



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н. Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н. Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

---

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

## ОТЧЕТ

по лабораторной работе № 7  
по курсу «Анализ Алгоритмов»  
на тему: «Алгоритмы поиска»

Студент ИУ7-51Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

А. А. Кузин  
(И. О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

Л. Л. Волкова  
(И. О. Фамилия)

2023 г.

# Содержание

<b>ВВЕДЕНИЕ</b>	<b>3</b>
<b>1 Аналитическая часть</b>	<b>4</b>
1.1 Словари . . . . .	4
1.2 Алгоритм полного перебора . . . . .	4
<b>2 Конструкторская часть</b>	<b>6</b>
2.1 Описание используемых типов данных . . . . .	6
2.2 Разработка алгоритмов . . . . .	6
<b>3 Технологическая часть</b>	<b>7</b>
3.1 Средства реализации . . . . .	7
3.2 Сведения о модулях программы . . . . .	7
3.3 Реализация алгоритмов . . . . .	8
<b>4 Исследовательская часть</b>	<b>9</b>
4.1 Технические характеристики . . . . .	9
4.2 Демонстрация работы программы . . . . .	9
4.3 Временные характеристики . . . . .	10
4.4 Вывод . . . . .	11
<b>ЗАКЛЮЧЕНИЕ</b>	<b>12</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>13</b>

# ВВЕДЕНИЕ

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

Целью данной лабораторной работы является сравнение производительности работы с обычным словарем и сегментированным словарем методом полного перебора.

Для поставленной цели необходимо выполнить следующие задачи:

- 1) изучить алгоритмы поиска в словарях;
- 2) описать алгоритмы в виде псевдокода;
- 3) реализовать алгоритм поиска значения по ключу в стандартном и сегментированном словарях;
- 4) провести сравнительный алгоритма поиска в стандартном и сегментированном словарях;
- 5) описать и обосновать полученные результаты.

# 1 Аналитическая часть

В этом разделе будут рассмотрена информация, касающаяся поиска значений в словарях.

## 1.1 Словари

Словарь [1] — абстрактный тип данных (интерфейс к хранилищу данных), позволяющий хранить пары вида «(ключ, значение)» и поддерживающий операции добавления пары, а также поиска и удаления пары по ключу:

- 1)  $insert(k, v)$ ;
- 2)  $find(k)$ ;
- 3)  $remove(k)$ .

В паре  $(k, v)$ :  $v$  называется значением, ассоциированным с ключом  $k$ . Где  $k$  — это ключ, а  $v$  — значение. Семантика и названия вышеупомянутых операций в разных реализациях ассоциативного массива могут отличаться.

Операция поиска  $find(k)$  возвращает значение, ассоциированное с заданным ключом, или некоторый специальный объект, означающий, что значения, ассоциированного с заданным ключом, нет. Две другие операции ничего не возвращают (за исключением, возможно, информации о том, успешно ли была выполнена данная операция).

Словарь с точки зрения интерфейса удобно рассматривать как обычный массив, в котором в качестве индексов можно использовать не только целые числа, но и значения других типов — например, строки (именно по этой причине словарь также иногда называют «ассоциативным массивом»).

## 1.2 Алгоритм полного перебора

Алгоритмом полного перебора [2] называют метод решения задачи, при котором по очереди рассматриваются все возможные варианты. В случае реализации алгоритма в рамках данной работы будут последовательно перебираться ключи словаря до тех пор, пока не будет найден нужный.

Трудоёмкость алгоритма зависит от того, присутствует ли искомый ключ в словаре, и, если присутствует — насколько он далеко от начала массива

ключей. Пусть на старте алгоритм затрагивает  $k_0$  операций, а при сравнении  $k_1$  операций.

Пусть алгоритм нашёл элемент на первом сравнении (лучший случай), тогда будет затрачено  $k_0 + k_1$  операций, на втором —  $k_0 + 2 \cdot k_1$ , на последнем (худший случай) —  $k_0 + N \cdot k_1$ . Если ключа нет в массиве ключей, то мы сможем понять это, только перебрав все ключи, таким образом трудоёмкость такого случая равно трудоёмкости случая с ключом на последней позиции. Трудоёмкость в среднем может быть рассчитана как математическое ожидание по формуле (1.1), где  $\Omega$  — множество всех возможных случаев.

$$\sum_{i \in \Omega} p_i \cdot f_i = k_0 + k_1 \cdot \left( 1 + \frac{N}{2} - \frac{1}{N+1} \right) \quad (1.1)$$

## Вывод

В данном разделе были рассмотрены словарь как структура данных и алгоритм полного перебора.

## 2 Конструкторская часть

В этом разделе будут представлено описание используемых типов данных, а также псевдокод алгоритма поиска в словаре.

К программе предъявлен ряд функциональных требований:

- иметь интерфейс ввода вопроса;
- работа с словарями и строками;

### 2.1 Описание используемых типов данных

При реализации алгоритмов будут использован словарь — встроенный тип `dict` в Python:

### 2.2 Разработка алгоритмов

В листинге 2.1 представлен псевдокод алгоритма поиска в словаре полным перебором.

Листинг 2.1 – Алгоритм поиска в словаре полным перебором

```
1  input: dictionary dict, string key
2  output: value
3
4  for each dict.key
5  begin
6    if dict[key] = key then
7    begin
8      return dict[key].value
9    end
10 end
11 return "no string found"
```

## Вывод

В данном разделе были представлены требования к программе, представлен псевдокод алгоритма поиска значения в словаре.

## 3 Технологическая часть

В данном разделе будут приведены средства реализации, листинг кода и функциональные тесты.

### 3.1 Средства реализации

В данном разделе рассмотрены средства реализации, а также представлены листинги реализаций алгоритма расчета термовой частота для всех термов из выборки документов.

В данной работе для реализации был выбран язык программирования *Python*. Данный выбор обусловлен соответствием с требованиями выдвинутыми в конструкторской части, а именно в языке программирование *Python* имеется встроенные типы данных — словарь и массив и инструменты для поиска подстроки в строке.

### 3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

- `main.py` — точка входа программы, пользовательское меню;
- `dict.py` — модуль с основными операциями над словарями;

### 3.3 Реализация алгоритмов

В листингах 3.1 – 3.2 приведены реализации алгоритмов поиска значения по ключу в обычном словаре и в сегментированном словаре.

Листинг 3.1 – Реализация алгоритма поиска в обычном словаре

```
1 def find_word(word_to_find, words_dicts):
2     start = timeit.default_timer()
3     val = "-1"
4     val = words_dicts.get(word_to_find, "-1")
5     stop = timeit.default_timer()
6
7     return val, (stop - start)
```

Листинг 3.2 – Реализация алгоритма поиска в сегментированном словаре

```
1 def find_word_seg(word_to_find, words_dicts):
2     segm = words_dicts.get(word_to_find[0].lower())
3     val = "-1"
4     if(segm != None):
5         start = timeit.default_timer()
6         val = segm.get(word_to_find, "-1")
7         stop = timeit.default_timer()
8     else:
9         stop, start = 0, 0
10
11
12     return val, (stop - start)
```

### Вывод

Были выбраны язык программирования и среда разработки, приведены сведения о модулях программы, листинги алгоритма.



## 4 Исследовательская часть

### 4.1 Технические характеристики

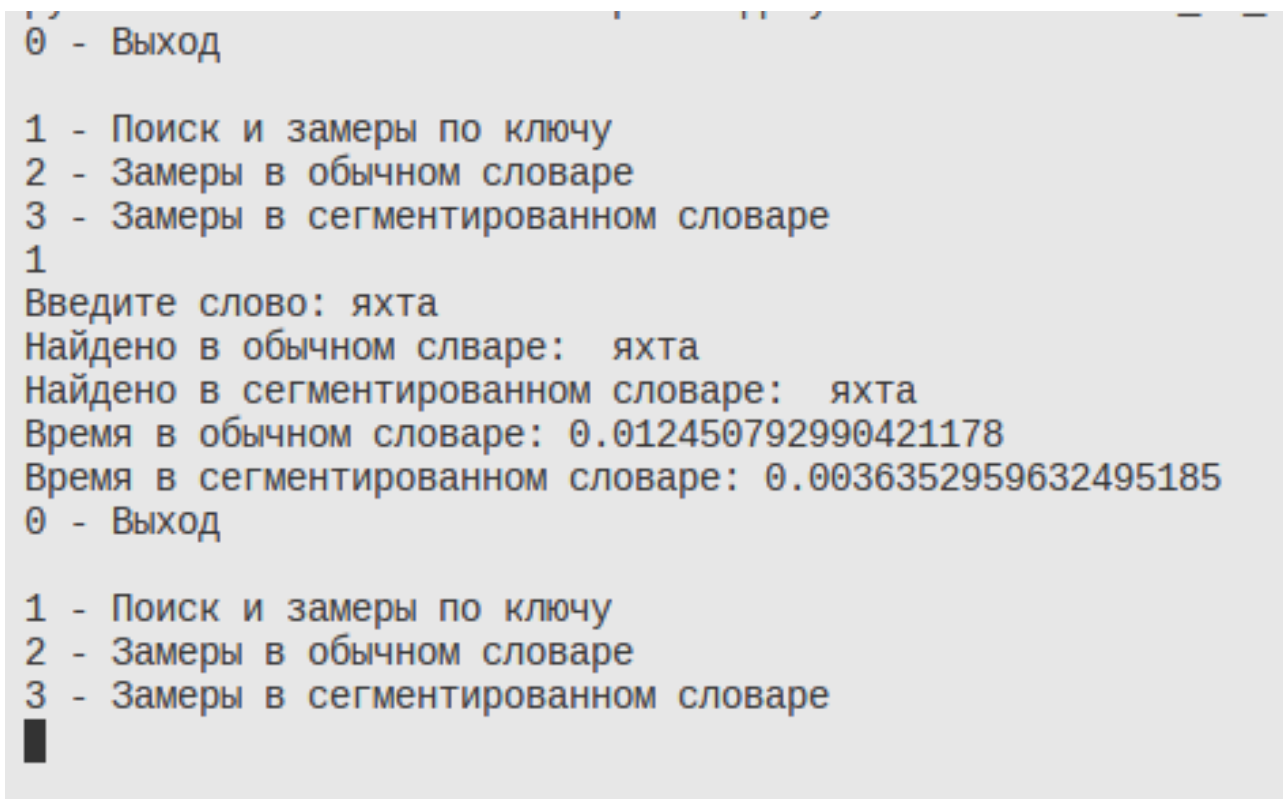
Технические характеристики устройства, на котором выполнялись замеры по времени:

- Процессор: Intel Core i7 9750H 2.6 ГГц;
- Оперативная память: 16 ГБ;
- Операционная система: Kubuntu 22.04.3 LTS x86\_64 Kernel: 6.2.0-36-generic

Во время проведения измерений времени ноутбук был подключен к сети электропитания и был нагружен только системными приложениями.

### 4.2 Демонстрация работы программы

На рисунке 4.1 показан пример работы с программой.



```
0 - Выход

1 - Поиск и замеры по ключу
2 - Замеры в обычном словаре
3 - Замеры в сегментированном словаре
1
Введите слово: яхта
Найдено в обычном слваре:  яхта
Найдено в сегментированном словаре:  яхта
Время в обычном словаре: 0.012450792990421178
Время в сегментированном словаре: 0.0036352959632495185
0 - Выход

1 - Поиск и замеры по ключу
2 - Замеры в обычном словаре
3 - Замеры в сегментированном словаре
```

Рисунок 4.1 – Демонстрация работы с программой

### 4.3 Временные характеристики

Исследование временных характеристик реализованных алгоритмов производилось на следующих запросах:

- 1) случайное слово из словаря;
- 2) первое слово из последнего сегмента;
- 3) последнее слово из первого сегмента.

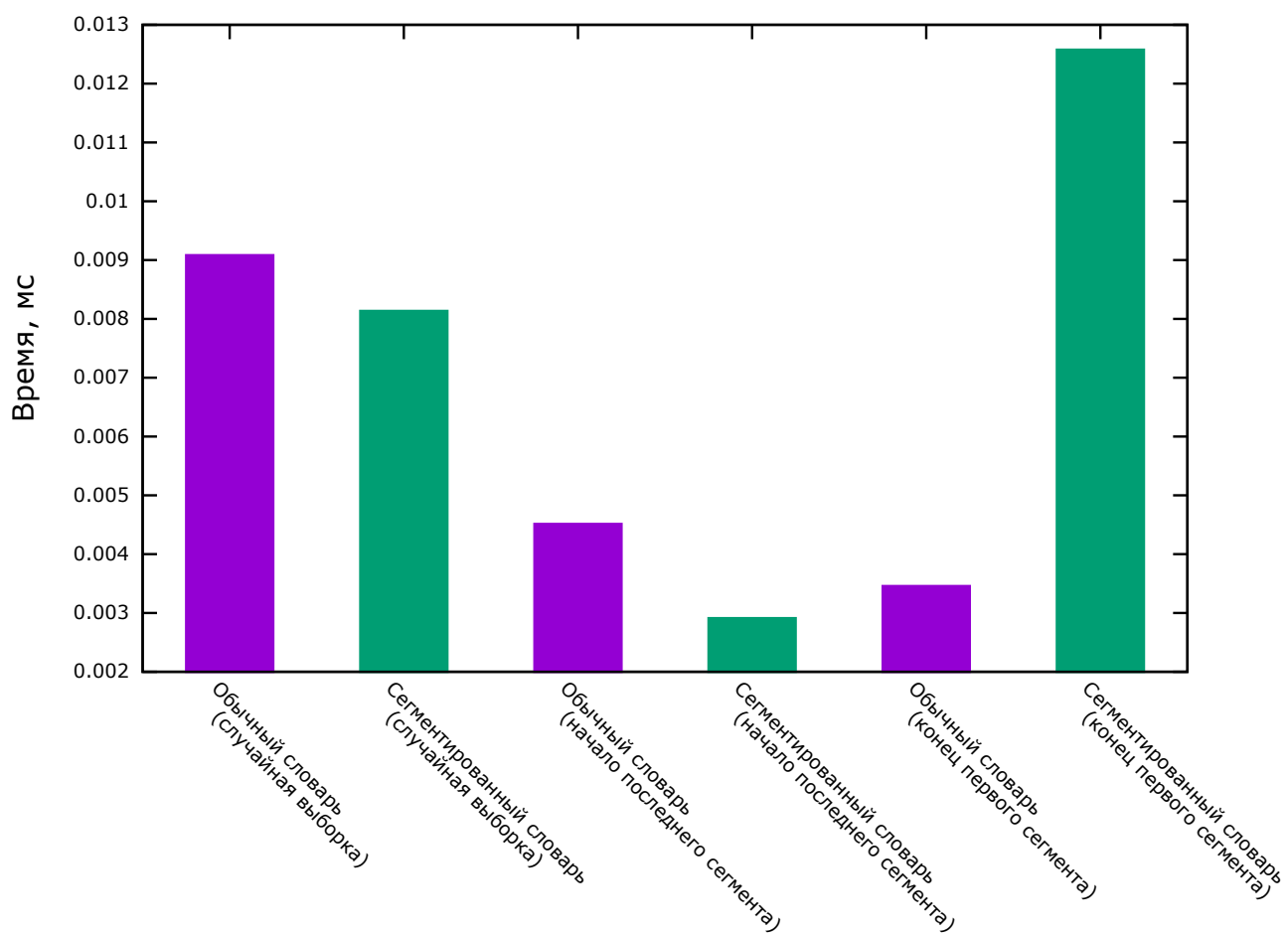


Рисунок 4.2 – Результат измерений времени работы (в мс) при разных запросах

## 4.4 Вывод

В среднем, использование сегментированного словаря занимает меньше времени, по сравнению с обычным словарем. В частности, если ключ ближе к последнему сегменту выигрыш сегментированного словаря становится в 1.5 раза больше. Однако, если поиск стремится к первому сегменту, то выполнение поиска в обычном словаре становится выгоднее.

## ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы поставленная ранее цель была достигнута: были изучены принципы поиска по обычному и сегментированному словарю.

В среднем, использование сегментированного словаря занимает меньше времени, чем использование обычного словаря. Особенно, если ключ ближе к последнему сегменту, выгода от использования сегментированного словаря увеличивается в 1.5 раза. Однако, если поиск направлен к первому сегменту, выполнение поиска в обычном словаре становится более выгодным.

В ходе выполнения лабораторной работы были решены следующие задачи:

- 1) изучены алгоритмы поиска в словарях;
- 2) описаны алгоритмы в виде псевдокода;
- 3) реализованы алгоритмы поиска значения по ключу в стандартном и сегментированном словарях;
- 4) проведен сравнительный анализ алгоритма поиска в стандартном и сегментированном словарях;
- 5) описаны и обоснованы полученные результаты.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *С. Ш.* Словари — Лаборатория линуксоида [Электронный ресурс]. — — Режим доступа: <https://younglinux.info/python/dictionary> (дата обращения: 28.01.2023).
2. *Н. Н.* Искусственный интеллект. Методы поиска решений. — М.: Мир, 1973. — С. 273.