Diskret Matematik og Algoritmer

Aflevering 6i (Genaflevering)

Adam Ingwersen,
GQR701

Datalogisk Institut
Københavns Universitet

November 3, 2016

Del 1

Vi betragter Fibonaccitallene og disses definition; $F_0 = 0, F_1 = 1$ og derefter:

$$F_n = F_{n-1} + F_{n-2} \quad \forall n \geq 2$$

(1)

Ved præsentation af induktionsbevis fremvises først og fremmest et basis-step, der efterviser, at $P(n_0)$ gælder. Herefter gøres en antagelse om $P(n)$. Ud fra denne antagelse forsøges det at vise, at $P(n) \implies P(n+1)$ er en tautologi - dette er sandt når implikationspilen er sand.

Basis-step

$$P(1): \quad F_2 = F_1 + F_0 = 2 \leq 2^1 = 2$$

Induktions-step

Vi gør os nu en antagelse om $P(k)$:

$$P(k): \quad F_k \leq 2^k$$

Nu skal det vises, at $P(k{+}1)$ gælder givet definitionen af Fibonacci-sekvensen defineret i Del 1:

$$P(k+1): \quad F_{k+1} = F_k + F_{k-1} \leq 2^{k+1} = 2 \cdot 2^k = F_k + Fk$$

Givet den rekursive definition af F_n gælder det, at:

$$P(k+1): \quad F_k + F_{k-1} \leq F_k + F_k \qquad \forall k \in \mathbb{Z}^+ \qquad Q.E.D.$$

Altså, er $P(k+1)$ sand. Efter princippet for matematisk induktion gælder det, at $P(n)$ er sand for alle $n \geq 1$

(2)

Basis-step

$$P(6): \quad F_6 = 8 \geq \frac{3}{2}^5 \approx 7,6$$

Base-case, $P(6)$, er sand. Nu skal det vises, at nedenstående gælder.

$$F_n \geq (\frac{3}{2})^{n-1} \quad \forall n \in \{6, 7, 8, ...\}$$

Induktions-step

Det vises, at også $P(7)$ sand:

$$P(7) : F_7 = 13 \geq \frac{3}{2}^6 \approx 11,39$$

Givet, $P(6)$ og $P(7)$ sand, antages $P(k-1)$ samt $P(k)$ sand, således at:

$$\begin{aligned}
F_{k+1} =& F_k + F_{k-1} \\
\geq& (\frac{3}{2})^{k-1} + (\frac{3}{2})^{k-2} \\
=& (\frac{3}{2})^{k-2} \cdot (\frac{3}{2} + 1) \\
>& (\frac{3}{2})^{k-2} \cdot (\frac{3}{2})^2 \\
=& (\frac{3}{2})^k \\
\Longrightarrow F_{k+1} \geq& (\frac{3}{2})^k \quad Q.E.D.
\end{aligned} \tag{1}$$

Hermed vist. Det bemærkes, at vores basis-step bryder sammen for $n < 6$ - hvorfor det gælder:

$$F_{n+1} \geq (\frac{3}{2})^n \quad \forall n \in \{6, 7, 8, ...\}$$

(3)

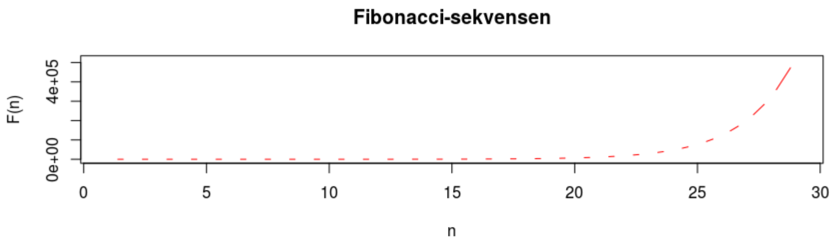
For at bestemme tids-kompleksiteten af Fibonacci-sekvensen, skal vi undersøge, hvor mange led der skal udregnes før et vilkårligt F_n er identificeret. Givet den rekursive definition af F_n, skal vi finde en funktion der tilfredsstiller:

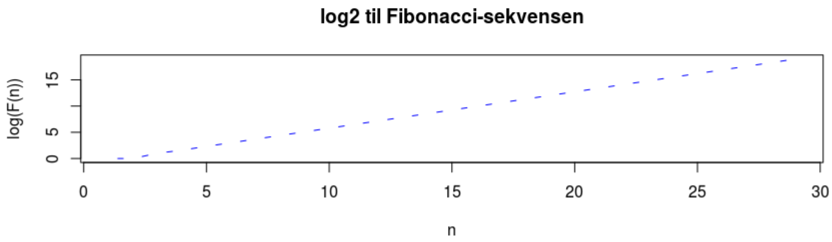
$$F_1 = 1 \quad \wedge \quad F_n = F_{n-1} + F_{n-2}$$

Vi ved, at F_n er $\Omega(\frac{3}{2}^{n-1})$, samt $O(2^n)$. Dette er F_n's lower- hhv. upper-bound. Mellem disse to funktioner findes en Θ-repræsentation af F_n. For at finde køretiden af $log_2 F_n$, tages 2-tals logaritmen til de to udtryk, hvor ulighedstegn anvendes for at indikere, at $log_2 F_n$ ligger imellem:

$$\begin{aligned}
\Omega(log_2 \frac{3}{2}^{n-1}) \leq log_2 F_n \leq O(log_2 2^n) \\
\Longrightarrow k \cdot (n-1) \leq log_2 F_n \leq n
\end{aligned} \tag{2}$$

Altså, ligger $log_2 F_n$ mellem to lineære funktioner af n, således at det asymptotisk gælder, at $log_2 F_n$ er $\Theta(n)$. Logaritmen til Fiboancci-sekvensen har konstant tids-kompleksitet, hvorved vi kan udlede, at tidskompleksiteten af Fibonacci-sekvensen må være non-lineær.





Del 2

(1)

Algoritmen MUL bestemmer hvorvidt to positive tal, $a, \quad b \in \mathbb{Z}^+$, er ligeligt divisible. Variablen y tæller antallet af gange, a kan divideres med b - eller rettere, hvor mange gange b går op i a. Det formodes, at der med illustrative eksempler menes et par repræsentative eksempler af algoritmens virken:

$$\begin{aligned} MUL(10,5) &\implies x = 10, y = 0 \to x = 10 - 5 = 5, y = 1 \to x = 5 - 5 = 0, y = 2 \\ &\implies x = 0, y = 2, RETURN = TRUE \\ MUL(11,6) &\implies x = 11, y = 0 \to x = 11 - 6 = 5, y = 1 \to ENDLOOP : x \ngeq b \\ &\implies x = 5, y = 1, RETURN = FALSE \end{aligned} \tag{3}$$

(2)

Basis-step

$$\begin{aligned} x_0 =& a \\ y_0 =& 0 \\ x_0 + b \cdot y_0 =& a \\ \implies x_0 =& a \end{aligned} \tag{4}$$

$P(0)$ er altså sand.

Induktions-step

Med udgangspunkt i $P(0)$ antager vi, at $P(k)$ er sand, sådan at:

$$P(k): \quad x_k + b \cdot y_k = a \iff x_k = a - b \cdot y_k$$

Det skal eftervises, at der efter n gennemløb af while-løkken gælder:

$$x_n + b \cdot y_n = a \tag{5}$$

For at vise det generelle tilfælde, betragtes udtrykket ved $n+1$'te gennemløb:

$$x_{n+1} + b \cdot y_{n+1} = a \tag{6}$$

Vi ved, at $y_{k+1} = y_k + 1$ pr. definition. Det ses, at der må gælde:

$$x_{n+1} = x_n - by_{n+1} = y_n + 1 \tag{7}$$

Lighederne i (7) substitueres ind i (6), hvorved vi får:

$$\begin{aligned} x_{n+1}b \cdot y_{n+1} &= x_n - b + b \cdot (y_n + 1) \\ x_{n+1}b \cdot y_{n+1} &= x_n - b + b \cdot y_n + b \\ x_{n+1}b \cdot y_{n+1} &= x_n + b \cdot y_n = a \end{aligned} \tag{8}$$

Hermed er det vist, induktivt, at $x_n + b \cdot y_n = a$ i det generelle tilfælde.

(3)

Algoritmen terminerer med FALSE/TRUE. For at dette lader sig gøre, kræves det at $b \geq x \geq 0$ i pseudokoden. Det bemærkes, at vi kan repræsentere a som $a = n \cdot b + r$, hvor $b > r \geq 0$. I udtrykket er r en rest - vi kan således repræsentere algoritmens udfald (FALSE/TRUE) vhja. r, altså to tilfælde:

$$b \mid a \implies r = 0 \quad \vee \quad b \nmid a \tag{9}$$

Lad det q'te iteration af while-løkken være den terminerende (sidste) iteration. Vi bruger, at y vokser med 1 pr. iteration, og må ved den q'te iteration være q:

$$\begin{aligned} x_q b \cdot y_q &= a \\ x_q + b \cdot y_q &= q \cdot b + r \\ x_q &= q \cdot b + r - b \cdot y_q \\ x_q &= q \cdot b + r - b \cdot q \\ x_q &= r \end{aligned} \tag{10}$$

Altså må algoritmen terminere, og derved returnere et af de to mulige udfald, FALSE/TRUE.

Bilag

```
fib  <- function(n)
{
  if (length(n) > 1) return(sapply(n, fib))
  if (n == 1) return(1)
  if (n == 2) return(1)
  return(fib(n-1)+fib(n-2))
}

n <- c(1:29)
fibn <- fib(1:29)

par(mfrow=c(2,1))
matplot(n, fibn, type = 'c',
        col = c("#ff0000"),
        ylab = "F(n)", xlab = "n")
title("Fibonacci-sekvensen")
matplot(n, log2(fibn), type = 'c',
        col = c("#0000ff"),
        ylab = "log(F(n))", xlab = "n")
title("log2 til Fibonacci-sekvensen")
```