

## Oefententamen IN2505-I Algoritmiek

Maart 2007

- Het gebruik van boek of aantekeningen tijdens dit tentamen is niet toegestaan.
- Dit tentamen heeft 15 meerkeuzevragen in totaal goed voor 3 punten en 4 open vragen in totaal goed voor 6 punten. Je krijgt sowieso 1 punt kado. Merk op dat je eindcijfer voor dit vak ook voor  $\frac{1}{3}$  uit het onafgeronde gemiddelde van het practicum bestaat (mits  $\geq 5.0$ ).
- Uiteraard komen in één tentamen niet alle onderwerpen aan bod. Trek daarom op basis van dit tentamen geen conclusies over stof die nooit getoetst wordt.
- Wat betreft de meerkeuzevragen:
  - Er is voor iedere vraag telkens maar één goed antwoord mogelijk.
  - Alle vragen tellen even zwaar, maar bij de puntentoekenning wordt wel gokkanscorrectie toegepast.
  - Schrijf je antwoorden eerst op kladpapier en neem ze later over op het antwoordformulier.
  - Vul je studienummer zowel met cijfers in als met streepjes/blokjes.
  - Vul het antwoordformulier bij voorkeur in met pen (geen rode pen) of potlood (goed zwart maken); gebruik geen doorhalingen.
  - Probeer in totaal niet meer dan een uur aan de meerkeuzevragen te besteden. De overige twee uur heb je nodig voor de open vragen.

- De open vragen worden als volgt gewogen:

Vraag:	1	2	3	4	Totaal:
Punten:	11	6	4	9	30
Score:					

- Geef antwoord in correct Nederlands of Engels en schrijf leesbaar (gebruik eerst kladpapier).
  - Geef geen irrelevante informatie. Dit kan leiden tot puntenaftrek.
  - Als er wordt gevraagd om een efficiënt algoritme, geef dan het meest efficiënte algoritme dat je kunt bedenken. Een langzamer algoritme kan minder punten opleveren.
  - Voordat je je antwoorden inlevert, controleer of op ieder blaadje je naam en studienummer staat en geef het aantal ingeleverde bladen voor de open vragen aan op (tenminste) de eerste pagina.
- De tentamenstof bestaat uit Chapters 1 to 7 uit Algorithm Design van Kleinberg en Tardos, behalve secties 4.9\*, 5.6, 6.10\*, 7.4\*, en 7.13\*, en bovendien de notities over de Master Method en over bewijstechnieken (Proving guide).
  - Totaal aantal pagina's: 8.



## Meerkeuzevragen

1. Laat het volgende gegeven zijn:

- de verzamelingen  $M = \{m_1, \dots, m_n\}$  en  $W = \{w_1, \dots, w_{n'}\}$ ,
- voor iedere  $m \in M$  een strikt geordende lijst met preferences (voorkeuren) over  $W$ , en voor iedere  $w \in W$  een strikt geordende lijst met preferences over  $M$ ,
- een matching  $S \subseteq M \times W$ .

We weten dan dat  $S$  stabiel is dan en slechts dan als

- A. de matching perfect is.
- B. er geen koppel  $(m, w) \in S$ , en  $m'$  en  $w'$  met  $m' \neq m$ , en  $w' \neq w$  te vinden zijn waarvoor geldt:  $m$  verkiest  $w'$  boven  $w$ , en  $w$  verkiest  $m'$  boven  $m$ .
- C. het aantal koppels in  $S$  gelijk is aan  $n$  en  $n = n'$ .
- D. zowel A als B geldt.

**Antwoord:** D (Ch.1)

2. Welk van de volgende beweringen is *niet* waar?

- A.  $n^3$  is  $\Omega(n^2)$ .
- B.  $n!$  is  $\Omega(2^n)$ .
- C.  $n \log n$  is  $O(n^c)$  voor  $c > 1$ .
- D.  $\log_3 n$  is  $\Theta(\log_5 n)$ .
- E.  $\sqrt{10n}$  is  $\Omega(n)$ .

**Antwoord:** E (Ch.2)

3. Laat een graaf  $G$  met  $n$  knopen (nodes) gegeven zijn. Welke van de volgende beweringen is niet equivalent met de andere?

1.  $G$  is verbonden (connected) en heeft  $n - 1$  kanten (edges).
2.  $G$  bevat geen cycle en is verbonden.
3.  $G$  is een boom.
4.  $G$  bevat geen cycle en heeft  $n - 1$  kanten.

Welke van deze beweringen is *niet* equivalent met de andere?

- A. 1
- B. 2
- C. 3
- D. 4
- E. De beweringen zijn allemaal equivalent.

**Antwoord:** E (Ch.3)

4. Je bent ingehuurd om een algoritme te schrijven om te bepalen of twee mensen in een groep van  $n$  mensen verwant zijn (direct of indirect). Je klant geeft je een lijst met namen van de personen  $p_1, \dots, p_n$  en een lijst met paren van de vorm  $(p_i, p_j)$  die aangeven dat persoon  $i$  verwant is met persoon  $j$ . Je besluit wijselijk om de data zo te processen dat je (geamortiseerd) in  $O(1)$  antwoord kunt geven op de vraag of twee gegeven personen verwant zijn. Welk algoritme is het meest efficiënt om te gebruiken in je programma voor deze preprocessing?
- A. Prim's algoritme voor de minimum spanning tree
  - B. Dijkstra's algoritme voor het kortste pad tussen twee knopen
  - C. Breadth-first search
  - D. Ford-Fulkerson voor maximum flow

**Antwoord:** C (Ch.3)

5. Gegeven een verbonden ongerichte bipartite graaf  $G = (V, E)$ .
1. Alle cycles in  $G$  hebben een oneven lengte.
  2. De knopen van  $G$  kunnen zo gekleurd worden met twee kleuren, dat er geen twee knopen  $u, v \in V$  met  $(u, v) \in E$  zijn die dezelfde kleur hebben.

Welke stelling(en) is (zijn) waar?

- A. Alleen stelling 1.
- B. Alleen stelling 2.
- C. Stellingen 1 en 2.
- D. Geen van beide stellingen 1 en 2.

**Antwoord:** Alleen stelling 2 is waar. Er bestaan juist geen cycles met oneven lengte. (Ch.3)

6. Gegeven een verzamelingen van  $n$  open intervallen (starttijd, eindtijd) in  $\mathbb{R}^+$ . Hoe zou je de intervallen sorteren om daarna in lineaire tijd te kunnen bepalen wat het grootste aantal overlappende intervallen is?
- A. Aflopend op starttijd (Latest starting time first)
  - B. Oplopend op eindtijd (Earliest finishing time first)
  - C. Oplopend op starttijd (Earliest starting time first)
  - D. Aflopend op eindtijd (Latest finishing time first)

**Antwoord:** C (Ch.4) hoewel er ook andere mogelijke algoritmen zijn (dit is dus niet zo'n goede vraag)

7. Gegeven een graaf  $G$  met unieke gewichten op iedere kant (edge). Gegeven ook een kant  $e$ . Wanneer weten we zeker dat  $e$  in iedere minimale opspannende boom zit (minimal spanning tree)?
- A. Als het gewicht van  $e$  het kleinste is van alle kanten in de graaf.
  - B. Als er een snede (cut)  $(A, B)$  van de knopen (nodes) in  $G$  bestaat waarbij  $e$  de kant is met het kleinste gewicht van alle kanten die een knoop in  $A$  verbinden met een knoop in  $B$ .
  - C. Als  $e$  voldoet aan de *cut-property*.
  - D. Ieder van de bovenstaande voorwaarden is voldoende.

**Antwoord:** D (Ch.4)

8. Laat de optimale prefix code gegeven zijn als een binaire boom.
- De twee laagst frequente letters hebben dezelfde parent (zijn siblings).
  - De twee laagst frequente letters zitten op het diepste niveau in de boom.
  - De boom is gebalanceerd.
  - Ieder van de bovenstaande beweringen is geldig.

**Antwoord:** B (Ch.4)

9. Gegeven zijn twee integers  $x$  en  $y$  van ieder  $n$  bits. We definiëren  $x_1$  als de  $\lfloor \frac{n}{2} \rfloor$  meest significante bits van  $x$  en  $x_0$  als de  $\lceil \frac{n}{2} \rceil$  minst significante bits van  $x$ . Analoog definiëren we  $y_1$  en  $y_0$ . Het algoritme om in  $O(n^{\log_2 3})$  tijd twee integers te vermenigvuldigen is gebaseerd op de volgende herschrijving van een vermenigvuldiging van  $x$  en  $y$  en een recursieve aanroep van de hierin voorkomende vermenigvuldigingen.
- $xy = (x_1 \cdot 2^{\frac{n}{2}} + x_0)(y_1 \cdot 2^{\frac{n}{2}} + y_0)$
  - $xy = x_1y_1 \cdot 2^n + ((x_1 + x_0)(y_1 + y_0) - x_1y_1 - x_0y_0) \cdot 2^{\frac{n}{2}} + x_0y_0$
  - $xy = x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{\frac{n}{2}} + x_0y_0$
  - $xy = x_1y_1 \cdot 2^n + (x_1y + x_0y) \cdot 2^{\frac{n}{2}}$

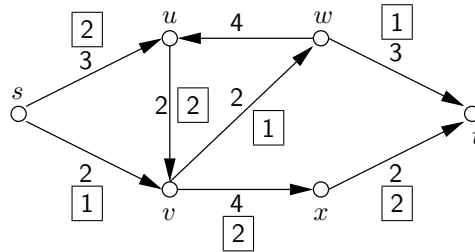
**Antwoord:** B (Ch.5)

10. Gegeven een verzameling punten  $(x_1, y_1), \dots, (x_n, y_n)$ . We definiëren  $e_{i,j}$  als de (minimale) fout die wordt verkregen als de punten  $i$  tot en met  $j$  worden benaderd met één lijnstuk.<sup>1</sup> We definiëren  $C$  als de kosten voor het introduceren van een extra lijnstuk. Welke van de volgende formules voor OPT geeft dan correct weer wat de minimale kosten zijn voor het benaderen van de punten 1 tot  $j$  met behulp van lijnstukken?
- $\text{OPT}(0) = 0$ , en voor  $j > 0$
- $\text{OPT}(j) = \min_{1 \leq i \leq j} (e_{i,j} + C + \text{OPT}(i - 1))$
  - $\text{OPT}(j) = e_{j-1,j} + C + \text{OPT}(j - 1)$
  - $\text{OPT}(j) = \min(e_{j-1,j} + C + \text{OPT}(j - 1), \text{OPT}(j - 1))$
  - $\text{OPT}(j) = e_{1,j} + C + \text{OPT}(\frac{j}{2})$

**Antwoord:** A (Ch.6)

11. Het probleem van het vinden van de ruimtelijke structuur van een RNA-molecuul (RNA Secondary Structure) is als volgt: gegeven een string basen  $B = b_1 \dots b_n$  vind een zo groot mogelijke verzameling  $S$  van paren  $(i, j)$  waarbij  $i, j \in \{1, \dots, n\}$ , die aan de volgende voorwaarden voldoen:
- Geen scherpe bochten: als  $(i, j) \in S$  dan  $|j - i| > 4$ .
  - Ieder paar bestaat ofwel uit een  $A$  en een  $U$ , ofwel uit een  $C$  en een  $G$ .
  - Geen enkele base  $b_i$  voor  $i \in \{1, \dots, n\}$  komt in meer dan één paar voor.
  - Geen kruisingen: Als  $(i, j)$  en  $(k, l)$  twee paren in  $S$  zijn, dan mag het niet zo zijn dat  $i < k < j < l$ .

<sup>1</sup>Hiertoe wordt de minimale som van de kwadraten van de Euclidische afstanden van de punten  $i$  tot en met  $j$  tot het lijnstuk berekend.



Figuur 1: Een stroom (flow) netwerk. De getallen bij de kanten zijn de capaciteiten. De stroom (flow) op iedere kant is weergegeven in een vierkantje.

De oplossing voor dit probleem met behulp van dynamisch programmeren slaat de resultaten van deelproblemen op in een tabel. Schrijf de recursieve functie die door dit algoritme gebruikt wordt op je kladpapier. Beantwoord dan de volgende vraag: wat is de complexiteit van dit algoritme?

- A.  $O(1)$
- B.  $O(n^3)$
- C.  $O(\log n)$
- D.  $O(n^2)$
- E.  $O(n)$

**Antwoord:** B (Ch.6)

12. Welke van de volgende beweringen is niet altijd geldig?

- A. De waarde van de stroom (flow) is gelijk aan de netto waarde van de stroom door iedere willekeurige  $s - t$  snede (cut).
- B. De waarde van de stroom kan nooit meer zijn dan de capaciteit van iedere willekeurige  $s - t$  snede (cut).
- C. De waarde van de maximale stroom is gelijk aan de rest-capaciteit over iedere willekeurige  $s - t$  snede (cut).
- D. De waarde van de maximale stroom in een netwerk is gelijk aan de capaciteit van de kleinste  $s - t$  snede (cut).

**Antwoord:** C (Ch.7)

13. Bekijk het flow netwerk in figuur 1 en beantwoord dan de volgende vraag. Wat is een mogelijk augmenting path?

- A.  $s - v - x - t$
- B.  $s - u - v - w - t$
- C.  $s - u - w - t$
- D.  $s - v - w - t$

**Antwoord:** D (Ch.7)

14. Stel dat we de capaciteit van iedere edge verdubbelen en we verdubbelen ook de maximale flow op iedere edge. Is dit dan nog steeds een maximale flow?

- A. Wel in dit voorbeeld, maar niet in het algemeen.
- B. Nee, nooit.
- C. Ja, dat geldt altijd.
- D. Alleen als de graaf geen cycles bevat.

**Antwoord:** C (Ch.7)

15. Stel dat voor iedere kant een minimale flow van 1 vereist is. Kunnen we het algoritme van Ford-Fulkerson dan nog steeds gebruiken om een geschikte stroom te vinden?
- A. Ja, maar we moeten dan overal de capaciteit met 1 verlagen.
  - B. Nee, we moeten dan een dynamisch programmeren benadering gebruiken.
  - C. Ja, we moeten dan een kant van  $t$  naar  $s$  toevoegen en in deze graaf een circulatie vinden.
  - D. Nee, er bestaat dan geen geldige stroom.

**Antwoord:** C (Ch.7)

## Open vragen

1. Je bedrijfje heeft de mogelijkheid om een groot aantal projecten uit te voeren. Het resultaat van sommige van die projecten is een voorwaarde voor andere projecten. Stel dat deze precedentierelaties zijn weergegeven in een a-cyclische gerichte (directed) graaf  $G = (V, E)$ . De knopen  $V$  representeren de projecten en iedere gerichte kant  $(u, v) \in E$  betekent dat  $u$  uitgevoerd moet worden voordat  $v$  uitgevoerd kan worden.

- (a) (6 punten) Bewijs dat als er geen cycles in de graaf  $G$  zitten, dat er dan altijd een knoop is zonder inkomende kanten (in maximaal 10 regels).

**Antwoord:** Dit is het bewijs voor Eq.3.19 uit het boek (p.102). Bewijs uit het ongerijmde:  
Stel dat je een graaf  $G = (V, E)$  hebt waarbij iedere knoop een inkomende kant heeft. Kies een willekeurige knoop  $v \in V$ . Deze knoop heeft tenminste één inkomende kant  $(u, v)$ . Volg deze inkomende kant terug naar de knoop  $u$ . Herhaal dit voor knoop  $u$  ipv  $v$ . We kunnen dit onbepert volhouden. Maar na  $|V| + 1$  stappen hebben we tenminste één knoop tweemaal bezocht. Laat  $C$  de serie knopen zijn tussen deze twee bezoeken.  $C$  is dan een cycle. Dit is tegenspraak met het feit dat  $G$  geen cycles heeft. Dus niet iedere knoop heeft een inkomende kant. (Ch.3)

- (b) (5 punten) Geef een algoritme in pseudocode dat een geschikte volgorde bepaalt om deze projecten uit te voeren (in maximaal 10 regels).

**Antwoord:** Dit is het algoritme voor het vinden van een topologische ordening.

```

while  $V \neq \emptyset$  do
  vind een knoop  $v$  zonder inkomende kant
  print  $v$ 
  wis  $v$  uit  $V$ 
end
(Ch.3)

```

2. (6 punten) Langs een lange weg staan hier en daar wat huizen. Uit ervaring blijkt dat er veel ongelukken gebeuren als mensen na een kroegbezoek in de auto stappen. De gemeente wil daarom dat er voor ieder huis een kroeg binnen fietsafstand is. De fietsafstand wordt gesteld op 5km. Gegeven deze randvoorwaarde, wil de gemeente zo min mogelijk kroegen langs de weg. Laat de afstand  $x_1, \dots, x_n$  tot ieder huis vanaf het begin van de weg gegeven zijn. Gegeven is het volgende (greedy) algoritme dat bepaalt op welke plaats(en) de kroeg(en) moet(en) komen.

sorteer de afstanden  $x_1, \dots, x_n$  (oplopend)

$l = -\infty$ ;  $j = 1$

**for**  $i = 1$  **to**  $n$  **do**

**if**  $|x_i - l| > 5$  **then**

$l_j = x_i + 5$

$l = l_j$

$j = j + 1$

**end**

**end**

Laat zien dat dit algoritme correct is. Geef hiertoe een bewijs volgens "greedy stays ahead" voor de volgende stelling (in maximaal 15 regels):

**Proposition.** Laat  $O = \{o_1, \dots, o_m\}$  de verzameling optimale locaties voor de kroegen zijn. Het greedy algoritme levert dan locaties  $L = \{l_1, \dots, l_k\}$  op, zodanig dat  $k \leq m$  en dat alle huizen  $x_i$  binnen fietsafstand zijn van een kroeg.



**Antwoord:** Het bewijs gaat met inductie over  $j$ . We laten zien dat altijd geldt:  $o_j \leq l_j$ .

- Basis: Als  $j = 1$  dan geeft het algoritme de locatie  $x_1 + 5$ . De optimale locatie  $o_1$  kan niet groter zijn, want dan wordt  $x_1$  niet afgedekt, dus  $o_1 \leq l_1$ .
- Stap: Beschouw  $j > 1$ .
  - Hypothese: Neem aan dat het algoritme locaties  $\{l_1, \dots, l_{j-1}\}$  heeft gevonden die alle huizen afdekt die ook worden afgedekt door  $\{o_1, \dots, o_{j-1}\}$  en dat  $o_{j-1} \leq l_{j-1}$ .
  - Alle huizen met  $x_i \leq l_{j-1} + 5$  worden nog door  $l_{j-1}$  afgedekt. Het algoritme kiest  $l_j$  zo, dat  $l_j$  precies 5km komt na het eerste huis  $i$  wat niet door  $l_{j-1}$  wordt afgedekt. Dit huis wordt ook niet door  $o_{j-1}$  afgedekt omdat  $o_{j-1} \leq l_{j-1}$ . Dus  $o_j \leq x_i + 5 = l_j$ .
- Met volledige inductie geldt nu dat dit algoritme locaties  $\{l_1, \dots, l_k\}$  heeft gevonden die alle huizen afdekt die ook worden afgedekt door  $\{o_1, \dots, o_k\}$  en dat  $o_k \leq l_k$ . Hieruit volgt direct dat  $L$  nooit meer locaties bevat dan  $O$  en dus  $k \leq m$ . (Ch.4)

3. Geef een asymptotische boven en ondergrens voor  $T(n)$  in ieder van de volgende recurrente betrekkingen. Neem aan dat  $T(n)$  constant is voor  $n \leq 10$ . Maak je grenzen zo krap (tight) mogelijk en geef aan hoe je aan je antwoord komt.

(a) (2 punten)  $T(n) = 4T(n/5) + n \log n$

**Antwoord:**  $\log_b a = \log_5 4 < 1$ , dus  $f(n) = \Omega(n^{\log_5 4})$ . Dit is geval 3 van de Master Method. Check regularity condition:  $4 \frac{n}{5} \log \frac{n}{5} \leq \frac{4}{5} n \log n$ . Is dus waar voor  $c = \frac{4}{5}$ . Dus  $T(n)$  is  $O(n \log n)$ .

(b) (2 punten)  $T(n) = 2T(n/3) + \log^2 n$

**Antwoord:**  $\log_b a = \log_3 2$ , en  $f(n) = O(n^{\log_3 2})$ . Dit is geval 1 van de Master Method. Dus  $T(n)$  is  $O(n^{\log_3 2})$ .

4. In een fabriek staan  $m$  machines. In de fabriek worden  $n$  soorten taken uitgevoerd. Voor iedere type taak  $j$  is bekend welke machines  $M_j$  die taak kunnen uitvoeren. Iedere dag moeten er tussen de  $c_j$  en  $c'_j$  taken worden afgehandeld. Iedere machine  $i$  kan maximaal  $m_i$  taken per dag uitvoeren. Het probleem is om taken aan machines toe te wijzen zodanig dat aan deze eisen voldaan wordt. Gegeven is dat er nooit meer dan  $k = \max_j c'_j$  taken van één type worden afgehandeld per dag.

(a) (6 punten) Geef een polynomiaal algoritme (in  $m$ ,  $n$  en/of  $k$ ) om een toewijzing te vinden. Doe hierin een aanroep van een bekend algoritme voor het bepalen van de maximale flow (in maximaal 20 regels).<sup>2</sup>

**Antwoord:** Deze opgave komt overeen met Survey Design uit het boek. Noem de verzameling taaksoorten  $X$ . Noem de verzameling machines  $Y$ . Construeer nu de volgende graaf  $G$ :

- Maak voor ieder taak type  $j \in X$  een knoop.
- Maak voor iedere machine  $i \in Y$  een knoop.
- Verbind nu iedere taakknop  $j \in X$  met een machineknop  $i \in M_j$  door een edge met capaciteit  $\infty$ .

<sup>2</sup>Je hoeft dit algoritme niet uit te schrijven.

- Verbind een knoop  $s$  met iedere taakknoop  $j$  door een edge met capaciteit  $[c_j, c'_j]$ .
- Verbind iedere machineknoop  $i$  met een knoop  $t$  door een edge met capaciteit  $m_i$ .
- Verbind tot slot de knoop  $t$  met de knoop  $s$  met capaciteit  $[\sum_j c_j, \sum_j c'_j]$ .
- Geef iedere knoop een demand van 0.

In de resulterende flow graaf  $G$  bestaat een geldige kringloop (circulatie) dan en slechts dan als er een geldige manier is om taken aan machines toe te wijzen. Schrijf dit probleem als volgt om in een netwerk flow probleem  $G'$ :

1. Werk de ondergrenzen als volgt weg:
  - (a) Verhoog de demand van  $t$  met  $\sum_{j \in X} c_j$ .
  - (b) Verlaag de capaciteit op de edge van  $t$  naar  $s$  met  $\sum_{j \in X} c_j$ .
  - (c) Laat de demand in  $s$  op 0 door de verlaging vanwege  $(t, s)$  weg te strepen tegen de verhoging door de ondergrenzen in de uitgaande kanten.
  - (d) Verlaag daartoe de demand in iedere taakknoop  $j$  met  $c_j$ . Die wordt nu dus  $-c_j$ .
  - (e) Maak de capaciteit op iedere edge van  $s$  naar  $j$  nu  $c'_j - c_j$ .
2. Werk de demands weg door het introduceren van een nieuwe knoop  $s'$  en  $t'$ . Verbind  $s'$  met iedere taakknoop  $j$  met capaciteit  $c_j$  en verbind  $t$  met  $t'$  met capaciteit  $\sum_{j \in X} c_j$ .

Als de maximale flow van  $s'$  naar  $t'$  in  $G'$  gelijk is aan  $\sum_{j \in X} c_j$  dan bestaat er een geldige kringloop en dus ook een geldige toewijzing van taken aan machines. (Ch.7)

- (b) (3 punten) Geef aan hoeveel de looptijd is van je algoritme en hoe je hieraan komt (in maximaal 5 regels).

**Antwoord:** De grafen  $G$  en ook  $G'$  bestaan uit  $O(m+n)$  knopen en  $O(mn)$  kanten. De maximale flow is  $nk$ . De looptijd van Ford-Fulkerson is het aantal kanten maal de maximale flow. Dat is hier dus  $O(mn^2k)$ . (Ch.7)