TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Tentamen IN2505-I Algoritmiek

19 juni 2008, 14.00-17.00

- Het gebruik van boek, aantekeningen of rekenmachine 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 beide ≥ 5.0).
- Wat betreft de meerkeuzevragen:
  - Er is voor iedere vraag telkens maar een 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	7	9	8	30

  - Geef antwoord in correct Nederlands of Engels en schrijf leesbaar (gebruik eerst kladpapier).
  - Geef geen irrelevante informatie. Dit kan leiden tot puntenaftrek. Het aangegeven maximaal aantal regels is van toepassing op (getypte) regels en aangepast aan de grootte van je handschrift en het gebruik van witruimte.
  - 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: 6.

Meerkeuzevragen

1. Wat is het belangrijkste argument voor het feit dat het algoritme van Gale en Shapley altijd een stabiele matching oplevert?
   A. Er bestaat altijd een stabiele matching.
   B. Er zijn geen twee mannen die precies dezelfde voorkeur hebben.
   C. Een man doet voorstellen aan vrouwen in aflopende volgorde van zijn voorkeuren en vrouwen accepteren alleen betere voorstellen.
   D. Er is geen man die liever aan een andere vrouw gekoppeld zou worden dan de vrouw aan wie hij door Gale en Shapley gekoppeld wordt.

2. Welke van de volgende beweringen is waar?
   A. 2^n is $O\left(n^{2000}\right)$.
   B. $\frac{1}{2}n^2$ is $\Theta\left(\frac{1}{3}n^3\right)$.
   C. $n^{1.5}$ is $\Omega\left(n \log n\right)$.
   D. $2n \log_2 n$ is $O\left(\log_4 n\right)$.

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. Laat een gerichte (directed) graaf G met $n \geq 3$ knopen (nodes) gegeven zijn. Welke bewering is correct?
   A. Als G een cycle heeft, dan is G sterk verbonden.
   B. Als G geen topologische volgorde heeft, dan bevat G precies één cycle.
   C. G is acyclisch dan en slechts dan als G een topologische volgorde heeft.
   D. G is acyclisch dan en slechts dan als G strikt minder dan n kanten heeft.

5. Gegeven een verbonden graaf G, een knoop s en een knoop t. Welk algoritme is ***ongeschikt*** om het minimale aantal kanten te bepalen van een pad van s naar t?
   A. Dijkstra
   B. Bellman-Ford
   C. Ford-Fulkerson
   D. Breadth-first search

6. Gegeven een graaf G met unieke gewichten op iedere kant (edge). Gegeven ook een kant e. Welk van de volgende algoritmen levert ***geen*** minimale opspannende boom (minimum spanning tree) op?
   A. Bouw een boom T volgens het algoritme van Prim.
   B. Bouw een boom T door telkens de kant (u, v) met $u \in T$ en $v \notin T$ met het laagste gewicht toe te voegen.
   C. Bouw een boom T door de kanten in oplopende volgorde van hun gewicht te beschouwen, en voeg alleen die kanten toe aan T die voldoen aan de *cut property* in T.
   D. Bouw een boom T door de kanten in oplopende volgorde van hun gewicht te beschouwen, en voeg alleen die kanten toe aan T die voldoen aan de *cycle property* in T.

7. In het algoritme van Huffman wordt een binaire prefixcodering gerepresenteerd door een boom. Welke strategie volgt Huffman om een *optimale* prefixboom te maken?

   A. Hij plaatst eerst de meest voorkomende letter bovenin de boom.

   B. Hij plaatst eerst de twee minst voorkomende letters onderin de boom.

   C. Hij plaatst recursief de minst voorkomende letters onderin de boom en de meest voorkomende letters bovenin de boom.

   D. Hij construeert een gebalanceerde boom, waarbij de som van de frequenties van de letters in het linkerdeel zoveel mogelijk gelijk is aan die in het rechterdeel.

8. Gegeven een volledig gevulde cache en de lijst van items die in het verleden opgevraagd zijn, maar ook die in de toekomst opgevraagd zullen worden. Als een item d_i opgeslagen moet worden in de cache, welk item uit de cache kan dan het best verwijderd worden om het aantal cache-misses in de toekomst te minimaliseren?

   A. Het item dat het kortst geleden gebruikt is.

   B. Het item dat het langst geleden gebruikt is.

   C. Het item dat het minst gebruikt gaat worden.

   D. Het item dat het laatst gebruikt gaat worden.

9. Laat een verzameling van n punten in het (twee-dimensionale) vlak gegeven zijn. Dan vindt het "Closest-Pair-Of-Points" algoritme de twee punten die het dichtste bij elkaar liggen. Het belangrijkste deel van dit efficiënte "Divide and Conquer" 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. Voor je verjaardag heb je een verlanglijst gemaakt van n kado's met ieder een prijs p_i. Je hebt echter geen enkele van deze kado's gekregen, maar wel een bedrag P (bestaande uit een aantal waardebonnen en wat geld). Je wilt uitzoeken welke deelverzameling van de kado's je nu kunt gaan kopen met een zo hoog mogelijk totaal bedrag (maar uiteraard niet meer dan P). Dit probleem kan efficiënt opgelost worden met behulp van dynamisch programmeren. Daartoe stellen we eerst een recursieve functie op. Welk van de volgende functie beschrijvingen voor OPT drukt uit welk bedrag je maximaal kwijt kunt?
    Als $i = 0$ dan $\mathsf{OPT}(i,p) = 0$, als $p_i > p$ dan $\mathsf{OPT}(i,p) = \mathsf{OPT}(i-1,p)$ en anders

   A. $\mathsf{OPT}(i,p) = \max_{1 \le j \le i} \{\mathsf{OPT}(j, p - p_j) + p_j\}$

   B. $\mathsf{OPT}(i,p) = \max \{\mathsf{OPT}(i-1,p),\ \mathsf{OPT}\,(i-1, p-p_i) + p_i\}$

   C. $\mathsf{OPT}(i,p) = \max_{1 \le j \le i} \{\mathsf{OPT}(i-j, p-p_j) + \mathsf{OPT}(j,p_j) + p_j\}$

   D. $\mathsf{OPT}(i,p) = \max \{\max_{1 \le j \le i} \{\mathsf{OPT}(j, p-p_j) + p_j\},\ \mathsf{OPT}\,(i-1,p)\}$

11. Het probleem van het vinden van de ruimtelijke structuur van een RNA-molecuul (RNA Secondary Structure) luidt als volgt: gegeven een rij basen $B = b_1, \ldots, b_n$ waarbij iedere $b_i \in \{A, U, C, G\}$, vind een zo groot mogelijke verzameling S van paren (i, j), waarbij $i, j \in \{1, \ldots, n\}$, die aan de volgende voorwaarden voldoet:

    1. Geen scherpe bochten: als $(i, j) \in S$ dan $|j - i| > 4$.
    2. Ieder paar $(i, j) \in S$ bestaat ofwel uit een A en een U, ofwel uit een C en een G.
    3. Geen enkele base b_i voor $i \in \{1, \ldots, n\}$ komt in meer dan een paar in S voor.
    4. Geen kruisingen: Als (i, j) en (k, l) twee paren in S zijn, dan mag het niet zo zijn dat $i < k < j < l$.

    Als een paar (i, j) aan deze voorwaarden voldoet, schrijven we $p(i, j)$. Dit probleem kan efficiënt opgelost worden met behulp van dynamisch programmeren. Daartoe stellen we eerst een recursieve functie op. Welke recursieve functie OPT geeft het grootste aantal paren in de rij die aan elkaar gekoppeld kunnen worden? Als $i \geq j - 4$ dan $\mathsf{OPT}(i, j) = 0$ en verder:

    A. $\mathsf{OPT}(i, j) = \max_t (1 + \mathsf{OPT}(i, t - 1) + \mathsf{OPT}(t + 1, j - 1))$
    B. $\mathsf{OPT}(i, j) = \max(\mathsf{OPT}(i, j - 1), \max_t (1 + \mathsf{OPT}(t + 1, j - 1)))$
    C. $\mathsf{OPT}(i, j) = \max(\mathsf{OPT}(i, j - 1), \max_t (1 + \mathsf{OPT}(i, t - 1) + \mathsf{OPT}(t + 1, j - 1)))$
    D. $\mathsf{OPT}(i, j) = \max(\mathsf{OPT}(i - 1, j), \mathsf{OPT}(i, j - 1), \max_t (1 + \mathsf{OPT}(t + 1, j - 1)))$

    waarbij $\max_t$ het maximum neemt over alle basen t waarvoor geldt $i \leq t < j - 4$ en $p(t, j)$.

12. We willen een serie punten $(x_1, y_1), \ldots, (x_n, y_n)$ gesorteerd op x-coördinaat benaderen met rechte lijnstukken. De (minimale) fout $e_{i,j}$ die wordt verkregen als de punten i tot en met j worden benaderd met één lijnstuk is gegeven voor iedere waarde van i en $j \geq i$. We definiëren C als de kosten voor het introduceren van een extra lijnstuk. De minimale kosten voor het benaderen van de punten 1 tot n met behulp van lijnstukken kunnen dan efficiënt berekend worden met behulp van dynamisch programmeren. Wat is de meest krappe (tight) worst-case grens voor de looptijd van dit algoritme?

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

13. Welke van de volgende beweringen is geldig?

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

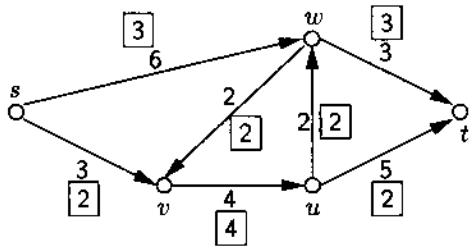
14. Bekijk het stroom netwerk in figuur 1 en teken de restgraaf (residual graph). Wat is een mogelijk stroomverhogend pad (augmenting path)?

    A. $s - w - t$

    B. $s - v - u - t$

    C. $s - w - u - t$

    D. Geen van deze paden is een stroomverhogend pad.



Figuur 1: Een stroomnetwerk. De getallen bij de kanten zijn de capaciteiten. De stroom op een kant is weergegeven in een vierkantje.

15. Bekijk nogmaals het stroomnetwerk in figuur 1. Wat is de capaciteit van de minimale snede?

    A. 5

    B. 6

    C. 7

    D. 8

Open vragen

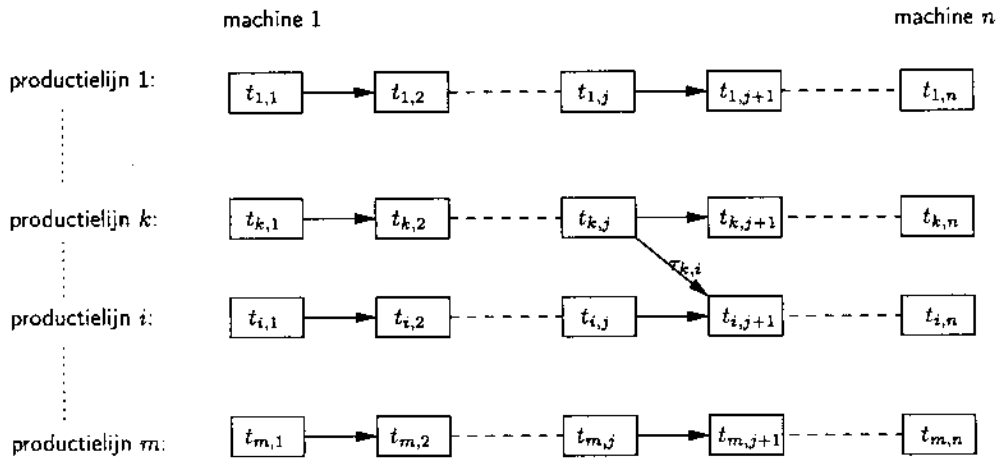
1. Geef een asymptotische boven- en ondergrens voor $T(n)$ in ieder van de volgende recurrente betrekkingen. Maak je grenzen zo krap (tight) mogelijk en geef aan hoe je aan je antwoord komt (in maximaal 5 regels per recurrente betrekking).
   (a) (3 punten) $T(n) = 3T(n/2) + n^2$
   (b) (3 punten) $T(n) = 8T(n/2) + n^3$

2. De komende zomervakantie heb je een bijbaantje bij een klein ICT bedrijf dat als volgt te werk gaat bij ieder project i. De manager maakt eerst het ontwerp in m_i dagen en direct daarna worden er werkkrachten ingehuurd om het project af te ronden in nog eens w_i dagen. Vanaf vandaag (dag 0) zijn er nog n projecten te doen. Het is jouw opdracht om het optimale rooster S^* voor de manager te maken zodat zo snel mogelijk alle projecten afgerond zijn.
   (a) (2 punten) Beschouw het voorbeeld van drie projecten (dus $n = 3$) in de onderstaande tabel. Het rooster waarbij de manager de projecten ontwerpt in volgorde van projectnummer leidt tot een eindtijd van 6 voor project 1, 17 voor project 2 en 25 voor project 3. Dus pas op dag 25 zijn alle projecten afgerond. Laat zien in welke volgorde deze 3 projecten het snelste zijn afgerond (in maximaal 3 regels).

project i :	m_i	w_i
1	2	4
2	5	10
3	11	7

   (b) (2 punten) Laat s_i de startdag zijn waarop de manager aan project i begint. Geef een formule voor de (vroegste) eindtijd f_i van een project i en druk de functie die we willen optimaliseren daarin uit (in maximaal 3 regels).
   (c) (3 punten) Schrijf een algoritme dat het optimale rooster S^* berekent voor de manager en geef een krappe bovengrens op de looptijd aan (in maximaal 5 regels).
   (d) (0 punten) (3 bonus punten) Bewijs met behulp van een exchange argument dat je algoritme uit de voorgaande vraag correct is (in maximaal 15 regels). Ga er voor het gemak van uit dat alle gegeven tijden uniek zijn.

3. Een autofabriek heeft m productielijnen met ieder een serie van n machines. In ieder van de productielijnen heeft de j^{de} machine (voor alle $1 \leq j \leq n$) precies dezelfde functionaliteit. Slechts de benodigde tijd per machine kan verschillen, omdat er soms machines vervangen zijn door nieuwere modellen. De benodigde tijd voor machine j in lijn i is gegeven en noteren we met $t_{i,j}$. Omdat de functionaliteit van de machines gelijk is, is het mogelijk om een auto na een machinehandeling van lijn k over te zetten op een andere lijn i. Dit kost $\tau_{k,i}$ tijd. Als $i = k$ dan $\tau_{k,i} = 0$. Het management wil graag weten wat de minimale doorlooptijd voor de productie van een auto is waarbij *alle* productielijnen gebruikt mogen worden. Figuur 2 (z.o.z.) geeft schematisch de belangrijkste elementen van dit probleem weer.



Figuur 2: In dit schema zijn alle productielijnen, machines en machinetijden $t_{i,j}$ weergegeven. Daarnaast zijn er tussen alle machines j en hun opvolgers $j+1$ in alle andere productielijnen nog overzettijden $\tau_{k,i}$ gegeven, maar hiervan zijn de meeste weggelaten (omdat dat er veel te veel zouden zijn).

(a) (1 punt) Laat een functie OPT (i, j) gegeven zijn die de minimale doorlooptijd geeft voor de eerste j machines waarbij als laatste machine j van lijn i wordt gebruikt. Laat zien hoe de minimale doorlooptijd door alle n machines met gebruik van alle lijnen in deze functie uitgedrukt kan worden (in maximaal 3 regels).

(b) (3 punten) Geef een recursieve definitie voor OPT(i, j) (in maximaal 5 regels).

(c) (3 punten) Geef een efficiënt iteratief algoritme met behulp van dynamisch programmeren dat de minimale doorlooptijd berekent (in maximaal 10 regels).

(d) (1 punt) Leg uit hoe je algoritme aangepast kan worden zodat na afloop ook het optimale traject gegeven kan worden (in maximaal 5 regels).

(e) (1 punt) Geef aan wat de looptijd is van je algoritme en hoe je hieraan komt (in maximaal 3 regels).

4. Je vriendin zit in het weekend meestal bij jou thuis en ze klaagt erover dat ze erg moe wordt van het heen en weer sjouwen van haar garderobe. Je biedt haar daarom aan om een deel van haar kleding bij jou in de kast te hangen. Ze komt er echter niet uit welke kleding ze voortaan waar wil opbergen.

   Voor ieder kledingstuk i heeft ze een waarde a_i voor het bij haar thuis houden (voor gebruik door-de-weeks) en een waarde b_i voor het opbergen van i bij jou (voor in het weekend). Bovendien kent ze per combinatie van twee kledingstukken i en j een getal w_{ij} toe dat uitdrukt hoeveel waarde ze eraan hecht dat die betreffende kledingstukken op dezelfde plaats zijn (omdat ze goed bij elkaar passen). De vraag is nu hoe ze de kledingstukken op zo'n manier in twee verzamelingen A en B kan opdelen dat de waarde van die opdeling (partition) maximaal is. Je gaat haar helpen door dit probleem te vertalen naar een stroomnetwerk (network flow) probleem.

   (a) (1 punt) Schrijf de som op van de waarden die je wil maximaliseren voor de opdeling in A en B (in maximaal 3 regels).

   (b) (3 punten) Door je te realiseren dat er een constante bovengrens is voor dit probleem, kun je inzien dat het *maximaliseren* van de som uit (a) overeenkomt met het *minimaliseren* van de volgende som: $\sum_{i \in B} a_i + \sum_{j \in A} b_j + \sum_{i \in A, j \in B} w_{ij}$. Gebruik deze eigenschap om het probleem te modelleren door middel van een stroomnetwerk (flow network) (in maximaal 5 regels).

   (c) (4 punten) Leg uit hoe je met behulp van dit stroomnetwerk nu efficiënt kunt bepalen wat de ideale opdeling in A en B is en waarom dit een geldige methode is (in maximaal 10 regels).

Einde tentamenopgaven