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
Сериков К.Г.

Преподаватель
Дик А.Г.

Санкт-Петербург
2025 г.

Оглавление

1. Цель работы.....3
2. Описание задачи (формализация задачи).....3
3. Основные шаги программы.....5
4. Блок схема программы.....6
5. Описание программы.....7
6. Рекомендации пользователя.....8
7. Рекомендации программиста.....9
8. Исходный код программы.....9
9. Контрольный пример.....10
10. Анализ.....12
11. Вывод.....13
12. Источники.....13

Цель работы

Целью данной лабораторной работы является изучение и реализация алгоритма ближайшего соседа для решения задачи коммивояжёра (TSP). В рамках данной работы была разработана программа, которая находит кратчайший гамильтонов цикл при помощи алгоритма ближайшего соседа и предоставляет возможность переключения между обычной и модифицированной версиями алгоритма.

Описание задачи (формализация задачи)

В данной лабораторной работе необходимо исследовать эффективность алгоритма ближайшего соседа для решения задачи коммивояжёра (TSP).

Формализация задачи включает следующие компоненты:

1. **Изучение особенностей алгоритма ближайшего соседа**: Исследовать принципы работы алгоритм ближайшего соседа, включая его применение для решения задачи коммивояжера. Рассмотреть эффективность алгоритма.
2. **Разработка алгоритма для нахождения кратчайшего гамильтонова цикла:** Создать программу, реализующую алгоритм ближайшего соседа для поиска решения задачи коммивояжера. Реализовать визуализацию исходного и итогового графов.
3. **Тестирование программы на взвешенном ориентированном графе:** Провести тестирование программы на контрольных примерах. Оценить точность и эффективность метода ближайшего соседа для решения задачи коммивояжёра (TSP).
4. **Анализ эффективности алгоритма**: Провести сравнительный анализ работы алгоритма ближайшего соседа с модификацией и без неё, а также исследовать влияние структуры графа на эффективность. Оценить влияние модификации на скорость и точность нахождения кратчайшего гамильтонова цикла.

Теоретическая часть

Задача коммивояжера

Задача коммивояжёра (Traveling Salesman Problem) является одной из классических задач комбинаторной оптимизации. Она формулируется следующим образом: имеется множество городов и известны расстояния между каждой парой городов. Требуется найти кратчайший маршрут, который проходит через каждый город ровно один раз и возвращается в начальную точку.

Алгоритм ближайшего соседа

Алгоритм ближайшего соседа с модификацией выбора стартовой вершины — это эвристический жадный метод решения задачи коммивояжёра (TSP). Алгоритм основан на поочередном выборе ближайшей непосёщенной вершины со стартовой точки. Этот метод прост в реализации, однако далеко не всегда находит оптимальный маршрут. Модификация заключается в переборе всех стартовых вершин вместо одной, для возможного увеличения качества нахождения пути.

Принцип работы

1. Выбирается стартовая вершина.
2. Из текущей вершины выбирается ближайшая непосещенная вершина.
3. Вершина помечается как посещенная, и маршрут продолжается.
4. Когда все вершины посещены, путь замыкается, возвращаясь в стартовую вершину.

Основные шаги программы

1. **Запуск программы:** Инициализируется графический интерфейс с элементами управления.
2. **Настройка параметров:** Пользователь в интерфейсе может задать граф вручную или загрузить JSON файл, также пользователь выбирает между стандартным и модифицированным режимами алгоритма.
3. **Запуск алгоритма:** При нажатии на кнопку "Рассчитать" программа запускает выполнение ближайшего соседа на заданном графе.
4. **Начало работы алгоритма:** Выбираем нулевую вершину за начальную. Помечаем её как посещённую и инициализируем путь с этой вершины.
5. **Построение маршрута:** Пока есть непосещённые вершины:
   - Находим ближайшую непосещённую вершину к текущей
   - Добавляем её в маршрут
   - Помечаем как посещённую
   - Обновляем общее расстояние
6. **Завершение цикла:** Добавляем начальную вершину в конец маршрута и вычисляем полное расстояние цикла.
7. **Отображение результатов:** Программа отображает кратчайший гамильтонов цикл в виде графа и длину это пути и вершины маршрута.

Блок схема программы

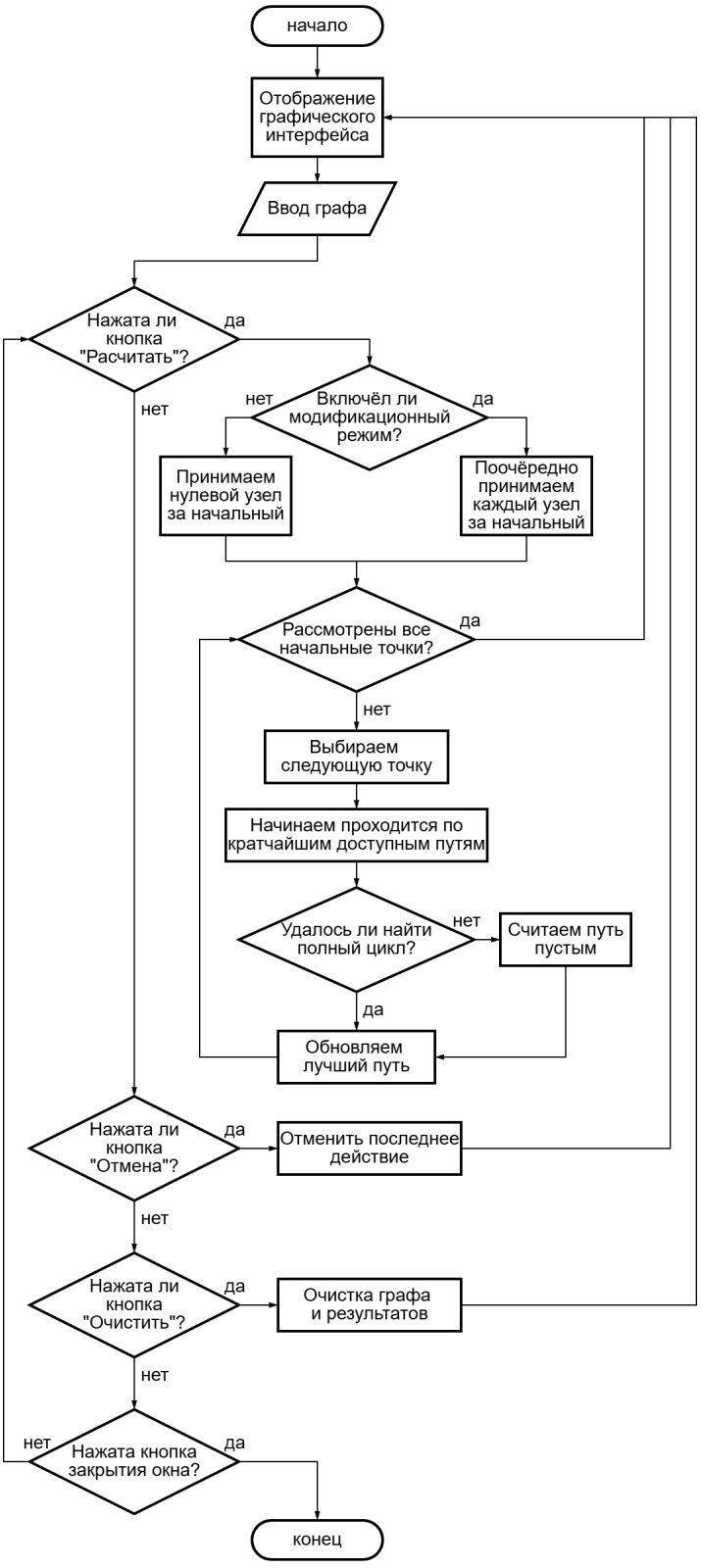

Рис 1. Блок-схема программы

Описание программы

Программная реализация написана на языке Python 3.13.0 с использованием следующих библиотек: tkinter [1], numpy [2], time [3], math [4] и json [5]. Программа организована в виде графического интерфейса для решения задачи коммивояжёра (TSP) с помощью алгоритма ближайшего соседа. В программе реализован только один класс – TSPApp, который отвечает за работу всей программы. В процессе разработки программы использовался main.py, включающий 12 функций, каждая из которых имеет чётко определённое назначение:

Таблица 1. main.py

Функция	Описание	Возвращаемое значение
__init__	Инициализация интерфейса.	None
add_node_or_edge	Добавляет вершину или ребро в зависимости от места клика	None
add_node	Добавляет новую вершину в заданной позиции	None
add_edge	Добавляет ребро между двумя вершинами с вычислением евклидова расстояния	None
find_clicked_node	Находит вершину по координатам клика	int или None
redraw_graph	Перерисовывает граф и обновляет таблицу рёбер	None
solve_tsp	Реализует алгоритм ближайшего соседа	None
visualize_solution	Визуализирует найденный маршрут на отдельном холсте	None
load_graph	Загружает граф из JSON-файла	None

get_distance	Возвращает расстояние между двумя вершинами	float
undo_action	Отменяет последнее действие	None
clear_graph	Полностью очищает граф, историю изменений и результаты	None

Рекомендации пользователя

Программа позволяет запустить алгоритма ближайшего соседа для решения задачи коммивояжёра (TSP) с использованием графического интерфейса.

1. **Запуск программы**: Откройте программу с помощью Python 3.13.0, чтобы инициализировать интерфейс. Интерфейс позволяет нарисовать граф и выбирать режим: стандартный или модифицированный.
2. **Настройка параметров**: В графическом интерфейсе реализованы следующий функции:
   - Добавление узлов: Для создания нового узла кликните левой кнопкой мыши на область графа.
   - Добавление ребер: Для создания ребра между узлами кликните сначала на один узел, затем на второй. Длина ребра будет автоматически подсчитана.
   - Выбор модификации: Для использования модификации перебора начальных вершин установите галочку в флажок «Использовать модификацию».
3. **Запуск алгоритма**: Нажмите кнопку «Рассчитать» для выполнения алгоритма. При этом будет вычислен и отображён кратчайший гамильтонов цикл. Программа отобразит последовательность вершин кратчайшего гамильтонова цикла и построенный маршрут с выделенными ребрами.
4. **Дополнительные элементы интерфейса**: Для отмены последнего действия нажмите кнопку «Отмена». Для сброса графа нажмите кнопку «Очистить».

Рекомендации программиста

Актуальность версии Python: Используйте обновленную версию Python, tkinter [1], numpy [2], time [3], math [4] и json [5]. Уделяйте внимание четкому именованию переменных и функций. Тщательно тестируйте на различных графах, чтобы удостовериться в корректной работе алгоритма и визуализации. Убедитесь, что интерфейс программы в Tkinter легко понимается пользователями. Разделите входные данные, управление алгоритмом и отображение результатов по разным секциям.

Исходный код программы

https://github.com/romplle/spbu-algorithms-and-data-structures/

Контрольный пример

1. Запуск программы и ввод параметров

   Для запуска программы откройте main.py. Программа откроет графический интерфейс для настройки и запуска алгоритма ближайшего соседа, предназначенного для решения задачи коммивояжёра (TSP) (Рис. 2).

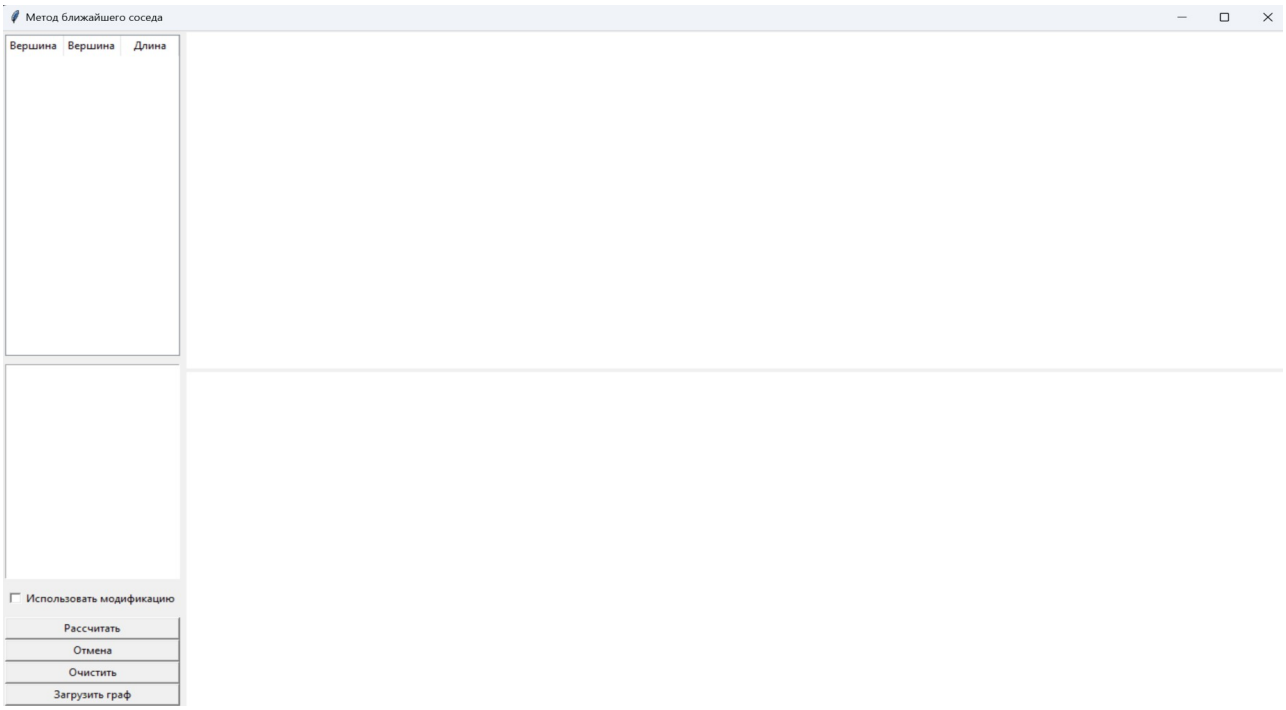

Рис. 2. Начальное окно программы

2. Ввод исходный данных

   После запуска программы можно добавить вершины и соединить их ребрами вручную, а можно нажать кнопку «Загрузить граф» и выбрать JSON-файл для импортирования готового графа. Вы также можете выбрать режим работы (стандартный или модифицированный) (Рис. 3).

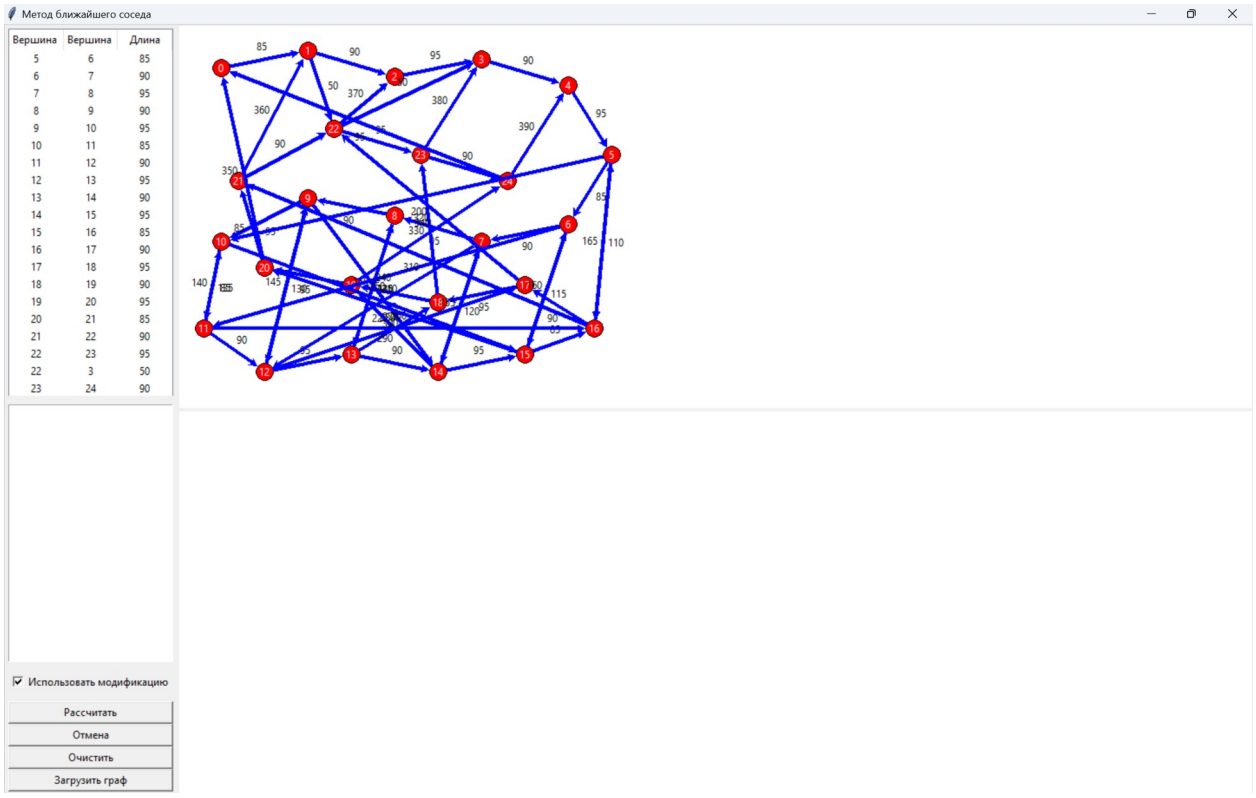

Рис. 3. Пример исходных данных программы

3. Запуск алгоритма

   После задания графа нажмите кнопку «Рассчитать», чтобы запустить алгоритм. Программа вычислит кратчайший гамильтонов цикл методом ближайшего соседа, обновляя интерфейс для отображения результатов (Рис. 4).

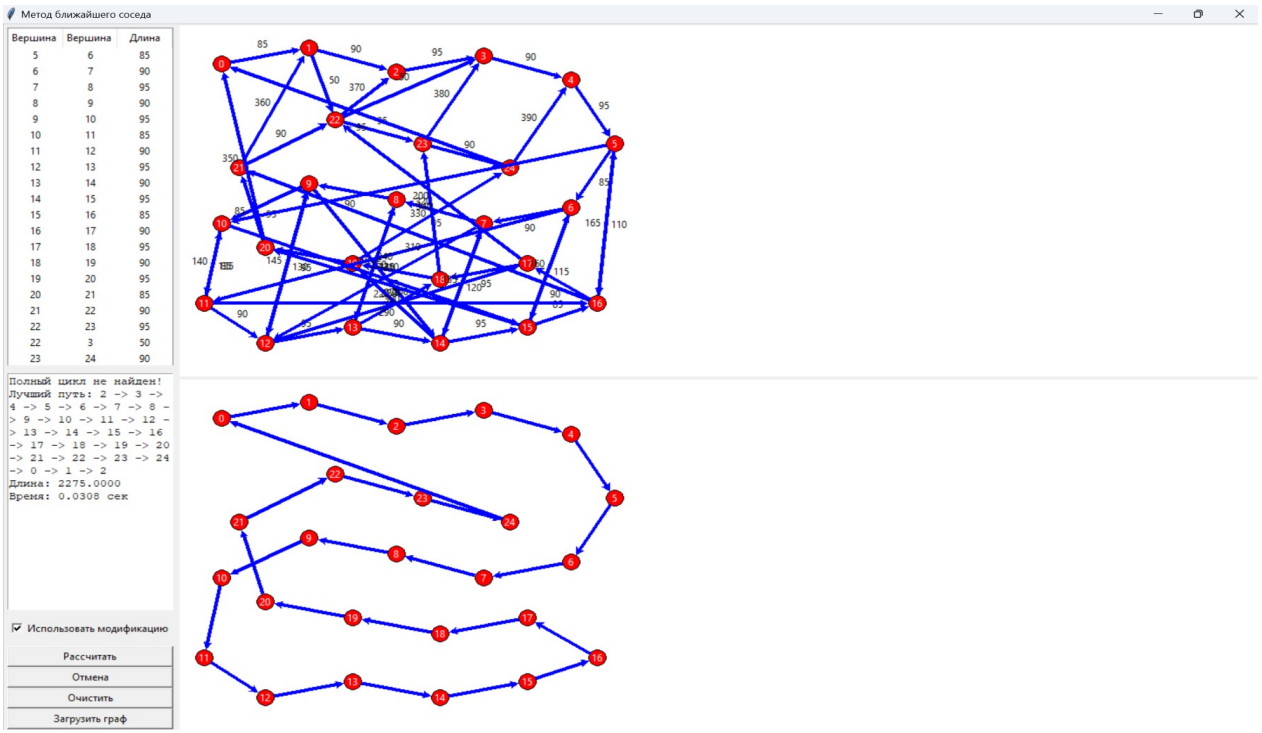

Рис. 4. Пример результатов программы

Анализ

В качестве модификации классического алгоритма ближайшего соседа был применён метод многократного запуска с каждой вершины графа. Вместо того чтобы начинать построение маршрута с одной фиксированной вершины, алгоритм запускается последовательно из каждой вершины, после чего выбирается наиболее короткий из полученных маршрутов. Такой подход позволяет частично компенсировать главный недостаток жадного алгоритма — сильную зависимость от начальной вершины.

Таблица 2: тестирование алгоритма ближайшего соседа

Кол-во вершин	Модификация	Время	Длина пути
6	Без модификации	0.0001 сек	12
6	С модификацией	0.0002 сек	12
15	Без модификации	0.0022 сек	18
15	С модификацией	0.0258 сек	17
25 (1)	Без модификации	Полный цикл не найден	
25 (1)	С модификацией	0.0456 сек	2275
25 (2)	Без модификации	0.0090 сек	31
25 (2)	С модификацией	0.1414 сек	25

Общие наблюдения

Из таблицы видно, что модифицированный алгоритм демонстрирует лучшую устойчивость и качество решений. При одинаковом числе вершин (например, 15) модификация позволила сократить длину маршрута, пусть и ценой увеличения времени выполнения.

При 25 вершинах в первом графе базовый алгоритм не смог построить полный цикл, тогда как модифицированный успешно справился с задачей. Это подчёркивает критическое ограничение жадной стратегии: выбор ближайшей вершины на каждом шаге может привести к тупику, если в дальнейшем не остаётся достижимых вершин, необходимых для завершения маршрута.

Вывод

В рамках работы был реализован алгоритм ближайшего соседа для решения задачи коммивояжёра, модифицированный путём запуска с каждой вершины графа. Также был разработан удобный графический интерфейс. Преимуществами алгоритма являются простота реализации и высокая вычислительная эффективность, что делает его подходящим для быстрого приближённого решения задачи.

Источники

1. tkinter — Python interface to Tcl/Tk // URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 26.03.2025).
2. NumPy documentation // URL: https://numpy.org/doc/stable/index.html (дата обращения: 26.03.2025).
3. time — Time access and conversions // URL: https://docs.python.org/3/library/time.html (дата обращения: 27.03.2025).
4. math — Mathematical functions // URL: https://docs.python.org/3/library/math.html (дата обращения: 27.03.2025).
5. json — JSON encoder and decoder // URL: https://docs.python.org/3/library/json.html(дата обращения: 27.03.2025).