Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИУК «Информатика и управление»**

КАФЕДРА **ИУК4 «Программное обеспечение ЭВМ, информационные технологии»**

ЛАБОРАТОРНАЯ РАБОТА

«Представление графов и операции над ними»

ДИСЦИПЛИНА: «Дискретная математика»

Выполнил: студент гр. ИУК4-31Б _____ (Суриков Н. С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Никитенко У. В.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Цель: закрепить умения в нахождении эйлеровых циклов

Задачи:

1. Назработать приложение построения замкнутой (разомкнутой) эйлеровой цепи.
2. Используя алгоритм Флёри, визуализировать этапы построения.

Вариант 11

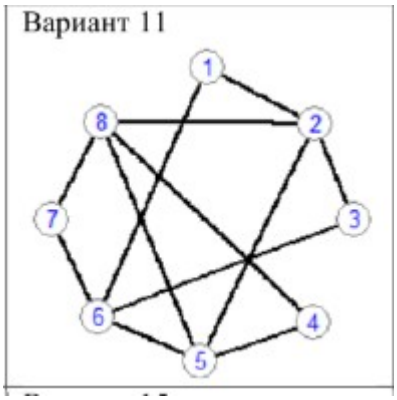
Задание

I. Проверить, что заданный граф является эйлеровым (полуэйлеровым); если построение замкнутой (разомкнутой) эйлеровой цепи невозможно – по согласованию с преподавателем изменить граф так, что построение замкнутой (разомкнутой) эйлеровой цепи стало возможным.

II. Найти замкнутую (разомкнутую) эйлерову цепь.

- написать алгоритм проверки моста;
- написать алгоритм Флёри и применить его к поиску замкнутой (разомкнутой) эйлеровой цепи.

III. Визуализировать последовательность построения замкнутой (разомкнутой) эйлеровой цепи (для Python возможно использование библиотек NetworkX, Igraph).



Листинг программы:

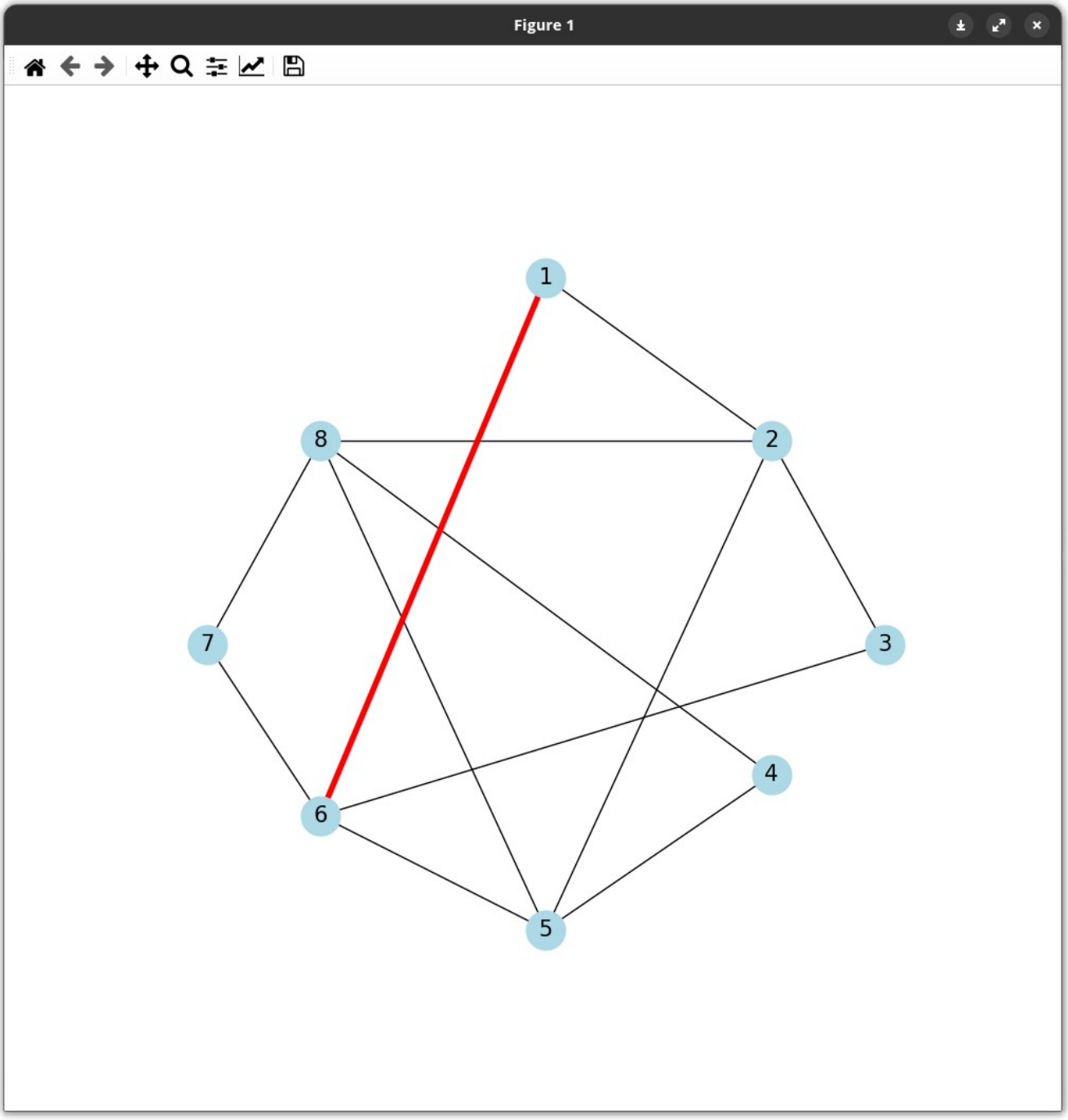
```python
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import time


class Graph:
    def __init__(self, adjacency_matrix):
        self.adjacency_matrix = np.array(adjacency_matrix)
        self.num_vertices = len(adjacency_matrix)

    def is_eulerian(self):
        odd_count = 0
```

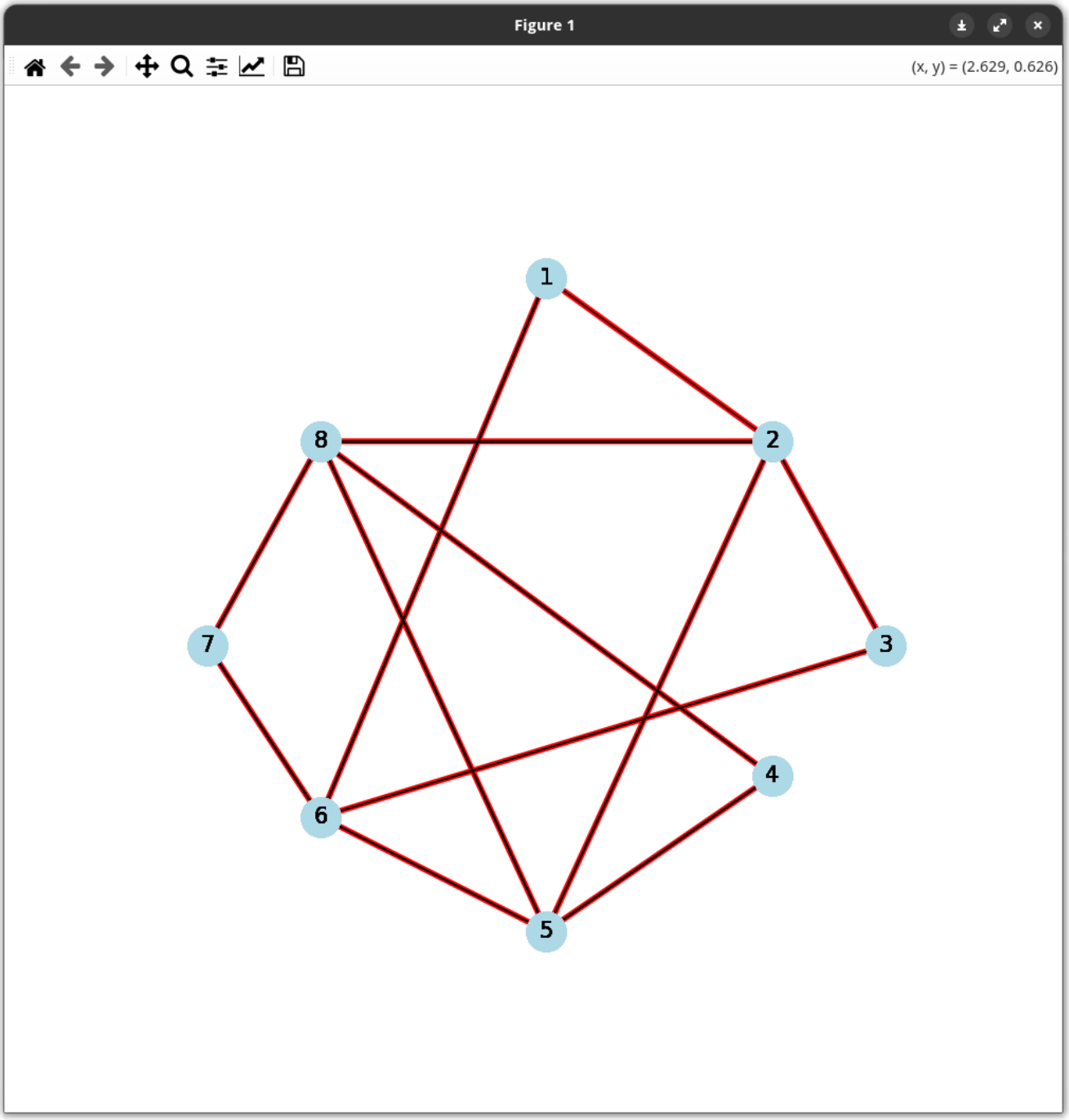
```
        for i in range(self.num_vertices):
            degree = sum(self.adjacency_matrix[i])
            if degree % 2 != 0:
                odd_count += 1
        if odd_count == 0:
            return "Замкнутый эйлеров граф"
        elif odd_count == 2:
            return "Полуэйлеров граф"
        else:
            return "Не является эйлеровым графом"

    def find_eulerian_circuit(self, start_vertex=None):
        if self.is_eulerian() == "Не является эйлеровым графом":
            return []

        temp_matrix = self.adjacency_matrix.copy()
        eulerian_circuit = []
        current_vertex = start_vertex if start_vertex is not None else 0

        stack = [current_vertex]
        while stack:
            current_vertex = stack[-1]
            next_vertex = self.find_next_vertex(current_vertex, temp_matrix)
            if next_vertex is None:
                eulerian_circuit.append(current_vertex + 1)
                stack.pop()
            else:
                temp_matrix[current_vertex][next_vertex] = 0
                temp_matrix[next_vertex][current_vertex] = 0
                stack.append(next_vertex)

        if eulerian_circuit[0] != eulerian_circuit[-1]:
            eulerian_circuit.append(eulerian_circuit[0])

        return eulerian_circuit

    def find_next_vertex(self, vertex, temp_matrix):
        for next_vertex in range(self.num_vertices):
            if temp_matrix[vertex][next_vertex] > 0:
                return next_vertex
        return None

    def visualize_eulerian_circuit(self, path):
        G = nx.Graph(self.adjacency_matrix)
        pos = {
(2, 3),
(2.5, 2),
(2.75, 0.75),
(2.5, -0.05),
(2, -1),
(1.5, -0.3),
(1.25, 0.75),
(1.5, 2),
        }
        labels = {
```

```
            i: str(i + 1) for i in range(self.num_vertices)
        }

        fig, ax = plt.subplots(figsize=(10, 10))
        nx.draw(
            G,
            pos,
            with_labels=True,
            labels=labels,
            node_color="lightblue",
            node_size=700,
            font_size=15,
            ax=ax,
        )

        for i in range(len(path)):
            path_edges = [(path[i] - 1, path[(i + 1) % len(path)] - 1)]
            nx.draw_networkx_edges(
                G, pos, edgelist=path_edges, edge_color="red", width=4, ax=ax
            )
            plt.draw()
            plt.pause(0.5)
            nx.draw(
                G,
                pos,
                with_labels=True,
                labels=labels,
                node_color="lightblue",
                node_size=700,
                font_size=15,
                ax=ax,
            )

        plt.title("Эйлерова цепь")
        plt.show()


if __name__ == "__main__":
    adjacency_matrix = [
        [0, 1, 0, 0, 0, 1, 0, 0],
        [1, 0, 1, 0, 1, 0, 0, 1],
        [0, 1, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 1],
        [0, 1, 0, 1, 0, 1, 0, 1],
        [1, 0, 1, 0, 1, 0, 1, 0],
        [0, 0, 0, 0, 0, 1, 0, 1],
        [0, 1, 0, 1, 1, 0, 1, 0],
    ]

    graph = Graph(adjacency_matrix)

    print(graph.is_eulerian())

    if graph.is_eulerian() != "Не является эйлеровым графом":
        eulerian_circuit = graph.find_eulerian_circuit()
```

```
124
125         print("Эйлерова цепь:", eulerian_circuit)
126
127         graph.visualize_eulerian_circuit(eulerian_circuit)
128     else:
129         print("Невозможно найти эйлерову цепь.")
```

Результат работы:





В результате мы получаем последовательное построение эйлеровой цепи на экране, и её текстовой отображение в консоли.

Вывод: В ходе работы были закреплены умения в нахождении эйлеровых циклов.