

Algorithms

G. E. FORSYTHE, J. G. HERRIOT, Editors

ALGORITHM 245

TREESORT 3 [M1]

ROBERT W. FLOYD (Recd. 22 June 1964 and 17 Aug. 1964)
Computer Associates, Inc., Wakefield, Mass.

```

procedure TREESORT 3 (M, n);
  value n; array M; integer n;
comment TREESORT 3 is a major revision of TREESORT
[R. W. Floyd, Alg. 113, Comm. ACM 5 (Aug. 1962), 434] sug-
gested by HEAPSORT [J. W. J. Williams, Alg. 232, Comm.
ACM 7 (June 1964), 347] from which it differs in being an in-place
sort. It is shorter and probably faster, requiring fewer compar-
isons and only one division. It sorts the array M[1:n], requiring
no more than  $2 \times (2 \uparrow p - 2) \times (p - 1)$ , or approximately  $2 \times$ 
 $n \times (\log_2(n) - 1)$  comparisons and half as many exchanges in
the worst case to sort  $n = 2 \uparrow p - 1$  items. The algorithm is
most easily followed if M is thought of as a tree, with M[j÷2]
the father of M[j] for  $1 < j \leq n$ ;
begin
  procedure exchange (x,y); real x,y;
    begin real t; t := x; x := y; y := t
    end exchange;
  procedure siftup (i,n); value i, n; integer i, n;
comment M[i] is moved upward in the subtree of M[1:n] of
which it is the root;
  begin real copy; integer j;
    copy := M[i];
    loop: j := 2 × i;
    if j ≤ n then
      begin if j < n then
        begin if M[j+1] > M[j] then j := j + 1 end;
        if M[j] > copy then
          begin M[i] := M[j]; i := j; go to loop end
        end;
        M[i] := copy
      end siftup;
      integer i;
      for i := n÷2 step -1 until 2 do siftup (i,n);
      for i := n step -1 until 2 do
        begin siftup (1,i);
          comment M[j÷2] ≥ M[j] for  $1 < j \leq i$ ;
          exchange (M[1], M[i]);
          comment M[1:n] is fully sorted;
        end
      end TREESORT 3

```

ALGORITHM 246

GRAYCODE [Z]

J. BOOTHROYD* (Recd. 18 Nov. 1963)

English Electric-Leo Computers, Kidsgrove, Stoke-on-
Trent, England

* Now at University of Tasmania, Hobart, Tasmania, Aust.

```

procedure graycode (a) dimension: (n) parity: (s); value n,s;
  Boolean array a; integer n; Boolean s;
comment elements of the Boolean array a[1:n] may together be

```

considered as representing a logical vector value in the Gray
cyclic binary-code. [See e.g. Phister, M., Jr., *Logical Design of*
Digital Computers, Wiley, New York, 1958. pp. 232, 399.] This
procedure changes one element of the array to form the next
code value in ascending sequence if the parity parameter *s*
= **true** or in descending sequence if *s* = **false**. The procedure
may also be applied to the classic "rings-o-seven" puzzle [see
K. E. Iverson, *A Programming Language*, p. 63, Ex. 1.5];

```

begin integer i,j; j := n + 1;
  for i := n step -1 until 1 do if a[i] then begin s := ¬ s;
    j := i end;
    if s then a[1] := ¬ a[1] else if j < n then a[j+1] := ¬ a[j+1]
    else a[n] := ¬ a[n]
  end graycode

```

ALGORITHM 247

RADICAL-INVERSE QUASI-RANDOM POINT SEQUENCE [G5]

J. H. HALTON AND G. B. SMITH (Recd. 24 Jan. 1964 and
21 July 1964)

Brookhaven National Laboratory, Upton, N. Y., and
University of Colorado, Boulder, Colo.

```

procedure QRPSH (K, N, P, Q, R, E);
integer K, N; real array P, Q; integer array R; real E;
comment This procedure computes a sequence of N quasi-
random points lying in the K-dimensional unit hypercube
given by  $0 < x_i < 1$ ,  $i = 1, 2, \dots, K$ . The ith component of
the mth point is stored in Q[m,i]. The sequence is initiated by a
"zero-th point" stored in P, and each component sequence is
iteratively generated with parameter R[i]. E is a positive error-
parameter. K, N, E, and the P[i] and R[i] for  $i = 1, 2, \dots, K$ ,
are to be given.

```

The sequence is discussed by J. H. Halton in *Num. Math.* 2
(1960), 84-90. If any integer *n* is written in radix-*R* notation as

$$n = n_m \dots n_2 n_1 n_0 . 0 = n_0 + n_1 R + n_2 R^2 + \dots + n_m R^m,$$

and reflected in the radical point, we obtain the *R*-inverse func-
tion of *n*, lying between 0 and 1,

$$\phi_R(n) = 0 . n_0 n_1 n_2 \dots n_m = n_0 R^{-1} + n_1 R^{-2} \\ + n_2 R^{-3} + \dots + n_m R^{-m-1}.$$

The problem solved by this algorithm is that of giving a com-
pact procedure for the addition of R^{-1} , in any radix *R*, to a frac-
tion, with downward "carry".

If $P[i] = \phi_{R[i]}(s)$, as will almost always be the case in practice,
with *s* a known integer, then $Q[m,i] = \phi_{R[i]}(s+m)$. For quasi-
randomness (uniform limiting density), the integers *R*[*i*] must
be mutually prime.

For exact numbers, *E* would be infinitesimal positive. In prac-
tice, round-off errors would then cause the "carry" to be in-
correctly placed, in two circumstances. Suppose that the stored
number representing $\phi_R(n)$ is actually $\phi_R(n) + \Delta$. (a) If $|\Delta|$
≥ R^{-m-1} , we see that the results of the algorithm become un-

predictable. It is necessary to stop before this event occurs. It may be delayed by working in multiple-length arithmetic. (b) If $n = R^{m+1} - 1$, so that $\phi_R(n) = 1 - R^{-m-1}$, and $\Delta < 0$, the computed successor of the stored value can be seen to be about R^{-m} , instead of $R^{-m-2} = \phi_R(n+1)$. This error can be avoided, without disturbing the rest of the computation, by adopting a value of E greater than any $|\Delta|$ which may occur, but smaller than the least $(nR)^{-1}$ (which is smaller than the least R^{-m-1}) to be encountered.

Small errors in the $P[i]$ will not affect the sequence. Any set of $P[i]$ in the computer may be considered as a set of $\phi_{R[i]}(s_i)$, for generally large and unequal integers s_i , with small round-off errors. The arguments used in J. H. Halton's paper to establish the uniformity of the sequence of points

$$[\phi_{R_1}(n), \phi_{R_2}(n), \dots, \phi_{R_K}(n)], \quad n = 1, 2, \dots, N$$

can be applied identically to the more general sequence

$$[\phi_{R_1}(s_1+n), \phi_{R_2}(s_2+n), \dots, \phi_{R_K}(s_K+n)], \quad n = 1, 2, \dots, N.$$

Thus, theoretically, any "zero-th point" P will do. However, the difficulty described in (a) above limits us to the use of $P[i]$ corresponding to relatively small integers s_i ;

```
begin integer i, m; real r, f, g, h;
for i := 1 step 1 until K do
begin r := 1.0/R[i];
for m := 1 step 1 until N do
begin if m > 1 then f := 1.0 - Q[m-1,i] else
f := 1.0 - P[i];
g := 1.0; h := r;
repeat: if f - h < E then
begin g := h; h := h × r; go to repeat end;
Q[m,i] := g + h - f
end
end
end QRPSh
```

CERTIFICATION OF ALGORITHM 181 [S15] COMPLEMENTARY ERROR FUNCTION—LARGE X [Henry C. Thacher, Jr., *Comm. ACM* 6 (June 1963), 315]

I. CLAUSEN AND L. HANSSON (Recd. 20 Aug. 1964)
DAEC, Risø, Denmark.

The procedure *erfcL* was tested in GIER-ALGOL with 29 significant bits and the number-range $abs(x) < 2 \uparrow 512$ (approx. 1.310154). The statement $m := R := 0$; was corrected to $m := 0$; $R := 0$; [Because m and R are of different type; cf. Sec. 4.2.4 of the ALGOL Report, *Comm. ACM* 6 (Jan. 1963), 1-17.—Ed.] After this the tests were successful. The procedure was checked a.o. for $x = 1.19 (-0.01) 0.72$. The differences from table values increased from 10^{-8} at $x = 1.1$ to $7 \cdot 10^{-8}$ at $x = 0.75$. Overflow occurred at $x = 0.71$.

CERTIFICATION OF ALGORITHM 224 [F3] EVALUATION OF DETERMINANT

[Leo J. Rotenberg, *Comm. ACM* 7 (Apr. 1964), 243]
VIC HASSELBLAD AND JEFF RULIFSON (Recd. 17 July 1964)
Computer Center, U. of Washington, Seattle, Wash.

The "Evaluation of Determinant" program was tested on an ALGOL 60 compiler for an IBM 709 (SHARE distribution #3032). When the 10th line on page 244 was changed to read:

```
begin if imax = r then go to resume else
correct results were obtained. It was tested up through  $4 \times 4$  matrices.
```

CERTIFICATION OF ALGORITHM 237 [A1]
GREATEST COMMON DIVISOR [J. E. L. Peck,
Comm. ACM 7 (Aug. 1964), 481]
T. A. BRAY (Recd. 8 Sept. 1964)
Boeing Scientific Research Laboratories, Seattle,
Washington

This procedure was translated into the FORTRAN IV language and tested on the Univac 1107. No corrections were required and the procedure gave correct results for all cases tested.

Revised Algorithms Policy • May, 1964

A contribution to the Algorithms department must be in the form of an algorithm, a certification, or a remark. Contributions should be sent in duplicate to the editor, typewritten double-spaced in capital and lower-case letters. Authors should carefully follow the style of this department, with especial attention to indentation and completeness of references. Material to appear in **boldface** type should be underlined in black. Blue underlining may be used to indicate *italic* type, but this is usually best left to the Editor.

An algorithm must be written in the ALGOL 60 Reference Language [*Comm. ACM* 6 (Jan. 1963), 1-17], and normally consists of a commented procedure declaration. Each algorithm must be accompanied by a complete driver program in ALGOL 60 which generates test data, calls the procedure, and outputs test answers. Moreover, selected previously obtained test answers should be given in comments in either the driver program or the algorithm. The driver program may be published with the algorithm if it would be of major assistance to a user.

Input and output should be achieved by procedure statements, using one of the following five procedures (whose body is not specified in ALGOL): [see "Report on Input-Output Procedures for ALGOL 60," *Comm. ACM* 7 (Oct. 1964), 628-629].

```
procedure inreal (channel, destination); value channel; integer channel;
real destination; comment the number read from channel channel is
assigned to the variable destination; . . . ;
procedure outreal (channel, source); value channel, source; integer channel;
real source; comment the value of expression source is output to channel
channel; . . . ;
procedure ininteger (channel, destination);
value channel; integer channel, destination; . . . ;
procedure outinteger (channel, source);
value channel, source; integer channel, source; . . . ;
procedure outstring (channel, string); value channel; integer channel;
string string; . . . ;
```

If only one channel is used by the program, it should be designated by 1
Examples:

```
outstring (1, 'x = '); outreal (1, x);
for i := 1 step 1 until n do outreal (1, A[i]);
ininteger (1, digit [17]);
```

It is intended that each published algorithm be a well-organized, clearly commented, syntactically correct, and a substantial contribution to the ALGOL literature. All contributions will be refereed both by human beings and by an ALGOL compiler. Authors should give great attention to the correctness of their programs, since referees cannot be expected to debug them. Because ALGOL compilers are often incomplete, authors are encouraged to indicate in comments whether their algorithms are written in a recognized subset of ALGOL 60 [see "Report on SUBSET ALGOL 60 (IFIP)," *Comm. ACM* 7 (Oct. 1964), 626-627].

Certifications and remarks should add new information to that already published. Readers are especially encouraged to test and certify previously uncertified algorithms. Rewritten versions of previously published algorithms will be refereed as new contributions, and should not be imbedded in certifications or remarks.

Galley proofs will be sent to the authors; obviously rapid and careful proofreading is of paramount importance.

Although each algorithm has been tested by its author, no liability is assumed by the contributor, the editor, or the Association for Computing Machinery in connection therewith.

The reproduction of algorithms appearing in this department is explicitly permitted without any charge. When reproduction is for publication purposes, reference must be made to the algorithm author and to the *Communications* issue bearing the algorithm.—G.E.F.

INDEX BY SUBJECT TO ALGORITHMS, 1964

<u>REAL ARITHMETIC, NUMBER THEORY</u>				<u>RANDOM NUMBER GENERATORS</u>			
A1	223	PRIME TWINS	4-64(243)	G5	247	QUASI-RANDOM POINT SEQUENCE	12-64(701)
A1	237	GREATEST COMMON DIVISOR	8-64(481),12-64(702)				
<u>TRIG AND INVERSE TRIG FUNCTIONS</u>				<u>PERMUTATIONS AND COMBINATIONS</u>			
B1	229	ELEMENTARY FCNS BY CONT. FRACT.	5-64(296)	G6	235	RANDOM PERMUTATION	7-64(420)
B1	241	ARCTAN(Z)	9-64(546)	G6	242	PERMUTATIONS WITH REPETITIONS	10-64(585)
<u>EXPONENTIAL AND LOGARITHMIC FUNCTIONS</u>				<u>OPERATIONS RESEARCH, GRAPH STRUCTURES</u>			
B3	48	LOG(Z), Z COMPLEX	4-61(179),6-62(347),	H	40	CRITICAL PATH SCHEDULING	3-61(152),9-61(392),
B3	48	7-62(391),8-64(485)		H	40	17-62(513),6-64(349)	
B3	243	LOGARITHM OF COMPLEX NUMBER	11-64(660)				
<u>SUMMATION OF SERIES, CONVERGENCE ACCELERATION</u>				<u>INPUT - COMPOSITE</u>			
C6	128	FOURIER SERIES SUMMATION	10-62(513),7-64(421)	I5	239	FREE-FIELD READ	6-64(481)
C6	215	EPSILON ALGORITHM	11-63(662),5-64(297)				
C6		EPSILON ALGORITHM	NUM.MATH.V6(22)				
<u>QUADRATURE</u>				<u>PLOTTING</u>			
D1	63	ROMBERG METHOD	6-61(255),3-62(166),	J6	162	XY PLOTTER	4-63(161),8-63(450),
D1	60	5-62(281),7-64(420)		J6	162	8-64(482)	
D1	145	MULTIPLE INTEGRAL	12-62(604),5-64(296)				
D1	182	ADAPTIVE SIMPSON	6-63(315),4-64(244)				
D1	233	MULTIPLE INTEG.-SIMPSONS RULE	6-64(348)				
D1		ROMBERG METHOD	NUM.MATH.V6(15)				
<u>ORDINARY DIFFERENTIAL EQUATIONS</u>				<u>SORTING</u>			
D2	218	KUTTA MERSON	12-63(737),10-64(585)	M1	175	SHUTTLE SORT	6-63(312),10-63(619),
				M1	175	12-63(739),5-64(296)	
				M1	201	SHELL SORT	8-63(445),6-64(349)
				M1	207	STRING SORT	10-63(615),10-64(585)
				M1	232	HEAPSORT	6-64(347)
				M1	245	TREESORT 3	12-64(701)
				M1		SEARCH IN A LIST	J.ACM-1962(23)
				M1		INSERTION IN A LIST	J.ACM-1962(23)
				M1		DELETION FROM A LIST	J.ACM-1962(24)
				M1		SORTING WITH MINIMUM STORAGE	J.ACM-1962(27)
<u>CURVE AND SURFACE FITTING</u>				<u>APPROXIMATION OF SPECIAL FUNCTIONS...</u>			
E2	91	CHEBYSHEV FIT	5-62(281),4-63(167),	S			
E2	91	5-64(296)		S			
E2		RATIONAL CHEBYSHEV APPROX.	J.ACM-1964(66)	S			
<u>MINIMIZING OR MAXIMIZING A FUNCTION</u>				<u>FUNCTIONS ARE CLASSIFIED S01 TO S23, FOLLOWING</u>			
E4	203	MINIMIZE FUNCT. OF N VARIABLES	9-63(517),10-64(585)	S			
E4		MINIMIZING FCN.-CONJ. GRAD.	COMP.J.V7(151)	S			
<u>MATRIX OPERATIONS, INCLUDING INVERSION</u>				S14	221	GAMMA FUNCTION	3-64(143),10-64(586)
F1	150	INVERSE OF SYMMETRIC MATRIX	2-63(67),7-63(390),	S14	222	INCOMPLETE BETA FCN. RATIOS	3-64(143),4-64(244)
F1	150	3-64(148)		S14	225	GAMMA FCN WITH CONTROLLED ACCY.	5-64(295),10-64(586)
F1	197	MATRIX DIVISION	8-63(443),3-64(148)	S15	123	ERROR FUNCTION	9-62(483),6-63(316),
F1	230	MATRIX PERMUTATION	6-64(347)	S15	123	10-63(618),3-64(145)	
F1	231	INVERSION-GAUSS. ELIM.-COMP. PIV.	6-64(347)	S15	181	ERROR FUNCTION	6-63(315),12-64(702)
F1		INVERSE-CONFL. VANDERMONDE MTX	NUM.MATH.V5(429)	S15	209	ERROR FUNCTION	10-63(616),3-64(148),
F1		EQUIVALENCE OF MATRICES	ICC BULL.-1964(62)	S15	209	8-64(482)	
<u>EIGENVALUES AND EIGENVECTORS OF MATRICES</u>				S15	226	NORMAL DISTRIBUTION FCN.	5-64(295)
F2	122	GIVENS TRIDIAGONAL REDUCTION	9-62(482),3-64(144)	S17	236	BESSEL FCNS OF FIRST KIND	8-64(479)
F2		EIGENVALUES-LAGUERRES METHOD	MOC 1964(474)	S18	214	BESSEL FUNCTION	11-63(662),6-64(349)
<u>DETERMINANTS</u>				S18	228	Q-BESSEL FUNCTION	5-64(295)
F3	41	DETERMINANT EVALUATION	4-61(176),9-63(520),	S20	213	FRESNEL INTEGRALS	10-63(617),11-64(661)
F3	41	3-64(144)		S20	244	FRESNEL INTEGRALS	11-64(660)
F3	170	DETERMINANT-POLYNOMIAL ELEMENTS	4-63(165),8-63(450),	S21		COMPLETE ELL. INT.-FIRST KIND(K)	NUM.MATH.V5(296)
F3	170	7-64(421)		S21		COMPLETE ELL. INT.-SECOND KIND(E)	NUM.MATH.V5(297)
F3	224	EVALUATION OF DETERMINANT	4-64(243),12-64(702)	S21		COMPLETE ELL. INT.(B)	NUM.MATH.V5(297)
<u>SIMULTANEOUS LINEAR EQUATIONS</u>				S21		INCOMPL. ELL. INT.-FIRST KIND(K)	NUM.MATH.V5(297)
F4	135	CROUT WITH EQUILIBRATION	11-62(553),11-62(557),	S21		INCOMPL. ELL. INT.-SECOND KIND(E)	NUM.MATH.V5(298)
F4	135	7-64(421)		S21		INCOMPL. ELL. INT.(B)	NUM.MATH.V5(299)
F4	220	GAUSS-SEIDEL	12-63(739),6-64(349)	S21		JACOBIAN ELLIPTIC SIN FCN.(SN)	NUM.MATH.V5(299)
F4	238	CONJUGATE GRADIENT METHOD	8-64(481)	S21		JACOBIAN ELLIPTIC COS FCN.(CN)	NUM.MATH.V5(300)
F4		CONJUGATE GRADIENT METHOD	NUM.MATH.V5(195)	S21		JACOBIAN ELLIPTIC FCN.(DN)	NUM.MATH.V5(301)
				S22	191	HYPERGEOMETRIC FCN.(COMPLEX)	7-63(388),4-64(244)
				S22	192	CONFLUENT HYPERG. FCN.(COMPLEX)	7-63(388),4-64(244)
				S22	227	CHEBYSHEV POLYNOMIAL COEFF.	5-64(295)
				S23	234	POISSON-CHARLIER POLYNOMIALS	7-64(420)
<u>ALL OTHERS</u>				<u>ALL OTHERS</u>			
				Z	199	CALENDAR CONVERSION	8-63(444),11-64(661)
				Z	240	COORDINATES ON AN ELLIPSOID	9-64(546)
				Z	246	GRAYCODE	12-64(701)

Key—1st column: A1, B1, B3, etc. is the key to the underlined Modified Share Classification heading each group of algorithms; 2d column: number of the algorithm in CACM; 3d column: title of algorithm; 4th column: month, year and page (in parens) in CACM, or reference elsewhere.