OTE: Ohjelmointitekniikka
Programming Techniques

course homepage: `http://www.cs.uku.fi/~mnykanen/OTE/`

Week 46/2008

Solution sketch 1.

(a) This initial subpart is the whole input.

(b) On line 6, $b > A[m]$ by line 4. Then the part $A[l \ldots m]$ cannot contain any b since A is nondescending. Hence dropping this part by $l \leftarrow m+1$ is allowed without dropping any place where b might be.

Solution sketch 2. By lines 2 and 3, $l \leq m < u$ since it is the midpoint of the part $A[l \ldots u]$ between l and u, rounded down if necessary. Hence line 5 strictly decreases u, as it should. Similarly, line 6 strictly increases l, as it should.

Solution sketch 3. The lectures and the two exercises above showed the correctness of the algorithm in question, if $l = u$ is true on line 7.

(a) The correctness argument holds whever $l \leq u$ in the beginning of the **while** loop. Is there an input where $l > u$ in the beginning? There is: the empty array has $N = 0$.

(b) The simplest fix would be to treat this special input separately and directly:

if $N = 0$
  then $r \leftarrow 0$
  else our original algorithm.

(c) The original argument works for all nonempty inputs, and hence suffices for the **else** branch. The **then** branch in turn handles correctly the one input which it does not.

Solution sketch 4.

(a) See Backhouse (2003, solution to Exercise 4.6).

(b) Probably because then `centre` = `lo`, so assigning the former into the latter would not be progress, but instead would leave the part as it was.

(c) By setting l to m *plus one* which guarantees that even the two-item case does progress further into the one-item case.

Solution sketch 5.

(a) The proof of Claim 1 for line 5 in the lectures divided further into two cases: $A[m] = b$ and $A[m] < b$, so there.

(b) Just treat them separately in the code as well:

$$
\begin{array}{l}
\textbf{if } b = A[m] \\
\quad \textbf{then } l \leftarrow m; u \leftarrow m \\
\textbf{elseif } b < A[m] \\
\quad \textbf{then } u \leftarrow m \\
\quad \textbf{else } \ l \leftarrow m + 1
\end{array}
$$

(c) By simply adding the argument that if $b = A[m]$ then the one-element part $A[m \dots m]$ does indeed contain b and hence satisfies Claim 1 too.

(d) The "right" choice is the one for which you can argue better, whichever that is...

References

Roland Backhouse. *Program Construction: Calculating Implementations from Specifications.* Wiley, 2003.