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»

Лабораторная работа № 3
по дисциплине «Анализ алгоритмов»

Тема Поиск в массиве

Студент Бугаков И. С.

Группа ИУ7-54Б

Преподаватели Строганов Ю. В., Волкова Л. Л.

Москва, 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель данной работы — анализ трудоемкости поиска в массиве относительно количества производимых операций сравнения для двух реализаций: линейного и бинарного поиска.

Задачи лабораторной работы:

1) описать алгоритмы линейного и бинарного поисков;
2) реализовать указанные алгоритмы;
3) провести тестирование реализаций алгоритмов;
4) провести анализ алгоритмов поиска относительно числа производимых ими операций сравнения.

1 Аналитическая часть

Ниже представлены два основных и наиболее распространенных [3] метода поиска элемента в массиве:

— линейный поиск;

— бинарный поиск.

1.1 Линейный поиск

1.1.1 Описание алгоритма

Линейный поиск — метод поиска, при котором последовательно просматриваются и сравниваются с искомым все элементы массива.

Соответственно для линейного поиска возможен $n + 1$ исход, где n — количество элементов в массиве. При этом лучший случай наступает, когда элемент обнаруживается на первой же позиции, т. е. производится единственное сравнение. Худших случаев два: элемент найден на последней позиции или не найден вообще. При этом оба случая обойдутся в n сравнений.

Пусть k_1 — число операций производимых линейным поиском до прохода по всем элементам, например, объявление и определение начального значения индекса. Пусть также k_2 — число операций производимых на каждой итерации просмотра очередного элемента, например, операции сравнения и инкремента индекса. Тогда для линейного поиска:

$$f_{best}(n) = k_1 + k_2 \tag{1.1}$$

$$f_{worst}(n) = k_1 + n \cdot k_2 \tag{1.2}$$

$$f_{mean}(n) = k_1 + \sum_{i \in \Omega} p_i \cdot \delta_i = k_1 + k_2 \cdot (1 + \frac{n}{2} - \frac{1}{n+1}) \tag{1.3}$$

где Ω — множество всех возможных случаев, p_i — вероятность i- случая, для $\forall i \; p_i = \frac{1}{n}$, т. к. все случаи равновозможны, δ_i — количество операций, которые необходимо выполнить для нахождения элемента в i-ом случае.

Формулы 1.1, 1.2 и 1.3 отражают лучший, худший и средний по числу операций случаи соответственно. Из худшего случая следует, что асимптотика такого поиска $O(n)$.

1.2 Бинарный поиск

1.2.1 Описание алгоритма

Алгоритм бинарного поиска применим только к упорядоченным массивам, в связи с чем, возникает необходимость предварительной сортировки, если таковая возможна. Основная идея алгоритма — возможность отбросить половину отрезка поиска, после сравнения его середины с искомым элементом [4]. Так, если массив упорядочен по неубыванию, и средний элемент отрезка элемента больше искомого, поиск необходимо продолжать в левой половине с меньшими среднего элементами. Иначе в правой с большими соответственно.

В этом случае последовательное уменьшение отрезка поиска можно представить в виде бинарного дерева, высота которого будет не более $\log_2 n$. Таким образом, бинарный поиск не проверяет более

$$\lfloor \log_2 n \rfloor + 1 \tag{1.4}$$

элементов [4]. Откуда асимптотика $O(\log n)$.

Вывод

Рассмотрены алгоритмы линейного и бинарного поисков элемента в массиве, приведены их асимптотики и оценки количества операций сравнения.

2 Конструкторская часть

Ниже приведены схемы алгоритмов, упомянутых в аналитической части.

2.1 Линейный поиск

На рисунке 2.1 приведена схема алгоритма линейного поиска.

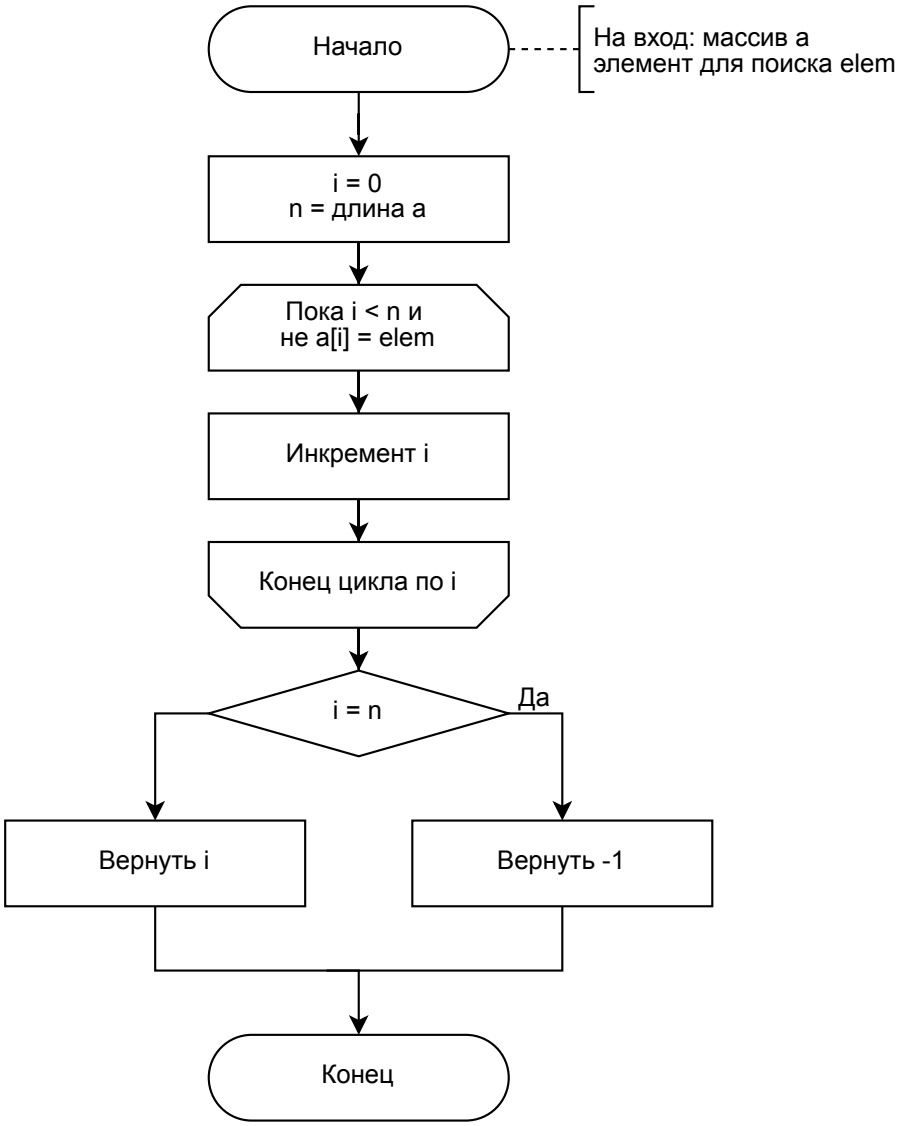

Рисунок 2.1 — Схема алгоритма линейного поиска

2.2 Бинарный поиск

На рисунке 2.2 приведена схема алгоритма бинарного поиска.

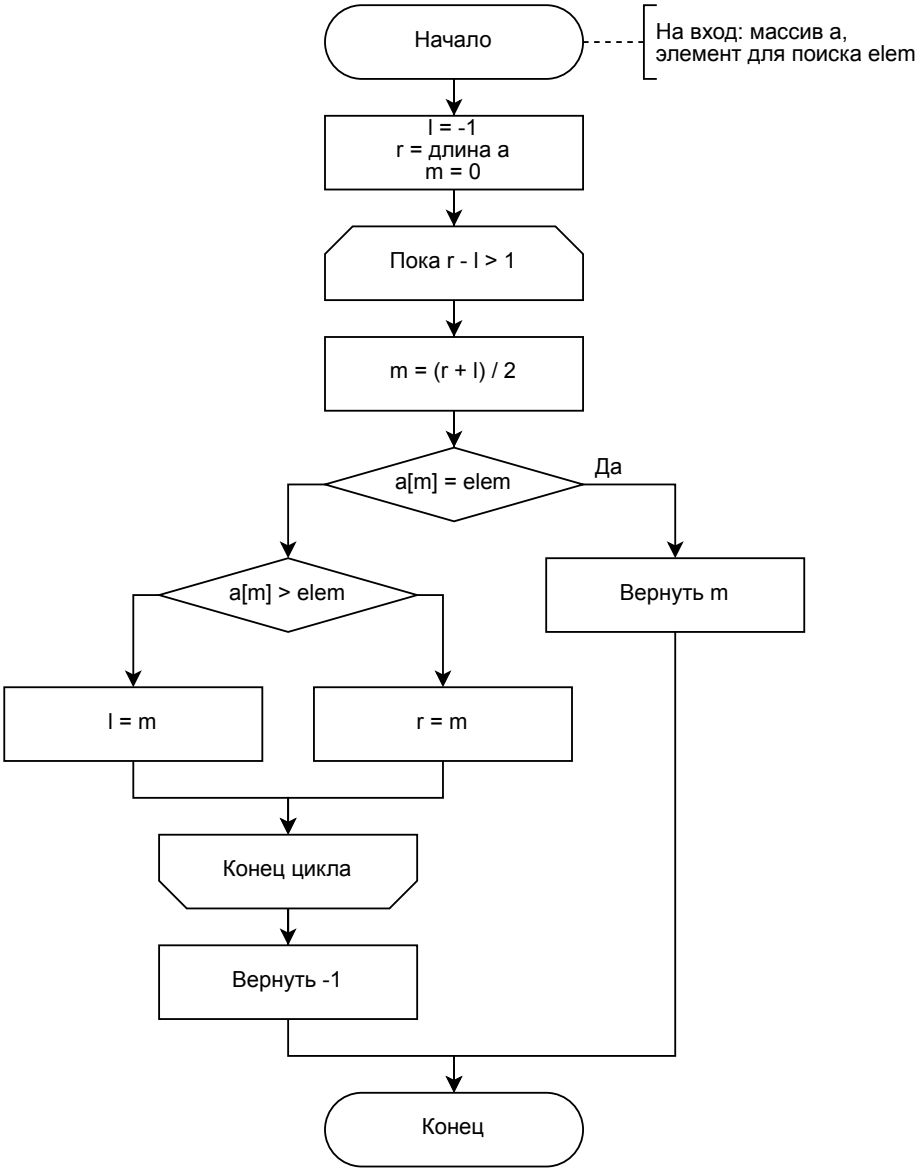

Рисунок 2.2 — Схема алгоритма бинарного поиска

2.3 Требования к реализации алгоритмов

Реализации алгоритмов должны удовлетворять следующим общим требованиям:

— входные данные: массив целых чисел, элемент для поиска, представленный целым числом;

— выходные данные: индекс вхождения элемента или -1, если элемент не найден.

К алгоритму бинарного поиска предъявляется дополнительное требование: входной массив должен быть упорядочен по неубыванию.

Вывод

На основе аналитической части построены схемы алгоритмов для реализации.

3 Технологическая часть

3.1 Выбор языка программирования

Для реализации указанных алгоритмов был выбран язык C++, т. к. данный язык предоставляет возможность работать с динамической памятью с помощью контейнерных классов.

Для визуализации данных использовался язык Python, т. к. данный язык обладает большим выбором библиотек для выполнения этой задачи. В лабораторной работе использовалась библиотека matplotlib [2].

В качестве IDE был выбран Clion, т. к. обладает функциями подсветки синтаксиса и автодополнения для обоих указанных языков. Также в данной IDE реализована поддержка средства автоматизированной сборки проектов cmake.

3.2 Реализации алгоритмов

В листингах ниже представлены реализации указанных алгоритмов:

— в листинге 3.1 — реализация линейного поиска;

— в листинге 3.2 — реализация бинарного поиска.

Данные реализации дополнительно возвращают число сравнений, необходимое для дальнейшего анализа.

Листинг 3.1 — Функция линейного поиска

```
pair<int, int> linear_search(const vector<int> &a, int elem_to_find)
   {
  int i, cnt = 1;
  for (i = 0; i < a.size() && a[i] != elem_to_find; ++i, cnt++) {}
  return {(i == a.size() ? -1 : i), (i == a.size() ? cnt - 1 : cnt)};
}
```

Листинг 3.2 — Функция бинарного поиска

```
pair<int, int> binary_search(const vector<int> &a, int elem_to_find)
   {
  int l = -1, r = a.size(), m, cnt = 0;
  while (r - l > 1) {
    m = (r + l) / 2;
    if (a[m] == elem_to_find) {
      ++cnt;
      return {m, cnt};
    } else if (a[m] > elem_to_find) {
      cnt += 2;
      r = m;
    } else {
      cnt += 2;
      l = m;
    }
  }
  return {-1, cnt};
}
```

3.3 Тестирование

Для тестирования алгоритмов, тестовые случаи были разбиты на классы эквивалентности. В таблице 3.1 приведены тесты для линейного поиска. В таблице 3.2 для бинарного поиска соответственно.

Таблица 3.1 — Тестовые случаи для алгоритма линейного поиска

Классы эквивалентности	Входные данные	Ожидаемые выходные данные		Полученные выходные данные	
		Индекс	Число сравнений	Индекс	Число сравнений
Искомый элемент первый	1, 2, 3, 4, 5 1	0	1	0	1
Искомый элемент последний	1, 2, 3, 4, 5 5	4	5	4	5
Искомый элемент центральный	1, 2, 3, 4, 5 3	2	3	2	3
Искомый элемент отсутствует	1, 2, 3, 4, 5 -1	-1	5	-1	5
Массив пуст	«» 5	-1	0	-1	0

Таблица 3.2 — Тестовые случаи для алгоритма бинарного поиска

Классы эквивалентности	Входные данные	Ожидаемые выходные данные		Полученные выходные данные	
		Индекс	Число сравнений	Индекс	Число сравнений
Искомый элемент первый	1, 2, 3, 4, 5 1	0	3	0	3
Искомый элемент последний	1, 2, 3, 4, 5 5	4	5	4	5
Искомый элемент центральный	1, 2, 3, 4, 5 3	2	1	2	1
Искомый элемент отсутствует	1, 2, 3, 4, 5 -1	-1	4	-1	4
Массив пуст	«» 5	-1	0	-1	0

Для проведения тестирования использовалась библиотека Google Tests [1].

Вывод

Были реализованы алгоритмы поиска элемента в массиве и проведено их тестирование. Все тесты были успешно пройдены.

4 Исследовательская часть

4.1 Характеристики устройства

Исследования проводились на машине со следующими характеристиками:

— процессор Intel(R) Core(TM) i5-10210U, тактовая частота 1.60 ГГц;

— оперативная память: 16 ГБ;

— операционная система: Ubuntu 22.04.4 LTS.

4.2 Замер числа операций сравнений

Длинна массива по варианту составляет 1041 элемент. Соответственно, было произведено 1042 запуска: для каждого элемента и один для элемента, которого в массиве нет.

Далее приведены графики, иллюстрирующие зависимость количества сравнений от индекса вхождения элемента (для элемента, которого нет в массиве, индекс принимается равным -1). На рисунке 4.1 представлен график для линейного поиска. На рисунке 4.2 для бинарного соответственно.

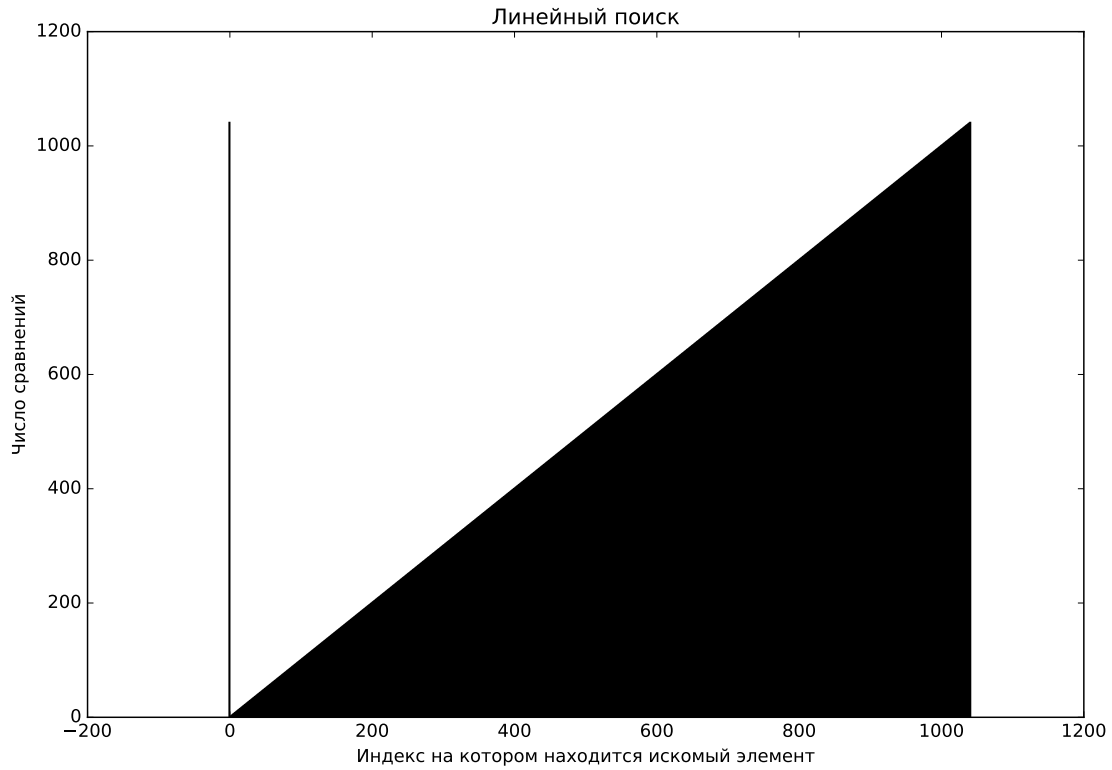

Рисунок 4.1 — График зависимости числа сравнений от индекса элемента для линейного поиска

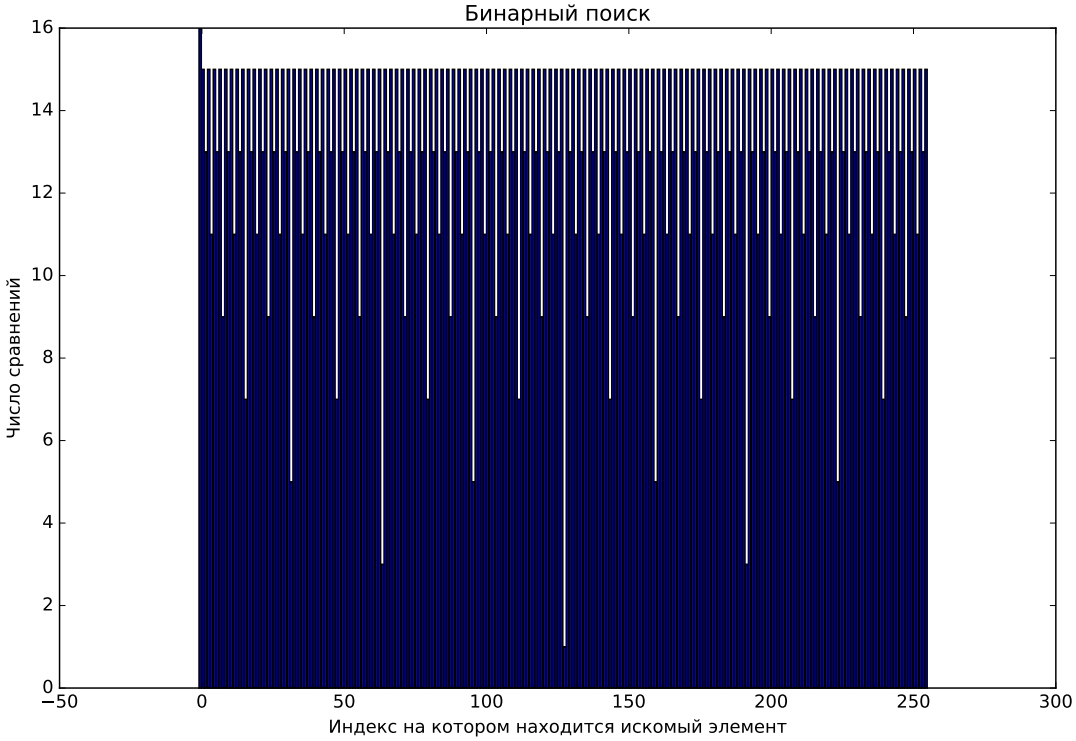

Рисунок 4.2 — График зависимости числа сравнений от индекса элемента для бинарного поиска

Среднее число операций сравнения полученное при замерах для бинарного поиска равно 17. Для линейного поиска это число составляет 521.

4.3 Вывод

Алгоритм бинарного поиска превосходит линейный поиск по эффективности относительно числа производимых сравнений. Так для нахождения максимального элемента бинарный поиск производит 21 операцию сравнения, тогда как линейный поиск осуществляет для этого 1041 операцию. При это для нахождения минимального элемента бинарный поиск производит 19 операций,а линейный одну. В среднем же бинарный поиск выполняет меньше операций сравнения, чем линейный алгоритм.

Если в формулу 1.3 подставить $k_1 = 0$ и $k_2 = 1$, т. е. одна операция сравнения на каждой итерации цикла, теоретическое среднее число операций сравнения получается равным $1 + \frac{1041}{2} + \frac{1}{1042} \approx 521$, что совпадает со средним числом сравнений полученным на практике.

Максимальное число операций сравнения для бинарного поиска составляет 21. При этом количество просматриваемых элементов равно 11, что также соответствует теоретической оценке при подстановке $n = 1041$ в формулу 1.4: $\lfloor \log_2 1041 \rfloor + 1 = 10 + 1 = 11$.

Однако, бинарный поиск требует упорядоченности массива, что сужает допустимый

спектр входных данных, тогда как линейный поиск таких требований не накладывает.

ЗАКЛЮЧЕНИЕ

Целью данной работы являлся анализ эффективности бинарного и линейного поиска в массиве относительно числа операций сравнения.

В ходе лабораторной работы были выполнены следующие задачи:

1) описаны алгоритмы линейного и бинарного поисков;
2) реализованы указанные алгоритмы;
3) проведено тестирование реализаций алгоритмов;
4) проведен анализ алгоритмов поиска относительно числа производимых ими операций сравнения.

По итогам анализа было выявлено, что бинарный поиск в среднем производит меньше операций сравнения, чем линейный. При этом теоретические оценки среднего числа операций сравнения для линейного поиска и максимального числа просматриваемых бинарным поиском элементов, совпали с практически полученными результатами.

Поставленная цель исследования алгоритмов поиска в массиве была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Googletest user's guide. Режим доступа: `https://google.github.io/googletest/`. Дата обращения 8.10.2024.

2. matplotlib. Режим доступа: `https://matplotlib.org/`. Дата обращения 24.10.2024.

3. Конспект лекций по дисциплине "Анализ алгоритмов". Режим доступа: `http://oii.iu7.bmstu.ru:3000/projects/aa2024_25_iu7_lab_aa_03/files`. Дата обращения 24.10.2024.

4. Седжвик Роберт. *Фундаментальные алгоритмы на С. Анализ/Структуры данных/Сортировка/Поиск*. ДиаСофтЮП, СПБ, 2003. 1136 с., ISBN 5-93772-081-4.