Departement Wiskunde, Faculteit Bètawetenschappen, UU.
In elektronische vorm beschikbaar gemaakt door de TBC van A–Eskwadraat.
Het college INFODS werd in 2009-2010 gegeven door Dr. H.L. Bodlaender.

1e deeltentamen Datastructuren (INFODS)
02-06-2010

Bij verschillende opgaven wordt om een algoritme gevraagd. Probeer hierbij duidelijke antwoorden te geven. Pseudocode met een toelichting in tekst is daarbij een goede optie. Als om een tijdgrens wordt gevraagd, geef dan een 'zo goed mogelijke grens', waarbij steeds naar keuze van de O-notatie of de Θ-notatie gebruikt mag worden.

Opgave 1. O, Ω en $\Theta - I$ (2pt)

Zeg van elk van e volgende beweringen of ze waar zijn of onwaar. Toelichting is *niet* nodig.

a) $2^n \cdot n = O(3^n)$

b) $3n^3 \log n^3 + 4n^3 = O(n^3)$

c) $3n^3 \log n^3 + 4n^3 = O(n^3 \log n)$

d) $3n^3 \log n^3 + 4n^3 = O(n^4)$

e) $2n\sqrt{n} = \Omega(n \log n)$

f) $2n \log n = \Omega(n\sqrt{n}$

g) $5n^2 + 2n \log n + n = \Theta(n^2)$

Opgave 2. Hoeveel zijn er groter? (2pt)

Bij beide onderdelen moet het antwoord uit 2 delen bestaan:

1. een tijdgrens (in O-notatie of Θ-notatie, naar keuze)
2. een algoritme (beschreven in woorden of in pseudocode)

a) Stel we hebben een array van n verschillende integers $A\,[1 \cdots n]$, en een integer x. Hoeveel tijd kost het om te bepalen hoeveel elementen in A minstens zo groot zijn als x?

b) Stel nu da tA gesorteerd is. De vraag is nu dus: stel we hebben een gesorteerde array van n verschillende integers $A\,[1 \cdots n]$, en een integer x. Hoeveel tijd kost het om te bepalen hoeveel elementen in A minstens zo groot zijn als x? Licht je antwoord toe.

Opgave 3. Analyse van algoritmen (1.25pt)

Maak een zo goed mogelijke tijd-analyse (worst-case) van algoritme **aaa**, in termen van O van een functie in n. Verklaar je antwoord. Je mag aannemen dat de berekening van $\sqrt{n}$ in $O(1)$ tijd kan.

Algoritme **aaa**(C)
Input: 2-dimensionaal $n \times n$ array van integers $C\,[1 \cdots n, 1 \cdots n]$.
Output: 2-dimensionaal $n \times n$ array van integers $A\,[1 \cdots n, 1 \cdots n]$ of foutmelding.
d=1; `while` $d < n$ `do`
    `for` $i = 1$ `to` n `do`
        `for` $j = 1$ `to` n^2 `do`
            $A\,[i, j] = C\,[i, j]$;

```
                    for k = 1 to ⌊√n⌋ do
                        A[i,j] = max{A[i,j], A[i,k] + A[k,j]};
                    enddo;
                    for k = 1 to n do
                        A[i,j] = min{A[i,j], A[i,k] + A[k,j]};
                    for k = 1 to n do
                    enddo;
                enddo;
            enddo;
            d = d * 2;
    enddo;
    for i = 1 to n do
            if A[i,i] < 0
            then return "foutmelding";
            endif
    enddo;
    return A
```

Je antwoord bevat dus zowel de tijdgrens als uitleg hoe je aan die tijdgrens gekomen bent.

Opgave 4. Analyse van algoritmen II (1.25pt)

Beschouw het volgende algoritme:

```
Algoritme silly(A)
Input: array A[1 ··· n] van integers
if n > 3
then int b = ⌈n/3⌉
        Maak een array B[1 ··· b]
        for k = 1 to b do
            B[k] = A[k]
        enddo
        for k = 1 to b do
            A[k] = B[b − k + 1]
        for k = 1 to b do
        enddo
        silly(B)
        enddo
        for k = 1 to b do
            B[k] = A[k + b]
        for k = 1 to b do
        enddo
        silly(B)
        for k = 1 to b do
            A[b + k] = B[b − k + 1]
        enddo
endif
return A
```

a) Stel een recurrente betrekking op die de tijd van dit algoritme uitdrukt als functie van n, evt. met behulp van Θ-notatie.

b) Hoeveel tijd gebruikt dit algoritme, in Θ-notatie.

Als vraag a) niet is gelukt mag je, in plaats van je antwoord op vraag a), voor een deelscore bij deze vraag, de volgende recurrente betrekking gebruiken:

$$T(n) = 4T(\lceil n/5 \rceil) + n$$

Opgave 5. Heaps (1.75pt)

Op de Universiteit van Harderwijk wordt het vak *Sorteren en Classificeren* gegeven. Dit vak heeft ook een practicum, waarbij de studenten een heap moeten uitprogrammeren. De practicumleiding wil automatisch testen of heaps, gemaakt door de code van studenten, inderdaad maxheaps zijn. Daarom moet het volgende probleem worden opgelost. Deze opgave vraagt om een oplossing die lineaire tijd kost.
Stel we hebben een array van n integers $A[1 \cdots n]$. Geef een algoritme dat in $O(n)$ tijd test of A aan de maxheap-eigenschap voldoet.
Je mag hierbij NIET gebruik makien van aanroepen naar subroutines die in het college behandeld zijn. Je hoeft alleen het algoritme te geven, en de tijdgrens niet te bewijzen.

Opgave 6. Ontwerp van algoritmen (1.75pt)

Geef een algoritme dat in $O(n)$ tijd het volgende probleem oplost. Het algoritme mag in pseudocode genoteerd zijn. Leg uit waarom het algoritme $O(n)$ tijd gebruikt.
Gegeven is een array $A[1 \cdots n]$ met n reeële getallen tussen de 0 en 1. (Je mag aannemen dat ze **double**'s zijn in Java.) We weten dus dat

$$\forall i : 1 \leq i \leq n : 0 \leq A[i] \leq 1$$

Bepaal of er in A twee elementen zijn die hooguit $\frac{1}{2}n$ van elkaar verschillen, d.w.z., test of de volgende bewering waar is:

$$\exists i, j : 1 \leq i \leq n \wedge 1 \leq j \leq n \wedge i \neq j \wedge |A[i] - A[j]| \leq \frac{1}{2n}$$

Je algoritme moet als returnwaarde **true** of **false** hebben: **true** als de test hierboven geldt voor A en **false** anders.

Theorem 4.1 (Master theorem)
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n),$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. $\square$