Chapter 39 : Fast Fibonacci

The fibonacci function defined on the natural numbers is as follows.

* (0) f.0	=	0
* (1) f.1	=	1
* (2) f.(n+2)	=	f.n + f.(n+1)

We take a conventional approach to choose x = f.n as part of the invariant. However we strengthen this with y = f.(n+1).

Invariants.

P0 : x = f.n $\wedge$ y = f.(n+1)
P1 : 0 $\leq$ n $\leq$ N

Establish invariants.

n, x, y := 0, 0, 1

Achieving postcondition.

Note that P0 $\wedge$ P1 $\wedge$ n = N => x = f.N

Guard.

n $\neq$ N

vf.

N - n

Loop body.

(n, x, y := n+1, E, E').P0
= {text substitution}
E = f.(n+1) $\wedge$ E'' = f.(n+2)
= {(2)}
E = f.(n+1) $\wedge$ E'' = f.n + f.(n+1)
= {P0}
E = y $\wedge$ E'' = x+y

Algorithm.

n, x, y := 0, 0, 1 ;
Do n ≠ N —>

n, x, y := n+1, y, x+y

Od
{x = f.N ∧ y = f.(N+1) }

The algorithm has complexity O(N).

Another approach.

We observe that the values assigned to x and y within the loop are linear combinations of x and y. We can express this in matrix form.

$$n,\begin{pmatrix} x \\ y \end{pmatrix} := n+1,\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix}$$

The postcondition can be expressed as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N * \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Where

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} f.N \\ f.(N+1) \end{pmatrix}$$

The invariant P0 can now be expressed as

$$\text{P0: } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n * \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

This invariants are established by the assignment

$$n,\begin{pmatrix} x \\ y \end{pmatrix} := 0,\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Now, in the same was that we constructed the fast exponentiation algorithm we propose that we try to construct a program to achieve the same post but this time using the following tail invariant.

Invariants.

$$\text{P0} : \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = A^n * \begin{pmatrix} x \\ y \end{pmatrix}$$

P1 : $0 \le n \le N$

Establish invariants.

$$n, x, y, A := N, 0, 1, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Achieving postcondition.

We note that $P0 \wedge P1 \wedge n = 0 \Rightarrow x = f.N$

Guard.

$n \neq 0$

vf.

n

Key Insight.

If A is a square matrix then we note the following properties.

$A^n = (A*A)^{(n \text{ div } 2)}$ <= even.n

$A^n = A^{(n-1)}*A$ <= odd.n

Loop body.

We observe

P0

= {definition}

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = A^n * \begin{pmatrix} x \\ y \end{pmatrix}$$

= {case even.n}

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = (A*A)^{(n div 2)} * \begin{pmatrix} x \\ y \end{pmatrix}$$

= {WP.}

(n, A := n div 2, A*A).P0

We further observe

P0

= {definition}

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = A^n * \begin{pmatrix} x \\ y \end{pmatrix}$$

= {case odd.n}

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = A * A^{(n-1)} * \begin{pmatrix} x \\ y \end{pmatrix}$$

= {WP.}

$$n, \begin{pmatrix} x \\ y \end{pmatrix} := n-1, A * \begin{pmatrix} x \\ y \end{pmatrix}$$

Algorithm.

$$n, x, y, A := N, 0, 1, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix};$$

Do n ≠ 0 —>

if even.n —> n, *A* := n div 2, *A* * *A*

[] odd.n —> $n, \begin{pmatrix} x \\ y \end{pmatrix} := n-1, A * \begin{pmatrix} x \\ y \end{pmatrix}$

fi

Od

{x = f.N}

The algorithm has complexity O(Log(N)).

Final refinement.

Our language however does not provide matrices, so it is necessary to try to remove them. We will represent the matrix using 4 variables.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Then

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} * \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a*a+b*c & (a+d)*b \\ (a+d)*c & b*c+d*d \end{pmatrix}$$

We can now eliminate the matrix operations in our algorithm.

A := $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ becomes a, b, c, d := 0,1, 1, 1

A := A*A becomes a, b, c, d := a*a+b*c , (a+d)*b, (a+d)*c, b*c+d*d

$\begin{pmatrix} x \\ y \end{pmatrix} := A * \begin{pmatrix} x \\ y \end{pmatrix}$ becomes x, y := a*x + b*y, c*x + d*y

Final algorithm.

```
n, x, y, a, b, c, d := N, 0, 1, 0, 1, 1, 1 ;
Do n ≠ 0 —>
        if even.n —> n, a, b, c, d  := n div 2,  a*a+b*c , (a+d)*b, (a+d)*c, b*c+d*d
        [] odd.n   —> n, x, y  := n-1,  a*x + b*y,  c*x + d*y

        fi
Od
{x = f.N}
```

The Algorithm involves no matrix operations and has complexity O(Log(N)).