**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «**Решение задачи коммивояжёра с помощью метода ближайшего соседа**»**

**Студент гр. 23Б15-пу**

**Голуб Г.Я.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 2](#_Toc280841624)
2. [Описание задачи (формализация задачи) 3](#_Toc1276834523)
3. [Теоретическая часть 3](#_Toc8348496)
4. [Основные шаги программы 4](#_Toc351813160)
5. [Блок схема программы 4](#_Toc1189755412)
6. [Описание программы 6](#_Toc388641127)
7. [Рекомендации пользователя 7](#_Toc1153691210)
8. [Рекомендации программиста 8](#_Toc1917301687)
9. [Исходный код программы 8](#_Toc1673149504)
10. [Контрольный пример 8](#_Toc54910285)
11. [Вывод 10](#_Toc1918507330)
12. [Источники 11](#_Toc1154950108)

# Цель работы

Целью данной лабораторной работы является разработка программного решения для задачи коммивояжёра с использованием метода ближайшего соседа.   
В рамках работы необходимо создать программное средство, которое позволяет пользователю вводить граф в виде списка вершин и матрицы смежности, решать задачу и визуализировать оптимальный путь на графе. Основная цель — исследовать эффективность метода ближайшего соседа, выявить его преимущества и недостатки, а также предложить возможные улучшения в алгоритме.

# Описание задачи (формализация задачи)

Задача коммивояжёра — это классическая NP-трудная задача комбинаторной оптимизации, в которой необходимо найти кратчайший гамильтонов цикл (путь, проходящий через все вершины ровно один раз и возвращающийся в исходную точку) в взвешенном графе.   
В данной работе рассматривается направленный граф, представленный в виде матрицы смежности, где каждая вершина обозначает пункт назначения, а рёбра имеют веса, соответствующие расстояниям или затратам на переход между вершинами.   
Основное требование — минимизация общей длины маршрута.

# Теоретическая часть

Метод ближайшего соседа (Nearest Neighbor Algorithm) является жадным алгоритмом, применяемым для решения задачи коммивояжёра.   
Он работает по простому принципу: начиная с произвольной вершины, алгоритм на каждом шаге выбирает ближайшую не посещённую вершину, повторяет этот процесс, пока не обойдёт все вершины, и затем возвращается в начальную точку. Этот метод является простым и быстрым в реализации, но не всегда приводит к оптимальному решению, так как жадный выбор на каждом шаге не учитывает глобальную оптимальность маршрута.   
В данной лабораторной работе рассматривается также модификация метода с перебором начальной вершины для повышения качества решения.

# Основные шаги программы

1. Ввод пользователем списка вершин и матрицы смежности.
2. Запуск алгоритма ближайшего соседа для поиска кратчайшего пути.
3. Опционально: включение модификации алгоритма с перебором начальной вершины для улучшения результата.
4. Отображение оптимального пути в текстовом формате.
5. Визуализация графа с подсвеченным маршрутом, где используются различные цвета для отображения рёбер, входящих в оптимальный путь.

# Блок схема программы

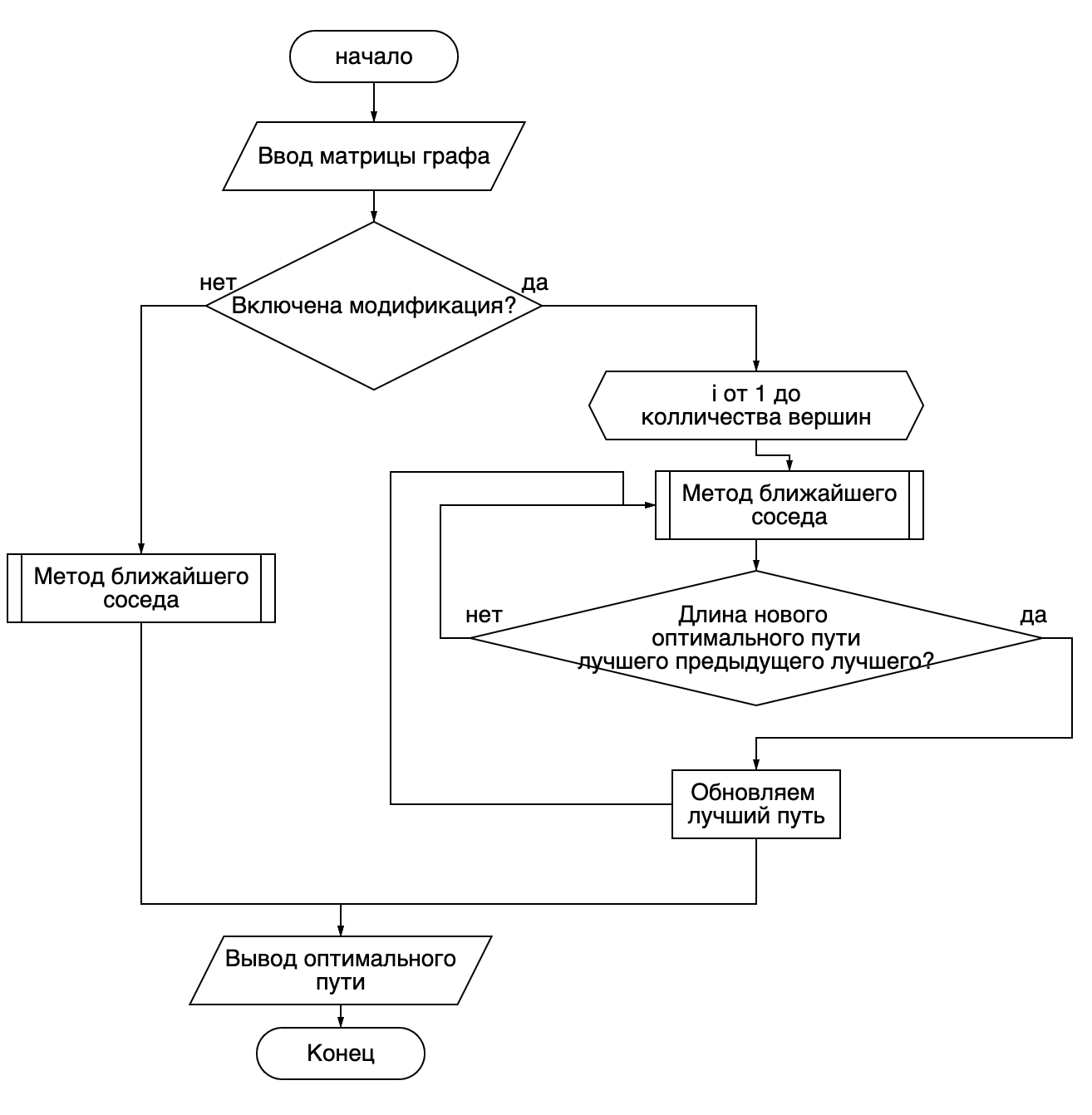


Рис 1. Блок-схема основной программы



Рис 2. Блок-схема подпрограмм

# Описание программы

Программа реализована на языке Python с использованием библиотек PyQt5[[1]](#_Источники) для пользовательского интерфейса и NetworkX[[2]](#_Источники) для построения и визуализации графов.   
 Пользователь вводит граф через текстовое поле, после чего алгоритм ближайшего соседа вычисляет маршрут и отображает его на графе.   
 Программа позволяет переключать режимы: стандартный алгоритм ближайшего соседа и улучшенную версию, использующую перебор всех стартовых вершин.

Таблица 1. main.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| parse\_graph\_input | Считывание графа из поля ввода | None |
| nearest\_neighbor\_tsp | Функция алгоритма tsp | list |
| draw\_graph | Отображение графа | None |
| initUI | Функция инициализации интерфейса | None |

# Рекомендации пользователя

- Вводите граф в корректном формате:   
 ```   
 A B C D E   
 0 3 1 4 8   
 2 0 5 1 3   
 6 2 0 3 7   
 4 3 8 0 2   
 5 6 3 4 0   
 ```   
 - Используйте опцию «Использовать улучшенную версию», если хотите получить более оптимальный маршрут.   
 - Следите за корректностью входных данных, чтобы избежать ошибок в вычислениях.

# Рекомендации программиста

1. Используйте актуальные версии библиотек PyQt5[[1]](#_Источники) и NetworkX[[2]](#_Источники).
2. Для улучшения визуализации можно добавить анимацию построения маршрута.
3. Возможны дальнейшие модификации алгоритма, например, с использованием динамического программирования для приближения к оптимальному решению.

# Исходный код программы

<https://github.com/zgjhz/algorithms_term_2>

# Контрольный пример

1. Запуск программы

Для запуска программы используйте файл **main.py.**

2. Ввод графа (Рис. 1)

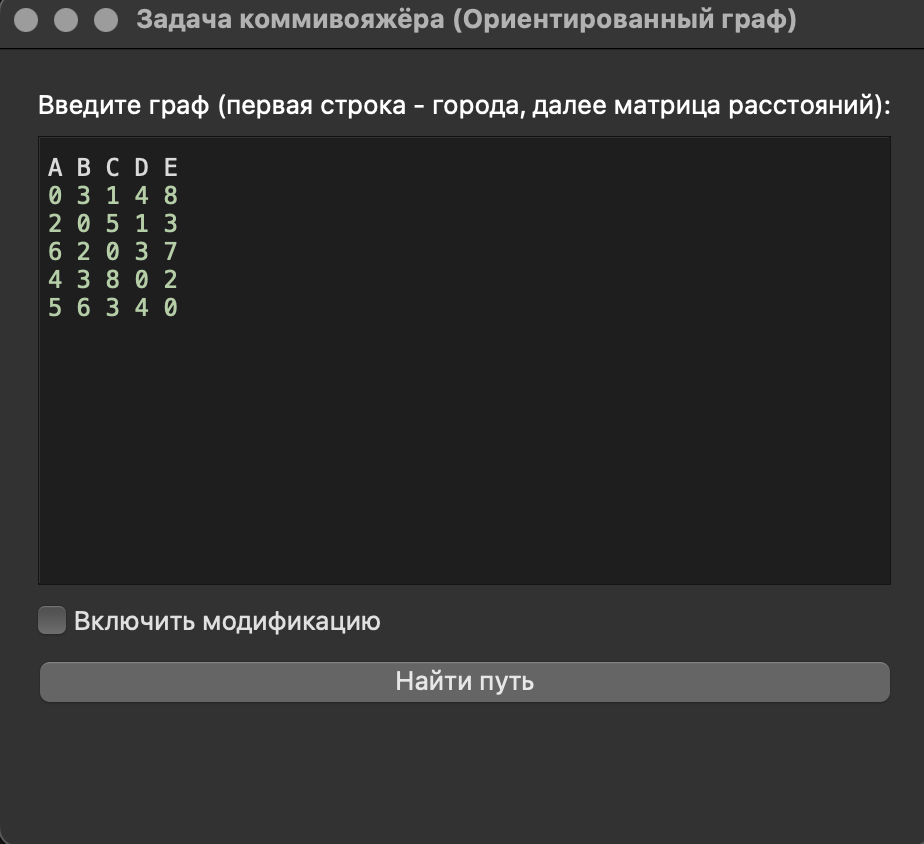
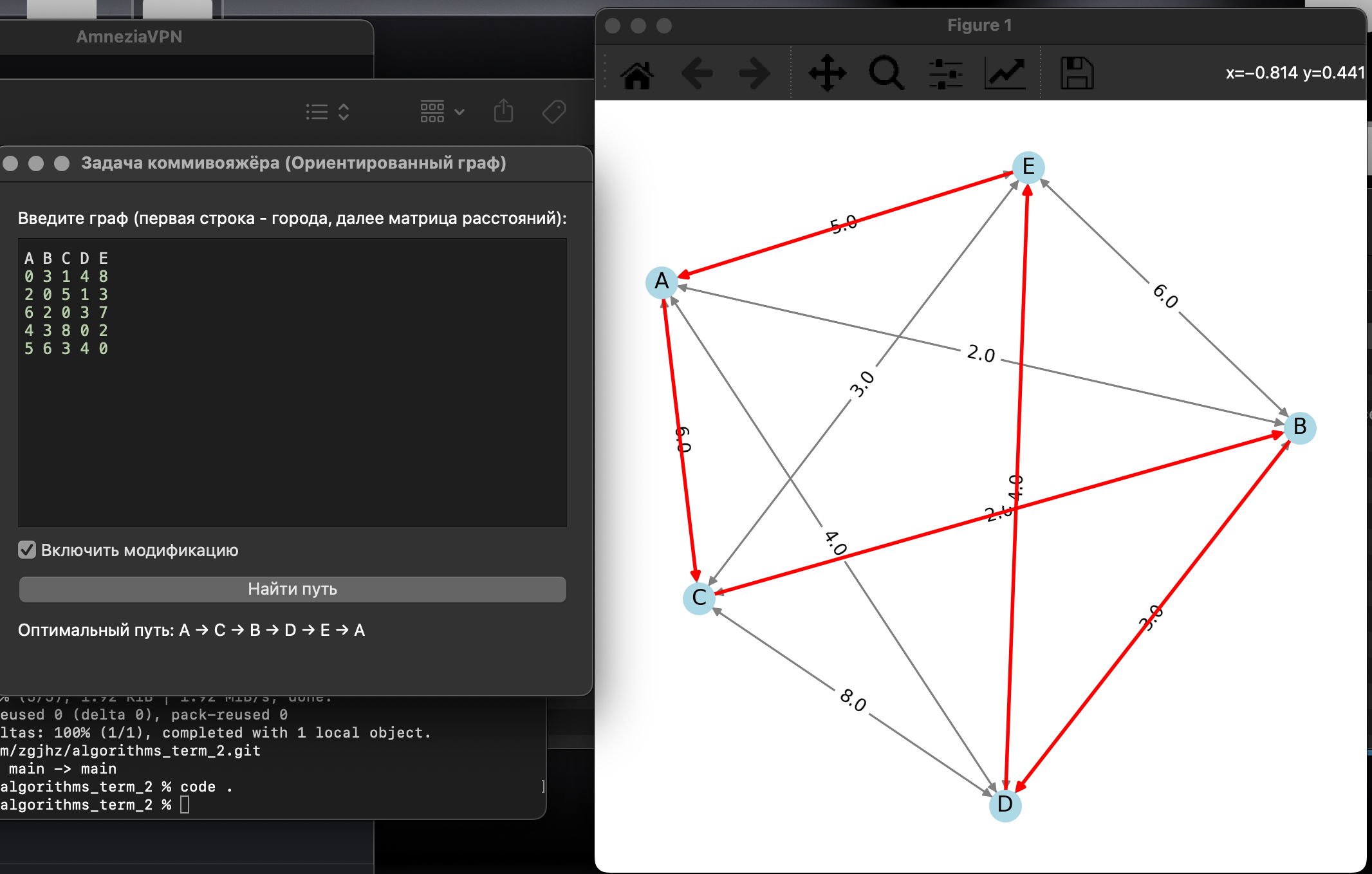


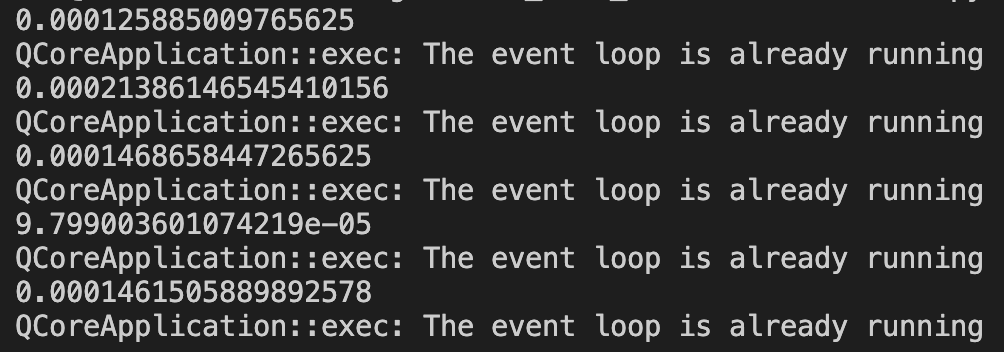
Рис. 1 (Ввод матрицы графа)

3. Нажмите кнопку “Найти путь”. По желание можно включить модификацию

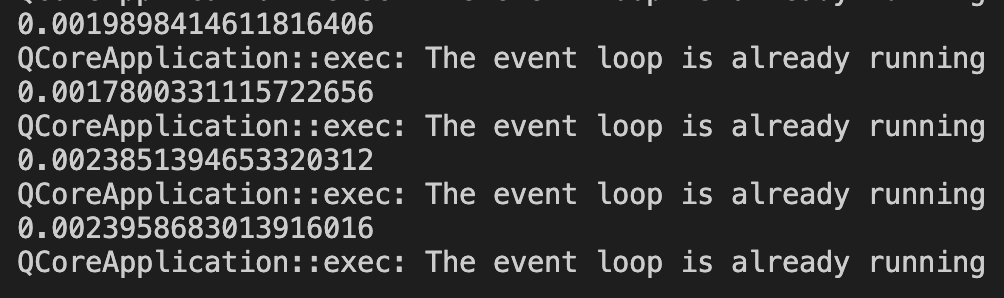
4. Вывод графа с найденным путём (Рис. 2)

Рис. 2 (Вывод программы)

# Сравнение алгоритма с и без модификации

1. Алгоритм без модификации имеет сложность O(n^2), когда как с модификацией имеет сложность O(n^3)  
   Среднее время выполнения без модификации равно 0.15 сек (Рис. 3)  
   Рис. 3 (Время выполнения алгоритма без модификации)

Среднее время выполнения с модификацией равно 0.20 сек (Рис. 4)

Рис. 4 (Время выполнения алгоритма с модификацией)

1. За счёт рассматривания разных начальных вершин алгоритм с модификацией чаще находил более оптимальные маршруты

# Вывод

В ходе лабораторной работы была реализована программа для решения задачи коммивояжёра методом ближайшего соседа.   
Программа позволяет пользователю вводить граф, использовать стандартный алгоритм или его модифицированную версию и визуализировать маршрут.   
Проведённые тесты показали, что модифицированный алгоритм с перебором начальной вершины приводит к более качественным результатам в большинстве случаев.

# Источники

**1. PyQt5 Documentation: https://doc.qt.io/qtforpython/**

**2. NetworkX Documentation: https://networkx.github.io**