Ministerul Educaţiei, Culturii și Cercetării
Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator nr. 1
la Matematici Speciale

Tema: Păstrarea grafurilor în memoria calculatorului

A efectuat: Cătălin Pleșu
st. gr. TI-206

A verificat: Lisnic Inga

Chișinău – 2021

1. **Scopul și obiectivele lucrării**
    a. Studierea metodelor de definire a unui graf:
        - Matricea de incidență
        - Matricea de adiacență
        - Liste

    b. Elaborarea unor proceduri de introducere, extragere și transformare a diferitelor forme de reprezentare internă a grafurilor cu scoaterea rezultatelor la display și imprimantă.
2. **Sarcina lucrării**
    a. Elaborarea a unui program care să permită introducerea unui graf arbitrar.

    b. Afișarea de la ecran a trei forme de stocare a grafului:
        - Matrice de incidență
        - Matrice de adiacență
        - Listă de adiacență

3. Întrebări de control + oleacă de teorie

Se numește graf, G=(X,U), ansamblu format dintr-o mulțime finită X si o aplicație U a lui X în X.

Elementele mulțimii X se numesc vârfurile grafului. Perechea de vârfuri (x,y) se numește arc, vârful x se numeste originea sau extremitatea inițială a arcului (x,y) iar vârful y se numește extremitatea finală sau terminală.

Dacă un vârf nu este extremitatea nici unui arc el se numeste vârf izolat, iar dacă este extremitatea a mai mult de două arce- nod. Un arc (x,y) pentru care extremitatea inițială coincide cu cea finală se numește buclă.

1. Care sunt metodele de bază de reprezentare a unui graf?

Exista 3 metode de baza de definire a unui graf:
1. Matricea de incidență;
2. Matricea de adiacentă;
3. Lista de adiacență.

2. Descrieţi fiecare din aceste metode.

Matricea de incidenţă

Este o matrice de tipul mxn, în care m este numărul de muchii sau arce (pentru un graf orientat), iar n este numărul vârfurilor. La intersecţia liniei i cu coloana j se vor considera valori de 0, 1 sau -1 în conformitate cu următoarea regulă:

- 1 - dacă muchia/arcul i "intră" în vârful j în cazul unui graf orientat;
- 0 - dacă muchia (arcul) i şi vârful j nu sunt incidente;
- -1 - dacă arcul i "iese" din vârful j.

MI	x_1	x_2	x_3	x_4	x_5
u_1	0	1	0	0	-1
u_2	1	0	0	0	-1
u_3	-1	1	0	0	0
u_4	0	0	1	0	-1
u_5	0	0	-1	1	0
u_6	0	1	-1	0	0
u_7	0	-1	0	1	0

Matricea de adiacenţă

Este o matrice pătrată nxn, aici n este numărul de vârfuri. Fiecare element poate fi 0, dacă vârfurile respective nu sunt adiacente, sau 1, în caz contrar.

MA	x_1	x_2	x_3	x_4	x_5
x_1	0	1	0	0	0
x_2	0	0	0	1	0
x_3	0	1	0	1	0
x_4	0	0	0	0	0
x_5	1	1	1	0	0

Lista de adiacenţă

Lista de adiacenţă este o listă cu n linii (după numărul de vârfuri n), în linia cu numărul i vor fi scrise numerele vârfurilor adiacente cu vârful i.

Xi	F(xi)
x_1	2,0
x_2	4,0
x_3	2,4,0
x_4	0
x_5	1,2,3,0

3. Cum se vor realiza aceste metode în limbajul de programare ales de tine?

Aceste metode au fost realizate în programul de la punctul 4.
Matricile în python sunt destul de elementare însă în urma unor cercetări am ajuns la concluzia că am nevoie de librăria **numpy**.
Matricea de incidență este o matrice **n**x**m** unde n este numărul de muchii iar m este numărul de vârfuri.
Matricea de adiacență este o matrice **m**x**m** unde m aceiași variabila ca mai sus este numărul de vârfuri.
Cu ajutorul librăriei menționate anerior nu a fost o problemă declararea sau modificarea matricelor.
Lista de adiacență este ce-a mai comodă metodă din toate deoarece ea conține doar legăturile dintre vârfuri adica nu are zerouri. Graful realizat în python este sub forma unui dicționar unde cheia este vârful inițial iar ei îi este atribuită o listă de vârfuri care sunt vârfurile terminale ale mârfului inițial.

4.Codul programului

```
# librarii care vor desena graful
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import json

fisier = "secret.txt"


def readIncidenceMatrix():
    print("citirea matricii de incidenta")
    print("dati numarul de varfuri :")
    x = int(input())
    matrice = np.zeros([x, x])
    n, m = (x, x)
```

```
    iese = "ca nu"
    i = 0
    while True:
        print("daca doriti sa finalizati tastati q")
        print("aveti la dispozitie", x,
              "varfuri ( !!!nu esiti din aceasta limita, pe urma veti avea pois
bilitatea de a modifica ceva ). \nMuchia,", i+1, ":")
        print("extrema initiala :")
        iese = input()
        if iese == "q":
            break
        if i == n:
            n += 1
            matrice = np.resize(matrice, (n, m))
            for j in range(0, m):
                matrice[i][j] = 0
        iese = int(iese)-1
        print("extrema terminala :")
        intra = int(input())-1
        matrice[i][iese] = -1
        matrice[i][intra] = 1
        if iese == intra:
            matrice[i][intra] = 2
        i += 1
    return (matrice, n, m)


def readAdjacencyMatrix():
    print("citirea matricii de adiacenta")
    print("dati numarul de varfuri :")
    x = int(input())
    matrice = np.zeros([x, x])
    muchia = 1
    for i in range(0, x):
        y = "text fara un scop anume"
        while True:
            print("\naveti la dispozitie", x,
                  "varfuri ( !!!nu esiti din aceasta limita, pe urma veti avea
poisbilitatea de a modifica ceva )\n. Muchia ,", muchia, ":")
```

```
            print("daca doriti sa treceti la urmatorul varf tastati n")
            print("daca doriti sa finisati tastati q")
            print("arcul cu extremitatea initiala",
                  i+1, "are extremitatea terminala :")
            y = input()
            if (y == "n" or y == "q"):
                break
            y = int(y)-1
            matrice[i][y] = 1
            muchia += 1
        if (y == "q"):
            break
    return (matrice, x)


def readAdjacencyList():
    print("citirea liseti de  adiacenta")
    G = {}
    e = "dummy text"
    a = "dummy text"
    n = 1
    while True:
        print("pentru a termina tastati q")
        print("dati varful initial ( poate fi orice numar pozitiv) se recomanda
 sa fie", n, " :")
        e = input()
        if e == "q":
            break
        # initiez o lista goala
        e = int(e)
        G[e] = []
        while True:
            print("pentru a trece la varful urmator tastati n")
            print("dati un element din lista de adiacenta pentru varful", e)
            a = input()
            if a == "n" or a == "q":
                break
            G[e].append(int(a))
        # este specific pentru python dar a exista si aici jos :/
```

```
        if a == "q":
            break
        n += 1
    return G


# cu ajutorul librariei importate va desena grahul ( foarte interesant )
def drawList(G):
    g = nx.DiGraph()
    for i in [*G]:
        for j in G[i]:
            g.add_edge(i, j)
    # deoarece varful 0 este adugat automat si conectat cu toate celelalte varf
uri
    if g.has_node(0):
        g.remove_node(0)
    nx.draw(g, with_labels=True)
    plt.draw()
    plt.show()
    return


def incidenceToAdjacency(matrix, n, m):
    # deoarece matricea adiacenta este una patrata
    matrixA = np.zeros([m, m])
    for i in range(0, n):
        for j in range(0, m):
            if matrix[i][j] == -1:
                # deoarece am nevoie sa pastrez randul pe care va trebui sa pun
 1
                a = j
            if matrix[i][j] == 1:
                # ca sa aflu in ce coloana sa pun acest 1
                b = j
            # deoarece daca avem o bucla aceasta nu va fi identificata
            elif matrix[i][j] == 2:
                a, b = (j, j)
        matrixA[a][b] = 1
    return (matrixA, m)
```

```
def listToIncidence(g):
    # deoarece mai jos am nevoie de aceste variabile initializate
    m = 0
    n = 0
    for i in [*g]:
        if int(i) > m:
            m = int(i)
            for j in g[i]:
                if int(j) > m:
                    m = int(j)
    # deoarece aceasta matrice poate avea oricare numar de linii
    matrix = np.zeros([1, m])
    for i in [*g]:
        for j in g[i]:
            # redimensionez aceasta matrice de fiecare data
            # de notat ca n la inceput este 0 iar prima redimensionare nu are n
ici un efect
            n += 1
            matrix = np.resize(matrix, (n, m))
            # am nevoie ca toate elementele de pe rand la inceput sa fie 0
            for z in range(0, m):
                matrix[n-1][z] = 0
            # n-1 - linia curenta
            # i-
1 - este varful din care iese muchia considerand ca matricea in memorie se ince
pe de la 0
            matrix[n-1][int(i)-1] = -1
            if i != j:
                matrix[n-1][int(j)-1] = 1
            else:
                matrix[n-1][int(j)-1] = 2
    return (matrix, n, m)


def adjacencyToList(matrix, n):
    # initiez cutia chestia asat grozava
    megaList = {}
```

```
    for i in range(0, n):
        # initiez o lista
        megaList[i+1] = []
        for j in range(0, n):
            if matrix[i][j] == 1:
                megaList[i+1].append(j+1)
    return megaList


def removeVertex(g):
    print(g)
    print("varful pe care doriti sa il stergeti :")
    v = int(input())
    for i in [*g]:
        if v in g[i]:
            g[i].remove(v)
    g.pop(v, None)
    return g


def removeEdge(g):
    print(g)
    print("varful din care iese muchia :")
    e = int(input())
    print("varful in care intra muchia :")
    i = int(input())
    if i in g[e]:
        g[e].remove(i)
    else:
        print("\nnu exista asa muchie\n")
    return g


def addEdge(g):
    print(g)
    print("puteti adauga orice muchie (chiar cu varfuri noi)")
    print("varful din care iese muchia :")
    e = int(input())
    print("varful in care intra muchia :")
    i = int(input())
```

```
    if e in g:
        print("varful va fi adaugat")
    else:
        g[e] = []
    g[e].append(i)
    return g


def edit(g):
    print("puteti :\nsterge un varf - v\nsterge o muchie - m\nadauga o muchie -
 a")
    o = input()
    if o == "v":
        g = removeVertex(g)
    elif o == "m":
        g = removeEdge(g)
    elif o == "a":
        g = addEdge(g)
    return g


def main():

    print("Lucrarea de laborator nr 1\nla Matematici speciale ")
    print("Daca veti dori sa iesiti din program scrieti - iSurrender sau qqq\nd
aca veti uita tastati ctrl + c sau restartati pc-ul")
    G = {}
    print("programul dat opereaza cu grafuri le ptueti citi; printa, edita, chi
ar si sa le vedeti in forma grafica \n( pentru ca nu am imaginatie sa mi le ima
ginez )")
    while True:
        print("\n\n")
        print("citirea din fisier ( r ) ; scrierea in fisier ( w )")
        print("citire matrice de incidenta tastati ( i ) ; printare ( ii )")
        print("matrice de adiacenta ( a ) ; printare ( aa )")
        print("lista de adiacenta ( l ) ; printare ( ll )")
        print("pentru a edita graful ( e )")
        print("bonus ( b )")
        print("( f )")
```

```
        o = input()

        if o == "w":
            haltura = np.zeros([m, m])
            for k in [*G]:
                haltura[k-1][0] = k
                j = 1
                for i in G[k]:
                    haltura[k-1][j] = i
                    j += 1
            np.savetxt("lista.txt", haltura, fmt="%d ")
            json.dump(G, open(fisier, 'w'))
        if o == "r":
            G = json.load(open(fisier))
            # convertirea in celelalte tipuri de matrice
            incidenceMatrix, n, m = listToIncidence(G)
            adjacencyMatrix, m = incidenceToAdjacency(incidenceMatrix, n, m)
            # pentru ca , cand  cheia se citeste din fisier automat este caract
er

            G = adjacencyToList(adjacencyMatrix, m)
            # convertirea in celelalte tipuri de matrice
        if o == "i":
            incidenceMatrix, n, m = readIncidenceMatrix()
            # convertirea in celelalte tipuri de matrice
            adjacencyMatrix, m = incidenceToAdjacency(incidenceMatrix, n, m)
            G = adjacencyToList(adjacencyMatrix, m)
            # convertirea in celelalte tipuri de matrice
            print(incidenceMatrix)
            drawList(G)
        if o == "a":
            adjacencyMatrix, m = readAdjacencyMatrix()
            # convertirea in celelalte tipuri de matrice
            G = adjacencyToList(adjacencyMatrix, m)
            incidenceMatrix, n, m = listToIncidence(G)
            # convertirea in celelalte tipuri de matrice
            print(adjacencyMatrix)
            drawList(G)
        if o == "l":
            G = readAdjacencyList()
```

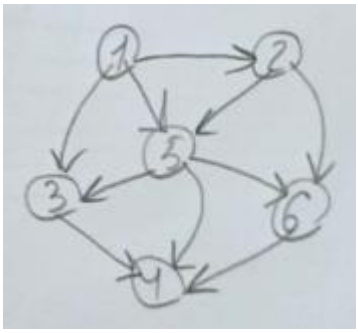
```
            # convertirea in celelalte tipuri de matrice
            incidenceMatrix, n, m = listToIncidence(G)
            adjacencyMatrix, m = incidenceToAdjacency(incidenceMatrix, n, m)
            # convertirea in celelalte tipuri de matrice
            print(G)
            drawList(G)
        if o == "ii":
            print("", end="     ")
            for i in range(0, m):
                print("X"+str(i+1), end="  ")
            print("\n")
            for i in range(0, n):
                print("U"+str(i+1), end="  ")
                for j in range(0, m):
                    print(int(incidenceMatrix[i][j]), end='   ')
                print("\n")
        if o == "aa":
            print("", end="   ")
            for i in range(0, m):
                print("X"+str(i+1), end="  ")
            print("\n")
            for i in range(0, m):
                print("X"+str(i+1), end="  ")
                for j in range(0, m):
                    print(int(adjacencyMatrix[i][j]), end='   ')
                print("\n")
        if o == "ll":
            for k in [*G]:
                print(k, '-', end=" ")
                for i in G[k]:
                    print(i, end=" ")
                print(0)
        if o == "e":
            G = edit(G)
            # convertirea in celelalte tipuri de matrice
            incidenceMatrix, n, m = listToIncidence(G)
            adjacencyMatrix, m = incidenceToAdjacency(incidenceMatrix, n, m)
            # convertirea in celelalte tipuri de matrice
        if o == "b":
```

```
            drawList(G)
        if o == "f":
            print(incidenceMatrix)
            print(adjacencyMatrix)
            print(G)
        if o == "iSurrender" or o == "qqq":
            break
    return


#  apeleaza functia principala
```

5.Executarea programului

Graficul folosit la testarea programului



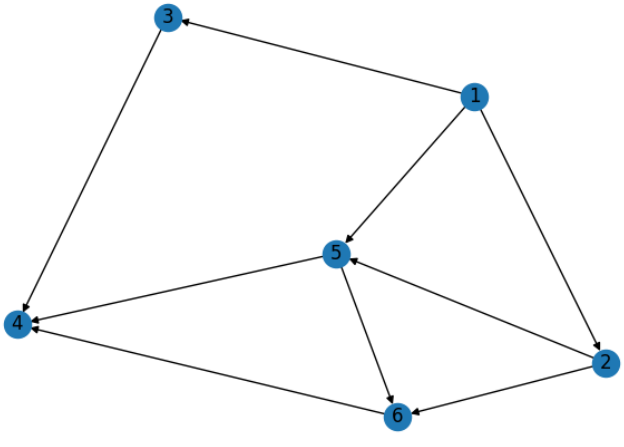
Meniul principal al programului

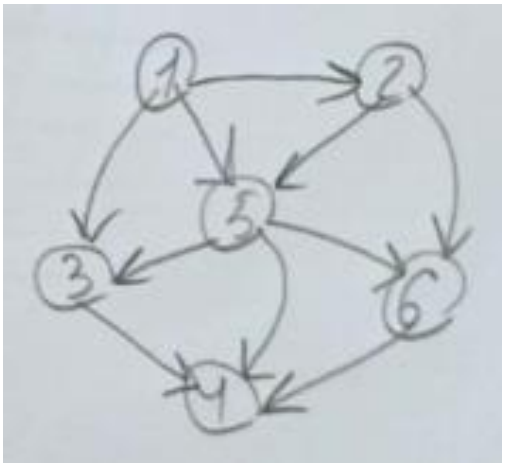
```
Lucrarea de laborator nr 1
la Matematici speciale
Daca veti dori sa iesiti din program scrieti - iSurrender sau qqq
daca veti uita tastati ctrl + c sau restartati pc-ul
programul dat opereaza cu grafuri le ptueti citi; printa, edita, chiar si sa le vedeti in forma grafica
( pentru ca nu am imaginatie sa mi le imaginez )



citirea din fisier ( r ) ; scrierea in fisier ( w )
citire matrice de incidenta tastati ( i ) ; printare ( ii )
matrice de adiacenta ( a ) ; printare ( aa )
lista de adiacenta ( l ) ; printare ( ll )
pentru a edita graful ( e )
bonus ( b )
( f )
```

Am introdus o matrice de incidenta insa am omis o muchie din greseala



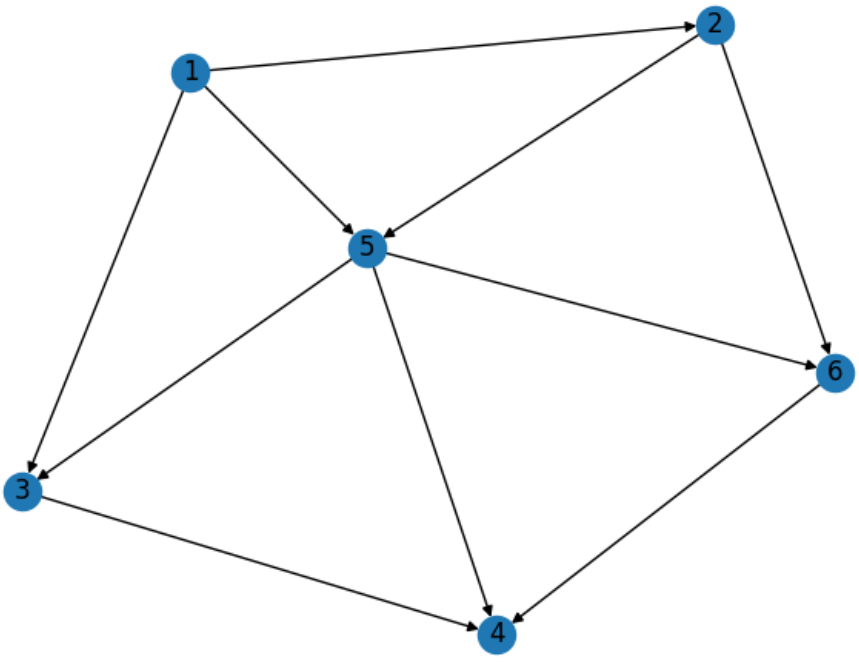


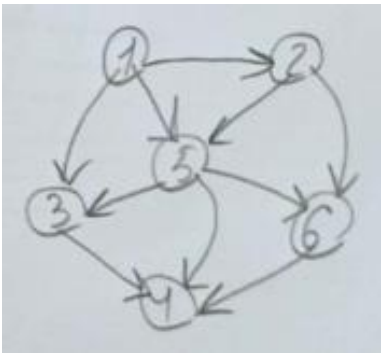
Arcul 5 -> 3 fiind cel omis

```
4
daca doriti sa finalizati tastati q
aveti la dispozitie 6 varfuri ( !!!nu esiti din aceasta limita, pe urma veti avea poisbilitatea de a modifica ceva ).
Muchia, 7 :
extrema initiala :
5
extrema terminala :
4
daca doriti sa finalizati tastati q
aveti la dispozitie 6 varfuri ( !!!nu esiti din aceasta limita, pe urma veti avea poisbilitatea de a modifica ceva ).
Muchia, 8 :
extrema initiala :
5
extrema terminala :
6
daca doriti sa finalizati tastati q
aveti la dispozitie 6 varfuri ( !!!nu esiti din aceasta limita, pe urma veti avea poisbilitatea de a modifica ceva ).
Muchia, 9 :
extrema initiala :
6
extrema terminala :
4
daca doriti sa finalizati tastati q
aveti la dispozitie 6 varfuri ( !!!nu esiti din aceasta limita, pe urma veti avea poisbilitatea de a modifica ceva ).
Muchia, 10 :
extrema initiala :
q
[[-1.  1.  0.  0.  0.  0.]
 [-1.  0.  1.  0.  0.  0.]
 [-1.  0.  0.  0.  1.  0.]
 [ 0. -1.  0.  0.  1.  0.]
 [ 0. -1.  0.  0.  0.  1.]
 [ 0.  0. -1.  1.  0.  0.]
 [ 0.  0.  0.  1. -1.  0.]
 [ 0.  0.  0.  0. -1.  1.]
 [ 0.  0.  0.  1.  0. -1.]]
```

Apoi am adăugat vârful muchia care lipsea.

```
{1: [2, 3, 5], 2: [5, 6], 3: [4], 4: [], 5: [4, 6], 6: [4]}
puteti adauga orice muchie (chiar cu varfuri noi)
varful din care iese muchia :
5
varful in care intra muchia :
3
varful va fi adaugat
```





Trebuie mentionat că programul desenează acest graf.

Afișarea grafului la ecran

Matrice de incidență

```
      X1   X2   X3   X4   X5   X6

U1    -1    1    0    0    0    0

U2    -1    0    1    0    0    0

U3    -1    0    0    0    1    0

U4    0    -1    0    0    1    0

U5    0    -1    0    0    0    1

U6    0    0    -1    1    0    0

U7    0    0    0    1    -1    0

U8    0    0    0    0    -1    1

U9    0    0    1    0    -1    0

U10   0    0    0    1    0    -1
```

Matrice de adiacență

```
    X1   X2   X3   X4   X5   X6

X1  0    1    1    0    1    0

X2  0    0    0    0    1    1

X3  0    0    0    1    0    0

X4  0    0    0    0    0    0

X5  0    0    1    1    0    1

X6  0    0    0    1    0    0
```

Listă de adiacență

```
1 - 2 3 5 0
2 - 5 6 0
3 - 4 0
4 - 0
5 - 4 6 3 0
6 - 4 0
```

Lista de adiacenta poate fi pastrata in memoria externă

```
1  2  3  5  0  0
2  5  6  0  0  0
3  4  0  0  0  0
4  0  0  0  0  0
5  3  4  6  0  0
6  4  0  0  0  0
```

Acestă listă de adiacență este generată doar in pentru a putea fi printată (adică nu știu cum să o citesc înapoi)

Însă cu ajutorul acestui fișier text pot și citi înapoi lista de adiacență

```
{"1": [2, 3, 5], "2": [5, 6], "3": [4], "4": [], "5": [3, 4, 6], "6": [4]}
```

Mi se pare că e chiar mai comod

Pentru a demonstra posibilitățile de editare am șters vârfurile 5 și 6

```
1 - 2 3 0
2 - 0
3 - 4 0
4 - 0
```

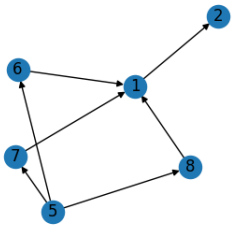
Apoi am șters muchia 1 -> 3

```
1 - 2 0
2 - 0
3 - 4 0
4 - 0
```

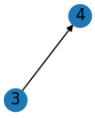
În ultimul rând voi adăuga niște muchii aleatori

```
1 - 2 0
2 - 0
3 - 4 0
4 - 0
5 - 6 7 8 0
6 - 1 0
7 - 1 0
8 - 1 0
```

și am arătat graful :



Care aparent nu era conectat



6.Concluzie:

Pentru că am efectuat acest laborator am acumulat cunoștințe despre grafuri, matrici de incidență, adiacență și listă de adiacență. Am Învățat un limbaj nou de programare, Python ceea ce nu ar fi fost posibil fără un motiv anume. Am ales python pentru că sunt o persoană care este atrasă de lucrurile vizuale și știam că in python cel mai probabil există o bibliotecă care să poată să îmi deseneze graful, evident că am avut dreptate și librăria respectivă a fost utilizată în program.

Efectuând lucrarea am creat niște algoritmi de a transforma graful în toate formele de reprezentare. Cred că nu este atât de eficinet să citesc graful în 3 moduri diferite luând în considerare că oricum pot folosi funcțiile de transformare, ceea ce am și facut.

Cred că nu este tare complicat de a îmbunătăți programul pentru a putea aplica alogritmul Hamiltonian.