```
algol,n<
begin
   comment

   Sudoku program

   Time1: 5269.1s = 1h 27m 49.1s
   Time4: 6551.3s = 1h 49m 11.3s
   Time4: 4188.7s = 1h 9m 48.7s
   Time5: 5245.2s = 1h 27m 25.2s
   Time6: 17130.0s = 4h 45m 30.0s
   Time6: 123...: 70029.5s = 19h 27m 9.5s
   Time6: 987...: 9803.0s = 2h 43m 23.0s
   Time6: 12651.3s = 3h 30m 51.3s
   Time6: 12142.5s = 3h 22m 22.5s
   Time7: 3388.7s = 56m 28.7s

   No buffer:

   Time classic:        24794.2
   Time turbo:          24784.9 0.04pct
   Tracks transferred:  972876

   Buffer:

   Time classic:        3407.5
   Time turbo:          3097.7 9.1pct
   Tracks transferred:  6255




   ;

   integer array board,rows,cols,submatrices[1:81],stack[0:161];
   integer n,i,j,k,l,s,p,digit,row1,col1,mat1,best n,best p;
   boolean m,best m,mask;
   boolean array possible[1:81];
   real procedure clock count;
   code clock count;
   1, 37;
     zl        , grf p-1   ; RF:=clock count; stack[p-1]:=RF;
   e;
   procedure print;
   begin
      integer i,j;
      writecr;
      writetext(«<Clock: »);
      write(«-ddddddddd.d», clock count);
      writecr;
      for i:=1 step 1 until 9 do
      begin
         for j:=1 step 1 until 9 do
         writeinteger(«dd», board[(i-1)×9+j]);
         writecr
      end
   end print;
   procedure nprint(n);
   value n;
   boolean n;
   begin
      integer i;
      writecr;
      for i:=0 step 1 until 39 do
```

```
      begin
         writechar(if n shift i then 1 else 16);
         if i mod 10=9 then writechar(0)
      end i
   end nprint;
   integer procedure nbits(n);
   value n;
   boolean n;
   begin
      n:=n shift -30;
      n:=boolean((integer(n ∧ 24045454545)) + (integer((n shift -1) ∧ 24045454545)));
      n:=boolean((integer(n ∧ 24043434343)) + (integer((n shift -2) ∧ 24043434343)));
      n:=boolean((integer(n ∧ 24040404040)) + (integer((n shift -4) ∧ 24040404040)));
      nbits:=(integer(n ∧ 24040404040)) + (integer((n shift -8) ∧ 24040404040));
   end nbits;

   select(16);

   n:=0;
   readgeneral(board,3 0 7 27 3 2 7 64 3 1 7 5 3 3 7 0,n);

   for i:=1 step 1 until 9 do
   begin
      for j:=1 step 1 until 9 do
      begin
         rows[(i-1)×9+j] := (i-1)×9+1;
         cols[(i-1)×9+j] := j
      end j
   end i;
   for i:=1 step 1 until 3 do
   for j:=1 step 1 until 3 do
   for k:=1 step 1 until 3 do
   for l:=1 step 1 until 3 do
   submatrices[(i-1)×27+(j-1)×3+(k-1)×9+l] := (i-1)×27+(j-1)×3+1;

   clock count;

   print;
   s:=0;
   p:=1;
a1:
a2:
   best p:=0;
   best n:=10;
   best m:=400;
   for p:=1 step 1 until 81 do
   if board[p]=0 then
   begin
      m:=1 0 9 m 30 0;
      row1:=rows[p];
      col1:=cols[p];
      mat1 := submatrices[p];
      mask:=1 0 39 m;
      i:= board[row1   ]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 1]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 2]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 3]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 4]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 5]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 6]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 7]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[row1+ 8]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[col1   ]; if i≠0 then m:=m∧(mask shift -i);
      i:= board[col1+ 9]; if i≠0 then m:=m∧(mask shift -i);
```

```
   i:= board[col1+18]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[col1+27]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[col1+36]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[col1+45]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[col1+54]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[col1+63]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[col1+72]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1   ]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+ 1]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+ 2]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+ 9]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+10]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+11]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+18]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+19]; if i≠0 then m:=m∧(mask shift -i);
   i:= board[mat1+20]; if i≠0 then m:=m∧(mask shift -i);
   n:=nbits(m);
   possible[p]:=m;
   if n<best n then
   begin
      best n:=n;
      best p:=p;
      best m:=m
   end better
end p free
else
possible[p]:=40 0;
if best n=10 then goto FOUND;
if best n=0 then
begin
   s:=s-2;
   if s<0 then goto BAD;
   board[stack[s]]:=0;
   goto a3
end dead end;
if best n>1 then
begin
   for p:=1 step 1 until 81 do
   if board[p]=0 then
   begin
      m:=possible[p];
      for j:=1 step 1 until 9 do
      if m shift j then
      begin
         row1:=rows[p];
         col1:=cols[p];
         mat1 := submatrices[p];
         k:=0;
         if possible[row1   ] shift j then k:=k+1;
         if possible[row1+ 1] shift j then k:=k+1;
         if possible[row1+ 2] shift j then k:=k+1;
         if possible[row1+ 3] shift j then k:=k+1;
         if possible[row1+ 4] shift j then k:=k+1;
         if possible[row1+ 5] shift j then k:=k+1;
         if possible[row1+ 6] shift j then k:=k+1;
         if possible[row1+ 7] shift j then k:=k+1;
         if possible[row1+ 8] shift j then k:=k+1;
         if k=1 then
         begin
            best p:=p;
            best m:=1 1 39 0 shift -j;
            goto better
         end only in row;
         k:=0;
```

```
            if possible[col1   ] shift j then k:=k+1;
            if possible[col1+ 9] shift j then k:=k+1;
            if possible[col1+18] shift j then k:=k+1;
            if possible[col1+27] shift j then k:=k+1;
            if possible[col1+36] shift j then k:=k+1;
            if possible[col1+45] shift j then k:=k+1;
            if possible[col1+54] shift j then k:=k+1;
            if possible[col1+63] shift j then k:=k+1;
            if possible[col1+72] shift j then k:=k+1;
            if k=1 then
            begin
               best p:=p;
               best m:=1 1 39 0 shift -j;
               goto better
            end only in col;
            k:=0;
            if possible[mat1   ] shift j then k:=k+1;
            if possible[mat1+ 1] shift j then k:=k+1;
            if possible[mat1+ 2] shift j then k:=k+1;
            if possible[mat1+ 9] shift j then k:=k+1;
            if possible[mat1+10] shift j then k:=k+1;
            if possible[mat1+11] shift j then k:=k+1;
            if possible[mat1+18] shift j then k:=k+1;
            if possible[mat1+19] shift j then k:=k+1;
            if possible[mat1+20] shift j then k:=k+1;
            if k=1 then
            begin
               best p:=p;
               best m:=1 1 39 0 shift -j;
               goto better
            end only in submatrix
         end j;
      end p;
better:
   end;
   stack[s]:=best p;
   stack[s+1]:=integer best m;
a3:
   p:=stack[s];
   m:=boolean stack[s+1];
   if (integer m)=0 then
   begin
      s:=s-2;
      if s<0 then goto BAD;
      board[stack[s]]:=0;
      goto a3
   end;
a4:
   for digit:=1 step 1 until 9 do
   if m shift digit then goto found digit;
found digit:
   m:=m∧(1 0 39 m shift -digit);
   board[p]:=digit;
   stack[s+1]:=integer m;
   s:=s+2;
   goto a1;

FOUND:
   writecr;
   writetext(‹<Finished clock: ›);
   write(‹-ddddddddd.d›, clock count);
   writecr;
   writetext(‹<Tracks transferred: ›);
   writeinteger(‹p›,tracks transferred);
```

```
   print;
   goto skip;
BAD:
   writecr;
   writetext(‡<Stack underflow‡);
skip:

end
t<
8,0,0,0,0,0,0,0,0,
0,0,3,6,0,0,0,0,0,
0,7,0,0,9,0,2,0,0,
0,5,0,0,0,7,0,0,0,
0,0,0,0,4,5,7,0,0,
0,0,0,1,0,0,0,3,0,
0,0,1,0,0,0,0,6,8,
0,0,8,5,0,0,0,1,0,
0,9,0,0,0,0,4,0,0;

0,0,0,0,0,0,0,0,0,
1,3,0,7,0,0,0,5,0,
0,4,0,0,0,0,9,0,7,
0,0,0,0,1,0,0,0,0,
0,0,0,0,0,0,0,4,2,
0,0,9,0,8,0,0,0,6,
0,0,8,0,0,0,0,0,0,
0,0,0,0,0,5,1,3,0,
6,0,0,2,0,0,0,0,0;

0,0,0,0,0,0,0,0,0,
9,7,0,3,0,0,0,5,0,
0,6,0,0,0,0,1,0,3,
0,0,0,0,9,0,0,0,0,
0,0,0,0,0,0,0,6,8,
0,0,1,0,2,0,0,0,4,
0,0,2,0,0,0,0,0,0,
0,0,0,0,0,5,9,7,0,
4,0,0,8,0,0,0,0,0;
```