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 Поиск по словарю

Студент Смирнов И.В.

Группа ИУ7-52Б

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

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **3**

1 Аналитическая часть **4**

1.1 Поиск полным перебором 4

1.2 Поиск в упорядоченном массиве бинарным поиском 4

2 Конструкторская часть **5**

2.1 Представление алгоритмов 5

2.2 Трудоемкость алгоритмов в лучшем и худшем случаях 8

2.2.1 Алгоритм, использующий полный перебор 8

2.2.2 Алгоритм, использующий бинарный поиск 8

3 Технологическая часть **9**

3.1 Требования к программному обеспечению 9

3.2 Средства реализации 9

3.3 Реализация алгоритмов 9

4 Исследовательская часть **12**

4.1 Технические характеристики 12

4.2 Описание используемых типов данных 12

4.3 Гистограммы 12

4.4 Вывод 14

ЗАКЛЮЧЕНИЕ **15**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **16**

ВВЕДЕНИЕ

Пусть дан словарь, в котором требуется найти элемент. Чтобы успешно его обнаружить или выяснить, что его нет, необходимо применить алгоритм поиска. В данной работе рассматриваются два стандартных подхода для поиска элементов в словаре: полный перебор и бинарный поиск.

Цель лабораторной работы — сравнить два алгоритма поиска элемента (полного перебора и бинарного поиска) при работе со словарем. Для достижения поставленной цели необходимо выполнить следующие задачи:

- выполнить оценку трудоемкости разрабатываемых алгоритмов;
- реализовать алгоритмы нахождения значения в словаре;
- выполнить сравнительный анализ сложности двух алгоритмов на основе замеров количества сравнений для каждого элемента массива и данных лучшего и худшего случаев

1 Аналитическая часть

В данном разделе будут рассмотрены алгоритмы нахождения заданного значения в словаре.

1.1 Поиск полным перебором

Для поиска заданного значения, алгоритм начинает перебирать все значения массива с первого до последнего, пока не найдет нужный элемент. Элементы, расположенные в начале массива, будут найдены быстрее тех, что расположены ближе к концу.

1.2 Поиск в упорядоченном массиве бинарным поиском

Для поиска заданного значения, массив должен быть изначально отсортирован. Вводятся левая и правая граница поиска (изначально левая граница — первый элемент, правая граница — последний элемент), а также центральный элемент, который сравнивается с искомым значением. Если искомое значение меньше центрального элемента, то правая граница передвигается на место центрального элемента. Если искомое значение больше центрального элемента, то левая граница передвигается на место центрального элемента. Вводится новый центральный элемент для изменившихся границ. Так повторяется, пока искомое значение не будет равно центральному элементу или левая граница станет больше или равна правой.

ВЫВОД

В данном разделе рассмотрены алгоритмы нахождения заданного значения в массиве.

2 Конструкторская часть

В данном разделе будут представлены схемы алгоритмов поиска заданного значения в массиве полным перебором и с помощью двоичного поиска, а также будут оценены трудоемкости алгоритмов в лучшем/худшем случаях.

2.1 Представление алгоритмов

На вход алгоритмы получают массив $array$ и значение $target$, которое необходимо найти.

На выходе алгоритмы возвращают найденный индекс, а также количество сравнений. В случае отсутствия заданного элемента в массиве алгоритмы, в качестве найденного индекса, возвращают -1.

На рисунках 2.1 — 2.2 приведены схемы двух алгоритмов нахождения значения в массиве: использующий полный перебор, использующий бинарный поиск.

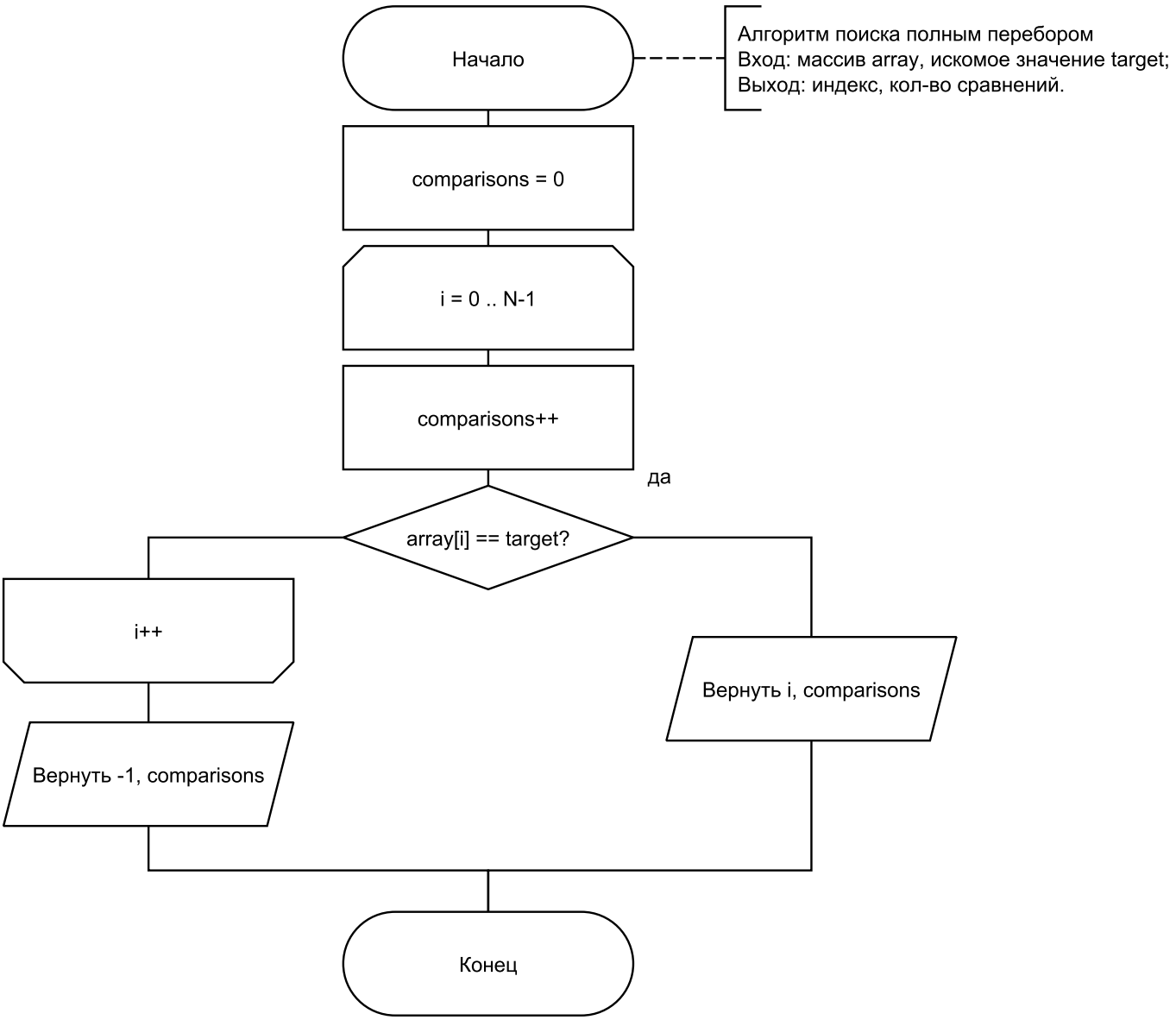

Рисунок 2.1 – Схема алгоритма поиска полным перебором

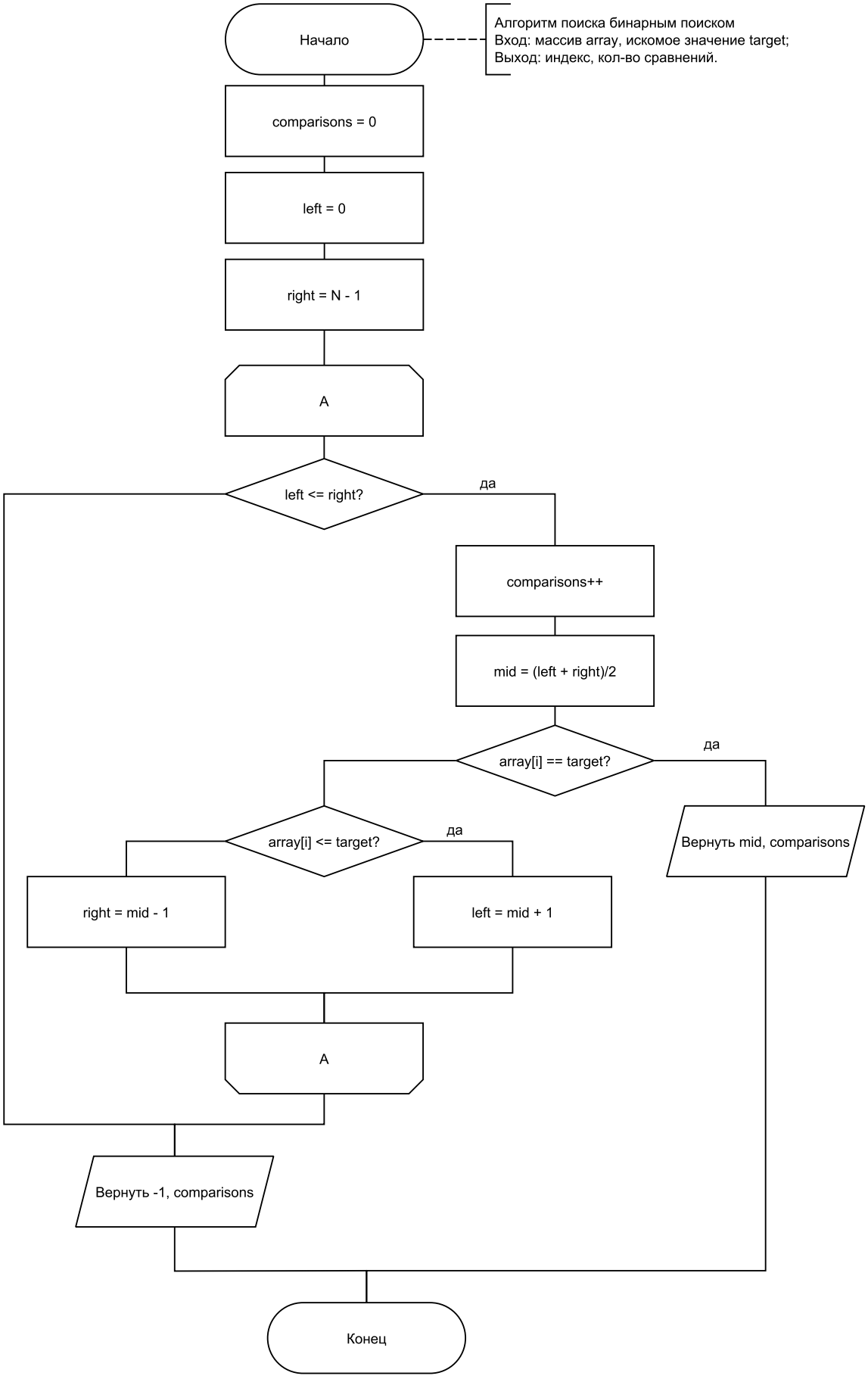

Рисунок 2.2 – Схема алгоритма поиска бинарным поиском

2.2 Трудоемкость алгоритмов в лучшем и худшем случаях

В данном подразделе будут оценены трудоемкости в лучшем и худшем случаях для двух алгоритмов.

2.2.1 Алгоритм, использующий полный перебор

В массиве из N элементов $\exists\, N+1$ возможный случай размещения исходного значения $target$ в массиве $array$, включая случай, когда $x \notin array$.

Лучший случай: исходное значение — первый элемент массива (с индексом 0). Трудоемкость в лучшем случае составляет $O(1)$.

Худший случай: исходное значение — последний элемент массива (с индексом $N-1$). Трудоемкость в худшем случае составляет $O(N)$.

2.2.2 Алгоритм, использующий бинарный поиск

Алгоритм бинарного поиска делит массив пополам на каждом шаге, уменьшая количество элементов, которые нужно проверить.

Лучший случай: искомое значение находится ровно в центре массива на первом шаге. Трудоемкость в лучшем случае составляет $O(1)$.

Худший случай: искомое значение отсутствует в массиве или расположено на границе одной из половин массива, что требует максимального количества шагов для поиска. Поскольку на каждом шаге массив делится на две части, общее количество шагов для поиска элемента в худшем случае равно $O(\log N)$.

ВЫВОД

В данном разделе были представлены схемы алгоритмов нахождения элемента в словаре, а также оценены трудоемкости для данных алгоритмов в лучшем/худшем случаях.

3 Технологическая часть

В данном разделе будут приведены требования к программному обеспечению, средства реализации, листинги кода.

3.1 Требования к программному обеспечению

Входные данные: массив и искомое значение;
Выходные данные: индекс найденного значения и количество сравнений.

3.2 Средства реализации

В данной работе для реализации был выбран язык программирования $C\#$ с использованием платформы $.NET$ 8 [1]. Выбор обусловлен наличием:

- возможности создавать массивы определенного размера: $int[N]$;
- встроенной поддержки $WindowsForms$, что позволяет создавать пользовательские интерфейсы и визуализировать данные. В частности, для построения гистограмм были использованы средства, представленные в пространстве имен $System.Windows.Forms.DataVisualization.Charting$ [2].

3.3 Реализация алгоритмов

В листингах 3.1 - 3.2 представлены реализации алгоритмов.

Листинг 3.1 – Алгоритм использующий полный перебор

```
1  private int FullSearch(int[] array, int target, out int comparisons)
2  {
3      comparisons = 0;
4      for (int i = 0; i < N; i++)
5      {
6          comparisons++;
7          if (array[i] == target)
8              return i;
9      }
10     return -1;
11 }
```

Листинг 3.2 – Алгоритм использующий полный перебор

```
1  private int BinarySearch(int[] array, int target, out int comparisons)
2  {
3      comparisons = 0;
4      int left = 0, right = N - 1;
5
6      while (left <= right)
7      {
8          comparisons++;
9          int mid = (left + right) / 2;
10         if (array[mid] == target)
11             return mid;
12         if (array[mid] < target)
13             left = mid + 1;
14         else
15             right = mid - 1;
16     }
17     return -1;
18 }
```

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 — Windows 11 Home [3]
- Память — 16 Гб.
- Процессор — Intel(R) Core(TM) i5-10300H CPU @ 2.50Ггц [4]

4.2 Описание используемых типов данных

Используемые типы данных: словарь — массив из N различных элементов типа int; искомый элемент — число типа int.

4.3 Гистограммы

На рисунках 4.1 — 4.3 приведены гистограммы, показывающие количество сравнений для каждого элемента словаря в трех вариациях:

- используется алгоритм полного перебора;
- используется алгоритм бинарного поиска на упорядоченном массиве;
- вторая вариация, где столбцы упорядочены в порядке возрастания.

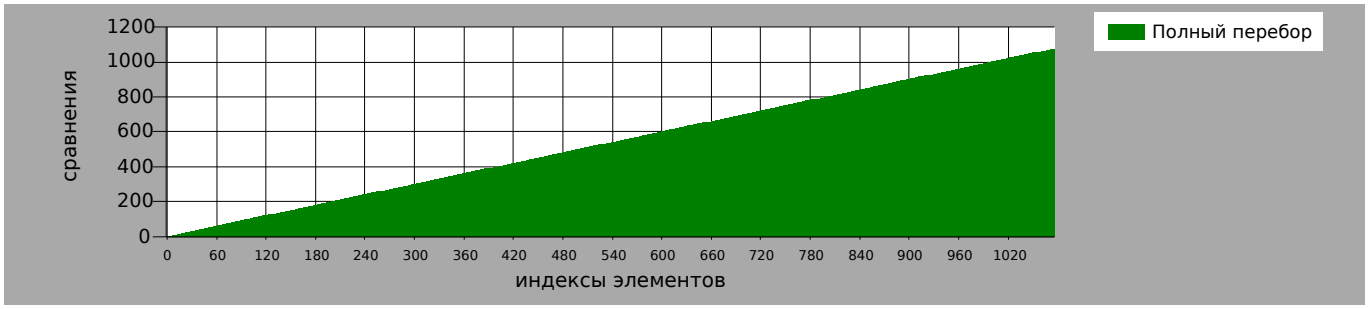

Рисунок 4.1 – Сравнения для каждого элемента в алгоритме полного перебора

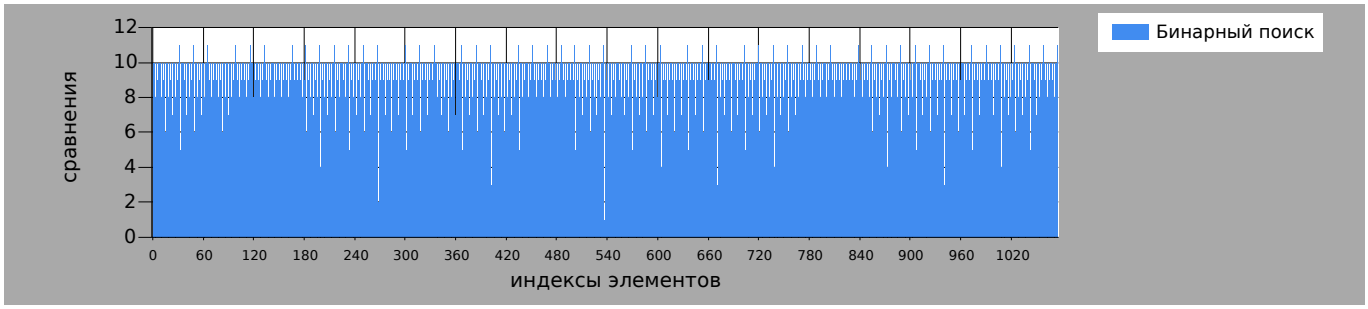

Рисунок 4.2 – Сравнения для каждого элемента в алгоритме бинарного поиска

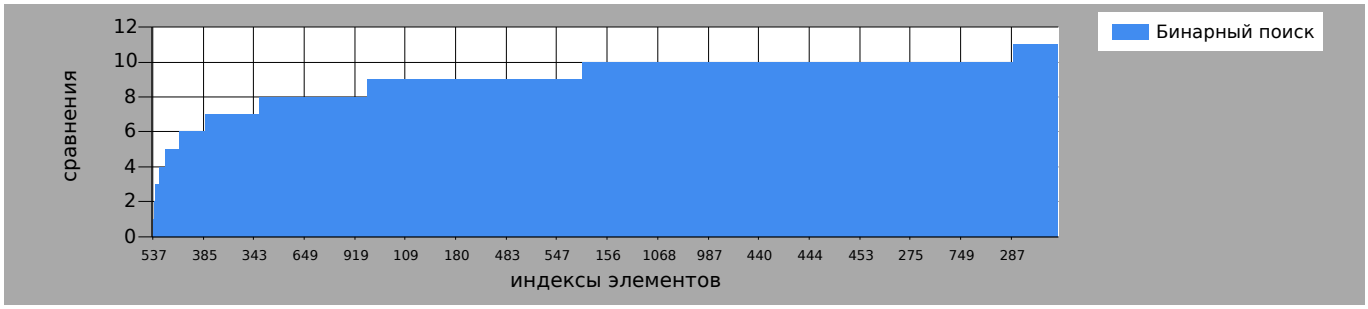

Рисунок 4.3 – Сравнения для каждого элемента в алгоритме полного перебора (отсортирован)

4.4 Вывод

На основе полученных гистограмм для двух алгоритмов поиска в массиве (полного перебора и бинарного поиска) можно сделать следующие выводы:

Полный перебор: В лучшем случае, если искомый элемент находится в начале массива, алгоритм выполняет минимальное количество операций, что соответствует трудоемкости $O(1)$. Гистограмма показала, что трудоемкость алгоритма линейно растет с увеличением индекса элемента, который необходимо найти. Это подтверждает теоретическую оценку трудоемкости $O(N)$ в худшем случае.

Бинарный поиск: В лучшем случае, если элемент находится в середине массива, бинарный поиск завершает работу, что соответствует трудоемкости $O(1)$. Гистограмма для бинарного поиска демонстрирует гораздо меньшую трудоемкость по сравнению с полным перебором (почти в 100 раз меньше сравнений для худшего случая). Количество операций увеличивается логарифмически, что соответствует трудоемкости $O(\log N)$.

Для конкретного размера массива (1076), заданного вариантом, алгоритм бинарного поиска находит искомое значение за не более, чем 11 шагов, так как размер массива лежит между следующими степенями двойки: $2^{10} < 1076 < 2^{11}$. Алгоритм полного перебора работает на первых 9 значениях (значениях с индексами от 0 до 8) лучше, чем алгоритм бинарного поиска.

ЗАКЛЮЧЕНИЕ

Исследование показало, что бинарный поиск демонстрирует значительно более высокую эффективность по сравнению с методом полного перебора. Трудоемкость бинарного поиска составляет $O(\log N)$, что делает его предпочтительным выбором для поиска элементов в отсортированных массивах, особенно при больших объемах данных.

По результатам сравнения двух алгоритмов можно сказать, что полный перебор, хотя и является простым в реализации, имеет линейную трудоемкость $O(N)$, что приводит к значительным затратам времени, особенно в худшем случае, когда элемент расположен в конце массива или отсутствует.

В ходе выполнения данной лабораторной работы были решены следующие задачи:

- проведена оценка трудоемкости разрабатываемых алгоритмов;
- реализованы алгоритмы поиска значений в словаре;
- выполнен сравнительный анализ сложности двух алгоритмов на основе замеров количества сравнений для каждого элемента массива и анализ данных лучшего и худшего случаев.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] C# Language documentation [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата обращения: 29.09.2024).

[2] Microsoft .NET Charting Namespace [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.datavisualization.charting?view=netframework-4.8.1 (дата обращения: 29.09.2024).

[3] Windows technical documentation for developers and IT pros [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/windows/ (дата обращения: 29.09.2024).

[4] Intel® Core™ i5-10300H Processor [Электронный ресурс]. URL: https://ark.intel.com/content/www/us/en/ark/products/201839/intel-core-i5-10300h-processor-8m-cache-up-to-4-50-ghz.html (дата обращения: 29.09.2024).