SI-möte #5, Programkonstruktion och datastrukturer
Lösningsförslag

Elias Castegren & Kristiina Ausmees
elca7381@student.uu.se ‖ krau6498@student.uu.se

Övningar

1.

$i)$ Antalet strumpor avgör hur lång tid det tar att genomföra algoritmen.

$ii)$ Antag att man börjar med n stycken strumpor. Antalet förflyttningar F kan då uttryckas som

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

Basfallet är när man inte har några strumpor kvar att flytta, och då flyttar man noll strumpor. I det allmänna fallet flyttar man en strumpa, och då återstår $n - 1$ strumpor att flytta.

$iii)$ En sluten formel för rekursionen är $F(n) = n$. Börjar man med n stycken strumpor måste man utföra n stycken strumpförflyttningar. Det kan bevisas med induktion på följande sätt:

Induktionsantagande (IA): $F(p) = p$
Basfall:
$F(0) \overset{Rek.}{=} 0$
$F(0) \overset{IA}{=} 0$ **VSV**
Induktionssteg: $F(p) = p \implies F(p+1) = p + 1$
Bevis:
$F(p+1) \overset{Rek.}{=} 1 + F(p) \overset{IA}{=} 1 + p$ **VSV**

$iv)$ $F(2n) = 2n$, alltså måste man utföra dubbelt så många förflyttningar om man dubblerar antalet strumpor.

2.

$i)$ 1. Plocka upp en strumpa.
2. Jämför strumpan med en annan strumpa. Om de är lika, lägg dem båda åt sidan och gå till steg 3, upprepa annars steg 2 med en ny strumpa.
3. Finns det strumpor kvar? Om ja, gå till steg 1, annars är du klar.

$ii)$ Antalet strumpor avgör problemstorleken.

$iii)$ Antag att man har s stycken strumpor kvar. Plockar man upp en strumpa måste man i värsta fall jämföra den med alla de återstående $s-1$ strumporna innan man hittar rätt.

$iv)$ Antag att man börjar med n stycken strumpor. Antalet jämförelser C kan då uttryckas:

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

Basfallet är när man inte har några strumpor kvar att flytta, och då jämför man noll strumpor. I det allmänna fallet gör man $n-1$ jämförelser innan man lägger undan de två matchande strumporna. Då återstår $n-2$ strumpor att sortera.

$v)$

$$C(2) = 1 + C(0) = 1 + 0 = 1$$
$$C(4) = 3 + C(2) = 3 + 1 = 4$$
$$C(8) = 7 + C(6) = 7 + (5 + C(4)) = 7 + 5 + 4 = 16$$
$$C(16) = 15 + C(14) = 15 + (13 + C(12)) = 15 + (13 + (11 + C(10))) =$$
$$15 + (13 + (11 + (9 + C(8)))) = 15 + (13 + (11 + (9 + 16))) = 64$$

$vi)$ Om vi utvecklar rekursionen får vi

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

Detta kan bevisas med rekursion:

Induktionsantagande: $C(p) = \dfrac{p^2}{4}$

Basfall:

$C(0) \overset{Rek.}{=} 0$

$C(0) \overset{IA}{=} \dfrac{0^2}{4} = 0 \quad \mathbf{VSV}$

Induktionssteg: $C(p) = \dfrac{p^2}{4} \implies C(p+2) = \dfrac{(p+2)^2}{4} = \dfrac{p^2+4p+4}{4}$

Notera att vi antar att vi har ett jämnt antal strumpor

Bevis:

$$C(p+2) \overset{Rek.}{=} p + 1 + C(p) \overset{IA}{=} p + 1 + \frac{p^2}{4} = \frac{4p}{4} + \frac{4}{4} + \frac{p^2}{4} = \frac{p^2+4p+4}{4} \quad \mathbf{VSV}$$

vii) 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.

Övriga kommentarer:
I det genomsnittliga fallet behöver man bara 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. Bägge fallen kan bevisas med induktion.

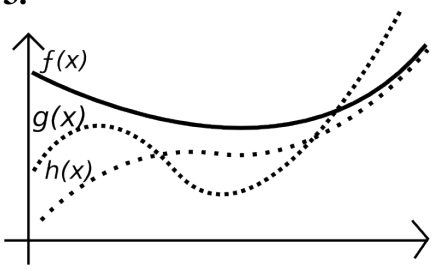
Sammanfattningsvis har den här metoden att sortera strumpor följande komplexitet:

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

Att flytta strumpor som i uppgift ett har alltid linjär komplexitet:

$$F(n) = n = \Theta(n)$$

3.



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 $f(n) = \Theta(g(n))$, så gäller också att $f(n) = \Omega(g(n))$, att $f(n) = O(g(n))$ samt att $g(n) = \Theta(f(n))$.

4.

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

5.

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

Notera att värdet funktionen beräknar inte spelar någon roll för komplexiteten! Att man multiplicerar värdet av det rekursiva anropet med 2 påverkar alltså inte analysen.

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**