ZALG - 11. cvičení

Quicksort

Založen na principu rozděl a panuj

1. Vezmeme libovolný prvek posloupnosti (pivot) a posloupnost rozdělíme na 3 skupiny. V první budou prvky menší než pivot, ve druhé prvky s hodnotou stejnou jako pivot a ve třetí prvky s hodnotou větší než pivot
2. Po 1. kroku jsou prvky se stejnou hodnotou jako pivot na správném místě. Prvky s menší hodnotou jsou nalevo, ale ne nutně seřazené a stejně tak prvky s větší hodnotou jsou napravo
3. Skupiny jsou vůči sobě seřazené - nyní stačí seřadit prvky v první a třetí skupině opět podle prvního kroku

Quicksort - rozdělení posloupnosti

- Začneme od prvního prvku posloupnosti a budeme hledat ten prvek, který má hodnotu rovno nebo větší než pivot
- Zároveň půjdeme od konce a budeme hledat prvek, který má hodnotu rovno nebo menší než pivot. Až najdeme tyto dva prvky, tak je prohodíme
- Až se oba směry prohledávání setkají uprostřed, tak skončíme

Quicksort - jiné rozdělení posloupnosti

- Pivot (jeho hodnotu) uložíme na první místo posloupnosti (pivot prohodíme s prvním prvkem).
- Pak postupně od druhého pole projdeme pole, ty prvky s hodnotou větší než má pivot odložíme na konec. První takový na poslední místo, druhý na předposlední atd…
- Na závěr vložíme pivota na správné místo - prohodíme ho s posledním prvkem s hodnotou menší než pivot

Quicksort - algoritmus

- Rekurzivní
  - Metodou rozděl rozdělení rozdělíme posloupnost na 3 skupiny
  - Zavoláme rekurzivně třídící funkci na 1. skupinu
  - Zavoláme rekurzivně třídící funkci na 3. skupinu
- Nerekurzivní
  - s využitím zásobníku
  - prvně do něj vložíme celý interval a pak v cyklu opakujeme, dokud není zásobník prázdný
    - vyjmeme horní interval a rozdělíme
    - vložíme do zásobníku 1. skupinu (pokud tam stále je)
    - vložíme do zásobníku 3. skupinu (pokud tam stále je)

Časová složitost

- záleží na tom, jak dobře jsme zvolili pivot
- **nejlepší případ**: pivot = medián prvků
  $\rightarrow T(n) = 2T(n/2) + O(n) = O(nlog_2 n)$
  (na rozmyšlenou, použijte master theorem = kuchařku)
- **nejhorší případ**: pivot = nejmenší nebo největší prvek úseku
  $\rightarrow T(n) = 2T(n-1) + (n-1) = O(n^2)$
  (například: již setříděné pole a jako pivot bereme první prvek úseku)
- **průměrný případ**: $T(n) = O(nlog_2 n)$ (viz skripta)

Prostorová složitost

- **nejhorší případ**: $S(n) = O(n)$ (maximální hloubka rekurze nebo velikost zásobníku)
- **průměrný případ**: $S(n) = O(log_2 n)$

Hoarův algoritmus

Metoda vyhledání k-tého nejmenšího prvku pole

Není nutné třídit úplně celé pole.

Základní myšlenka:

- Pole na intervalu [l,r], Opakuj:
- Zvol pivot a proveď PARTITION (z quicksortu)
- Je-li pivot k-tým prvkem -> return k
- Je-li index (p) pivotu > k => přesuň se na interval [l, p-1]
- Jinak se přesuň na interval [p+1, r]

Radix sort

- jako klíče používá číslice (nejčastěji v desítkové soustavě) tříděných čísel
- předpokládá znalost maximálního možného počtu (m) číslic v číslech

1. Vytvoříme pole 10 front. Proměnné *z* přiřadíme hodnotu 10 a proměnné *d* hodnotu 1
2. Pro *l = 0,…,m-1* provedeme kroky 3-9. Pak algoritmus skončí
3. Je-li vstupní posloupnost prázdná, jdi na krok 8
4. Vyjmi další číslo ze vstupní posloupnosti a vlož ho do x
5. Vypočti *i = [x/d] mod z*
6. Vlož *x* na konec *i*-té fronty
7. Vrať se na krok 3.
8. Přejdi na další číslici – Do *d* ulož *dz*
9. Přenes hodnoty z nulté, pak z první, …, deváté fronty – v pořadí v jakém jsou v těchto frontách – zpět do vstupní posloupnosti

Bucket sort

- V případě, že prvky nabývají obecně malého počtu různých hodnot (K)
- Vytvoříme si K bucketů (front nebo vektorů)
- Projdeme celou posloupnost a jednotlivé prvky uložíme do příslušných bucketů. Prvek s hodnotou 0 do nultého bucketu, prvek s hodnotou 1 do prvního atd…
- Nakonec buckety sloučíme popořadě od nultého bucketu až na závěr K-tý bucket
- Složitost je O(n)

Časová složitost

1. přesypání prvků do přihrádek: $T(n) = O(n + k)$
2. vysypání prvků z přihrádek: $T(n) = O(n)$
3. celkem: $T(n) = O(n + k)$

Prostorová složitost $S(n) = O(n + k)$ (na přihrádky)