Deeltntamen 1 - Algoritmiek

8 maart 2016

Dit is het eerste deeltentamen van Algoritmiek 2015/2016. Je hebt drie uur voor dit tentamen. Schrijf duidelijk en in begrijpelijk Nederlands of Engels. Onduidelijke antwoorden kunnen fout gerekend worden. Schrijf op elk vel dat je inlevert je naam en studentnummer, en op het eerste vel het aantal bladen dat je inlevert.

Geef waar gevraagd wordt om een algoritme een duidelijke omschrijving ervan. Geef waar gevraagd wordt om pseudocode duidelijk leesbare pseudocode of C#-code.

Veel succes!

1. **Looptijden (2 punten):** (a) **Los op met de Master Theorem:** $T(n) = 5T(n/3) + \Theta(n \log n)$. Geef zowel de uitkomst, en leg uit hoe je aan deze uitkomst komt.
   (b) **Bewijs** met de Substitutiemethode dat de oplossing van $T(n) = T(n/2) + d.n$ voldoet aan $T(n) = O(n)$.

2. **A naar B (2 punten):** Gegeven gehele getallen $0 < A < B$, is een kwadratenrijtje van A naar B een rij tekens I (increment) en K (kwadraat)) zo, dat als je de operaties toepast op A, er B uitkomt. Voorbeeld: KIK is een $(2, 25)$-kwadratenrijtje, want $2^2 = 4$, $4 + 1 = 5$ en $5^2 = 25$.
   (a) **Geef een efficient greedy algoritme om het *kortste* kwadratenrijtje van A naar B te berekenen.**
   (b) **Bewijs de correctheid van je algoritme** (bijvoorbeeld door een Greedy Choice Property te bewijzen).

3. **Algoritme ontwerp (1 punt):** Gegeven is een array $A[1 \cdots n]$ met n integer waardes. Stel $A[1]$ is een even getal en $A[n]$ is een oneven getal. (a) **Geef een in O-notatie zo efficient mogelijk algoritme dat een oneven getal vindt dat direct komt na een even getal.** D.w.z., we zoeken een i, $2 \leq i \leq n$, zodat $A[i]$ oneven is, en $A[i-1]$ even is.
   (b) **Hoeveel tijd in O-notatie gebruikt je algoritme?** (Geef de tijdgrens, toelichting niet nodig.)

4. **Dynamisch Programmeren I: Opstellen recurrente betrekking (1.5 punten):** Een inbreker heeft de mogelijkheid om n kunstvoorwerpen $1, \ldots, n$ te stelen bij een inbraak. Het stelen van voorwerp i kost t_i minuten, het voorwerp i weegt w_i kilo en brengt op de zwarte markt p_i euro op. Om niet betrapt te worden, mag de inbraak niet langer dan M minuten duren, en in de vluchtauto past hoogstens K kilo aan voorwerpen. De vraag die we willen oplossen is:

   > Wat is de maximale waarde van een deelverzameling van de voorwerpen, waarbij de totale tijd van het stelen van de voorwerpen in de deelverzameling hooguit M is en het totale gewicht van de voorwerpen in de deelverzameling hooguit K is?

   Alle waardes t_i, w_i, p_i, M en K zijn niet-negatieve gehele getallen. Elk voorwerp kan hooguit één keer gekozen worden.
   Laat voor $0 \leq i \leq n$, $0 \leq t \leq M$, $0 \leq g \leq K$, $Z(i, t, g)$ de maximale waarde van een deelverzameling van kunstvoorwerpen $1, \ldots, i$ zijn met totale tijd hooguit t en totaal gewicht hooguit g. Als $i = 0$, dan is de deelverzameling leeg.
   (a) **Geef een uitdrukking voor $Z(0, t, g)$.**
   (b) **Geef een recursieve betrekking voor $Z(i, t, g)$**, die gebruikt kan worden voor het ontwerp van een algoritme gebaseerd op dynamisch programmeren. (Je hoeft bij deze opgave het algoritme zelf niet te geven.)

5. **Dynamisch programmeren II: Chained Matrix Multiplication (1.5 punten):** We hebben een serie matrices van verschillend formaat, $A_1, \ldots, A_r$. Matrix A_i is van formaat d_{i-1} bij d_i. We willen het product van de matrices uitrekenen: $A_1 \cdot A_2 \cdot A_3 \cdots A_{r-1} \cdot A_r$. Echter, in plaats van het 'klassieke' algoritme voor het vermenigvuldigen van twee matrices, gebruiken we de volgende methode: we vergelijken wat sneller is: de klassieke matrixvermenigvuldiging, of het opvullen van de matrices tot vierkante matrices, dan Strassen's algoritme gebruiken, en dan de niet-relevante posities weer weglaten.
Gegeven is een constante c_0, zodat geldt dat het vermenigvuldigen van een matrix van formaat k_0 bij k_1 met een matrix van formaat k_1 bij k_2 in het totaal $c_0 \cdot \min(k_0 \cdot k_1 \cdot k_2, (\max\{k_0, k_1, k_2\})^{2.81})$ operaties kost.
Het probleem wat we bekijken is het bepalen van een volgorde van vermenigvuldingen (het 'haakjes' zetten) zodat het totaal aantal operaties zo klein mogelijk is. Bijvoorbeeld, als $r = 3$ kunnen we het product uitrekenen als $A_1 \cdot (A_2 \cdot A_3)$ of als $(A_1 \cdot A_2) \cdot A_3$. We gaan dit probleem aanpakken met behulp van dynamisch programmeren. In deze tentamenopgave beperken we ons tot het maken van een algoritme dat het minimum aantal operaties uitrekent, gegeven een recurrente betrekking.
Definieer OPT(i,j) als het minimum aantal operaties nodig om $A_i \cdot A_{i+1} \cdots A_j$ te berekenen, $1 \leq i \leq j \leq r$. De volgende recurrente betrekking hoef je niet te bewijzen, maar mag je als gegeven beschouwen.

- Voor elke i, $1 \leq i \leq r$: OPT(i,i) $= 0$.
- Voor elke i, $1 \leq i \leq r-1$:

$$OPT(i, i+1) = c_0 \cdot \min\left\{d_{i-1} \cdot d_i \cdot d_{i+1}, (\max\left\{d_{i-1}, d_i, d_{i+1}\right\})^{2.81}\right\}$$

- Voor elke i, $1 \leq i < r-1$, en voor elke j, $i+1 < j \leq r$: $OPT(i,j) =$

$$\min_{k \in \{i,\ldots,j-1\}} \{ \; OPT(i,k) + OPT(k+1,j) + \\ + c_0 \cdot \min\{d_{i-1} \cdot d_k \cdot d_j, (\max\{d_{i-1}, d_k, d_j\})^{2.81}\}\}$$

(a) Stel $d_0, \ldots, d_n$ zijn gegeven, evenals c_0. **Geef een efficient algoritme** dat uitrekent wat het minimum aantal operaties is dat nodig is om $A_1 \cdots A_n$ te berekenen, onder de aannames als boven. Je algoritme moet dynamisch programmeren gebruiken.
(b) **Hoeveel tijd** gebruikt je algoritme (in O-notatie)? (Geef de tijdgrens, toelichting niet nodig.)

6. **Gerichte acyclische grafen en uitvindingen (2 punten):** In een computerspel kan een speler *uitvindingen* doen. Voor sommige uitvindingen geldt dat ze pas kunnen worden uitgevonden nadat sommige andere uitvindingen gedaan zijn. We hebben een verzameling uitvindingen U, en een verzamelingen P van geordende paren (u_1, u_2). Als er een paar (u_1, u_2) is, dan zal uitvinding u_2 altijd gedaan moeten worden op een later tijdstip dan uitvinding u_1 gedaan is. Gegeven is dat er geen cyclische afhankelijkheid in de uitvindingen zit, d.w.z., er is altijd een manier om alles uit te vinden.
De AI van het computerspel wil nu de volgende vraag beantwoorden. Gegeven zijn U, de verzameling paren, en een *doel-uitvinding* u^*: welke uitvindingen moet ik allemaal doen voordat ik u^* kan uitvinden?
Voorbeeld: $U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$, $P = \{(u_1, u_2), (u_1, u_4), (u_3, u_5), (u_2, u_5)$, en doeluitvinding $u^* = u_5$. Het antwoord is nu: $\{u_1, u_2, u_3\}$.
Geef een algoritme dat dit probleem oplost in tijd, lineair in het aantal uitvindingen plus het aantal paren. Leg ook uit welke datastructuur je gebruikt. Zorg dat je beschrijving van je algoritme duidelijk is; geef eventueel zowel een beschrijving in woorden als (pseudo)code.