The top of the hill

Introduction

In this note, we demonstrate an application of the Binary Search algorithm, as well as an application of the design technique known as stepwise refinement.

The problem we are about to tackle may be informally specified as follows: Given that array $f[a..b]$, $a \leq b$, is the concatenation of an increasing and a decreasing sequence, design a program to compute the index of the maximal element of f .

As always, we begin by getting a better grip on the problem through a formal specification.

A formal specification

We specify our program as follows :

|[**con** a, b, N : **int** ; **con** f : **array** $[a..b]$ of **int** ;
var x : **int** ;
{ Q }
Top of the hill
{ R }
]| .

Our postcondition is:

R : $a \leq x \leq b \quad \wedge \quad C.x$.

Our precondition is:

Q : $Q0 \quad \wedge \quad Q1$

$Q0$: $a \leq b$

$Q1$: $a \leq N \leq b \quad \wedge \quad C.N$.

The predicate C used above is specified as follows:

$C.m$: $I.a.m \quad \wedge \quad D.m.b$,

for m an integer. The expression $I.a.b$ denotes "$f[a..b]$ is increasing", and $D.a.b$ denotes "$f[a..b]$ is decreasing".

Applying the Binary Search

Since the problem involves searching for an element in an array, we may employ one of our searching algorithms to solve it. In this case, we choose to use the Binary Search . We recall that the specification of the Binary Search is :

|[**con** a, b : **int** ; **con** f : **array** $[a..b]$ of **int** ;

var x : **int** ;

$\{\ a < b \quad \wedge \quad a\ \mathcal{Z}\ b\ \}$

Binary Search

$\{\ a \leq x < b \quad \wedge \quad x\ \mathcal{Z}\ (x+1)\ \}$

]| .

The first step toward applying the Binary Search is to design a suitable relation $\mathcal{Z}$ for the problem at hand. What properties will this relation need to satisfy? Comparing the pre- and postconditions of the Binary Search with those of our program, we see that our $\mathcal{Z}$ needs to satisfy:

- $a\ \mathcal{Z}\ (b+1)$.
- $x\ \mathcal{Z}\ (x+1) \quad \equiv \quad I.a.x \ \wedge\ D.x.b \quad \equiv \quad C.x$.

Looking at these specifications, we propose:

(0) $\quad x\ \mathcal{Z}\ y \quad \equiv \quad I.a.x \quad \wedge \quad D.(y-1).b \quad .$

This specification of $\mathcal{Z}$ satisfies our requirements provided we have

$$I.a.a \quad \wedge \quad D.b.b$$

as properties of I and D . We shall assume these properties.

Thus our precondition becomes:

Q : $\quad Q0 \quad \wedge \quad Q1$

$Q0$: $\quad a < b+1 \quad \wedge \quad a\ \mathcal{Z}\ (b+1)$

$Q1$: $\quad a \leq N \leq b \quad \wedge \quad C.N \quad .$

Our postcondition becomes:

R : $\quad a \leq x < b+1 \quad \wedge \quad x\ \mathcal{Z}\ (x+1) \quad .$

Thus we may instantiate the Binary Search with $a, b := a, b+1$. And so we arrive at the following program.

Program

$$
\begin{array}{l}
|[\ \textbf{var}\ \ x, y, m :\ \textbf{int} \\
\quad x, y\ :=\ a, b+1 \\
\quad \{\ \text{Inv:}\ \ P\ \ \equiv\ \ P0\ \wedge\ P1\ \} \\
\quad \{\ P0 :\ a \le x\ \wedge\ x < y\ \wedge\ y \le b+1\ \} \\
\quad \{\ P1 :\ x\ \mathcal{Z}\ y\ \} \\
\quad ;\ \textbf{do}\ \ x+1 \neq y\ \ \rightarrow \\
\qquad m := (x+y)\ \textbf{div}\ 2 \\
\qquad \{\ x < m < y\ \} \\
\qquad ;\ \textbf{if}\ \ m\ \mathcal{Z}\ y\ \ \rightarrow\ \ x := m \\
\qquad\ \ []\ \ x\ \mathcal{Z}\ m\ \ \rightarrow\ \ y := m \\
\qquad\ \ \textbf{fi} \\
\qquad \{\ P\ \} \\
\quad \textbf{od} \\
\quad \{\ P\ \ \text{and}\ \ x+1 = y\ ,\ \text{hence}\ \ R\ \} \\
]|\ \ \ .
\end{array}
$$

From (0) , $P0$, and $x < m < y$ it is clear that this program does not inspect f values outside the range $[a..b]$. Thus our only remaining obligation is to prove the non-abortion of the selection statement.

Proving non-abortion

To prove non-abortion, we must prove the disjunction of the guards. Thus we calculate:

$$
\begin{array}{ll}
 & m\ \mathcal{Z}\ y\ \ \vee\ \ x\ \mathcal{Z}\ m \\
\equiv & \quad \{\ \ (0)\ ;\ \ x\ \mathcal{Z}\ y\ \ \text{from the invariant}\ \} \\
 & I.a.m\ \ \vee\ \ D.(m-1).b
\end{array}
$$

$\equiv$ { (1) and (2) ; $a < m < b$ }

$m \leq N \quad \vee \quad N \leq (m-1)$

$\equiv$ { Arithmetic }

true .

In the above, we have used the following properties of predicates I and D. For $a \leq p \leq q \leq b$, we have :

(1) $I.p.q \quad \equiv \quad q \leq N$

(2) $D.p.q \quad \equiv \quad N \leq p$.

We enjoy the use of these properties thanks to the conjunct $Q1$ of the precondition.

Refining the guards

Finally, it would be nice to refine the guards into easily executable code. Toward this end we calculate with guard $m \mathcal{Z} y$:

$m \mathcal{Z} y$

$\equiv$ { (0) ; $x \mathcal{Z} y$ from the invariant }

$I.a.m$

$\equiv$ { (1) ; $a < m < b$ }

$m \leq N$

$\equiv$ { (1) ; $a \leq m-1 < m < b$ }

$I.(m-1).m$

$\equiv$ { Property of I }

$f.(m-1) < f.m$.

We can calculate

$$x \mathcal{Z} m \quad \equiv \quad f.m < f.(m-1)$$

in a similar manner, as the interested reader may verify.

The final program

And so we arrive at our final —un-annotated— program:

$$
\begin{array}{l}
|[\ \textbf{var}\ \ x, y, m :\ \textbf{int} \\
\quad x, y := a, b+1 \\
\quad ;\ \textbf{do}\ \ x+1 \neq y \quad \rightarrow \\
\qquad\quad m := (x+y)\ \textbf{div}\ 2 \\
\qquad\quad ;\ \textbf{if}\ \ f.(m-1) < f.m \quad \rightarrow \quad x := m \\
\qquad\qquad []\ \ f.m < f.(m-1) \quad \rightarrow \quad y := m \\
\qquad\qquad \textbf{fi} \\
\quad \textbf{od} \\
]|
\end{array}
$$

Postscript: An observation

Finally, in addition to (1) , we have used the following properties of I :

- $I.p.p$
- $I.(p-1).p \quad \equiv \quad f.(p-1) < f.p$

We wish to note that these properties are both satisfied by the conventional specification of an ascending sequence:

$$I.p.q \quad \equiv \quad \langle \forall i :\ p \leq i < q :\ f.i < f.(i+1) \ \rangle \qquad .$$

The same observation applies to predicate D .

Acknowledgment

I am indebted to the Eindhoven Tuesday Afternoon Sessions for the original version of this design. In addition, Tom Verhoeff and Jeremy Weissmann provided extensive comments on earlier versions of this document. Their feedback has been invaluable.

Original: Mumbai, 28 August 2006

Revision: Mumbai, March 27-28 2007

Apurva Mehta
apurva@mathmeth.com