MINISTERUL EDUCAŢIEI și CERCETĂRII al REPUBLICII MOLDOVA UNIVERSITATEA TEHNICĂ a MOLDOVEI   
FACULTATEA CALCULATOARE, INFORMATICǍ şi MICROELECTRONICǍ Departamentul Informatică și Ingineria Sistemelor

**Lucrare de laborator nr.4**

La Matematica Discretă

Tema: ***Păstrarea grafurilor în memoria calculatorului.***

***Matricea drumurilor***

A realizat Chistol Maxim, st. gr. IA-233  
A verificat Vladimir Melnic

**Chișinău 2024**

**Cuprins**

[**1. Scopul lucrării** 3](#_Toc158886713)

[**2. Sarcina** 4](#_Toc158886714)

[**3. Concluzii** 13](#_Toc158886715)

# **Scopul lucrării**

1. Studierea algoritmilor de determinare a drumurilor minime și maxime întrun graf;

2. Elaborarea programelor de determinare a drumului minim și maxim întrun graf ponderat

# **Sarcina**

1. Elaboraţi procedura introducerii unui graf ponderat.

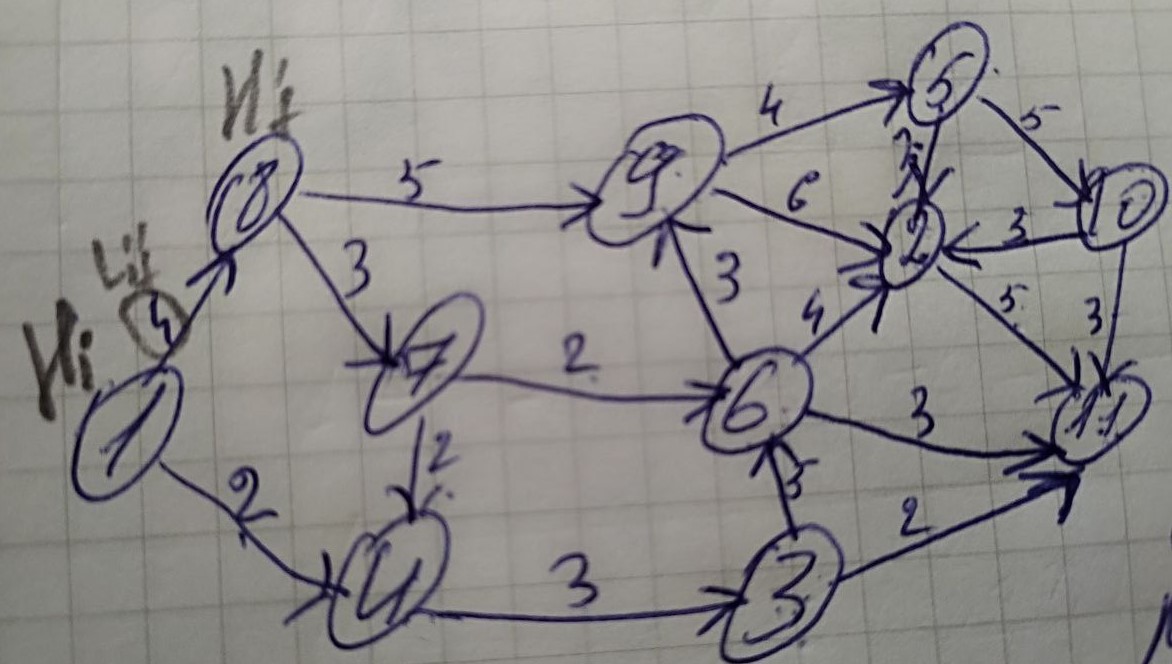
2. Elaboraţi procedurile determinării drumului minim utilizând algoritmul Ford.

3. Realizaţi un program cu următoarele funcţii:

➢ Introducerea grafului ponderat cu posibilităţi de analiză sintactică şi semantică şi de corectare a informaţiei;

➢ Determinarea drumului minim utilizând algoritmul Ford

➢ Extragerea informaţiei la display (valoarea drumului minim şi succesiunea vârfurilor care formează acest drum);

****

**Codul:**

#include <stdio.h>

#include <limits.h>

#define NR\_MAX\_VARFURI 10

#define INF INT\_MAX

struct Muchie {

    int sursa;

    int destinatie;

    int pondere;

};

// Funcția 'afiseazaDrum' afișează drumul minim de la nodul sursă la nodul j folosind informațiile despre părinți.

 // Dacă nodul j nu are părinte, înseamnă că am ajuns la nodul sursă și îl afișăm.

    // Apelăm recursiv funcția pentru a afișa drumul de la sursă la părintele nodului j, apoi afișăm nodul j.

void afiseazaDrum(int parinte[], int j) {

    if (parinte[j] == -1) {

        printf("%d ", j);

        return;

    }

    afiseazaDrum(parinte, parinte[j]);

    printf("%d ", j);

}

void Ford(struct Muchie muchii[], int nr\_muchii, int sursa, int nr\_varfuri, int dist[], int parinte[]) {

    for (int i = 0; i < nr\_varfuri; i++) {

        dist[i] = INF;//ca fiecarui varf ii se va atasa un numar foarte mare Hj=INF

        //dist=distanta  il initializam cu infinit,din cauza ca el trebuei sa fie mai mare ca ponderea .....din cauza ca avem formula Hj-Hi>Lij, in dist se salveaza costul minim a drumului unde el parcurgand le actualizeaza pe pozitie

        parinte[i] = -1;//-1 Ca parinte  este  nedefinit

    }

    dist[sursa] = 0;

// Pentru fiecare muchie din graf, încercăm să îmbunătățim distanța până la nodul destinație.

// Dacă distanța până la nodul sursă nu este infinită și putem îmbunătăți distanța până la nodul destinație

    for (int i = 0; i < nr\_varfuri - 1; i++) {// conditia  i < nr\_varfuri - 1 cand se va indiplini se va opri algoritmul

        for (int j = 0; j < nr\_muchii; j++) { // implimentarea for care patrunde prin toate muchile care sunt posibile la varful dat

            int u = muchii[j].sursa;

            int v = muchii[j].destinatie;

            int pondere = muchii[j].pondere;

            if (dist[u] != INF && dist[u] + pondere < dist[v]) { // Actualizăm distanța și părintele nodului destinație.

                dist[v] = dist[u] + pondere;

                parinte[v] = u;

            }

        }

    }

}

// Citim numărul de noduri și muchii din graf.

 // Citim muchiile și ponderile lor.

  // Citim nodurile sursă și destinație pentru care dorim să calculăm drumul minim.

  // Apelăm algoritmul Ford pentru a calcula drumurile minime de la nodul sursă la toate celelalte noduri.

 // Afișăm drumul minim de la nodul sursă la nodul destinație.

int main() {

    int nr\_varfuri, nr\_muchii, sursa, destinatie;

    struct Muchie muchii[NR\_MAX\_VARFURI \* NR\_MAX\_VARFURI]; // Lista de muchii din graf.

    int dist[NR\_MAX\_VARFURI]; // Vectorul de distanțe de la nodul sursă la fiecare alt nod.

    int parinte[NR\_MAX\_VARFURI]; // Vectorul de părinți pentru a reconstitui drumul minim.

    printf("Introduceti numarul de varfuri si muchii: ");

    scanf("%d %d", &nr\_varfuri, &nr\_muchii);

    printf("Introduceti muchiile si ponderile lor (ex: sursa destinatie pondere):\n");

    for (int i = 1; i <= nr\_muchii; i++) {

        scanf("%d %d %d", &muchii[i].sursa, &muchii[i].destinatie, &muchii[i].pondere);

    }

    printf("Introduceti varful sursa si varful destinatie pentru calculul drumului minim: ");

    scanf("%d %d", &sursa, &destinatie);

       Ford(muchii, nr\_muchii, sursa, nr\_varfuri, dist, parinte);

    printf("Drumul minim de la varful %d la varful %d este: ", sursa, destinatie);

    if (dist[destinatie] != INF) {

        afiseazaDrum(parinte, destinatie);

    } else {

        printf("Nu exista drum.\n");

    }

    return 0;

}

# **3. Concluzii**

Lucrarea a reușit să atingă obiectivele propuse, focalizându-se pe elaborarea procedurilor și a programelor pentru lucrul cu grafuri ponderate și determinarea drumurilor minime și maxime în acestea.

Prin elaborarea procedurilor de introducere a unui graf ponderat, s-a asigurat o analiză completă a informației, inclusiv verificarea sintactică și semantică, precum și corectarea eventualelor erori. De asemenea, s-a dezvoltat o procedură eficientă pentru determinarea drumului minim utilizând algoritmul Ford, care să ofere rezultate precise și fiabile.

Implementarea programului a permis integrarea acestor proceduri într-un instrument practic și ușor de utilizat. Utilizatorul poate introduce un graf ponderat, să determine drumul minim folosind algoritmul Ford și să afișeze informațiile relevante despre acesta, contribuind astfel la înțelegerea și utilizarea conceptelor de teorie a grafurilor într-un mod practic și eficient.

Studiul algoritmilor de determinare a drumurilor minime și maxime a oferit o viziune mai profundă asupra acestei ramuri a informaticii, facilitând înțelegerea principiilor și mecanismelor din spatele acestor procese.

În final, lucrarea reprezintă un pas important în direcția aplicării practice a conceptelor teoretice și algoritmice asociate cu teoria grafurilor, oferind un cadru solid pentru dezvoltarea ulterioară a aplicațiilor și algoritmilor în domenii precum rețelele de transport, logistica sau analiza rețelelor sociale.