Tentamen Complexiteit
Vrijdag 10 juni 2016, 10.00 – 13.00 uur

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

Opgave 1. (30 punten)
We hebben in deze opgave (behalve in **1d**) te maken met arrays $A = A[1]$, $A[2]$, ..., $A[n]$ die n (met n *oneven*) verschillende gehele getallen bevatten en die de volgende vorm hebben. Op de oneven posities $(1, 3, \ldots, n)$ staan negatieve (< 0) getallen en op de even posities staan positieve (> 0) getallen. We bekijken algoritmen die voor dit soort arrays de *mediaan* (= de middelste in grootte) bepalen.

a. (3 punten)
Gegeven een beslissingsboom corresponderend met een algoritme dat gebaseerd is op arrayvergelijkingen. Wat stelt –in termen van het algoritme– de hoogte van zo'n boom voor en wat kun je zeggen over de bladeren?
b. (10 punten)
Bewijs met behulp van een *beslissingsboomargument* dat elk algoritme voor bovenstaand probleem dat is gebaseerd op arrayvergelijkingen, voor de bekeken invoerrijtjes ten minste $\lceil \lg(n+1) \ -1 \rceil$ vergelijkingen moet doen in de worst case. Formuleer je bewijs zorgvuldig en geef aan welke stellingen/eigenschappen je gebruikt. *Hint:* Welke array-elementen zijn zeker niet de mediaan?
c. (4 punten)
Geef een eenvoudig, iteratief algoritme (in C++ of pseudocode), dat voor een rijtje van de gegeven vorm de index van de mediaan oplevert met $\lceil \frac{n}{2} \rceil - 1 = \frac{n-1}{2}$ arrayvergelijkingen.
d. (7 punten)
We bekijken in dit onderdeel een willekeurig array $A = A[1], A[2], \ldots, A[n]$ met n ($n \geq 1$, geheel) verschillende waarden. Bewijs nu uit het ongerijmde: elk algoritme gebaseerd op arrayvergelijkingen dat de grootste waarde uit A bepaalt doet in de worst case ten minste $n - 1$ arrayvergelijkingen.
e. (6 punten)
(i) Wat kun je op basis van de ondergrens uit **b** zeggen over de optimaliteit van het algoritme uit **c**? (ii) En op basis van de stelling uit **d**?

Opgave 2. (15 punten)
Gegeven is een array $A = A[1], A[2], \ldots, A[n]$, waarbij $n = 2^k$ met k geheel en $n \geq 2$. Het array bevat afwisselend positieve en negatieve getallen. Op de oneven posities (dus $1, 3, \ldots, n-1$) staan positieve getallen en op de even posities (dus $2, 4, \ldots, n$) negatieve getallen. De bedoeling is om het array door het verwisselen van elementen zo te reorganiseren dat alle negatieve getallen vooraan staan en de positieve achteraan. De onderlinge volgorde van de negatieve, resp. positieve getallen maakt niet uit. We doen dit recursief als volgt. Voor $n \geq 4$ reorganiseren we eerst *recursief* de linkerhelft en de rechterhelft. Vervolgens zorgen we dat de hele rij de gewenste vorm krijgt door de nog verkeerd staande array-elementen met elkaar te verwisselen. Het basisgeval is $n = 2$. Het aantal verwisselingen dat dit algoritme doet noemen we $W(n)$.

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

$$W(n) = \begin{cases} 1 & n = 2 \\ 2W(\frac{n}{2}) + \frac{n}{4} & n \geq 4, n = 2^k \end{cases}$$

b. (10 punten)
Los de recurrente betrekking exact op door deze herhaald in zichzelf in te vullen (substitutiemethode). Schrijf $W(n)$ als functie van n. Bewijs vervolgens met behulp van volledige inductie dat de aldus gevonden oplossing inderdaad voldoet.
Gebruik: $n = 2^k$, dus $\frac{n}{2} = 2^{k-1}$ en $k = \lg n$.

Opgave 3. (24 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)      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.
Het is eenvoudig in te zien dat het *vergelijken* van array-elementen (regel (5)) maatgevend is voor de complexiteit van dit algoritme.

a. (4 punten)
Wat weet je van de elementen $A[i]$ t/m $A[k]$ direct voor regel (13), dus na de binnenste

while-loop? Leg je antwoord uit.
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? Geef *alle* gevallen en leg op basis van het algoritme uit hoe je aan je antwoord komt.

In onderdeel **c** en **d** kijken we naar het aantal verwisselingen, dus het aantal aanroepen van de functie `wissel`.

c. (7 punten)
Hoeveel verwisselingen van array-elementen (regels (6) en (11)) doet het algoritme in het *beste* geval voor algemene $n > 1$? En voor wat voor invoerrijtjes komt dat voor? Leid dit af uit het algoritme en geef *alle* gevallen.
d. (5 punten)
Is het verwisselen van array-elementen een goede maat voor de complexiteit van het algoritme? Motiveer je antwoord.

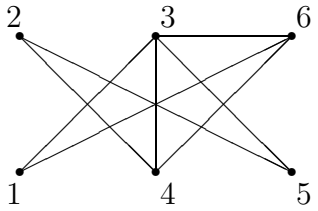
Opgave 4. (31 punten)
We bekijken de volgende twee beslissingsproblemen:

Kliek: Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een positief geheel getal $k \leq |V|$. Heeft $\mathcal{G}$ een *kliek* bestaande uit k knopen?

Half-Kliek: Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$, met een even aantal knopen (dus $|V|$ is even). Heeft $\mathcal{G}$ een kliek bestaande uit $|V|/2$ knopen?

Een *kliek* is een deelverzameling V' van V zodat voor elk tweetal knopen $u, v \in V'$ met $u \neq v$ geldt dat $(u, v) \in E$ (m.a.w. tussen elk tweetal knopen uit V' zit een tak).

Voorbeeld: Nevenstaande graaf met 6 knopen heeft een kliek die de helft van de knopen bevat, namelijk $\{1, 3, 6\}$. Overigens is ook $\{3, 4, 6\}$ een kliek met 3 knopen.

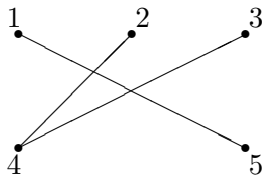


a. (10 punten)
Toon aan dat Half-Kliek $\in \mathcal{NP}$ door een *niet-deterministisch polynomiaal* algoritme voor Half-Kliek te geven. Het algoritme heeft dus als invoer een ongerichte graaf $\mathcal{G} = (V, E)$ met een even aantal knopen en moet "ja" opleveren dan en slechts dan als $\mathcal{G}$ een ja-instantie is (&). We mogen wel aannemen dat $V = \{1, 2, \ldots, n\}$, dus het aantal knopen is bekend. 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.

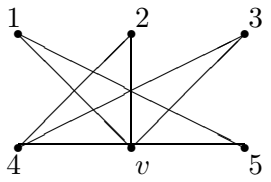
We bekijken een transformatie T die instanties van Kliek transformeert naar instanties van Half-Kliek. Gegeven een instantie $x = \langle \mathcal{G}, k \rangle$ van Kliek, dus een ongerichte graaf $\mathcal{G} = (V, E)$ met een even aantal knopen, en een positief geheel getal $k \leq |V|$. Het aantal knopen $|V|$ noemen we n. Uit $x = \langle \mathcal{G}, k \rangle$ construeren we een ongerichte graaf $T(x) = \mathcal{G}'$ als volgt.

- Als $k \geq \frac{n}{2}$ voegen we $2k - n$ geïsoleerde knopen toe aan $\mathcal{G}$ (dus knopen zonder takken).
- Als $k < \frac{n}{2}$ voegen we $n - 2k$ knopen toe en verbinden elke nieuwe knoop met *alle* andere knopen, dus zowel met de knopen die al in $\mathcal{G}$ zaten als met de nieuwe knopen.

Voorbeeld: met $\mathcal{G}$ als hier linksonder en $k = 2$:



wordt afgebeeld op:



Er wordt $5 - 2 \cdot 2 = 1$ knoop v toegevoegd en v wordt verbonden met knoop 1 t/m 5. Merk op: $\mathcal{G}$ heeft een kliek ter grootte 2, bijvoorbeeld $\{2, 4\}$, en het beeld heeft een kliek ter grootte $3 = 6/2$, bijvoorbeeld $\{2, 4, v\}$.

Het is duidelijk dat de constructie van $\mathcal{G}'$ uit $\mathcal{G}$ polynomiaal is. Samen met **b** volgt dan dat Kliek $\leq_{\mathrm{P}}$ Half-Kliek (#).

b. (9 punten)
Toon aan dat geldt: $\langle \mathcal{G}, k \rangle$ is een ja-instantie van Kliek $\iff$ $\mathcal{G}'$ is een ja-instantie van Half-Kliek.
Behandel de gevallen $k \geq \frac{n}{2}$ en $k < \frac{n}{2}$ apart.
c. (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 Kliek $\in \mathcal{NPC}$. Gegeven is verder een of ander probleem $Q \in \mathcal{P}$.

d. (10 punten)
(i) Er geldt ook dat Half-Kliek $\leq_{\mathrm{P}}$ Kliek. Leg uit waarom dit zo is zonder een concrete reductie te geven.
(ii) Volgt uit **a** en **d(i)** dat Half-Kliek $\in \mathcal{NPC}$, of volgt dat uit **a** en (#)? Leg je antwoord duidelijk uit.
(iii) Stel dat bewezen wordt dat Half-Kliek $\leq_{\mathrm{P}} Q$. Wat is daarvan het gevolg?
(iv) Bestaat er een polynomiale reductie van Q naar Kliek? Waarom wel/niet?

EINDE