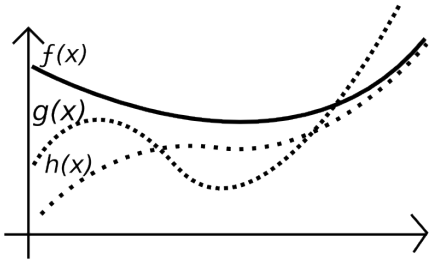
SI-möte #6, Programkonstruktion och datastrukturer
Lösningsförslag

Elias Castegren
elca7381@student.uu.se

Övningar

1.



i) $g(x) = O(f(x))$ **Falskt**

ii) $f(x) = \Theta(h(x))$ **Sant**

iii) $g(x) = \Theta(h(x))$ **Falskt**

iv) $h(x) = \Omega(f(x))$ **Sant**

v) $h(x) = O(f(x))$ **Sant**

Kommentar till uppgiften:
Kom ihåg att definitionerna av Θ, Ω och O är att det finns konstanter c_1 och c_2 så att

$$f(n) = \Theta(g(n)) \implies c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

$$f(n) = \Omega(g(n)) \implies c_1 \cdot g(n) \leq f(n)$$

$$f(n) = O(g(n)) \implies f(n) \leq c_2 \cdot g(n)$$

för n större än något n_0. Alltså, om $f(n) = \Theta(g(n))$ gäller, så gäller också att $f(n) = \Omega(g(n))$, att $f(n) = O(g(n))$ samt att $g(n) = \Theta(f(n))$.

2.

i) $3n^3 + 15n^2 + 3 = \Theta(\underline{n^3})$

ii) $\underline{1} = O(42)$ (notera att $100 = O(42)$ också stämmer)

iii) $n \cdot \log_{10} n + 3 = \Omega(\underline{n \cdot \log_{10} n})$, ($\Omega(n)$, $\Omega(\log n)$ och $\Omega(1)$ är också korrekta svar)

iv) $n^2 + 100n + 15 = O(\underline{n^2})$ (Alla $O(n^k)$, där $k \geq 2$ stämmer)

v) $n \cdot \log_2 n + n = \Theta(\underline{n \cdot \log n})$

vi) $n^2 + \log_2 n = \underline{\Omega}(n \cdot \log_2 n)$

vii) $2^n = \underline{O}(n!)$

3.

i) $T(n) = 2T(\frac{n}{2}) + n$

$a = 2, \qquad b = 2, \qquad f(n) = n$

$n^{log_2 2} = n, \qquad f(n) = \Theta(n) \implies$ **Case 2**

$T(n) = \Theta(n \cdot \log n)$

ii) $T(n) = 4T(\frac{n}{2}) + n^2 \cdot \lg n$

$a = 4, \qquad b = 2, \qquad f(n) = n^2 \cdot \lg n$

$n^{log_2 4} = n^2, \qquad \frac{n^2 \cdot \lg n}{n^2} = \lg n \not> n^\epsilon \implies$ **Master theorem ej applicerbart**

iii) $T(n) = 10T(\frac{n}{3}) + 2n^2$

$a = 10, \qquad b = 3, \qquad f(n) = 2n^2$

$n^{log_3 10} > n^2, \qquad \frac{n^{log_3 10}}{f(n)} > n^\epsilon \implies$ **Case 1**

$T(n) = \Theta(n^{log_3 10})$

iv) $T(n) = T(\frac{n}{2}) + n$

$a = 1, \qquad b = 2, \qquad f(n) = n$

$n^{log_2 1} = n^0 = 1, \qquad \frac{f(n)}{1} > n^\epsilon \implies$ **Case 3**

$T(n) = \Theta(n)$

v) $T(n) = 2^n \cdot T(\frac{n}{2}) + n^4$

$$a = 2^n \text{ inte en konstant } \implies \textbf{Master theorem ej applicerbart}$$

4.

I det värsta fallet måste man jämföra varje strumpa med alla andra strumpor innan man hittar rätt strumpa. När man till slut hittar rätt strumpa tar man förstås bort det paret ur högen (vi antar att det från början är ett jämnt antal strumpor). Antalet jämförelser $C(n)$ av n strumpor beskrivs rekursivt som:

$$C(n) = \begin{cases} 0 & \text{om } n = 0 \\ n - 1 + C(n-2) & \text{om } n > 0 \end{cases}$$

Om vi utvecklar rekursionen får vi

$$C(n) = (n-1) + (n-3) + \cdots + 3 + 1 + 0 = \left(\frac{n}{2}\right)^2 = \frac{n^2}{4}$$

Om man ska para ihop $2n$ strumpor blir antalet jämförelser $\frac{(2n)^2}{4} = n^2$, alltså fyra gånger så många som för n strumpor.

I det genomsnittliga fallet behöver man jämföra varje strumpa med hälften av de resterande strumporna. Med samma resonemang som ovan får man:

$$C(n) = \begin{cases} 0 & \text{om } n = 0 \\ \frac{n-1}{2} + C(n-2) & \text{om } n > 0 \end{cases}$$

$$C(n) = \frac{(n-1)}{2} + \frac{(n-3)}{2} + \cdots + \frac{3}{2} + \frac{1}{2} + 0 = \frac{\left(\frac{n}{2}\right)^2}{2} = \frac{\frac{n^2}{4}}{2} = \frac{n^2}{8}$$

I bästa fall plockar man upp två stumpor åt gången och har den ofantliga turen att de alltid matchar (eller så är man inte så himla petig). Man gör alltså en jämförelse per par strumpor, vilket blir totalt $\frac{n}{2}$ jämförelser för n stycken strumpor.

Sammanfattningsvis har den här metoden att sortera strumpor följande komplexitet:

$$C(n) = \begin{cases} \frac{n^2}{4} = \Theta(n^2) & \text{i värsta fall} \\ \frac{n^2}{8} = \Theta(n^2) & \text{i genomsnittliga fall} \\ \frac{n}{2} = \Theta(n) & \text{i bästa fall} \end{cases}$$

5.

```
fun f(0) = 0
  | f(n) = if 2*n>42 then f(n-1) else f(n-1)+1;
```

Låt k_1 och k_2 vara (de konstanta) tiderna för mönstermatchning, multiplikation och så vidare.

$$T(n) = \begin{cases} k_1 & n = 0 \\ k_2 + T(n-1) & n > 0 \end{cases}$$

Eftersom varje steg "kostar" k_2 och n minskar med ett varje steg är en rimlig gissning att den totala kostnaden kommer vara $n \cdot k_2 + k_1$, alltså att komplexiteten kommer vara linjär ($\Theta(n)$). Ett induktionsbevis följer nedan.

Induktionsantagande (IA): $T(p) = p \cdot k_2 + k_1$

Basfall: $T(0) = 0 \cdot k_2 + k_1 = k_1$

Induktionssteg: $T(p) = p \cdot k_2 + k_1 \implies T(p+1) = (p+1) \cdot k_2 + k_1$

Bevis: $T(p+1) \overset{rek.}{=} k_2 + T(p) \overset{IA}{=} k_2 + p \cdot k_2 + k_1 = (p+1) \cdot k_2 + k_1$ **VSV**

```
fun g(0) = 1
  | g(n) = g(n-1) + g(n-1);
```

Låt k_1 och k_2 vara (de konstanta) tiderna för mönstermatchning och addition.

$$T(n) = \begin{cases} k_1 & n = 0 \\ k_2 + 2T(n-1) & n > 0 \end{cases}$$

Om man utvecklar rekursionen får man:

$$T(n) = k_2 + 2(k_2 + 2(k_2 + \cdots + 2(k_2 + k_1)\dots) = \sum_{i=0}^{n} 2^i \cdot k_2 + 2^n \cdot k_1 = (2^n - 1) \cdot k_2 + 2^n \cdot k_1$$

Detta räcker egentligen som bevis för att $T(n) = \Theta(2^n)$. Vill man vara säker på att man har räknat rätt kan man utföra följande induktionsbevis:

Induktionsantagande (IA): $T(p) = (2^p - 1) \cdot k_2 + 2^p \cdot k_1$

Basfall: $T(0) = (2^0 - 1) \cdot k_2 + k_1 = 0 \cdot k_2 + k_1 = k1$

Induktionssteg: $T(p) = (2^p - 1) \cdot k_2 + 2^p \cdot k_1 \implies (2^{p+1} - 1) \cdot k_2 + 2^{p+1} \cdot k_1$

Bevis:

$T(p+1) \overset{rek.}{=} k_2 + 2T(p) \overset{IA}{=} k_2 + 2((2^p - 1) \cdot k_2 + 2^p \cdot k_1) =$

$k_2 + 2^{p+1} \cdot k_2 - 2k_2 + 2^{p+1} \cdot k_1 = 2^{p+1} \cdot k_2 - k_2 + 2^{p+1} \cdot k_1 =$

$(2^{p+1} - 1) \cdot k_2 + 2^{p+1} \cdot k_1$ **VSV**