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»

Тема Поиск по словарю

Студент Калашков П. А.

Группа ИУ7-56Б

Оценка (баллы)

Преподаватели Волкова Л. Л., Строганов Ю. В.

Москва — 2022 г.

Содержание

Введение

В процессе развития компьютерных систем количество обрабатываемых данных увеличивалось, вледствие чего множество операций над наборами данных стали выполняться очень долго, поскольку чаще всего это был обычный перебор. Это вызвало необходимость создать новые алгоритмы, которые решают поставленную задачу на порядок быстрее стандартного решения прямого обхода. В том числе это касается и словарей, в которых одной из основных операций является операция поиска.

Цель работы: получить навык поиска по словарю при ограничении на значение признака, заданном при помощи лингвистической переменной. Для достижения поставленной цели необходимо выполнить следующие задачи:

1) формализовать объект и его признак;
2) составить анкету для заполнения респондентом;
3) провести анкетирование респондентов;
4) построить функцию принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов;
5) описать 3–5 типовых вопросов на русском языке, имеющих целью запрос на поиск в словаре;
6) описать алгоритм поиска в словаре объектов, удоволетворяющих ограничению, заданному в вопросе на ограниченном естественном языке;
7) описать структуру данных словаря, хранящего наименование объектов согласно варианту и числовое значение признака объекта;
8) реализовать описанный алгоритм поиска в словаре;
9) привести примеры запросов пользователя и сформированной реализацией алгоритма поиска выборки объектов из словаря, используя составленные респондентами вопросы;

10) дать заключение о применимости предложенного алгоритма и его ограничениях;

11) описать и обосновать полученные результаты в виде отчёта о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В этом разделе будет представлена информация о формализации объекта «человек» и его признака «рост» по варианту, составлена анкета для заполнения респондентами, проведено анкетирование респондентов и построена функция принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов. Также будет описаны структура данных словаря и алгоритм поиска в словаре.

1.1 Формализация объекта и его признака

Согласно согласованному варианту, формализуем объект «человек» следующим образом: определим набор данных и признак объекта, на основании которого составим набор термов. Набор данных:

1) имя человека — строка;
2) пол человека — строка;
3) родная страна человека — строка;
4) профессия человека — строка.

Согласно варианту, признаком, по которому будет производиться поиск объектов, будет являться *рост* в сантиметрах — целое число.

Определим следующие термы, соответствующие признаку «рост»:

1) «Очень низкий»;
2) «Низкий»;
3) «Средний»;
4) «Высокий»;
5) «Очень высокий».

Также введём универсальное для данной задачи множество оцениваемой величины (роста) H:

$$H = \{80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220\} \quad (1.1)$$

1.2 Анкетирование респондентов

Было проведено анкетирование следующих респондентов:

1) Соловов Юрий, группа ИУ7-56Б — Респондент 1;
2) Чепрасов Кирилл, группа ИУ7-56Б — Респондент 2;
3) Виноградов Алексей, группа ИУ7-56Б — Респондент 3;
4) Авсюнин Алексей, группа ИУ7-56Б — Респондент 4;
5) Ковель Александр, группа ИУ7-56Б — Респондент 5;
6) Комаров Никита, группа ИУ7-52Б — Респондент 6.

Респонденты, выступающие в качестве экспертов, для каждого из приведённых выше термов указали соответствующий промежуток, элементами которого являются числа из введённого для поставленной задачи множества оценимоемой величины.

Результаты анкетирования перечисленных респондентов продемонстрированы в таблице 1.1. В данной таблице Респ. — сокращение от «Респондент», термы 1 – 5 — термы, соответствующие обозначенным в п. 1.1 термам.

Таблица 1.1 – Результаты анкетирования

Терм	Респ. 1	Респ. 2	Респ. 3	Респ. 4	Респ. 5	Респ.6
1	[80; 110)	[80; 140]	[80; 150]	[80; 140]	[80; 130]	[80; 149]
2	[110; 150)	(140; 160]	(150; 165)	(140; 160]	(130; 150]	[150; 164]
3	[150; 180)	(160; 180]	[165; 190]	(160 ; 180]	(150; 170]	[165; 179]
4	[180; 190)	(180; 190]	[190; 200]	(180; 195]	(170; 190]	[180; 195]
5	[190; 220]	(190; 220]	(200; 220]	(195; 220]	(190; 220]	[196; 220]

1.2.1 Построение функции принадлежности термам

Построим графики функций принадлежности числовых значений переменной термам, описывающим группы значений лингвистической переменной.

Для этого для каждого значения из H для каждого терма из перечисленных найдём количество респондентов, согласно которым значение из H удоволетворяет сопоставляемому терму. Данное значение поделим на количество респондентов — это и будет значением функции μ для терма в точке.

1.3 Словарь

Словарь (англ. *dictionary*) [1] — тип данных, который позволяет хранить пары вида «ключ — значение» (англ. *key — value*) (k, v). Он поддерживает три операции:

1) добавление пары;

2) поиск по ключу;

3) удаление по ключу.

В паре (k, v) v — значение, ассоциируемое с k.

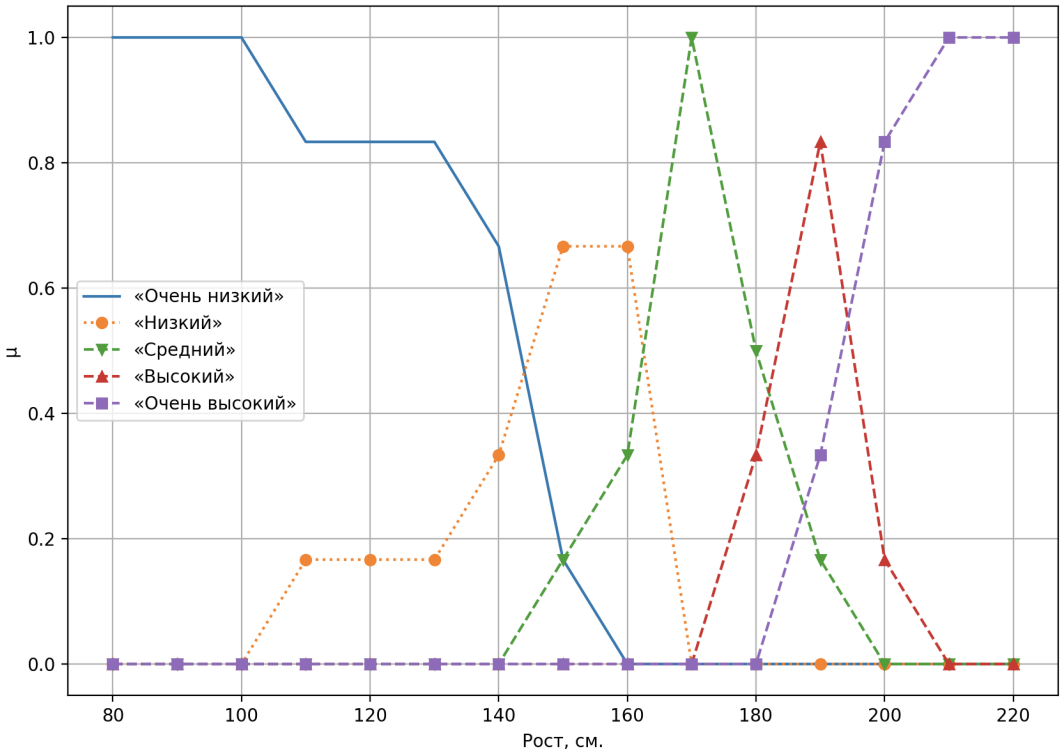

Рисунок 1.1 – Графики функций принадлежности числовых значений переменной термам, описывающим группы значений лингвистической переменной

1.3.1 Алгоритм поиска в словаре

Рассмотрим алгоритм поиска в словаре объектов, удоволетворяющих ограничению, заданному в вопросе на ограниченном естественном языке. Предлагается использовать метод полного перебора, рассмотренный ниже.

Метод полного перебора [2] — метод решения поиска значения в словаре, при котором поочерёдно перебираются все ключи словаря, пока не будет найден нужный.

Чем дальше искомый ключ от начала словаря, тем выше трудоемкость алгоритм. Так, если на старте алгоритм затрагивает b операций, а при сравнении k операций, то:

— элемент найден на первом сравнении за $b+k$ операций (лучший случай);

— элемент найден на i-ом сравнении за $b + i \cdot k$ операций;

— элемент найден на последнем сравнении за $b + N \cdot k$ операций, где N – размер словаря (худший случай).

Если обозначить трудоёмкость выполняемых операций равной единице, то средняя трудоёмкость равна:

$$f = b + k \cdot \left(1 + \frac{N}{2} - \frac{1}{N+1}\right) \tag{1.2}$$

Прежде, чем искать необходимые значения для отдельных ключей, необходимо выделить набор отдельных ключей как ограничение, заданное на ограниченном естественном языке (русском). Рассмотрим следующие этапы выделения ограничения:

1) найти вхождение слова, описывающего объект (т. е. в запросе должна идти речь о людях);

2) найти вхождение слов или словосочетаний, указывающих на признак (т. е. в запросе должна идти речь о росте);

3) найти терм;

Для каждого из пунктов определим набор слов и словосочетаний, которые должны присутствовать в запросе для того, чтобы он считался корректным, т. е. гарантированно содержал информацию, достаточную для определения терма, поиск по которому будет осуществляться в словаре.

Для определения, что в запросе идёт речь об объекте «человек» достаточно, чтобы в нём присутствовал ходя бы один из приведённых далее элементов: «человек», «людей».

Для определения, что в запросе идёт речь о росте, в запросе должно присутствовать слово с корнем«рост».

Для определения терма, о котором идёт речь в запросе, необходимо проверить присутствие следующих элементов для каждого из термов соответственно:

1) «очень низкого»;

2) «низкого» (причём перед этим словом не стоит слово «очень»);

3) «среднего»;

4) «высокого» (причём перед этим словом не стоит слово «очень»);

5) «очень высокого»;

Также необходимо обработать ситуации, в которых перед выделенными словами или словосочетаниями стоит слово «не» — это будет говорить о том, что необходимо найти все объекты, не соответствующие указанному терму.

Запросы, в которых отсутствует информация о термах, идёт речь не об объектах типа «человек» или не об их росте, или в которых присутствует информация о нескольких термах сразу, будут считаться некорректными запросами.

1.4 Требования к программе

Выделим ряд требований к разрабатываемой программе:

— программа должна получать на вход матрицу смежности, для которой можно будет выбрать один из алгоритмов поиска оптимальных путей — полным перебором или муравьиным алгоритмом;

— программа должна позволять пользователю определять коэффициенты и количество дней для муравьиного алгоритма;

— программа должна давать возможность получить минимальную сумму пути, а также сам путь, используя один из алгоритмов. Также должна присутствовать возможность провести тестирование по времени выполнения для разных размеров матриц.

Вывод

В данном разделе была рассмотрена задача коммивояжёра, а также полный перебор для её решения и муравьиный алгоритм. Были представлены требования к разрабатываемому программному обеспечению.

2 Конструкторская часть

В данном разделе будут представлены требования к разрабатываемому программному обеспечению, описание используемых типов данных, а также схемы алгоритма полного перебора и муравьиного алгоритма и классы эквивалентности для функционального тестирования.

2.1 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие типы данных:

— размер матрицы смежности — целое число;

— имя файла — строка;

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

— матрица смежности — матрица целых чисел.

2.2 Разработка алгоритмов

На рисунке 2.1 представлена схема алгоритма полного перебора путей, а на рисунках 2.2–2.3 схема муравьиного алгоритма поиска путей. Также на рисунках 2.4–2.6 представлены схемы вспомогательных функций для муравьиного алгоритма.

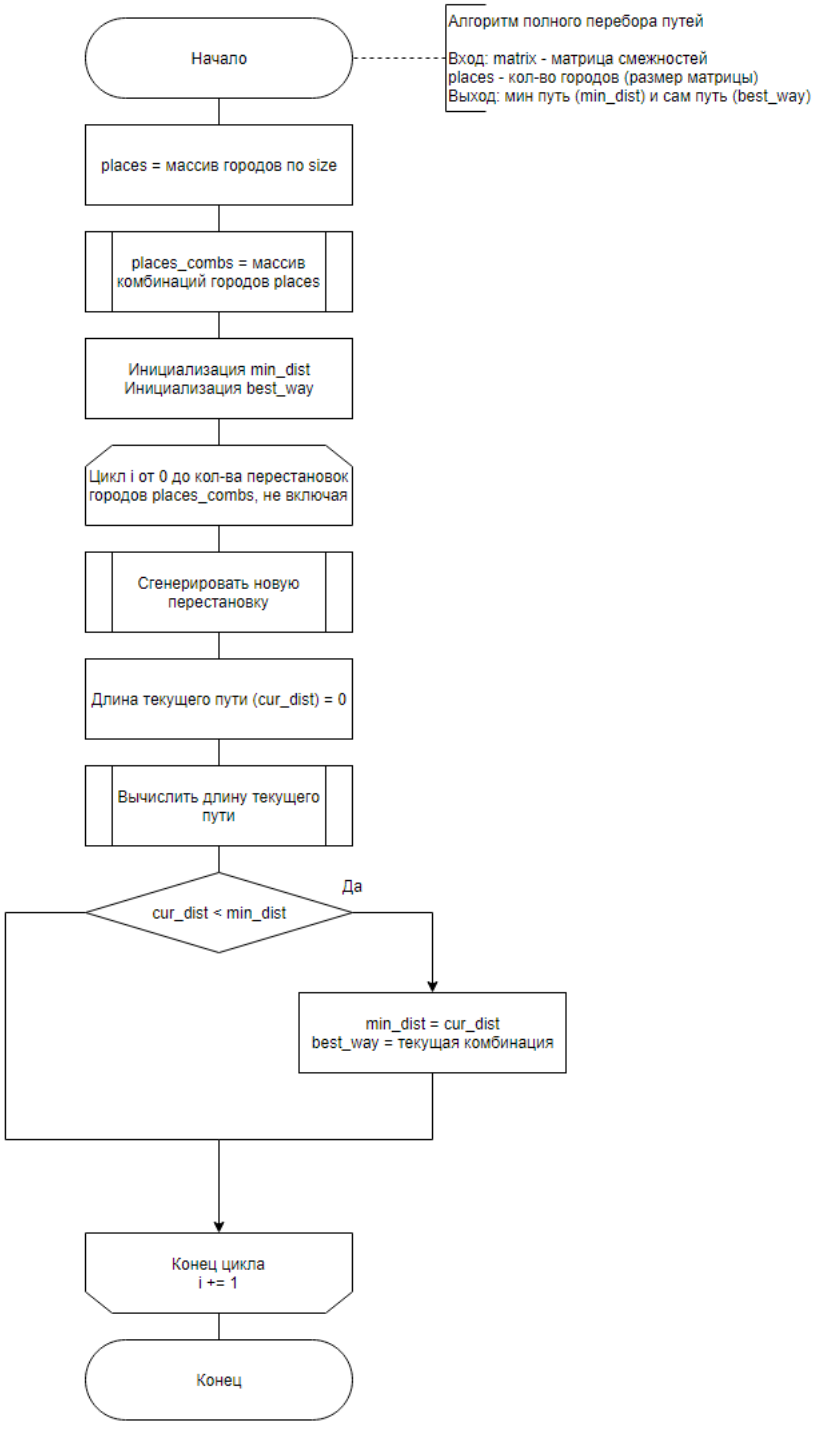

Рисунок 2.1 – Схема алгоритма полного перебора путей

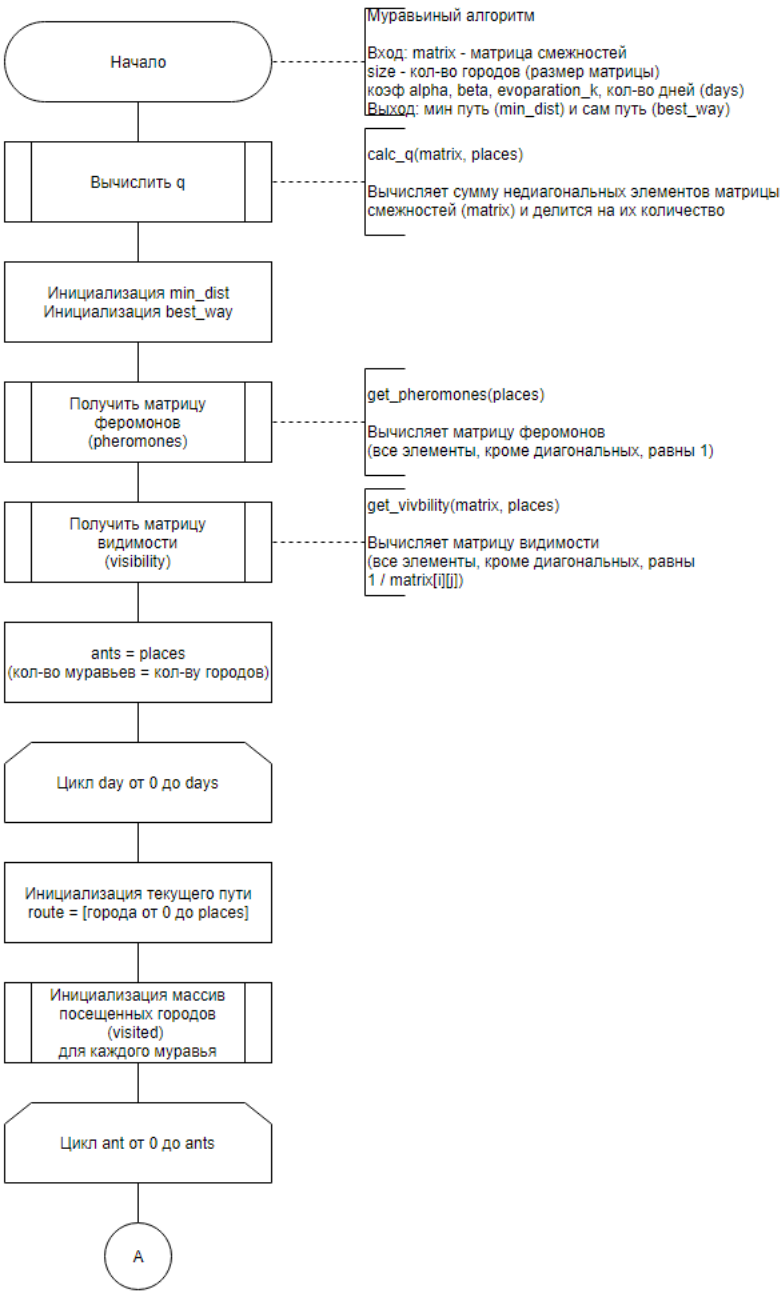

Рисунок 2.2 – Схема муравьиного алгоритма (часть 1)

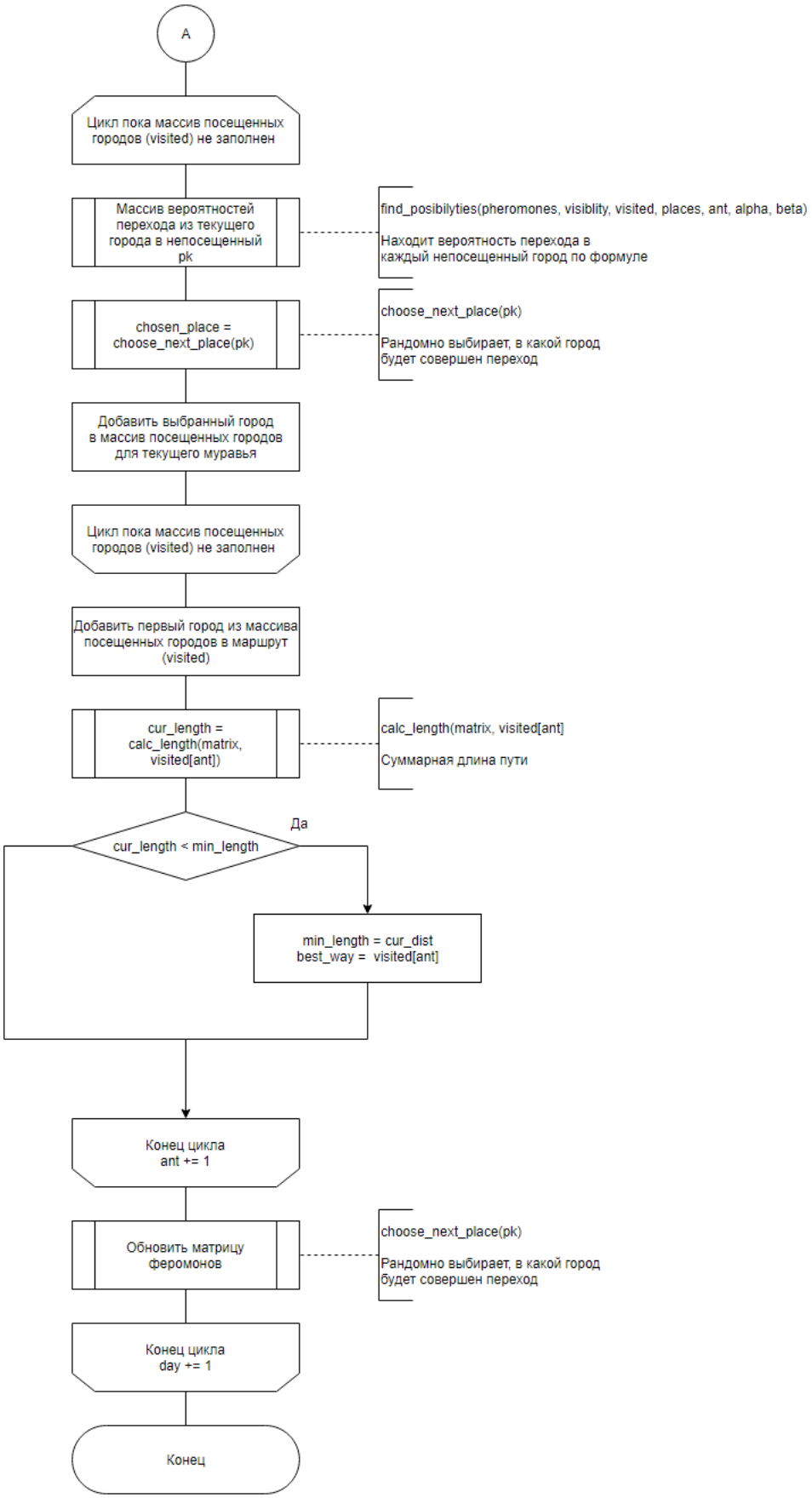

Рисунок 2.3 – Схема муравьиного алгоритма (часть 2)

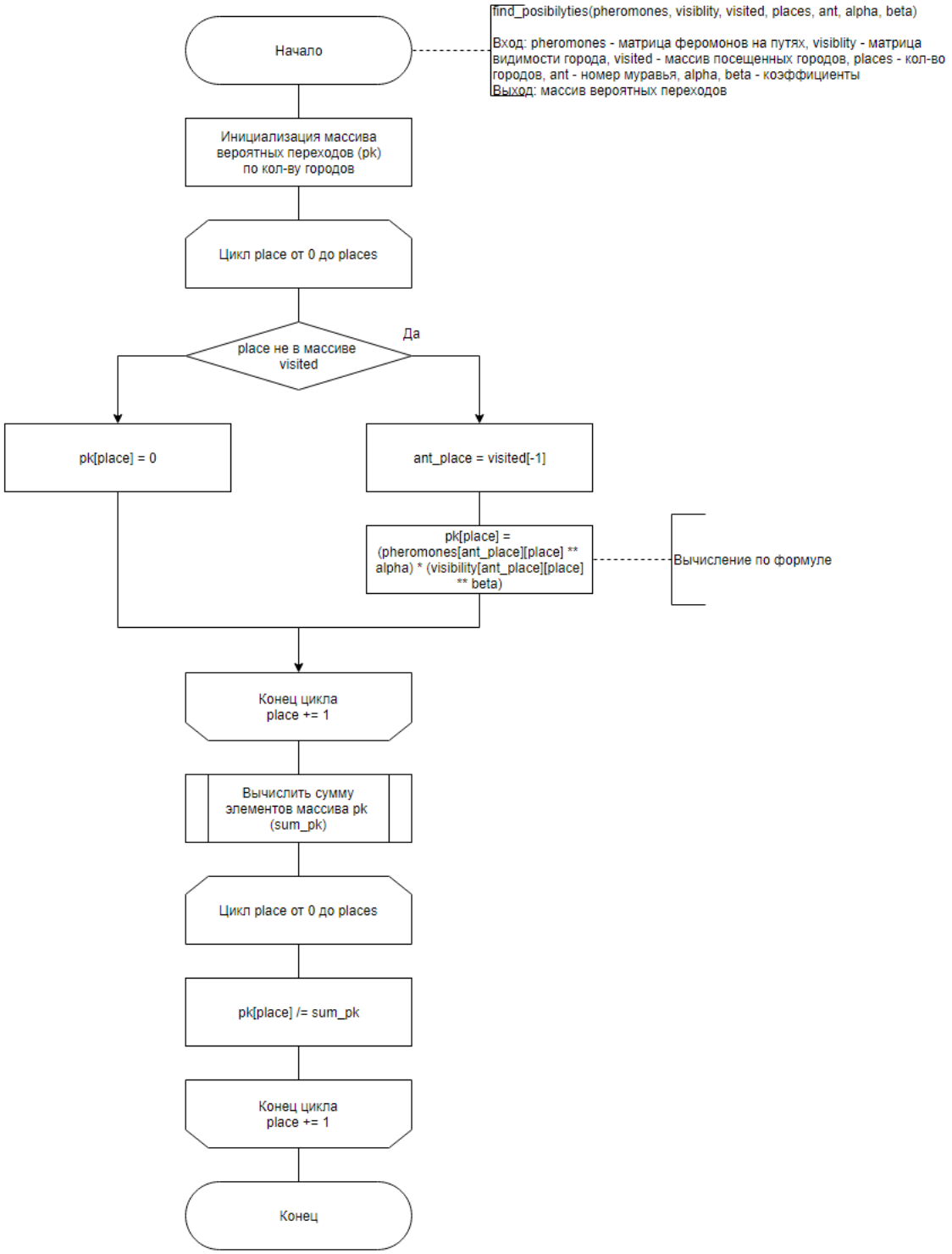

Рисунок 2.4 – Схема алгоритма нахождения массива вероятностей переходов в непосещенные города

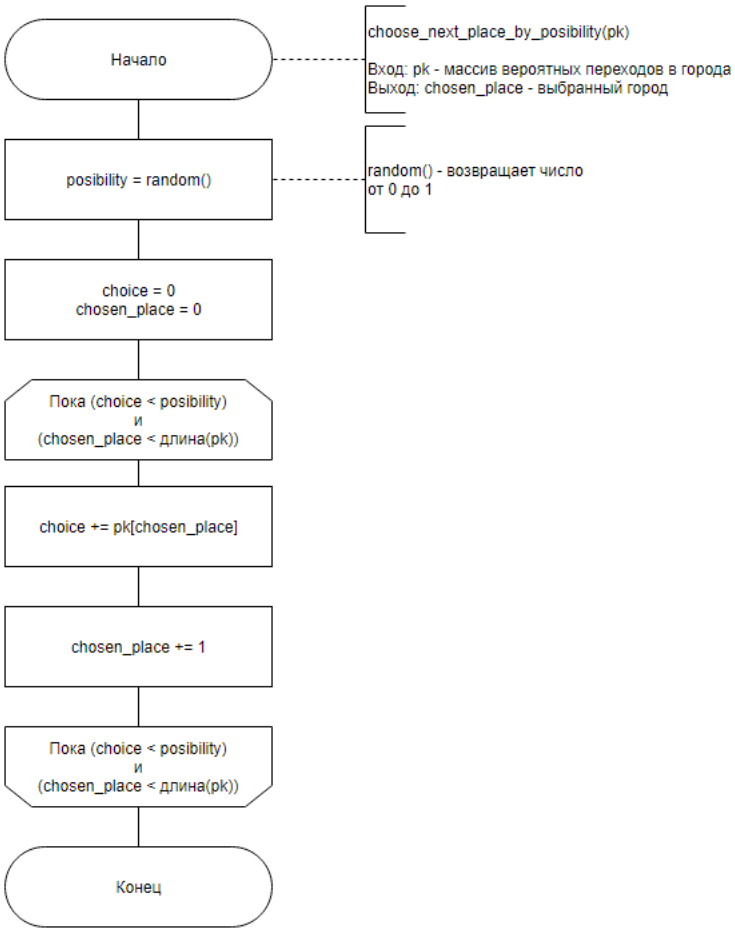

Рисунок 2.5 – Схема алгоритма выбора следующего города

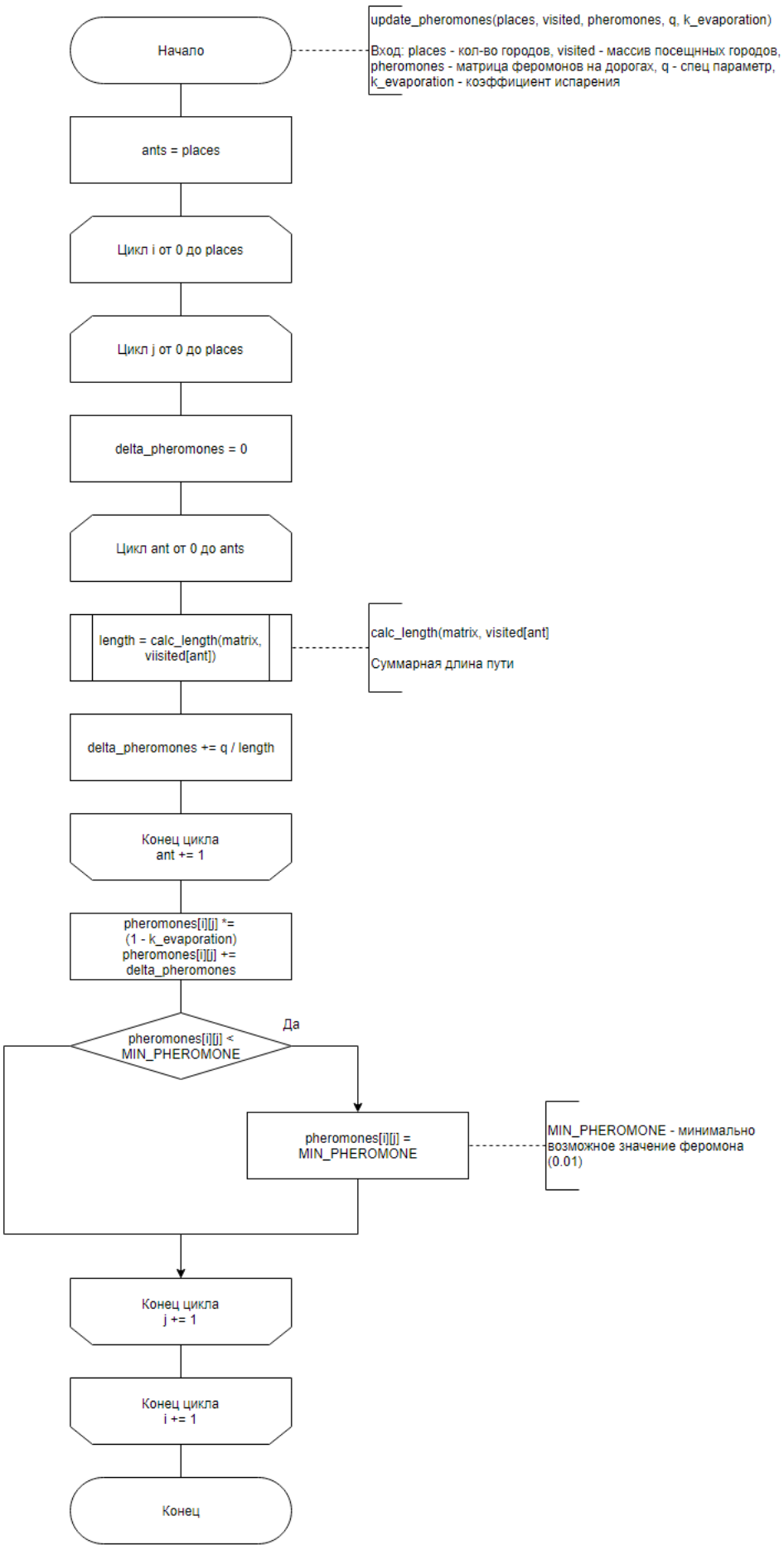

Рисунок 2.6 – Схема алгоритма обновления матрицы феромонов

2.3 Классы эквивалентности при функциональном тестировании

Для функционального тестирования выделены классы эквивалентности, представленные ниже.

1. Неверно выбран пункт меню — не число или число, меньшее 0 или большее 8.
2. Неверно введены коэффициенты α, β, $evaporation\_koef$ — не число или число, меньшее 0.
3. Неверно введено количество дней — не число или число, меньшее 0.
4. Неверно введен размер матрицы — не число или число, меньшее 2.
5. Корректный ввод всех параметров.

Вывод

В данном разделе были представлены описания используемых типов данных, а также схемы алгоритма полного перебора и муравьиного алгоритма и классы эквивалентности для функционального тестирования.

3 Технологическая часть

В данном разделе будут рассмотрены средства реализации, а также представлены листинги алгоритма, осуществляющего полный перебор путей, и муравьиного алгоритма.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python* [3]. В текущей лабораторной работе требуется замерить процессорное время работы выполняемой программы и визуализировать результаты при помощи графиков. Инструменты для этого присутствуют в выбранном языке программирования.

Время работы было замерено с помощью функции *process_time(...)* из библиотеки *time* [4].

3.2 Сведения о модулях программы

Программа состоит из шести модулей:

1) *main.py* — файл, содержащий точку входа;
2) *menu.py* — файл, содержащий код меню программы;
3) *test.py* — файл, содержаший код тестирования алгоритмов;
4) *utils.py* — файл, содержащий служебные алгоритмы;
5) *constants.py* — файл, содержаший константы программы;
6) *algorythms.py* — файл, содержащий код всех алгоритмов.

3.3 Реализации алгоритмов

В листинге 3.1 представлен алгоритм полного перебора путей, а в листингах 3.2–3.6 — муравьиный алгоритм и дополнительные к нему функции.

Листинг 3.1 – Реализация алгоритма полного перебора путей

```
1         def fullCombinationAlg(matrix, size):
2             places = np.arange(size)
3             placesCombinations = list()
4
5             for combination in it.permutations(places):
6                 combArr = list(combination)
7                 placesCombinations.append(combArr)
8
9             minDist = float("inf")
10
11            for i in range(len(placesCombinations)):
12                placesCombinations[i].append(\
13                placesCombinations[i][0])
14                curDist = 0
15                for j in range(size):
16                    startCity = placesCombinations[i][j]
17                    endCity = placesCombinations[i][j + 1]
18                    curDist += matrix[startCity][endCity]
19
20                if (curDist < minDist):
21                    minDist = curDist
22                    bestWay = placesCombinations[i]
23
24            return minDist, bestWay
```

Листинг 3.2 – Реализация муравьиного алгоритма

```
1     def antAlgorithm(matrix, places, alpha, beta, k_evaporation,
          days):
2         q = calcQ(matrix, places)
3         bestWay = []
4         minDist = float("inf")
5         pheromones = calcPheromones(places)
6         visibility = calcVisibility(matrix, places)
7         ants = places
8         for day in range(days):
9             route = np.arange(places)
10            visited = calcVisitedPlaces(route, ants)
11            for ant in range(ants):
12                while (len(visited[ant]) != ants):
13                    pk = findWays(pheromones, visibility, visited,
                          places, ant, alpha, beta)
14                    chosenPlace = chooseNextPlaceByPosibility(pk)
15                    visited[ant].append(chosenPlace - 1)
16
17                visited[ant].append(visited[ant][0])
18
19                curLength = calcLength(matrix, visited[ant])
20
21                if (curLength < minDist):
22                    minDist = curLength
23                    bestWay = visited[ant]
24
25            pheromones = updatePheromones(matrix, places, visited,
                  pheromones, q, k_evaporation)
26
27        return minDist, bestWay
```

Листинг 3.3 – Реализация алгоритма нахождения массива вероятностей переходов в непосещенные города

```
def findWays(pheromones, visibility, visited, places, ant, alpha,
    beta):
     pk = [0] * places

     for place in range(places):
         if place not in visited[ant]:
             ant_place = visited[ant][-1]
             pk[place] = pow(pheromones[ant_place][place], alpha) * \
             pow(visibility[ant_place][place], beta)
         else:
             pk[place] = 0

     sum_pk = sum(pk)

     for place in range(places):
         pk[place] /= sum_pk

     return pk
```

Листинг 3.4 – Реализация алгоритма нахождения массива вероятностей переходов в непосещенные города

```
def calcPheromones(size):
min_phero = 1
pheromones = [[min_phero for i in range(size)] for j in range(size)]
return pheromones
```

Листинг 3.5 – Реализация алгоритма выбора следующего города

```
def chooseNextPlaceByPosibility(pk):
    posibility = random()
    choice = 0
    chosenPlace = 0

    while ((choice < posibility) and (chosenPlace < len(pk))):
        choice += pk[chosenPlace]
        chosenPlace += 1

    return chosenPlace
```

Листинг 3.6 – Реализация алгоритма обновления матрицы феромонов

```
    def updatePheromones(matrix, places, visited, pheromones, q,
        k_evaporation):
        ants = places

        for i in range(places):
            for j in range(places):
                delta = 0
                for ant in range(ants):
                    length = calcLength(matrix, visited[ant])
                    delta += q / length

                pheromones[i][j] *= (1 - k_evaporation)
                pheromones[i][j] += delta
                if (pheromones[i][j] < MIN_PHEROMONE):
                    pheromones[i][j] = MIN_PHEROMONE

        return pheromones
```

3.4 Функциональные тесты

В таблице 3.1 приведены тесты для функций программы. Все функциональные тесты пройдены *успешно*.

Таблица 3.1 – Функциональные тесты

Матрица смежности	Ожидаемый результат	Результат программы
$\begin{pmatrix} 0 & 4 & 2 & 1 & 7 \\ 4 & 0 & 3 & 7 & 2 \\ 2 & 3 & 0 & 10 & 3 \\ 1 & 7 & 10 & 0 & 9 \\ 7 & 2 & 3 & 9 & 0 \end{pmatrix}$	15, [0, 2, 4, 1, 3, 0]	15, [0, 2, 4, 1, 3, 0]
$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$	4, [0, 1, 2, 0]	4, [0, 1, 2, 0]
$\begin{pmatrix} 0 & 15 & 19 & 20 \\ 15 & 0 & 12 & 13 \\ 19 & 12 & 0 & 17 \\ 20 & 13 & 17 & 0 \end{pmatrix}$	64, [0, 1, 2, 3, 0]	64, [0, 1, 2, 3, 0]

3.5 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

В данном разделе будет приведён пример работы программы, а также проведён сравнительный анализ алгоритмов при различных ситуациях на основе полученных данных.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры времени работы реализаций описанных алгоритмов, представлены далее:

— операционная система Mac OS Monterey Версия 12.5.1 (21G83) [5] x86_64;

— память 16 ГБ;

— четырёхъядерный процессор Intel Core i7 с тактовой частотой 2,7 ГГц [6].

При тестировании ноутбук был включен в сеть электропитания. Во время замеров и нагрузочного тестирования ноутбук был нагружен только встроенными приложениями окружения, а также системой тестирования.

4.2 Демонстрация работы программы

На рисунке 4.1 представлен пример работы программы для обоих алгоритмов — полного перебора и муравьиного. Осуществляется выбор файла с данными, ввод коэффициентов для муравьиного алгоритма, а также выполнение алгоритма решения задачи коммивояжёров методом полного перебора и методом на основе муравьиного алгоритма.

```
p.kalashkov in ~/Desktop/fifthTerm/bmstu-aa/lab06/src on branch master > python3 main.py
Меню
1. Полный перебор
2. Муравьиный алгоритм
3. Все алгоритмы
4. Параметризация
5. Замерить время
6. Обновить данные
7. Распечатать матрицу
0. Выход
Выбор: 3


Доступные файлы:  3  штук
1. 1.csv
2. mat9_highdif.csv
3. mat9_lowdif.csv

Выберите файл: 3


Введите коэффициент alpha: 0.7
Введите коэффициент evaporation: 0.9
Введите кол-во дней: 200


Алгоритм полного перебора
        Минимальная длина пути =  9
        Путь:  [0, 1, 8, 4, 3, 6, 2, 5, 7, 0]


Муравьиный алгоритм
        Минимальная длина пути =  9
        Путь:  [2, 6, 3, 4, 8, 1, 5, 7, 0, 2]
```

Рисунок 4.1 – Пример работы программы

4.3 Время выполнения алгоритмов

Для замера процессорного времени используется функция *process_ time(...)* из библиотеки *time* на *Python*. Функция возвращает процессорное время типа float в секундах.

Использовать функцию приходится дважды, затем из конечного времени нужно вычесть начальное, чтобы получить результат.

Замеры проводились для разного размера матриц, чтобы определить, когда наиболее эффективно использовать муравьиный алгоритм.

Результаты замеров приведены в таблице 4.1 (время в с).

Таблица 4.1 – Результаты замеров времени

Размер	Полный перебор	Муравьиный
2	0.000130	0.019932
3	0.000138	0.031615
4	0.000104	0.044361
5	0.000420	0.089291
6	0.002390	0.152131
7	0.019703	0.254059
8	0.162850	0.398472
9	1.637611	0.594024
10	18.207853	0.857666

Также на рисунке 4.2 приведены графические результаты замеров.

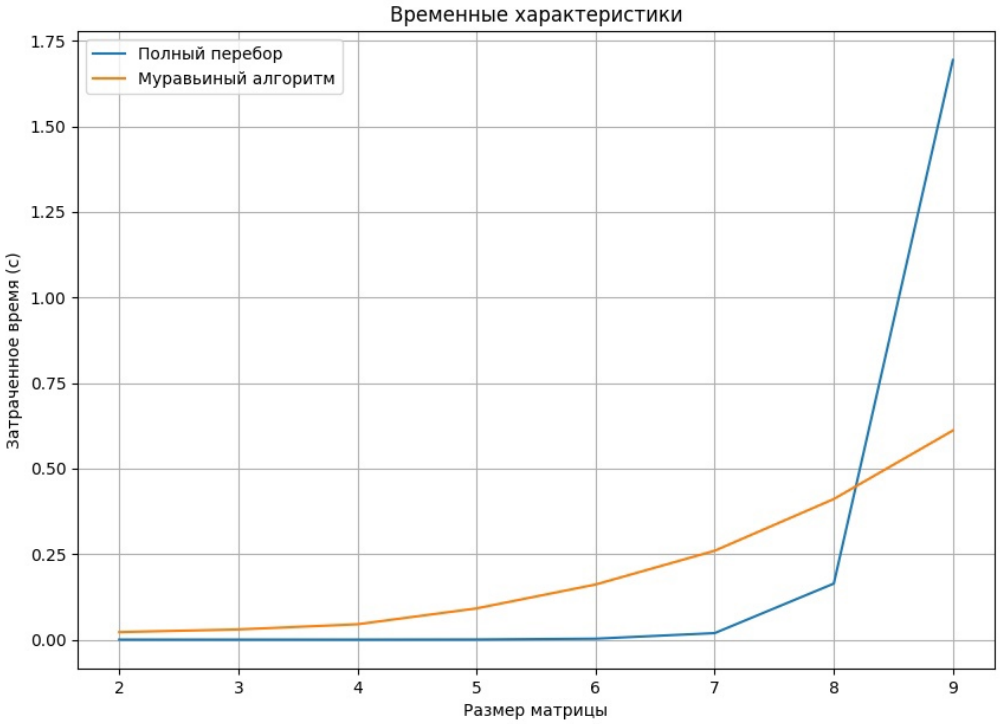

Рисунок 4.2 – Сравнение по времени алгоритмов полного перебора путей и муравьиного на разных размерах матриц

4.4 Постановка эксперимента

Автоматическая параметризация была проведена на двух классах данных — 4.4.1 и 4.4.2. Алгоритм будет запущен для набора значений $\alpha, \rho \in (0, 1)$.

Итоговая таблица значений параметризации будет состоять из следующих колонок:

- α — коэффициент жадности;
- ρ — коэффициент испарения;
- *days* — количество дней жизни колонии муравьёв;
- *Result* — эталонный результат, полученный методом полного перебора для проведения данного эксперимента;
- *Mistake* — разность полученного основаным на муравьином алгоритме методом значения и эталонного значения на данных значениях параметров, показатель качества решения.

Цель эксперимента — определить комбинацию параметров, которые позволяют решаить 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 9 элементов (небольшой разброс значений — от 1 до 2), которая представлена далее.

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

Для данного класса данных приведена таблица 4.2 с выборкой параметров, которые наилучшим образом решают поставленную задачу, полные результаты параметризации приведены в приложении А. Использованы следующие обозначения: Days — количество дней, Result — результат работы, Mistake — ошибка как отклонение решения от эталонного .

Таблица 4.2 – Параметры для класса данных 1

α	ρ	Days	Result	Mistake
0.1	0.3	10	9	0
0.1	0.3	50	9	0
0.1	0.3	100	9	0
0.1	0.3	300	9	0
0.1	0.3	500	9	0
0.1	0.4	10	9	0
0.1	0.4	50	9	0
0.1	0.4	100	9	0
0.1	0.4	300	9	0
0.1	0.4	500	9	0
0.1	0.7	10	9	0
0.1	0.7	50	9	0
0.1	0.7	100	9	0
0.1	0.7	300	9	0
0.1	0.7	500	9	0

0.2	0.5	10	9	0
0.2	0.5	50	9	0
0.2	0.5	100	9	0
0.2	0.5	300	9	0
0.2	0.5	500	9	0
0.2	0.7	10	9	0
0.2	0.7	50	9	0
0.2	0.7	100	9	0
0.2	0.7	300	9	0
0.2	0.7	500	9	0
0.3	0.4	10	9	0
0.3	0.4	50	9	0
0.3	0.4	100	9	0
0.3	0.4	300	9	0
0.3	0.4	500	9	0
0.3	0.5	10	9	0
0.3	0.5	100	9	0
0.3	0.5	300	9	0
0.3	0.5	500	9	0
0.4	0.5	10	9	0
0.4	0.5	50	9	0
0.4	0.5	100	9	0
0.4	0.5	300	9	0
0.4	0.5	500	9	0
0.6	0.1	10	9	0
0.6	0.1	50	9	0
0.6	0.1	100	9	0
0.6	0.1	300	9	0
0.6	0.1	500	9	0

4.4.2 Класс данных 2

Класс данных 2 представляет собой матрицу смежности размером 9 элементов (большой разброс значений - от 1000 до 9999), которая представлена далее.

$$K_1 = \begin{pmatrix} 0 & 9271 & 8511 & 2010 & 1983 & 7296 & 7289 & 3024 & 1011 \\ 9271 & 0 & 7731 & 4865 & 5494 & 6812 & 4755 & 7780 & 7641 \\ 8511 & 7731 & 0 & 1515 & 9297 & 7506 & 5781 & 5804 & 7334 \\ 2010 & 4865 & 1515 & 0 & 3662 & 9597 & 2876 & 8188 & 9227 \\ 1983 & 5494 & 9297 & 3662 & 0 & 8700 & 4754 & 7445 & 3834 \\ 7296 & 6812 & 7506 & 9597 & 8700 & 0 & 4216 & 5553 & 8215 \\ 7289 & 4755 & 5781 & 2876 & 4754 & 4216 & 0 & 4001 & 4715 \\ 3024 & 7780 & 5804 & 8188 & 7445 & 5553 & 4001 & 0 & 9522 \\ 1011 & 7641 & 7334 & 9227 & 3834 & 8215 & 4715 & 9522 & 0 \end{pmatrix} \quad (4.2)$$

Для данного класса данных приведена таблица с выборкой параметров, которые наилучшим образом решают поставленную задачу Days — количество дней, Result — результат работы, Mistake — ошибочность полученного результата).

Таблица 4.3 – Параметры для класса данных 2

α	ρ	Days	Result	Mistake
0.1	0.3	100	34192	0
0.1	0.3	300	34192	0
0.1	0.3	500	34192	0
0.1	0.7	100	34192	0
0.1	0.7	300	34192	0
0.1	0.7	500	34192	0
0.2	0.1	100	34192	0
0.2	0.1	300	34192	0
0.2	0.1	500	34192	0

0.2	0.2	100	34192	0
0.2	0.2	300	34192	0
0.2	0.2	500	34192	0
0.2	0.5	100	34192	0
0.2	0.5	300	34192	0
0.2	0.5	500	34192	0
0.2	0.8	100	34192	0
0.2	0.8	300	34192	0
0.2	0.8	500	34192	0
0.3	0.1	100	34192	0
0.3	0.1	300	34192	0
0.3	0.1	500	34192	0
0.3	0.2	5	34192	0
0.3	0.2	50	34192	0
0.3	0.2	100	34192	0
0.3	0.2	300	34192	0
0.3	0.2	500	34192	0
0.4	0.5	50	34192	0
0.4	0.5	300	34192	0
0.4	0.5	500	34192	0
0.5	0.2	100	34192	0
0.5	0.2	300	34192	0
0.5	0.2	500	34192	0
0.6	0.2	100	34192	0
0.6	0.2	300	34192	0
0.6	0.2	500	34192	0
0.6	0.3	300	34192	0
0.6	0.3	500	34192	0
0.6	0.4	100	34192	0
0.6	0.4	500	34192	0
0.6	0.5	100	34192	0
0.6	0.5	300	34192	0
0.6	0.5	500	34192	0

4.5 Вывод

В результате эксперимента было получно, что использование муравьиного алгоритма наиболее эффективно при больших размерах матриц. Так, при размере матрицы, равном 2, муравьиный алгоритм медленее алгоритма полного перебора в 153 раза, а при размере матрицы, равном 9, муравьиный алгоритм быстрее алгоритма полного перебора в раз, а при размере в 10 – уже в 21 раз. Следовательно, при рамзерах матриц больше 8 следует использовать муравьиный алгоритм, но стоит учитывать, что он не гарантирует получения глобального оптимума при решении задачи.

Также при проведении эксперимента с классами данных было получено, что на первом классе данных (см. п. 4.4.1) муравьиный алгоритм лучше всего показывает себя при параметрах:

— $\alpha = 0.1, \rho = 0.3, 0.4, 0.7$;

— $\alpha = 0.2, \rho = 0.5, 0.7$;

— $\alpha = 0.3, \rho = 0.4, 0.5$;

— $\alpha = 0.4, \rho = 0.5$;

— $\alpha = 0.6, \rho = 0.1$.

Следовательно, для класса данных 1 рекомендуется использовать данные параметры.

Для класса данных 2 (см. п. 4.4.2) было получено, что наилучшим образом алгоритм работает на значениях параметров, которые представлены далее:

— $\alpha = 0.1, \rho = 0.3, 0.7$;

— $\alpha = 0.2, \rho = 0.1, 0.2, 0.5, 0.8$;

— $\alpha = 0.3, \rho = 0.1, 0.2$;

— $\alpha = 0.4, \rho = 0.5$;

— $\alpha = 0.5, \rho = 0.2$;

— $\alpha = 0.6, \rho = 0.2, 0.3, 0.4$.

Для второго класса данных 2 рекомендуется использовать данные параметры.

Также во время исследования было замечено — чем меньше α, тем меньше погрешностей возникает. При этом число дней жизни колонии значительно влияет на качество решения: чем значение параметра $Days$ больше, тем меньше отклонение решения от эталонного.

Заключение

Поставленная цель достигнута: получен навык параметризации методов на примере решения задачи коммивояжёра методом на основе муравьиного алгоритма.

В ходе выполнения лабораторной работы были решены все задачи:

1) описана задача коммивояжёра;
2) описаны методы решения задачи коммивояжёра — метод полного перебора и метод на основе муравьиного алгоритма;
3) приведены схемы муравьиного алгоритма и алгоритма, позволяющего решить задачу коммивояжёра методом полного перебора;
4) описаны используемые типы и структуры данных;
5) описана структура разрабатываемого программного обеспечения;
6) реализованы разработанные алгоритмы;
7) проведено функциональное тестирование разработанного алгоритма;
8) проведен сравнительный анализ выполненных реализаций по затрачиваемому времени;
9) полученные результаты описаны и обоснованы в отчёте о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе..

Исходя из полученных результатов, использование муравьиного алгоритма наиболее эффективно по времени при больших размерах матриц. Так при размере матрицы, равном 2, муравьиный алгоритм медленее алгоритма полного перебора в 153 раза, а при размере матрицы, равном 9, муравьиный алгоритм быстрее алгоритма полного перебора в раз, а при размере в 10 – уже в 21 раз. Следовательно, при размерах матриц больше 8 следует использовать муравьиный алгоритм, но стоит учитывать, что он не гарантирует оптимального решения, в отличие от метода полного перебора.

Даны рекомендации о применимости метода на основании муравьиного алгоритма к решению задачи коммивояжёра.

Список использованных источников

1. С. Шапошникова. Словари — Лаборатория линуксоида [Электронный ресурс]. 2022. URL: Режим доступа: `https://younglinux.info/python/dictionary` (дата обращения: 19.12.2022).

2. Семёнов С. С. Педан А. В. Воловиков В. С. Климов И. С. Анализ трудоёмкости различных алгоритмических подходов для решения задачи коммивояжёра. — ООО «Корпорация «Интел Групп», Системы управления, связи и безопасности [Электронный ресурс], 2022. URL: Режим доступа: `https://cyberleninka.ru/article/n/analiz-trudoemkosti-razlichnyh-algoritmicheskih-podhodov-dlya-\resheniya-zadachi-kommivoyazhera` (дата обращения: 18.11.2022).

3. Welcome to Python [Электронный ресурс]. Режим доступа: `https://www.python.org` (дата обращения: 11.10.2022).

4. time — Time access and conversions [Электронный ресурс]. Режим доступа: `https://docs.python.org/3/library/time.html#functions` (дата обращения: 17.09.2022).

5. macOS Monterey [Электронный ресурс]. Режим доступа: `https://www.apple.com/macos/monterey/` (дата обращения: 11.10.2022).

6. Процессор Intel® Core™ i7 [Электронный ресурс]. Режим доступа: `https://www.intel.com/processors/core/i7/docs` (дата обращения: 17.09.2022).