Minsegsumtwodim

Remark about notation By way of experiment I shall denote functional application by an infix period; it has the highest binding power of all and is (as usual) left-associative. By way of further experiment, I shall also use that period for subscription. (End of Remark about notation.)

This note is a sequel to EWD897, in which the one-dimensional "minsegsum" is solved in linear time.

The functional specification for minsegsumtwodim is

|[M, N: int {M ≥ 0 ∧ N ≥ 0}
; C(m,n: 0 ≤ m < M ∧ 0 ≤ n < N) array of int
; |[x: int
  ; minsegsumtwodim
   {R: x = (MIN m0, m1, n0, n1: 0 ≤ m0 ≤ m1 ≤ M ∧ 0 ≤ n0 ≤ n1 ≤ N :
         (S m,n: m0 ≤ m < m1 ∧ n0 ≤ n < n1: C.m.n))}
  ]|
]|

in which (as usual) the constants of the environment are declared in the outer block.

Remark. For M=1, the specification of minsegsumtwodim reduces to that of aforementioned one-dimensional minsegsum. (End of Remark.)

We remark that in $\underline{\mathrm{MIN}}$ the ranges have been given including the bounds because, without further conventions, the minimum over an empty range is undefined. In $\underline{S}$ the ranges exclude the upper bound, summation over the empty range being defined to yield 0 (i.e. the identity element of the addition).

Since summations over empty ranges ($m0 = m1 \vee n0 = n1$) are included and all yield 0, we can take them out of the range for $\underline{\mathrm{MIN}}$ and rewrite R

$$\begin{aligned} R\colon\ x = 0\ \underline{\min}\ (\underline{\mathrm{MIN}}\ m0, m1, n0, n1 :\ & 0 \le m0 < m1 \le M\ \wedge \\ & 0 \le n0 < n1 \le N : \\ & (\underline{S}\ m, n : m0 \le m < m1 \wedge n0 \le n < n1 : C.m.n)) \end{aligned}$$

provided we extend the definition of $\underline{\mathrm{MIN}}$ with a suitable identity element if its range is empty; from our last form of R we see that 0 will do in this context.

* * *

We know that, for $M=1$, there exists a solution linear in N (and we would like our solution for minsegsumtwodim to reduce to that linear one if $M=1$). In a rectangle with two rows, however, the solution for minsegsumtwodim has little relation to the solutions for minsegsum in the individual rows, as is shown by the following examples:

—for each rectangle we give the number of rightmost columns that realize the minsegsums and minsegsumtwodim respectively—

	rectangle			minsegsum	minsegsumtwodim
(i)	−2	+3	−6	1	3
	+1	−3	−6	2	
(ii)	+2	+1	−5	1	2
	−1	−2	−2	3	
(iii)	+4	−1	−4	2	1
	−3	+2	−6	3	

The moral of the above observation seems to be that if we wish to apply minsegsum computations in the N-direction, they can share in the M-direction little more than the summation. In the following this will be made more precise.

* * *

Our next step is to rewrite R in terms of nested <u>MIN</u>'s and <u>S</u>'s. (At this stage, the reader is invited to convince himself of the correctness of my rewriting and urged to shelve for the time being the question <u>why</u> R is rewritten <u>this</u> way. He is, however, welcome to observe that we have concentrated the quantifications in the N-direction in the middle.) Again, MIN over the empty range is defined to yield a suitable value (i.e. ≥ 0)

$$R:\ x = 0 \ \underline{\min}\ (\underline{\mathrm{MIN}}\ m0 : 0 \leq m0 < M :$$
$$(\underline{\mathrm{MIN}}\ m1 : m0 < m1 \leq M :$$
$$(\underline{\mathrm{MIN}}\ n1 : 0 \leq n1 < N : MS.m0.m1.(n1+1)))) \qquad {}^{*)}$$

$$MS.m0.m1.n1 =$$
$$(\underline{\mathrm{MIN}}\ n0 : 0 \leq n0 < n1 : (\underline{S}\ n : n0 \leq n < n1 : Q.m0.m1.n))$$

$$Q.m0.m1.n = (\underline{S}\ m : m0 \leq m < m1 : C.m.n) \quad .$$

*) Note that this line is equivalent to
$$(\underline{\mathrm{MIN}}\ n1 : 0 < n1 \leq N : MS.m0.m1.n1)))$$

Analogously to EWD897 we now derive two relations for MS .

$$MS.m0.m1.0 = 0 \quad (\text{or any value} \geq 0) \qquad (0)$$

and for $0 \leq n1 < N$

$MS.m0.m1.(n1+1)$
$= \{\text{definition of } MS\}$
$(\underline{\mathrm{MIN}}\ n0 : 0 \leq n0 < n1+1 : (\underline{S}\ n : n0 \leq n < n1+1 : Q.m0.m1.n))$
$= \{\text{isolation of last term of summation}\}$
$(\underline{\mathrm{MIN}}\ n0 : 0 \leq n0 < n1+1$
$\quad (\underline{S}\ n : n0 \leq n < n1 : Q.m0.m1.n) + Q.m0.m1.n1)$
$= \{Q.m0.m1.n1$ does not depend on $n0$, and addition then distributes over minimization over non-empty range$\}$

$= (\underline{\mathrm{MIN}}\ n0: 0 \le n0 < n1+1: (\underline{S}n: n0 \le n < n1: Q.m0.m1.n)) +$
$\quad Q.m0.m1.n1$

$= \{$isolation of last term of minimization; for $n0 = n1$, the summation yields $0\}$

$((\underline{\mathrm{MIN}}\ n0: 0 \le n0 < n1: (\underline{S}n: n0 \le n < n1: Q.m0.m1.n))\ \underline{\min}\ 0) +$
$Q.m0.m1.n1$

$= \{$definition of MS$\}$

$$(MS.m0.m1.n1\ \ \underline{\min}\ 0) + Q.m0.m1.n1 \qquad (1)$$

For Q we derive directly from its definition

$$Q.m0.m0.n1 = 0 \qquad (2)$$

$$Q.m0.(m1+1).n1 = Q.m0.m1.n1 + C.m1.n1 \qquad (3)$$

Our rewritten R tells us that for any m0,m1 combination we have to find the minimum of $MS.m0.m1.(n1+1)$ for all $n1: 0 \le n1 < N$, while (1) gives a recurrence relation for that sequence of values. The snag is that that recurrence relation contains the term $Q.m0.m1.n1$, which – see (3)– satisfies a recurrence relation over m1. In order to exploit these two "orthogonal" recurrence relations to the fullest, the program evaluates for each value of m0 the $M-m0$ recurrences (1) that correspond to the different m1-values in synchrony. To this end, a local array $MSv(m1: m0 < m1 \le M)$ is introduced such that, whenever $MSv.m1$ is adjusted, its value changes from $MS.m0.m1.n1$ to $MS.m0.m1.(n1+1)$. Furthermore a local scalar Qv is introduced that satisfies

Qv = Q.m0.m1.n1 . The program is given below; it reveals why we have written line *) in R the way we did: we want to use each MS-value as soon as it has been computed, i.e. equals MS.m0.m1.(n1+1) .

```
|[ m0: int ; x:=0 ; m0:=0
; do m0 ≠ M →
    |[ MSv(m: m0 < m ≤ M) array of int
    ; m1, n1 : int
    ; m1:= m0 ; do m1 ≠ M → m1:= m1 + 1; MSv.m1 := 0 od
    ; n1:= 0 {(A m1: m0 < m1 ≤ M: MSv.m1 = MS.m0.m1.n1), see (0)}
    ; do n1 ≠ N →
        |[ Qv: int ; Qv,m1 := 0, m0  {Qv = Q.m0.m1.n1, see (2)}
        ; do m1 ≠ M →
              Qv, m1 := Qv + C.m1.n1 , m1 + 1
                { Qv = Q.m0.m1,n1 , see (3)}
            ;MSv.m1 := (MSv.m1 min 0) + Qv
                {MSv.m1 = MS.m0.m1.(n1+1) , see (1)}
            ; x := x min MSv.m1
          od ; n1 := n1 + 1
        ]|
      od ; m0 := m0 + 1
    ]|
  od
]|
```

* * *

<u>History and acknowledgements</u> I had just discussed (the linear) minsegsum in my lectures when Jon Bentley's article [0] appeared, in which he mentioned in passing the two-dimensional generalization. Solutions

of time-complexity $M^2 \cdot N$ were independently found by Jayadev Misra, Mohamed G. Gouda and me. (Gouda's solution generates in turn the $M \cdot (M-1)/2$ sequences of length N, to which minsegsum is applied; his solution needs all the time a linear array of length N, but is somewhat simpler than mine. How much store Misra intended to use I don't know.)

The above presentation has been influenced - and hopefully improved - by my course's audience, with which I discussed an earlier version. I am grateful to Shmuel Katz for insisting on an early inclusion of some heuristics for not trying to incorporate a "tighter coupling" in the M-direction and for accepting an asymmetric solution. The three 2×3 examples have been constructed in response to his request.

* * *

My earlier version refrained from defining a minimum for an empty set, and the formal derivation was less smooth. Halfway this note I got in panic: I failed to see how my derivation could ever yield the program I had derived earlier. The experience was instructive: I just went on and ..... derived a program different from my earlier one! It so happens that replacing

$$Msv.m1 := (Msv.m1 \ \underline{\min} \ 0) + Qv$$

by

$$Msv.m1 := (Msv.m1 + Qv) \ \underline{\min} \ 0$$

changes the program into my earlier version. (For the earlier version to be correct, it is essential that the elements of Msv are initialized at 0.) There seems to be little point in comparing the relative merits of the two programs. The important lesson is that the formal derivation not only leads to new programs, but may even lead to programs one did not intend to derive! And this experience seems the most effective rebuttal of an opinion voiced by J.J. Seidel -a former colleague of mine- many years ago. I told him about formal program derivation when that was still very new; he reacted with "But, sure, you'll never formally derive a program that you have not found already otherwise.". At the time, that reaction struck me as an underestimation of the power of formal techniques (an underestimation which was perhaps typical for the mathematicians of his generation).

Completed on 2 Dec. 1984

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