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 №3 по курсу «Анализ Алгоритмов»

Студент Паламарчук А.Н.

Группа ИУ7-53Б

Преподаватель Волкова Л. Л.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является исследование алгоритмов линейного и бинарного поиска. Для достижения поставленной цели необходимо выполнить следующие задачи:

— разработать алгоритмы линейного и бинарного поиска;

— разработать программное обеспечение, содержащее алгоритмы линейного и бинарного поиска;

— провести функциональное тестирование реализованных алгоритмов;

— провести сравнительный анализ алгоритмов.

1 Аналитическая часть

1.1 Словарь

Словарь – тип данных, который позволяет хранить пары вида «ключ-значение» – *(k, v)*. В паре *(k, v)* – v это значение, которое ассоциируется с ключом k.

В данной лабораторной работе словарь представлен следующим образом:

— ключ – VIN номер автомобиля;

— значение – информация об автомобиле.

1.2 Линейный поиск

Линейный поиск [1] – это метод поиска элемента в списке. Он последовательно проверяет каждый элемент массива до тех пор, пока не будет найдено совпадение или пока не будет просмотрен весь массив. В массиве из n элементов существует $n+1$ возможных случаев размещения искомого значения.

Алгоритм при начале работы затрачивает k операций, при сравнении q операций, тогда:

— лучший случай – элемент найден на первом сравнении ($k+q$ операций);

— общий случай – элемент найден на *i-ом* сравнении ($k + i \cdot q$ операций);

— худший случай – элемент найден на последнем сравнении, либо не найден ($k + n \cdot q$ операций).

1.3 Бинарный поиск

Бинарный поиск [1] – поиск в заранее отсортированном словаре, который заключается в сравнении со средним ключом, в результате этого сравнения определить, в какой половине словаря находится искомый ключ, и снова применить ту же процедуру к половине словаря.

Алгоритм при начале работы затрачивает k операций, тогда:

— лучший случай – элемент найден на первом сравнении со средним элементом $(b + \log_2 1)$;

— общий случай – элемент найден на *i-ом* сравнении $(b + \log_2 i)$;

— худший случай – элемент найден на последнем сравнении $(b + \log_2 n)$, где n – размер словаря.

2 Конструкторская часть

2.1 Описание используемых структур данных

При реализации алгоритмов будут использованы следующие структуры данных:

— искомый VIN номер — строковый тип;

— словарь для хранения информации об автомобиле по его VIN — тип словарь.

2.2 Разработка алгоритмов

На рисунках 2.1 и 2.2 представлены алгоритмы линейного и бинарного поиска.

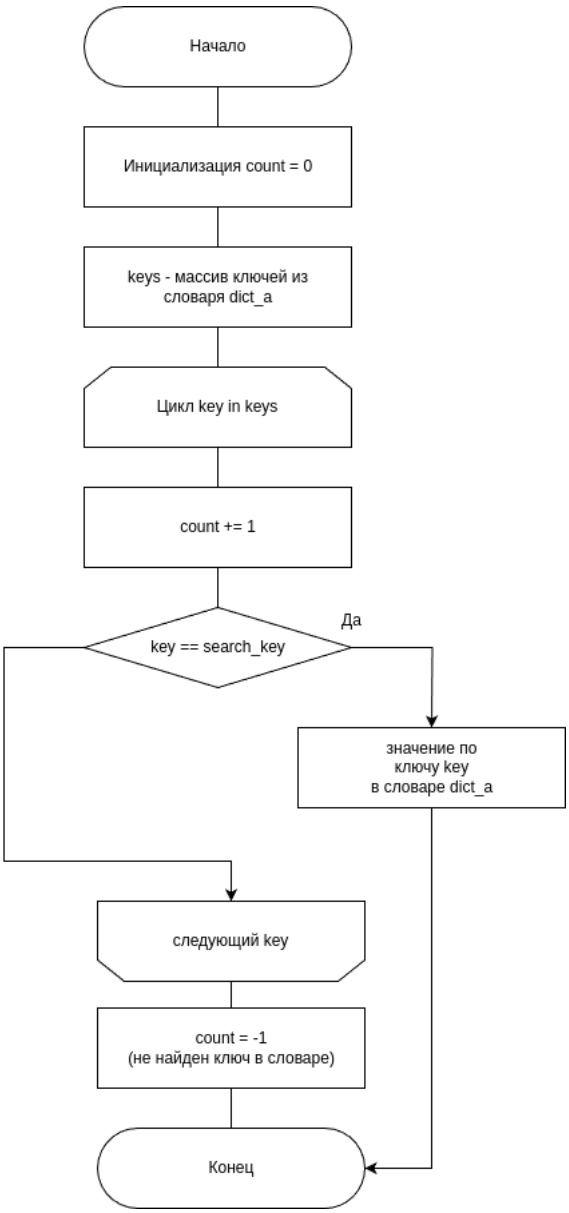

Рисунок 2.1 – Схема алгоритма поиска полным перебором

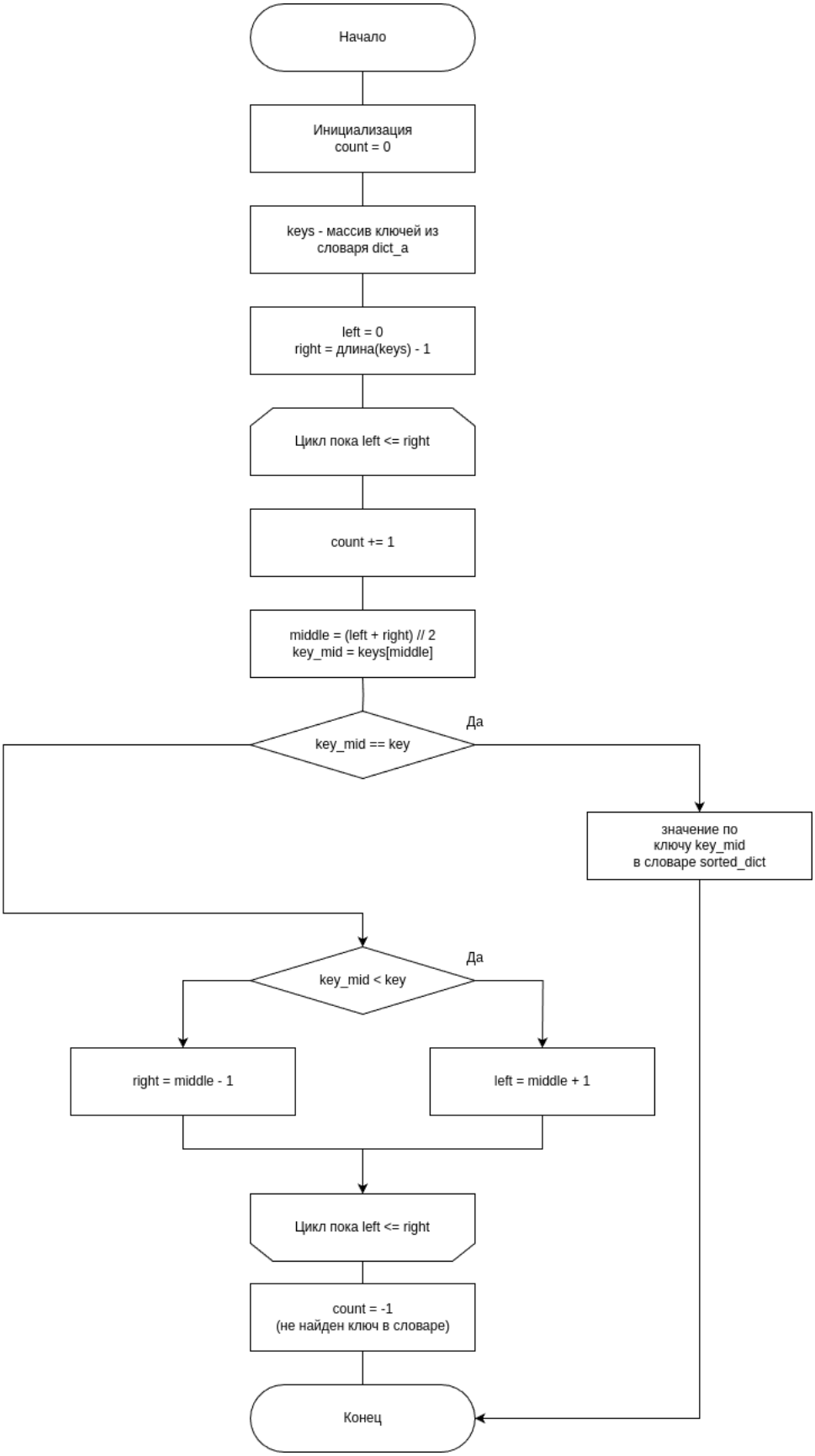

Рисунок 2.2 – Схема алгоритма бинарного поиска

3 Технологическая часть

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python*. Требуется замерить количество сравнений и построить графики. Для построения графиков использовалась библиотека *mathplotlib*.

3.2 Реализация алгоритмов

В листингах 3.1- 3.2 представлены реализации алгоритмов поиска полным перебором и бинарного поиска.

Листинг 3.1 – Алгоритм поиска в словаре полным перебором

```
1 def full_search(dict_a: Dictionary, search_key):
2     count = 0
3     keys = dict_a.data.keys()
4     for key in keys:
5         count += 1
6         if (key == search_key):
7             return [dict_a.data[key], count]
8     return [NOT_FOUND, count]
```

Листинг 3.2 – Алгоритм бинарного поиска

```
def bin_search(dict_a: Dictionary, key):
    count = 0
    sort_dict = Dictionary()
    sort_dict.data = dict_a.sort()
    keys = list(sort_dict.data.keys())
    left = 0
    right = len(keys)

    while (left <= right):
        count += 1
        mid = (left + right) // 2
        key_mid = keys[mid]
        if (key_mid == key):
            return [sort_dict.data[key_mid], count]
        elif (key_mid < key):
            left = mid + 1
        else:
            right = mid - 1

    return [NOT_FOUND, count]
```

3.3 Функциональные тесты

В таблице 3.1 приведены тесты для функций программы. Тесты для всех функций пройдены успешно.

Таблица 3.1 – Функциональные тесты

Ключ	Результат	Пояснение
emptyVIN	Нет такого ключа	Ключ отсутствует
WZSLH08W7RNBMEEE2	Инф-ия и кол-во сравнений	Первый ключ
KTMY9BVE3K3ARBKE8	Инф-ия и кол-во сравнений	Последний ключ
7LB08NXS8G9UBGNZE	Инф-ия и кол-во сравнений	Ключ в словаре

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства:

— операционная система Manjaro Linux x86_64;

— процессор Ryzen 5500U 6 ядер, тактовая частота 2.1 ГГц;

— оперативная память 16 Гбайт.

При тестировании ноутбук был включён в сеть электропитания. Во время тестирования ноутбук был нагружен только системными приложениями окружения, а также системой тестирования.

4.2 Количество сравнений в алгоритмах

В ходе исследования исследуется количество сравнений необходимых для нахождения искомого элемента.

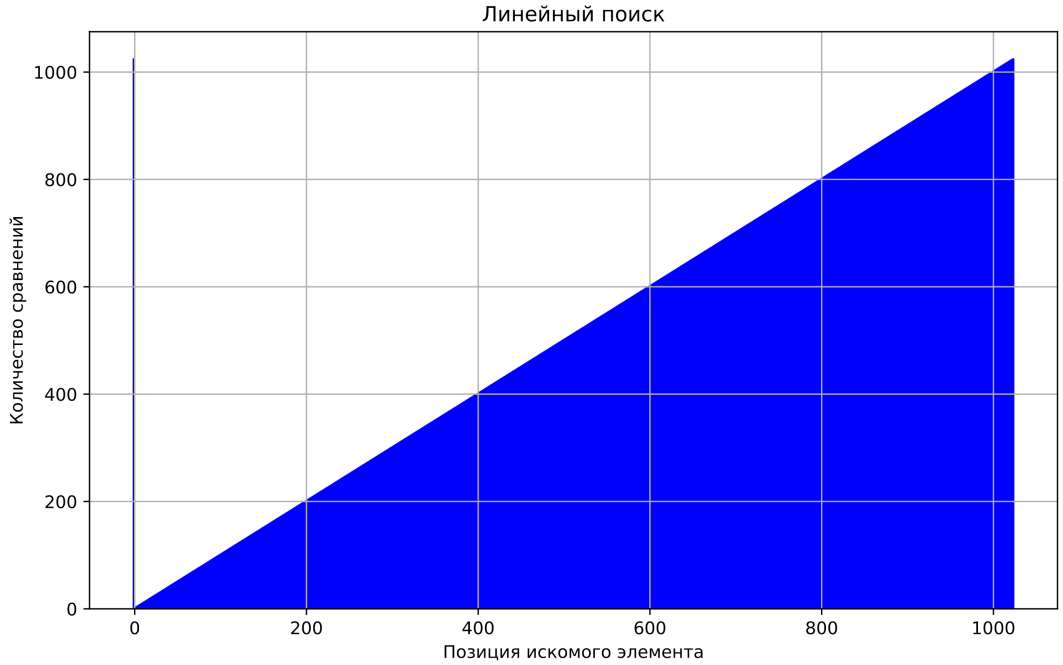

Рисунок 4.1 – Алгоритмом линейного поиска

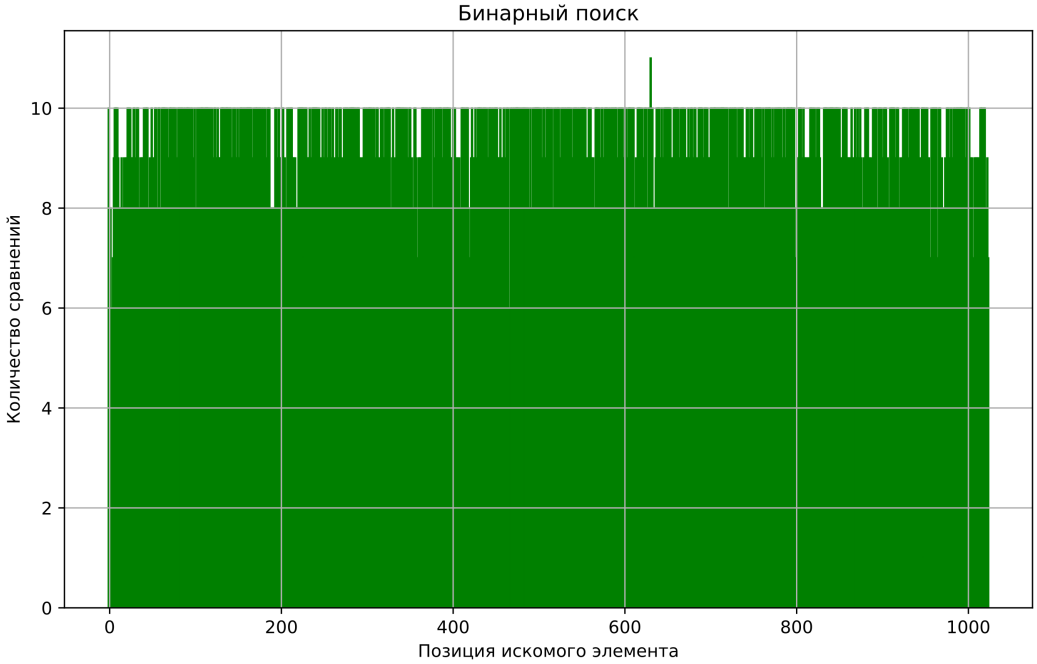

Рисунок 4.2 – Алгоритм бинарного поиска

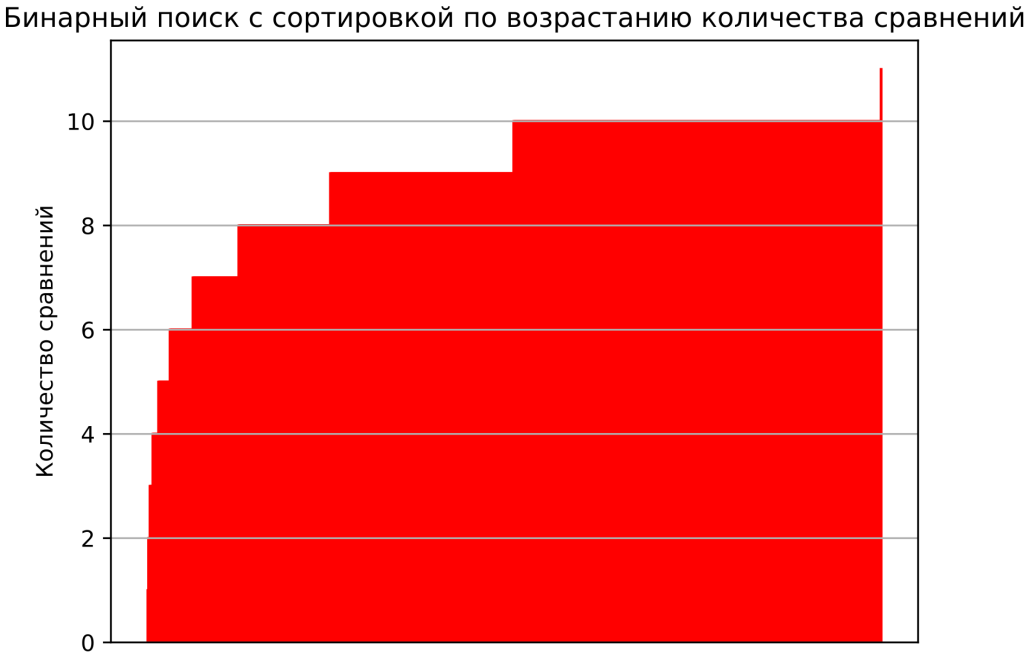

Рисунок 4.3 – Алгоритм бинарного поиска с сортировкой по возрастанию количества сравнений

4.3 Результаты проводимых исследований

В результаты измерений количества сравнений, представлены на рисунках 4.1, 4.2 и 4.3. Среднее количество сравнений для нахождения элемента в словаре: алгоритм линейного поиска — 512, бинарного поиска — 8. Алгоритм

бинарного поиска эффективнее линейного поиска по количеству сравнений. Но для работы бинарного поиска нужно предварительно отсортировать исходный словарь.

ЗАКЛЮЧЕНИЕ

В результате исследования было определено, что алгоритм бинарного поиска эффективнее линейного поиска по количеству сравнений примерно в 60 раз. Но для работы бинарного поиска нужно предварительно отсортировать исходный словарь.

Цель достигнута, были исследованы алгоритмы линейного и бинарного поиска, а также в ходе выполнения лабораторной работы были решены следующие задачи:

— разработаны алгоритмы линейного и бинарного поиска;

— разработано программное обеспечение, содержащее алгоритмы линейного и бинарного поиска;

— проведено функциональное тестирование реализованных алгоритмов;

— проведён сравнительный анализ алгоритмов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Кнут Д.Э. Исскуство программирования, том 3. Сортировка и поиск, 2-е изд. Москва: ООО И.Д. Вильямс, 2009. с. 832.