|  |  |
| --- | --- |
|  | **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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ДИСЦИПЛИНА «Анализ алгоритмов»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 7**

|  |  |
| --- | --- |
| **Тема «**Поиск в словаре»  **Студент**  Чалый А.А.  **Группа** ИУ7 – 52Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватели** Волкова Л.Л.  Строганов Ю.В. |  |

Москва.

2020 г.

**Оглавление**

[Введение 3](#_Toc1)

[1. Аналитическая часть 4](#_Toc2)

[1.1 Описание задачи 4](#_Toc3)

[1.2 Метод полного перебора 4](#_Toc4)

[1.3 Муравьиный алгоритм 5](#_Toc5)

[Вывод 8](#_Toc6)

[2. Конструкторская часть 9](#_Toc7)

[2.1 Схемы алгоритмов 9](#_Toc8)

[Вывод 12](#_Toc9)

[3.Технологическая часть 14](#_Toc10)

[3.1 Требования к программному обеспечению 14](#_Toc11)

[3.2 Средства реализации 14](#_Toc12)

[Вывод 19](#_Toc13)

[4. Экспериментальная часть 20](#_Toc14)

[4.1 Примеры работы 20](#_Toc15)

[4.2 Параметризация метода 21](#_Toc16)

[4.3 Сравнительный анализ на материале экспериментальных данных 22](#_Toc17)

[Вывод 23](#_Toc18)

[Заключение 24](#_Toc19)

[Список использованных источников 25](#_Toc20)

## Введение

Целью данной работы является изучение и применение на практике алгоритма поиска элемента в массиве словарей по ключу. В данной лабораторной работе рассматривается алгоритм перебора, алгоритм бинарного поиска и сегментный поиск с частным анализом данных.

Поиск — обработка некоторого множества данных с целью выявления подмножества данных, соответствующего критериям поиска. [1]

Задачи работы:

1. Реализовать алгоритм полного перебора.
2. Реализовать алгоритм бинарного поиска.
3. Реализовать алгоритм сегментного поиска, основанного на результатах частного анализа.
4. Сравнить алгоритмы по затраченным ресурсам.

# 1. Аналитическая часть

В данном разделе содержится описание методов поиска в словаре.

## 1.1 Поиск полным перебором

Алгоритм заключается в том, что для поиска заданного элемента из множества, происходит непосредственное сравнивание каждого элемента этого множества с искомым, до тех пор, пока искомый не найдется или множество закончится.

## 1.2 Двоичный поиск в упорядоченном словаре

Двоичный поиск заключается в том, что на каждом шаге множество объектов делится на две части и в работе остаётся та часть множества, где находится искомый объект.

## 1.3 Сегментный поиск с частным анализом

Сегментный поиск с частным анализом заключается в том, что множество делится на подмножества (сегменты), затем сегменты сортируются по частоте обращения (использования). Это позволяет снизить трудоемкость у часто вызываемых элементов. Для данного метода необходима первичная обработка данных, которая требует дополнительных ресурсов.

## Вывод

В данном разделе были описаны такие методы поиска в словаре как: поиск полным перебором, двоичный поиск в упорядоченном словаре и сегментный поиск с частным анализом.

# 2. Конструкторская часть

Описание индивидуального задания: множества представляют собой массив словарей, которые состоят из Id доставщика и времени доставки. На вход алгоритмы принимают ключ. На выходе алгоритм выдает словарь, соответствующий ключу. Далее в данном разделе будут рассмотрены схемы алгоритмов и описан частный анализ данных для задачи сегментного поиска.

## 2.1 Схемы алгоритмов

На рис. 1 приведена схема алгоритма полного перебора.

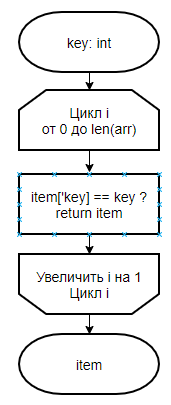


Рисунок 1. Алгоритма полного перебора

На рис. 2 приведена схема алгоритма бинарного поиска в упорядоченном множестве.

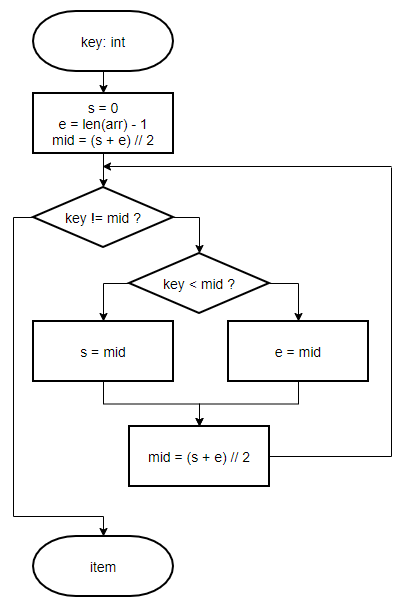


Рисунок 2. Схема алгоритма бинарного поиска в упорядоченном множестве

Схема алгоритма сегментного поиска с частным анализом ничем не отличается от алгоритма полного перебора, так как разница лишь в том, что в данном алгоритме происходит предобработка данных.

## 2.2 Частный анализ данных

Данные разбиваются на сегменты, каждому сегменты присваивается своя вероятность. Сегменты с наивысшей вероятностью ставятся в начало массива.

## Вывод

В данном разделе было описано индивидуальное задание, рассмотрены схемы алгоритмов и описан частный анализ данных для задачи сегментного поиска.

# 3.Технологическая часть

В данном разделе будут рассмотрены требования к программному обеспечению, средства реализации и представлен листинг кода, также будет проведено тестирование по методу черного ящика.

## 3.1 Требования к программному обеспечению

Входные данные: Ключ.

Выходные данные: Словарь по данному ключу.

Функциональная схема решения задачи поиска в словаре IDEF0 представлена на рис. 3.

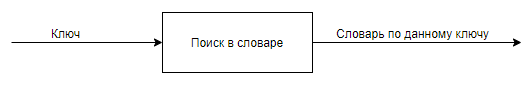


Рисунок 3. Функциональная схема решения задачи поиска в словаре

## 3.2 Средства реализации

В качестве языка программирования был выбран python т.к. данный язык часто использовался мною. Данный язык позволяет писать высокоуровневый абстрактный код.

Среда разработки – PyCharm, которая предоставляет умную проверку кода, быстрое выявление ошибок и оперативное исправление, вкупе с автоматическим рефакторингом кода и богатыми возможностями в навигации.

Время работы было замерено с помощью функции process\_time() из библиотеки time . [2]

**3.3 Листинг кода**

В данном пункте представлен листинг кода функций. В листинге 1 представлена реализация алгоритма полного перебора. В листинге 2 представлена реализация алгоритма бинарного поиска. В листинге 3 представлена реализация сегментного алгоритма с частным анализом.

Листинг 1. Реализация алгоритма полного перебора

**def** brute**(**mas**,** key**):**

**for** d **in** mas**:**

**if** d**[**'key'**]** **==** key**:**

**return** d

**return** "No such deliveryman"

Листинг 2. Реализация алгоритма бинарного поиска

**def** binary**(**mas**,** key**):**

s **=** 0

e **=** **len(**mas**)** **-** 1

mid **=** **(**s **+** e**)** **//** 2

**if** mas**[**s**][**'key'**]** **>** key**:**

**return** "No such deliveryman"

**elif** mas**[**e**][**'key'**]** **<** key**:**

**return** "No such deliveryman"

**if** mas**[**s**][**'key'**]** **==** key**:**

**return** mas**[**s**]**

**elif** mas**[**e**][**'key'**]** **==** key**:**

**return** mas**[**e**]**

tmp **=** mas**[**mid**][**'key'**]**

**while** key **!=** tmp**:**

**if** key **<** tmp**:**

e **=** mid

**else:**

s **=** mid

mid **=** **(**s **+** e**)** **//** 2

tmp **=** mas**[**mid**][**'key'**]**

**return** mas**[**mid**]**

Листинг 3. Реализация сегментного алгоритма с частным анализом

**def** segalg**(**mas**,** key**,** seg\_cnt**):**

pmas **=** preproc**(**seg\_cnt**)**

**for** d **in** mas**:**

**if** d**[**'key'**]** **==** key**:**

**return** d

**return** "No such deliveryman"

**def** preproc**(**seg\_cnt**):**

p **=** 100 **/** **((**1 **+** seg\_cnt**)** **\*** seg\_cnt **//** 2**)** **/** 100

pmas **=** **[]**

**for** i **in** **range(**seg\_cnt**):**

pmas**.**append**((**seg\_cnt **-** i**)** **\*** p**)**

**for** i **in** **range(**1**,** seg\_cnt**):**

pmas**[**i**]** **+=** pmas**[**i **-** 1**]**

**return** pmas

**3.4 Тестирование алгоритмов**

В тестах будет рассмотрена проверка работы алгоритмов при:

* несуществующем ключе;
* ключе, находящимся на границе;
* нормальном случае.

Тестирование производится методом черного ящика.

На рисунке ниже будет приведены результат тестирования алгоритмов с целью демонстрации корректности работы программы.



Рисунок 4. Результат тестирования алгоритмов

## Вывод

В данном разделе были рассмотрены требования к программному обеспечению, средства реализации и представлен листинг кода, также было проведено тестирование по методу черного ящика.

# 4. Экспериментальная часть

В данном разделе будет проведен эксперимент и сравнительный анализ полученных данных.

## 4.1 Сравнительный анализ на материале экспериментальных данных

В рамках данной части был проведен эксперимент, описанный ниже.

Сравнение времени работы алгоритма полного перебора, бинарного поиска, сегментного алгоритма с частным анализом (при 5 и 10 сегментах). Создаются массивы размерностью 10000. Эксперимент по замеру времени ставился 10000 раз.

На рис. 10 представлено Среднее время работы алгоритмов.

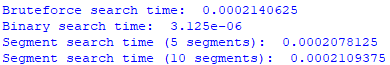


Рисунок 5. Среднее время работы алгоритмов

Как можно наблюдать на рис. 5 бинарный поиск является самым быстрым алгоритмом поиска. Алгоритм Сегментного поиска с частным анализом немного обгоняет алгоритм полного перебора, однако стоит учесть, что дополнительная посегментная сортировка также требует время. Если в программе требуется частое обращение к некоторым частям данных, то целесообразно проанализировать их и использовать сегментный поиск.

## Вывод

В данном разделе был проведен эксперимент и сравнительный анализ полученных данных в ходе которого было выявлено, что алгоритм бинарного поиска является самым быстрым, однако алгоритм полного перебора также возможен в использовании если известно к каким частям данных требуется обращаться чаще.

# Заключение

В ходе работы были изучены и реализованы алгоритмы поиска в массиве словаря по ключу. Было выявлено, что бинарный поиск является самым быстрым, однако алгоритм полного перебора также возможен в использовании если известно к каким частям данных требуется обращаться чаще.. Все поставленные задачи были выполнены. Целью лабораторной работы являлось поиск словаря по ключу, что также было достигнуто.

# Список использованных источников

1. Алгоритмы сортировки и поиска. [электронный ресурс]. Режим доступа: https://prog-cpp.ru/algorithm-sort/, свободный (Дата обращения: 11.12.20)
2. time – Time access and conversions: сайт. – URL: [https://docs.python.org/3/library/time.html](https://docs.python.org/3/library/time.html(дата) (дата обращения 16.09.2020). – Текст: электронный.