**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Lucrarea de laborator nr.4

*Metoda programarii dinamice*

A efectuat:

st. gr. SI-211 Vozian Vladimir

A verificat:

asist. univ. Sergiu Scrob

Chişinău 2022

**Scopul lucrării:**

1. Studierea metodei programării dinamice.

2. Analiza şi implementarea algoritmilor de programare dinamică.

3. Compararea tehnicii greedy cu metoda de programare dinamică.

**Codul programului:**

import random

def createGraf(varf, dens):

    G = [[0] \* varf] \* varf

    for i in range(varf):

        for j in range(varf):

            if (j > i):

                r = random.randint(0,100)

                if(r <= dens):

                    r = random.randint(1,10000)

                    G[i][j] = r

                else:

                    G[i][j] = G[j][i]

    for i in range(varf):

        for j in range(varf):

            if(G[i][j] == 0 and i != j):

                G[i][j] = 99999

    return G

def floyd(G,nV):

    iter = 0

    distance = list(map(lambda i: list(map(lambda j: j, i)), G))

    for k in range(nV):

        for i in range(nV):

            for j in range(nV):

                iter += 1

                distance[i][j] = min(distance[i][j], distance[i][k] + distance[k][j])

    print("Floyd Iteratii",iter)

iterdijkstra = 0

def minCost(cost, visited, V):

    global iterdijkstra

    min = 99999

    minIndex = 0

    for i in range(V):

        iterdijkstra += 1

        if(visited[i] == 0 and cost[i] <= min):

            iterdijkstra += 1

            min = cost[i]

            minIndex = i

    return minIndex

def dijkstra(G, V):

    global iterdijkstra

    for i in range(V):

        cost = [99999] \* V

        visited = [0] \* V

        cost[i] = 0

        for i in range(V - 1):

            min = minCost(cost, visited, V)

            visited[min] = 1

            for j in range(V):

                iterdijkstra += 1

                if(visited[j] == 0 and G[min][j] != 0 and cost[min] != 99999 and (cost[min] + G[min][j] < cost[j])):

                    iterdijkstra += 1

                    cost[j] = cost[min]+G[min][j]

    print("Dijkstra Iteratii : ",iterdijkstra)

varf = [10,15,20]

dens = [20,50,100]

for i in dens:

    for j in varf:

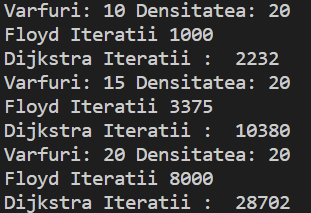
        print("Varfuri:",j,"Densitatea:",i)

        G = createGraf(j,i)

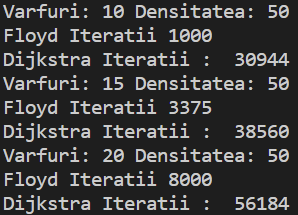
        floyd(G,j)

        dijkstra(G,j)

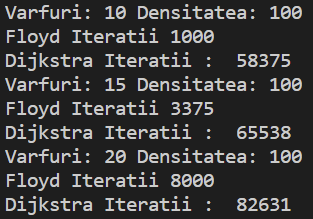
**Rezultatele:**



Densitatea 20, varfuri [10,15,20]



Densitatea 50, varfuri [10,15,20]



Densitatea 100, varfuri [10,15,20]

**Concluzii**

Pentru realizarea acestei lucrări de laborator, am studiat programarea dinamica prin elaborarea a doua algoritmi bazate pe programarea dinamica in problema de a afla drumul minim in un graf. Din rezultate vedem diferenta numarului de iteratii exprimat prin numarul de varfuri si densitatea grafului, in toate cazurile algoritmul Floyd fiind unul mai bun avand un numar cu mult mai mic de iteratii decat algoritmul Dijkstra, indeosebi la un numar mai mare de varfuri si densitate mai mare, acest fapt demonstrand simplitatea lui si aratand pe practica rapiditatea lui expusa in teorie.