KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

DISKREČIOSIOS STRUKTŪROS (P170B008)

KURSINIS DARBAS

Užduoties nr. B15

Atliko:

IFF-1/1 gr. studentas

Vytenis Kriščiūnas

Priėmė:

Doc. Martynas Patašius

KAUNAS

2022

Turinys

1. Užduotis (B15)

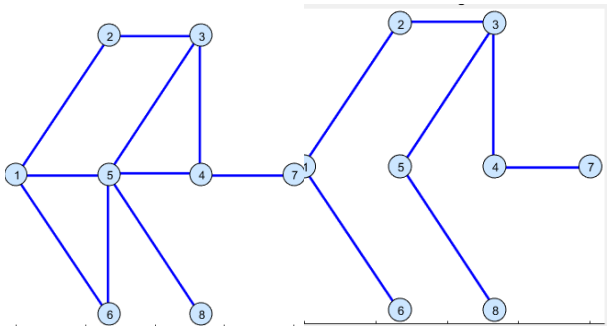
Rasti visus duoto grafo dalinius grafus, kurie yra medžiai.

2. Užduoties analizė

Dalinis grafas – tai grafas, turintis tą pačią viršūnių aibę ir dalį pradinio grafo briaunų (lankų). Medis – tai grafas, kuris yra jungus ir neturi ciklų. Medžio radimui grafe galima naudoti kelis algoritmus, tačiau čia naudosimės paieška į gylį.

Uždavinys. Duotas grafas $G = (V,U)$, kur V – viršūnių, o U – briaunų aibės. Reikia rasti visus įmanomus medžius duotame grafe ir juos pateikti medžio briaunų matricos formatu.

Metodo idėja. Pradžioje iš duotų viršūnių ir briaunų aibių yra sudaroma gretimumo struktūra. Tada atitinkami gretimumo struktūros nariai yra maišomi vietomis, kad paieška į gylį būtų unikali ir rastume unikalius grafo medžius. Turint vis besikeičiančia gretimumo struktūra galima kreiptis į paieškos į gylį algoritmą, kuris randa ieškomą dalinį grafą, kuris yra medis. Šis algoritmas veikia (n – 1) kartų, kur n yra viršūnių skaičius. Aplankytos viršūnės yra talpinamos masyve, kad nesusidarytų ciklai. Į matricą yra talpinamos medžio viršūnės. Galiausiai gavus visą atsakymo matricą, galima kurti briaunų aibes. Gautos medžių matricos yra talpinamos masyve ir jų pagalba galima nuspręsti ar tam tikras grafo medis yra rastas ar ne. Gautos medžio briaunų aibės yra talpinama atsakymo masyve. Atsakyme yra pateikiamas rastų medžių skaičius ir medžių briaunų aibės. Norint patikrinti, kiek grafas turi dalinių grafų, kurie yra medžiai galima naudotis internete pateiktu skaičiuotuvu, suvedant -1 reikšmes į atitinkamas gretimumo matricos vietas ir šios matricos įstrižainę pakeitus atitinkančiais viršūnių laipsniais.



1 pav. Duotasis grafas ir gautasis dalinis grafas, kuris yra medis

3. Programos algoritmo aprašymas

Programa gaus viršūnių ir briaunų aibes, taip pat vektorių matricą, kad grafo taškai būtų tam tikrose vietose ir reikalingus parametrus grafo nubraižymui.

Pradžioje yra nubraižomas duotasis grafas pagal teorinėje medžiagoje dėstytojo pateiktą kodą. Tada yra randama pradinio grafo gretimumo struktūra (GAM), kurios radimo algoritmas yra randamas pateiktoje teorinėje medžiagoje. Jis naudoja viršūnių ir briaunų aibes, taip pat narį nusakantį, ar grafas orientuotas ar ne.

Pirmojo ciklo ribų nustatymui yra gaunamas (count) skaičius, kuris randamas apskaičiavus viršūnių skaičių. Atsakymo formavimui yra sukuriamas (atsMas) rastų medžių masyvas ir (arrayOfMatrix) rastų medžių gretimumo matricų masyvas, taip pat (index) indeksas, reikalingas nurodyti vietą (atsMas), jis nuolatos didėja.

Pirmasis ciklas įvyksta dešimt kartų. Antrasis ciklas yra pradedamas nuo vieneto. Šio ciklo iteracijų indeksas, kuris nuolatos didėja yra priskiriamas kintamajam (v) nurodančiam, nuo kurios viršūnės bus pradėta paieška į gylį. Ciklo viduje esantis ciklas (trečiasis ciklas) nuolatos iteruoja per gretimumo struktūros viršūnes, o ketvirtasis ciklas esantis trečiojo ciklo viduje iteruoja per gretimumo struktūros viršūnių gretimas viršūnes. Ketvirtojo ciklo viduje vyksta pirmosios viršūnės sukeitimas su tolesnėmis viršūnėmis, kitaip tariant, gretimumo struktūros viršūnių gretimos viršūnės yra maišomos viena su kita, norint išgauti unikalias gretimumo struktūras, kurios bus naudojamos paieškos į gylį algoritme. Toliau yra kreipiamasi į paieškos į gylį algoritmą, kuriam yra duodami trys nariai: esama gretimumo struktūra (GAM), viršūnė, nuo kurios bus pradedama paieška į gylį (v) ir viršūnių skaičius (count). Šis algoritmas gražina medžio briaunų aibę ir medžio matricą.

Radus dalinį grafą, kuris yra medis, reikia patikrinti ar rastasis medis jau buvo rastas seniau. Tam yra naudojamas medžių gretimumo matricų masyvas (arrayOfMatrix). Paleidus ciklą yra iteruojama per medžių matricų masyvą ir tikrinama ar gautasis medis yra naujas šiam masyvui. Jei gautasis medis yra naujas, šio medžio briaunų aibė yra talpinama (atsMas) medžių masyve ir (arrayOfMatrix) rastų medžių gretimumo matricoje.

Paieškos į gylį algoritmas veikia gavus jau minėtuosius narius ir susikūrus kelis naujus narius. Kuriamas nulių masyvas (pastIndexes), į kurį bus talpinamos aplankytos viršūnės. Sukuriama nulių gretimumo matrica (atsMas), kurioje bus vienetais žymimos medžių viršūnės, taip pat kintamasis (placeOfPastIndex), kuris nuolatos didės vienetu ir bus reikalingas nustatyti masyvo (pastIndexes) viršūnių vietai ir jų patalpinimui masyve. Kintamasis (goBackPerOne) didėja vienetu tam tikrose ciklo vietose ir taip pat yra skirtas nustatyti masyvo (pastIndexes) viršūnių vietai. (counter) kintamasis didėja vienetu radus neaplankytą viršūnę ir yra reikalingas ciklo užbaigimui radus n – 1 viršūnių skaičių.

Pirmajame cikle (Ind) kintamajam yra priskiriama paieškos gretimumo struktūroje viršūnė (v). Antrajame cikle yra tikrinama ar esama viršūnė (Ind) egzistuoja (pastIndexes) masyve. Jei ji neegzistuoja, ji yra talpinama aplankytų viršūnių masyve (pastIndexes). Trečiajame cikle yra iteruojama per gretimumo struktūros viršūnių gretimas viršūnes ir ketvirtajame cikle tikrinama ar esama viršūnė jau buvo aplankyta, kitaip tariant yra patalpinta (pastIndexes) masyve. Jei šios viršūnės aplankytų viršūnių masyve nėra ji yra pažymima vienetu (atsMas) medžio gretimumo matricoje. Kintamajam (v) yra priskiriama esama viršūnė, nuo kurios kitoje ciklo iteracijoje bus ieškomos jai gretimos viršūnės. Grįžus į pirmąjį ciklą yra tikrinama ar (v) ir (Ind) kintamieji yra lygus, jei jie lygūs yra daroma išvada, kad esama viršūnė patalpinta kintamajame (Ind) neturi neaplankytų gretimų viršūnių – jos visos yra patalpintos (pastIndexes) masyve, todėl šiame masyve yra ieškoma prieš tai buvusios viršūnės iš (placeOfPastIndex) kintamojo atimant (goBackPerOne) kintamąjį ir rezultatą priskiriant (v). (goBackPerOne) kintamasis didėja vienetu tol, kol nėra randama nauja viršūnė.

Norint išvengti pasikartojančių medžių reikia susikurti naują, iš nulių sudarytą gretimumo matricą (perMatr), į kurią bus talpinami atitinkamose vietose rasti vienetai. Pradinis ciklas iteruoja per (atsMas) pradinę medžio gretimumo matricą vertikaliai, o kiti du ciklai horizontaliai. Pirmas ciklas esanti pradinio ciklo viduje ieško vienetų ir kai juos randa atitinkamose vietose vienetus įrašo į (perMatr) matricą, o (atsMas) pradinėje matricoje tose vietose įrašo nulius. Prieš pradedant antrąjį ciklą yra invertuojama (atsMas) pradinė matrica ir visi veiksmai, kurie buvo atlikti pirmajame cikle yra kartojami.

Galiausiai reikia iš rastojo medžio viršūnių sudarytos matricos sukurti briaunų aibių masyvą. Šio masyvo kūrimui yra naudojami du ciklai iteruojantys per gretimumo matricą. Į atskiras masyvo briaunų

aibes yra talpinamos dvi atitinkamos viršūnės, tai atliekama tada, kai gretimumo matricoje yra randami vienetai.

Kadangi iš vieno grafo galima gauti labai daug dalinių grafų, kurie yra medžiai, programos veikimas tampa pakankamai lėtas. Ši programa yra pagrysta principu – sukurti kuo įmanoma daugiau skirtingų gretimumo struktūrų variantų, kad paieškos į gylį algoritmas rastų visus medžius.

Pasirinkau „MATLAB“ programavimo kalbą, nes jos suteikiamos galimybės šios programos įgyvendinimui pasirodė plačiausios. Matricų naudojimas labai palengvina algoritmo įgyvendinimą, kitose programavimo kalbose tai padaryti būtų sudėtingiau.

4. Programos tekstas

B15.m

```
clc; close all; clear all

%Galima atkomentuoti virš?ni? ir briaun? aibes arba sukurti naujus U ir V
V=[1 2 3 4 5 6 7 8];% grafo virsuniu aibe
U={[1 2],[1 5],[1 6],[2 3],[3 4],[3 5],[4 5],[4 7],[5 6],[5 8]};

%V=[1 2 3 4 5 6];
%U={[1 2],[1 6],[2 3],[2 4],[3 4],[3 5],[3 6],[4 5]};

%V=[1 2 3 4];
%U={[1 2],[1 4],[2 3],[3 4]};

%V=[1 2 3 4 5];
%U={[1 2],[1 4],[2 3],[2 5],[3 4],[4 5]};

Vkor=[-1 0; -0.33 1; 0.33 1; 0.33 0; -0.33 0; -0.33 -1; 1 0; 0.33 -1];

arc=0; poz=0; orgraf=0; Fontsize=10; storis=2; spalva='b';
figure(1)
title('Duotasis grafas')
plotGraphVU1(V,U,orgraf,arc,Vkor,poz,Fontsize,storis,spalva)
GAM = UtoGAM(V,U,orgraf);
count = length(V);
atsMas = {};
arrayOfMatrix = {};
index = 1;
for o = 1:10
    for v = 1:count
        for i = 1:length(GAM)
            for j = 1:length(GAM{i})

                n = GAM{i}(1);
                GAM{i}(1) = GAM{i}(j);
                GAM{i}(j) = n;
                [struktura, matrix] = PaieskaIGyli(GAM, v, count);
                is = false;

                for z = 1:length(arrayOfMatrix)
                    if isequal(arrayOfMatrix{z}, matrix)
                        is = true;
                        break;
                    end
                end
                if is == false
```

```
                    arrayOfMatrix{index} = matrix;
                    atsMas{index} = struktura;
                    index = index + 1;
                end
            end
        end
    end
end

%Spausdinimas ? Command window
fprintf('Dalini? graf?, kurie yra medžiai skai?ius: %d\n\n', length(atsMas));
disp('Medži? briaun? aib?s:');
for i = 1:length(atsMas)
    for j = 1:length(atsMas{i})
        a = atsMas{i};
         fprintf('[%d %d]', a{j}(1), a{j}(2));
    end
    fprintf('\n');
end

%Pasirinkto medžio grafo spausdinimas (šiuo atv?ju - tai 20 medis)
figure(2)
title('Gautasis medžio grafas')
plotGraphVU1(V,atsMas{20},orgraf,arc,Vkor,poz,Fontsize,storis,spalva)

function [struktura, perMatr] = PaieskaIGyli(GAM, v, count)
pastIndexes = zeros(count);

atsMas = zeros(count,count);
placeOfPastIndex = 0;
goBackPerOne = 1;
counter = 0;

while(counter < count - 1)
    Ind = v;
    IsItIn = false;
    for i = 1:length(pastIndexes)
        if pastIndexes(i) == Ind
            IsItIn = true;
            break;
        end
    end

    if IsItIn == false
        placeOfPastIndex = placeOfPastIndex + 1;
        pastIndexes(placeOfPastIndex) = Ind;
    end
    for j = 1:length(GAM{Ind})
        Is = false;
        current = GAM{Ind}(j);
            for z = 1:length(pastIndexes)
                if  pastIndexes(z) == current
                    Is = true;
                    break;
                end
            end
            if Is == false
                atsMas(Ind, current) = 1;
                v = current;
                goBackPerOne = 1;
                counter = counter + 1;
                break;
            end
    end
```

```
    if v == Ind
        v = pastIndexes(placeOfPastIndex - goBackPerOne);
        goBackPerOne = goBackPerOne + 1;
    end
end

perMatr = zeros(count,count);
for i = 1:length(atsMas)
    for j = 1:length(atsMas(i,:))
        if atsMas(i, j) == 1
            atsMas(i, j) = 0;
            perMatr(i, j) = 1;
        end
    end
    atsMas = atsMas.';
    for j = 1:length(atsMas(i,:))
        if atsMas(i, j) == 1
            atsMas(i, j) = 0;
            perMatr(i, j) = 1;
        end
    end
    atsMas = atsMas.';
end

x1 = 1;
for i = 1:length(perMatr)
    for j = 1:length(perMatr(i,:))
            if perMatr(i, j) ~= 0
                struktura{x1}(1) = i;
                struktura{x1}(2) = j;
                x1 = x1 + 1;
            end

    end
end
end
```

plotGraphVU1.m

```
function plotGraphVU1(V,U,orgraf,arc,Vkor,poz,Fontsize,lstor,spalva)
% plotGraphVU1 funkcija piesia grafa,kai jis nusakomas virsuniu ir
% briaunu(lanku) aibe.
% Braizomas grafas, duotas virsuniu aibe V ir dvielemenciu poaibiu aibe U
%       Jeigu poaibis is 2 elementu, jis zymi briauna (lanka);
%       Jeigu poaibis is 3 elementu, jis zymi briauna (lanka) ir jo svori;
%
% orgraf  = 0 arba 1 - neorientuotas ar orientuotas grafas;
% arc = 0 arba 1 - tiesus ar apvalinti lankai;
% Vkor(1:nv,2) - grafo virsuniu koordinates plokstumoje.
%                nebutinas parametras,pagal nutylejima virsunes
%                isdestomos apskritimu. Jeigu ivestas Vkor=[], virsunes
%                isdestomos taip pat pagal nutylejima (t.y. apskritimu)
% poz =0, jei briaunos virsunes dazomos spalva, apibrezta pagal nutylejima,
%     =1, jei briaunos virsunes dazomos raudona spalva,
% Fontsize - srifto aukstis (pagal nutylejima 10), nebutinas parametras
% lstor - linijos storis, - nebutinas parametras (pagal nutylejima "1")
% spalva - linijos spalva (pagal nutylejima "b") nebutinas parametras
%*********************************************************************

hold on; axis equal; axis([-1.1,1.1,-1.1,1.1]);
s=size(V);nv=s(2);  % grafo eile
s=size(U);nb=s(2);  % briaunu(lanku) skaicius

dphi=2*pi/nv;   % virsuniu isdestymo kampinis zingsnis
```

```
R=1;            % virsuniu isdestymo apskritimo spindulys
r=0.08;         % virsunes spindulys

if nargin < 5 || isempty(Vkor)  % virsuniu koordinates isdestomos ratu
    for i=1:nv, Vkor(i,:)=[R*cos((i-1)*dphi),R*sin((i-1)*dphi)];  end
end

if nargin<6, Fontsize=10; storis=1; spalva='b'; end

% briaunu braizymas:
for i=1:nb      % ciklas per briaunas
  a=U{i};       % "i-os" briaunos virsuniu numeriai
  jr=a(1); % "i-os" briaunos pirmos virsunes numeris
  j=find(V==jr); % virsunes "jr" eiles numeris vektoriuje V
  jr1=a(2); % "i-os" briaunos antros virsunes numeris
  j1=find(V==jr1); % antros virsunes eiles numeris vektoriuje V
          x=Vkor(j,1);y=Vkor(j,2);          % briaunos pirmos virsunes koordinates
          x1=Vkor(j1,1);y1=Vkor(j1,2);      % briaunos antros virsunes koordinates
  L=norm([x1-x,y1-y]);              % briaunos ilgis

  n=[x1-x,y1-y]/L;                  % briaunos krypties vektorius
  nt=[n(2),-n(1)];                  % vienetinis vektorius, statmenas n
  v1=[x+r*n(1)+0.1*r*nt(1),y+r*n(2)+0.1*r*nt(2)];v2=[x1-r*n(1)+0.1*r*nt(1),y1-
r*n(2)+0.1*r*nt(2)];  % briaunos galu koordinates

  if ~arc   % tiesios briaunos braizymas
plot([v1(1),v2(1)],[v1(2),v2(2)],spalva,'LineWidth',lstor);
      if orgraf % rodykles braizymas
         ang=pi/10; % 1/2 rodykles kampo
         n1=[cos(ang),-sin(ang);sin(ang),cos(ang)]*n';
 plot([v2(1),v2(1)-1*n1(1)*r ],[v2(2),v2(2)-1*n1(2)*r],spalva,'LineWidth',lstor);
         ang=-pi/10;
         n1=[cos(ang),-sin(ang);sin(ang),cos(ang)]*n';
plot([v2(1),v2(1)-1*n1(1)*r ],[v2(2),v2(2)-1*n1(2)*r],spalva,'LineWidth',lstor );
      end
      s=size(a);sv=[];if s(2)>2, sv=a(3);end  % lanko svoris
              if ~isempty(sv)   % uzrasomas lanko svoris
                  v=v1+0.4*(v2-v1);ll=2.5*r;hh=1.5*r;
                  rectangle('Position',[v(1)-ll/2,v(2)-
hh/2,ll,hh],'Curvature',[0,0],'FaceColor',[1. 0.9 0.8]);
                  str=sprintf('%-6.2g',sv);
                  text(v(1)-ll/2+hh/4,v(2)-hh/4*0,str,'FontSize',Fontsize);
              end

   else     % briaunos lanko braizymas
      vm=v1+(v2-v1)/2+nt*r;
      x=[v1(1),vm(1),v2(1)]; y=[v1(2),vm(2),v2(2)]; t=0:2;
      tt=0:0.1:2; xx=spline(t,x,tt); yy=spline(t,y,tt);
      plot(xx,yy,spalva);
      if orgraf % rodykles braizymas
         ang=pi/10; % 1/2 rodykles kampo
         dang=2*atan(2*r/L); % rodykles kampo korekcija del lanko kreivumo
         n1=[cos(ang+dang),-sin(ang+dang);sin(ang+dang),cos(ang+dang)]*n';
 plot([v2(1),v2(1)-1*n1(1)*r ],[v2(2),v2(2)-1*n1(2)*r],spalva,'LineWidth',lstor);
         n1=[cos(-ang+dang),-sin(-ang+dang);sin(-ang+dang),cos(-ang+dang)]*n';
plot([v2(1),v2(1)-1*n1(1)*r ],[v2(2),v2(2)-1*n1(2)*r],spalva,'LineWidth',lstor);
      end
      s=size(a);sv=[];if s(2)>2, sv=a(3);end  % lanko svoris
              if ~isempty(sv)   % uzrasomas lanko svoris
                  v=v1+0.8*(vm-v1);ll=2.5*r;hh=1.5*r;
                  rectangle('Position',[v(1)-ll/2,v(2)-
hh/2,ll,hh],'Curvature',[0,0],'FaceColor',[1. 0.9 0.8]);
                  str=sprintf('%-6.2g',sv);
                  text(v(1)-ll/2+hh/4,v(2)-hh/4*0,str,'FontSize',Fontsize);
```

```
            end
    end

end

% virsuniu braizymas:

for i=1:nv      % ciklas per virsunes
    x=Vkor(i,1);y=Vkor(i,2); % V(i) virsunes koordinates
 ccc=[0.8 0.9 1];if poz==1, ccc='r'; end
    rectangle('Position',[x-r,y-r,2*r,2*r],'Curvature',[1,1],'FaceColor',ccc);
          % uzrasomas virsunes numeris:
          if abs(V(i))<10,str=sprintf('%d',abs(V(i)));shiftx=0.2*r;
          elseif abs(V(i))<100,str=sprintf('%2d',abs(V(i)));shiftx=0.4*r;
          else, str=sprintf('%3d',abs(V(i)));shiftx=0.6*r;
          end
          text(x-shiftx,y,str); %
end
return
```

UtoGAM.m

```
function GAM = UtoGAM(V,U,orgrafGAM)
% GAM funkcija perskaiciuoja grafo G=(V,U),nusakyto virsuniu V ir
% briaunu U aibemis, uzrasa i gretimumo struktura GAM.
% Pavyzdziui, tegu grafas nusakytas virsuniu aibe V=[1,2,3,4,5] ir briaunu
% aibe U={[1,2],[1,3],[1,5],[2,3],[2,4],[3,4],[3,5],[4,5]}.Tada funkcija
% UtoGAM apskaiciuoja gretimumo struktura
% GAM={[2,3,5],[1,3,4],[1,2,4,5],[2,3,5],[1,3,4]}
% "i-tasis" GAM narys GAM{i} yra vienmatis masyvas, kuriame surasytos
%  virsunes, gretimos virsunei "i"
%
% Formalus parametrai ------------------------
% V - vienmatis masyvas,- grafo virsuniu aibe,
% U - grafo briaunu (lanku) masyvas (zr. pavyzdi),
% orgrafGAM == 0 - jei grafas neorientuotasis,
% orgrafGAM == 1 - jei grafas orientuotasis.

s = size(V);  nv = s(2);  % grafo eile
s = size(U);  nb = s(2);  % lanku skaicius
GAM = cell(nv);
for i = 1:nv
    nvr = abs(V(i));
    for j = 1:nb
        a1 = U{j};  a = a1(1:2);
        s = size(a1); if s(2)==3, sv = a1(3); else, sv = [];  end %jei grafas
svorinis sv briaunos svoris, jei ne tuscia
        if ~orgrafGAM  %jei 1 - orgrafas
            ind = (abs(a)==nvr);
            if sum(ind), GAM{i} = [GAM{i},[a(find(~ind));sv]];  end
        else %jei 0 - neorgrafas
            if abs(a(1))==nvr, GAM{i} = [GAM{i},[a(2);sv]];  end
        end
    end
end
return
```

5. Testavimo pavyzdžiai

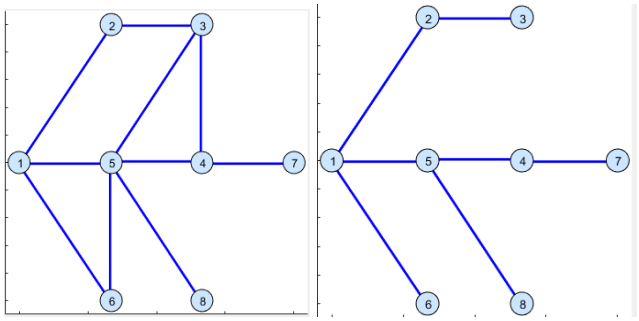
Buvo panaudoti trys testavimo pavyzdžiai:

Pirmas testas

Pirmojo pavyzdžio pradinis grafas ir vienas iš daugelio medžio grafų pavaizduoti 2 pav., kai duotos viršūnių ir briaunų aibės:

```
V=[1 2 3 4 5 6 7 8];
U={[1 2],[1 5],[1 6],[2 3],[3 4],[3 5],[4 5],[4 7],[5 6],[5 8]};
```

Programos rezultatas:



2 pav. Pradinis grafas ir medžio grafas

```
Dalinių grafų, kurie yra medžiai skaičius: 28


Medžių briaunų aibės:

[1 2][2 3][3 4][4 5][4 7][5 6][5 8]

[1 5][2 3][3 4][3 5][4 7][5 6][5 8]

[1 6][2 3][3 4][3 5][4 7][5 6][5 8]

[1 6][2 3][3 4][4 5][4 7][5 6][5 8]

[1 5][1 6][2 3][3 5][4 5][4 7][5 8]

[1 2][1 5][3 4][3 5][4 7][5 6][5 8]

[1 2][1 5][3 4][4 5][4 7][5 6][5 8]

[1 2][1 6][3 4][4 5][4 7][5 6][5 8]

[1 2][1 6][3 5][4 5][4 7][5 6][5 8]

[1 2][1 5][2 3][4 5][4 7][5 6][5 8]

[1 2][1 5][2 3][3 4][4 7][5 6][5 8]

[1 2][1 6][2 3][3 4][4 7][5 6][5 8]

[1 2][1 6][3 4][3 5][4 7][5 6][5 8]

[1 2][1 6][2 3][4 5][4 7][5 6][5 8]

[1 2][1 5][1 6][3 4][3 5][4 7][5 8]

[1 2][1 6][2 3][3 4][3 5][4 7][5 8]

[1 2][1 6][2 3][3 4][4 5][4 7][5 8]
```

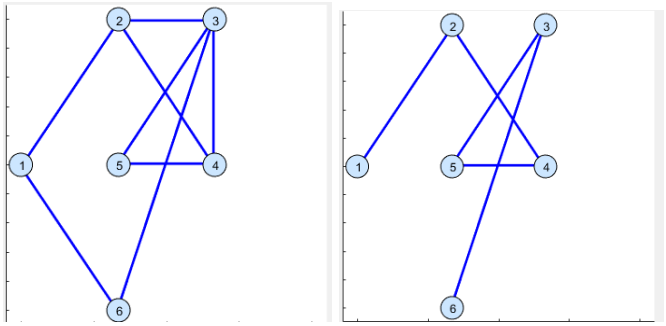
```
[1 2][1 5][1 6][2 3][3 4][4 7][5 8]

[1 2][2 3][3 4][3 5][4 7][5 6][5 8]

[1 2][1 5][1 6][2 3][4 5][4 7][5 8]

[1 5][2 3][3 4][4 5][4 7][5 6][5 8]

[1 2][2 3][3 5][4 5][4 7][5 6][5 8]

[1 2][1 5][1 6][3 5][4 5][4 7][5 8]

[1 2][1 5][1 6][3 4][4 5][4 7][5 8]

[1 5][1 6][2 3][3 4][3 5][4 7][5 8]

[1 2][1 6][2 3][3 5][4 5][4 7][5 8]

[1 5][1 6][2 3][3 4][4 5][4 7][5 8]

[1 6][2 3][3 5][4 5][4 7][5 6][5 8]
```

Antras testas

Antrojo pavyzdžio pradinis grafas ir vienas iš daugelio medžio grafų pavaizduoti 3 pav., kai duotos viršūnių ir briaunų aibės:

```
V=[1 2 3 4 5 6];
U={[1 2],[1 6],[2 3],[2 4],[3 4],[3 5],[3 6],[4 5]};
```

Programos rezultatas:



3 pav. Pradinis grafas ir medžio grafas

```
Dalinių grafų, kurie yra medžiai skaičius: 24


Medžių briaunų aibės:

[1 2][2 3][3 4][3 6][4 5]

[1 6][2 3][2 4][3 6][4 5]

[1 6][2 4][3 4][3 6][4 5]

[1 6][2 4][3 5][3 6][4 5]

[1 2][1 6][3 4][3 6][4 5]

[1 6][2 3][3 4][3 6][4 5]

[1 6][2 3][3 5][3 6][4 5]
```

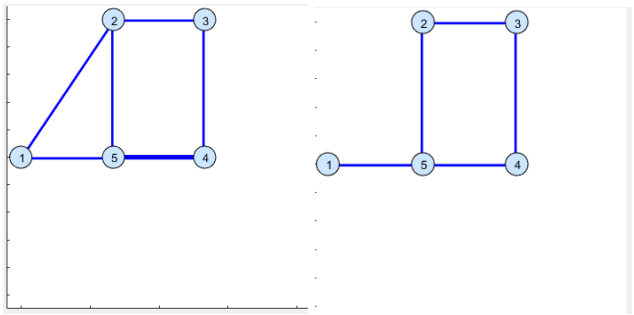
```
[1 2][1 6][2 4][3 5][4 5]

[1 2][1 6][2 4][3 6][4 5]

[1 2][1 6][2 3][2 4][4 5]

[1 2][1 6][2 4][3 4][4 5]

[1 2][1 6][2 4][3 5][3 6]

[1 6][2 3][2 4][3 5][3 6]

[1 2][1 6][2 3][3 4][3 5]

[1 2][1 6][2 3][3 5][4 5]

[1 2][1 6][2 3][2 4][3 5]

[1 2][1 6][2 4][3 4][3 5]

[1 2][2 3][2 4][3 6][4 5]

[1 2][2 4][3 4][3 6][4 5]

[1 2][2 4][3 5][3 6][4 5]

[1 2][2 3][3 5][3 6][4 5]

[1 2][1 6][3 4][3 5][3 6]

[1 2][1 6][3 5][3 6][4 5]

[1 2][1 6][2 3][3 4][4 5]
```

Trečias testas

Trečiojo pavyzdžio pradinis grafas ir vienas iš daugelio medžio grafų pavaizduoti 4 pav., kai duotos viršūnių ir briaunų aibės:

```
V=[1 2 3 4 5];
U={[1 2],[1 4],[2 3],[2 5],[3 4],[4 5]};
```

Programos rezultatas:



4 pav. Pradinis grafas ir medžio grafas

```
Dalinių grafų, kurie yra medžiai skaičius: 12


Medžių briaunų aibės:

[1 2][2 3][3 4][4 5]

[1 4][2 3][2 5][3 4]
```

```
[1 4][2 3][2 5][4 5]
[1 4][2 5][3 4][4 5]
[1 2][1 4][3 4][4 5]
[1 4][2 3][3 4][4 5]
[1 2][1 4][2 3][4 5]
[1 2][2 5][3 4][4 5]
[1 2][1 4][2 5][3 4]
[1 2][1 4][2 3][2 5]
[1 2][2 3][2 5][3 4]
[1 2][2 3][2 5][4 5]
```

6. Išvados

Ši programa įgyvendina paieškos į gylį algoritmą teisingai, kuris geba rasti dalinį grafą, kuris yra medis. Bandydami rasti visas įmanomas gretimumo struktūras, keičiant viršūnių išsidėstymo tvarką galima surasti visus egzistuojančius grafo medžius. Vis dėlto šioje programoje nepavyksta sugeneruoti visų įmanomų unikalių gretimumo struktūrų, todėl yra randami ne visi daliniai grafai, kurie yra medžiai. Tik iš mažesnių ir mažiau sudėtingų grafų galima gauti visus medžius, nes yra didesnė tikimybė juos rasti, keičiant gretimumo struktūrą. Iš testavimo pavyzdžių tik trečiasis testas pasiteisino – buvo rasti visi medžiai duotame grafe. Ši programa gana lėta, nes yra atliekamas paieškos į gylį algoritmas su visomis gretimumo struktūromis, nors kai kurios iš jų ir nesugeneruoja unikalaus medžio – randami jau seniau rasti medžiai (pateiktuose atsakymuose medžiai nesidubliuoja).

7. Literatūros sąrašas

1. Grafo dalinių grafų, kurie yra medžiai skaičiuotuvas https://www.emathhelp.net/calculators/linear-algebra/cofactor-matrix-calculator/ (žiūrėta 2022-11-22)
2. „Diskrečiųjų struktūrų“ modulis „Moodle“ aplinkoje https://moodle.ktu.edu/course/view.php?id=39 (žiūrėta 2022-11-22)