Tentamen Complexiteit
Dinsdag 14 augustus 2007, 10.00-13.00 uur

Geef een *duidelijke* toelichting bij al je antwoorden.

Opgave 1. (25 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)
Gegeven een binaire boom met hoogte h en b bladeren. Bewijs dat geldt: $h \geq \lceil \lg b \rceil$.
c. (5 punten)
Bewijs met behulp van een beslissingsboomargument dat *elk* algoritme dat van n verschillende waarden de grootste en de op een na grootste waarde bepaalt m.b.v. arrayvergelijkingen, ten minste $\lceil 2 \cdot \lg(n-1) \rceil$ vergelijkingen moet doen in de worst case.
d. (5 punten)
We bekijken de methode van Horner voor polynoomevaluatie. Leg uit hoe de methode werkt (uitleg, geen algoritme!), en laat zien hoeveel $(+,-)$'s en hoeveel $(*,/)$'s de methode voor algemene n doet. Illustreer de werking met het volgende voorbeeldpolynoom: $3x^5 - 4x^4 + 8x^3 - 6x^2 + 7x + 4$.
e. (5 punten)
De volgende stelling geldt: een algoritme dat polynomen van graad n evalueert doet ten minste n $(+,-)$'s en n $(*,/)$'s. Het polynoom $p(x) = x^n + x^{n-1} + \cdots + x + 1$ van graad n kan echter geëvalueerd worden met 2 $(+,-)$'s en $n+1$ $(*,/)$'s.
(i) Geef aan op welke manier dit kan.
(ii) Is dit in tegenspraak met de genoemde stelling? Waarom (niet)?

Opgave 2. (20 punten)

Gegeven is een array met n verschillende getallen. Laat $n = 3^k$ en $n \geq 3$. We bekijken een recursief algoritme dat de grootste en de op een na grootste waarde uit het array bepaalt. De methode werkt als volgt. Voor $n > 3$ berekenen we *recursief* de grootste en de op een na grootste waarde uit de eerste $n/3$ elementen, en vervolgens doen we hetzelfde met de middelste $n/3$ arrayelementen, en met de laatste $n/3$ arrayelementen. Tenslotte bepalen we, door de 6 aldus gevonden waarden handig onderling te vergelijken, het grootste en het op een na grootste getal uit het hele array. Laat $W(n)$ het aantal arrayvergelijkingen (in de worst case) zijn dat dit algoritme doet.

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

$$W(n) = \begin{cases} 3 & n = 3 \\ 3 \cdot W(\frac{n}{3}) + 4 & n > 3,\ n = 3^k \end{cases}$$

Hierbij moeten dus ook de beginvoorwaarde $W(3) = 3$ en de factor 4 worden uitgelegd.

b. (10 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 (uitgedrukt in n) inderdaad voldoet.
Hint bij het uitrekenen:

$$\sum_{i=0}^{l-1} 3^i \quad = \quad \frac{1}{2} \cdot (3^l - 1)$$

c. (4 punten)
In **b.** heb het je het aantal vergelijkingen in de worst case berekend van de hierboven beschreven methode. Kun je, alleen op grond van het bewezene in opgave **1.c.**, concluderen dat deze methode niet optimaal is? Zo ja waarom, zo nee waarom niet?

Opgave 3. (30 punten)

Gegeven een array A ($A[1], A[2], \ldots, A[n]$), met $n \geq 1$, dat n *verschillende* getallen bevat. Onderstaand algoritme sorteert het array oplopend door alle posities van rechts naar links af te lopen en array-elementen op hun juiste plek neer te zetten. In regel (5) t/m (9) wordt de juiste plek voor de waarde $A[i]$ bepaald door het aantal elementen $\leq A[i]$ te tellen. In regel (10) wordt $A[i]$ daadwerkelijk op zijn juiste positie —aangegeven door *index*— gezet door deze te verwisselen met $A[index]$. Vervolgens (indien $index \neq i$) wordt de nieuwe $A[i]$ op dezelfde manier op zijn juiste positie geplaatst. Als echter $index = i$ in regel (11), dan hebben we $A[i]$ met zichzelf verwisseld en stond deze blijkbaar al op de juiste plek. We kunnen dan dus verdergaan met de volgende i.
Twee van de drie while-loops, namelijk de binnenste en de buitenste, stellen for-loops voor. Merk verder op dat een array-element dat eenmaal op de juiste positie staat daar nooit meer verdwijnt.

```
(1)   i := n;
(2)   while i ≥ 2 do
(3)       doorgaan := True;
(4)       while doorgaan do
(5)           index := 1; j := 1;
(6)           while j < i do
(7)               if A[j] < A[i] then
(8)                   index := index + 1;
                  fi
(9)               j := j + 1;
              od
(10)          wissel(A[index], A[i]);
(11)          if index = i then
(12)              doorgaan := False;
              fi
          od
(13)      i := i − 1;
      od
```

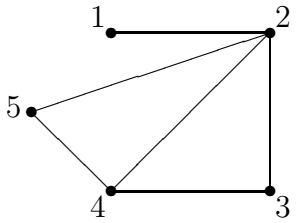
a. (3 punten)
Ga na hoe het algoritme werkt voor het array $5, 6, 1, 2, 3, 4$. Laat zien hoe het array eruit ziet elke keer nadat regel (10) is uitgevoerd.
b. (6 punten)
Toon aan dat het *vergelijken* van array-elementen in regel (7) een goede basisoperatie is als $n \geq 2$, door het aantal keer dat een regel wordt uitgevoerd af te schatten (in orde van grootte) op het aantal keer dat de test in regel (7) gebeurt. Gebruik hierbij dat deze test voor elke doorgang door de middelste while-lus ten minste één keer gedaan wordt vanwege $i \geq 2$, en overigens dus ook ≥ 1 keer per doorgang door de buitenste while-lus.
c. (7 punten)
Hoe vaak kan regel (10) maximaal uitgevoerd worden en waarom? Gebruik dit om een redelijke bovengrens te bepalen voor het aantal vergelijkingen (regel (7)) dat het algoritme maximaal kan doen.
d. (7 punten)
Hoeveel vergelijkingen tussen array-elementen (regel (7)) doet het algoritme in het beste geval (algemene n)? Voor wat voor invoerrijtjes komt dat voor? Leg duidelijk uit hoe je aan je antwoord komt.
e. (7 punten)
Hoeveel vergelijkingen tussen array-elementen doet het algoritme in het slechtste geval (algemene n)? Voor wat voor type invoerrijtjes wordt dit maximale aantal vergelijkingen gedaan? Geef een duidelijke toelichting.

Opgave 4. (25 punten)

We bekijken het volgende beslissingsprobleem:

Independent Set (IS): Gegeven een ongerichte graaf $G = (V, E)$ en een geheel getal K met $K > 0$.
Vraag: is er een onafhankelijke verzameling I ($I \subseteq V$) met $|I| = K$?
Een deelverzameling I van de knopen V heet *onafhankelijk* als geldt: voor alle $i, j \in I$ is er *geen* tak tussen i en j. Verder wordt met $|I|$ het aantal knopen van I bedoeld.

Voorbeeld : zij G als hieronder en $K = 3$, dan is $I = \{1, 3, 5\}$ een onafhankelijke knoopverzameling ter grootte K.



a. (10 punten)
Toon aan dat IS $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor IS te geven. Het algoritme heeft dus als invoer een ongerichte graaf $G = (V, E)$ en een geheel getal K met $K > 0$, en moet "ja" opleveren d.e.s.d.a. $< G, K >$ een ja-instantie is. Leg onder andere uit wat in elke fase van het algoritme gebeurt, en in het bijzonder wat in fase 2 wordt gecontroleerd en hoe.

b. (3 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.)
c. (6 punten)
Het is bekend dat IS $\in \mathcal{NPC}$. Gegeven is nu een beslissingsprobleem P. Geef van de volgende beweringen aan of ze waar zijn of niet (waarom/waarom niet)? Motiveer je antwoord en formuleer duidelijk gebruikte stellingen.
(i) Als gegeven is dat $P \in \mathcal{NP}$ en $P \leq_P$ IS, dan is P NP-volledig.
(ii) Als gegeven is dat $P \in \mathcal{NP}$ en IS $\leq_P P$, dan is P NP-volledig.
(iii) Uit de NP-volledigheid van IS volgt onmiddelijk dat $P \leq_P$ IS.
d. (6 punten)
We fixeren nu de invoerwaarde K voor het Independent Set probleem. Voor $K = 2$ krijgen we dan het beslissingsprobleem 2IS.

2IS: Gegeven een ongerichte graaf G. Bevat G een onafhankelijke knoopverzameling bestaande uit 2 knopen?

Geheel analoog definiëren we 3IS, 4IS, 5IS, ... Beantwoord nu de volgende twee vragen.
(i) Is 2IS NP-volledig? Leg uit waarom/waarom niet.
(ii) Zijn 3IS, 4IS, 5IS, ... NP-volledig of niet? Motiveer je antoord.

Veel succes!