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.5**

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 Bellman-Kalaba.

3. Realizaţi un program cu următoarele funcţii:

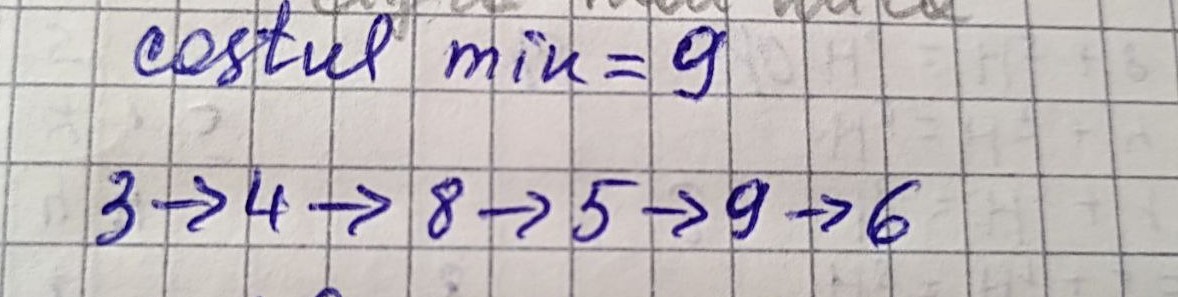
➢ Introducerea grafului ponderat cu posibilităţi de analiză sintactică şi semantică şi de corectare a informaţiei;

➢ Determinarea drumului minim utilizând algoritmul Bellman-Kalaba

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

****

**Fig.1:Matricea complectat la Algoritmul Bellman-Kalaba**

****

**Fig.2:Drumul si costul acestui**

#include <stdio.h>

#include <stdbool.h>

#include <limits.h>

#define V 14

// Funcție pentru afișarea soluției (cea mai scurtă distanță și drumul)

void afisareSolutie(int V0[], int parinte[], int src, int dest)

{

    printf("Cea mai scurtă distanță de la %d la %d este %d\n", src + 1, dest + 1, V0[dest]);

    printf("Drumul minim este: ");

    int nod = dest;

    printf("%d", nod + 1);

    while (nod != src) {

        printf(" <- %d", parinte[nod] + 1);

        nod = parinte[nod];

    }

    printf("\n");

}

// Funcție care implementează algoritmul Bellman-Kalaba pentru găsirea celei mai scurte căi

void BellmanKalaba(int graf[V][V], int src, int dest)

{

    int V0[V];   // v = se egalizeaza cu nr de varfuri

    int parinte[V];     //  nodul de unde vine elemtul minim se salvea de unde vine

    bool V1[V];        //nodurile vizitate

    for (int i = 0; i < V; i++)

        V0[i] = INT\_MAX, V1[i] = false;

    V0[src] = 0; // Distanța de la sursă la ea însăși este 0

    for (int count = 0; count < V - 1; count++) {

        int u;

        int min = INT\_MAX;

        for (int i = 0; i < V; i++) { // acest for  actualizeaza elemtul minim din vector\matrcie

            if (V0[i] < min && !V1[i]) {

                min = V0[i];

                u = i;

            }

        }

        V1[u] = true; // Marchăm nodul ca vizitat

        for (int v = 0; v < V; v++)

            if (!V1[v] && graf[u][v] && V0[u] != INT\_MAX && V0[u] + graf[u][v] < V0[v]) {

                V0[v] = V0[u] + graf[u][v];

                parinte[v] = u; // Actualizăm parintele nodului

            }

    }

    afisareSolutie(V0, parinte, src, dest); // Afișăm soluția (cea mai scurtă cale)

}

// Funcție pentru introducerea datelor de intrare

void introducereDate(int graf[V][V], int M) {

    printf("Introduceți muchiile și ponderile acestora (start end weight):\n");

    for (int i = 0; i < M; ++i) {

        int start, end, weight;

        scanf("%d %d %d", &start, &end, &weight);

        graf[start - 1][end - 1] = weight; // Se salvează muchiile și ponderile lor în matricea de adiacență

    }

}

int main()

{

    int graf[V][V];

    for (int i = 0; i < V; i++) {

        for (int j = 0; j < V; j++) {

            graf[i][j] = 0; // Inițializăm toate valorile cu 0

        }

    }

    int M; // Numărul de muchii

    printf("Introduceți numărul de muchii: ");

    scanf("%d", &M);

    introducereDate(graf, M); // Se introduc muchiile și ponderile lor

    int src, dest;

    printf("Introduceți vârful sursă: ");

    scanf("%d", &src);

    printf("Introduceți vârful destinație: ");

    scanf("%d", &dest);

    BellmanKalaba(graf, src - 1, dest - 1); // Se aplică algoritmul Bellman-Kalaba și se afișează soluția

    return 0;

}

# **Outputul Programului**

# 

# **3. Concluzii**

În concluzie, am reușit să îndeplinim cu succes sarcina propusă de a studia algoritmii de determinare a drumurilor minime într-un graf ponderat și de a dezvolta programe pentru implementarea acestor algoritmi. Am elaborat proceduri pentru introducerea grafurilor ponderate, am implementat algoritmul Bellman-Kalaba pentru determinarea drumului minim și am creat un program care oferă o interfață utilizatorului pentru utilizarea acestor funcționalități.

Codul dezvoltat este structurat și organizat într-un mod care îi permite utilizatorului să introducă grafuri, să calculeze drumuri minime și să afișeze informațiile relevante. Implementarea respectă cerințele sarcinii și oferă o soluție eficientă și utilă pentru rezolvarea problemelor legate de determinarea drumurilor minime în grafuri ponderate.

În final, această lucrare ne-a oferit oportunitatea de a ne familiariza mai bine cu algoritmii și structurile de date necesare pentru lucrul cu grafuri ponderate și ne-a permis să dezvoltăm abilități practice în implementarea acestora în programe funcționale.