Algorithms

L.D. Fosdick
Editor

Algorithm 411

Three Procedures for the Stable Marriage Problem [H]

D.G. McVitie* and L.B. Wilson (Recd. 12 Aug. 1968 and 15 July 1969)
Computing Laboratory, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, England

Key Words and Phrases: assignment problems, assignment procedures, combinatorics, discrete mathematics, operations research, stable marriage problem, university entrance
CR Categories: 5.30

Part 1

```
procedure GS (malechoice, femalechoice, marriage, count, n);
  value n;  integer count, n;
  integer array malechoice, femalechoice, marriage;
comment This procedure finds the male optimal stable marriage
  solution using the Gale and Shapley algorithm. The result is
  left in the integer array marriage. Thus marriage [i] is the man
  whom the ith woman marries. n is the size of the problem,
  count is the number of proposals made before the stable marriage
  is found. malechoice and femalechoice are the choice matrices
  for the men and women respectively, i.e. femalechoice[i, j]
  is the jth choice of the ith woman. The femalechoice array is
  changed to the integer array fc, where fc[i, j] is the choice num-
  ber (first, second, third, . . .) of the jth man to woman i. This
  new arrangement is adopted for convenience when the women
  compare proposals. All the women keep a dummy man 0 in sus-
  pense initially. This dummy man is given a choice number n + 1
  so that he will be given up as soon as any other offer is made;
begin
  integer i, m, j;  Boolean array refuse [0:n];
  integer array fc [1:n, 0:n], proposal, malecounter [1:n];
  for i := 1 step 1 until n do
  begin
    for j := 1 step 1 until n do
      fc [i, femalechoice [i, j]] := j;
    comment The femalechoice array is rearranged for conven-
      ience in the marriage part of the procedure;
      refuse [i] := true; marriage [i] := 0;
      malecounter [i] := 1;  fc [i, 0] := n + 1
  end;
  count := 0;
PROPOSE:
  m := 0;
  comment Now the rejected men propose to the next woman in
    their choice lists. Initially all the men propose to their first
    choices;
  for i := 1 step 1 until n do
    if refuse [i] then
    begin
      proposal [i] := malechoice [i, malecounter [i]];
      malecounter [i] := malecounter [i] + 1;
      m := m + 1; refuse [i] := false
    end
    else proposal [i] := -1;
  if m = 0 then go to FINISH;
  comment The procedure terminates if at any stage no proposals
    are made by the men;
  count := count + m;
  comment In the next part of the procedure all the women who
    have had a proposal decide whether to reject it or the one
    they are keeping in suspense;
  for i := 1 step 1 until n do
    if proposal [i] > 0 then
    begin
      j := proposal [i];
      if fc [j, i] > fc [j, marriage [j]] then refuse [i] := true
      else
      begin refuse [marriage [j]] := true; marriage [j] := i end
    end;
  go to PROPOSE;
FINISH:
end of procedure GS
```

Part 2

```
procedure MW(malechoice, femalechoice, marriage, count, n);
  value n;  integer count, n;
  integer array malechoice, femalechoice, marriage;
comment The heading is the same as for the GS procedure and
  the formal parameters have the same meaning. Also the female-
  choice array has been rearranged in the array fc as before, and
  the women given initially a dummy man 0 with choice number
  n + 1;
begin
  integer i, j;
  integer array fc [1:n, 0:n], malecounter [1:n];
  procedure PROPOSAL (i);  value i;  integer i;
  comment This procedure makes the next proposal for man i
    and calls the procedure REFUSAL to see what effect this
    proposal will have. The procedure does nothing if man i is
    the dummy man 0;
  if i ≠ 0 then
  begin
    integer j;  count := count + 1;
    j := malecounter[i];  malecounter[i] := j + 1;
    REFUSAL(i, malechoice[i, j])
  end;
  procedure REFUSAL(i, j);  value i, j;  integer i, j;
  comment This procedure decides whether woman j should keep
    the man she is holding in suspense in marriage[j] or man i
    who has just proposed to her. Whichever she rejects goes
    back to the procedure PROPOSAL to make his next proposal;
  if fc[j, marriage[j]] > fc[j, i] then
  begin
    integer k;
    k := marriage[j];  marriage[j] := i;
    PROPOSAL(k)
  end
  else PROPOSAL(i);
  for i := 1 step 1 until n do
  begin
    for j := 1 step 1 until n do
      fc[i, femalechoice[i,j]] := j;
    marriage[i] := 0;  malecounter[i] := 1; fc[i, 0] := n + 1
  end;
  count := 0;
  for i := 1 step 1 until n do PROPOSAL(i);
  comment This for statement operates the algorithm and after
    the ith cycle a set of stable marriages exists for the men 1 to i
    and i of the women;
end of procedure MW
```

*Now at Software Science, Ltd., Wilmslow, Cheshire, England.

```
Part 3
procedure ALL STABLE MARRIAGES (malechoice, femalechoice,
  n, STABLE MARRIAGE);
  value n;
  integer array malechoice, femalechoice;
  integer n; procedure STABLE MARRIAGE;
comment malechoice and femalechoice are the same arrays as were
  used in GS and MW, n is size of problem. STABLE MARRIAGE
  (marriage, n, count) is a procedure (with three parameters) writ-
  ten by the user which is entered when a new stable marriage is
  formed after count proposals. The marriage is stored such that
  marriage[i] contains the number of the man married to woman i.
  The locally declared Boolean array unchanged is used to make
  sure Rule (2) is not violated; i.e. during a breakmarriage opera-
  tion started on man i only men ≥ i may propose. The locally
  declared Boolean success is set true if breakmarriage to man i
  leads to a new stable marriage, otherwise it is set false;
begin
  integer array marriage, malecounter [0: n], fc [1: n, 0: n];
  Boolean array unchanged [0: n];
  integer i, j, k; Boolean success;
  procedure breakmarriage(malecounter,marriage,i,n,count);
    value malecounter, marriage, i, n, count;
    integer i, n, count; integer array malecounter, marriage;
  comment This procedure breaks the marriage of man i;
  begin
    integer j;
    marriage [malechoice [i, malecounter [i]-1]] := -i;
    proposal (i,malecounter,marriage,count);
    if ¬ success then go to EXIT;
    STABLE MARRIAGE (marriage,n,count);
    for j := i step 1 until n - 1 do
      breakmarriage (malecounter,marriage,j,n,count);
    comment The lower limit i in the above for statement is the
      application of Rule(1) which after a successful break-
      marriage operation on man i restricts further breakmarriages
      to men ≥ i;
    for j := i + 1 step 1 until n - 1 do
      unchanged [j] := true;
EXIT:
    unchanged [i] := false;
  end of breakmarriage;
  procedure proposal (i, malec, marriage, c);
    value i;
    integer i,c; integer array malec, marriage;
  comment In this procedure man i proposes to the next woman
    in his choice list, and calls the procedure refusal for this
    woman. If i is negative on entry then a successful break-
    marriage operation has been completed and a new stable
    marriage found. If the Boolean success is made false during
    a breakmarriage operation then it means that this break-
    marriage has failed;
  if i < 0 then success := true
  else if i = 0 ∨ malec [i] = n + 1 ∨ ¬ unchanged [i]
    then success := false
  else
  begin
    c := c + 1; j := malec [i]; malec [i] := j + 1;
    refusal (i,malechoice[i,j],malec,marriage,c)
  end of proposal;
  procedure refusal (i,j,malec,marriage,c);
    value i,j;
    integer i,j,c; integer array malec, marriage;
  comment This procedure decides whether woman j prefers man
    i or the man in marriage [j]. Whichever she rejects goes back
    to the procedure proposal to make his next choice;
  if fc [j, abs (marriage [j])] > fc[j,i] then
  begin
    k := marriage [j]; marriage [j] := i;
    proposal (k,malec,marriage,c)
  end
  else proposal (i,malec,marriage,c);
  for i := 1 step 1 until n do
  begin
    for j := 1 step 1 until n do
      fc[i,femalechoice [i,j]] := j;
    marriage [i] := 0; malecounter [i] := 1;
    fc[i,0] := n + 1; unchanged [i] := true;
  end;
  count := 0;
  for i := 1 step 1 until n do
      proposal (i,malecounter,marriage,count);
  comment Male optimal stable solution found;
  STABLE MARRIAGE (marriage,n,count);
  for i := 1 step 1 until n - 1 do
    breakmarriage (malecounter, marriage, i,n,count);
end of procedure ALL STABLE MARRIAGES
```

Algorithm 412

Graph Plotter [J6]

Josef Čermák (Recd. 19 Mar. 1970 and 12 Nov. 1970)
Department of Physics, University of Chemical Technology, Pardubice, ČSSR.

Key Words and Phrases: plot, graph, lineprinter plot
CR Categories: 4.41

```
procedure graphplotter (N, x, y, m, n, xerror, yerror, g, L, S, EM,
  C0, C1, C2, C3, C4, label);
  value N, m, n, xerror, yerror, g, L, S;
  array x, y; integer N, g, m, n, L, S; real xerror, yerror;
  string EM, C0, C1, C2, C3, C4; label label;
comment This procedure is functionally identical with Algorithm
  278. It needs, however, a significantly smaller array than Al-
  gorithm 278 for storage of the graph before it is printed. The
  procedure is intended to be used to give an approximate graph-
  ical display of a multivalued function y[i, j] of x[i], on a line
  printer. Output channel N is used for all output. The graph is
  plotted for those points such that 1 ≤ i ≤ m and 1 ≤ j ≤ n
  where 2 ≤ n ≤ 4. If n = 1, then y must be a one-dimensional
  array y[i] and the graph is plotted for x[i] and y[i] for 1 ≤ i ≤
  m. The format of the output is arranged so that a margin of g
  spaces appears on the left-hand edge of the graph. L and S
  specify the number of lines down the page and the number of
  spaces across the page which the graph is to occupy, respec-
  tively. The graph is printed so that lines 1 and L correspond to
  the minimum and maximum values of x, and character posi-
  tions 1 and S correspond to the minimum and maximum values
  of y. That is to say, y is plotted across the page and x is plotted
  down the page. After the entire graph has been plotted, the
  minimum and maximum values for x and y are printed out in
  order xmin, xmax, ymin, ymax. The argument EM represents
  the character which is printed on the perimeter of the display.
  The argument C0 represents the character printed at empty
  positions. The arguments, C1, C2, C3, C4, represent the charac-
  ters printed for y[i, 1], y[i, 2], y[i, 3], and y[i, 4], respectively.
  At those points at which more than one character would ap-
  pear, the order of preference is C1, C2, C3, C4. Control is
```