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 № 3

по курсу «Анализ алгоритмов»

на тему: «Поиск в массиве»

Вариант № 8186

Студент ИУ7-56Б (Группа) _____ (Подпись, дата) Александрова А. Д. (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Строганов Д. В. (И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **3**

1 Аналитический раздел **4**

1.1 Поиск полным перебором 4

1.2 Бинарный поиск 4

2 Конструкторский раздел **5**

2.1 Проектирование алгоритмов 5

3 Технологический раздел **8**

3.1 Требования к программе 8

3.2 Средства реализации 8

3.3 Программные модули 8

3.4 Реализация алгоритмов 8

3.5 Функциональные тесты 10

4 Исследовательский раздел **11**

4.1 Временные характеристики 11

ЗАКЛЮЧЕНИЕ **13**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **14**

ВВЕДЕНИЕ

Целью данной лабораторной работы является исследование алгоритмов поиска элемента в массиве.

Задачи лабораторной работы:

1) реализация алгоритмов поиска элемента в массиве — полного перебора и бинарного поиска;

2) исследование ресурсной эффективности реализованных алгоритмов, сравнительный анализ результатов;

3) описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1 Аналитический раздел

1.1 Поиск полным перебором

Алгоритм поиска элемента в массиве методом **полного перебора**, также известный как **линейный поиск**, используется для нахождения значения в массиве. Он проходит по каждому элементу, начиная с первого, и сравнивает его с искомым значением. Если элемент найден, возвращается его индекс; если нет — возвращается -1.

Так как алгоритм перебирает элементы последовательно, доступ к элементам в начале массива будет получен быстрее, чем к элементам в конце. То есть лучшим будет случай, когда искомый элемент является первым элементом массива. В таком случае трудоёмкость алгоритма равна $O(1)$. Худшим же будет случай, когда искомый элемент последний или отсутствует в массиве. Трудоёмкость в таком случае будет равна $O(n)$.

Этот метод подходит для небольших или неотсортированных массивов.

1.2 Бинарный поиск

Бинарный поиск [1] — это эффективный алгоритм поиска элемента в отсортированном массиве. Он работает по принципу деления массива на две половины и сравнения искомого значения с элементом, находящимся в середине массива. Если искомое значение больше, то поиск продолжится в правой части массива, иначе — в левой.

Так как начинаем сравнивать искомый элемент мы с середины массива, то лучшим случаем будет тот, где он находится в середине массива, и трудоёмкость будет равна $O(1)$. Худшим, с трудоёмкостью $O(log(n))$ соответственно, случай, где элемента нет в массиве, или он находится на позиции, для получения которой необходимо сделать $log(n)$ сравнений.

Бинарный поиск значительно быстрее линейного поиска для больших отсортированных массивов.

2 Конструкторский раздел

2.1 Проектирование алгоритмов

В процессе проектирования программы были разработаны алгоритмы поиска элемента в массиве, представленные на рисунках 2.1 и 2.2

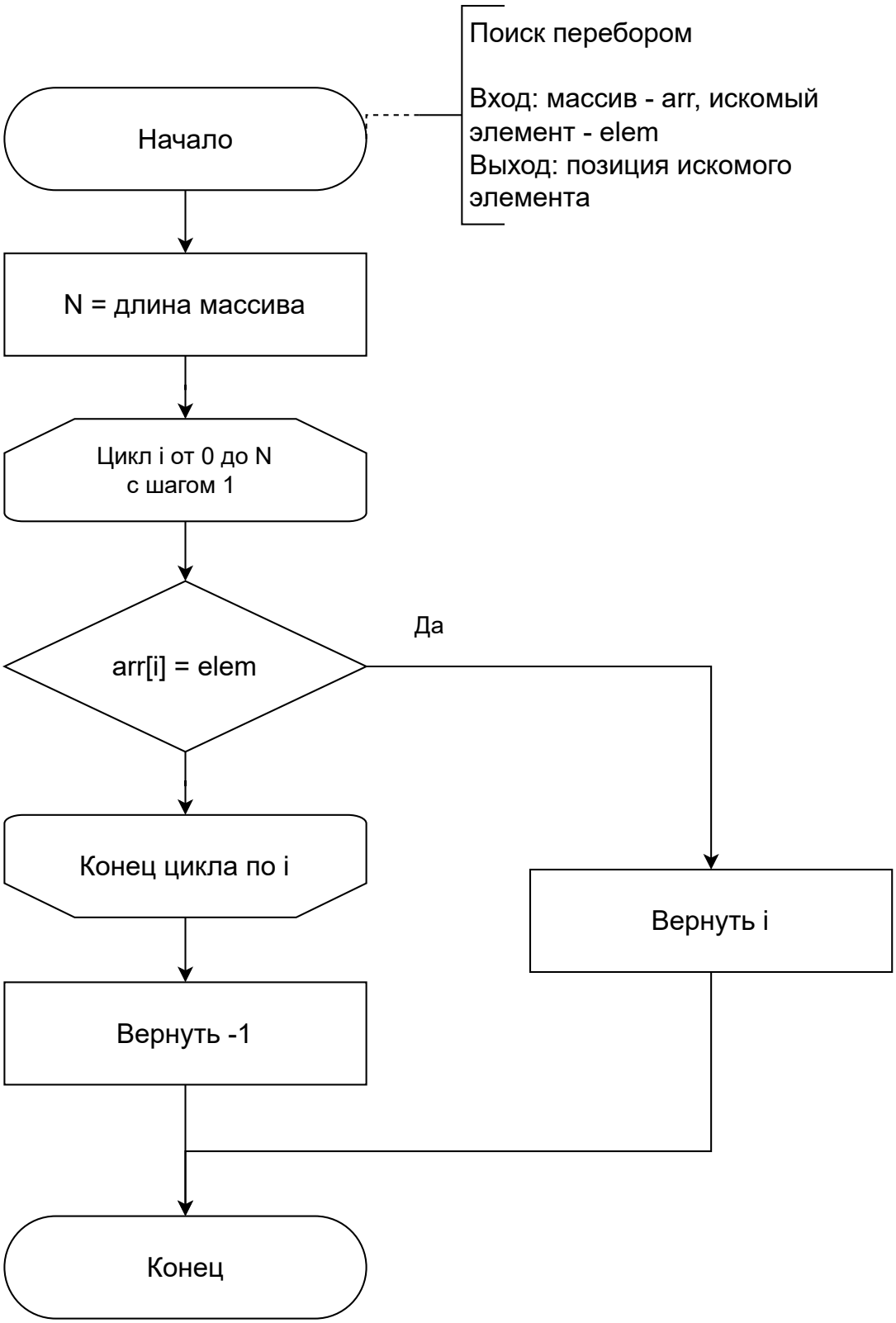

Рисунок 2.1 – Алгоритм поиска в массиве полным перебором

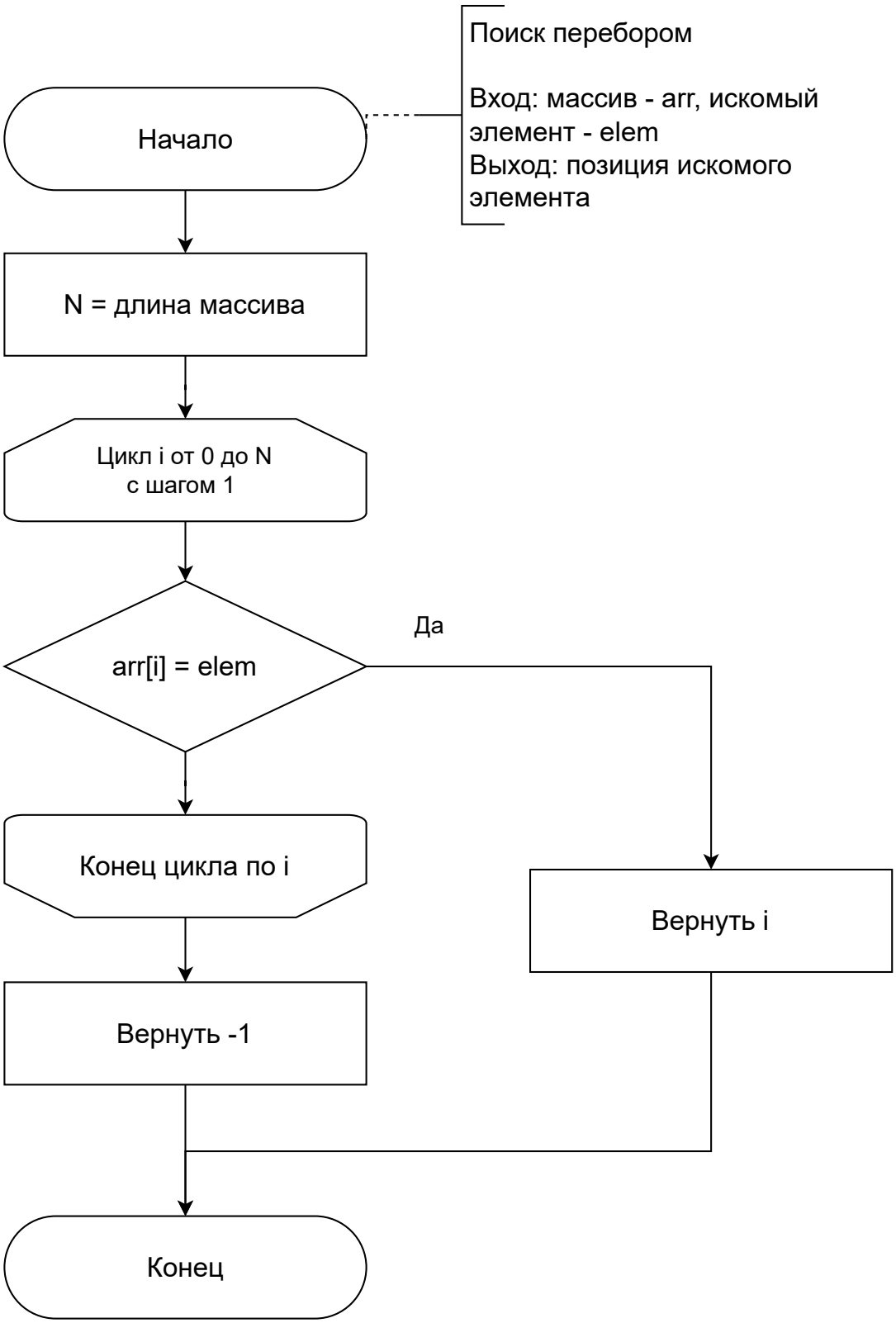

Рисунок 2.2 – Алгоритм бинарного поиска в массиве

3 Технологический раздел

3.1 Требования к программе

К программе предъявлены следующие требования:

— на вход подается массив и элемент, который необходимо найти;

— результатом работы программы является целое число — индекс искомого элемента в массиве, или -1 в случае его отсутствия;

— необходимо наличие пользовательского интерфейса для выбора действий и алгоритма работы;

— необходимо наличие замера количества сравнений, необходимых для поиска элемента в массиве для каждого из реализованных алгоритмов.

3.2 Средства реализации

Для реализации данной лабораторной был выбран язык Python [2], так как он содержит все необходимые для реализаций алгоритмов инструменты.

3.3 Программные модули

Программа разбита на следующие модули:

— **main.py** — модуль, реализующий пользовательский интерфейс и содержащий точку входа в программу;

— **funcs.py** — модуль, содержащий функции алгоритмов поиска элемента в массиве;

— **cmp_counter.py** — модуль, содержащий позволяющие подсчитать количество сравнений функции алгоритмов поиска элемента в массиве

3.4 Реализация алгоритмов

Реализации алгоритмов представлены в листингах 3.1 и 3.2.

Листинг 3.1 – Реализация алгоритма поиска элемента в массиве полным перебором

```
def linear_search(arr, elem):
    cmp = 0
    res = -1
    for i in range(len(arr)):
        cmp += 1
        if elem == arr[i]:
            res = i
            return res, cmp
    return res, cmp
```

Листинг 3.2 – Реализация алгоритма бинарного поиска элемента в массиве

```
def bin_search(arr, elem):
    cmp = 0
    pairs = list(enumerate(arr))
    pairs.sort(key=lambda x: x[1])
    l, r = 0, len(arr) - 1
    while l <= r:
        m = (l + r) // 2
        cmp += 1
        if pairs[m][1] == elem:
            return pairs[m][0], cmp
        elif pairs[m][1] < elem:
            l = m + 1
        else:
            r = m - 1
    return -1, cmp
```

3.5 Функциональные тесты

Функциональные тесты приведены в таблице 3.1 и были пройдены успешно всеми реализациями алгоритмов.

Таблица 3.1 – Функциональные тесты

Массив	**Элемент**	**Индекс в массиве**	**Количество сравнений**	
			Перебор	**Двоичный поиск**
1 2 3 4 5 6	1	0	1	3
1 2 3 4 5 6	0	-	6	3
1 2 3 4 5 6	4	3	4	1
1 2 3 4 5 6	6	5	6	3

4 Исследовательский раздел

4.1 Временные характеристики

В качестве временной характеристики в этой работе используется количество сравнений, произведенное для поиска элемента. Для построения графиков были использованы функции библиотеки Matplotlib [3].

Исследование производится на массиве размером 1062.

На рисунках 4.1 и 4.2 представлены зависимости количества сравнений от позиции искомого элемента в массиве для поиска перебором и бинарного поиска соответственно. На рисунке 4.3 представлена упорядоченная зависимость количества сравнений от позиции искомого элемента в массиве.

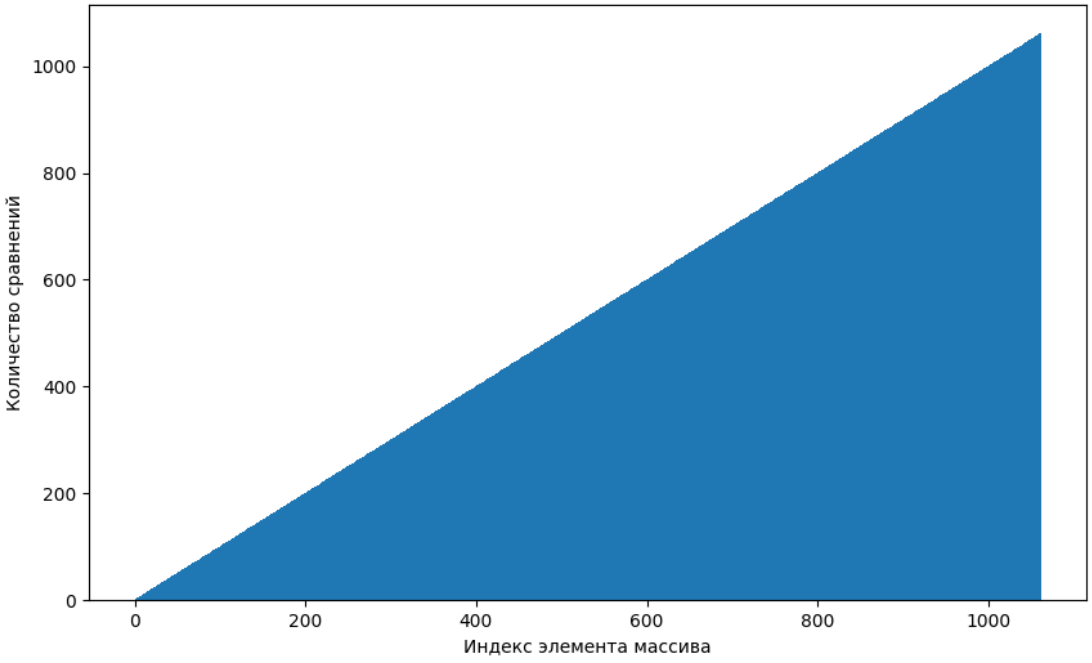

Рисунок 4.1 – Гистограмма количества сравнений для поиска перебором

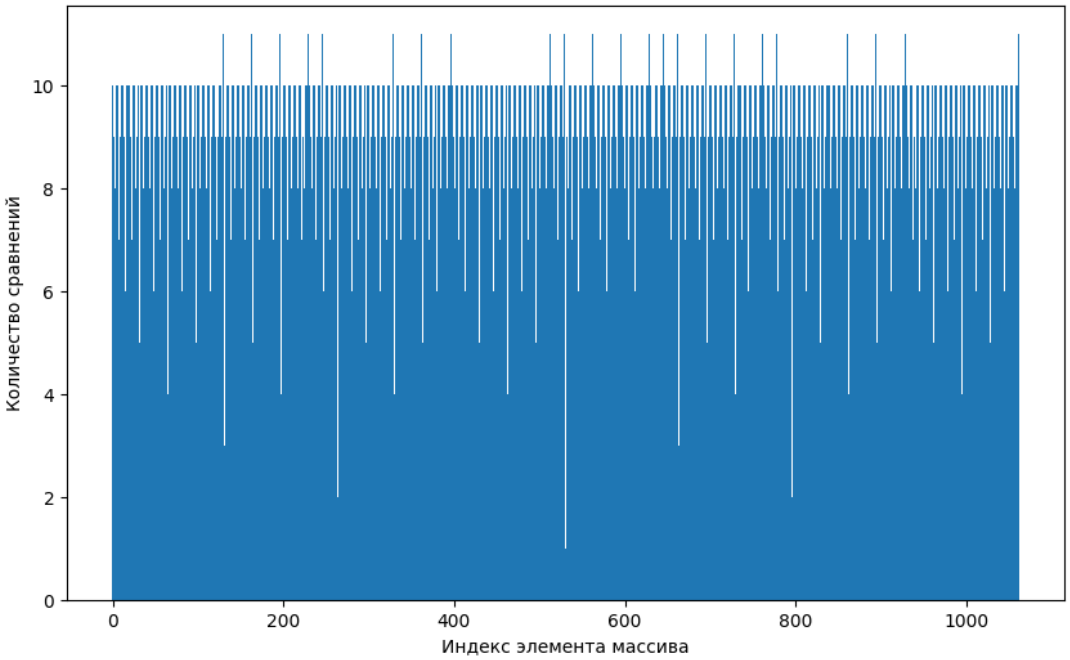

Рисунок 4.2 – Гистограмма количества сравнений для бинарного поиска

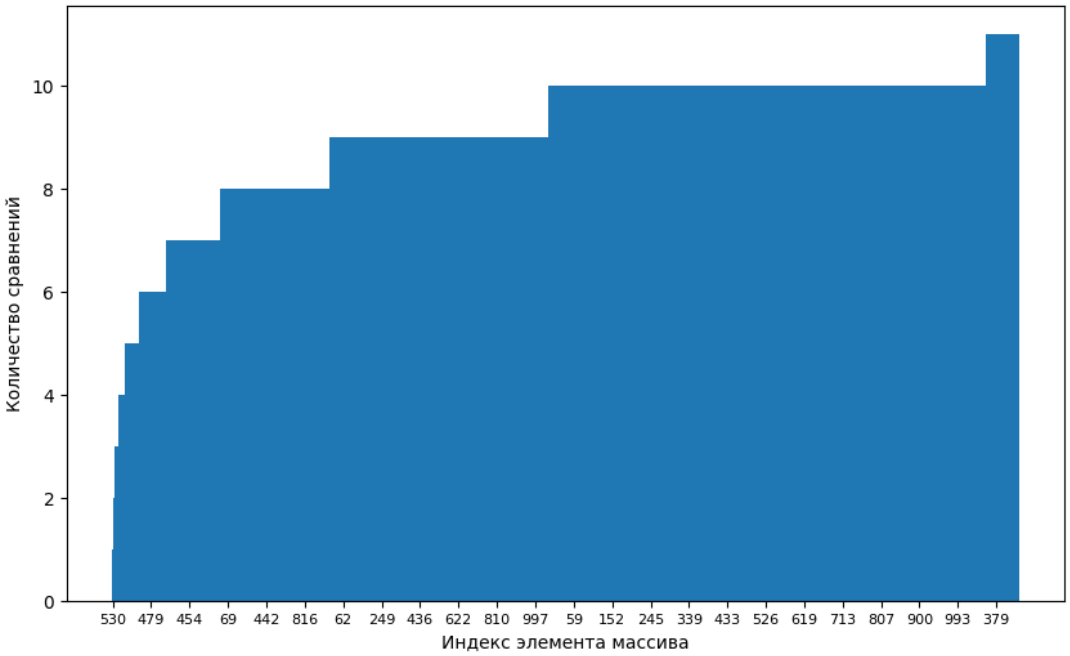

Рисунок 4.3 – Упорядоченная гистограмма количества сравнений для бинарного поиска

ЗАКЛЮЧЕНИЕ

В результате исследования сделан вывод, что так как для работы бинарного поиска требуется выполнять меньше сравнений, чем для поиска перебором, бинарный поиск является более быстрым алгоритмом. Но он требует предварительной сортировки массива, что приводит к дополнительным затратам времени.

Таким образом, в результате выполнения лабораторной работы были исследованы и реализованы алгоритмы поиска в массиве — поиск перебором и бинарный поиск, и проведена оценка их ресурсной эффективности, то есть выполнены все поставленные задачи, и цель выполнения работы — исследование алгоритмов поиска в массиве — можно считать достигнутой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Левитин А. В.* Алгоритмы. Введение в разработку и анализ //. — Вильямс, 2006. — С. 180—183.
2. Welcome to Python [Электронный ресурс]. — — Режим доступа: `https://www.python.org` (Дата обращения: 20.10.2024).
3. Документация модуля Matplotlib [Электронный ресурс]. — — Режим доступа: `https://matplotlib.org/stable/` (Дата обращения: 20.10.2024).