



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»

# Отчёт

## по лабораторной работе № 7

Название: Поиск по словарю

Дисциплина: Анализ алгоритмов

Студент

ИУ7-55Б

(Группа)

Д.О. Склифасовский

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Л.Л. Волкова

(Подпись, дата)

(И.О. Фамилия)

*Москва, 2020*

## Содержание

<b>Введение</b>	<b>3</b>
<b>1 Аналитический раздел</b>	<b>4</b>
1.1 Общие сведения . . . . .	4
1.2 Алгоритм полного перебора . . . . .	4
1.3 Алгоритм двоичного поиска . . . . .	5
1.4 Алгоритм поиска по сегментам . . . . .	5
<b>2 Конструкторский раздел</b>	<b>6</b>
2.1 Разработка алгоритмов . . . . .	6
2.2 Вывод . . . . .	8
<b>3 Технологический раздел</b>	<b>9</b>
3.1 Общие требования . . . . .	9
3.2 Средства реализации . . . . .	9
3.3 Сведения о модулях программы . . . . .	9
3.4 Листинг кода программы . . . . .	10
3.5 Вывод . . . . .	12
<b>4 Экспериментальный раздел</b>	<b>13</b>
4.1 Описание экспериментов . . . . .	13
4.2 Примеры работы программы . . . . .	13
4.3 Вывод . . . . .	14
<b>5 Вывод</b>	<b>15</b>
<b>Литература</b>	<b>16</b>

## **Введение**

Цель работы: изучение алгоритмов поиска слов в словаре

В ходе лабораторной работы требуется:

- 1) описать алгоритм полного перебора;
- 2) описать алгоритм двоичного поиска;
- 3) описать алгоритм поиска слов по сегментам;
- 4) реализовать 3 алгоритма поиска по словарю;
- 5) провести замеры времени работы алгоритмов.

## **1 Аналитический раздел**

В данном разделе представлено описание трех выбранных алгоритмов.

### **1.1 Общие сведения**

Словарь — книга или любой другой источник, информация в котором упорядочена с помощью разбивки на небольшие статьи, отсортированные по названию или тематике. Различают энциклопедические и лингвистические словари. С развитием компьютерной техники всё большее распространение получают электронные словари и онлайн-словари. Первым русским словарём принято считать Азбуковник, помещённый в списке Кормчей книги 1282 года и содержащий 174 слова. Задача состоит в поиске слов из словаря в случайных данных любого размера(напр. в файле). Поскольку словарь меняется редко, то можно его подготовить (напр. отсортировать, создать дерево итд). Это зависит от алгоритма поиска, который будет использован.

### **1.2 Алгоритм полного перебора**

Алгоритм полного перебора — это алгоритм разрешения математических задач, который можно отнести к классу способов нахождения решения рассмотрением всех возможных вариантов. Полный перебор (или метод «грубой силы», англ. brute force) — метод решения математических задач. Относится к классу методов поиска решения исчерпыванием всевозможных вариантов. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий.

В нашем же случае следует перебирать слова циклом в словаре, пока не встретится нужное слово.

### **1.3 Алгоритм двоичного поиска**

Целочисленный двоичный поиск (бинарный поиск) (англ. binary search) — алгоритм поиска объекта по заданному признаку в множестве объектов, упорядоченных по тому же самому признаку, работающий за логарифмическое время.

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. Или же, в зависимости от постановки задачи, мы можем остановить процесс, когда мы получим первый или же последний индекс вхождения элемента. Последнее условие — это левосторонний/правосторонний двоичный поиск.

### **1.4 Алгоритм поиска по сегментам**

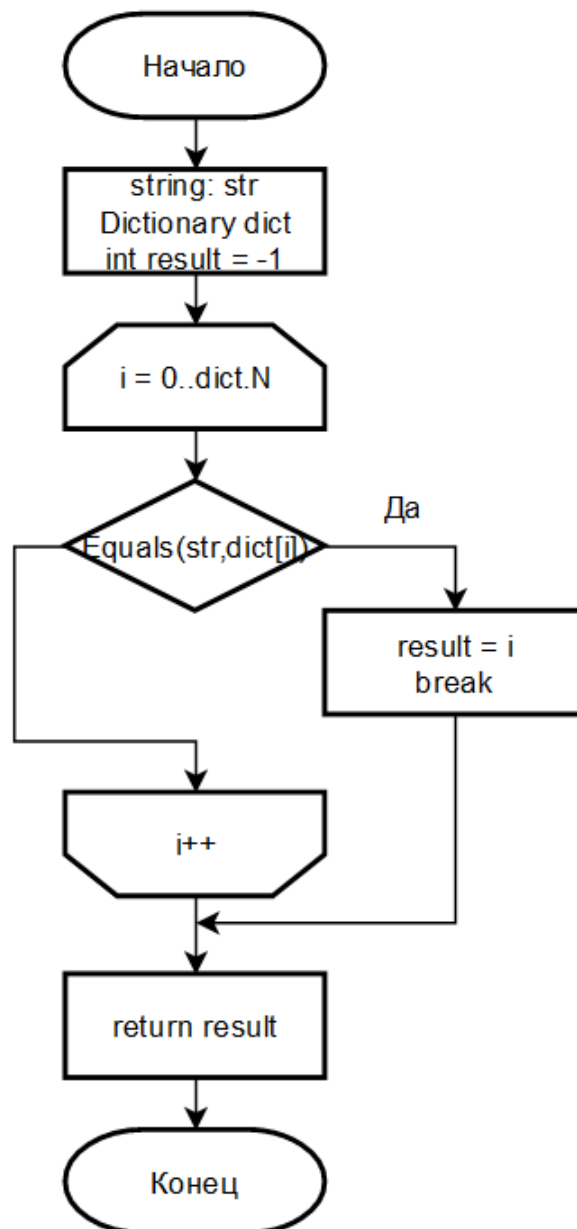
Суть данного алгоритма заключается в том, что необходимо разбить словарь на сегменты. Каждый сегмент определяет первую букву слов, которые находятся в нем. Для того, чтобы найти в словаре нужно сначала проходить по сегментам, когда найден нужный - искать слово в нем.

## 2 Конструкторский раздел

В данном разделе представлены схемы разработанных алгоритмов. Также описывается часть алгоритма, которая будет распараллеливаться.

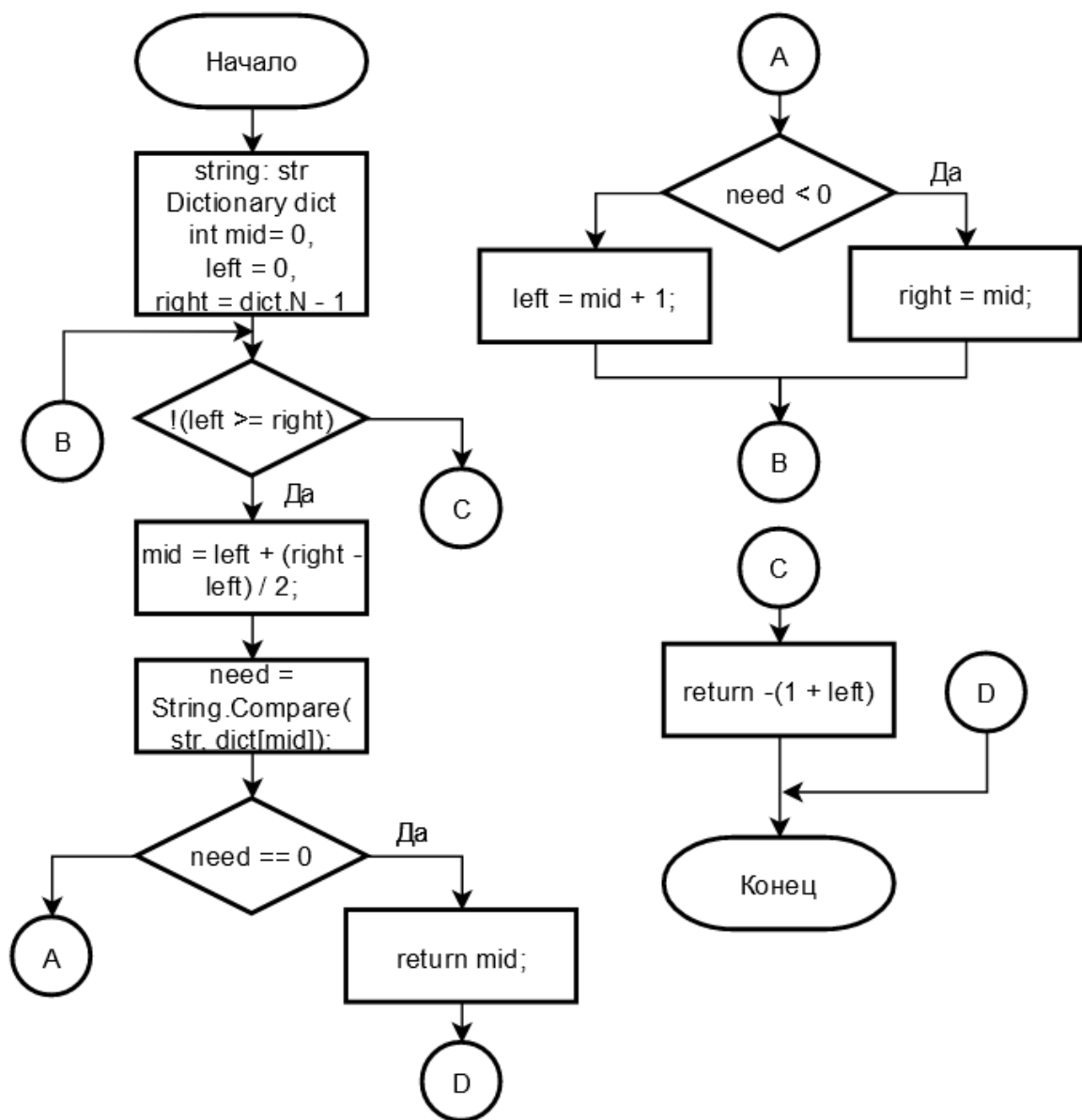
### 2.1 Разработка алгоритмов

На рисунке 1 изображена схема алгоритма поиска полным перебором.



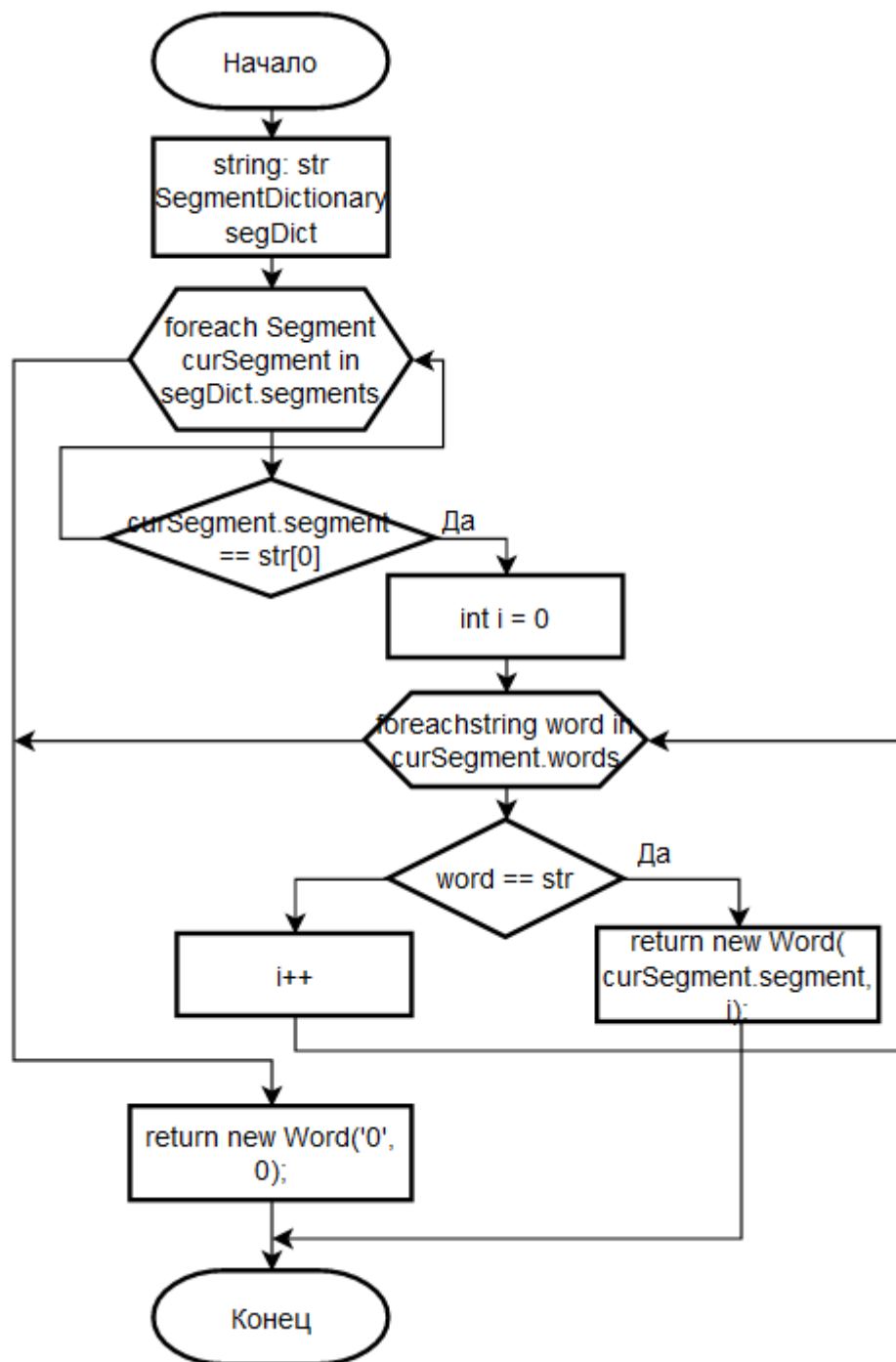
**Рисунок 1** – Схема алгоритма поиска полным перебором

На рисунке 2 изображена схема алгоритма двоичного поиска.



**Рисунок 2** – Схема алгоритма двоичного поиска

На рисунке 3 изображена схема алгоритма поиска по сегментам.



**Рисунок 3** – Схема алгоритма поиска по сегментам

## 2.2 Вывод

В данном разделе была рассмотрена схема алгоритма Винограда и описалась часть алгоритма, которая будет распараллеливаться.



### **3 Технологический раздел**

В данном разделе даны общие требования к программе, средства реализации и сама реализация алгоритмов.

#### **3.1 Общие требования**

##### **Требования к вводу:**

- 1) вводится слово;
- 2) в первых 2 алгоритмах результатом будет являться ключ (в первом по обычному словарю, а во втором по отсортированному словарю), в третьем результатом будет являться ключ (первая буква или же сегмент) и индекс слова в сегменте.

##### **Требования к программе**

- 1) при вводе слова, которого нет в словаре, программа не должна завершиться аварийно;
- 2) ключ или же ключ-индекс должны быть корректными (соответствовать нужному слову).

#### **3.2 Средства реализации**

В качестве языка программирования был выбран C#[1], так как я знаком с данным языком программирования, имею представление о способах тестирования программы.

Средой разработки Visual Studio.[2]

Для замеров процессорного времени используется функция *Stopwatch*. [3][4]

#### **3.3 Сведения о модулях программы**

Программа состоит из:

- 1) Program.cs - главный файл программы, в котором располагается точка входа в программу;
- 2) Dictionary.cs - класс, который хранит массив слов и методы для работы с этими словами;
- 3) Search.cs - класс, в котором хранятся алгоритмы поиска по словарю;
- 4) Segment.cs - класс сегмента, который хранит название сегмента (буква) и массив слов данного сегмента;
- 5) SegmentDictionary.cs - класс, который хранит массив сегментов и методы для работы с этими сегментами;
- 5) Word.cs - класс, который хранит название сегмента (буква) и индекс слова в сегменте.

### 3.4 Листинг кода программы

В листинге 1 реализован алгоритм полного перебора слов.

**Листинг 1** – Алгоритм полного перебора слов

```
1 public static int BruteForce(string str, Dictionary dict)
2 {
3     int result = -1;
4     for (int i = 0; i < dict.N; i++)
5     {
6         if (Equals(str, dict[i]))
7         {
8             return(i);
9         }
10    }
11    return result + 1;
12 }
```

В листинге 2 реализован алгоритм двоичного поиска

## Листинг 2 – Алгоритм двоичного поиска

```
1 public static int BinaryFind(string str, Dictionary dict)
2 {
3     int result, left = 0, right = dict.N - 1;
4     while (true)
5     {
6         if (left > right)
7         {
8             return 0;
9         }
10        result = left + (right - left) / 2;
11        int need = String.Compare(str, dict[result]);
12        if (need < 0)
13        {
14            right = result + 1;
15        }
16        if (need > 0)
17        {
18            left = result - 1;
19        }
20        if (need == 0)
21        {
22            return result;
23        }
24    }
25 }
```

В листинге 3 реализован алгоритм поиска по сегментам

## Листинг 3 – Алгоритм двоичного поиска

```
1 public static Word FindInSegment(string str, SegmentDictionary
   segDict)
2 {
3     foreach (Segment curSegment in segDict.segments)
4     {
```

```

5      if (curSegment.segment == str[0])
6      {
7          int i = 0;
8          foreach (string word in curSegment.words)
9          {
10             if (word == str)
11             {
12                 return new Word(curSegment.segment, i);
13             }
14             i++;
15         }
16         return new Word('0', 0);
17     }
18 }
19 return new Word('0', 0);
20 }

```

### 3.5 Вывод

В данном разделе были даны общие требования к программе, описаны средства реализации, были представлены сведения о модулях программы, а также реализованы алгоритмы поиска по словарю.

## 4 Экспериментальный раздел

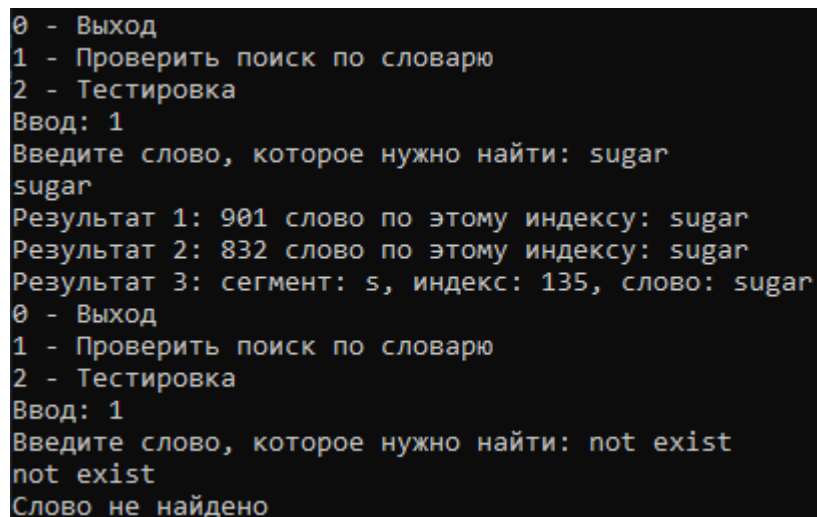
В данном разделе представлены результаты работы программы и приведен анализ времени работы алгоритмов.

### 4.1 Описание экспериментов

Производится замер времени для  $n + 1$  возможных случаев, где  $n$  - длина словаря.

### 4.2 Примеры работы программы

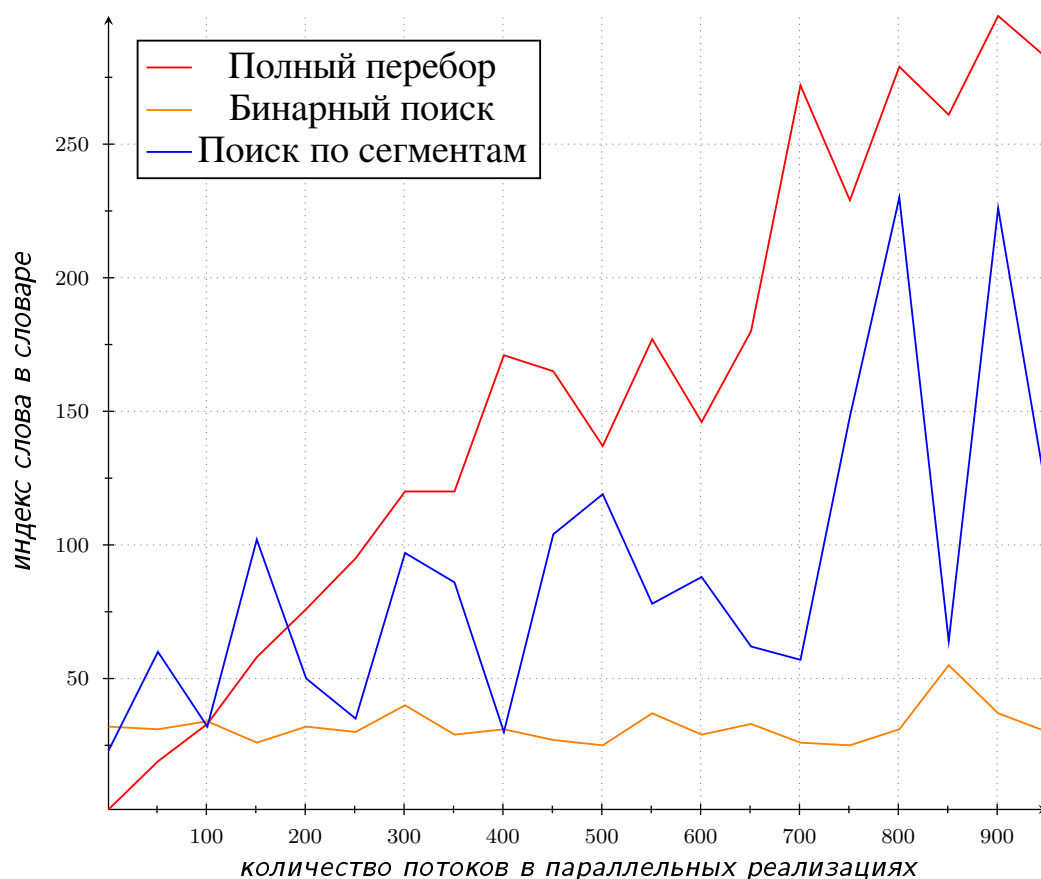
На рисунке 4 представлен результат работы алгоритмов.



```
0 - Выход
1 - Проверить поиск по словарю
2 - Тестировка
Ввод: 1
Введите слово, которое нужно найти: sugar
sugar
Результат 1: 901 слово по этому индексу: sugar
Результат 2: 832 слово по этому индексу: sugar
Результат 3: сегмент: s, индекс: 135, слово: sugar
0 - Выход
1 - Проверить поиск по словарю
2 - Тестировка
Ввод: 1
Введите слово, которое нужно найти: not exist
not exist
Слово не найдено
```

**Рисунок 4** – Первый результат работы программы

На рисунке 5 представлены результаты сравнения трех алгоритмов поиска.



**Рисунок 5** – Результаты замеров процессорного времени.

### 4.3 Вывод

Результаты тестирования показывают, что самым эффективным и "стабильным" является бинарный поиск. Самым долгим является алгоритм полного перебора. Его время возрастает каждый раз из-за того, что слова находятся дальше в словаре, а каждый поиск начинается с самого начала. По графику алгоритма поиска по сегментам можно увидеть резкие скачки. Это происходит из-за того, что слово находится в конце сегмента. Можно сделать вывод, что самым эффективным является алгоритм бинарного поиска.

## **5 Заключение**

В ходе выполнения данной лабораторной работы были изучены три алгоритмы поиска по словарю. Были описаны все алгоритмы и реализованы. Также были изучены способы хранения слов, то есть в случаях первого и второго алгоритмов слова хранились в массивах, а в третьем хранилось по сегментам. Сравнили время работы алгоритмов, в результате которого стало понятно, что самым эффективным алгоритм был бинарный поиск.

## Литература

1. Документация по C#. -URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 24.10.2020). -Текст: электронный.
2. Документация по семейству продуктов Visual Studio. -URL: <https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019> (дата обращения: 01.10.2020). -Текст: электронный.
3. Stopwatch Класс. -URL: <https://goo.su/2e99> (дата обращения: 24.10.2020). -Текст: электронный.
4. Под капотом у Stopwatch. -URL: <https://habr.com/ru/post/226279/> (дата обращения: 24.10.2020). Текст: электронный.