**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №3**

з дисципліни

«Алгоритми і структури даних»

Виконав: Перевірив:

Студент групи ІМ-42 Сергієнко А. М.

Лобань Михайло Юрійович

номер у списку групи: 20

Київ 2025

**Загальна постановка завдання**

1. Представити у програмi напрямлений i ненапрямлений графи з заданими параметрами:

* кiлькiсть вершин n;
* розмiщення вершин;
* матриця сумiжностi A.

1. Створити програму для формування зображення напрямленого i ненапрямленого графiв у графiчному вiкнi. Згаданi вище параметри графа задаються на основi чотиризначного номера варiанту n1n2n3n4, де n1n2 це десятковi цифри номера групи, а n3n4 — десятковi цифри номера варiанту, який був у студента для двох попереднiх робiт (див. таблицю з поточними оцiнками з АСД, надану викладачем на початку поточного семестру).

**Завдання за варіантом**

**Варіант 20**

n1n2n3n4 = 4220

Кількість вершин – 10 + 2 = 12

Розміщення вершин – колом, n4 = 0

**Текст програми**

import random

import math

import tkinter as tk

variant = 4220

random.seed(variant)

n3 = 2

n4 = 0

vertexes = n3 + 10

k = 1 - n3 \* 0.02 - n4 \* 0.005 - 0.25

def calculate\_element():

    return math.floor(random.random() \* 2 \* k)

matrix\_dir = [[0 for \_ in range(vertexes)] for \_ in range(vertexes)]

matrix\_undir = [[0 for \_ in range(vertexes)] for \_ in range(vertexes)]

print("\nDirected matrix:\n")

for i in range(vertexes):

    for j in range(vertexes):

        matrix\_dir[i][j] = calculate\_element()

        print(matrix\_dir[i][j], end=" ")

    print()

print("\nUndirected matrix:\n")

for i in range(vertexes):

    for j in range(vertexes):

        matrix\_undir[i][j] = matrix\_dir[i][j] or matrix\_dir[j][i]

        print(matrix\_undir[i][j], end=" ")

    print()

root = tk.Tk()

root.title("Graph")

canvas = tk.Canvas(root, width=800, height=800, bg="white")

canvas.pack()

mid\_x = mid\_y = 400

angle = math.pi \* 2 / vertexes

R = 20

def get\_x(i):

    return mid\_x + math.sin(i \* angle) \* 200

def get\_y(i):

    return mid\_y - math.cos(i \* angle) \* 200

def rotate\_around\_center(x, y, cx, cy, theta):

    x -= cx

    y -= cy

    new\_x = x \* math.cos(theta) - y \* math.sin(theta) + cx

    new\_y = x \* math.sin(theta) + y \* math.cos(theta) + cy

    return new\_x, new\_y

def draw\_graph(matrix, vertexes, is\_directed):

    for i in range(vertexes):

        x = get\_x(i) - R

        y = get\_y(i) - R

        canvas.create\_oval(x, y, x + 2 \* R, y + 2 \* R, fill="white")

        canvas.create\_text(x + R, y + R, text=str(i + 1), font=("Montserrat", 12))

    for i in range(vertexes):

        for j in range(vertexes):

            if matrix[i][j] == 1:

                if i == j:

                    cx, cy = get\_x(i), get\_y(i)

                    theta = i \* angle

                    cx += R \* math.sin(theta)

                    cy -= R \* math.cos(theta)

                    dx = 3 \* R / 4

                    dy = R \* (1 - math.sqrt(7)) / 4

                    p1 = (cx - dx, cy - dy)

                    p2 = (cx - 3 \* dx / 2, cy - R / 2)

                    p3 = (cx + 3 \* dx / 2, cy - R / 2)

                    p4 = (cx + dx, cy - dy)

                    p1 = rotate\_around\_center(p1[0], p1[1], cx, cy, theta)

                    p2 = rotate\_around\_center(p2[0], p2[1], cx, cy, theta)

                    p3 = rotate\_around\_center(p3[0], p3[1], cx, cy, theta)

                    p4 = rotate\_around\_center(p4[0], p4[1], cx, cy, theta)

                    canvas.create\_line(p1[0], p1[1], p2[0], p2[1], width=2)

                    canvas.create\_line(p2[0], p2[1], p3[0], p3[1], width=2)

                    if (is\_directed):

                        canvas.create\_line(p3[0], p3[1], p4[0], p4[1], width=2, arrow=tk.LAST)

                    else:

                        canvas.create\_line(p3[0], p3[1], p4[0], p4[1], width=2)

                else:

                    x1, y1 = get\_x(i), get\_y(i)

                    x2, y2 = get\_x(j), get\_y(j)

                    dx, dy = x2 - x1, y2 - y1

                    length = math.sqrt(dx \*\* 2 + dy \*\* 2)

                    dx /= length

                    dy /= length

                    x1 += dx \* R

                    y1 += dy \* R

                    x2 -= dx \* R

                    y2 -= dy \* R

                    if (is\_directed):

                        canvas.create\_line(x1, y1, x2, y2, width=2, arrow=tk.LAST)

                    else:

                        canvas.create\_line(x1, y1, x2, y2, width=2)

    root.mainloop()

draw\_graph(matrix\_dir, vertexes, 1)

#draw\_graph(matrix\_undir, vertexes, 0)

**Матриці суміжності**

Матриця суміжності напрямленого графа:

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Вміст, створений ШІ, може бути неправильним.

Матриця суміжності ненапрямленого графа:

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Вміст, створений ШІ, може бути неправильним.

**Графи**

Напрямлений граф:

Зображення, що містить ескіз, малюнок, коло, схема

Вміст, створений ШІ, може бути неправильним.

Ненапрямлений граф:

Зображення, що містить ескіз, коло, Симетрія, оригамі

Вміст, створений ШІ, може бути неправильним.

**Висновок**

Написав програму, яка генерує матриці суміжності та візуалізує їх за допомогою бібліотеки tkinter на мові python. Програма використовує генерацію випадкових значень для побудови зв'язків між вершинами графа, які відображаються на площині у вигляді кіл (вершин) та ліній/стрілок (ребер).