The Longest Upsequence: a derivation

The other day, Wim Feijen was wondering whether we can *derive* – by calculation – the solution for the problem of the *longest upsequence*. The answer is: "Yes, we can", and in this note I will supply proof. I will do so in a functional setting, but to obtain the (desired) $\mathcal{O}(n * \log \cdot n)$ time complexity I will eventually confine myself to the world of sequential programs, where we have random-access arrays at our disposal. A purely functional solution with the same time complexity is feasible too; this will require some kind of balanced trees, the elaboration of which I consider too much trouble (for now, that is).

The derivation requires two generalizations, which is why I present the solution in three steps. Pursuing these generalizations requires a little courage, because it is not at all clear at the outset that they will successfully lead to a solution. On the other hand, however, there seems to be little else we can do.

0 Specification

A *subsequence* of a given sequence is any sequence obtained by omitting several – zero or more – elements in arbitrary positions from the given sequence, while maintaining the relative order of the remaining elements.

We are only interested in sequences of integers, and for simplicity's sake (and "without loss of generality") I even confine myself to sequences of naturals. This has the technical advantage that the maximum operator has 0 as its identity element, whereas within the integers we are forced to use (the somewhat awkward) $-\infty$. The difference, however, is not essential.

An *upsequence* is a subsequence that is ascending. The problem of the longest upsequence is to (construct a program to) calculate the maximal length over all upsequences of a given sequence. Once this problem has been solved, it can be extended without too much trouble to calculate a *specimen* of a longest upsequence as well.

* * *

We formalize the problem in terms of (finite) lists of naturals: a sequence just is a list. We introduce a relation $\subseteq$ on lists with the intended meaning that $t \subseteq s$ ("t sub s") is the proposition "t is a subsequence of s".

type convention: Variables s, t, x, y denote lists of naturals, and b, c, k, l are naturals.

□

We now postulate that the following recursive definition of $\subseteq$ adequately characterizes all subsequences of a list; as usual, this (implicitly) defines $\subseteq$ as the *strongest* relation having the given properties. Clause (0) expresses that the empty list is the only subsequence of the empty list; (1) expresses that every subsequence of s is a subsequence of $b \triangleright s$ as well, whereas (2) expresses that every extension with b of a subsequence of s yields a subsequence of $b \triangleright s$ too. Thus, all subsequences of s are obtained, for all b, s, t:

(0) $\quad t \subseteq [\,] \;\equiv\; t = [\,]$

(1) $\quad t \subseteq s \;\Rightarrow\; t \subseteq (b \triangleright s)$

(2) $\quad t \subseteq s \;\Rightarrow\; (b \triangleright t) \subseteq (b \triangleright s)$

properties (without proof): Because $\subseteq$ is the strongest of all relations satisfying (0) through (2), this enumeration is exhaustive. As a result, formulae quantified over the subsequences of a list may be rewritten – mainly by range split and dummy transformation – as follows, for any binary operator (and quantor) $\mathbf{q}$, and for all b, s:

(3) $\quad (\mathbf{q}t : t \subseteq [\,] : F{\cdot}t) \;=\; F{\cdot}[\,]$, and:

(4) $\quad (\mathbf{q}t : t \subseteq (b \triangleright s) : F{\cdot}t) \;=\; (\mathbf{q}t : t \subseteq s : F{\cdot}t)\ \mathbf{q}\ (\mathbf{q}t : t \subseteq s : F{\cdot}(b \triangleright t))$.

Property (4) remains valid in the presence of additional conjuncts in the range of the quantification[0].

From (0) and (1) we also obtain (by simple mathematical induction), for all s:

(5) $\quad [\,] \subseteq s$.

□

Ascendingness of sequences can be defined recursively too, as follows. Here we use $b \sqsubseteq s$ ("b under s") as a shorthand for $(\forall i : 0 \leq i < \#s : b \leq s{\cdot}i)$. This relation can be defined recursively too. Thus we obtain, for all b, c, s:

[0] Dit heeft niets met het onderhavige probleem van doen en het is nogal elementair: mij staat zelfs een apart stukje voor ogen, want ik voorzie hier een generalisatie van de range/term-trading rule die (bij mijn weten) nog niet eerder expliciet is geformuleerd!

(6) $asc\cdot[\,] \equiv \mathsf{true}$

(7) $asc\cdot(b \triangleright s) \equiv b \sqsubseteq s \ \wedge \ asc\cdot s$

(8) $c \sqsubseteq [\,] \equiv \mathsf{true}$

(9) $c \sqsubseteq (b \triangleright s) \equiv c \leq b \ \wedge \ c \sqsubseteq s$

Relation $\sqsubseteq$ has these simple properties, for all b, c, s:

(10) $0 \sqsubseteq s$, and:

(11) $c \leq b \ \Rightarrow \ (b \sqsubseteq s \Rightarrow c \sqsubseteq s)$.

We now introduce function *lup* for the length of a longest upsequence of its parameter, which is a list of naturals.

specification: Function *lup* satisfies, for all natural lists s:

(12) $lup\cdot s \ = \ (\mathsf{max}\, t : t \subseteq s \wedge asc\cdot t : \#t)$.

□

1 First step

To satisfy *lup*'s specification we derive, by induction on s:

$lup\cdot[\,]$

= { specification (12) of *lup* }

$(\mathsf{max}\, t : t \subseteq [\,] \wedge asc\cdot t : \#t)$

= { property (3), using definition (6) of *asc* }

$\#[\,]$

= { definition of # }

0 ,

and:

$lup\cdot(b \triangleright s)$

= { specification (12) of *lup* }

$(\mathsf{max}\, t : t \subseteq (b \triangleright s) \wedge asc\cdot t : \#t)$

= { property (4) }

$$
\begin{array}{ll}
 & (\mathsf{max}\, t : t \sqsubseteq s \wedge asc\cdot t : \#t)\ \mathsf{max}\ (\mathsf{max}\, t : t \sqsubseteq s \wedge asc\cdot(b \triangleright t) : \#(b \triangleright t)) \\
= & \quad \{ \text{ specification (12) of } lup, \text{ by Induction Hypothesis } \} \\
 & lup\cdot s\ \mathsf{max}\ (\mathsf{max}\, t : t \sqsubseteq s \wedge asc\cdot(b \triangleright t) : \#(b \triangleright t)) \\
= & \quad \{ \text{ definition (7) of } asc \text{ and definition of } \# \ \} \\
 & lup\cdot s\ \mathsf{max}\ (\mathsf{max}\, t : t \sqsubseteq s \wedge b \sqsubseteq t \wedge asc\cdot t : 1+\#t) \\
= & \quad \{ (1+) \text{ over } \mathsf{max} \text{ (see below) } \} \\
 & lup\cdot s\ \mathsf{max}\ (1 + (\mathsf{max}\, t : t \sqsubseteq s \wedge b \sqsubseteq t \wedge asc\cdot t : \#t)) \quad .
\end{array}
$$

Distribution of $(1+)$ over max is allowed because the range of quantification is non-empty: it contains $[\,]$, as $t \sqsubseteq s$, $b \sqsubseteq t$, and $asc\cdot t$ are true for $t = [\,]$.

2 Second step

Here we are stuck, because of the occurrence of the additional conjunct $b \sqsubseteq t$ in the right-hand side subexpression. The only way out seems to generalize function lup so as to incorporate this. Because $b \sqsubseteq t$ expresses that t is bounded (from below) by b we call the new function blp, with this specification.

specification: Function blp satisfies, for all c and s:

(13) $\quad blp\cdot c\cdot s \ = \ (\mathsf{max}\, t : t \sqsubseteq s \wedge c \sqsubseteq t \wedge asc\cdot t : \#t) \quad .$

□

Function blp is a true generalization of lup: by property (10) we have $0 \sqsubseteq t$ for all t and, therefore, we have, for all s:

(14) $\quad lup\cdot s \ = \ blp\cdot 0\cdot s \quad .$

A direct consequence of property (11) and blp's specification is that blp is monotonic in its first parameter, in the following way, for all b, c, s:

(15) $\quad c \leq b \ \Rightarrow \ blp\cdot b\cdot s \leq blp\cdot c\cdot s \quad .$

* * *

For function blp we now derive, in the same way as we did for lup:

$$
\begin{array}{ll}
 & blp{\cdot}c{\cdot}[\,] \\
= & \quad \{ \text{ specification (13) of } blp \ \} \\
 & (\,\mathsf{max}\, t : t \sqsubseteq [\,] \wedge c \sqsubseteq t \wedge asc{\cdot}t : \#t\,) \\
= & \quad \{ \text{ property (3), using definitions (8) of } \sqsubseteq \text{ and (6) of } asc \ \} \\
 & \#[\,] \\
= & \quad \{ \text{ definition of } \# \ \} \\
 & 0 \ ,
\end{array}
$$

and:

$$
\begin{array}{ll}
 & blp{\cdot}c{\cdot}(b \triangleright s) \\
= & \quad \{ \text{ specification (13) of } blp \ \} \\
 & (\,\mathsf{max}\, t : t \sqsubseteq (b \triangleright s) \wedge c \sqsubseteq t \wedge asc{\cdot}t : \#t\,) \\
= & \quad \{ \text{ property (4) } \} \\
 & (\,\mathsf{max}\, t : t \sqsubseteq s \wedge c \sqsubseteq t \wedge asc{\cdot}t : \#t\,) \ \mathsf{max} \\
 & (\,\mathsf{max}\, t : t \sqsubseteq s \wedge c \sqsubseteq (b \triangleright t) \wedge asc{\cdot}(b \triangleright t) : \#(b \triangleright t)\,) \\
= & \quad \{ \text{ specification (13) of } blp, \text{ by Induction Hypothesis } \} \\
 & blp{\cdot}c{\cdot}s \ \mathsf{max} \ (\,\mathsf{max}\, t : t \sqsubseteq s \wedge c \sqsubseteq (b \triangleright t) \wedge asc{\cdot}(b \triangleright t) : \#(b \triangleright t)\,) \\
= & \quad \{ \text{ definitions (9) of } \sqsubseteq \text{ and (7) of } asc, \text{ and definition of } \# \ \} \\
 & blp{\cdot}c{\cdot}s \ \mathsf{max} \ (\,\mathsf{max}\, t : t \sqsubseteq s \wedge c \leq b \wedge c \sqsubseteq t \wedge b \sqsubseteq t \wedge asc{\cdot}t : 1{+}\#t\,) \ .
\end{array}
$$

The conjunct $c \leq b$ in the last formula is constant – independent of the dummy t –; this calls for case analysis: if $b < c$ the conjunct is false and, hence, the range of quantification is empty. So, the whole quantified formula equals the identity element of max and the result of the derivation is just $blp{\cdot}c{\cdot}s$.

If $c \leq b$ then this conjunct can be eliminated from the quantification's range; with this last formula we continue the calculation as follows:

$$
\begin{array}{ll}
 & (\,\mathsf{max}\, t : t \sqsubseteq s \wedge c \sqsubseteq t \wedge b \sqsubseteq t \wedge asc{\cdot}t : 1{+}\#t\,) \\
= & \quad \{ \ c \leq b \text{: property (11) } \} \\
 & (\,\mathsf{max}\, t : t \sqsubseteq s \wedge b \sqsubseteq t \wedge asc{\cdot}t : 1{+}\#t\,) \\
= & \quad \{ \ (1+) \text{ over } \mathsf{max} \text{ (range is non-empty) } \} \\
 & 1 + (\,\mathsf{max}\, t : t \sqsubseteq s \wedge b \sqsubseteq t \wedge asc{\cdot}t : \#t\,) \\
= & \quad \{ \text{ specification (13) of } blp, \text{ by Induction Hypothesis } \} \\
 & 1 + blp{\cdot}b{\cdot}s \ .
\end{array}
$$

Combining the results of these calculations we obtain this recursive definition:

$$
\begin{array}{lll}
(16) & blp\cdot c\cdot[\,] & = \quad 0 \\
(17) & blp\cdot c\cdot(b \triangleright s) & = \quad \mathsf{if}\ \ b<c \ \ \rightarrow \ \ blp\cdot c\cdot s \\
 & & \quad [\!]\ \ c\leq b \ \ \rightarrow \ \ blp\cdot c\cdot s\ \ \mathsf{max}\ \ (1+blp\cdot b\cdot s) \\
 & & \quad \mathsf{fi}
\end{array}
$$

3 Third step

However nice and simple the definition for blp may be, it is not very attractive from the point of view of efficiency, because of the two recursive occurrences of blp. These two recursive applications have different arguments $-c$ and $b-$ for their first parameter but the argument for the second parameter is the *same* list s. This gives some hope that a further generalization may be feasible.

Apparently, we are not interested in $blp\cdot c\cdot s$ for a *single* natural c but for a whole *set* of naturals. In addition, we are not interested in $blp\cdot c\cdot s$ in its pure form but in expressions of the shape $i+blp\cdot c\cdot s$, for several, hopefully *consecutive*, natural i. After all, we can rewrite $blp\cdot c\cdot s$ as $0+blp\cdot c\cdot s$ and, hence, the expression $blp\cdot c\cdot s\ \ \mathsf{max}\ \ (1+blp\cdot b\cdot s)$ can also be written as:

$$(0+blp\cdot c\cdot s)\ \ \mathsf{max}\ \ (1+blp\cdot b\cdot s)\quad .$$

Further unfolding of the subexpression $1+blp\cdot b\cdot s$ may be expected to yield an additional $1+$, which in combination with the $1+$ already present will yield $2+$, and so on: my hope that the natural numbers thus emerging remain consecutive seems (somewhat) justified.

The next step is quite bold, but I see no alternatives. The above considerations buffer the shock of invention to some extent, but whether this generalization leads to an efficient solution remains to be seen. The only way to know, however, is by giving it a try.

Inspired by the shape of the definition blp, we introduce a new function $blps$, with natural parameter c replaced by a list x of naturals. As we will see, in all applications of $blps$ list x is non-empty and ascending. In addition, x will also satisfy $x_0=0$. (To save a few parentheses I use subscription as an alternative to function application: x_i is the same as $x\cdot i$.)

specification: Function $blps$ satisfies, for all s and for all non-empty and ascending x with $x_0=0$:

$$(18)\qquad blps\cdot x\cdot s \ \ = \ \ (\,\mathsf{max}\, i : 0\leq i<\#x :\ i+blp\cdot x_i\cdot s\,)\quad .$$

□

As to its relevance for our original function lup, we calculate:

$$
\begin{array}{ll}
 & lup\cdot s \\
= & \quad \{ \text{ generalization to } blp\text{: property (14) } \} \\
 & blp\cdot 0\cdot s \\
= & \quad \{ \text{ a little (list) algebra and the one-point rule } \} \\
 & (\,\mathsf{max}\, i : 0\le i<1 :\ i+blp\cdot [0]_i\cdot s\,) \\
= & \quad \{ \text{ specification (18) of } blps \} \\
 & blps\cdot [0]\cdot s \quad .
\end{array}
$$

Hence, our original function lup can be defined in terms of $blps$ by:

$$lup\cdot s \;=\; blps\cdot [0]\cdot s \quad .$$

Note that list $[0]$ is non-empty and ascending, and that its first element is 0.

* * *

Now we try to derive a recursive definition for $blps$; in this derivation we use the definition for blp. As x is a non-empty list, I use $l+1$ for the length of x (where l, $0\le l$, is an auxiliary variable):

$$
\begin{array}{ll}
 & blps\cdot x\cdot [\,] \\
= & \quad \{ \text{ specification (18) of } blps \} \\
 & (\,\mathsf{max}\, i : 0\le i\le l :\ i+blp\cdot x_i\cdot [\,]\,) \\
= & \quad \{ \text{ definition (16) of } blp \} \\
 & (\,\mathsf{max}\, i : 0\le i\le l :\ i\,) \\
= & \quad \{ \ \mathsf{max}\text{-calculus } (0\le l) \ \} \\
 & l \quad .
\end{array}
$$

The derivation for the case $blps\cdot x\cdot (b\triangleright s)$ requires some preparation, because of the case analysis in definition (17). Here this amount to distinguishing the cases $x_i\le b$ and $b<x_i$; now, because x is ascending an (even unique) value k exists with the following properties:

(19) $\quad 1\le k\le l+1$

(20) $\quad (\forall i : 0\le i<k : x_i\le b\,)$

(21) $\quad (\forall i : k\le i\le l : b<x_i\,)$

That $1 \leq k$ follows from the precondition $x_0 = 0$: as a consequence, the range of quantification in (20) is non-empty. Now we derive:

$$
\begin{array}{ll}
 & blps \cdot x \cdot (b \triangleright s) \\
= & \quad \{ \text{ specification (18) of } blps \ \} \\
 & (\mathsf{max}\, i : 0 \leq i \leq l : \ i + blp \cdot x_i \cdot (b \triangleright s)\ ) \\
= & \quad \{ \text{ range split, preparing for use of definition (17) } \} \\
 & (\mathsf{max}\, i : 0 \leq i < k : \ i + blp \cdot x_i \cdot (b \triangleright s)\ )\ \mathsf{max} \\
 & (\mathsf{max}\, i : k \leq i \leq l : \ i + blp \cdot x_i \cdot (b \triangleright s)\ ) \\
= & \quad \{ \text{ definition (17) of } blp \text{, using (20) and (21) } \} \\
 & (\mathsf{max}\, i : 0 \leq i < k : \ i + (\ blp \cdot x_i \cdot s\ \mathsf{max}\ (1 + blp \cdot b \cdot s)\ )\ )\ \mathsf{max} \\
 & (\mathsf{max}\, i : k \leq i \leq l : \ i + blp \cdot x_i \cdot s\ ) \\
= & \quad \{ \ (i+) \text{ over (the innermost) } \mathsf{max} \ \} \\
 & (\mathsf{max}\, i : 0 \leq i < k : \ (i + blp \cdot x_i \cdot s)\ \mathsf{max}\ (i + 1 + blp \cdot b \cdot s)\ )\ \mathsf{max} \\
 & (\mathsf{max}\, i : k \leq i \leq l : \ i + blp \cdot x_i \cdot s\ ) \\
= & \quad \{ \text{ term split! } (blp \cdot b \cdot s \text{ does not depend on } i\ ) \ \} \\
 & (\mathsf{max}\, i : 0 \leq i < k : \ i + blp \cdot x_i \cdot s\ )\ \mathsf{max} \\
 & (\mathsf{max}\, i : 0 \leq i < k : \ i + 1 + blp \cdot b \cdot s\ )\ \mathsf{max} \\
 & (\mathsf{max}\, i : k \leq i \leq l : \ i + blp \cdot x_i \cdot s\ ) \\
= & \quad \{ \text{ range unsplit on the first and the last terms } \} \\
 & (\mathsf{max}\, i : 0 \leq i \leq l : \ i + blp \cdot x_i \cdot s\ )\ \mathsf{max}\ (\mathsf{max}\, i : 0 \leq i < k : \ i + 1 + blp \cdot b \cdot s\ ) \\
= & \quad \{ \ \mathsf{max}\text{-calculus, using } 1 \leq k \ \} \\
(22) & \quad (\mathsf{max}\, i : 0 \leq i \leq l : \ i + blp \cdot x_i \cdot s\ )\ \mathsf{max}\ (\ k + blp \cdot b \cdot s\ )\ \ .
\end{array}
$$

To be able to proceed we need some
case analysis:

- If $k \leq l$ then list x has an element x_k. We now define a new list y, of length $l+1$, thus:

  $$(\forall i : 0 \leq i \leq l \wedge i \neq k : y_i = x_i\ ) \ \ \wedge \ \ y_k = b \ \ .$$

  So, y is x where element x_k has been replaced by b. Property (21) implies $b < x_k$; hence, by (15), we have $blp \cdot x_k \cdot s \leq blp \cdot b \cdot s$ and, hence, that $(\ k + blp \cdot x_k \cdot s\ )\ \mathsf{max}\ (\ k + blp \cdot b \cdot s\ )$ equals $k + blp \cdot b \cdot s$. From this

we conclude that result (22) can be reformulated in terms of y as:

$$(\,\mathsf{max}\, i : 0 \le i \le l : i + blp \cdot y_i \cdot s\,) \quad .$$

- If $k = l+1$ then list x has no element x_k, as its last element is x_l. We now define a new list y, of length $l+2$, thus:

  $$(\forall i : 0 \le i \le l : y_i = x_i\,) \;\wedge\; y_{l+1} = b \quad .$$

  So, in this case y is just x extended with b. Now result (22) can be reformulated in terms of y as:

  $$(\,\mathsf{max}\, i : 0 \le i \le l+1 : i + blp \cdot y_i \cdot s\,) \quad .$$

□

In either case, as a result of properties (19) through (21) and the assumptions about x, list y is non-empty, is ascending, and satisfies $y_0 = 0$. And in either case, the above calculation can now be continued:

$$\begin{array}{ll} & (\,\mathsf{max}\, i : 0 \le i \le l : i + blp \cdot x_i \cdot s\,) \;\mathsf{max}\; (k + blp \cdot b \cdot s) \\ = & \quad \{ \text{ combining the results of the above case analysis } \} \\ & (\,\mathsf{max}\, i : 0 \le i < \#y : i + blp \cdot y_i \cdot s\,) \\ = & \quad \{ \text{ specification (18) of } blps\text{, by Ind. Hyp. } \} \\ & blps \cdot y \cdot s \quad . \end{array}$$

As the above case analysis shows, list y is a function of natural b and list x. Calling this function *adjust* we formulate this relation as:

$$y \;=\; adjust \cdot b \cdot x \quad .$$

The specification of *adjust* is then given by the above case analysis, and in terms of it we now obtain the following recursive definition for *blps*:

$$\begin{array}{lcl} blps \cdot x \cdot [\,] & = & \#x - 1 \\ blps \cdot x \cdot (b \triangleright s) & = & blps \cdot y \cdot s \;\; \mathsf{whr}\; y = adjust \cdot b \cdot x \;\mathsf{end} \end{array}$$

4 Implementation Issues

The computation of $adjust \cdot b \cdot x$ requires the computation of a natural k with properties (19) through (21). If list x is implemented as a random-access array this computation requires no more than $\mathcal{O}(\log \cdot l)$ steps, namely by means

of a Binary Search. Once this k is available, the subsequent modification of x into y –that is, the assignment $x := adjust{\cdot}b{\cdot}x$– is an $\mathcal{O}(1)$ operation, as it amounts to a simple assignment $x_k, l := b, l \mathsf{max} k$. Hence, a sequential implementation of the whole algorithm has time complexity $\mathcal{O}(n * \log{\cdot}n)$, where n is the size of the input.

* * *

Apart from its first element, which is invariantly 0, list x is a specimen of a longest subsequence of the "part of the list already processed". This can be rendered formally, either by adding an extra invariant to the sequential program, or by equipping function *blps* with an additional parameter that represents the "part of the list already processed". Calling this function (with the extra parameter) *blps1*, its definition becomes –with $x := 0 \triangleright x$–:

$$
\begin{array}{lcl}
blps1{\cdot}t{\cdot}(0 \triangleright x){\cdot}[\,] & = & x \\
blps1{\cdot}t{\cdot}(0 \triangleright x){\cdot}(b \triangleright s) & = & blps1{\cdot}(t \triangleleft b){\cdot}(0 \triangleright y){\cdot}s \\
 & & \mathsf{whr}\ \ 0 \triangleright y \ = \ adjust{\cdot}b{\cdot}(0 \triangleright x)\ \ \mathsf{end}
\end{array}
$$

Now, if in an application $blps1{\cdot}t{\cdot}(0 \triangleright x){\cdot}(b \triangleright s)$ it is given that x is a longest upsequence of t, then y, with $0 \triangleright y = adjust{\cdot}b{\cdot}(0 \triangleright x)$, is a longest upsequence of $t \triangleleft b$ in the recursive application $blps1{\cdot}(t \triangleleft b){\cdot}(0 \triangleright y){\cdot}s$. Thus, we can prove that $blps1{\cdot}[\,]{\cdot}[0]{\cdot}s$ yields a specimen of a longest upsequence of s (but I will not do so here).

Eindhoven, 11 april 2003

Rob R. Hoogerwoord
department of mathematics and computing science
Eindhoven University of Technology
postbus 513
5600 MB Eindhoven