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.6**

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 noţiunilor de bază legate de reţelele de transport;

2. Programarea algoritmului Ford-Fulkerson pentru determinarea fluxului maxim într-o reţea de transport .

**2. Sarcina:**

1. Realizaţi procedura introducerii unei reţele de transport cu posibilităţi de verificare a corectitudinii datelor introduse.

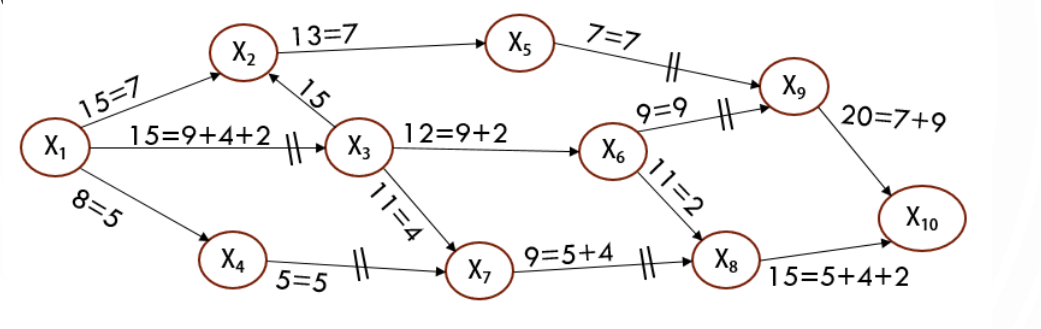
2. În conformitate cu algoritmul Ford-Fulkerson elaboraţi procedura determinării fluxului maxim pentru valori întregi ale capacităţilor arcelor. 3. Realizaţi un program cu următoarele funcţii:

➢ Introducerea reţelei de transport în memorie,

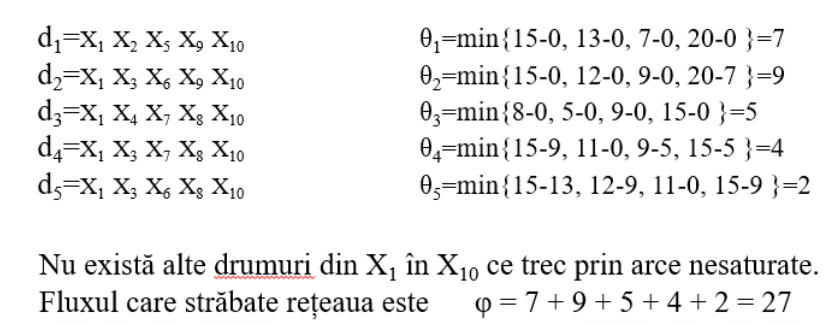
➢ Determinarea fluxului maxim pentru reţeaua concretă;

➢ Extragerea informaţiei la display.

**Fig.1:Graful la Algoritmul Ford-Fulkerson**

****

**Fig.2:Calculul**

****

**Codul:**

#include <stdio.h>

#define A 0

#define B 1

#define C 2

#define MAX\_NODURI 1000

#define O 1000000000

int n = 6;

int capacitate[MAX\_NODURI][MAX\_NODURI];

int flux[MAX\_NODURI][MAX\_NODURI];

int culoare[MAX\_NODURI];

int predecesor[MAX\_NODURI];

int minim(int x, int y) {

  return x < y ? x : y;

}

int cap, coada;

int q[MAX\_NODURI + 2];

void adaugaLaCoada(int x) {

  q[coada] = x;

  coada++;

  culoare[x] = B;// nod in  coada

}

int extrageDinCoada() {

  int x = q[cap];

  cap++;

  culoare[x] = C; // nod extras din nod

  return x;

}

// Folosind BFS ca algoritm de căutare

int bfs(int start, int scop) {

  int u, v;

  for (u = 0; u < n; u++) {

    culoare[u] = A;//Markeaza toate Nodurile cu A ca neatinse

  }

  cap = coada = 0; //Inițializarea capului și cozii la 0.

  adaugaLaCoada(start);//adaug nodul start la coada

  predecesor[start] = -1;

  while (cap != coada) {// Intrare in while atâta timp cât coada nu este goală.

    u = extrageDinCoada();//se extrage un nod din coada

    for (v = 0; v < n; v++) {// for unde trecem prin toate nodurile

      if (culoare[v] == A && capacitate[u][v] - flux[u][v] > 0) { // verifica daca este legatura intre nod u si v si ca nodul v nu a fost vizitat

        adaugaLaCoada(v);

        predecesor[v] = u;

      }

    }

  }

  return culoare[scop] == C;//returneaza C daca nodul a fost vizitat

}

// Aplicarea algoritmului Ford-Fulkerson

int fordFulkerson(int sursa, int scurgere) {

  int i, j, u;

  int fluxMaxim = 0;

  for (i = 0; i < n; i++) {

    for (j = 0; j < n; j++) {

      flux[i][j] = 0;

    }

  }

  // Actualizarea valorilor reziduale ale muchiilor

  while (bfs(sursa, scurgere)) {//se executa acest while atata timp cat este drum de la sursa la scurgere

    int increment = O;//

    for (u = scurgere; predecesor[u] >= 0; u = predecesor[u]) {//Parcurge drumul de la scurgere la sursă și găsește incrementul minim.

      increment = minim(increment, capacitate[predecesor[u]][u] - flux[predecesor[u]][u]);

    }

    for (u = scurgere; predecesor[u] >= 0; u = predecesor[u]) {

      flux[predecesor[u]][u] += increment; //se aduna la o muchie capacitatea or

       flux[u][predecesor[u]] -= increment;//se scade din capacitatea muchilor

    }

    // Adăugarea fluxurilor de pe traseu

    fluxMaxim += increment;

  }

  return fluxMaxim;

}

int main() {

  for (int i = 0; i < n; i++) {

    for (int j = 0; j < n; j++) {

      capacitate[i][j] = 0;

    }

  }

  capacitate[0][1] = 8;

  capacitate[0][4] = 3;

  capacitate[1][2] = 9;

  capacitate[2][4] = 7;

  capacitate[2][5] = 2;

  capacitate[3][5] = 5;

  capacitate[4][2] = 7;

  capacitate[4][3] = 4;

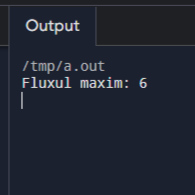
  int sursa = 0, scurgere = 5;

  printf("Fluxul maxim: %d\n", fordFulkerson(sursa, scurgere));

  return 0;

}

# **Outputul Programului**



# **3. Concluzii**

Concluzia acestei lucrări este că obiectivele stabilite au fost îndeplinite cu succes. Prin studierea noțiunilor de bază legate de rețelele de transport și implementarea algoritmului Ford-Fulkerson, a fost posibilă dezvoltarea unui program funcțional pentru gestionarea rețelelor de transport și determinarea fluxului maxim.

Această lucrare oferă o bază solidă pentru înțelegerea și aplicarea conceptelor legate de rețelele de transport în practică. Implementarea algoritmului Ford-Fulkerson într-un program util și eficient reprezintă un instrument valoros pentru rezolvarea problemelor practice asociate gestionării rețelelor de transport și determinării fluxului maxim în aceste rețele.