TEMA: Algoritmurile de parcgurgere a drumulurilor minime folosind: algoritmul Bellman-Kalaba si algoritmul Ford

1. **SCOPUL LUCRĂRII:**

* Studierea algoritmilor de determinare a drumurilor minime într-un graf.
* Elaborarea programelor de determinare a drumului minim într-un graf ponderat.

1. **NOTE DE CURS**

## Noţiune de drum minim

Pentru un graf orientat *G = (X,U)* se va numi *drum* un şir de vârfuri *D* = *(x0, x1,..., xr)* cu proprietatea că *(x0, x1)*, *(x1, x2)*,..., *(xr-1, xr)* aparţin lui *U*, deci sunt arce ale grafului şi extremitatea finală a arcului precedent coincide cu extremitatea iniţială a arcului următor.

Vârfurile *x0* şi *xr* se numesc extremităţile drumului *D*. Lungimea unui drum este dată de numărul de arce pe care le conţine. Dacă vârfurile *x0, x1,..., xr* sunt distincte două câte două drumul *D* este elementar.

Adeseori, fiecărui arc (muchii) i se pune în corespondenţă un număr real care se numeşte *ponderea* (lungimea) arcului. Lungimea arcului *(xi, xj)* se va nota *w(i,j)*, iar în cazul în care un arc este lipsă ponderea lui va fi considerată foarte mare (pentru calculator cel mai mare număr pozitiv posibil). În cazul grafurilor cu arce ponderate (grafuri ponderate) se va considera lungime a unui drum suma ponderilor arcelor care formează acest drum. Drumul care uneşte două vârfuri concrete şi are lungimea cea mai mică se va numi *drum minim* iar lungimea drumului minim vom numi *distanţă*. Vom nota distanţa dintre *x* şi *t* prin *d(x, t)*, evident, *d(x,x)=0*.

## Algoritmul lui Ford pentru detrminarea drumului minim

Permite determinarea drumului minim care începe cu un vârf iniţial *xi* până la oricare vârf al grafului *G*. Dacă prin *Lij* se va nota ponderea arcului *(xi, xj)* atunci algoritmul conţine următorii paşi:

1. Fiecărui vârf *xj* al grafului *G* se va ataşa un număr foarte mare *Hj(∞)*. Vârfului iniţial i se va ataşa *Ho = 0*;
2. Se vor calcula diferenţele *Hj - Hi* pentru fiecare arc *(xi, xj)*. Sunt posibile trei cazuri:
3. *Hj - Hi < Lij,*
4. *Hj - Hi = Lij,*
5. *Hj - Hi > Lij.*

Cazul "*c*" permite micşorarea distanţei dintre vârful iniţial şi *xj* din care cauză se va realiza *Hj = Hi + Lij*.

Pasul 2 se va repeta atâta timp cât vor mai exista arce pentru care are loc inegalitatea “c”. La terminare, etichetele *Hi* vor defini distanţa de la vârful iniţial până la vârful dat *xi*.

3. Acest pas presupune stabilirea secvenţei de vârfuri care va forma drumul minim. Se va pleca de la vârful final *xj* spre cel iniţial. Predecesorul lui *xj* va fi considerat vârful *xi* pentru care va avea loc *Hj - Hi = Lij*. Dacă vor exista câteva arce pentru care are loc această relaţie se va alege la opţiune.

## Algoritmul Bellman - Kalaba

Permite determinarea drumului minim dintre oricare vârf al grafului până la un vârf, numit vârf final.

Etapa iniţială presupune ataşarea grafului dat *G* a unei matrice ponderate de adiacenţă, care se va forma în conformitate cu următoarele:

1. *M(i,j)* = *Lij*, dacă există arcul *(xi, xj)* de pondere *Lij*;
2. *M(i,j)* = ∞, unde ∞ este un număr foarte mare (de tip întreg maximal pentru calculatorul dat), dacă arcul *(xi, xj)* este lipsă;
3. *M(i,j)* = *0*, dacă *i = j*.

La etapa a doua se va elabora un vector *V0* în felul următor:

1. *V0(i) = Lin*, dacă există arcul *(xi, xn)*, unde *xn* este vârful final pentru care se caută drumul minim, *Lin* este ponderea acestui arc;
2. *V0(i) =* ∞, dacă arcul *(xi, xn)* este lipsă;
3. *V0(i) = 0*, dacă *i = j*.

### Algoritmul constă în calcularea iterativă a vectorului *V* în conformitate cu următorul procedeu:

1. *Vk(i) = min{Vk-1; Lij+Vk-1(j)}*, unde *i = 1, 2,…, n - 1, j = 1, 2,..., n*; *i<>j;*
2. *Vk(n) = 0*.

Când se va ajunge la *Vk = Vk-1* - STOP.

Componenta cu numărul *i* a vectorului *Vk* cu valoarea diferită de zero ne va da valoarea minimă a drumului care leagă vârful *i* cu vârful *n*.

1. **SARCINA DE BAZĂ**
2. Elaboraţi procedura introducerii unui graf ponderat;
3. Elaboraţi procedurile determinării drumului minim;
4. Realizaţi un program cu următoarele funcţii:

* introducerea grafului ponderat cu posibilităţi de analiză sintactică şi semantică şi de corectare a informaţiei;
* determinarea drumului minim;
* extragerea informaţiei la display şi printer (valoarea drumului minim şi succesiunea vârfurilor care formează acest drum).

**De exemplu:**

**8**

**9 11 17**

**11 15**

**6 10 14**

**12 11 17 11 8**

**6**

**Fig1 Graf arbitrar**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MA** | **X1** | **X2** | **X3** | **X4** | **X5** | **X6** | **X7** | **X8** |
| **X1** | **0** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **X2** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **0** |
| **X3** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** |
| **X4** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** |
| **X5** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** |
| **X6** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** |
| **X7** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| **X8** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

**Fig.2 Matrice de adiacenta**

**4.Concluzie**

Efecuand aceasta lucrare de laborator,am obtinut practica de lucru metodele de reprezentare ale grafului,si metodele lui de stocare in memoria calculatorului.

Pe parcursul lucrarii, am ajuns la concluzia ca cea mai simpla metoda de introducere a grafului de la tastatura este lista adiacenta,de asemenea aceasta metoda utilizeaza si cel mai putin din memoria calculatorului.

Dupa parerea mea, cel mai usor putem desena graful,daca ne orientam dupa matricea de incidenta,insa ea ocupa un volum mai mare din memoria calculatorului.

**5.Bibliografie**

1.Conspectul de matematica discreta predat de dr.conf.univ. Dohotaru L.

2.Conspetul de SDA predat de dr.conf.univ.Kulev M.

**Anexa A**

**Listing-ul programului**

#include <stdio.h>

#include <conio.h>

#include <process.h>

#include<dos.h>

#include<time.h>

#include<windows.h>

void setcolor(unsigned short color)

{

HANDLE hcon = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hcon,color);

}

int a[20][20],x,c[20][20];

int vector(int a[15],int u)

{int m,i,t=a[0];

for(m=u;m>=2;m--)

for(i=0;i<m-1;i++)

if(t>a[i+1])

t=a[i+1];

return t;

}

int vector1(int a[15],int u)

{int m,i,t=a[0];

for(m=u;m>=2;m--)

for(i=0;i<m-1;i++)

if(t<a[i+1])

t=a[i+1];

return t;

}

void bkmin()

{ int i,j,x1,x2;

int m[15],m1[15],m2[15];

int k,l,t,o,n,p[15],t1;

for(i=0;i<x;i++)

for(j=0;j<x;j++)

{if(a[i][j]==0)

c[i][j]=1000;

if(i==j) c[i][j]=0;}

r: printf("Dati virful final:");

scanf("%d",&x2);

if(x2<0||x2>x) {printf("\nAsa virf graful dat nu Exista!\n");goto r;}

for(i=0;i<x;i++)

m[i]=c[i][x2-1];

a:for(i=0;i<x;i++)

{for(j=0;j<x;j++)

m1[j]=c[i][j]+m[j];

m2[i]=vector(m1,x);

}

k=0;

for(j=0;j<x;j++)

if(m2[j]==m[j]) k++;

for(j=0;j<x;j++)

m[j]=m2[j];

if(k!=x) goto a;

for(x1=1;x1<=x;x1++){

if(m[x1-1]==1000){printf("Drumul din x%d in x%d nu exista in graful dat\n",x1,x2);

getch(); goto b; }

if(x1==x2) goto b;

printf("p[dmin(x%d,x%d)]=%d\n",x1,x2,m[x1-1]);

l=x1-1;o=0;

c:for(j=0;j<x;j++)

{m2[j]=m[j]+c[l][j];

p[j]=j;}

for(n=x;n>=2;n--)

for(k=0;k<n-1;k++)

{if(m2[k]>m2[k+1])

{t=m2[k+1];m2[k+1]=m2[k];m2[k]=t;

t1=p[k+1];p[k+1]=p[k];p[k]=t1;}

if(m2[k]==m2[k+1]&&p[k]==l)

{t1=p[k+1];p[k+1]=p[k];p[k]=t1;}

}

m1[o]=p[0];o++;l=p[0];

if(l!=x2-1) goto c;

printf(" Drumul de valoarea minima din virful x%d in virful x%d este\n",x1,x2);

printf("(x%d",x1);

for(j=0;j<o;j++)

printf(",x%d",m1[j]+1);

printf(")\n");

k=getch();

b: if(k==27) break;

}

}

void bkmax()

{ int i,j,x1,x2;

int m[15],m1[15],m2[15];

int k,l,t,o,n,p[15],t1;

for(i=0;i<x;i++)

for(j=0;j<x;j++)

{if(a[i][j]==0)

c[i][j]=-1000;

if(i==j) c[i][j]=0;}

r: printf("Dati virful final:");

scanf("%d",&x2);

if(x2<0||x2>x) {printf("\nAsa virf graful dat nu Exista!\n");goto r;}

for(i=0;i<x;i++)

m[i]=c[i][x2-1];

a:for(i=0;i<x;i++)

{for(j=0;j<x;j++)

m1[j]=c[i][j]+m[j];

m2[i]=vector1(m1,x);

}

k=0;

for(j=0;j<x;j++)

if(m2[j]==m[j]) k++;

for(j=0;j<x;j++)

m[j]=m2[j];

if(k!=x) goto a;

for(x1=1;x1<=x;x1++){

if(x1==x2) goto b;

if(m[x1-1]==-1000){printf("Drumu din x%d in x%d nu exista in graful dat",x1,x2);

getch(); goto b;}

if(m[x1-1]>1000){printf("Graful dat contine circuit!!!");

int delay(1000);

break;}

printf("p[dmax(x%d,x%d)]=%d\n",x1,x2,m[x1-1]);

l=x1-1;o=0;

c:for(j=0;j<x;j++)

{m2[j]=m[j]+c[l][j];

p[j]=j;}

for(n=x;n>=2;n--)

for(k=0;k<n-1;k++)

{if(m2[k]<m2[k+1])

{t=m2[k+1];m2[k+1]=m2[k];m2[k]=t;

t1=p[k+1];p[k+1]=p[k];p[k]=t1;}

if(m2[k]==m2[k+1]&&p[k]==l)

{t1=p[k+1];p[k+1]=p[k];p[k]=t1;}

}

m1[o]=p[0];o++;l=p[0];

if(l!=x2-1) goto c;

printf(" Drumul de valoarea maxima din virful x%d in virful x%d este\n",x1,x2);

printf("(x%d",x1);

for(j=0;j<o;j++)

printf(",x%d",m1[j]+1);

printf(")\n");

k=getch();

b: if(k==27) break;

}

}

void ford()

{struct arc{int u;

int o;}b[30];

int i,j,x1,x2,t=0;

int p1[15],p[15],k,A,l,d[15],y1,y2;

clrscr();

for(i=0;i<x;i++)

for(j=0;j<x;j++)

if(a[i][j]!=0)

{b[t].u=i;b[t].o=j;t++;}

b: printf("Introduceti virful initial:\n");

scanf("%d",&x1);

if(x1<0||x1>x) {printf("\nAsa virf graful dat nu contine!\n");goto b;}

for(j=0;j<x;j++)

{p[j]=1000;

if(j==x1-1) p[j]=0;

}

p:for(i=0;i<x;i++)

p1[i]=p[i];

for(i=0;i<t;i++)

{k=p[b[i].o]-p[b[i].u];

if(k>c[b[i].u][b[i].o]) {p[b[i].o]=p[b[i].u]+c[b[i].u][b[i].o];}

}

k=0;

for(i=0;i<x;i++)

if(p1[i]==p[i]) k++;

if(k!=x) goto p;

for(x2=1;x2<=x;x2++)

{ if(x1==x2) goto a;

if(p[x2-1]==1000) {printf("Nu exista drumul din x%d in x%d\n",x1,x2);getch();goto a;}

if(x2<0||x2>x) {printf("\nAsa virf graful dat nu Exista!\n");getch();goto a;}

printf("Valoarea drumului minim din virful x%d in virful x%d este\n",x1,x2);

printf("p[dmin(x%d,x%d)]=%d",x1,x2,p[x2-1]);

printf("\nDrumul de valoarea minima din virful x%d in virful x%d este\n",x1,x2);

l=x2-1;k=0;

for(i=t-1;i>=0;i--)

if(b[i].o==l)

if(p[b[i].o]-p[b[i].u]==c[b[i].u][b[i].o]) {d[k]=b[i].o;k++;l=b[i].u;}

printf("dmin=(x%d",x1);

for(i=k-1;i>=0;i--)

printf(",x%d",d[i]+1);

printf(")\n");

A=getch();

a: if(A==27) break;

}

}

void intr()

{ int xx,yy,A,t,n,m,m1,n1,i1,j1,i,j;

clrscr();

setcolor(10);

printf("\n\n\n\n\n\

ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»\n\

º º\n\

º Lucrare de laborator Nr4 la Matematica Discreta. º\n\

º º\n\

ÌÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ%c\n\

º TEMA: º\n\

º Algoritmul de parcurgere a drumurilor minime folosind º\n\

º algoritmul Bellman-Kalaba si algoritmul Ford º\n\

º º\n\

º º\n\

º A efectuat : st.gr. º\n\

º º\n\

º A verificat: Lector asistent º\n\

º º\n\

º º\n\

ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼\n",185);

getch();

clrscr();

printf("\n\

ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»\n\

º INTRODUCETI DATELE º\n\

ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼\n",185);

printf(" Numarul de virfuri este:");

scanf("%d",&x);

printf("Introduceti matricea de adiacenta:");

printf("\n\nA ");

for (i=0;i<x;i++)

printf("x%d ",i+1);

printf("\n");

yy=9;xx=0;

for (i=0;i<x;i++)

{ yy=yy+1;

xx=0;

printf("\nx%d",i+1);

for (j=0;j<x;j++)

{ xx=xx+6;

a:gotoxy(xx,yy);

A=getch();

if(A==48){a[i][j]=0;printf("%d",a[i][j]);}

if(A==49){if(i==j) goto a;a[i][j]=1;printf("%d",a[i][j]);}

if((A!=48)&&(A!=49)){xx=xx-6;j--;}

}

} n=xx,m=yy;m1=0,n1=0;

for(;;)

{ b: t=getch();

if (t==13) break;

if(t==72) {m--;m1++;gotoxy(n,m); }

if(t==80) {m++;m1--;gotoxy(n,m);}

if(t==77) {n=n+6;n1--;gotoxy(n,m);}

if(t==75) {n=n-6;n1++;gotoxy(n,m);}

if(t==48) {i1=i-m1-1;j1=j-n1-1; a[i1][j1]=0; printf("%d",a[i1][j1]);gotoxy(n,m);}

if(t==49){i1=i-m1-1;j1=j-n1-1;if(i1==j1) goto b;a[i1][j1]=1; printf("%d",a[i1][j1]);gotoxy(n,m); }

if(t==83) {printf(" ");}

}

}

void main()

{int c1,i,j,A;

intr();

clrscr();

printf("\n");

for(i=0;i<x;i++)

for(j=0;j<x;j++)

if(a[i][j]!=0)

{printf("p(x%d,x%d)=",i+1,j+1);

scanf("%d",&c[i][j]);}

a:printf("\nDoriti sa corectati poderea unui arc(enter-da;orice tasta-nu;)?\n");

A=getch();

if(A==13){printf("Dati virful initial=");scanf("%d",&i);

printf("Dati virful final=");scanf("%d",&j);

printf("p(x%d,x%d)=",i,j);scanf("%d",&c[i-1][j-1]);goto a;

}

for(;;) { clrscr();

printf("\n\t\tAlegeti algoritmul care sa prelucreze graful dat:\n");

printf("\n\t1-Algoritmul Bellman-Kalaba de determinare a drumurilor minime;\n");

printf("\n\t2-Algoritmul Bellman-Kalaba de determinare a drumurilor maxime;\n");

printf("\n\t3-Algoritmul Ford de determinare a drumurilor de valoare minima;\n");

printf("\n\t4-Exit;\n");

scanf("%d",&c1);

switch(c1)

{ case 1:bkmin();break;

case 2:bkmax();break;

case 3:ford();break;

case 4:exit(0);break;

}

}

}

ANEXA B

Screenshoturi













