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 №7 по курсу "Анализ алгоритмов"

Тема Муравьиный алгоритм

Студент Артюхин Н.П.

Группа ИУ7-51Б

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

Москва — 2022 г.

Оглавление

Введение

Целью данной работы является получение навыка параметризации методов на примере решения задачи коммивояжера методом на основе муравьиного алгоритма.

Одной из важных задач является поиск оптимальных маршрутов. Такую задачу можно решать полным перебором, но данное решение является крайне неэффективным по времени (имеют большую трудоемкость) при большом числе вершин (городов) в графе (задача поиска оптимального маршрута представляется в виде графа — набора вершин и ребер). Существуют эвристические методы решения данной задачи, они не не гарантируют нахождение глобального оптимума (в данном случае — кратчайшего маршрута), но они эффективнее по времени (имеют более низкую трудоемкость).

Для достижения поставленной цели требуется решить следующие задачи:

1) изучение задачи коммивояжера и основ муравьиного алгоритма;

2) разработка методов решения задачи коммивояжера — метод полного перебора и метод на основе муравьиного алгоритма;

3) реализация разработанных алгоритмов для решения задачи коммивояжера;

4) выполнение оценки трудоемкости разработанных алгоритмов;

5) выполнение параметризации метода на основе муравьиного алгоритма по трем его параметрам;

6) сравнительный анализ реализации алгоритмов решения задачи коммивояжера по полученным результатам (кратчайшее расстояние).

1 Аналитическая часть

В данном разделе будут описаны задача коммивояжера, а также методы ее решения — метод полного перебора и метод на основе муравьиного алгоритма.

1.1 Задача коммивояжера

Коммивояжер — бродячий торговец. **Задача коммивояжера** — важная задача транспортной логистики, отрасли, занимающейся планированием транспортных перевозок. Коммивояжеру, чтобы распродать нужные и не очень нужные в хозяйстве 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 суммарная стоимость пути, или, в простейшем случае, длина маршрута [1]. Таким образом, задача коммивояжера заключается в том, чтобы найти такой порядок посещения вершин графа, при котором путь будет минимален по стоимости (у каждого ребра графа есть стоимость — расстояние между городами, по варианту — неориентированный граф, то есть в обе стороны одинаковое расстояние), каждая вершина будет посещена лишь один раз, возврат в начальную вершину учитываться не будет, так как по варианту лабораторной работы маршрут — незамкнутый. Данная задача является NP-трудной [2].

1.2 Метод полного перебора для решения задачи коммивояжера

Суть алгоритма полного перебора для решения задачи коммивояжера заключается в переборе всех вариантов путей и нахождении кратчайшего из них. Преимущество данного метода — гарантируется нахождение глобального оптимума (в данном случае – кратчайший путь), недостаток — большая трудоемкость $O(n!)$, где n — число городов [3].

1.3 Метод на основе муравьиного алгоритма для решения задачи коммивояжера

Муравьиный алгоритм [4] — метод решения задачи оптимизации, основаный на моделировании поведения колонии муравьев.

Муравьи действуют, руководствуясь органами чувств. Каждый муравей оставляет на своём пути феромоны, чтобы другие могли ориентироваться. При большом количестве муравьев наибольшее количество феромона остаётся на наиболее посещаемом пути, посещаемость же может быть связана с длинами рёбер (чем короче ребро, тем привлекательнее оно для муравья).

Суть в том, что отдельно взятый муравей мало что может, поскольку он способен выполнять только максимально простые задачи. Но при большом числе других таких муравьев они могут выступать самостоятельными вычислительными единицами. Муравьи используют непрямой обмен информацией через огружающую среду посредством феромона.

Пусть муравей обладает следующими свойствами:

1) зрение — способность определить привлекательность ребра по его длине;

2) обоняние — способность чуять концентрацию феромона;

3) память — способность запомнить пройденный маршрут за текущий день.

Функция, характеризующая привлекательность ребра:

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

где D_{ij} — расстояние от текущего города (вершины графа) i до заданного города j.

Формула вычисления вероятности перехода в заданную точку:

$$p_{k,ij} = \begin{cases} 0, j \in J_k \\ \frac{\eta_{ij}^{\alpha} \cdot \tau_{ij}^{\beta}}{\sum_{q \notin J_k} \eta_{iq}^{\alpha} \cdot \tau_{iq}^{\beta}}, j \notin J_k \end{cases} \tag{1.2}$$

где a — параметр влияния длины пути (коэффициент жадности), b — параметр влияния феромона (коэффициент стадности), τ_{ij} — количество феромонов на ребре ij, η_{ij} — привлекательность ребра ij, J_k — список посещённых за текущий день городов.

После завершения передвижения колонии муравьев (ночью, перед наступлением следующего дня), феромон обновляется по формуле:

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

где $p \in (0,1)$ — коэффициент испарения.

При этом

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

где

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} Q/L_k, \text{ребро посещено муравьем } k \text{ в текущий день } t, \\ 0, \text{иначе} \end{cases} \tag{1.5}$$

Поскольку вероятность перехода в заданную точку 1.2 не должна быть равна нулю, необходимо обеспечить неравенство $\tau_{ij}(t)$ нулю посредством введения дополительного минимально возможного значения феромона τ_{min} и в случае, если $\tau_{ij}(t+1)$ принимает значение, меньшее τ_{min}, откатывать значение феромона до этой величины.

Путь выбирается по следующей схеме.

1. Каждый муравей имеет список запретов — список уже посещенных городов (вершин графа).

2. Муравьиное зрение отвечает за эвристическое желание посетить вершину.

3. Муравьиное обоняние отвечает за ощущение феромона на определенном пути (ребре). При этом количество феромона на пути (ребре) в день t обозначается как $\tau_{i,j}(t)$.

4. После прохождения определенного ребра муравей откладывает на нем некотрое количество феромона, которое показывает оптималь-

ность сделанного выбора, это количество вычисляется по формуле (1.5).

Вывод

В данном разделе были описаны задача коммивояжера, а также методы ее решения — метод полного перебора и метод на основе муравьиного алгоритма.

2 Конструкторская часть

В данном разделе будут представлены схемы алгоритма полного перебора и муравьиного алгоритма решения задачи коммиявояжера, а также проведена оценка трудоемкости алгоритмов.

2.1 Алгоритм полного перебора

На рисунке 2.1 приведена схема алгоритма полного перебора, на рисунке 2.2 — схема алгоритма генерации маршрутов (для перебора всех возможных вариантов) и на рисунке 2.3 — схема алгоритма подсчета длины пути.

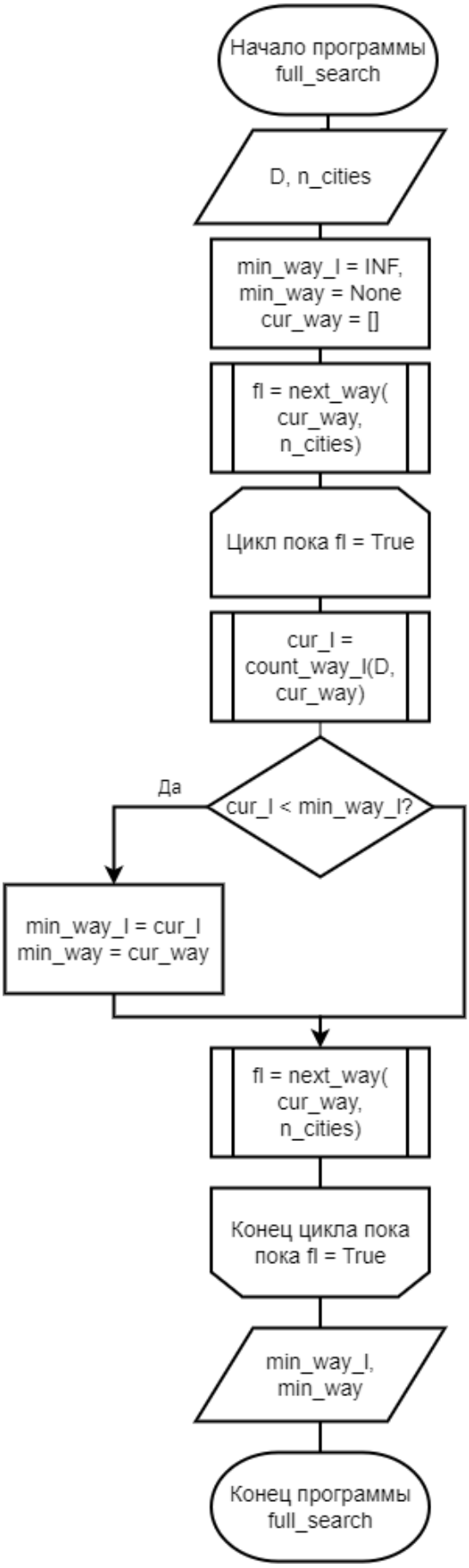

Рисунок 2.1 – Схема алгоритма полного перебора

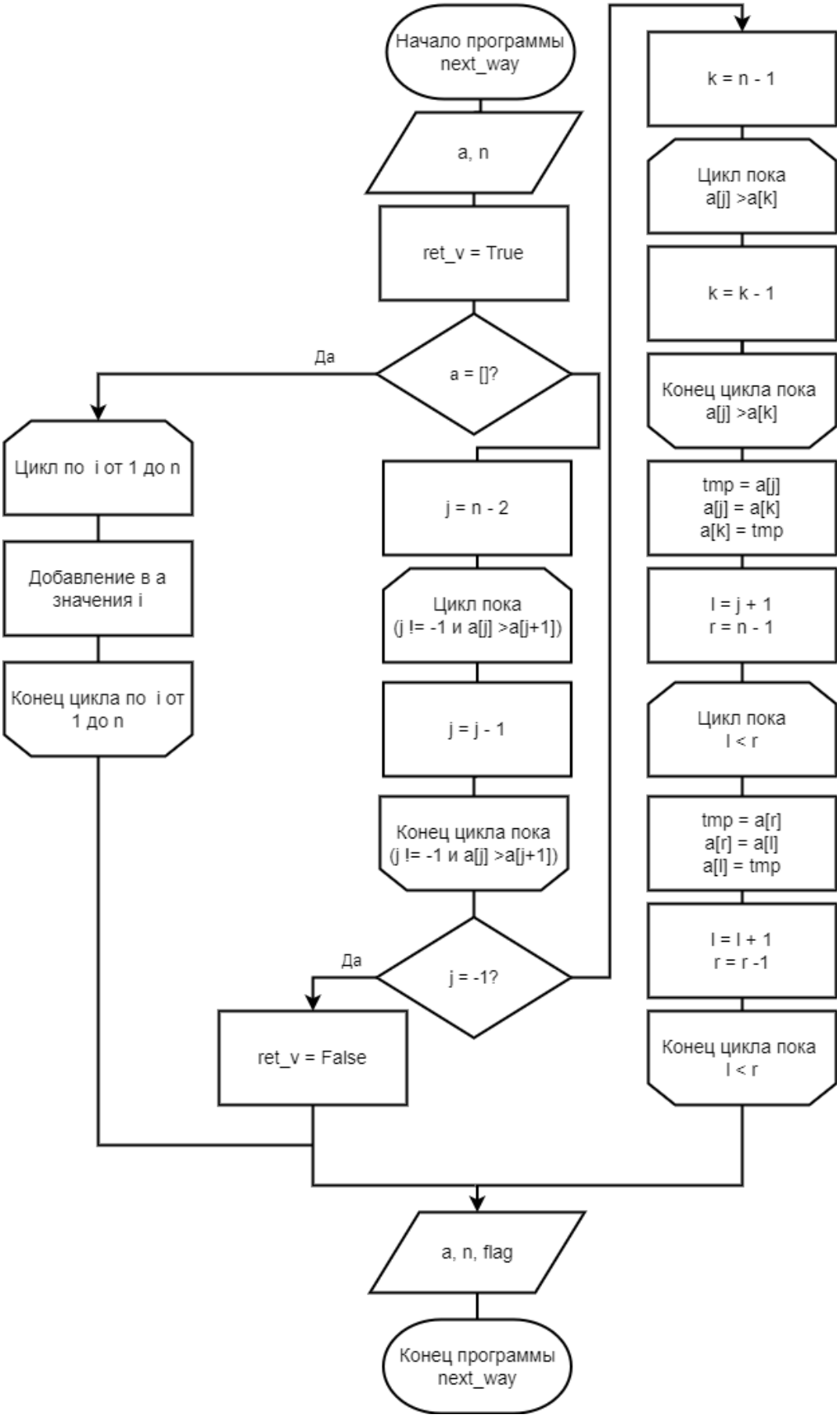

Рисунок 2.2 – Схема алгоритма генерации маршрутов

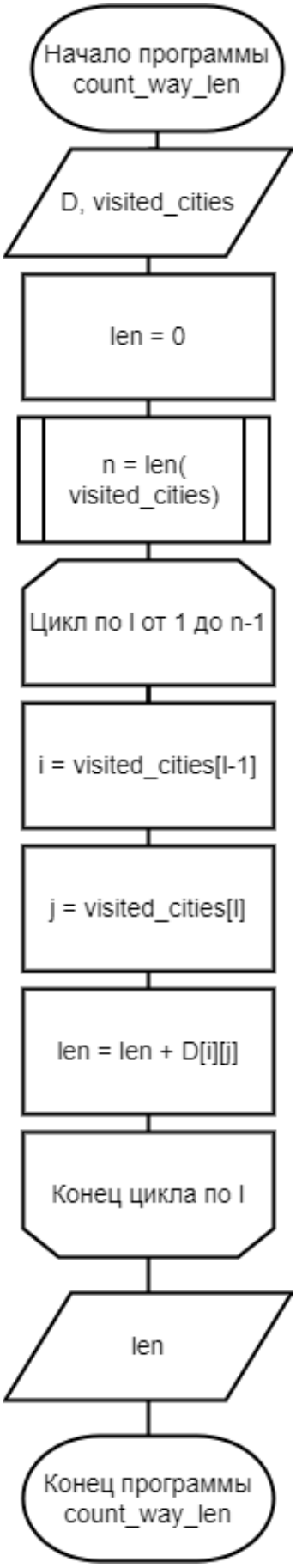

Рисунок 2.3 – Схема алгоритма подсчета длины пути

2.2 Муравьиный алгоритм

На рисунке 2.4 приведена схема муравьиного алгоритма решения задачи коммивояжера.

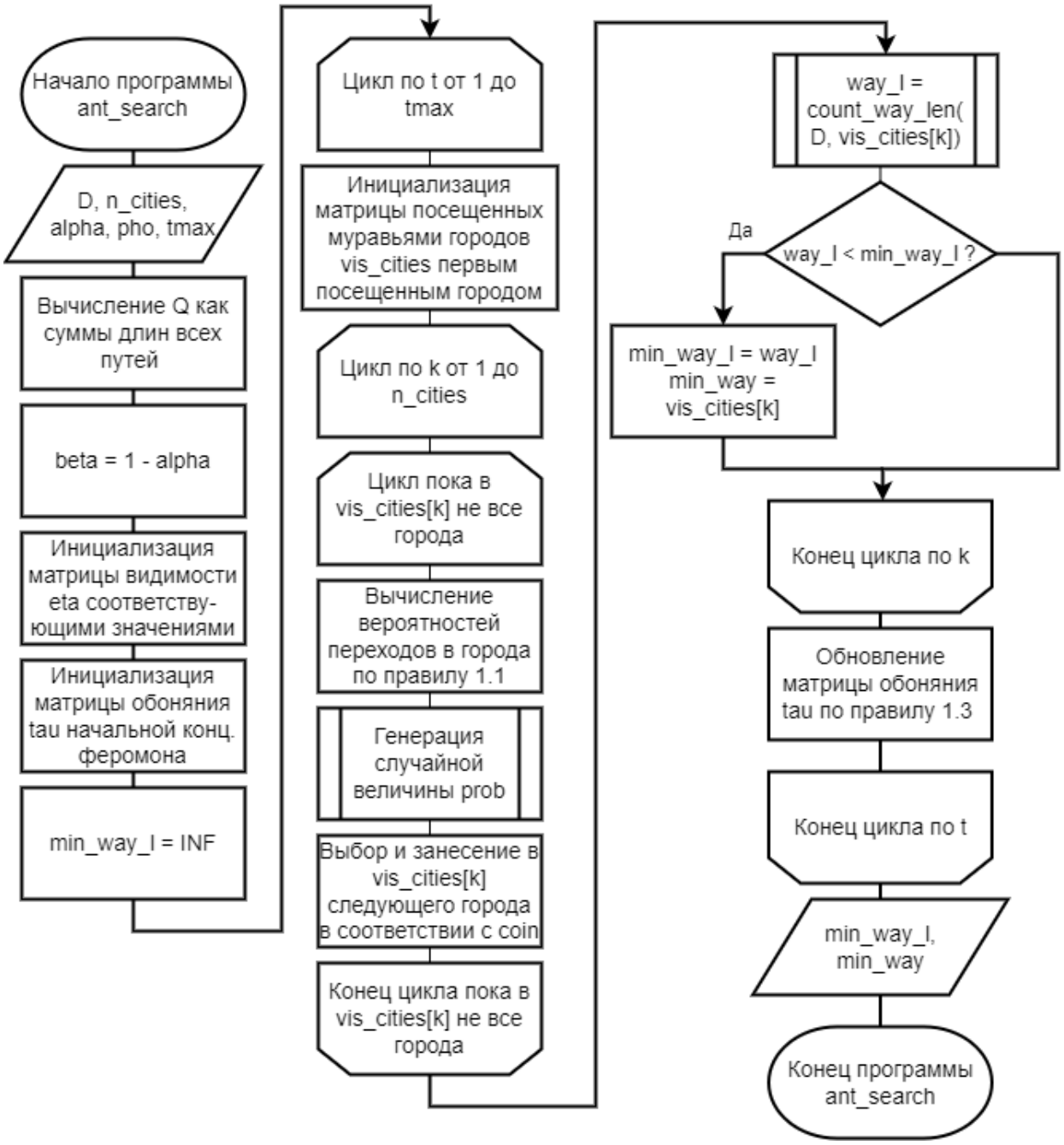

Рисунок 2.4 – Схема алгоритма муравьиного алгоритма решения задачи коммивояжера

2.3 Оценка трудоемкости алгоритмов

Задача коммивояжера является NP-трудной, и точный переборный алгоритм ее решения имеет сложность равную $O(n!)$, где n — число городов. Сложность муравьиного алгоритма равна $O(t_{max} * m * n^2))$, то есть она зависит от времени жизни колонии, количества городов и количества мура-

вьев в колонии [2]. В данной реализации количество муравьев равно количеству городов, и трудоемкость муравьиного алгоритма равна $O(t_{max} * n^3)$.

Вывод

В данном разделе были разработаны схемы алгоритма полного перебора и муравьиного алгоритма решения задачи коммивояжера, а также была проведена оценка трудоемкости алгоритмов.

3 Технологическая часть

В данном разделе будут представлены требования к программному обеспечению, средства реализации, листинги кода и тесты.

3.1 Требования к программному обеспечению

Вход: количество городов (целое положительноне число); симметричная матрица смежности, задающая граф; параметры для муравьиного алгоритма — α (коэффициент жадности, вещественное число от 0 до 1), p (коэффициент испарения, вещественное число от 0 до 1) и t_max (время жизни колонии муравьев, целое положительное число).

Выход: порядок посещенных вершин в кратчайшем пути и его длина для каждого из реализованных алгоритмов, таблица параметризации для муравьиного алгоритма.

3.2 Выбор средств реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран язык программирования Python [5]. Данный язык программирования позволяет реализовать муравьиный алгоритм и имеет необходимые библиотеки для формирования таблиц.

В качестве среды разработки был выбран PyCharm Professional [6]. Данная среда разработки является кросс-платформенной, предоставляет функциональный отладчик, средства для рефакторинга кода и возможность установки необходимых библиотек при необходимости.

3.3 Реализация алгоритмов

В листингах 3.1 – 3.2 представлена реализация алгоритма полного перебора, в листинге 3.3 — реализация муравьиного алгоритма, а в листинге

3.4 — реализация алгоритма подсчета длины пути.

Листинг 3.1 – Реализация алгоритма полного перебора

```
def full_search(matr, n_cities):
    min_way_length = INF
    min_way = None
    cur_way = []

    while next_way(cur_way, n_cities):
        cur_way_lenth = count_way_lenth(matr, cur_way)
        if cur_way_lenth < min_way_length:
            min_way_length = cur_way_lenth
            min_way = cur_way

    return min_way_length, min_way
}
```

Листинг 3.2 – Реализация алгоритма полного перебора (генератор маршрутов)

```
def next_way(a, n):
    if not a:
        for i in range(n):
            a.append(i)
        return True
    j = n - 2
    while j != -1 and a[j] > a[j + 1]:
        j -= 1
    if j == -1:
        return False
    k = n - 1
    while a[j] > a[k]:
        k -= 1
    a[j], a[k] = a[k], a[j]
    l = j + 1
    r = n - 1
    while l < r:
        a[l], a[r] = a[r], a[l]
        l += 1
        r -= 1
    return True
}
```

Листинг 3.3 – Реализация муравьиного алгоритма

```
def ant_search(D, n_cities, alpha=ALPHA, pho=PHO, tmax=TMAX):
    Q = 0
    for i in range(n_cities):
        for j in range(i):
            if D[i][j] < INF:
                Q += D[i][j]
    beta = 1 - alpha

    eta = [[0 for i in range(n_cities)] for j in range(n_cities)]
    tau = [[0 for i in range(n_cities)] for j in range(n_cities)]
    for i in range(n_cities):
        for j in range(i):
            eta[i][j] = 1 / D[i][j]
            eta[j][i] = 1 / D[j][i]
            tau[i][j] = 2 * EPS
            tau[j][i] = 2 * EPS

    min_way_length = INF

    for t in range(tmax):
        vis_cities = [[i] for i in range(n_cities)]
        for k in range(n_cities):
            while len(vis_cities[k]) != n_cities:
                P_ch = [0 for i in range(n_cities)]
                for j in range(n_cities):
                    if j not in vis_cities[k]:
                        i = vis_cities[k][-1]
                        P_ch[j] = (tau[i][j] ** alpha) * (eta[i][j
                            ] ** beta)
                P_zn = sum(P_ch)
                for j in range(n_cities):
                    P_ch[j] /= P_zn

                probability = random()
                summ, j = 0, 0
                while summ < probability:
                    summ += P_ch[j]
                    j += 1
                vis_cities[k].append(j - 1)
            way_length = count_way_lenth(D, vis_cities[k])
```

```
40
41              if way_length < min_way_length:
42                  min_way_length = way_length
43                  min_way = vis_cities[k]
44
45          for i in range(n_cities):
46              for j in range(i):
47                  delta_tau = 0
48                  for k in range(n_cities):
49                      way_length = count_way_lenth(D, vis_cities[k])
50                      for m in range(1, len(vis_cities[k])):
51                          if (vis_cities[k][m], vis_cities[k][m -
                                1]) in ((i, j), (j, i)):
52                              delta_tau += Q / way_length
53                              break
54
55                  tau[i][j] = tau[i][j] * (1 - pho) + delta_tau
56                  if tau[i][j] < EPS:
57                      tau[i][j] = EPS
58                  tau[j][i] = tau[i][j]
59
60      return min_way_length, min_way
```

Листинг 3.4 – Реализация алгоритма подсчета длины пути

```
1 def count_way_lenth(D, vis_cities):
2     length = 0
3
4     for l in range(1, len(vis_cities)):
5         i = vis_cities[l - 1]
6         j = vis_cities[l]
7         length += D[i][j]
8
9     return length
```

3.4 Тестирование

В таблице 3.1 приведены функциональные тесты для алгоритма полного перебора и муравьиного алгоритма. Все тесты были пройдены успешно.

Таблица 3.1 – Тестирование реализаций алгоритмов решения задачи коммивояжера

Матрица смежности	Ожидаемый результат	Результат программы
$\begin{pmatrix} 0 & 3 & 4 & 7 \\ 3 & 0 & 3 & 7 \\ 4 & 3 & 0 & 7 \\ 7 & 7 & 7 & 0 \end{pmatrix}$	13, [0, 1, 2, 3]	13, [0, 1, 2, 3]
$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$	5, [0, 1, 2, 3, 4, 5]	5, [0, 1, 2, 3, 4, 5]
$\begin{pmatrix} 0 & 15 & 19 & 20 \\ 15 & 0 & 12 & 13 \\ 19 & 12 & 0 & 17 \\ 20 & 13 & 17 & 0 \end{pmatrix}$	44, [0, 1, 2, 3]	44, [0, 1, 2, 3]

Вывод

В данном разделе были представлены требования к программному обеспечению и средства реализации, реализованы и протестированы алгоритмы решения задачи коммивояжера.

4 Исследовательская часть

В текущем разделе будут представлены пример работы разработанного программного обеспечения, постановка эксперимента и сравнительный анализ реализованных алгоритмов.

4.1 Пример работы программного обеспечения

На рисунке 4.1 представлен результат работы программы, на вход подавалась матрица смежности графа, состоящего из 4 вершин. Результат был получен для алгоритма полного перебора и муравьиного алгоритма в виде длины кратчайшего пути и порядка посещения вершин. В качестве эталонного результата считается результат алгоритма полного перебора, ошибка показывает отклонение результа муравьиного алгоритма от алгоритма полного перебора.

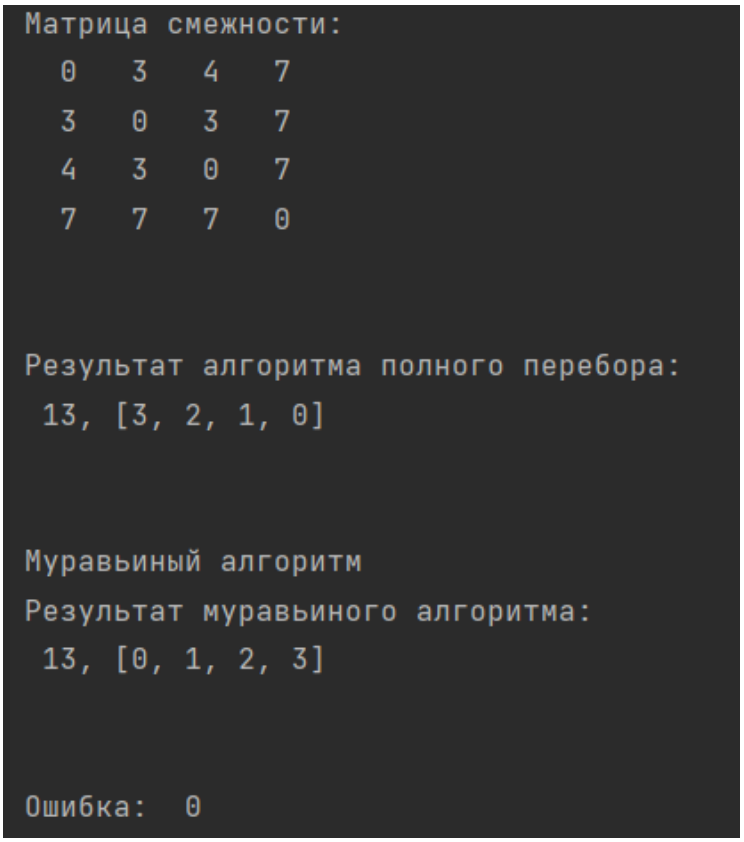

Рисунок 4.1 – Пример работы программы

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялась параметризация муравьиного алгоритма решения задачи коммивояжера:

- операционная система — Windows 10 [7];
- оперативная память — 16 Гб;
- процессор — Intel® Core™ i5 10300H 2.5 ГГц;
- 4 физических ядра, 4 логических ядра.

Во время выполнения параметризации алгоритма ноутбук был включен в сеть питания и нагружен только встроенными приложениями окружения и системой тестирования.

4.3 Постановка эксперимента

Автоматическая параметризация была проведена на трех классах данных — 4.3.1 и 4.3.2. Алгоритм будет запущен для набора значений $\alpha, \rho \in (0.1, 0.25, 0.5, 0.75, 0.9), t_{max} \in (20, 50, 100, 200, 500)$.

Итоговая таблица значений параметризации будет состоять из следующих столбцов:

— α — коэффициент жадности;

— ρ — коэффициент испарения;

— t_{max} — время жизни колонии муравьёв;

— граф 1 — отклонение результата муравьиного алгоритма от эталонного (полученного алгоритмом полного перебора) на 1-ом классе данных;

— граф 2 — отклонение результата муравьиного алгоритма от эталонного (полученного алгоритмом полного перебора) на 2-ом классе данных;

— граф 3 — отклонение результата муравьиного алгоритма от эталонного (полученного алгоритмом полного перебора) на 3-ем классе данных;

— медиана — медианное значение ошибки муравьиного алгоритма для всех трех классов данных.

Цель эксперимента — определить комбинацию параметров, которые позволяют решать задачу коммивояжера наилучшим образом для выбранного класса данных.

Полная таблица параметризации представлена в «Приложении А».

4.3.1 Класс данных 1

Класс данных 1 представляет собой матрицу смежности с линейной размерностью 5 (небольшой разброс значений — от 1 до 2).

При проведении эксперимента с классами данных было получено, что на первом классе данных муравьиный алгоритм лучше всего показывает себя при параметрах:

— $\alpha = 0.1, \rho = 0.1, 0.5, 0.75, 0.9$;

— $\alpha = 0.25, \rho = 0.25, 0.9$;

— $\alpha = 0.5, \rho = 0.1, 0.75, 0.9$;

— $\alpha = 0.75, \rho = 0.5$;

— $\alpha = 0.9, \rho = 0.1, 0.25, 0.9$.

Следовательно, для первого класса данных рекомендуется использовать данные параметры.

4.3.2 Класс данных 2

Класс данных 2 представляет собой матрицу смежности с линейной размерностью 9 (маленький разброс значений — от 9 до 11).

Для класса данных 2 было получено, что наилучшим образом алгоритм работает на значениях параметров, которые представлены далее:

— $\alpha = 0.1, \rho = 0.75$;

— $\alpha = 0.25, \rho = 0.1$;

— $\alpha = 0.5, \rho = 0.1, 0.25, 0.5, 0.9$;

— $\alpha = 0.75, \rho = 0.1, 0.25, 0.5, 0.9$;

— $\alpha = 0.9, \rho = 0.5$.

Следовательно, для второго класса данных рекомендуется использовать данные параметры.

4.3.3 Класс данных 3

Класс данных 3 представляет собой матрицу смежности с линейной размерностью 9 (большой разброс значений — от 0 до 1000).

Для класса данных 3 было получено, что наилучшим образом алгоритм работает на значениях параметров, которые представлены далее:

— $\alpha = 0.1, \rho = 0.1, 0.25, 0.5, 0.75$;

— $\alpha = 0.25, \rho = 0.25$;

— $\alpha = 0.5, \rho = 0.1, 0.5, 0.75$;

— $\alpha = 0.75, \rho = 0.5, 0.75$;

— $\alpha = 0.9, \rho = 0.1, 0.9$.

Следовательно, для третьего класса данных рекомендуется использовать данные параметры.

4.4 Вывод

В данном разделе будут представлены пример работы разработанного программного обеспечения, постановка эксперимента и сравнительный анализ отклонения решений задачи коммивояжера реализованными алгоритмами. В рамках эксперимента были получены рекомендованные параметры для каждого из трех классов данных, также было выяснено, что для всех классов данных время жизни муравьиной колонии (число дней) значительно влияет на точность решения: чем оно больше, тем меньше отклонение результата работы реализации муравьиного алгоритма от эталонного решения, полученного полным перебором.

Заключение

В результате выполнения лабораторной работы цель достигнута: получен навык параметризации методов на примере решения задачи коммивояжера методом на основе муравьиного алгоритма.

В ходе выполнения данной работы были решены все задачи:

1) изучены задача коммивояжера и основы муравьиного алгоритма;
2) разработаны методы решения задачи коммивояжёра — метод полного перебора и метод на основе муравьиного алгоритма;
3) реализованы разработанные алгоритмы для решения задачи коммивояжера;
4) выполнена оценка трудоемкости разработанных алгоритмов;
5) выполнена параметризация метода на основе муравьиного алгоритма по трём его параметрам;
6) проведен сравнительный анализ реализации алгоритмов решения задачи коммивояжера по полученным результатам (кратчайшее расстояние), получены оптимальные параметры для метода на основе муравьиного алгоритма к решению задачи коммивояжера.

Список использованных источников

1. Задача коммивояжёра [Электронный ресурс]. Режим доступа: http://mech.math.msu.su/~shvetz/54/inf/perl-examples/PerlExamples_CommisVoyageur.xhtml (дата обращения: 18.12.2022).

2. Ульянов М. В. РЕСУРСНО-ЭФФЕКТИВНЫЕ КОМПЬЮТЕРНЫЕ АЛГОРИТМЫ. РАЗРАБОТКА И АНАЛИЗ // НАУКА ФИЗМАТЛИТ. 2007. С. 201–205.

3. Алгоритмы решения задачи коммивояжера [Электронный ресурс]. Режим доступа: https://scienceforum.ru/2021/article/2018025171 (дата обращения: 28.10.2021).

4. Штовба С. Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. 2003. С. 70–75.

5. Лутц, Марк. Изучаем Python, том 1, 5-е изд. Пер. с англ. — СПб.: ООО “Диалектика”, 2019 — С. 832.

6. Узнайте все о PyCharm [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/learn/ (дата обращения: 20.09.2022).

7. Windows 10 [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/windows/ (дата обращения: 20.09.2022).

Приложение А

В данное приложение входит таблица параметризации муравьиного алгоритма для решения задачи коммивояжера в виде рисунков.

# 1	alpha	ро	tmax	граф 1	граф 2	граф 3	медиана
0	0.1	0.1	20.0	1.0	4.0	227.0	4.0
1	0.1	0.1	50.0	0.0	2.0	57.0	2.0
2	0.1	0.1	100.0	0.0	2.0	0.0	0.0
3	0.1	0.1	200.0	0.0	1.0	38.0	1.0
4	0.1	0.1	500.0	0.0	1.0	0.0	0.0
5	0.1	0.25	20.0	1.0	3.0	170.0	3.0
6	0.1	0.25	50.0	1.0	2.0	96.0	2.0
7	0.1	0.25	100.0	0.0	2.0	0.0	0.0
8	0.1	0.25	200.0	0.0	2.0	38.0	2.0
9	0.1	0.25	500.0	0.0	2.0	0.0	0.0
10	0.1	0.5	20.0	1.0	2.0	258.0	2.0
11	0.1	0.5	50.0	0.0	2.0	224.0	2.0
12	0.1	0.5	100.0	0.0	2.0	0.0	0.0
13	0.1	0.5	200.0	0.0	2.0	0.0	0.0
14	0.1	0.5	500.0	0.0	1.0	0.0	0.0
15	0.1	0.75	20.0	0.0	2.0	0.0	0.0
16	0.1	0.75	50.0	1.0	0.0	0.0	0.0
17	0.1	0.75	100.0	0.0	1.0	38.0	1.0
18	0.1	0.75	200.0	0.0	2.0	0.0	0.0
19	0.1	0.75	500.0	0.0	1.0	38.0	1.0
20	0.1	0.9	20.0	1.0	3.0	57.0	3.0
21	0.1	0.9	50.0	0.0	2.0	124.0	2.0
22	0.1	0.9	100.0	0.0	1.0	62.0	1.0
23	0.1	0.9	200.0	1.0	1.0	57.0	1.0
24	0.1	0.9	500.0	0.0	1.0	0.0	0.0

# 1	alpha	po	tmax	граф 1	граф 2	граф 3	медиана
25	0.25	0.1	20.0	0.0	3.0	162.0	3.0
26	0.25	0.1	50.0	1.0	1.0	124.0	1.0
27	0.25	0.1	100.0	0.0	1.0	57.0	1.0
28	0.25	0.1	200.0	1.0	0.0	0.0	0.0
29	0.25	0.1	500.0	0.0	1.0	0.0	0.0
30	0.25	0.25	20.0	1.0	3.0	93.0	3.0
31	0.25	0.25	50.0	0.0	1.0	0.0	0.0
32	0.25	0.25	100.0	1.0	2.0	104.0	2.0
33	0.25	0.25	200.0	0.0	1.0	0.0	0.0
34	0.25	0.25	500.0	0.0	1.0	0.0	0.0
35	0.25	0.5	20.0	1.0	2.0	62.0	2.0
36	0.25	0.5	50.0	1.0	3.0	93.0	3.0
37	0.25	0.5	100.0	0.0	1.0	51.0	1.0
38	0.25	0.5	200.0	0.0	1.0	38.0	1.0
39	0.25	0.5	500.0	0.0	1.0	0.0	0.0
40	0.25	0.75	20.0	1.0	3.0	304.0	3.0
41	0.25	0.75	50.0	1.0	2.0	38.0	2.0
42	0.25	0.75	100.0	0.0	2.0	0.0	0.0
43	0.25	0.75	200.0	0.0	2.0	38.0	2.0
44	0.25	0.75	500.0	0.0	2.0	0.0	0.0
45	0.25	0.9	20.0	1.0	1.0	213.0	1.0
46	0.25	0.9	50.0	0.0	3.0	124.0	3.0
47	0.25	0.9	100.0	0.0	2.0	93.0	2.0
48	0.25	0.9	200.0	0.0	2.0	0.0	0.0
49	0.25	0.9	500.0	0.0	1.0	0.0	0.0

# 1	alpha	po	tmax	граф 1	граф 2	граф 3	медиана
50	0.5	0.1	20.0	1.0	3.0	38.0	3.0
51	0.5	0.1	50.0	0.0	3.0	0.0	0.0
52	0.5	0.1	100.0	0.0	2.0	104.0	2.0
53	0.5	0.1	200.0	0.0	1.0	38.0	1.0
54	0.5	0.1	500.0	0.0	0.0	0.0	0.0
55	0.5	0.25	20.0	1.0	4.0	268.0	4.0
56	0.5	0.25	50.0	1.0	2.0	93.0	2.0
57	0.5	0.25	100.0	1.0	2.0	62.0	2.0
58	0.5	0.25	200.0	0.0	0.0	38.0	0.0
59	0.5	0.25	500.0	0.0	1.0	0.0	0.0
60	0.5	0.5	20.0	1.0	3.0	57.0	3.0
61	0.5	0.5	50.0	1.0	2.0	0.0	1.0
62	0.5	0.5	100.0	0.0	2.0	0.0	0.0
63	0.5	0.5	200.0	0.0	0.0	0.0	0.0
64	0.5	0.5	500.0	0.0	2.0	38.0	2.0
65	0.5	0.75	20.0	1.0	2.0	62.0	2.0
66	0.5	0.75	50.0	0.0	2.0	156.0	2.0
67	0.5	0.75	100.0	0.0	2.0	0.0	0.0
68	0.5	0.75	200.0	0.0	2.0	0.0	0.0
69	0.5	0.75	500.0	0.0	1.0	0.0	0.0
70	0.5	0.9	20.0	1.0	2.0	38.0	2.0
71	0.5	0.9	50.0	0.0	2.0	51.0	2.0
72	0.5	0.9	100.0	0.0	2.0	104.0	2.0
73	0.5	0.9	200.0	0.0	2.0	0.0	0.0
74	0.5	0.9	500.0	0.0	0.0	0.0	0.0

# 1	alpha	po	tmax	граф 1	граф 2	граф 3	медиана
75	0.75	0.1	20.0	1.0	2.0	354.0	2.0
76	0.75	0.1	50.0	1.0	1.0	162.0	1.0
77	0.75	0.1	100.0	0.0	2.0	62.0	2.0
78	0.75	0.1	200.0	0.0	0.0	51.0	0.0
79	0.75	0.1	500.0	0.0	1.0	0.0	0.0
80	0.75	0.25	20.0	1.0	2.0	304.0	2.0
81	0.75	0.25	50.0	1.0	1.0	170.0	1.0
82	0.75	0.25	100.0	0.0	1.0	232.0	1.0
83	0.75	0.25	200.0	0.0	2.0	104.0	2.0
84	0.75	0.25	500.0	0.0	0.0	0.0	0.0
85	0.75	0.5	20.0	0.0	1.0	378.0	1.0
86	0.75	0.5	50.0	1.0	3.0	0.0	1.0
87	0.75	0.5	100.0	0.0	2.0	57.0	2.0
88	0.75	0.5	200.0	0.0	2.0	38.0	2.0
89	0.75	0.5	500.0	0.0	0.0	57.0	0.0
90	0.75	0.75	20.0	1.0	2.0	319.0	2.0
91	0.75	0.75	50.0	1.0	2.0	38.0	2.0
92	0.75	0.75	100.0	0.0	2.0	38.0	2.0
93	0.75	0.75	200.0	0.0	2.0	0.0	0.0
94	0.75	0.75	500.0	0.0	1.0	0.0	0.0
95	0.75	0.9	20.0	1.0	3.0	437.0	3.0
96	0.75	0.9	50.0	1.0	2.0	213.0	2.0
97	0.75	0.9	100.0	0.0	3.0	51.0	3.0
98	0.75	0.9	200.0	0.0	0.0	51.0	0.0
99	0.75	0.9	500.0	0.0	0.0	0.0	0.0

# 1	alpha	po	tmax	граф 1	граф 2	граф 3	медиана
100	0.9	0.1	20.0	0.0	3.0	464.0	3.0
101	0.9	0.1	50.0	1.0	2.0	51.0	2.0
102	0.9	0.1	100.0	1.0	1.0	93.0	1.0
103	0.9	0.1	200.0	0.0	1.0	0.0	0.0
104	0.9	0.1	500.0	0.0	1.0	38.0	1.0
105	0.9	0.25	20.0	1.0	2.0	213.0	2.0
106	0.9	0.25	50.0	0.0	2.0	170.0	2.0
107	0.9	0.25	100.0	1.0	2.0	213.0	2.0
108	0.9	0.25	200.0	0.0	2.0	38.0	2.0
109	0.9	0.25	500.0	0.0	1.0	38.0	1.0
110	0.9	0.5	20.0	2.0	3.0	57.0	3.0
111	0.9	0.5	50.0	1.0	3.0	124.0	3.0
112	0.9	0.5	100.0	0.0	3.0	93.0	3.0
113	0.9	0.5	200.0	1.0	0.0	104.0	1.0
114	0.9	0.5	500.0	0.0	2.0	38.0	2.0
115	0.9	0.75	20.0	2.0	2.0	385.0	2.0
116	0.9	0.75	50.0	1.0	3.0	353.0	3.0
117	0.9	0.75	100.0	0.0	3.0	38.0	3.0
118	0.9	0.75	200.0	0.0	1.0	38.0	1.0
119	0.9	0.75	500.0	0.0	1.0	38.0	1.0
120	0.9	0.9	20.0	2.0	4.0	592.0	4.0
121	0.9	0.9	50.0	0.0	1.0	0.0	0.0
122	0.9	0.9	100.0	1.0	2.0	0.0	1.0
123	0.9	0.9	200.0	0.0	1.0	57.0	1.0
124	0.9	0.9	500.0	0.0	2.0	0.0	0.0