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"

Тема Поиск по словарю

Студент Гаврилюк В. А.

Группа ИУ7-51Б

Оценка (баллы) _____

Преподаватель Волкова Л. Л.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель лабораторной работы — сравнение алгоритмов нахождения заданного значения методами линейного и бинарного поиска. Для достижения поставленной цели необходимо выполнить следующие задачи:

— построить схемы для алгоритмов нахождения заданного значения методами линейного и бинарного поиска;

— создать программное обеспечение (ПО), реализующее перечисленные выше алгоритмы;

— провести анализ алгоритмов по количеству необходимых сравнений для нахождения каждого элемента массива.

— описать и обосновать полученные результаты в отчёте.

1 Аналитическая часть

В данном разделе будут рассмотрены алгоритмы линейного и бинарного поиска.

1.1 Линейный поиск

Алгоритм, основанный на линейном поиске, или поиске полным перебором, проходит по всему массиву, пытаясь отыскать целевой элемент [1]. Из этого следует, что если искомое значение находится в начале массива, то оно будет найдено быстрее, нежели если оно было бы расположено в конце. Линейный поиск работает как с отсортированными, так и с неотсортированными данными.

1.2 Бинарный поиск

Алгоритм бинарного (двоичного) поиска позволяет осуществлять быстрый поиск в массиве S отсортированных ключей. Чтобы найти ключ q, мы сравниваем значение q со средним ключом массива $S[n/2]$. Если значение ключа q меньше, чем значение ключа $S[n/2]$, значит, данный ключ должен находиться в верхней половине массива S; в противном случае он должен находиться в его нижней половине. Повторяется данный процесс на половине, гипотетически содержащей элемент q, пока значение ключа $S[n/2]$ не станет равным q или область поиска не станет пустой [2].

2 Конструкторская часть

В данном разделе будут приведены требования к входным, выходным параметрам и представлены схемы для алгоритмов линейного и бинарного поиска.

2.1 Требования к входным и выходным параметрам

Требования к входным и выходным параметрам:

— в качестве входных параметров алгоритм принимает массив и искомое значение;
— пустой массив является корректным входным значением;
— для бинарного поиска массив должен быть отсортирован;
— выходными параметрами являются два числа — индекс искомого значения и количество сравнений, потребовавшихся для нахождения данного значения;
— если искомое значение не найдено, в качестве индекса возвращается -1.

2.2 Схемы алгоритмов

На рисунках 1 и 2 представлены схемы алгоритмов линейного и бинарного поиска соответственно.

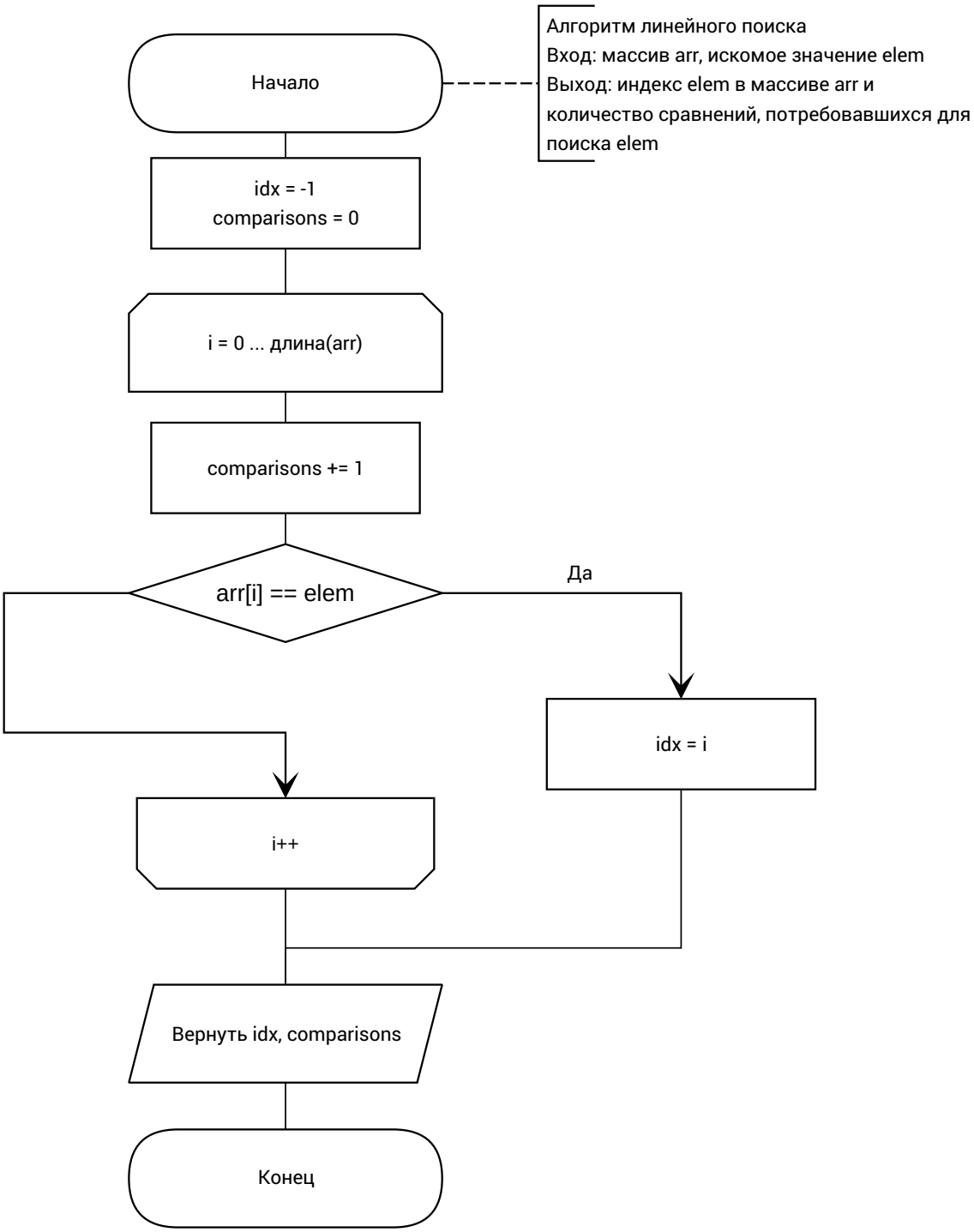


Рисунок 1 — Схема алгоритма линейного поиска

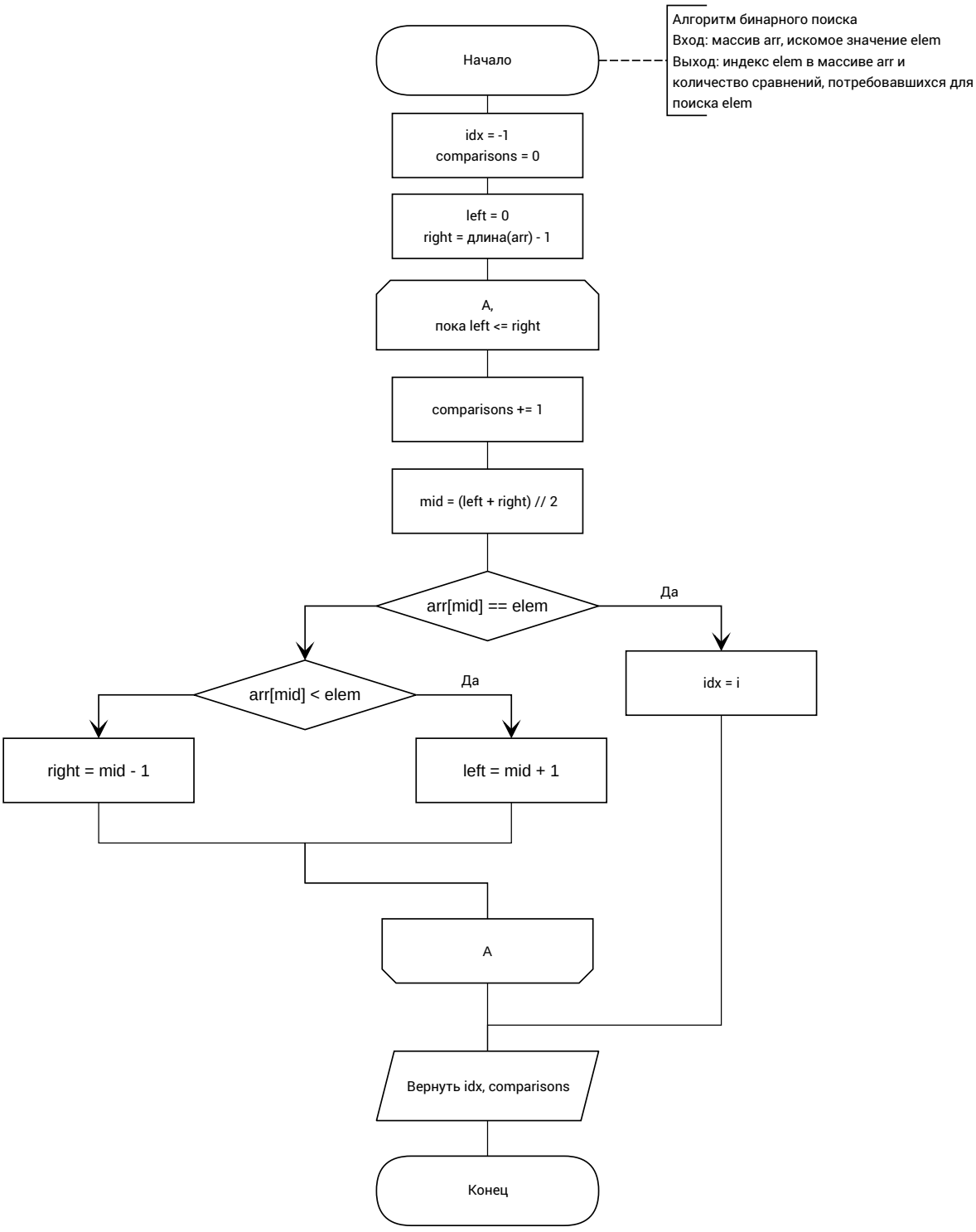


Рисунок 2 — Схема алгоритма бинарного поиска

3 Технологическая часть

В данном разделе будет представлена реализация алгоритмов линейного и бинарного поиска. Также будут указаны средства реализации и результаты тестирования.

3.1 Средства реализации

Для реализации был выбран язык программирования $Python$ [3]. Выбор обусловлен наличием библиотеки $matplotlib$ [4]. Для построения гистограмм использовалась функция bar [5].

3.2 Реализация алгоритмов

В листингах 1 — 2 представлены реализации алгоритмов линейного и бинарного поиска.

Листинг 1: Реализация алгоритма линейного поиска

```
1  def linearSearch(arr: list[int], elem: int) -> tuple[int, int]:
2      idx, comparisons = -1, 0
3
4      for i in range(len(arr)):
5          comparisons += 1
6          if arr[i] == elem:
7              idx = i
8              break
9
10     return idx, comparisons
```

Листинг 2: Реализация алгоритма бинарного поиска

```python
def binarySearch(arr: list[int], elem: int) -> tuple[int, int]:
    idx, comparisons = -1, 0
    left, right = 0, len(arr) - 1

    while left <= right:
        comparisons += 1
        mid = (left + right) // 2
        if arr[mid] == elem:
            idx = mid
            break
        elif arr[mid] < elem:
            left = mid + 1
        else:
            right = mid - 1

    return idx, comparisons
```

3.3 Тестирование

В таблице 2 представлены тесты для алгоритмов линейного и бинарного поиска. Тестирование проводилось по методологии чёрного ящика. В качестве входных данных использовались массивы из таблицы 1. Все тесты пройдены успешно.

Таблица 1 — Входные массивы для тестирования алгоритмов

Название массива	**Массив**
sorted_array	`[-1, -2, 3, 4, 5, 6, 7, 8, 9, 10]`
array	`[1, -10, 3, 4, 8, -6, 2, 7, 9, 8, -5]`
empty_array	`[]`

Таблица 2 — Тесты для алгоритмов линейного и бинарного поиска

№	Алгоритм	Описание	Название массива	Результат
1	Линейный поиск	Поиск значения 7 в отсортированном массиве.	sorted_array	(6, 7)
2	Линейный поиск	Поиск значения 7 в неотсортированном массиве.	array	(7, 8)
3	Линейный поиск	Поиск значения 11 в неотсортированном массиве, когда значение отсутствует.	array	(-1, 11)
4	Линейный поиск	Поиск значения 11 в пустом массиве.	empty_array	(-1, 0)
5	Бинарный поиск	Поиск значения 7 в отсортированном массиве.	sorted_array	(6, 4)
6	Бинарный поиск	Поиск значения 11 в отсортированном массиве, когда значение отсутствует.	sorted_array	(-1, 4)
7	Бинарный поиск	Поиск значения 11 в пустом массиве.	empty_array	(-1, 0)

4 Исследовательская часть

4.1 Оценка алгоритмов

В данном разделе будет проведено сравнение алгоритмов линейного и бинарного поиска по количеству сравнений, потребовавшихся для получения ответа. Длина массива равна $n = 1020$.

Для алгоритма поиска полным перебором существует $n + 1$ возможных исходов: n случаев расположения ключа в массиве и случай, когда ключ не найден. В худшем случае ключ может либо находиться в самом конце массива, либо отсутствовать вовсе, при этом количество сравнений в худшем случае будет равно n. В лучшем случае ключ будет находиться в начале массива и для его нахождения потребуется только одно сравнение. На рисунке 3 представлена гистограмма, отображающая количество сравнений для каждого элемента в массиве при линейном поиске. Крайний правый и крайний левый столбцы гистограммы соответствуют худшим случаям.

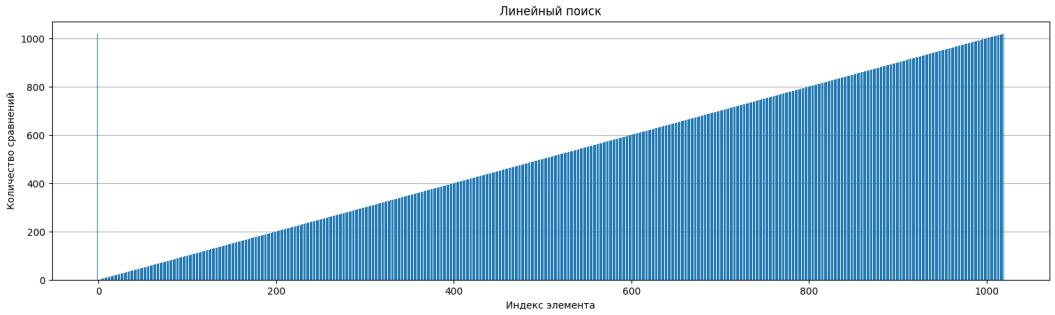


Рисунок 3 — Количество сравнений для каждого элемента в массиве при линейном поиске

На рисунке 4 представлена гистограмма, отображающая количество сравнений для каждого элемента в массиве при бинарном поиске. Для алгоритма бинарного поиска количество сравнений в худшем случае не превышает $log_2(n)$.

На рисунке 5 представлена гистограмма, на которой элементы отсортированы по количеству сравнений при бинарном поиске. В таком случае количество сравнений под индексом i ($i = \overline{0, n-1}$) соответствует максимальному количеству сравнений для массива длины $i + 1$.

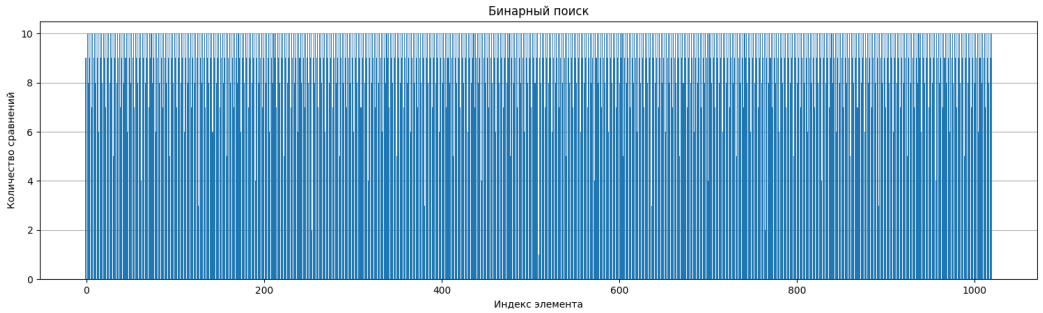


Рисунок 4 — Количество сравнений для каждого элемента в массиве при бинарном поиске

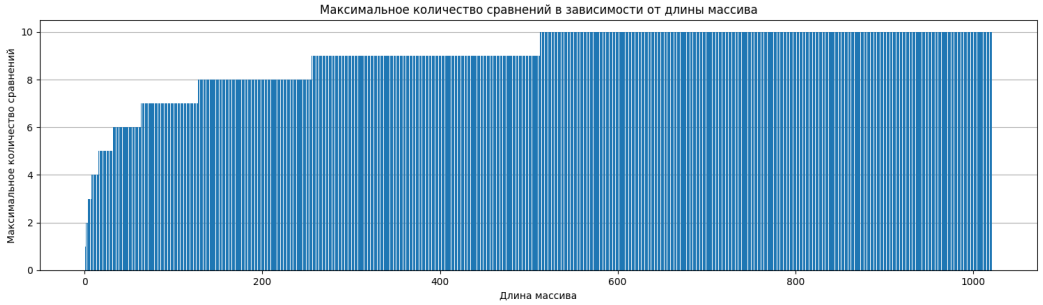
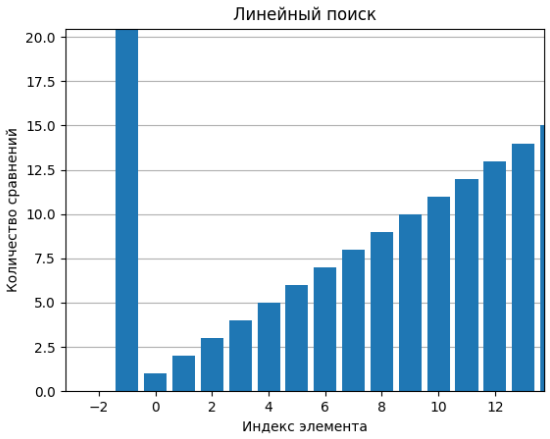
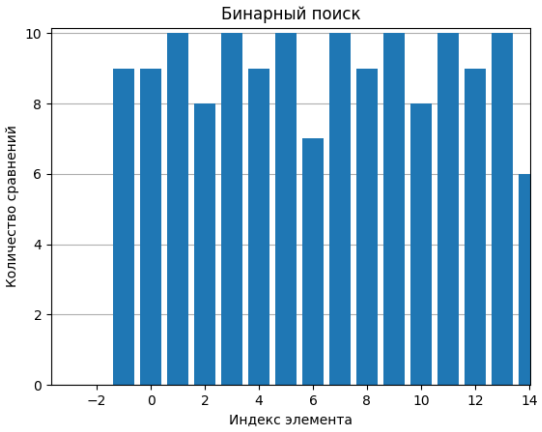


Рисунок 5 — Максимальное количество сравнений при бинарном поиске в зависимости от длины массива

На рисунке 6 представлены приближенные версии гистограмм, отображающих количество сравнений для каждого элемента при линейном и бинарном поиске на отсортированном массиве.



а) Линейный поиск



б) Бинарный поиск

Рисунок 6 — Количество сравнений при линейном и бинарном поиске на отсортированном массиве

В таблице 3 продемонстрировано количество сравнений при линейном и бинарном поиске при индексах $i = \overline{0{,}10}$. Для индексов, меньших $log_2(1020) \approx 9.994$, количество сравнений при линейном поиске меньше или равно количеству сравнений при бинарном поиске.

Таблица 3 — Количество сравнений при линейном и бинарном поиске при индексах $i = \overline{0, 10}$

Индекс	**Линейный поиск**	**Бинарный поиск**
0	1	9
1	2	10
2	3	8
3	4	10
4	5	9
5	6	10
6	7	7
7	8	10
8	9	9
9	10	10
10	11	8

4.2 Вывод

При поиске заданного ключа в массиве полным перебором количество сравнений растёт линейно с увеличением индекса ключа. В случае, если массив отсортирован и для нахождения элемента используется бинарный поиск, то количество сравнений не будет превышать $log_2(n)$, где n — размер массива. Хотя скорость роста функции $log_2(n)$ меньше (при увеличении n), чем у функции n, для отсортированного массива количество сравнений при линейном поиске может быть меньше или равно количеству сравнений при бинарном поиске, если индекс искомого ключа меньше $log_2(n)$. В исходном массиве, длиной $n = 1020$, для элементов, расположенных по индексам $i = \overline{0, 9}$, количество сравнений при линейном поиске будет меньше или равно количеству сравнений при бинарном поиске.

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы были проанализированы алгоритмы линейного и бинарного поиска посредством сопоставления количества сравнений для поиска заданного элемента в массиве, длина которого составляла $n = 1020$. Экспериментально было подтверждено, что линейный поиск требует n сравнений в худшем случае, тогда как бинарный поиск не превышает $log_2(n)$.

Исследование также показало, что алгоритм линейного поиска может иметь меньшее количество сравнений, чем в бинарном поиске при индексах, меньших $log_2(n)$. При больших объемах данных предпочтительным остаётся бинарный поиск, но для его использования данные должны быть отсортированы.

В ходе лабораторной работы были выполнены все поставленные задачи, а именно:

— построены схемы для алгоритмов нахождения заданного значения методами линейного и бинарного поиска;

— создано программное обеспечение (ПО), реализующее перечисленные выше алгоритмы;

— проведён анализ алгоритмов по количеству необходимых сравнений для нахождения каждого элемента массива.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стивенс Р. Алгоритмы. Теория и практическое применение. 2-е издание. — Москва : Эксмо, 2024.
2. Скиена С. Алгоритмы. Руководство по разработке. — Санкт-Петербург : БХВ, 2022.
3. Welcome to Python [Электронный ресурс]. — Режим доступа: https://www.python.org/ (дата обращения: 07.11.2024).
4. Matplotlib 3.9.2 documentation [Электронный ресурс]. — Режим доступа: https://matplotlib.org/stable/#matplotlib-release-documentation (дата обращения: 07.11.2024).
5. matplotlib.pyplot.bar [Электронный ресурс]. — Режим доступа: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html# (дата обращения: 07.11.2024).