TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Tentamen TI2300 Algoritmiek

8 april 2013, 14.00-17.00

- Het gebruik van boek, aantekeningen of rekenmachine tijdens dit tentamen is ***niet*** toegestaan.
- Dit tentamen heeft 15 meerkeuzevragen goed voor 3/10 van je cijfer en 4 open vragen in totaal goed voor 6/10 van je cijfer (onderverdeeld in 30 punten). Je krijgt sowieso 1/10 kado. Merk op dat je eindcijfer voor dit vak ook voor $\frac{1}{3}$ uit het onafgeronde gemiddelde van het practicum bestaat (mits beide ≥ 6.0).
- Voor de open vragen heb je ongeveer twee uur nodig:
  - Geef antwoord in correct Nederlands of Engels en schrijf leesbaar (gebruik eerst potlood of 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.
- 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.
- 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): 6.

Open vragen

1. Bij NedTrain kan op een dag aan n treinstellen tegelijk gewerkt worden. Echter, voorafgaand aan de reparaties, ondergaat ieder treinstel $i \in \{1, \ldots, n\}$ een uitgebreide test van t_i minuten. Er kan slechts één treinstel tegelijk getest worden door de beperkte ruimte. De reparaties aan ieder treinstel i kunnen verder parallel gedaan worden. (Hiervoor is altijd voldoende capaciteit.) Ook de duur van de reparaties aan een treinstel i is gegeven: p_i minuten.

   Het doel is nu om de starttijden s_i voor het testen van de treinstellen i zo te bepalen dat de testfase van geen enkel treinstel overlapt en alle treinstellen zo snel mogelijk klaar zijn. Bijvoorbeeld, stel dat we twee treinstellen hebben met $t_1 = 6$, $p_1 = 5$ en $t_2 = 4$ en $p_2 = 8$ en stel dat we eerst treinstel 2 doen en dan treinstel 1. In dat geval zijn de starttijden $s_2 = 0$ en $s_1 = 4$, is treinstel 2 klaar na $4 + 8 = 12$ uur en treinstel 1 na $4 + 6 + 5 = 15$ uur. Na 15 uur zijn dus alle treinen klaar.

   (a) (4 punten) Geef een (greedy) algoritme voor het bepalen van een rooster dat de laatste eindtijd minimaliseert (in maximaal 6 regels). Bereken hiertoe in het algoritme de starttijden $s_1, s_2, \ldots, s_n$.

   (b) (4 punten) Beschouw nu het volgende (andere) probleem. Er zijn een aantal taken i gegeven, met voor iedere taak i een deadline d_i en een duur t_i. Het volgende algoritme is gegeven.

```
1 Sorteer de taken oplopend op deadline d_i (hernummer de indices)
2 s_0 := 0
3 t_0 := 0
4 for j := 1 to n do
5     s_j := s_{j-1} + t_{j-1}
6     c_j := s_j + t_j
7     l_j = max{0, c_j - d_j}
8 end
```

   In dit algoritme wordt voor iedere taak j ook de eindtijd c_j en de lateness l_j berekend. Bewijs dat dit algoritme de maximale lateness, dus $\max_j l_j$ minimaliseert (in maximaal 15 regels).

   - Geef aan welke bewijstechniek je gebruikt.
   - Introduceer notatie die niet in de opgave voorkomt.
   - Maak duidelijk wanneer je een aanname doet.
   - Zorg dat iedere stap in je bewijs volgt uit voorgaande beweringen.

2. (a) (4 punten) Je hebt twee zeer grote, gesorteerde arrays met getallen. Geef een $O(\log(n))$ recursief algoritme dat de mediaan van de verzameling van de elementen van beide arrays op de uitvoer zet.[1]

   (b) (3 punten) Geef een asymptotische boven- en ondergrens voor $T(n)$ in de volgende recurrente betrekking. Maak je grenzen zo krap (tight) mogelijk en geef aan hoe je aan je antwoord komt (in maximaal 5 regels).

   $T(n) = 4T(n/2) + n^3 \log n$

[1] De mediaan van een verzameling is het middelste element van een gesorteerde lijst van alle elementen.

3. Binnenkort kun je kiezen voor een energiecontract waarbij de prijs van elektriciteit (afhankelijk van hoe hard het waait en of zonnecellen veel opleveren) varieert per kwartier. Als het verschil tussen de prijzen groot genoeg is, kun je geld verdienen door een accu te laden als de prijs laag is en de energie terug te leveren als de prijs hoog is.

   De prijzen $p(i)$ voor ieder kwartier $0 \leq i \leq n$ zijn van tevoren gegeven. De accu kan maximaal een lading bevatten die gelijk is aan W kwartier laden. Ga ervan uit dat de accu in het eerste kwartier ($i = 0$) leeg is en dat er geen verlies bij het laden en ontladen plaats vindt. We kunnen per kwartier aangeven of de accu moet laden of juist kan ontladen.

   (a) (4 punten) Geef een recursieve formule voor het bepalen van de maximale winst die behaald kan worden. Geef ook aan met welk(e) argument(en) de eerste aanroep gedaan wordt.

   (b) (3 punten) Geef pseudocode voor een efficiënt iteratief algoritme op basis van dynamisch programmeren en de functie van de vorige vraag dat de maximale winst berekent. Je mag alleen gebruik maken van opdrachten (statements) die standaard in een taal als Java beschikbaar zijn.

   (c) (1 punt) Geef de krapste bovengrens op de looptijd van je algoritme uit de vorige vraag.

4. Om in een digitale foto voorgrond van achtergrond te onderscheiden, wordt het volgende model gebruikt. Voor iedere pixel i is een waarschijnlijkheid gegeven dat deze bij de voorgrond hoort, a_i, en dat deze bij de achtergrond hoort, b_i. Verder wordt voor iedere twee pixels i en j die naast of boven elkaar liggen een boete p_{ij} gegeven als een van beide tot de voorgrond en de ander tot de achtergrond wordt gerekend. De vraag is om een partitie (A, B) van de pixels te maken zodanig dat de kwaliteit $q(A, B) = \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{i \text{ boven/naast } j,\ |A \cap \{i,j\}|=1} p_{ij}$ zo hoog mogelijk is.

   (a) (4 punten) Modelleer dit probleem door middel van een stroomnetwerk (flow network) zodat uit de minimale snede afgeleid kan worden welke pixels het beste bij de voorgrond en welke bij de achtergrond kunnen worden ingedeeld.

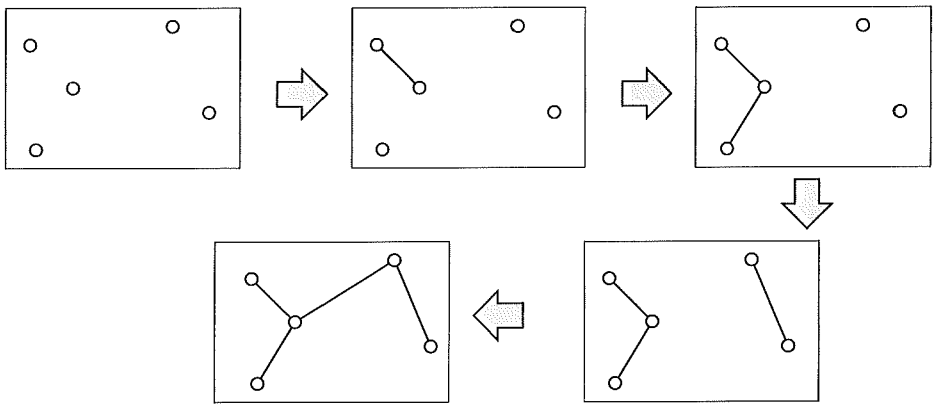
   - Beschrijf wat de knopen zijn,
   - wat de kanten zijn,
   - wat de capaciteiten op de kanten zijn en
   - hoe je de partitie van de pixels kunt maken (in maximaal 8 regels).
   - Maak ook een schets van een eenvoudig voorbeeld.

   (b) (3 punten) Bewijs dat de minimale snede (A, B) de partitie is met maximale kwaliteit $q(A, B)$ (in maximaal 5 regels).

Meerkeuzevragen

1. Welke van de volgende beweringen is waar?
   A. n^{3000} is $\Omega(2^n)$.
   B. $\frac{1}{3}n^3$ is $\Theta(\frac{2}{3}n^2)$.
   C. $n^{\frac{3}{2}}$ is $O(n \log n)$.
   D. $\log_3 n$ is $O(\log_2 n)$.

2. Beschouw deze stappen van de uitvoering van een Minimum Spanning Tree algoritme, beginnend linksboven en met de klok mee. De kosten van een kant zijn evenredig met de afstand tussen de knopen. Met welk greedy algoritme correspondeert deze uitvoer?



   A. Prim's algoritme.
   B. Kruskal's algoritme.
   C. Reverse-delete algoritme.
   D. Het voorbeeld beschrijft geen greedy algoritme.

3. Gegeven is een snede (cut) (A, B) in een graaf G met op iedere kant andere kosten. Noem e de kant in de snedeverzameling (cut set) van (A, B) met de minste kosten. Wat kunnen we nu concluderen?
   A. Voor e geldt de cycle property.
   B. Alle minimale opspannende bomen (MSTs) bevatten e.
   C. Er is precies één minimale opspannende boom die e bevat.
   D. Er is een kant f in de snedeverzameling die in geen enkele minimale opspannende boom voorkomt.

4. Op welke manier kun je efficiënt een k-clustering met maximale afstand (max spacing) vinden van een verzameling punten in twee dimensies?
   A. Op basis van het algoritme van Kruskal.
   B. Op basis van dynamisch programmeren.
   C. Op basis van het algoritme van Bellman-Ford.
   D. Op basis van het closest-pair-of-points algoritme.

5. Welke van de volgende coderingen is een *prefix* codering van minimale lengte voor de string *abccdebeeebe*?
   A. $a \to 000, b \to 10, c \to 01, d \to 001, e \to 11$
   B. $a \to 000, b \to 01, c \to 001, d \to 011, e \to 1$
   C. $a \to 000, b \to 001, c \to 010, d \to 011, e \to 1$
   D. $a \to 0001, b \to 01, c \to 001, d \to 0000, e \to 1$

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

7. Het algoritme van Karatsuba om efficient twee integers van n bits te vermenigvuldigen is gebaseerd op een herschrijving van de vermenigvuldiging en een recursieve aanroep van de vermenigvuldiging van integers bestaande uit minder bits. Hoeveel recursieve aanroepen worden er gedaan en wat is de grootte van de invoer van deze aanroepen?
   A. drie aanroepen op twee getallen van ieder $\frac{1}{2}n$ bits
   B. drie aanroepen op twee getallen van ieder $\frac{1}{3}n$ bits
   C. twee aanroepen op twee getallen van ieder $\frac{1}{2}n$ bits
   D. twee aanroepen op twee getallen van ieder $\frac{1}{3}n$ bits

8. 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 coördinaat en berekent dan de kleinste afstand van p tot de eerste van de gesorteerde lijst.

9. Bij welk soort algoritme wordt gebruik gemaakt van de oplossingen van deelproblemen?
   A. Greedy algoritmen.
   B. Dynamisch programmeren.
   C. Verdeel-en-heers algoritmen.
   D. Stroomnetwerken algoritmen.

10. 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 $\mathrm{OPT}(0) = 0$ en verder:
    A. $\mathrm{OPT}(j) = \max_{1 \le i < j} \{v_i + \mathrm{OPT}(p(i))\}$
    B. $\mathrm{OPT}(j) = \max_{1 \le i < j} \{v_j + \mathrm{OPT}(p(i))\}$
    C. $\mathrm{OPT}(j) = \max\{v_j + \mathrm{OPT}(p(j)),\ \mathrm{OPT}(j-1)\}$
    D. $\mathrm{OPT}(j) = \max\{v_j + \mathrm{OPT}(j-1),\ \mathrm{OPT}(p(j-1))\}$

11. Voor het berekenen van de afstand tussen twee strings, zoeken we naar de beste alignment: 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 $j \in \{1, \ldots, n\}$ en iedere i en iedere j hooguit eenmaal voorkomt. 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 $\alpha_{i,j} \geq 0$ 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. $\mathsf{OPT}(i, j) = \min_{i \leq t \leq j}\{\alpha_{t,t} + \mathsf{OPT}(t, j-t), \delta + \mathsf{OPT}(t, j), \delta + \mathsf{OPT}(t, j-t)\}$

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. $\mathsf{OPT}(i, j) = \min_{1 \leq t_1 \leq i}\{\min_{1 \leq t_2 \leq j}\{\alpha_{t_1,t_2} + \mathsf{OPT}(t_1, t_2), \delta + \mathsf{OPT}(t_1, j), \delta + \mathsf{OPT}(i, t_2)\}\}$

12. Het probleem van het vinden van de ruimtelijke structuur van een RNA-molecuul (RNA Secondary Structure) is 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) die aan een aantal voorwaarden voldoen. Als een paar (i, j) aan deze voorwaarden voldoet, schrijven we $p(i, j)$. Hiertoe wordt de volgende functie geimplementeerd met behulp van dynamisch programmeren.

$$\mathsf{OPT}(i, j) = \max\left(\mathsf{OPT}(i, j-1), \max_{\{t | i \leq t < j-4 \wedge p(t,j)\}}(1 + \mathsf{OPT}(i, t-1) + \mathsf{OPT}(t+1, j-1))\right)$$

Wat is de juiste asymptotische grens (bound) op de looptijd (running time) van het resulterende dynamisch programmeren algoritme?

A. $\Theta(n)$

B. $\Theta(n^2)$

C. $\Theta(n^3)$

D. $\Theta(n^2 \log n)$

13. Welke van de volgende beweringen over een stroomnetwerk is geldig?

A. De waarde van de stroom is gelijk aan de capaciteit van iedere willekeurige $s - t$ snede (cut).

B. De waarde van de maximale stroom is gelijk aan de capaciteit van de grootste $s - t$ 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 stroom is gelijk aan de netto waarde van de stroom door iedere willekeurige $s - t$ snede (cut).

14. Gegeven een stroomnetwerk $G = (V, E)$ met voor iedere kant (edge) $e \in E$ een capaciteit c_e, maar ook een vraag (demand) $d_v \in \mathbb{Z}$ voor iedere knoop (vertex) $v \in V$, zodanig dat $\sum_{v \in V} d_v = 0$. Voor een geldige stroom in G bestaat de extra voorwaarde dat de netto stroom in iedere knoop gelijk is aan de vraag van die knoop. Om te bepalen of G een geldige stroom heeft, maken we een netwerk G' als volgt. Neem netwerk G als basis voor G' en voeg daarna een super-source knoop s^* en een super-sink knoop t^* toe, en voor iedere knoop $v \in V$ met $d_v > 0$ een gerichte kant (v, t^*) met capaciteit d_v, en voor iedere knoop $v \in V$ met $d_v < 0$ een gerichte kant (s^*, v) met capaciteit $-d_v$. Wanneer bestaat er een geldige stroom in G?

    A. Er bestaat een geldige stroom in G dan en slechts dan als er een geldige stroom in G' bestaat.

    B. Er bestaat een geldige stroom in G dan en slechts dan als de maximale stroom in G' gelijk is aan de som van de vraag d_v van alle knopen $v \in V$.

    C. Er bestaat een geldige stroom in G dan en slechts dan als de maximale stroom in G' gelijk is aan de som van de vraag d_v van alle knopen $v \in V$ met $d_v > 0$.

    D. Er bestaat een geldige stroom in G dan en slechts dan als er een geldige stroom in G' bestaat die gelijk is aan de som van de vraag d_v van alle knopen $v \in V$ met $d_v > 0$.

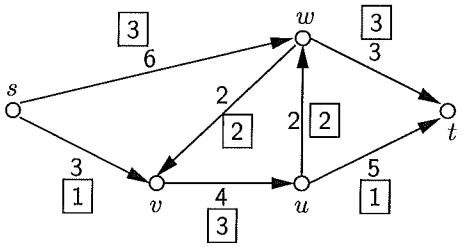
15. Bekijk het stroomnetwerk in figuur 1 en teken de restgraaf (residual graph). Met hoeveel kan de stroom hier nog verhoogd worden door het vinden van stroomverhogende paden (augmenting paths) in de restgraaf?

    A. met 1

    B. met 2

    C. met 3

    D. met 4



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

Einde tentamenopgaven