ALGORITHM 76
SORTING PROCEDURES
IVAN FLORES
Private Consultant, Norwalk, Connecticut

comment The following ALGOL 60 algorithms are procedures for
the sorting of records stored within the memory of the computer.
These procedures are described in detail, flow-charted, com-
pared, and contrasted in ‘‘Analysis of Internal Computer Sort-
ing’’ by Ivan Flores (J. ACM 8 (Jan. 1961)). Although sorting is
usually a business computer application, it can be described
completely in ALGOL if we stretch our imagination a little.
Sorting is ordering with respect to a key contained within the
record. If the key is the active record, the sorting is trivial.
A means is required to extract the key from the record. This is
essentially string manipulation, for which no provision, as yet,
has been made in ALGOL. We circumambulate this difficulty by
defining an integer procedure K(I) which “creates” a key
from the record, I. ALGOL does provide for machine language
code substitutions, which is one way to think of K(I).
This could be more accurately represented by using the string nota-
tion proposed by Julien Green [‘‘Remarks on ALGOL and Sym-
bol Manipulation,’’ Comm. ACM 2 (Sept. 1959), 23-27]. The
function \texttt{sub} $(S,j,g)$ represents the procedure, $K(I)$. $S$

\texttt{sort} corresponds to the record $I$, $I$ corresponds to the starting position
of the key and $g$ corresponds to the length of the key. Both $i$ and $g$
are values which must be specified when the sort procedure
is called for as a statement instead of a declaration.

Another factor, which might vex some, is that the key might
be alphabetic instead of numeric. Then, of course, K(I) would
not be integer. It would, however, be string when such is defined
eventually. Note, also, that keys are frequently compared.
This is done using the ordering relations “$>$” for “greater than,”
etc. These are not really defined in the ALGOL statement [NAUR,
PETER, ET AL. “Report on the Algorithmic Language ALGOL
60″. Comm. ACM 3 (May 1960), 294-314]. They can simply be
defined so that $Z > Y > \cdots > A > 9 > \cdots > 1 > 0$. Also the
assignment $X[I] := x$ should be interpreted as “Assign the key
‘x’ which is larger than any other key.” For any sort procedure
(I,N,S), “I” is the set of unsorted records, “N” is their
number, and “S” is the sorted set of records.

Caution, these algorithms were developed purely for the love
of it: No one was available with the combined knowledge of
sorting and ALGOL to check this work. Hence each algorithm
should be carefully checked before use. I will be glad to answer
any questions which may arise;

Sort insert (I,N,S); value N; array I[1:N], S[1:N];
\textbf{integer procedure } K(I); \textbf{integer } N;
begin \textbf{integer } i, j, k;
\textbf{S[I]} := \textbf{I[I]};
for \textbf{i} := 2 \textbf{step 1} until \textbf{N} do begin
for \textbf{j} := I-I, I-1 \textbf{while } K(I[I]) > K(S[I]) \textbf{do}
for \textbf{k} := \textbf{i} \textbf{step 1} until \textbf{j} + 1 do
\textbf{S[k]} := \textbf{S[K-1]};
\textbf{S}[j + 1] := \textbf{I[I]} \textbf{end end}
end

Sort count (I,N,S); value N; array I[1:N], S[1:N];
\textbf{integer procedure } K(I); \textbf{integer } N;
begin \textbf{integer array } C(I[N]); \textbf{integer } i, j;
for \textbf{i} := 1 \textbf{step 1} until \textbf{N} do \textbf{C[I]} := 0;
for \textbf{i} := 2 \textbf{step 1} until \textbf{N} do
for \textbf{j} := 1 \textbf{step 1} until \textbf{i} + 1 do
\textbf{if } K(I[I]) > K(I[I]) \textbf{then } \textbf{C[I]} := \textbf{C[I]} + 1
\textbf{else } \textbf{C[I]} := \textbf{C[I]} + 1;
for \textbf{i} := 1 \textbf{step 1} until \textbf{N} do
\textbf{S}[C[I]] := \textbf{I}[I]
end

Sort select (I,N,S); value N; array I[1:N], S[1:N];
\textbf{integer procedure } K(I); \textbf{integer } N;
begin \textbf{integer } i, j, k;
for \textbf{i} := 1 \textbf{step 1} until \textbf{N} do begin
\textbf{h} := \textbf{K(I[I])};
for \textbf{j} := 2 \textbf{step 1} until \textbf{N} do
\textbf{if } \textbf{h} := \textbf{K(I[I])} \textbf{then begin } \textbf{h} := \textbf{K(I[I])}; \textbf{A} := \textbf{j}\textbf{ end;
S[I] := I[A];
I[A] := z \textbf{end end}
end

Sort select exchange (I,N,S); value N; array I[1:N], S[1:N];
\textbf{integer procedure } K(I); \textbf{integer } N;
begin \textbf{integer } i, j, h; \textbf{real } T;
for \textbf{i} := 1 \textbf{step 1} until \textbf{N} do begin
\textbf{H} := \textbf{K(I[I])}; \textbf{h} := \textbf{i};
for \textbf{j} := i + 1 \textbf{step 1} until \textbf{N} do
\textbf{if } \textbf{K(I[I])} < \textbf{H} \textbf{then begin
\textbf{H} := \textbf{K(I[I])}; \textbf{h} := \textbf{j}\textbf{ end;
\textbf{T} := \textbf{I[h]; \textbf{I}[i] := \textbf{I}[h]; \textbf{I}[A] := \textbf{T}\textbf{ end end
end

Sort binary insert (I,N,S); value N; array I[1:N], S[1:N];
\textbf{integer procedure } K(I); \textbf{integer } N;
begin \textbf{integer } i, j, l;
\textbf{if } \textbf{K(I[I])} < \textbf{K(I[I])} \textbf{then begin
start: \textbf{for } i := 3 \textbf{step 1} until \textbf{N} do begin
\textbf{j} := (i + 1) + 2;
\textbf{find spot:} \textbf{for } k := (i + 1) + 2, (k + 1) + 2 \textbf{while } \textbf{k} > \textbf{i} \textbf{do
\textbf{if } \textbf{K(I[I])} < \textbf{K(S[j])} \textbf{then begin } \textbf{j} := \textbf{j} - \textbf{k}
\textbf{else } \textbf{j} := \textbf{j} + \textbf{k};
\textbf{if } \textbf{K(I[I])} \textbf{\geq K(S[j])} \textbf{then begin } \textbf{j} := \textbf{j} - 1;
\textbf{move items: } \textbf{for } l := \textbf{i} \textbf{step 1} until \textbf{j} + 1 do
\textbf{S}[l + 1] := \textbf{S}[l];
\textbf{enter this one: S}[j] := \textbf{I}[i]\textbf{ end end
\textbf{Sort address calculation (I,N,S,F); value N;
array \textbf{S}[1:M], \textbf{I}[1:N]; \textbf{integer procedure } F(K, I); \textbf{integer } N,M;\begin{array}{l}
\textbf{begin } i, j, G, H, F, M; \textbf{M} := \textbf{entier}(2.5 \times \textbf{N})
\textbf{for } i := 1 \textbf{step 1} until \textbf{M} do \textbf{S}[i] := 0;
\textbf{for } i := 1 \textbf{step 1} until \textbf{N} do begin
\textbf{F} := \textbf{F}(\textbf{K}(\textbf{I}[\textbf{I}]));
\textbf{if } \textbf{S}[\textbf{F}] = 0 \textbf{then begin } \textbf{S}[\textbf{F}] := \textbf{I}[\textbf{i}];
\textbf{go to NEXT end end
\textbf{LARGER:} \textbf{for } H := \textbf{F}, \textbf{H} + 1 \textbf{while } \textbf{K(S[H])} < \textbf{K(I[I])} \textbf{do
\textbf{for } G := \textbf{H}, \textbf{G} + 1 \textbf{while } \textbf{K(S[G])} \textbf{\neq 0 do
\textbf{for } j := \textbf{G step 1} until \textbf{H} + 1 do
\textbf{S}[j] := \textbf{S}[j] + 1;
\textbf{S}[H] := \textbf{I}[\textbf{i}]; \textbf{go to NEXT};
\textbf{SMALLER:} \textbf{for } H := \textbf{F}, \textbf{H} - 1 \textbf{while } \textbf{K(S[H])} > \textbf{K(I[I])} \textbf{do
%}}
for G := H, G - 1 while K[S(G)] ≠ 0 do
for j := G step 1 until H - 1 do
S[j] := S[j] + 1;
S[H] := |I[j];
NEXT: end end

Sort quadratic selection (I,N,S); value N; array I[1:N], S[1:N];
integer procedure K(i); integer N;
begin integer i,j,k,C,D,J,M;
integer array C[1:M], D[1:M];
array I[1:M], I[1:1];
Divide inputs: M := enter (sqrt(N)) + 1; j := k := 1;
for i := 1 step 1 until N do begin
I[i] := I[i];
k := k + 1;
if k > M then begin k := 1;
j := j + 1 end end
Fill up inputs: I[i,k] := z; k := k + 1;
if k > M then begin k := 1; j := j + 1 end
if j ≤ M then go to Fill up inputs;
Set controls: for j := 1 step 1 until M do begin
C[j] := K(I[j], I[1]); D[j] := 1;
for k := 1 step 1 until M do begin
if C[j] > K(I[j],k) then begin
C[j] := K(I[j],k);
D[j] := k end end;
i := 1;
Find least: C := C[1]; D := D[1]; J := 1;
for j := 1 step 1 until M do begin
if C[j] < C[j] then begin C := C[j];
D := D[j]; J := j end end;
Fill file: S[i] := I[J,D]; i := i + 1;
if i = N + 1 go to STOP;
Reset controls: for j := 1 step 1 until M do begin
C[j] := K(I[j], I[1]); D[j] := 1;
for k := 1 step 1 until M do begin
if C[j] > K(I[j],k) then begin
C[j] := K(I[j],k);
D[j] := k end end;
go to Find least;
STOP: end end

Sort binary merge (I,N,S); value N; array I[1:N];
integer procedure K(i); integer N;
begin real array $S[1:N]$;
integer array A[0:1, 0;J[a]], B[0:1, 0;K[b]], Aloc[0:1, 0;J[a]],
Bloc[0:1, 0;K[b]], J[0:1], K[0:1], J[0:1], K[0:1];
integer a,b,i,j,k;
for i := 1 step 1 until N do begin
if K[I[i]] < K[I[i-1]] then
if a = 1 then a := 0 else a := 1;
A[a, J[a]] := K[I[i]];
A[a, J[a]] := i;
J[a] := j[a] + 1 end;
J[0] := j[0]; J[1] := j[1];
k[1] := 1;
two inputs: if A[1, J[1]] ≤ A[0, J[0]] then a := 1 else
a := 0;
B[0, K[b]] := A[a, J[a]];
B[0, J[b]] := A[a, J[a]];
j[a] := j[a] + 1; k[b] := k[b] + 1;
if A[a, J[a]] ≤ A[a, J[a] - 1] then go to two
inputs else
if a = 1 then a := 0 else a := 1;
single step: B[0, K[b]] := A[a, J[a]];
B[0, J[b]] := A[a, J[a]];
j[a] := j[a] + 1;
k[b] := k[b] + 1;
if A[a, J[a]] ≥ A[a, j[a] - 1] then go to single step;
switch file: if b = 1 then b := 0 else b := 1;
check rollout: for a := 0, 1 do
if [a] = J[a] then go to rollout;
go to two inputs;
rollout: B[0, K[b]] := A[a, J[a]];
B[0, J[b]] := A[a, J[a]];
k[b] := k[b] + 1;
j[a] := j[a] + 1;
if [a] = J[a] then go to interchange files;
if A[a, J[a]] < A[a, j[a] - 1] then
if b = 1 then b := 0 else b := 1;
go to rollout;
interchange files: K[0] := k[0]; K[1] := k[1];
if K[0] = 1 then go to output end
for b := 1, 0 do begin
for k[b] := 1 step 1 until K[b] do begin
A[b, k[b]] := B[b, k[b]];
A[b, k[b]] := B[b, k[b]];
J[b] := K[b] end end;
go to next sort;
output: for i := 1 step 1 until N do
S[i] := I[Blc[b][i]];
end end

REMARK ON ALGORITHM 76
SORTING PROCEDURES (Ivan Flores, Comm. ACM
5, Jan. 1962)
B. Randle
Atomic Power Div., The English Electric Co., Whetstone,
England
The following types of errors have been found in the Sorting
Procedures:
1. Procedure declarations not starting with procedure.
2. Bound pair list given with array specification.
3. := used instead of := in assignment statements, and in a for clause.
4. A large number of semicolons missing (usually after end).
5. Expressions in bound pair lists in array declarations depending on local variables.
6. Right parentheses missing in some procedure statements.
7. Conditional statement following a then.
8. No declarations for \( A \), or \( z \), which is presumably a misprint.
9. In several procedures attempt is made to use the same identifier for two different quantities, and sometimes to declare an identifier twice in the same block head.
10. In the Presort quadratic selection procedure an array, declared as having two dimensions, is used by a subscripted variable with only one subscript.
11. At one point a subscripted variable is given as an actual parameter corresponding to a formal parameter specified as an array.
12. In several of the procedures, identifiers used as formal parameters are redeclared, and still assumed to be available as parameters.
13. In every procedure \( K \) is given in the specification part, with a parameter, whilst not given in the formal parameter list.

No attempt has been made to translate, or even to understand the logic of these procedures. Indeed it is felt that such a grossly inaccurate attempt at Algol should never have appeared as an algorithm in the Communications.