Tentamen Complexiteit
Dinsdag 31 mei 2005, 14.00-17.00 uur

Geef een duidelijke toelichting bij al je antwoorden.

Opgave 1. (20 punten)
a. (5 punten)
Leg uit wat een beslissingsboom is (voor algoritmen gebaseerd op arrayvergelijkingen): wat stellen de interne knopen en de bladeren voor, een pad van de wortel naar een blad, linker- en rechtersubboom van een knoop, de hoogte van de boom, etcetera.
b. (5 punten)
Bewijs met behulp van een beslissingsboomargument dat *elk* algoritme dat van n verschillende waarden de grootste *en* de op een na grootste bepaalt (m.b.v. arrayvergelijkingen) ten minste $\Omega(\lg n)$ vergelijkingen moet doen in de worst case. Gebruik hierbij dat voor een binaire boom met hoogte h en b bladeren geldt dat $h \geq \lceil \lg b \rceil$.
c. (8 punten)
De *toernooimethode* vindt het grootste en het op een na grootste element uit n stuks met $n + \lceil \lg n \rceil - 2$ vergelijkingen. Neem voor het gemak aan dat $n = 2^k$.
(i) Beschrijf het algoritme en maak duidelijk dat de methode werkt (dus inderdaad de grootste en de op een na grootste vindt).
(ii) Laat zien dat de toernooimethode inderdaad $n + \lceil \lg n \rceil - 2$ vergelijkingen doet.
d. (2 punten)
Volgt uit **b.** dat de toernooimethode *niet* optimaal is? Zo ja waarom, zo nee waarom niet.

Opgave 2. (10 punten)
Bewijs dat het aantal vergelijkingen dat *Shellsort* met sprongafstanden (= stapgroottes = increments) $n/7$, $n/49$, $n/343$, $\ldots$, 49, 7, 1 in de worst case doet $\Omega(n^2)$ is. Neem aan dat n een macht van 7 is.

Opgave 3. (15 punten)
Bekijk het volgende *recursieve* algoritme voor het oplopend sorteren van een rij van n getallen met $n = 3^k$ (k geheel, ≥ 0).
Als $n \geq 3$ sorteren we eerst recursief de eerste $n/3$ elementen, vervolgens de middelste $n/3$ en tenslotte de resterende $n/3$. Daarna voegen we de drie gesorteerde deelrijen samen, analoog aan de methode Merge. Steeds wordt de kleinste van de drie voorste elementen van de deelrijtjes bepaald en doorgeschoven naar een hulparray. Dit kost dus in elke stap 2 vergelijkingen (totdat één van de deelrijtjes leeg is, dan nog 1 vergelijking). Als er nog maar één deelrijtje over is kan dit uiteraard zonder verdere vergelijkingen naar het hulparray worden gekopieerd. Aan het eind van dit proces zijn alle deelrijtjes leeg en bevat het hulparray de oplopend gesorteerde rij van n elementen. Noem het aantal vergelijkingen dat dit algoritme in de worst case doet $S(n)$.
Opmerking. De in **b.** gevraagde oplossing kun je voor 5 punten vragen aan de surveillant; je moet die dan nog wel verifiëren met inductie.

a. (6 punten)
Leg duidelijk uit waarom $S(n)$ voldoet aan de volgende recurrente betrekking:

$$S(n) = \begin{cases} 0 & n = 1 \\ 3\,S(\frac{n}{3}) + 2n - 3 & n > 1, n = 3^k \end{cases}$$

b. (9 punten)
Los de recurrente betrekking uit **a.** op door deze herhaald in zichzelf in te vullen en bewijs met behulp van volledige inductie dat de aldus gevonden oplossing inderdaad voldoet. Schrijf S als functie van n. *Hint*: gebruik dat $\sum_{i=0}^{l} 3^i = \frac{1}{2}(3^{l+1} - 1)$.

Opgave 4. (25 punten)
We hebben een rij met n (≥ 1) positieve gehele getallen, opgeslagen in een array A met n elementen. Tevens is een geheel getal $t \geq 1$ gegeven. We willen nu indices $1 \leq \mathit{ene} \leq \mathit{andere} \leq n$ vinden waarvoor geldt dat $A[\mathit{ene}] + \cdots + A[\mathit{andere}] = t$. Als zulke indices niet bestaan krijgen *ene* en *andere* de waarde 0.
In onderstaand algoritme wordt voor elke i de som $A[i] + \cdots + A[j]$ berekend voor $j \geq i$. Zodra deze som groter wordt dan t wordt j niet meer opgehoogd. Er wordt dan gecontroleerd of de som gelijk is aan t (en in dat geval worden de gevonden i en j bewaard) en verder gegaan met de volgende i. Het algoritme gaat als volgt:

```
(1)     i := 1; ene := 0; andere := 0;
(2)     while i ≤ n do
(3)         j := i; hoeveel := 0;
(4)         while j ≤ n and hoeveel < t do
(5)             hoeveel := hoeveel + A[j];
(6)             j := j + 1;
            od
            // nu j = n + 1 en/of hoeveel ≥ t
(7)         if hoeveel = t then
(8)             ene := i; andere := j − 1;
            fi
(9)         i := i + 1;
        od
```

a. (6 punten)
Toon aan dat het *optellen* van array-elementen (uit regel (5)) een goede basisoperatie is om de complexiteit van dit algoritme mee te beschrijven. Geef eerst kort in woorden aan waarom dit een goede basisoperatie zou zijn, en maak dit vervolgens hard door het aantal keer dat een regel wordt uitgevoerd af te schatten op het aantal keer dat (5) wordt gedaan. Laat hiertoe onder andere zien dat de optelling uit regel (5) ten minste n keer gebeurt.
b. (6 punten)
Hoeveel optellingen van array-elementen (regel (5) dus) worden er gedaan in het beste geval, voor algemene n? En voor wat voor invoerrijtjes A komt dat voor? Geef alle gevallen.
c. (6 punten)
Hoeveel optellingen van array-elementen worden er gedaan in het slechtste geval? En voor wat voor invoerrijtjes A komt dat voor? Beschrijf alle gevallen. Neem voor het gemak aan dat $t \geq n$.

Een efficiënter algoritme voor het probleem is het volgende:

```
(1)   ene := 0; andere := 0; hoeveel := 0;
(2)   i := 1; j := 0; klaar := False;
(3)   while i ≤ n and j ≤ n and not klaar do
          // hier is hoeveel = A[i] + ··· + A[j]
(4)       if hoeveel = t then
(5)           ene := i; andere := j; klaar := True;
              // stop zodra indices gevonden
          else
(6)           if hoeveel < t then
(7)               j := j + 1;
(8)               if j < n then
(9)                   hoeveel := hoeveel + A[j];
                  fi
              else
(10)              hoeveel := hoeveel − A[i];
(11)              i := i + 1;
              fi
          fi
      od
```

d. (7 punten)
Om de analyse te vergemakkelijken veronderstellen we weer dat $t \geq n$. Wat is het aantal optellingen (regel (9) en (10)) samen) dat dit algoritme doet in de worst case? Geef hierbij een invoerrijtje waarvoor dit voorkomt.

Opgave 5. (30 punten)
We bekijken de volgende twee beslissingsproblemen:

Hamiltoncircuitprobleem (HC): Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$.
Vraag: Heeft $\mathcal{G}$ een Hamiltonkring?
Een Hamiltonkring in een ongerichte graaf is een kring die *elke* knoop precies *één* keer bevat.

Handelsreizigersprobleem met driehoeksongelijkheid (TSP+$\triangle$): Gegeven een *volledige*, ongerichte graaf $\mathcal{G} = (V, E)$ met gewichten op de takken, en een geheel getal $k \geq 0$. De gewichten $c(u, v)$ op de takken voldoen aan de driehoeksongelijkheid, dat wil zeggen: voor alle knopen u, v en w moet gelden: $c(u, w) \leq c(u, v) + c(v, w)$.

Vraag: bestaat er in $\mathcal{G}$ een Hamiltonkring met totaalgewicht $\leq k$?

a. (10 punten)
Toon aan dat TSP+$\triangle \in \mathcal{NP}$ door te laten zien dat er een *niet-deterministisch polynomiaal* algoritme voor TSP+$\triangle$ bestaat.

We construeren een transformatie T van instanties van HC naar instanties van TSP+$\triangle$. Een ongerichte graaf $\mathcal{G}$ wordt afgebeeld op de volledige ongerichte graaf $\mathcal{G}'$ met $\mathcal{G}' =$

(V', E'), waarbij $V' = V$ en $E' = \{(u, v) : u, v \in V, u \neq v\}$ (dus er is een tak tussen elk tweetal knopen), en een gewichtsfunctie c met

$$c(u, v) = \begin{cases} 1 & \text{als} \quad (u, v) \in E \\ 2 & \text{als} \quad (u, v) \notin E \end{cases}$$

Merk op dat de gewichtsfunctie c aan de driehoeksongelijkheid voldoet, aangezien $c(u, w) \leq c(u, v) + c(v, w)$ steeds geldt omdat $c(u, v) + c(v, w)$ altijd ≥ 2 is.
Definieer tenslotte $T(\mathcal{G}) = \langle \mathcal{G}', c, |V| \rangle$, met $|V|$ = het aantal knopen van $\mathcal{G}$.

b. (5 punten)
Laat zien dat $T(\mathcal{G})$ in polynomiaal begrensde tijd geconstrueerd kan worden.
c. (8 punten)
Toon aan dat geldt: $\mathcal{G}$ is een ja-instantie van HC $\Longleftrightarrow T(\mathcal{G})$ is een ja-instantie van TSP+$\triangle$.
Ofwel: $\mathcal{G}$ heeft een Hamiltonkring $\Longleftrightarrow \mathcal{G}'$ heeft een Hamiltonkring met totaalgewicht ten hoogste $|V|$.
Merk op: uit **b.** en **c.** volgt nu dat HC $\leq_P$ TSP+$\triangle$.
d. (2 punten)
Wanneer is een beslissingsprobleem P NP-hard? En wanneer NP-volledig? (Geef de definitie van NP-hard/ NP-volledigheid; de definitie van NP hoeft niet te worden gegeven.)
e. (5 punten)
Welke van de twee volgende beweringen is waar en welke niet (waarom/waarom niet)? Motiveer je antwoord en formuleer duidelijk gebruikte stellingen.
(i) Als gegeven is dat HC $\in \mathcal{NP}$ en TSP+$\triangle \in \mathcal{NPC}$, dan volgt uit **b.** en **c.** dat HC NP-volledig is.
(ii) Als gegeven is dat HC $\in \mathcal{NPC}$, dan volgt uit **a.**, **b.** en **c.** dat TSP+$\triangle$ NP-volledig is.

Veel succes!