Third example.

For $N \geq 1$ we consider a sequence of N elements: $A[0],\ldots,A[N-1]$. The order of increasing subscript value will be called "the order from left to right". For any s satisfying $0 \leq s \leq N$, we can take from $A[0],\ldots,A[N-1]$ so-called "subsequences of length s " by removing an arbitrary collection of $N-s$ elements and retaining the remaining s elements in the order in which they occurred in the original sequence. As a result, $A[0],\ldots,A[N-1]$ contains 2^N subsequences. When, in addition, each element has an integer value, we call a subsequence an "upsequence" if and only if it contains no element with a right-hand neighbour smaller than itself.

Note. According to this definition, all N subsequences of length 1 --and even the empty subsequence-- are upsequences. (End of note.)

Our problem is the design of an algorithm that determines for any such sequence the maximum length of an upsequence contained in it.

Note. Although there need not be a unique longest upsequence, the maximum length is unique, e.g. the given sequence $(3,1,1,2,5,3)$ yields 4 for the maximum length, realised either by $(1,1,2,5)$ or by $(1,1,2,3)$. (End of note.)

Let the final value of the variable k represent the answer we are looking for, i.e. we seek to establish the relation

R: k = the maximum length of an upsequence contained in $A[0],\ldots,A[N-1]$.

A moment's reflection tells us that each element of the given sequence has to be considered, and we only make the (modest) assumption that we can get away with taking the elements into consideration in the order from left to right. More formally, we propose to introduce a second variable, n say, and to establish initially and to maintain subsequently the so-called "invariant relation"

```
P1:     k = the maximum length of an upsequence contained
            in  A[0],...,A[n-1]  and
        1 ≤ n ≤ N      ,
```

to be used in a program of the structure --assertions having been inserted between braces--

```
        "establish  P1  for  n = 1 "; {P1}
        do n ≠ N → {P1 and 1 ≤ n < N}
                 "increase  n  by  1  under invariance of  P1 " {P1}
        od {P1 and n = N}     .
```

Relation P1 has been inspired by the fact that R contains the parameter N ; it has been derived from R by the standard technique of replacing a constant (here N) by a variable (here n) --and (as usual) restricting its range-- so that, as a result

$$(P1 \ \underline{\text{and}}\ n = N) \Rightarrow R \qquad ,$$

from which we deduce that the above program would do the job; this inspiration is further encouraged by the observation that P1 is easily established initially as $(n = 1 \ \underline{\text{and}}\ k = 1) \Rightarrow P1$.

The repeatable statement "increase n by 1 under invariance of P1 " is the algorithmic equivalent of the induction step: given the solution for n it has to construct the solution for n+1 . The required invariance of P1 means that the increase n:= n + 1 may have to be accompanied by an adjustment of the value of k (the only other variable occurring in P1 !). Because extension of the sequence considered with a next element can never decrease the maximum length of an upsequence contained in it, and can increase it by at most 1 , the adjustment of k , when needed, will have the form k:= k + 1 . For the repeatable statement we can take the form

```
{P1 and 1 ≤ n < N}
if .... → k:= k + 1 [] .... → skip fi;
n:= n + 1 {P1}
```

and our only task is now to fill in the dots, i.e. to decide under which circumstances k has to be increased by 1 , or can remain unchanged respectively, such that after the subsequent increase n:= n + 1 the relation P1 is again guaranteed to hold.

Because A[n] is the next element to be considered, we can fill in the dots as follows:

```
{P1 and 1 ≤ n < N}
if m ≤ A[n] → k:= k + 1 [] A[n] < m → skip fi;
n:= n + 1 {P1}
```

provided the value of m is defined by

D: m = the minimum right-most element of an upsequence of length k contained in A[0],...,A[n-1] .

In other words: the obligation to keep P1 invariant requires, besides the value k , the value m as an additional derivative from A[0],...,A[n-1] . Introducing m as a variable, and replacing the original invariant relation P1 by the stronger P1 and D , we find ourselves considering the program

```
n:= 1; k:= 1; m:= A[0]; {P1 and D}
do n ≠ N → {P1 and D and 1 ≤ n < N}
            if m ≤ A[n] → k:= k + 1; m:= A[n]
             [] A[n] < m → ....
            fi;
            n:= n + 1 {P1 and D}
od {R}
```

Note. Strengthening the invariant relation is the computational analogue to the strengthening of an induction hypothesis. (End of note.)

Note that now we have to increase n by 1 under invariance of

P1 <u>and</u> D . In the first alternative the invariance of D presents no problem: all upsequences of the increased length have $A[n]$ as their right-most element, and this is therefore the proper new value for m .

But what in the second alternative, when $A[n] < m$? The new element $A[n]$ cannot be used to form a longer upsequence, but should it be used to lower m because, thanks to its inclusion, the right-most element of an upsequence of length k can now be smaller than was possible before the extension? A moment's reflection will tell us that for our last dots we can fill in

```
{$A[n] < m$}
if $m' \le A[n] \rightarrow m := A[n]$
 ▯ $A[n] < m' \rightarrow$ skip
fi
```

provided the value of m' is defined by

```
D':   $m'$ = if $k = 1 \rightarrow$ minus infinity
           ▯ $k > 1 \rightarrow$ the minimum right-most element of an upsequence
                       of length $k-1$ contained in $A[0],...,A[n-1]$
           fi    .
```

In other words: the obligation to keep D invariant requires, besides the values k and m , the value m' as an additional derivative from $A[0],...,A[n-1]$. After the introduction of m' as a variable and of the strengthened relation P1 <u>and</u> D <u>and</u> D' , the obligation to maintain the invariance of D' requires an m" , etc., and by mathematical induction we conclude that we could use a whole array of m-values, and combine D <u>and</u> D' <u>and</u> D" <u>and</u> ... into

P2: ($\underline{A}$ j: $1 \le j \le k$: $m[j]$ = the minimum right-most element of an upsequence of length j contained in $A[0],...,A[n-1]$)

where the old m is now $m[k]$, the old m' is now $m[k-1]$, etc.

With the new invariant relation P1 <u>and</u> P2 our program takes its final shape:

```
n:= 1; k:= 1; m[1]:= A[0]; {P1 and P2}
do n $\neq$ N $\rightarrow$ "increase n by 1 under invariance of P1 and P2" od {R}
```

We leave to the reader the now straightforward verification that for the repeatable statement the following suffices:

```
"increase n by 1 under invariance of P1 and P2":
if m[k] $\leq$ A[n] $\rightarrow$ k:= k + 1; m[k]:= A[n]
 [] A[n] < m[1] $\rightarrow$ m[1]:= A[n]
 [] m[1] $\leq$ A[n] < m[k] $\rightarrow$ "establish j such that m[j-1] $\leq$ A[n] < m[j]";
                            m[j]:= A[n]
fi;
n:= n + 1 {P1 and P2}
```

Using

P3: $m[i] \leq A[n] < m[j]$ and $1 \leq i < j \leq k$

as the invariant relation in the binary search for our last refinement

```
"establish j such that m[j-1] $\leq$ A[n] < m[j]":
{m[1] $\leq$ A[n] < m[k]}
i:= 1; j:= k; {P3}
do i $\neq$ j - 1 $\rightarrow$ h:= (i + j) div 2; {i < h < j}
              if m[h] $\leq$ A[n] $\rightarrow$ i:= h {P3}
               [] A[n] < m[h] $\rightarrow$ j:= h {P3}
              fi {P3}
od {m[j-1] $\leq$ A[n] < m[j]}
```

we have solved our original problem with an N.log(N)-algorithm.

<u>Note</u>. Because the difference $j - i$ decreases each time by at least 1 and remains positive, termination of our last refinement is guaranteed, and the existence of a j such that $m[j-1] \leq A[n] < m[j]$ is thereby proved. (End of note.)

If we assume the elements of the array m initialized to plus infinity, we observe that in the above algorithm the adjustment of the array m , prior to the increase $n:= n + 1$, boils down to the effective decrease of exactly one element of the array m to the value $A[n]$.

Suppose that we determine in parallel h = the maximum length of a downsequence contained in $A[0],...,A[N-1]$. That computation would comprise a corresponding array, p say, --with elements initialized to minus infinity-- such that each adjustment of it boils down to an effective increase of exactly one element of the array p to the value $A[n]$.

We call a pair (i, j) --with $1 \leq i \leq k$ and $1 \leq j \leq h$ -- such that $m[i] \leq p[j]$ "an inversion". Because m-values never increase and p-values never decrease, an inversion, once introduced, remains in existence. Furthermore the double adjustment introduces at least one new inversion (via the m- and p-elements that are effectively decreased and increased to $A[n]$ respectively). Hence for the combined computation the relation

$$n \leq \text{the number of inversions}$$

is an invariant. Because by definition

$$\text{the number of inversions} \leq h * k$$

we conclude $n \leq h * k$. Hence we have proved the

Theorem. A sequence of length $> M^2$ contains a monotonic subsequence of length $> M$.

And this concludes my treatment of the third example.

(Note added during revision. In reponse to the original manuscript, J.Misra of the University of Texas at Austin, Texas, U.S.A., communicated to me a very nice proof of the last theorem that was based on the Pidgeonhole Principle.)

Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow