

datastructuren en algoritmen
2011 09 29
college 8

- **datastructuren: lineair en hierarchisch**
stacks, queues, vectoren, lijsten, rijtjes, priority queues
bomen, binaire bomen, binaire zoekbomen, AVL-bomen, heaps
- **sorteer-algoritmen**
heap-sort, selection sort, insertion sort, merge sort, quick sort,
quick select
- **hashing**

schema

- recap
- verdeel en heers
- maximum-subarray probleem
- integer vermenigvuldiging
- matrix vermenigvuldiging
- geld wisselen
- greedy algoritmen
- materiaal

verdeel en heers



- **divide:**
verdeel probleem P in deelproblemen
- **recur:**
los met recursie deelproblemen op
- **conquer:**
voeg oplossingen van deelproblemen samen
tot oplossing van P

verdeel en heers: voorbeelden al gezien

- vierkantjes betegelen
- binary search
- merge sort

oplossen van recurrente betrekkingen

- substitutie methode
- recursie-boom
- master method

tijdscomplexiteit: vaak met recurrente betrekkingen

voorbeeld: recurrente betrekking voor merge sort

$$T(n) = \begin{cases} 1 & \text{als } n = 1 \\ 2T(\frac{n}{2}) + n & \text{als } n > 1 \end{cases}$$

eigenlijk: met floor en ceiling, constante factoren, levert Θ

substitutie methode

- 'gok' een oplossing $S(n)$ voor $T(n)$
- laat zien $T(n) \leq S(n)$
geeft grote-Oh
- wij doen meestal: zie patroon opdoemen, vul basisgeval in

recursie-boom

expandeer, bekijk werk per laagje, en diepte boom

maximum-subarray probleem

gegeven een array met integers

geef een niet-leeg, (aaneensluitend), subarray met grootste som

- **toepassing:**
koop en verkoop met maximale winst
niet perse te komen van goedkoopste aankoop of duurste verkoop
- **alleen interessant als ook negatieve integers**
- **brute force:** in $\Theta(n^2)$

master method

boek Theorem 5.6

wordt niet expliciet gebruikt in college of werkcollege

aanpak: met verdeel en heers?

hoe ziet een oplossing eruit?

subarray $A[i \dots j]$ van $A[low \dots high]$

- links,
in $A[low \dots mid]$ oftewel $low \leq i < j \leq mid$
met recursie
- rechts,
in $A[mid \dots high]$ oftewel $mid \leq i < j \leq high$
met recursie
- midden,
met $low \leq i \leq mid < j \leq high$
met hulpprocedure

vind max-subarray in het midden

Algorithm middenMaxSubArray($A, low, mid, high$):

$leftsum := -\infty$

$sum := 0$

for $i := mid$ **downto** low **do**

$sum := sum + A[i]$

if $sum > leftsum$ **then**

$leftsum := sum$

$maxleft := i$

$rightsum := -\infty$

$sum := 0$

for $j := mid + 1$ **to** $high$ **do**

$sum := sum + A[j]$

if $sum > rightsum$ **then**

$rightsum := sum$

$maxright := j$

return ($maxleft, maxright, leftsum + rightsum$)

schema

- recap
- verdeel en heers
- maximum-subarray probleem
- integer vermenigvuldiging
- matrix vermenigvuldiging
- geld wisselen
- greedy algoritmen
- materiaal

tijdscomplexiteit

van hulprocedure:

in $\mathcal{O}(n)$, zelfs $\Theta(n)$

van totaal:

$$T(n) = \begin{cases} 1 & \text{als } n = 1 \\ 2T(\frac{n}{2}) + n & \text{als } n > 1 \end{cases}$$

dus in $\mathcal{O}(n \log(n))$

dus asymptotisch beter dan brute-force

integer vermenigvuldiging

- vermenigvuldigen van grote integers
in binaire representatie van zeg 1024 bits
- waarom zou je?
bijv voor public-key cryptografische toepassingen

integer arithmetiek

gegeven: twee n -bit integers A en B

- plus in $\mathcal{O}(n)$
- min in $\mathcal{O}(n)$
- keer in $\mathcal{O}(n^2)$
kan het beter?

integer vermenigvuldiging

- dan berekenen we $A \cdot B$:

$$\begin{aligned} A \cdot B &= (A_h \cdot 2^{\frac{n}{2}} + A_l) \cdot (B_h \cdot 2^{\frac{n}{2}} + B_l) \\ &= A_h \cdot B_h \cdot 2^n + \\ &\quad A_h \cdot B_l \cdot 2^{\frac{n}{2}} + \\ &\quad A_l \cdot B_h \cdot 2^{\frac{n}{2}} + \\ &\quad A_l \cdot B_l \end{aligned}$$

- in het voorbeeld:

$$\begin{aligned} A \cdot B &= 10110101 \cdot 10101101 \\ &= (1011) \cdot (1010) \cdot 2^8 + \\ &\quad (1011) \cdot (1101) \cdot 2^4 + \\ &\quad (0101) \cdot (1010) \cdot 2^4 + \\ &\quad (0101) \cdot (1101) \\ &= \dots \end{aligned}$$

integer vermenigvuldiging

we nemen aan: A en B bestaan uit n , een 2-macht, bits

- splits A en B in higher-order bits en lower-order bits
 $A = A_h \cdot 2^{\frac{n}{2}} + A_l$
 $B = B_h \cdot 2^{\frac{n}{2}} + B_l$
- voorbeeld:
 $A = 10110101 = (1011) \cdot 2^4 + 0101$
 $B = 10101101 = (1010) \cdot 2^4 + 1101$
- vermenigvuldigen met 2^p van binair getal
is shift left van p stappen
is in $\mathcal{O}(n)$ met n aantal bits van getal

integer vermenigvuldiging: tijdscomplexiteit

- recurrente betrekking:
 $T(n) = 4 \cdot T(\frac{n}{2}) + n$
- worst-case tijdscomplexiteit:
 $\mathcal{O}(n^2)$
- nog niet beter dan naieve aanpak

integer vermenigvuldiging: betere methode

- andere haakjes:

$$\begin{aligned} A \cdot B &= A_h \cdot B_h \cdot 2^n + \\ &\quad (A_h \cdot B_l + A_l \cdot B_h) \cdot 2^{\frac{n}{2}} + \\ &\quad A_l \cdot B_l \\ &= A_h \cdot B_h \cdot 2^n + \\ &\quad ((A_h - A_l) \cdot (B_l - B_h) + A_h \cdot B_h + A_l \cdot B_l) \cdot 2^{\frac{n}{2}} + \\ &\quad A_l \cdot B_l \end{aligned}$$

- in het voorbeeld:

$$\begin{aligned} A \cdot B &= 10110101 \cdot 10101101 \\ &= (1011) \cdot (1010) \cdot 2^3 + \\ &\quad (((1011) - (0101)) \cdot ((1101) - (1010)) + \\ &\quad (1011) \cdot (1010) + (0101) \cdot (1101)) \cdot 2^2 + \\ &\quad (0101) \cdot (1101) \\ &= \dots \end{aligned}$$

integer vermenigvuldigen: state of the art

- een eerste verdeel-en-heers poging levert geen verbetering
- andere verdeel-en-heers (Karatsuba) levert wel verbetering
- nu: nèt geen $\mathcal{O}(n \log n)$

integer vermenigvuldiging: nieuwe tijdscomplexiteit

- recurrente betrekking:

$$T(n) = 3 \cdot T\left(\frac{n}{2}\right) + n$$

- worst-case tijdscomplexiteit:

$$\mathcal{O}(n^{\log 3}) = n^{1.585}$$

- nu wel beter dan naieve aanpak

schema

- recap
- verdeel en heers
- maximum-subarray probleem
- integer vermenigvuldiging
- matrix vermenigvuldiging
- geld wisselen
- greedy algoritmen
- materiaal

matrix vermenigvuldiging

- veel nodig bijv voor lineaire transformaties in visualisatie
- vermenigvuldiging twee $n \times n$ matrices $Z = XY$:

$$Z[i, j] = \sum_{k=0}^{n-1} X[i, k] \cdot Y[k, j]$$

- n^2 inproducten van n vermenigvuldigingen dus $\mathcal{O}(n^3)$
- kan het beter?

matrix vermenigvuldiging: Strassen

- door ingenieus inzicht slechts 7 vermenigvuldigingen nodig
- tijdscomplexiteit:
 $T(n) = 7 \cdot T(\frac{n}{2})$ levert $\mathcal{O}(n^{\log 7})$ oftewel $\mathcal{O}(n^{2.808})$

matrix vermenigvuldiging: verdeel en heers

- neem aan: n is 2-macht
- schrijf $Z = XY$ als:

$$\begin{pmatrix} I & J \\ K & L \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

- dan hebben we:

$$\begin{aligned} I &= AE + BG \\ J &= AF + BH \\ K &= CE + DG \\ L &= CF + DH \end{aligned}$$

- tijdscomplexiteit?
 $T(n) = 8 \cdot T(\frac{n}{2}) + n^2$ levert nog steeds $\mathcal{O}(n^3)$

matrix vermenigvuldiging: state of the art

- een eerste verdeel-en-heers poging levert geen verbetering
- een andere (verassende) levert wel verbetering
- nu: een nog geavanceerder algoritme levert $\mathcal{O}(n^{2.376})$

schema

- recap
- verdeel en heers
- maximum-subarray probleem
- integer vermenigvuldiging
- matrix vermenigvuldiging
- geld wisselen
- greedy algoritmen
- materiaal

geld wisselen: altijd goede methode?

- **kies steeds grootste munt/briefje**
is goede methode voor euro-munten
- **munten 1, 3, 4 en maak 6**
dan werkt die methode niet
levert 3 munten in plaats van 2
- **in het algemeen:**
hangt van het munten-systeem af of het werkt
- **puzzel: voor welke munten-systemen werkt t?**

geld wisselen

- **probleem:**
maak bedrag N met zo min mogelijk muntjes/briefjes
- **voorbeeld:**
maak bedrag van 4,35 euro
- **in het algemeen:**
vul steeds aan met zo groot mogelijke eenheid
dit levert bedrag met zo min mogelijk munten/briefjes

schema

- recap
- verdeel en heers
- maximum-subarray probleem
- integer vermenigvuldiging
- matrix vermenigvuldiging
- geld wisselen
- greedy algoritmen
- materiaal

gulzigheid: terzijde

een van de 'zeven hoofdzonden'

- Gula (onmatigheid - gulzigheid - vraatzucht)

plaatje Hieronymus Bosch, Allegorie op de Gulzigheid, 1495



greedy choice property

- **probleem heeft greedy choice property**
als een greedy algoritme tot optimale oplossing leidt
oftwel: lokaal beste stap geeft globaal beste oplossing
- **voorbeelden problemen met greedy choice property:**
geld wisselen met euro-munten
maak grootste binaire getal
- **voorbeelden problemen zonder greedy choice property:**
geld wisselen met 1, 3, 4
ga van A naar C via $\{B_1, \dots, B_n\}$

greedy algoritmen

- **optimalisatie probleem**
er zijn verschillende configuraties
één of meer daarvan vinden we het beste
- **oplossing:**
exhaustive search, maar dat kan misschien beter?
- **greedy algoritmen: idee**
kies in elke stap de beste mogelijkheid

schema

- recap
- verdeel en heers
- maximum-subarray probleem
- integer vermenigvuldiging
- matrix vermenigvuldiging
- geld wisselen
- greedy algoritmen
- materiaal

materiaal

- boek 5.2, begin van 5.1

extra materiaal

- [wiki Karatsuba's algoritme voor integer vermenigvuldiging](#)
- [wiki matrix vermenigvuldiging](#)
- [wiki Strassen's algoritme voor matrix vermenigvuldiging](#)