SMALL PROGRAMMING EXERCISES 11

M. REM

Department of Mathematics and Computing Science, Eindhoven University of Technology,
5600 MB Eindhoven, Netherlands

Of Exercise 21, appearing in Small Programming Exercises 8, I wrote that it “seems to belong to the folklore”. In the meantime, however, I have been informed of its origins: Prof. E.M. Reingold gave it as one of the problems for the final exam on combinatorial algorithms at the University of Illinois on December 18, 1973. He had made up the problem while driving from Urbana to Chicago about two weeks before.

We have three new exercises, all allowing very compact solutions. In the first exercise the maximal length of a certain type of array segment has to be determined. The problem has a surprisingly simple solution, which is linear in the size of the array and does not use auxiliary arrays. I owe this problem to A. Kaldewaij.

Exercise 28 addresses the well-known problem of the longest common subsequence of two arrays. In this case, however, we do not require that a longest common subsequence be produced: we are interested in its length only. This makes the problem much simpler. One auxiliary array suffices for a solution whose computation time is proportional to the product of the sizes of the arrays given.

The binomial coefficients may be grouped into a triangle that is known as Pascal’s triangle. Since \( \binom{i}{1} = i \), each positive integer occurs in at least one row of this triangle. Exercise 29 concerns determining the topmost row in which a given number \( N \) occurs. It allows, without introducing auxiliary arrays, a solution whose execution time is proportional to the row number computed. I owe this exercise to J.L.A. van de Snepscheut.

Exercise 27: A-segments

With \( AS(p, q) \) denoting

\[
(Ai: p \leq i \leq q: X(i) \leq \text{abs}(X(q)))
\]

for \( 0 \leq p \leq q \leq N \), we are requested to find a statement list \( S \) such that

\[
\begin{align*}
\{N: \text{int}; \{N \geq 0\} \\
X(i): 0 \leq i \leq N): \text{array of int};& \\
\{r: \text{int}; \\
S \\
\{r = (\text{MAX } p, q: 0 \leq p \leq q \leq N \wedge AS(p, q): q - p + 1)\}
\end{align*}
\]
Exercise 28: Length of a longest common subsequence

An array $X(i: 0 \leq i < M)$ is a sequence of $M$ elements. A subsequence of $X$ is obtained by removing zero or more elements from $X$ and leaving the others in the original order. A common subsequence of arrays $X$ and $Y$ is a sequence that is both a subsequence of $X$ and a subsequence of $Y$. We have to determine $S$ such that

$$\forall [M, N: \text{int}; \{0 \leq M \leq N\}, X(i: 0 \leq i < M), Y(j: 0 \leq j < N): \text{array of int};$$

$$\forall [r: \text{int};$$

$$S$$

$$\{r = l(M, N)\}$$

$$]]$$

where $l(m, n)$ denotes, for $0 \leq m \leq M$ and $0 \leq n \leq N$, the maximal length of a common subsequence of $X(i: 0 \leq i < m)$ and $Y(j: 0 \leq j < n)$. (Hint: determine a recurrence relation for $l(m, n)$.)

Exercise 29: A search in Pascal's triangle

Find a statement list $S$ such that

$$\forall [N: \text{int}; \{N \geq 1\},$$

$$\forall [k: \text{int};$$

$$S$$

$$\{k = (\text{MIN } x, y: 0 \leq x \leq y \wedge N = (\text{MIN } x, y))\}$$

$$]]$$

Solution of Exercise 25 (Strahler number of a binary tree)

The Strahler number of an empty binary tree is 0. If tree $T$ has subtrees $T_0$ and $T_1$, its Strahler number $\sigma(T)$ is given by

$$\sigma(T) = \begin{cases} \sigma(T_0) \lor \sigma(T_1) & \text{if } \sigma(T_0) \neq \sigma(T_1), \\ \sigma(T_0) + 1 & \text{if } \sigma(T_0) = \sigma(T_1). \end{cases}$$

A binary tree with vertices $0, 1, \ldots, N - 1$, of which $0$ is the root, is recorded in array $v(i: 1 \leq i < N)$, where $v(i)$ is the ‘father vertex’ of vertex $i$:

(Ai: $1 \leq i < N$: the tree with vertex $v(i)$ as its root has a subtree with vertex $i$ as its root)

We have to determine $S$ such that

$$\forall [N: \text{int}; \{N \geq 1\},$$

$$v(i: 1 \leq i < N): \text{array of int};$$
We may associate a Strahler number with each vertex, viz. the Strahler number of the tree of which this vertex is the root. A leaf is a vertex with two empty subtrees. Since an empty tree has Strahler number 0, each leaf has Strahler number 1. We compute the Strahler numbers of all vertices, starting with the leaves. The Strahler number of a vertex can be computed as soon as the Strahler numbers of its subtrees have been determined.

As usual, we partition the set of vertices into sets $V_0$, $V_1$, and $V_2$. Initially $V_2$ is empty and $V_1$ contains all leaves. Call a subtree black if either it is empty or its root is in $V_2$ (and nonblack, otherwise). We introduce auxiliary arrays $t$ and $q$. Array $t$ records for each vertex the number of nonblack subtrees ($0 \leq t(i) \leq 2$). If $t(i) = 1$ the computation of the Strahler number of vertex $i$ has partly progressed:

$$ P: \quad t(i) = 1 \Rightarrow q(i) = \text{Strahler number of the black subtree of } i $$
$$ \land t(i) = 0 \Rightarrow q(i) = \text{Strahler number of vertex } i. $$

Set $V_1$ contains all vertices $i$, $i \notin V_2$, with $t(i) = 0$:

$$ V_0 = \{i \in V_0 \cup V_1 \mid t(i) \geq 1\}, $$
$$ V_1 = \{i \in V_0 \cup V_1 \mid t(i) = 0\}. $$

Set $V_1$ is, as usual, recorded in segment $v1(i: 0 \leq i < nv1)$ of array $v1(i: 0 \leq i < N)$.

The initial value of array $t$ is given by

$$ (Ai: 0 \leq i < N: t(i) = (Nh: 1 \leq h < N: v(h) = i)). $$

Per step of the repetition a vertex $j \in V1 \setminus \{0\}$ is selected and moved to $V_2$. Let $k$ be its ‘father vertex’. By moving $j$ to $V_2$ the number of nonblack subtrees of $k$ decreases by 1, by which $t(k)$ becomes 1 or 0. In both cases the invariance of $P$ requires an update of $q(k)$. If $t(k) = 0$ vertex $k$ is moved from $V_0$ to $V_1$. The repetition terminates when the root (vertex 0) is moved to $V_1$.

The code of our solution is

$$ S: \quad [nv1: \text{int}; \\
q, t, v1(i: 0 \leq i < N): \text{array of int}; \\
\text{INIT} \\
; \text{do } t(0) \neq 0 \\
\rightarrow [j, k: \text{int}; nv1 := nv1 - 1; j := v1(nv1); k := v(j) \\
; t: (k) = t(k) - 1 \] $$
where the initialization is given by

\[
\text{INIT} \quad \begin{cases} 
\text{i: int; } i := 0; \text{do } i \neq N \rightarrow t:(i) = 0; i := i + 1 \od \ \\
\text{i := 1; do } i \neq N \rightarrow t:(v(i)) = t(v(i)) + 1; i := i + 1 \od \ \\
\text{i, nv1 := = 0,0} \ \\
\text{do } i \neq N \ \\
\quad \rightarrow \text{if } t(i) = 0 \rightarrow q:(i) = 1; v1:(nv1) = i; nv1 := nv1 + 1 \ \\
\quad \rightarrow \text{if } t(i) = 1 \rightarrow q:(i) = 0 \ \\
\quad \rightarrow \text{if } t(i) = 2 \rightarrow \text{skip} \ \\
\quad \od \ \\
\text{i := i + 1} \ \\
\od \ 
\end{cases}
\]

Per step of the repetition one vertex is moved from \(V1\) to \(V2\). The computation time of our solution is, consequently, \(O(N)\).

**Solution of Exercise 26 (K-sequel of an acyclic digraph)**

The \(K\)-sequel of a weighted acyclic directed graph is the set of all vertices \(j\) for which the weight of each path from a source to \(j\) is at least \(K\).

We have to solve \(S\) in

\[
\begin{align*}
[N, M, K: \text{int}; \{N \geq 1 \land M \geq 0 \land K \geq 1}\} \\
b(j: 0 \leq j \leq N): \text{array of int; } \\
e, w(i: 0 \leq i < M): \text{array of int;} \\
\{\text{suc} (G, b, e) \land G \text{ acyclic } \land (Ai: 0 \leq i < M: w(i) \geq 1)}\} \\
[a(j: 0 \leq j < N): \text{array of bool}; \\
S \{((Aj: 0 \leq j < N: a(j) \equiv (j \text{ in the } K\text{-sequel of } G))\} \end{align*}
\]

Again we partition the vertex set into three (possibly empty) subsets \(V0\), \(V1\), and \(V2\). Let a **black path** be a path from a source of which all vertices, except possibly
the last one, are in V2. We introduce an integer array $q(j: 0 \leq j < N)$ and maintain as an invariant

$$P0: \quad (A j: 0 \leq j < N: q(j) = \text{minimum of the weights of the black paths to vertex } j).$$

From $P0$ we conclude that all sources $j$ have $q(j) = 0$. The vertices $j$ for which $q(j) \geq K$ constitute set $V0$, which is also recorded in boolean array $a$:

$$P1: \quad V0 = \{j | q(j) \geq K\} = \{j | a(j)\}.$$

Call the vertices in $V0$, $V1$, and $V2$ white, grey, and black respectively. Initially the sources are grey and all other vertices are white. The latter have, according to $P0$, their (initial) $q$-values at $\inf$.

Per step of the repetition an arbitrary grey vertex is coloured black. This requires, by $P0$, for each successor $k$ of $j$ an adjustment of $q(k)$. If this results for a white successor $k$ in $q(k) < K$ vertex $k$ is coloured grey. The repetition terminates when there are no grey vertices left. Consider in that case the subgraph induced by the white vertices. Its sources $j$ have $q(j) \geq K$. As a consequence, all white vertices then belong to the $K$-sequel and, by $P1$, array $a$ has the appropriate final value.

We can now code our solution. As in the preceding exercise, set $V1$ is recorded in array segment $v1(j: 0 \leq j < nv1)$.

$$S: \quad \begin{cases} \text{[nv1: int];} \\ q, v1(j: 0 \leq j < N): \text{array of int}; \\ \text{INIT} \\ \text{do} \begin{cases} \text{nvl} \neq 0 \\ \begin{cases} \text{[j, i: int; nvl := nvl - 1; j := v1(nvl)} \\ i := b(j) \\ \text{do} \begin{cases} \text{i \neq b(j + 1)} \\ \begin{cases} \text{[k: int; k := e(i); q(k) = q(k) min(q(j) + w(i))} \\ \text{if} \ a(k) \land q(k) < K \rightarrow v11(nv11) = k ; \ nv11 := nv11 + 1 \\ \text{a(k) = false} \\ \text{\square neg a(k) \lor q(k) \geq K \rightarrow skip} \\ \text{fi} \\ \text{i := i + 1} \end{cases} \\ \text{od} \\ \text{od} \end{cases} \\ \text{[i := i + 1} \end{cases} \\ \text{ od} \end{cases} \\ \text{INIT is given by} \\ \text{INIT: \begin{cases} \text{[j, i: int;} \\ t(j: 0 \leq j < N): \text{array of int}; \\ j := 0 ; \text{do} \ j \neq N \rightarrow t(j) = 0 ; \ j := j + 1 \text{ od} \end{cases}}\end{cases}$$
This is an example of a computation in which, although the graph is weighted, an arbitrary grey vertex may be selected to be coloured black. This keeps the computation time of the solution linear in the number of vertices and arcs of the graph.