ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XIII, 3 (1973)

M. M. SYSŁO (Wrocław)

FUNDAMENTAL SET OF CYCLES OF A GRAPH

1. The procedure *cycle* finds a fundamental set of cycles of a connected undirected finite graph and — as the additional result — the spanning tree of the examined graph.

Data:

n	—	the number of nodes of the graph,
$a[1:n, 1:n]$	—	the array of the incidence matrix of the graph (in the computation is required its upper triangle only),
s	—	an integer number from the interval $[1, n]$ (root of the spanning tree).

Remark. The elements of the array *precede* $[1:n]$ must be equal zero before the first call (for the given graph) of the procedure *cycle*.

Results:

Each execution of the statement $cycnum := cycnum + 1$; means that a successive cycle of the graph was found. In the procedure *cycle* after this statement there follow 8 statements after execution of which there will be printed the sequence of symbols

$$cycle\ i_1 i_2 i_3 \ldots i_k i_1,$$

where $(i_1, i_2, i_3, \ldots, i_k, i_1)$ is the currently obtained cycle.

Additional results:

cycnum	—	the number of independent cycles of the graph,
precede $[1:n]$	—	the array of node numbers such that *precede* $[k] \neq 0$ contains the number of the node successing the node k on the chain from the node k to the node s (root of the spanning tree).

2. Method used and certification. The algorithm of Paton [2] has been used in the procedure *cycle*. Paton [2] has assessed the required store

```
procedure cycle(n,a,s,cycnum,precede);
 value n,s;
 integer n,s,cycnum;
 integer array a,precede;
 begin
  integer i,i1,j,k,l,m,sj;
  integer array p[1:n];
  cycnum:=0;
  precede[s]:=-1;
  k:=1;
  p[1]:=s;
  for i:=2,i+1 while i≤n∧k>0 do
   begin
    i1:=p[k];
    k:=k-1;
    precede[i1]:=-precede[i1];
    for j:=1 step 1 until n do
     if (if j<i1 then a[j,i1] else a[i1,j])=1
      then
      begin
       sj:=precede[j];
       if sj=0
        then
       begin
         precede[j]:=-i1;
         k:=k+1;
         p[k]:=j;
        end sj=0
        else
         if sj<0
```

```
                then
                begin
                 cycnum:=cycnum+1;
                 outstring(1,'cycle');
                 sj:=-sj;
                 outinteger(1,sj);
                 outinteger(1,j);
                 outinteger(1,i1);
                 l:=i1;
                 for m:=precede[l] while m≠sj do
                  begin
                   outinteger(1,m);
                   l:=m
                  end m;
                 outinteger(1,sj)
                end sj < 0
            end j
       end i;
      precede[s]:=0;
      i1:=p[1];
      precede[i1]:=abs(precede[i1])
     end cycle
```

by his method as n^2+3n words and this realization of his algorithm requires only $n^2/2+2n$ words of the machine store (the lower triangle of the array *a* could be used for other edge informations of the examined graph). However, according to Paton, for random graphs of various densities the time required by the procedure *cycle* increases as n^2, namely for the ODRA 1204 computer $time(n) = .0027n^2$.

3. Modifications and additional applications.

(a) The procedure *cyclecomp* finds a fundamental set of cycles of an undirected finite graph (not necessarily connected) and — as the additional result — the spanning forest of the examined graph.

```
procedure cyclecomp(n,a,cycnum,compnum,precede);
 value n;
 integer n,cycnum,compnum;
 integer array a,precede;
 begin
  integer r,s;
  for s:=1 step 1 until n do
   precede[s]:=0;
  cycnum:=compnum:=0;
  for s:=1 step 1 until n do
   if precede[s]=0
    then
    begin
     compnum:=compnum+1;
     cycle(n,a,s,r,precede);
     cycnum:=cycnum+r
    end precede[s]=0,s
 end cyclecomp
```

Data:

$\left.\begin{array}{r} n \\ a[1:n,\ 1:n] \end{array}\right\}$ as in the procedure *cycle*.

Results:

The procedure *cyclecomp* prints, for each component of the examined graph, the fundamental set of cycles (see Section 1, Results of the procedure *cycle*).

Additional results:

cycnum	– the number of independent cycles of the graph,
compnum	– the number of components of the graph,
precede $[1:n]$	– the array of node numbers such that *precede* $[k] \neq 0$ contains the number of the node successing the node k on the chain from the node k to the root of the spanning forest (the node k is the root of the spanning forest if *precede* $[k]$ is equal 0),

$cycnum + n - compnum$ – the number of edges of the graph.

(b) The fundamental cycle generation procedure could be used in an algorithm for generation of all elementary cycles from the fundamental set of cycles. The procedure *setcycles* in [5], using an above-described version of Paton's method, generates all elementary cycles of a graph by Gibbs' method [1].

(c) The fundamental cycle generation procedure could also be applied in an algorithm for finding bridges, blocks and cutnodes of a graph (see [3] and [4]). For this purpose the following properties are used:

(i) Any edge which is not a bridge belongs to at least one fundamental cycle.

(ii) Two edges are in the same block if there is a fundamental cycle which contains both edges.

(iii) Any node which is a cutnode belongs to at least two blocks.

The procedure *connectivity* finds the bridges, blocks and cutnodes of a graph.

Data:

See Section 1, data for the procedure *cycle.*

Remark. The elements of the array *precede* $[1:n]$ must be equal zero before each call of the procedure *connectivity.*

Results:

lbr	–	the number of bridges of the graph,
lb	–	the number of blocks of the graph,
lcn	–	the number of cutnodes of the graph,
$b\,[1:n]$	–	the integer array of node labels such that: if $b[k]=0$, then nodes k and s do not belong to the same component, if $b[k]\neq 0$, then $abs(b[k])$ is the number of one of the blocks (or exactly one block) which contains node k, and if $b[k]<0$, then the node k is the cutnode,
precede $[1:n]$	–	see Additional results of the procedure *cycle.*

Remark. The blocks of the graph are numbered by consecutive numbers $1, 2, \ldots, lbr$, and the fixed node k belongs to blocks the numbers of which appear at the nodes which are in incidence with the node k and at the node k.

The algorithm of Paton [3] has been used in the procedure *connectivity.* This realization of Paton's algorithm requires $5n+n^2/2$ words of machine store.

```
procedure connectivity(n,a,s,lbr,lb,lcn,b,precede);
 value n,s;
 integer n,s,lbr,lb,lcn;
 integer array a,b,precede;
 begin
  integer bm,i,i1,i2,j,k,L,L1,m,s1,s2,sj;
  integer array d,p[1:n];
  Boolean array A[1:n];
  for i:=1 step 1 until n do
   begin
    b[i]:=0;
    A[i]:=false
   end i;
  s1:=s-1;
  s2:=s+1;
  d[s]:=0;
  precede[s]:=-1;
  b[s]:=-s;
  k:=1;
  p[1]:=s;
  for i:=2,i+1 while i≤n∧k>0 do
   begin
    i1:=p[k];
    k:=k-1;
    precede[i1]:=-precede[i1];
    L:=0;
    for j:=1 step 1 until n do
     if (if j<i1 then a[j,i1] else a[i1,j])≠0
      then
      begin
      sj:=precede[j];
      if sj=0
       then
       begin
        precede[j]:=-i1;
        k:=k+1;
        p[k]:=j;
        d[j]:=d[i1]+1;
        b[j]:=-j
       end sj=0
       else
        if sj<0
         then
         begin
          i2:=b[j];
          if i2>0
           then A[i2]:=true;
          b[j]:=i1;
          L1:=d[i1]-d[-sj];
          if L1>L
           then L:=L1;
         end sj < 0
    end j,(if j < i1...);
  if L>0
   then
   begin
    m:=i1;
    for L:=L-1 while L≥0 do
     begin
      bm:=b[m];
```

```
        if bm>0
         then A[bm]:=true;
        b[m]:=i1;
        m:=precede[m]
       end L;
      for m:=1 step 1 until n do
       begin
        bm:=b[m];
        if bm>0
         then
         begin
          if A[bm]
           then b[m]:=i1
         end sj > 0...
       end m
     end L > 0
   end i;
  precede[s]:=0;
  i1:=p[1];
  precede[i1]:=abs(precede[i1]);
  lbr:=lb:=lcn:=0;
  for i:=1 step 1 until s1,s2 step 1 until n do
   begin
    i1:=b[i];
    if i1<0
     then
     begin
      lb:=lb+1;
      lbr:=lbr+1;
      b[i]:=n+lb
   end i1 < 0
   else
    if i1≤n∧i1>0
     then
     begin
      lb:=lb+1;
      i2:=n+lb;
      for j:=i step 1 until n do
       if b[j]=i1
        then b[j]:=i2
     end i1 ≤ n
 end i;
for m:=1 step 1 until n do
 begin
  bm:=b[m];
  if bm>0
   then b[m]:=bm-n
 end m;
i:=0;
for i:=i+1 while precede[i]≠s do;
b[s]:=b[i];
for i:=1 step 1 until n do
 begin
  sj:=precede[i];
  if sj>0
   then
   begin
    bm:=abs(b[sj]);
    if abs(b[i])≠bm
```

```
            then b[sj]:=-bm
          end sj > 0
        end i;
       for i:=1 step 1 until n do
        if b[i]<0
         then lcn:=lcn+1
       end connectivity
```

Example. For the graph shown in Fig. 1 the following results were obtained:

s	*lbr*	*lb*	*lcn*		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	4	3	*b*	1,	−1,	1,	−2,	−3,	4,	4,	0,	0,	0,	0,	0,	0,	0
				precede	0,	1,	1,	2,	4,	5,	5,	0,	0,	0,	0,	0,	0,	0
8	2	3	2	*b*	0,	0,	0,	0,	0,	0,	0,	1,	−1,	1,	−2,	3,	0,	0
				precede	0,	0,	0,	0,	0,	0,	0,	0,	8,	8,	9,	11,	0,	0
13	1	1	0	*b*	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	1,	1
				precede	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	13

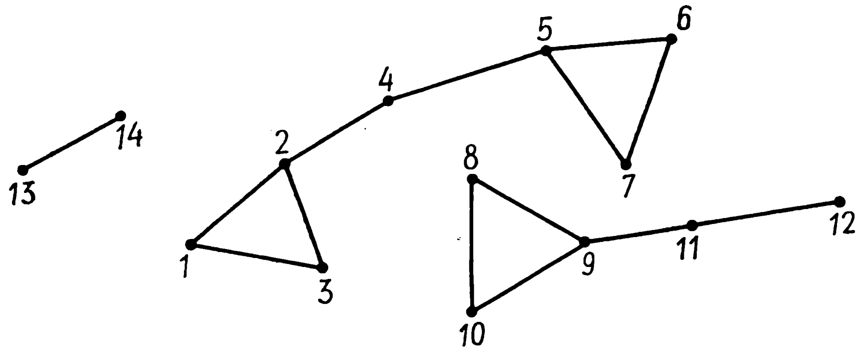

Fig. 1

References

[1] N. E. Gibbs, *A cycle generation algorithm for finite undirected linear graphs*, J. ACM 16 (1969), p. 564-568.

[2] K. Paton, *An algorithm for finding a fundamental set of cycles of a graph*, Comm. ACM 12 (1969), p. 514-518.

[3] — *An algorithm for the blocks and cutnodes of a graph*, ibidem 14 (1971), p. 468--475.

[4] R. C. Read, *Teaching graph theory to a computer* in *Recent progress in combinatorics* (Tutte, Ed.), Academic Press, New York 1969, p. 161-173.
[5] M. M. Sysło, *Procedure setcycles* in *The library of ALGOL programs and procedures of the ODRA 1204 computer*, Elwro-Wrocław 1972.

DEPT. OF NUMERICAL METHODS
UNIVERSITY OF WROCŁAW

Received on 7. 7. 1972

M. M. SYSŁO (Wrocław) ALGORYTMY 20-22

WYZNACZANIE FUNDAMENTALNEGO ZBIORU CYKLI GRAFU

STRESZCZENIE

Procedura *cycle* (*cyclecomp*) wyznacza fundamentalny zbiór cykli niezorientowanego grafu spójnego (niekoniecznie spójnego), a jako wynik dodatkowy otrzymujemy drzewo częściowe (las częściowy) badanego grafu.

Dane:

n —	liczba wierzchołków grafu,
$a\,[1:n,\,1:n]$ —	tablica całkowita zawierająca macierz incydencji grafu (w obliczeniach wykorzystywana jest tylko górna część trójkątna),
s —	(dla procedury *cycle*) liczba całkowita z przedziału $[1, n]$ (korzeń drzewa częściowego).

Wyniki:

Procedury *cycle* i *cyclecomp* drukują fundamentalny zbiór cykli grafu.

Wyniki dodatkowe:

cycnum —	liczba niezależnych cykli grafu,
compnum —	(dla procedury *cyclecomp*) liczba komponent spójności grafu,
precede $[1:n]$ —	tablica zawierająca takie numery wierzchołków, że *precede* $[k] \neq 0$ jest numerem wierzchołka następującego po wierzchołku k w łańcuchu prowadzącym z wierzchołka k do korzenia drzewa (lasu) częściowego. Wierzchołek k jest korzeniem drzewa (lasu) częściowego, jeśli *precede* $[k] = 0$.

W paragrafie 3 opisane zostały dalsze zastosowania procedury wyznaczającej fundamentalny zbiór cykli grafu: wyznaczanie zbioru wszystkich cykli elementarnych grafu (procedura *setcycles* [5]), wyznaczanie mostów, bloków i wierzchołków rozspajających grafu (procedura *connectivity*).

Obliczenia, wykonane na maszynie cyfrowej ODRA 1204, wykazały poprawność przedstawionych procedur.
