Tentamen Complexiteit
Dinsdag 1 juli 2025, 9.00 – 12.00 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 verschillende gehele getallen bevat (n is even, $n \geq 6$). Gegeven is verder dat de eerste $\frac{n}{2}$ elementen van het array oplopend gesorteerd zijn. We willen de indices van de grootste en de op twee na grootste waarde uit A weten, dus we zoeken (p, q) met $A[p]$ de grootste waarde en $A[q]$ de op twee na grootste waarde.

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. (8 punten)
We bekijken nu bovenstaand probleem. Hoeveel antwoorden zijn er mogelijk voor invoerrijtjes van het aangegeven type? Geef een duidelijke berekening.
Hint: Maak bij het tellen onderscheid tussen het geval dat de grootste waarde in de linkerhelft van A zit, en het geval dat de grootste waarde zich in de rechterhelft bevindt.
c. (3 punten)
Bewijs nu (*beslissingsboomargument*) dat elk algoritme voor het hierboven beschreven probleem dat is gebaseerd op arrayvergelijkingen, in het slechtste geval (worst case) ten minste $\lceil \lg(\frac{n^2}{4} + n + 2) \rceil$ arrayvergelijkingen moet doen. Maak gebruik van **a** en **b** en geef aan welke stellingen/eigenschappen je gebruikt.
d. (5 punten)
Geef in woorden een eenvoudig algoritme dat, gebruikmakend van de speciale vorm van de invoer, de (indices van de) grootste en de op twee na grootste waarde vindt in $\frac{3n}{2} + 3$ arrayvergelijkingen. Betekent dit dat de ondergrens uit **c** niet scherp is?

Opgave 2. (15 punten)
Gegeven is een array $A = A[1], A[2], \ldots, A[n]$ ($n > 0$, $n = 2^k$), met op de oneven posities de kleinste $\frac{n}{2}$ getallen in oplopende volgorde, en op de even posities de overige getallen, ook in oplopende volgorde. Het array is dus 2-gesorteerd. Voorbeeldrijtje voor $n = 8$ met de getallen 1 t/m 8: $1, 5, 2, 6, 3, 7, 4, 8$.

a. (5 punten)
Leg uit hoe de sorteermethode *Shellsort*, met de stapgroottes zoals door Donald Shell gekozen, werkt op algemene rijtjes ter lengte n met $n = 2^k$. Geef onder andere aan wat h-sorteren inhoudt en hoe dat gebruikt wordt binnen Shellsort.

We bekijken in **b** en **c** de speciale rijtjes zoals hierboven aangegeven.

b. (5 punten)
Hoeveel inversies hebben zulke rijtjes voor algemene $n = 2^k$? Leg je antwoord uit.

c. (5 punten)
Bewijs, door onder andere het verband tussen het aantal inversies en het aantal arrayvergelijkingen verstandig te gebruiken (licht dit verband ook toe en tevens hoe je het gebruikt), dat Shellsort op dit soort rijtjes ten minste $\frac{n^2}{8} - \frac{n}{4}$ arrayvergelijkingen moet doen.

Opgave 3. (20 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)      // hier geldt dat A[i] = A[i + 1] = ··· = A[k] < A[k + 1], ··· , A[n]
(14)      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 worden de array-elementen gaandeweg het algoritme flink door elkaar gehusseld.
Het is eenvoudig in te zien dat het *vergelijken* van array-elementen (de test in regel (5)) maatgevend is voor de complexiteit van dit algoritme.

a. (5 punten)
Is het verwisselen van array-elementen (regel (11)) ook een goede maat voor de complexiteit van het algoritme, dat wil zeggen is die operatie ook maatgevend? Leg duidelijk uit waarom (wel/niet).
b. (3 punten)
Laat zien dat de test in regel (5) altijd ten minste $n - 1$ keer plaatsvindt.
c. (4 punten)
Hoeveel vergelijkingen tussen array-elementen (regel (5)) 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. Gebruik de commentaarregel (13).
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? Leid dit af uit het algoritme en geef *alle* gevallen.

Opgave 4. (15 punten)
Gegeven een array A met n gehele getallen, waarbij we aannemen dat n even is. We gaan het array als volgt sorteren. Eerst sorteren we alle elementen behalve de buitenste twee, d.w.z. behalve $A[1]$ en $A[n]$. Vervolgens schuiven we $A[1]$ naar rechts tot deze op de juiste plaats staat, en ten slotte schuiven we $A[n]$ naar links totdat deze ook goed staat.

a. (5 punten)
Laat $T(n)$ het aantal stappen zijn dat door bovenstaande methode in de worst case wordt uitgevoerd. Beredeneer dat $T(n)$ voldoet aan de recurrente betrekking gegeven door

$$T(n) = \begin{cases} 1 & n = 2, \\ T(n-2) + 2n - 3 & n > 2 \text{ even.} \end{cases}$$

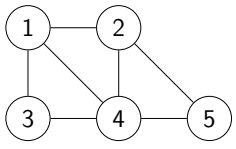
b. (10 punten)
Los de recurrente betrekking exact op door deze herhaald in zichzelf in te vullen (substitutiemethode). Schrijf $T(n)$ als functie van n. Bewijs vervolgens met behulp van volledige inductie dat de aldus gevonden oplossing inderdaad voldoet.
Je mag gebruiken dat $\sum_{i=1}^{\ell-1} i =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

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

Kliek: Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k \geq 1$. Bevat $\mathcal{G}$ een kliek van grootte minstens k? Dat wil zeggen: is er een $U \subseteq V$ met $|U| \geq k$ zodanig dat alle knopen in U onderling verbonden zijn?

Bijna-Kliek (BK): Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k \geq 1$. Bevat $\mathcal{G}$ een *bijna-kliek* van grootte minstens k? Een bijna-kliek is een verzameling knopen U waarvoor alle onderlinge takken aanwezig zijn behalve één.

Voorbeeld: beschouw de volgende graaf $\mathcal{G}$:

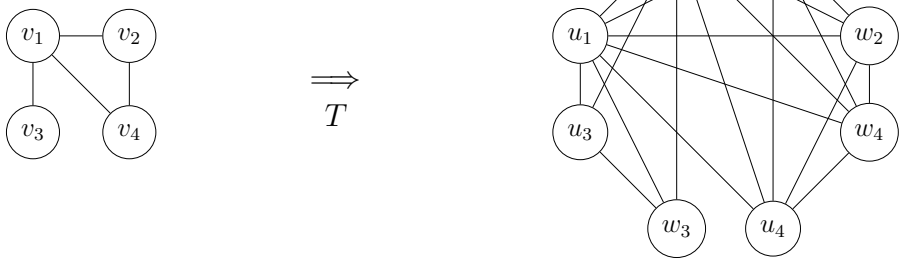


$\mathcal{G}$ bevat een kliek van grootte 3, bijvoorbeeld $U = \{1, 2, 4\}$. De graaf bevat geen kliek van grootte 4, maar wél een bijna-kliek van grootte 4: in de verzameling $U = \{1, 2, 3, 4\}$ zijn alle onderlinge takken aanwezig, behalve $(2, 3)$.

a. (10 punten)
Toon aan dat BK $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor BK 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 de invoer $(\mathcal{G}, k)$ 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 $m = |V|$.

We bekijken nu een transformatie T die instanties $(\mathcal{G}, k)$ van Kliek transformeert naar instanties $T(\mathcal{G}, k) = (\mathcal{G}', k')$ van BK.
Voor iedere knoop v_i in de oorspronkelijke graaf $\mathcal{G}$ definiëren we in onze nieuwe graaf $\mathcal{G}'$ twee knopen, die we u_i en w_i zullen noemen. We verbinden deze knopen met een tak (u_i, w_i), en voegen voor elke oorspronkelijke tak (v_i, v_j) ook de vier takken (u_i, u_j), (u_i, w_j), (w_i, u_j) en (w_i, w_j) toe. Ten slotte kiezen we het aantal knopen k' waar onze bijna-kliek in $\mathcal{G}'$ uit moet bestaan als $k' = 2k$.
Duidelijk: de constructie van $T(\mathcal{G}, k)$ uit $(\mathcal{G}, k)$ is polynomiaal begrensd in $|(\mathcal{G}, k)|$. $(*)$

Voorbeeld: Beschouw de instantie van Kliek gegeven door $k = 3$ en $\mathcal{G}$ als hieronder links. Deze wordt door T omgezet in de instantie van BK met $k' = 6$ en de graaf $\mathcal{G}'$ rechtsonder:



b. (4 punten)
Bewijs: als $\mathcal{G}$ een kliek van grootte k bevat, dan bevat $\mathcal{G}'$ een kliek van grootte $2k$.
c. (4 punten)
Bewijs: als $\mathcal{G}'$ een kliek van grootte $2k - 1$ bevat, dan bevat $\mathcal{G}$ een kliek van grootte k.
d. (3 punten)
Bewijs nu: als $\mathcal{G}'$ een bijna-kliek van grootte $2k$ bevat, dan bevat $\mathcal{G}$ een kliek van grootte k.
e. (9 punten)
Gegeven dat Kliek $\in \mathcal{NPC}$. Zij P nu een willekeurig beslissingsprobleem waarvan we weten dat $P \in \mathcal{P}$. Welke van de volgende vier uitspraken is/zijn waar, onwaar, of zeer waarschijnlijk onwaar? Leg je antwoorden uit met behulp van **a**, **b**, **d** en $(*)$.

1. Kliek $\leq_{\mathrm{P}}$ BK.
2. BK $\leq_{\mathrm{P}}$ Kliek
3. BK $\in \mathcal{NPC}$
4. Kliek $\leq_{\mathrm{P}} P$
5. $P \leq_{\mathrm{P}}$ Kliek

EINDE