Tentamen Complexiteit
dinsdag 4 juni 2019, 10:00–13:00 uur

Universiteit Leiden
The Netherlands

Geef een *duidelijke* toelichting bij al je antwoorden!

Opgave 1. (25 punten)
Gegeven is een array $A = A[1], A[2], \ldots, A[n]$ dat n verschillende gehele getallen bevat, met $n \geq 2$ even. Het array bestaat uit $\frac{n}{2}$ opeenvolgende deelrijen, waarbij elke deelrij 2 elementen bevat. Gegeven is verder dat steeds beide waardes uit een deelrij groter zijn dan alle waardes ervoor, en kleiner dan alle waardes erna. We willen de rij *oplopend* sorteren. Een voorbeeld van zo'n rij met $n = 10$: $-3, -6, -1, 4, 9, 8, 14, 11, 17, 20$.

a. (6 punten)
Hoeveel vergelijkingen doet *Quicksort* op rijtjes zoals hierboven beschreven in het slechtste geval voor algemene (even) n? We bekijken de versie van *Quicksort* waarbij we steeds het eerste element van het betreffende rijtje als spil kiezen.

b. (10 punten)
Toon aan met behulp van een *beslissingsboomargument* dat elk sorteeralgoritme gebaseerd op arrayvergelijkingen voor deze invoerrijtjes ten minste $\frac{n}{2}$ vergelijkingen moet doen in de worst case. Formuleer je bewijs zorgvuldig.

c. (3 punten)
Geef aan waarom deze ondergrens niet in strijd is met de theoretische ondergrens voor sorteren zoals die op college is besproken.

d. (6 punten)
Ook door naar het aantal *inversies* te kijken, kun je voor dit probleem een ondergrens van $\frac{n}{2}$ bewijzen. Toon aan: elk sorteeralgoritme dat gebaseerd is op arrayvergelijkingen en dat per arrayvergelijking hooguit één inversie opheft, doet in de worst case voor de hier bekeken invoerrijtjes ten minste $\frac{n}{2}$ van deze vergelijkingen. In je bewijs moet in elk geval voorkomen wat een inversie is, hoeveel inversies de bekeken rijtjes maximaal hebben en wat de relatie is tussen inversies en het aantal vergelijkingen in de worst case.

Opgave 2. (20 punten)
Neem aan dat $n = 2^k$ met $k \geq 0$. Gegeven zijn verder twee $n \times n$ matrices A en B, bestaande uit reële getallen. We willen het matrixproduct $C = A \cdot B$ berekenen. We splitsen daartoe de matrices A, B en C ieder op in vier $\frac{n}{2} \times \frac{n}{2}$ deelmatrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \; B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \; C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Vervolgens berekenen we $C = A \cdot B$ recursief via:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \text{ met } \begin{array}{ll} C_{11} & = A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \\ C_{12} & = A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ C_{21} & = A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \\ C_{22} & = A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{array}$$

Zij $P(n)$ (respectievelijk $V(n)$) het aantal optellingen (respectievelijk vermenigvuldigingen) van array-elementen nodig om twee $n \times n$ matrices op bovenstaande manier recursief te vermenigvuldigen.

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

$$P(n) = \begin{cases} 0 & n = 1 \\ 8P(\frac{n}{2}) + n^2 & n \geq 2, n = 2^k \end{cases}$$

b. (5 punten)
Er geldt dat $P(n) = n^2(n-1)$. Los de recurrente betrekking op door deze herhaald in zichzelf in te vullen (substitutiemethode), en maak zo deze formule aannemelijk.
Hint: $\Sigma_{i=0}^{\ell-1} 2^i = 2^\ell - 1$.

c. (5 punten)
Bewijs vervolgens met behulp van volledige inductie dat de gegeven oplossing inderdaad voldoet.

d. (5 punten)
Leid een (eenvoudigere) recurrente betrekking af voor $V(n)$. Los deze ook exact op. (Een inductie-bewijs is niet nodig.)

Opgave 3. (30 punten)
Gegeven is een rij (array) A met n verschillende elementen $A[1], \ldots, A[n]$, waarbij $n \geq 2$ even is. We sorteren A oplopend met het volgende (wonderlijke) algoritme:

```
(1)   r := n/2;
(2)   while r ≥ 1 do
(3)       i := r + 1;
(4)       while i ≤ n − r + 1 do
              // nu A[i] op de juiste plek in A[r] ... A[i − 1] invoegen
(5)           x := A[i];
(6)           j := i − 1;
(7)           while j > r − 1 and A[j] > x do
(8)               A[j + 1] := A[j];
(9)               j := j − 1;
(10)          od
(11)          A[j + 1] := x;
(12)          i := i + 1;
(13)      od
(14)      r := r − 1;
(15)  od
```

We doen hierbij herhaald, in regel (3)–(13), *Insertion sort* op het "middengedeelte" $A[r] \ldots A[n - r + 1]$ van het array, dat we steeds laten groeien.

a. (10 punten)
Laat zien dat de operatie $A[j] > x$ in regel (7) maatgevend is voor de complexiteit van het algoritme. Breng hiertoe alle regels van het algorime in verstandig verband met deze operatie.

b. (8 punten)
Wat is de exacte best-case complexiteit van het algoritme, en geef alle bijbehorende instanties. Neem hierbij aan dat het array A een permutatie van $\{1, 2, \ldots, n\}$ is.

c. (12 punten)
Wat is de exacte worst-case complexiteit van het algoritme (eventuele sommaties mogen blijven staan), en geef enkele bijbehorende instanties. Neem hierbij wederom aan dat het array A een permutatie van $\{1, 2, \ldots, n\}$ is. Geef tevens *alle* instanties voor het geval $n = 4$.

Opgave 4. (25 punten)
Een logische formule ϕ staat in 3-CNF als ϕ een conjunctie (AND) is van clausules, waarbij elke clausule een disjunctie (OR) is van drie verschillende literals. Een literal is een x_i of een $\neg x_i$. We bekijken de volgende twee beslissingsproblemen:

3SAT: Gegeven een logische formule ϕ in 3-CNF.
Vraag: Bestaat er een waardering (waarheidstoekenning) van de in ϕ voorkomende logische variabelen x_i, die alle clausules in ϕ `True` maakt, dat wil zeggen dat per clausule minstens één literal waar is?

HALF3SAT: Gegeven een logische formule ϕ in 3-CNF met een *even* aantal clausules.
Vraag: Bestaat er een waardering (waarheidstoekenning) van de in ϕ voorkomende logische variabelen x_i, die precies de helft van de clausules in ϕ `True` maakt en de andere helft `False`?

Voorbeeld: $\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$ is een ja-instantie voor HALF3SAT. Immers, kies maar $x_1 = x_3 =$ `True` en $x_2 = x_4 =$ `False`.

a. (10 punten)
Toon aan dat HALF3SAT $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor HALF3SAT te geven. Het algoritme heeft als invoer een logische formule ϕ in 3-CNF met een even aantal clausules en moet "ja" opleveren dan en slechts dan als ϕ 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 jouw niet-deterministische algoritme aan (&) voldoet en waarom het polynomiaal is.

We bekijken een transformatie T die instanties van 3SAT transformeert naar instanties van HALF3SAT. Gegeven een logische formule ϕ in 3-CNF met m clausules en n logische variabelen x_i. We construeren daarbij een logische formule $T(\phi) = \psi$ in 3-CNF met $4m$ (dus een even aantal) clausules en $n + 3$ logische variabelen als volgt. Introduceer drie nieuwe variabelen u, v en w, die niet in ϕ voorkomen. $T(\phi)$ wordt dan een conjunctie van ϕ, m (dezelfde) clausules $(u \vee \neg u \vee v)$ en $2m$ (dezelfde) clausules $(u \vee v \vee w)$. De transformatie T beeldt zo ϕ in 3-CNF af op $\psi = T(\phi)$ in 3-CNF met een even aantal clausules. Het is duidelijk dat de constructie van $T(\phi)$ uit ϕ polynomiaal is.
Voorbeeld: de formule $\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$ wordt afgebeeld op
$\psi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (u \vee \neg u \vee v) \wedge (u \vee \neg u \vee v) \wedge (u \vee v \vee w) \wedge$
$(u \vee v \vee w) \wedge (u \vee v \vee w) \wedge (u \vee v \vee w)$b

b. (4 punten)
Toon aan dat geldt: ϕ is een ja-instantie van 3SAT $\Longrightarrow$ $T(\phi) = \psi$ is een ja-instantie van HALF3SAT.

c. (4 punten)
Toon aan dat geldt: $T(\phi) = \psi$ is een ja-instantie van HALF3SAT $\Longrightarrow$ ϕ is een ja-instantie van 3SAT.

d. (2 punten)
Wanneer is een beslissingsprobleem P NP-hard? En wanneer NP-volledig? (De definitie van $\mathcal{NP}$ hoeft niet te worden gegeven.)

e. (5 punten)
Van college is bekend dat 3SAT $\in \mathcal{NPC}$ en uit **a** dat HALF3SAT $\in \mathcal{NP}$. Tevens weten we via **b** en **c** dat 3SAT $\leq_P$ HALF3SAT (#).
(i) Er geldt ook: HALF3SAT $\leq_P$ 3SAT (##). Leg uit waarom dit het geval is.
(ii) Toon aan dat HALF3SAT $\in \mathcal{NPC}$. Gebruik je daarbij (#) of (##)?