Zesde college complexiteit

4 maart 2008

Inversies, mergesort en merge

Definitie: een inversie van de permutatie $A[1], A[2], \ldots, A[n]$ is een paar $(A[i], A[j])$ waarvoor $i < j$ en $A[i] > A[j]$. M.a.w.: een inversie is een paar $(A[i], A[j])$ dat verkeerd om staat.

Merk op: *elk* sorteeralgoritme moet *alle* aanwezige inversies opheffen.

Verder: als een sorteeralgoritme altijd hooguit één inversie opheft per arrayvergelijking, dan is het aantal vergelijkingen dat wordt gedaan om $A[1], \ldots, A[n]$ te sorteren *ten minste* het aantal inversies van A.

Bovendien: een buursverwisseling (zoals bij Insertion sort) heft altijd precies één inversie op (indien de buurelementen verkeerd om staan).

Stelling. Het maximale aantal inversies dat kan voorkomen in een rijtje van n verschillende waarden is $\binom{n}{2} = \frac{1}{2}n(n-1)$.

Gevolg. Elk sorteeralgoritme (gebaseerd op arrayvergelijkingen) dat hooguit één inversie opheft per vergelijking doet ten minste $\frac{1}{2}n(n-1)$ vergelijkingen in de worst case.

Conclusie: Insertion sort is optimaal voor wat betreft de worst case, binnen de klasse van algoritmen gebaseerd op het doen van arrayvergelijkingen, waarbij per vergelijking hooguit één inversie wordt opgeheven (bijvoorbeeld via buursverwisselingen).

Stelling. Het *gemiddeld* aantal inversies in een permutatie van n verschillende waarden (bijvoorbeeld de getallen 1 t/m n) is $\frac{1}{4}n(n-1)$. Dit onder de aanname dat alle $n!$ permutaties even waarschijnlijk zijn.

Gevolg. Elk algoritme dat sorteert met behulp van arrayvergelijkingen en dat per vergelijking ten hoogste één inversie opheft, moet ten minste $\frac{1}{4}n(n-1)$ vergelijkingen doen in de average case.

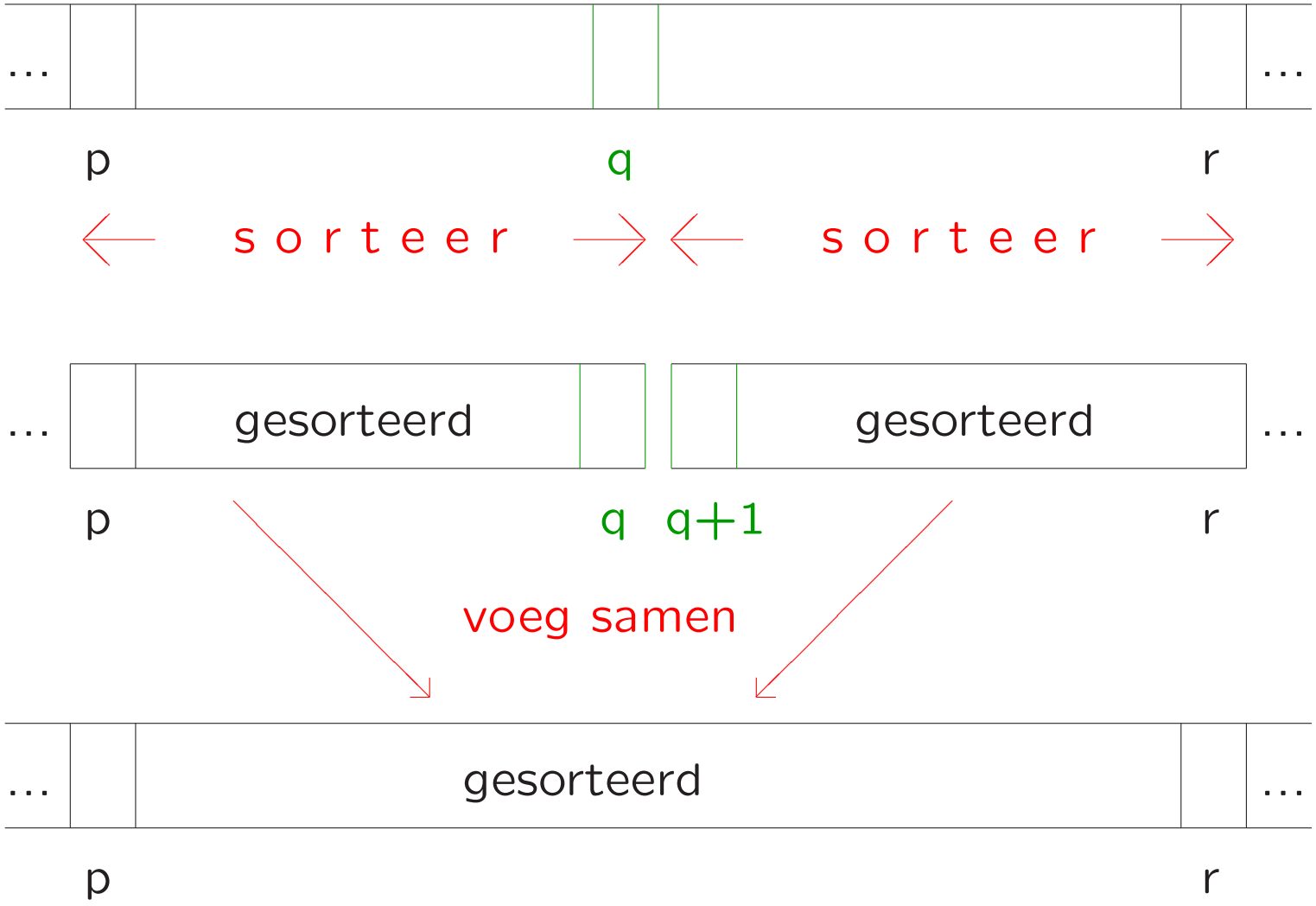
Insertion sort doet gemiddeld $\frac{1}{4}n(n-1)+n-\sum_{i=1}^{n}\frac{1}{i}$ vergelijkingen, dus Insertion sort is in de average case in orde van grootte optimaal (binnen de betreffende klasse van algoritmen), namelijk $\Theta(n^2)$.

Het (recursieve) algoritme:

MergeSort(A, p, r)::
// sorteert $A[p], \ldots, A[r]$

if $p < r$ **then**
$q := \lfloor \frac{p+r}{2} \rfloor$;
MergeSort(A, p, q); verdeel
MergeSort$(A, q+1, r)$; en
Merge(A, p, q, r); heers (voeg samen)
fi

Aanroep: MergeSort$(A, 1, n)$.


...
...
p
q
r
sorteer
sorteer
...
gesorteerd
gesorteerd
...
p
q
q+1
r
voeg samen
...
gesorteerd
...
p
r

Merge(A, p, q, r)::

$i := p$; $j := q + 1$; $k := p$;
while $i \leq q$ **and** $j \leq r$ **do**
  if $A[i] < A[j]$ **then**
    $hulp[k] := A[i]$; $i := i + 1$; $k := k + 1$;
  else
    $hulp[k] := A[j]$; $j := j + 1$; $k := k + 1$;
  fi
od
if $i > q$ **then** // eerste helft is op
  kopieer $A[j], \ldots, A[r]$ naar $hulp$;
else // tweede helft is op
  kopieer $A[i], \ldots, A[q]$ naar $hulp$;
fi
kopieer $hulp[p], \ldots, hulp[q]$ terug naar A;

- Merge(A,p,q,r) voegt de reeds gesorteerde deelrijtjes $A[p], \ldots, A[q]$ en $A[q+1], \ldots, A[r]$ samen tot een gesorteerd stuk $A[p], \ldots, A[r]$

- $hulp$ is een hulparray ter grootte n (net als A)

- Voor het bepalen van de complexiteit van Merge tellen we het aantal vergelijkingen van de vorm: $A[i] < A[j]$

- Is het aantal arrayvergelijkingen hier wel een goede maat voor de complexiteit?

Stel dat we met behulp van Merge twee gesorteerde rijtjes van respectievelijk k en m elementen (met $k + m = n$) samenvoegen tot één gesorteerde rij. Dan geldt:

1. Het aantal vergelijkingen in de worst case is $n - 1$

2. Het aantal vergelijkingen in de best case is $\min\{k, m\}$

Zij $T(n) =$ aantal vergelijkingen in de worst case van Mergesort op n elementen, met $n = 2^k$.

Dan geldt:

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T(\frac{n}{2}) + n - 1 & n = 2^k > 1 \end{cases}$$

Oplossing: $T(n) = n \lg n - n + 1 = \Theta(n \lg n)$

Als n geen tweemacht is, wordt de recurrente betrekking:

$$T(n) = \begin{cases} 0 & n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n - 1 & n > 1 \end{cases}$$

Dan geldt eveneens: $T(n) = \Theta(n \lg n)$. (Zelfs: $T(n) = n\lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1$)

Mergesort is in orde van grootte optimaal voor wat betreft de worst case (zie later). Er is echter extra geheugenruimte ter grootte $\Theta(n)$ nodig.

Zij $B(n) =$ aantal vergelijkingen in de best case, met $n = 2^k$. Dan geldt:

$$B(n) = \begin{cases} 0 & n = 1 \\ 2B(\frac{n}{2}) + \frac{n}{2} & n = 2^k > 1 \end{cases}$$

Oplossing: $B(n) = \frac{n}{2} \lg n = \Theta(n \lg n)$.

Opmerking: het aantal vergelijkingen in de worst case is $\Theta(n \lg n)$, evenals het aantal vergelijkingen in de best case.
Gevolg: het aantal vergelijkingen in de average case is ook $\Theta(n \lg n)$.

Stelling.

1. Elk algoritme, alleen gebaseerd op arrayvergelijkingen, dat twee gesorteerde arrays (rijen) van lengte m samenvoegt tot één gesorteerd array, doet in het slechtste geval ten minste $2m-1$ van zulke vergelijkingen.
   Voor $m = \frac{n}{2}$ (n even) is dit dus ten minste $n-1$.

2. Voor het samenvoegen van twee rijtjes ter lengte $m-1$ respectievelijk m is dat ten minste $2m-2$.
   Voor $m = \lceil \frac{n}{2} \rceil$ (n oneven) is dit ten minste $n-1$.

Gevolg. Binnen de klasse van samenvoegalgoritmen gebaseerd op arrayvergelijkingen is het beschreven Mergealgoritme optimaal.

We geven een klasse van invoerrijtjes waarop *elk* samenvoegalgoritme (gebaseerd op arrayvergelijkingen) *ten minste* $2m-1$ vergelijkingen doet. Dat bewijst dan de stelling.

Kies de rijtjes $A = (a_1, a_2, \ldots, a_m)$ en $B = (b_1, b_2, \ldots, b_m)$ zo dat

$$b_1 < a_1 < b_2 < \cdots < a_{i-1} < b_i < a_i < b_{i+1} < \cdots < b_m < a_m$$

Dan *moet* elk samenvoegalgoritme a_i met b_i vergelijken $(i = 1, 2, \ldots, m)$ en a_i met b_{i+1} $(i = 1, 2, \ldots, m-1)$.

Het bewijs van deze bewering gaat uit het ongerijmde.

We bekijken sorteeralgoritmen gebaseerd op het doen van vergelijkingen van de vorm $A[i] < A[j]$.

Aannames:

- A bevat n verschillende waarden. (We gaan immers een ondergrens voor de worst case bepalen.)
- het sorteeralgoritme stopt zodra de sortering (onderlinge ordening) gevonden is.

Zo'n algoritme correspondeert (voor elke n) met een beslissingsboom die de series vergelijkingen representeert die het algoritme uitvoert voor elke mogelijke invoer.

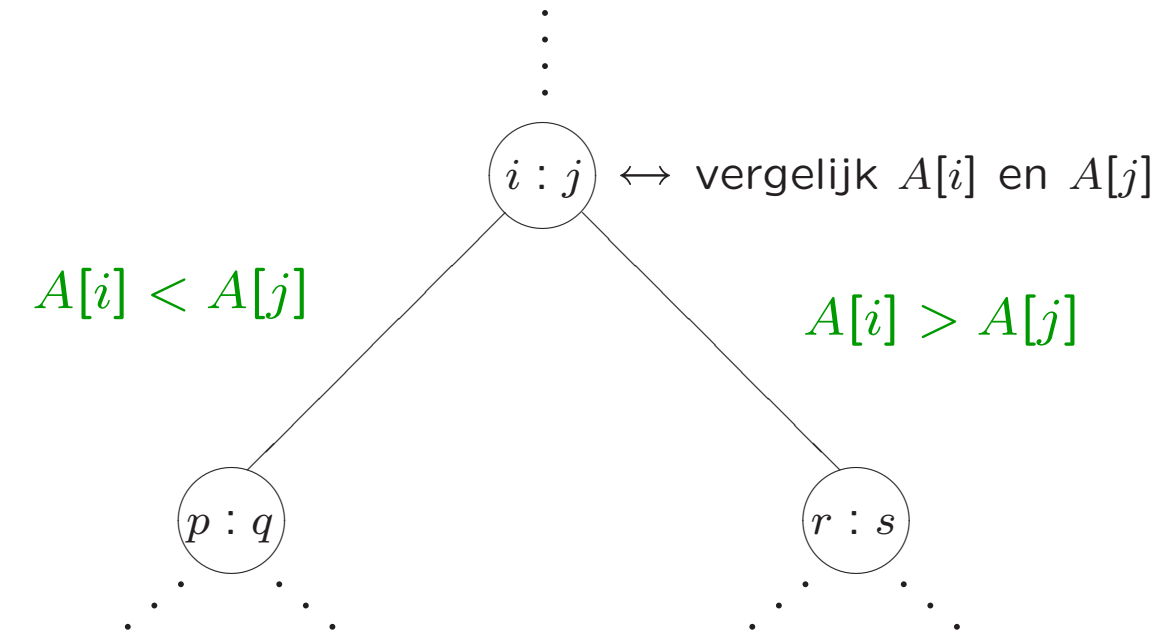
algoritme gebaseerd op het doen van arrayvergelijkingen $A[i] < A[j]$

$$\Downarrow$$

beslissingsboom*: binaire boom waarin de interne knopen corresponderen met arrayvergelijkingen en de bladeren/externe knopen met het eindresultaat†; een pad vanaf de wortel naar een blad correspondeert met een executie van het algoritme

*alle $A[i]$ zijn verschillend

†eindresultaat = de gevonden sorteringen/ordeningen (in dit geval)



Beslissingsboom voor algoritmen gebaseerd op arrayvergelijkingen

1. - alleen de onderlinge volgorde van de array-elementen is van belang; niet de waarde
   - het rijtje $1, 5, 3, 4, 2$ wordt bijvoorbeeld precies zo behandeld door het sorteeralgoritme als het rijtje $3, 15, 8, 11, 6$
   - ze volgen dan ook precies hetzelfde pad in de beslissingsboom
   - er zijn in essentie $n!$ mogelijke te onderscheiden invoeren

2. - sorteren = vind de oplopende ordening
   - er zijn dus $n!$ verschillende eindantwoorden (=ordeningen) mogelijk
   - een sorteeralgoritme moet die allemaal kunnen bereiken
   - de bijbehorende beslissingsboom moet dus minstens $n!$ bladeren hebben

3. conclusie uit het voorgaande: een beslissingsboom corresponderend met een sorteeralgoritme heeft precies $n!$ bladeren (n = aantal array-elementen)

Stelling: Het aantal vergelijkingen in de worst case is voor elk algoritme dat sorteert middels arrayvergelijkingen ten minste $\lceil \lg n! \rceil$ (dus $\Omega(n \lg n)$).

Stelling: Het aantal vergelijkingen in de average case is voor elk algoritme dat sorteert middels arrayvergelijkingen $\Omega(n \lg n)$.
Dit onder de aanname dat alle $n!$ mogelijke volgordes als invoerrijtje even waarschijnlijk zijn.