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
Федоренко Іван Русланович
номер у списку групи: 29

Перевірив:

Сергієнко А. М.

Київ 2025

Завдання
Лабораторна робота 3.

Графічне представлення графів

Мета лабораторної роботи

Метою лабораторної роботи No3 «Графічне представлення графів» є набуття практичних навичок представлення графів у комп'ютері та ознайомлення з принципами роботи ОС.

Постановка задачі

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 (у мене група 42), а n3n4 — десяткові цифри номера варіанту (мій варіант 29).

Кількість вершин n дорівнює 10 + n3.

Розміщення вершин:

• колом при n4 = 0, 1;

• квадратом (прямокутником) при n4 = 2, 3;

• трикутником при n4 = 4, 5;

• колом з вершиною в центрі при n4 = 6, 7;

• квадратом (прямокутником) з вершиною в центрі при n4 = 8, 9.

Наприклад, при n4 9 розміщення вершин прямокутником з вершиною в центрі

Матриця суміжності Adir напрямленого графа за варіантом формується таким чином:

1) встановлюється параметр (seed) генератора випадкових чисел, рівний номеру варіанту n1n2n3n4 — детальніше див. с. 12;

2) матриця розміром n*n заповнюється згенерованими випадковими числами в діапазоні [0, 2.0);

3) обчислюється коефіцієнт n3 * 0.02 - n4 * 0.005 - 0.25

4) кожен елемент матриці множиться на коефіцієнт k;

5) елементи матриці округлюються: 0 — якщо елемент менший за 1.0, 1 — якщо елемент більший або дорівнює 1.0.

Матриця суміжності Aundir ненапрямленого графа одержується з матриці Adir:

a(dir i,j) = 1 => (aundir i,j) = 1, aundir j,i) = 1.

Наприклад, якщо запрограмувати додаткові функції randm та mulmr, у програмі мовою С генерація матриці Adir напрямленого графа може виглядати так:

srand(n1n2n3n4);

T = randm(n);

k = 1.0 - n3*0.02 - n4*0.005 - 0.25;

A = mulmr(T, k);

Тут randm(n) — розроблена функція, яка формує матрицю розміром nn, що складається з випадкових чисел у діапазоні (0, 2.0);mulmr(T, k) — розроблена функція множення матриці на коефіцієнт та округлення результату до 0 чи 1.

Код програми:

```
import numpy as np

import matplotlib.pyplot as plt

import networkx as nx

import math


class GraphVisualizer:

    def __init__(self):

        self.n1 = 4

        self.n2 = 2

        self.n3 = 2

        self.n4 = 9
```

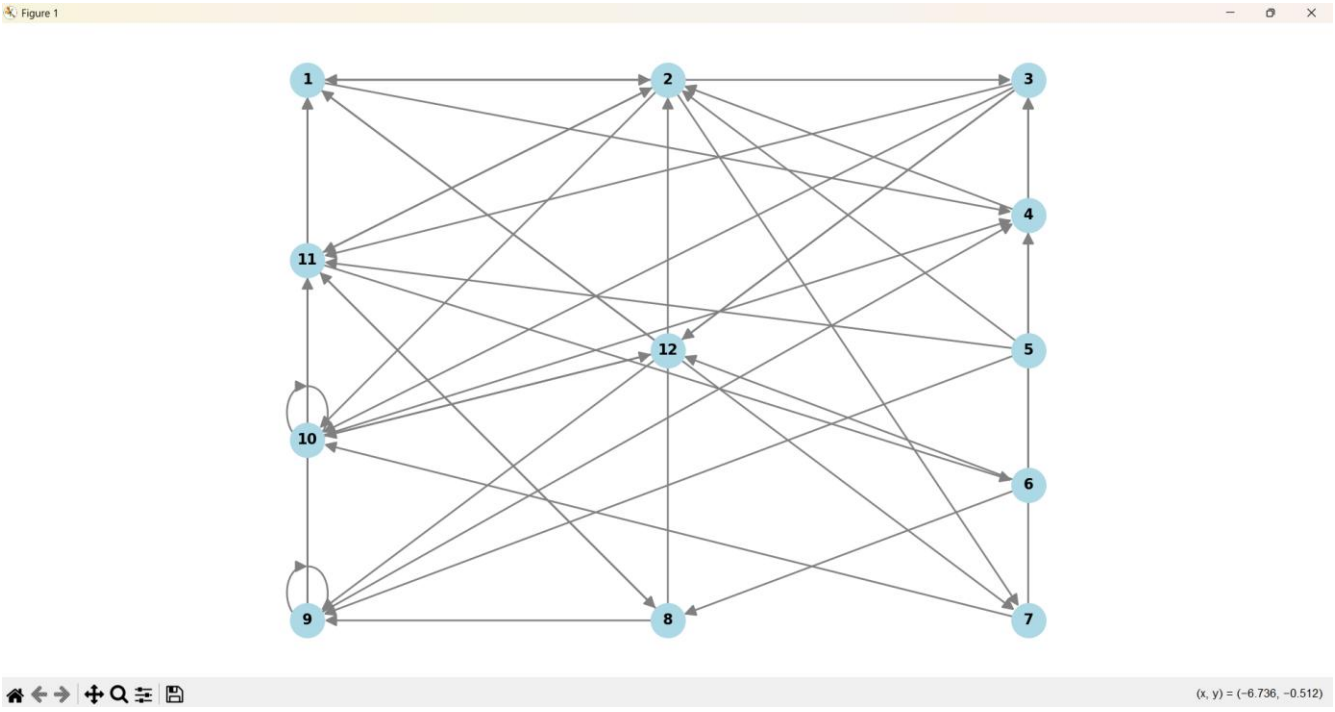
```
        self.n = 10 + self.n3

        self.k = 1.0 - self.n3 * 0.02 - self.n4 * 0.005 - 0.25

    def generate_adjacency_matrix(self):
        np.random.seed(self.n1 * 1000 + self.n2 * 100 + self.n3 * 10 + self.n4)

        T = np.random.uniform(0, 2.0, size=(self.n, self.n))

        A_dir = np.where(T * self.k >= 1.0, 1, 0)

        A_undir = np.maximum(A_dir, A_dir.T)

        return A_dir, A_undir

    def get_vertex_positions(self):
        positions = {}
        if self.n4 in [8, 9]:
            width, height = 8, 6
            outer_vertices = self.n - 1

            vertices_top = math.ceil(outer_vertices / 4)
            vertices_right = math.ceil((outer_vertices - vertices_top) / 3)
            vertices_bottom = math.ceil((outer_vertices - vertices_top - vertices_right) / 2)
            vertices_left = outer_vertices - vertices_top - vertices_right - vertices_bottom

            current_vertex = 1
```

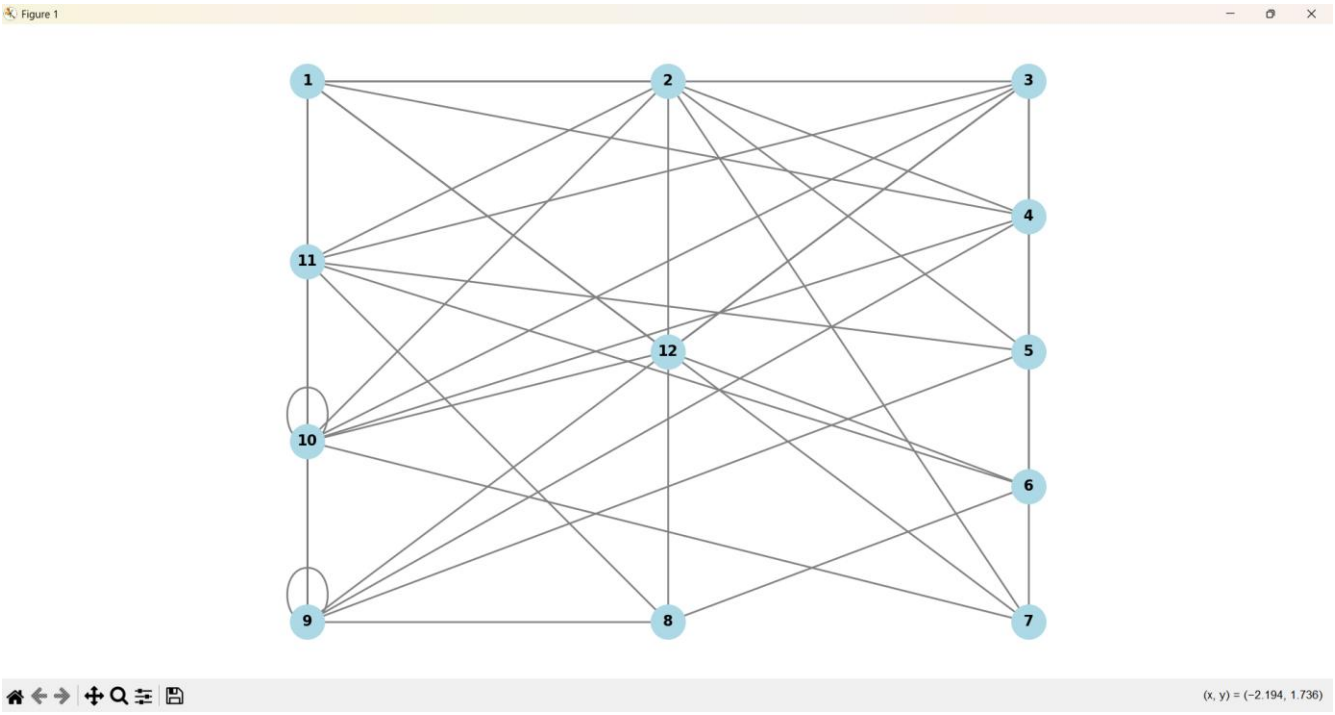
```
    for i in range(vertices_top):
        if current_vertex > outer_vertices: break
        x = -width/2 + i * (width/(vertices_top-1) if vertices_top > 1 else 0)
        positions[current_vertex] = (x, height/2)
        current_vertex += 1

    for i in range(vertices_right):
        if current_vertex > outer_vertices: break
        y = height/2 - (i+1) * (height/(vertices_right+1))
        positions[current_vertex] = (width/2, y)
        current_vertex += 1

    for i in range(vertices_bottom):
        if current_vertex > outer_vertices: break
        x = width/2 - i * (width/(vertices_bottom-1) if vertices_bottom > 1 else 0)
        positions[current_vertex] = (x, -height/2)
        current_vertex += 1

    for i in range(vertices_left):
        if current_vertex > outer_vertices: break
        y = -height/2 + (i+1) * (height/(vertices_left+1))
        positions[current_vertex] = (-width/2, y)
        current_vertex += 1

    positions[self.n] = (0, 0)

return positions
```

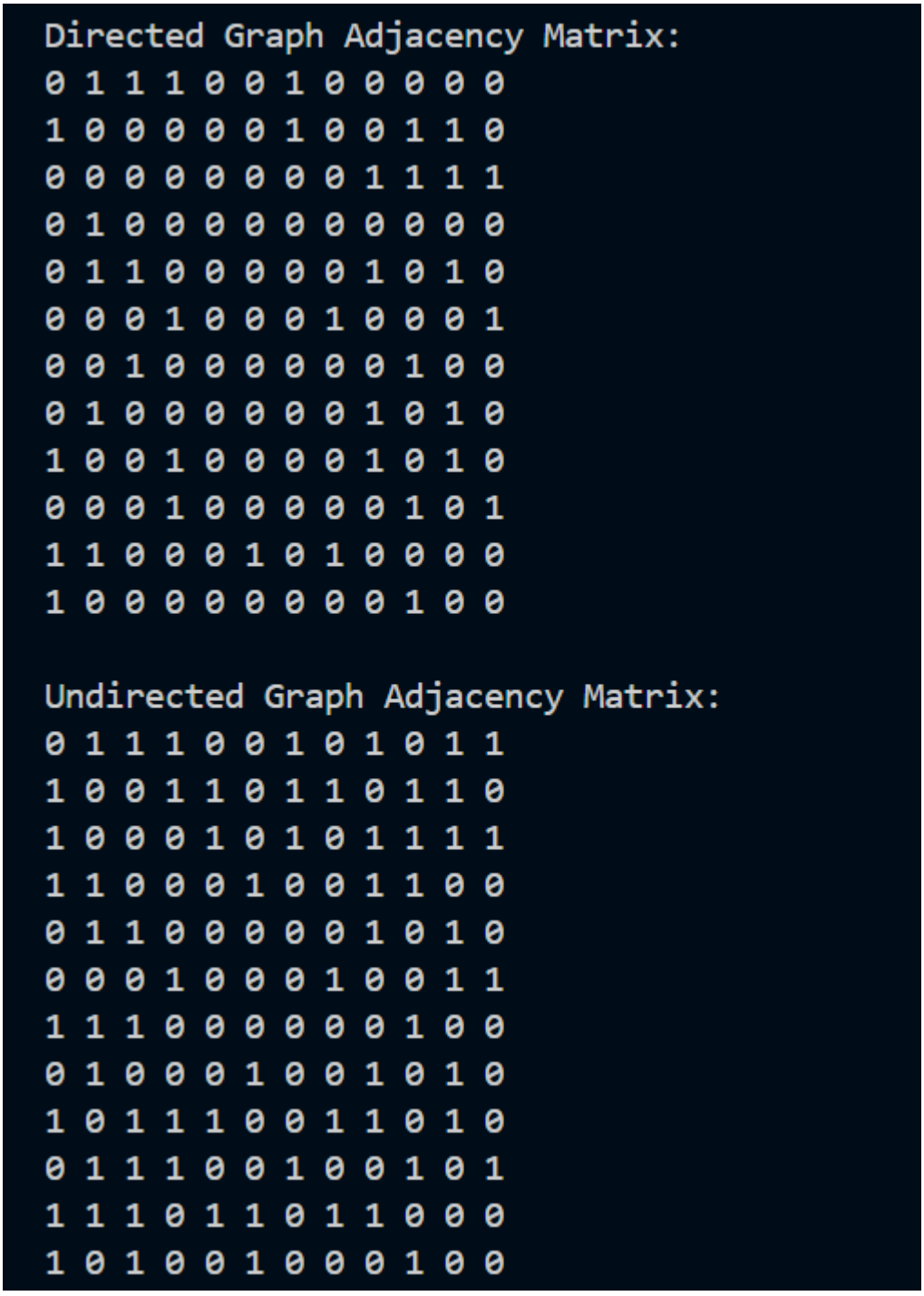
```
def draw_graph(self, adj_matrix, directed=True):
    plt.figure(figsize=(12, 8))
    G = nx.DiGraph() if directed else nx.Graph()

    for i in range(1, self.n + 1):
        G.add_node(i)

    for i in range(self.n):
        for j in range(self.n):
            if adj_matrix[i][j] == 1:
                G.add_edge(i + 1, j + 1)

    pos = self.get_vertex_positions()

    nx.draw(G, pos,
            with_labels=True,
            node_color='lightblue',
            node_size=800,
            arrowsize=20 if directed else 0,
            font_size=12,
            font_weight='bold',
            arrows=directed,
            edge_color='gray',
            width=1.5,
            node_shape='o')

    plt.title(f'{"Directed" if directed else "Undirected"} Graph')
    plt.axis('equal')
```

```
        plt.show()

    def print_matrix(self, matrix, title):
        print(f"\n{title}:")
        for row in matrix:
            print(" ".join(map(str, row)))

def main():
    visualizer = GraphVisualizer()

    A_dir, A_undir = visualizer.generate_adjacency_matrix()

    visualizer.print_matrix(A_dir, "Directed Graph Adjacency Matrix")
    visualizer.print_matrix(A_undir, "Undirected Graph Adjacency Matrix")

    visualizer.draw_graph(A_dir, directed=True)
    visualizer.draw_graph(A_undir, directed=False)

if __name__ == "__main__":
    main()
```

Результати тестування:







Висновки:

У ході виконання лабораторної роботи було реалізовано програму для створення та графічного зображення напрямленого й ненапрямленого графів із заданими параметрами. Було сформовано матриці суміжності на основі генерації випадкових чисел з урахуванням номера варіанту, а також реалізовано алгоритм візуалізації графів у графічному вікні з використанням мови програмування Python та бібліотеки matplotlib.

Виконана робота сприяла закріпленню практичних навичок у представленні графів у комп’ютерних програмах, формуванні матриць суміжності та застосуванні базових графічних примітивів. Отримані знання можуть бути застосовані в подальшій роботі з алгоритмами на графах та побудові візуалізацій у складніших програмних проектах.