Tentamen Complexiteit IBC028

= herkansing Analyse van Algoritmen IBC013
23 juni 2015, 8.30 - 11.30 uur
Met uitwerkingen

Dit tentamen bestaat uit vijf opgaven die alle even zwaar tellen.

Het tentamen is een gesloten-boek-tentamen, dat wil zeggen dat er tijdens het tentamen geen gebruik mag worden gemaakt van het boek en/of aantekeningen.

Voor alle vragen geldt: motiveer uw antwoord.

Opgave 1.

a. De functie T is gegeven door $T(n) = 1$ als $n \leq 2$ en $T(n) = T(\lfloor n/2 \rfloor) + T(\lfloor n/3 \rfloor) + n$ als $n > 2$. Bewijs dat $T(n) = O(n)$.

Uitwerking:

We bewijzen met inductie naar n dat $T(n) \leq 6n$. Voor $n = 1, 2$ klopt dit, voor $n > 2$ mogen we als inductiehypothese aannemen dat $T(k) \leq 6k$ voor $k < n$. In het bijzonder geldt $\lfloor n/2 \rfloor < n$ en $\lfloor n/3 \rfloor < n$, dus geldt

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T(\lfloor n/3 \rfloor) + n \\ &\leq 6\lfloor n/2 \rfloor + 6\lfloor n/3 \rfloor + n \\ &\leq 6(n/2) + 6(n/3) + n \\ &= 3n + 2n + n \\ &= 6n, \end{aligned}$$

waarmee het gevraagde is bewezen.

b. De functie T is gegeven door $T(1) = 1$ en $T(n) = 5T(\lfloor n/2 \rfloor) + n^2$ als $n > 1$. Bepaal een functie f zodanig dat $T(n) = \Theta(f(n))$.

Uitwerking:

We stellen vast dat $\log_2 5 > 2$, dus is voor $\epsilon = \log_2 5 - 2 > 0$ geldt $n^2 = O(n^{\log_2 5 - \epsilon})$. Volgens de Master stelling geldt nu $T(n) = \Theta(n^{\log_2 5})$, dus voor f gedefinieerd door $f(n) = n^{\log_2 5}$ geldt het gevraagde.

Opgave 2.

Er is een methode om twee 2×2 matrices met elkaar te vermenigvuldigen door naast een eindig aantal optellingen en aftrekkingen, 7 vermenigvuldigingen uit te voeren. Geef een algoritme dat gebruik makend hiervan het product van twee $n \times n$ matrices bepaalt en laat zien dat hiervan de complexiteit $O(n^{2,81})$ is. Hierbij hoeft n geen 2-macht te zijn; er geldt $2^{2,81} > 7$.

Uitwerking:

We geven een methode die werkt voor 2-machten; als n geen 2-macht is kiezen we de kleinste k met $n \leq 2^k$. We breiden de $n \times n$ matrices uit tot $2^k \times 2^k$ matrices door de diagonalen aan te vullen met enen en de rest met nullen. Het gevraagde product is dan het $n \times n$ stuk linksboven van het berekende product. Aangezien $2^k \leq 2n$ heeft dit geen invloed op de O van het uiteindelijke algoritme. Vanaf nu nemen we aan dat n een 2-macht is.

Het algoritme splitst beide matrices op in vier $n/2 \times n/2$ deelmatrices. Op deze deelmatrices wordt het gegeven algoritme toegepast, waarbij de 7 vermengvuldigingen uitgevoerd worden als recursieve aanroep van het te bouwen algoritme. De complexiteit $T(n)$ hiervan voldoet aan

$$T(n) = 7T(n/2) + O(n^2);$$

de $7T(n/2)$ vanwege de 7 recursieve aanroepen, en de $O(n^2)$ omdat de rest bestaat uit eindig aantal optellingen en aftrekkingen van $n/2 \times n/2$ matrices.

Vanwege $\log_2 7 > 2$ kunnen we kiezen $\epsilon = \log_2 7 - 2 > 0$ en geldt hiervoor $n^2 = O(n^{\log_2 7 - \epsilon})$. Volgens de Master stelling geldt nu $T(n) = \Theta(n^{\log_2 7}) = O(n^{2,81})$, gebruik makend van $2^{2,81} > 7$.

Opgave 3.

Gegeven zijn n cirkels in het platte vlak, elk gegeven door hun middelpunt en straal. Geef een $O(n \log n)$ algoritme dat als uitvoer een getal y geeft waarvoor de horizontale lijn door $(0, y)$ een zo groot mogelijk aantal van deze cirkels snijdt.

Uitwerking:

Voor alle cirkels met middelpunt (x_i, y_i) en straal r_i sorteren we de getallen $y_i - r_i$ en $y_i + r_i$. Vervolgens gaan we hier met een sweep line doorheen met oplopende y-waarde. Hierbij houden we het aantal snijdende cirkels bij door te beginnen met 0, en elke keer als we een $y_i - r_i$ tegenkomen deze waarde een op te hogen en elke keer als we een $y_i + r_i$ tegenkomen deze waarde een te verminderen.

Het gevraagde algoritme wordt verkregen door in dit proces de maximale waarde van deze aantallen en de bijbehorende y-waarde bij te houden.

Het voorwerk van dit algoritme bestaat uit sorteren en is $O(n \log n)$; in het algoritme zelf kost elke stap constante tijd, wat $O(n)$ oplevert. De totale complexiteit is dus $O(n \log n)$.

Opgave 4.

a. Geef de definitie van NP.

Uitwerking:

NP is de klasse van alle talen L waarvoor er een polynomiaal algoritme A en getallen c, N bestaan zodanig dat

$$L \;=\; \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^* : |y| < N|x|^c \wedge A(x,y) = 1\}.$$

b. Laat ϕ een CNF zijn waarin alle clauses uit precies vier literals bestaan. Geef een polynomiale constructie f die ϕ omzet naar een 3-CNF $f(\phi)$ waarvoor ϕ vervulbaar is dan en slechts dan als $f(\phi)$ vervulbaar is.

Uitwerking:

Voer voor elke clause C_i in ϕ een verse variabele a_i in. Laat f de operatie zijn die elke clause $C_i = l_1 \vee l_2 \vee l_3 \vee l_4$ in ϕ vervangt door door de twee clauses $l_1 \vee l_2 \vee a_i$ en $\neg a_i \vee l_3 \vee l_4$.

Als ϕ vervulbaar is, is $f(\phi)$ dat ook, met dezelfde vervulling, aangevuld met de waarde true voor a_i als $l_3 \vee l_4$ waar is, en anders false. Omgekeerd, als $f(\phi)$ vervulbaar is, zijn voor die vervulling voor elke i de beide clauses $l_1 \vee l_2 \vee a_i$ en $\neg a_i \vee l_3 \vee l_4$ waar, waaruit volgt (met gevalsonderscheid voor a_i) dat ook de oorspronkelijke clause $l_1 \vee l_2 \vee l_3 \vee l_4$ waar is. Dit geldt voor elke i; deze vervulling is hierdoor ook een vervulling voor ϕ. Hiermee is aangetoond dat f de gevraagde eigenschap heeft.

Opgave 5.

Het *partition problem* (PART) luidt als volgt:

> Gegeven een eindige verzameling natuurlijke getallen, stel vast of deze verzameling opgesplitst kan worden in twee deelverzamelingen die dezelfde som hebben.

a. Geef aan wat er bewezen moet worden als we willen concluderen dat PART NP-compleet is, gebruikmakend van het feit dat het *subset sum problem* NP-compleet is.

Uitwerking:

We moeten laten zien dat PART in NP zit, en we moeten een polynomiale f geven zodanig dat voor elke $x \in \{0,1\}^*$ geldt

$$x \in \text{SSS} \iff f(x) \in \text{PART},$$

waarin SSS staat voor subset sum problem. Hierin zijn x en $f(x)$ coderingen van instanties van de problemen. Als we deze codering wegwerken is de eis dat f losgelaten wordt op een verzameling S en een getal t en resulteert in een verzameling, en is de eis dat voor elke S, t geldt

$$\exists A \subseteq S : \sum_{s \in A} s \;=\; t \iff \exists B \subseteq f(S,t) : \sum_{s \in B} s \;=\; \frac{1}{2} \sum_{s \in f(S,t)} s.$$

b. Geef dit bewijs. (Aanwijzing: voeg een of twee grote getallen toe aan de verzameling)

Uitwerking:

Er geldt dat PART in NP zit omdat we de codering van een van beide deelverzamelingen als certificaat y kunnen kiezen. Het checken dat de som van deze deelverzameling precies de helft is van de som van de hele verzameling kan in polynomiale tijd gedaan worden.

Voor de constructie van f is het idee dat we twee grote getallen aan S toevoegen zodanig dat S een deelverzameling heeft met som S dan en slechts dan als de uitgebreide verzameling in twee gelijke stukken opgesplitst kan worden. Laat $k = \sum_{s \in S} s$. Als $k = 2t$ hoeven we niets te doen. Anders kiezen we N een heel groot getal, bijvoorbeeld $N = 10k$, en definiëren

$$f(S,t) \;=\; S \cup \{N - k + t, N - t\}.$$

Merk op dat $\sum_{s\in f(S,t)} s = 2N$. Als er een $A \subseteq S$ is met $\sum_{s\in A} s = t$ dan kiezen we $B = A \cup \{N-t\}$ en geldt $\sum_{s\in B} s = N = \frac{1}{2}\sum_{s\in f(S,t)} s$.

Omgekeerd, als er een $B \subseteq f(S,t)$ is met $\sum_{s\in B} s = \frac{1}{2}\sum_{s\in f(S,t)} s = N$, dan dan bevat B precies een van de twee grote toegevoegde getallen $N-k+t$ en $N-t$. Als het $N-t$ is, dan kiezen we $A = B \setminus \{N-t\}$ en geldt $\sum_{s\in A} s = t$. Anders kiezen we $A' = B \setminus \{N-k+t\}$ en geldt $\sum_{s\in A'} s = k-t$. Door dan te kiezen $A = S \setminus A'$ hebben we $\sum_{s\in A} s = k-(k-t) = t$. In alle gevallen hebben we dus een $A \subseteq S$ met $\sum_{s\in A} s = t$, waarmee het gevraagde is bewezen.

Opmerking: er is ook een constructie waarbij er maar één getal wordt toegevoegd, maar dan vergt de redenering wat meer gevalsonderscheid en moet je toestaan dat eenzelfde getal ook meerdere keren voor mag komen.