**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 метода имитации отжига»**

**Студент гр. 23Б15-пу**

**Абрахин Е.Д.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2025 г.**

Оглавление

1. [Цель работы 3](#_Toc195299963)
2. [Описание задачи (формализация задачи) 3](#_Toc195299964)
3. [Теоретическая часть 4](#_Toc195299965)
4. [Основные шаги программы 6](#_Toc195299966)
5. [Блок схема программы 8](#_Toc195299967)
6. [Описание программы 9](#_Toc195299968)
7. [Рекомендации пользователя 11](#_Toc195299969)
8. [Рекомендации программиста 11](#_Toc195299970)
9. [Исходный код программы 12](#_Toc195299971)
10. [Контрольный пример 13](#_Toc195299972)
11. [Анализ результатов работы алгоритма 16](#_Toc195299973)
12. [Вывод 20](#_Toc195299974)
13. [Источники 21](#_Toc195299975)

# Цель работы

Целью данной лабораторной работы является исследование метода имитации отжига для решения задачи коммивояжера, а также анализ эффективности его модифицированной версии с больцмановским охлаждением. В ходе работы будет разработана программа, реализующая алгоритм имитации отжига, с возможностью визуализации найденного маршрута. Программа будет протестирована на заданном взвешенном орграфе для оценки качества полученного решения и сравнения его с оптимальным маршрутом.

# Описание задачи (формализация задачи)

1. Изучение задачи коммивояжера и метода имитации отжига.

Ознакомиться с алгоритмом имитации отжига и его применением для поиска приближенного решения. Исследовать влияние выбора схемы охлаждения на сходимость алгоритма.

1. Разработка программы для нахождения кратчайшего гамильтонова цикла.

Написать программу, реализующую алгоритм имитации отжига для задачи коммивояжера. Добавить возможность использования Больцмановской схемы охлаждения в качестве модификации алгоритма. Реализовать визуализацию построенного маршрута на графе.

1. Тестирование программы на взвешенном орграфе.

Провести тестирование программы на контрольном примере, представленном во взвешенном орграфе. Сравнить полученный маршрут с оптимальным решением, оценив точность и эффективность метода.

1. Анализ результатов

Оценить влияние различных схем охлаждения на качество решения. Проанализировать, при каких параметрах алгоритм работает достаточно эффективно.

# Теоретическая часть

**Алгоритм имитации отжига**

Алгоритм имитации отжига (Simulated Annealing, SA) — это стохастический метод оптимизации, вдохновленный процессом термического отжига в металлургии. Он применяется для приближенного решения задачи коммивояжера, когда необходимо найти кратчайший гамильтонов цикл в графе. В отличие от жадных алгоритмов, имитация отжига позволяет выходить из локальных минимумов за счет принятия ухудшающих решений с определенной вероятностью.

**Параметры алгоритма и их роль**

1. Начальная температура

Определяет вероятность принятия неоптимальных решений. Чем выше температура, тем выше вероятность исследования различных состояний.

1. Функция охлаждения

Контролирует снижение температуры. В коде реализованы два варианта:

* + **Стандартная охлаждение**: , где k — номер итерации.
  + **Больцмановское охлаждение**: , где температура снижается медленнее, что позволяет алгоритму дольше исследовать пространство решений.

1. **Функция стоимости (длины пути)**

Определяет, насколько хорош текущее решение. В коде стоимость пути вычисляется как сумма весов рёбер в текущем маршруте.

1. **Критерий вероятности перехода**

Если новое решение хуже текущего, оно принимается с вероятностью:

где ΔE — разница между длинами маршрутов.

**Принцип работы**

1. Генерируется начальный маршрут с помощью поиска в глубину.
2. На каждой итерации выбираются два случайных узла, и производится перестановка.
3. Вычисляется разница в стоимости пути после перестановки.
4. Если новый путь короче — он принимается.
5. Если новый путь длиннее, он принимается с вероятностью .
6. Температура снижается по заданной схеме (экспоненциальной или больцмановской).
7. Процесс повторяется, пока температура не достигнет порогового значения или не будет выполнено максимальное число итераций.

**Оценка эффективности**

Основные параметры, влияющие на качество решения:

1. Скорость сходимости
   * Экспоненциальное охлаждение быстрее приводит к стабилизации, но может застревать в локальных минимумах.
   * Больцмановское охлаждение дольше исследует пространство решений, что повышает вероятность нахождения глобального оптимума.
2. Качество решения
   * Возможность выхода из локальных минимумов за счёт принятия ухудшающих решений на ранних итерациях.
   * Больцмановская схема может давать более точные результаты за счёт плавного снижения температуры.
3. Число итераций
   * В коде используется ограничение по итерациям и температуре, что позволяет контролировать время работы алгоритма.

# Основные шаги программы

Основные шаги программы The\_simulated\_annealing\_algorithm.pyw

1. Запуск программы:

Инициализация графического интерфейса (GUI) с элементами управления:

Область визуализации графа.

Таблица для редактирования весов рёбер.

Поле для вывода результатов.

Флажок для включения модификации.

Кнопки управления: «Рассчитать», «Отмена», «Очистить», «Загрузить», «Сохранить».

1. Настройка графа:

Добавление вершин кликом по области визуализации.

Создание рёбер путем выбора двух вершин (автоматически рассчитывается расстояние).

Отображение данных о вершинах и рёбрах в таблице.

Возможность редактирования весов рёбер.

1. Запуск алгоритма:

При нажатии кнопки «Рассчитать» выполняется алгоритм поиска пути.

Если включена модификация, алгоритм использует снижение температуры методом Больцмана.

1. Поиск решения:

Генерация начального маршрута с помощью поиска в глубину.

Итеративное улучшение маршрута методом имитации отжига:

Перестановка двух случайных узлов.

Вычисление разницы в стоимости пути.

Если новый путь короче — он принимается.

Если длиннее, то принимается с вероятностью, зависящей от температуры.

Охлаждение температуры по экспоненциальному или Больцмановскому закону.

1. Обновление интерфейса:

Визуализация найденного пути на графе.

Заполнение таблицы актуальными данными.

Вывод результатов: маршрут, длина пути, время выполнения.

1. Дополнительные возможности:

Отмена последнего действия (удаление вершины или ребра).

Очистка графа для нового расчёта.

Загрузка и сохранение графа в Excel.

# Блок схема программы

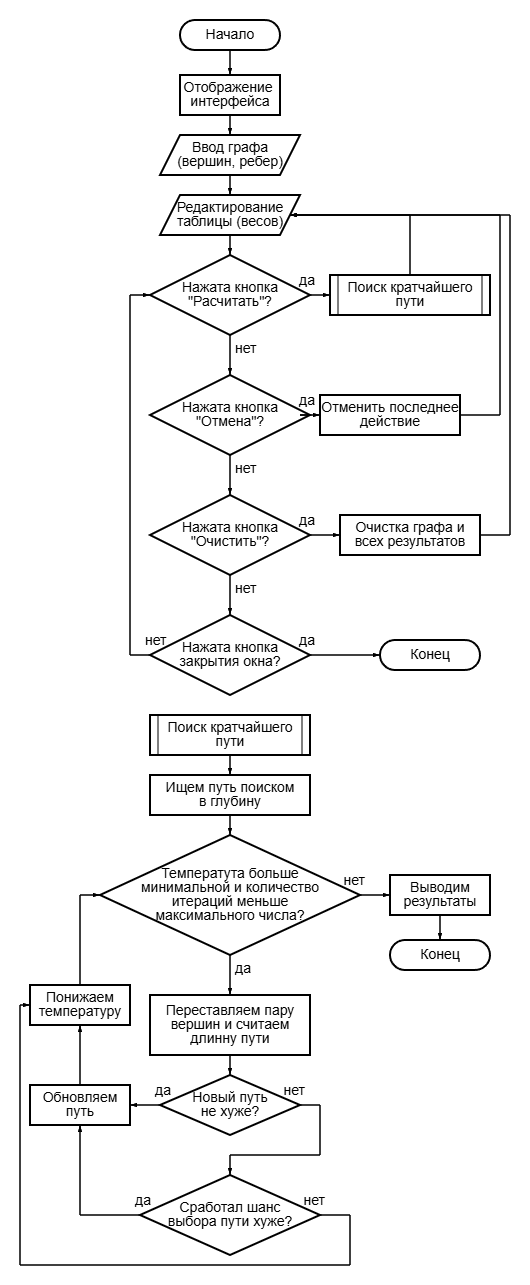


Рис 1. Блок-схема The\_nearest\_neighbor\_method.py

# Описание программы

Программа написана на языке Python 3.12.6 и использует библиотеки numpy [[2]](#_Источники_1) и PyQt5 [[3]](#_Источники_1) для реализации графического интерфейса решения задачи о коммивояжере с использованием алгоритма ближайшего соседа. В программе реализован только один класс – **TSPApp**, который отвечает за создание интерфейса, обработку графических событий, построение графа, вычисление кратчайшего пути и визуализацию результатов. Класс включает в себя 18 методов, каждый из которых выполняет свою задачу в процессе работы приложения:

Таблица 1. TSPApp

|  |  |  |
| --- | --- | --- |
| Метод | Описание | Возвращаемое значение |
| \_\_init\_\_ | Инициализация интерфейса и параметров приложения. | None |
| initUI | Настройка графического интерфейса приложения, создание и размещение виджетов. | None |
| addEdge | Добавляет ребро между двумя выбранными вершинами, вычисляя расстояние между ними. | None |
| addNode | Добавляет новую вершину в граф по заданной позиции. | None |
| addNodeOrEdge | Определяет, следует ли добавить новую вершину или создать ребро между существующими вершинами, в зависимости от положения курсора. | None |
| clearGraph | Очищает граф, таблицу ребер, историю действий и текстовое поле с результатом. | None |
| drawArrow | Отрисовывает стрелку между вершинами, корректируя начало и конец линии, чтобы не накладываться на узлы. | None |
| drawSolution | Отображает найденный маршрут и вершины графа на отдельной сцене для визуализации решения. | None |
| findClickedNode | Определяет, на какую вершину произведён клик; возвращает идентификатор вершины или None, если клик не попал на узел. | int или None |
| getDistance | Возвращает расстояние между двумя вершинами, если соответствующее ребро существует, иначе — float('inf'). | float |
| generateTestGraph | Генерирует случайный тестовый граф с вершинами и рёбрами, а также сохраняет его в Excel. | None |
| loadGraphFromExcel | Загружает граф из файла Excel, обновляет список вершин и рёбер. | None |
| lockColumns | Делает первые два столбца таблицы недоступными для редактирования. | None |
| redrawGraph | Перерисовывает граф и таблицу ребер после внесённых изменений. | None |
| saveGraphToExcel | Сохраняет текущий граф (матрицу смежности) в файл Excel | None |
| solveTsp | Вычисляет кратчайший путь (гамильтонов цикл) с помощью алгоритма ближайшего соседа, измеряет время выполнения и выводит результаты. | None |
| undoAction | Отменяет последнее действие (добавление вершины или ребра) и обновляет граф. | None |
| updateEdgeWeight | Обновляет вес ребра в таблице при изменении значения, если ввод корректен. | None |

# Рекомендации пользователя

Программа позволяет решить задачу о коммивояжере с помощью алгоритма имитации отжига. С ее помощью можно построить граф, добавить узлы и ребра, а затем найти кратчайший гамильтонов цикл.

1. Запустите программу, например, выполнив команду:

python The\_simulated\_annealing\_algorithm.pyw

1. В графическом интерфейсе:
   * Добавление узлов:

Щелкайте на области графа для создания новых узлов.

* + Добавление ребер:

Для соединения узлов выберите сначала один узел, затем второй – программа автоматически создаст ребро между ними.

* + Редактирование:

При необходимости измените вес ребра в таблице.

* + Выбор модификации:

Для использования модификации Больцмановского отжига установите галочку в «Использовать модификацию».

1. Нажмите кнопку **«Рассчитать»** для выполнения алгоритма. Программа вычислит кратчайший путь (гамильтонов цикл) и отобразит:
   * В поле **«Рассчитанный путь»** — последовательность вершин кратчайшего маршрута.
   * На графике — построенный маршрут с выделенными ребрами.
2. Используйте кнопку **«Отмена»** для удаления последнего действия или **«Очистить»** для сброса всего графа.

# Рекомендации программиста

Для корректной работы программы убедитесь, что установлены следующие компоненты и выполнены необходимые шаги:

* Требования:
  + Python версии 3.12.0 или выше.
  + Необходимые библиотеки: PyQt5, numpy.
* Шаги по установке:
  + Выполните команду для запуска программы:

python The\_simulated\_annealing\_algorithm.pyw

* + Проверьте, что графический интерфейс запускается корректно и результаты алгоритма (нахождение кратчайшего гамильтонова цикла) отображаются в соответствии с заданием.

# Исходный код программы

<https://github.com/FasterXaos/Algorithms_and_Data_Structures>

# Контрольный пример

The\_simulated\_annealing\_algorithm.py

1. Запуск программы и формирование графа

Для запуска программы откройте файл, содержащий код класса TSPApp. Программа откроет графический интерфейс, где в левой части окна расположены таблица для ввода данных о ребрах и панель для отображения вычисленного маршрута, в правой – область для построения графа (Рис. 2).

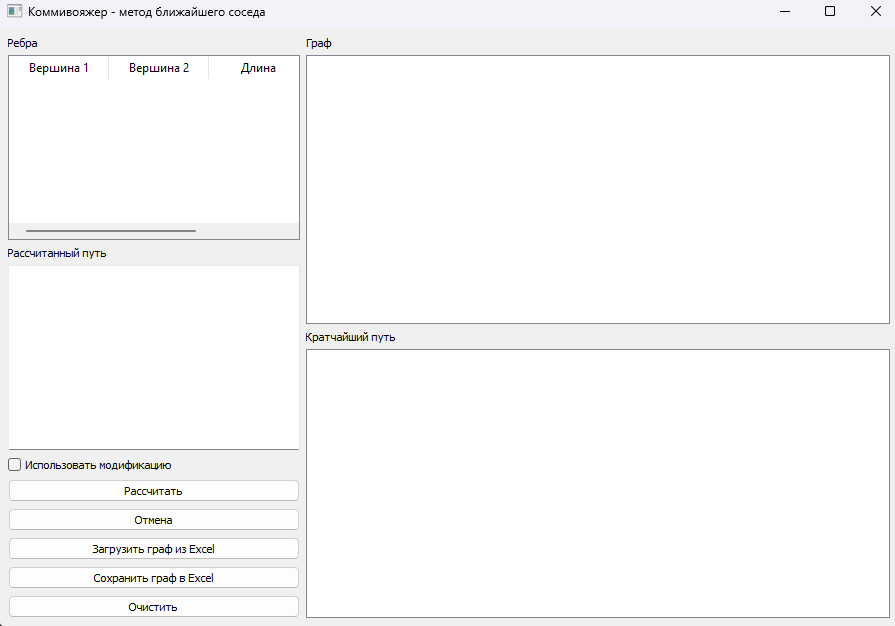


Рис 2. Пример окна программы

1. Ввод исходных данных

После запуска программы можно добавить вершины и соединить их ребрами следующим образом:

* Добавление вершин:

щёлкните в области графа (правой части окна) для создания новой вершины.

* Соединение вершин:

после выбора первой вершины щёлкните по другой, чтобы добавить ребро между ними. Вес ребра вычисляется автоматически как расстояние между центрами вершин.

1. Запуск алгоритма

После формирования графа нажмите кнопку «Рассчитать». Программа выполнит вычисление кратчайшего гамильтонова цикла с помощью алгоритма имитации отжига, определяя маршрут, проходящий через все вершины и возвращающийся в исходную точку (Рис. 3). Если поставить галочку «Использовать модификацию», то понижение температуры будет рассчитываться с помощью Больцмановской модификации.

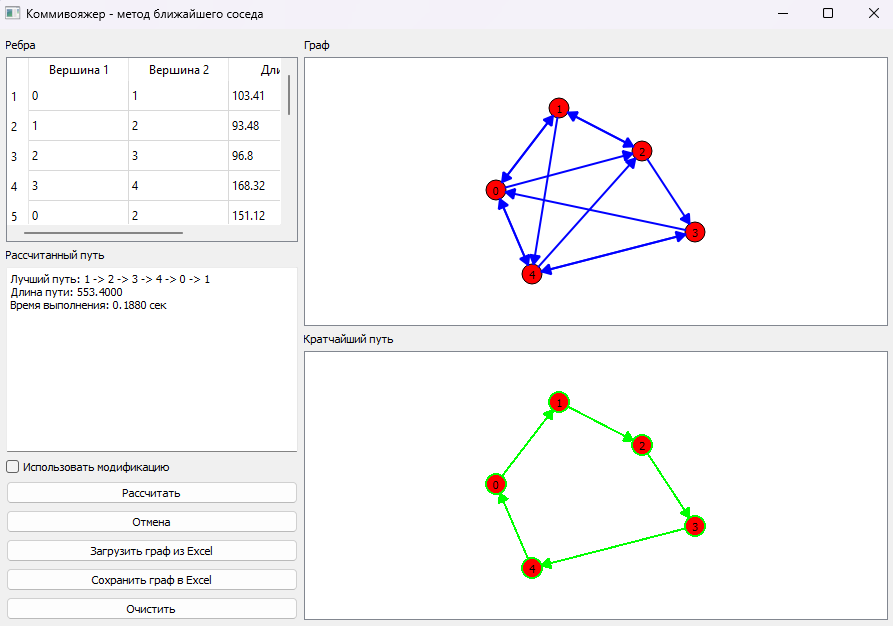


Рис 3. Пример результатов программы

1. Просмотр результатов

После завершения расчёта в нижней части окна будут выведены следующие данные:

* Найденный кратчайший маршрут (последовательность вершин);
* Общая длина маршрута с заданной точностью;
* Время выполнения алгоритма.

Дополнительно в правой части окна отображается графическое представление маршрута с выделенными вершинами и соединяющими их стрелками.

1. Загрузка и сохранение графа.

Для сохранения нарисованного графа или загрузки его из файла Excel нужно нажать кнопки «Загрузить граф в Excel» и «Сохранить граф в Excel». Таблица смежность должна иметь следующий вид для корректной загрузки:

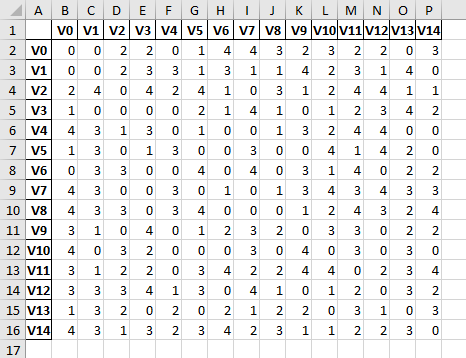


Рис 4. Пример задания таблицы графа

# Анализ результатов работы алгоритма

Анализ работы алгоритма имитации отжига и его модификации приведены примеры результатов для нахождения гамильтонова цикла методом ближайших соседей на разных графах:

Таблица 2. Тесты на тридцати вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | 43 | 0.0054 сек | 34 | 0.1655 сек |
| 2 | Не удалось найти путь | | 32 | 0.1635 сек |
| 3 | 37 | 0.0057 сек | 33 | 0.1643 сек |
| 4 | Не удалось найти путь | | 32 | 0.1607 сек |

Таблица 3. Тесты на пятидесяти вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | Не удалось найти путь | | 52 | 1.9866 сек |
| 2 | 59 | 0.0423 сек | 52 | 1.9910 сек |
| 3 | 58 | 0.0396 сек | 52 | 1.9427 сек |
| 4 | Не удалось найти путь | | 54 | 2.0333 сек |

Таблица 4. Тесты на пятнадцати вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | 27 | 0.0004 сек | 18 | 0.0061 сек |
| 2 | Не удалось найти путь | | 18 | 0.0059 сек |
| 3 | 21 | 0.0004 сек | 19 | 0.0060 сек |
| 4 | 24 | 0.0005 сек | 20 | 0.0063 сек |

Таблица 5. Тесты на шести вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | 12 | 0.0001 сек | 12 | 0.0002 сек |
| 2 | Не удалось найти путь | | 13 | 0.0002 сек |
| 3 | 9 | 0.0001 сек | 9 | 0.0002 сек |
| 4 | 11 | 0.0001 сек | 11 | 0.0002 сек |

Для схождения метода нужно подобрать оптимальную температуру для обоих формул расчета температуры.

Стандартная: , где k — номер итерации.

Больцмановская: .

Вероятность выбора неоптимального решения: , где — на сколько ухудшилось решение.

Если для стандартной формулы достаточно взять , и , для относительно плавного снижения температуры и вероятности выбора неоптимального решения к нулю, то для Больцмановской модификации все сложнее.

Таблица 6. Подбор температур при = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Итерация | T (Эксп.) | P (Эксп.) | T (Больцм.) | P (Больцм.) |
| 0 | 1000 | 0,9990005 | 4,328085123 | 0,793700526 |
| 1000 | 367,6954248 | 0,997284053 | 0,434168903 | 0,099933422 |
| 2000 | 135,1999254 | 0,992630833 | 0,394638081 | 0,079343614 |
| 3000 | 49,712394 | 0,980085263 | 0,37467057 | 0,069320726 |
| 4000 | 18,27901983 | 0,946762008 | 0,361683295 | 0,062985557 |
| 5000 | 6,72111196 | 0,861754455 | 0,352212176 | 0,058472559 |
| 6000 | 2,471322117 | 0,667215827 | 0,34483361 | 0,055026007 |
| 7000 | 0,908693836 | 0,332711121 | 0,338831779 | 0,052270818 |
| 8000 | 0,334122566 | 0,050141128 | 0,333798916 | 0,049995834 |
| 9000 | 0,122855339 | 0,000291738 | 0,32948198 | 0,048071425 |
| 10000 | 0,045173346 | 2,43246E-10 | 0,325713789 | 0,046412794 |

Как видно из расчетов, чтобы Больцмановский вариант хотя бы сходился к 10000 итераций с вероятностью 4.6% приходится брать , причем из-за того, что логарифм существенно растет лишь в начале, только на первой тысяче итераций имеется высокий шанс выбора неоптимального решения, за тем рост логарифма замедляется и снижение шанса идет крайне медленно. С этими значениями были получены следующие результаты:

Таблица 7. Тесты на шести вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | 13 | 0.1877 сек | 13 | 0.2042 сек |
| 2 | 13 | 0.1578 сек | 13 | 0.1731 сек |
| 3 | 9 | 0.1638 сек | 10 | 0.1622 сек |
| 4 | 9 | 0.1805 сек | 9 | 0.1876 сек |

Таблица 8. Тесты на пятнадцати вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | 19 | 0.8301 сек | 20 | 0.8162 сек |
| 2 | 18 | 0.8063 сек | 19 | 0.8463 сек |
| 3 | 18 | 0.7855 сек | 17 | 0.8522 сек |
| 4 | 21 | 0.8295 сек | 21 | 0.8012 сек |

Таблица 9. Тесты на тридцати вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | 35 | 4.6611 сек | 33 | 4.6212 сек |
| 2 | 35 | 4.6814 сек | 37 | 4.7954 сек |
| 3 | 34 | 4.5799 сек | 32 | 4.6005 сек |
| 4 | 34 | 4.6718 сек | 36 | 4.4452 сек |

Таблица 10. Тесты на пятидесяти вершинах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер графа | Расстояние без модификации | Время без модификации | Расстояние с модификацией | Время с модификацией |
| 1 | 55 | 19.3073 сек | 56 | 19.0850 сек |
| 2 | 55 | 19.4550 сек | 57 | 19.0367 сек |
| 3 | 56 | 19.2024 сек | 57 | 19.5447 сек |
| 4 | 54 | 19.2696 сек | 57 | 19.8725 сек |

Алгоритм имитации отжига показывает результаты значительно лучше, чем метод ближайшего соседа, но затрачивает намного больше времени из-за необходимости расчетов большого числа итераций. Больцмановская модификация зачастую находит пути несколько хуже, чем стандартный алгоритм. Скорее всего, это происходит из-за раннего и быстрого понижения температуры и схождения в локальный минимум. Поскольку единственное отличие модификации — это метод расчета температуры, то фактическое время в обоих случаях сопоставимо.

# Вывод

В ходе работы был реализован алгоритм имитации отжига для решения задачи коммивояжера, а также создан графический интерфейс для визуализации построения графа и вывода решения. Экспериментальные результаты показали, что метод эффективнее справляется с поиском оптимального решения нежели метод ближайшего соседа, но его затрачиваемое время значительно выше.

Преимуществом алгоритма являются его точность, но может возникнуть проблема с поиском начального пути для инициализации поиска через имитацию отжига. Его модификация показала отсутствие явных преимуществ и проявила недостатки в точности и скорости схождения.

# Источники

1. NumPy documentation // numpy.org URL: https://numpy.org/doc/stable/ (дата обращения: 04.03.2025).
2. PyQt5 Reference Guide // www.riverbankcomputing.com URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата обращения: 04.03.2025).
3. Алгоритм имитации отжига // ru.wikipedia.org URL: https://ru.wikipedia.org/wiki/Алгоритм\_имитации\_отжига (дата обращения: (04.03.2025).