s prvním prvkem v neseřazené čísti
  - zmenšíme neseřazenou část
- složitost O(n^2)
- konstantní paměťová složitost

Select sort

Nesetříděná část
Prvky se vybírají postupně

15	4	28	9	51	21	14
4	15	28	9	51	21	14
4	9	28	15	51	21	14
4	9	14	15	51	21	28
4	9	14	15	51	21	28
4	9	14	15	21	51	28
4	9	14	15	21	28	51
4	9	14	15	21	28	51

Setříděná část
Prvky vkládané na místo podle velikosti

Double select sort

- vylepšení algoritmu select sort
- současně hledáme oba extrémy
- setříděné části vznikají na obou stranách

Insert sort

- řazení vkládáním
- pole rozdělíme
  - seřazená část
  - neseřazená část
- princip
  - bereme prvky z neseřazené části
  - vkládáme je na místo v seřazené části
- složitost O(n^2)
- u téměř seřazeného pole se blížíme k O(n)

Insert sort

Nesetříděná část
Prvky se vybírají postupně

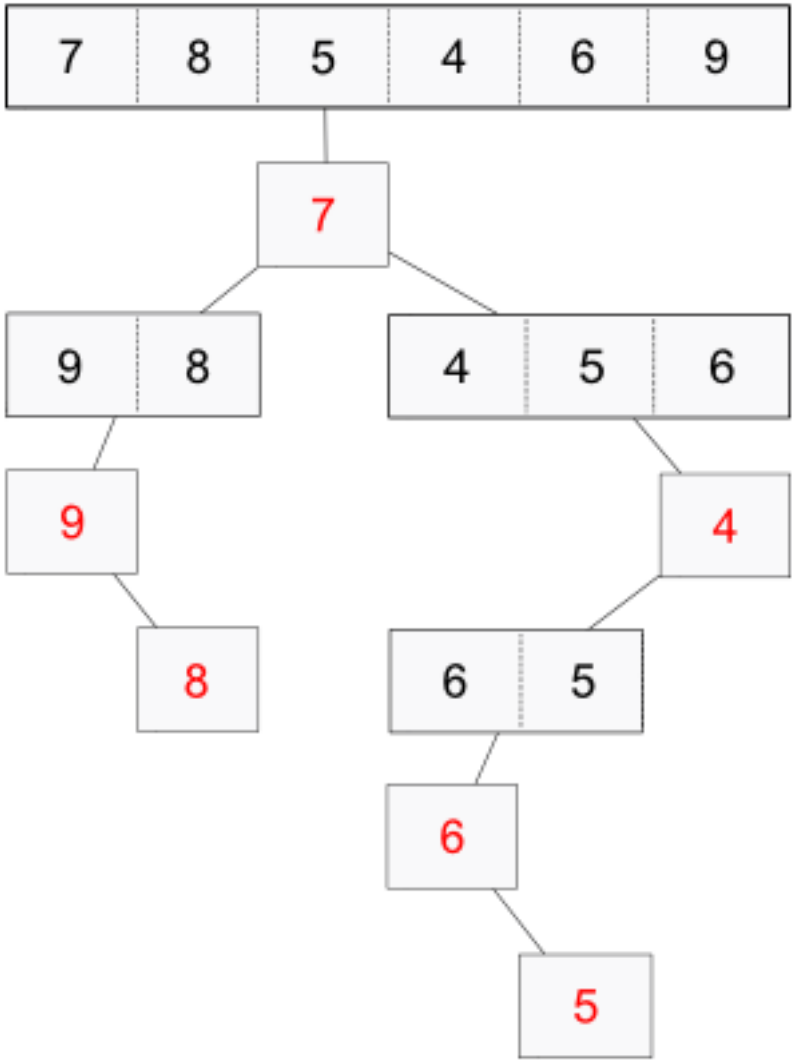
15	4	28	9	51	21	14
4	15	28	9	51	21	14
4	15	28	9	51	21	14
4	9	15	28	51	21	14
4	9	15	28	51	21	14
4	9	15	21	28	51	14
4	9	14	15	21	28	51

Setříděná část
Prvky vkládané na místo podle velikosti

Quick sort

- Maximální složitost: O(n^2)
- Ve většině případů se ale blížíme k: O(n.log n)
- nejrychlejší mezi běžnými algoritmy
- pokud v datech hodně stejných hodnot, horší vlastnosti
- princip
  - vyber pivot
    - ideálně rozděl data na 2 stejné části
    - např. první prvek
    - poslední prvek
    - medián (nejlepší), nutná apriorní znalost
  - rozděl hodnoty na menší a větší jak pivot
    - menší přemísti vlevo od pivotu
    - větší přemísti vpravo od pivotu
  - obě části - rekurze

Quick sort



Merge sort

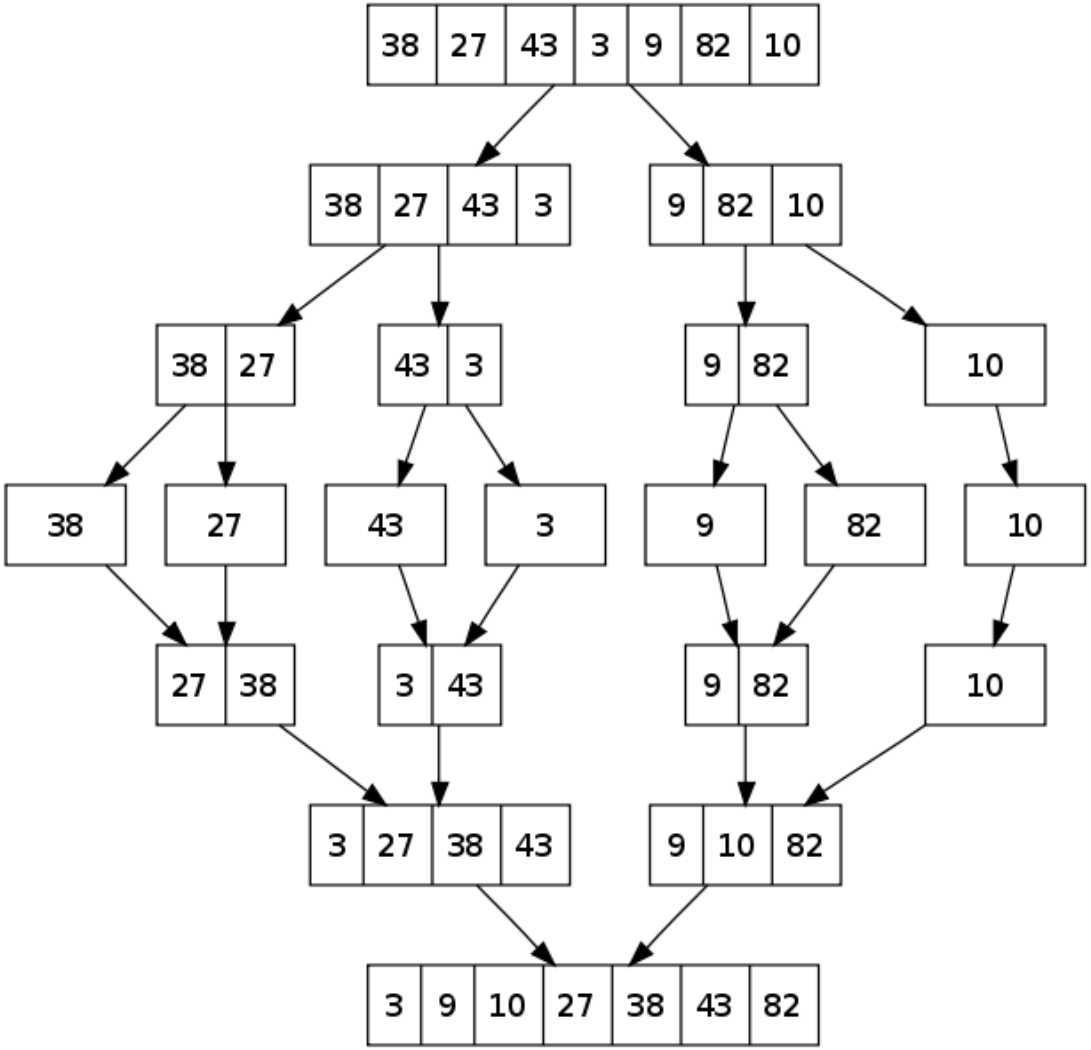
- složitost O(n.log n)
- potřebuje pomocnou paměť
- vhodný k paralelizaci
- princip
  - pokud jsou dva (nebo jeden) prvky
    - seřaď je
    - vrať se o úroveň výše
  - rozděl data na polovinu
  - na každou polovinu zavolej tuto funkci
  - spoj obě poloviny
    - poloviny jsou setříděné
    - ➢ jejich spojení je tedy rychlé

Merge sort

- spojování seznamu
  - máme 2 setříděné seznamy
  - máme výstupní seznam

  - porovnáváme první hodnoty z obou seznamů
  - menší vložíme na první pozici výstupního seznamu
  - po vyprázdnění jednoho seznamu
    - kopírujeme zbytek dat z druhého

Merge sort


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

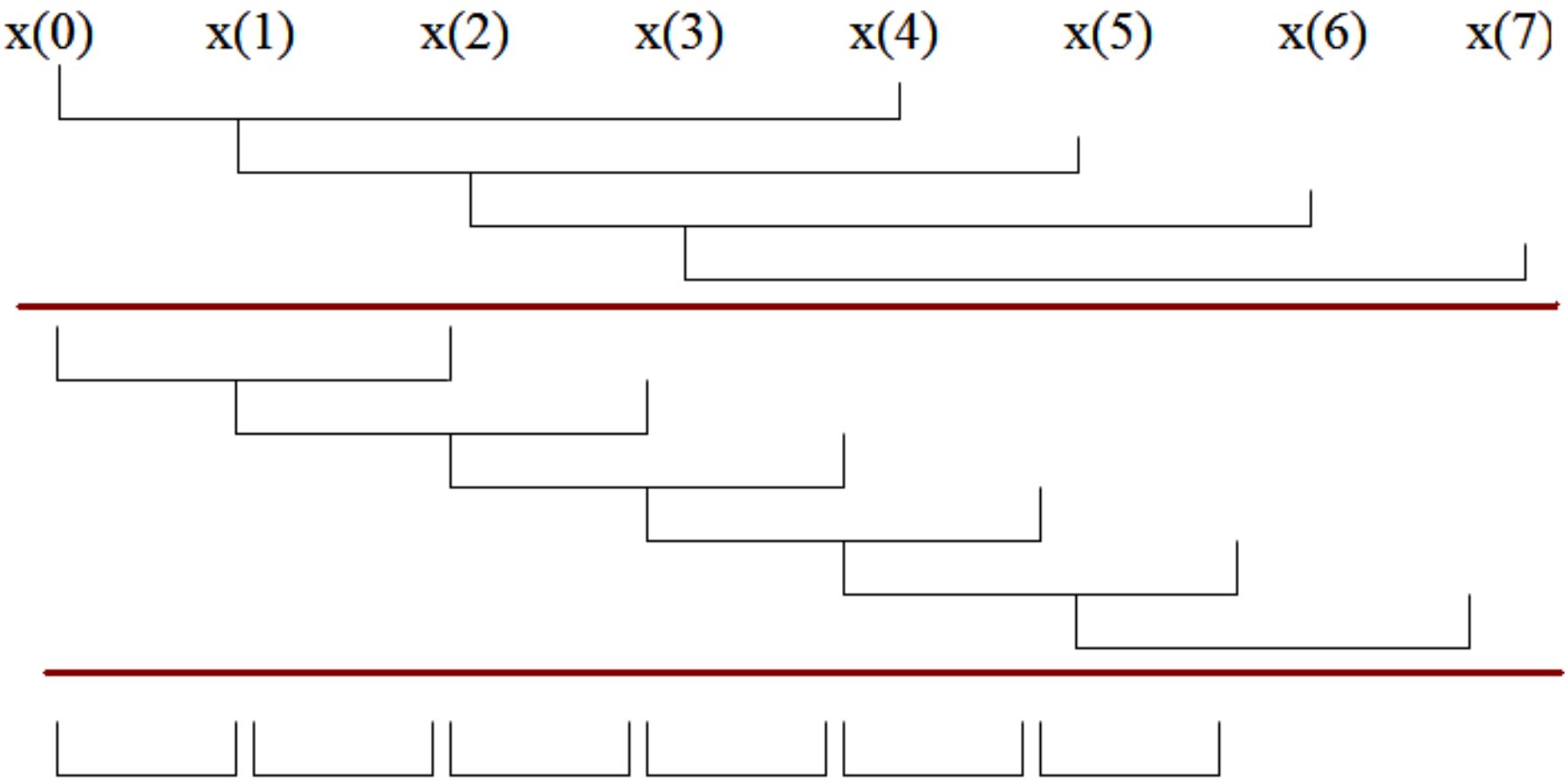
Shell sort

- složitost O(n^(1.5))
- Modifikace Insert sortu
  - menší nároky na kód
  - menší nároky na zásobník
- algoritmus:
  - zvolí se původní vzdálenost (např. 4)
  - začne se na prvním prvku
    - vezmou se prvky s danou vzdáleností
    - seřadí se pomocí insert sortu
  - posunujeme se o jeden prvek
    - předchozí krok se opakuje
    - provedeme řazení vybraných prvků jako v minulém bodě
  - snížíme vzdálenost
  - opakujeme celý postup
  - poslední iterace -> klasický insert sort

Shell sort

- krok
  - celočíselné násobky čísla 2.2
  - začínáme od největšího možného kroku
- výhoda
  - extrémy velmi rychle přemístěny na odpovídající stranu pole
  - poslední iterace algoritmu
    - ➢ přesouváme naprosté minimum prvků

Shell sort



Bucket sort

- podle klíče se rozřadí do sekcí
  - například podle hodnoty v nejvyšším řádu
- hodnoty v sekci se seřadí nějakým z řadících algoritmů
  - podle charakteru dat možno pro každou sekci jiný algoritmus
- hodnoty se nakopírují do výsledného pole
- musíme dále řadit?
  - každá sekce pokrývala určitý rozsah
  - rozsahy se nepřekrývaly
  - není nutné další zpracování
- vhodné pro paralelní zpracování
  - ➢co bucket to vlákno
- Složitost O(m*C(n/m))
  - C(n) – složitost vnitřního třídícího algoritmu

Counting sort

- O(n+k)
  - n- počet prvků
  - k - počet různých prvků
- vhodný pro velká pole s opakujícími se hodnotami
- dají se řadit jen diskrétní hodnoty
- algoritmus:
  - nejprve se zjistí četnost jednotlivých hodnot v poli
  - vytvoří se pole posledních indexů
    - tj. hodnota značí poslední index s danou hodnotou
  - pole četností a výskytů se použije pro vytvoření seřazeného pole

Counting sort

Vstupní pole	9	6	6	3	2	0	4	2	9	3
Výčet prvků	0	1	2	3	4	5	6	7	8	9
Pole četností	1	0	2	2	1	0	2	0	0	2
Pole výskytů	0	0	2	4	5	5	7	7	7	9
Seřazené pole	0	2	2	3	3	4	6	6	9	9

Radix sort

- řazení řetězců totožné délky
- přímo z definice stabilního řazení
  - postupně jdeme znak po znaku a řadíme
- algoritmus
  - vezmi první znaky všech řetězců
  - řetězce seřaď podle tohoto znaku
    - volá jiný stabilní algoritmus
    - často Counting sort
  - na řetězce se stejným prvním znakem zavolej tento algoritmus
- O(m.C(n))

Intro sort

- v knihovních funkcích C++
- kombinace více metod
  - kombinace quick a heap sort
  - insert sort – použitý na úseky kratší jak 16 prvků
- princip
  - ke třídění se používá quick sort
  - je-li při kontrole zjištěn nevhodný případ dat pro quicksort, vyřeší se daná množina pomocí heap sortu
  - je-li hloubka rekurze větší než doporučená mez (stanovená podle počtu dat), zbývající data seřadíme pomocí heapsort.

Heap sort

- jeden z nejlepších algoritmů
- využívá současně reprezentace dat pomocí
  - pole
  - stromu/binární haldy
- popis fungování v příští přednášce

Děkuji za pozornost

zdroj obrázků a další materiály: https://algoritmy.net