**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 №6**

з дисципліни  
«Дискретні структури»

на тему «Пошук найкоротшого шляху»

Виконав: Перевірила:

студент групи ІП-93 Сергієнко А. А.  
Домінський Валентин Олексійович   
номер залікової книжки: 9311

Київ 2020

**Завдання на лабораторну роботу:**

1. Представити зважений ненапрямлений граф із заданими параметрами

так само, як у лабораторній роботі №5.

2. Створити скрипт для Scilab для пошуку найкоротшого шляху до кожної

вершини. Пошук виконувати з першої вершини. Якщо вона ізольована, то з

другої і так далі. При цьому в скрипті:

— встановити функцію halt у точці призначення чергової активної

вершини,

— виводити зображення графа у графічному вікні перед кожною

зупинкою по функції halt, виділяючи окремими кольорами тимчасові й постійні

вершини.

3. Під час обходу графа побудувати до кожної вершини графа найкоротший

шлях від стартової вершини.

Варіант 9311:

*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 = 3;**

**N3 = 1;**

**N4 = 1;**

**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**

**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)**

**Maximum = 999;**

**Root = 1;**

**//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**

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

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

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

**W=Wt+Wt';**

**//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**

**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))**

**function[]=Configuring(Marker)**

**Shape=get("current\_entity");**

**Shape.tag=Marker;**

**endfunction**

**function[Land]=getObjByTag(Marker)**

**Land="";**

**Shape=get("current\_figure");**

**Numbers=length(Shape.children.children)**

**for x=1:1:Numbers**

**if Shape.children.children(x).tag==Marker**

**Land=Shape.children.children(x)**

**end**

**end**

**endfunction**

**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);**

**Configuring("NodePoint"+string(x));**

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

**//(х,у).**

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

**end**

**function [MinimalCharacteristics] = LowestPath(FinalLoc, SettingUp)**

**Minimum = Maximum;**

**MinimalCharacteristics = -1;**

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

**if SettingUp(x) == 0 && FinalLoc(x) <= Minimum**

**Minimum = FinalLoc(x);**

**MinimalCharacteristics = x;**

**end**

**end**

**endfunction**

**WorkingNodes=list();**

**FinalLoc=list();**

**LowPath=[];**

**Root=1;**

**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);**

**Configuring("LinePoint"+string(i)+ " To "+string(j));**

**//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**

**Configuring("LinePoint"+string(i)+ " To "+string(j));**

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

**end**

**end**

**end**

**end**

**function[]=ShortestPath(W,Root)**

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

**FinalLoc(x)=Maximum;**

**SettingUp(x)=0;**

**end**

**FinalLoc(Root)=0;**

**for Sth=1:1:N-1**

**z=LowestPath(FinalLoc,SettingUp);**

**WorkingNodes($+1)=z;**

**SettingUp(z)=1;**

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

**if FinalLoc(z)~=Maximum && FinalLoc(z)+W(z,y)<FinalLoc(y) && W(z,y)~=0 && ~SettingUp(y)**

**FinalLoc(y)=FinalLoc(z)+W(z,y);**

**if z==Root**

**LowPath($+1)=string(z)+" To "+string(y)+" | weight = "+string(FinalLoc(y));**

**else**

**Progress=" To "+string(z)+" | weight = ";**

**Hill=grep(LowPath,Progress);**

**if (Hill>0)**

**Last=strsplit(LowPath(Hill),' To ')(size(strsplit(LowPath(Hill),' To '))(1)-1)**

**Progress=" To "+string(y)+" | weight = ";**

**Element=grep(LowPath,Progress);**

**if (Element>0)**

**Line=getObjByTag("LinePoint"+Last+" To "+string(y));**

**Line.segs\_color=ColorOfArrow\_1;**

**Line.thickness=1;**

**if (Last~=string(z) && Last~=string(Root))**

**LowPath(Element)=string(Root)+" To "+Last+" To "+string(z)+" To "+string(y)+" | weight = "+string(FinalLoc(y));**

**else**

**LowPath(Element)=Last+" To "+string(z)+" To "+string(y)+" | weight = "+string(FinalLoc(y));**

**end**

**else**

**Partic=string(Root)+" To "+string(z)+" To "+string(y)+" | weight = "+string(FinalLoc(y));**

**LowPath($+1)=Partic;**

**disp(LowPath)**

**end**

**end**

**end**

**Node=getObjByTag("NodePoint"+string(z));**

**Node.fill\_mode="on";**

**Node.thickness=5;**

**Node.background=3;**

**LineN=getObjByTag("LinePoint"+string(z)+" To "+string(y));**

**LineN.thickness=5;**

**halt()**

**end**

**end**

**Node=getObjByTag("NodePoint"+string(z));**

**Node.fill\_mode="off";**

**Node.background=3;**

**Node.thickness=5;**

**end**

**Node=getObjByTag("NodePoint4")**

**Node.thickness=5;**

**disp("Path and weight");**

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

**disp(string(x)+"-"+string(FinalLoc(x)));**

**end**

**disp(LowPath)**

**endfunction**

**ShortestPath(W,Root)**

**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.

**Перелік знайдених шляхів та їх ваг:**

"Path and weight"

"1-0"

"2-20"

"3-22"

"4-65"

"5-8"

"6-3"

"7-38"

"8-47"

"9-2"

"10-62"

"11-61"

"1 To 9 To 3 | weight = 22"

"1 To 5 | weight = 8"

"1 To 6 | weight = 3"

"1 To 7 | weight = 38"

"1 To 9 | weight = 2"

"1 To 6 To 10 | weight = 62"

"1 To 11 | weight = 61"

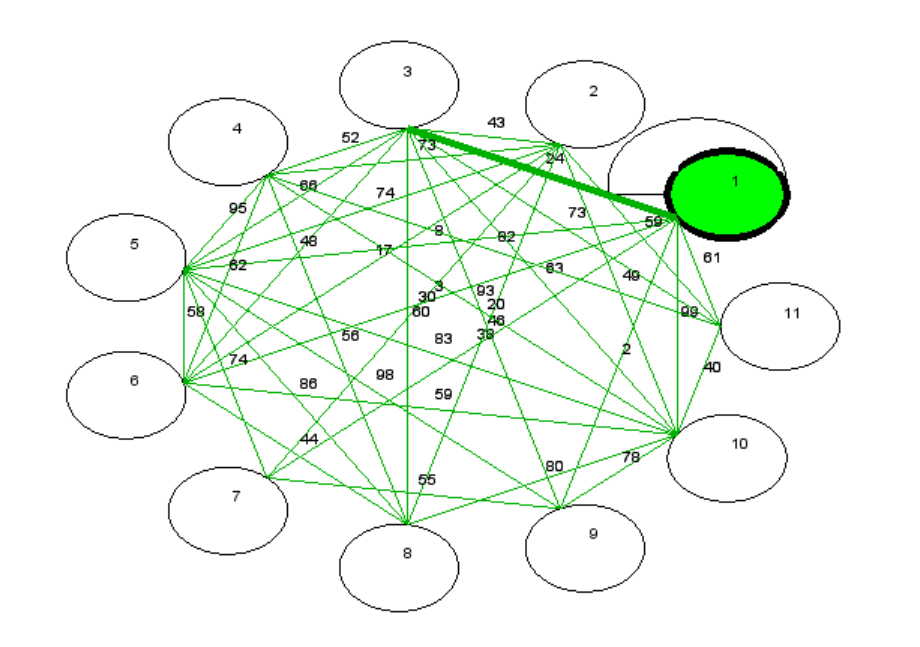
"1 To 6 To 2 | weight = 20"

"1 To 6 To 4 | weight = 65"

"1 To 6 To 8 | weight = 47"

**Результати виконання програми:**

Неорграф (та початок пошуку шляхів):



У процесі виконання (завершено):

