



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 в словаре»

Дисциплина «Анализ алгоритмов»

Студент ИУ7-55Б

(подпись, дата)

Бугаенко А.П.
(Фамилия И.О.)

Преподаватель

(подпись, дата)

Волкова Л.Л.
(Фамилия И.О.)

Москва, 2022

Содержание

Введение	3
1 Аналитический раздел	4
1.1 Понятие словаря	4
1.2 Алгоритм полного перебора	4
1.3 Алгоритм двоичного поиска в упорядоченном словаре	4
1.4 Алгоритм частичного анализа для поиска в словаре	4
1.5 Вывод	5
2 Конструкторский раздел	6
2.1 Тестирование алгоритмов	6
2.2 Алгоритм полного перебора	6
2.3 Алгоритм бинарного поиска	7
2.4 Алгоритм частичного анализа	9
2.5 Функциональная схема ПО	10
2.6 Вывод	10
3 Технологический раздел	11
3.1 Выбор языка программирования	11
3.2 Сведения о модулях программы	11
3.3 Реализация алгоритмов	11
3.4 Результаты тестирования алгоритмов	12
3.5 Вывод	12
4 Экспериментальный раздел	13
4.1 Технические характеристики	13
4.2 Результаты экспериментов	13
4.2.1 Результаты экспериментов для ключа в начале словаря	14
4.2.2 Результаты экспериментов для ключа в середине словаря	16
4.2.3 Результаты экспериментов для ключа в конце словаря	18
4.2.4 Результаты экспериментов для ключа в не из словаря	20
4.3 Вывод	21
Заключение	22
Список литературы	23

Введение

Словарь - тип структуры данных, позволяющей хранить пары вида ключ-значение. Также словарь позволяет добавлять новые пары ключ-значение, удалять старые пары и изменять пары. Одним из основных условий работы словаря является уникальность ключа относительно остального множества ключей, которые на данный момент присутствуют в словаре. В этой лабораторной работе мы будем рассматривать алгоритмы поиска значения по ключу в словаре на основе словаря связей студентов с их научными руководителями.

Целью данной лабораторной работы является изучение алгоритмов поиска значения по ключу в словаре с помощью алгоритмов полного перебора, двоичного поиска в упорядоченном словаре и алгоритме поиска с использованием частичного анализа. Для того, чтобы достичь поставленной цели, необходимо выполнить следующие задачи:

- 1) провести анализ алгоритмов поиска в массиве;
- 2) описать используемые в алгоритмах структуры данных;
- 3) привести схемы рассматриваемых алгоритмов;
- 4) программно реализовать описанные выше алгоритмы;
- 5) провести сравнительный анализ алгоритмов относительно скорости работы.

1 Аналитический раздел

В данном разделе будут рассмотрены теоретические основы работы алгоритмов поиска значения в словаре по ключу.

1.1 Понятие словаря

Словарь - это структура данных, представляющая из себя ассоциативный массив, позволяющий хранить пары вида ключ-значение. Операции, которые поддерживает словарь, включают в себя добавление пары, удаление пары по ключу, поиск пары по ключу. Также в словарях не может быть двух пар с одинаковыми ключами.

На данный момент существует множество вариантов поиска в словаре по ключу. В этой лабораторной работе будут рассмотрены следующие алгоритмы:

- поиск полным перебором;
- двоичный поиск в упорядоченном словаре;
- частичный анализ для поиска в словаре.

1.2 Алгоритм полного перебора

Алгоритм полного перебора подразумевает последовательный проход по словарю, в процессе которого каждый ключ из каждой пары сравнивается с искомым ключом, пока не будет найдено совпадение. Лучшим случаем для данного алгоритма является ситуация, когда искомым ключ находится в начале словаря, в результате происходит одно сравнения. Худших случаев два - ключ не найден и ключ находится в конце.

1.3 Алгоритм двоичного поиска в упорядоченном словаре

Двоичный поиск использует отсортированный массив ключей. Идея заключается в выделении среднего для фрагмента элемента и сравнения ключа с ним. Так как список отсортирован, мы можем с лёгкостью определять при каждом сравнении, в какой половине фрагмента находится элемент, пока не дойдём до самого элемента. Лучший случай алгоритма - элемент находится в центре, в следствие чего находится за один прогон цикла. Худшим случаем является нахождение элемента на краях массива, в начале или в конце. Так как длина части, в которой мы ищем элемент, каждый раз сокращается в два раза, то общая сложность будет задаваться как $O(\log_2 n)$.

1.4 Алгоритм частичного анализа для поиска в словаре

Использование частичного анализа подразумевает, что словарь, подающийся на вход, был отсортирован по частоте встречаемости первого символа каждого ключа словаря. Ключами нового словаря являются буквы алфавита. Новый словарь формируется таким образом, что первая буква ключей основного словаря совпадает с ключами нового словаря в результате

чего такая организация напоминает толковый словарь. Изначально поиск ведётся по первой букве, а затем полным перебором. В худшем случае первая буква у всех ключей будет одна и та же, что приведёт к тому, что алгоритм будет работать, как алгоритм полного перебора. В лучшем случае ключ в новом словаре будет указывать на словарь из одного слова, которое и является искомым ключом.

1.5 Вывод

В данном разделе были рассмотрены основные теоретические сведения об алгоритмах полного перебора, бинарного поиска и частичного анализа. В результате были сделаны выводы о том, что на вход алгоритму подаётся словарь и искомый ключ, а на выходе программа возвращает значение, которое соответствует данному ключу. Алгоритмы работают на словарях размерностью от 0 до физически возможного предела для используемой машины. В качестве критерия для сравнения эффективности алгоритмов будет использоваться время работы на словарях различных размерностей.

2 Конструкторский раздел

В данном разделе будут рассмотрены схемы, структуры данных, способы тестирования, описания памяти для следующих алгоритмов:

- алгоритм полного перебора;
- алгоритм бинарного поиска;
- алгоритм частичного анализа.

2.1 Тестирование алгоритмов

Описание тестов:

- 1) проверка работы на ключе из начала словаря;
- 2) проверка работы на ключе из середины словаря;
- 3) проверка работы на ключе из конца словаря;
- 4) проверка работы на несуществующем ключе.

2.2 Алгоритм полного перебора

Используемые типы и структуры данных включают в себя:

- 1) `integer`, целое число - используется для хранения индексов массива, размера массива;
- 2) `dict`, словарь - стандартная структура данных, используемая для хранения словаря.

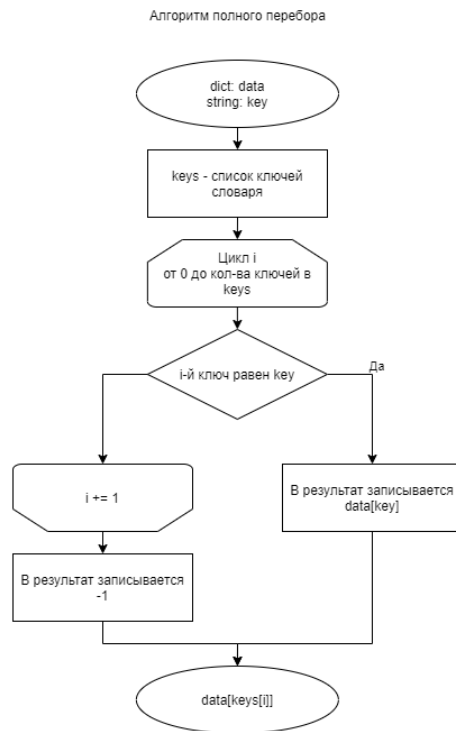


Рисунок 2.1 — Схема алгоритма полного перебора

2.3 Алгоритм бинарного поиска

Используемые типы и структуры данных включают в себя:

- 1) integer, целое число - используется для хранения индексов массива, размера массива;
- 2) dict, словарь - стандартная структура данных, используемая для хранения словаря.

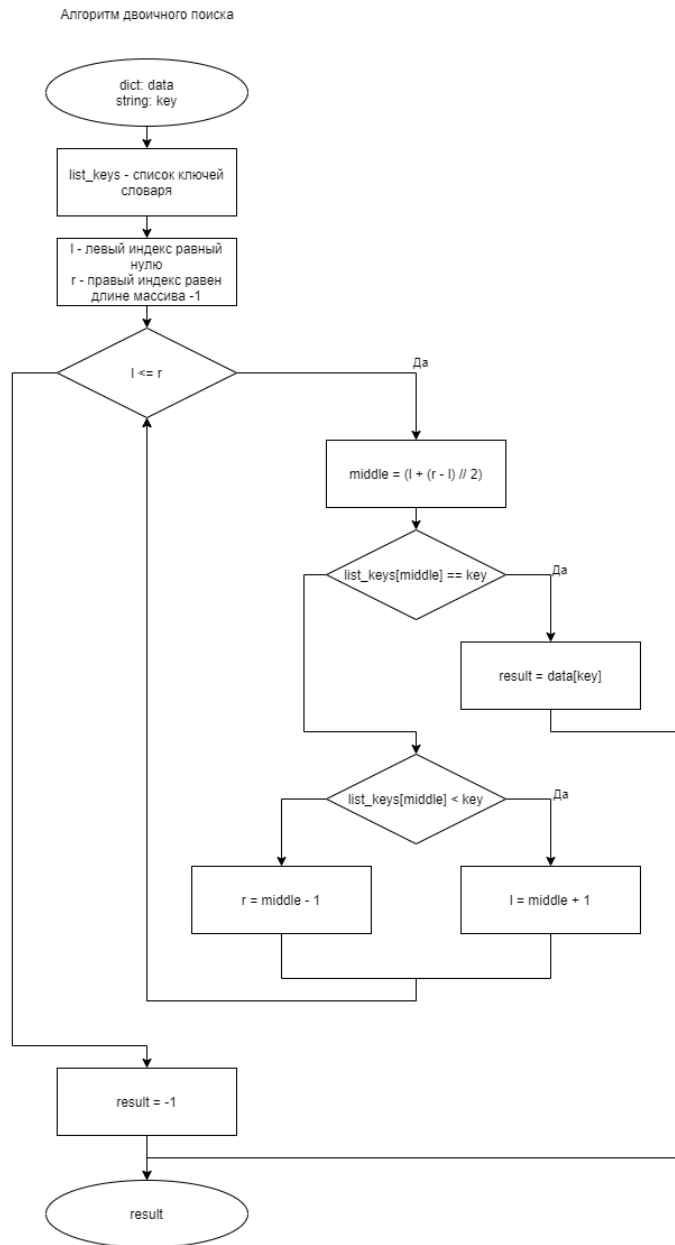


Рисунок 2.2 — Схема алгоритма бинарного поиска

2.4 Алгоритм частичного анализа

Используемые типы и структуры данных включают в себя:

- 1) integer, целое число - используется для хранения индексов массива, размера массива;
- 2) dict, словарь - стандартная структура данных, используемая для хранения словаря.

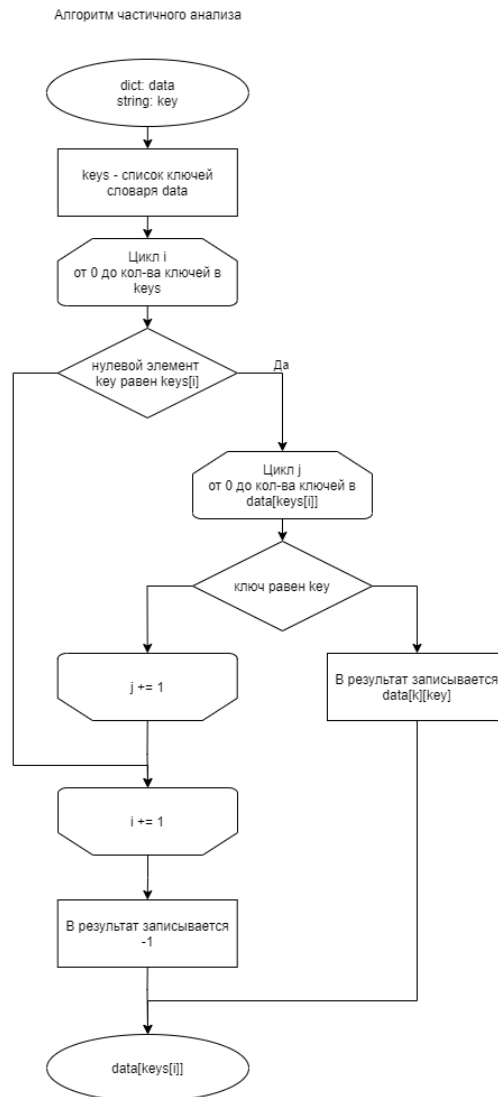


Рисунок 2.3 — Схема алгоритма частичного анализа

2.5 Функциональная схема ПО

На изображении ниже представлена функциональная схема разрабатываемого ПО. На вход подаётся словарь и ключ, при помощи алгоритмов, реализованных на языке Python мы получаем в результате работы значение, соответствующее ключу.



Рисунок 2.4 — IDEF0 диаграмма разрабатываемой программы

2.6 Вывод

В данном разделе были рассмотрены схемы алгоритмов полного перебора, бинарного поиска и частичного анализа. Были определены тесты для каждого алгоритма и описаны типы и структуры данных, использующихся в алгоритмах. Также была приведена функциональная схема разрабатываемого ПО.

3 Технологический раздел

В данном разделе будут рассмотрены подробности реализации описанных выше алгоритмов. Также будут обоснованы выбор языка программирования для реализации, выбор библиотек для проведения экспериментов и представлены важные фрагменты кода написанной в рамках работы программы.

3.1 Выбор языка программирования

В качестве языка программирования для реализации данной лабораторной работы использовался язык программирования Python поскольку он предоставляет широкие возможности для эффективной реализации алгоритмов. В качестве среды разработки использовалась Visual Studio Code по причине того, что данная среда имеет встроенные средства отладки и анализа работы программы, позволяющие быстро и эффективно писать код.

3.2 Сведения о модулях программы

Реализованное ПО состоит из трёх модулей:

- 1) main - основной файл программы, где находится точка входа;
- 2) tests - реализация тестов алгоритмов;
- 3) algos - реализация алгоритмов.

3.3 Реализация алгоритмов

Листинг 3.1 — Реализация алгоритма полного перебора

```
1 def seq_search(data, key):
2     for elem in data:
3         if key == elem:
4             return data[elem]
5     return -1
```

Листинг 3.2 — Реализация алгоритма бинарного поиска

```
1 def bin_search(data, key):
2     list_keys = sorted(list(data.keys()))
3     l, r = 0, len(list_keys) - 1
4     while l <= r:
5         middle = (l + (r - 1) // 2)
6
7         if list_keys[middle] == key:
8             return data[key]
9         elif list_keys[middle] < key:
10            l = middle + 1
11        else:
12            r = middle - 1
```

```
13
14     return -1
```

Листинг 3.3 — Реализация алгоритма частичного анализа

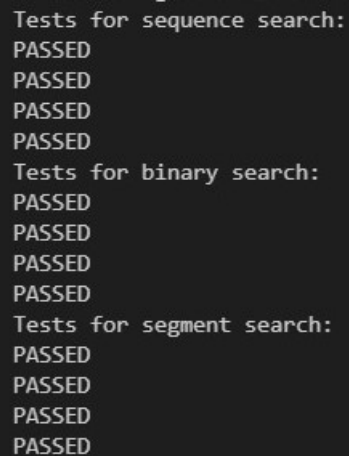
```
1 def seg_search(data, key):
2     for k in data:
3         if key[0] == k:
4             for elem in data[k]:
5                 if elem == key:
6                     return data[k][elem]
7     return -1
```

3.4 Результаты тестирования алгоритмов

Для тестирования алгоритмов было реализованы следующие тесты:

- 1) проверка работы на ключе из начала словаря;
- 2) проверка работы на ключе из середины словаря;
- 3) проверка работы на ключе из конца словаря;
- 4) проверка работы на несуществующем ключе.

Результаты тестов:



```
Tests for sequence search:
PASSED
PASSED
PASSED
PASSED
Tests for binary search:
PASSED
PASSED
PASSED
PASSED
Tests for segment search:
PASSED
PASSED
PASSED
PASSED
```

Рисунок 3.1 — Результаты тестирования алгоритмов

3.5 Вывод

В данной разделе были представлены реализации алгоритма полного перебора, бинарного поиска и частичного анализа, а также представлены результаты работы модуля тестирования реализованных алгоритмов.

4 Экспериментальный раздел

В данном разделе описываются измерения временных характеристики алгоритмов полного перебра, бинарного поиска и частичного анализа для различных положений ключа в словаре, а также делается вывод об эффективности алгоритмов для соответствующих случаев.

4.1 Технические характеристики

- Операционная система - Windows 10, 64-bit;
- Оперативная память - 16 GiB;
- Процессор - Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz, 6 ядер, 12 потоков.

4.2 Результаты экспериментов

4.2.1 Результаты экспериментов для ключа в начале словаря

Таблица 4.1 — Время работы алгоритмов при предварительной сортировке с ключом в начале

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	0	0	0
150000	0	0	0
200000	0	0	0
250000	0	0	3125000
300000	0	0	0
350000	0	0	0
400000	0	0	0
450000	0	0	0
500000	0	0	0

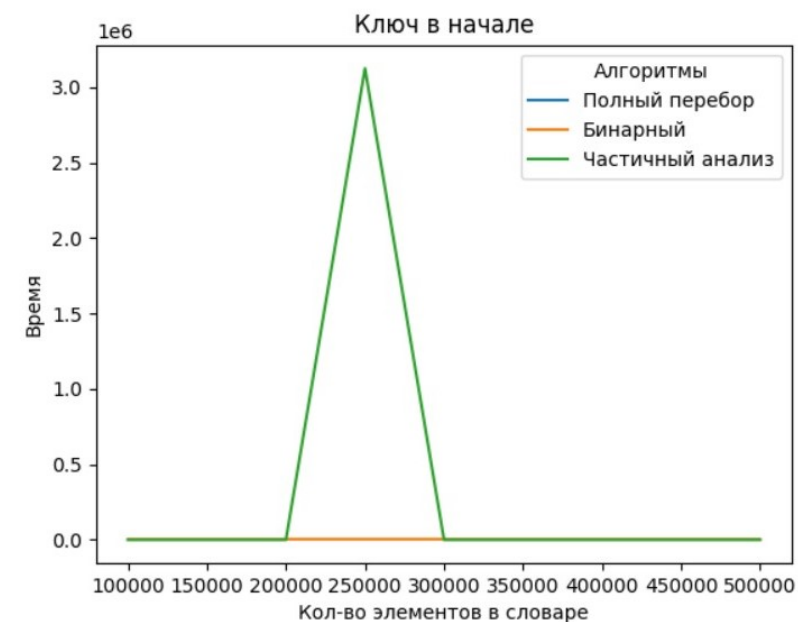


Рисунок 4.1 — График зависимости времени работы от размерности словарей при предварительной сортировке с ключом в начале

Таблица 4.2 — Время работы алгоритмов без предварительной сортировки с ключом в начале

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	0	78125000	62500000
150000	0	134375000	96875000
200000	0	187500000	137500000
250000	0	246875000	187500000
300000	0	312500000	218750000
350000	0	425000000	265625000
400000	0	450000000	296875000
450000	0	537500000	340625000
500000	0	584375000	415625000

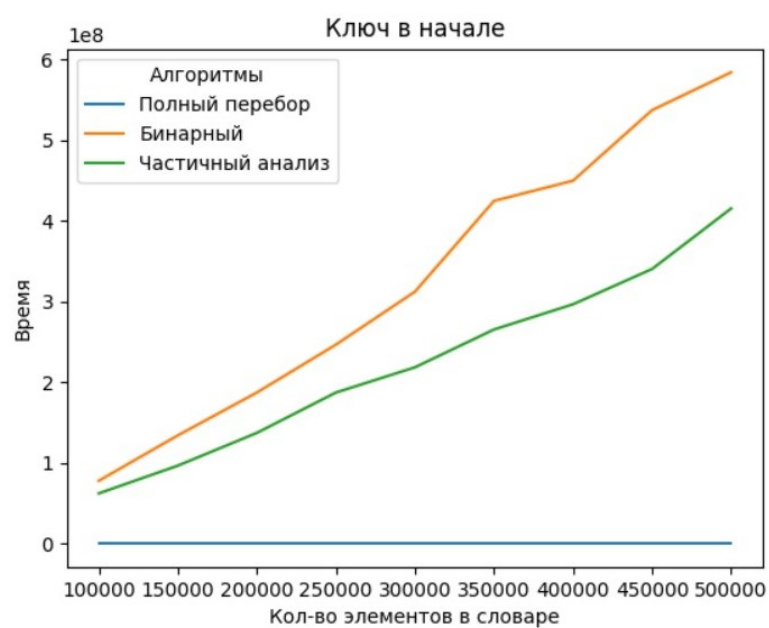


Рисунок 4.2 — График зависимости времени работы от размерности словарей без предварительной сортировки с ключом в начале

4.2.2 Результаты экспериментов для ключа в середине словаря

Таблица 4.3 — Время работы алгоритмов при предварительной сортировке с ключом в середине

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	3125000	0	0
150000	3125000	0	0
200000	3125000	0	0
250000	12500000	0	0
300000	9375000	0	0
350000	9375000	0	0
400000	12500000	0	3125000
450000	12500000	0	3125000
500000	12500000	0	6250000

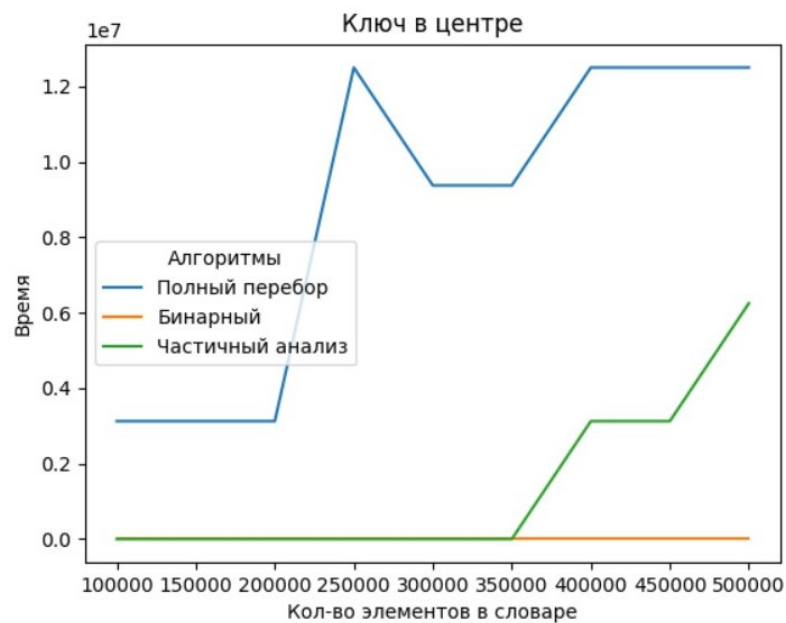


Рисунок 4.3 — График зависимости времени работы от размерности словарей при предварительной сортировке с ключом в середине

Таблица 4.4 — Время работы алгоритмов без предварительной сортировки с ключом в середине

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	0	81250000	62500000
150000	3125000	131250000	96875000
200000	6250000	187500000	150000000
250000	9375000	259375000	193750000
300000	9375000	337500000	225000000
350000	9375000	381250000	259375000
400000	9375000	437500000	300000000
450000	12500000	534375000	359375000
500000	12500000	578125000	412500000

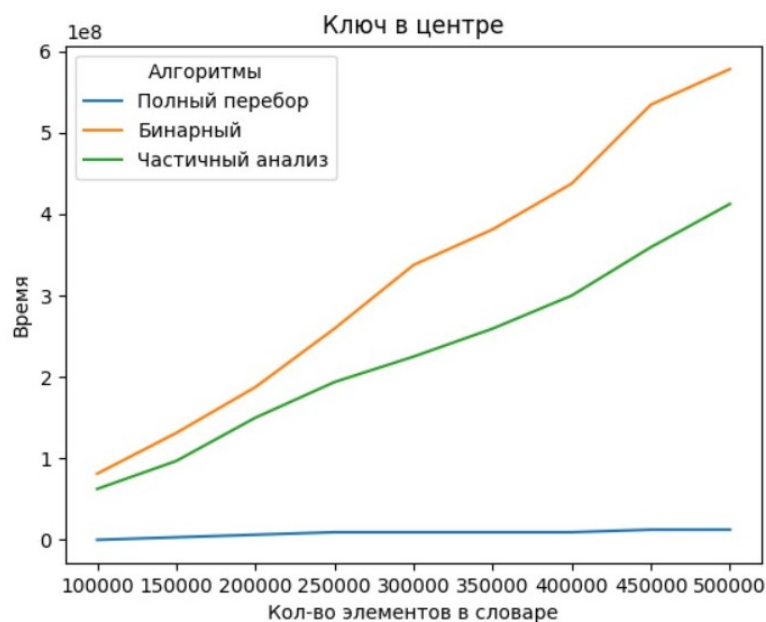


Рисунок 4.4 — График зависимости времени работы от размерности словарей без предварительной сортировки с ключом в середине

4.2.3 Результаты экспериментов для ключа в конце словаря

Таблица 4.5 — Время работы алгоритмов при предварительной сортировке с ключом в конце

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	6250000	0	3125000
150000	6250000	0	0
200000	9375000	0	0
250000	18750000	0	3125000
300000	15625000	0	0
350000	21875000	3125000	0
400000	25000000	0	3125000
450000	25000000	0	3125000
500000	34375000	0	3125000

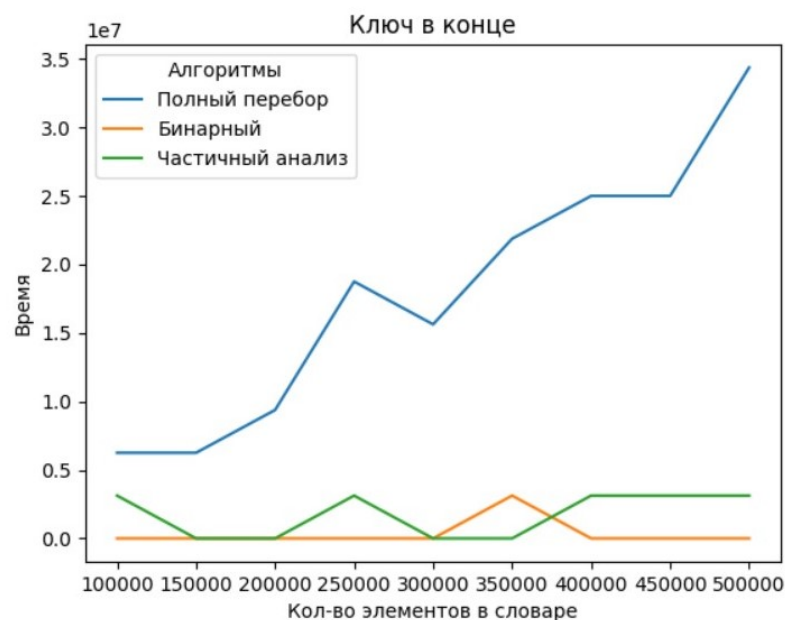


Рисунок 4.5 — График зависимости времени работы от размерности словарей при предварительной сортировке с ключом в конце

Таблица 4.6 — Время работы алгоритмов без предварительной сортировки с ключом в конце

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	6250000	75000000	65625000
150000	6250000	131250000	100000000
200000	9375000	196875000	137500000
250000	12500000	246875000	190625000
300000	12500000	312500000	215625000
350000	15625000	375000000	265625000
400000	21875000	437500000	321875000
450000	21875000	509375000	346875000
500000	25000000	646875000	418750000

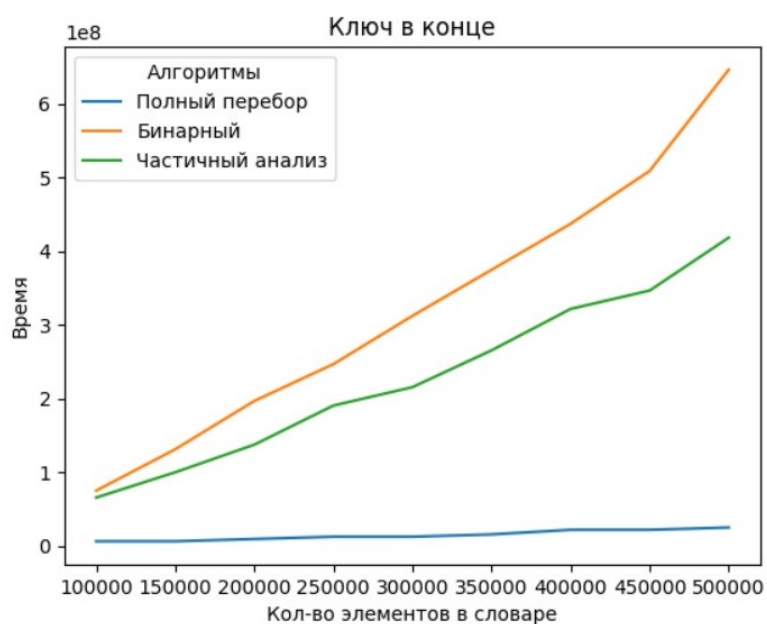


Рисунок 4.6 — График зависимости времени работы от размерности словарей без предварительной сортировки с ключом в конце

4.2.4 Результаты экспериментов для ключа в не из словаря

Таблица 4.7 — Время работы алгоритмов при предварительной сортировке с ключом не из словаря

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	6250000	0	3125000
150000	6250000	0	0
200000	9375000	0	0
250000	18750000	0	3125000
300000	15625000	0	0
350000	21875000	3125000	0
400000	25000000	0	3125000
450000	25000000	0	3125000
500000	34375000	0	3125000

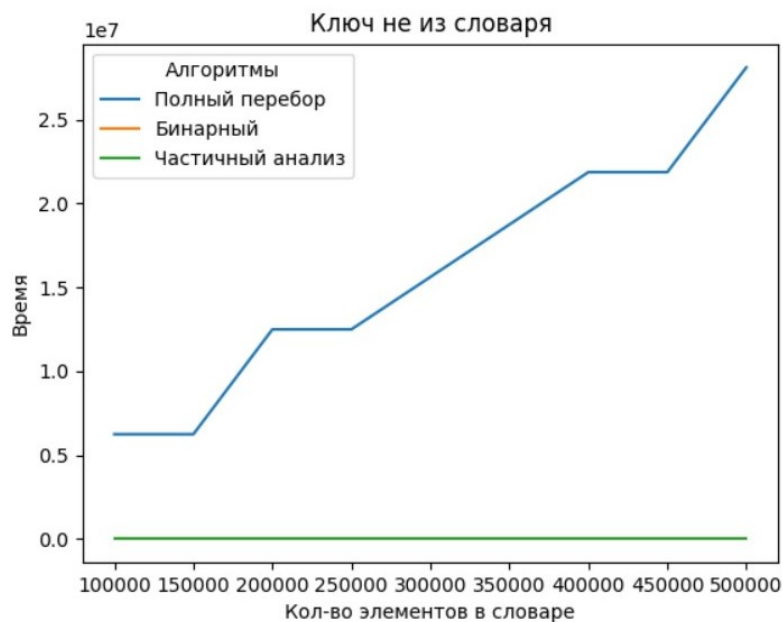


Рисунок 4.7 — График зависимости времени работы от размерности словарей при предварительной сортировке с ключом не из словаря

Таблица 4.8 — Время работы алгоритмов без предварительной сортировки с ключом в начале

кол-во пар	t полного перебора (нс)	t бинарного поиска (нс)	t частичного анализа (нс)
100000	3125000	78125000	65625000
150000	6250000	131250000	100000000
200000	9375000	187500000	137500000
250000	9375000	268750000	190625000
300000	15625000	309375000	218750000
350000	15625000	371875000	271875000
400000	18750000	456250000	318750000
450000	21875000	537500000	337500000
500000	25000000	625000000	390625000

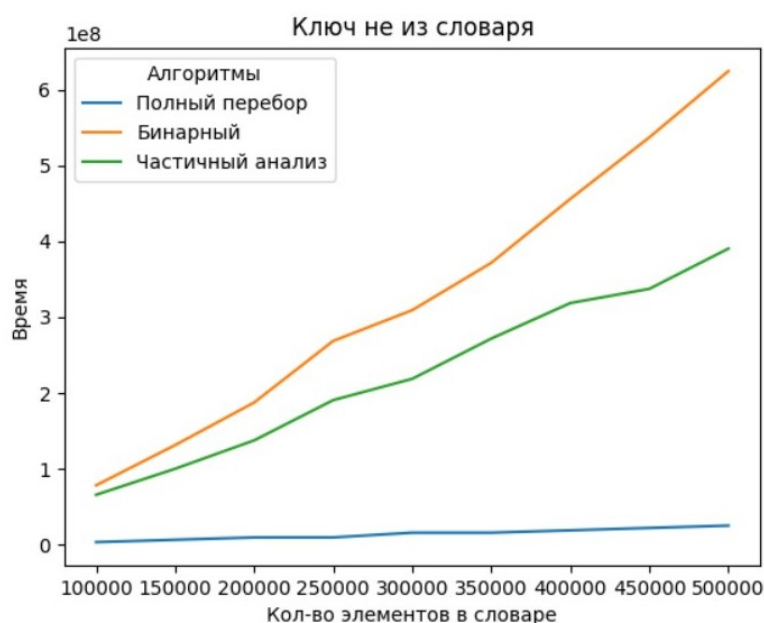


Рисунок 4.8 — График зависимости времени работы от размерности словарей без предварительной сортировки с ключом не из словаря

4.3 Вывод

В результате эксперимента было получено, что при предварительной сортировке данных наиболее медленным алгоритмом является алгоритм полного перебора. В случае, когда подготовка исходного массива включается во временные измерения самым быстрым алгоритмом во всех случаях является алгоритм полного перебора, а самым медленным - бинарный поиск.

Заключение

В процессе выполнения данной лабораторной работы были изучены алгоритмы полного перебора, бинарного поиска и частичного анализа. Были выполнены анализ алгоритмов и представлены схемы алгоритмов, а также функциональная схема ПО. После чего эти алгоритмы были реализованы при помощи языка Python в IDE Visual Studio Code. Помимо этого были произведены эксперименты с целью получить информацию о временной производительности алгоритмов. В результате эксперимента было получено, что при предварительной сортировке данных наиболее медленным алгоритмом является алгоритм полного перебора. В случае, когда подготовка исходного массива включается во временные измерения самым быстрым алгоритмом во всех случаях является алгоритм полного перебора, а самым медленным - бинарный поиск. Целью данной лабораторной работы являлось изучение алгоритмов полного перебора, бинарного поиска и частичного анализа на примере задачи поиска по ключу в словаре, что было успешно достигнуто.

Список литературы

- [1] NIST's Dictionary of Algorithms and Data Structures: Associative Array [Электронный доступ], режим доступа: <https://xlinux.nist.gov/dads/HTML/assocarray.html> (дата обращения: 21.12.2021)
- [2] The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с.
- [3] Документация языка Python [Электронный доступ], режим доступа: <https://docs.python.org/3/index.html> - 16.12.2021
- [4] Magnus Lie Hetland. Python Algorithms: Mastering Basic Algorithms in the Python Language. — Apress, 2010. — 336 с.
- [5] Документация к среде разработки Visual Studio Code [Электронный доступ], режим доступа: <https://code.visualstudio.com/docs> (дата обращения: 21.12.2021)