Tentamen Complexiteit
Donderdag 14 juni 2018, 10.00 – 13.00 uur

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

Opgave 1. (20 punten)
Gegeven een array $A = A[1], A[2], \ldots, A[n]$ dat $n \geq 8$ verschillende getallen bevat (n is een viervoud). Gegeven is dat A bestaat uit $n/4$ opeenvolgende deelrijtjes ter lengte 4, waarbij elk deelrijtje oplopend gesorteerd is.
Voorbeeld met $n = 12$: $12, 20, 25, 36, 14, 16, 28, 42, 7, 8, 19, 26$.
Gevraagd worden de grootste en de op-twee-na-grootste waarde uit A. Preciezer: voor invoerrij A zoeken we het paar (p, q) waarvoor geldt dat $A[p]$ de grootste waarde uit A is en $A[q]$ de op-twee-na-grootste. Voor het voorbeeld wordt dat $(p, q) = (8, 7)$.

a. (4 punten)
Gegeven een beslissingsboom corresponderend met een algoritme dat gebaseerd is op arrayvergelijkingen. In termen van het algoritme, wat stelt een pad van de wortel naar een blad voor, wat is de betekenis van de lengte van zo'n pad, wat kun je zeggen over de bladeren en wat stelt de hoogte van zo'n boom voor?
b. (6 punten)
Hoeveel verschillende antwoorden (dus paren (p, q) met p de positie van de grootste en q de positie van de op-twee-na-grootste waarde) kunnen voorkomen bij invoerrijtjes van het bekeken type?
c. (4 punten)
Bewijs met behulp van een *beslissingsboomargument* dat elk algoritme voor bovenstaand probleem dat is gebaseerd op arrayvergelijkingen, voor de bekeken soort invoerrijtjes ten minste $\lceil 2 \lg n \rceil - 3$ vergelijkingen moet doen in de worst case. Formuleer je bewijs zorgvuldig, maak gebruik van **a** en **b** en geef aan welke stellingen/eigenschappen je gebruikt.
d. (6 punten)
Laat $\mathcal{B}$ een algoritme zijn (gebaseerd op het doen van arrayvergelijkingen) dat bovenstaand probleem oplost. Welke van onderstaande beweringen over $\mathcal{B}$ is/zijn in tegenspraak met het bewezene in **c** en welke kan/kunnen wel waar zijn? Licht je antwoorden toe!

(i) Het aantal arrayvergelijkingen dat $\mathcal{B}$ doet in de worst case is $O(\sqrt{\lg n})$.
(ii) $\mathcal{B}$ doet in de worst case $3n - 6$ arrayvergelijkingen en is derhalve niet optimaal.

Opgave 2. (15 punten)
Gegeven is een array $A = A[1], A[2], \ldots, A[n]$, dat verschillende getallen bevat en waarbij $n = 2^k$ voor een geheel getal $k \geq 0$. We gaan het array sorteren met behulp van Mergesort. Het aantal arrayvergelijkingen dat Mergesort doet in de *worst case* noemen we $M(n)$.

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

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

b. (10 punten)
Los de recurrente betrekking exact op door deze herhaald in zichzelf in te vullen (substitutiemethode). Schrijf $M(n)$ als functie van n. Bewijs vervolgens met behulp van volledige inductie dat de aldus gevonden oplossing inderdaad voldoet.
Je mag gebruiken dat $\sum_{i=0}^{\ell-1} 2^i = 2^\ell - 1$.

Opgave 3. (25 punten)
Gegeven is een array $A = A[1], A[2], \ldots, A[n]$ dat $n \geq 2$ gehele getallen bevat. We noemen een array-element $A[i]$ met $i < n$ een *leider* als geldt dat $A[i] > 2{*}A[j]$ voor alle $i{+}1 \leq j \leq n$.

Voorbeeld: de rij $15, 7, 3, 11, 3, 5, 2$ heeft twee leiders, namelijk $A[4] = 11$ en $A[6] = 5$.

We bekijken een algoritme dat de index van de voorste *leider* (in het voorbeeld 4) oplevert indien die bestaat, en anders -1. $A[n]$ is per definitie geen leider.
In onderstaand algoritme geldt zowel in regel (3) als in regel (5) dat als de eerste test onwaar is de tweede test niet meer wordt gedaan.

```
(1)   i := 1; leider := −1;
(2)   gevonden := False;
(3)   while (i < n) and not gevonden do
(4)       j := i + 1;
(5)       while (j ≤ n) and (A[i] > 2 ∗ A[j]) do
(6)           j := j + 1;
          od
(7)       if (j > n) then
(8)           leider := i; gevonden := True;
          fi
(9)       i := i + 1;
      od
(10)  return leider;
```

Toelichting: we lopen alle array-elementen af, kijken of de waarde minstens twee keer zo groot is als de waarde van elk element dat er rechts van staat en stoppen zodra we een leider gevonden hebben. Het vergelijken van array-elementen in regel (5) (tweede test) is maatgevend voor de complexiteit van het algoritme.

a. (6 punten)
Toon aan dat er *altijd* ten minste $n - 1$ arrayvergelijkingen (2^e test uit regel (5)) worden gedaan. Maak hierbij onderscheid tussen (i) het geval dat A geen leider heeft en (ii) het geval dat A wel een leider heeft.
b. (6 punten)
Voor algemene n worden in het *beste* geval precies $n - 1$ arrayvergelijkingen gedaan. Voor wat voor invoerrijtjes komt dat voor? Maak hetzelfde onderscheid als in **a.** Geef een karakterisering van *alle* best case gevallen en leg op basis van het algoritme uit hoe je aan je antwoord komt. Geef ten slotte twee concrete voorbeelden van best case invoerrijtjes met $n = 8$: de ene moet een leider hebben op een positie > 2, de andere heeft juist geen leider.

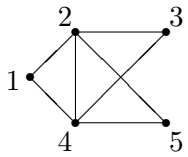
c. (3 punten)
In welke twee gevallen (algemene n) wordt de buitenste while-loop het vaakst uitgevoerd?
d. (5+5 punten)
Beantwoord voor elk van beide situaties uit **c** de volgende vraag.
Hoeveel vergelijkingen tussen array-elementen worden er gedaan in het *slechtste* geval? Voor wat voor invoerrijtjes komt dat voor? Beschrijf *alle* gevallen en leg op basis van het algoritme uit hoe je aan je antwoord komt. Geef ook een concreet voorbeeld met $n = 8$.

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

Independent Set (InSet): Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal k met $k > 0$. **Vraag:** bestaat er in $\mathcal{G}$ een onafhankelijke knoopverzameling I ($I \subseteq V$) ter grootte k?
Definitie. 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.

3SAT: Gegeven een logische formule ϕ in 3-CNF. **Vraag:** bestaat er een waardering van de in ϕ voorkomende logische variabelen x_i die ϕ True maakt?
Definitie. Een logische formule ϕ staat in 3-CNF als ϕ een conjunctie ($\wedge$) van clausules ($\vee$) is, waarbij *elke clausule* bestaat uit *drie verschillende literals*.

Voorbeeld:

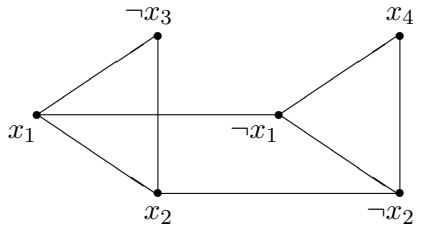


Nevenstaande graaf $\mathcal{G}_1$ heeft een onafhankelijke knoopverzameling ter grootte 3, namelijk $\{1, 3, 5\}$.
$\langle \mathcal{G}_1, 3 \rangle$ is dus een ja-instantie van InSet.

a. (8 punten)
Toon aan dat InSet $\in \mathcal{N}P$ door een *niet-deterministisch polynomiaal* algoritme voor InSet te geven. Het algoritme heeft dus als invoer een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k > 0$. Het algoritme moet "ja" opleveren dan en slechts dan als $x = \langle \mathcal{G}, k \rangle$ een ja-instantie is voor InSet (&). Je mag er van uitgaan dat het aantal knopen van $\mathcal{G}$ bekend is en dat $V = \{1, 2, \ldots, n\}$.
Leg uit wat in elke fase van het algoritme gebeurt en hoeveel stappen dat kost. Leg ook uit waarom jouw algoritme aan (&) voldoet en waarom het polynomiaal is in $|x|$.

We transformeren instanties van 3SAT naar instanties van InSet. Gegeven een logische formule ϕ in 3-CNF. Stel dat de formule m clausules bevat: $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, waarbij $C_r = l_1^r \vee l_2^r \vee l_3^r$ ($r = 1, \ldots, m$). We construeren daaruit een ongerichte graaf $\mathcal{G}_\phi = (V_\phi, E_\phi)$ als volgt. Voor elke clausule C_r maken we 3 knopen v_1^r, v_2^r en v_3^r in V_ϕ (deze corresponderen dus met l_1^r, l_2^r en l_3^r). Tussen elk tweetal van deze drie knopen brengen we een tak aan; zo wordt elke clausule dus afgebeeld op een aparte driehoek in $\mathcal{G}_\phi$. Verder brengen we een tak tussen twee knopen uit verschillende driehoeken aan als die knopen corresponderen met tegengestelde literals, dus met x_j en $\neg x_j$. Definieer nu de transformatie T als: $T(\phi) = \langle \mathcal{G}_\phi, m \rangle$. Het is duidelijk dat de constructie van $T(\phi)$ uit ϕ polynomiaal is. Samen met **b** volgt dan dat 3SAT $\leq_{\mathrm{P}}$ InSet.

Voorbeeld met $m = 2$: $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_1 \vee x_4)$ wordt afgebeeld op:



b. (5+5 punten)
Toon aan dat geldt: ϕ is een ja-instantie van 3SAT $\iff$ $T(\phi)$ is een ja-instantie van InSet. Formuleer wat het betekent als ϕ resp. $T(\phi)$ een ja-instantie is. Bewijs eerst "$\Longrightarrow$" en dan "$\Longleftarrow$".
c. (2 punten)
Wanneer is een beslissingsprobleem Q NP-hard? En wanneer NP-volledig? (De definitie van $\mathcal{NP}$ hoeft niet te worden gegeven.)

In vraag **d** hieronder mag je gebruiken dat (1) 3SAT $\in \mathcal{NPC}$, (2) InSet $\in \mathcal{NP}$ en (3) 3SAT $\leq_{\mathrm{P}}$ InSet. Verder definiëren we het probleem In3Set als volgt:

In3Set: Gegeven een ongerichte graaf $G = (V, E)$. **Vraag:** bestaat er in $\mathcal{G}$ een onafhankelijke knoopverzameling I ($I \subseteq V$) ter grootte 3?

d. (10 punten)
Beantwoord de volgende vragen en geef bij elk antwoord een *duidelijke uitleg.*
(i) Waarom volgt uit (1) en (3) dat In3Set $\leq_{\mathrm{P}}$ InSet?
(ii) Volgt uit (1), (2) en (3) dat InSet $\in \mathcal{NPC}$ of hebben we daar de omgekeerde reductie (dat wil zeggen InSet $\leq_{\mathrm{P}}$ 3SAT) voor nodig?
(iii) Stel dat InSet $\leq_{\mathrm{P}}$ In3Set. Wat betekent dat voor InSet in het bijzonder en voor $\mathcal{NP}$ in het algemeen?

Opgave 5. (10 punten)
Op de band van een DTM staat een string x bestaande uit nullen en enen en mogelijk precies één 2. Schrijf een DTM-programma dat de 2 uit die invoerstring x verwijdert indien x inderdaad een 2 bevat, en in dat geval alle karakters rechts ervan naar links aanschuift. Het programma termineert in dat geval in toestand q_Y. Als x geen 2 blijkt te bevatten wordt geëindigd in toestand q_N en blijft x onveranderd. De lees- en schrijfkop staat bij aanvang op het eerste karakter van x en na afloop op het laatste karakter van de uitvoerstring.
Geef de transitiefunctie $\delta\colon Q \setminus \{q_Y, q_N\} \times \Gamma \longrightarrow Q \times \Gamma \times \{-1, 0, 1\}$ 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 $\Gamma = \{0, 1, 2, b\}$ de verzameling gebruikte tape-symbolen.
Voorbeeld: de string 0 1 1 1 0 2 1 0 0 wordt 0 1 1 1 0 1 0 0 .

EINDE