Tentamen Complexiteit
Dinsdag 30 mei 2006, 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)
Bewijs met behulp van een beslissingsboomargument dat *elk* algoritme dat n waarden (oplopend) sorteert m.b.v. arrayvergelijkingen ten minste $\Omega(n \lg n)$ 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. (10 punten)
Bewijs dat het aantal vergelijkingen dat *Shellsort* met sprongafstanden (= stapgroottes = increments) $n/3, n/9, n/27, \ldots, 9, 3, 1$ in de worst case doet $\Omega(n^2)$ is. Neem aan dat n een macht van 3 is.
d. (5 punten)
We hebben de volgende sorteermethoden: Counting sort, Shellsort met increments $n/3^k$ (zie **c.**), Mergesort, Binary Insertion sort, Shellsort met Hibbard's increments $(2^k - 1, 2^{k-1} - 1, \ldots, 7, 3, 1)$ en Quicksort. Hoeveel arrayvergelijkingen doet elk van deze sorteermethoden in de worst case (alles in O/Ω-notatie)? Vergelijk ze vervolgens voor wat betreft hun worst case *complexiteit* (!): welke is het efficiëntst, welke de op een na beste, etcetera. Motiveer je antwoord.

Opgave 2. (15 punten)

Bekijk het volgende *recursieve* algoritme voor het aflopend sorteren van een rij van n verschillende getallen met $n \geq 0$ en n even.
Als $n \geq 2$ bepalen we eerst de grootste en de kleinste waarde door twee keer het hele array langs te lopen, en verwisselen we die met resp. het voorste en het achterste array-element. Nu staan $A[1]$ en $A[n]$ dus goed. Daarna sorteren we *recursief* de rest van het array. Noem het aantal vergelijkingen dat dit algoritme doet $S(n)$.

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

$$S(n) = \begin{cases} 0 & n = 0 \\ S(n-2) + 2n - 3 & n \geq 2,\ n \text{ even} \end{cases}$$

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 inderdaad voldoet. Laat bij het herhaald invullen de termen met n, $n-2$, $n-4$, etc. gewoon staan en veeg de 3-en bij elkaar. Dan is de algemene vorm beter te herkennen. Gebruik tenslotte dat $S(0) = 0$.

Hint bij het uitrekenen: gebruik –met even n– een van de volgende sommaties:

$$\sum_{i=1}^{\frac{n}{2}} i \;=\; \frac{n}{4}\cdot(\frac{n}{2}+1); \qquad \sum_{i=2,\ i \text{ even}}^{n} i \;=\; \frac{n}{2}\cdot(\frac{n}{2}+1)$$

Opgave 3. (30 punten)

We bekijken een algoritme (*zie volgende pagina*) dat de mediaan van n verschillende gehele getallen $A[1], A[2], \ldots, A[n]$ uitrekent; hierbij is $n > 1$ oneven. De *mediaan* is het unieke array-element waarvoor geldt dat er precies $(n-1)/2$ array-elementen groter (en dus ook precies $(n-1)/2$ array-elementen kleiner) zijn. In het algoritme zal *groter*$[i]$ het aantal array-elementen *groter* dan $A[i]$ (of een ondergrens daarvoor) gaan bevatten; analoog *kleiner*$[i]$. Als een van deze aantallen in regel (7) te groot blijkt, kan $A[i]$ zeker de mediaan niet zijn en wordt dit getal overgeslagen: *gedaan*$[i]$ blijft dan **False**. Anders worden alle tot dan toe nog niet met $A[i]$ vergeleken array-elementen, waaronder degene die eerder overgeslagen waren, alsnog in regel (12) met $A[i]$ in contact gebracht, en zo worden de precieze *groter*$[i]$ en *kleiner*$[i]$ bepaald voor deze i. In regel (16) blijkt of $A[i]$ de mediaan is. Merk op: in (11) betekent *gedaan*$[j]$ ($j \neq i$) is **True** zo altijd dat $A[j]$ al bekeken is (dus $j < i$) en dat precies bekend is hoeveel array-elementen groter resp. kleiner zijn.

a. (4 punten)
Voer het algoritme uit voor het rijtje $1, 2, 9, 3, 4, 6, 7, 8, 5$, met $n = 9$. Geef enkele tussenresultaten en het eindresultaat in de vorm van de inhoud van de arrays *gedaan*, *kleiner* en *groter*.

b. (8 punten)
Laat regel (7) (en bijbehorende **fi**) weg—het algoritme blijft dan correct. Laat zien dat het aantal keren dat de test in regel (11) wordt uitgevoerd in dat geval 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 regel (11) wordt uitgevoerd. Laat hiertoe onder andere zien dat regel (11) altijd minstens n keer gedaan wordt.

Vanaf nu zetten we regel (7) en bijbehorende **fi** er weer bij.

c. (4 punten)
Hoe vaak wordt de test in regel (11) minimaal uitgevoerd? En in welk geval gebeurt dat? Dezelfde vraag voor het aantal vergelijkingen tussen array-elementen (test in regel (12)).

d. (8 punten)
Neem hier aan dat $A[n]$ de mediaan is.
(i) Hoe vaak wordt de test in regel (11) dan minimaal uitgevoerd? Wanneer komt dat voor? Geef ook een voorbeeldrijtje waarvoor dat minimale aantal bereikt wordt (met $n = 9$).
(ii) Als (i), maar nu maximaal in plaats van minimaal.

e. (6 punten)
Het algoritme bekijkt elk tweetal verschillende array-elementen ofwel 0 keer ofwel 1 keer. Wat is derhalve het maximale aantal arrayvergelijkingen (regel (12)) dat het algoritme kan doen (algemene n), en voor wat voor soort invoer wordt dit maximale aantal onder andere gehaald? Toon dit laatste ook aan.

f. (**bonusopgave**, 5 punten)
Behalve de in **e.** gevonden worst case invoer bestaan er nog meer worst case gevallen waarvoor het maximale aantal arrayvergelijkingen gedaan wordt. Beschrijf deze allemaal.

Het algoritme voor de bepaling van de mediaan gaat als volgt:

```
(1)   i := 1;
(2)   while i ≤ n do
(3)       kleiner[i] := 0; groter[i] := 0; gedaan[i] := False;
(4)       i := i + 1;
      od
(5)   i := 1; found := False;
(6)   while not found do
(7)       if groter[i] ≤ (n − 1)/2 and kleiner[i] ≤ (n − 1)/2 then
(8)           gedaan[i] := True;
(9)           j := 1;
(10)          while j ≤ n do
(11)              if not gedaan[j] then
(12)                  if A[i] > A[j] then
(13)                      kleiner[i] := kleiner[i] + 1; groter[j] := groter[j] + 1;
                      else
(14)                      groter[i] := groter[i] + 1; kleiner[j] := kleiner[j] + 1;
                      fi
                  fi
(15)              j := j + 1;
              od
(16)          if groter[i] = (n − 1)/2 then
(17)              found := True;
              fi
          fi
(18)      i := i + 1;
      od
(19)  return i − 1;
```

Opgave 4. (30 punten)

We bekijken de volgende twee beslissingsproblemen:

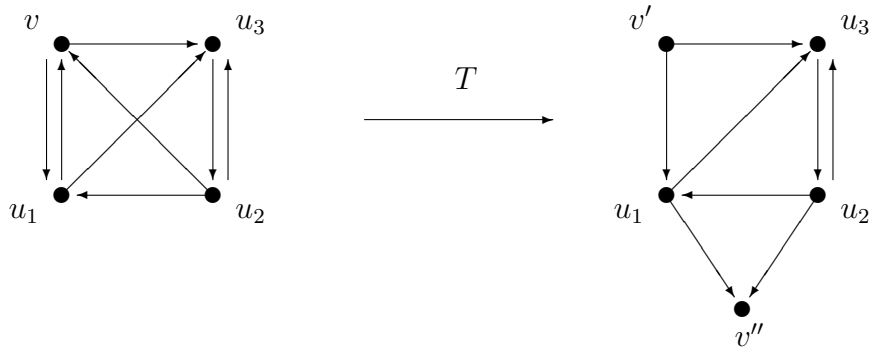
Gericht Hamiltoncircuitprobleem (GHC): Gegeven een gerichte graaf $\mathcal{G} = (V, E)$. Heeft $\mathcal{G}$ een (gerichte) Hamiltonkring, d.w.z. een (gerichte) kring die *elke* knoop precies *één* keer bevat?

Gericht Hamiltonpadprobleem (GHP): Gegeven een gerichte graaf $\mathcal{G} = (V, E)$. Bestaat er in $\mathcal{G}$ een (gericht) Hamiltonpad, d.w.z. een (gericht) pad dat *elke* knoop precies *één* keer bevat?

a. (10 punten)
Toon aan dat GHP $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor GHP te geven. Het algoritme heeft dus als invoer een gerichte graaf $\mathcal{G}$ en moet "ja" opleveren d.e.s.d.a. $\mathcal{G}$ 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.

We bekijken een transformatie T van instanties van GHC naar instanties van GHP.
Een gerichte graaf $\mathcal{G}$ wordt als volgt afgebeeld op een gerichte graaf $T(\mathcal{G}) = \mathcal{G}'$: kies een willekeurige knoop v in $\mathcal{G}$ en vervang deze door twee nieuwe knopen v' en v''. Alle uitgaande takken van v worden nu uitgaande takken van v' en alle inkomende takken van v worden inkomende takken in v''. Ofwel, elke pijl (v, w) in $\mathcal{G}$ wordt vervangen door een pijl (v', w), en elke pijl (u, v) in $\mathcal{G}$ door een pijl (u, v''). Zo heeft dus v' alleen uitgaande pijlen en v'' alleen inkomende.

Voorbeeld:



b. (5 punten)
Laat zien dat $T(\mathcal{G})$ in $O(|\mathcal{G}|^k)$ stappen (polynomiaal) uit $\mathcal{G}$ geconstrueerd kan worden.
c. (8 punten)
Toon aan dat geldt: $\mathcal{G}$ is een ja-instantie van GHC $\Longleftrightarrow$ $T(\mathcal{G})$ is een ja-instantie van GHP.
Ofwel: $\mathcal{G}$ heeft een (gerichte) Hamiltonkring $\Longleftrightarrow$ $\mathcal{G}'$ heeft een (gericht) Hamiltonpad.
Hint bij "$\Longleftarrow$": gegeven een Hamiltonpad in $\mathcal{G}'$, welke knopen moeten dan de uiteinden wel zijn?
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 GHC $\in \mathcal{NPC}$, dan volgt uit **a.**, **b.** en **c.** dat GHP NP-volledig is.
(ii) Als gegeven is dat GHC $\in \mathcal{NP}$ en GHP $\in \mathcal{NPC}$, dan volgt uit **b.** en **c.** dat GHC NP-volledig is.

Veel succes!