

1

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*¹ and *Programming, The Derivation of Algorithms*²: 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:

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

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.

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

² A. Kaldewaij. *Programming, The Derivation of Algorithms*. Prentice Hall, 1990

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 notation³ :

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

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*⁴.

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

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

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

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:

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

So $f.N = f.(N-1) \text{ max } p.N$ and $p.N = (x_{N-1} + p.(N-1)) \text{ max } 0$. 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.