# PROIECT NR.4:ALGORITMUL LUI DIJKSTRA

A elaborat: Dmitri COJOCARI, gr. IS11Z

A verificat: Vitalie ȚÎCĂU, lector universitar

## 1. Formularea problemei

De realizat un program ce va efectua algoritmul lui Dijkstra pentru un graf dat.

## 2. Studiu preliminar

Algoritmul lui Dijkstra determină drumurile de cost minim dintr-un nod start(nod-sursă), în toate celelalte noduri ale grafului. La fiecare iterație se alege muchia de cost minim.

## 3. Programul

#include <stdio.h>

#include <conio.h>

#include <mem.h>

**int** a[20][20], d[20], i, j, k, n, x, y, min, imin;

**char** c[20], varf[20];

**char** local[20][20] = {"Dmitri", "Balti", "Orhei", "Sangerei", "Rascani", "Ungheni", "Chisinau", "Soroca", "Falesti", "Glodeni", "Drochia"};

**int** main()

{

    printf("Lucrare nr.4");

    printf("\nTema: Algoritmul lui Dijkstra");

    printf("\nElaborat de Cojocari Dmitri");

*//citirea din fisier*

    FILE \*f;

    f = fopen("dijkstra.in", "r");

    if (f == NULL)

    {

        printf("Eroare la deschidere");

        return -1;

    }

*//citim numarul de noduri*

    fscanf(f, "%d", &n);

*//citim matricea de costuri*

    for (i = 1; i <= n; i++)

        for (j = 1; j <= n; j++)

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

*//citim nodul sursa*

    fscanf(f, "%d", &x);

    memset(d, 0, sizeof(d));

    memset(varf, 0, sizeof(varf));

    for (i = 1; i <= n; i++)

    {

        c[i] = 1;

        if (a[x][i])

            varf[i] = x;

    }

*//nodul curent se seteaza cu 0*

    c[x] = 0;

    for (i = 1; i <= n; i++)

        if (c[i])

            d[i] = a[x][i];

    for (y = 1; y <= n; y++)

        if (x != y)

        {

            for (k = 1; k <= n - 2; k++)

            {

                min = 10000;

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

                    if (c[i] && d[i] > 0 && d[i] < min)

                    {

                        min = d[i];

                        imin = i;

                    }

                if (imin == y)

                    break;

                c[imin] = 0;

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

                    if (c[i] && d[imin] != 0 && a[imin][i] != 0)

                        if (d[i] == 0 || d[i] > d[imin] + a[imin][i])

                        {

                            d[i] = d[imin] + a[imin][i];

                            varf[i] = imin;

                        }

            }

            printf("\nDistanta minima intre %s si %s este: %d \n", local[x], local[y], d[y]);

            c[i = 0] = y;

            while (varf[i] != x && varf[c[i]])

                c[++i] = varf[c[i - 1]];

            if (c[i] != x)

                c[++i] = x;

            for (; i >= 0;)

                printf("%s ", local[c[i--]]);

        }

    getch();

    return 0;

}

Fisierul text

dijkstra.in

10

999 86 999 999 62 136 999 999 999 999

999 999 63 999 999 999 110 999 999 999

999 999 999 999 999 999 999 56 999 999

999 999 65 999 999 999 999 999 20 999

62 999 999 103 999 999 999 999 999 999

136 999 999 999 999 999 999 999 173 999

999 999 999 999 999 999 999 999 88 44

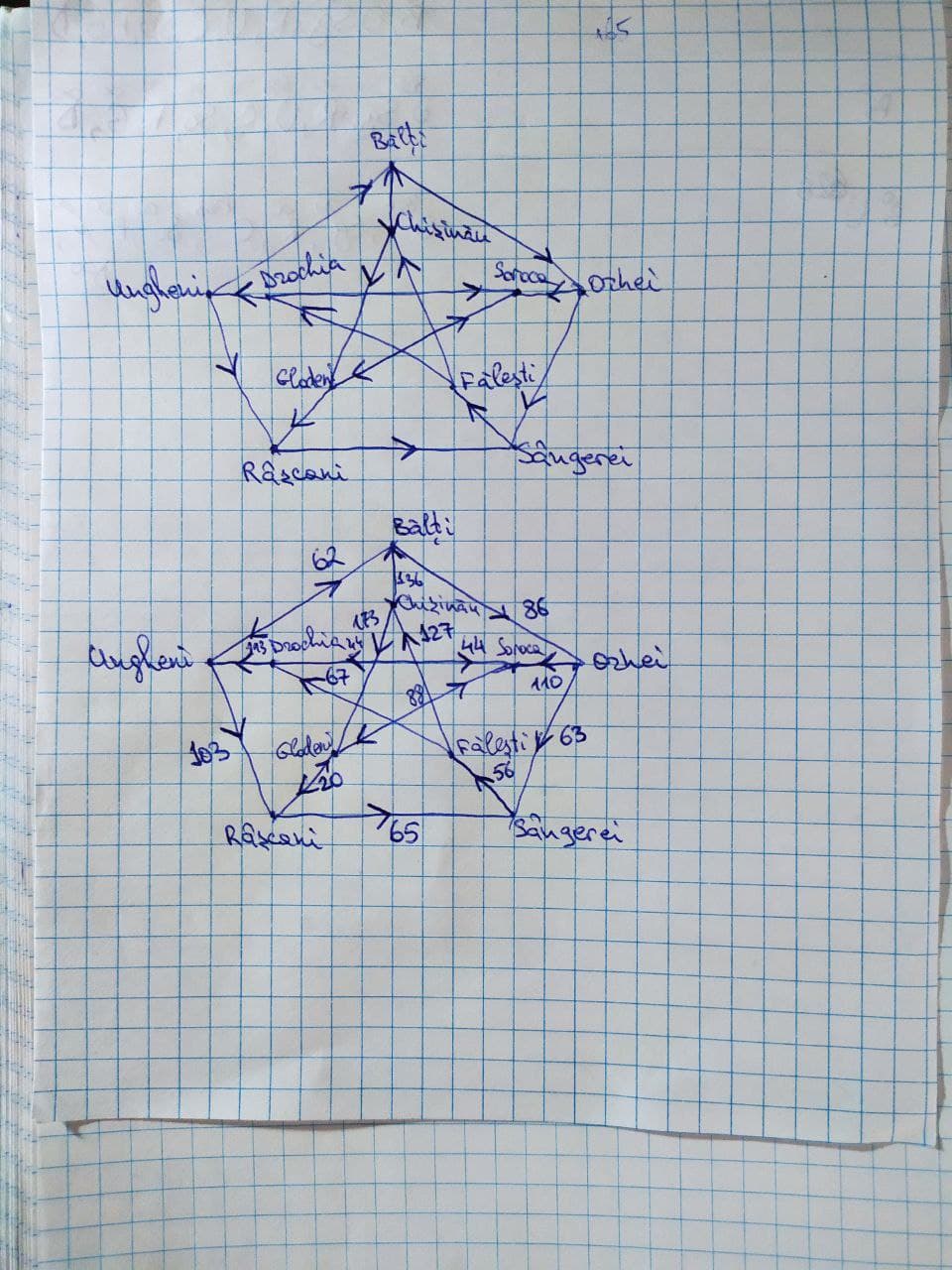
999 999 999 999 999 127 999 999 999 67

999 999 999 20 999 999 88 999 999 999

999 999 999 999 113 999 44 999 999 999

1

Graful corespunzător



## 4. Rezultatele si analiza lor

Lucrare nr.4

Tema: Algoritmul lui Dijkstra

Elaborat de Cojocari Dmitri

Distanta minima intre Balti si Orhei este: 86

Balti Orhei

Distanta minima intre Balti si Sangerei este: 149

Balti Orhei Sangerei

Distanta minima intre Balti si Rascani este: 165

Balti Ungheni Rascani

Distanta minima intre Balti si Ungheni este: 62

Balti Chisinau Balti Ungheni

Distanta minima intre Balti si Chisinau este: 136

Balti Chisinau Balti Chisinau

Distanta minima intre Balti si Soroca este: 196

Balti Orhei Soroca

Distanta minima intre Balti si Falesti este: 205

Balti Orhei Sangerei Falesti

Distanta minima intre Balti si Glodeni este: 185

Balti Ungheni Rascani Glodeni

Distanta minima intre Balti si Drochia este: 240

Balti Orhei Soroca Drochia

## 5. Concluzii

În urma realizării prezentei lucrări, am observat că:

* Prezentul algoritm determină drumul de cost minim dintr-un nod al grafului la toate celelalte noduri ale grafului,
* La fiecare pas se alege drumul de cost minim, deci algoritmul este de tip Greedy,
* Pentru a găsi drumurile minime cu originea într-un alt nod, algoritmul trebuie repornit cu nodul sursă setat ca nodul respectiv.