Laboratorul 2

Problema tăieturii minime

Considerăm un graf (neorientat) $G = (V, E)$ (V e mulţimea vârfurilor, E e mulţimea muchiilor) care este conex (adică, între oricare două vârfuri există cel puţin un drum). Spunem că $C \subseteq M$ este o mulţime de tăietură pentru G, dacă graful $G_C = (V, E \setminus C)$ nu este conex.

Problema tăieturii minime: să se determine o mulţime de tăietură C pentru G astfel încât $|C|$ (i.e. cardinalul lui C) este minim. O astfel de mulţime de tăietură spunem că este *minimă*.

Ideea de bază a algoritmului aleator *Random Min-Cut* este de a contracta, la fiecare iteraţie, câte o muchie aleasă aleator, până când mai rămân doar 2 vârfuri. O contracţie a unei muchii $e = (u, v)$, care leagă vârfurile u şi v, presupune: eliminarea muchiilor care conectează pe u cu v, înlocuirea nodurilor u şi v cu un nou nod w şi păstrarea tuturor celorlalte muchii, inclusiv a celor care erau conectate cu u sau v, acestea fiind conectate acum cu w.

Pseudocodul algoritmului:

```
RndMinCut(G)
begin
  repeat
   randomly pick an edge and contract it
  until |V|=2
  return the set of remaining edges
end
```

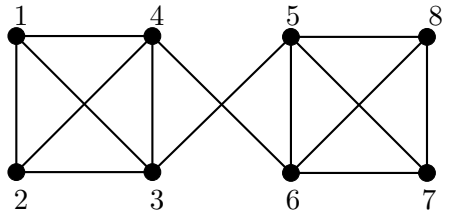
Fie $|V| = n$ ($n \in \mathbb{N}, n > 2$).

Din Seminarul 2 avem:

• Probabilitatea ca `RndMinCut(G)` să genereze o mulţime de tăietură minimă pentru graful $G = (V, E)$ este cel puţin $\frac{2}{n(n-1)}$.

• Dacă rulăm `RndMinCut(G)` de $C_n^2 \cdot [\ln n]$ ori şi returnăm dintre toate mulţimile de tăietură generate una cu cardinal minim, atunci probabilitatea ca mulţimea de tăietură returnată să nu fie minimă este este cel mult $\frac{1}{n}$.

a) Implementaţi în Matlab `RndMinCut(G)` şi testaţi-l pe graful următor.



Pentru a rezolva cerinţa dată la a), puteţi completa spaţiile punctate din codul următor, care foloseşte funcţia *graph* pentru a genera o *clasă* asociată unui graf (puteţi apela *help graph* pentru a afla mai informaţii depre *obiectele, metodele* şi *proprietăţile* corespunzătoare clasei generate de *graph*).

```
%Cod pentru Matlab 2018a sau 2018b.

%Exemplu de generare a unui graf si de apel pentru RndMinCut:
%>>E=[1 2;1 3;1 4;2 3;2 4;3 4;3 5;4 6;5 6;5 7;5 8;6 7;6 8;7 8];
%>>G=graph(E(:,1),E(:,2));
%>>H=RndMinCut(G);
%>>H.Nodes
%>>H.Edges

function G=RndMinCut(G)
n=numnodes(G); G.Nodes.Name(:,1)=cellstr(string(1:n));
subplot(1,n-1,1); plot(G,'-k');
axis square; axis off;
for i=2:n-1
    e=randi(numedges(...));
    u=findnode(G,G.Edges.EndNodes(...,1)); v=findnode(G,G.Edges.EndNodes(...,2));
    G=rmedge(G,...,...);
    G = addnode(G,strcat(G.Nodes.Name(...,1),',',G.Nodes.Name(...,1)));
    G_old=G;
    G=rmnode(G,G.Nodes.Name([...,...],1));
    for j=1:numedges(...)
        if ismember(G_old.Edges.EndNodes(...,1),G_old.Nodes.Name([...,...],1))
            G=addedge(G,G_old.Edges.EndNodes(...,2),G_old.Nodes.Name(numnodes(...),1));
        elseif ismember(G_old.Edges.EndNodes(...,2),G_old.Nodes.Name([...,...],1))
            G=addedge(G,G_old.Edges.EndNodes(...,1),G_old.Nodes.Name(numnodes(...),1));
        end
    end
    subplot(1,n-1,i); plot(G,'-k');
    axis square; axis off;
end
```

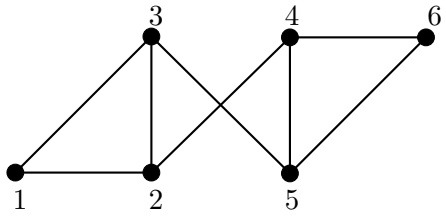
```
%Cod pentru versiuni de Matlab intre 2015b si 2017b.

%Exemplu de generare a unui graf si de apel pentru RndMinCut:
%>>E=[1 2;1 3;1 4;2 3;2 4;3 4;3 5;4 6;5 6;5 7;5 8;6 7;6 8;7 8];
%>>G=graph(E(:,1),E(:,2),ones(size(E,1),1));
%>>H=RndMinCut(G);
%>>H.Nodes
%>>H.Edges.Weight

function G=RndMinCut(G)
n=numnodes(G); G.Nodes.Name(:,1)=cellstr(string(1:n));
subplot(1,n-1,1); plot(G,'-k');
axis square; axis off;
```

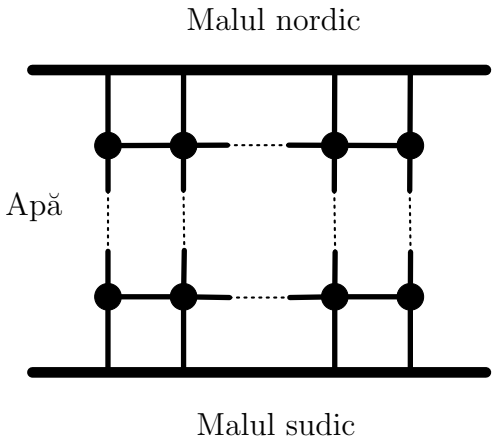
```
for i=2:n-1
    e=randsample(1:numedges(...),1,true,G.Edges.Weight/sum(G.Edges.Weight));
    u=findnode(G,G.Edges.EndNodes(...,1)); v=findnode(G,G.Edges.EndNodes(...,2));
    G=rmedge(G,...,...);
    G = addnode(G,strcat(G.Nodes.Name(...,1),',',G.Nodes.Name(...,1)));
    G_old=G;
    for j=1:numedges(...)
        nodes_edge_j=findnode(G,G_old.Edges.EndNodes(...,:));
        if ismember(nodes_edge_j(1),[...,...])
           ide=findedge(G,nodes_edge_j(2),numnodes(G));
           if ide==0
             G=addedge(G,nodes_edge_j(...),numnodes(...),G_old.Edges.Weight(...));
           else
             G.Edges.Weight(ide)=G.Edges.Weight(ide)+G_old.Edges.Weight(j);
           end
        elseif ismember(nodes_edge_j(2),[...,...])
           ide=findedge(G,nodes_edge_j(1),numnodes(G));
           if ide==0
             G=addedge(G,nodes_edge_j(...),numnodes(G),G_old.Edges.Weight(...));
           else
             G.Edges.Weight(ide)=G.Edges.Weight(ide)+G_old.Edges.Weight(j);
           end
        end
    end
    G=rmnode(G,G.Nodes.Name([...,...],1));
    subplot(1,n-1,i);
    plot(G,'-k','LineWidth', G.Edges.Weight);
    axis square; axis off;
end
```

b) Rulaţi, fără reprezentări grafice, RndMinCut(G) de $C_n^2 \cdot [\ln n]$ ori pentru graful următor şi afişaţi cardinalul cel mai mic dintre cardinalele mulţimilor de tăietură generate. Este cardinalul afişat al unei mulţimi de tăietură minimă?

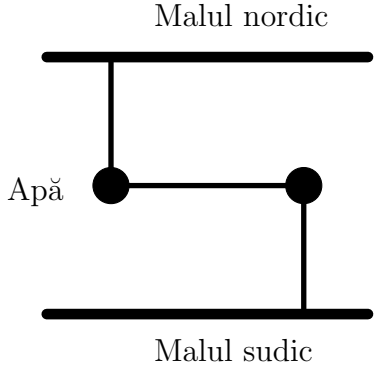


Problema podurilor

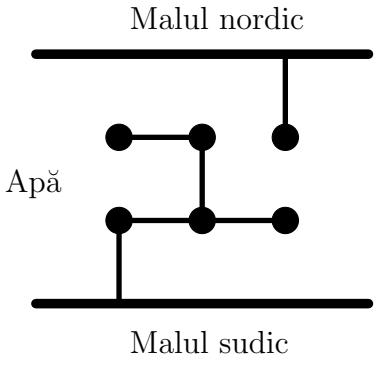
Un călător trebuie să ajungă de pe malul sudic pe malul nordic ale unei mări, traversând o reţea de poduri şi insule, reprezentate în figura de mai jos. Insulele sunt distribuite pe n linii şi $n+1$ coloane ($n \in \mathbb{N}^*$).



Cu o zi înaintea sosirii călătorului, a avut loc o furtună, în urma căreia fiecare pod, independent de celelalte, a fost distrus cu probabilitate 50%. În figura următoare avem reprezentată o reţea de insule şi poduri după furtună, pentru $n = 2$, care îi permite călătorului să treacă de pe un mal pe altul.



În figura următoare avem reprezentată o reţea de insule şi poduri după furtună, pentru $n = 3$, care nu îi permite călătorului să treacă de pe un mal pe altul.



a) Simulaţi grafic, pentru n dat, configuraţia podurilor după furtună şi afişaţi dacă este posibilă sau nu trecerea de pe malul sudic pe malul nordic pentru călător.

Pentru a rezolva cerinţa dată la a), puteţi să completaţi spaţiile punctate din codul următor.

```
function storm(n)
clf; axis equal; axis off;hold on;
axis([-2 n+3 -2 n+3]);
plot(repmat(1:n,n+1,1),'ok','MarkerFaceColor','k','MarkerSize',8);
line([0 n+2],[0 0],'Color','k','LineWidth',3);
text(2,-1,'Malul sudic');
line([0 n+2],[n+1 n+1],'Color','k','LineWidth',3);
text(2,n+2,'Malul nordic');
G=graph;
G=addnode(G,(n+2)*(n+1));
for i=1:n+1
    for j=1:n+1
        if binornd(1,0.5)
            line([j j],[i-1 i],'Color','k','LineWidth',1.5);
            G=addedge(G,...,...);
        end
    end
end
for i=1:n
    for j=1:n
        if binornd(1,0.5)
            line([j j+1],[i i],'Color','k','LineWidth',1.5);
            G=addedge(G,...,...);
        end
    end
end
if any(any(distances(G,...,...)~=Inf))
    disp('Se poate trece de pe un mal pe altul.');
else
    disp('Nu se poate trece de pe un mal pe altul.');
end
```

b) Simulaţi, pentru un n fixat, fără reprezentări grafice, de $N \in \{100, 1000\}$ ori configuraţia podurilor după furtună şi afişaţi procentul configuraţiilor în care este posibilă trecerea de pe malul sudic pe malul nordic pentru călător.

c) Determinaţi probabilitatea evenimentului A: "călătorul poate să treacă de pe malul sudic pe malul nordic".

Rezolvare:

c) Considerăm evenimentul B: "călătorul poate să treacă pe mare cu o barcă din partea stângă în partea dreaptă ale reţelei de insule", presupunând că un pod care nu este distrus blochează trecerea bărcii, iar un pod distrus permite trecerea bărcii.

Se observă că evenimentul B este echivalent cu evenimentul $\bar{A}$, identificând fiecare zonă de apă, delimitată de poduri, cu un nod, la fel ca în figura de mai jos, deoarece reţeaua de noduri pe apă are aceeaşi structură ca reţeaua de insule (n linii şi $n+1$ coloane, din punctul de vedere al călătorului atât pe uscat, cât şi pe apă) şi orice pod distrus, respectiv intact, între două insule, reprezintă legătura, respectiv lipsa ei, între două noduri corespunzătoare pe apă. Deoarece fiecare pod este distrus, independent de celelalte, cu probabilitatea 0,5, deducem că $P(A) = P(\bar{A})$. Deci $P(A) = 0{,}5$.