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»

Студент ИУ7-52Б _____ Новиков А. А.
(Группа) (Подпись, дата) (И. О. Фамилия)

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

2024 г.

Оглавление

ВВЕДЕНИЕ

Поиск заданного значения в массиве или проверка существования этого значения в нем — часто встречающаяся задача при работе с массивами. Существуют различные алгоритмы для поиска заданного значения.

Цель лабораторной работы — сравнение алгоритмов нахождения заддного элемента в массиве методом полного перебора и методом бинарного поиска. Для достижения поставленной цели необходимо выполнить следующие задачи:

— реализовать алгоритмы нахождения заданного элемента в массиве методом линейного поиска и методом бинарного поиска;

— проанализировать реализации алгоритмов по количеству понадобившихся сравнений для нахождения каждого элемента;

— описать полученные результаты в отчете.

1 Аналитический раздел

В данном разделе рассмотрены алгоритмы нахождения заданного значения в множестве.

1.1 Описание алгоритмов

1.1.1 Алгоритм линейного поиска

При использовании алгоритма линейного поиска происходит перебор всех элементов множества [1]. Поэтому скорость нахождения значения зависит от его расположении в множестве.

1.1.2 Алгоритм бинарного поиска

В алгоритме бинарного поиска перебор всех элементов множества не осуществляется. Для его работы множество должно быть отсортировано. Определяются левая и правая границы поиска, после чего выбирается элемент, находящийся в середине этого диапазона, и он сравнивается с искомым значением. Если искомое значение меньше среднего элемента, то правая граница сдвигается влево, за средний элемент. Если искомое значение больше, то левая граница перемещается вправо. Этот процесс повторяется до тех пор, пока искомое значение не совпадет с центральным элементом или область поиска не сократится до нуля [2].

ВЫВОД

В данном разделе были рассмотрены алгоритм линейного поиска и алгоритм бинарного поиска заданного значения в множестве.

2 Конструкторский раздел

В данном разделе будут приведены схемы алгоритмов поиска заданного элемента в массиве методом линейного и бинарного поисков.

2.1 Представление алгоритмов

Алгоритмы на вход получают массив array, искомый элемент item. Возвращают индекс найденного элемента или иное значение, сигнализирующее об отсутствии искомого элемента в массиве. На рисунках 2.1 — 2.2 представлены схемы алгоритмов.

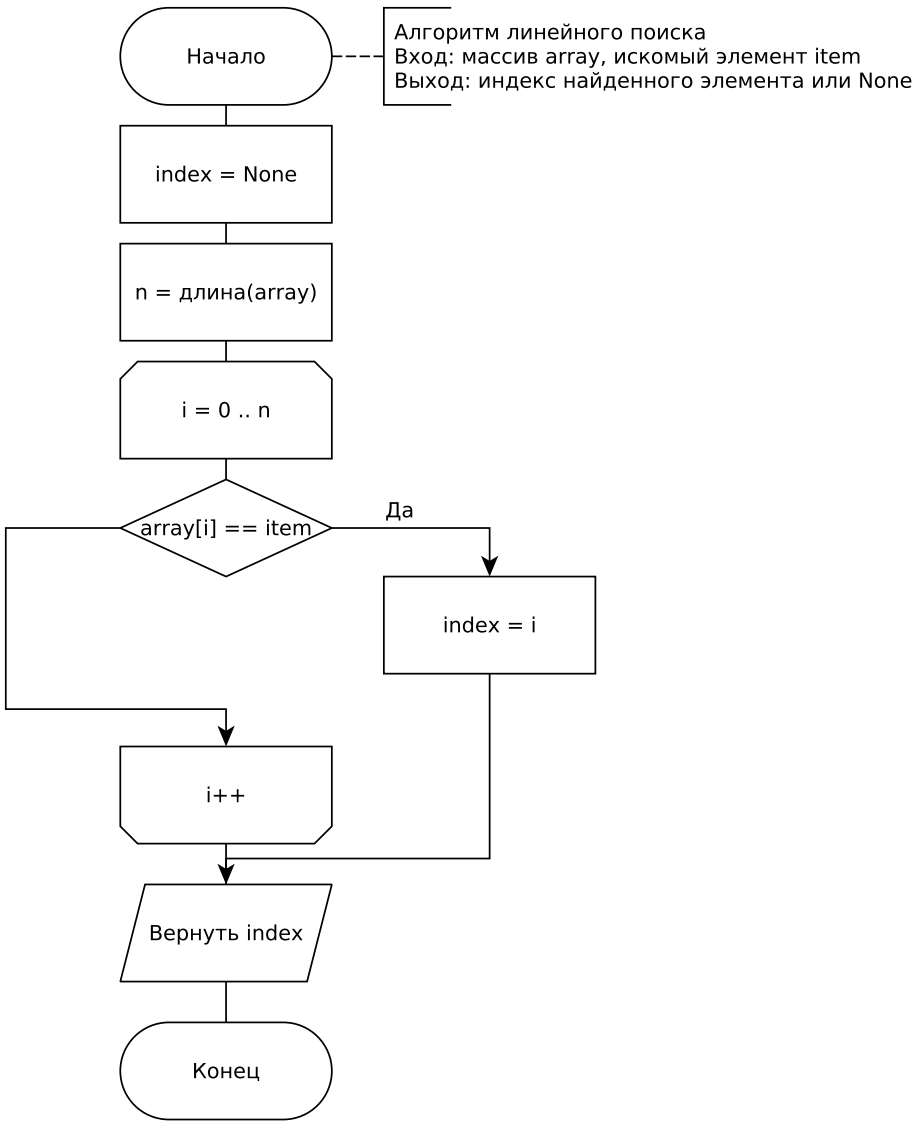

Рисунок 2.1 – Схема алгоритма линейного поиска в массиве

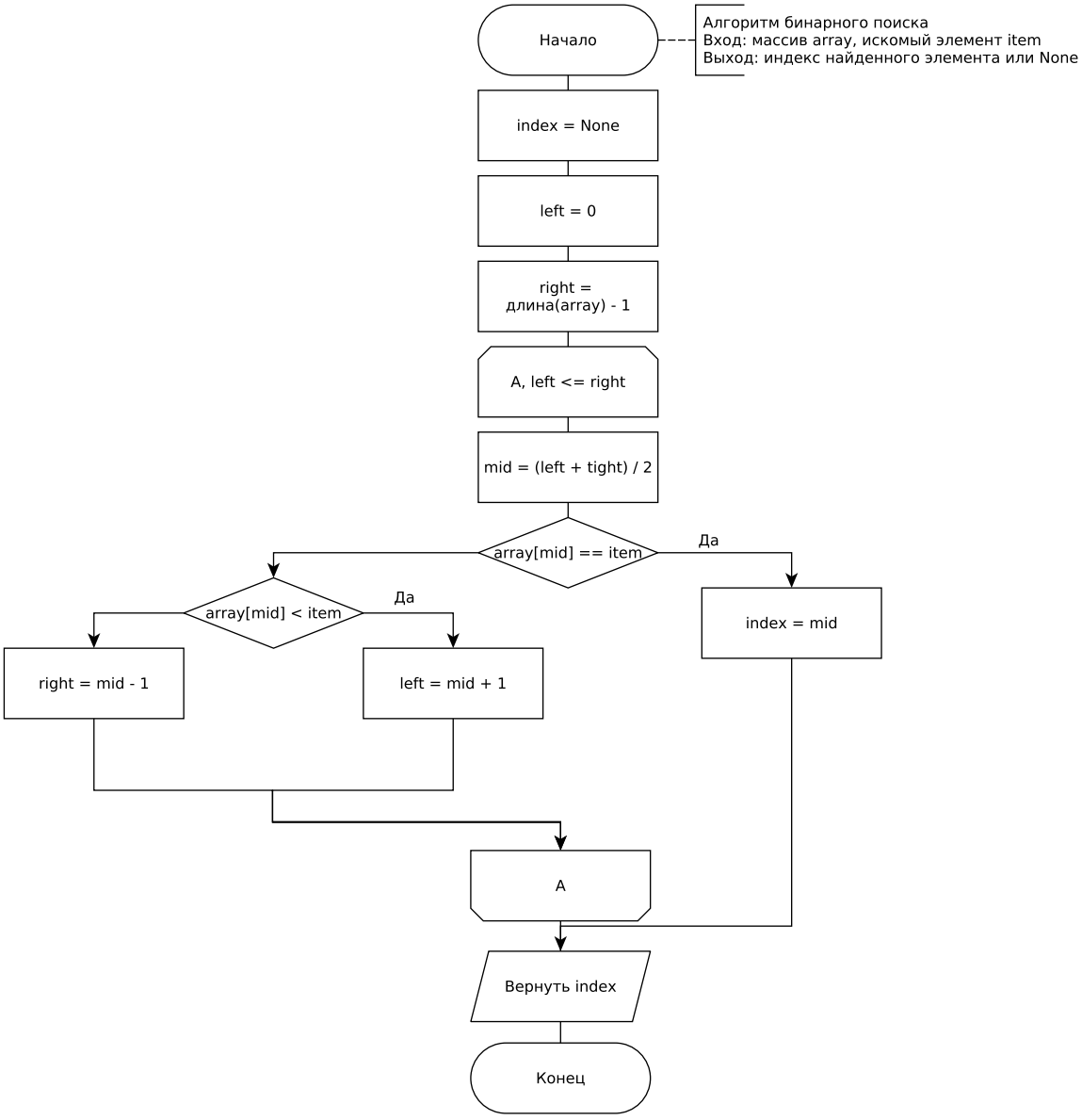

Рисунок 2.2 – Схема алгоритма бинарного поиска в массиве

ВЫВОД

В данном разделе были представлены схемы алгоритмов поиска заданного элемента в массиве методом линейного поиска и методом бинарного поиска.

3 Технологический раздел

В данном разделе описаны требования к программному обеспечению, реализация алгоритмов и средства реализации.

3.1 Требования к программному обеспечению

Входные данные: массив и искомый элемент; Выходные данные: индекс искомого элемента или значение, сигнализирующие об отсутствии искомого элемента в массиве.

3.2 Средства реализации

Для реализации данной лабораторной работы выбран язык программирования высокого уровня Python [3]. Выбор был обусловлен наличием библиотеки *matplotlib* [4]. Для построения графиков была выбрана функция *bar* [5].

3.3 Реализация алгоритмов

В листингах 3.1 — 3.2 представлены представлены реализации алгоритмов.

Листинг 3.1 – Реализация алгоритма линейного поиска

```
1  def linear_search(array: list, item: int) -> int | None:
2      n = len(array)
3      i = 0
4      index = None
5
6      while i < n:
7          if larray[i] == item:
8              index = i
9              break
10         i += 1
11
12     return index
```

Листинг 3.2 – Реализация алгоритма бинарного поиска

```
def binary_search(array: list, item: int) -> int | None:
    left = 0
    right = len(array) - 1
    index = None

    while left <= right:
        mid = (left + right) // 2
        if array[mid] == item:
            item = mid
            break

        if item < array[mid]:
            right = mid - 1
        else:
            left = mid + 1

    return item
```

ВЫВОД

В данном разделе были представлены реализации алгоритмов поиска заданного элемента в массиве методом линейного поиска и методом бинарного поиска, были рассмотрены средства реализации, предъявлены требования к программному обеспечению.

4 Исследовательский раздел

4.1 Технические характеристики

Технические характеристики, используемого устройства:

— Операционная система — Ubuntu Linux x86_64 [6];

— Память — 16 Гб;

— Процессор — AMD Ryzen 5 5500U (6x2.10 ГГц) [7].

4.2 Оценка алгоритмов

В данном разделе оценку трудоемкости алгоритма будем делать по количеству числа сравнений, понадобившихся для нахождения заданного элемента. Размер массива равен 1013.

В алгоритме линейного поиска количество возможных исходов равно длине массива: это n вариантов, если элемент присутствует в множестве, плюс еще один исход, когда элемент отсутствует. В худшем случае число сравнений составит n. Наихудший сценарий возникает, когда элемент либо отсутствует, либо находится в самом конце массива. Линейный поиск на отсортированном множестве работает быстрее, если индекс искомого элемента меньше приблизительно, чем $\log_2(n)$. На рисунке 4.1 показана гистограмма работы данного алгоритма.

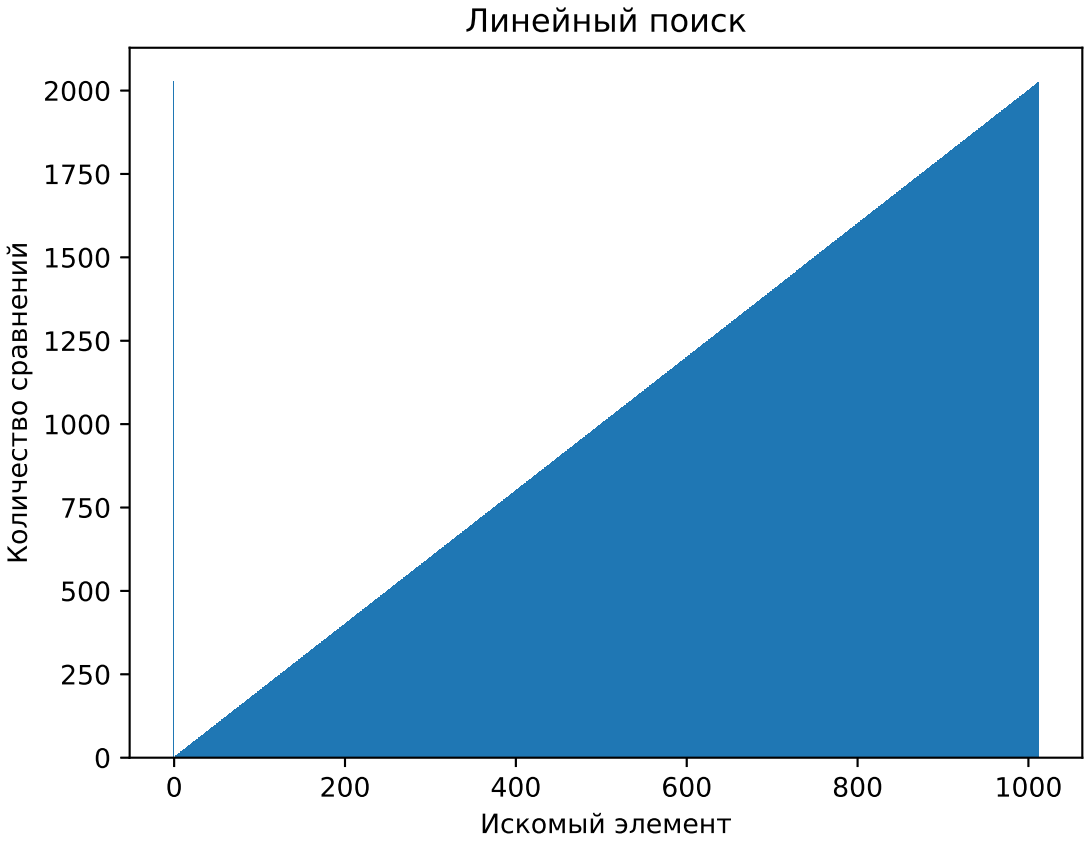

Рисунок 4.1 – Гистограмма алгоритма линейного поиска

В алгоритме двоичного поиска максимальное количество сравнений в худшем случае не превышает $\log_2(n)$. На рисунках 4.2 и 4.3 представлены гистограммы для этого алгоритма, при этом рисунок 4.3 демонстрирует график, где количество сравнений отсортировано по возрастанию.

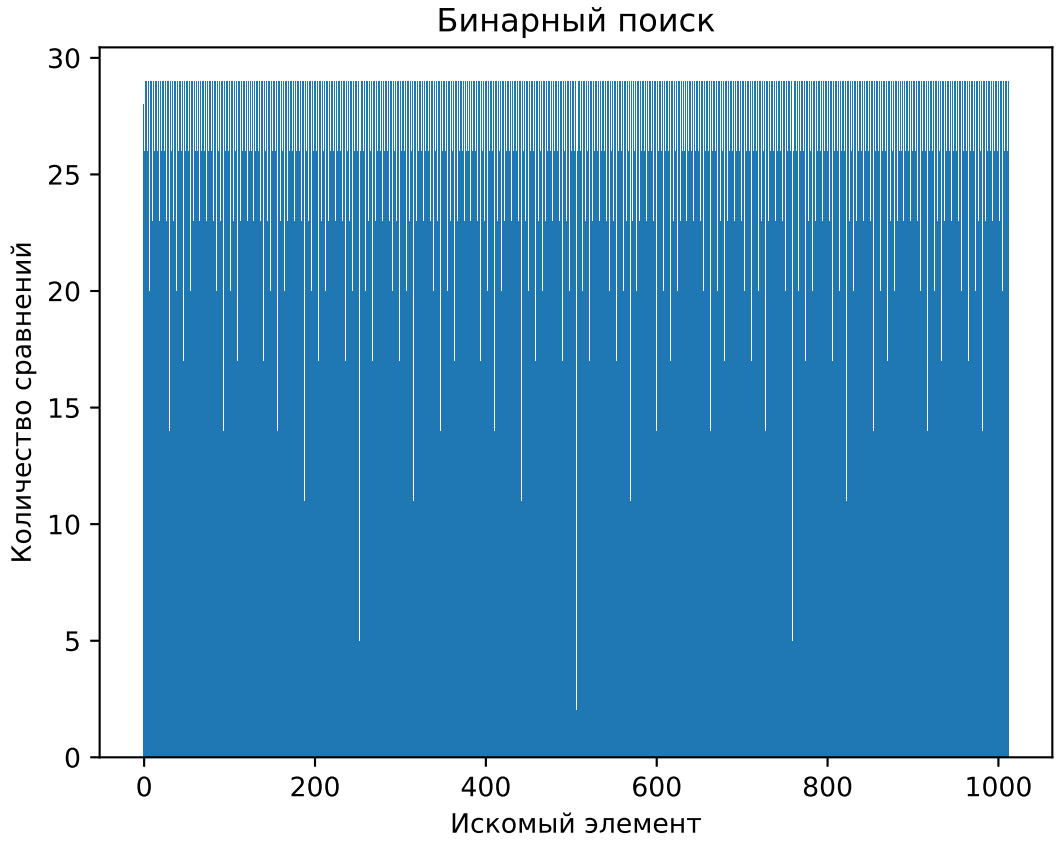

Рисунок 4.2 – Гистограмма алгоритма с двоичным поиском

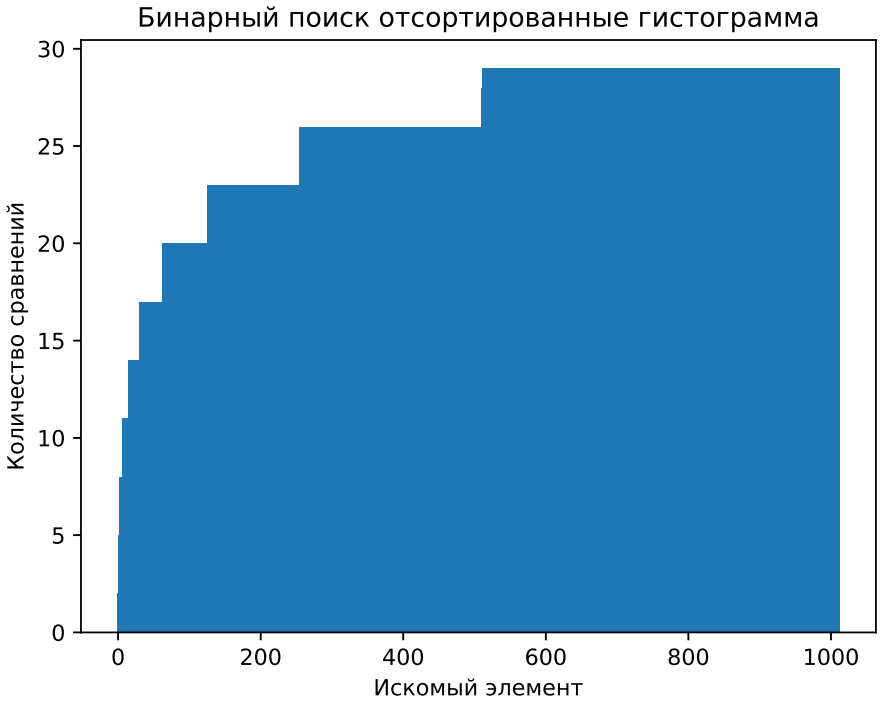

Рисунок 4.3 – Гистограмма алгоритма с двоичным поиском, отсортированная по количеству сравнений

4.3 Вывод

При линейном поиске количество сравнений постепенно увеличивается с ростом индекса искомого элемента. Для двоичного поиска количество сравнений, в целом, не зависит от расположения элемента относительно начала или конца массива, хотя элементы, находящиеся ближе к середине массива, будут найдены быстрее. В некоторых случаях линейный поиск может оказаться быстрее бинарного на отсортированном множестве, если искомое значение имеет индекс, меньший, чем $\log_2(n)$. Например, для массива из 1013 элементов линейный поиск будет эффективнее на индексах от 1 до 12.

В целом, двоичный поиск является более эффективным, поскольку в худшем случае требует не более $\log_2(n)$ сравнений, тогда как для полного перебора это значение составляет n.

ЗАКЛЮЧЕНИЕ

Экспериментально были выявлены различия между алгоритмами линейного поиска и бинарного поиска при поиске заданного элемента в множестве с использованием специально разработанного программного обеспечения.

Исследования подтвердили, что алгоритм полного перебора уступает бинарному поиску по числу сравнений, необходимых для нахождения элемента в массиве.

В ходе выполнения данной лабораторной работы были решены следующие задачи:

— реализованы алгоритмы нахождения заданного значения в массиве методом линейного поиска и методом бинарного поиска;

— проанализированы реализации алгоритмов по количеству сравнений для нахождения каждого элемента;

— описаны полученные результаты в отчете.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритм линейного поиска [Электронный ресурс]. — URL: https://kvodo.ru/lineynyiy-poisk.html (дата обращения 01.10.2024).
2. Алгоритм бинарного поиска [Электронный ресурс]. — URL: https://ru.hexlet.io/courses/basic-algorithms/lessons/binary-search/theory$_u nit$(01.10.2024).
3. Язык программирования Python [Электронный ресурс]. — URL: https://docs.python.org/3/ (дата обращения 01.10.2024).
4. Библиотека Python matplotlib [Электронный ресурс]. — URL: https://matplotlib.org/stable/index.html (дата обращения 01.10.2024).
5. matplotlib, функция bar [Электронный ресурс]. — URL: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html (дата обращения 01.10.2024).
6. Ubuntu technical documentation for developers and IT pros [Электронный ресурс]. — URL: https://ubuntu.com/tutorials (дата обращения 01.10.2024).
7. AMD Ryzen 5 5500U Processor [Электронный ресурс]. — https://www.amd.com/en/products/apu/amd-ryzen-5-5500u (дата обращения 01.10.2024).