**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.03.04 Программная инженерия | |
| **Профиль** | Разработка программно-информационных систем | |
| **Факультет** | КТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | А.А. Лисс |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРА

Тема: Разработка библиотеки для решения NP-трудной задачи «О рюкзаке»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Е.А. Максимов |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н., доцент |  |  | С.А. Беляев |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты |  |  |  | А.М. Шевелева |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  | к.э.н., доцент |  |  | Г.В. Голигузова |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  | к.т.н. |  |  | М.М. Заславский |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | Максимов Е.А. | | | |  | Группа | 0304 | |
| Тема работы: Разработка библиотеки для решения NP-трудной задачи «О рюкзаке» | | | | | | | | |
| Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МО ЭВМ | | | | | | | | |
| Исходные данные (технические требования):  разработать библиотеку на языке программирования Java для решения задачи «О рюкзаке» через сведение к задаче коммивояжёра | | | | | | | | |
| Содержание ВКР: Введение, Обзор и сравнение аналогов, Формулировка требований к решению и постановка задачи, Архитектура программной реализации, Исследование результата, Заключение | | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал | | | | | | | | |
| Дополнительные разделы: Экономическое обоснование ВКР | | | | | | | | |
|  | | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | | |
| «18» марта 2024 г. | | | «18» июня 2024 г. | | | | | |
|  | | |  | | | | | |
| Студент | |  | | Е.А. Максимов | | | |  |
| Руководитель к.т.н., доцент | |  | | С.А. Беляев | | | |  |
| *(Уч. степень, уч. звание)* | |  | |  | | | |  |
| Консультант | |  | | А.М. Шевелева | | | |  |
| *(Уч. степень, уч. звание)* | |  | |  | | | |  |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Максимов Е.А. |  | Группа | 0304 |
| Тема работы: Разработка библиотеки для решения NP-трудной задачи «О рюкзаке» | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор предметной области | 18.03 – 23.03 |
| 2 | Обзор и сравнение аналогов | 24.03 – 6.04 |
| 3 | Формулировка требований к решению и постановка задачи | 7.04 – 13.04 |
| 4 | Архитектура программной реализации | 14.04 – 27.04 |
| 5 | Исследование результата | 28.04 – 4.05 |
| 6 | Экономическое обоснование ВКР | 5.05 – 7.05 |
| 7 | Оформление пояснительной записки | 8.05 – 16.05 |
| 8 | Оформление иллюстративного материала | 17.05 – 10.06 |
| 9 | Предзащита | 11.06 |

|  |  |  |
| --- | --- | --- |
| Студент |  | Е.А. Максимов |
| Руководитель к.т.н., доцент |  | С.А. Беляев |
| *(Уч. степень, уч. звание)* |  |  |
| Консультант |  | А.М. Шевелева |
| *(Уч. степень, уч. звание)* |  |  |

**РЕФЕРАТ**

Пояснительная записка 66 стр., 7 рис., 9 табл., 21 ист., 2 прил.

ЗАДАЧА О РЮКЗАКЕ, ЗАДАЧА КОММИВОЯЖЁРА, ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ, АЛГОРИТМ СВЕДЕНИЯ, БИБЛИОТЕКА, NP-ТРУДНАЯ ЗАДАЧА.

**Объектом исследования** является алгоритм решения задачи о рюкзаке.

**Предметом исследования** является эффективность алгоритма решения задачи о рюкзаке через сведение к задаче коммивояжёра.

**Цель работы:** разработать библиотеку на языке программирования Java для решения задачи «О рюкзаке» через сведение к задаче коммивояжёра.

В работе были проанализированы существующие точные и эвристические алгоритмы решения задачи о рюкзаке; описана и реализована математическая модель алгоритмов решения задачи о рюкзаке; разработаны, реализованы и протестированы на корректность алгоритмы сведения задачи о рюкзаке к задаче коммивояжёра. Была проанализирована работа математической модели на наборах с различной грузоподъёмностью, различным количеством предметов случайного веса и стоимости. Была реализована библиотека на языке программирования Java, включающая в себя разработанные математические модели.

**ABSTRACT**

This paper analyzes existing exact and heuristic algorithms for solving the knapsack problem; a mathematical model of algorithms for solving the knapsack problem is described and implemented; algorithms for reducing the knapsack problem to the traveling salesman problem are designed, implemented and tested for correctness. The performance of the mathematical model on sets with different numbers of random weight items and backpack capacity was analyzed. A library in Java programming language including the developed mathematical models was implemented.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 8](#_Toc23742)

[1 ОБЗОР И СРАВНЕНИЕ АНАЛОГОВ 10](#_Toc12511)

[1.1 Основные определения 10](#_Toc13225)

[1.2 Обзор алгоритмов и методов решения 10](#_Toc26036)

[1.3 Критерии сравнения алгоритмов 14](#_Toc14476)

[1.4 Выводы по итогам обзора алгоритмов 15](#_Toc31886)

[1.5 Обзор библиотек для решения задачи 16](#_Toc21657)

[1.6 Критерии сравнения библиотек 17](#_Toc4918)

[1.7 Выводы по итогам обзора библиотек 18](#_Toc10831)

[2 ФОРМУЛИРОВКА ТРЕБОВАНИЙ К РЕШЕНИЮ И ПОСТАНОВКА ЗАДАЧИ 20](#_Toc13988)

[2.1 Постановка задачи 20](#_Toc32116)

[2.2 Алгоритм сведения задачи о рюкзаке к задаче коммивояжёра 20](#_Toc903)

[2.3 Алгоритм сведения решения задачи коммивояжёра к задаче о рюкзаке 21](#_Toc18420)

[2.4 Обзор методов решения задачи коммивояжёра 22](#_Toc1073)

[2.5 Выбор методов решения задачи о рюкзаке и задачи коммивояжёра 25](#_Toc19106)

[3 АРХИТЕКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ 27](#_Toc32450)

[3.1 Используемые технологии 27](#_Toc8504)

[3.2 Структура программной реализации 27](#_Toc15813)

[3.2.1 Классы моделей задач 27](#_Toc23296)

[3.2.2 Классы алгоритмов 31](#_Toc28318)

[3.2.3 Вспомогательные классы 37](#_Toc26118)

[3.2.4 Класс ввода-вывода данных 37](#_Toc19576)

[3.2.5 Класс интерфейса библиотеки 38](#_Toc18645)

[3.3 Взаимодействие элементов программной реализации 39](#_Toc17301)

[3.4 Структура тестирования 40](#_Toc18176)

[1 ИССЛЕДОВАНИЕ РЕЗУЛЬТАТА 42](#_Toc22573)

[4.1 Исследование программного решения 42](#_Toc12039)

[4.1.1 Тестирование на корректность 42](#_Toc13637)

[4.1.2 Сравнение программного решения с аналогами 42](#_Toc17500)

[4.2 Исследование полученного метода решения задачи о рюкзаке 43](#_Toc11032)

[4.2.1 Оценка вычислительной сложности 44](#_Toc3727)

[4.2.2 Измерение зависимости вычислительной сложности от количества предметов 45](#_Toc25110)

[4.2.3 Измерение зависимости вычислительной сложности от грузоподъёмности рюкзака 48](#_Toc24305)

[4.2.4 Сравнение вычислительной сложности полученного метода решения с аналогами 51](#_Toc28286)

[4.2.5 Сравнение времени вычисления полученного метода с аналогами 52](#_Toc10282)

[4.3 Общие выводы по исследованию 53](#_Toc4637)

[5 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ВКР 54](#_Toc24465)

[5.1 Концепция экономического обоснования разработки научно-технического продукта 54](#_Toc9404)

[5.2 Составление плана-графика работ 54](#_Toc7761)

[5.3 Расчёт расходов на оплату труда 55](#_Toc8512)

[5.4 Расчёт отчислений на социальные нужды 56](#_Toc10394)

[5.5 Расчёт расходов на материалы 57](#_Toc1163)

[5.6 Расчёт затрат по работам, выполняемым сторонними организациями 58](#_Toc29548)

[5.7 Спецоборудование 58](#_Toc21659)

[5.8 Расчёт амортизационных отчислений 58](#_Toc17864)

[5.9 Расчёт накладных расходов 59](#_Toc6530)

[5.10 Итоговые затраты проведения работ 60](#_Toc14403)

[ЗАКЛЮЧЕНИЕ 62](#_Toc7971)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 64](#_Toc11853)

[ПРИЛОЖЕНИЕ А 67](#_Toc19933)

[ПРИЛОЖЕНИЕ Б 71](#_Toc32437)

## ВВЕДЕНИЕ

Решения NP-трудных задач оптимизации играют ключевую роль во многих областях практической деятельности. Алгоритмы, способные эффективно решать задачу о рюкзаке, могут быть применены в маркетинге с целью увеличения прибыли компании [1]. Алгоритмы решения задачи о рюкзаке могут быть использованы в вычислительной технике для создания планировщика задач и алгоритмов оптимизации используемых ресурсов устройства [2]. Также задача о рюкзаке является частным случаем задачи о контейнерах, поэтому это также может быть использовано в решении.

Реальные задачи дискретной оптимизации бывают довольно сложны, и существующие методы оптимизации не всегда могут обеспечить их решение этих задач в разумные сроки. Быстрое развитие технологий повышает требования к быстродействию используемых информационных систем.

Поэтому новые алгоритмы для решения данных задач, способных демонстрировать высокую эффективность на различных наборах данных, позволит значительно оптимизировать существующие процессы и системы.

Во второй главе будут рассматриваться задачи и подходы к нахождению оценки длины пути. В третьей главе будет проводиться исследование работы подхода на различных NP-полных задачах, сведённых к задаче коммивояжёра, и на матрицах со случайными значениями.

В первой главе работы будут даны основные определения, использующиеся в задаче о рюкзаке, представлены известные алгоритмы решения задачи коммивояжёра и существующие библиотеки и пакеты в публичном доступе, специализирующихся на решении задачи о рюкзаке. Данные алгоритмы будут сравниваться по критериям вычислительной сложности, точности и применимости. Данные программные решения будут сравниваться по критериям обеспечения корректной работы путём валидации входных параметров, количеству реализованных способов решения задачи о рюкзаке, наличию инструментов для импорта и экспорта данных.

Во второй главе будет рассматриваться алгоритм сведения задачи о рюкзаке к задаче коммивояжёра. Будут представлены методы решения задачи коммивояжёра, и на основе выбранных методов решения задачи о рюкзаке будут выбраны соответствующие методы решения задачи коммивояжёра.

В третьей главе будет рассматриваться архитектура программной реализации библиотеки. В четвёртой главе будет проводиться исследование корректной работоспособности реализованного программного решения на основе юнит-тестирования. Также в четвёртой главе будет проводиться исследование вычислительной сложности полученного метода решения рюкзаке на основе измерений и анализа полученных данных.

Целью данной работы является разработка библиотеки для решения задачи NP-трудной задачи о рюкзаке.

Задачи данной работы:

Провести обзор существующих алгоритмов и методов решения задачи о рюкзаке.

Определить критерии сравнения эффективности алгоритмов и методов.

Провести обзор существующих библиотек, специализирующихся на решении задачи о рюкзаке.

Определить критерии сравнения библиотек, специализирующихся на решении задачи о рюкзаке.

Реализовать библиотеку для решения задачи о рюкзаке методом сведения к задаче коммивояжёра.

Исследовать полученное программное решение.

**Объектом исследования** является алгоритм решения задачи о рюкзаке.

**Предметом исследования** является эффективность решения задачи о рюкзаке.

**Практическая ценность данной работы** состоит в исследовании новых методов решения задачи о рюкзаке и в расширении доступного разнообразия библиотек для решения этой задачи.

## 1 ОБЗОР И СРАВНЕНИЕ АНАЛОГОВ

### 1.1 Основные определения

*Определение 1.* Класс NP – класс [задач](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8" \o "Задача разрешимости), решение которых возможно проверить на [машине Тьюринга](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0" \o "Детерминированная машина Тьюринга) за время, не превосходящее значения некоторого [многочлена](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD" \o "Многочлен) от размера входных данных, при наличии некоторых дополнительных сведений.

*Определение 2.* Задача является NP-трудной, если любая задача из NP может быть сведена к за время, не превосходящее значения некоторого [многочлена](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD" \o "Многочлен) от размера входных данных.

*Определение 3.* Задача о рюкзаке.

Пусть имеется набор объектов, у которых есть определённые положительные масса и стоимость, и рюкзак определённой положительной грузоподъёмности. Необходимо выбрать объекты из набора таким образом, чтобы максимизировать суммарную стоимость, но не превышая грузоподъёмность рюкзака.

Математическая формулировка задачи о рюкзаке приведена ниже.

Имеется предметов. Для каждого предмета определены его масса и ценность Также имеется максимальная грузоподъёмность . Необходимо максимизировать (1)

|  |  |
| --- | --- |
|  | (1) |

со следующими ограничениями (2, 3)

|  |  |
| --- | --- |
|  | (2) |
|  | (3) |

### 1.2 Обзор алгоритмов и методов решения

В качестве аналогов рассматривались алгоритмы и методы решения задачи о рюкзаке. Отбор аналогов производился посредством использования системы Google Scholar, электронных библиотек eLibrary и КиберЛенинка. Для поиска аналогов использовались следующие поисковые запросы: «задача о рюкзаке», «knapsack problem».

*Метод полного перебора*

Естественный способ решения подобного рода задач. Относится к классу методов поиска решения исчерпыванием всевозможных вариантов. Данный метод гарантирует оптимальное решение задачи, однако высокие вычислительные затраты не позволяют широко использовать его для решения прикладных задач, поэтому эффективен лишь на сравнительно малом объёме входных данных.

*Метод ветвей и границ*

В работе Kolesar P. J. приводится алгоритм метода ветвей и границ [3]. Все предметы отсортировываются в порядке убывания удельной ценности. Затем рекурсивно строится дерево полного перебора, отдавая приоритет предметам с большей удельной стоимостью. Если в узле дерева на текущем шаге вес упакованных предметов превышает грузоподъёмность рюкзака, то этот узел пропускается и предметы более не добавляются.

Представляет собой разновидность метода полного перебора, имеет такую же вычислительную сложность, но на практике работает гораздо быстрее за счёт игнорирования ветвей, которые гарантированно не содержат оптимальное решение.

*Жадный алгоритм*

Алгоритм приближенного и сравнительно быстрого решения задачи о рюкзаке. Все предметы отсортировываются в порядке убывания удельной ценности, после этого последовательно упаковываются в рюкзак, пропуская предметы при нехватке места. В работе Al Etawi N. A., Aburomman F. T. авторы оценивают сложность жадного алгоритма решения задачи о рюкзаке как линейно-логарифмическую, минимальную точность в 50% в случае неограниченной задачи [4].

*Метод динамического программирования*

В работе М.А. Куприяшина, Г.И. Борзунова авторы приводят алгоритм точного решения задачи о рюкзаке методом динамического программирования [5].

Пусть – матрица стоимости укладок размерностью Столбцы отображают вес упакованных предметов. По строкам записываются частные решения упаковки рюкзака для первых предметов из заданного списка. В ячейках матрицы записывается стоимость упакованных в рюкзак предметов, общий вес которых не превышает . Решение состоит из 2 этапов: построение матрицы стоимости укладок и восстановление предметов.

1. Построение матрицы стоимости укладок.

Ячейки матрицы инициализируется (4, 5)

|  |  |
| --- | --- |
|  | (4) |
|  | (5) |

Далее матрица, начиная с ячейки , заполняется последовательно слева направо и сверху вниз. Для каждого проверяется следующее условие для нового предмета :

а) если вес нового предмета , то данный элемент матрицы приравнивается к элементу того же столбца и предыдущей строки – предмет не влезает в рюкзак;

б) если вес нового предмета , то при помещении его в рюкзак, остаётся доступной грузоподъёмности. Возможно, стоимость набора предметов, составленного на рассматриваемом шаге окажется меньше, чем на предыдущем без учёта текущего предмета Тогда необходимо определить наиболее выгодный набор предметов: текущий и все предметы из шага, который соответствует ячейке или оставить предыдущий результат . Данный алгоритм построения матрицы стоимости укладок можно перезаписать в упрощённой форме (6)

|  |  |
| --- | --- |
|  | (6) |

1. Восстановление предметов.

Необходимо обойти матрицу стоимости укладок начиная с последней ячейки , сравнивая текущее значение со значением для того же максимального веса, но с меньшим числом предметов. Для каждого , проверяется следующее условие для предмета :

а) если выполняется равенство (7)

|  |  |
| --- | --- |
|  | (7) |

то рюкзак пуст, и все предметы с номером, меньшим или равным не входят в решение;

б) если выполняется равенство (8)

|  |  |
| --- | --- |
|  | (8) |

то предмет не влияет на итоговый результат; этот предмет не входит в решение, необходимо пропустить этот предмет и перейти к следующему предмету , то есть к ячейке матрицы ;

в) если равенство (8) не выполняется, то предмет влияет на итоговый результат, поэтому входит в решение задачи. В таком случае необходимо уменьшить текущий вес на величину веса предмета и перейти к следующему предмету , то есть к ячейке матрицы .

Данный метод решения является точным, его сложность является псевдополиномиальной и зависит от исходного количества предметов и грузоподъёмности рюкзака.

*Генетический алгоритм*

В статье Yadav V., Singh S. авторы приводят описание метода решения задачи о рюкзаке при помощи генетического алгоритма [6]. Основные шаги алгоритма: вычисление функции приспособления для текущего поколения популяции, отбор кандидатов по значению приспособления, генерация следующего поколения популяции, мутация. В качестве вектора генов выступает массив из нулей и единиц, который соответствует решению задачи о рюкзаке.

Авторы отмечают, что несмотря на полиномиальную сложность шагов алгоритма, время работы не ограничено сверху и процесс выполнения может выполняться сколь угодно долго. Для завершения выполнения вводятся критерии завершения: исчерпание ресурсов времени или памяти, построение функции оценки решения, устойчивое состоянии популяции.

### 1.3 Критерии сравнения алгоритмов

*Вычислительная сложность*

Данный критерий оценивает зависимость объёма вычислений, который потребуется на решение задачи, от размера входных данных (в данном случае — от количества предметов).

*Точность решения*

Данный критерий показывает, насколько решение в худшем случае приближается к истинному значению. Оценивается как отношение минимальной стоимости полученного набора к стоимости оптимального набора.

*Тип задачи*

Данный критерий определяет типы задачи для которых предназначен алгоритм и на которых алгоритм демонстрирует точность выше по сравнению с другими типами. Задача коммивояжёра делится по типам, перечисленные ниже.

1. Классическая (0-1) – предмет либо есть в рюкзаке, либо его нет.
2. Ограниченная (0-*N*) – в рюкзак можно поместить до копий предмета.
3. Неограниченная – в рюкзак можно поместить любое количество копий предмета.

В таблице 1 указаны все вышеперечисленные методы и их сравнительная характеристика. Для вычислительной сложности методов, – количество предметов, – грузоподъёмность рюкзака.

Таблица 1 – Сравнение методов решения задачи о рюкзаке

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Вычислительная сложность | Точность решения | Тип задачи |
| Метод полного перебора |  | точный | классическая, ограниченная |
| Метод ветвей и границ |  | точный | классическая, ограниченная, неограниченная |
| Жадный алгоритм |  | 50 % | неограниченная |
| Метод динамического программирования |  | точный | классическая, ограниченная |
| Генетический алгоритм | зависит от времени выполнения | зависит от времени выполнения | классическая, ограниченная |

### 1.4 Выводы по итогам обзора алгоритмов

Для задач с малым количеством предметов или когда требуется абсолютная точность, предпочтительнее использовать методы полного перебора или ветвей и границ. Этот метод также применим к другим типам задач, включая неограниченный вариант.

Когда время имеет решающее значение или задача имеет большой размер, жадный алгоритм может быть лучшим выбором, несмотря на потерю точности.

Метод динамического программирования является лучшим среди представленных, обеспечивая точность и приемлемую скорость для задач среднего размера. Он особенно эффективен, когда вместимость рюкзака и количество предметов не слишком велики.

### 1.5 Обзор библиотек для решения задачи

В качестве аналогов рассматривались библиотеки, пакеты и прочие импортируемые модули с открытым исходным кодом, реализующие решение задачи о рюкзаке. Отбор аналогов производился посредством использования поисковых систем Google и Yandex. Для поиска аналогов использовались следующие поисковые запросы: «задача о рюкзаке библиотека», «задача о рюкзаке пакет», «knapsack problem library», «knapsack problem package».

*OR-Tools*

Популярный пакет программного обеспечения с открытым исходным кодом для решения различных NP-трудных задач оптимизации и программирования в ограничениях [7]. Реализована на языке программирования C++, однако имеет готовые интерфейсы для многих других языков программирования.

*go-knapsack*

Представляет собой пакет для решения задачи о рюкзаке методом динамического программирования [8]. Реализован на языке программирования Go и решает исключительно классическую задачу о рюкзаке.

*mknapsack*

Библиотека для языка программирования Python, включающая в алгоритмы решения задачи о рюкзаке и связанных с ней задача о контейнерах, многомерная задача о рюкзаке и другие [9]. Большинство алгоритмов реализовано с применением языка программирования Fortran, что значительно ускоряет выполнение алгоритмов.

*knapsacksolver*

Библиотека для языка программирования C++, реализующая решение классической задачи о рюкзаке и задачу о сумме подмножеств с множественным выбором [10]. В документации библиотеки представлено подробное описание методов для вызова с примерами.

*knapsack-pip*

Пакет для языка программирования Python, в которой представлено единственный метод решения задачу о рюкзаке – метод ветвей и границ [11]. Отличается сравнительной компактностью своего исходного кода.

### 1.6 Критерии сравнения библиотек

*Языки программирования*

Данный критерий отображает языки программирования, для которых данный программный продукт применим.

*Количество методов решения*

Данный критерий отображает количество различных методов, при помощи которых пользователь библиотеки может решить классическую задачу о рюкзаке.

*Валидация входных параметров*

Данный бинарный критерий отображает наличие проверки входных параметров для реализованных алгоритмов. В случае некорректного ввода пользователь должен быть оповещён сообщением, которое содержит описание и причину возникновения ошибки.

*Импорт и экспорт данных*

Данный бинарный критерий отображает наличие инструментов в библиотеке, реализующих импорт условия задачи и экспорт полученного решения в отдельный файл.

В таблице 2 указаны все вышеперечисленные библиотеки и их сравнительная характеристика.

Таблица 2 – Сравнение библиотек для решения задачи о рюкзаке

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Языки программирования | Количество методов решения | Валидация входных параметров | Импорт и экспорт данных |
| OR-Tools | C++, C#,  Java, Python | 1 | Да | Нет |
| go-knapsack | Go | 1 | Нет | Нет |
| mknapsack | Python | 2 | Да | Нет |
| knapsacksolver | C++ | 4 | Да | Да |
| knapsack-pip | Python | 1 | Нет | Нет |

### 1.7 Выводы по итогам обзора библиотек

Если важным критерием выбора является библиотека с наибольшим количеством методов решения и поддержкой импорта/экспорта данных, то knapsacksolver на языке программирования C++ может быть лучшим выбором.

Если важна простота использования и возможность работы с языком программирования Python, то библиотеки OR-Tools или mknapsack могут также быть рассмотрены.

Для объектно-ориентированных языков программирования C# и Java разнообразие библиотек по решению задачи о рюкзаке слишком мало.

## 2 ФОРМУЛИРОВКА ТРЕБОВАНИЙ К РЕШЕНИЮ И ПОСТАНОВКА ЗАДАЧИ

### 2.1 Постановка задачи

Необходимо разработать библиотеку на языке программирования Java, реализующую решение задачи о рюкзаке методом сведения к задаче коммивояжёра. Необходимо происследовать реализованный метод решения путём измерения времени работы на различных наборах данных и сравнить полученные результаты с аналогами.

### 2.2 Алгоритм сведения задачи о рюкзаке к задаче коммивояжёра

В работе Беляева С.А., Шевелевой А.М. приводятся алгоритмы сведения задачи о рюкзаке к задаче коммивояжёра, и алгоритм обратного сведения [12].

Пусть – количество предметов, – грузоподъёмность рюкзака, – вес предмета, - стоимость предмета, . Алгоритм сведения задачи описан ниже.

1. Если минимальный вес предмета среди множества заданных предметов не равен 1, то вес и стоимость каждого предмета пересчитывается по формулам (9, 10) соответственно.

|  |  |
| --- | --- |
|  | (9) |
|  | (10) |

1. Построить вершин графа, пронумерованных от 0 до
2. От каждой пронумерованной вершины провести дуги () с весом (11)

|  |  |
| --- | --- |
|  | (11) |

для каждого из предметов , если выполняется неравенство (12)

|  |  |
| --- | --- |
|  | (12) |

1. Построить дополнительно вершин, пронумерованных от до .
2. От каждой пронумерованной вершины провести дуги с весом 0. Провести дугу весом 0, соединяющую последнюю и первую вершину графа.
3. От каждой пронумерованной вершины провести дуги с весом 0:
4. От каждой пронумерованной вершины провести дуги с весом 0. Провести дугу весом 0.

### 2.3 Алгоритм сведения решения задачи коммивояжёра к задаче о рюкзаке

Пусть – весовая матрица графа задачи коммивояжёра размерности , где вес отсутствующих рёбер равен бесконечности, – сумма весов всех не бесконечных рёбер, – решение задачи коммивояжёра, представленное в виде матрицы размерности со значениями 0 и 1, – список предметов задачи о рюкзаке, содержащий номера вершин оптимального маршрута графа коммивояжёра, вес и стоимость (в виде матрицы размерности ). Алгоритм сведения решения представлен ниже.

1. Для каждого , если элемент весовой матрицы не равен бесконечности, то в список добавляется предмет, вес которого 1, стоимость предмета равна (13)

|  |  |
| --- | --- |
|  | (13) |

2. Если индексы предмета из списка соответствуют ребру оптимального пути матрицы решения задачи коммивояжёра , то этот предмет добавляется в рюкзак.

### 2.4 Обзор методов решения задачи коммивояжёра

Поскольку решение сведённой задачи включает в себя алгоритм решения задачи коммивояжёра, то для реализации данного метода возникает необходимость рассмотреть методы решения задачи коммивояжёра.

*Метод полного перебора*

Естественный способ решения подобного рода задач. Относится к классу методов поиска решения исчерпыванием всевозможных вариантов. Данный метод гарантирует оптимальное решение задачи, однако высокие вычислительные затраты не позволяют широко использовать его для решения прикладных задач, поэтому эффективен лишь на сравнительно малом объёме входных данных [13].

*Модифицированный алгоритм Литтла*

В статье Костюка Ю.Л. “Эффективная реализация алгоритма решения задачи коммивояжёра методом ветвей и границ” [14] автор предлагает решение на основе метода ветвей и границ Литтла [15]. Модификация основана на следующем факте: если из всех элементов строки или столбца весовой матрицы задачи коммивояжёра вычесть одно и то же число, то все маршруты сократятся на это число. Построении дерева поиска производится отсечение частично построенных маршрутов, у которых оценка снизу длины маршрута больше или равна длине ранее построенного полного наилучшего маршрута.

Алгоритм инициализируется списком решений, который содержит решение с исходной матрицей смежности, с пустым списком рёбер маршрута и с базовым весом маршрута, равным 0. На каждом шаге для решения с наименьшим базовым весом маршрута выполняет два основных действия: редукция матрицы смежности и ветвление.

1. Редукция матрицы смежности.

В каждой строке матрицы выбирается наименьшее число; из каждого элемента строки вычитается это число, и затем добавляется к базовому весу маршрута. Затем производятся те же действия со столбцами матрицы.

а) Если после этого шага базовая стоимость маршрута данного решения больше, чем у текущего лучшего решения, то решение исключается и далее не рассматривается.

б) Если после этого шага количество рёбер маршрута решения равно количеству вершин, то производится сравнение с текущим лучшим решением. Если базовый вес маршрута меньше текущего лучшего результата, то лучшее решение заменяется, иначе исключается.

1. Ветвление.

В матрице смежности производится поиск элемента со значением 0 такого, что «штраф» – сумма наименьших элементов в соответствующих строке и столбце, исключая сам элемент, – максимален. Индексам данного элемента соответствует ребро графа задачи коммивояжёра. Создаётся два новых решения, которые добавляются в список.

а) Решение, которое содержит это ребро в списке рёбер маршрута. Для данного решения удаляются столбец и строка матрицы смежности, на которых расположен данный элемент, а ребро, обратное данному, исключается – нельзя вернуться назад.

б) Решение, которое не содержит это ребро в списке рёбер маршрута. В таком случае данное ребро исключается, а к базовому весу маршрута добавляется «штраф».

Алгоритм выполняется до тех пор, пока не закончатся решения в списке возможных решений.

Вычислительная сложность данного метода в худшем случае сопоставима с методом полного перебора.

*Динамический метод Хелда-Карпа*

Данный метод основан на идее перебора всех возможных путей, проходящих через все города ровно один раз [16]. Суть алгоритма заключается в динамическом заполнении таблицы подзадач, где каждое состояние таблицы представляет собой пару (*B, j*), где *B* – подмножество городов, которые коммивояжёр уже посетил, а *j* – последний город, в котором находится коммивояжёр. Алгоритм последовательно увеличивает размер подмножества, перебирая все возможные варианты таких подмножеств в зависимости от предыдущего состояния таблицы.

Является точным методом решения задачи коммивояжёра и имеет вычислительную сложность по времени

*Алгоритм Лина-Кернигана*

В статье Lin S., Kernighan B.W. “An Effective Heuristic Algorithm for the Traveling-Salesman Problem” приводится эвристический алгоритм решения задачи [17]. Основная идея алгоритма заключается в замене дуг с целью приближения к оптимальному решению. Алгоритм предполагает наличие исходного допустимого решения. Необходимо выделить два множества дуг таких, что если все дуги первого множества удалить из найденного тура и заменить их дугами из второго множества, то результатом будет тур, который лучше предыдущего. Процесс переноса дуг повторяется до тех пор, пока становится невозможно сформировать такие два множества дуг. Как показали экспериментальные результаты, этот алгоритм достаточно часто находит даже глобально оптимальные решения.

*Жадный алгоритм*

Для получения приближенной оценки жадным алгоритмом просматриваются все ребра полного графа, выходящие из вершин, и выбираются, самые короткие ребра в количестве, равном количеству вершин. Если ребра формируют гамильтонов цикл, то это будет решением задачи [18].

*Антижадный алгоритм*

В статье Чусовлянкина А.А., Морозенко В.В. “Анализ точности и времени решения задачи коммивояжёра с помощью "антижадного" алгоритма” рассматривается алгоритм, основывающийся на последовательном приближении к оптимальному решению за счёт удаления рёбер в графе по определенным правилам при помощи матрицы состояний [19]. На каждом шаге алгоритма производится поиск самого длинного нерассмотренного ребра в графе. Ребро либо фиксируется, либо удаляется из графа, и в матрице состояний делается соответствующая пометка. Если невозможно ни фиксирование, ни удаление, то решение из предыдущего шага оказалось неверным.

### 2.5 Выбор методов решения задачи о рюкзаке и задачи коммивояжёра

С целью обеспечения дальнейших корректных расчётов эффективности метода, были выбраны пары из эквивалентных точных методов решения задачи о рюкзаке и задачи коммивояжёра. В данном случае таковыми являются методы, приведённые ниже.

1. Метод полного перебора для задачи о рюкзаке и метод полного перебора для задачи коммивояжёра.
2. Метод ветвей и границ для задачи о рюкзаке и модифицированный алгоритм Литтла для задачи коммивояжёра.
3. Метод динамического программирования для задачи о рюкзаке и динамический метод Хелда-Карпа для задачи коммивояжёра.

Вычислительная сложность алгоритмов в зависимости от количества вершин графа представлена в таблице 3.

Таблица 3 – Вычислительная сложность методов решения задачи коммивояжёра

|  |  |
| --- | --- |
| Название метода | Вычислительная сложность |
| Метод полного перебора |  |
| Модифицированный алгоритм Литтла |  |
| Динамический метод Хелда-Карпа |  |

## 3 АРХИТЕКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ

### 3.1 Используемые технологии

Библиотека написана на языке программирования Java версии 17 с использованием фреймворка для сборки пакетов Maven версии 3.9.6.

### 3.2 Структура программной реализации

Библиотека представляет собой Java-пакет с набором классов, которые реализуют решение задачи о рюкзаке методом сведения к задаче коммивояжёра двумя способами, перечисленными в предыдущей главе. Исходный код библиотеки представлен в репозитории GitHub [20].

#### 3.2.1 Классы моделей задач

Ниже перечислены классы, представляющие собой объекты, над которыми производятся основные операции в процессе решения.

**Knapsack**

Класс, представляющий собой рюкзак из задачи о рюкзаке.

В классе определены следующие приватные поля:

* *int capacity* – грузоподъёмность рюкзака.
* *long timeUs* – время выполнения алгоритма над объектом, мкс.

В классе определены следующие публичные поля:

* *ArrayList<Item> items* – список предметов, которые содержатся в рюкзаке.

В классе определены следующие публичные методы:

* Конструктор класса *Knapsack(int capacity)* инициализирует приватные поля объекта. Список предметов *items* инициализируется пустым списком.
* *boolean equals(Object o)* представляет собой переопределенный метод равенства объектов. Два объекта типа *Knapsack* равны, если это один и тот же объект или если все значения полей и предметы *Item* в динамическом массиве *items* у объектов равны.
* *int getCapacity()* возвращает значение приватного поля *capacity*.
* *float getTotalValue()* возвращает суммарную стоимость предметов из списка *items*.
* *int getTotalWeight()* возвращает суммарный вес предметов из списка *items*.
* *boolean isPacked()* возвращает *true,* если суммарный вес предметов из списка *items* не превышает грузоподъёмность рюкзака *capacity*, и *false* в противном случае.
* *void setTimeUs(long timeUs)* устанавливает значение времени в приватное поле *timeUs*.
* *long getTimeUs()* возвращает значение поля *timeUs*.

**Item**

Класс, представляющий собой предмет из задачи о рюкзаке.

В классе определены следующие приватные поля:

* *int weight* – вес предмета.
* *float value* – стоимость предмета.

В этом классе определены следующие публичные методы:

* Конструктор класса *Item(int weight, float value)* инициализирует приватные поля объекта.
* *boolean equals(Object o)* представляет собой переопределенный метод равенства объектов. Два объекта типа *Item* равны, если это один и тот же объект или значения полей *weight* и *value* у объектов равны.
* *int compareTo(Item that)* представляет собой переопределенный метод сравнения. Объекты сравниваются сначала по полю *weight*; если они значения равны, то сравнивается по значению поля *value*. Необходим для реализации метода ветвей и границ для задачи о рюкзаке.
* *int getWeight()* возвращает значение приватного поля *weight*.
* *float getValue()* возвращает значение приватного поля *value*.

**ItemTSP**

Класс представляет собой предмет задачи о рюкзаке, содержащий номера вершин оптимального маршрута графа коммивояжёра.

В классе определены следующие публичные поля:

* *int weight* – вес предмета.
* *float value* – стоимость предмета.
* *int amount* – количество предметов.
* *int i* – индекс начальной вершины дуги из графа задачи коммивояжёра.
* *int j* – индекс конечной вершины дуги из графа задачи коммивояжёра.

В классе определены следующие публичные методы:

* Конструктор класса *ItemTSP(int i, int j, int weight, float value)* поля объекта. Количество предметов *amount* приравнивается 1.
* *static Item toItemKP(ItemISP item)* возвращает объект типа *Item* по заданному объекту типа *ItemTSP.*

**PathDataBnB**

Класс представляет собой одно из рассматриваемых решений задачи коммивояжёра для метода ветвей и границ.

В классе определены следующие публичные поля:

* *float[][] adjacencyMatrix* – весовая матрица графа текущего решения коммивояжёра.
* *ArrayList<ArrayList<Integer>> currentPath* – индексы рёбер графа коммивояжёра, которое включает в себя текущее решение.
* *int baseCost* – текущий вес решения;
* *int penaltyI, penaltyJ* – индексы следующего оптимального ребра на данном данного этапа решения;
* *float maxPenalty* – величина, на которую возрастёт вес решения в случае удаления ребра (*penaltyI, penaltyJ*) из графа коммивояжёра.

В классе определены следующие приватные поля:

* *int nodesAmount* – количество вершин в графе коммивояжёра.

В классе определены следующие публичные методы:

* Конструктор *PathDataBnB(float[][] adjacencyMatrix, ArrayList <ArrayList<Integer>> currentPath, int baseCost)* инициализирует соответствующие поля класса.
* Конструктор *PathDataBnB(float[][] adjacencyMatrix)* инициализирует соответствующее поле класса, остальные требуемые поля принимают значения по умолчанию: значение *baseCost* равно *0,* динамический список *currentPath* пуст.
* *int compareTo(PathDataBnB o)* представляет собой переопределенный компаратор для использования структуры данных с сортировкой *TreeSet* из встроенного пакета *util*.
* *PathDataBnB clone()* возвращает новый объект, который представляет собой копию исходного объекта.
* *void reduce()* производит операцию редукции весовой матрицы графа коммивояжёра. В результате редукции текущий вес решения *baseCost* увеличивается на значения редукции по каждой строке и каждому столбцу матрицы, а соответствующие значения в строках и столбцах уменьшаются.
* *void setPenalty()* производит поиск ребра, которое при исключении добавит наибольшее значение к текущему весу решения.
* *void setPathEdge(int indexI, int indexJ)* добавляет ребро в список рёбер текущего решения и исключает те рёбра из весовой матрицы, которые образуют циклы из 3 вершин с существующими в массиве рёбрами.
* *boolean isPathFull(int startNode)* проверяет решение на наличие всех вершин в маршруте коммивояжёра. Возвращает истину, если начальная вершина встречается дважды, а любая другая только один раз.
* *ArrayList<Integer> restorePath(int startNode*) возвращает динамический массив индексов вершин, которые соответствуют решению задачи коммивояжёра, по индексам начальной и конечной вершин рёбер в поле *currentPath*.

#### 3.2.2 Классы алгоритмов

Ниже перечислены классы, содержащие реализацию методов решения задачи о рюкзаке.

**AlgorithmKnapsack**

Данный абстрактный базовый класс определяет структуру класса-потомка для решения задачи о рюкзаке.

В классе определены следующие публичные методы:

* *abstract void run(Knapsack knapsack, ArrayList<Item> items)* является методом, который должен реализовывать алгоритм решения задачи о рюзаке.

**AlgorithmKnapsackBnB**

Класс, реализующий решение задачи о рюкзаке методом ветвей и границ. Наследуется от класса *AlgorithmKnapsack*.

В классе определены следующие публичные методы:

* *void run()* реализует алгоритм решения задачи коммивояжёра методом ветвей и границ. В результате выполнения алгоритма формируется массив с оптимальным маршрутом коммивояжёра.

В классе определены следующие приватные методы:

* *void packKnapsackFromBinaryString(Knapsack knapsack, ArrayList <Item> items, String binaryString)* добавляет предметы в рюкзак, в соответствии со строкой упаковки которая состоит из единиц иулей, где «1» соответствует присутствию предмета в рюкзаке и «0» отсутствию предмета в рюкзаке.

**AlgorithmKnapsackBruteForce**

Класс, реализующий решение задачи о рюкзаке методом полного перебора. Наследуется от класса AlgorithmKnapsack.

В классе определены следующие публичные методы:

* void run() реализует алгоритм решения задачи коммивояжёра методом полного перебора. В результате выполнения алгоритма формируется массив с оптимальным маршрутом коммивояжёра.

В классе определены следующие приватные методы:

* void packKnapsackFromSolutionInteger(Knapsack knapsack, ArrayList <Item> items, int solution) добавляет предметы в рюкзак, в соответствии с порядковым номером упаковки, который переводится в двоичную систему исчисления, где «1» соответствует присутствию предмета в рюкзаке и «0» отсутствию предмета в рюкзаке.

**AlgorithmKnapsackDynProg**

Класс, реализующий решение задачи о рюкзаке методом динамического программирования. Наследуется от класса *AlgorithmKnapsack*.

В классе определены следующие публичные методы:

* *void run(Knapsack knapsack, ArrayList<Item> items)* реализует алгоритм решения задачи о рюкзаке методом динамического программирования. В процессе решения используется массив *ArrayList<ArrayList<Integer>> matrix* для хранения промежуточных результатов вычисления.

**AlgorithmKnapsackReduction**

Данный абстрактный класс определяет структуру класса-потомка для решения сведения задачи о рюкзаке к задаче коммивояжёра и предоставляет методы сведения задач. Наследуется от абстрактного класса *AlgorithmKnapsack*.

В классе определены следующие публичные поля:

* *ArrayList<ItemTSP> currentResult* – промежуточный результат решения задачи о рюкзаке.
* *ArrayList<ItemTSP> itemsTSP* – список предметов задачи о рюкзаке, содержащих номера вершин оптимального маршрута графа коммивояжёра.
* *int maxWeight* – вес промежуточного решения.
* *float maxValue* – стоимость промежуточного решения.
* *ArrayList<ItemTSP> result –* оптимальный результат*.*

В классе определены следующие защищённые методы:

* *ArrayList<ArrayList<Float>> buildMatrixForTSP(Knapsack knapsack, ArrayList<Item> items)* сводит задачу о рюкзаке к задаче коммивояжёра: строит весовую матрицу графа задачи коммивояжёра по заданному списку предметов задачи о рюкзаке.
* *void convertSolutionToKP(ArrayList<ArrayList<Float>> matrixTSP, ArrayList<Integer> tour, Knapsack knapsack, ArrayList<Item> items)* сводит задачу коммивояжёра к задаче о рюкзаке: по заданным весовой матрице и маршруту коммивояжёра задаёт рюкзак и содержащиеся в нём предметы.

**AlgorithmKnapsackReductionBnB**

Класс, реализующий решение сведённой задачи о рюкзаке модифицированным алгоритмом Литтла. Наследуется от класса *AlgorithmKnapsackReduction*.

В классе определены следующие публичные методы:

* *void run(Knapsack knapsack, ArrayList<Item> items)* реализует решение сведённой задачи о рюкзаке модифицированным алгоритмом Литтла. Метод сводит задачу, создаёт соответствующий объект класса для решения и сводит решение обратно.

**AlgorithmKnapsackReductionBruteForce**

Класс, реализующий решение сведённой задачи о рюкзаке методом полного перебора. Наследуется от класса *AlgorithmKnapsackReduction*.

В классе определены следующие публичные методы:

* *void run(Knapsack knapsack, ArrayList<Item> items)* реализует решение сведённой задачи о рюкзаке методом полного перебора, проводя действия, аналогично классу выше.

**AlgorithmKnapsackReductionDynProg**

Класс, реализующий решение сведённой задачи о рюкзаке методом динамического программирования Хелда-Карпа. Наследуется от класса *AlgorithmKnapsackReduction*.

В классе определены следующие публичные методы:

* *void run(Knapsack knapsack, ArrayList<Item> items)* реализует решение сведённой задачи о рюкзаке методом динамического программирования Хелда-Карпа, проводя действия, аналогично классу выше.

**AlgorithmTSP**

Данный абстрактный базовый класс определяет структуру класса-потомка для решения задачи коммивояжёра.

В классе определены следующие публичные поля:

* *float minTourCost* – вес оптимального маршрута коммивояжёра.
* *ArrayList<Integer> minTour* – список вершин, последовательно образующих маршрут коммивояжёра.

В классе определены следующие защищённые поля:

* *int startNode* – начальная вершина маршрута коммивояжёра.
* *float [][] distance* – весовая матрица графа задачи коммивояжёра.
* *int nodesAmount* – количество вершин.

В классе определены следующие публичные методы:

* Конструктор класса *AlgorithmTSP(int startNode, ArrayList<ArrayList <float>> distance)* инициализирует публичные и приватные поля класса.
* *abstract void run()* является методом, который должен реализовывать алгоритм решения задачи коммивояжёра.

**AlgorithmTSPBnB**

Класс, реализующий решение задачи коммивояжёра модифицированным алгоритмом Литтла. Используется в классе *AlgorithmKnapsackReductionBnB*. Наследуется от класса *AlgorithmTSP*.

В классе определены следующие публичные поля:

* *TreeSet<PathDataBnB> data* – структура данных, содержащая все решения *PathDataBnB* в порядке возрастания веса текущего решения.

В классе определены следующие публичные методы:

* Конструктор *AlgorithmTSPBnB(int startNode, float[][] distance)* вызывает конструктор родительского класса, таким образом инициализируя поля класса. Дополнительно инициализирует поле *data* созданным объектом типа *PathDataBnB*, содержащим весовую матрицу задачи коммивояжёра.
* *void run()* реализует решение задачи коммивояжёра модифицированным алгоритмом Литтла, используя вышеперечисленные методы и объекты, производя последовательность действий над первым объектом из *data*, до тех пор, пока не будут рассмотрены все решения.

**AlgorithmTSPBruteForce**

Класс, реализующий решение задачи коммивояжёра методом полного перебора. Используется в классе *AlgorithmKnapsackReductionBruteForce*. Наследуется от класса *AlgorithmTSP*.

В классе определены следующие приватные поля:

* *ArrayList<Integer> currentTour* – промежуточное решение.

В классе определены следующие публичные методы:

* Конструктор *AlgorithmTSPBruteForce(int startNode, float[][] distance)* вызывает конструктор родительского класса, таким образом инициализируя поля класса.
* *void run()* реализует решение задачи коммивояжёра методом полного перебора.

В классе определены следующие приватные методы:

* *void checkTour()* проверяет, является ли маршрут, полученный на данном шаге, лучше, чем текущий оптимальный путь. Если это так, то осуществляет замену.
* *void tspRecurcive()* представляет собой рекурсивный метод, реализующий решение методом полного перебора.

**AlgorithmTSPDynProg**

Класс, реализующий решение задачи коммивояжёра методом динамического программирования Хелда-Карпа итеративным подходом. Используется в классе *AlgorithmKnapsackReductionDynProg*. Наследуется от класса *AlgorithmTSP*.

В классе определены следующие приватные поля:

* *float[][] dp* – двумерный массив кратчайших путей. Значения в этом массиве обновляются и заполняются по ходу выполнения алгоритма для хранения промежуточных результатов.
* *int[][] parent* – двумерный массив, в котором для каждого узла и состояния хранится предыдущий узел в кратчайшем пути. Этот массив помогает восстановить оптимальный маршрут после завершения алгоритма.

В классе определены следующие публичные методы:

* *void run()* реализует решение задачи коммивояжёра методом динамического программирования Хелд-Карпа.

В классе определены следующие приватные методы:

* *void reconstructPath(int mask, int current)* восстанавливает найденный оптимальный путь коммивояжёра после завершения основной части алгоритма.

#### 3.2.3 Вспомогательные классы

Ниже перечислены классы, используемые для удобного представления или хранения данных в процессе решения.

**Constant**

Содержит глобальную константы в виде статического поля, необходимого для работы алгоритмов.

В классе определены следующие публичные поля:

* *static int INFINITY* – значение, принимаемое в алгоритмах за бесконечно большую величину (108).

**Pair**

Шаблонный класс, представляющий собой объединение двух объектов в один. Необходим в некоторых методах для одновременного получения двух возвращаемых значений из метода.

В классе определены следующие публичные поля:

* *final T t, final U u* – пара объектов.

В классе определены следующие публичные методы:

* Конструктор класса *Pair(T t, U u)* инициализирует публичные поля объекта.

#### 3.2.4 Класс ввода-вывода данных

Класс ввода-вывода **FileIO** реализует взаимодействие с текстовыми файлами.

В классе определены следующие публичные методы:

* *static Pair<Knapsack, ArrayList<Item>> readDataKP(String filename)* осуществляет импорт данных из файла по пути, указанного в параметре *filename.* Метод возвращает объект типа *Pair*, представляющий собой пару объектов: рюкзак типа *Knapsack*, и список предметов типа *ArrayList<Item>*. В случае ошибки чтения (нет доступа к файлу, неправильный формат данных) вызывается исключение.
* *static void saveResultKP(String output, Knapsack knapsack)* осуществляет экспорт данных в файл, указанного в параметре *output.* В случае ошибки (невозможность записи) вызывается исключение.

#### 3.2.5 Класс интерфейса библиотеки

Класс **API** представляет собой программный интерфейс библиотеки, с которым взаимодействует пользователь.

В классе определены следующие публичные методы:

* *static void bnb(Knapsack knapsack, ArrayList<Item> items)* вызывает приватный метод *run* с объектом класса, соответствующим решению задачи о рюкзаке методом ветвей и границ.
* *static void bruteForce(Knapsack knapsack, ArrayList<Item> items)* вызывает приватный метод *run* с объектом класса, соответствующим решению задачи о рюкзаке методом полного перебора.
* *static void dynamicProgramming(Knapsack knapsack, ArrayList<Item> items)* аналогичен методу выше, решает задачу о рюкзаке методом динамического программирования.
* *static void reductionBnB(Knapsack knapsack, ArrayList<Item> items)* сводит задачу о рюкзаке к задаче коммивояжёра, решает сведённую задачу о рюкзаке модифицированным алгоритмом Литтла и сводит обратно к задаче о рюкзаке.
* *static void reductionBruteForce(Knapsack knapsack, ArrayList<Item> items)* аналогичен методу выше, решает сведённую задачу методом полного перебора.
* *static void reductionDynamicProgramming(Knapsack knapsack, ArrayList<Item> items)* аналогичен методу выше, решает сведённую задачу методом динамического программирования Хелда-Карпа.
* Вышеперечисленные методы переопределены для аргумента типа *Pair* и выполняют те же действия.
* *static void run(Knapsack knapsack, ArrayList<Item> items, AlgorithmKnapsack algorithm)* вызывает решение задачи о рюкзаке требуемым методом для заданных параметров. Время работы алгоритма записывается в соответствующее поле объекта рюкзака. Решение задачи о рюкзаке содержится в поле *items* объекта рюкзака*.*
* *static Pair<Knapsack, ArrayList<Item>> generateRandomKP(int capacity, int itemsAmount, int weightMin, int weightMax, float valueMin, float valueMax)* генерирует объект типа *Pair*, содержащий рюкзак с заданными параметрами и список предметов со случайными величинами веса и стоимости в указанных пределах. Если минимальная величина меньше максимальной, грузоподъёмность рюкзака меньше 1, минимальный вес предмета меньше 1, минимальная стоимость предмета меньше 0, то генерируется ошибка и выводится соответствующее сообщение.

### 3.3 Взаимодействие элементов программной реализации

Все описанные выше классы взаимодействуют друг с другом, согласно UML-диаграмме классов, представленной на рис. 1.

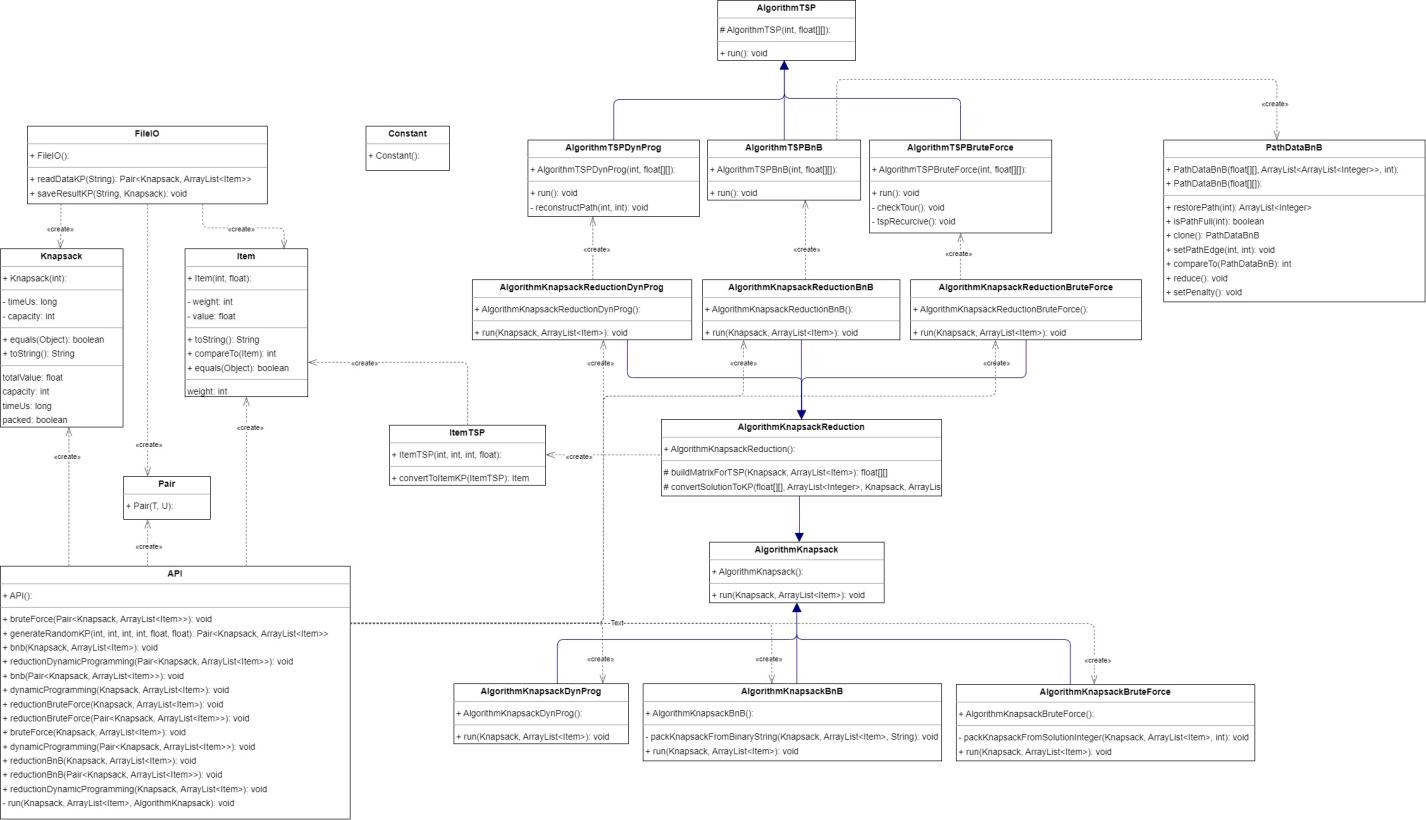


Рисунок 1 – UML-диаграмма классов библиотеки

### 3.4 Структура тестирования

Для классов алгоритмов, класса ввода-вывода и класса интерфейса библиотеки были реализованы классы юнит-тесты. Для проведения юнит-тестирования был использован фреймворк для автоматического юнит-тестирования JUnit. Классы юнит-тестов перечислены ниже.

* *AlgorithmKnapsackBnBTest* – корректность реализованного метода ветвей и границ для задачи о рюкзаке;
* *AlgorithmKnapsackBruteForceTest* – корректность реализованного метода полного перебора для задачи о рюкзаке;
* *AlgorithmKnapsackDynProgTest* – корректность реализованного метода динамического программирования для задачи о рюкзаке;
* *AlgorithmTSPBnBTest* – корректность реализованного модифицированного алгоритма Литтла для задачи коммивояжёра;
* *AlgorithmTSPBruteForceTest* – корректность реализованного метода грубой силы для задачи коммивояжёра;
* *AlgorithmTSPDynProgTest* – корректность реализованного метода динамического программирования для задачи коммивояжёра;
* *FileIOTest* – методы чтения и записи;
* *APITest* – метод генерации данных для задачи о рюкзаке.

## 1 ИССЛЕДОВАНИЕ РЕЗУЛЬТАТА

### 4.1 Исследование программного решения

Программное решение было протестировано на корректность посредством юнит-тестирования, реализация которого описана в предыдущей главе. Далее было произведено сравнение полученного решения с аналогами.

#### 4.1.1 Тестирование на корректность

Тестирование методов решения было основано на проверке полученного ответа к задачам, составленных с учётом различных параметров. Для методов решения задачи о рюкзаке – грузоподъёмность рюкзака, количество предметов, вес предметов. Для методов решения задачи коммивояжёра – количество вершин, вес рёбер графа коммивояжёра, связность графа.

По результатам тестирования реализованных методов решения задач ошибок обнаружено не было.

Тестирование методов, реализующих работу библиотеки, проводилось на заготовленных данных. Основная цель данного тестирования – проверка работоспособности функций библиотеки на значениях, близким к граничным, или их превышающие. Были протестированы методы генерации данных задачи о рюкзаке, ввод и вывод данных.

По результатам тестирования реализованных методов для работы библиотеки ошибок обнаружено не было.

#### 4.1.2 Сравнение программного решения с аналогами

Сравнение полученного программного решения с аналогами представлено в таблице 4.

Таблица 4 – Сравнение программного решения с аналогами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Языки программирования | Количество методов решения | Валидация входных параметров | Импорт и экспорт данных |
| OR-Tools | C++, C#,  Java, Python | 1 | Да | Нет |
| go-knapsack | Go | 1 | Нет | Нет |
| mknapsack | Python | 2 | Да | Нет |
| knapsacksolver | C++ | 4 | Да | Да |
| knapsack-pip | Python | 1 | Нет | Нет |
| Реализованное решение | Java | 6 | Да | Да |

### 4.2 Исследование полученного метода решения задачи о рюкзаке

Реализованные методы решения задачи о рюкзаке путём сведения к задаче коммивояжёра были исследованы на зависимость от параметров количества предметов и грузоподъёмности рюкзака аналитически и экспериментально. Измерения проводились на данных, сгенерированных случайным образом по закону равномерного распределения в заданных пределах. Для каждого набора данных проводилось 10 испытаний, в качестве результата указано среднее арифметическое времени выполнения метода.

Измерения проводились на устройстве (ноутбук) с характеристики, приведёнными ниже.

* Операционная система: Windows 11 23H2;
* процессор: Intel® Core™ i5-10210U CPU @ 1.60GHz;
* графический процессор: Intel UHD Graphics 620;
* оперативная память: 8 ГБ;
* жёсткий диск: 512 ГБ (NVMe SSD).

#### 4.2.1 Оценка вычислительной сложности

Определим аналитически сложность алгоритма сведения задачи о рюкзаке к задаче коммивояжёра относительно параметров задачи.

1. Сведение предметов рюкзака к графу коммивояжёра.

Граф задачи коммивояжёра состоит из вершин. На первом шаге построения графа добавляются рёбра с приведённым весом предметов рюкзака. Для каждой из первых вершин графа проверяются все предметов рюкзака на возможность добавить ребро. На втором шаге построения графа добавляются рёбра с нулевым весом для первой вершины графа и каждой из последних вершин. Вычислительная сложность сведения к задаче коммивояжёра составляет (14)

|  |  |
| --- | --- |
|  | (14) |

1. Сведение полученного маршрута коммивояжёра к предметам рюкзака.

По полученному маршруту графа коммивояжёра строится предметов графа коммивояжёра. Для этого необходимо найти сумму весов всех рёбер графа коммивояжёра, представленных в форме матрицы размерностью Далее необходимо обойти маршрут коммивояжёра длиной , чтобы получить для получения предметов графа коммивояжёра. Вычислительная сложность сведения маршрута коммивояжёра к предметам рюкзака составляет (15)

|  |  |
| --- | --- |
|  | (15) |

1. Алгоритм решения задачи коммивояжёра.

Количество вершин графа коммивояжёра основывается на грузоподъёмности рюкзака . Вычислительная сложность большинства алгоритмов решения задачи коммивояжёра зависит от количества вершин, поэтому грузоподъёмность рюкзака будет влиять на вычислительную сложность алгоритма.

В результате вычислительная сложность решения задачи о рюкзаке методом сведения к задаче коммивояжёра может быть записана в следующем виде (16)

|  |  |
| --- | --- |
|  | (16) |

где– вычислительная сложность алгоритма решения задачи коммивояжёра.

#### 4.2.2 Измерение зависимости вычислительной сложности от количества предметов

Были проведены измерения с целью выявить экспериментальную зависимость времени выполнения работы методов от количества предметов и сравнить с аналитической оценкой. Экспериментальные данные и результаты измерений представлены в приложении А.

На рис. 2 представлен график зависимости времени вычисления от количества предметов методом полного перебора для сведённой задачи. На графике наблюдается линейная зависимость.



Рисунок 2 – График зависимости времени вычисления от количества предметов методом полного перебора для сведённой задачи

На рис. 3 представлена точечная диаграмма времени вычисления от количества предметов модифицированным алгоритмом Литтла для сведённой задачи. Диаграмма демонстрирует отсутствие корреляции между временем вычисления и количеством предметов. Наблюдаемый результат может быть связан с зависимостью времени работы от свойств входных данных (вес и стоимость предметов рюкзака): алгоритм пропускает множества маршрутов коммивояжёра, которые гарантированно имеют вес больше, чем у текущего лучшего маршрута.

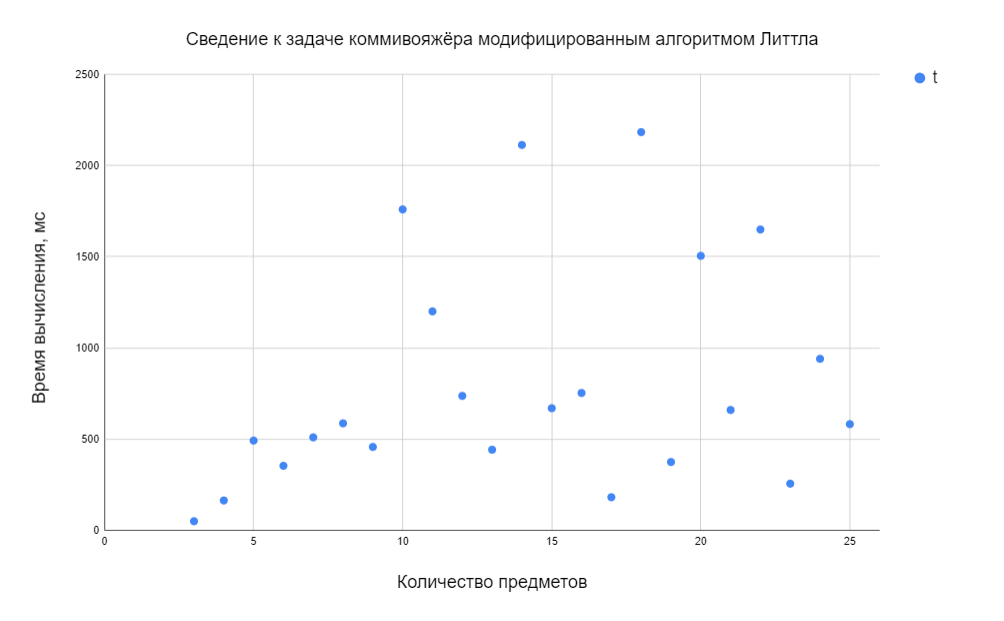


Рисунок 3 – Точечная диаграмма времени вычисления от количества предметов модифицированным алгоритмом Литтла для сведённой задачи

На рис. 4 представлен график зависимости времени вычисления от количества предметов методом Хелда-Карпа для сведённой задачи. На графике также наблюдается линейная зависимость.



Рисунок 4 – График зависимости времени вычисления от количества предметов динамическим методом Хелда-Карпа для сведённой задачи

#### 4.2.3 Измерение зависимости вычислительной сложности от грузоподъёмности рюкзака

Были проведены измерения с целью выявить экспериментальную зависимость времени выполнения работы методов от грузоподъёмности рюкзака и сравнить с аналитической оценкой. Экспериментальные данные и результаты измерений представлены в приложении Б.

На рис. 5 представлен график зависимости времени вычисления от грузоподъёмности рюкзака методом полного перебора для сведённой задачи в логарифмическом масштабе. График демонстрирует зависимость, близкую к факториальной (нелинейный характер на логарифмическом масштабе), что также соответствует методу решения задачи коммивояжёра.

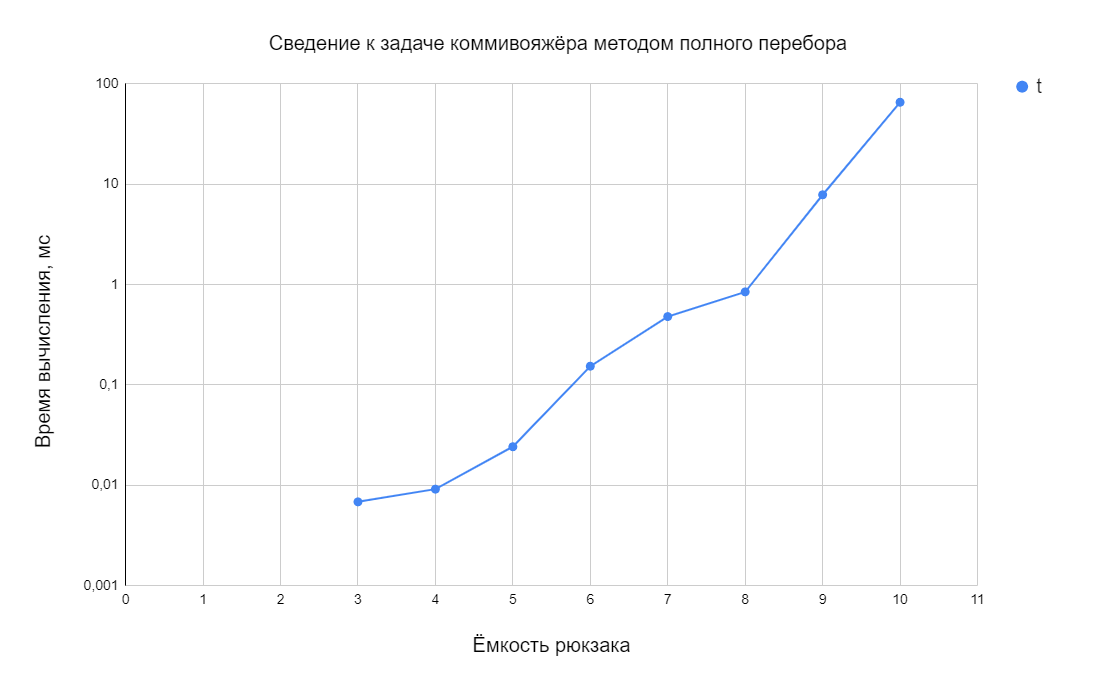


Рисунок 5 – График зависимости времени вычисления от грузоподъёмности рюкзака методом полного перебора для сведённой задачи

На рис. 6 представлен график зависимости времени вычисления от грузоподъёмности рюкзака модифицированным алгоритмом Литтла для сведённой задачи в логарифмическом масштабе. График демонстрирует вычислительную сложность выше экспоненциальной, что соответствует факториальной сложности, соответствующей этому методу.

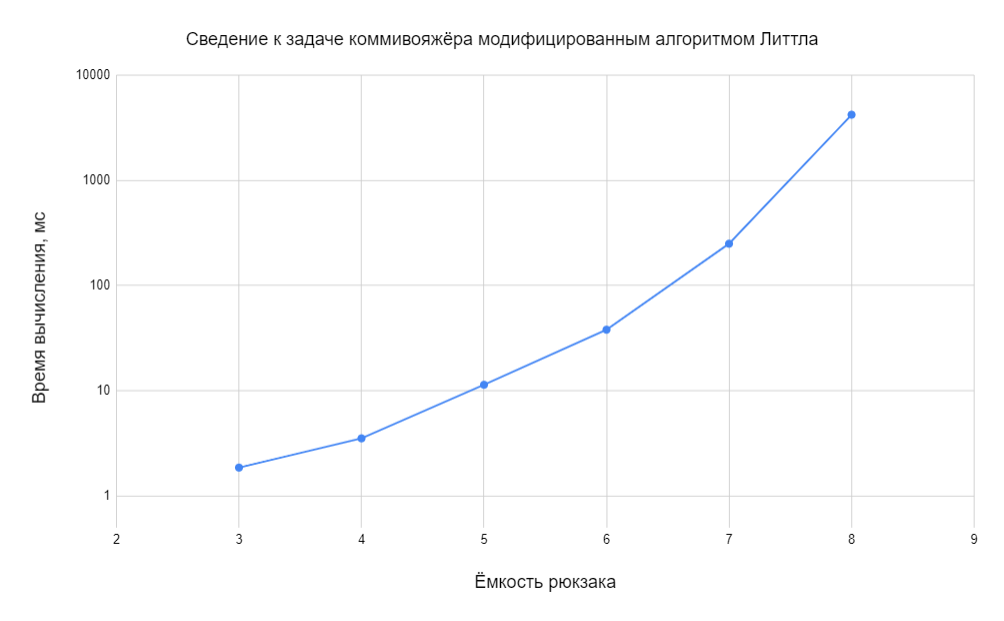


Рисунок 6 – График зависимости времени вычисления от грузоподъёмности рюкзака модифицированным алгоритмом Литтла для сведённой задачи

На рис. 7 представлен график зависимости времени вычисления от грузоподъёмности рюкзака динамическим методом Хелда-Карпа для сведённой задачи в логарифмическом масштабе. График демонстрирует явно экспоненциальную вычислительную сложность, что соответствует вычислительной сложности этого метода решения задачи коммивояжёра.

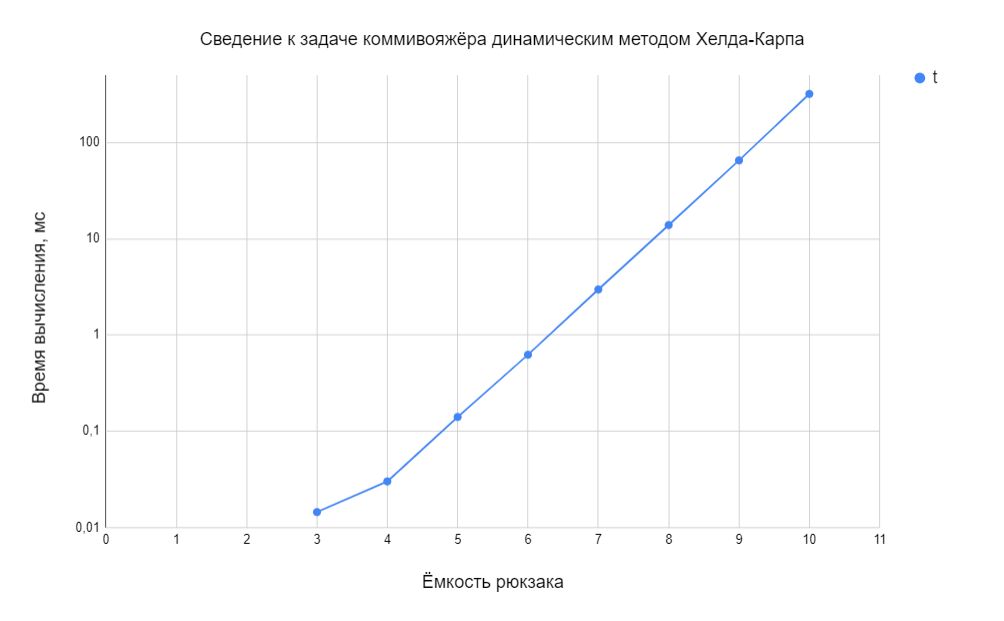


Рисунок 7 – График зависимости времени вычисления от грузоподъёмности рюкзака динамическим методом Хелда-Карпа для сведённой задачи

#### 4.2.4 Сравнение вычислительной сложности полученного метода решения с аналогами

Сравнение полученного метода решения задачи о рюкзаке с аналогами представлено в таблице 5 ниже.

Точность решения задачи методом сведения соответствует используемому методу решения задачи коммивояжёра.

Только классический тип задачи подходит для решения задачи о рюкзаке методом сведения, потому что алгоритм построения графа коммивояжёра по предметам рюкзака не расширяется для построения на мультиграфе.

Таблица 5 – Сравнение полученного метода решения с аналогами

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Вычислительная сложность | Точность решения | Тип задачи |
| Метод полного перебора |  | точный | классическая, ограниченная |
| Метод ветвей и границ |  | точный | классическая, ограниченная, неограниченная |
| Метод динамического программирования |  | точный | классическая, ограниченная |
| Сведение к задаче коммивояжёра методом полного перебора |  | точный | классическая |
| Сведение к задаче коммивояжёра модифицированным алгоритмом Литтла |  | точный | классическая |
| Сведение к задаче коммивояжёра алгоритмом Хелда-Карпа |  | точный | классическая |

#### 4.2.5 Сравнение времени вычисления полученного метода с аналогами

Были проведены измерения времени вычисления аналогичных методов на том же наборе параметров с целью сравнения времени работы и выявления практических преимуществ и недостатков метода. Экспериментальные данные и результаты измерений представлены в приложении А и в приложении Б.

По мере роста количества предметов методы сведения демонстрируют линейный рост времени вычисления. Это гораздо эффективнее метода полного перебора и метода ветвей и границ. Данная зависимость соответствует методу динамического программирования.

При изменении грузоподъёмности рюкзака методы сведения демонстрируют стремительный рост времени вычисления, что делает их использование нецелесообразным.

Проведённые измерения позволяют сделать вывод о применимости методов сведения для высокого значения параметра при сравнительно малом значении параметра .

### 4.3 Общие выводы по исследованию

Полученное программное решение имеет реализовано на языке программирования Java, включает в себя 6 методов решения задачи о рюкзаке (3 из которых реализованы через сведение к задаче коммивояжёра), имеет валидацию входных данных, также был реализован импорт и экспорт данных в отдельный внешний файл.

Решение отличается от аналогов легковесностью (отсутствием прочих реализованных математических моделей, не имеющих отношение к решению задачи о рюкзаке), гибкостью в выборе метода решения задачи о рюкзаке и наличием валидации входных данных. Решение было реализовано на языке программирования Java, что расширяет разнообразие библиотек в открытом доступе для данного языка программирования.

Полученные методы решения задачи о рюкзаке путём сведения к задаче коммивояжёра неэффективны при большом количестве предметов и сравнительно эффективны для малого значения грузоподъёмности рюкзака при большом количестве предметов, обладая в данных условиях вычислительной сложностью, подобной методу динамического программирования. Также полученные методы применимы только для классической (0-1) задачи о рюкзаке.

## 5 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ВКР

### 5.1 Концепция экономического обоснования разработки научно-технического продукта

Выпускная квалификационная работа направлена на исследование эффективности метода решения задачи о рюкзаке путём сведения её к задаче коммивояжёра и на разработку программного продукта, который реализует представленную математическую модель метода. Разработанный продукт предназначен для ускорения процесса разработки информационных систем, использующих алгоритмы решения задачи о рюкзаке.

### 5.2 Составление плана-графика работ

Для определения совокупной длительности работы был составлен детализированный план работ (начиная с составления технического задания (ТЗ) и до оформления технической документации включительно). Продолжительность работ определена по факту. Длительность работы каждой работы в человеко-днях отражена в таблице 6.

Таблица 6 ‒ Длительность этапа разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование работы | Длительность работы, чел.-дн. | |
| Руководитель | Исполнитель |
| 1 | Составление ТЗ | 1 | 1 |
| 2 | Анализ существующих решений ТЗ | 2 | 3 |
| 3 | Разработка математической модели | 1 | 2 |
| 4 | Разработка алгоритмов сведения | 2 | 6 |
| 5 | Реализация математической модели | 2 | 3 |
| 6 | Реализация алгоритмов сведения | 3 | 9 |
| 7 | Отладка и тестирование программы | 1 | 4 |

Продолжение таблицы 6

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | Исследование работы математической модели | 1 | 3 |
| 9 | Оформление технической документации | 1 | 4 |
| ИТОГО | | 14 | 35 |

### 5.3 Расчёт расходов на оплату труда

Должностной оклад согласно приказу ОД/0432 от 15.10.2020 г. равен 14500 руб./мес. для исполнителя (инженер) и 40600 руб./мес. для руководителя (доцент, к.т.н.). Дневная ставка из расчёта 21 рабочего дня в месяц для исполнителя 690,48 руб./день и для руководителя 1933,33 руб./день.

На основе данных о трудоёмкости выполняемых работ и ставки (за день или час) соответствующих исполнителей определены расходы на заработную плату исполнителей и отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование [21].

Расходы на основную заработную плату исполнителей определяются по формуле (17)

|  |  |
| --- | --- |
|  | (17) |

где – расходы на основную заработную плату исполнителей (руб.); – количество исполнителей; – время, затраченное *i*-м исполнителем на проведение исследования (дни или часы); – ставка *i*-го исполнителя (руб./день или руб./час).

На основании данных в таблице 6 и дневной ставки исполнителей значение основной заработной платы (18)

|  |  |
| --- | --- |
|  | (18) |

Расходы на дополнительную заработную плату исполнителей определяются по формуле (19)

|  |  |
| --- | --- |
|  | (19) |

где – расходы на дополнительную заработную плату исполнителей (руб.); – расходы на основную заработную плату исполнителей (руб.); – норматив дополнительной заработной платы (%). При выполнении расчётов в ВКР норматив дополнительной заработной платы принимаем равным 14%.

Значение дополнительной заработной платы равно (20)

|  |  |
| --- | --- |
|  | (20) |

### 5.4 Расчёт отчислений на социальные нужды

Отчисления на страховые взносы на обязательное социальное, страхование с основной и дополнительной заработной платы исполнителей определяются по формуле:

где – отчисления на социальные нужды с заработной платы (руб.); – расходы на основную заработную плату исполнителей (руб.); – расходы на дополнительную заработную плату исполнителей (руб.); – норматив отчислений в процентах на страховые взносы в Фонд Социального Страхования, Пенсионный Фонд России и Федеральный Фонд Обязательного Медицинского Страхования. Норматив отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование 30%.

Отчисление на страховые взносы (21)

|  |  |
| --- | --- |
|  | (21) |

### 5.5 Расчёт расходов на материалы

Оценка потребности в ресурсах устанавливается в натуральных и стоимостных показателях.

Себестоимость по каждой из статей рассчитана нормативным методом. Для этого были найдены параметры продукции (нормы расхода), заложенные в конструкторско-технологической документации. Расчеты по сырью и материалам производятся по следующей формуле (22)

|  |  |
| --- | --- |
|  | (22) |

где – затраты на сырье и материалы (руб.); – индекс вида сырья или материала; – норма расхода -го материала на единицу продукции (ед.); – цена приобретения единицы -го материала (руб./ед.); – норма транспортно-заготовительных расходов (%). При выполнении расчётов в ВКР норму транспортно-заготовительных расходов () принимаем равной 10%. Расчёт затрат на сырье и материалы представлен в таблице 7.

Таблица 7 – Расходы на сырье и материалы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Изделие | Тип (профиль, сорт, марка, размер) | Норма расхода на изделие, ед. | Цена за единицу, руб. | Сумма на изделие, руб. |
| Бумага для принтера | SVETOCOPY, A4 | 1 пачка | 378 | 270 |
| Тетрадь 48 листов А5 | ErichKrause | 1 штука | 48 | 48 |
| Ручка синяя | PILOT | 4 штуки | 79 | 316 |
| Транспортные расходы | | | | 70 |
| Расходы на печать | | | | 2340 |
| Итого | | | | 3044 |

### 5.6 Расчёт затрат по работам, выполняемым сторонними организациями

Услуги сторонних организаций учитываются по их фактической стоимости за вычетом НДС (20%).

Для выполнения ВКР потребовалась услуга доступа в Интернет от интернет-провайдера «МТС». Стоимость услуги составила 1090 руб./месяц или 35,2 руб./день.

Тогда затраты по работам, выполняемым сторонними организациями, равны (23)

|  |  |
| --- | --- |
|  | (23) |

### 5.7 Спецоборудование

Для проведения работы потребовалась покупка специального оборудования. Затраты на оборудование представлены в таблице 8.

Таблица 8 – Затраты на специальное оборудование

|  |  |  |  |
| --- | --- | --- | --- |
| Оборудование | Норма расхода на единицу продукции, шт. | Цена, руб./шт. | Сумма на единицу продукции, руб. |
| Ноутбук HUAWEI MateBook D 14 NbB-WAH9 | 1 | 36799 | 36799 |
| Итого | | | 36799 |

### 5.8 Расчёт амортизационных отчислений

Амортизационные отчисления по основному средству i за год определяются как (24)

|  |  |
| --- | --- |
|  | (24) |

где – амортизационные отчисления за год по -му основному средству (руб.); – первоначальная стоимость -го основного средства; – годовая норма амортизации -го основного средства 33.3%. Согласно постановлению Правительства РФ от 01.01.2002 №1 (ред. от 18.11.2022) «О классификации основных средств, включаемых в амортизационные группы» нормативный срок службы используемого персонального компьютера, ноутбука и печатающих устройств равен 2-3 года.

Амортизационные отчисления за год составят (25)

|  |  |
| --- | --- |
|  | (25) |

Величина амортизационных отчислений по -му основному средству, используемому студентом при работе над ВКР, определяется по формуле (26)

|  |  |
| --- | --- |
|  | (26) |

где – амортизационные отчисления по -му основному средству, используемому студентом в работе над ВКР (руб.); – амортизационные отчисления за год по -му основному средству (руб.); – время, в течение которого студент использует -ое основное средство (мес.).

Тогда величина амортизационных отчислений по -му основному средству (27)

|  |  |
| --- | --- |
|  | (27) |

### 5.9 Расчёт накладных расходов

Накладные расходы – расходы на управление и хозяйственное обслуживание, вычисляются как процент от суммы заработной платы основной и дополнительной по формуле (28)

|  |  |
| --- | --- |
|  | (28) |

где – накладные расходы (руб.), – полная заработная плата (руб.), – норматив накладных расходов 20%.

Накладные расходы (29)

|  |  |
| --- | --- |
|  | (29) |

### 5.10 Итоговые затраты проведения работ

Сумма затрат на выполнение ВКР по статьям расходов представлена в таблице 9.

Таблица 9 – Смета затрат на ВКР

|  |  |  |
| --- | --- | --- |
| № | Наименование статьи | Сумма, руб. |
| 1 | Расходы на оплату труда | 58406,1 |
| 2 | Отчисления на социальные нужды | 17521,83 |
| 3 | Материалы | 3044 |
| 4 | Затраты по работам, выполняемым сторонними организациями | 1026,67 |
| 5 | Накладные расходы | 11681,22 |
| 6 | Амортизационные отчисления | 1143,71 |
| Итого затрат | | 92 823,53 |

Пояснение к статьям сметы:

* на статью “Расходы на оплату труда” относится заработная плата научных сотрудников, инженеров и прочего инженерно-технического персонала, непосредственно занятых выполнением работы;
* на статью “Отчисления на социальные нужды” относят затраты, связанные с выплатой социальных отчислений с заработной платы;
* на статью “Материалы” относятся затраты на сырье, основные и вспомогательные материалы, покупные полуфабрикаты и комплектующие изделия, необходимые для выполнения работы c учётом транспортно-заготовительных расходов;
* на статью “Затраты по работам, выполняемым сторонними организациями” относят затраты по оплате всех видов работ, выполняемые сторонними организациями;
* в статью “Накладные расходы” включаются расходы на управление и хозяйственное обслуживание. Величина накладных расходов определяется на основе норматива, установленного в конкретной организации, где производится выполнение работ;
* по статье “Амортизационные отчисления” учитываются амортизационные отчисления по всем видам основных средств, используемых при выполнении ВКР.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была достигнута поставленная цель – была успешно разработана библиотека на языке программирования Java для решения задачи о рюкзаке.

Были проанализированы существующие точные и эвристические алгоритмы решения задачи о рюкзаке, описана и реализована математическая модель для решения задачи. Разработаны, реализованы и протестированы на корректность алгоритмы решения задачи о рюкзаке путём сведения к задаче коммивояжёра.

Были проанализированы существующие библиотеки и пакеты для различных языков программирования, реализующих решения задачи о рюкзаке. Была разработана библиотека на языке программирования Java, реализующая рассматриваемые методы решения задачи о рюкзаке, в том числе методом сведения к задаче коммивояжёра.

Была проанализирована работа математической модели сведения задачи коммивояжёра к задаче о рюкзаке на различных наборах предметов случайного веса и стоимости, выявлены зависимости времени вычисления от количества предметов и грузоподъёмности рюкзака. Было получено, что методы решения задачи о рюкзаке путём сведения к задаче коммивояжёра обладают линейной зависимостью от количества предметов и зависимостью от грузоподъёмности рюкзака, соответствующей используемому методу решения задачи коммивояжёра. В результате анализа данных проведённых измерений было получено, что методы применимы на практике для малых значениях грузоподъёмности рюкзака при сравнительно высоких значениях количества предметов.

Для обеспечения корректной работы библиотеки были реализовано юнит-тестирование с использованием фреймворка JUnit. Полученное решение отличается от аналогов легковесностью, сравнительным разнообразием выбора решения задачи и наличием автоматизированного импорта и экспорта данных.

В дальнейшем планируется найти значения параметров задачи о рюкзаке для каждого из рассмотренных методов решения для составления критериев выбора метода с целью оптимизации времени решения.

На основе данной математической модели разработать алгоритм выбора метода решения задачи о рюкзаке в зависимости от свойств входных данных

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Коваль Юлия Вадимовна Применение задачи о рюкзаке для выбора местоположения торговой точки // StudNet. 2020. №8. URL: https://cyberleninka.ru/article/n/primenenie-zadachi-o-ryukzake-dlya-vybora-mestopolozheniya-torgovoy-tochki (дата обращения: 28.01.2024).
2. Laalaoui Y., M’Hallah R. A binary multiple knapsack model for single machine scheduling with machine unavailability // Computers & Operations Research. – 2016. – Т. 72. – С. 71-82.
3. Kolesar P. J. A branch and bound algorithm for the knapsack problem // Management science. – 1967. – Т. 13. – №. 9. – С. 723-735.
4. Al Etawi N. A., Aburomman F. T. 0/1 KNAPSACK PROBLEM: GREEDY VS. DYNAMIC-PROGRAMMING // Int J Adv Eng Manag Res. – 2020. – Т. 5. – №. 2. – С. 1-10.
5. Куприяшин М.А., Борзунов Г.И. Исследование алгоритма точного решения задачи о рюкзаке методом динамического программирования // Вестник ПНИПУ. Электротехника, информационные технологии, системы управления. 2016. №17. URL: https://cyberleninka.ru/article/n/issledovanie-algoritma-tochnogo-resheniya-zadachi-o-ryukzake-metodom-dinamicheskogo-programmirovaniya (дата обращения: 15.05.2024).
6. Yadav V., Singh S. ‘Genetic algorithms based approach to solve 0-1 Knapsack problem optimization problem //Int. J. Innov. Res. Comput. Commun. Eng. – 2016. – Т. 4. – №. 5. – С. 8595-8602.
7. OR-Tools - Google Optimization Tools // GitHub. URL: https://github.com/google/or-tools (дата обращения: 16.05.2024).
8. go-knapsack // GitHub. URL: https://github.com/mattschofield/go-knapsack (дата обращения: 16.05.2024).
9. [mknapsack](https://github.com/jmyrberg/mknapsack) // GitHub. URL: https://github.com/jmyrberg/mknapsack (дата обращения: 16.05.2024).
10. [knapsacksolver](https://github.com/fontanf/knapsacksolver) // GitHub. URL: https://github.com/fontanf/knapsacksolver (дата обращения: 16.05.2024).
11. [knapsack-pip](https://github.com/Alieladi/knapsack-pip) // GitHub. URL: https://github.com/Alieladi/knapsack-pip (дата обращения: 16.05.2024).
12. Sheveleva A. M., Belyaev S. A. Development of the Software for Solving the Knapsack Problem by Solving the Traveling Salesman Problem //2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). – IEEE, 2021. – С. 652-656.
13. Saiyed A. R. The traveling salesman problem // Indiana State University. – 2012. – Т. 2. – С. 1-15.
14. Костюк Юрий Леонидович Эффективная реализация алгоритма решения задачи коммивояжёра методом ветвей и границ // ПДМ. 2013. №2 (20).
15. Little J. D. C., Murty K. G., Sweeney D. W., and Karel C. An Algorithm for the Traveling Salesman Problem // Operations Research. 1963. No. 11. P. 972–989.
16. Канцедал Сергей Андреевич, Костикова Марина Владимировна Динамическое программирование для задачи коммивояжера // АСУ и приборы автоматики. 2014. №166. URL: https://cyberleninka.ru/article/n/dinamicheskoe-programmirovanie-dlya-zadachi-kommivoyazhera (дата обращения: 17.05.2024).
17. Lin S., Kernighan B.W. An effective heuristic algorithm for the traveling-salesman problem // Operations Research. 1973. Vol. 21. No. 2. P. 498–516.
18. Gutin G.,Y eo A. The Greedy Algorithm for the Symmetric TSP. University of London, 2002. 1–2.
19. Чусовлянкин А.А., Морозенко В.В. Анализ точности и времени решения задачи коммивояжера с помощью "антижадного" алгоритма // Вестник Пермского университета. Серия: Математика. Механика. Информатика. 2016. №4 (35).
20. Исходный код библиотеки, представленной в ВКР // GitHub. URL: https://github.com/ZhenjaMax/graduate-work-knapsack (дата обращения: 15.05.2024).
21. Алексеева О.Г. Экономическое обоснование ВКР: Учебно-методическое пособие, СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2023.

## ПРИЛОЖЕНИЕ А

В таблицах А.1-А.6 представлены экспериментальные данные по исследованию зависимости времени выполнения методов решения задачи о рюкзаке от количества предметов В таблицах представлено среднее квадратическое отклонение В данных экспериментах грузоподъёмность рюкзака постоянна,

Таблица А.1 – Экспериментальные данные для метода полного перебора

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0002 | 0,0006 | 15 | 3,4872 | 0,0734 |
| 4 | 0,0012 | 0,0014 | 16 | 7,1756 | 0,1622 |
| 5 | 0,0018 | 0,0007 | 17 | 14,9672 | 0,2241 |
| 6 | 0,0033 | 0,0005 | 18 | 30,9869 | 0,2874 |
| 7 | 0,0075 | 0,0007 | 19 | 63,2931 | 0,2804 |
| 8 | 0,017 | 0,001 | 20 | 130,8495 | 2,3131 |
| 9 | 0,0445 | 0,004 | 21 | 265,1889 | 2,8126 |
| 10 | 0,0967 | 0,0065 | 22 | 536,5095 | 2,7581 |
| 11 | 0,2049 | 0,01426 | 23 | 1098,2137 | 4,7193 |
| 12 | 0,4089 | 0,0218 | 24 | 2242,7334 | 7,6994 |
| 13 | 0,8304 | 0,0101 | 25 | 4607,4673 | 19,1352 |
| 14 | 1,7045 | 0,0602 | – | – | – |

Таблица А.2 – Экспериментальные данные для метода ветвей и границ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0004 | 0,0007 | 15 | 0,1225 | 0,0512 |
| 4 | 0,001 | 0,0009 | 16 | 0,2027 | 0,1162 |
| 5 | 0,0027 | 0,0016 | 17 | 0,2238 | 0,0818 |
| 6 | 0,0045 | 0,0017 | 18 | 0,2618 | 0,1694 |
| 7 | 0,0078 | 0,0035 | 19 | 0,3103 | 0,1803 |
| 8 | 0,0128 | 0,0098 | 20 | 0,5129 | 0,2662 |

Продолжение таблицы А.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 9 | 0,0174 | 0,0051 | 21 | 0,7205 | 0,4571 |
| 10 | 0,02 | 0,0057 | 22 | 0,6909 | 0,5118 |
| 11 | 0,0405 | 0,0186 | 23 | 0,8448 | 0,5625 |
| 12 | 0,0672 | 0,0234 | 24 | 0,8887 | 0,4253 |
| 13 | 0,0676 | 0,0368 | 25 | 1,169 | 1,2703 |
| 14 | 0,0815 | 0,0393 | – | – | – |

Таблица А.3 – Экспериментальные данные для метода динамического программирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0005 | 0,0004 | 15 | 0,0026 | 0,0005 |
| 4 | 0,0006 | 0,0006 | 16 | 0,0025 | 0,0004 |
| 5 | 0,0013 | 0,0004 | 17 | 0,0022 | 0,0004 |
| 6 | 0,0013 | 0,0004 | 18 | 0,0024 | 0,0004 |
| 7 | 0,0015 | 0,0004 | 19 | 0,0029 | 0,0004 |
| 8 | 0,0012 | 0,0004 | 20 | 0,0033 | 0,0004 |
| 9 | 0,0012 | 0,0004 | 21 | 0,0036 | 0,0005 |
| 10 | 0,0022 | 0,0025 | 22 | 0,0032 | 0,0004 |
| 11 | 0,0017 | 0,0008 | 23 | 0,0033 | 0,0004 |
| 12 | 0,0023 | 0,0004 | 24 | 0,0034 | 0,0005 |
| 13 | 0,003 | 0,0021 | 25 | 0,004 | 0,0005 |
| 14 | 0,0032 | 0,002 | – | – | – |

Таблица А.4 – Экспериментальные данные для сведения к задаче коммивояжёра методом полного перебора

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,4834 | 0,221 | 15 | 1,1184 | 0,3543 |
| 4 | 0,5491 | 0,255 | 16 | 1,1359 | 0,2668 |
| 5 | 0,5507 | 0,1591 | 17 | 1,1964 | 0,2415 |
| 6 | 0,6212 | 0,3077 | 18 | 1,247 | 0,1366 |

Продолжение таблицы А.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 | 0,6806 | 0,4013 | 19 | 1,294 | 0,033 |
| 8 | 0,7914 | 0,2262 | 20 | 1,3404 | 0,1329 |
| 9 | 0,7705 | 0,4387 | 21 | 1,3268 | 0,095 |
| 10 | 0,8473 | 0,2383 | 22 | 1,2954 | 0,0151 |
| 11 | 0,977 | 0,2757 | 23 | 1,3801 | 0,1723 |
| 12 | 1,1268 | 0,2383 | 24 | 1,3693 | 0,2232 |
| 13 | 1,0198 | 0,3957 | 25 | 1,3708 | 0,2238 |
| 14 | 1,1599 | 0,259 | – | – | – |

Таблица А.5 – Экспериментальные данные для сведения к задаче коммивояжёра модифицированным алгоритмом Литтла

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 49,8358 | 50,9207 | 15 | 669,5086 | 507,5393 |
| 4 | 163,5863 | 119,2224 | 16 | 753,0797 | 604,7758 |
| 5 | 492,0969 | 333,3535 | 17 | 181,401 | 183,0243 |
| 6 | 353,7114 | 193,1314 | 18 | 2183,4175 | 1985,0046 |
| 7 | 509,583 | 550,8311 | 19 | 374,6108 | 375,0691 |
| 8 | 586,5434 | 154,6899 | 20 | 1505,2593 | 1598,0539 |
| 9 | 457,195 | 167,0373 | 21 | 659,817 | 588,0045 |
| 10 | 1759,6417 | 160,9691 | 22 | 1649,5067 | 1800,4795 |
| 11 | 1200,5828 | 1181,8488 | 23 | 255,8504 | 270,1222 |
| 12 | 737,0553 | 376,5671 | 24 | 940,4221 | 1030,4554 |
| 13 | 441,9159 | 271,8608 | 25 | 582,2083 | 600,1536 |
| 14 | 2113,1445 | 1130,6754 | – | – | – |

Таблица А.6 – Экспериментальные данные для сведения к задаче коммивояжёра динамическим методом Хелда-Карпа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 2,8581 | 0,1081 | 15 | 2,988 | 0,2773 |
| 4 | 2,9243 | 0,1885 | 16 | 3,003 | 0,4517 |
| 5 | 2,9234 | 0,2468 | 17 | 3,073 | 0,5161 |
| 6 | 2,9518 | 0,2053 | 18 | 3,0293 | 0,2767 |
| 7 | 2,934 | 0,1639 | 19 | 3,1005 | 0,418 |
| 8 | 2,9251 | 0,1962 | 20 | 3,1398 | 0,4142 |
| 9 | 2,9463 | 0,2493 | 21 | 3,1724 | 0,8111 |
| 10 | 2,9576 | 0,262 | 22 | 3,0987 | 0,3885 |
| 11 | 2,9588 | 0,2162 | 23 | 3,0928 | 0,491 |
| 12 | 2,904 | 0,1546 | 24 | 3,1158 | 0,2892 |
| 13 | 2,9675 | 0,2704 | 25 | 3,2323 | 0,2743 |
| 14 | 2,97 | 0,2883 | – | – | – |

## ПРИЛОЖЕНИЕ Б

В таблицах Б.1-Б.6 представлены экспериментальные данные по исследованию зависимости времени выполнения методов решения задачи о рюкзаке от грузоподъёмности рюкзака В таблицах представлено среднее квадратическое отклонение В данных экспериментах количество предметов постоянно, .

Таблица Б.1 – Экспериментальные данные для метода полного перебора

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0201 | 0,0027 | 7 | 0,0211 | 0,0037 |
| 4 | 0,0191 | 0,0015 | 8 | 0,0166 | 0,0013 |
| 5 | 0,0191 | 0,0009 | 9 | 0,016 | 0,0012 |
| 6 | 0,0195 | 0,0049 | 10 | 0,0203 | 0,0013 |

Таблица Б.2 – Экспериментальные данные для метода ветвей и границ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0284 | 0,0051 | 7 | 0,0297 | 0,0073 |
| 4 | 0,0313 | 0,0208 | 8 | 0,033 | 0,0136 |
| 5 | 0,0336 | 0,0224 | 9 | 0,0262 | 0,0159 |
| 6 | 0,03 | 0,0136 | 10 | 0,0211 | 0,0094 |

Таблица Б.3 – Экспериментальные данные для метода динамического программирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0004 | 0,0073 | 7 | 0,0008 | 0,0002 |
| 4 | 0,0006 | 0,0110 | 8 | 0,0009 | 0,0005 |
| 5 | 0,0007 | 0,0001 | 9 | 0,0011 | 0,0003 |
| 6 | 0,0006 | 0,0001 | 10 | 0,0011 | 0,0003 |

Таблица Б.4 – Экспериментальные данные для сведения к задаче коммивояжёра методом полного перебора

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0068 | 0,0044 | 7 | 0,4775 | 0,2695 |
| 4 | 0,0091 | 0,003 | 8 | 0,8417 | 0,6222 |
| 5 | 0,0241 | 0,0234 | 9 | 7,8094 | 3,9184 |
| 6 | 0,1529 | 0,142 | 10 | 65,2542 | 15,346 |

Таблица Б.5 – Экспериментальные данные для сведения к задаче коммивояжёра модифицированным алгоритмом Литтла

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 1,8598 | 1,9543 | 7 | 250,46141 | 85,0526 |
| 4 | 3,5293 | 2,2412 | 8 | 4221,4004 | 344,155 |
| 5 | 11,405 | 12,0526 | 9 | – | – |
| 6 | 38,0878 | 34,994 | 10 | – | – |

Таблица Б.6 – Экспериментальные данные для сведения к задаче коммивояжёра динамическим методом Хелда-Карпа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | , мс | , мс |  | , мс | , мс |
| 3 | 0,0145 | 0,0058 | 7 | 2,981 | 0,1903 |
| 4 | 0,0302 | 0,0042 | 8 | 13,901 | 0,6031 |
| 5 | 0,1411 | 0,0048 | 9 | 65,362 | 3,4543 |
| 6 | 0,6252 | 0,0077 | 10 | 320,4324 | 9,4816 |