TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Tentamen IN2505-A en TI2300 Algoritmiek

31 januari 2012, 14.00-17.00

- Het gebruik van boek, aantekeningen of rekenmachine tijdens dit tentamen is ***niet*** toegestaan.
- Schrijf tenminste op je eerste antwoordvel en op het formulier voor de multiple choice de vakcode van het vak waarvoor je tentamen wil doen: IN2505-A (oud) of TI2300 (nieuw).
- 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 ≥ 5.8).
- Als je voor de huiswerkopgaven van dit collegejaar gemiddeld minstens een 5.8 hebt, dan wordt je cijfer voor dit tentamen met 1.0 verhoogd (maar nooit hoger dan 10.0).
- 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. Als je geen enkel antwoord geeft, wordt dit altijd fout gerekend.
  - 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.
- Voor de open vragen heb je ongeveer twee uur nodig:
  - 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 wordt dus 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.
  - Als er wordt gevraagd om pseudocode, dan mag je daarin algoritmen uit het boek aanroepen, tenzij anders vermeld.
  - Voordat je je antwoorden inlevert, controleer of op ieder blaadje je naam en studienummer staat en geef de vakcode en het aantal ingeleverde bladen voor de open vragen aan op (tenminste) de eerste pagina.
- De tentamenstof bestaat uit Chapters 4–7 uit Algorithm Design van Kleinberg en Tardos (behalve secties 4.4, 4.9*, 5.6, 6.8–6.10*, 7.4*, en 7.13*), en bovendien de notities over de Master Method en over bewijstechnieken (Proving guide).
- Totaal aantal pagina's (zonder dit voorblad): 7.

Meerkeuzevragen

1. Gegeven een verzameling van n open intervallen (starttijd, eindtijd) in $\mathbb{R}^+$. In welke volgorde zou je de intervallen aflopen om efficiënt te kunnen bepalen wat het grootste aantal niet-overlappende intervallen is (interval scheduling)?

   A. In de volgorde van de invoer

   B. Sorteer aflopend op eindtijd (Latest finishing time first)

   C. Sorteer oplopend op starttijd (Earliest starting time first)

   D. Sorteer oplopend op eindtijd (Earliest finishing time first)

2. Laat een Huffman codering van een alfabet S met unieke frequenties gegeven zijn. In deze codering wordt er geen onderscheid gemaakt tussen het karakter c-cédille (ç) en c. In een nieuwe versie van de codering worden deze letters wel apart gecodeerd. Als c de letter is met de laagste frequentie in S, wat weten we dan over de Huffman codering van het uitgebreide alfabet $(S \cup \{ç\})$?

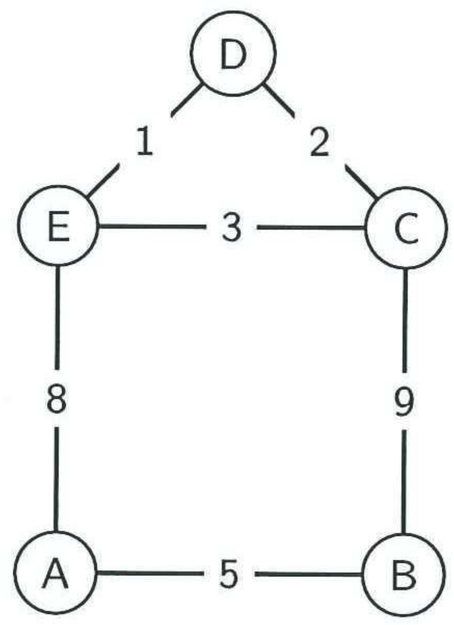
   A. Het is geen prefix codering meer.

   B. Voor alle andere letters verandert de codering.

   C. Voor geen van de andere letters verandert de codering.

   D. Voor enkele van de andere letters verandert de codering.

3. Beschouw de graaf $G = (V, E)$ in Figuur 1. Wat weten we over een minimale opspannende boom (MST)?



Figuur 1: Graaf G

   A. De kanten (C,E) en (A,E) komen in iedere minimale opspannende boom voor.

   B. De kanten (C,E) en (A,E) komen in geen enkele minimale opspannende boom voor.

   C. De kant (C,E) komt in iedere minimale opspannende boom voor en de kant (A,E) in geen enkele.

   D. De kant (A,E) komt in iedere minimale opspannende boom voor en de kant (C,E) in geen enkele.

4. Welke van de volgende beweringen over de efficiënte implementatie met behulp van pointers van de Union-Find data structuur van n elementen is ***on*waar**?

   A. Met padcompressie kost een `Find` $O(1)$.

   B. Met padcompressie kost een `Union` $O(1)$.

   C. Zonder padcompressie kost een `Union` $O(1)$.

   D. Zonder padcompressie kost een `Find` $O(\log n)$.

5. Het algoritme (van Karatsuba en Ofman) om in minder dan kwadratische tijd twee integers te vermenigvuldigen van n bits is gebaseerd op een herschrijving van de vermenigvuldiging en een recursieve aanroep van de vermenigvuldiging van integers bestaande uit minder bits. Wat is de krapste bovengrens op de looptijd van dit algoritme? (Hint: gebruik de master methode.)

   A. $O\left(n \log n\right)$
   B. $O\left(n \log^2 n\right)$
   C. $O\left(n^{\log_2 3}\right)$
   D. $O\left(n^{\log_3 2}\right)$

6. Laat een recursief algoritme gegeven zijn met een looptijd beschreven door de recurrente betrekking $T(n) = T(n-1) + n$. Wat is de krapste bovengrens op de looptijd van dit algoritme?

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

7. Gegeven een lijn L, leveren de twee recursieve aanroepen voor het bepalen van de twee dichtstbijzijnde punten in het platte vlak (het Closest-Pair-Of-Points algoritme) de kleinste afstand tussen twee punten ofwel links ofwel rechts van L op. Het minimum hiervan noemen we δ. Wat gebeurt er vervolgens voor ieder punt p binnen een afstand δ van L om efficiënt de twee dichtstbijzijnde punten in het *hele* vlak te vinden?

   A. Het algoritme berekent de kleinste afstand van p tot een constant aantal punten niet verder dan δ van L, in volgorde van y coördinaat.
   B. Het algoritme berekent de kleinste afstand van p tot alle punten met een afstand van minder dan δ van p, in volgorde van y coördinaat.
   C. Het algoritme berekent de kleinste afstand van p tot alle punten aan de andere kant van L en niet verder dan δ van L, in volgorde van y coördinaat.
   D. Het algoritme sorteert alle punten niet verder dan δ van L op y coordinaat en berekent dan de kleinste afstand van p tot de eerste ïan de gesorteerde lijst.

8. Beschouw het volgende algoritme voor het bepalen van de maximale som niet groter dan W van een deelverzameling van de getallen $\{w_1, w_2, \ldots, w_m\}$ (Subset Sum). Wat zijn correcte grenzen op de gebruikte hoeveelheid geheugen (space) en looptijd (run time)?

```
for j ← 1 to n do
    M[j, 0] ← 0
end
for w ← 0 to W do
    M[0, w] ← 0
end
for j ← 1 to n do
    for w ← 1 to W do
        if w_j ≤ w then
            M[j, w] ← max(M[j − 1, w], w_j + M[j − 1, w − w_j])
        end
        else
            M[j, w] ← M[j − 1, w]
        end
    end
end
print M[n, W]
```

A. beide $O\left(n^2\right)$

B. beide $\Theta\left(nW\right)$

C. $O\left(W\right)$ geheugen en $O\left(nW\right)$ looptijd

D. $\Omega\left(nW\right)$ geheugen en $\Omega\left(n^2W\right)$ looptijd

9. Het probleem van het vinden van de ruimtelijke structuur van een RNA-molecuul (RNA Secondary Structure) is als volgt: gegeven een string basen $B = b_1 \ldots b_n$ vind een zo groot mogelijke verzameling S van paren (i, j) waarbij $i, j \in \{1, \ldots, n\}$, die aan de volgende voorwaarden voldoen:

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

   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. $O\left(nW\right)$

   B. $O\left(n^2\right)$

   C. $O\left(n^2 \log n\right)$

   D. $O(n^3)$

10. Voor het berekenen van de afstand tussen twee strings X en Y van lengte m en n, zoeken we naar een alignment met de kleinste boete. Een alignment is een verzameling van koppels (i, j) waarbij $i \in \{1, \ldots, m\}$ en $j \in \{1, \ldots, n\}$, iedere i en iedere j hooguit eenmaal voorkomt en er geen kruisingen zijn (dus als $(i, j), (i', j') \in M$ en $i < i'$ dan $j < j'$). De boete voor koppelingen 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 $\alpha_{i,j} \geq 0$ voor iedere $(i, j) \in M$ waarvoor karakter i in X ongelijk is aan karakter j in Y, en waar $\alpha_{i,j} = 0$ als karakter i in X gelijk is aan karakter j in Y.

Met welke van de volgende formules voor OPT kun je (door middel van dynamisch programmeren) efficiënt bepalen wat de boete is voor de beste alignment van X tot en met positie i met Y tot en met positie j?

$\mathsf{OPT}(0,j) = j\delta$ en $\mathsf{OPT}(i,0) = i\delta$ en voor $i,j > 0$:

A. $\mathsf{OPT}(i,j) = \min\{\alpha_{i,j} + \mathsf{OPT}(i-1,j-1),\ \delta + \mathsf{OPT}(i-1,j-1)\}$
B. $\mathsf{OPT}(i,j) = \min_{i \le t \le j}\{\alpha_{t,t} + \mathsf{OPT}(t,j-t),\ \delta + \mathsf{OPT}(t,j),\ \delta + \mathsf{OPT}(t,j-t)\}$
C. $\mathsf{OPT}(i,j) = \min\{\alpha_{i,j} + \mathsf{OPT}(i-1,j-1),\ \delta + \mathsf{OPT}(i-1,j),\ \delta + \mathsf{OPT}(i,j-1)\}$
D. $\mathsf{OPT}(i,j) = \min_{1 \le t_1 \le i}\{\min_{1 \le t_2 \le j}\{\alpha_{t_1,t_2} + \mathsf{OPT}(t_1,t_2),\ \delta + \mathsf{OPT}(t_1,j),\ \delta + \mathsf{OPT}(i,t_2)\}\}$

11. Laat n taken gegeven zijn met voor iedere taak j een waarde v_j, en een (vaste) begin en eindtijd, respectievelijk s_j en f_j. Geef een recursieve functie voor het bepalen van de grootste waarde van een deelverzameling van niet-overlappende taken. Je mag gebruik maken van een predecessor functie $p(j)$ die voor taak j de niet-overlappende taak geeft voorafgaand aan j met de laatst mogelijke eindtijd. Gegeven is ook $\mathsf{OPT}(0) = 0$ en verder:

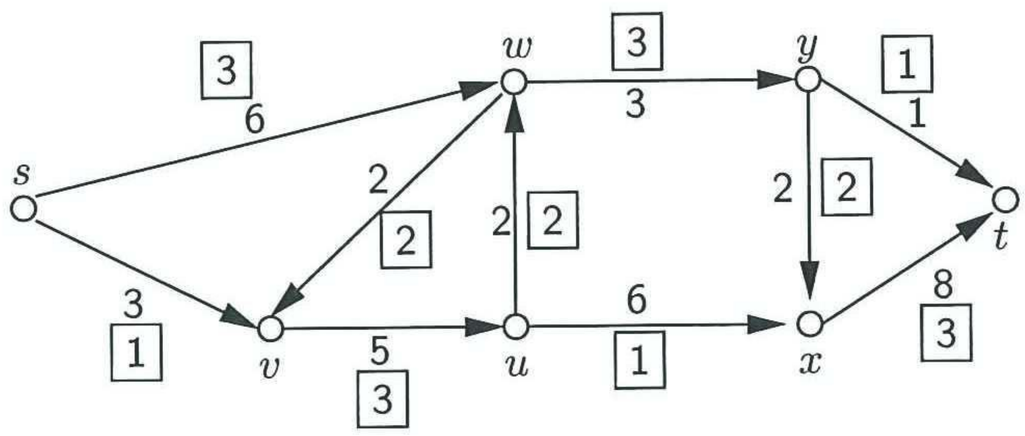
A. $\mathsf{OPT}(j) = \max_{1 \le i < j}\{v_i + \mathsf{OPT}(p(i))\}$
B. $\mathsf{OPT}(j) = \max_{1 \le i < j}\{v_j + \mathsf{OPT}(p(i))\}$
C. $\mathsf{OPT}(j) = \max\{v_j + \mathsf{OPT}(p(j)),\ \mathsf{OPT}(j-1)\}$
D. $\mathsf{OPT}(j) = \max\{v_j + \mathsf{OPT}(j-1),\ \mathsf{OPT}(p(j-1))\}$

12. Laat een stroom (flow) f en een s-t snede (A,B) gegeven zijn. Welke bewering is correct?

A. De waarde van de stroom f is nooit meer dan de capaciteit van de snede (A,B).
B. De waarde van de stroom f is nooit minder dan de capaciteit van de snede (A,B).
C. Als f een maximale stroom is dan is de capaciteit van (A,B) gelijk aan de waarde van f.
D. Als (A,B) een de minimale snede is (min-cut) dan is de waarde van f gelijk aan de capaciteit van de snede (A,B).

13. Bekijk het stroomnetwerk in figuur 2 en teken de restgraaf (residual graph). Met hoeveel kan de stroom hier nog verhoogd worden door het vinden van stroomverhogende paden (augmenting paths) in de restgraaf?

A. met 2
B. met 3
C. met 4
D. de stroom kan hier niet meer verhoogd worden



Figuur 2: Een stroomnetwerk. De getallen bij de kanten zijn de capaciteiten. De stroom op een kant is weergegeven in een vierkantje.

14. Bekijk nogmaals het stroomnetwerk in figuur 2 en je getekende restgraaf. Wat is een minimale snede?

A. $(\{s\}, \{u,v,w,x,y,t\})$
B. $(\{s,u,v,w\}, \{x,y,t\})$
C. $(\{s,v,w,y\}, \{u,x,t\})$
D. $(\{s,u,v,w,x,y\}, \{t\})$

15. Gegeven een stroomnetwerk met n knopen en m kanten met geheeltallige capaciteiten kleiner dan of gelijk aan c^* (let op: het boek gebruikt $C = \sum_{e \text{ out of } s} c_e$). Van het algoritme van Ford-Fulkerson bestaat ook een variant die een stuk efficiënter kan zijn door het gebruik van een schaling (scaling) techniek met een factor Δ. Wat is de krapste bovengrens op de looptijd van het algoritme met en zonder schaling?

    A. Zonder schaling is $O(mc^*)$, met schaling is $O(m \log c^*)$.

    B. Zonder schaling is $O(mc^*)$, met schaling is $O(m^2 \log c^*)$.

    C. Zonder schaling is $O(mnc^*)$, met schaling is $O(m \log c^*)$.

    D. Zonder schaling is $O(mnc^*)$, met schaling is $O(m^2 \log c^*)$.

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) $T(n) = 8T(n/2) + n^3\sqrt{n}$
   (b) (3 punten) $T(n) = 5T(n/5) + n$

2. Een vak in de masterfase wordt getoetst met een mondeling direct gevolgd door een kleine schriftelijke opdracht. Beide opdrachten zijn individueel, maar de schriftelijke opdracht kan wel door meerdere studenten tegelijkertijd gemaakt worden, terwijl dat bij het mondeling niet kan. Hoewel er een maximum tijdsduur voor beide is, wordt dit over het algemeen in de praktijk niet gehaald. In de loop van het vak heeft de docent een goede indruk gekregen van iedere student $i \in \{1, \ldots, n\}$ en heeft een tabel aangelegd met de verwachte duur van het mondelinge (m_i) en van het schriftelijke deel (s_i) van de toets. De docent kan pas naar huis als de laatste student ook zijn schriftelijke toets heeft afgemaakt.

   Beschouw het voorbeeld van drie studenten (dus $n = 3$) in de onderstaande tabel. Het rooster waarbij de studenten in volgorde van hun nummer worden getoetst leidt tot een eindtijd van 12 voor student 1, 34 voor student 2 en 50 voor student 3. Dus na 50 minuten zijn alle studenten klaar en kan de docent naar huis. Dit is niet de meest efficiënte volgorde. De beste volgorde is 2, 3, 1: de eindtijden zijn dan 30, 46 en 44.

student i:	m_i	s_i
1	4	8
2	10	20
3	22	14

   (a) (1 punt) Stel dat een student i op tijdstip $S[i]$ begint met zijn mondeling, wanneer is hij of zijn dan klaar met beide onderdelen? Geef een eenvoudige formule voor de tijd f_i dat een student i klaar is.
   (b) (3 punten) Geef de pseudocode voor een efficiënt algoritme dat de starttijd voor iedere student i in een array S invult zodat de docent zo snel mogelijk naar huis kan (in maximaal 8 regels). (Je mag hierbij eventueel bekende algoritmen aanroepen.)
   (c) (1 punt) Geef een krappe bovengrens op de looptijd van je algoritme.
   (d) (3 punten) Gegeven zijn de optimale starttijden $S^*[i]$ voor $1 \leq i \leq n$. Bewijs dat de starttijden $S[i]$ van je algoritme uit de voorgaande vraag nooit tot een later einde leiden met behulp van een exchange argument (in maximaal 15 regels). Ga er voor het gemak van uit dat alle gegeven tijden uniek zijn.

3. In het kader van de onderwijsverbetering worden studenten meer op hun eigen tempo onderwezen. We nemen aan dat voor ieder van de n studenten een snelheid v_i bekend is en dat de studenten $\{1, 2, \ldots, n\}$ gesorteerd zijn op deze snelheid. Het idee is om groepen te maken van studenten die min of meer dezelfde snelheid hebben. Iedere groep zal dan een aaneengesloten deelreeks van de gegeven invoer zijn. De kwaliteit van de indeling is door de opleidingsdirectie gedefiniëerd door de som van een tweetal kostenelementen per groep: de vaste (staf) kosten C en de spreiding $s(i, j)$ van de snelheid van de studenten binnen de groep $\{i, i+1, \ldots, j\}$.
   (a) (3 punten) Geef een (formule voor een) recursieve functie die de minimale kosten bepaalt voor het (optimaal) indelen van de studenten in groepen. Geef ook aan welke waarde(n) je als argument(en) meegeeft bij de eerste aanroep van je recursieve functie en beschrijf het deelprobleem dat wordt opgelost door je functie kort in woorden (in maximaal 6 regels).
   (b) (3 punten) Geef pseudocode van een efficiënt iteratief algoritme met behulp van dynamisch programmeren op basis van je functie uit a) met een looptijd polynomiaal in n dat de minimale kosten van een indeling berekent en print (in maximaal 10 regels). Je mag gebruik maken van een tabel s[][] waarin voor iedere $i \leq j$ de spreiding van groep $\{i, i+1, \ldots, j\}$ is opgeslagen op positie s[i][j]. Je mag alleen gebruik maken van opdrachten (statements) die standaard in een taal als Java beschikbaar zijn.

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

(d) (2 punten) Geef pseudocode voor een algoritme dat (na het uitvoeren van het vorige algoritme) de indices print van het begin van de groepen van de indeling met de minimale kosten (in maximaal 12 regels).

4. Een fabriek in de petrochemische industrie produceert vier tussenproducten (A,B,C en D) op basis van vier verschillende grondstoffen (a, b, c en d). Grondstof a kan omgezet worden in ieder tussenproduct, grondstof b alleen in B, C en D, grondstof c alleen in C en D en grondstof d slechts in D. Er is een probleem met de aanlevering van de grondstoffen, maar de hoeveelheden voor de komende week zijn bekend. Ook de bestellingen voor die periode zijn bekend.

(a) (4 punten) Beschrijf hoe je het algoritme van Ford-Fulkerson kunt gebruiken om uit te zoeken of de toevoer van grondstoffen voldoende is om aan alle bestellingen van tussenproducten te kunnen voldoen.

(b) (3 punten) Neem aan dat er de volgende bestellingen zijn binnen gekomen: 45 ton A, 42 ton B, 10 ton C, en 3 ton D. De beschikbare grondstoffen zijn: 50 ton a, 36 ton b, 11 ton c, en 8 ton d.

- Geef een argument op basis van de snede met minimale capaciteit (min-cut) om te bewijzen waarom niet aan alle bestellingen voldaan kan worden.
- Geef een productieplan (dat is een toewijzing van grondstoffen aan producten) zodat zoveel mogelijk ton van de bestelde tussenproducten geproduceerd kan worden.
- Geef ook een korte uitleg voor de manager van de fabriek wat er aan de hand is (zonder de woorden stroom, snede en capaciteit).

Einde tentamenopgaven