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 о коммивояжере с помощью метода ближайшего соседа»

Студент гр. 23Б15-пу
Трофимов И. А.

Преподаватель
Дик А.Г.

Санкт-Петербург

2025 г.

Оглавление

Цель работы

Целью данной работы является исследование и разработка алгоритма ближайшего соседа для решения задачи коммивояжёра, а также анализ его эффективности в поиске кратчайшего гамильтонова цикла. В ходе выполнения работы предполагается изучить особенности алгоритма, реализовать программу с графическим интерфейсом для интерактивного построения графа и поиска оптимального пути, а также оценить качество и быстродействие полученного решения.

Теоретическая часть

Эта часть работы посвящена рассмотрению задачи коммивояжёра и анализу алгоритма ближайшего соседа как одного из эвристических методов её решения. Задача коммивояжёра заключается в поиске кратчайшего маршрута, проходящего через все заданные вершины графа ровно один раз и возвращающегося в исходную точку, что является NP-полной задачей. В данной работе рассматривается применение метода ближайшего соседа, который на каждом шаге выбирает следующую вершину с минимальным весом ребра, обеспечивая быстрое получение приближённого решения.

Также в теоретической части рассматриваются особенности работы алгоритма на ориентированных графах, где между парой вершин могут существовать ребра с различными весами в зависимости от направления. Такой подход требует учета асимметрии при построении гамильтонова цикла и позволяет провести анализ влияния выбора направления и значений весов на итоговое качество решения.

Описание задачи

Задача коммивояжёра заключается в нахождении минимального циклического маршрута, проходящего через все заданные вершины графа ровно один раз и возвращающегося в исходную точку. Данная задача относится к NP-полным, что обуславливает применение эвристических методов для получения приближённого решения за приемлемое время. В работе исследуется алгоритм ближайшего соседа, который на каждом шаге выбирает вершину с минимальным весом ребра, при этом особое внимание уделяется ориентированным графам, где между парой вершин могут существовать ребра с различными весами в зависимости от направления.

Основные шаги:

1. **Формирование графа**

   Пользователь интерактивно создаёт граф, добавляя вершины кликами по рабочему полю. Каждая вершина получает уникальный номер, а её координаты сохраняются для последующей отрисовки.

2. **Добавление ребер с заданием весов**

   В режиме добавления ребер пользователь выбирает пару вершин, после чего задаются веса для направлений. Для двунаправленных соединений предусмотрена возможность указания различных весов для каждого направления, что позволяет моделировать асимметричные графы.

3. **Удаление элементов**

   Реализованы функции удаления как отдельных вершин, так и конкретных ребер. При удалении вершины происходит обновление матрицы смежности и перерисовка графа, а удаление ребра приводит к

его исключению и корректной перерисовке соединений на экране.

4. **Генерация случайного графа**

   Программа позволяет автоматически сгенерировать граф заданного размера, при этом вершины размещаются случайным образом, а ребра создаются с заданной вероятностью появления и случайными весами.

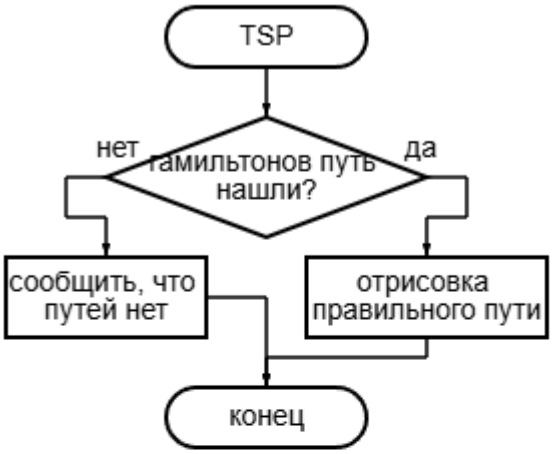
5. **Решение задачи коммивояжёра**

   Алгоритм ближайшего соседа последовательно перебирает возможные стартовые вершины, выбирая на каждом шаге следующую вершину с минимальным весом ребра, что позволяет найти приближённое решение задачи нахождения кратчайшего гамильтонова цикла.

6. **Отображение результата**

   Найденный гамильтонов цикл визуализируется на отдельном рабочем поле (нижнем Canvas), где выделяются все вершины и ребра маршрута, а также выводится суммарный вес пути.

Блок схема программы



Описание программы

Программная реализация написана на языке Python версии 3.13.0 с использованием библиотек random[1], tkinter[2]. Программа представляет собой GUI-приложение, реализующее алгоритм TSP для поиска минимального гамильтонова пути. Пользователь может настраивать параметры алгоритма, количество вершин и ребер, а также веса. Кроме того, предусмотрена возможность автоматической генерации графа, для ускорения тестирования. В процессе разработки программы использовался следующий модуль:

Таблица1 main.py

Функция	Описание	Возвращаемое значение
add_vertex()	Добавляет новую вершину в граф при клике на поле.	None
add_edge()	Включает режим добавления рёбер между вершинами.	float
set_edge_weight()	Устанавливает веса рёбер в обоих направлениях.	None
delete_vertex()	Удаляет выбранную вершину и все связанные с ней рёбра.	None
delete_edge()	Удаляет выбранное ребро из графа.	None

clear_all()	Полностью очищает граф, удаляя все вершины и рёбра.	None
generate_random_graph()	Создаёт случайный граф с заданным числом вершин и рёбер.	None
nearest_neighbor_algorithm()	Выполняет алгоритм ближайшего соседа для поиска пути.	None
draw_graph()	Перерисовывает граф, отображая вершины, рёбра и их веса.	None
draw_solution()	Отображает найденный гамильтонов цикл на графе.	None

Рекомендации пользователя

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки, такие как tkinter[2]. Код можно запустить в среде

разработки или через командную строку. Запуск программы производится через файл main.py.

При запуске программы вы можете сами создать граф или получить его автоматически. Вводите соответствующие значения поля в GUI, если хотите сами создать граф.

Рекомендации программиста

Для поддержания актуальности и работоспособности программы используйте последние версии библиотек, особенно tkinter[2]. Применяйте практики надлежащего именования переменных и функций для улучшения читаемости кода.

Исходный код программы:

Гитхаб

Контрольный пример

1. **Запуск программы**

   Для запуска программы используйте файл main.py. Программа загружает GUI. (Рис. 1)

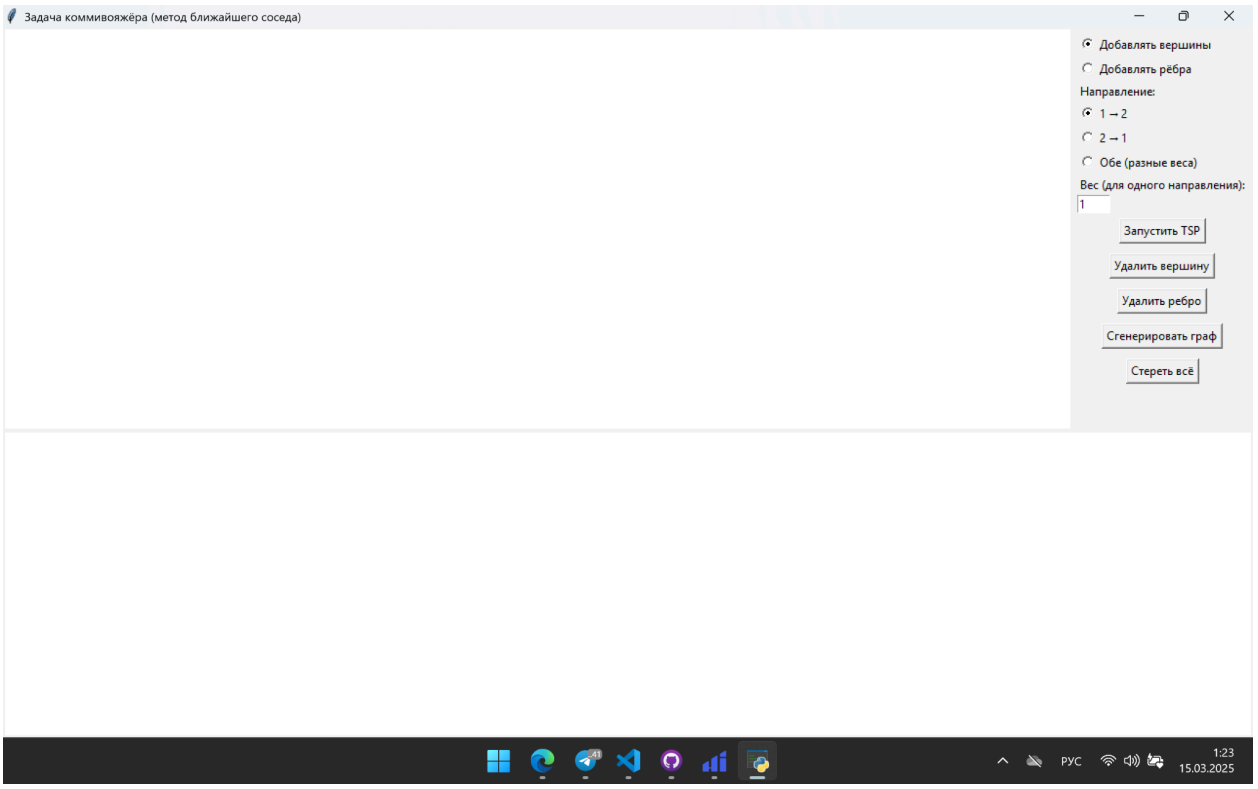


Рис 1. Окно GUI

2. **Выбор параметров**

После запуска программы пользователю будет предложено выбрать параметры и создать граф вручную (Рис. 2)

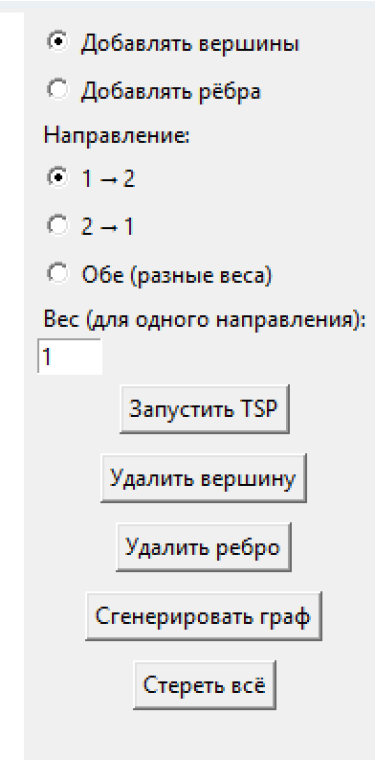


Рис 2. Меню настройки

3. **Обработка графа и вывод результатов**

   После создания графа программа запускает алгоритм, а затем выводит результат, найденный путь (Рис. 3) или сообщение, что такого нет (Рис. 4).

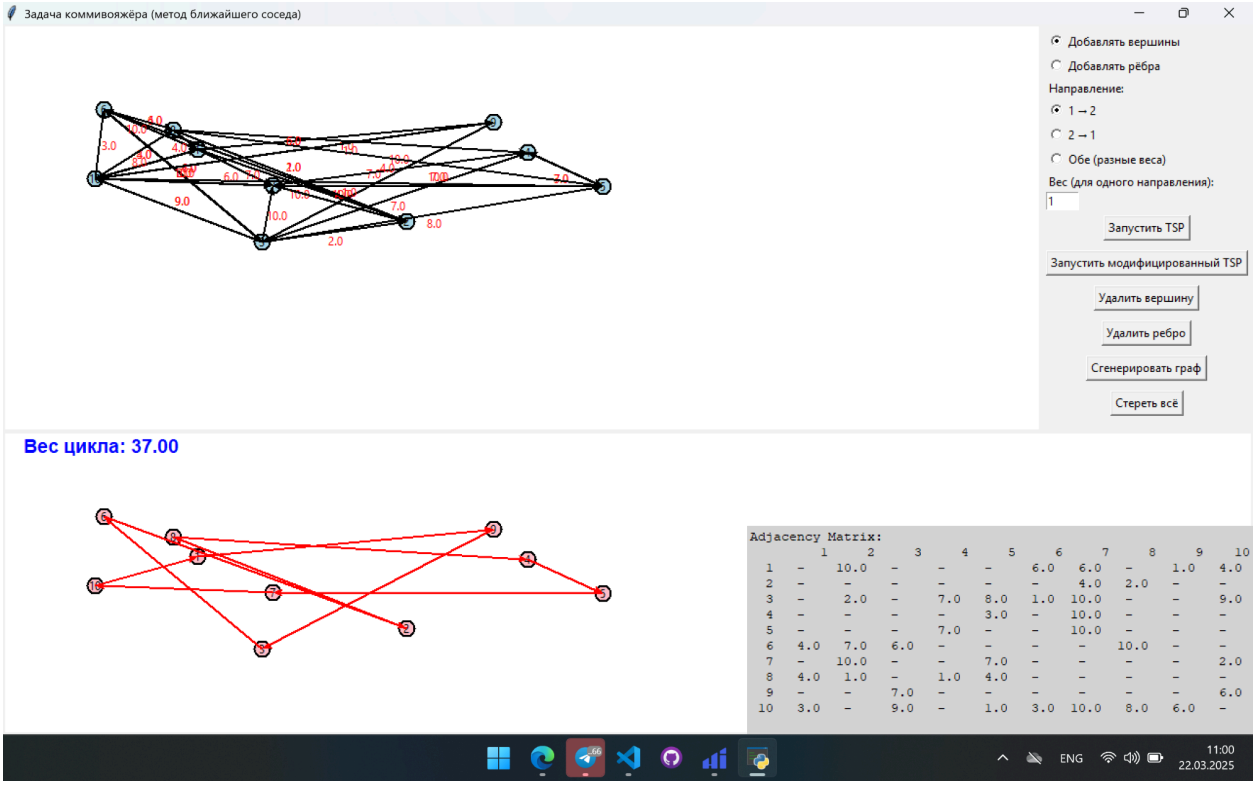


Рис 3. Результат работы алгоритма

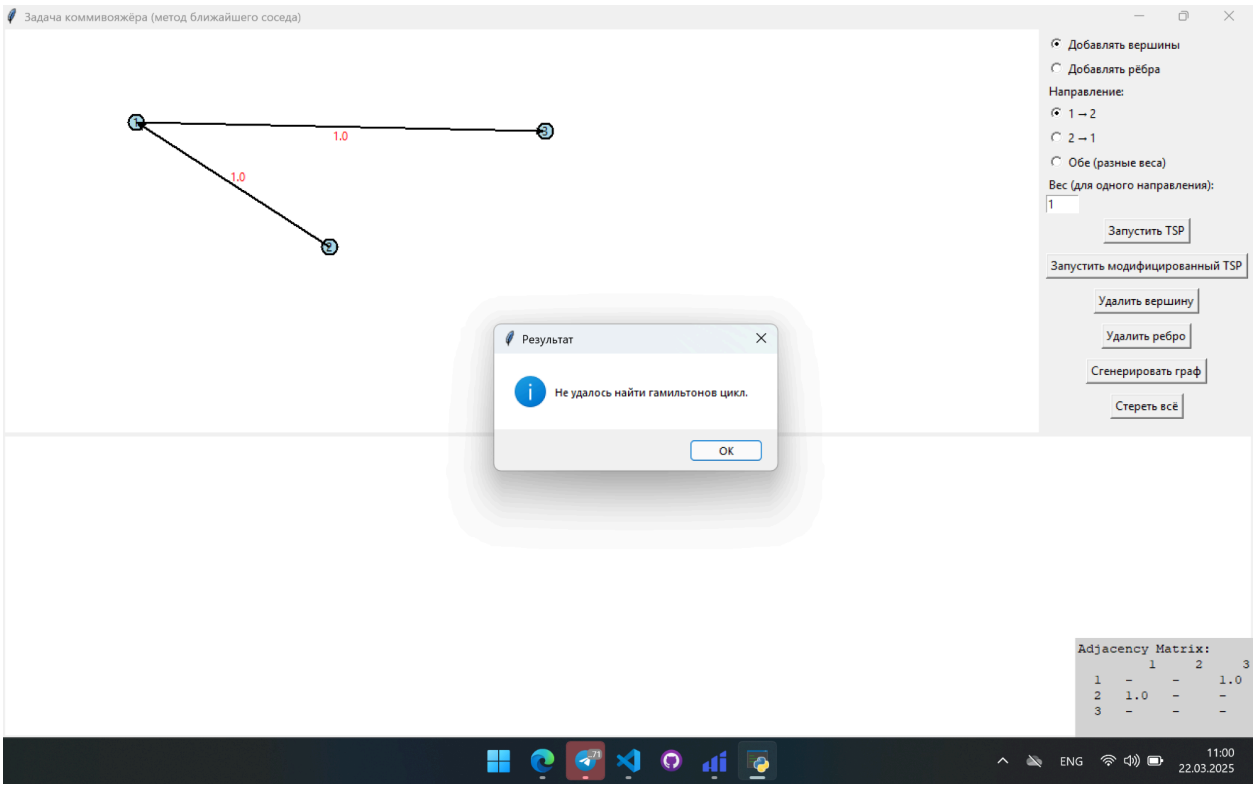


Рис 4. Результат работы алгоритма

Исследование

Количество вершин	Без модификации	С модификацией
10	55	61
10 другой граф	51	51
20	58	56
20 другой граф	50	50
30	75	60
30 другой граф	76	73
50	84	77
50 другой граф	102	99
100	164	151
100 другой граф	139	135-178

Вывод

В ходе выполнения лабораторной работы была реализована программа для решения задачи коммивояжёра с использованием алгоритма ближайшего соседа. Приложение предоставляет удобный графический интерфейс, позволяющий вручную задавать граф, а также генерировать его случайным образом. Реализованы функции добавления и удаления вершин и рёбер, установка направленных рёбер с разными весами, а также визуализация найденного маршрута.

Источники

1. Random documentation // RandomURL: https://random.org (дата обращения: 14.03.2025).
2. Tkinter documentation // Tkinter URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 14.03.2025)