**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №5**

з дисципліни  
«Дискретні структури»

на тему «Мінімальний кістяк графа»

Виконала: Перевірила:

студентка групи ІП-93 Сергієнко А. А.  
Узунлу Ґамзенур   
номер залікової книжки: 9882

Київ 2020

**Завдання на лабораторну роботу:**

1. Представити зважений ненапрямлений граф із заданими параметрами

так само, як у лабораторній роботі №1. Відміна: матриця А за варіантом

формується за командами Scilab:

rand("seed", п1п2п3п4);

T = rand(n,n) + rand(n,n);

A = floor((1.0 - п3\*0.01 - п4\*0.005 - 0.05)\*T)

Матриця ваг W формується за наступними командами:

Wt = round(rand(n,n)\*100 .\* A);

B = Wt & ones(n,n);

Wt = (bool2s(B & ~B') + bool2s(B & B') .\* tril(ones(n,n),-1)) .\* Wt;

W = Wt + Wt';

2. Створити скрипт для Scilab для знаходження мінімального кістяка за

алгоритмом Краскала при п4 — парному і за алгоритмом Пріма — при

непарному. При цьому у скрипті:

— встановити функцію halt у точці додавання чергового ребра до кістяка,

— виводити зображення графа у графічному вікні перед кожною

зупинкою по функції halt.

3. Під час обходу графа побудувати дерево його кістяка. Вивести

побудоване дерево у графічному вікні. При зображенні як графа, так і його

кістяка, вказати ваги ребер.

Варіант 9882:  
*п =* 11  
розміщення вершин: колом

**Текст програми для Scilab**

**clf; //clearing canvas**

**clear;**

**TypeOfGraph = 0; //0 is for non-oriented,1 for oriented**

**N1 = 9; //numbers of My Zalikovka to make figure**

**N2 = 8;**

**N3 = 8;**

**N4 = 2;**

**N = 10 + N3; //number of vertexes**

**//preparing for making canvas**

**//making values for plot2d**

**//x on left is 0**

**ForPlot\_X1 = 0;**

**//x on right is 70**

**ForPlot\_X2 = 70;**

**//y on down is 0**

**ForPlot\_Y1 = 0;**

**//y on upper is 70**

**ForPlot\_Y2 = 70;**

**RadiusOfCircle = 5; //Woah,its radius of circle**

**DiameterOfCircle = 2\*RadiusOfCircle; //Woah,its diameter of circle**

**TopLimit = 999;**

**S=2;**

**ColorOfArrow\_1 = 14; //color of arrow\_1 //1 is good**

**ColorOfArrow\_2 = 9; //color of arrow\_2 //3,**

**N1N2N3N4 = N1\*1000 + N2\*100 + N3\*10 + N4**

**//generating matrix for or-graph(from the konspekt)**

**rand("seed", N1N2N3N4);**

**T = rand(N,N) + rand(N,N);**

**A\_oriented = floor((1.0 - N3\*0.01 - N4\*0.005 - 0.05)\*T)**

**//generating matrix for non\_or-graph**

**At=A\_oriented'; //transpose matrix**

**A\_non\_oriented=At+A\_oriented; //final non\_oriented matrix(adding regular matrix and transpose)**

**disp(A\_non\_oriented)**

**//loop,where we changing 2,which we get by adding**

**//two matrixes,for every element,that = 2**

**for v=1:1:N**

**for v1=1:1:N**

**if A\_non\_oriented(v,v1) == 2**

**A\_non\_oriented(v,v1)=1**

**end**

**end**

**end**

**//choosing matrix and width of arrows**

**if TypeOfGraph == 0**

**Matrix = A\_non\_oriented;**

**WidthOfArrow = 0;**

**elseif TypeOfGraph == 1**

**Matrix = A\_oriented;**

**WidthOfArrow = 30;**

**end**

**//making canvas with our varibles**

**//plot2d([0,200],[0,100],[-1,-1],"022") — формує**

**//прямокутне поле у графічному вікні розміром 100х200 одиниць.**

**//xsetech-ділить вікно на області**

**xsetech([0,0,0.5,1]);**

**plot2d([ForPlot\_X1;ForPlot\_X2],[ForPlot\_Y1;ForPlot\_Y2],0);**

**//looking for a center**

**//(100-0)/2=50**

**X\_O = (ForPlot\_X2-ForPlot\_X1)/2;**

**Y\_O = (ForPlot\_Y2-ForPlot\_Y1)/2;**

**//radius of the big circle,on which we will place our vertexes**

**BigRadius = 2/5\*sqrt((ForPlot\_X2-ForPlot\_X1)\*(ForPlot\_Y2-ForPlot\_Y1))**

**Wt = round(rand(N,N)\*100.\* Matrix);**

**disp(Wt)**

**B = Wt & ones(N,N);**

**disp(B)**

**Wt = (bool2s(B & ~B') + bool2s(B & B').\* tril(ones(N,N),-1)).\* Wt;**

**disp(Wt)**

**W = Wt + Wt';**

**function [MinimalIndicator]=MinimalConf(Lead,Congifure)**

**Minimal=TopLimit;**

**MinimalIndicator=-1;**

**for x=1:1:N**

**if Congifure(x)==0 && Lead(x)<Minimal**

**Minimal=Lead(x);**

**MinimalIndicator=x;**

**end**

**end**

**endfunction**

**Prim\_Matrix=zeros(N,N);**

**for x=1:1:N//making operations for each element**

**X\_Position = cos(2 \* %pi \* x / N) \* BigRadius + X\_O;//calculating coordinates**

**Y\_Position = sin(2 \* %pi \* x / N) \* BigRadius + Y\_O;**

**//drawing vertexes**

**// xarc(x,y,w,h,a1,a2) — малювання дуги еліпса, який вписаний у прямокутник**

**//з координатами (х,у) верхнього лівого кута і шириною w та висотою h,**

**//яка обмежена променями з кутами а1 і а2, які задаються у 64-х частках градуса.**

**//Наприклад, коло діаметром 4 одиниць з центром у точці (10,20) задається**

**//функцією xarc(8,22,4,4,0,360\*64);**

**xarc(X\_Position,Y\_Position,DiameterOfCircle,DiameterOfCircle,0,360\*64);**

**// xnumb(x,y,data) — вивід числа data починаючи з позиції з координатами**

**//(х,у).**

**xnumb(X\_Position+RadiusOfCircle,Y\_Position-RadiusOfCircle,x);**

**end**

**for x=1:1:N**

**Root(x)=0;**

**Lead(x)=TopLimit;**

**Congifure(x)=0;**

**end**

**for i=1:1:N**

**for j=1:1:N**

**if Matrix(i,j) == 1;**

**Position\_X\_I = cos(2 \* %pi \* i / N) \* (BigRadius-RadiusOfCircle) + X\_O;**

**Position\_Y\_I = sin(2 \* %pi \* i / N) \* (BigRadius-RadiusOfCircle) + Y\_O;**

**Position\_X\_J = cos(2 \* %pi \* j / N) \* (BigRadius-RadiusOfCircle) + X\_O;**

**Position\_Y\_J = sin(2 \* %pi \* j / N) \* (BigRadius-RadiusOfCircle) + Y\_O;**

**if i == j**

**Position\_X\_I = cos(2 \* %pi \* i / N) \* (BigRadius) + X\_O;**

**Position\_Y\_I = sin(2 \* %pi \* i / N) \* (BigRadius) + Y\_O;**

**Position\_X\_J = cos(2 \* %pi \* j / N) \* (BigRadius) + X\_O;**

**Position\_Y\_J = sin(2 \* %pi \* j / N) \* (BigRadius) + Y\_O;**

**xarc(Position\_X\_I-RadiusOfCircle,Position\_Y\_I+RadiusOfCircle\*3/4,3/2\*DiameterOfCircle,3/2\*DiameterOfCircle,0,190\*64);**

**//xarrows([x1;x2], [y1;y2], w, c) — стрілка з координатами початку**

**//(x1;y1), кінця (x2;y2), розмірами вістря w і кольором с (с=1 –чорний, 2– синій і**

**//т.д.).**

**xarrows([Position\_X\_I-RadiusOfCircle; Position\_X\_J],[Position\_Y\_I-RadiusOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, 1);**

**else**

**xarrows([Position\_X\_I+RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_1);**

**CentrePosition\_X = (Position\_X\_I+RadiusOfCircle+Position\_X\_J+RadiusOfCircle)/2;**

**CentrePosition\_Y = (Position\_Y\_I+RadiusOfCircle+Position\_Y\_J+RadiusOfCircle)/2;**

**FirstCoord=((Position\_X\_I+RadiusOfCircle)+(Position\_X\_J+RadiusOfCircle))/2;**

**SecondCoord=((Position\_Y\_I-RadiusOfCircle)+(Position\_Y\_J-RadiusOfCircle))/2;**

**xnumb(FirstCoord,SecondCoord,W(i,j));**

**end**

**end**

**end**

**end**

**Root(1)=-1;**

**Lead(1)=0;**

**xsetech([0.5,0,0.5,1]);**

**plot2d([ForPlot\_X1;ForPlot\_X2],[ForPlot\_Y1;ForPlot\_Y2],0);**

**//looking for a center**

**//(100-0)/2=50**

**X\_O = (ForPlot\_X2-ForPlot\_X1)/2;**

**Y\_O = (ForPlot\_Y2-ForPlot\_Y1)/2;**

**//radius of the big circle,on which we will place our vertexes**

**BigRadius = 2/5\*sqrt((ForPlot\_X2-ForPlot\_X1)\*(ForPlot\_Y2-ForPlot\_Y1))**

**for Tick=1:1:N-1**

**L=MinimalConf(Lead,Congifure);**

**Congifure(L)=1;**

**for P=1:1:N**

**if W(L,P)~=0 && Congifure(P)==0 && W(L,P)<Lead(P)**

**Root(P)=L;**

**Lead(P)=W(L,P);**

**end**

**end**

**end**

**for x=1:1:N//making operations for each element**

**X\_Position = cos(2 \* %pi \* x / N) \* BigRadius + X\_O;//calculating coordinates**

**Y\_Position = sin(2 \* %pi \* x / N) \* BigRadius + Y\_O;**

**//drawing vertexes**

**// xarc(x,y,w,h,a1,a2) — малювання дуги еліпса, який вписаний у прямокутник**

**//з координатами (х,у) верхнього лівого кута і шириною w та висотою h,**

**//яка обмежена променями з кутами а1 і а2, які задаються у 64-х частках градуса.**

**//Наприклад, коло діаметром 4 одиниць з центром у точці (10,20) задається**

**//функцією xarc(8,22,4,4,0,360\*64);**

**xarc(X\_Position,Y\_Position,DiameterOfCircle,DiameterOfCircle,0,360\*64);**

**// xnumb(x,y,data) — вивід числа data починаючи з позиції з координатами**

**//(х,у).**

**xnumb(X\_Position+RadiusOfCircle,Y\_Position-RadiusOfCircle,x);**

**end**

**disp("This is table of weights and arrows")**

**for y=S:1:N**

**disp(string(W(y,Root(y)))+'(Weight):'+string(y)+'(First Vertex):'+string(Root(y))+'(Second Vertex)');**

**M=W(y,Root(y));**

**Prim\_Matrix(y,Root(y))=M;**

**Prim\_Matrix(Root(y),y)=M;**

**end**

**disp("This is the "+char(39)+"кістяк"+char(39)+" of graph",Prim\_Matrix)**

**disp("This is an adjacency matrix of oriented graph",A\_oriented)**

**disp("This is an adjacency matrix of non oriented graph",A\_non\_oriented)**

**disp("This is a weight matrix",W)**

**for i=1:1:N**

**for j=1:1:N**

**if Prim\_Matrix(i,j) > 0 || Prim\_Matrix(i,j) < 0**

**X\_Position = cos(2 \* %pi \* x / N) \* BigRadius + X\_O;//calculating coordinates**

**Y\_Position = sin(2 \* %pi \* x / N) \* BigRadius + Y\_O;**

**Position\_X\_I = cos(2 \* %pi \* i / N) \* (BigRadius-RadiusOfCircle) + X\_O;**

**Position\_Y\_I = sin(2 \* %pi \* i / N) \* (BigRadius-RadiusOfCircle) + Y\_O;**

**Position\_X\_J = cos(2 \* %pi \* j / N) \* (BigRadius-RadiusOfCircle) + X\_O;**

**Position\_Y\_J = sin(2 \* %pi \* j / N) \* (BigRadius-RadiusOfCircle) + Y\_O;**

**halt**

**xarrows([Position\_X\_I+RadiusOfCircle; Position\_X\_J+RadiusOfCircle],[Position\_Y\_I-RadiusOfCircle; Position\_Y\_J-RadiusOfCircle], WidthOfArrow, ColorOfArrow\_1);**

**CentrePosition\_X = (Position\_X\_I+RadiusOfCircle+Position\_X\_J+RadiusOfCircle)/2;**

**CentrePosition\_Y = (Position\_Y\_I-RadiusOfCircle+Position\_Y\_J)/2;**

**FirstCoord=((Position\_X\_I+RadiusOfCircle)+(Position\_X\_J+RadiusOfCircle))/2;**

**SecondCoord=((Position\_Y\_I-RadiusOfCircle)+(Position\_Y\_J-RadiusOfCircle))/2;**

**xnumb(FirstCoord,SecondCoord,W(i,j));**

**end**

**end**

**end**

**Згенерована матриця суміжності орграфу:**

1. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0.

0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1.

0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 1.

0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1.

0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0.

1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0.

0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0.

0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0.

1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.

1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1.

1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0.

**Згенерована матриця суміжності неорграфу:**

1. 0. 1. 0. 1. 1. 1. 0. 1. 1. 1.

0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1.

1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1.

0. 1. 1. 0. 1. 1. 0. 1. 0. 1. 1.

1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0.

1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0.

1. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0.

0. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0.

1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0.

1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1.

1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0.

**Матриця знайденого кістяка графу:**

0. 0. 0. 0. 15. 3. 14. 0. 2. 0. 0.

0. 0. 0. 0. 0. 0. 30. 9. 0. 30. 44.

0. 0. 0. 0. 0. 0. 0. 0. 17. 0. 0.

0. 0. 0. 0. 0. 0. 0. 32. 0. 0. 0.

15. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

14. 30. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 9. 0. 32. 0. 0. 0. 0. 0. 0. 0.

2. 0. 17. 0. 0. 0. 0. 0. 0. 0. 0.

0. 30. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 44. 0. 0. 0. 0. 0. 0. 0. 0. 0.

**Матриця ваг:**

0. 0. 21. 0. 15. 3. 14. 0. 2. 99. 61.

0. 0. 43. 49. 74. 68. 30. 9. 0. 30. 44.

21. 43. 0. 52. 66. 48. 0. 60. 17. 63. 73.

0. 49. 52. 0. 95. 62. 0. 32. 0. 93. 82.

15. 74. 66. 95. 0. 58. 74. 86. 68. 83. 0.

3. 68. 48. 62. 58. 0. 0. 74. 0. 59. 0.

14. 30. 0. 0. 74. 0. 0. 0. 55. 0. 0.

0. 9. 60. 32. 86. 74. 0. 0. 0. 80. 0.

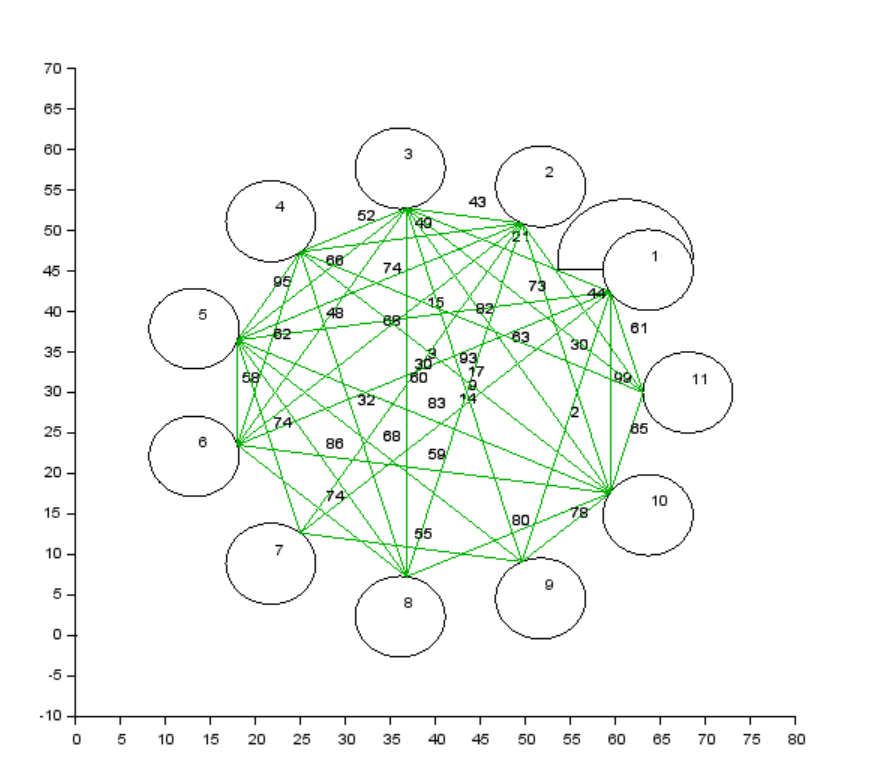
2. 0. 17. 0. 68. 0. 55. 0. 0. 78. 0.

99. 30. 63. 93. 83. 59. 0. 80. 78. 0. 65.

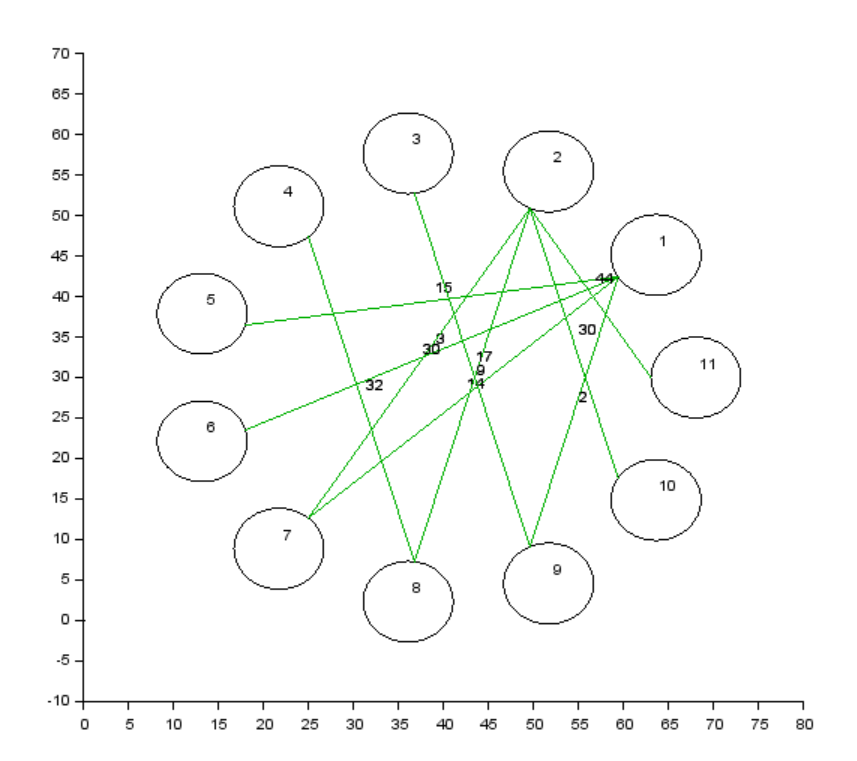
61. 44. 73. 82. 0. 0. 0. 0. 0. 65. 0.

**Результати виконання програми:**

Неорграф:



Кістяк:



**Таблиця відповідності ваги до стрілки(перша та друга вершини):**

"30(Weight):2(First Vertex):7(Second Vertex)"

"17(Weight):3(First Vertex):9(Second Vertex)"

"32(Weight):4(First Vertex):8(Second Vertex)"

"15(Weight):5(First Vertex):1(Second Vertex)"

"3(Weight):6(First Vertex):1(Second Vertex)"

"14(Weight):7(First Vertex):1(Second Vertex)"

"9(Weight):8(First Vertex):2(Second Vertex)"

"2(Weight):9(First Vertex):1(Second Vertex)"

"30(Weight):10(First Vertex):2(Second Vertex)"

"44(Weight):11(First Vertex):2(Second Vertex)"