Tentamen Complexiteit
Woensdag 28 mei 2025, 9.00 – 12.00 uur

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

Opgave 1. (10 punten)
Bewijs met behulp van een *adversary-argument* dat elk algoritme dat een array met n verschillende waarden sorteert met behulp van arrayvergelijkingen, in de worst case ten minste $\lceil \lg n! \rceil \in \Theta(n \lg n)$ arrayvergelijkingen doet. Geef duidelijk aan wat de strategie van de adversary is en waarom deze de ondergrens $\lceil \lg n! \rceil \in \Theta(n \lg n)$ voor de worst case oplevert.
Ter herinnering: sorteren komt neer op het vinden van de juiste (oplopende) ordening tussen de elementen van A. Voorbeeld: voor $9, 6, 7, 1, 3$ is dat $A[4] < A[5] < A[2] < A[3] < A[1]$.

Opgave 2. (20 punten)
Gegeven een array $A = A[1], A[2], \ldots, A[n]$ dat n verschillende gehele getallen bevat ($n \geq 3$ en deelbaar door 3). Het array bestaat uit $\frac{n}{3}$ opeenvolgende deelrijen, waarbij elke deelrij 3 elementen bevat. Gegeven is verder dat de drie waardes uit een deelrij altijd groter zijn dan de waardes uit de deelrijen ervoor, en kleiner dan die uit de deelrijen erna. We willen de rij *oplopend* sorteren.
Een voorbeeld van zo'n rij met $n = 9$ is: $5, 2, 1, 9, 8, 11, 17, 20, 33$.

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. (9 punten)
Bewijs met behulp van een *beslissingsboomargument* dat elk algoritme voor het hierboven beschreven probleem dat is gebaseerd op arrayvergelijkingen, in het slechtste geval (worst case) ten minste $\lceil \frac{n}{3}(1+\lg 3) \rceil$ arrayvergelijkingen moet doen. Formuleer je bewijs zorgvuldig, maak gebruik van **a** en geef aan welke stellingen/eigenschappen je gebruikt.
Hint: bereken eerst hoeveel antwoorden er mogelijk zijn voor de bekeken soort invoerrijtjes.
c. (3 punten)
Kan er een algoritme voor het probleem bestaan dat in de worst case $\Theta(n^2)$ arrayvergelijkingen doet? Indien een $\Theta(n^2)$-algoritme mogelijk is, kan dit dan optimaal zijn?
d. (4 punten)
Op college hebben we gezien dat *Quicksort* (met als spil steeds het voorste array-element) en *Insertion sort* in de worst case $\Theta(n^2)$ arrayvergelijkingen doen. Denk je dat het toepassen van deze sorteermethoden op bovenstaande speciale invoerrijtjes nog steeds $\Theta(n^2)$ is in de worst case? Motiveer je antwoord.

Opgave 3. (25 punten)
Gegeven is een array $A = A[1], A[2], \ldots, A[n]$ dat $n \geq 1$ positieve (> 0) gehele getallen bevat. Verder is een geheel getal $t \geq n$ gegeven. We willen indices $1 \leq \texttt{ene} \leq \texttt{andere} \leq \texttt{n}$ vinden waarvoor geldt dat $A[\texttt{ene}] + \cdots + \texttt{A}[\texttt{andere}] = t$. Als zulke indices niet bestaan krijgen

`ene` en `andere` de waarde 0.
In onderstaand algoritme wordt voor elke i de som $A[i] + \cdots + A[j]$ berekend voor $j \geq i$. Zodra deze som groter wordt dan t wordt j niet meer opgehoogd. Er wordt dan gecontroleerd of de som gelijk is aan t (en in dat geval worden de gevonden i en j bewaard) en verder gegaan met de volgende i. Het algoritme gaat als volgt:

```
(1)   i := 1; ene := 0; andere := 0;
(2)   while i ≤ n do
(3)       j := i; hoeveel := 0;
(4)       while j ≤ n and hoeveel < t do
          // als de eerste test False is wordt de tweede test niet meer gedaan
(5)           hoeveel := hoeveel + A[j];
              // nu hoeveel = A[i] + ··· + A[j]
(6)           j := j + 1;
          od
(7)       if hoeveel = t then
(8)           ene := i; andere := j − 1;
          fi
(9)       i := i + 1;
      od
```

Het optellen van array-elementen (regel (5)) is maatgevend voor de complexiteit van dit algoritme. Dit moet je in onderdeel **a.** bewijzen. Vervolgens bekijken we hoe vaak deze operatie gebeurt in het beste en het slechtste geval.

a. (10 punten)
Toon aan dat het optellen van array-elementen op regel (5) altijd ten minste n keer gebeurt. Laat vervolgens zien dat deze operatie maatgevend is voor de complexiteit van het algoritme. Breng daartoe het aantal keer dat *elke* andere operatie/regel van het algoritme plaatsvindt in verstandig verband met het aantal keer dat de operatie uit regel (5) gebeurt. Je mag de dubbele test in regel (4) als één operatie tellen.
b. (5 punten)
Hoeveel optellingen van array-elementen (regel (5) dus) worden er gedaan in het beste 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.
c. (10 punten)
Hoeveel optellingen van array-elementen worden er gedaan in het slechtste geval? En voor wat voor invoerrijtjes A komt dat voor? Leid dit af uit het algoritme en beschrijf *alle* gevallen.

Opgave 4. (15 punten)
Gegeven een array A met $n = 2^k$ gehele getallen. We zoeken een deelarray met maximale som. Met andere woorden, we zoeken indices a en b zó dat $\sum_{i=a}^{b} A[i]$ maximaal is.

Voorbeeld: gegeven het array $A = 1, -2, 6, -1, 4, -7, 2, 3$. Het (hier unieke) maximale deelarray is $6, -1, 4$ met som 9.

Om het probleem op te lossen splitsen we het array eerst in twee stukken van gelijke lengte en bepalen een maximaal deelarray van de ene helft en van de andere helft. Vervolgens bepalen we een deelarray met zo groot mogelijke som dat een deel van beide helften beslaat. Ten slotte vergelijken we de drie resultaten met elkaar, en geven we als antwoord het beste (met maximale som) deelarray dat we hebben gevonden.

a. (5 punten)
Laat $T(n)$ het aantal stappen zijn dat door bovenstaande methode wordt gedaan. Een stap is hier een optelling of een vergelijking van twee getallen. Het optellen van m getallen tellen we als m stappen. Beredeneer dat $T(n)$ voldoet aan:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\frac{n}{2}) + n + 3 & n \geq 2, n = 2^k \end{cases}$$

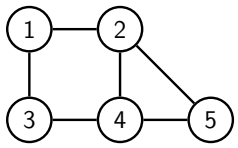
b. (10 punten)
Los de recurrente betrekking exact op door deze herhaald in zichzelf in te vullen (substitutiemethode). Schrijf $T(n)$ als functie van n. Bewijs vervolgens met behulp van volledige inductie dat de aldus gevonden oplossing inderdaad voldoet.
Je mag gebruiken dat $\sum_{i=0}^{\ell-1} 2^i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

Opgave 5. (30 punten)
We bekijken in deze opgave de volgende twee beslissingsproblemen.

3-SAT: Gegeven een logische formule ϕ met variabelen $x_1, \ldots, x_n$ in 3-CNF, waarbij elke clausule uit drie *verschillende* variabelen bestaat. Is er een waarheidstoekenning van de variabelen die ϕ waar maakt?

Dominating set (DS): Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k \geq 1$. Bestaat er een *dominating set* van k knopen? Een verzameling knopen $U \subseteq V$ heet een *dominating set* als iedere knoop in V ofwel zelf bevat is in U, ofwel direct verbonden is met een knoop in U.

Voorbeeld: beschouw als invoer $(\mathcal{G}, 2)$ (dus $k = 2$), met $\mathcal{G}$ als hieronder links:



De verzameling $U = \{1, 4\}$ is een dominating set in deze graaf: iedere knoop uit V is bevat in U of direct verbonden met een knoop in U. Dus $(\mathcal{G}, 2)$ is een ja-instantie van DS.

a. (10 punten)
Toon aan dat DS $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor DS te geven. Het algoritme heeft dus als invoer een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k \geq 1$, en moet "ja" opleveren dan en slechts dan als $(\mathcal{G}, k)$ 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 "ja" oplevert dan en slechts dan als $(\mathcal{G}, k)$ een ja-instantie is

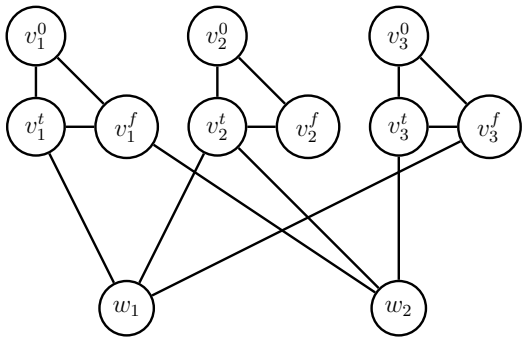
en waarom het polynomiaal is. Je mag aannemen dat $V = \{1, 2, \ldots, m\}$, met $m = |V|$.

Bekijk de transformatie T die instanties van 3-SAT transformeert naar instanties van DS. Gegeven een formule ϕ in 3-CNF. We construeren hieruit een ongerichte graaf $\mathcal{G}$ als volgt. Voor iedere variabele x_i definiëren we drie knopen, v_i^t, v_i^f en v_i^0, en voegen we de takken (v_i^t, v_i^f), (v_i^t, v_i^0) en (v_i^f, v_i^0) toe. Voor elke clausule $C_j = \ell_1 \vee \ell_2 \vee \ell_3$ voegen we bovendien een knoop w_j toe. Vervolgens voegen we een tak (v_i^t, w_j) toe als x_i in zuivere vorm voorkomt in C_j, en een tak (v_i^f, w_j) als de negatie van x_i in C_j voorkomt. Ten slotte kiezen we k gelijk aan het aantal variabelen in ϕ. Dus $T(\phi) = (\mathcal{G}, n)$.
Het is duidelijk dat de constructie van $T(\phi)$ uit ϕ polynomiaal begrensd is in $|\phi|$. $(*)$

Voorbeeld: Beschouw de instantie van 3-SAT gegeven door

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3).$$

Deze wordt door T omgezet in de instantie van DS met $k = 3$ en de volgende graaf $\mathcal{G}$:



b. (5 punten)
Bewijs: als ϕ een ja-instantie is van 3-SAT, dan is $T(\phi)$ een ja-instantie van DS.
c. (2 punten)
Beredeneer dat, als $T(\phi)$ een ja-instantie is voor DS, de dominating set U met n knopen dan precies één knoop uit ieder drietal $\{v_i^0, v_i^t, v_i^f\}$ moet bevatten.
d. (4 punten)
Bewijs nu: als $T(\phi)$ een ja-instantie is van DS, dan is ϕ een ja-instantie van 3-SAT.
e. (9 punten)
Gegeven dat 3-SAT $\in \mathcal{NPC}$. Zij P nu een willekeurig beslissingsprobleem waarvan we weten dat $P \in \mathcal{P}$. Welke van de volgende vijf uitspraken is/zijn waar, onwaar, of zeer waarschijnlijk onwaar? Leg je antwoorden kort uit met behulp van **a**, $(*)$, **b** en **d**.

1. 3-SAT $\leq_{\mathrm{P}}$ DS
2. DS $\leq_{\mathrm{P}}$ 3-SAT
3. DS $\in \mathcal{NPC}$
4. $P \leq_{\mathrm{P}}$ 3-SAT
5. 3-SAT $\leq_{\mathrm{P}} P$

EINDE