Tentamen Complexiteit
Donderdag 4 juni 2009, 14.00-17.00 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. (6 punten)
Bewijs met behulp van een beslissingsboomargument: *elk* algoritme dat van n verschillende waarden de grootste, de een na grootste en de twee na grootste bepaalt (m.b.v. arrayvergelijkingen) moet ten minste $\lceil 3 \cdot \lg(n-2) \rceil$ vergelijkingen doen in de worst case.

Onderdeel **d.** staat los van de vorige drie vragen.
d. (9 punten)
Stel dat we een array A hebben dat n verschillende gehele getallen bevat, en waarvan gegeven is dat de waarden op de even posities allemaal kleiner zijn dan alle waarden op de oneven posities. Het aantal n kan zowel even als oneven zijn.
(i) Geef een algoritme dat in $n-2$ arrayvergelijkingen zowel het minimum als het maximum van deze n getallen oplevert.
(ii) Op college is aangetoond dat elk algoritme gebaseerd op arrayvergelijkingen dat het minimum en het maximum vindt van n elementen, ten minste $\lceil \frac{3n}{2} \rceil$ vergelijkingen moet doen in de worst case. Is dit in tegenspraak met het algoritme uit (i)? Motiveer je antwoord.

Opgave 2. (20 punten)
Gegeven is een array A met $n = 2^k$ $(k \geq 1)$ verschillende gehele getallen.
De bedoeling is het array zo te reorganiseren dat na afloop alle elementen op de even posities oplopend gesorteerd zijn, en alle elementen op de oneven posities aflopend. De rij $3, 4, 6, 8, 1, 7, 5, 2$ wordt dus gereorganiseerd tot $6, 2, 5, 4, 3, 7, 1, 8$. We doen dit als volgt. Voor $n > 2$ reorganiseren we *recursief* de eerste en tweede helft van A zoals hierboven bedoeld. Vervolgens gebruiken we Merge op de elementen op de even posities, en vervolgens op de elementen op de oneven posities.
Laat $W(n)$ het aantal arrayvergelijkingen zijn dat dit algoritme doet in de worst case.

a. (5 punten)
Leg uit waarom $W(n)$ voldoet aan de volgende recurrente betrekking:

$$W(n) = \begin{cases} 0 & n = 2 \\ 2W(\frac{n}{2}) + n - 2 & n \geq 4,\ n = 2^k \end{cases}$$

Leg ook uit hoe je aan de beginvoorwaarde $W(2) = 0$ en de term $n - 2$ komt.

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: $\sum_{i=1}^{l} 2^i \;=\; 2^{l+1} - 2$
c. (5 punten)
Het gereorganiseerde array kunnen we nu gemakkelijk (= lineair) oplopend sorteren.
(i) Leg uit hoe je dat kan doen en hoeveel vergelijkingen dit dan maximaal kost.
(ii) Denk je dat deze methode (eerst reorganiseren als boven en dan het algoritme uit (i)) beter dan, slechter dan of even goed is als Mergesort op het oorspronkelijke array?

Opgave 3. (30 punten)
Gegeven is een array A dat $n \geq 2$ gehele getallen $A[1]$, $A[2]$, ..., $A[n]$ bevat. We bekijken een algoritme dat bepaalt welke waarde het vaakst in A voorkomt. (Indien meerdere waardes even vaak voorkomen, dan geeft het algoritme er gewoon een.) Hiertoe lopen we het array van links naar rechts af, bepalen van elk getal hoe vaak dat voorkomt en onthouden de waarde die tot dusver het vaakst is aangetroffen. Om te zorgen dat we dezelfde getallen niet nog een keer bekijken gebruiken we een boolean hulparray `gehad` waarin we per index aangeven of het getal A[index] al geweest is. Bij aanvang zijn alle elementen van `gehad` op `False` geïnitialiseerd. Het algoritme gaat als volgt:

```
(1)   vaakst := 1; waarde := A[1];
(2)   j := 1;
(3)   while j ≤ n − 1 do
(4)       if not gehad[j] then
(5)           telA := 1; gehad[j] := True;
(6)           i := j + 1;
(7)           while i ≤ n do
(8)               if A[i] = A[j] then
(9)                   gehad[i] := True;
(10)                  telA := telA + 1;
(11)              fi
(12)              i := i + 1;
(13)          od
(14)          if telA > vaakst then
(15)              vaakst := telA; waarde := A[j];
(16)          fi
(17)      fi
(18)      j := j + 1;
(19)  od
```

a. (6 punten)
Toon aan dat het *vergelijken* van array-elementen (test in regel (8)) maatgevend is voor de complexiteit van het algoritme. Doe dit door het aantal keren dat een regel wordt gedaan in verband te brengen met het aantal keren dat de test in regel (8) wordt uitgevoerd.
Laat hiertoe onder andere zien dat de test in regel (8) ten minste $n - 1$ keer gebeurt.
b. (6 punten)
Hoeveel vergelijkingen tussen array-elementen worden er gedaan in het beste geval, voor

algemene n? En voor wat voor soort invoerrijtjes A komt dat voor? Geef *alle* gevallen en leg daarbij ook uit waarom dit alle gevallen zijn.
c. (6 punten)
Als **b.**, maar nu in het slechtste geval.
d. (6 punten)
Voeg in het algoritme tussen regel (7) en regel (8) de test **if not** `gehad`$[i]$ **then** toe. Beantwoord nu vraag **b.** (best case) voor het aangepaste algoritme. Wat is er veranderd?
e. (6 punten)
Hoeveel vergelijkingen (regel (8)) doet het aangepaste algoritme minder dan het oorspronkelijke algoritme op het rijtje $1, 2, \ldots, \frac{n}{2}, 1, 1, \ldots, 1$?

Opgave 4. (25 punten)
We bekijken het volgende beslissingsprobleem:

MAX2SAT: Gegeven is een logische formule ϕ in 2-CNF (dus een AND van clausules waarbij elke clausule de OR van *precies twee* verschillende literals is). Gegeven is verder een geheel getal $k > 0$.
Vraag: is er een waardering (waarheidstoekenning) van de in ϕ voorkomende variabelen $x_1, \ldots, x_n$, zodanig dat ten minste k clausules van ϕ worden waargemaakt?

Merk op: onder een literal verstaan we een logische variabele of zijn ontkenning (x_i of $\neg x_i$).

a. (10 punten)
Toon aan dat MAX2SAT $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor MAX2SAT te geven. Het algoritme heeft als invoer een logische formule ϕ in 2-CNF en een geheel getal $k > 0$. Het moet "ja" opleveren dan en slechts dan als $\langle \phi, k \rangle$ 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, en hoeveel stappen dat kost. Leg ook uit waarom je algoritme aan (&) voldoet en waarom het polynomiaal is.
b. (5 punten)
(i) Leg uit wat een polynomiale reductie is.
(ii) 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.)

Van college weten we dat 3SAT NP-volledig is. Bovendien is bekend dat 2SAT in $\mathcal{P}$ zit. Uit **a.** weten we dat MAX2SAT $\in \mathcal{NP}$. Beantwoord nu de volgende vragen en licht je antwoord toe.

c. (10 punten)
(i) Stel dat MAX2SAT $\leq_P$ 3SAT. Volgt nu dat MAX2SAT NP-volledig is?
(ii) Stel dat 3SAT $\leq_P$ MAX2SAT. Volgt nu dat MAX2SAT NP-volledig is?
(iii) Bestaat er een polynomiale reductie van 2SAT naar 3SAT? Waarom (niet)?
(iv) Bestaat er een polynomiale reductie van 3SAT naar 2SAT? Waarom (niet)?
In (iii)/(iv) wordt alleen naar het bestaan gevraagd. Je hoeft dus geen reductie te geven.

Veel succes!