Tentamen Complexiteit
11 augustus 2005, 10.00-13.00 uur

Geef een **duidelijke toelichting** bij al je antwoorden.

Opgave 1. (25 punten)

a. (5 punten)
Met behulp van een *adversary-argument* kun je een ondergrens op de worst case complexiteit van een *probleem* bewijzen. Leg in woorden uit wat zo'n adversary-argument is en hoe het werkt.
b. (5 punten)
Bewijs met een adversary-argument dat *elk* algoritme (gebaseerd op het doen van arrayvergelijkingen) dat een array van n verschillende elementen sorteert in de worst case ten minste $\lceil \lg n! \rceil = \Theta(n \lg n)$ vergelijkingen moet doen. Geef hierbij in elk geval aan wat de strategie van de adversary is (dus wat voor antwoord geeft hij op een vraag van de vorm $A[i] < A[j]$) en waarom deze strategie leidt tot de ondergrens $\lceil \lg n! \rceil$.
c. (9 punten)
Leg aan de hand van het rijtje $6, 1, 5, 2, 5, 3, 5, 4$ ($n = 8$) uit hoe de volgende sorteermethodes werken (sorteer oplopend):
(i) Shellsort met als stapgroottes $n/2, n/2^2, \ldots, 2, 1$ (hier dus: $4, 2, 1$)
(ii) Counting sort
d. (6 punten)
Beantwoord de volgende vragen en geef een daarbij een korte toelichting.
(i) Is de ondergrens uit **b.** (te weten $\Theta(n \lg n)$) scherp?
(ii) Is Shellsort met als stapgroottes $n/2, n/2^2, \ldots, 2, 1$ ($n = 2^k$) optimaal voor wat betreft de worst case?
(iii) De worst case complexiteit van Counting sort is $O(n)$ (zie college). Is dit in tegenspraak met de ondergrens uit **b.**?

Opgave 2. (10 punten)

Gegeven een array A met n verschillende waarden $A[1], A[2], \ldots, A[n]$. We willen A oplopend sorteren. Een *inversie* is gedefinieerd als een paar $(A[i], A[j])$ waarvoor $i < j$ en $A[i] > A[j]$, m.a.w. een paar dat verkeerd om staat.
a. (5 punten)
Bewijs: 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. (5 punten)
(i) Laat zien dat Insertion sort ten minste $\frac{1}{2}n(n-1)$ arrayvergelijkingen moet doen in de worst case.
(ii) Moet Shellsort op grond van de stelling uit a. ook minstens $\frac{1}{2}n(n-1)$ arrayvergelijkingen doen in de worst case? Waarom (niet)?

Opgave 3. (15 punten)

We bekijken het volgende *recursieve* algoritme om de grootste en de op een na grootste waarde te vinden uit een rij van n getallen met $n = 2^k$ (k geheel, ≥ 1).
Als $n \geq 4$ bepalen we eerst recursief de grootste en de op een na grootste van de eerste $n/2$ elementen, vervolgens die uit de resterende $n/2$. Daarna moeten we hieruit nog de grootste en de op een na grootste van de gehele rij afleiden. Noem het aantal (array)vergelijkingen dat dit algoritme doet $G(n)$.
a. (4 punten)
Leg duidelijk uit waarom $G(n)$ voldoet aan de volgende recurrente betrekking:

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

b. (8 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 G als functie van n. *Hint*: gebruik dat $\sum_{i=1}^{l-1} 2^i = 2^l - 2$.
c. (3 punten)
Zoals bekend bepaalt de *toernooimethode* het grootste en het op een na grootste element van n stuks ($n = 2^k$) met $n + \lg n - 2$ vergelijkingen. Is bovenstaand recursieve algoritme optimaal als we kijken naar het precieze aantal vergelijkingen? En als we kijken naar het aantal vergelijkingen in orde van grootte (dus in de Θ-notatie)?

Opgave 4. (25 punten)

Gegeven een array A ($A[1], A[2], \ldots, A[n]$, met n een geheel getal > 1). Onderstaand algoritme "sorteert" A zodanig dat na afloop alle even elementen van A *vooraan* staan en de oneven elementen van A *achteraan*. De functie *wissel* verwisselt uiteraard de waarden van zijn argumenten. Verder betekent $A[i]$ **mod** 2: de rest bij deling van $A[i]$ door 2.

```
(1)   i := 1; j := n;
(2)   while i < j do
(3)       if (A[i] mod 2 = 0) then
(4)           i := i + 1;
(5)       else
(6)           if (A[j] mod 2 = 0) then
(7)               wissel(A[i], A[j]);
(8)               i := i+1;
(9)           fi
(10)          j := j − 1;
(11)      fi
(12)  od
```

We bekijken drie mogelijke operaties die in aanmerking komen om de complexiteit van het algoritme mee te beschrijven:
(I) het vergelijken van i en j in regel (2)
(II) het vaststellen of een array-element even of oneven is in regel (3) en regel (6)
(III) het verwisselen van array-elementen in regel (7)

a. (8 punten)
Geef van elk van deze drie operaties aan of ze de complexiteit van het algoritme goed beschrijven of niet. Zo ja, schat dan voor elke regel (die met **fi** of **od** uiteraard niet) het aantal keren dat deze wordt uitgevoerd af op het aantal keren dat de betreffende operatie ((I) of (II) of (III)) wordt gedaan. Zo nee, motiveer waarom niet.
b. (8 punten)
We bekijken operatie (I). Hoe vaak wordt deze operatie gedaan in de *best case* en voor wat voor invoerrijtjes komt dat voor (alle gevallen)? Idem voor de *worst case*. Let op de situatie n even of oneven. Motiveer je antwoord!
c. (5 punten)
Dezelfde vraag als **b**, maar nu voor operatie III.
d. (4 punten)
Hoe vaak wordt operatie II (regel (3) en regel (6) samen) gedaan in de worst case en voor wat voor invoerrijtjes komt dat voor? Geef alle gevallen.

Opgave 5. (25 punten)

We bekijken de volgende twee beslissingsproblemen:

Hamiltonkringprobleem (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.

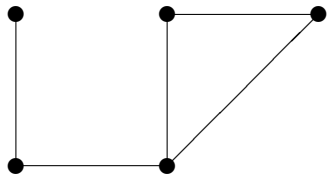
Hamiltonpadprobleem (HP): Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$.
Vraag: Heeft $\mathcal{G}$ een Hamiltonpad?
Een Hamiltonpad in een ongerichte graaf is een pad dat *elke* knoop precies *één* keer bevat.

a. (10 punten)
Toon aan dat HP $\in \mathcal{NP}$ door te laten zien dat er een *niet-deterministisch polynomiaal* algoritme voor HP bestaat. Het algoritme heeft dus als invoer een ongerichte graaf. Leg uit wat in elke fase van het algoritme gebeurt, en in het bijzonder wat in fase 2 wordt gecontroleerd en hoe. Wanneer is het antwoord van dit algoritme op een gegeven invoer $\mathcal{G}$ "ja" en wanneer "nee"? Geef duidelijk aan waarom je algoritme polynomiaal begrensd is.

De volgende transformatie beeldt een instantie van HP (een ongerichte graaf dus) af op een instantie van HC (ook een ongerichte graaf). Aan een ongerichte graaf $\mathcal{G}$ voegen we één knoop toe, en we verbinden deze door een tak met alle andere knopen. Dit levert een ongerichte graaf $\mathcal{G}'$. Noem deze transformatie T, dan $T(\mathcal{G}) = \mathcal{G}'$. Het is eenvoudig in te zien dat T polynomiaal is. Om te bewijzen dat HP $\leq_P$ HC moet verder nog bewezen worden: $\mathcal{G}$ is een ja-instantie van HP $\Longleftrightarrow$ $T(\mathcal{G})$ is een ja-instantie van HC, ofwel:
$\mathcal{G}$ heeft een Hamiltonpad $\Longleftrightarrow$ $\mathcal{G}'$ heeft een Hamiltonkring (*).

b. (6 punten)
Aan de hand van onderstaande voorbeeldgraaf $\mathcal{G}$ moet je aantonen dat (*) geldt. Geef daartoe eerst het beeld $T(\mathcal{G})$ van $\mathcal{G}$. Neem vervolgens een willekeurig Hamiltonpad in $\mathcal{G}$ en geef aan hoe hieruit een Hamiltonkring in $T(\mathcal{G})$ kan worden verkregen. Omgekeerd:

neem een willekeurige (maar andere dan zojuist bekeken) Hamiltonkring in $T(\mathcal{G})$ en laat zien hoe men daaruit een Hamiltonpad in $\mathcal{G}$ kan afleiden.



c. (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.)
d. (7 punten)
Motiveer bij onderstaande vragen je antwoord en formuleer duidelijk eventueel gebruikte stellingen.
(i) Als gegeven is dat HC $\in \mathcal{NPC}$ en HP $\in \mathcal{NP}$, volgt dan uit HP $\leq_P$ HC dat HP NP-volledig is?
(ii) Als gegeven is dat HC $\in \mathcal{NPC}$ en HP $\in \mathcal{NP}$, volgt dan uit HC $\leq_P$ HP dat HP NP-volledig is?
(iii) Stel dat we nu niet weten of HC $\in \mathcal{NPC}$. Laat wel gegeven zijn dat 3SAT $\leq_P$ HC en 3SAT $\in \mathcal{NPC}$. Volgt dan uit HC $\leq_P$ HP dat HP NP-volledig is?

Veel succes!