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 № 1

на тему :

„ Графи ”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. 1. Зчитування графу з вхідного файлу. На вхід подається текстовий файл наступного вигляду: n m

v1 u1

v2 u2

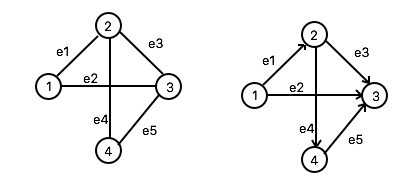
....

vm um

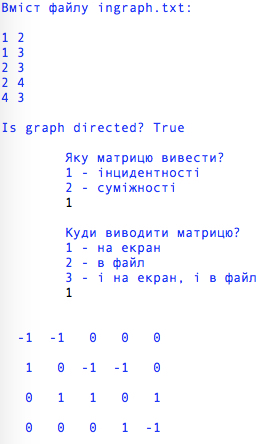
Тут n – кількість вершин графу (ціле число, більше нуля), m – кількість ребер графу (ціле число, більше нуля), vi та ui – початкова та кінцева вершина ребра i (1≤vi≤n, 1≤ui≤n, цілі числа). Індексація вершин у файлі ведеться з 1. Вважається, що граф є орієнтованим. Таким чином можна сказати, що граф задається у файлі списком ребер.

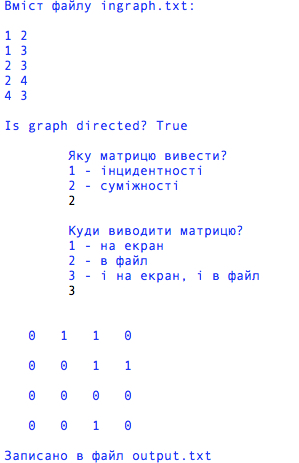
2. Вивід матриць інцидентності та суміжності. За вимогою користувача програма повинна виводити матриці інцидентності та суміжності (окремі функції) на екран та/або у текстовий файл, який вказує користувач.

# Задані графи



# робота програми





# тексти програмного коду

def output\_file\_content():

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

# get file content

file = open("input\_direct.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\_list = []

for i in lines:

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

graph\_list.append(i.split(" "))

return graph\_list

def directed\_graph(graph\_list):

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

counter = 0

for i in range(len(graph\_list)):

a = graph\_list[i][0]

b = graph\_list[i][1]

rev\_line = [b, a]

for line in graph\_list:

if line == rev\_line:

counter += 1

return (True, False)[counter == len(graph\_list)]

def ask\_matr\_questions(n, graph\_list, is\_directed):

""" What matrix create and how it displays """

# ask what matrix create

choice = None

while True:

choice = input("""

Яку матрицю вивести?

1 - інцидентності

2 - суміжності

""")

if choice == "1":

matrix = incidence\_matrix(n, graph\_list, is\_directed)

break

elif choice == "2":

matrix = adjacency\_matrix(n, graph\_list)

nn = n

break

else:

print("В меню немає пункту", choice)

# ask show matrix on screen or save in the file

choice = None

while True:

choice = input("""

Куди виводити матрицю?

1 - на екран

2 - в файл

3 - і на екран, і в файл

""")

if choice == "1":

out\_on\_screen(matrix)

break

elif choice == "2":

out\_in\_file(matrix)

break

elif choice == "3":

out\_on\_screen(matrix)

out\_in\_file(matrix)

break

else:

print("В меню немає пункту", choice)

def number\_of\_vertexes(lst):

""" Calculates number of bertexes in the graph """

maximum = 0

local\_max = 0

for i in range(len(lst)):

local\_max = int(lst[i][0] > lst[i][1] and lst[i][0] or lst[i][1])

maximum = maximum > local\_max and maximum or local\_max

return maximum

def incidence\_matrix(n, graph\_list, is\_directed):

""" Creates incidence matrix """

# check is graph directed

if not is\_directed:

# new list without extra elements

new\_graph = []

for i in range(len(graph\_list)):

a = graph\_list[i][0]

b = graph\_list[i][1]

rev\_line = [b, a]

is\_alone = True

for j in range(i, len(graph\_list)):

if graph\_list[j] == rev\_line:

is\_alone = False

if is\_alone:

new\_graph.append(rev\_line)

graph\_list = new\_graph

# create blank matrix n x nn

nn = len(graph\_list)

matrix = [[0 for row in range(0,nn)] for col in range(0,n)]

# fill matrix

counter = 0

for i in graph\_list:

if is\_directed:

matrix[int(i[0]) - 1][counter] = -1

matrix[int(i[1]) - 1][counter] = 1

counter += 1

else:

matrix[int(i[0]) - 1][counter] = 1

matrix[int(i[1]) - 1][counter] = 1

counter += 1

return matrix

def adjacency\_matrix(n, graph\_list):

""" Creates adjacency matrix """

# create blank matrix n x n

matrix = [[0 for row in range(0,n)] for col in range(0,n)]

# fill matrix

for k in range(len(graph\_list)):

i = int(graph\_list[k][0]) - 1

j = int(graph\_list[k][1]) - 1

matrix[i][j] = 1

return matrix

def out\_on\_screen(m):

""" Displays matrix on screen """

n = len(m)

nn = len(m[0])

print("\n")

for i in range(n):

for j in range(nn):

print("%4i" % m[i][j], end = "")

print("\n")

def out\_in\_file(m):

""" Saves matrix in file """

# create list from matrix to write in file

n = len(m)

nn = len(m[0])

m\_list = []

for i in range(n):

help\_str = ""

for j in range(nn):

#help\_str += str(m[i][j]) + " "

help\_str += str("%4i" % m[i][j])

help\_str += "\n"

m\_list.append(help\_str)

# write list in file

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

file.writelines(m\_list)

file.close()

print("Записано в файл output.txt")

def main():

# show what is in the file and get graphs list

graph\_list = output\_file\_content()

is\_directed = directed\_graph(graph\_list)

print("Is graph directed?", is\_directed)

# number of vertexes

n = number\_of\_vertexes(graph\_list)

# asks questions about matrix

ask\_matr\_questions(n, graph\_list, is\_directed)

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

main()