Tentamen Complexiteit
Maandag 5 juni 2023, 9.00 – 12.00

Geef een *duidelijke* toelichting/uitleg bij al je antwoorden!

Opgave 1. (24 punten)

Gegeven is een array $A = A[1], \ldots, A[n]$ dat n verschillende getallen bevat, met $n = 2^k$ en $n \geq 2$. Gegeven is dat $A[i] > A[i + \frac{n}{2}]$ voor alle $1 \leq i \leq \frac{n}{2}$, m.a.w., elk i-de getal in de linkerhelft is groter dan het i-de getal in de rechterhelft.
Voorbeeld: $2, 10, 8, 5, 7, 1, 9, 3, 4, 6$ ($n = 10$).

We zoeken in dit array de indices van het grootste en het op-één-na-grootste element; in het voorbeeld respectievelijk 2 (van $A[2] = 10$) en 7 (van $A[7] = 9$).

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. (4 punten)
Welke elementen uit A kunnen het grootste zijn? Welke elementen kunnen daarna nog het op-één-na-grootste zijn?

c. (6 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 $2 \lg n - 2$ 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. (4 punten)
Met de toernooimethode zijn de grootste en de op-één-na-grootste elementen van een arbitrair rijtje van $m = 2^\ell$ getallen te vinden in $m + \lg m - 2$ vergelijkingen. Leg uit hoe je de toernooimethode kan gebruiken om de grootste en de op-één-na-grootste elementen van A te vinden in $\frac{n}{2} + \lg n - 2$ vergelijkingen.

e. (3 punten)
Wat kun je op basis van de ondergrens uit **c** zeggen over de optimaliteit van het algoritme uit **d**?

f. (3 punten)
De toernooimethode is tijdens het college optimaal bewezen. Leg uit waarom het algoritme uit **d**, dat minder vergelijkingen doet om hetzelfde probleem op te lossen, niet in tegenspraak is met die optimaliteit.

Opgave 2. (15 punten)

Gegeven is een graaf $\mathcal{G}$ met $n = 2^k$ knopen ($n \geq 2$). Alle knopen hebben een kleur: de ene helft is wit en de andere helft is zwart. We tellen recursief het aantal takken tussen knopen met dezelfde kleur. Voor $n \geq 4$ verdelen we de knopen in twee gelijke helften, waarbij elke helft evenveel witte als zwarte knopen heeft. We tellen *recursief* het aantal takken tussen knopen van dezelfde kleur in beide helften. Vervolgens kijken we voor alle relevante paren knopen (u, v), waarbij u en v uit verschillende helften komen, of er een tak tussen loopt. Het basisgeval is $n = 2$. Het aantal keren dat we opzoeken of tussen twee knopen een tak loopt noemen we $T(n)$.

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

$$T(n) = \begin{cases} 0 & n = 2 \\ 2T(\frac{n}{2}) + \frac{n^2}{8} & n \geq 4, n = 2^k \end{cases}$$

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 hierbij gebruiken dat, voor $\ell \geq 1$:

$$\sum_{i=0}^{\ell-1} (\frac{1}{2})^i = 2 - \frac{1}{2^{\ell-1}}$$

Opgave 3. (29 punten)

Gegeven is een array $A = A[1], \ldots, A[n]$ dat $n \geq 2$ verschillende gehele getallen bevat. Het *gewicht* van een element $A[i]$ is het aantal elementen direct volgend op $A[i]$ dat kleiner is dan $A[i]$. Het tellen houdt dus op bij het eerstvolgende element dat groter is dan $A[i]$ of aan het einde van het rijtje.

Voorbeeld: in de rij $8, 3, 2, 7, 5, 9, 1, 4$ hebben de elementen respectievelijk de gewichten $4, 1, 0, 1, 0, 2, 0, 0$.

We bekijken een algoritme dat de index van het *zwaarste* element (het element met het grootste gewicht) oplevert. In het voorbeeld is dit 1 (van $A[1] = 8$), met gewicht 4. Als er meerdere zwaarste elementen zijn, wordt de index van het meest linker zwaarste element geretourneerd.
Op regel 4 geldt dat, als de eerste test onwaar is, de tweede test niet gebeurt.

```
 1 i := 1; zwaarst := 0; zwaarst_gewicht := −1;
 2 while i < n do
 3 |  j := i + 1; gewicht := 0;
 4 |  while j ≤ n and A[j] < A[i] do
 5 |  |  gewicht := gewicht + 1;
 6 |  |  j := j + 1;
 7 |  od
 8 |  if gewicht > zwaarst_gewicht then
 9 |  |  zwaarst := i;
10 |  |  zwaarst_gewicht := gewicht;
11 |  fi
12 |  i := i + 1;
13 od
14 return zwaarst;
```

a. (5 punten)
Laat zien dat de vergelijking $A[j] < A[i]$ op regel 4 maatgevend is voor de complexiteit van het algoritme. Breng hiertoe alle regels van het algoritme in verstandig verband met deze operatie.

b. (7 punten)
Hoeveel arrayvergelijkingen doet dit algoritme in de *best case*? Geef een exact antwoord en beschrijf *alle* bijbehorende invoerrijtjes. Leg op basis van het algoritme uit hoe je aan je antwoord komt.

c. (9 punten)
Hoeveel arrayvergelijkingen doet dit algoritme in de *worst case*? Geef een exact antwoord en beschrijf *alle* bijbehorende invoerrijtjes. Leg op basis van het algoritme uit hoe je aan je antwoord komt.

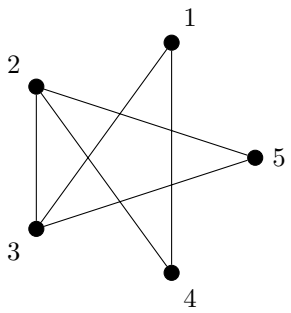
d. (8 punten)
De test op regel 2 kan worden aangescherpt naar $i < n - zwaarst\_gewicht$. Als $zwaarst\_gewicht \geq n - i$ zijn er immers niet genoeg elementen over om nog een zwaarder element te vinden. Leg uit of deze aanpassing de antwoorden op **b** en **c** verandert. Onderbouw je antwoord. In geval van verandering volstaat het om te onderbouwen dát het verandert, het is onnodig om een volledige nieuwe analyse te geven. Bespreek zowel het aantal vergelijkingen als de invoerrijtjes.

Opgave 4. (32 punten)

We bekijken de volgende twee beslissingsproblemen:

> **3SAT**: gegeven een logische formule ϕ in 3-CNF.
> **Vraag**: bestaat er een waardering (waarheidstoekenning) w van de in ϕ voorkomende logische variabelen x_i, die alle clausules in ϕ waar maakt, dat wil zeggen per clausule minstens één literal waar maakt?
>
> Een logische formule ϕ staat in 3-CNF als ϕ een conjunctie $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ is van clausules, waarbij elke clausule C_r een disjunctie $\ell_1^r \vee \ell_2^r \vee \ell_3^r$ is van drie verschillende literals. Een literal is een x_i of een $\neg x_i$, met x_i een logische variabele.
>
> **VertexCover (VC)**: gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k > 0$.
> **Vraag**: heeft $\mathcal{G}$ een *vertex cover* bestaande uit hooguit k knopen?
>
> Een vertex cover is een deelverzameling V' van V zodat voor elke tak $(u, v) \in E$ minstens één van de uiteinden u of v bevat is in V'.

Voorbeeld: onderstaande graaf $\mathcal{G}_1$ heeft een vertex cover ter grootte 3, namelijk $\{3, 4, 5\}$. $\langle \mathcal{G}_1, 3 \rangle$ is dus een ja-instantie van VC.



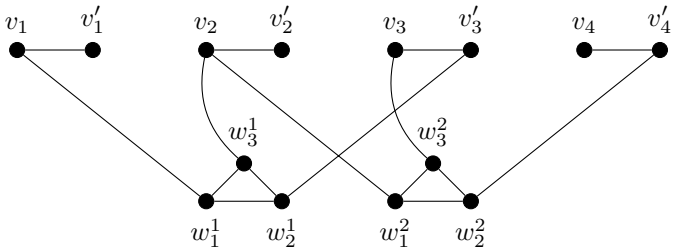
a. (10 punten)
Toon aan dat VC $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme te geven voor VC. Het algoritme heeft dus als invoer een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k > 0$. Het algoritme moet "ja" opleveren dan en slechts dan als $x = \langle \mathcal{G}, k \rangle$ een ja-instantie is voor VC (&). 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 hoe*, en hoeveel stappen dat kost. Leg ook uit waarom jouw algoritme voldoet aan (&) en waarom het polynomiaal is in $|x|$.

We bekijken een transformatie T die instanties van 3SAT transformeert naar instanties van VC. Gegeven een logische formule ϕ in 3-CNF met m clausules en n logische variabelen x_i. We construeren daaruit een ongerichte graaf $\mathcal{G}_\phi = (V_\phi, E_\phi)$ als volgt.

- Voor elke variabele x_i maken we twee knopen v_i en v_i' (deze corresponderen met de literals x_i en $\neg x_i$). Tussen deze knopen brengen we een tak aan.
- Voor elke clausule $C_r = \ell_1^r \vee \ell_2^r \vee \ell_3^r$ maken we drie knopen w_1^r, w_2^r en w_3^r (deze corresponderen met ℓ_1^r, ℓ_2^r en ℓ_3^r). Tussen elk tweetal van deze drie knopen brengen we een tak aan.
- We brengen een tak aan tussen knopen v_i (of v_i') en w_j^r als deze dezelfde literal voorstellen.

Definieer nu de transformatie T als: $T(\phi) = \langle \mathcal{G}_\phi, n + 2m \rangle$.

Voorbeeld ($m = 2$ en $n = 4$): $(x_1 \vee \neg x_3 \vee x_2) \wedge (x_2 \vee \neg x_4 \vee x_3)$ wordt de graaf:



b. (5 punten)
Toon aan: ϕ is een ja-instantie van 3SAT $\Longrightarrow$ $T(\phi)$ is een ja-instantie van VC.

c. (5 punten)
Toon aan: $T(\phi)$ is een ja-instantie van VC $\Longrightarrow$ ϕ is een ja-instantie van 3SAT.

Het is duidelijk dat de constructie van $T(\phi)$ uit ϕ polynomiaal is. Samen met **b** en **c** volgt dan dat 3SAT $\leq_{\mathsf{P}}$ VC.

d. (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 (1) 3SAT $\in \mathcal{NPC}$. Van hierboven mag je gebruiken dat (2) VC $\in \mathcal{NP}$ en dat (3) 3SAT $\leq_{\mathsf{P}}$ VC. Gegeven is verder een of ander probleem $Q \in \mathcal{P}$.

e. (10 punten)

(i) Kan je uit (1), (2) en (3) concluderen dat VC $\leq_{\mathsf{P}}$ 3SAT? Leg uit.

(ii) Kan je uit (1), (2), (3) en (i) concluderen dat VC $\in \mathcal{NPC}$? Leg uit.

(iii) Stel dat iemand bewijst dat VC $\leq_{\mathsf{P}} Q$. Is dit bijzonder? Waarom wel/niet?

(iv) Stel dat iemand bewijst dat $Q \leq_{\mathsf{P}}$ VC. Is dit bijzonder? Waarom wel/niet?

EINDE