Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

з практикуму з дискретних структур № 2

на тему :

„ Характеристики графів ”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | *ІП-61 Кушка Михайло Олександрович* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | *Гавриленко О. В.* |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2017

ЗМІСТ

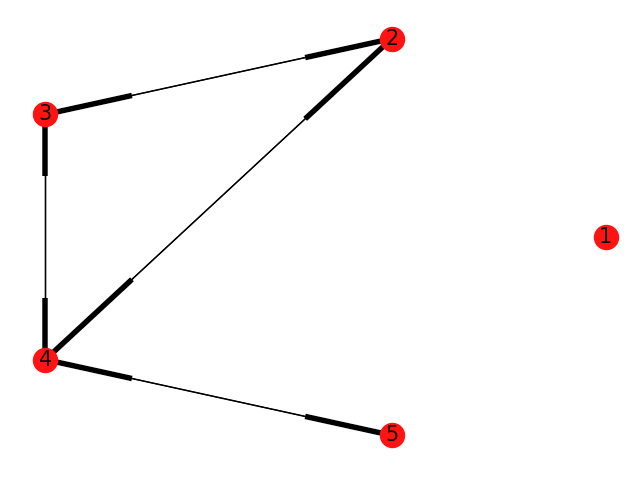
1. постановка задачі 3
2. задані Графи 4
3. робота програми 5
4. тексти програмного коду 6

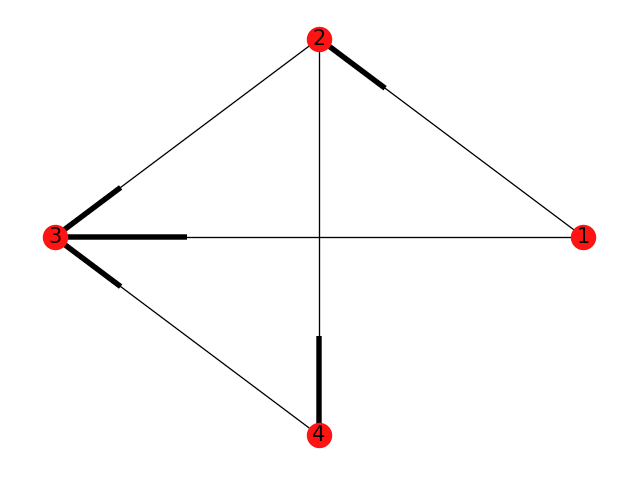
# постановка задачі

Реалізувати програмне застосування (програму), яке виконує наступні функції. Причому на вхід програми подається вхідний файл з описом графу, зі структурою, яка вказана у практичному завданні No1 «Представлення графів».

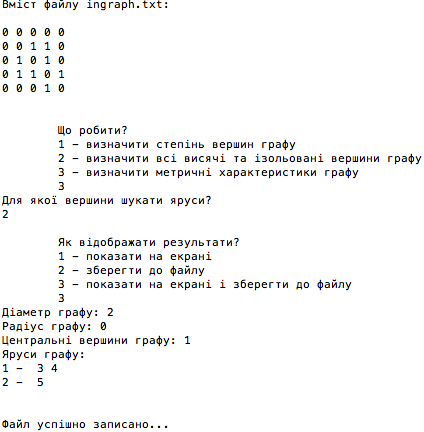
1. Визначити степінь вершин графу. За запитом користувача програма на екран та/або у файл виводить степінь усіх вершин графу (напівстепені виходу та заходу). Визначити, чи графу є однорідним та якщо так, то вказати степінь однорідності графу.
2. Визначити всі висячі та ізольовані вершини. За запитом користувача програма на екран виводить перелік усіх висячих та ізольованих вершин графу.
3. Визначення метричних характеристик графу. Програма виводить наступні характеристики:
   1. Діаметр графу
   2. Радіус графу
   3. Центр графу
   4. Яруси графу із переліком вершин, які входять до кожного ярусу

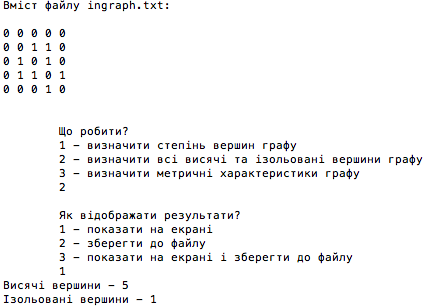
# Задані графи





# робота програми





# тексти програмного коду

import numpy as np

import matplotlib.pyplot as plt

import networkx as nx

def output\_file\_content():

""" Displays file content and sets graphs list """

# get file content

file = open("input/1.txt", "r")

lines = file.readlines()

file.close()

# display lines

print("Вміст файлу ingraph.txt:\n")

for line in lines:

print(line, end="")

print("\n")

# change list for better processing

graph\_matr = []

for i in lines:

i = i.replace("\n", "")

graph\_matr.append(i.split())

return graph\_matr

def strmatr\_to\_intmatr(Z):

""" Try to convert all elements in matrix to int type """

n = len(Z)

new\_Z = []

for i in range(n):

help\_Z = []

for j in range(n):

help\_Z.append(int(Z[i][j]))

new\_Z.append(help\_Z)

return new\_Z

def draw\_graph(Z):

""" Draws matrix """

Z = strmatr\_to\_intmatr(Z)

G = nx.from\_numpy\_matrix(np.array(Z), create\_using=nx.MultiDiGraph())

pos = nx.circular\_layout(G)

nx.draw\_circular(G)

labels = {i : i + 1 for i in G.nodes()}

nx.draw\_networkx\_labels(G, pos, labels, font\_size=15)

plt.show()

def is\_graph\_directed(matr):

""" Check is graph directed or not """

n = len(matr)

number\_of\_edjes = 0

not\_orient = 0

for i in range(n):

for j in range(n):

if matr[i][j] == "1":

number\_of\_edjes += 1

if matr[j][i] == "1":

not\_orient += 1

return (True, False)[number\_of\_edjes == not\_orient]

def degree\_of\_vertexes(matr, is\_directed):

""" Calc degree of every vertex """

n = len(matr)

result = []

if is\_directed:

for i in range(n):

counter1 = 0

counter2 = 0

for j in range(n):

print(matr[i][j])

if matr[i][j] == "1":

counter1 += 1

elif matr[j][i] == "1":

counter2 += 1

result.append(str(i + 1) + " - " + str(counter1) + ", " + str(counter2))

else:

for i in range(n):

counter = 0

for j in range(n):

if matr[i][j] == "1":

counter += 1

result.append(str(i + 1) + " - " + str(counter))

# is graph regular

new\_res = []

for i in result:

new\_res.append(i.split(" - "))

edjes\_in\_1 = new\_res[0][1]

counter = 0

for i in new\_res:

if edjes\_in\_1 == i[1]:

counter += 1

if counter == len(new\_res):

print("Граф однорідний.")

print("Степінь вершини:", edjes\_in\_1)

else:

print("Граф не є однорідним.")

show(result)

def hanging\_and\_isolated\_vertexes(matr):

""" Find hanging and isolated vertexes in the graph """

handle = []

isolated = []

n = len(matr)

for i in range(n):

counter = 0

for j in range(n):

if matr[i][j] == "0":

counter += 1

if counter == n:

isolated.append(str(i + 1))

elif counter == n - 1:

handle.append(str(i + 1))

to\_write = ""

to\_write +="Висячі вершини - "

for i in handle:

to\_write += i + " "

to\_write += "\nІзольовані вершини - "

for i in isolated:

to\_write += i + " "

show(to\_write, "string")

def metric\_parameters(matr, start\_vrt):

""" Calc diameter, radius, center vertexes and storeys of the graph """

n = len(matr)

end\_lst = []

storeys = {}

for k in range(n):

queue = []

black\_list = []

path = {}

queue.append(k)

path[k] = 0

while len(queue) != 0:

parent = queue[0]

for child in range(n):

if matr[parent][child] == "1" and not child in black\_list:

if not child in path:

queue.append(child)

path[child] = path[parent] + 1

black\_list.append(parent)

del queue[0]

del path[k]

if k == start\_vrt - 1:

storeys = path

maximum = 0

for key in path.values():

if key > maximum:

maximum = key

end\_lst.append(maximum)

minimum = 99999

maximum = 0

for i in end\_lst:

if i > maximum:

maximum = i

if i < minimum:

minimum = i

center = ""

for i in range(len(end\_lst)):

if end\_lst[i] == minimum:

center += str(i + 1) + " "

# output results

out = "Діаметр графу: " + str(maximum) + "\n"

out += "Радіус графу: " + str(minimum) + "\n"

out += "Центральні вершини графу: " + str(center) + "\n"

out += "Яруси графу:" + "\n"

# number of storeys for start\_vrt

max\_depth = 0

for i in storeys.values():

if i > max\_depth:

max\_depth = i

# storeys of the graph

for i in range(1, max\_depth + 1):

out += str(i) + " - " + " "#print(i, "-", end = " ")

for key, value in storeys.items():

if value == i:

out += str(key + 1) + " "#print(key, end = " ")

out += "\n"

show(out, "string")

def show(result, mode = "list"):

""" Ask where output result and output result """

def out\_on\_screen(content):

""" Displays content on screen """

if mode == "list":

for i in content:

print(i)

elif mode == "string":

print(content)

def out\_in\_file(content):

""" Save content in the file """

file = open("output.txt", "w")

if mode == "list":

for i in content:

file.write(i + "\n")

elif mode == "string":

file.write(content)

print("\nФайл успішно записано...")

file.close()

choice = None

while True:

choice = input("""

Як відображати результати?

1 - показати на екрані

2 - зберегти до файлу

3 - показати на екрані і зберегти до файлу

""")

if choice == "1":

out\_on\_screen(result)

break

elif choice == "2":

out\_in\_file(result)

break

elif choice == "3":

out\_on\_screen(result)

out\_in\_file(result)

break

else:

print("В меню відсутній пункт", choice)

def main():

# input graph

graph\_matr = output\_file\_content()

is\_directed = is\_graph\_directed(graph\_matr)

choice = None

while True:

choice = input("""

Що робити?

1 - визначити степінь вершин графу

2 - визначити всі висячі та ізольовані вершини графу

3 - визначити метричні характеристики графу

""")

if choice == "1":

degree\_of\_vertexes(graph\_matr, is\_directed)

break

elif choice == "2":

hanging\_and\_isolated\_vertexes(graph\_matr)

break

elif choice == "3":

start\_vertex = int(input("Для якої вершини шукати яруси?\n"))

metric\_parameters(graph\_matr, start\_vertex)

break

else:

print("В меню відсутній пункт", choice)

draw\_graph(graph\_matr)

if \_\_name\_\_ == "\_\_main\_\_":

main()