Föreläsning
Datastrukturer (DAT037)

Nils Anders Danielsson

2015-12-07

Tidigare

Sortering:

- Insättningssortering.
- Mergesort.
- Stabilitet.
- Heapsort.
- Undre gräns.
- Radixsortering.

Idag

Sortering:

- Heapsort.
- Quicksort.
- Adaptiva sorteringsalgoritmer.
- Sortering i JDK och Haskells standardbibliotek.

Idag

Antagande: Jämförelser tar konstant tid.

Heapsort

Prioritetskösortering

- Enkel sorteringsalgoritm: Sätt in varje element, kör `delete-min` tills kön är tom.
- $O(n \log n)$ om `insert` och `delete-min` har tidskomplexiteten $O(\log n)$.

Heapsort

- Variant av den enkla prioritetskösorteringsalgoritmen.
- *In-place*: Kräver $O(1)$ extra minne.

Heapsort

- Bygg binär maxheap in-place med `build-heap` (med roten på position 0).
- Kör `delete-max` upprepade gånger.
  Lägg största elementet sist i arrayen,
  nästa element näst sist, och så vidare.

Är heapsort stabil?

- Ja.
- Nej.

Heapsort

- Notera att algoritmen inte är in-place om den implementeras med rekursion (i avsaknad av ”tail-call optimisation” e d): rekursion använder minne (stack).

Quicksort

Quicksort

Idé (ej implementation):

```
quicksort :: Multiset a -> List a
quicksort ∅   = []
quicksort {x} = [x]
quicksort S   = quicksort S₁ ++ [p] ++ quicksort S₂
  where
  p ∈ S
  S₁ ∪ S₂ = S \ p
  ∀ x ∈ S₁. x ≤ p
  ∀ x ∈ S₂. x ≥ p
```

Quicksort

- Sorterar array.
- Effektiv om implementerad på bra sätt (se boken för några detaljer).
- Partitionerar array in-place.
- Rekursiva anrop för arraysegment.
- Tight inre loop (liten konstant).

Vad är värstafallstidskomplexiteten för quicksort om `p` alltid är arraysegmentets första element? Anta att partitionering tar linjär tid.

- $\Theta(n)$.
- $\Theta(n \log n)$.
- $\Theta(n^2)$.
- $\Theta(n^2 \log n)$.
- $\Theta(n^3)$.

Partitioneringsstrategi

- Viktigt välja pivotelementet $\mathtt{p}$ på ett bra sätt, $\mathtt{S}_1$ och $\mathtt{S}_2$ ska helst vara ungefär lika stora.
- Rekommenderade val:
  - Slumpmässigt element.
  - Medianen av tre element:
    första, sista och ett av de mittersta.
- Bör även ha bra strategi för element lika med $\mathtt{p}$.

Quicksort

Tidskomplexitet:

- I värsta fallet (bara dåliga val av $\mathtt{p}$): $\Theta(n^2)$.
- Bara bra val av $\mathtt{p}$, $-1 \leq |\mathtt{S}_1| - |\mathtt{S}_2| \leq 1$: $O(n \log n)$.
- Medelkomplexitet (med bra implementation): $O(n \log n)$.

Quicksort

Många implementationer ej stabila.

Mer sortering

Adaptiva sorteringsalgoritmer

- Adaptiva sorteringsalgoritmer:
  Går snabbare för listor med viss struktur,
  t ex listor som redan är sorterade.
- Insättningssortering:
  $\Theta(n + i)$, där i är antalet *inversioner*:
  par $j < k$ med $\mathtt{xs}[j] > \mathtt{xs}[k]$.
  $\Theta(n^2)$ för omvänt sorterade listor
  (med distinkta element).
- Det finns fler adaptiva sorteringsalgoritmer.

Animationer

http://www.sorting-algorithms.com/

Sortering i Haskell

Data.List.sort:

- En variant av mergesort som först hittar växande/strikt avtagande segment:
  $[1, 2, 3, 4, 3, 2, 1, 1, 1, 2, 3] \mapsto$
  $[[1, 2, 3, 4], [1, 2, 3], [1, 1, 2, 3], []]$
- Stabil, $O(n \log n)$, adaptiv.

Sortering i Java

JDK 7, arrayer med primitiva typer
(med reservation för fel):

- För långa arrayer med `byte` (≥ 31 element), `short`, `char` (≥ 3202):
  Räknesortering (counting sort), en variant av hinksortering med räknare istället för hinkar.
- För korta `byte`-arrayer: Insättningssortering.
- För långa arrayer (≥ 287) med få (≤ 66) segment ($\leq/\geq/$korta $=$): Variant av mergesort.
- Annars: Quicksort med två pivotelement.
- Quicksort använder variant av insättnings-sortering för korta arrayer (≤ 46).

Sortering i Java

JDK 7, arrayer med primitiva typer
(med reservation för fel):

- Adaptiv.
- `byte`, `short`, `char`: $O(n)$.
- Andra primitiva typer: $O(n^2)$ i värsta fallet, medelkomplexitet gissningsvis $O(n \log n)$.
- Stabilitet ointressant för primitiva typer.

Sortering i Java

JDK 7, arrayer med objekt:

- Timsort: Stabil, $O(n \log n)$, adaptiv.
- Drar nytta av ordnade segment.
- Baserad på mergesort och (binär) insättningssortering.

Sammanfattning

Idag:

- Heapsort.
- Quicksort.
- Adaptiva sorteringsalgoritmer.
- Sortering i JDK och Haskells standardbibliotek.