ALGORITHM 81
ECONOMISING A SEQUENCE 1

BRIAN H. MAYOH
Digital Computer Laboratory, University of Illinois, Urbana, Ill.

```
procedure ECONOMISER 1 (desired property, costs, n, C);
  array costs;  integer n;
    Boolean procedure desired property;
  Boolean array C;
begin comment Given a finite, monotonely increasing
  sequence of positive numbers, looked upon as prices, ECONO-
  MISER 1 selects the cheapest subsequence with a given prop-
  erty. The formal parameters are: Desired property, a function
  designator to answer the question: Does the subsequence held
  in array C possess the required property? n is (number of ele-
  ments in the sequence) + 1.  Costs is an array of size [1:n].
  Costs[1] to costs[n−1] hold the numbers of the sequence and
  costs[n] is any arbitrary number greater than the sum of all
  other elements of costs.  C is an array of the same size and indi-
  cates a subsequence by the rule: C[i] ≡ element i of the original
  sequence is in the subsequence. At exit from ECONOMISER 1,
  C indicates the cheapest subsequence. It is supposed that the
  original sequence has the desired property.;
  integer d, j, k, ℓ;  real i;
  for j := 1 step 1 until n do C[j] := j = 1; d := 0;
  reenter:  d := d+1;
  INSIDE:  begin own real array prices [1:d];
              own Boolean array alternatives[1:d, 1:n];
              procedure ENTER SUCCESSORS;
              begin k := n−1;
                A:  if ¬ C[k] then
                    begin k := k−1;  go to A end;  i := 0;
                    for j := 1 step 1 until n do
                    begin alternatives[ℓ,j]
                      := j ≠ k ∧ j ≠ k−1 ≡ C[j];
                      if alternatives[ℓ,j] then
                      i := i + costs[j]
                    end;
                B:  k := k−1;
                    go to if k = 0 then find cheapest
                      else if C[k] then (if k=1 then
                        find cheapest else B)
                      else if k=1 then E
                        else if C[k−1] then D
                      else find cheapest;
                D:  C[k−1] := false;
                E:  C[k] := true;  go to reenter
              end of ENTER SUCCESSORS;
              i := 0;  for j := 1 step 1 until n do
              begin alternatives[d,j] := C[j];  if C[j] then
                i := i + costs[j]
              end;  prices[d] := i;
            find cheapest:  i := 0;  for j := 1 step 1 until d do
              begin if prices[j] < i then
                begin ℓ := j;  i := prices[ℓ] end
              end;
              for j := 1 step 1 until n do
              C[j] := alternatives[ℓ,j];
              if ¬ desired property then
                ENTER SUCCESSORS
            end of INSIDE;
end of ECONOMISER 1.
```