The Invariant Game Problem Book

Matteo Vaccari <m.vaccari@sourcesense.com>
http://matteo.vaccari.name/

Eindhoven, 20th November 2008

1 Problems with numbers

1.1 Solved example: compute factorial – value: 0 points

The factorial function is defined by

$$n! = 1 * 2 * \cdots * (n-1) * n$$

Examples:

- input 0, output 1
- input 1, output 1
- input 3, output 6

Spec:

$$\{\ true\ \}\ S\ \{\ post\colon x = n!\ \}$$

Invariant:

$$inv\colon x = k! \qquad \text{introducing new variable } k$$

Implementation:

$k := 0; x := 1$
$\{\ x = k!\ \Leftarrow\ 0! = 1\ \}$
while $k \neq n$
  $\{\ x = k! \wedge k \neq n\ \}$
  $x := x * k$
  $k := k + 1$
  $\{\ x = k!\ \}$
end
$\{\ x = k! \wedge k = n\ \}$
$\{$ therefore, $x = n!\ \}$

1.2 Compute Fibonacci – value: 100 points

The Fibonacci function is defined by

$$\begin{aligned} F(0) &= 0 \\ F(1) &= 1 \\ F(n+2) &= F(n+1) + F(n) \qquad \text{for } n \geqslant 0 \end{aligned}$$

Examples:

- input: 0, output: 0
- input: 1, output: 1
- input: 2, output: 1
- input: 6, output: 8

An extra requirement: recursive programs are not allowed! You are restricted to iterative programs.

1.3 Compute a table of squares – value: 100 points

You are given an array a of size $a.size$. Initialize it to the sequence of squares. Examples:

$a.size = 0$	output: $a = []$
$a.size = 1$	output: $a = [0]$
$a.size = 6$	output: $a = [0, 1, 4, 9, 16, 25]$

Additional requirement: you are not allowed to use multiplication!

1.4 Compute c^n – value: 100 points

Examples

- input $(2, 0)$, output 1
- input $(2, 3)$, output 8
- input $(3, 4)$, output 81

1.5 Compute c^n in logarithmic time – value: 200 points

The spec and examples are the same as in the previous problem, but you have the additional requirement of executing in logarithmic time. You can do it because when the exponent is even,

$$c^{2*n} = c^n * c^n$$

which means that, for instance, $c^{16} = (c^8)^2 = ((c^4)^2)^2 = (((c^2)^2)^2)^2$, which can be computed with 4 multiplications instead of 15. *Hint: let the running variable decrease from* n *to* 0 *instead of the opposite.*

2 Arrays

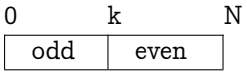
2.1 Solved example: Separate odd and even numbers – value: 0 points

Rearrange an array in place so that the even values are to the left and the odd values to the right.

Examples

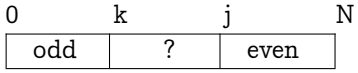
- input [], output []
- input [1,2,3,4,5], output [2,4,1,3,5]

Spec



var a : **array** $[0, N)$ **of integer**
$\{ \; true \; \} \; S \; \{ \; all\_even[0, k) \wedge all\_odd[k, N) \; \}$
where
$all\_even[x, y) \quad \equiv \quad \forall i : x \leqslant i < y : even(a[i])$

Invariant



inv: $all\_even[0, k) \wedge all\_odd[j, N)$

Implementation

$k := 0; j := N$
$\{ \; inv: \; all\_even[0, 0) \wedge all\_odd[N, N) \; \}$
while $j \neq k$
$\{ \; inv \wedge j \neq k \; \}$
if $even(a[k])$
$k := k + 1$
else
$swap(a[k], a[j])$
$j := j - 1$
end
$\{ \; inv \; \}$
end
$\{ \; inv \wedge j = k \; \}$
$\{ \; post \; \}$

2.2 Remove zeros – value: 100 points

You must change array a in place by moving all the non-zero elements to the left. You must also set k to the number of non-zero elements. Examples:

- input $a = []$, output $a = [], k = 0$
- input $a = [3]$, output $a = [3], k = 1$
- input $a = [0]$, output $a = [?], k = 0$
- input $a = [1, 0, 0, 3, 0]$, output $a = [1, 3, ?, ?, ?], k = 2$

2.3 Compress sequences – value: 150 points

Change array a in place by substituting sequences of 2 or more equal elements with a single element. Set k to the number of remaining elements. Examples:

- input $a = [2]$, output $a = [2], k = 1$
- input $a = [3, 3, 3]$, output $a = [3, ?, ?], k = 1$
- input $a = [1, 3, 3, 3, 1, 1, 3]$, output $a = [1, 3, 1, 3, ?, ?, ?], k = 4$

2.4 The famous *Dutch national flag* problem – value: 150 points

You are given an array of elements that can be red, blue or white. The object is to reorder the array so that the red elements are left, the white in the middle and the blue to the right. The reordering must be done by swaps.

Examples

input: []	output: []
input: $[w]$	output: $[w]$
input: $[r, b, r, w, b, w]$	output: $[r, r, w, w, b, b]$

2.5 Merge two arrays – value: 150 points

You are given three arrays a, b, c, with $c.size = a.size + b.size$. Assume that a and b are sorted. Copy them to c, preserving the order.

Examples:

$a = [3, 5]$	$b = []$	$c = [3, 5]$
$a = [3, 8]$	$b = [3, 7, 9]$	$c = [3, 3, 7, 8, 9]$

2.6 Binary search – value: 200 points

Given a sorted array a and value v, write a program that will find the index of v in the array. The program should execute in logarithmic time.

Examples:

$a = [1, 3, 5]$	$v = 3$	$k = 1$
$a = [1, 3, 5]$	$v = 2$	$k = -1$

3 References

Recommended textbooks:

- Roland Backhouse, *Algorithmic Problem Solving*, `http://www.cs.nott.ac.uk/~rcb/G51APS/aps.ps`. A fun textbook that introduces programming techniques as ways to solve mathematical puzzles.
- Roland Backhouse, *Program Construction: Calculating Implementations from Specifications*, Wiley, 2003. The sequel to Algorithmic Problem Solving; applies the same techniques to the construction of programs.
- Edward Cohen, *Programming in the 1990s*, Springer 1990. Another good textbook.

Other books:

- Edsger W. Dijkstra, *A Discipline of Programming*, Prentice-Hall, 1976. Introduces an incremental way of constructing programs, as opposed to the mathematical tradition of writing a program first and the proof later. "To develop a proof and program hand in hand".
- Antonietta van Gasteren, *On the Shape Of Mathematical Arguments*, Springer, 1990. A dissertation on how to present mathematical arguments in a way that is simple and speaking, by sharing the workload more equally between the author and the reader.
- Jon Bentley, *Programming Pearls*, 2nd edition, Addison-Wesley, 2000. Chapters 4 and 5 are about writing a correct implementation of the *binary search* algorithm, which is surprisingly difficult to get right. There are useful hints on how to apply invariants from the point of view of a programming practictioner.