The problem of the Longest Ribbon

This is a continuation of EWD942 in the sense that we shall use its notation and results.

In an integer sequence a segment is "a ribbon of width w" if no two of its elements are more than w apart. Formally

(0) $b.x.y \equiv (\underline{A}\, i,j : x \leq i < j < y : abs.(f.i - f.j) \leq w)$.

The program should determine the maximum length of a ribbon of width w in $f(i : 0 \leq i < N)$.

The monotonicity assumption is satisfied, so the theory of EWD942 is applicable, and our job is the implementation of

$$\{h = g.n\}\ h := g.(n+1)\ \{h = g.(n+1)\}$$.

A quadratic algorithm readily presents itself; the purpose of this note is to develop an $N \cdot \log.N$ algorithm.

* * *

Our first step is inspired by the knowledge -gathered in experience- that the absolute value is an awkward function that we therefore would like to eliminate from the functional specification. The best start is usually the use of

(1) $abs.X = X \ \underline{max}\ -X$.

This yields for the term in (0) the rewriting

$$\begin{aligned} & abs.(f.i - f.j) \leq w \\ = & \{(1)\} \\ & (f.i - f.j \ \underline{max}\ f.j - f.i) \leq w \end{aligned}$$

$= \{$properties of $\underline{\max}\}$
$f.i - f.j \leq w \;\wedge\; f.j - f.i \leq w$
$= \{$arithmetic$\}$
$f.i \leq f.j + w \;\wedge\; f.i \geq f.j - w$.

That is, we have

$$b.x.y \equiv b'.x.y \wedge b''.x.y \quad , \text{ with}$$

$$b'.x.y \equiv (\underline{A}\, i,j : x \leq i < j < y : f.i \leq f.j + w) \quad \text{and}$$

$$b''.x.y \equiv (\underline{A}\, i,j : x \leq i < j < y : f.i \geq f.j + w) \quad .$$

This looks like a nice result in view of Theorem 1, EWD942. [I should probably not have introduced the term "right-strengthening":

$$(\underline{A}\, X,Y : \mathcal{B}.(XY) : \mathcal{B}.X)$$

is usualy rendered by saying that "(the set of strings satisfying) $\mathcal{B}$ is prefix-closed".]

Does this help? Can we solve the problem for $b.x.y$ if we can solve it for the simpler $b'.x.y$ and $b''.x.y$? Let us explore this, in view of Theorem 1, EWD942, for disjunction and conjunction.

Let $g.n$, $g'.n$, and $g''.n$ be the smallest natural solutions of $x : b.x.n$, $x : b'.x.n$, and $x : b''.x.n$ respectively. Then

(i) for $b.x.y \equiv b'.x.y \vee b''.x.y$, $g.n$ is the smallest natural value that solves $x : b.x.n$ or solves $x : b''.x.n$, i.e. $g.n = g'.n \;\underline{\min}\; g''.n$

(ii) for $b.x.y \equiv b'.x.y \wedge b''.x.y$, $g.n$ is the smallest common solution of both $x : b'.x.n$ and $x : b''.x.n$,

i.e. $g.n \geq g'.n \max g''.n$,

where we have the inequality in order to cater for the case that $g'.n \max g''.n$ solves only one of the equations. We have

$$g.n = g'.n \max g''.n$$

if $(\underline{A}i: g'.n < i \leq n: b'.i.n) \wedge (\underline{A}i: g''.n < i \leq n: b''.i.n)$, i.e.

if $(\underline{A}X,Y: \mathcal{B}'.(XY): \mathcal{B}'.Y) \wedge (\underline{A}X,Y: \mathcal{B}''.(XY): \mathcal{B}''.Y)$,

i.e. if both $\mathcal{B}'$ and $\mathcal{B}''$ are postfix-closed as well!

Since our $\mathcal{B}'$ and $\mathcal{B}$ are postfix-closed, a legitimate implementation of

$$h := g.(n+1)$$

is therefore

$$h' := g'.(n+1) \; ; \; h'' := g''.(n+1) \; ; \; h := h' \max h'' \; ,$$

i.e. by thus merging the solutions of the primed problems, we can solve the original one. The two primed problems are so similar — the one for f is the other one for $-f$ — that we shall devote the rest of this note to solving the maximum segment length for $\mathcal{B}'$.

* * *

From EWD942 we know that our task is to implement

$$\{h' = g'.n\} \; h := g'.(n+1) \; ;$$

let us elaborate $b'.x.(n+1)$ since $g'.(n+1)$ is defined as the smallest natural solution of $x: b'.x.(n+1)$.

$$
\begin{aligned}
& b'.x.(n+1) \\
= & \{\text{definition of } b'\} \\
& (\underline{A}i,j: x \le i < j < n+1: f.i \le f.j + w) \\
= & \{\text{isolation largest } j\text{-value}\} \\
& (\underline{A}i,j: x \le i < j < n: f.i \le f.j + w) \wedge (\underline{A}i: x \le i < n: f.i \le f.n + w) \\
= & \{\text{definition of } b'\} \\
& b'.x.n \wedge (\underline{A}i: x \le i < n: f.i \le f.n + w) \quad .
\end{aligned}
$$

Because B' is postfix-closed

$$b'.x.n \equiv g'.n \le x$$

Combining the two results, we find that

$$b'.x.(n+1) \equiv g'.n \le x \wedge (\underline{A}i: x \le i < n: f.i \le f.n + w) \quad .$$

So far, we have simplified – which is always safe – and even quite nicely so in view of the precondition $h' = g'.n$.

Let us now investigate whether we can optimize. As $b'.x.(n+1)$ has been given, finding the smallest solution of $x: b'.x.(n+1)$ for a single value of n requires a linear search, which would lead to a quadratic algorithm. But that smallest solution has to be determined for a whole sequence of successive values of n, and the question is whether the algorithm can collect additional information about $f(i: 0 \le i < n)$ as it goes along, with the aid of which the determination of the smallest solution of $x: b'.x.(n+1)$ can be speeded up. Because that additional information is to be used over and over again, for successive values of n we would like to reduce its dependence of n, which occurs in $b'.x.(n+1)$ in no less than 3 different places. Furthermore, it is reasonable to ex-

pect that, somewhere along the line, we should exploit the transitivity of $\leq$.

The first conjunct of $b!x.(n+1)$ being simple to evaluate and the second conjunct being the one that would give rise to the linear search, it is the latter one on which we should focus our attention. It is of the form $Q0$, given by

$$Q0 \equiv (\underline{A}i: x \leq i < n: f.i \leq H)$$

in which we consider H as a parameter, of which it is temporarily irrelevant that it depends on $f.n$ (which does not enter $b!x.(n+1)$ in any other way). The introduction of H is nothing deep: it is a convenient abbreviation.

Because -like $b!x.(n+1)$- both its conjuncts weaken with increasing x, $g!(n+1)$ is the maximum of $g.n$ and the smallest (natural) solution of $x: Q0$.

In view of the very special form of $Q0$, the smallest natural solution of $x: Q0$ is almost the largest solution of

$$x: (f.(x-1) > H) \ ;$$

I said "almost" because in this formulation we have to ensure that the equation <u>has</u> a solution, no matter how large H . This we can ensure by defining for the purpose of this discussion

$$f.(-1) = +\infty \ .$$

This being done, we can define the smallest natural solution of $x: Q0$ as the <u>unique</u> q satisfying

(0) $\quad 0 \leq q \leq n \;\wedge\; f.(q-1) > H \;\wedge\; (\underline{A}i: q \leq i < n: f.i \leq H)$.

We would like to tackle the last conjunct with the aid of collected information, which should be independent of H. Because

$$f.(q-1) > H \;\wedge\; f.i \leq H \;\Rightarrow\; f.i < f.(q-1)$$

we can "eliminate" H from the quantified expression by concluding that q is a solution of $x: Q1$ with $Q1$ given by

$$Q1 \equiv 0 \leq x \leq n \;\wedge\; f.(x-1) > H \;\wedge\; (\underline{A}i: x \leq i < n: f.i < f.(x-1)) \quad .$$

Let $q1$ be an arbitrary solution of $x: Q1$; then

$$q1-1 < n \;\wedge\; f.(q1-1) > H \quad ;$$

from the last conjunct of (0) we then conclude $q1-1 < q$ or, equivalently $q1 \leq q$, i.e. the q we are looking for is now identified as the largest solution of $x: Q1$.

And now we are in good shape. Let us write

$$Q1 \equiv f.(x-1) > H \;\wedge\; Q2 \qquad \text{with}$$

$$Q2 \equiv 0 \leq x \leq n \;\wedge\; (\underline{A}i: x \leq i < n: f.i < f.(x-1))$$

— i.e. we have extracted the conjuncts independent of H —, and suppose that we have somehow tabulated the roots of $x: Q2$ in increasing order. In order to determine q one has to find the largest root x such that $f.(x-1) > H$.

Let $x0$ and $x1$ be 2 different roots of $x: Q2$ such that $x0 < x1 \leq n$ or $x0 \leq x1-1 < n$. Because $x0$ is a root we then have $f.(x1-1) < f.(x0-1)$.

In other words, for x in increasing order ranging over the roots of x: Q2, the values f.(x-1) form a decreasing sequence, and as a result q can be determined in logarithmic time.

The adjustment of the set of roots of x: Q2 in preparation of n:=n+1 has now to be considered. From Q2 we see that

(i) roots x with $f.(x-1) > f.n$ also solve $x: Q2^{n}_{n+1}$ and hence have to be maintained

(ii) roots x with $f.(x-1) \leq f.n$ don't solve $x: Q2^{n}_{n+1}$ and hence have to be discarded

(iii) the only new root of $x: Q2^{n}_{n+1}$ is n+1, which has to be added.

Hence the roots of x: Q2 can be stored in a stack; since (according to (iii) stack growth is linear in N, the unstacking involved in (i) and (ii) can be done in the most naive fashion, one element at a time, and the manipulations remain linear in N.

Note Instead of stacking individual roots, one can stack root intervals, e.g. pairs (i,j) such that the pair (i,j) in the stack represents the roots x satisfying $i < x \leq j$ (or the subscripts x-1 satisfying $i \leq x-1 < j$). S. Doaitse Swierstra draw our attention to this possibility at the ATAC session of 15 October 1985. The searching for q becomes a two-stage process - two binary searches in succession, one for the interval and one for the place in the interval. It remains logarithmic. Worst case storage demands remain linear in N, the average case might be better. (End of Note.)

Concluding remarks

When I raised the problem of the longest ribbon length I had the feeling that it somehow was a stinker. Yet it took me some time to admit that -at least for the time being- could not find a shorter heuristically convincing argument that leads to Q2.

Writing the above was much harder than I expected. (Took also more time than intended.) The introduction of the x-1 was a nuisance; I considered for quite some time whether I would try to eliminate it. But with the h as introduced in EWD942 I would get +1's in other places.

The discovery that the conjunction of prefix-closed predicates is only helpful if they are postfix-closed as well was only made after I embarked on this problem. The moral of the story seems to be that a compact theory about extreme solutions of equations with various forms of monotonicity would be welcome.

I am indebted to the ATAC, to Chris Lengauer in particular.

Austin, 18 October 1985

prof. dr. Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
USA