

## Maximum subsequence

### Problem

Given a sequence of integer numbers  $x_0, x_1, \dots, x_{N-1}$  (not necessarily positive) find a subsequence  $x_i, \dots, x_{j-1}$  such that the sum of numbers in it is maximum over all subsequences of consecutive elements.

We adopt the same notation used in *Programming in the 1990s*<sup>1</sup> and *Programming, The Derivation of Algorithms*<sup>2</sup>: The notation of function application is the "dot" notation with name of function, followed by arguments, each separated by a dot. The notation of quantified expressions has the operator followed by the bounded variables, then a colon followed by the range for the bounded variables and ended with a colon and the actual expression. So

$$(\sum k : i \leq k < j : x_k)$$

corresponds to the more classical mathematical notation  $\sum_{k=i}^{j-1} x_k$ . For our derivation steps in predicate calculus we will use the following notation:

$$\begin{array}{l} A \\ = \{ \text{reason why A equals B} \} \\ B \\ \leq \{ \text{reason why B is less than C} \} \\ C \end{array}$$

If all the numbers are positive then the maximum sum is the sum of the whole initial sequence. If all the numbers are negative then the maximum sum is 0 (by definition 0 is the sum over an empty range). So the interesting case is a sequence with positive and negative numbers in it.

We hope to find an algorithm that visits every number in the sequence only once, so with runtime  $O(n)$ . Let's introduce some notation: Let's introduce some notation<sup>3</sup> :

$$f.n = (\text{MAX} i, j : 0 \leq i \leq j \leq n : s.i.j)$$

with

$$s.i.j = (\sum k : i \leq k < j : x_k).$$

We will use properties of quantified expressions as covered in Chapter 3 of *Programming in the 1990s*<sup>4</sup>.

<sup>1</sup> Edward Cohen. *Programming in the 1990s, An Introduction to the Calculation of Programs*. Springer-Verlag, 1990

<sup>2</sup> A. Kaldewaij. *Programming, The Derivation of Algorithms*. Prentice Hall, 1990

<sup>3</sup> Our problem can be stated as finding  $f.N$  given  $x_i \in \mathbb{Z}, 0 \leq i < N, N \in \mathbb{N}$ .

<sup>4</sup> Edward Cohen. *Programming in the 1990s, An Introduction to the Calculation of Programs*. Springer-Verlag, 1990

$$\begin{aligned}
& f.N \\
= & \text{< definition of } f \text{ >} \\
& (MAXi, j : 0 \leq i \leq j \leq N : s.i.j) \\
= & \text{< range nesting >} \\
& (MAXj : 0 \leq j \leq N : (MAXi : 0 \leq i \leq j : s.i.j)) \\
= & \text{< defining } p.j = (MAXi : 0 \leq i \leq j : s.i.j) \text{ >} \\
& (MAXj : 0 \leq j \leq N : p.j) \\
= & \text{< range split, 1-point rule >} \\
& (MAXj : 0 \leq j < N : p.j) \max p.N \\
= & \text{< definition of } f \text{ >} \\
& f.(N-1) \max p.N
\end{aligned}$$

We now have a recursive expression for  $f$ , which still depends on a newly introduced function  $p$ . Let's see if we can get a recursive expression for  $p$  too:

$$\begin{aligned}
& p.N \\
= & \text{< definition of } p \text{ >} \\
& (MAXi : 0 \leq i \leq N : s.i.N) \\
= & \text{< range split, 1-point rule >} \\
& (MAXi : 0 \leq i < N : s.i.N) \max s.N.N \\
= & \text{< definition of } s \text{ and } s.N.N = 0 \text{ by definition of sum over empty range >} \\
& (MAXi : 0 \leq i < N : (\sum k : i \leq k < N : x_k)) \max 0 \\
= & \text{< range split in sum >} \\
& (MAXi : 0 \leq i < N : (\sum k : i \leq k < N-1 : x_k) + x_{N-1}) \max 0 \\
= & \text{< + distributes over max >} \\
& (x_{N-1} + (MAXi : 0 \leq i < N : (\sum k : i \leq k < N-1 : x_k)) \max 0 \\
= & \text{< definition of } p \text{ >} \\
& (x_{N-1} + p.(N-1)) \max 0
\end{aligned}$$

So  $f.N = f.(N-1) \max p.N$  and  $p.N = (x_{N-1} + p.(N-1)) \max 0$ . The base cases are  $f.0 = 0$  and  $p.0 = 0$ .

Armed with these recursive relations we can provide a Haskell program that solves the problem:

Listing 1.1: Haskell code

```

maxSum :: [Int] -> (Int, Int)

maxSum (x:xs) = let (a, b) = maxSum xs
                  c = x + b
                  in (max c (max a 0), max c 0)

maxSum [] = (0, 0)

```

The `maxSum` function calculates the tuple  $(f.N, p.N)$ .

## *Bibliography*

Edward Cohen. *Programming in the 1990s, An Introduction to the Calculation of Programs*. Springer-Verlag, 1990.

A. Kaldewaij. *Programming, The Derivation of Algorithms*. Prentice Hall, 1990.