A queuing problem in a post office.

Given an integer t0 , and two integer arrays a, b(0..N-1), such that

- $N \geq 1$
- $a(0) < a(1) < \ldots < a(N-1)$
- $(\underline{A}i: 0 \leq i < N: b(i) > 0)$.

A post office, equipped with four mutually equivalent service desks, is used by N customers numbered from 0 through N-1 .
The post office opens at moment t0 .
Customer i arrives at moment a(i) and requests a service of b(i) time units. Customers are served in the order of their arrival.
The desk officers are never idle when a customer is waiting for service,
The post office closes as soon as all N customers have been served.

Write a program to compute the total amount of time during which at least one customer is waiting for service.

*

It is requested to compute r , where

r = (the total amount of time during which at least one customer is waiting for service) .

One of the few openings to the solution of this problem seems to be to study how r functionally depends on the waits of the individual customers.
Thus we are led to define, for all $i: 0 \leq i < N$,

s_i = (the moment at which the service of customer i begins) ,

in terms of which we can grasp the wait imposed on customer i by the length of the interval $(a(i), s_i)$.
If the N intervals defined like this are coloured black on a time axis which is white to begin with, then we agree, assuming that black plus black gives black, that the value of r to be computed is the total length of whatever looks black. Shorter,

R: r = (the length of the superposition of the intervals
$(a(i), s_i)$: $0 \leq i < N$) .

In bypassing we observe two possibly useful properties of the numbers s_i :

$$(\underline{A}i: 0 \leq i < N: a(i) \leq s_i) \qquad (0)$$

$$s_0 \leq s_1 \leq \ldots \leq s_{N-1} \quad ; \qquad (1)$$

the first one stems from the thought that a customer's service cannot start before its arrival, and the second one from the rule that customers are served in the order of their arrival.
The convenience with which R is realised critically depends on the convenience with which the numbers s_i are determined. So let us embark on the computation of the s-sequence first.

+

We imagine that, upon its arrival, a customer i finds the residues of what could have happened in the past: a chaotic variety of possibilities, like a not yet opened post office or zero, more or even four desks occupied or a perhaps empty queue or combinations of these.
A moment's reflection, which --it should be admitted-- can last a long time, tells us that what matters about the earlier customers j: $0 \leq j < i$ is their departure times: these entirely describe the changing occupancy of the four desks. More specificly even, inspired by (1), the only aspect determinative of the moment s_i is the fourth largest departure time among those of the earlier customers: this describes the earliest instant at which a desk is available to customer i .
Thus we are led to define, for all i: $4 \leq i < N$,

$d4_i$ = (the fourth largest departure time among those of
customers j: $0 \leq j < i$) ,

and similarly, for reasons that become apparent soon, $d3_i$, $d2_i$, $d1_i$ for the third, second and first largest departure time among those of the earlier customers.
In terms of these numbers we then grasp s_i by

$$(\underline{A}i:\ 4 \leqslant i < N:\ s_i = \max(a(i), d4_i)) \qquad (2a)$$

$$(\underline{A}i:\ 0 \leqslant i < 4:\ s_i = \max(a(i), t0)) \ , \qquad (2b)$$

in obedience to the rule that desk officers are never idle when a customer is waiting for service.

Herewith we have an effective way of computing the s-sequence , provided we have an effective way to compute the d4-sequence .
The d4-sequence is computed very effectively if we realise that, in terms of s_i , customer i's departure time is conveniently expressed as $s_i + b(i)$. Since $b(i) > 0$, we have $s_i + b(i) > d4_i$ on account of (2a), so that it is not difficult to see that

$$(\underline{A}i:\ 4 \leqslant i < N-1:\ d4_{i+1} = \min(s_i+b(i), d3_i, d2_i, d1_i)) \ , \qquad (3)$$

or, to accomodate the other three sequences as well, the more general

$$(\underline{A}i:\ 4 \leqslant i < N-1:\ (d4_{i+1}, d3_{i+1}, d2_{i+1}, d1_{i+1}) = \text{nondecreasing arrangement of } (s_i+b(i), d3_i, d2_i, d1_i)) \ . \qquad (4a)$$

+

For the establishment of R we propose as an invariant relation

P: r = (the length of the superposition of the intervals $(a(i), s_i):\ 0 \leqslant i < n$) <u>and</u> $0 \leqslant n \leqslant N$,

and investigate an increase of n by 1 .
We have to examine which part of the interval $(a(n), s_n)$ is not covered by any of the intervals $(a(i), s_i):\ 0 \leqslant i < n$. Properties (0) and (1) infer that for $a(n) \geqslant s_{n-1}$ the interval $(a(n), s_n)$ is disjoint with the other ones. For $a(n) < s_{n-1}$ we conclude from (1) and the monotonicity of the a-sequence that the part $(a(n), s_{n-1})$ is contained in $(a(n-1), s_{n-1})$ and that the part (s_{n-1}, s_n) is disjoint with all intervals $(a(i), s_i):\ 0 \leqslant i < n$.

Hence, the part $(\max(a(n), s_{n-1}), s_n)$ is not covered by any of the other intervals, and we conclude:

$$(P \ \underline{\text{and}} \ 1 \leq n < N) \Rightarrow wp(\text{"}r{:=}\ r + s_n - \max(a(n), s_{n-1});\ n{:=}\ n + 1\text{"}, P)$$

or, if we are willing to define: $s_{-1} \leq a(0)$,

$$(P \ \underline{\text{and}} \ 0 \leq n < N) \Rightarrow wp(\text{"}r{:=}\ r + s_n - \max(a(n), s_{n-1});\ n{:=}\ n + 1\text{"}, P) .$$

The corresponding program then is

$$r, n := 0, 0;\ \underline{\text{do}}\ n \neq N \rightarrow r{:=}\ r + s_n - \max(a(n), s_{n-1});\ n{:=}\ n + 1\ \underline{\text{od}} ,$$

or, if P is strengthened -- for the sake of efficiency for instance-- with

QO: $\quad q = s_{n-1}$,

$r, n := 0, 0;$ "q: $q \leq a(0)$";
$\underline{\text{do}}\ n \neq N \rightarrow p{:=}\ s_n;\ r{:=}\ r + p - \max(a(n), q);\ n{:=}\ n + 1;\ q{:=}\ p\ \underline{\text{od}}$.

A minor complication concerns the incorporation of the computation of s_m as given via the relations (2a), (2b) and (4a). The inhomogeneity of the equations (2a) and (2b) on the one hand and the undefinedness of our four d-sequences on the other hand would force us to withdraw the nice initialisation "n:= 0" in the program above and peplace it by something like "n:= 4", with all awkward consequences. The remedy, however, is universal and consists of an appropriate enlargement of the domains of the painful functions. If we define

$$(\underline{A}i\colon 0 \leq i < 4\colon d4_i = t0) , \tag{5}$$

then the formulae (2a) and (2b) collapse into

$$(\underline{A}i\colon 0 \leq i < N\colon s_i = \max(a(i), d4_i)) , \tag{2}$$

and the definition of the d-sequences into

$$(d4_0, d3_0, d2_0, d1_0) = (t0, t0, t0, t0) \ ,$$
$$(\underline{A}i:\ 0 \leqslant i < N-1:\ (d4_{i+1}, d3_{i+1}, d2_{i+1}, d1_{i+1}) = \text{nondecreasing arrangement of } (s_i + b(i), d3_i, d2_i, d1_i)) \ . \qquad (4)$$

Strengthening P further with the relation

Q1: $h4,\ h3,\ h2,\ h1 \ = \ d4_n,\ d3_n,\ d2_n,\ d1_n$

results in what we consider as our ultimate program:

```
r, n := 0, 0; q:= a(0); h4, h3, h2, h1 := t0, t0, t0, t0;
do n ≠ N → p:= max(a(n),h4); r:= r + p - max(a(n),q); q:= p;
           h4:= p + b(n);
           do h4 > h3 → h4, h3 := h3, h4
            [] h3 > h2 → h3, h2 := h2, h3
            [] h2 > h1 → h2, h1 := h1, h2
           od;
           n:= n + 1
od.
```

*

Memorable of this exercise is that it has served as a problem on a written examination for third years students in mathematics, and that it has cut as a sharp knife through that group of students: those who passed the exam were invariably the most brilliant students in the other fields of applied mathematics. This observation is generally not taken in gratitude, but it does give an indication on what programming is about.

W.H.J. Feijen,
Sterksel, july 23, 1979.