Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №2

з дисципліни «Дискретна математика»

на тему

«Характеристики графів»

Виконав:

студент гр. ІС-91

Косюк Михайло

Викладач:

доц. Рибачук Л.В.

Київ – 2019

### Зміст

[Зміст 2](#_Toc190845267)

[1 Постановка задачі 3](#_Toc190845268)

[2 Результати виконання програми 4](#_Toc190845269)

[3 Лістинг програми](#_Toc190845271) 5

### 1 Постановка задачі

Ціль

Дослідити характеристики графів та навчитись визначати їх на конкретних прикладах.

Завдання

Реалізувати програмне застосування (програму), яке виконує наступні функції:

**Перша частина:** на вхід програми подається вхідний файл з описом **неорієнтованого** графу, зі структурою, яка вказана у практичному завданні №1 «Представлення графів» (списком ребер)

1. Визначити матриці відстаней та досяжності графу. Програма за запитом користувача виводить на екран та/або у файл матрицю відстаней *D* та матрицю досяжності *R* графу, за алгоритмом піднесення в степінь (конспект стор. 155-156).

2. Визначення метричних характеристик графу. Програма виводить наступні характеристики

a. Діаметр графу

b. Радіус графу

c. Центр графу

d. Яруси графу із переліком вершин, які входять до кожного ярусу

**Друга частина:** на вхід програми подається вхідний файл з описом **орієнтованого** графу, зі структурою, яка вказана у практичному завданні №1 «Представлення графів» (списком ребер)

1. Визначити матриці відстаней та досяжності графу. Програма за запитом користувача виводить на екран та/або у файл матрицю відстаней *D* та матрицю досяжності *R* графу, за алгоритмом піднесення в степінь (конспект стор. 155-156).

2. Визначити тип зв’язності графу. Програма виводить на екран тип зв’язності графу.

Тестовий граф, що використовувався для обробки заданий такою послідовністю(граф, що містить 5 вершин і 7 ребер):

**6 6**

**1 5**

**2 3**

**2 4**

**3 3**

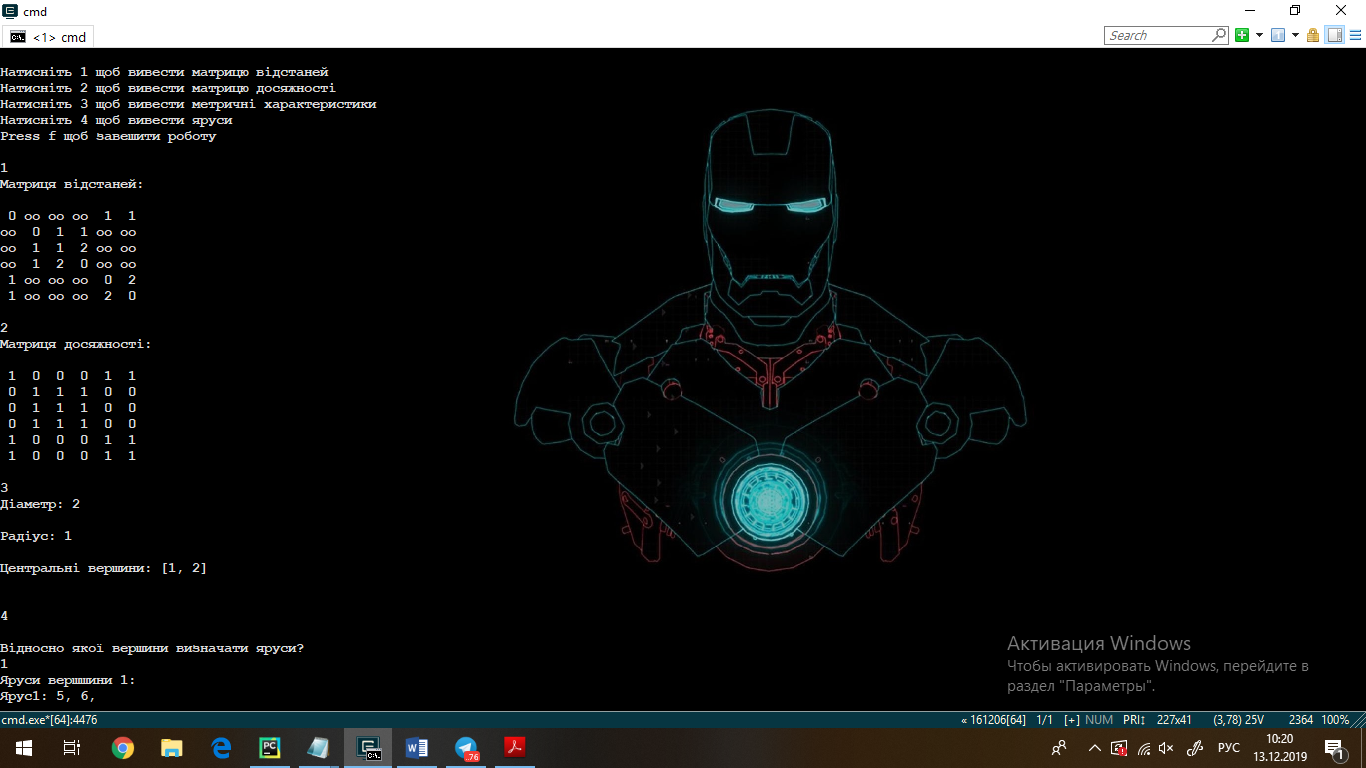
**5 1**

**6 1**

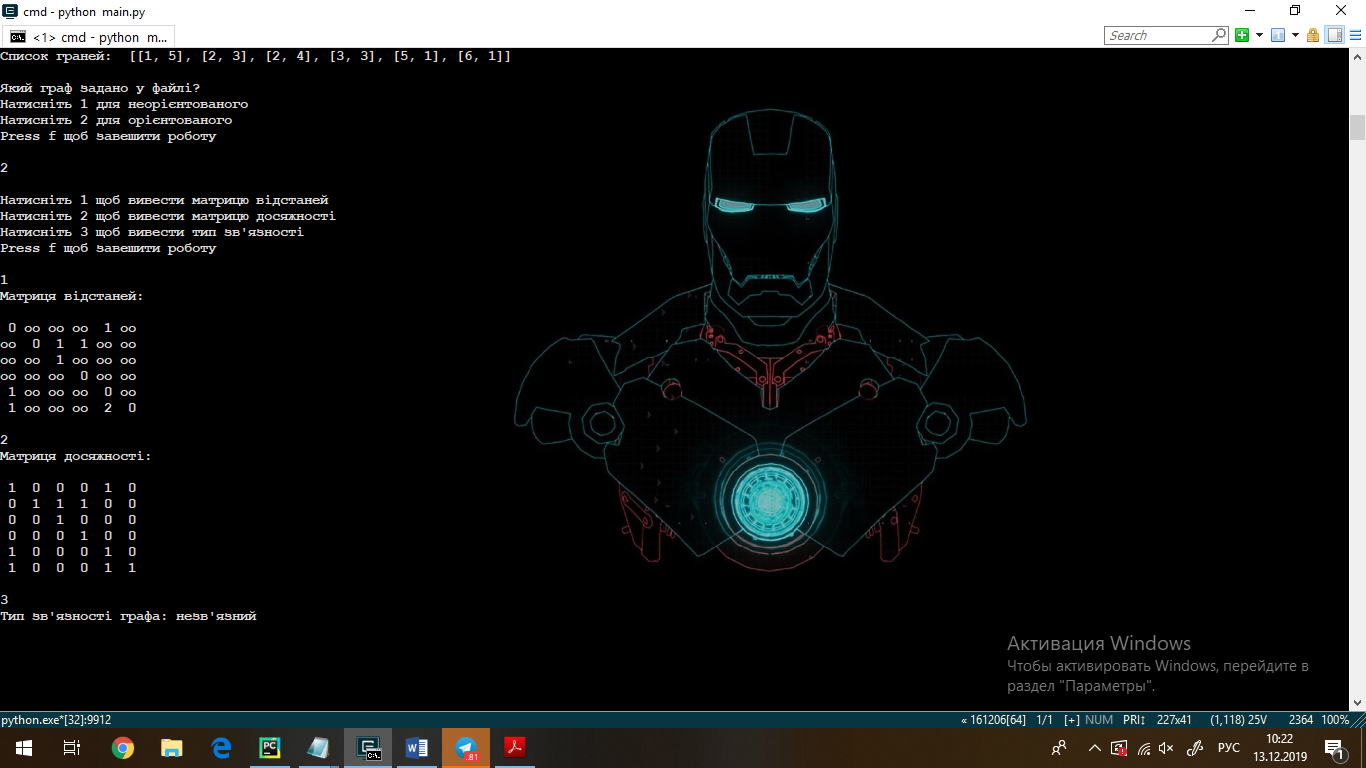
### 

### 2 Результати виконання програми

### Для неорграфа:



1. Для орграфа:



### 3 Лістинг програми

from itertools import islice  
import numpy as np  
  
  
def print\_matrix(mat):  
 for r in mat:  
 for el in r:  
 print("%2s" % el, end=' ')  
 print()  
  
  
def gen\_dist\_matrix(mat, degree, prt):  
 dist\_mat = mat  
 res\_mat = [[0]\*len(mat) for i in range(len(mat))]  
 temp\_mat = mat  
 done = 1  
 while done < degree:  
 for r in range(len(res\_mat)):  
 for d in range(len(res\_mat[r])):  
 for i in range(len(mat)): res\_mat[r][d] += temp\_mat[r][i]\*mat[i][d]  
 temp\_mat = res\_mat  
 for r in range(len(res\_mat)):  
 for d in range(len(res\_mat[r])):  
 if res\_mat[r][d] != 0 and r != d and dist\_mat[r][d] == 0: dist\_mat[r][d] = done + 1  
 res\_mat = [[0]\*len(mat) for i in range(len(mat))]  
 done += 1  
 p\_dist\_mat = [['oo' if dist\_mat[d][t] == 0 and t != d else dist\_mat[d][t] for t in range(len(dist\_mat[d]))] for d in range(len(dist\_mat))]  
 if prt:  
 print("Матриця відстаней: \n")  
 print\_matrix(p\_dist\_mat)  
 return dist\_mat  
  
  
def gen\_reach\_mat(mat, prt):  
 reach\_mat = [[1 if mat[i][j] != 0 or j == i else 0 for j in range(len(mat[i]))] for i in range(len(mat))]  
 if prt:  
 print("Матриця досяжності: \n")  
 print\_matrix(reach\_mat)  
 return reach\_mat  
  
  
def find\_center\_radius(mat):  
 cent = []  
 tops\_exc = []  
 for t in range(len(mat)):  
 tops\_exc.append(max(mat[t]))  
 rad = min(tops\_exc)  
 if tops\_exc.count(rad) > 1:  
 start = tops\_exc.index(rad)  
 for i in range(tops\_exc.count(rad)):  
 c = tops\_exc.index(rad, start)  
 cent.append(c+1)  
 start = tops\_exc.index(rad, c) + 1  
 return rad, cent  
  
  
def find\_diameter(mat):  
 diam = 0  
 for t in mat:  
 if diam < max(t): diam = max(t)  
 return diam  
  
  
def get\_stages(mat):  
 t = int(input('\nВідносно якої вершини визначати яруси?\n'))  
 print("Яруси вершшини " + str(t) + ":")  
 ds = mat[t-1]  
 stages = [[ds.index(ds[d], d)+1 for d in range(len(ds)) if ds[d] == st] for st in range(1, max(ds)+1)]  
 for i in range(len(stages)):  
 print("Ярус" + str(i+1) + ":", end=" ")  
 for j in stages[i]: print(str(j) + ",", end=" ")  
 print()  
  
  
def get\_connect\_type(mat, reach\_mat\_no):  
 type = 0  
 con\_tops = 0  
 for t in reach\_mat\_no:  
 if t.count(1) == len(reach\_mat\_no): con\_tops += 1  
 if con\_tops == len(reach\_mat\_no): type = 1  
 r = np.array(mat)  
 tr = r.transpose()  
 single\_con = r + tr  
 single\_con = single\_con.tolist()  
 ch1 = [1 for r in single\_con if r.count(0) == 0]  
 if len(ch1) == len(single\_con): type = 2  
 ch2 = [1 for r in mat if r.count(1) == len(r)]  
 if len(ch2) == len(mat): type = 3  
 print("Тип зв'язності графа: ", end='')  
 if type == 1: print("слабко зв'язаний")  
 elif type == 2: print("однобічно зв'язаний")  
 elif type == 3: print("сильно зв'язаний")  
 else: print("незв'язний")  
  
  
def gen\_adj\_mat(tops, e\_list, org):  
 adj\_mat = [[0]\*tops for i in range(tops)]  
 for e in e\_list:  
 adj\_mat[e[0]-1][e[1]-1] = 1  
 if not org: adj\_mat[e[1]-1][e[0]-1] = 1  
 return adj\_mat  
  
  
file = open('condition.txt')  
tops\_num = int(file.read(1))  
edges\_num = int(file.read(3))  
edge\_list = []  
for edge in islice(file, 0, edges\_num):  
 ep = []  
 for top in edge.split(' ', 1):  
 ep.append(int(top))  
 edge\_list.append(ep)  
if edges\_num != len(edge\_list):  
 print("!!!Дана програма використовує python модуль numpy, "  
 "\nякщо він не встановлений на вашому комп'ютері виконайте команду"  
 "\npip install numpy!!!\n")  
 print("\nКількість ребер не співпадає у із заданим списком, перевірте файл condition.txt")  
 exit()  
print("Кількість вершин: " + str(tops\_num))  
print("Кількість ребер: " + str(edges\_num))  
print("Список граней: ", edge\_list)  
  
print("\nЯкий граф задано у файлі?"  
 "\nНатисніть 1 для неорієнтованого"  
 "\nНатисніть 2 для орієнтованого"  
 "\nPress f щоб завешити роботу")  
graph\_type = input('\n')  
if graph\_type == '1':  
 adj\_matrix = gen\_adj\_mat(tops\_num, edge\_list, False)  
 dist\_matrix = gen\_dist\_matrix(adj\_matrix, tops\_num, False)  
 print("\nНатисніть 1 щоб вивести матрицю відстаней"  
 "\nНатисніть 2 щоб вивести матрицю досяжності"  
 "\nНатисніть 3 щоб вивести метричні характеристики"  
 "\nНатисніть 4 щоб вивести яруси"  
 "\nPress f щоб завешити роботу")  
 control = ''  
 while control != 'f':  
 control = input('\n')  
 if control == '1': dist\_matrix = gen\_dist\_matrix(adj\_matrix, tops\_num, True)  
 elif control == '2': reach\_matrix = gen\_reach\_mat(dist\_matrix, True)  
 elif control == '3':  
 diameter = find\_diameter(dist\_matrix)  
 print("Діаметр: " + str(diameter) + "\n")  
 radius, centers = find\_center\_radius(dist\_matrix)  
 print("Радіус: " + str(radius) + "\n")  
 print("Центральні вершини: " + str(centers) + "\n")  
 elif control == '4': get\_stages(dist\_matrix)  
elif graph\_type == '2':  
 reach\_mat\_no = gen\_reach\_mat(gen\_dist\_matrix(gen\_adj\_mat(tops\_num, edge\_list, False), tops\_num, False), False)  
 adj\_matrix = gen\_adj\_mat(tops\_num, edge\_list, True)  
 dist\_matrix = gen\_dist\_matrix(adj\_matrix, tops\_num, False)  
 reach\_matrix = gen\_reach\_mat(dist\_matrix, False)  
 print("\nНатисніть 1 щоб вивести матрицю відстаней"  
 "\nНатисніть 2 щоб вивести матрицю досяжності"  
 "\nНатисніть 3 щоб вивести тип зв'язності"  
 "\nPress f щоб завешити роботу")  
 control = ''  
 while control != 'f':  
 control = input('\n')  
 if control == '1': dist\_matrix = gen\_dist\_matrix(adj\_matrix, tops\_num, True)  
 elif control == '2': reach\_matrix = gen\_reach\_mat(dist\_matrix, True)  
 elif control == '3': get\_connect\_type(reach\_matrix, reach\_mat\_no)