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

по курсу «Анализ алгоритмов»

на тему: «Методы решения задачи коммивояжера»

Вариант № 21

Студент ИУ7-51Б (Группа) _____ (Подпись, дата) Д. В. Шубенина (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Л. Л. Волкова (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной лабораторной работы является параметризация метода решения задачи коммивояжера на основе муравьиного метода.

Для достижения поставленной цели, необходимо решить следующие задачи:

1) описать задачу коммивояжера;
2) описать методы решения задачи коммивояжера: метод полного перебора и метод на основе муравьиного алгоритма;
3) реализовать данные алгоритмы;
4) сравнить по времени метод полного перебора и метод на основе муравьиного алгоритма

Выданный индивидуальный вариант для выполнения лабораторной работы:

— ориентированный граф;

— с элитными муравьями;

— незамкнутый маршрут;

— города России XVI века;

— зимой можно ходить по рекам в обе стороны за равную цену, летом по течению в 2 раза быстрее, против — в 4 раза медленнее.

1 Аналитическая часть

Задача коммивояжера — задача комбинаторной оптимизации, цель которой — нахождение самого выгодного маршрута, проходящего через указанные точки по одному разу с возвращением в исходную точку [1].

Задача рассматривается как в симметричном, так и в ассиметричном варианте. В асимметричном варианте задача представляется в виде ориентированного графа, где веса рёбер между одними и теми же вершинами зависят от направления движения [1].

1.1 Методы решения задачи коммивояжера

Полный перебор. Этот метод заключается в переборе всех возможных маршрутов в графе и выборе кратчайшего из них. Сложность такого алгоритма — $O(n!)$ [2], где n — количество городов.

Муравьиный алгоритм. Данный метод основан на принципах поведения колонии муравьев [2].

Муравьи, двигаясь из муравейника в поисках пищи, откладывают феромоны на своем пути. При этом, чем больше плотность феромона, тем короче путь, и, соответственно, чем длиннее путь, тем быстрее феромон испарится, и его плотность будет меньше [3].

Со временем муравьи оставят наибольшее количество феромонов на самом коротком участке пути, что приведет к тому, что большинство муравьев выберет этот самый короткий путь, и, следовательно, они оставят еще больше феромонов на нем, что уменьшит вероятность выбора других маршрутов [3].

При сведении алгоритма к математическим формулам, сначала определяется целевая функция:

$$n = \frac{1}{D}, \tag{1.1}$$

где D — расстояние до заданного пункта [3].

Далее вычисляются вероятности перехода в заданную точку:

$$P = \frac{t^e \cdot n^b}{\sum_{i=1}^{m} t_i^a \cdot n_i^b}, \tag{1.2}$$

где a, b — настраиваемые параметры,
t — концентрация феромона [3].

При $a = 0$ выбирается ближайший город и алгоритм становится «жадным» (выбирает только оптимальные или самые короткие расстояния), при $b = 0$ будут учитываться только след феромона, что может привести к сужению пространства поиска оптимального решения [3].

Последним производится обновление феромона:

$$t_{i+1} = (1 - p) \cdot t_i + \frac{Q}{L_0}, \tag{1.3}$$

где p настраивает скорость испарения феромона,
Q настраивает концентрацию нанесения феромона,
L_0 — длина пути на определенном участке [3].

Последовательность вышеизложенных действий повторяется, пока не будет найден оптимальный маршрут.

Одной из модификаций муравьиного алгоритма является элитарная муравьиная система. При таком подходе искусственно вводятся «элитные» муравьи, усиливающие уровень феромонов, оптимального на данный момент маршрута [4].

Вывод

В данном разделе была рассмотрена задача коммивояжера, а также способы её решения: полным перебором и муравьиным алгоритмом.

2 Конструкторская часть

В данном разделе будут описаны требования к программному обеспечению и представлены схемы алгоритма полного перебора и муравьиного алгоритма.

2.1 Требования к программному обеспечению

К программе предъявлен ряд требований:

1) наличие интерфейса для выбора действий;
2) возможность сохранения сгенерированной матрицы смежности в файл и загрузки готовых матриц из файла;
3) возможность выбора алгоритма решения задачи коммивояжера.

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие типы данных:

— размер матрицы смежности — целое число;

— имя файла — строка;

— коэффициенты α, β, $k\_evaporation$ — вещественные числа;

— матрица смежности — матрица вещественных чисел.

2.3 Разработка алгоритмов

На рисунке 2.1 представлена схема алгоритма полного перебора, а на рисунке 2.2 — схема муравьиного алгоритма поиска путей.

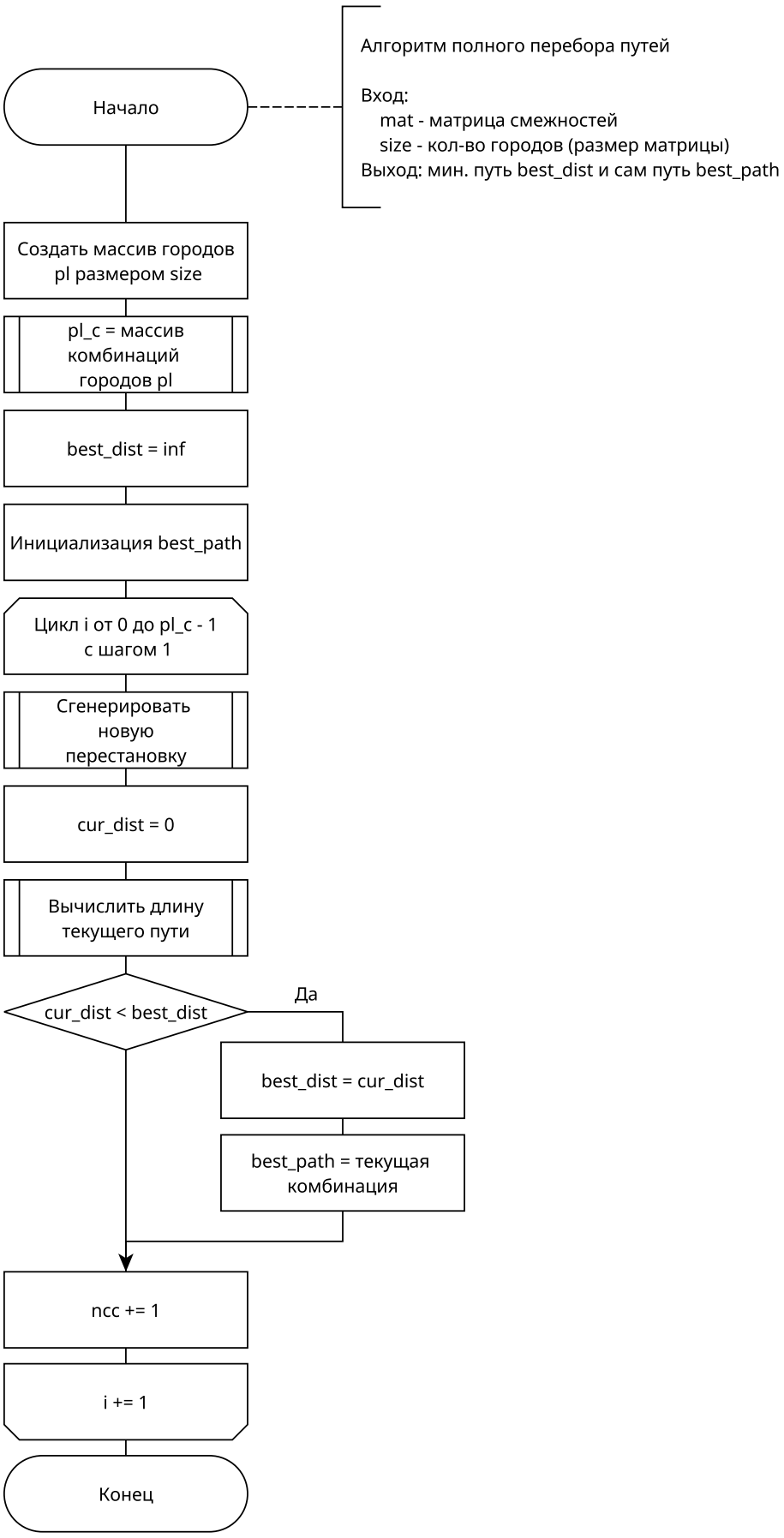

Рисунок 2.1 – Схема алгоритма полного перебора

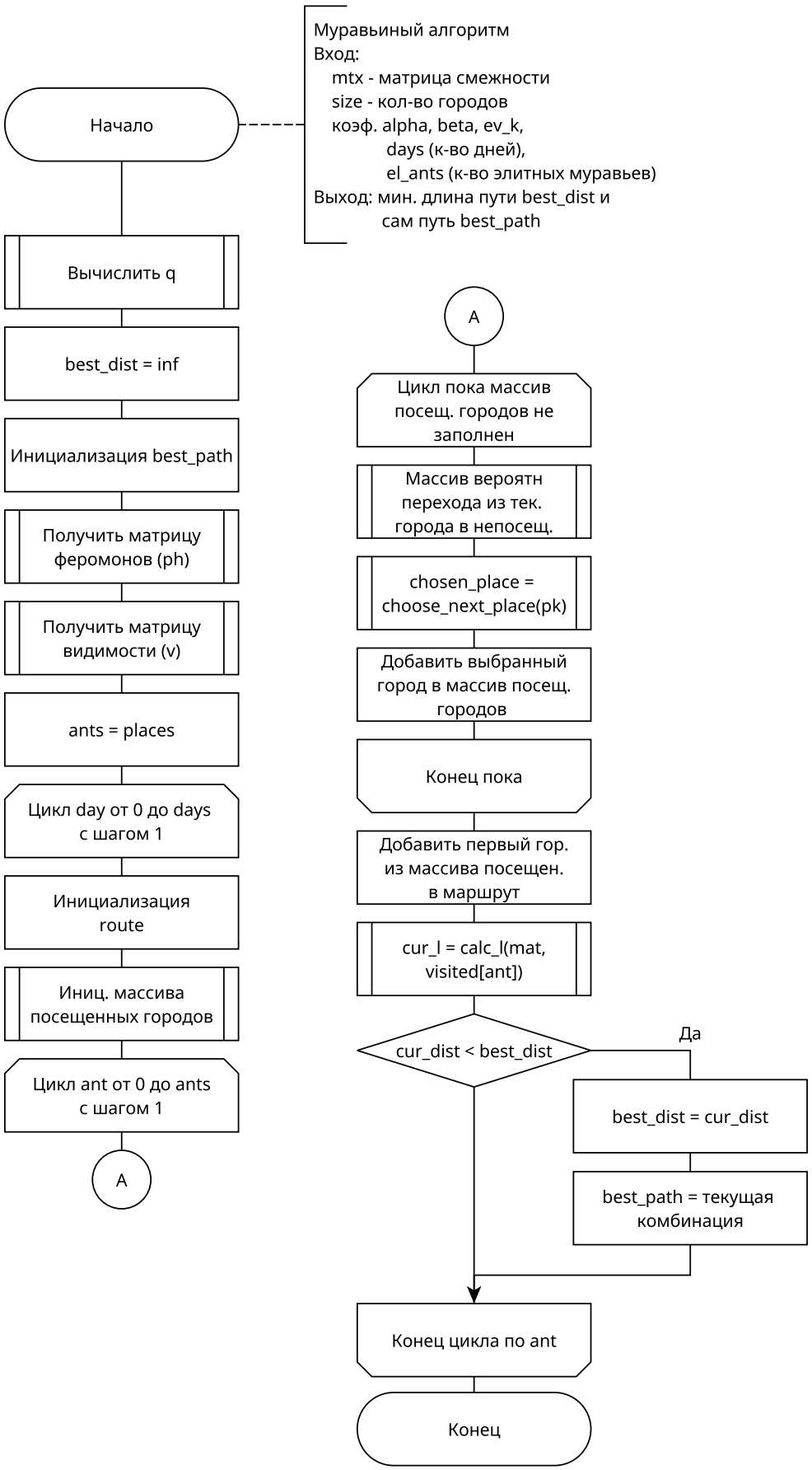

Рисунок 2.2 – Схема муравьиного алгоритма поиска путей

На рисунке 2.3 представлена схема алгоритма выбора следующего города, на рисунке 2.4 — схема алгоритма нахождения массива вероятностей переходов и на рисунке 2.5 показана схема алгоритма обновления матрицы феромонов.

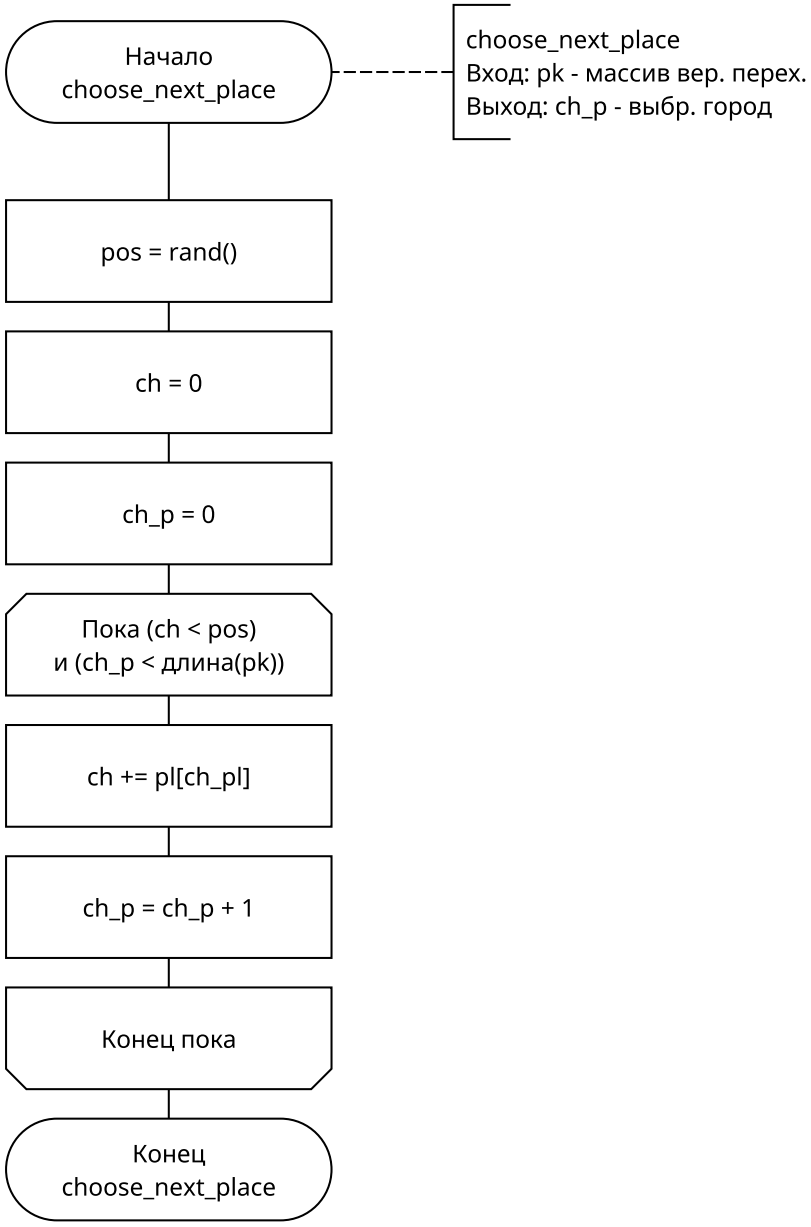

Рисунок 2.3 – Схема алгоритма выбора следующего города

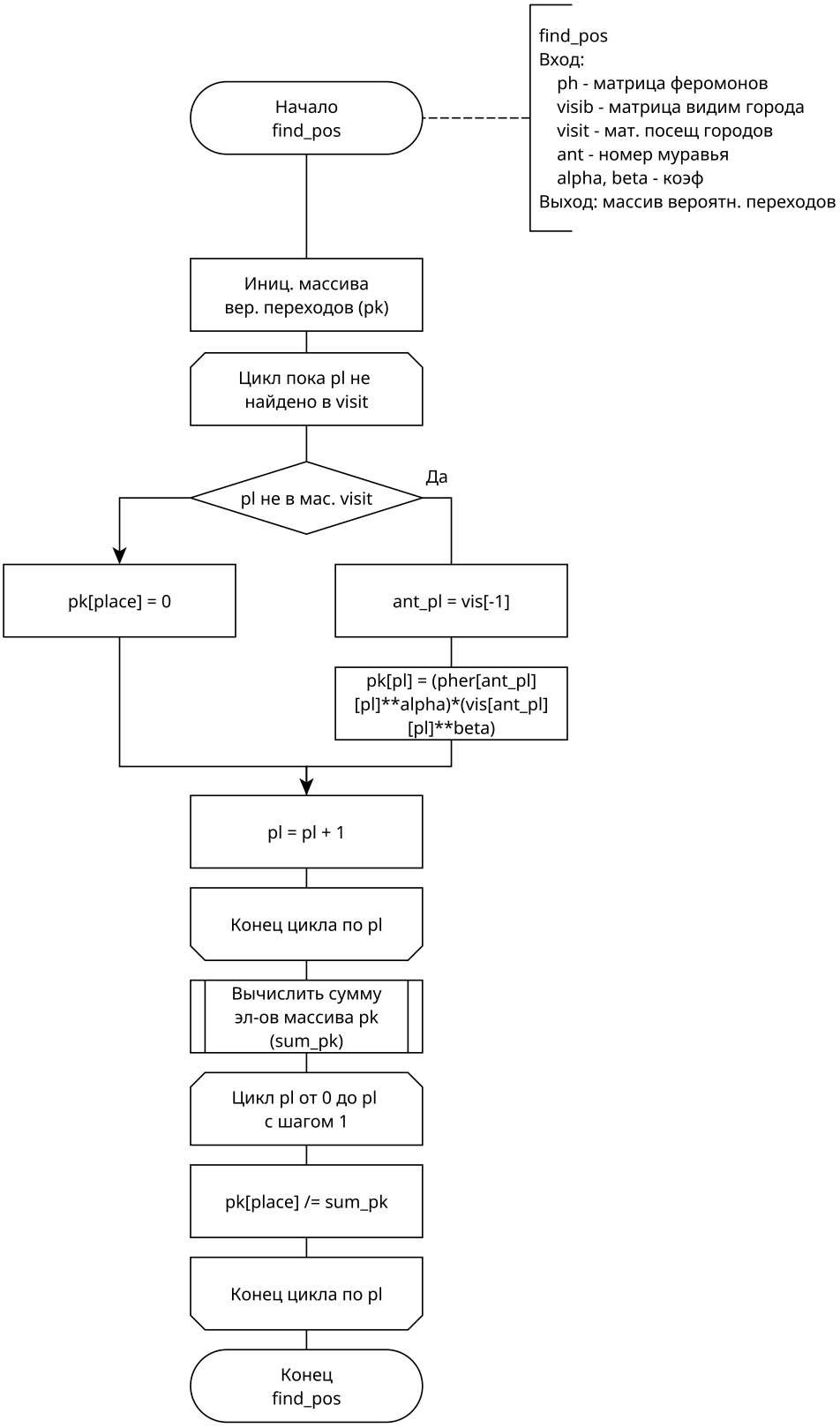

Рисунок 2.4 – Схема алгоритма нахождения массива вероятностей переходов

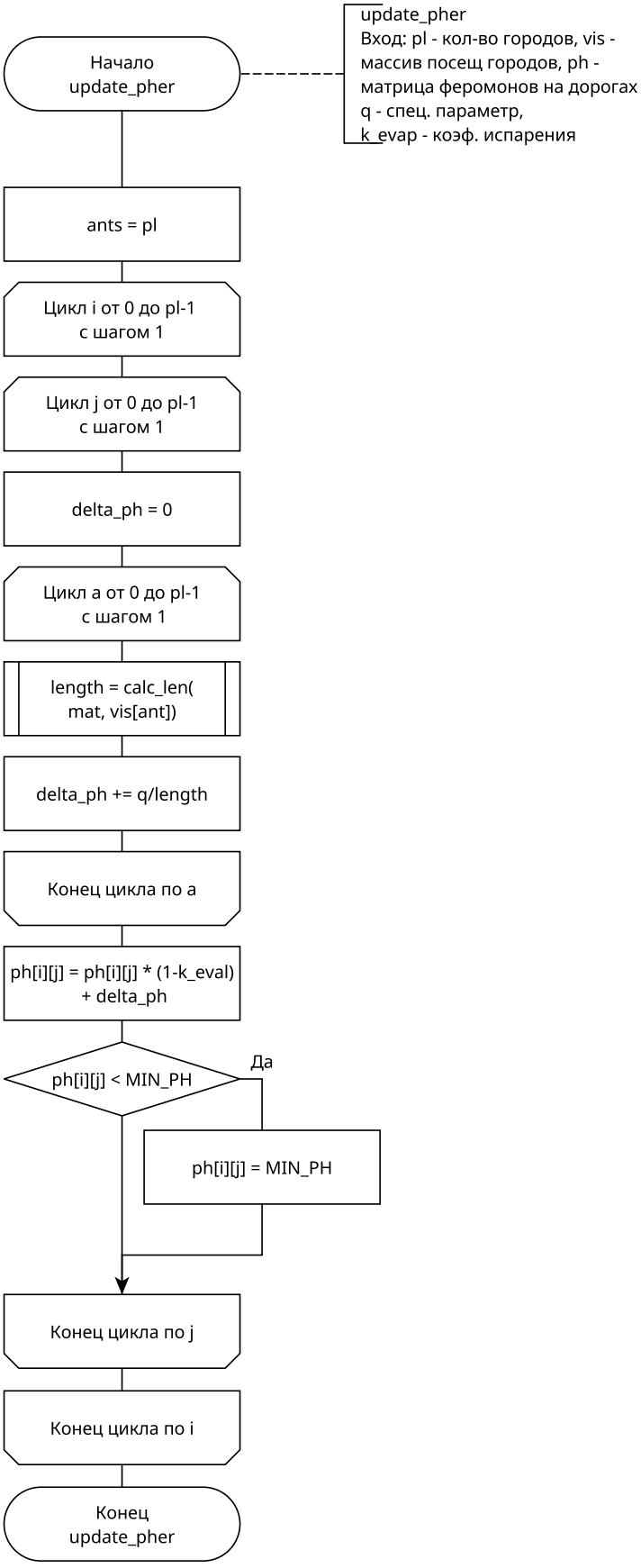

Рисунок 2.5 – Схема алгоритма обновления матрицы феромонов

Вывод

В данном разделе были перечислены требования к программному обеспечению и построены схемы алгоритмов решения задачи коммивояжера.

3 Технологическая часть

3.1 Средства реализации

В данной работе для реализации был выбран язык *Python* [5], поскольку стандартная библиотека данного языка содержит все инструменты, требующиеся для реализации программного обеспечения.

3.2 Средства замера времени

Для выполнения измерений процессорного времени работы выполняемой программы использовалась функция *process_time* из модуля *time* [6]. На листинге 3.1 приведен пример замера процессорного времени с помощью функции *process_time*.

Листинг 3.1 – Пример замера затраченного времени

```
REPS = 100
t = 0
for i in range(0, REPS):
    beg = process_time()
    func(...)
    end = process_time()
    t += end - beg
print('Время:', t / REPS)
```

3.3 Сведения о модулях программы

Программа состоит из следующих модулей:

— `main.py` — файл, содержащий точку входа в программу;

— `utils.py` — файл, содержащий вспомогательные функции;

— `bruteforce.py` — файл, содержащий реализацию алгоритма полного перебора;

— `ants.py` — файл, содержащий реализацию муравьиного алгоритма;

— `test.py` — файл, содержащий функции замера процессорного времени работы алгоритмов и подбора параметров.

3.4 Реализация алгоритмов

3.5 Функциональные тесты

В таблице 3.1 приведены результаты функционального тестирования реализованных алгоритмов. Все тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

Матрица смежности	Полный перебор	Муравьиный алгоритм
$\begin{pmatrix} 0 & 4 & 2 & 1 & 7 \\ 4 & 0 & 3 & 7 & 2 \\ 2 & 3 & 0 & 10 & 3 \\ 1 & 7 & 10 & 0 & 9 \\ 7 & 2 & 3 & 9 & 0 \end{pmatrix}$	15, [0, 2, 4, 1, 3]	15, [0, 2, 4, 1, 3]
$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$	4, [0, 1, 2]	4, [0, 1, 2]
$\begin{pmatrix} 0 & 15 & 19 & 20 \\ 15 & 0 & 12 & 13 \\ 19 & 12 & 0 & 17 \\ 20 & 13 & 17 & 0 \end{pmatrix}$	64, [0, 1, 2, 3]	64, [0, 1, 2, 3]

Вывод

Были представлены листинги всех реализаций алгоритмов — полного перебора и муравьиного. Также в данном разделе была приведена информации о выбранных средствах для разработки алгоритмов и сведения о модулях программы, проведено функциональное тестирование.

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры по времени:

— процессор: AMD Ryzen 7 5800X (16) @ 3.800 ГГц.

— оперативная память: 32 ГБайт.

— операционная система: Manjaro Linux x86_64 (версия ядра Linux 6.5.12-1-MANJARO).

Измерения проводились на стационарном компьютере. Во время проведения измерений устройство было нагружено только системными приложениями.

4.2 Демонстрация работы программы

4.3 Временные характеристики

4.4 Постановка исследования

Автоматическая параметризация была проведена на двух классах данных — 4.4.1 и 4.4.2. Алгоритм будет запущен для набора значений $\alpha, \rho \in (0, 1)$, $elite\_amt \in \{1, 5, 10\}$ и $elite\_depos \in \{0.2, 0.4, 0.8, 1\}$.

Итоговая таблица значений параметризации будет состоять из следующих колонок:

— α — коэффициент жадности;

— ρ — коэффициент испарения;

— $elite\_amt$ — количество элитных муравьев;

— $elite\_depos$ — коэффициент усиления феромонов у элитных муравьев;

— $days$ — количество дней жизни колонии муравьёв;

— $Result$ — эталонный результат, полученный методом полного перебора для проведения данного исследования;

— *Mistake* — разность полученного основанным на муравьином алгоритме методом значения и эталонного значения на данных значениях параметров, показатель качества решения.

Цель исследования — определить комбинацию параметров, которые позволяют решить задачу наилучшим образом для выбранного класса данных. Качество решения зависит от количества дней и погрешности измерений.

4.4.1 Класс данных 1

Класс данных 1 представляет собой матрицу смежности размером 10 элементов (небольшой разброс значений: от 1 до 2), которая представлена в (4.1).

$$K_1 = \begin{pmatrix} 0 & 1 & 1 & 2 & 2 & 1 & 1 & 1 & 2 \\ 1 & 0 & 1 & 2 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 0 & 1 & 2 & 1 & 2 & 2 \\ 2 & 1 & 2 & 1 & 0 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 & 0 & 1 & 1 & 2 \\ 1 & 2 & 1 & 1 & 2 & 1 & 0 & 2 & 2 \\ 1 & 1 & 2 & 2 & 1 & 1 & 2 & 0 & 2 \\ 2 & 1 & 2 & 2 & 1 & 2 & 2 & 2 & 0 \end{pmatrix} \tag{4.1}$$

4.4.2 Класс данных 2

$$K_2 = \begin{pmatrix} 0 & 9271 & 8511 & 2010 & 1983 & 7296 & 7289 & 3024 & 1011 \\ 9271 & 0 & 7731 & 4865 & 5494 & 6812 & 4755 & 7780 & 7641 \\ 8511 & 7731 & 0 & 1515 & 9297 & 7506 & 5781 & 5804 & 7334 \\ 2010 & 4865 & 1515 & 0 & 3662 & 9597 & 2876 & 8188 & 9227 \\ 1983 & 5494 & 9297 & 3662 & 0 & 8700 & 4754 & 7445 & 3834 \\ 7296 & 6812 & 7506 & 9597 & 8700 & 0 & 4216 & 5553 & 8215 \\ 7289 & 4755 & 5781 & 2876 & 4754 & 4216 & 0 & 4001 & 4715 \\ 3024 & 7780 & 5804 & 8188 & 7445 & 5553 & 4001 & 0 & 9522 \\ 1011 & 7641 & 7334 & 9227 & 3834 & 8215 & 4715 & 9522 & 0 \end{pmatrix} \tag{4.2}$$

4.4.3 Класс данных 3

4.5 Вывод

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Киселева А. А.* Обзор методов решения задачи коммивояжера //. — Издательство «Научно-исследовательские публикации» (ООО «Вэлборн»), 2019.
2. *Борознов В. О.* Исследование решения задачи коммивояжера. — АГТУ, Вестник Астраханского государственного технического университета. — Режим доступа: `https://cyberleninka.ru/article/n/issledovanie-resheniya-zadachi-kommivoyazhera/viewer` (дата обращения: 12.12.2023).
3. *Дробот К. С*, *Едемская Е. Н.* Решение задачи коммивояжера с помощью муравьиного алгоритма // Сборник материалов XI Международной научно-технической конференции в рамках VI Международного Научного форума Донецкой Народной Республики. — 2020. — С. 344—348.
4. *Коцюбинская С. А.* Задача глобальной оптимизации. Муравьиный алгоритм // Modern Science. — 2020. — С. 537—540.
5. Welcome to Python [Электронный ресурс]. — URL: `https://www.python.org`.
6. time — Time access and conversions [Электронный ресурс]. — URL: `https://docs.python.org/3/library/time.html#functions`.