Programozás I.

Oszd meg és uralkodj elvű algoritmusok

Sergyán Szabolcs
sergyan.szabolcs@nik.uni-obuda.hu

Óbudai Egyetem
Neumann János Informatikai Kar

November 24, 2013

1. Oszd meg és uralkodj elv
2. Rekurzív rendezések
   - Merge sort
   - Quicksort
3. *k*-adik legkisebb elem keresése

Oszd meg és uralkodj elv

- Oszd meg és uralkodj elv:
  - A megoldandó problémát felosztjuk kisebb részfeladatokra
  - Az egyes részfeladatokat rekurzív módon megoldjuk
  - A részfeladatok megoldásait egyesítjük

- Ezzel a megközelítéssel olyan problémák is megoldhatók, amiket más módszerrel is meg lehet oldani

Példa: maximumkiválasztás

Bemenet: X - tömb, N - X elemszáma
Kimenet: X elemeinek maximuma

Függvény FelezőMaximumkiválaszt(X,N)
    Ha $N = 1$ **akkor**
        return(X[1])
    Különben
        $K \leftarrow \lfloor N/2 \rfloor$
        $BalMax \leftarrow$ FelezőMaximumkiválaszt($X[1..K]$,K)
        $JobbMax \leftarrow$ FelezőMaximumkiválaszt($X[K+1..N]$,$N-K$)
        Ha $BalMax \geq JobbMax$ **akkor**
            return(BalMax)
        Különben
            return(JobbMax)
        Elágazás vége
    Elágazás vége
Függvény vége

Merge sort

Alapötlet

- Az n elemű tömböt felosztjuk két ($n/2$ elemű) résztömbre
- A résztömböket rekurzív módon rendezzük, azaz
  - továbbosztjuk fele olyan hosszú résztömbökre
  - 1 elemű tömb önmagában rendezett
- A rendezett résztömböket összefuttatjuk megtartva a rendezettséget

Merge sort algoritmus

Pszeudokód

Eljárás Merge-sort(X, E, U)
    Ha $E < U$ **akkor**
        $K \leftarrow \left\lfloor \frac{E+U}{2} \right\rfloor$
        Merge-sort(X, E, K)
        Merge-sort(X, $K+1$, U)
        Merge(X, E, K, U)
    Elágazás vége
Eljárás vége

Az eljárás első hívásakor Merge-sort(X, 1, N) módon hívjuk, mivel a teljes tömböt akarjuk rendezni. X az A tömb elemszáma.

Összefuttatás

Pszeudokód

Eljárás Merge(X, E, K, U)
  $N_1 \leftarrow K - E + 1$; $N_2 \leftarrow U - K$
  Ciklus $i \leftarrow$ **1-től** N_1**-ig**
    $L[i] \leftarrow X[E + i - 1]$
  Ciklus vége
  Ciklus $j \leftarrow$ **1-től** N_2**-ig**
    $R[j] \leftarrow X[K + j]$
  Ciklus vége
  $L[N_1 + 1] \leftarrow +\infty$; $R[N_2 + 1] \leftarrow +\infty$
  $i \leftarrow 1$; $j \leftarrow 1$
  Ciklus $k \leftarrow E$**-től** U**-ig**
    Ha $L[i] \leq R[j]$ **akkor**
      $X[k] \leftarrow L[i]$
      $i \leftarrow i + 1$
    Különben
      $X[k] \leftarrow R[j]$
      $j \leftarrow j + 1$
    Elágazás vége
  Ciklus vége
Eljárás vége

Megjegyzések

- Az algoritmus futási ideje $O(N \cdot \log N)$-es
- A megvalósításhoz szükségünk van segédtömbökre, így nagyméretű tömbök esetén helyfoglalás szempontjából nem hatékony

Alapötlet

- Válogassuk szét úgy a rendezendő X tömb elemeit, hogy az első elemnél kisebb értékű elemek az első elem elé, a nagyobbak pedig mögé kerüljenek.
- Végezzük el ezt a szétválogatást az első elemnél kisebbekre, illetve nagyobbakra külön-külön.
- Ez az eljárás az *Oszd meg és uralkodj!* elvet használja.

Quicksort

Szétválogató eljárás

Eljárás Szétválogat(X, E, U, K)
    $K \leftarrow E$; $L \leftarrow U$; $A \leftarrow X[K]$
    Ciklus amíg $K < L$
        Ciklus amíg $(K < L)$ **és** $(X[L] \geq A)$
            $L \leftarrow L - 1$
        Ciklus vége
        Ha $K < L$ **akkor**
            $X[K] \leftarrow X[L]$; $K \leftarrow K + 1$
            Ciklus amíg $(K < L)$ **és** $(X[K] \leq A)$
                $K \leftarrow K + 1$
            Ciklus vége
            Ha $K < L$ **akkor**
                $X[L] \leftarrow X[K]$; $L \leftarrow L - 1$
            Elágazás vége
        Elágazás vége
    Ciklus vége
    $X[K] \leftarrow A$
Eljárás vége

Rekurzív hívás

Eljárás Quick(X, E, U)
  Szétválogat(X, E, U, K)
  Ha $K - E > 1$ **akkor**
    Quick(X, E, $K - 1$)
  Elágazás vége
  Ha $U - K > 1$ **akkor**
    Quick(X, $K + 1$, U)
  Elágazás vége
Eljárás vége

Megjegyzés

- A Quicksort-nál alkalmazott Szétválogat metódus mindig a vizsgált résztömb első eleméhez viszonyítva válogatja két részre a résztömböt.
- Emiatt pl. elve rendezett tömb esetén az algoritmus futási ideje $O(N^2)$-es, míg átlagos esetben csak $O(N \cdot \log N)$-es.
- Javíthatunk az algoritmuson, ha véletlenszerűen jelöljük ki, hogy melyik legyen az az elem a vizsgált résztömbből, amelyhez viszonyítva szétválogatjuk a résztömb elemeit.

Oszd meg és uralkodj elvű algoritmusok

k-adik legkisebb elem keresése

- Feladat: egy tömbben a k-adik legkisebb elemet szeretnénk megkeresni
- Megoldásai javaslat: rendezzük a tömböt, majd vegyük a (rendezett) elemek közül a k-adikat
- Eddigi ismereteink alapján ez átlagosan $O(N \cdot \log N)$ időt igényel
- Nem lehetne ezen valamelyest gyorsítani?

Pszeudokód

Függvény KiválasztK-adik(X, E, U, k)
  Ha $E = U$ **akkor**
    return $X[E]$
  Elágazás vége
  Szétválogat(X, E, U, K)
  Elágazás
    $k = K$ **esetén**
      return $X[K]$
    $k < K$ **esetén**
      KiválasztK-adik(X, E, $K - 1$, k)
    $k > K$ **esetén**
      KiválasztK-adik(X, $K + 1$, U, $k - K$)
  Elágazás vége
Függvény vége

k-adik legkisebb elem keresése

Megjegyzések

- Az ismertetett algoritmus átlagos esetben $O(N)$-es, de legrosszabb esetben $O(N \log N)$-es.
- Nem lehetne valamilyen módon úgy javítani, hogy legrosszabb esetben is csak $O(N)$-es legyen?
- Probléma ott van, mint Quicksort algoritmusnál, hogy a szétválogatásnál használt viszonyítási (ún. őrszem) elemet miként választjuk ki.

k-adik legkisebb elem keresése

Ötlet

1. A bemeneti tömb n darab elemét rendezzük $\lfloor n/5 \rfloor$ darab 5 elemből álló csoportba, a maradék $n \mod 5$ darab elemből alkossunk egy újabb csoportot (ha $n \mod 5 \neq 0$).
2. Az $\lceil n/5 \rceil$ darab csoportnak keressük meg a mediánját[a] (3. legkisebb) elemét. Ehhez pl. rendezzük az 5 darab elemet javított beillesztéses rendezéssel, majd válasszuk ki minden csoportból a mediánt.
3. A KiválasztK-adik függvény rekurzív használatával határozzuk meg az előző lépésben kapott $\lceil n/5 \rceil$ darab medián x-szel jelölt mediánját.
4. A Szétválogatást valósítsuk meg úgy, hogy x-et használjuk őrszem elemként.

[a] Medián: sorba rendezve a tömb elemeit az $\lceil N/2 \rceil$-edik elem, ahol N a tömb elemszáma.

k-adik legkisebb elem keresése

Az ismertetett algoritmus azért működik megfelelően, mert az őrszem elem ilyen választásával elérhető, hogy az elemek legalább 3/10-e egy csoportba kerül a szétválogatás során.