TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Tentamen IN2505-I Algoritmiek

5 april 2007, 14.00-17.00

- 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.
- 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:	6	10	8	6	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 notitie over de Master Method en over bewijstechnieken (Proving guide).
- Totaal aantal pagina's: 5.

Meerkeuzevragen

1. Laat het volgende gegeven zijn:

   - de verzamelingen $M = \{m_1, \ldots, m_n\}$ en $W = \{w_1, \ldots, w_{n'}\}$,
   - voor iedere $m \in M$ een preferentieordening (preferences order) over W, en voor iedere $w \in W$ een preferentieordening over M,
   - een matching $S \subseteq M \times W$.

   Welke van de volgende beweringen is waar?

   A. Matching S is stabiel als $n = n'$.

   B. Matching S is perfect als $n = n'$.

   C. In iedere stabiele matching is er een koppel (m, w) zodat m op de eerste plaats komt bij w en w op de eerste plaats in de preferentieordening van m.

   D. Als er een $m \in M$ en een $w \in W$ zijn zodat m op de eerste plaats komt bij w en w op de eerste plaats in de preferentieordening van m komt, dan zullen m en w in iedere stabiele matching aan elkaar gekoppeld worden.

2. Welk van de volgende beweringen is waar?

   A. n^2 is $\Omega\left(n^3\right)$.

   B. $n!$ is $O\left(n^{1000}\right)$.

   C. $\log_7 n$ is $\Theta\left(\log_3 n\right)$.

   D. $n \log n$ is $O\left(n^c\right)$ voor $0 < c < 1$.

3. Laat een ongerichte (undirected) graaf G met $n \geq 3$ knopen (nodes) gegeven zijn. Stel dat G ook n kanten heeft. Wat weten we dan over G?

   A. G is een boom

   B. G is verbonden

   C. G bevat een cycle

   D. G is bipartiet (bipartite)

4. Gegeven een verbonden graaf G en een knoop s. De afstand tussen twee knopen is gedefinieerd als het aantal kanten van het kortste pad tussen die twee knopen. Wat is de meest efficiënte manier om de knoop te vinden die het verst verwijderd ligt vanaf s? Een variant op:

   A. Depth-first search

   B. Breadth-first search

   C. Dijkstra's shortest-path algorithm

   D. Bellman-Ford's shortest-path algorithm

5. Laat een gerichte graaf $G = (V, E)$ en een knoop $s \in V$ gegeven zijn. We zeggen dan dat G sterk verbonden (strongly connected) is, dan en slechts dan als:

   A. Als iedere knoop $v \in V$ direct verbonden is met s.

   B. Als er voor iedere knoop $v \in V$ een pad is van v naar s en ook een pad van s naar v.

   C. Als het kortste pad van iedere knoop $v \in V$ naar s korter is dan het kortste pad van s naar v.

   D. Als er voor iedere twee knopen $v, w \in V$ meerdere (tenminste twee) paden zijn van v naar w.

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 (interval partitioning)?
   A. Aflopend op eindtijd (Latest finishing time first)
   B. Aflopend op starttijd (Latest starting time first)
   C. Oplopend op eindtijd (Earliest finishing time first)
   D. Oplopend op starttijd (Earliest starting time first)

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

8. Huffman's algoritme voor het construeren van een prefix code maakt een binaire boom van de te coderen letters. Bij het construeren wordt de volgende greedy regel toegepast.
   A. De twee letters met de laagste frequentie komen naast elkaar onderin de boom.
   B. De letters worden gesorteerd op aflopende frequentie en daarna een voor een vanaf bovenaan in de boom gehangen: dus highest frequency first (hoogste frequentie eerst).
   C. De letters worden gesorteerd op oplopende frequentie en daarna een voor een vanaf bovenaan in de boom gehangen: dus lowest frequency first (laagste frequentie eerst).
   D. De te coderen letters worden zodanig in twee groepen verdeeld dat te coderen letters ongeveer even vaak in de ene groep zitten als in de andere. Het algoritme wordt dan recursief op beide groepen aangeroepen.

9. Gegeven een verzameling van n punten in het twee dimensionale vlak, hebben we een $O(n \log n)$ algoritme gevonden om de twee punten te vinden die het dichtste bij elkaar liggen. Het belangrijkste deel van dit algoritme is de combineer (merge) stap van de twee recursieve aanroepen. Deze combineerstap wordt gedaan in:
   A. $O(1)$
   B. $O(n)$
   C. $O(\log n)$
   D. $O(n \log n)$

10. Direct na je studie werk je twee maanden als freelancer. Aan het begin van die periode kies je de projecten waar je aan gaat werken. Je kunt kiezen uit een verzameling van n projecten. Voor ieder project $1 \leq i \leq n$ schat je in hoeveel tijd t_i het project zal kosten en hoe leuk je het vindt om aan dat project te werken $0 \leq l_i \leq 10$.[1] Je wilt graag een verzameling projecten $P \subseteq \{1, \ldots, n\}$ kiezen die je in T tijd kunt afronden, en waaraan je zoveel mogelijk plezier zult hebben, dus je wilt de volgende functie maximaliseren: $\sum_{i \in P} l_i$. Met welke van de volgende formules voor OPT kun je bepalen wat het maximum is voor deze functie?
    A. $\mathsf{OPT}(0) = 0$ en $\mathsf{OPT}(j) = \max_{1 \leq i \leq j} (l_i + \mathsf{OPT}(i - t_i))$
    B. $\mathsf{OPT}(0) = 0$ en $\mathsf{OPT}(j) = \max (l_j + \mathsf{OPT}(j - t_j), \mathsf{OPT}(j - 1))$
    C. $\mathsf{OPT}(0, t) = 0$ en $\mathsf{OPT}(j, t) = \begin{cases} 0 & \text{if } t \leq 0 \\ \max_{1 \leq i \leq j} (l_i + \mathsf{OPT}(i, t - t_i)) & \text{otherwise} \end{cases}$
    D. $\mathsf{OPT}(0, t) = 0$ en $\mathsf{OPT}(j, t) = \begin{cases} 0 & \text{if } t \leq 0 \\ \max (l_j + \mathsf{OPT}(j - 1, t - t_j), \mathsf{OPT}(j - 1, t)) & \text{otherwise} \end{cases}$

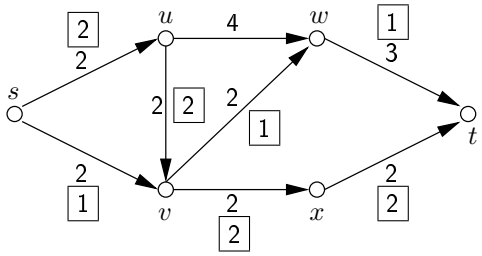
[1] Je uurloon voor alle projecten is gelijk.

11. Het probleem van het vinden van de beste alignment van twee strings is als volgt: gegeven strings X en Y van lengte m en n, match posities i van X met posities j van Y zodanig dat de boete voor mismatches in een alignment geminimaliseerd wordt. Laat een alignment M een verzameling van koppels (i, j) zijn waarbij $i \in \{1, \ldots, m\}$ en $j \in \{1, \ldots, n\}$. De boete voor mismatches in een alignment M wordt als volgt berekend:

    1. een boete $\delta \geq 0$ voor iedere positie $i \in \{1, \ldots, m\}$ of $j \in \{1, \ldots, n\}$ die in geen enkel koppel in M voorkomt, plus
    2. een boete $\alpha_{x_i y_j} \geq 0$ voor iedere $(i, j) \in M$ waarvoor $x_i \neq y_j$.

    De oplossing voor dit probleem door middel 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. $O(m + n)$
    B. $O(mn)$
    C. $O\left(n^2\right)$
    D. $O(mn^2)$



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.

12. Bekijk het flow netwerk in figuur 1 en beantwoord dan de volgende vraag. Wat is de capaciteit van de s-t snede (cut) $(\{s, v\}, \{u, w, x, t\})$?

    A. 2
    B. 4
    C. 6
    D. 8

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. $s - v - u - w - t$
    D. Geen van deze paden is een augmenting path.

14. Bekijk het flow netwerk in figuur 1 en beantwoord dan de volgende vraag. Stel dat we de capaciteit van iedere edge verdubbelen. Is de minimum cut dan nog dezelfde partitie van knopen?

    A. Nee, dat is nooit het geval.
    B. Niet in dit voorbeeld, maar soms wel.
    C. Wel in dit voorbeeld, maar niet in het algemeen.
    D. Ja, deze stelling is geldig voor ieder stroom netwerk.

15. Gegeven een stroom (flow) netwerk G met ondergrenzen (lower bounds) op de kanten en vraag (demand) op de knopen. Om te bepalen of er een geldige (feasible) kringloop (circulation) in dit netwerk bestaat moeten eerst de ondergrenzen weggewerkt worden. We construeren daartoe een nieuw netwerk G' als volgt. We verlagen de capaciteit van iedere kant met de ondergrens. Om te zorgen dat er een geldige kringloop in G bestaat dan en slechts dan als er een geldige kringloop in G' bestaat, passen we G' verder als volgt aan:

    A. Voor iedere gerichte kant (u, v) met een ondergrens lb, verhoog de vraag (demand) in u met lb en verlaag de vraag in v met lb.

    B. Voor iedere gerichte kant (u, v) met een ondergrens lb, verhoog de vraag (demand) in v met lb en verlaag de vraag in u met lb.

    C. Introduceer een nieuwe bron (source) s en afvoer (sink) t. Voor iedere gerichte kant (u, v) met een ondergrens lb, verbind s met u met als capaciteit lb en v met t met als capaciteit lb.

    D. Introduceer een nieuwe bron (source) s en afvoer (sink) t. Voor iedere gerichte kant (u, v) met een ondergrens lb, verbind s met v met als capaciteit lb en u met t met als capaciteit lb.

Open vragen

1. 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 (in maximaal 5 regels per recurrente betrekking).
   (a) (3 punten) $T(n) = 8T(n/2) + n^3$
   (b) (3 punten) $T(n) = 2T(n/3) + n^2 \log n$

2. Een paar vrienden van je doen onderzoek naar verborgen berichten. Ze willen daarom weten of een bepaalde tekst verborgen is in een grotere tekst door soms wat letters over te slaan. (Zoals de Omega code in de Torah.)

   We zeggen dat een string S' van lengte m *verstopt is* in een grotere string S van lengte n op posities $k_1 < k_2 < \ldots < k_m$ als voor iedere letter $j = 1, \ldots, m$ in S' geldt dat $s'[j] = s[k_j]$, waarbij we met $s[j]$ de letter op positie j in S en met $s'[k_j]$ de letter op positie k_j in S' aanduiden. Bijvoorbeeld is "hoi" verstopt in "honderdduizend" op de plaatsen $k_1 = 1$, $k_2 = 2$, en $k_3 = 10$.
   (a) (5 punten) Gegeven een bericht S van lengte n en S' van lengte m, geef een $O(m + n)$ algoritme dat kan bepalen of S' in S verstopt zit (in maximaal 10 regels).
   (b) (5 punten) Je vrienden willen er echt zeker van zijn dat je programma correct is. Geef daarom een bewijs met inductie ("greedy stays ahead") voor de volgende stelling (in maximaal 15 regels):
   Proposition. *Gegeven dat S' verstopt is op posities $l_1 < \ldots < l_m$ in S (en dus voor iedere letter $j = 1, \ldots, m$ in S' geldt dat $s'[j] = s[k_j]$), dan geldt voor iedere $j = 1, \ldots, m$ dat je algoritme een $k_j \leq l_j$ vindt waarvoor $s'[j] = s[k_j]$.*

3. Het loopt tegen het einde van het kwartaal en je moet nog voor n projecten ieder één laatste opdracht doen. Iedere opdracht wordt beoordeeld op een schaal van 1 tot g waarbij $g > 1$. Je doel is natuurlijk om je gemiddelde cijfer te maximaliseren. Je wil in totaal niet meer dan H uur aan alle opdrachten samen besteden. Om het eenvoudig te houden ga je er vanuit dat je aan iedere opdracht een geheel aantal uren zal besteden. Om goed te kunnen beslissen hoeveel tijd je het beste aan iedere opdracht kunt besteden, heb je voor iedere opdracht een functie gemaakt: als je $0 \leq h_i \leq H$ uur aan project i besteedt (met $i \in \{1, \ldots, n\}$), verwacht je een cijfer $f_i(h)_i$. De functies zijn monotoon stijgend (dus als $h < h'$ dan is $f_i(h) \leq f_i(h')$).

   Het probleem is nu als volgt: gegeven deze functies, beslis hoeveel uur je aan ieder project gaat besteden zodat je gemiddelde cijfer (berekend met de functies f_i) zo hoog mogelijk is. Merk op dat dit probleem hetzelfde is als het optimaliseren van het totaal aantal behaalde punten, dus $\sum_i f_i(h_i)$.
   (a) (6 punten) Geef een iteratief algoritme met behulp van dynamisch programmeren met een looptijd polynomiaal in n en H dat het maximaal totaal aantal te halen punten berekent (in maximaal 15 regels).
   (b) (2 punten) Geef aan wat de looptijd is van je algoritme en hoe je hieraan komt (in maximaal 3 regels).

4. Laat een stroom (flow) netwerk $G = (V, E)$ met capaciteiten $c : E \to \mathbb{N}$, een stroom (flow) $f : E \to \mathbb{N}$ en een s-t snede (cut) (A, B) gegeven zijn, waarbij $s \in A$, en $t \in B$. We definiëren $f^{\text{ out of }}(v) = \sum_{e \text{ out of } v} f(e)$ als de totale stroom die uit een knoop v vertrekt en $f^{\text{ into }}(v) = \sum_{e \text{ into } v} f(e)$ als de totale stroom die een knoop v binnenkomt. Voor verzamelingen knopen $A \subset V$ definiëren we analoog $f^{\text{ out of }}(A) = \sum_{e \text{ out of } A} f(e)$ en $f^{\text{ into }}(A) = \sum_{e \text{ into } A} f(e)$. Gebruik deze notatie zo mogelijk in je antwoord op de volgende vragen.
   (a) (1 punt) Geef de twee voorwaarden voor een geldige stroom f volgens de definitie (in maximaal 4 regels).
   (b) (1 punt) Geef de definitie van de waarde (value) van de stroom f, dus $v(f)$ (in maximaal 2 regels).
   (c) (1 punt) Geef de definitie van de capaciteit van een s-t snede (A, B), dus $\text{cap}(A, B)$ (in maximaal 2 regels).
   (d) (3 punten) Je mag er vanuit gaan dat de waarde van de stroom gelijk is aan de netto waarde van de stroom door de s-t snede (A, B): $v(f) = f^{\text{ out of }}(A) - f^{\text{ into }}(A)$. Gebruik dit resultaat en de voorgaande definities (voor zover nodig) om te bewijzen dat de netto waarde van de stroom niet groter is dan de capaciteit van de s-t snede (A, B), oftewel bewijs dat $v(f) \leq \text{cap}(A, B)$ (in maximaal 4 regels).

Einde tentamenopgaven