1e deeltentamen Datastructuren 2010/2011

U heeft twee uur voor dit deeltentamen. 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 je steeds naar keuze van de O-notatie of de Θ-notatie gebruik mag maken.

Schrijf leesbaar en netjes. Zet uw mobiel uit voor aanvang van het tentamen. Veel succes!

1. O, Ω, en Θ - I

(2 punten.) **Zeg van elk van de volgende beweringen of ze waar zijn of onwaar.** U hoeft uw antwoord niet toe te lichten.

1. $5n^3 \log n^3 + 4n^3 = \Omega(n^3 \log n)$: WAAR of ONWAAR
2. $5n^3 \log n^3 + 4n^3 = O(n^3)$: WAAR of ONWAAR
3. $\frac{\log n}{2} + 6 = \Theta(\log n)$: WAAR of ONWAAR
4. $\frac{\log n}{2} + 6 = \Omega(1)$: WAAR of ONWAAR
5. $2^n = O(n^2)$: WAAR of ONWAAR
6. $2^n = \Omega(n^2)$: WAAR of ONWAAR
7. $n^2 = O(2^n)$: WAAR of ONWAAR
8. $n^2 = \Omega(2^n)$: WAAR of ONWAAR

2. Analyse van Algoritmen

(2 punten) **Maak een zo goed mogelijke tijd-analyse** (worst-case) van algoritme **Toets**, in termen van grote O of Θ (naar keuze) van een functie in n. Uw beantwoording moet ook uitleg bevatten hoe u aan het antwoord komt. We nemen aan dat operaties op integers elk $O(1)$ tijd kosten.

```
Algoritme Toets(int n)
    int x = 0;
    for i = 1 to n do
        int y = 0;
        while y < n do
            x = x + 1;
            y = y + i;
    return x;
```

Uw antwoord bevat dus zowel de tijdgrens als uitleg hoe u tot dit antwoord komt.

3. Analyse van Algoritmen -II

(2 punten) Beschouw het volgende algoritme **Kluts**.

```
Algorithm Kluts(integer array A[], int onder, int boven)
    if (boven − onder) ≤ 2 then
        x = A[boven];
        A[boven] = A[onder];
        A[onder] = x;
    else
        z = ⌊(boven + onder)/2⌋ ;
        Kluts(A, onder, z);
        Kluts(A, z + 1, boven);
        for i = onder to boven - 1 do
            x = A[i];
            A[i + 1] = A[i];
            A[i] = x;
        Kluts(A, z + 1, boven);
        Kluts(A, onder, z);
```

(i) Stel een recurrente betrekking op die de tijd van dit algoritme uitdrukt als functie van n, evt. met behulp van Θ-notatie. We nemen aan dat operaties op integers elk $O(1)$ tijd kosten.

(ii) Hoeveel tijd gebruikt dit algoritme **Kluts**, in Θ-notatie.

Als onderdeel (i) niet is gelukt mag u, in plaats van uw antwoord op onderdeel (i), voor een deelscore bij deze vraag, de volgende recurrente betrekking gebruiken.

$$T(n) = 4T(\lceil n/8 \rceil) + \log n$$

2

4. Heaps

(2 punten) Stel we hebben een **min-heap** S. We nemen aan dat S geimplementeerd is met de (behandelde) implementatie in een array.

Stel x is een of andere key, niet noodzakelijk een element van S. **Geef een algoritme, dat alle keys in S afdrukt die kleiner dan of gelijk zijn aan x.** Uw algoritme moet in $O(r)$ tijd werken, als het aantal antwoorden $r \geq 1$ is.

Het is mogelijk dat waardes vaker dan één keer voorkomen.

Bij uw antwoord mag u de beschikbare operaties op een min-heap gebruiken, zonder voor die operaties de pseudocode (opnieuw) te geven.

Probeer door compact te formuleren voldoende details te geven, zo mogelijk in pseudocode. Maar uw algoritme mag niet meer dan 1 kantje A4 beslaan. Het bewijs dat uw algoritme $O(r)$ tijd gebruikt hoeft u **niet** te geven.

5. Inductie

(2 punten) Een 2-3-boom is een boom met een wortel, die voldoet aan de volgende twee eigenschappen:

- Elke knoop heeft nul, twee of drie kinderen.
- Alle bladeren hebben dezelfde diepte.

Bewijs met inductie:

Stel T is een 2-3-boom met diepte r en s bladeren. Dan

$$2^r \leq s \leq 3^r$$

Stelling 1 *Stel $a \geq 1$ en $b \geq 1$ zijn constantes, $f(n)$ is een functie en $T(n)$ is een functie van de niet-negatieve (of positieve) integers, gedefinieerd met*

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

waarbij n/b zowel omhoog als omlaag kan afgerond worden.

1. *Als $f(n) = O(n^{\log_b a - \epsilon})$ voor een constante $\epsilon > 0$, dan geldt $T(n) = \Theta(n^{\log_b a})$.*
2. *Als $f(n) = \Theta(n^{\log_b a})$, dan geldt $T(n) = \Theta(n^{\log_b a} \log n)$.*
3. *Als $f(n) = \Omega(n^{\log_b a + \epsilon})$ voor een constante $\epsilon > 0$ en als $af(n/b) \leq cf(n)$ voor een constante $c < 1$ en alle voldoende grote n, dan geldt $T(n) = \Theta(f(n))$.*