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 дисциплине «Анализ Алгоритмов»

Тема

Студент Козырнов А.Д.

Группа ИУ7-52Б

Преподаватель Волкова Л. Л., Строганов Д.В.

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Очень часто требуется найти в массиве заданное значение или сообщить, что его там нет. Существуют разные алгоритмы поиска заданного значения.

Цель лабораторной работы — сравнение алгоритмов нахождения заданного значения способом полного перебора и методом двоичного поиска. Для достижения поставленной цели необходимо выполнить следующие задачи:

- реализовать указанные алгоритмы нахождения заданного значения;
- проанализировать реализации алгоритмов по количеству необходимых сравнений для нахождения каждого элемента множества.

1 Аналитическая часть

В данном разделе рассмотрены алгоритмы нахождения заданного значения в множестве.

1.1 Описание алгоритмов

Данные алгоритмы используются для поиска заданного значения в множестве.

1.1.1 Алгоритм поиска полным перебором

При использовании алгоритма поиска полным перебором происходит перебор всех элементов множества [1]. Это означает, если искомое значение лежит в начале множества, то оно будет найдено быстрее, чем если оно лежало в конце множества.

1.1.2 Алгоритм нахождения с помощью бинарного поиска

При использовании алгоритма с бинарным поиском не происходит перебор всех элементов множества. В этом алгоритме множество должно быть отсортированным. Вводятся левая и правая граница поиска. Выбирается центральный элемент в границе поиска и сравнивается с искомым значением. Если искомое значение меньше центрального элемента, то правая граница множества передвигается левее центрального элемента. Если искомое значение больше центрального элемента, то левая граница множества передвигается правее центрального элемента. Так повторяется, пока искомое значение не будет равно центральному элементу или область поиска сужается до нуля [2].

ВЫВОД

В данном разделе рассмотрены алгоритмы нахождения заданного значения в множестве.

2 Конструкторская часть

В данном разделе будут представлены схемы алгоритмов поиска заданного значения в массиве полным перебором и с помощью двоичного поиска.

2.1 Представление алгоритмов

Алгоритмы на вход получают массив array, значение value, а на выходе возвращают найденный индекс и количество потребованных операций. На рисунках 2.1 — 2.2 представлены схемы алгоритмов.

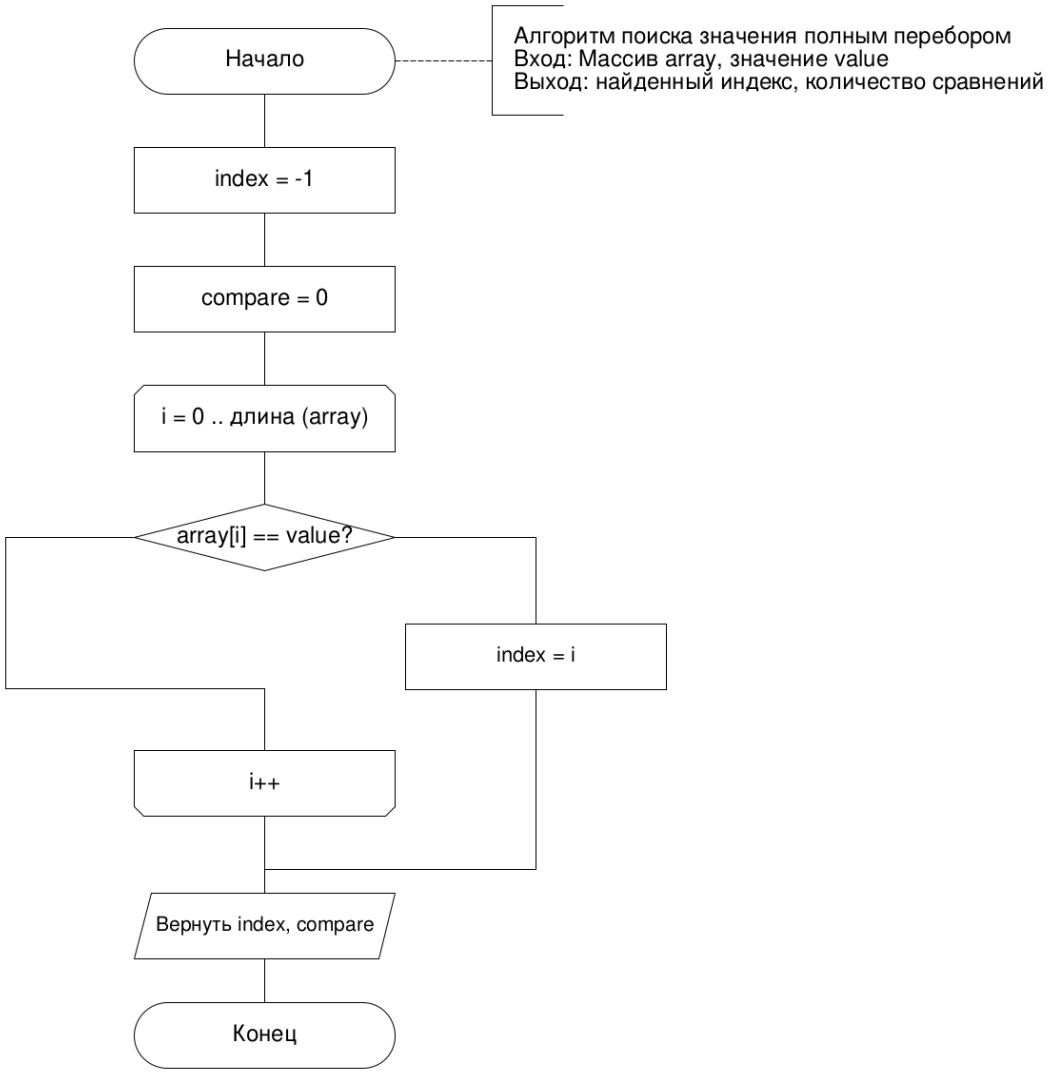

Рисунок 2.1 – Схема алгоритма поиска в массиве полным перебором

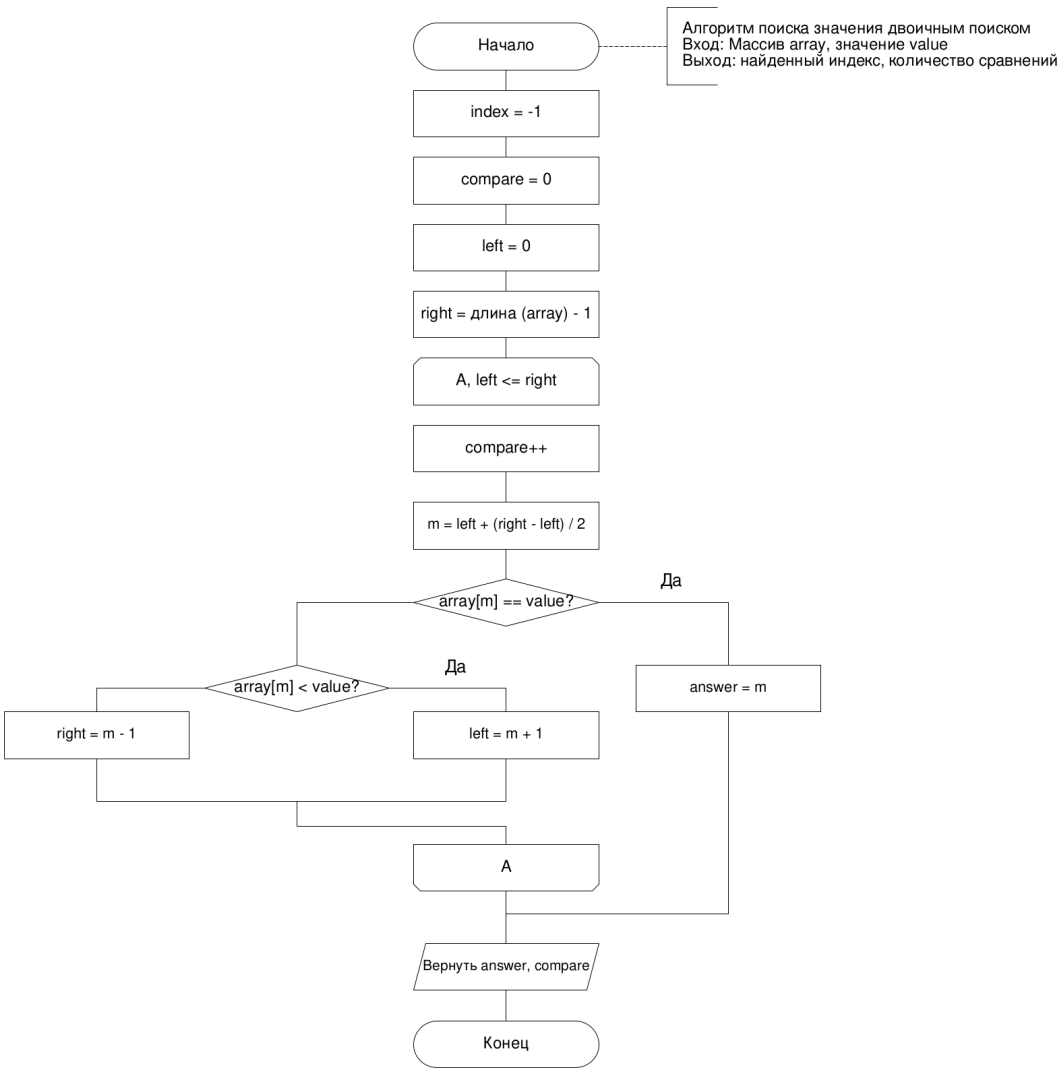

Рисунок 2.2 – Схема алгоритма поиска в массиве с помощью двоичного поиска

ВЫВОД

В данном разделе были представлены схемы алгоритмов поиска заданного значения в массиве полным перебором и с помощью двоичного поиска.

3 Технологическая часть

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

3.1 Требования к программному обеспечению

Входные данные: Массив и искомое значение;
Выходные данные: Индекс и количество потребованных операций.

3.2 Средства реализации

Для реализации был выбран язык программирования Python [3]. Выбор обусловлен наличием библиотеки *matplotlib* [4]. Для построения графиков была использована функция *bar* [5].

3.3 Реализация алгоритмов

В листингах 3.1 — 3.2 представлены реализации алгоритмов.

Листинг 3.1 – Алгоритм нахождения значения полным перебором

```
1  def locate(source: list[int], x: int) -> (int, int):
2      index = -1
3      compare = 0
4
5      for i in range(len(source)):
6          compare += 1
7          if source[i] == x:
8              index = i
9              break
10
11     return index, compare
```

Листинг 3.2 – Алгоритм нахождения значения с помощью бинарного поиска

```
def bin_locate(source: list[int], x: int) -> (int, int):
    answer =-1
    compare = 0

    left = 0
    right = len(source) - 1
    while left <= right:
        compare += 1
        middle = (left + right) // 2
        if source[middle] == x:
            answer = middle
            break
        elif source[middle] < x:
            left = middle + 1
        else:
            right = middle - 1

    return answer, compare
```

ВЫВОД

В данном разделе были реализованы алгоритмы поиска заданного значения в массиве полным перебором и с помощью двоичного поиска, рассмотрены средства реализации, предусмотрены требования к программному обеспечению.

4 Исследовательская часть

4.1 Технические характеристики

Характеристики используемого оборудования:

- Операционная система — Linux [6]
- Память — 16 Гб.
- Процессор — AMD Ryzen 7 5800H [7]

4.2 Оценка алгоритмов

В данном разделе оценку трудоемкости алгоритма будем дана в терминах числа сравнений, которые понадобились для нахождения ответа. Размер массива равен 113.

Для алгоритма с полным перебором количество количество исходов равно длине массиве: n исходов, если элемент в множестве и еще один исход, когда элемент не в множестве. Причем количество сравнений в худшем случае равно n. Худший случай - это случай, когда элемента или нет в множестве, или находится в самом его конце. Также линейный поиск работает быстрее на отсортированном множестве, если искомое значение лежит на индексе, примерно меньшем $\log_2(n)$. На рисунке 4.1 представлена гистограмма алгоритма.

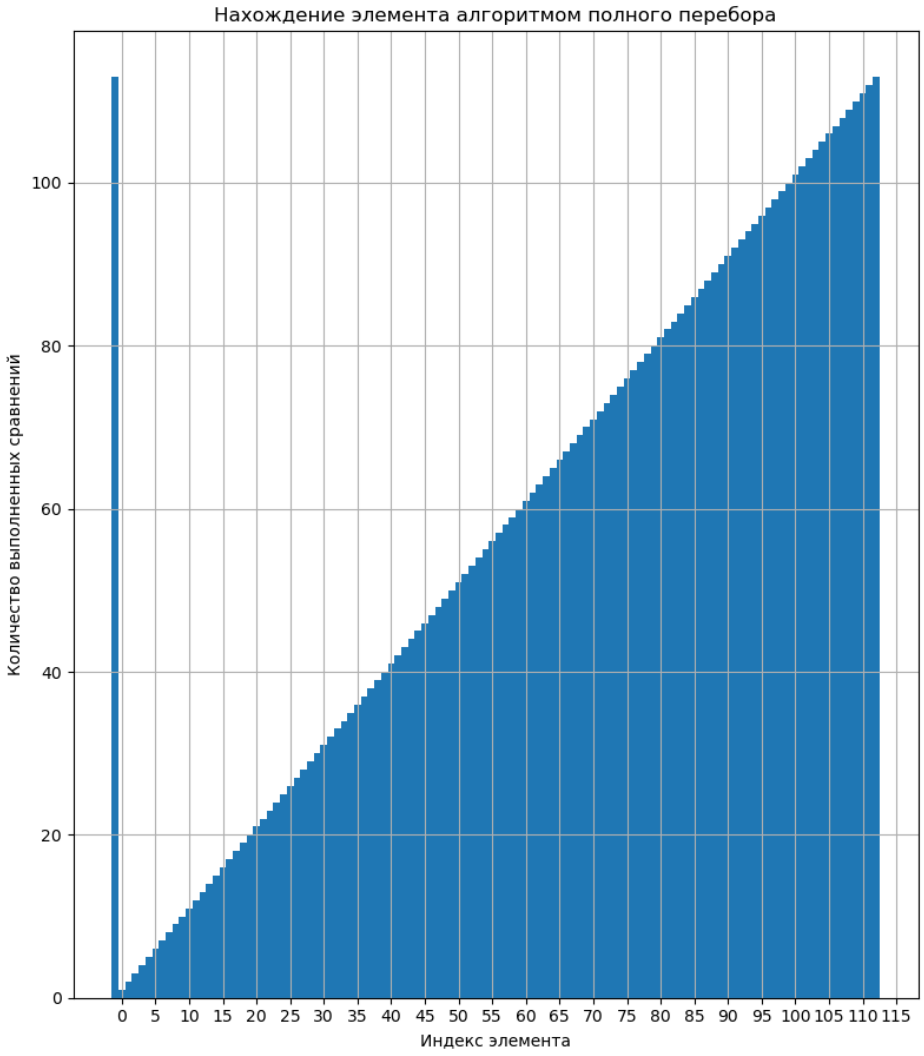

Рисунок 4.1 – Гистограмма алгоритма с полным перебором

Для алгоритма с двоичным поиском наибольшее количество сравнений не превышает $log_2(n)$ в худшем случае. На рисунке 4.2 и 4.3 представлены гистограммы алгоритма двоичного поиска. На рисунке 4.3 изображена гистограмма алгоритма с двоичным поиском, где количество сравнений отсортировано по возрастанию.

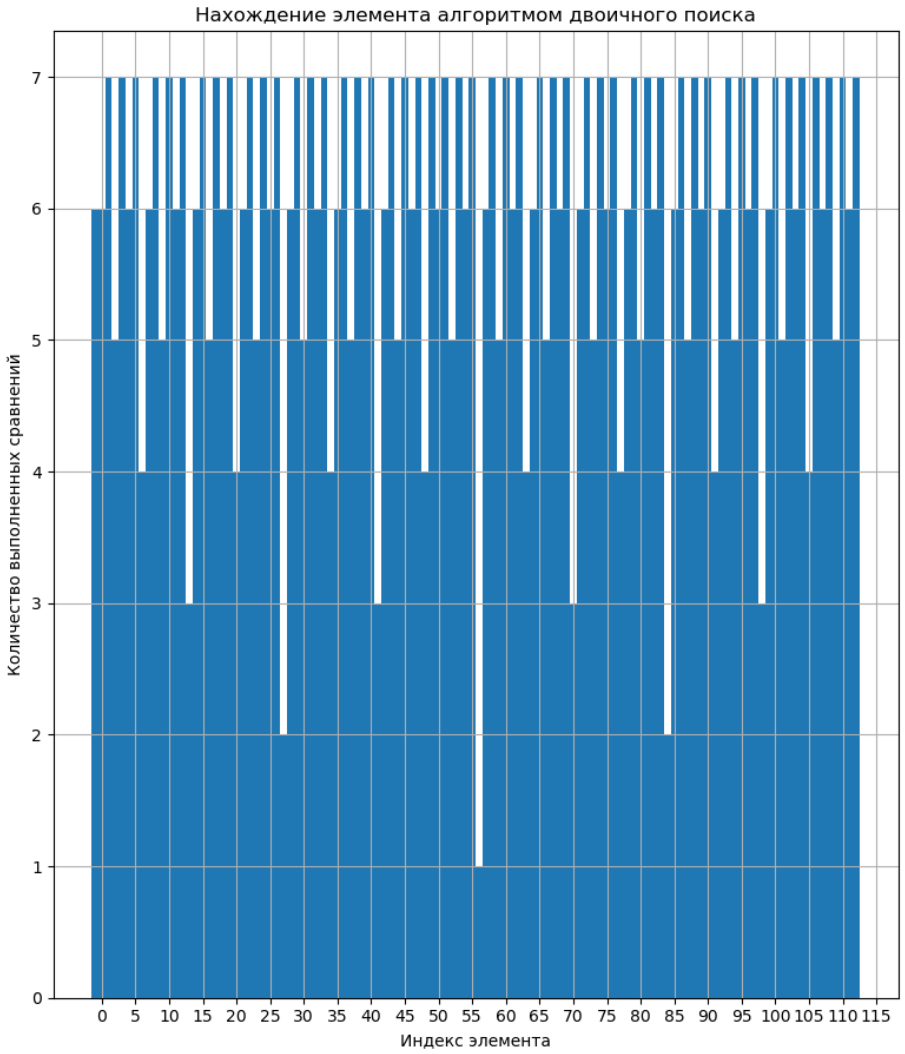

Рисунок 4.2 – Гистограмма алгоритма с двоичным поиском

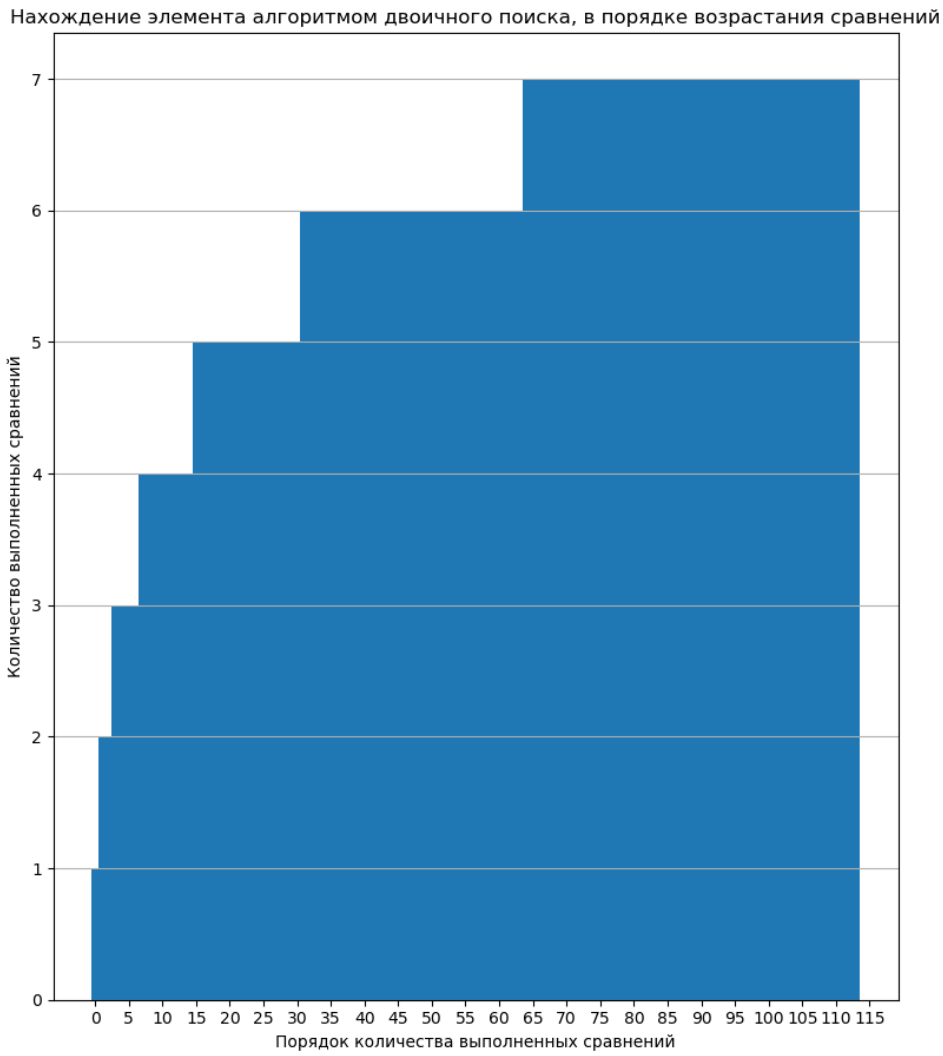

Рисунок 4.3 – Гистограмма алгоритма с двоичным поиском, отсортированная по количеству сравнений

4.3 Вывод

Для поиска заданного значения в массиве полным перебором количество сравнений растет постепенно с увеличением индекса искомого элемента. Для поиска заданного значения в массиве с помощью двоичного поиска количество сравнений в целом не зависит от его расположения относительно начала и конца массива. Однако оно будет найдено быстрее, если лежит в середине массива, в четвертях и так далее. Однако линейный поиск может работать быстрее бинарного поиска в случаях на отсортированном наборе, если искомое значение примерно меньше индекса $\log_2(n)$, где n — размер набора. В случае, если размер набора равен 113, то при индексах 1 — 6 линейный поиск быстрее работает, чем двоичный поиск.

Наиболее эффективным алгоритмом является алгоритм с использованием двоичного поиска, потому что количество сравнений в худшем случае не превышает $log_2(n)$, когда в алгоритме с полным перебором количество в худшем случае равно n.

ЗАКЛЮЧЕНИЕ

Было экспериментально подтверждены различия между алгоритмами поиска заданного значения в множестве алгоритмами с полным перебором и с бинарным поиском при помощи разработанного программного обеспечения.

В результате исследований можно сделать вывод о том, что алгоритм с полным перебором уступает алгоритму с бинарным поиском по количеству проводимых сравнений для нахождения заданного значения.

В ходе выполнения данной лабораторной работы были решены следующие задачи:

- реализованы указанные алгоритмы нахождения заданного значения;
- проанализированы реализации алгоритмов по количеству необходимых сравнений для нахождения каждого значения;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Алгоритм линейного поиска [Электронныи ресурс] URL: https://kvodo.ru/lineynyiy-poisk.html (дата обращения: 29-09-2024)

[2] Алгоритм бинарного поиска [Электронныи ресурс] URL: https://ru.hexlet.io/courses/basic-algorithms/lessons/binary-search/theory_unit (дата обращения: 29-09-2024)

[3] Язык программирования Python [Электронныи ресурс] URL: https://docs.python.org/3/ (дата обращения: 29-09-2024)

[4] Библиотека Python matplotlib [Электронныи ресурс] URL: https://matplotlib.org/stable/index.html (дата обращения: 29-09-2024)

[5] matplotlib, функция bar [Электронныи ресурс] URL: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html#matplotlib.pyplot.bar (дата обращения: 29-09-2024)

[6] Операционная система Linux [Электронныи ресурс] URL: https://archlinux.org/ (дата обращения: 29-09-2024)

[7] Процессор AMD Ryzen 7 5800H, бенчмарк [Электронныи ресурс] URL: https://www.notebookcheck-ru.com/AMD-Ryzen-7-5800H.519526.0.html (дата обращения: 29-09-2024)