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:

Студент групи ІМ-42

Лобань Михайло Юрійович

номер у списку групи: 20

Перевірив:

Сергієнко А. М.

Київ 2025

Загальна постановка завдання

1. Представити у програмі напрямлений і ненапрямлений графи з заданими параметрами:
    - кількість вершин n;
    - розміщення вершин;
    - матриця суміжності A.
2. Створити програму для формування зображення напрямленого і ненапрямленого графів у графічному вікні. Згадані вище параметри графа задаються на основі чотиризначного номера варіанту n1n2n3n4, де n1n2 це десяткові цифри номера групи, а n3n4 — десяткові цифри номера варіанту, який був у студента для двох попередніх робіт (див. таблицю з поточними оцінками 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:

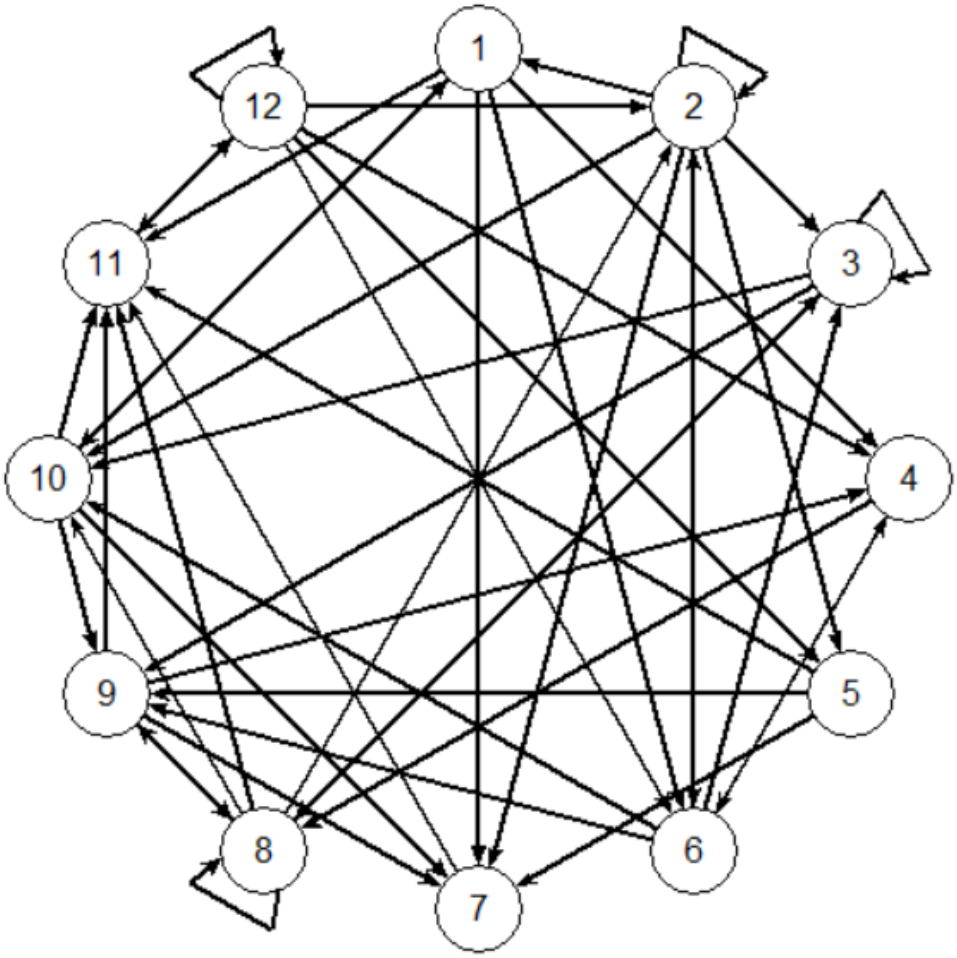
```
Directed matrix:

0 0 0 1 0 1 1 0 0 1 1 0
1 1 1 0 1 1 1 0 0 1 0 0
0 0 1 0 0 0 0 1 1 1 0 0
0 0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 1 0 1 0 1 0
0 1 1 1 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0
0 1 1 0 0 0 0 1 1 1 1 0
0 0 0 1 0 0 1 1 0 0 1 0
1 0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 1
0 1 0 1 1 1 0 0 0 0 1 1
```

Матриця суміжності ненапрямленого графа:

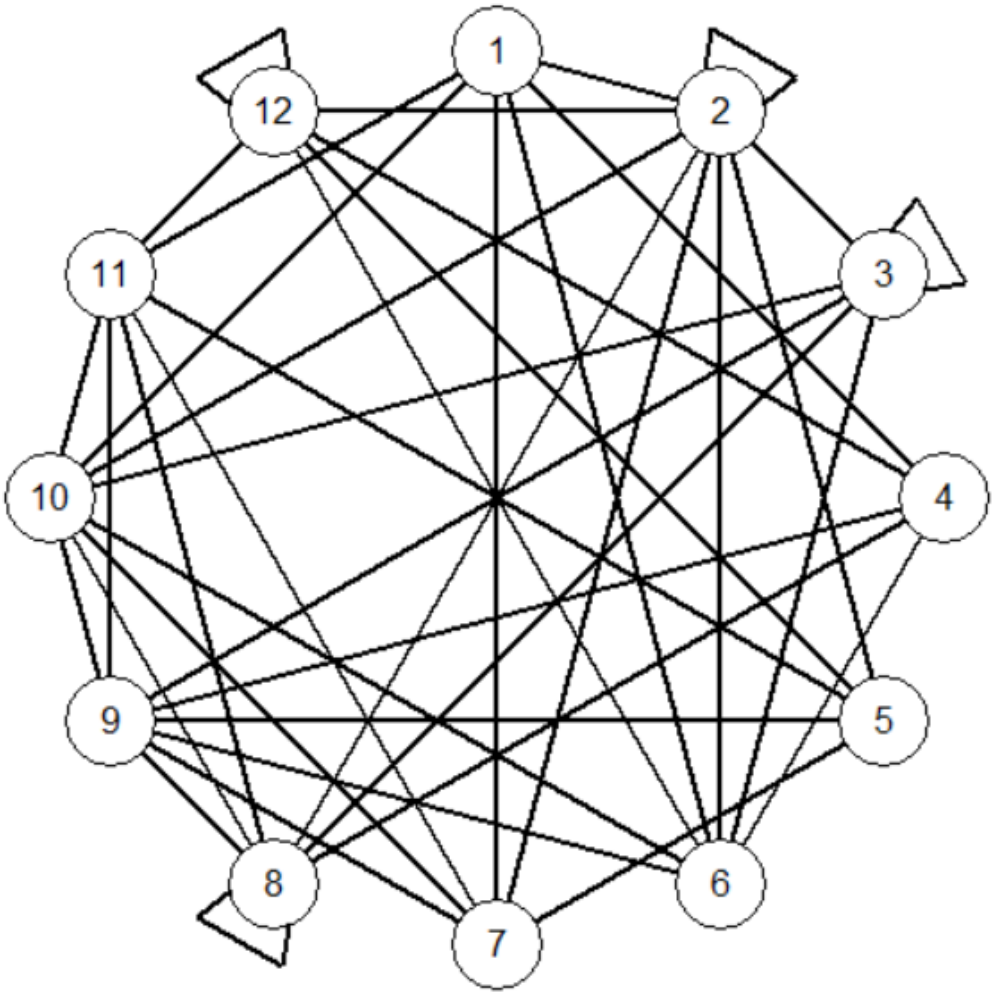
```
Undirected matrix:

0 1 0 1 0 1 1 0 0 1 1 0
1 1 1 0 1 1 1 1 0 1 0 1
0 1 1 0 0 1 0 1 1 1 0 0
1 0 0 0 0 1 0 1 1 0 0 1
0 1 0 0 0 0 1 0 1 0 1 1
1 1 1 1 0 0 0 0 1 1 0 1
1 1 0 0 1 0 0 0 1 1 1 0
0 1 1 1 0 0 0 1 1 1 1 0
0 0 1 1 1 1 1 1 0 1 1 0
1 1 1 0 0 1 1 1 1 0 1 0
1 0 0 0 1 0 1 1 1 1 0 1
0 1 0 1 1 1 0 0 0 0 1 1
```

Графи

Напрямлений граф:



Ненапрямлений граф:



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).