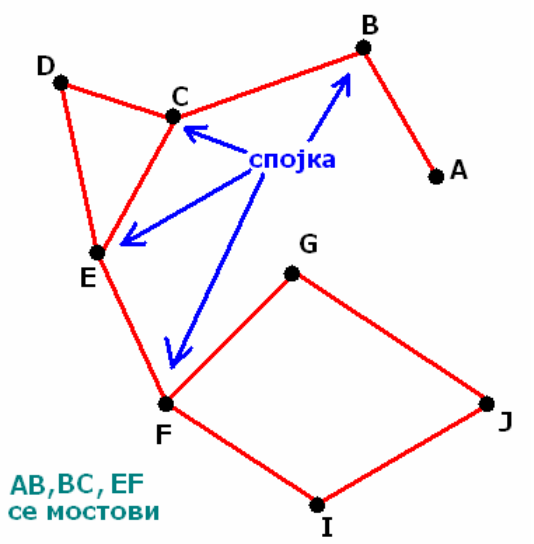
algoritam.blog.com.mk
2007

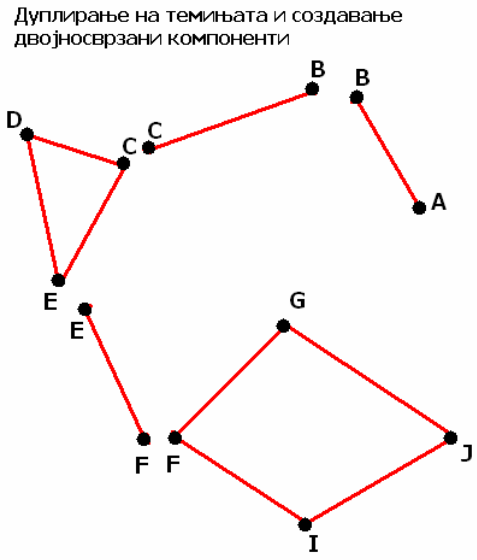
Мостови и спојки

Нека дадениот граф G(V,E) е сврзан. Реброто r кај графот G го нарекуваме **мост**, ако доколку го извадиме реброто r, графот ќе биде поделен на две сврзани подкомпоненти. Втората можност е графот од сврзан да се направи несврзан со едноставно поместување(отстранување) на теме. Ваквите темиња ги нарекуваме **спојки**.



Слика 1

Ако спојките ги дуплираме добиваме подкомпоненти од графот кои не содржат спојки. Графот е двојносврзан ако нема спојки.



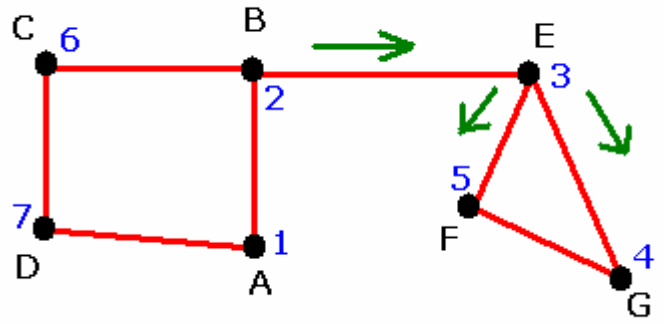
Слика 2

Пронаоѓањето на мостови и спојки ќе оди според следнава аналогија: Замислете дека графот ги претставува познанствата меѓу луѓето од некоја група. Темињата се луѓето, притоа врска помеѓу ***i***-тиот и ***j***-тиот човек постои доколку тие се познаваат, и обратно; врска не постои доколку тие не се познаваат.
Секој човек од групата на секој познаник му го вели следново:

> *Тргни од тие луѓе што ти ги познаваш, распрашувај се натака за нивните познанства, па и нека те запознаваат со нивните познаници,и така натака, па ако на тој начин дојдеш до познанство кое е и мое познанство тогаш значи дека јас не сум единствена врска која ќе ти ја заврши работата т.е. не претставуваме мост.*

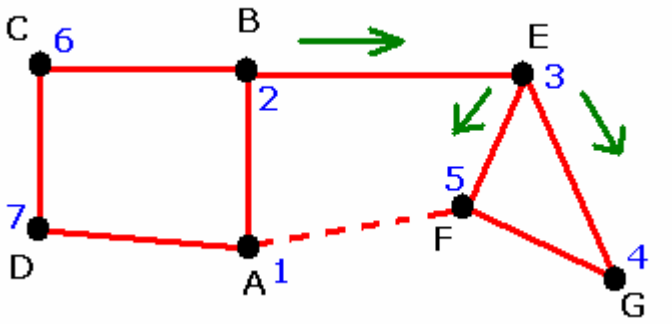
Токму во тоа се крие и решението за откривање на мостовите и спојките во графот.

Со помош на DFS, темињата од графот ги нумерираме од 1 до |V|. Значи, едно теме ***v*** има нумерација **num[*v*]**. Покрај нумерацијата, постои и друг елемент кој му го придружуваме на секое теме. Овој елемент ќе го означуваме како **mpt[*v*]**.



Слика 3

Нумерацијата е означена со сини бројки. Темето **B** го повикува својот "познаник" **E**(слика 3), и му вика: Започни си засебен DFS, и памти го најмалиот број(**mpt[*v*]**) од нумерацијата на темињата што ќе ги сретнш во текот на засебниот DFS. Ако откако ќе завршиш со DFS, твоето **mpt** е поголем или еднаков од твојата нумерација, тогаш јас и вие градиме мост. Тоа е причината зошто **BE** престставува мост. Бројот **mpt** што **E**, ќе го врати како резултат на ваквиот процес, ќе изнесува 3 што е еднакво на неговата нумерација . Значи **BE** е мост.



Слика 4

Нека сега, поставиме уште едно ребро помеѓу темињата **A** и **F**(слика4). Кога темето **E** ќе ги пушти DFS барањата преку **F** и **G**, како одговор за најмало бројче ќе добие 1,(До оваа вредност се доаѓа преку темето **F** кое е во врска со **A**, чија нумерација е 1) што е помало од неговата нумерација, што значи дека во овој случај **BE** не е мост.

Пример програма

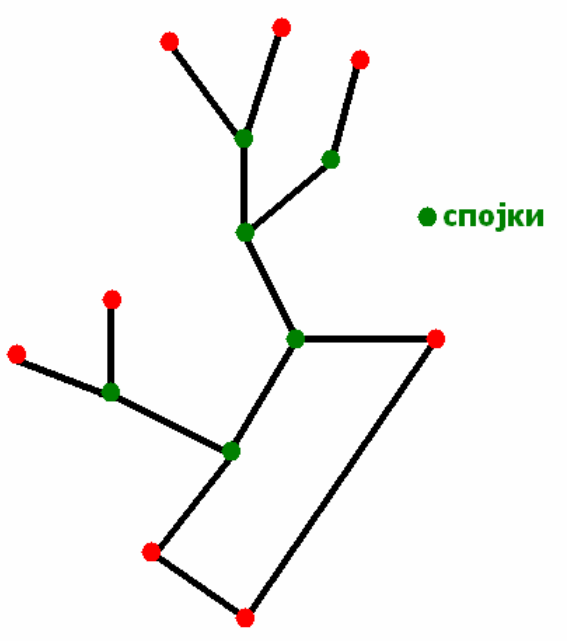
```
Program mostovi;
Var
      matrica:array[1..20,1..20]of boolean;{матрица на соседство}
      num:array[1..20]of integer;{ низа за нумерација}
      mpt:array[1..20]of integer;{низа за минимален број од нумерацијата на
следбениците}
      br,i,j,prva,vtora:integer;
      vrski:text;

procedure baraj_most(t,p:integer);{p се однесува на темето претходник на темето t}
var
  i:integer;
begin
            inc(br);
            num[t]:=br;{го нумерираме темето}
            mpt[t]:=br;{истата вредност на почеток ја задаваме и за mpt}
            for i:=1 to 20 do
              begin
                    if matrica[t,i]and(i<>p) {продолжуваме со теме со кое има
врска}
                      then
                          begin
                                if num[i]=0 then{ако следното теме не е
нумерирано}
                                  begin
                                        baraj_most(i,t);{се повикува рекурзивно
истата процедура но сега за темето i и неговиот родител t}
                                if (mpt[i]>=num[i]) then {услов за мост}
                         writeln(i,' ',t);
                                        if mpt[t]>mpt[i]{ако после завршувањето на
рек. повик детето теме i има помал mpt отколку на t}
                                           then
                                               mpt[t]:=mpt[i];
                                  end
                                      else
                                          if (num[i]<mpt[t])[1]and(i<>p){ ако темето
t дојде до теме кое е веке нумерирано, а не е негов родител}
                                             then
                                                 mpt[t]:=num[i];
                          end;
              end;
end;

begin
      for i:=1 to 20 do
            begin
                  num[i]:=0;
```

[1] Во книгата што служи како извор за оваа тема[1], во приложениот код од истата книга, стои кодот **if (num[i]<num[t])and…** . но ако го земеме фактот дека **mpt[t]<=num[t]** за секое t, а **mpt[t]** го менуваме само во случај кога неговата вредност е помала од **num[i]**, значи нема потреба да се проверува **num[i]<num[t]**, туку да се искористи условот **num[i]<mpt[t].**

```
                mpt[i]:=0;
                for j:=1 to 20 do
                      matrica[i,j]:=false;
          end;
      assign(vrski,'vrski.in');
      reset(vrski);
      while not eof(vrski) do
          begin
                read(vrski,prva);
                read(vrski,vtora);
                matrica[prva,vtora]:=true;
                matrica[vtora,prva]:=true;
          end;
      br:=0;
      baraj_most(1,0);
      close(vrski);
end.
```



Слика 5

Спојките се препознаваат полесно. Кога ќе се пушти повик рекурзивно во рамки на процедурата за теме **k**, за теме кое не е нумерирано, нека тоа го означиме со **i**, по извршувањето на повикот, се проверува дали **mpt** на детето-теме **i** е поголемо или еднакво од нумерацијата на темето родител **k**. Ако ова е точно, тогаш **k** е спојка.

Процедура за пронаоѓање на спојките

```
procedure baraj_spojka(t,p:integer);
var
  i:integer;
begin
          inc(br);
          num[t]:=br;{го нумерираме темето}
          mpt[t]:=br;{истата вредност на почеток ја задаваме и за mpt}
          for i:=1 to 20 do
            begin
                  if matrica[t,i]and(i<>p) {продолжуваме со теме со кое има
врска}
```

```
                    then
                        begin
                              if num[i]=0 then{ако следното теме не е
нумерирано}
                                begin
                                      baraj_spojka(i,t);{се повикува рекурзивно
истата процедура но сега за темето i и неговиот родител t}
                                      if (num[t]<=mpt[i])and(not oznacen[t]){ако
после завршувањето на рек. повик детето теме i има помал mpt отколку нумерацијата
на t, услов за спојка}
                                        then
                          begin
                                writeln(t);
                                oznacen[t]:=true;
                          end;
                       if mpt[t]>mpt[i]
                         then
                          mpt[t]:=mpt[i];
                  end
                                    else
                                        if (num[i]<mpt[t])and(i<>p){ ако темето
t дојде до теме кое е веке нумерирано, а не е негов родител}
                                          then
                                              mpt[t]:=num[i];
                            end;
              end;
end;
```

Забелешка: Како почетен јазол од кој треба да се започне со пребарувањето на спојки, треба да се избере јазол кој учествува во барем две ребра, односно има степен кој е поголем или еднаков на 2. Затоа наместо веднаш да се започне со јазолот 1, би требало најпрво да се провери дали јазолот 1 учествува во најмалку две ребра. Тоа е затоа што доколку тргнеме да бараме спојки, од завршно теме во графот(лист; теме со степен 1), тогаш алгоритмот ќе заврши во истото тоа теме. Но откако алгоритмот ќе ја префрли контролта на почетното теме, неговата нумерација ќе биде сигурно помала или еднаква од mpt на единственото дете по кое е пуштено пребарувањето на спојките. Тоа доведува до означување на почетното теме и испишување на почетното теме-лист како спојка, што воопшто не е случај.

Користена литература

1. *Теорија на графови* – Проф. Д-р. Душан Чакмаков