Tentamen Complexiteit
Vrijdag 8 juli 2016, 14.00 – 17.00 uur

Geef een *duidelijke* toelichting bij al je antwoorden! Veel succes!

Opgave 1. (13 punten)
Gegeven een array A dat $2n$ verschillende getallen bevat (n is even). De eerste $n/2$ elementen van A zijn oplopend gesorteerd, de tweede $3n/2$ elementen ook. Voorbeeld met $n = 6 : 2, 6, 11, 3, 5, 7, 8, 9, 12, 16, 18, 20$. We zoeken het n-de getal in grootte uit A; we zoeken dus het element waarvoor er precies $n-1$ waarden groter zijn (en dan ook precies n waarden kleiner).

a. (3 punten)
Gegeven een beslissingsboom corresponderend met een algoritme dat gebaseerd is op arrayvergelijkingen. Wat stelt –in termen van het algoritme– de hoogte van zo'n boom voor en wat kun je zeggen over de bladeren?
b. (10 punten)
Bewijs met behulp van een *beslissingsboomargument* dat elk algoritme voor bovenstaand probleem dat is gebaseerd op arrayvergelijkingen, voor de bekeken invoerrijtjes ten minste $\lceil \lg(n+1) \rceil$ vergelijkingen moet doen in de worst case. Formuleer je bewijs zorgvuldig en geef aan welke stellingen/eigenschappen je gebruikt. *Hint:* Welke array-elementen zijn zeker niet de n-de in grootte?

Opgave 2. (18 punten)
Gegeven een array A dat n (met n is even) verschillende getallen $A[1], A[2], \ldots, A[n]$ bevat. Het array is al enigszins voorgesorteerd: de eerste helft is oplopend gesorteerd en de tweede helft ook.

a. (6 punten)
Hoeveel inversies bevat een array van bovenstaande vorm maximaal? Geef een beschrijving van het soort rijtjes waarvoor dit maximale aantal gehaald wordt.
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.
b. (8 punten)
Bewijs het volgende: *elk* algoritme dat een array A van bovenbeschreven vorm oplopend sorteert met behulp van arrayvergelijkingen en dat per vergelijking ten hoogste één inversie opheft, moet in de worst case minstens $\frac{1}{4}n^2$ vergelijkingen doen.
c. (4 punten)
Welke van de volgende twee sorteermethoden voldoen/voldoet aan de (twee) voorwaarden van de stelling uit **b**? Motiveer je antwoord.
1. Bubble sort 2. Quicksort

Opgave 3. (27 punten)
Gegeven is een array A dat $n > 1$ gehele getallen bevat. We gaan het array sorteren met een aangepaste versie van Selection sort. We hebben daarbij steeds een reeds gesorteerd stuk $A[1]$ t/m $A[i-1]$, en een nog ongesorteerd stuk (de rest). In elke ronde doorlopen we het ongesorteerde stuk, op zoek naar de kleinste waarde. Zodra we een waarde kleiner dan $A[i]$ tegenkomen zetten we die vooraan in het ongesorteerde stuk op positie i. Verder verplaatsen we elk ander element dat gelijk is aan de huidige $A[i]$ ook meteen naar voren. Het gevolg is dat je mogelijk rondes kunt overslaan. Zie verder het algoritme hieronder.

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

De functie `wissel` verwisselt de waarden van zijn argumenten. Merk op dat de binnenste while-lus (regel (3) t/m (12)) eigenlijk een for-loop is. Verder geldt na de binnenste while-loop, dus direct voor regel (13): $A[i-1] < A[i] = A[i+1] = \cdots = A[k]$ en $A[k] < A[k+1], A[k+2], \ldots, A[n]$.

a. (8 punten)
Hoeveel vergelijkingen tussen array-elementen (regel (5)) worden er gedaan in het *beste* geval, voor algemene n? Voor wat voor invoerrijtjes komt dat voor? Geef *alle* gevallen en leg op basis van het algoritme uit hoe je aan je antwoord komt.
b. (8 punten)
Hoeveel vergelijkingen tussen array-elementen (regel (5)) worden er gedaan in het *slechtste* geval, voor algemene n? En voor wat voor invoerrijtjes komt dat voor? Geef *alle* gevallen en leg op basis van het algoritme uit hoe je aan je antwoord komt.

In onderdeel **c** en **d** kijken we naar het aantal verwisselingen, dus het aantal aanroepen van de functie `wissel`.

c. (5 punten)
Hoeveel verwisselingen doet het algoritme in ronde 1 op het rijtje $n, n-1, n-2, \ldots, 2, 1$? En in ronde 2? Laat na elk van beide rondes zien hoe het array er op dat moment uitziet. Hoeveel verwisselingen doet het algoritme in totaal op deze invoer?

d. (6 punten)
Hoeveel verwisselingen van array-elementen (regels (6) en (11)) doet het algoritme maximaal voor algemene $n > 1$? Beredeneer op basis van het algoritme voor wat voor invoerrijtjes dat voorkomt (*alle* gevallen). We nemen daarbij aan dat het array waardes uit $\{1, \ldots, n\}$ bevat (algemene n). Geef behalve het rijtje uit **c** nog een illustratief voorbeeld van een dergelijk worst case rijtje.

Opgave 4. (30 punten)
We bekijken de volgende twee beslissingsproblemen:

NAE4SAT: Gegeven een logische formule ϕ in 4-CNF (dus een AND van clausules waarbij elke clausule de OR van *precies vier* verschillende literals[1] is).
Vraag: is er een waardering (waarheidstoekenning) van de in ϕ voorkomende logische variabelen x_i, die per clausule minstens één literal `true` maakt en minstens één literal `false`?[2] (In het bijzonder maakt zo'n waardering dus ϕ waar.)
Opmerking: zo'n waardering zullen we een NAE-waardering noemen.

NAE3SAT: Gegeven een logische formule ϕ in 3-CNF (dus een AND van clausules waarbij elke clausule de OR van *precies drie* verschillende literals is).
Vraag: is er een waardering van de in ϕ voorkomende logische variabelen x_i, die per clausule minstens één literal `true` maakt en minstens één literal `false`?

Voorbeeld: $\phi = (x_1 \vee x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee x_4 \vee \neg x_3)$ heeft een NAE-waardering, namelijk bijvoorbeeld: $x_1 = \texttt{true}; x_2 = \texttt{false}; x_3 = \texttt{true}; x_4 = \texttt{false}$.

a. (10 punten)
Toon aan dat NAE4SAT $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor NAE4SAT te geven. Het algoritme heeft dus als invoer een logische formule ϕ in 4-CNF en moet "ja" opleveren dan en slechts dan als ϕ een ja-instantie is voor NAE4SAT (&). Het aantal verschillende in ϕ voorkomende logische variabelen is op voorhand niet bekend.
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 jouw niet-deterministische algoritme aan (&) voldoet en waarom het polynomiaal is.

We bekijken een transformatie T die instanties van NAE4SAT transformeert naar instanties van NAE3SAT. Gegeven een logische formule ϕ in 4-CNF. Noem de in ϕ voorkomende logische variabelen $x_1, x_2, \ldots, x_n$. We construeren uit ϕ een logische formule $\phi' = T(\phi)$ in 3-CNF. Voor elk van de clausules van ϕ maken we twee nieuwe clausules. De conjunctie van deze clausules is dan $T(\phi)$. We doen dit als volgt: de j-de clausule $(l_j^1 \vee l_j^2 \vee l_j^3 \vee l_j^4)$ wordt overgezet in $(l_j^1 \vee l_j^2 \vee y_j) \wedge (\neg y_j \vee l_j^3 \vee l_j^4)$. Alle variabelen y_j (één per clausule van ϕ) zijn nieuw gekozen.
Het is duidelijk dat de constructie van ϕ' uit ϕ polynomiaal is. Samen met **b** volgt dan dat NAE4SAT $\leq_{\mathrm{P}}$ NAE3SAT (#).

[1] Een literal is een variable x_i of zijn negatie $\neg x_i$.
[2] Met andere woorden: de literals zijn niet allemaal gelijk (Not All Equal).

Voorbeeld: De formule $\phi = (x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3 \vee x_4)$ wordt afgebeeld op $\phi' = (x_1 \vee \neg x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee y_2) \wedge (\neg y_2 \vee x_3 \vee x_4)$

b. (8 punten)
Toon aan dat geldt: ϕ is een ja-instantie van NAE4SAT $\iff$ ϕ' is een ja-instantie van NAE3SAT (ofwel: ϕ heeft een NAE-waardering $\iff$ ϕ' heeft een NAE-waardering).
c. (2 punten)
Wanneer is een beslissingsprobleem P NP-hard? En wanneer NP-volledig? (De definitie van $\mathcal{NP}$ hoeft niet te worden gegeven.)

$(*)$ Van college is bekend dat 3SAT $\in \mathcal{NPC}$. Tevens kan op dezelfde manier als bij **a** aangetoond worden dat NAE3SAT $\in \mathcal{NP}$. Ten slotte mag je als bekend veronderstellen dat 3SAT $\leq_{\mathrm{P}}$ NAE4SAT.

d. (10 punten)
Beantwoord de volgende vragen en geef een duidelijke toelichting.
(i) Leg uit waarom uit $(\#)$ en $(*)$ volgt dat NAE3SAT $\in \mathcal{NPC}$.
(ii) Er geldt ook dat NAE4SAT $\in \mathcal{NPC}$. Volgt dit uit **a**, $(\#)$ en **d**(i)?
(iii) Gegeven een beslissingsprobleem Q $\in \mathcal{P}$. Bestaat er een polynomiale reductie van NAE4SAT naar Q? Waarom wel/niet?

Opgave 5. (12 punten)
Schrijf een DTM-programma dat voor een gegeven invoerstring x bestaande uit nullen en enen nagaat of x van de vorm $(101)^n\,1$ is, dat wil zeggen bestaat uit nul of meer keer 101 en ten slotte één 1. Het programma termineert in toestand q_Y als x deze vorm heeft en anders in toestand q_N. De invoerstring mag tijdens of na de werking van het programma niet veranderd worden. De lees- en schrijfkop staat bij aanvang en na afloop op het eerste karakter van x.
Geef de transitiefunctie $\delta : Q - \{q_Y, q_N\} \times \Gamma \longrightarrow Q \times \Gamma \times \{-1, 0, 1\}$ in de vorm van een tabel, en beschrijf de werking van je DTM-programma (kort) in woorden door aan te geven waar elke toestand globaal voor dient. Q stelt de verzameling toestanden van de DTM voor, en $\Gamma = \{0, 1, b\}$ de verzameling gebruikte tape-symbolen.

EINDE