**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

**DISKREČIOSIOS STRUKTŪROS (P170B008)**

**KURSINIS DARBAS**

**Užduoties nr. B23**

Atliko: IFF–5/4 gr. studentas Mantas Damijonaitis

Priemė dėst. Martynas Patašius

KAUNAS

2016

1. **Užduotis (nr. B23)**

Sudaryti algoritmą ir programą, nustatančią, ar duotas orgrafas yra stipriai jungus, vienakryptiškai jungus, silpnai jungus.

1. **Užduoties analizė**

Remiantis trečiosios laboratorinių darbų temos skaidrėmis, orientuotas grafas yra stipriai jungus tada, jei bet kokios dvi viršūnės x ir y yra pasiekiamos viena iš kitos, vienakryptiškai jungus, jei bet kokiai porai viršūnių x ir y jos yra pasiekiamos bent viena kryptimi, ir silpnai jungus, jei yra jungus neorientuotasis grafas gautasis iš orientuotojo, pakeitus lankus briaunomis. Taip pat žemiau buvo pateiktos tokios iliustracijos:



Žvelgiant į šias iliustracijas, grafo tipą atskirti nesunku, tačiau teiginiai man pasirodė neaiškūs, ir nepakankami sugalvoti algoritmo idėjai. Tačiau paieškojęs ilgiau, Aleksandro Krylovo vadovėlyje „Diskrečioji matematika“ radau tokias teoremas:

1. „Grafas yra stiprusis tada ir tik tada, kai egzistuoja einantis per visas jo viršūnės ciklas“.
2. „Grafas yra vienakryptiškai stiprusis tada ir tik tada, kai jis turi einantį per visas viršūnes maršrutą“.
3. „Grafas yra silpnasis tada ir tik tada, kai per visas jo viršunes eina pusmaršutis“.

Remdamasis šiomis teoremomis, realizavau grafo stiprumo nustatymą. Algoritmas veikia tokiu principu:

1. Turime dvi matricas – gretimumo, kurioje apibūdintas turimas grafas, bei kelių matrica. Iteruodami per gretimumo matricos elementus, kuriame kelių matricą. Kiekvienas naujas atrastas kelias įrašomas į kelių matricos eilutę. Iteruojama tol, kol išanalizuojamos visos grafo briaunos.
2. Iteruojame per sudarytą kelių matricą. Tikriname, ar šioje matricoje egzistuoja eilutė, kurioje paminėtos visos grafo viršūnės, o eilutė prasideda ir baigiasi 1. Jei taip – šis grafas turi ciklą. Taip pat ieškome tokios eilutės, kurioje yra visos grafo viršūnės.
3. Iš gautų rezultatų darome išvadas. Jei buvo tokia eilutė, kurioje buvo paminėti visi elementai, ji prasidėjo ir baigėsi 1, be to, taip pat egzistuoja eilutė, kurioje buvo paminėti visi elementai, pagal pirmą teoremą galime teigti, kad grafas yra stiprusis.
4. Jei grafas neturi ciklo, tačiau turi eilutę, kurioje paminėti visi elementai, reiškia, kad grafas turi per visas viršunes einantį maršrutą. Tada, pagal antra teoremą, jis – vienakryptiškai jungus.
5. O jei grafas neatitinka nei trečiame, nei ketvirtame punkte paminėtų salygų – jis turi tik pusmaršrutį, todėl pagal trečią teoremą galime teigti, kad jis silpnai jungus.

**3. Programos tekstas**

diskreciuNd.m

clc

%graphMatrix = [0 1 1 0; 0 0 1 0; 0 0 0 1; 1 0 0 0]; %STIPRIAI JUNGI% %

%graphMatrix = [0 1 1 1; 0 0 1 0; 0 0 0 1; 0 0 0 0]; %VIENAKRYPTISKAI JUNGI% %

%graphMatrix = [0 1 1 1; 0 0 1 0; 0 0 0 0; 0 0 1 0]; %SILPNAI JUNGUS% %

%graphMatrix = [0 1 1 1 0 0 0; ... % SUDETINGA STIPRIAI JUNGI% %

%0 0 1 0 0 0 0; ...

%0 0 0 1 0 0 0; ...

%0 0 0 0 1 1 0; ...

%0 1 0 0 0 1 0; ...

%0 0 0 0 0 0 1; ...

%1 0 0 0 0 0 0];

graphMatrix = [0 1 1 1 0 0 1; ... % SUDETINGA VIENAKRYPTISKAI JUNGI% %

0 0 1 0 0 0 0; ...

0 0 0 1 0 0 0; ...

0 0 0 0 1 1 0; ...

0 1 0 0 0 1 0; ...

0 0 0 0 0 0 1; ...

0 0 0 0 0 0 0];

%graphMatrix = [0 1 1 1 0 0 1; ... % SUDETINGA SILPNAI JUNGI% %

% 0 0 1 0 0 0 0; ...

% 0 0 0 1 0 0 0; ...

% 0 0 0 0 1 1 0; ...

% 0 1 0 0 0 1 0; ...

% 0 0 0 0 0 0 0; ...

% 0 0 0 0 0 1 0];

drawableMatrix = convertMatrixToDrawable(graphMatrix);

Vkor = [];

V = 1:7;

orgraf = 1;

arc=0; poz=0; Fontsize=10; lstor=1; spalva='b';

%figure(1)

matchMatrix = setMatchMatrix(graphMatrix);

result = analyzeMatchMatrix(matchMatrix);

title(result);

plotGraphVU1(V,drawableMatrix,orgraf,arc,Vkor,poz,Fontsize,lstor,spalva);

hold on;

convertMatrixToDrawable.m

function y = convertMatrixToDrawable(graphMatrix)

returnMatrix = [,];

[rows\_amount, columns\_amount] = size (graphMatrix);

for rows\_index = 1 : rows\_amount

for columns\_index = 1 : columns\_amount

if graphMatrix(rows\_index, columns\_index) == 1

tempSet(1,1) = rows\_index;

tempSet(1,2) = columns\_index;

tempVar = [rows\_index, columns\_index]

returnMatrix = horzcat(returnMatrix, {tempVar});

end;

end;

end;

%returnMatrix

%[rows\_amount, columns\_amount] = size(returnMatrix);

%for rows\_index = 1 : rows\_amount

% returnMatrix(rows\_index, 3) = 1;

%end;

y = returnMatrix

end

plotGraphVU1.m (biblioteka nekeista)

setMatchMatrix.m

function y = setMatchMatrix (graph)

[rows\_amount,columns\_amount] = size(graph)

matchMatrix = zeros (1, columns\_amount + 1);

for rows\_index = 1 : rows\_amount

for columns\_index = 1 : columns\_amount

if (graph(rows\_index, columns\_index) == 1)

matchMatrix = appendToMatchMatrix(matchMatrix, rows\_index, columns\_index);

end;

end;

end;

y = matchMatrix;

end

analyzeMatchMatrix.m

function y = analyzeMatchMatrix(matchMatrix)

connectedOneSide = 0;

loopDetected = 0;

[row\_amount, column\_amount] = size (matchMatrix);

for rows\_index = 1 : row\_amount

oneSideChainAmount = 0;

for columns\_index = 1 : column\_amount

if matchMatrix(rows\_index, columns\_index) == columns\_index

oneSideChainAmount = oneSideChainAmount + 1;

if (oneSideChainAmount == column\_amount - 1)

connectedOneSide = 1;

end;

end;

end

if matchMatrix(rows\_index, 1) == 1 && matchMatrix(rows\_index, column\_amount) == 1

loopDetected = 1;

end;

end;

y = '';

if (loopDetected == 1 && connectedOneSide == 1)

y = 'Grafas yra stipriai jungus';

end;

if (loopDetected == 0 && connectedOneSide == 1)

y = 'Grafas yra vienakryptiškai jungus';

end;

if (loopDetected == 0 && connectedOneSide == 0)

y = 'Grafas yra silpnai jungus';

end;

end

appendToMatchMatrix.m

function y = appendToMatchMatrix(matchMatrix, from, to)

[rowsAmount, columnsAmount] = size (matchMatrix);

wasInserted = 0;

for rows\_index = 1 : rowsAmount

for columns\_index = 1 : columnsAmount

if (columns\_index ~= 1 && ...

matchMatrix(rows\_index, columns\_index - 1) ~= 0) && ...

(matchMatrix(rows\_index, columns\_index - 1) == from) && ...

(matchMatrix(rows\_index, columns\_index) == 0)

tempMatrix = matchMatrix (rows\_index,:);

tempMatrix (1, columns\_index) = to;

matchMatrix = [matchMatrix; tempMatrix];

wasInserted = 1;

end;

end;

end;

if (wasInserted == 0)

for rows\_index = 1 : rowsAmount

if (matchMatrix(rows\_index, 1) == 0 && wasInserted == 0)

tempMatrix = zeros(1, columnsAmount);

tempMatrix (1, 1) = from;

tempMatrix (1, 2) = to;

matchMatrix = [matchMatrix; tempMatrix];

wasInserted = 1;

end;

end;

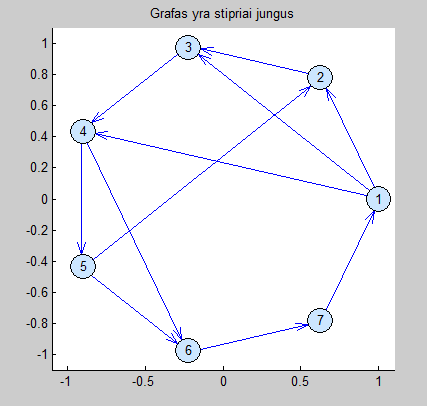
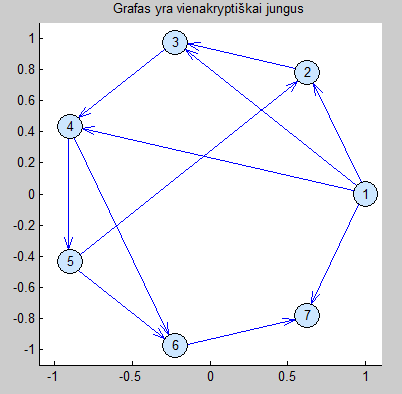
end;

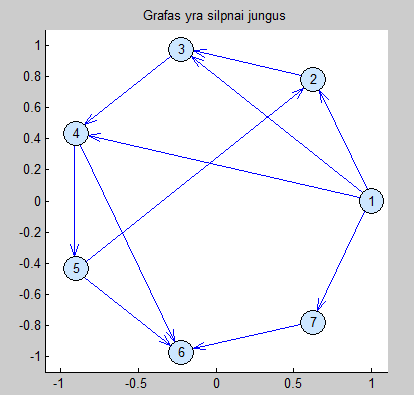
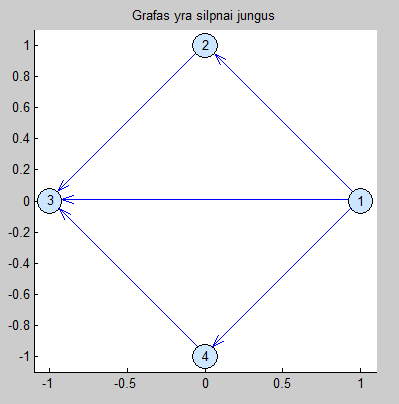
y = matchMatrix;

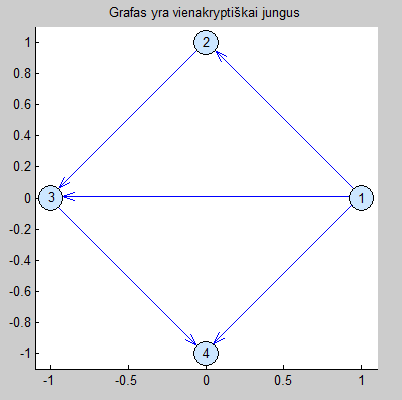
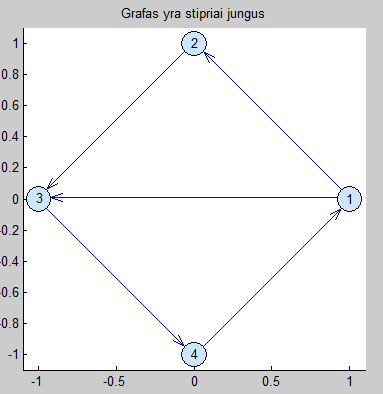
end

# 4. Testiniai pavyzdžiai

Buvo panaudoti šeši testiniai pavyzdžiai:

**5. Išvados**

Sudaryta programa veikia teisingai. Su pateiktais testiniais grafais rezultatus apskaičiuoja akimirksniu. Išmokau MATLAB pradmenų, ir praplėčiau žinias grafų teorijoje. Atlikti šią užduotį užtrukau apie penkias dienas.

**6. Literatūrinis sąrašas**

1. “Diskrečioji matematika” – Aleksandras Krylovas

http://techmat.vgtu.lt/~akrl/Medziaga/Konspektai/DiskrMatematVadovelis/AKrylovas\_DiskrMatem\_20080316.pdf

1. „Diskrečiųjų struktūrų“ modulis „Moodle“ aplinkoje https://moodle.ktu.edu/course/view.php?id=763 (žiūrėta 2016-11-22)
2. Matlab dokumentacija <http://www.mathworks.se/help/index.html> (žiūrėta 2016-11-22)