Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ
по лабораторной работе №6

Название: Муравьиный алгоритм

Дисциплина: Анализ алгоритмов

Студент ИУ7-54б _____ Ларин В.Н.
(группа) (подпись, дата) (фамилия, и.о.)

Преподаватель _____ Волкова Л.Л.
(подпись, дата) (фамилия, и.о.)

Москва, 2021

СОДЕРЖАНИЕ

Введение 2

1 Аналитический раздел 3

1.1 Задача коммивояжера 3

1.2 Алгоритм полного перебора 3

1.3 Муравьиный алгоритм 3

1.4 Вывод 6

2 Конструкторский раздел 7

2.1 Схемы алгоритмов 7

2.2 Классы эквивалентности 10

2.3 Описание используемых типов данных 10

2.4 Структура ПО 10

2.5 Вывод 11

3 Технологический раздел 12

3.1 Средства реализации 12

3.2 Листинги кода 12

3.3 Функциональные тесты 16

3.4 Вывод 16

4 Экспериментальный раздел 17

4.1 Технические характеристики 17

4.2 Демонстрация работы программы 17

4.3 Время выполнения алгоритмов 17

4.4 Автоматическая параметризация 20

4.4.1 Класс данных 1 20

4.4.2 Класс данных 2 27

4.5 Вывод 34

Заключение 35

Список литературы 36

ВВЕДЕНИЕ

Задача поиска оптимальных маршрутов является одной из важных. Муравьиный алгоритм – один из эффективных полиномиальных алгоритмов для нахождения приближённых решений задачи коммивояжёра, а также решения аналогичных задач поиска маршрутов на графах. Суть подхода заключается в анализе и использовании модели поведения муравьёв, ищущих пути от колонии к источнику питания, и представляет собой метаэвристическую оптимизацию.

Целью данной лабораторной работы является изучение муравьиного алгоритма на примере задачи коммивояжера.

Для достижения поставленной цели необходимо выполнить следующие задачи:

— исследовать задачу коммивояжера;

— изучить алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжера;

— провести параметризацию муравьиного алгоритма на двух классах данных;

— привести схемы используемых алгоритмов;

— описать используемые структуры данных;

— описать структуру разрабатываемого ПО;

— определить средства программной реализации;

— провести сравнительный анализ времени работы алгоритмов;

— провести функциональное тестирование;

— описать и обосновать полученные результаты в отчете о выполненной лабораторной работе.

1 Аналитический раздел

В данном разделе будет представлено описание задачи коммивояжера и используемых для её решения алгоритмов (полного перебора и муравьиного алгоритма).

1.1 Задача коммивояжера

Коммивояжёр — бродячий торговец. Задача коммивояжёра [1] — одна из самых важных задач транспортной логистики, отрасли, занимающейся планированием транспортных перевозок. Коммивояжёру, чтобы распродать товары, следует объехать n пунктов и в конце концов вернуться в исходный пункт. Требуется определить наиболее выгодный маршрут объезда. В качестве меры выгодности маршрута может служить суммарное время в пути, суммарная стоимость дороги, или, в простейшем случае, длина маршрута.

В описываемой задаче рассматривается несколько городов и матрица попарных расстояний между ними (матрица смежности).

1.2 Алгоритм полного перебора

Алгоритм полного перебора [2] для решения задачи коммивояжера предполагает рассмотрение всех возможных путей в графе и выбор наименьшего из них. Смысл перебора состоит в том, что мы перебираем все варианты объезда городов и выбираем оптимальный. Однако, при таком подходе количество возможных маршрутов очень быстро возрастает с ростом n (сложность алгоритма равна $n!$).

Алгоритм полного перебора гарантирует точное решение задачи, однако, уже при небольшом числе городов будут большие затраты по времени выполнения.

1.3 Муравьиный алгоритм

Муравьиный алгоритм [3] – метод решения задач коммивояжера, в основе которого лежит моделирование поведения колонии муравьев.

Каждый муравей определяет для себя маршрут, который необходимо пройти на основе феромона, который он ощущает во время прохождения, каждый муравей оставляет феромон на своем пути, чтобы остальные муравьи могли по нему ориентироваться. В результате при прохождении каждым муравьем различного маршрута наибольшее число феромона остается на оптимальном пути.

Пусть муравей имеет следующие характеристики:

— зрение – способен определить длину ребра;

— память – запоминает пройденный маршрут;

— обоняние – чувствует феромон.

Также введем целевую функцию (1.1).

$$\eta_{ij} = 1/D_{ij}, \tag{1.1}$$

где D_{ij} — расстояние из текущего пункта i до заданного пункта j.

А также понадобится формула вычисления вероятности перехода в заданную точку (1.2).

$$P_{kij} = \begin{cases} \frac{\tau_{ij}^a \eta_{ij}^b}{\sum_{q=1}^m \tau_{iq}^a \eta_{iq}^b}, \text{вершина не была посещена ранее муравьем k,} \\ 0, \text{иначе} \end{cases} \tag{1.2}$$

где a – параметр влияния длины пути, b – параметр влияния феромона, τ_{ij} – расстояния от города i до j, η_{ij} – количество феромонов на ребре ij.

После завершения движения всех муравьев, формула обновляется феромон по формуле (1.3):

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \Delta\tau_{ij}. \tag{1.3}$$

При этом

$$\Delta\tau_{ij} = \sum_{k=1}^{N} \tau_{ij}^k, \tag{1.4}$$

где

$$\Delta\tau_{ij}^k = \begin{cases} Q/L_k, \text{ребро посещено k-ым муравьем,} \\ 0, \text{иначе} \end{cases} \tag{1.5}$$

Описание поведения муравьев при выборе пути.

1) Муравьи имеют собственную «память». Поскольку каждый город может быть посещён только один раз, то у каждого муравья есть списокуже посещенных городов - список запретов. Обозначим через J_{ik} список городов, которые необходимо посетить муравью k, находящемуся в городе i.

2) Муравьи обладают «зрением» - желание посетить город j, если муравей находится в городе i. Будем считать, что видимость обратно пропорциональна расстоянию между городами.

3) Муравьи обладают «обонянием» - они могут улавливать след феромона, подтверждающий желание посетить город j из города i на основании опыта других муравьёв. Количество феромона на ребре (i, j) в момент времени t обозначим через $\tau_{i,j}(t)$.

4) Пройдя ребро (i, j), муравей откладывает на нём некоторое количество феромона, которое должно быть связано с оптимальностью сделанного выбора. Пусть $T_k(t)$ есть

маршрут, пройденный муравьем k к моменту времени t, $L_k(t)$ - длина этого маршрута, а Q - параметр, имеющий значение порядка длины оптимального пути. Тогда откладываемое количество феромона может быть задано формулой (1.5).

1.4 Вывод

В этом разделе было рассмотрено описание задачи коммивояжера и используемых для её решения алгоритмов (полного перебора и муравьиного алгоритма).

На вход программе будет поступать матрица смежности и её размер. Также для работы муравьиного алгоритма будут поступать коэффициенты и количество дней. При попытке задать некорректные данные, будет выдано сообщение об ошибке. Реализуемое ПО будет давать возможность выбрать алгоритм решения задачи коммивояжера (полного перебора или муравьиный алгоритм) и вывести для него результат вычисления, а также возможность произвести сравнение алгоритмов по затраченному времени.

2 Конструкторский раздел

В данном разделе будут приведены схемы алгоритма полного перебора и муравьиного алгоритма для решения задачи коммивояжера, приведено описание используемых типов данных, классов эквивалентности, а также описана структура ПО.

2.1 Схемы алгоритмов

На рис. 2.1 - 2.2 приведены схемы алгоритмов для решения задачи коммивояжера (полного перебора и муравьиного алгоритма).

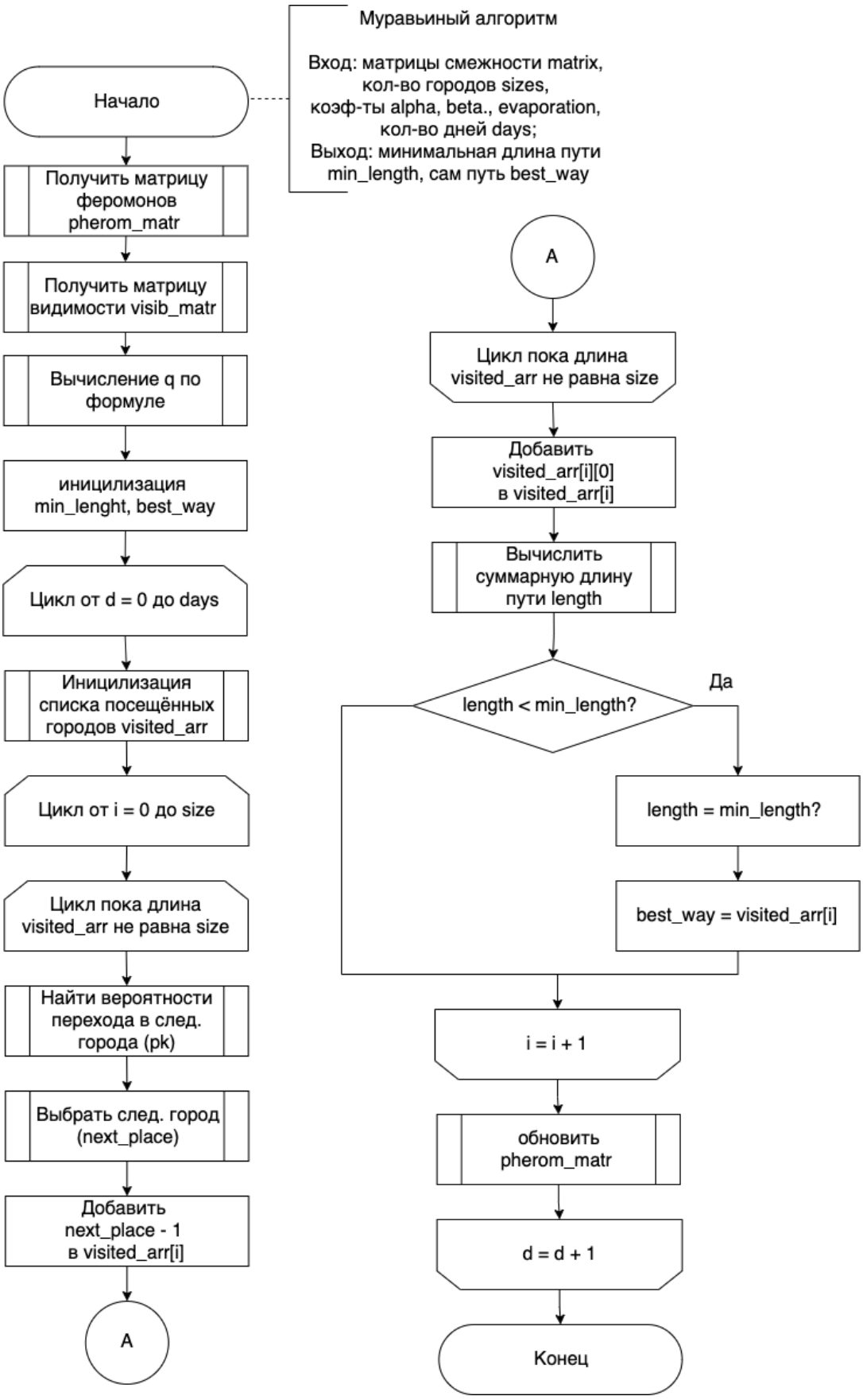

Рисунок 2.1 — Схема муравьиного алгоритма

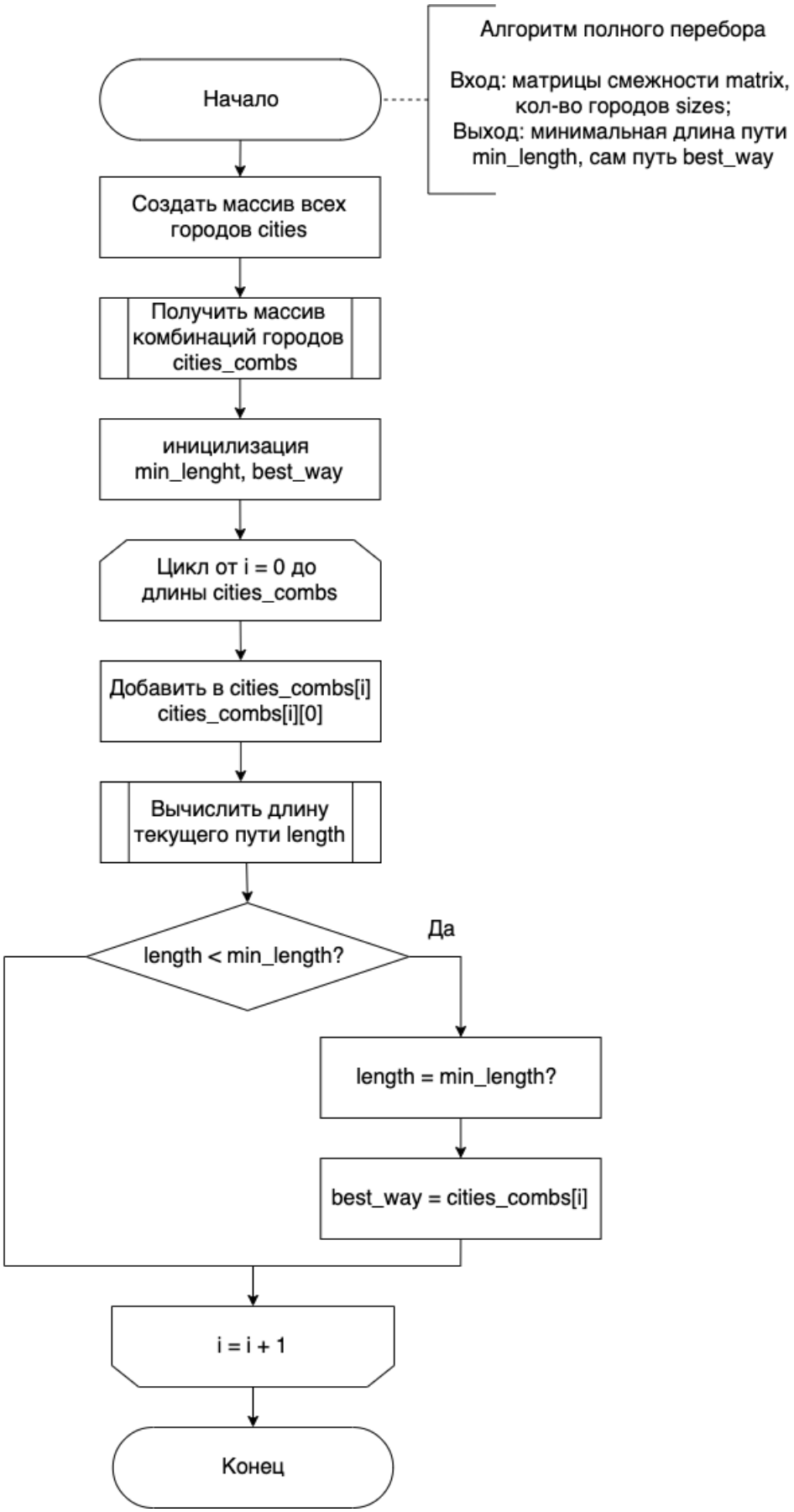

Рисунок 2.2 — Схема алгоритма полного перебора

2.2 Классы эквивалентности

Выделенные классы эквивалентности для тестирования:

— коэффицент α <= 0;

— коэффицент α не является числом;

— коэффицент β <= 0;

— коэффицент β не является числом;

— коэффицент $evaporation$ <= 0;

— коэффицент $evaporation$ не является числом;

— номер команды < 0 или > 5;

— номер команды не является целым числом;

— корректный ввод всех параметров;

2.3 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие структуры данных:

— размер матрицы смежности - целое число типа int;

— матрица смежности - матрица, элементы которой имеют тип int;

— коэффиценты α, β, $evaporation$ - действительные числа типа $float$;

— название файла - строка типа str.

2.4 Структура ПО

ПО будет состоять из следующих модулей:

— $main.py$ – файл, содержащий функцию $main$;

— $generate.py$ – файл, содержащий функцию для генирации матриц;

— $ant\_algorythm.py$ – файл, в котором содержатся функции муравьиного алгоритма для решения задачи коммивояжера;

— $full\_combinations.py$ – файл, в котором содержатся функции алгоритма полного перебора для решения задачи коммивояжера;

— $parametrization.py$ – файл, содержащий функцию параметризации для муравьиного алгоритма;

— $compare\_time.py$ – файл, в котором содержатся функции для замера времени работы алгоритмов и построения графика зависимоти времени выполнения от размера матрицы;

— $read\_print.py$ – файл, в котором содержатся функции ввода вывода данных;

— $color.py$ – файл, который содержит переменные типа $string$ для цветного вывода результата работы программы в консоль.

2.5 Вывод

В данном разделе на основе теоретических данных были построены схемы требуемых алгоритмов решения задачи коммивояжера (муравьиного алгоритма и алгоритма полного перебора), выбраны используемые типы данных, выделены классы эквивалентности для тестирования, а также была описана структура ПО.

3 Технологический раздел

В данном разделе будут приведены средства реализации, листинги кода, а также функциональные тесты.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования $Python$ [4]. Выбор обсуловлен наличием опыта работы с ним. Время работы было замерено с помощью функции *process_time* из библиотеки $time$ [5].

3.2 Листинги кода

В листингах 3.2 - 3.2 представлены функции для конвейерного и ленейного алгоритмов обработки матриц.

Листинг 3.1 — Алгоритм полного перебора

```
1  def full_combinations_alg(matrix, size):
2
3      cities = np.arange(size)
4      cities_combs = []
5
6      for combination in it.permutations(cities):
7          cities_combs.append(list(combination))
8
9      best_way = []
10     min_length = float("inf")
11
12     for i in range(len(cities_combs)):
13         cities_combs[i].append(cities_combs[i][0])
14
15         length = calc_length(matrix, size, cities_combs[i])
16
17         if length < min_length:
18             min_length = length
19             best_way = cities_combs[i]
20
21     return min_length, best_way
```

Листинг 3.2 — Муравьиный алгоритм

```
def ant_alg(matrix, size, alpha, beta, evaporation, days):

    pherom_matr = get_pherom_matr(size)
    visib_matr = get_visib_matr(matrix, size)

    q = calc_q(matrix, size)

    best_way = []
    min_length = float("inf")

    for _ in range(days):
        visited_arr = get_visited_places(np.arange(size), size)

        for i in range(size):
            while (len(visited_arr[i]) != size):
                pk = search_probability(pherom_matr, visib_matr,
                    visited_arr, size, i, alpha, beta)
                next_place = choose_next_place(pk)

                visited_arr[i].append(next_place - 1)

            visited_arr[i].append(visited_arr[i][0])

            length = calc_length(matrix, size, visited_arr[i])

            if length < min_length:
                min_length = length
                best_way = visited_arr[i]

        pherom_matr = update_pherom_matr(matrix, size, visited_arr,
            pherom_matr, q, evaporation)

    return min_length, best_way
```

Листинг 3.3 — Функция для обновления феромонов

```
def update_pherom_matr(matrix, size, visited_arr, pherom_matr, q,
    evaporation):

    ants = size

    for i in range(size):
        for j in range(size):
            delta_pherom_matr = 0

            for ant in range(ants):
                length = calc_length(matrix, size, visited_arr[ant])
                delta_pherom_matr += q / length

            pherom_matr[i][j] *= (1 - evaporation)
            pherom_matr[i][j] += delta_pherom_matr

            if pherom_matr[i][j] < MIN_PHEROMONE:
            pherom_matr[i][j] = MIN_PHEROMONE

    return pherom_matr
```

Листинг 3.4 — Функция для нахождения вероятней перехода в каждый из городов

```
def search_probability(pherom_matr, visib_matr, visited_arr, size, ant,
    alpha, beta):
    pk = [0] * size

    for i in range(size):
        if i not in visited_arr[ant]:
            ant_i = visited_arr[ant][-1]

            pk[i] = pow(pherom_matr[ant_i][i], alpha) * \
                    pow(visib_matr[ant_i][i], beta)
        else:
            pk[i] = 0

    pk_sum = sum(pk)

    for i in range(size):
        pk[i] /= pk_sum

    return pk
```

Листинг 3.5 — Функция выбора следующего города

```
def choose_next_place(pk):

    size = len(pk)
    numb = 0
    i = 0

    probability = random()

    while numb < probability and i < size:
        numb += pk[i]
        i += 1

    return i
```

Листинг 3.6 — Функция нахождения длины пути

```
def calc_length(matrix, size, way):

    length = 0

    for i in range(size):
        beg_city = way[i]
        end_city = way[i + 1]

        length += matrix[beg_city][end_city]

    return length
```

3.3 Функциональные тесты

В таблице 3.1 приведены функциональные тесты для алгоритмов решения задачи коммивояжера (муравьиного алгоритма и алгоритма полного перебора). Все тесты пройдены успешно.

Таблица 3.1 — Функциональные тесты

Матрица смежности	Результат	Ожидаемый результат
$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{pmatrix}$	6, [0, 1, 2, 0]	6, [0, 1, 2, 0]
$\begin{pmatrix} 0 & 3 & 4 & 1 \\ 3 & 0 & 1 & 1 \\ 4 & 1 & 0 & 2 \\ 1 & 1 & 2 & 0 \end{pmatrix}$	7, [0, 1, 2, 3, 0]	7, [0, 1, 2, 3, 0]
$\begin{pmatrix} 0 & 11 & 12 & 14 & 13 \\ 11 & 0 & 15 & 10 & 10 \\ 12 & 15 & 0 & 14 & 13 \\ 14 & 10 & 14 & 0 & 10 \\ 13 & 10 & 13 & 10 & 0 \end{pmatrix}$	56, [0, 1, 3, 4, 2, 0]	56, [0, 1, 3, 4, 2, 0]

3.4 Вывод

В данном разделе были разработаны алгоритмы решения задачи коммивояжера (муравьиный алгоритм и алгоритм полного перебора), проведено тестирование, описаны средства реализации.

4 Экспериментальный раздел

4.1 Технические характеристики

Тестирование выполнялось на устройстве со следующими техническими характеристиками:

— Операционная система Ubuntu 21.04;

— Память 16 GiB (4,5 GiB выделено для нужд графического ядра)

— Процессор AMD® Ryzen 5 5500u with radeon graphics × 12

4.2 Демонстрация работы программы

4.3 Время выполнения алгоритмов

Результаты замеров времени работы алгоритмов решения задачи коммивояжера представлены на рисунках 4.3 - 4.5. Замеры времени проводились в секундах и усреднялись для каждого набора одинаковых экспериментов.

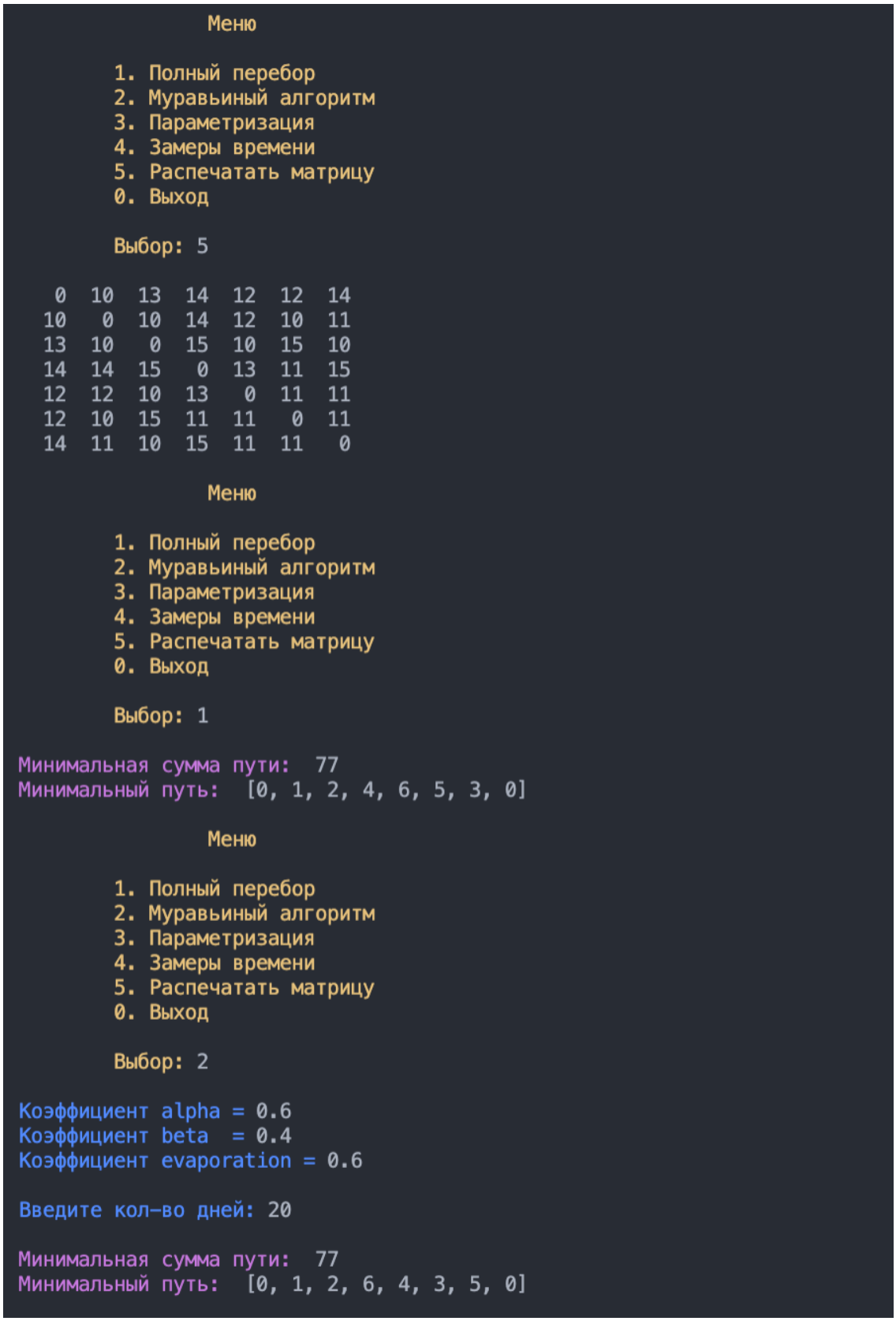

Рисунок 4.1 — Пример работы программы

Размер	Муравьиный алг.	полный перебор
2	1.1800e-04	1.6029e-02
3	5.4000e-05	3.0766e-02
4	9.6000e-05	7.1131e-02
5	4.4000e-04	1.4442e-01
6	1.3202e-02	2.6173e-01
7	2.2243e-02	4.7923e-01
8	2.2290e-01	8.2016e-01
9	2.3747e+00	1.1539e+00
10	2.4016e+01	1.5597e+00

Рисунок 4.2 — Замеры времени работы для муравьиного алгоритиа и алгоритма полного перебора

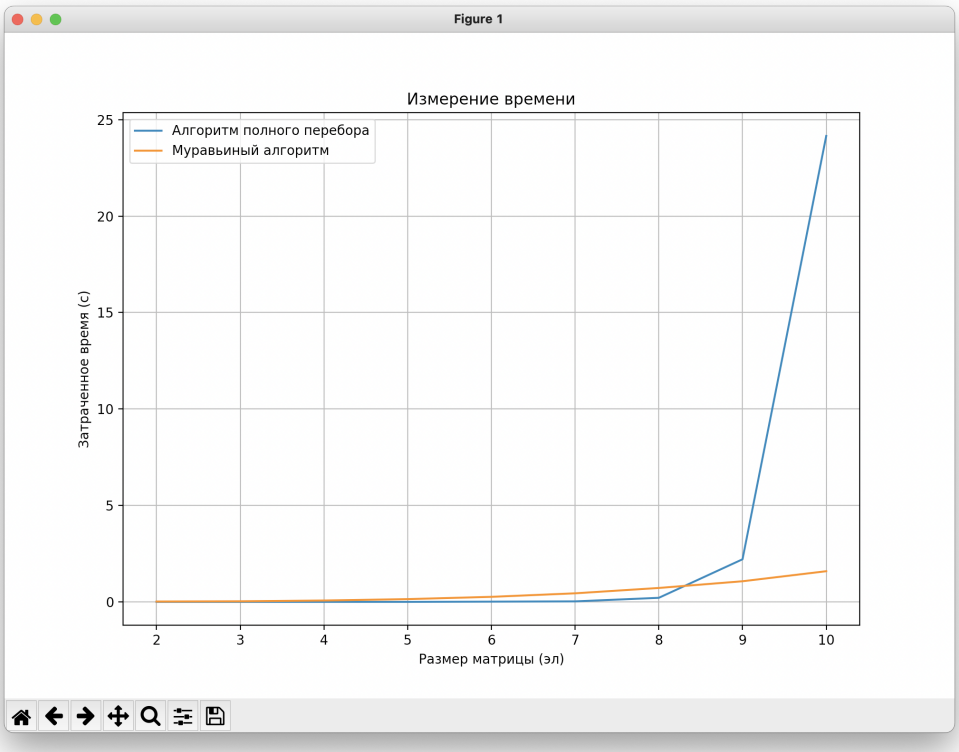

Рисунок 4.3 — Зависимость времени работы алгоритмов от размера матриц

4.4 Автоматическая параметризация

Автоматическая параметризация была проведена на двух классах данных. Для проведение эксперимета были взяты матрицы размером 10x10. Муравьиный алгоритм был запущен для всех значений $\alpha, \rho \in (0, 1)$ с шагом 0.1.

В качестве эталонного значения был взят результат работы алгоритма полного перебора.

Далее будут представлены матрицы смежности (матрица 4.1 для первого класса данных и 4.2 для второго), на которых происходила параметризация и таблицы с результатами её выполнения.

4.4.1 Класс данных 1

В качестве первого класса данных была взята матрица смежности, в которой все значения незначительно отличаются друг от друга, находятся в диапазоне [1, 3].

$$M_1 = \begin{pmatrix} 0 & 2 & 1 & 1 & 3 & 3 & 2 & 2 & 3 & 3 \\ 2 & 0 & 3 & 3 & 3 & 1 & 3 & 2 & 2 & 3 \\ 1 & 3 & 0 & 2 & 3 & 3 & 1 & 1 & 2 & 3 \\ 1 & 3 & 2 & 0 & 2 & 3 & 2 & 3 & 1 & 3 \\ 3 & 3 & 3 & 2 & 0 & 3 & 2 & 3 & 1 & 1 \\ 3 & 1 & 3 & 3 & 3 & 0 & 1 & 1 & 2 & 1 \\ 2 & 3 & 1 & 2 & 2 & 1 & 0 & 2 & 1 & 2 \\ 2 & 2 & 1 & 3 & 3 & 1 & 2 & 0 & 2 & 3 \\ 3 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 0 & 2 \\ 3 & 3 & 3 & 3 & 1 & 1 & 2 & 3 & 2 & 0 \end{pmatrix} \tag{4.1}$$

Таблица 4.1 — Параметры для класса данных 1

α	β	ρ	Дней	Результат	Ошибка
0.1	0.9	0.1	50	12	1
0.1	0.9	0.1	100	12	1
0.1	0.9	0.1	200	12	0
0.1	0.9	0.2	50	12	0
0.1	0.9	0.2	100	12	0
0.1	0.9	0.2	200	12	0
0.1	0.9	0.3	50	12	1
0.1	0.9	0.3	100	12	1
0.1	0.9	0.3	200	12	0
0.1	0.9	0.4	50	12	1
0.1	0.9	0.4	100	12	0
0.1	0.9	0.4	200	12	0
0.1	0.9	0.5	50	12	0
0.1	0.9	0.5	100	12	0
0.1	0.9	0.5	200	12	0
0.1	0.9	0.6	50	12	1
0.1	0.9	0.6	100	12	1
0.1	0.9	0.6	200	12	0
0.1	0.9	0.7	50	12	1
0.1	0.9	0.7	100	12	0
0.1	0.9	0.7	200	12	0
0.1	0.9	0.8	50	12	1
0.1	0.9	0.8	100	12	0
0.1	0.9	0.8	200	12	0
0.2	0.8	0.1	50	12	0
0.2	0.8	0.1	100	12	1
0.2	0.8	0.1	200	12	1
0.2	0.8	0.2	50	12	1
0.2	0.8	0.2	100	12	0
0.2	0.8	0.2	200	12	0
0.2	0.8	0.3	50	12	0
0.2	0.8	0.3	100	12	0
0.2	0.8	0.3	200	12	0
0.2	0.8	0.4	50	12	0
0.2	0.8	0.4	100	12	0
0.2	0.8	0.4	200	12	0

0.2	0.8	0.5	50	12	1
0.2	0.8	0.5	100	12	0
0.2	0.8	0.5	200	12	1
0.2	0.8	0.6	50	12	0
0.2	0.8	0.6	100	12	0
0.2	0.8	0.6	200	12	0
0.2	0.8	0.7	50	12	1
0.2	0.8	0.7	100	12	1
0.2	0.8	0.7	200	12	0
0.2	0.8	0.8	50	12	1
0.2	0.8	0.8	100	12	0
0.2	0.8	0.8	200	12	0
0.3	0.7	0.1	50	12	0
0.3	0.7	0.1	100	12	1
0.3	0.7	0.1	200	12	1
0.3	0.7	0.2	50	12	0
0.3	0.7	0.2	100	12	0
0.3	0.7	0.2	200	12	0
0.3	0.7	0.3	50	12	1
0.3	0.7	0.3	100	12	1
0.3	0.7	0.3	200	12	0
0.3	0.7	0.4	50	12	1
0.3	0.7	0.4	100	12	0
0.3	0.7	0.4	200	12	1
0.3	0.7	0.5	50	12	0
0.3	0.7	0.5	100	12	1
0.3	0.7	0.5	200	12	0
0.3	0.7	0.6	50	12	2
0.3	0.7	0.6	100	12	1
0.3	0.7	0.6	200	12	1
0.3	0.7	0.7	50	12	1
0.3	0.7	0.7	100	12	1
0.3	0.7	0.7	200	12	0
0.3	0.7	0.8	50	12	1
0.3	0.7	0.8	100	12	1
0.3	0.7	0.8	200	12	1
0.4	0.6	0.1	50	12	1
0.4	0.6	0.1	100	12	0

0.4	0.6	0.1	200	12	0
0.4	0.6	0.2	50	12	1
0.4	0.6	0.2	100	12	0
0.4	0.6	0.2	200	12	1
0.4	0.6	0.3	50	12	0
0.4	0.6	0.3	100	12	0
0.4	0.6	0.3	200	12	1
0.4	0.6	0.4	50	12	1
0.4	0.6	0.4	100	12	0
0.4	0.6	0.4	200	12	0
0.4	0.6	0.5	50	12	1
0.4	0.6	0.5	100	12	1
0.4	0.6	0.5	200	12	0
0.4	0.6	0.6	50	12	0
0.4	0.6	0.6	100	12	1
0.4	0.6	0.6	200	12	0
0.4	0.6	0.7	50	12	2
0.4	0.6	0.7	100	12	1
0.4	0.6	0.7	200	12	0
0.4	0.6	0.8	50	12	1
0.4	0.6	0.8	100	12	1
0.4	0.6	0.8	200	12	0
0.5	0.5	0.1	50	12	2
0.5	0.5	0.1	100	12	0
0.5	0.5	0.1	200	12	1
0.5	0.5	0.2	50	12	1
0.5	0.5	0.2	100	12	0
0.5	0.5	0.2	200	12	1
0.5	0.5	0.3	50	12	2
0.5	0.5	0.3	100	12	1
0.5	0.5	0.3	200	12	1
0.5	0.5	0.4	50	12	2
0.5	0.5	0.4	100	12	1
0.5	0.5	0.4	200	12	0
0.5	0.5	0.5	50	12	2
0.5	0.5	0.5	100	12	1
0.5	0.5	0.5	200	12	1
0.5	0.5	0.6	50	12	2

0.5	0.5	0.6	100	12	1
0.5	0.5	0.6	200	12	1
0.5	0.5	0.7	50	12	2
0.5	0.5	0.7	100	12	0
0.5	0.5	0.7	200	12	0
0.5	0.5	0.8	50	12	1
0.5	0.5	0.8	100	12	1
0.5	0.5	0.8	200	12	1
0.6	0.4	0.1	50	12	1
0.6	0.4	0.1	100	12	0
0.6	0.4	0.1	200	12	1
0.6	0.4	0.2	50	12	2
0.6	0.4	0.2	100	12	1
0.6	0.4	0.2	200	12	1
0.6	0.4	0.3	50	12	1
0.6	0.4	0.3	100	12	2
0.6	0.4	0.3	200	12	1
0.6	0.4	0.4	50	12	1
0.6	0.4	0.4	100	12	0
0.6	0.4	0.4	200	12	0
0.6	0.4	0.5	50	12	1
0.6	0.4	0.5	100	12	1
0.6	0.4	0.5	200	12	1
0.6	0.4	0.6	50	12	1
0.6	0.4	0.6	100	12	1
0.6	0.4	0.6	200	12	1
0.6	0.4	0.7	50	12	1
0.6	0.4	0.7	100	12	1
0.6	0.4	0.7	200	12	1
0.6	0.4	0.8	50	12	1
0.6	0.4	0.8	100	12	2
0.6	0.4	0.8	200	12	1
0.7	0.3	0.1	50	12	2
0.7	0.3	0.1	100	12	1
0.7	0.3	0.1	200	12	1
0.7	0.3	0.2	50	12	1
0.7	0.3	0.2	100	12	1
0.7	0.3	0.2	200	12	1

0.7	0.3	0.3	50	12	2
0.7	0.3	0.3	100	12	1
0.7	0.3	0.3	200	12	1
0.7	0.3	0.4	50	12	2
0.7	0.3	0.4	100	12	1
0.7	0.3	0.4	200	12	0
0.7	0.3	0.5	50	12	2
0.7	0.3	0.5	100	12	1
0.7	0.3	0.5	200	12	1
0.7	0.3	0.6	50	12	3
0.7	0.3	0.6	100	12	2
0.7	0.3	0.6	200	12	1
0.7	0.3	0.7	50	12	2
0.7	0.3	0.7	100	12	2
0.7	0.3	0.7	200	12	1
0.7	0.3	0.8	50	12	2
0.7	0.3	0.8	100	12	1
0.7	0.3	0.8	200	12	0
0.8	0.2	0.1	50	12	1
0.8	0.2	0.1	100	12	2
0.8	0.2	0.1	200	12	1
0.8	0.2	0.2	50	12	2
0.8	0.2	0.2	100	12	2
0.8	0.2	0.2	200	12	1
0.8	0.2	0.3	50	12	1
0.8	0.2	0.3	100	12	2
0.8	0.2	0.3	200	12	1
0.8	0.2	0.4	50	12	1
0.8	0.2	0.4	100	12	2
0.8	0.2	0.4	200	12	1
0.8	0.2	0.5	50	12	2
0.8	0.2	0.5	100	12	1
0.8	0.2	0.5	200	12	0
0.8	0.2	0.6	50	12	2
0.8	0.2	0.6	100	12	1
0.8	0.2	0.6	200	12	0
0.8	0.2	0.7	50	12	2
0.8	0.2	0.7	100	12	2

0.8	0.2	0.7	200	12	2
0.8	0.2	0.8	50	12	0
0.8	0.2	0.8	100	12	1
0.8	0.2	0.8	200	12	1
0.9	0.1	0.1	50	12	2
0.9	0.1	0.1	100	12	1
0.9	0.1	0.1	200	12	2
0.9	0.1	0.2	50	12	2
0.9	0.1	0.2	100	12	1
0.9	0.1	0.2	200	12	0
0.9	0.1	0.3	50	12	3
0.9	0.1	0.3	100	12	1
0.9	0.1	0.3	200	12	1
0.9	0.1	0.4	50	12	3
0.9	0.1	0.4	100	12	1
0.9	0.1	0.4	200	12	2
0.9	0.1	0.5	50	12	2
0.9	0.1	0.5	100	12	2
0.9	0.1	0.5	200	12	2
0.9	0.1	0.6	50	12	2
0.9	0.1	0.6	100	12	2
0.9	0.1	0.6	200	12	1
0.9	0.1	0.7	50	12	2
0.9	0.1	0.7	100	12	1
0.9	0.1	0.7	200	12	1
0.9	0.1	0.8	50	12	2
0.9	0.1	0.8	100	12	2
0.9	0.1	0.8	200	12	1

4.4.2 Класс данных 2

В качестве второго класса данных была взята матрица смежности, в которой все значения отличюатся на большое значение друг от друга, находятся в диапазоне [1, 1000].

$$M_2 = \begin{pmatrix} 0 & 157 & 611 & 117 & 341 & 452 & 579 & 773 & 370 & 343 \\ 157 & 0 & 170 & 696 & 238 & 669 & 302 & 633 & 111 & 427 \\ 611 & 170 & 0 & 95 & 829 & 14 & 661 & 118 & 871 & 754 \\ 117 & 696 & 95 & 0 & 37 & 535 & 308 & 996 & 419 & 456 \\ 341 & 238 & 829 & 37 & 0 & 854 & 454 & 908 & 806 & 455 \\ 452 & 669 & 14 & 535 & 854 & 0 & 646 & 262 & 400 & 799 \\ 579 & 302 & 661 & 308 & 454 & 646 & 0 & 139 & 982 & 423 \\ 773 & 633 & 118 & 996 & 908 & 262 & 139 & 0 & 887 & 59 \\ 370 & 111 & 871 & 419 & 806 & 400 & 982 & 887 & 0 & 578 \\ 343 & 427 & 754 & 456 & 455 & 799 & 423 & 59 & 578 & 0 \end{pmatrix} \tag{4.2}$$

Таблица 4.2 — Параметры для класса данных 2

α	β	ρ	Дней	Результат	Ошибка
0.1	0.9	0.1	50	1809	0
0.1	0.9	0.1	100	1809	0
0.1	0.9	0.1	200	1809	0
0.1	0.9	0.2	50	1809	0
0.1	0.9	0.2	100	1809	32
0.1	0.9	0.2	200	1809	32
0.1	0.9	0.3	50	1809	0
0.1	0.9	0.3	100	1809	0
0.1	0.9	0.3	200	1809	81
0.1	0.9	0.4	50	1809	32
0.1	0.9	0.4	100	1809	0
0.1	0.9	0.4	200	1809	0
0.1	0.9	0.5	50	1809	32
0.1	0.9	0.5	100	1809	0
0.1	0.9	0.5	200	1809	0
0.1	0.9	0.6	50	1809	0
0.1	0.9	0.6	100	1809	0
0.1	0.9	0.6	200	1809	0
0.1	0.9	0.7	50	1809	205
0.1	0.9	0.7	100	1809	32
0.1	0.9	0.7	200	1809	0

0.1	0.9	0.8	50	1809	0
0.1	0.9	0.8	100	1809	0
0.1	0.9	0.8	200	1809	0
0.2	0.8	0.1	50	1809	0
0.2	0.8	0.1	100	1809	32
0.2	0.8	0.1	200	1809	0
0.2	0.8	0.2	50	1809	146
0.2	0.8	0.2	100	1809	0
0.2	0.8	0.2	200	1809	0
0.2	0.8	0.3	50	1809	32
0.2	0.8	0.3	100	1809	0
0.2	0.8	0.3	200	1809	0
0.2	0.8	0.4	50	1809	81
0.2	0.8	0.4	100	1809	32
0.2	0.8	0.4	200	1809	0
0.2	0.8	0.5	50	1809	0
0.2	0.8	0.5	100	1809	0
0.2	0.8	0.5	200	1809	0
0.2	0.8	0.6	50	1809	32
0.2	0.8	0.6	100	1809	32
0.2	0.8	0.6	200	1809	0
0.2	0.8	0.7	50	1809	32
0.2	0.8	0.7	100	1809	0
0.2	0.8	0.7	200	1809	0
0.2	0.8	0.8	50	1809	162
0.2	0.8	0.8	100	1809	0
0.2	0.8	0.8	200	1809	0
0.3	0.7	0.1	50	1809	81
0.3	0.7	0.1	100	1809	0
0.3	0.7	0.1	200	1809	32
0.3	0.7	0.2	50	1809	131
0.3	0.7	0.2	100	1809	0
0.3	0.7	0.2	200	1809	0
0.3	0.7	0.3	50	1809	32
0.3	0.7	0.3	100	1809	0
0.3	0.7	0.3	200	1809	0
0.3	0.7	0.4	50	1809	0
0.3	0.7	0.4	100	1809	0

0.3	0.7	0.4	200	1809	0
0.3	0.7	0.5	50	1809	208
0.3	0.7	0.5	100	1809	32
0.3	0.7	0.5	200	1809	0
0.3	0.7	0.6	50	1809	131
0.3	0.7	0.6	100	1809	32
0.3	0.7	0.6	200	1809	0
0.3	0.7	0.7	50	1809	32
0.3	0.7	0.7	100	1809	0
0.3	0.7	0.7	200	1809	0
0.3	0.7	0.8	50	1809	131
0.3	0.7	0.8	100	1809	0
0.3	0.7	0.8	200	1809	32
0.4	0.6	0.1	50	1809	81
0.4	0.6	0.1	100	1809	0
0.4	0.6	0.1	200	1809	0
0.4	0.6	0.2	50	1809	81
0.4	0.6	0.2	100	1809	159
0.4	0.6	0.2	200	1809	0
0.4	0.6	0.3	50	1809	0
0.4	0.6	0.3	100	1809	161
0.4	0.6	0.3	200	1809	0
0.4	0.6	0.4	50	1809	32
0.4	0.6	0.4	100	1809	270
0.4	0.6	0.4	200	1809	0
0.4	0.6	0.5	50	1809	188
0.4	0.6	0.5	100	1809	32
0.4	0.6	0.5	200	1809	0
0.4	0.6	0.6	50	1809	81
0.4	0.6	0.6	100	1809	81
0.4	0.6	0.6	200	1809	32
0.4	0.6	0.7	50	1809	32
0.4	0.6	0.7	100	1809	188
0.4	0.6	0.7	200	1809	32
0.4	0.6	0.8	50	1809	378
0.4	0.6	0.8	100	1809	161
0.4	0.6	0.8	200	1809	32
0.5	0.5	0.1	50	1809	146

0.5	0.5	0.1	100	1809	0
0.5	0.5	0.1	200	1809	32
0.5	0.5	0.2	50	1809	273
0.5	0.5	0.2	100	1809	0
0.5	0.5	0.2	200	1809	0
0.5	0.5	0.3	50	1809	131
0.5	0.5	0.3	100	1809	32
0.5	0.5	0.3	200	1809	0
0.5	0.5	0.4	50	1809	205
0.5	0.5	0.4	100	1809	0
0.5	0.5	0.4	200	1809	32
0.5	0.5	0.5	50	1809	223
0.5	0.5	0.5	100	1809	159
0.5	0.5	0.5	200	1809	0
0.5	0.5	0.6	50	1809	224
0.5	0.5	0.6	100	1809	81
0.5	0.5	0.6	200	1809	0
0.5	0.5	0.7	50	1809	289
0.5	0.5	0.7	100	1809	0
0.5	0.5	0.7	200	1809	0
0.5	0.5	0.8	50	1809	205
0.5	0.5	0.8	100	1809	32
0.5	0.5	0.8	200	1809	81
0.6	0.4	0.1	50	1809	219
0.6	0.4	0.1	100	1809	131
0.6	0.4	0.1	200	1809	0
0.6	0.4	0.2	50	1809	162
0.6	0.4	0.2	100	1809	237
0.6	0.4	0.2	200	1809	146
0.6	0.4	0.3	50	1809	81
0.6	0.4	0.3	100	1809	161
0.6	0.4	0.3	200	1809	0
0.6	0.4	0.4	50	1809	269
0.6	0.4	0.4	100	1809	81
0.6	0.4	0.4	200	1809	131
0.6	0.4	0.5	50	1809	349
0.6	0.4	0.5	100	1809	219
0.6	0.4	0.5	200	1809	0

0.6	0.4	0.6	50	1809	32
0.6	0.4	0.6	100	1809	0
0.6	0.4	0.6	200	1809	81
0.6	0.4	0.7	50	1809	0
0.6	0.4	0.7	100	1809	0
0.6	0.4	0.7	200	1809	32
0.6	0.4	0.8	50	1809	224
0.6	0.4	0.8	100	1809	224
0.6	0.4	0.8	200	1809	131
0.7	0.3	0.1	50	1809	384
0.7	0.3	0.1	100	1809	146
0.7	0.3	0.1	200	1809	287
0.7	0.3	0.2	50	1809	0
0.7	0.3	0.2	100	1809	287
0.7	0.3	0.2	200	1809	0
0.7	0.3	0.3	50	1809	341
0.7	0.3	0.3	100	1809	237
0.7	0.3	0.3	200	1809	341
0.7	0.3	0.4	50	1809	239
0.7	0.3	0.4	100	1809	389
0.7	0.3	0.4	200	1809	81
0.7	0.3	0.5	50	1809	310
0.7	0.3	0.5	100	1809	353
0.7	0.3	0.5	200	1809	0
0.7	0.3	0.6	50	1809	313
0.7	0.3	0.6	100	1809	269
0.7	0.3	0.6	200	1809	0
0.7	0.3	0.7	50	1809	555
0.7	0.3	0.7	100	1809	225
0.7	0.3	0.7	200	1809	0
0.7	0.3	0.8	50	1809	470
0.7	0.3	0.8	100	1809	316
0.7	0.3	0.8	200	1809	243
0.8	0.2	0.1	50	1809	81
0.8	0.2	0.1	100	1809	569
0.8	0.2	0.1	200	1809	389
0.8	0.2	0.2	50	1809	162
0.8	0.2	0.2	100	1809	225

0.8	0.2	0.2	200	1809	348
0.8	0.2	0.3	50	1809	411
0.8	0.2	0.3	100	1809	450
0.8	0.2	0.3	200	1809	287
0.8	0.2	0.4	50	1809	539
0.8	0.2	0.4	100	1809	32
0.8	0.2	0.4	200	1809	237
0.8	0.2	0.5	50	1809	489
0.8	0.2	0.5	100	1809	32
0.8	0.2	0.5	200	1809	161
0.8	0.2	0.6	50	1809	208
0.8	0.2	0.6	100	1809	254
0.8	0.2	0.6	200	1809	146
0.8	0.2	0.7	50	1809	559
0.8	0.2	0.7	100	1809	161
0.8	0.2	0.7	200	1809	237
0.8	0.2	0.8	50	1809	161
0.8	0.2	0.8	100	1809	487
0.8	0.2	0.8	200	1809	188
0.9	0.1	0.1	50	1809	223
0.9	0.1	0.1	100	1809	146
0.9	0.1	0.1	200	1809	254
0.9	0.1	0.2	50	1809	433
0.9	0.1	0.2	100	1809	529
0.9	0.1	0.2	200	1809	413
0.9	0.1	0.3	50	1809	423
0.9	0.1	0.3	100	1809	458
0.9	0.1	0.3	200	1809	161
0.9	0.1	0.4	50	1809	482
0.9	0.1	0.4	100	1809	420
0.9	0.1	0.4	200	1809	387
0.9	0.1	0.5	50	1809	672
0.9	0.1	0.5	100	1809	319
0.9	0.1	0.5	200	1809	81
0.9	0.1	0.6	50	1809	322
0.9	0.1	0.6	100	1809	612
0.9	0.1	0.6	200	1809	146
0.9	0.1	0.7	50	1809	349

0.9	0.1	0.7	100	1809	467
0.9	0.1	0.7	200	1809	310
0.9	0.1	0.8	50	1809	237
0.9	0.1	0.8	100	1809	703
0.9	0.1	0.8	200	1809	387

4.5 Вывод

В этом разделе были указаны технические характеристики машины, на которой происходило сравнение времени работы алгоритмов (муравьиного алгоритма и алгоритма полного перебора) решения задачи коммивояжера, также была рассмотрена автоматическая параметризация.

В результате замеров времени было установлено, что муравьиный алгоритм работает хуже алгоритма полного перебора на матрицах, размер которых меньше 9 (при размере 8 он работает хуже в 3.7 раза). Но при больших размерах муравьиный алгоритм существенно превосходит алгоритм полного перебора (на матрицах 9x9 он лучше в 2.1 раз, а на матрицах 10x10 уже в 15.4 раза). Во время проведения замеров времени кол-во дней для муравьиного алгоритма было взято 300.

На основе проведённой параметризации по двум классам данных можно сделать следующие выводы.

Для первого класса данных 4.1 лучше всего подходят следующие параметры:

— $\alpha = 0.1, \beta = 0.9, \rho = 0.2, 0.5$;

— $\alpha = 0.2, \beta = 0.8, \rho = 0.3, 0.4, 06$;

— $\alpha = 0.3, \beta = 0.7, \rho = 0.2$;

Для второго класса данных 4.2 лучше всего подходят следующие коэффиценты:

— $\alpha = 0.1, \beta = 0.9, \rho = 0.1, 0.6, 0.8$;

— $\alpha = 0.2, \beta = 0.8, \rho = 0.5$;

— $\alpha = 0.3, \beta = 0.7, \rho = 0.4$;

Таким образом, можно сделать вывод о том, что для лучшей работы муравьиного алгоритма на используемых классах данных (4.1 и 4.2) нужно использовать полученные коэффиценты.

ЗАКЛЮЧЕНИЕ

Было экспериментально подтверждено различие во временной эффективности муравьиного алгоритма и алгоритма полного перебора решения задачи коммивояжера. В результате исследований можно сделать вывод о том, что при матрицах большого размера (больше 9) стоит использовать муравьиный алгоритм решения задачи коммивояжера, а не алгоритм полного перебора (на матрице размером 10x10 он работает в 15.4 раза быстрее). Также было установлено по результатам параметризации на экспериментальных класса данных, что при коэффициенте α = 0.1, 0.2, 0.3 муравьиный алгоритм работает наилучшим образом.

В ходе выполнения данной лабораторной работы были решены следующие задачи:

— изучены алгоритмы решения задачи коммивояжера (муравьиный алгоритм и алгоритм полного перебора);

— применены изученные основы для реализации описанных алгоритмов;

— произведен сравнительный анализ муравьиного алгоритма и алгорита полного перебора для решения задачи коммивояжера;

— экспериментально подтверждено различие во временной эффективности рассматриваемых алгоритмов при помощи разработанного программного обеспечения на материале замеров процессорного времени;

— проведена параметризация муравьиного алгоритма, которая показала лучшие наборы параметров для работы алгоритма на экспериментальных классах данных;

— описаны и обоснованы полученные результаты в отчете о выполненной лабораторной работе.

Поставленная цель была достигнута.

СПИСОК ЛИТЕРАТУРЫ

1. Задача коммивояжёра [Электронный ресурс]. Режим доступа: http://galyautdinov.ru/post/zadacha-kommivoyazhera (дата обращения: 05.12.2021).

2. Решаем задачу коммивояжёра простым перебором [Электронный ресурс]. Режим доступа: https://thecode.media/path-js/ (дата обращения: 05.12.2021).

3. Муравьиные алгоритмы [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/105302/ (дата обращения: 05.12.2021).

4. Welcome to Python [Электронный ресурс]. Режим доступа: https://www.python.org (дата обращения: 18.10.2021).

5. time — Time access and conversions [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/time.html#functions (дата обращения: 18.10.2021).