Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 6. PREDAVANJA (8.11.2007)

Študijsko leto 2007/08

Sortiranje tabel – Sortiranje s kopico

Lastnosti kopice:

- binarno drevo
- vozlišče drevesa ustreza objektu v tabeli
- za vsako vozlišče velja, da je objekt večji ali enak od vseh naslednikov
- dolžini dveh poljubnih vej v kopici se razlikujeta kvečjemu za ena, daljše veje so skrajno levo

Ponazoritev kopice v računalniku:

- še vedno sortiramo tabelo `Element[] a`
- koren je v `a[0]`
- levi sin vozlišča `a[i]` je v `a[2*i+1]`
- desni sin vozlišča `a[i]` je v `a[2*i+2]`

Sortiranje tabel – Sortiranje s kopico 24

Opis algoritma: 4 koraki

1. vhodno zaporedje uredimo v kopico
2. zamenjamo prvi in zadnji element v kopici
3. popravimo kopico
4. ponavljamo koraka 2 in 3, dokler ne zmanjka podatkov

Popravljanje kopice (del koraka 1 in 3)

- kopico z neustreznim korenom popravimo tako, da koren pogreznemo po veji z večjim naslednikom
- postopek ponavljamo, dokler ne popravimo kopice (ali pridemo do lista)
- **Podprogram za popravljanje kopice: `heapify`**

Sortiranje tabel – Sortiranje s kopico 25

Gradnja kopice

- uporabimo postopek pogrezanja
- objekti v desni polovici tabele (`i>=a.length/2`) nimajo naslednikov
- objekte pogrezamo v zanki od sredine tabele proti začetku
- **del metode za ta del ...**

Zanka, v kateri ponavljamo koraka 2 in 3

- **Del programa za ta del ...**

Celoten algoritem ...

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda Heapsort ...

Sortiranje tabel – Sortiranje s kopico 26

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Heapsort),
- sprememba razreda GlavniProgram ...

Izboljšava navadnega izbiranja

Analiza časovne kompleksnosti:

- gradnja kopice na začetku: $O(\log_2 n)$
- popravljanje kopice: $O(\log_2 n)$
- zamenjava prvega in zadnjega objekta: $O(n)$
- **Skupaj:** $T_w = O(n*\log_2 n)$

Sortiranje tabel – Quicksort 27

Osnova: porazdelitveni algoritem, ki tabelo razdeli na 2 dela:

1. izberemo poljubni element tabele **`x`** (npr. srednji)
2. pregledujemo tabelo od leve proti desni, dokler ne najdemo **`a[i] > x`**
3. pregledujemo tabelo od desne proti levi, dokler ne najdemo **`a[j] < x`**
4. zamenjamo **`a[i]`** in **`a[j]`**
5. ponavljamo korake 2 – 4, dokler se pregledovanji ne srečata

   Porazdelitveni algoritem ...

   Prikaz delovanja algoritma ...
6. postopek **rekurzivno** ponavljamo na levem in desnem delu tabele

Ralizacija metode v Javi: metodi Quicksort in Sort ...

Sortiranje tabel – Quicksort 28

Primer:

- dopolnitev razreda Sortiranje objektov (z metodama Quicksort in Sort),
- sprememba razreda GlavniProgram ...

Izboljšava principa navadne zamenjave

Analiza časovne kompleksnosti:

- ena porazdelitev: O(n)
- število rekurzivnih klicev:
  - ugoden (povprečen) primer: $O(\log_2 n)$
  - najslabši primer: O(n)
- **Skupaj:** $\mathbf{T_a = O(n*\log_2 n)}$, vendar $\mathbf{T_w = O(n^2)}$
- **Iskanje najslabše permutacije ...**

Sortiranje tabel – Mergesort 29

Ideja iz sortiranja datotek

Še ena implementacija metode Deli in vladaj:

1. Razdeli tabelo na dve (približno) enako veliki podtabeli
2. Rekurzivno uredi levo podtabelo
3. Rekurzivno uredi desno podtabelo
4. Zlij levo in desno podtabelo tako, da ohraniš urejenost

Prikaz delovanja algoritma ...

Problem zlivanja: pomožna tabela `temp`

Ralizacija metode v Javi: metode Mergesort, Msort in Merge ...

Primer:

- dopolnitev razreda Sortiranje objektov
  (z metodami Mergesort, Msort in Merge),
- sprememba razreda GlavniProgram ...

Izboljšava principa navadne zamenjave

Analiza časovne kompleksnosti:

- porazdelitev ni
- število rekurzivnih klicev Msort: $O(\log_2 n)$
- zlivanje: $O(n)$
- **Skupaj:** $\mathbf{T = O(n*\log_2 n)}$, vendar slabše kot Quicksort