Central European Olympiad in Informatics

28 July – 4 August 2005 Sárospatak, Hungary
http://ceoi.inf.elte.hu

Critical Network Lines

Consider the graph G whose nodes are the network nodes and its edges are the direct communication lines.
A critical network line must be a bridge in G. Let (u,v) be a bridge of G. Removing the bridge (u,v) from G splits G into two disjoint subgraphs, say G_u and G_v. (u,v) is a critical network line if and only if

$$nA = 0 \vee nA = K \vee nB = 0 \vee nB = L$$

where K is the number of nodes that provide service type **A**, L is the number of nodes that provide service type **B**, nA is the number of nodes in the subgraph G_v that provide service type **A** and nB is the number of nodes in the subgraph G_v that provide service type **B**.
The bridges of G can be detected by depth first search. If (u,v) is a bridge then it is a tree edge of the depth first spanning tree. Assume that the orientation in the tree is $u \rightarrow v$. Then the subtree of the spanning tree rooted at v contains exactly the nodes of the subgraph G_v. Hence it is enough to compute for each node p the number of service providing nodes of type A and of type B that are in the subtree rooted at p. This computation can be incorporated into the depth first search procedure.

Complexity:
Time: $O(n+m)$
Memory: $O(n+m)$
Implementation

```
Program Net;
Const
  MaxN=100000;
Type
  PointType=0..MaxN;
  List=^Cell;
  Cell=Record P:PointType; link:List End;
  Graph=Array[1..MaxN] Of List;
  Palette=(White, Gray, Black);
Var
  N,M,Ka,Lb:longint;
  G:Graph;
  Parent,D,L:Array[1..MaxN] of longint;
  Color:Array[1..MaxN] Of Palette;
  A,B:Array[1..MaxN] Of boolean;
  E:array[1..MaxN] of
      record p,q: longint end; {the critical lines}
  S,Time,u,Na,Nb:longint;
  outFile:Text;

procedure ReadIn;
var inFile:Text;
  Guv,Gvu:List;
  u,v,i:longint;
begin
  assign(inFile,'net.in'); reset(inFile);
```

```
  readln(inFile,N,M,Ka,Lb);
  For u:=1 To N Do begin
    G[u]:=Nil;
    A[u]:=false;
    B[u]:=false;
  end;
  For i:=1 To Ka Do Begin
    read(inFile,u);
    A[u]:=true;
  end;
  For i:=1 To Lb Do Begin
    read(inFile,u);
    B[u]:=true;
  end;
  readln(inFile);

  For i:=1 To M Do Begin
    ReadLn(inFile, u,v);
    New(Guv);
    Guv^.p:=v; Guv^.link:=G[u];
    G[u]:=Guv;
    New(Gvu);
    Gvu^.p:=u; Gvu^.link:=G[v];
    G[v]:=Gvu;
  End{for i};
  close(inFile);
end{Readin};

procedure DFS(p:PointType; var Na,Nb:longint);
 {Global: G, Color, Time, D, L, A,B}
 {Out: Na is the number of nodes in the subtree that provide service A,
       Nb is the number of nodes in the subtree that provide service B}
 Var
  q : PointType;
  pq:List;
  Naq,Nbq,Nap,Nbp:longint;
Begin{DFS}
  inc(Time);
  D[p]:=Time;
  L[p]:=Time;
  Color[P]:=Gray;
  pq:=G[p];
  Nap:=0; Nbp:=0;
  while pq<>nil  Do begin
    q:=pq^.p;
    If Color[q]=White Then begin
      parent[q]:=p;
```

```
      DFS(q, Naq, Nbq);
      Nap:=Nap+Naq; Nbp:=Nbp+Nbq;
      if L[q]<L[p] then L[p]:=L[q];
    end else if (Color[q]=Gray)and(parent[p]<>q)and(D[q]<L[p]) then
      L[p]:=D[q];
    pq:=pq^.link;
  end{while};
  if A[p] then inc(Nap);
  if B[p] then inc(Nbp);
  if (p<>1)and(L[p]=D[p])and ((Nap=0)or(Nap=Ka)or(Nbp=0)or(Nbp=Lb)) then begin
    {parent[p]-p is a bridge and a critical line}
     inc(S);
     E[S].p:=p; E[S].q:=Parent[p];
  end;
  Na:=Nap;
  Nb:=Nbp;
  Color[p]:=Black;
End {DFS};

Begin{Prog}
  ReadIn;

  Time:=0;
  for u:=1 to N do Color[u]:=White;
  S:=0;
  DFS(1,Na,Nb);

  assign(outFile,'net.out'); rewrite(outFile);
  writeln(outFile, S);
  for u:=1 to S do
    writeln(outFile, E[u].p,' ',E[u].q);
  close(outFile);
End.
```