Minsegsumtwodim

Just when I had posed "minsegsum" to my students, Jon Bentley's article [0] appeared in the CACM. The article mentioned the analogous problem in two dimensions and more or less suggested that it was a difficult one. As it mentioned no complexity results for the two-dimensional problem, both Jayadev Misra and I started to think about it and arrived independently at the same solution. Since the solution is complicated enough to present a problem of presentation, I decided to devote a note to it.

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

The functional specification for minsegsumtwodim is

```plaintext
1[ M,N: int { M ≥ 1 ∧ N ≥ 1 } ]
  ; C(m,n: 0 ≤ m < M ∧ 0 ≤ n < N) array of int
  ; 1[ x: int ]
    ; minsegsumtwodim
      { R: x = (MIN m0,m1, n0, n1: 0 ≤ m0 ≤ m1 ≤ M ∧ 0 ≤ n0 ≤ n1 ≤ N:
             (Ś 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.

To begin with we observe that \( R \Rightarrow x \leq 0 \) since the summation may be over an empty range \( (m_0 = m_1 \lor n_0 = n_1) \). Furthermore we observe that \( R \) is not changed if we change the range for \( m_0 \) and \( m_1 \) into \( 0 \leq m_0 < m_1 \leq M \), since then the summation over the empty rectangle is still included. We adopt this change. (Condition \( M > 1 \) has been chosen, rather than \( M > 0 \), to make this change permissible; \( N > 1 \) has been chosen because, in general, our solution is most attractive for \( N > M \).)

Our next step is to rewrite \( R \) in terms of nested \( \text{MIN}'s \). (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 why \( R \) is rewritten this way.)

\[
R: x = (\text{MIN} m_0: 0 \leq m_0 < M:
(\text{MIN} m_1: m_0 < m_1 \leq M:
(\text{MIN} n_1: 0 \leq n_1 \leq N: \text{MS.} m_0, m_1, n_1)))
\]

\[
\text{MS.} m_0, m_1, n_1 =
(\text{MIN} n_0: 0 \leq n_0 \leq n_1: (\text{SN:} n_0 \leq n < n_1: \text{Q.} m_0, m_1, n))
\]

\[
\text{Q.} m_0, m_1, n = (\text{SM:} m_0 \leq m < m_1: \text{C.} m, n)
\]

Lines 2 through 4 of the above describe the linear minsegsum "in the n-direction" with \( m_0 \) and \( m_1 \) as parameters: the \( m \)-direction has been
"pushed to the sides", i.e. to the outer minimizations and the inner summation respectively.

We next derive —analogously to EWD897— two relations for MS.

\[ \text{MS.} \, m_0. \, m_1. \, 0 = 0 \]  
and for \( 0 \leq n < N \)

\[ \text{MS.} \, m_0. \, m_1. \, (n_1+1) \]

\[ = \text{definition of } \text{MS} \]

\[ (\text{MIN } n_0:0 \leq n_0 \leq n_1+1:(\text{SUM} \, n_0 \leq n < n_1+1:Q \cdot m_0. \, m_1. \, n)) \]

\[ = \{ \text{For } n_0 = n_1 + 1, \text{ the summation yields } 0 \} \]

\[ (\text{MIN } n_0:0 \leq n_0 \leq n_1:(\text{SUM} \, n_0 \leq n < n_1+1:Q \cdot m_0. \, m_1. \, n)) \, \text{ min } 0 \]

\[ = \{ \text{isolation of last term of summation} \} \]

\[ (\text{MIN } n_0:0 \leq n_0 \leq n_1:(\text{SUM} \, n_0 \leq n < n_1:Q \cdot m_0. \, m_1. \, n) \]

\[ + Q \cdot m_0. \, m_1. \, n_1) \, \text{ min } 0 \]

\[ = \{ Q \cdot m_0. \, m_1. \, n_1 \text{ does not depend on } n_0 \text{ and} \]

\[ \text{addition then distributes over minimization} \} \]

\[ (\text{MIN } n_0:0 \leq n_0 \leq n_1:(\text{SUM} \, n_0 \leq n < n_1:Q \cdot m_0. \, m_1. \, n) \]

\[ + Q \cdot m_0. \, m_1. \, n_1) \, \text{ min } 0 \]

\[ = \text{definition of } \text{MS} \]

\[ (\text{MS.} \, m_0. \, m_1. \, n_1 + Q \cdot m_0. \, m_1. \, n_1) \, \text{ min } 0 \]. \hspace{1cm} (1) \]

For \( Q \) we derive (directly from its definition)

\[ Q \cdot m_0. \, m_0. \, n_1 = 0 \] \hspace{1cm} (2)

\[ Q \cdot m_0. \, (m_1+1). \, n_1 = Q \cdot m_0. \, m_1. \, n_1 + C \, m_1. \, n_1 \] \hspace{1cm} (3)
Our rewritten R tells us that for any \( m_0, m_1 \) combination we have to find the minimum of \( MS.m_0.m_1.n_1 \) for all \( n_1 \), while (3) gives a recurrence relation for that sequence of values. The snag is that that recurrence relation contains the term \( Q.m_0.m_1.n_1 \), which - see (3) - satisfies a recurrence relation over \( m_1 \). In order to exploit these two "orthogonal" recurrence relations to the fullest, the program evaluates for each value of \( m_0 \) the \( M-m_0 \) recurrences (1) in synchrony. To this end a local array \( MSv(m; m_0 < m \leq M) \) is introduced, such that whenever \( MSv.m_1 \) is adjusted, its value changes from \( MS.m_0.m_1.n_1 \) to \( MS.m_0.m_1.(n_1+1) \). Furthermore a local scalar \( Qv \) is introduced such that whenever \( Qv \) is adjusted, its value changes from \( Q.m_0.m_1.n_1 \) to \( Q.m_0.(m_1+1).n_1 \).

The annotated program is given below. In the initialization of \( x \), \( x \leq 0 \) is used; in its adjustment we use that \( \min \) is associative.

Because we wish to use each \( MS \)-value as soon as it is computed, i.e. equals \( MS.m_0.m_1.(n_1+1) \), line 2 of the rewritten \( R \) is rewritten once more:

\[
0 \min (\text{MIN } n_1: 0 \leq n_1 < N: \text{ MS. } m_0.m_1.(n_1+1))
\]

where the initial 0 can be taken out and taken care of by the initialization.
\[ \begin{array}{l}
\text{let } \mathbf{m}_0, x = 0; \mathbf{m}_0 := 0 \\
\text{while } \mathbf{m}_0 \neq M \rightarrow \\
\quad \text{let } \mathbf{m}, \mathbf{n} : \text{int} \\
\quad \quad \mathbf{m}_1, \mathbf{n}_1 : \text{int} \\
\quad \quad \mathbf{m}_1 := \mathbf{m}_0; \text{while } \mathbf{m}_1 \neq M \rightarrow \mathbf{m}_1 := \mathbf{m}_1 + 1; \text{MSv.m} := 0 \text{ od} \\
\quad \quad \mathbf{n}_1 := 0 \{ \exists \mathbf{m} : \mathbf{m}_0 \leq \mathbf{m} \leq \mathbf{M} : \text{MSv.m} = \text{MS.m}_0.m.m.n1 \} \\
\quad \text{while } \mathbf{n}_1 \neq N \rightarrow \\
\quad \quad \text{let } \mathbf{Q}_v : \text{int}; \mathbf{Q}_v, \mathbf{m}_1 := 0, \mathbf{m}_0 \{ \mathbf{Q}_v = \mathbf{Q}.m_0.m_1.n1 \} \\
\quad \quad \text{while } \mathbf{m}_1 \neq M \rightarrow \\
\quad \quad \quad \mathbf{Q}_v.m_1 := \mathbf{Q}_v + \mathbf{C}.m_1.n1, \mathbf{m}_1 + 1 \\
\quad \quad \quad \{ \mathbf{Q}_v = \mathbf{Q}.m_0.m_1.n1 \} \\
\quad \quad \quad \text{MSv.m}_1 := (\text{MSv.m}_1 + \mathbf{Q}_v) \min 0 \\
\quad \quad \text{od} \\
\quad \mathbf{n}_1 := \mathbf{n}_1 + 1 \\
\quad \text{od} \\
\text{od} \\
\text{od} \\
\end{array} \]

\[ \]

\textit{Side remark:} Eventually, we came up with a program that works for \( M=0 \vee N=0 \) as well. We could have made our analysis also applicable in that case by defining the minimum of an empty set as "big enough" -0, say-. (End of Side remark.)