Tentamen Complexiteit
Dinsdag 18 augustus 2009, 10.00-13.00 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: *elk* algoritme dat de mediaan (= middelste in grootte) van n verschillende waarden, opgeslagen in een array, bepaalt m.b.v. arrayvergelijkingen moet ten minste $\lceil \lg n \rceil$ vergelijkingen doen in de worst case.
c. (5 punten)
(i) Laat zien dat de mediaan van n verschillende waarden met maximaal $\Theta(n \lg n)$ arrayvergelijkingen *kan* worden gevonden. (ii) Betekent dit dat de ondergrens uit **b.** niet scherp is? Licht je antwoord toe.
d. (5 punten)
Laat zien hoe je de mediaan van 5 verschillende waarden a, b, c, d en e met maximaal 6 vergelijkingen kunt vinden.

Opgave 2. (15 punten)
Gegeven een array A met n verschillende waarden $A[1], A[2], \ldots, A[n]$. Een *inversie* is een paar $(A[i], A[j])$ met $i < j$ en $A[i] > A[j]$, m.a.w. een paar dat verkeerd om staat indien we oplopend willen sorteren.
a. (7 punten)
Bewijs nu het volgende: elk algoritme dat A oplopend sorteert met behulp van arrayvergelijkingen en dat na elke vergelijking ten hoogste één inversie opheft, moet in de worst case ten minste $\frac{1}{2}n(n-1)$ vergelijkingen doen.
b. (8 punten)
Zowel voor Bubblesort als voor Quicksort geldt dat ze ten minste $\frac{1}{2}n(n-1)$ arrayvergelijkingen moeten doen in de worst case.
Kan men dit voor Bubblesort op grond van de stelling uit **a.** concluderen? En voor Quicksort? Als het antwoord "nee" luidt moet je een ander, eenvoudig, argument geven dat het gestelde bewijst. Als het antwoord "ja" is moet je duidelijk aangeven waarom.

Opgave 3. (15 punten)
We bekijken het volgende *recursieve* algoritme voor het vinden van het maximum van een rij van n verschillende getallen met $n = 3^k$ (k geheel, ≥ 0). Als $n \geq 3$ bepalen we eerst recursief de grootste van de eerste $n/3$ elementen, vervolgens doen we hetzelfde voor de volgende $n/3$ elementen, en ten slotte voor de resterende $n/3$. Uit de drie gevonden maxima bepalen we dan het maximum van *alle* n elementen. Noem het aantal vergelijkingen dat dit algoritme doet $M(n)$.
a. (3 punten)
Leg uit waarom $M(n)$ voldoet aan de volgende recurrente betrekking:

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

Leg ook uit hoe je aan de beginvoorwaarde $M(1) = 0$ en de term 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 inductie dat de aldus gevonden oplossing (uitgedrukt in n) inderdaad voldoet. *Hint* bij het uitrekenen: $\sum_{i=0}^{\ell} 3^i = \frac{1}{2} \cdot (3^{\ell+1} - 1)$.
c. (2 punten)
Is dit algoritme optimaal voor wat betreft de worst case?

Opgave 4. (25 punten)
In deze opgave hebben we twee rijen met n (≥ 1) positieve gehele getallen (opgeslagen in arrays A en B met elk n elementen), waarvan we willen weten of de een een permutatie van de ander is. We laten hierbij toe dat waarden meer dan één keer voorkomen. Dat betekent dat A en B permutaties van elkaar zijn als elk getal precies even vaak in A als in B voorkomt. Voorbeeld: 3 8 3 3 7 is een permutatie van 7 3 3 8 3.
Om te bepalen of A en B permutaties van elkaar zijn lopen we het array A van links naar rechts af en bepalen van elk getal hoe vaak dat voorkomt. Vervolgens wordt in B gekeken of dat getal daar evenvaak voorkomt. 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 het array `gehad` op `False` geïnitialiseerd. Het algoritme gaat als volgt:

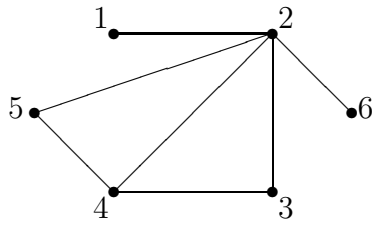
```
(1)   hetzelfde := True; j := 1;
(2)   while j ≤ n and hetzelfde do
(3)       if not gehad[j] then
(4)           telA := 0; i := j;
(5)           while i ≤ n do
(6)               if A[i] = A[j] then
(7)                   gehad[i] := True;
(8)                   telA := telA + 1;
(9)               fi
(10)              i := i + 1;
(11)          od
(12)          telB := 0; i := 1;
(13)          while i ≤ n do
(14)              if B[i] = A[j] then
(15)                  telB := telB + 1;
(16)              fi
(17)              i := i + 1;
(18)          od
(19)          hetzelfde := (telA = telB);
(20)      fi
(21)      j := j + 1;
(22)  od
(23)  return hetzelfde;
```

a. (7 punten)
Toon aan dat het *vergelijken* van array-elementen (test in regel (6) en test in regel (14) samen) een goede basisoperatie is om de complexiteit van dit algoritme mee te beschrijven. Doe dit door het aantal keren dat een regel wordt gedaan in verband te brengen met het aantal keren dat de test in regel (6) of de test in regel (14) wordt uitgevoerd.
Laat hiertoe onder andere zien dat de test in regel (6) ten minste n keer gebeurt.
b. (9 punten)
Hoeveel vergelijkingen tussen array-elementen worden er gedaan in het beste geval, voor algemene n? En voor wat voor invoerrijtjes A en B komt dat voor? Geef *alle* gevallen en leg daarbij ook uit waarom dit alle gevallen zijn.
c. (9 punten)
Idem, maar nu in het slechtste geval.

Opgave 5. (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 hiernaast en $k = 4$, dan is $I = \{1, 3, 5, 6\}$ 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 $k > 0$. Het moet "ja" opleveren dan en slechts dan als $\langle G, k \rangle$ 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.)

Het is bekend dat IS $\in \mathcal{NPC}$. We gaan nu bekijken wat er gebeurt als we de invoerwaarde k voor het Independent Set probleem fixeren; de invoer van het probleem is dan alleen een ongerichte graaf. Voor $k = 3$, bijvoorbeeld, krijgen we zo het beslissingsprobleem 3IS:

3IS: Gegeven een ongerichte graaf G. Bevat G een onafhankelijke knoopverzameling bestaande uit 3 knopen?

c. (5 punten)
Toon aan dat 3IS $\in \mathcal{P}$.

d. (5 punten)
Beantwoord de volgende vragen:

(i) Bestaat er een polynomiale reductie van IS naar 3IS? Waarom (niet)?
(ii) Bestaat er een polynomiale reductie van 3IS naar IS? Waarom (niet)?

Er wordt hier alleen naar het bestaan gevraagd. Je hoeft dus geen reductie te geven.

Veel succes!