**Ministerul Educaţiei Tineretului şi Sportului al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**RAPORT**

**LA MATEMATCA DISCRETA**

Tema: ALGORITMUL DE DETERMINARE A DRUMULUI MINIM/MAXIM

Lucrarea de Laborator nr. 4

A efectuat Studentul grupei \_SI-212\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Vozian Vladimir\_\_\_\_

*semnătura nume, prenume*

A verificat \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*nume, prenume profesor*

**Chisinău 2021**

1. **SCOPUL LUCRĂRII:**

* Studierea algoritmilor de determinare a drumurilor minime și maxime într-un graf.
* Elaborarea programelor de determinare a drumului minim și maxim într-un graf ponderat.

1. **SARCINA DE BAZĂ**
2. Elaboraţi procedura introducerii unui graf ponderat;
3. Elaboraţi procedurile determinării drumului minim;
4. Realizaţi un program cu următoarele funcţii:

* introducerea grafului ponderat cu posibilităţi de analiză sintactică şi semantică şi de corectare a informaţiei;
* determinarea drumului minim;
* extragerea informaţiei la display şi printer (valoarea drumului minim şi succesiunea vârfurilor care formează acest drum).

import networkx as nx

import matplotlib.pyplot as plt

edges\_nr = int(input("Introduceti numarul de muchii=>"))

G = nx.DiGraph()

path = ""

while path != "maxim" or path != "minim":

    print("Introduceti dupa modelul din paranteze (maxim / minim)")

    path = input("Drumul minim sau maxim?(maxim/minim): ").lower()

    if  path == "minim":

        for x in range(edges\_nr):

            G.add\_edge(int(input("Primnul nod=>")),

                       int(input("Nodul adicent=>")),

                       weight = int(input("Ponderea=>")))

            x += 1

    elif path == "maxim":

        for x in range(edges\_nr):

            G.add\_edge(int(input("Primnul nod=>")),

                       int(input("Nodul adicent=>")),

                       weight = -abs(int(input("Ponderea=>"))))

            x += 1

    length = dict(nx.all\_pairs\_bellman\_ford\_path\_length(G))

    pos = nx.circular\_layout(G)

    nx.draw\_networkx(G, pos)

    labels = nx.get\_edge\_attributes(G, ('weight'))

    nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels=labels)

    plt.show()

    break

x = int(input("Nodul de start: "))

y = int(input("Ultimul nod: "))

if path == "maxim":

    print("Drumul maxim din {} in {} ".format(x,y))

    print(f" {x}->{y}: {abs(length[x][y])}")

    pathnodes = dict(nx.all\_pairs\_bellman\_ford\_path(G))

    print(f"Drumul cel mai lung drum este \n{pathnodes[x][y]}")

else:

    print("Drumul minim din {} in {} ".format(x,y))

    print(f" {x}->{y}: {length[x][y]}")

    print("Cele mai scurte drumuri")

    print([p for p in nx.all\_shortest\_paths(G,source=x,target=y,weight='weight')])

## Exemple de executie a programului:

## Graful a fost introdus de la tastatură fiind date nodurile intre care sunt muchiile si ponderea lor, logic introducerea trece asemănător cu algoritmul ford, ca exemplu de graf a fost luat un exemplu rezolvat la cers/seminar:

## Drumul minim:

## 

## Am folosit și o bibliotecă care automat deseneaza graful introdus:

## 

## Deci rezultatul

## Drumul minim din 1 in 8 1->8: 13

## Cele mai scurte drumuri [[1, 2, 4, 8], [1, 5, 8], [1, 3, 7, 8], [1, 2, 4, 6, 8], [1, 5, 6, 8]]

## 

## Deci rezultatul programului coincide cu rezultatul rezolvat la curs.

## Drumul Maxim:

## 

## 

## 

## La fel a fost un exemplu de graf rezolvat la curs.

## Drumul maxim din 1 in 8 1->8: 32

## Drumul cel mai lung drum este [1, 2, 3, 4, 5, 7, 8]

## Concluzii:

Lucrarea dată era efectuată in python, sa exersat asupra cunoștiințelor determinării drumului minim/maxim cu efectuarea lui prin program, am folosit o bibliotecă importată care automat determină și calculează drumul și lungimea lui.