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 массиву

Студент Нисуев Н.Ф.

Группа ИУ7-52Б

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

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Зачастую необходимо найти определённое значение в массиве или определить, что оно отсутствует. Существуют разные алгоритмы поиска заданного значения.

Цель лабораторной работы — сравнение алгоритмов нахождения заданного значения методом линейного поиска 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 массива.

1 Аналитическая часть

В данном разделе рассмотрены алгоритмы нахождения заданного значения в массиве[1].

1.1 Описание алгоритмов

Данные алгоритмы используются для поиска заданного значения в массиве.

1.1.1 Алгоритм линейного поиска

При линейном поиске перебираются все элементы массива. Это означает, что если искомое значение находится в начале, оно будет найдено быстрее, чем если бы оно располагалось в конце [2].

1.1.2 Алгоритм нахождения с помощью бинарного поиска

При использовании алгоритма бинарного поиска перебор всех элементов не требуется. Для его работы множество должно быть отсортировано. Задаются левая и правая границы поиска, после чего выбирается центральный элемент в этом диапазоне и сравнивается с искомым значением. Если искомое значение меньше центрального элемента, правая граница смещается влево от него. Если больше — левая граница сдвигается вправо. Этот процесс продолжается до тех пор, пока искомое значение не совпадёт с центральным элементом или пока область поиска не сузится до нуля [3].

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 с помощью линейного и двоичного поисков.

2.1 Представление алгоритмов

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

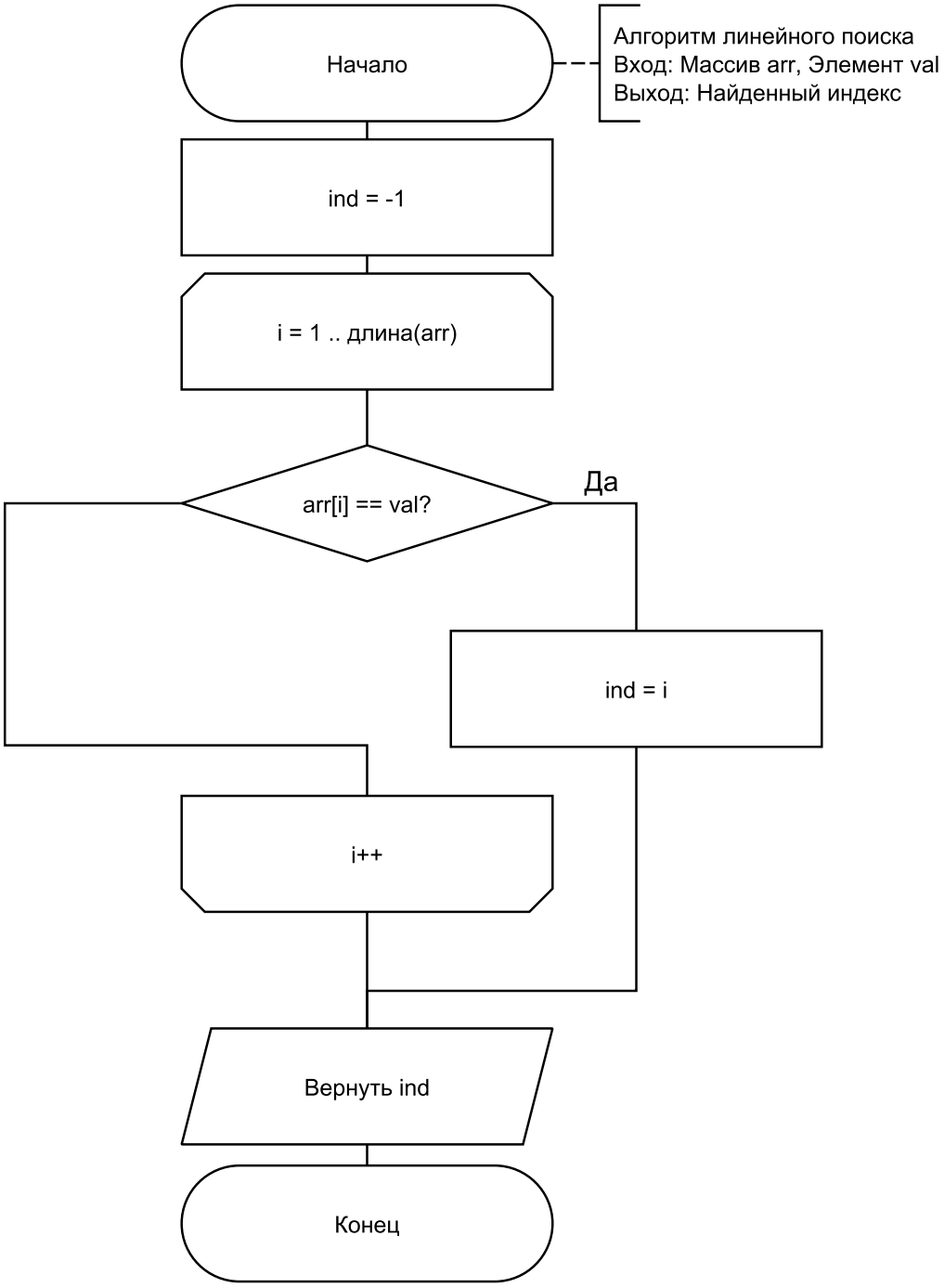

Рисунок 2.1 – Схема алгоритма поиска в массиве с помощью линейного поиска

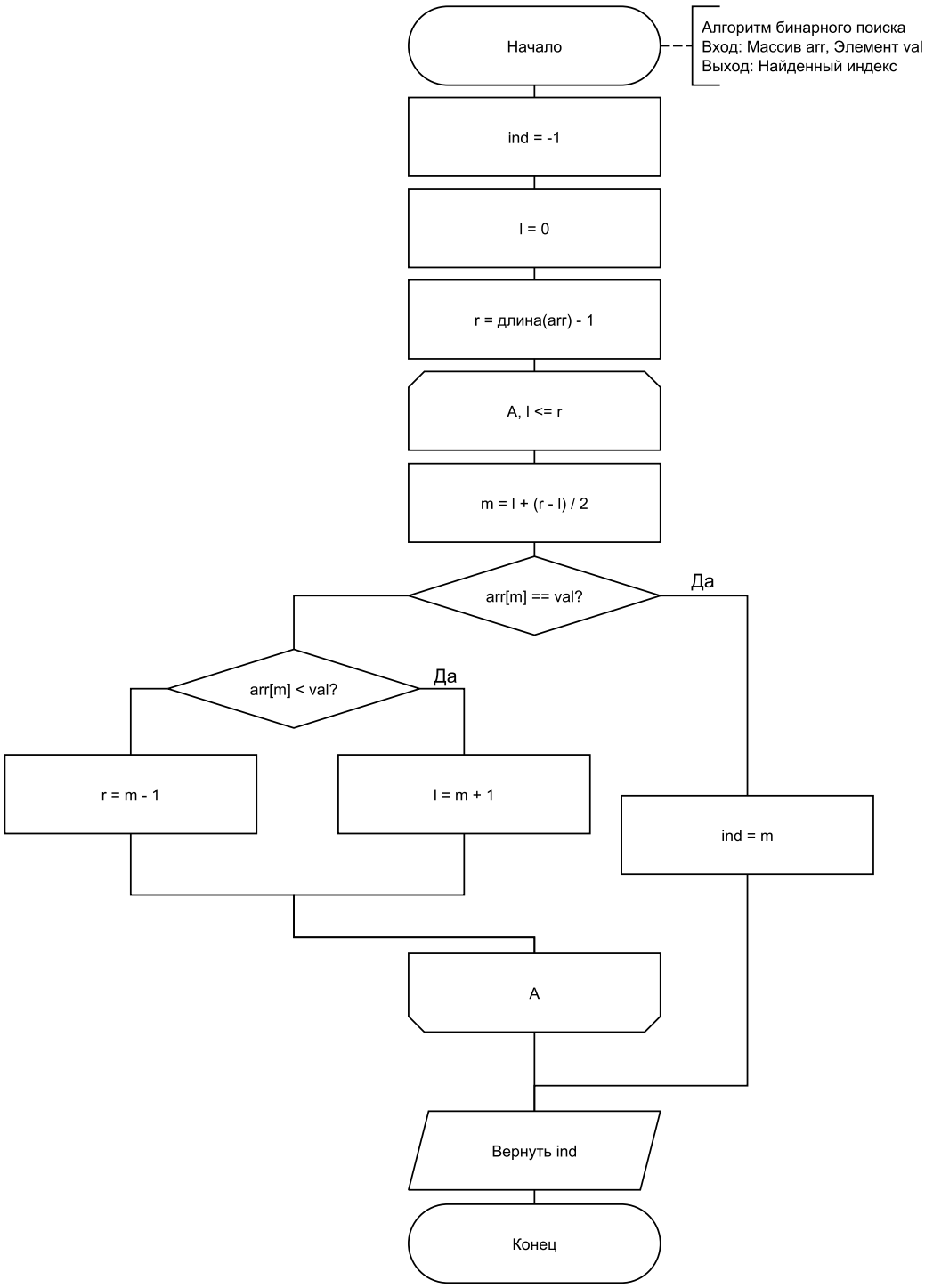

Рисунок 2.2 – Схема алгоритма поиска в массиве с помощью линейного поиска

ВЫВОД

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

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 сравнений.

3.2 Средства реализации

Для реализации был выбран язык программирования *Rust* [4]. Выбор обусловлен наличием библиотеки *plotly* [5] для построения графиков.

3.3 Реализация алгоритмов

В листингах 3.1 - 3.2 представлены реализации алгоритмов.

Листинг 3.1 – Алгоритм нахождения объектов линейным поиском

```
pub fn linear_search<T: PartialEq>(arr: &[T], target: &T) ->
   (Option<usize>, usize) {
    let mut iterations = 0;

    for (index, item) in arr.iter().enumerate() {
        iterations += 1;
        if item == target {
            return (Some(index), iterations);
        }
    }

    (None, iterations)
}
```

Листинг 3.2 – Алгоритм нахождения объектов бинарным поиском

```
pub fn binary_search<T: PartialOrd>(arr: &[T], target: &T) ->
    (Option<usize>, usize) {
    let mut low = 0;
    let mut high = arr.len() as isize - 1;
    let mut iterations = 0;

    while low <= high {
        iterations += 1;
        let mid = ((low + high) / 2) as usize;

        if &arr[mid] == target {
            return (Some(mid), iterations);
        } else if &arr[mid] < target {
            low = mid as isize + 1;
        } else {
            high = mid as isize - 1;
        }
    }

    (None, iterations)
}
```

ВЫВОД

В данном разделе были реализованы алгоритмы поиска заданного значения в массиве линейным и двоичным поисками, рассмотрены средства реализации, предусмотрены требования к программному обеспечению.

4 Исследовательская часть

4.1 Технические характеристики

Характеристики используемого оборудования:

- Операционная система — Windows 11 Home [6]
- Память — 16 Гб.
- Процессор — 12th Gen Intel(R) Core(TM) i7-12700H @ 2.30 ГГц [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 1084.

Для алгоритма линейного поиска количество возможных исходов соответствует длине массива: n исходов, если элемент присутствует в массиве, и ещё один исход, если элемента нет. В худшем случае потребуется выполнить n сравнений, что происходит, если элемент отсутствует в массиве или находится в его конце. Линейный поиск может работать быстрее на отсортированном массиве, если искомый элемент находится на индексе, меньшем $\log_2(n)$. На рисунке 4.1 представлена гистограмма, показывающая работу алгоритма линейного поиска.

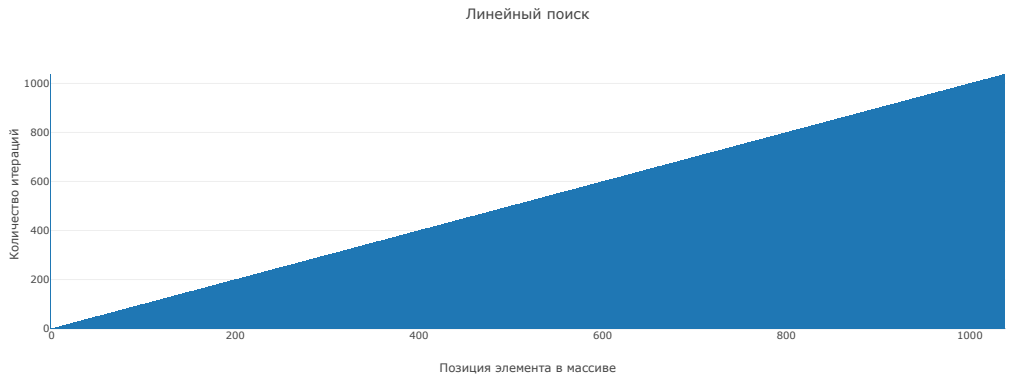

Рисунок 4.1 – Гистограмма алгоритма линейного поиска

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.

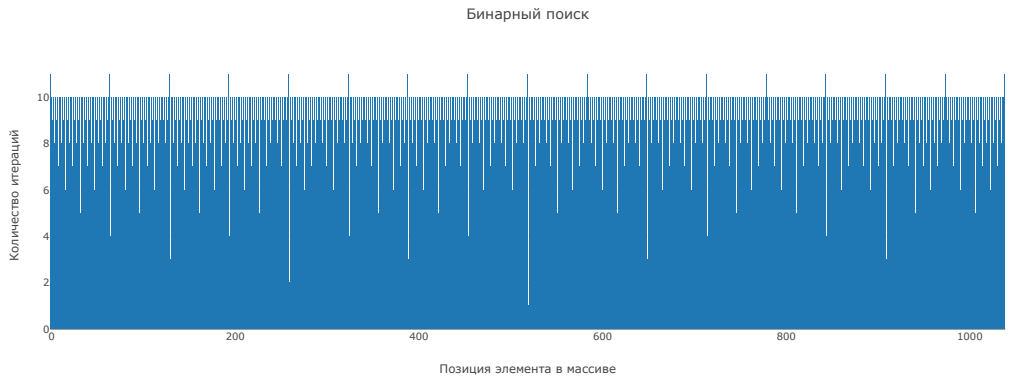

Рисунок 4.2 – Гистограмма алгоритма с бинарным поиском

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

4.3 Вывод

При полном переборе количество сравнений для поиска значения в массиве увеличивается по мере роста индекса искомого элемента. В отличие от этого, в бинарном поиске число сравнений практически не зависит от положения элемента в начале или конце массива. Однако элемент будет найден быстрее, если он находится ближе к середине или в её четвертях. Линейный поиск может быть быстрее бинарного на отсортированном массиве, если искомый элемент расположен на индексе, меньшем $\log_2(n)$, где n — размер массива. Например, при размере массива 1084, линейный поиск будет эффективнее двоичного для индексов от 1 до 10.

Тем не менее, наиболее эффективным считается бинарный поиск, так как в худшем случае количество сравнений не превышает $log_2(n)$, тогда как при полном переборе оно может достигать n.

ЗАКЛЮЧЕНИЕ

Экспериментально были подтверждены различия между алгоритмами поиска заданного значения с использованием линейного и бинарного поисков, что было продемонстрировано с помощью разработанного программного обеспечения.

На основании исследований можно сделать вывод, что алгоритм линейного поиска уступает бинарному поиску по количеству сравнений, необходимых для нахождения искомого 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. Поиск [Электронный ресурс]. — Режим доступа: https://kvodo.ru/algorithms/search_algorithms (дата обращения: 05.10.2024).
2. Алгоритм линейного поиска [Электронный ресурс]. — Режим доступа: https://kvodo.ru/lineynyiy-poisk.html (дата обращения: 05.10.2024).
3. Двоичный (бинарный) поиск [Электронный ресурс]. — Режим доступа: https://kvodo.ru/dvoichnyiy-poisk-2.html (дата обращения: 05.10.2024).
4. The Rust Programming Language [Электронный ресурс]. — Режим доступа: https://doc.rust-lang.org/book/ (дата обращения: 05.10.2024).
5. Бтблиотека Plotly [Электронный ресурс]. — Режим доступа: https://docs.rs/plotly/latest/plotly/ (дата обращения: 05.10.2024).
6. Windows technical documentation for developers and IT pros [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/windows/ (дата обращения: 05.10.2024).
7. Intel® Core™ i7-12700H Processor [Электронный ресурс]. — Режим доступа: https://ark.intel.com/content/www/us/en/ark/products/132228/intel-core-i7-12700h-processor-24m-cache-up-to-4-70-ghz.html (дата обращения: 05.10.2024).