ALGORITHM 116

COMPLEX DIVISION

ROBERT L. SMITH

Stanford University, Stanford, Calif.

procedure complexdiv (a, b, c, d); real a, b, c, d;
comment complexdiv yields the complex quotient of a + ib
 divided by c + id. The method used here tends to avoid arithmetic
overflow or underflow. Such spills could otherwise occur when squaring
the component parts of the denominator if the usual method were used;

begin real r, den;
if abs (c) >= abs (d) then
begin r := d/c;
den := c + r X d;
q := (a + b X r)/den;
f := (b - a X r)/den;
end
else
begin r := c/d;
den := d + r X c;
e := (a X r + b)/den;
f := (b X r - a)/den;
end
end complexdiv

ALGORITHM 117

MAGIC SQUARE (EVEN ORDER)

D. M. COLLISON

Elliott Brothers (London) Limited, Borehamwood, Herts., England

procedure magiceven (n, z); value n; integer array z; integer n;
comment the method of Devedec for even n is described in
"Mathematical Recreations" by M. Kraitchik, pp. 150-2. Enter with side of square n to produce a magic square of the integers
1 - n ^ 2 in z, where n ^ 2 >= 4;

begin integer a, b, n2, nm; Boolean p, q, r;
n2 := n + 2; nm := n X n;

begin procedure alpha (p, q, a, h); value p, q, a, h; integer p, q, a;
Boolean h;
Comment pattern 0/0/0/ ... ;
begin integer r;
for r := p step 1 until q do begin
z[r, a] := if h then (a X n + n + r) else (nm - a X n + 1 + n - r); h := ~h end;
end alpha;

procedure beta (p, q, a, h); value p, q, a, h; integer p, q, a;
Boolean h;
comment pattern 1 - 1 - 1 .... ;
begin integer r;
for r := p step 1 until q do begin
z[r, a] := if h then [nm - a X n + r + 1] else (a X n + 1 - r); h := ~h end;
end beta;

procedure gamma (p, q, a, h); value p, q, a, h; integer p, q, a;
Boolean h;
comment pattern --/--/-- .... ;
begin integer r;
for r := p step 1 until q do begin
z[r, a] := if h then [(nm - a X n + n - r + 1)] else (a X n + 1 - r); h := ~h end;
end gamma;

comment program begins;
p := q := (n - (n + 4) X 4 - 4 = 0); r := true;
for a := 1 step 1 until (n - 1) do begin
beta (1, a - 1, a, r); alpha (a, n2 - 1, a, true);
z[n2, a] := if q then (nm + n + n2 + 1) else (a X n + 1 - r); h := ~h end;
end gamma;

END

Communications of the ACM 435
ALGORITHM 118
MAGIC SQUARE (ODD ORDER)
D. M. COLLISON
Elliott Brothers (London) Limited, Borehamwood, Herts., England

procedure magicodd (n, x); value n; integer n; integer
array x;
comment for given side n the procedure generates a magic
square of the integers 1 - n T 2. For the method of De la
Loubère, see M. Kraitchik, “Mathematical Recreations,” p.
140. n must be odd and n => 3;
begin integer i, j, k;
for i := 1 step 1 until n do
for j := 1 step 1 until n do
x[i, j] := 0;
end;
for k := 1 step 1 until n X n do begin
if x[i, j] # 0 then begin i := i -- 1; j := j -- 2;
if i < 1 then i := i + n; if j < 1 then j := j + n end;
x[i, j] := k;
i := i + 1; if i > n then i := i -- n;
j := j + 1; if j > n then j := j -- n;
end;
end magicodd

ALGORITHM 119
EVALUATION OF A PERT NETWORK
BURTON EISENMAN AND MARTIN SHAPIRO
United Nuclear Corp., White Plains, N. Y.

procedure pert (nmax, i, j, te, st, emax, l, es, at);
comment An algorithm describing an iterative procedure for
evaluating a PERT network that permits the use of arbitrarily
ordered activities and event identifiers such that an upper
triangular matrix type of solution is unnecessary.
It has been observed by investigations of PERT networks,
that an N X N matrix whose rows are designated as predecessor
and whose columns are designated as successor events, has an
entry in the (i, j)-element representing the activity time re-
quired in going from event i to event j. By elementary transfor-
mations, the matrix is transformed generally into an upper
triangular matrix. The resultant upper triangular matrix is well
ordered (i.e. any activity time appearing in a column is not
dependent upon those activity times which appear in columns
to the right of it). This precise manipulation generally demands considerable running time. By direct evaluation not requiring a collection of
elementary transformations, it is possible to evaluate the network with considerable reduction of running time;
begin integer nmax, emax;
real at, st;
integer array i, j, l;
real array te, es, at;
comment Given the total number of activities, nmax, the pre-
ceding and succeeding event identifiers, i, and j, the cor-
respective expected time, te, for each activity, and the starting
time, st, of the network, this procedure computes the early start
and late finishes, es, and at, for each event, l, in the net-
work;
begin procedure scan (e, t, l);
integer e, t;
integer array l;
comment Given the number of events, e-1, contained in vector array, l, and an event identifier le or j, stored in l,
this procedure scans the existing array, l, to determine whether the
event should be added to the list or not. If it is to be added,
it becomes le, and e replaces the event identifier. If it is not
added, k replaces the event identifier. ; begin integer k;
if e = 1 then go to add;
for k := e -- 1 step --1 until 1 do
begin
add:
begin
l[e] := t;
t := e;
e := e + 1;
end scan;
integer n, e, s, t, k;
real a, x;
begin
s := l;
a := st;
k := emax;
end;
for e := 1 step 1 until emax do
begin
a[e] := a;
s1:
begin
for e := 1 step 1 until emax do
a[e] := a;
s2:
begin
for n := 1 step 1 until nmax do
begin
if l[i[n]] > 0 then
for s := b1, b2;
b1: x := abs (a[i[n]]) + es[n];
if x > abs (a[i[n]]) then go to l1;
go to l2;
b2: x := abs (a[i[n]]) -- es[n];
if x < abs (a[i[n]]) then
l1: a[i[n]] := -- x;
l2: a[i[n]] := x;
for e := 1 step 1 until emax do
begin
if l[e] < 0 then
begin
end;
begin
if a[e] < 0 then
begin
l[e] := abs (l[e]);
end;
begin
end;
for e <= 0 then
begin
end;
begin
end;
begin
end;
begin
436 Communications of the ACM