F. Drewes 2006-05-08

Datavetenskapens grunder, VT06
– Gruppövning 4 –

1. Låt $f(n) = (500\sqrt{n} + n/3)^2 - 7n$. Hitta en lämplig funktion g där $f(n) = O(g(n))$. Motivera ditt val av g.

   OBS! Utelämna uppgiften om ni känner er tillräckligt förtrogna med stora ordo-begreppet.

2. Vilken tidskomplexitet har $\{uu^{\mathcal{R}} \mid u \in \{a, b\}^*\}$ på en vanlig TM (där $u^{\mathcal{R}}$ vänder på u)? Går det fortare om TM:en får ha två band istället för ett?

3. Samma uppgift som föregående fast med språket $\{uu \mid u \in \{a, b\}^*\}$.

4. En liten utmaning: Visa att P är sluten under Kleene-stjärnan.

   Tips! Det är viktigare att fundera på problemet än att lösa det. Diskutera först vad påståendet innebär och vad svårigheten är. Fundera sedan på hur problemet kan reduceras till problemet PATH. (Kom ihåg att PATH är mängden av alla $\langle G, x, y\rangle$ där G är en riktad graf och x, y är noder i G så att det finns en väg från x till y. Problemet tillhör P.)

Kommentarer...

1. Man borde komma fram till att $f(n) = O(n^2)$. I ett första steg kan $f(n)$ skrivas som $f(n) = cn + c'n\sqrt{n} + c''n^2$ där c, c', c'' är lämpliga konstanter. De första två termerna kan sedan strykas (n^2 växer snabbare än både n och $n\sqrt{n}$), så att bara $c''n^2 = O(n^2)$ finns kvar.

2. Komplexiteten är $O(n^2)$: Kolla om första symbolen är lika med sista ($O(n)$ steg), stryk dem och upprepa tills inga symboler är kvar ($O(n)$ upprepningar). Sammanlagt alltså $O(n) \cdot O(n) = O(n^2)$.

   Med två band räcker $O(n)$ om man först kopierar inputsträngen till andra bandet och sedan kollar likheten genom att läsa från vänster till höger på band ett och från höger till vänster på band två.

3. Förhållandena är samma som ovan fast det är lite svårare att se eftersom man först måste hitta strängens mitt. Det kan dock t ex göras genom att märka symbolerna. Bandets första symbol $x \in \{a, b\}$ ersätts med x' och sista symbolen $y \in \{a, b\}$ ersätts med y'' (där a', b', a'', b'' är nya symboler). Efter $O(n) \cdot O(n) = O(n^2)$ steg har mitten hittats. Ytterligare $O(n^2)$ steg behövs för att avgöra om hälfterna är identiska.

   Med två band går det igen fortare. Först kopieras inputsträngen igen. Sedan hittas mitten genom att läsa från vänster med båda huvud, där den ena alltid gör två steg till höger medan den andra bara gör ett. Om och när första huvudet når första mellanslaget står andra huvudet på fösta symbolen av andra halvan. Sedan förflyttas första huvudet tillbaka till strängens början och hälfterna jämförs stegvis.

4. Utsagans innebörd kan beskrivas så här. Vi har ett problem $A \subseteq \Sigma^*$ som tillhör P. Det finns alltså en TM M_A som avgör A på polynomtid, säg $O(n^l)$. Det som ska visas är att även A^* tillhör P. Vid input u gäller det alltså att kolla om $u = u_1 \cdots u_m$ för några $u_1, \ldots, u_m \in A^*$. Svårigheten är att vi inte känner till uppdelningen av inputsträngen u.

   För att se hur det kan lösas effektivt, antar att $u = a_1 \cdots a_n$, där $a_1, \ldots, a_n \in \Sigma$. Om lämpliga $u_1, \ldots, u_m \in A^*$ finns är var och en av dem en delsträng $u_{i,j} = a_{i+1} \cdots a_j$, där $0 \leq i < j \leq n$. Dessutom måste u_1 börja med första symbolen ($i = 0$), u_m måste sluta med sista symbolen ($j = m$) och varje u_{p+1} måste börja där u_p slutar (dvs $u_p = u_{i,j}$ implicerar $u_{p+1} = u_{j,k}$ för något k). De här kraven kan representeras som en graf G_u enligt följande:

   Noderna är $v_0, \ldots, v_n$. Intuitionen är att noden v_p representerar delsträngen av u som består av de första p symbolerna. I synnerhet representerar v_0 ε och v_n hela inputsträngen u. Det finns en kant från v_i till v_j ($0 \leq i < j \leq n$) om och endast om $u_{i,j} \in A$. Enligt det som nämndes medför det att $u \in A^*$ om och endast om $\langle G_u, 1, n \rangle \in \text{PATH}$.

   För att kolla om $u \in A^*$ kan vi alltså gå till väga så här.

(a) Konstruera grafen G_u. Detta görs genom att, för varje i, j $(0 \leq i < j \leq n)$, räkna ut $M_A(u_{i,j})$ för att få reda på om det ska finnas en kant från v_i till v_j. Eftersom det finns $O(n^2)$ par (i, j) och M_A måste exekveras för var och en av dem blir tidskomplexiteten $O(n^2 n^l) = O(n^{l+2})$.

(b) Använd TM:en M_{PATH} som avgör PATH för att avgöra om $\langle G_u, 1, n\rangle \in \text{PATH}$ och returnera resultatet. Om tidskomplexiteten av M_{PATH} är $O(n^d)$ blir komplexiteten av det här steget också $O(n^d)$ (eftersom G_u har $n + 1$ noder).

Den totala tidsåtgången är alltså $O(n^r)$ där $r = \max(l + 2, d)$.