**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления Программа бакалавриата**

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

**ОТЧЕТ**

**по лабораторной работе №2**

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

**на тему**

**«Решение задачи о коммивояжере с помощью алгоритма имитации отжига»**

**Студент гр. 23Б15-пу**

**Сурин И. С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург 2025 г**

**Оглавление**

1. [**Цель работы** 3](#_bookmark0)
2. [**Теоретическая часть** 3](#_bookmark1)
3. [**Описание задачи** 3](#_bookmark2)
4. [**Основные шаги программы** 3](#_bookmark3)
5. [**Описание программы** 6](#_bookmark4)
6. [**Рекомендации пользователя** 8](#_bookmark5)
7. [**Рекомендации программиста** 8](#_bookmark6)
8. [**Исходный код программы** 8](#_bookmark7)
9. [**Контрольный пример** 8](#_bookmark8)
10. [**Исследование** 10](#_bookmark9)
11. [**Вывод** 10](#_bookmark10)
12. [**Источники** 10](#_bookmark11)

# **Цель работы**

Цель данной лабораторной работы — изучение и практическая реализация метода имитации отжига для решения задачи коммивояжёра. В ходе выполнения реализуется алгоритм поиска приближённого оптимального маршрута по ориентированному графу с заданными весами рёбер. Также рассматривается вариант использования распределения Больцмана для более эффективного выбора переходов. Программа снабжена графическим интерфейсом, позволяющим пользователю наглядно задать структуру графа, настроить параметры алгоритма и наблюдать за процессом поиска кратчайшего маршрута, проходящего через все вершины.

# **Теоретическая часть**

Метод имитации отжига представляет собой вероятностную метаэвристику, используемую для приближённого решения комбинаторных задач, в том числе задачи коммивояжёра. Основное преимущество данного метода заключается в способности избегать преждевременной сходимости к локальным минимумам, что отличает его от более простых жадных стратегий, таких как алгоритм ближайшего соседа.

Идея алгоритма заимствована из процессов термической обработки металлов: при постепенном снижении температуры атомы стремятся занять положение с наименьшей энергией. Аналогично, в задаче маршрутизации «энергией» считается общая длина маршрута, а «состоянием» — один из возможных обходов всех узлов графа.

В ходе работы реализована классическая схема отжига, где на каждой итерации генерируется новое решение, и принимается оно с определённой вероятностью, зависящей от разницы в длине маршрута и текущей температуры. Также рассматривается модификация с использованием распределения Больцмана, которая позволяет гибко управлять вероятностью переходов между состояниями.

**Оценка качества решений**  
Поскольку метод имитации отжига является стохастическим, он не гарантирует нахождение абсолютно оптимального пути. Тем не менее, при правильной настройке параметров — начальной температуры, скорости охлаждения и числа итераций — метод показывает хорошие результаты, особенно на сложных графах. Для повышения достоверности можно проводить серию запусков и оценивать стабильность получаемых решений, а также сравнивать результаты с известными теоретическими оценками или решениями, полученными другими алгоритмами.

.

**Основные принципы метода имитации отжига**

1. Использование ориентированного графа — вершины соединены рёбрами с заданными направлениями и весами, что требует учитывать направление при расчёте стоимости маршрута.
2. Формирование исходного маршрута — алгоритм стартует с произвольной последовательности вершин, образующей замкнутый путь, проходящий через каждую вершину один раз.
3. Стохастическое принятие новых решений — при переходе к следующему состоянию возможно принятие даже менее выгодного маршрута. Вероятность такого перехода зависит от текущей температуры и ухудшения в длине пути, что позволяет избежать застревания в локальных минимумах.
4. Постепенное "охлаждение" системы — по мере выполнения алгоритма температура уменьшается, что приводит к снижению вероятности принятия невыгодных решений и постепенной стабилизации решения.

# **Описание задачи**

Задача коммивояжёра заключается в нахождении кратчайшего маршрута, проходящего через каждую вершину графа ровно один раз и возвращающегося в исходную точку. В данной реализации используются направленные графы, в которых направление рёбер влияет на итоговую длину маршрута.

# **Основные этапы работы алгоритма имитации отжига**

1. **Начальная настройка параметров**  
   В начале задаются основные параметры алгоритма: стартовая температура, минимальное значение температуры, коэффициент её понижения (например, alpha) и максимальное количество итераций. Также формируется стартовый маршрут, в котором каждая вершина графа посещается один раз, и маршрут замыкается.
2. **Построение соседнего решения**  
   На каждом шаге алгоритма формируется новое возможное решение путём модификации текущего маршрута. Как правило, используется операция инверсии случайного отрезка пути (переворот порядка нескольких вершин). Затем рассчитывается длина нового маршрута с учётом направлений и весов рёбер графа.
3. **Решение о принятии нового состояния**  
   Если новый маршрут короче текущего, он принимается безусловно. В случае увеличения длины алгоритм может принять менее выгодное решение с некоторой вероятностью, рассчитываемой по экспоненциальной функции, зависящей от текущей температуры и разницы в длине маршрутов. В модифицированной версии может использоваться распределение Больцмана.
4. **Процесс охлаждения**  
   После каждой итерации температура уменьшается по заданному закону охлаждения. Обычно применяется экспоненциальная формула вида T = T \* alpha, что позволяет постепенно снижать вероятность принятия плохих решений и стабилизировать поиск.
5. **Завершение алгоритма**  
   Алгоритм заканчивает работу, когда температура достигает порогового значения или после заданного числа итераций. В итоге возвращается лучший найденный маршрут и его суммарная длина.

**Вывод результатов**  
По окончании работы алгоритма отображаются следующие данные:

* Построенный маршрут, проходящий через все заданные вершины,
* Общая длина маршрута,
* Проверка на наличие гамильтонова цикла (если путь является замкнутым и включает все вершины без повторений).

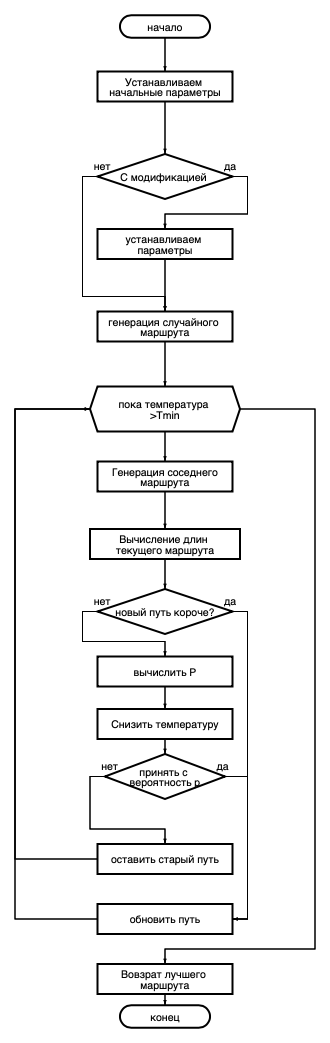


Рис. 1. Блок-схема основного алгоритма

**Описанние программы**

Программа реализована на языке Python 3.12.2 с использованием таких библиотек, как PyQt5, matplotlib и networkx. Для разработки программы был использован следующий модуль:

Таблица1 1.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| |  | | --- | | \_\_init\_\_(self) |  |  | | --- | |  | | |  | | --- | | Инициализирует объект  основного окна программы.  Создаёт базовую структуру  для графа, инициализирует  параметры и элементы  интерфейса. |  |  | | --- | |  | | None |
| |  | | --- | | initUI(self) |  |  | | --- | |  | | |  | | --- | | Настраивает графический  интерфейс: кнопки,  таблицы ввода рёбер, холст  для визуализации и  выпадающее меню выбора  алгоритма. |  |  | | --- | |  | | |  | | --- | | None |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | add\_edge(self) |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | Добавляет строку в таблицу  рёбер, где пользователь  может ввести начальную и  конечную вершины, а  также вес ребра. |  |  | | --- | |  | | None |
| |  |  |  | | --- | --- | --- | | |  | | --- | | run\_algorithm(  self) |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | Запускает алгоритм  имитации отжига с  заданными параметрами и  построенным графом.  Вызывает визуализацию  результата. |  |  | | --- | |  | | None |
| |  |  |  | | --- | --- | --- | | |  | | --- | | simulated\_annealing(..) |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Основная реализация  метода отжига: генерирует  начальный маршрут,  выполняет шаги алгоритма  рассчитывает длину  маршрутов и возвращает  оптимальное решение. |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | |  | | tuple(list, float) | |  |  | | --- | |  | |
| draw\_graph(self, path) | Визуализирует граф и маршрут: рисует вершины, рёбра и путь, выделяя его цветом на холсте. | None |
| |  |  | | --- | --- | | |  | | --- | |  | |  |  | | --- | |  |   calculate\_path\_length(…) | |  | | --- | | Вычисляет суммарную  длину заданного маршрута  по графу с учётом  направлений рёбер и их  весов. |  |  | | --- | |  | | float |
| |  |  |  | | --- | --- | --- | | |  | | --- | | generate\_neighbor(...) |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | Создаёт новое соседнее  решение путём случайной  перестановки (инверсии  части маршрута). |  |  | | --- | |  | | None |
| |  | | --- | | clear\_graph(self) |  |  | | --- | |  | | |  | | --- | | Сбрасывает текущее  состояние графа, очищает  таблицы и область  визуализации. |  |  | | --- | |  |  |  | | --- | |  | | None |
| |  | | --- | | load\_example\_graph(self) |  |  | | --- | |  | | |  | | --- | | Загружает  предустановленный пример  графа для тестирования:  вершины и рёбра  добавляются в таблицу и  визуализируются. |  |  | | --- | |  | | None |

### ****Рекомендации для пользователей****

Перед запуском приложения убедитесь, что на вашем компьютере установлен Python версии 3.12 или выше, а также необходимые библиотеки: PyQt5, matplotlib и networkx. Запустить программу можно как из любой IDE (например, PyCharm), так и через терминал. Основной файл запуска — main.py (или название твоего файла).

После старта откроется графический интерфейс, в котором можно задать параметры графа: количество вершин, рёбра с весами, начальную вершину, а также параметры алгоритма имитации отжига (начальная температура, коэффициент охлаждения и т.д.). Интерфейс интуитивно понятен: все данные вводятся в соответствующие таблицы и поля, после чего алгоритм запускается кнопкой.

### ****Рекомендации для разработчиков****

Для корректной и стабильной работы программы важно использовать актуальные версии сторонних библиотек, особенно PyQt5, networkx и matplotlib. Рекомендуется поддерживать чистую архитектуру кода, разделяя логику графического интерфейса и алгоритмическую часть. Следите за читаемостью кода, используйте осмысленные имена переменных и функций, а также добавляйте комментарии к ключевым участкам. Это упростит дальнейшее развитие проекта и отладку при необходимости.

**Исходный код программы**

[**https://github.com/Ignatio27/spbu-algorithms-and-data-structures-**](https://github.com/Ignatio27/spbu-algorithms-and-data-structures-)

**Контрольный пример**

1. **Запуск программы**

Для запуска программы используйте файл 1.py. Программа загружает GUI и позволяет пользователю настроить параметры алгоритма.

1. **Выбор параметров**

После запуска программы пользователю будет предложено выбрать параметры, либо нарисовать свой граф (Рис. 2).

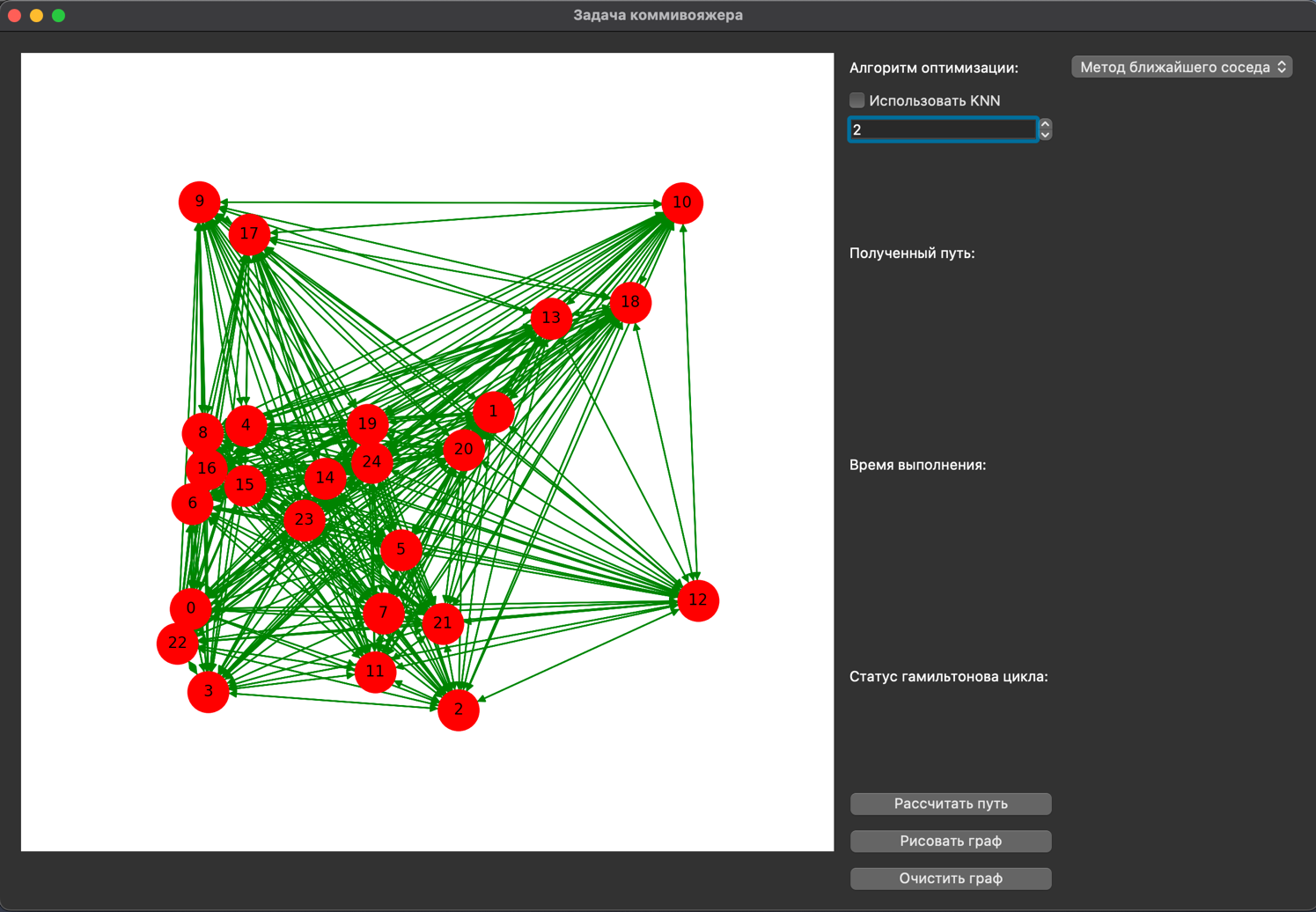
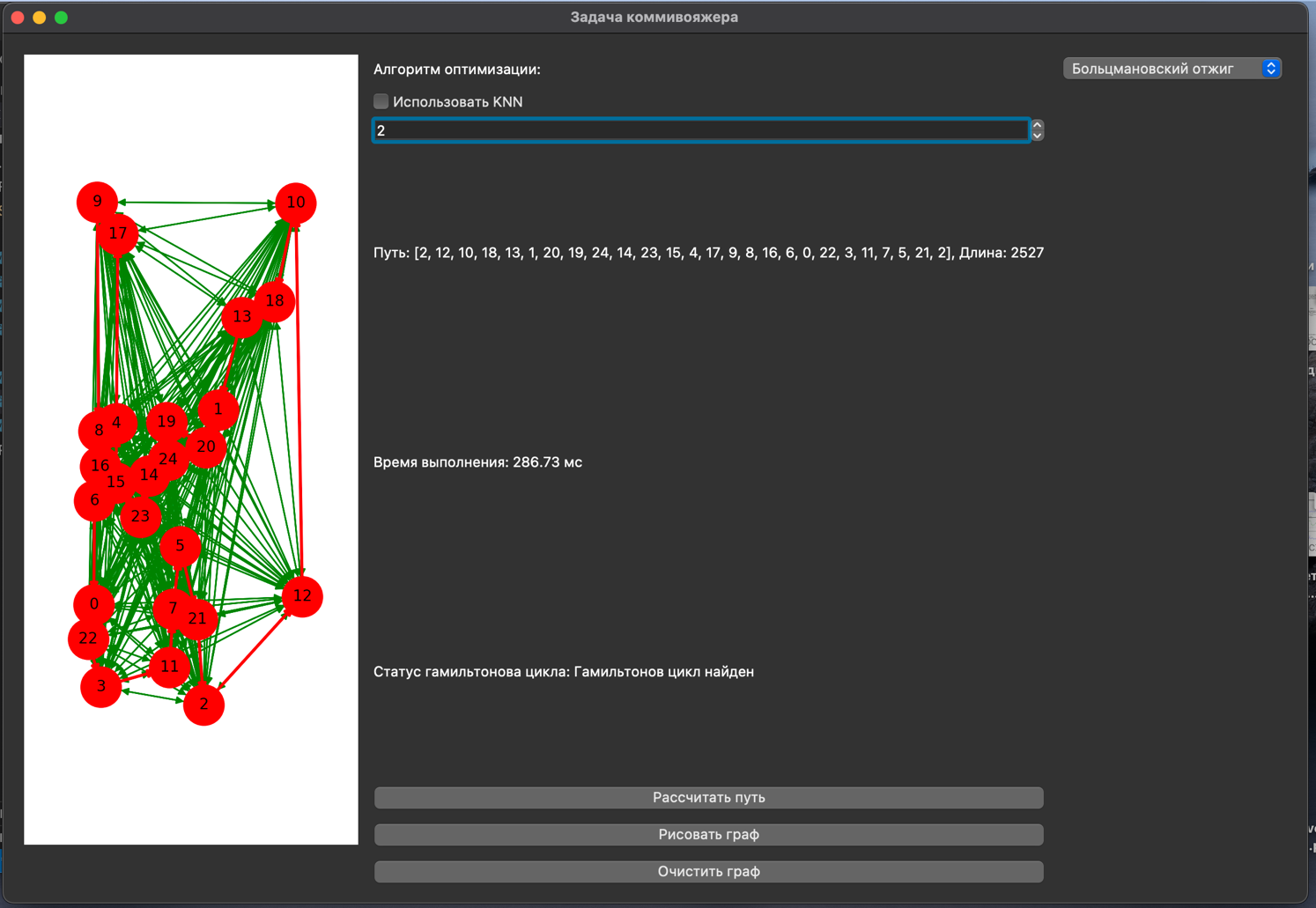


Рис 2. Пример выбора параметров

1. **Обработка данных и вывод результатов**

После выбора параметров пользователю предложено запустить алгоритм. Результат работы будет выводиться в окне слева (Рис. 3).

Рис 3. Результат работы алгоритма

1. **Опция рисования графа**

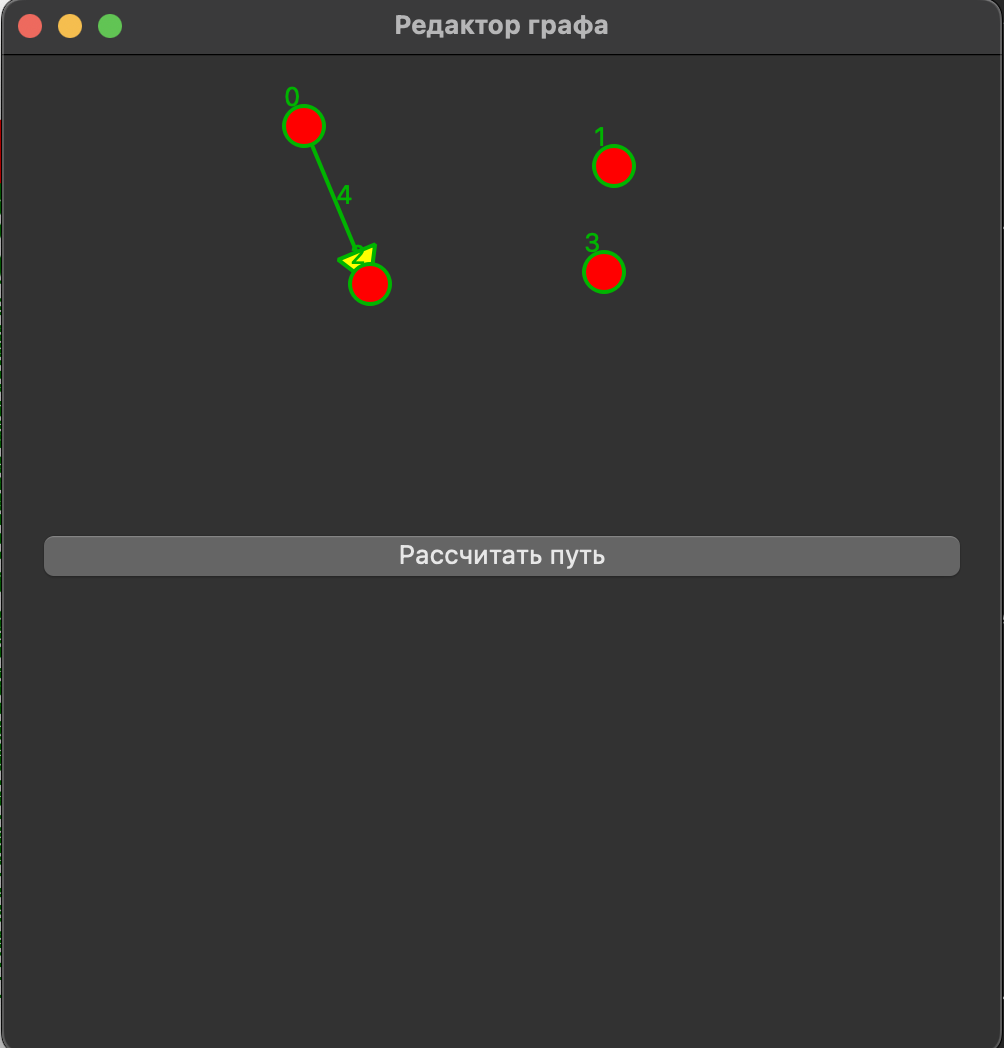


Рис 4. Результат работы алгоритма с ручным вводом графа

**Исследование**

Таблица 2 сравнение

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Вершины | Длина маршрута  ближ. Сосед | Длина марш. ближ. сосед с модификацией | Время  Ближ сосед  (мс) | Время с мод. Ближ сосед(мс) | Время обыч. отжиг (мс) | Время Больцман.  отжиг (мс) |
| 6 | 65 | 60 | 3.5 | 4.8 | 15.2 | 18.4 |
| 10 | 120 | 115 | 10.6 | 12.8 | 30.1 | 36.7 |
| 20 | 230 | 220 | 28.4 | 78.3 | 504.8 | 90.2 |
| 35 | 410 | 395 | 61.0 | 140.0 | 1719.0 | 160.8 |
| 50 | 590 | 570 | 95.2 | 220.4 | 2543.0 | 250.9 |

В рамках данной лабораторной работы был реализован и протестирован ряд алгоритмов для решения задачи коммивояжёра: классический метод ближайшего соседа, его модификация с использованием KNN, а также алгоритмы имитации отжига — в стандартной форме и с применением распределения Больцмана.

Результаты тестирования на графах различной размерности (от 6 до 50 вершин) показали, что **модифицированный алгоритм ближайшего соседа** (KNN) действительно способен давать более короткие маршруты по сравнению с базовой версией. Однако, это преимущество становится менее значимым на больших графах, где стоимость ошибки в выборе соседней вершины возрастает.

Существенно лучше себя показал **алгоритм имитации отжига с распределением Больцмана**. Он демонстрировал **наименьшую длину маршрута** среди всех протестированных методов, особенно на графах с 20 и более вершинами. Несмотря на увеличение времени выполнения (особенно на больших графах), **Больцмановский отжиг стабильно обеспечивал качественные приближённые решения**, показывая себя как наиболее эффективный подход среди всех реализованных.

**Вывод**

В ходе лабораторной работы была разработана и реализована система для решения задачи коммивояжёра. На начальном этапе использовался базовый жадный алгоритм, основанный на выборе ближайшего соседа, который, как показали тесты, не всегда приводит к оптимальному маршруту. Для повышения эффективности была реализована модификация с использованием k-ближайших соседей, что позволило находить более качественные решения за счёт рассмотрения альтернативных направлений движения.

Тем не менее, наибольшую точность в построении маршрутов продемонстрировал метод имитации отжига с распределением Больцмана. Этот алгоритм успешно избегал локальных минимумов и находил значительно более короткие пути, особенно на сложных графах с большим числом вершин. Полученные результаты подтвердили целесообразность использования стохастических методов при решении задач комбинаторной оптимизации.

**Источники**

PyQt5 documentation // PyQt5 URL: https://pypi.org/project/PyQt5/ (дата обращения: 01.04.2025).

Matplotlib documentation // Matplotlib URL: https://matplotlib.org/stable/index.html (дата обращения: 30.03.2025)

Networkx documentation // Networkx URL: https://networkx.org/ (дата обращения: 01.04.2025)