****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Calancea Cătălin**

**MI-222**

**Raport**

**pentru lucrarea de laborator Nr.4**

***la cursul de “Matematica Discreta”***

Verificat:

**Orindas Victoria,** *asist. univ.*

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău – 2023**

**Scopul lucrării:** Studierea algoritmului Ford.Elaborarea programului de cautare a drumului minim si maxim folosind algoritmul Ford.

**Sarcini:**

1. Elaborati procedura care va realiza algoritmul Ford de cautare a drumului maxim si minim.
2. Folosind procedurile din lucrarile precedente , elaborati programul care va permite:

* Introducerea grafului in calculator;
* Cautarea drumului maxim si minim;
* Extreagerea datelor la display.

**Codul programului:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <conio.h>

#define max 100

int a[100][100],n,m,j,i,k=0,numar=0,loc=0,h[100],h2[100],elem=1001,b[100],c[100],q=1;

void d\_min\_Ford( int j2)

{

int i2 ;

if(elem<1000){

for(i2=0;i2<numar;i2++)

if(c[i2]==elem) loc= i2; // introduc elementele ce lipsesc in drumul minim

// adica elementele de dupa el.recursie din drumul minim precedent

for(i2=numar-1;i2>=loc;i2--)

b[k++]= c[i2];

elem=1001; }

b[k++]=j2;

if(j2==1) {

printf("\n#Drumul minim[ %d ] : ",q++);

for(i2=k-1;i2>=0;i2-- )

printf("%3d",b[i2]);

numar = k;

for(i2=k-1;i2>=0;i2--)

c[k-i2-1]=b[i2]; //salvez drumul minim precedent

k=0;

}

for(i2=n-1;i2>=1;i2--){

if(a[i2][j2]>0)

if( h2[j2]-h2[i2] == a[i2][j2] ){

d\_min\_Ford(i2); elem=j2; // salvez elementul de la care se apeleaza recursia

}

}

}

void afisare\_Ford (int a[100][100], int n , int m)

{

printf("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n| | | | |\n");

printf("| (xi,xj) |"); printf(" Pij |"); printf(" Hj - Hi |"); printf(" Hj - Hi |");

printf("\n|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

// afisarea rezultatelor

for(i=1;i<n;i++)

for(j=1;j<n;j++)

if(a[i][j]>0&&a[i][j]<max)

{

printf("\n| (%2d,%2d) |",i,j); printf(" %3d |",a[i][j]);

if( h[j]-h[i] == a[i][j] ){ printf(" H%d - H%d = %3d = %2d |",j,i,a[i][j],a[i][j]); }

if( h[j]-h[i] < a[i][j] ) { printf(" H%d - H%d = %3d < %2d |",j,i,h[j]-h[i],a[i][j]); }

if( h[j]-h[i] > a[i][j] ) {

if(h[j]-h[i]>500)

printf(" H%d - H%d = inf > %2d |",j,i,a[i][j]);

else printf(" H%d - H%d = %3d > %2d |",j,i,h[j]-h[i],a[i][j]);

h[j]=h[i]+a[i][j]; }

// ----------------------------------------------------------

if( h2[j]-h2[i] == a[i][j] ){ printf(" H%d - H%d = %3d = %2d |",j,i,a[i][j],a[i][j]); }

if( h2[j]-h2[i] < a[i][j] ) { printf(" H%d - H%d = %3d < %2d |",j,i,h2[j]-h2[i],a[i][j]); }

printf("\n|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

}

}

void main ()

{

system("COLOR 2");

printf ("\nIntrodu numarul de virfuri : "); scanf ( "%d",&n); n++;

printf("\nIntroduce-ti Graful Ponderat : \n\n");

//introduc graful ponderat

for(i=1;i<n;i++)

for(j=1;j<n;j++)

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

// initializez lungimea drumurilor

h[1]=0;

for(i=2;i<n;i++) h[i]=max;

h2[1]=0;

for(i=2;i<n;i++) h2[i]=max;

// a doua coloana

for(i=1;i<n;i++)

for(j=1;j<n;j++)

if(a[i][j]>0)

{

if( h2[j]-h2[i]>a[i][j] ) h2[j]=h2[i]+a[i][j];

}

afisare\_Ford(a,n,m);

printf("\n\n\n\tLungimea drumului minim = %d",h2[n-1]);

printf ("\n\n\*\*\*Drumurile Minime\*\*\*\n");

d\_min\_Ford(n-1);

getch() ;

}

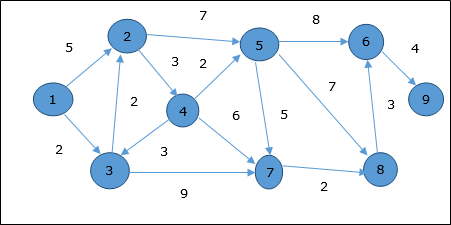
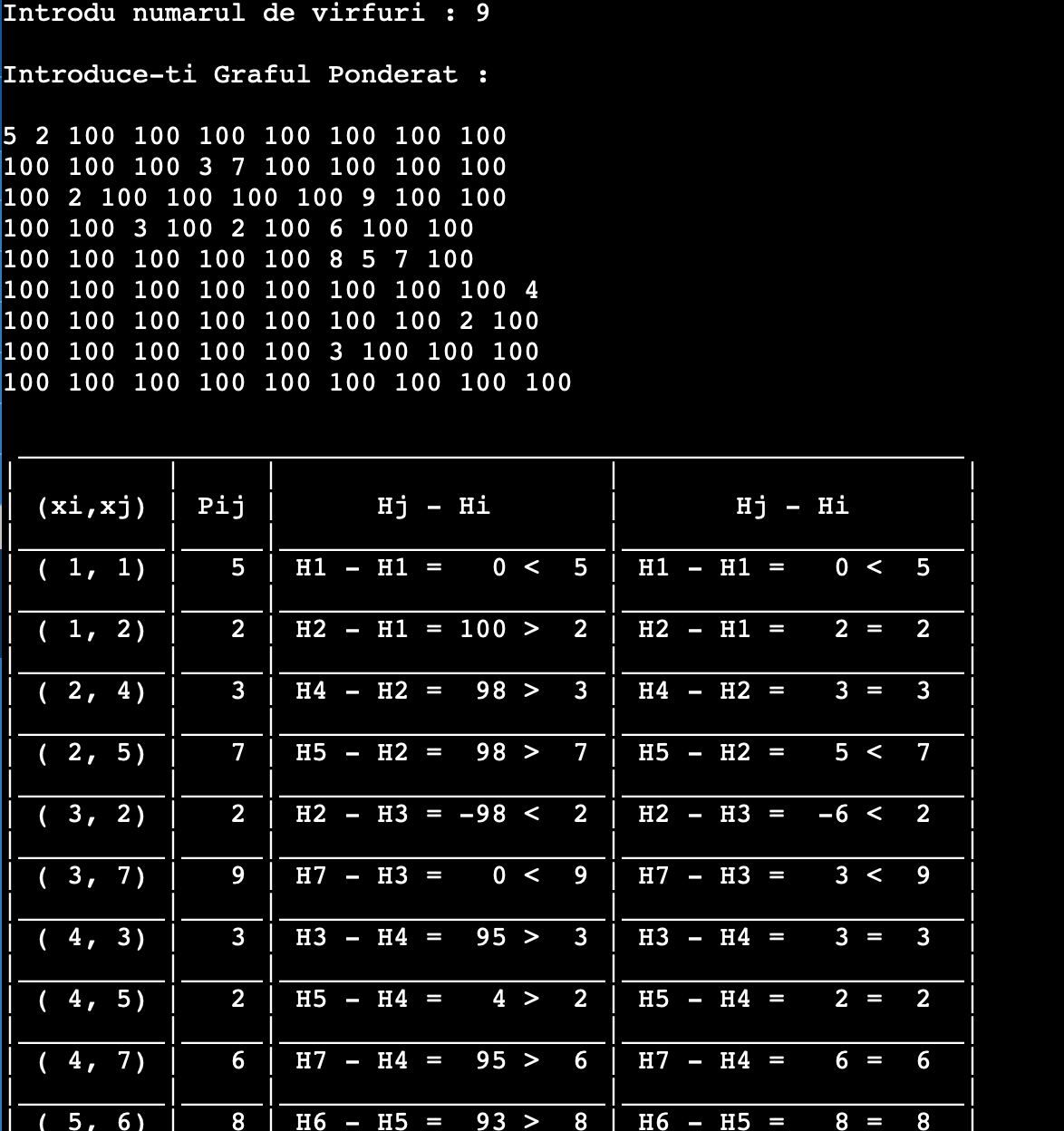
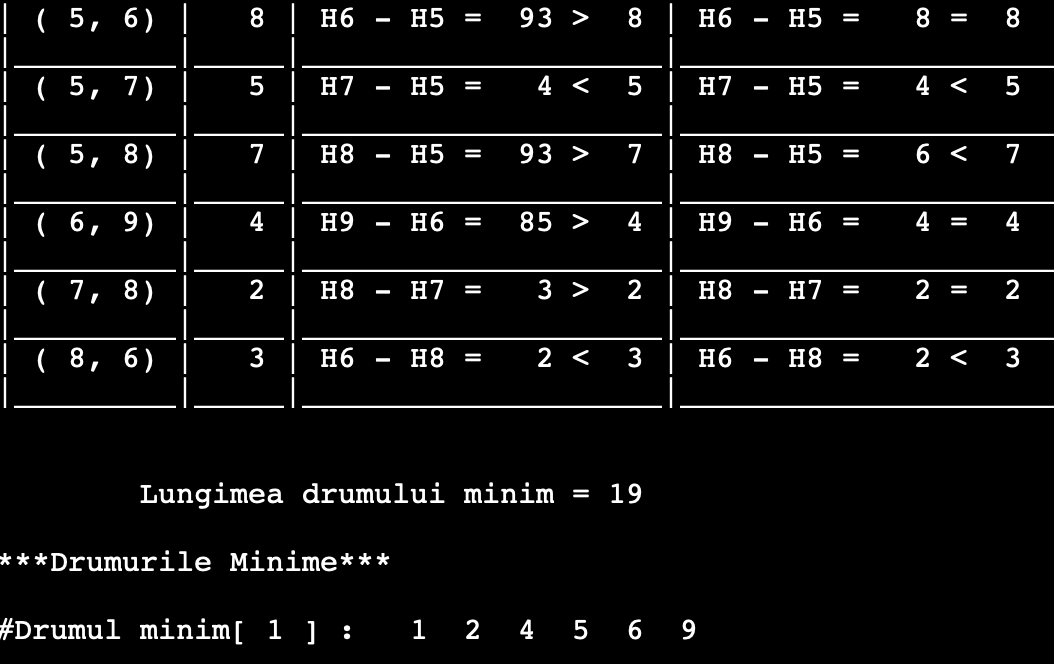


Figura 1. Graf

****

****

**Concluzie:**În urma executarii lucrării de laborator nr.4 , algoritmul lui Ford este o metodă eficientă de găsire a drumului minim și maxim într-un graf ponderat, prin utilizarea unei abordări iterative și actualizarea continuă a distanțelor și a predecesorilor nodurilor. Acesta poate fi utilizat într-o varietate de aplicații practice, cum ar fi rețelele de transport sau comunicații. Elaborarea unui program de căutare a drumului minim și maxim folosind algoritmul lui Ford poate fi realizată în diferite limbaje de programare și platforme, iar performanța acestuia depinde de eficiența implementării și de complexitatea grafurilor de analizat. În general, algoritmul lui Ford este o tehnologie fundamentală și utilă în domeniul informaticii și algoritmică, care poate fi aplicată într-o varietate de situații și domenii practice.