



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: «Задача коммивояжера»

Студент ИУ7-54Б
(Группа)

(Подпись, дата)

Булдаков М.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Волкова Л. Л.
(И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Аналитический раздел	4
1.1 Описание задачи	4
1.2 Алгоритмы решения задачи	4
2 Конструкторский раздел	7
2.1 Требования к программному обеспечению	7
2.2 Описание используемых типов данных	7
2.3 Разработка алгоритмов	8
3 Технологический раздел	13
3.1 Средства реализации	13
3.2 Сведения о модулях программы	13
3.3 Реализация алгоритмов	13
3.4 Функциональные тесты	14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	15

ВВЕДЕНИЕ

Задача коммивояжера является одной из классических задач комбинаторной оптимизации, привлекающей внимание исследователей и практиков в области логистики, транспорта и информационных технологий. В контексте логистики, эта задача применяется для оптимизации маршрутов доставки товаров и грузов, что позволяет сократить затраты на перевозку, уменьшить время доставки и улучшить эффективность работы логистических компаний [1].

Цель данной лабораторной работы — рассмотреть алгоритмы решения задачи коммивояжера в случае построения карты перемещений для воздухоплавателей. Для достижения поставленной цели необходимо выполнить следующие задачи:

- описать алгоритмы решения задачи коммивояжера;
- спроектировать программное обеспечение, реализующее алгоритмы решения задачи коммивояжера;
- выбрать инструменты для реализации и замера процессорного времени выполнения реализаций решения задачи;
- проанализировать затраты реализаций алгоритмов по времени.

1 Аналитический раздел

В данном разделе будут описаны алгоритмы решения задачи коммивояжера.

1.1 Описание задачи

Задача коммивояжера — одна из самых известных задач комбинаторной оптимизации, заключающаяся в поиске наиболее выгодного маршрута, проходящего через указанные города и возвращающегося в начальный пункт.

В общем случае задача формулируется следующим образом: имеется N городов, для каждой пары которых известно расстояние между ними. Требуется найти такой маршрут, проходящий через каждый город по одному разу (и возвращающийся в исходный город), при этом сумма всех расстояний на этом маршруте должна быть наименьшей из возможных [1].

В данной работе будет рассматриваться вариант задачи с незамкнутым маршрутом, т. е. без одного последнего перехода. Стоимость пути между городами будет отличаться в различных направлениях, поэтому граф будет ориентированным.

1.2 Алгоритмы решения задачи

Алгоритм полного перебора

Задача может быть решена перебором всех вариантов объезда и выбором оптимального. Очевидно, что при полном переборе будет найден самый кратчайший маршрут, но при этом для перебора необходимо будет выполнить порядка $O(N!)$ операций, где N — количество городов, что является тяжелой задачей даже для современных ЭВМ при N порядка сотни.

Муравьиный алгоритм (без элитных муравьев)

Муравьиный алгоритм — алгоритм решения задачи коммивояжера, основанный на принципе поведения колонии муравьев [2].

Муравьи действуют, руководствуясь органами чувств. Каждый муравей оставляет на своем пути феромоны, чтобы другие могли ориентироваться. При большом количестве муравьев наибольшее количество феромона остается на наиболее посещаемом пути, посещаемость же может быть связана с длинами ребер. Муравьи используют не прямой обмен информацией через окружающую

среду посредством феромона.

Основная идея заключается в том, что выделяются две фазы: день и ночь. В фазу дня каждый муравей k строит один маршрут, вечером обновляется лучшая траектория. В фазу ночи обновляется матрица феромона.

Каждый муравей имеет 3 способности:

- 1) Зрение — муравей k , стоя в городе i , может оценить привлекательность ребра $i - j$;
- 2) Обоняние — муравей чувствует концентрацию феромона $\tau_{ij}(t)$ на ребре $i-j$ в текущий день t ;
- 3) Память — муравей запоминает список, посещенных за текущий день t городов — $J_k(t)$.

Привлекательность ребра $i-j$ оценивается по формуле (1.1).

$$\eta_{ij} = \frac{1}{D_{ij}}, \quad (1.1)$$

где D_{ij} — метка ребра, D — матрица смежности.

Стоя в городе i , муравей k выбирает следующий город на основе вероятностного правила (1.2).

$$p_{ij,k} = \begin{cases} 0, j \in J_k, \\ \frac{\eta_{ij}^\alpha \cdot \tau_{ij}^\beta(t)}{\sum_{q \in J_k} \eta_{iq}^\alpha \cdot \tau_{iq}^\beta(t)}, \text{ иначе,} \end{cases} \quad (1.2)$$

где α — коэффициент жадности, β — коэффициент стадности, причем $\alpha + \beta = 1$.

После завершения движения всех муравьев (ночью, перед наступлением следующего дня), феромон обновляется по формуле (1.3).

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot (1 - \rho) + \Delta\tau_{ij}(t), \quad (1.3)$$

где $\rho \in [0, 1]$ — коэффициент испарения феромона.

$$\Delta\tau_{ij}(t) = \sum_{k=1}^N \Delta\tau_{ij,k}(t). \quad (1.4)$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} 0, \text{ муравей } k \text{ в день } t \text{ не ходил по ребру } i-j, \\ Q/L_k, \text{ иначе,} \end{cases} \quad (1.5)$$

где Q — квота феромона одного муравья на день, Q выбирается соразмерной длине лучшего маршрута в графе.

Чтобы значение феромона не обнулилось и не повлекло обнуление вероятности перехода по ребру, после расчета значения $\tau_{ij}(t+1)$ необходимо выполнять проверку матрицы феромона и все значения, которые меньше заданного порога, заменить на порог [2].

В данном алгоритме отсутствует полный перебор, что означает меньшую трудоемкость, чем для алгоритма полного перебора, но при этом лучшее решение не гарантируется.

2 Конструкторский раздел

В этом разделе будет представлено описание используемых типов данных, а также схематические изображения алгоритмов решения задачи коммивояжера.

2.1 Требования к программному обеспечению

Программа должна поддерживать два режима работы: режим массового замера времени и режим решения задачи коммивояжера.

Режим массового замера времени должен обладать следующей функциональностью:

- генерировать графы различного размера для проведения замеров;
- осуществлять массовый замер, используя сгенерированные данные;
- результаты массового замера должны быть представлены в виде таблицы и графика.

К режиму решения задачи коммивояжера выдвигается следующий ряд требований:

- возможность работать графами, записанными в файл;
- возможность вводить матрицы смежности графов;
- наличие интерфейса для выбора действий;
- на выходе программы, стоимость и маршрут кратчайшей длины.

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие структуры и типы данных:

- граф — множество вершин и ребер между ними, задается с помощью матрицы смежности;
- матрица — двумерный массив чисел.

2.3 Разработка алгоритмов

На рисунке 2.1 представлена схема алгоритма решения задачи коммивояжера полным перебором. На рисунке 2.2 представлена схема муравьиного алгоритма.

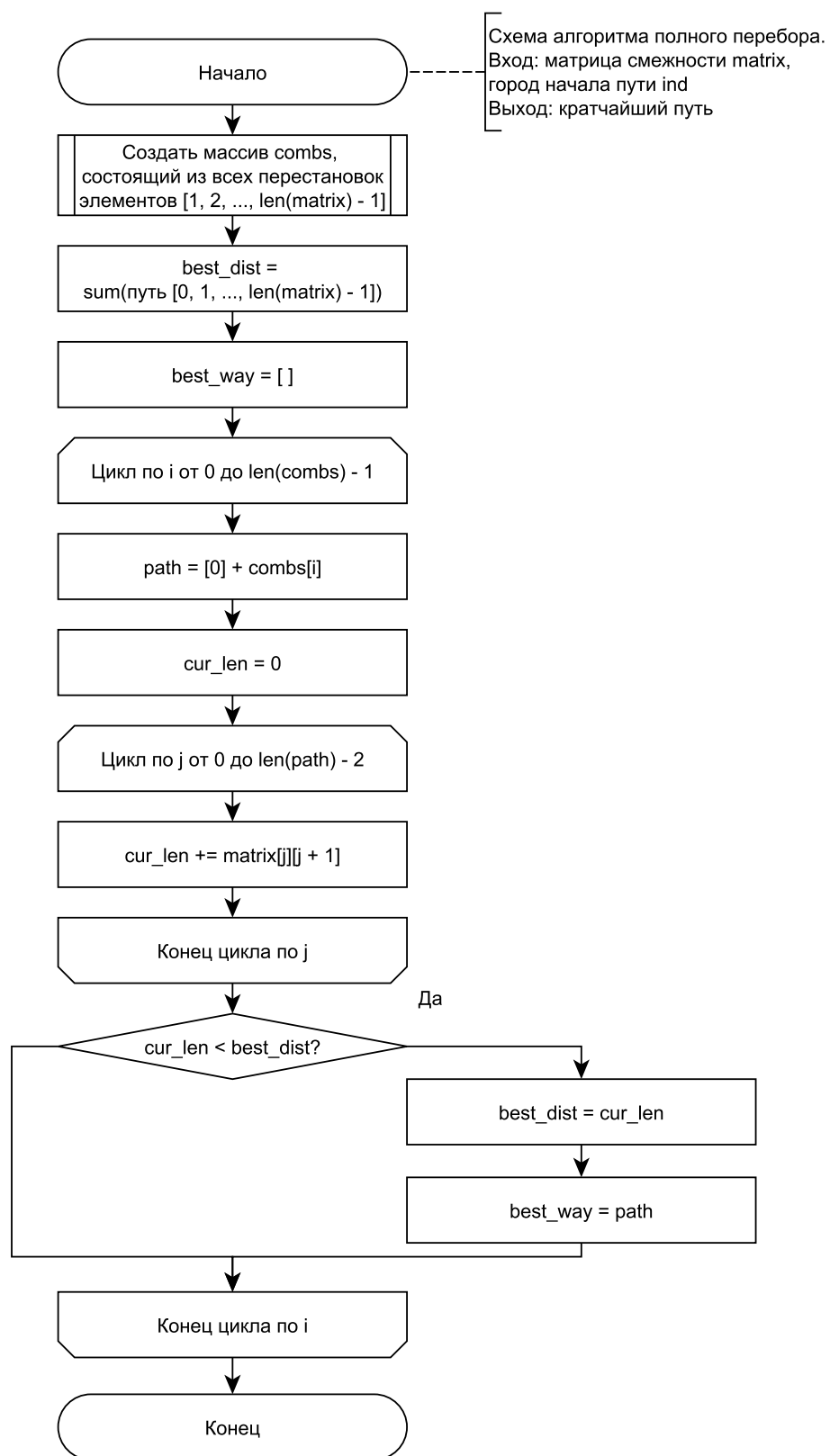


Рисунок 2.1 – Схема алгоритма решения задачи коммивояжера полным перебором

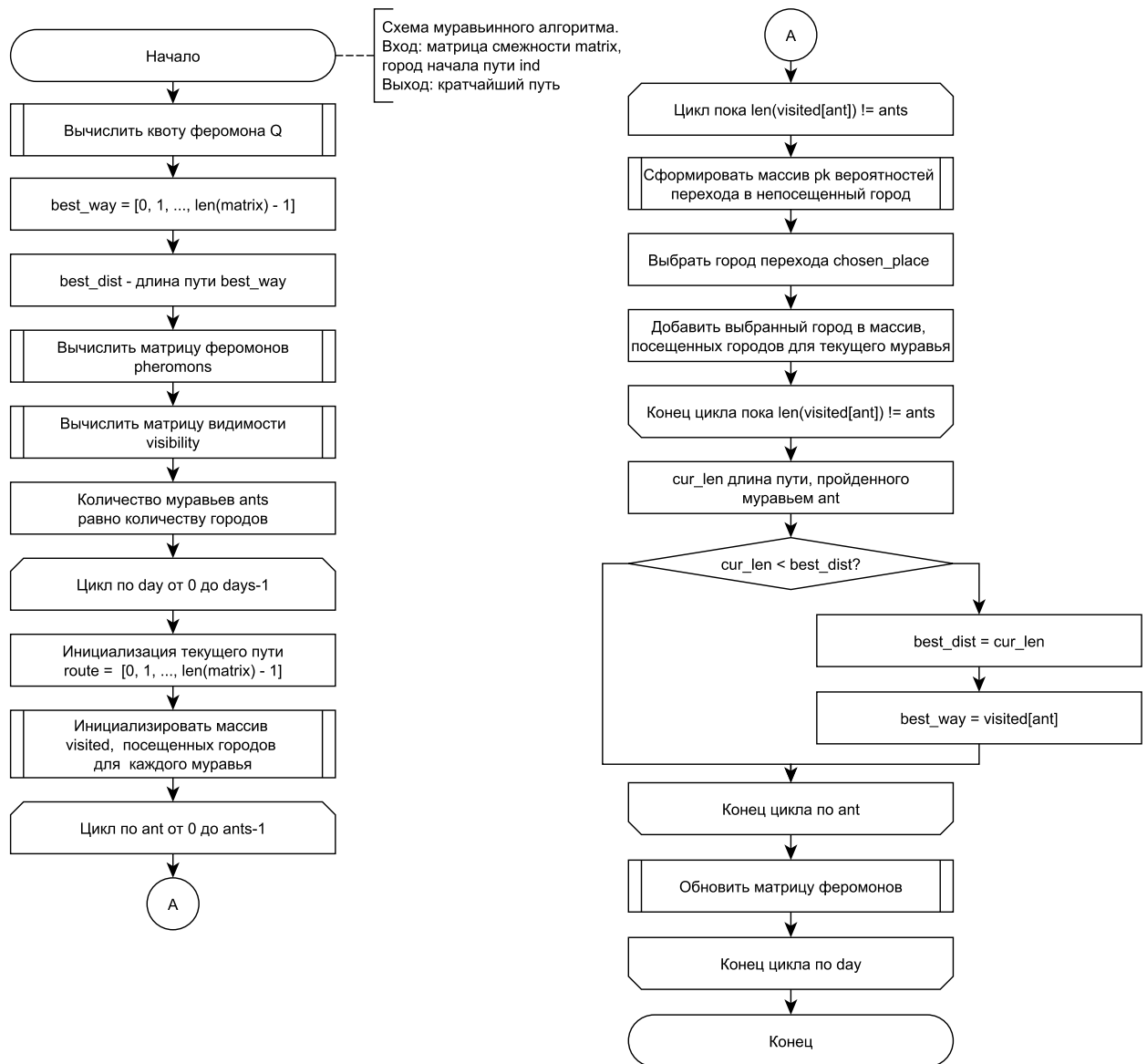


Рисунок 2.2 – Схема муравьиного алгоритма

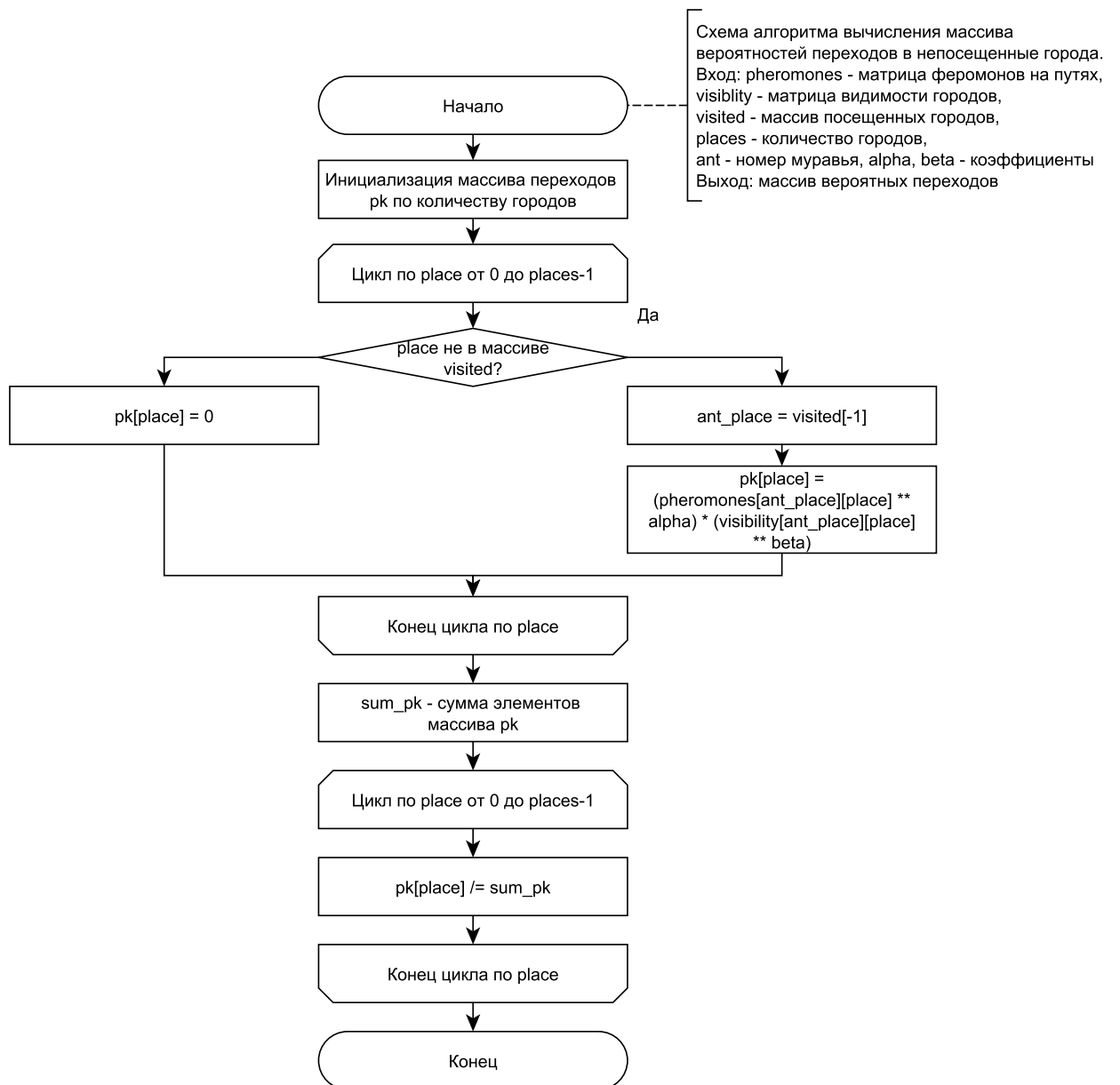


Рисунок 2.3 – Схема алгоритма вычисления массива вероятностей переходов в непосещенные города

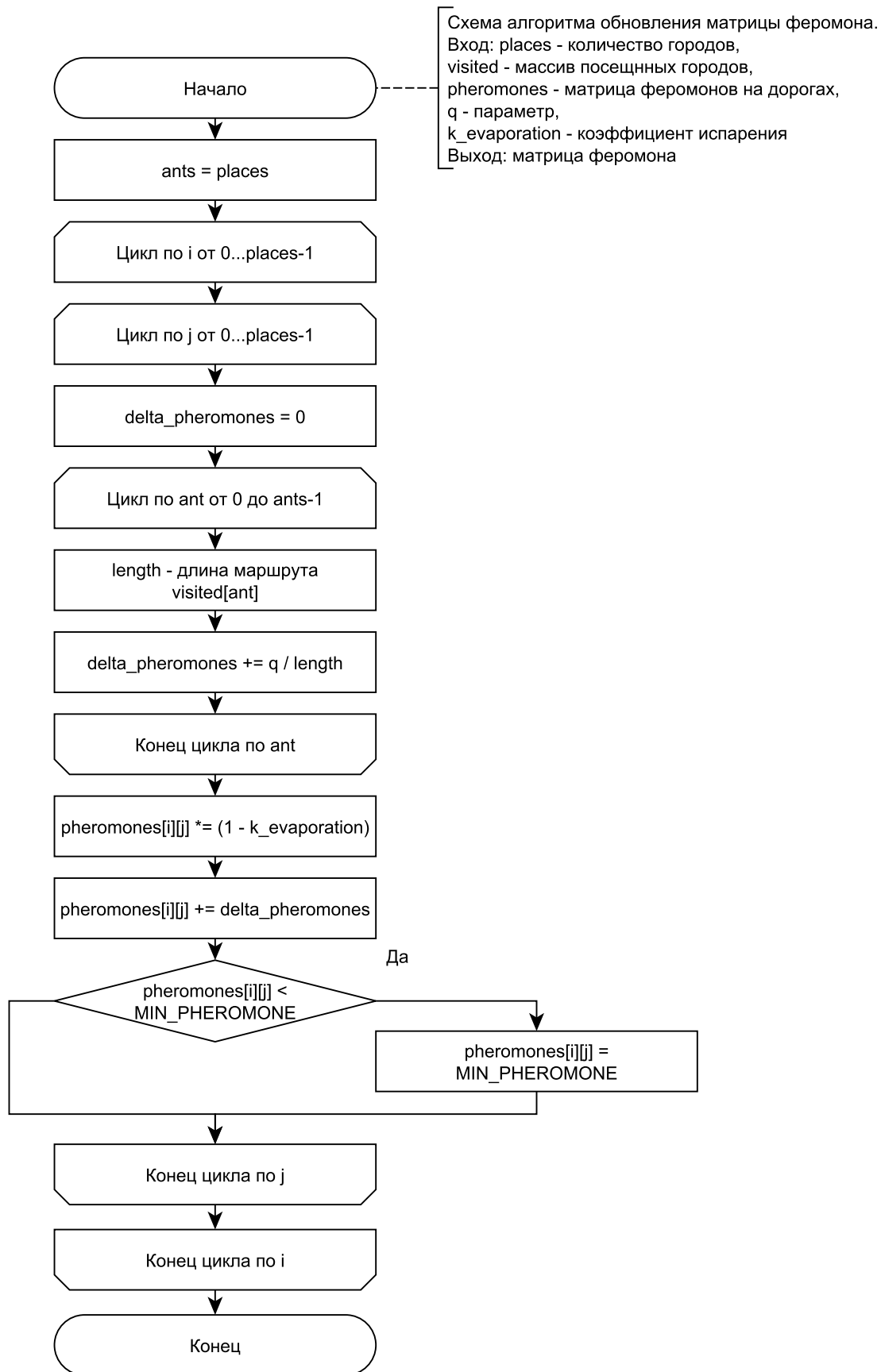


Рисунок 2.4 – Схема алгоритма обновления матрицы феромона

3 Технологический раздел

В данном разделе будут приведены требования к программному обеспечению, средства реализации, листинг кода и функциональные тесты.

3.1 Средства реализации

Для реализации данной работы был выбран язык *Python* [3]. Данный выбор обусловлен следующим:

- язык поддерживает все структуры данных, которые выбраны в результате проектирования;
- язык позволяет реализовать все алгоритмы, выбранные в результате проектирования;
- язык позволяет замерять процессорное время с помощью модуля *time*.

Процессорное время было замерено с помощью функции *process_time()* из модуля *time* [4].

3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

- *main.py* — файл, содержащий функцию *main*;
- *algorithms.py* — файл, содержащий код реализаций всех алгоритмов сортировок;
- *compare_time.py* — файл, в котором содержатся функции для замера и вывода времени выполнения реализаций алгоритмов.

3.3 Реализация алгоритмов

3.4 Функциональные тесты

В таблице 3.1 приведены функциональные тесты для разработанных алгоритмов сортировок. Все тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

Массив	Ожидаемый результат	Фактический результат
[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]	[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]
[]	[]	[]
[1]	[1]	[1]
[4, 1, 2, 3]	[1, 2, 3, 4]	[1, 2, 3, 4]
[2, 1]	[1, 2]	[1, 2]
[31, 57, 24, -10, 59]	[-10, 24, 31, 57, 59]	[-10, 24, 31, 57, 59]

Вывод

Были разработаны и протестированы спроектированные алгоритмы сортировок: Шелла, гномья и пирамидальная.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Володина Е.В. С. Е.* Практическое применение алгоритма решения задачи коммивояжера // ИВД. — 2015. — № 2.
2. *Colormi A., Dorigo M., Maniezzo V.* Distributed Optimization by Ant Colonies // Proceedings of the First European Conference on Artificial Life. — 1991.
3. The official home of the Python Programming Language [Электронный ресурс]. — Режим доступа: <https://www.python.org/> (дата обращения: 19.09.2023).
4. time — Time access and conversions [Электронный ресурс]. — Режим доступа: <https://docs.python.org/3/library/time.html> (дата обращения: 19.09.2023).