



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 № 7
по курсу «Анализ алгоритмов»
на тему: «Алгоритмы поиска»

Студент ИУ7-56Б
(Группа)

(Подпись, дата)

М. Ю. Вольняга
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Л. Л. Волкова
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Ю. В. Строганов
(И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Аналитическая часть	4
1.1 Алгоритм бинарного поиска	4
2 Конструкторская часть	5
2.1 Требования к программному обеспечению	5
2.2 Разработка алгоритмов	5
3 Технологическая часть	8
3.1 Средства реализации	8
3.2 Сведения о модулях программы	8
3.3 Реализация алгоритмов	8
4 Исследовательская часть	11
4.1 Демонстрация работы программы	11
4.2 Количество сравнений	12
4.3 Вывод	14
ЗАКЛЮЧЕНИЕ	16
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	17

ВВЕДЕНИЕ

Целью данной лабораторной работы является исследование лучших и худших случаев работы алгоритмов поиска.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) описать используемые алгоритмы поиска;
- 2) описать худший и лучший случаи работы рассмотренных алгоритмов;
- 3) определить средства программной реализации;
- 4) реализовать данные алгоритмы поиска;
- 5) проанализировать алгоритмы по количеству сравнений.

1 Аналитическая часть

В данном разделе приведена информация, касающаяся алгоритма бинарного поиска.

1.1 Алгоритм бинарного поиска

Алгоритм бинарного поиска является алгоритмом поиска значения в отсортированном массиве данных.

Алгоритм бинарного поиска начинает сравнение целевого значения с элементом в середине массива. Если они не равны, половина массива, в которой целевое значение не может находиться, удаляется, и поиск продолжается в оставшейся половине. Затем цикл повторяется: снова берется элемент в середине оставшейся половины, сравнивается с целевым значением, и процесс продолжается до тех пор, пока не будет найдено целевое значение. Если поиск завершается с пустой оставшейся половиной, значит, целевое значение отсутствует в массиве.

В лучшем случае искомое значение находится в середине массива — тогда алгоритм отрабатывает за 1 итерацию [1].

В худшем случае искомый элемент отсутствует в массиве или является последним — тогда количество итераций равно $\text{ceil}(\log_2(n) + 1)$ [2].

2 Конструкторская часть

В данном разделе будут представлены псевдокоды алгоритмов двоичного поиска с одним и двумя сравнения с медианным элементом.

2.1 Требования к программному обеспечению

К программному обеспечению предъявлен ряд требований:

- 1) наличие интерфейса для выбора действий;
- 2) возможность ввода массива;
- 3) возможность выполнения операции поиска по массиву.

2.2 Разработка алгоритмов

На рисунке 1 представлен псевдокод алгоритма двоичного поиска с двумя сравнениями с медианным элементом.

В качестве инструмента для создания псевдокода использован пакет `algorithmx`.

Листинг 1 Псевдокод алгоритма бинарного поиска с двумя сравнениями с медианным элементом

Вход: ссылка на массив arr размером n и искомый элемент x

Выход: индекс найденного элемента ind

```
1: function BINSEARCHTWOCOMP( $arr, n, x$ )
2:    $l \leftarrow 0$ 
3:    $h \leftarrow n - 1$ 
4:   while  $l \leq h$  do
5:      $m \leftarrow l + (h - l)/2$ 
6:     if  $arr[m] = x$  then return  $m$ 
7:     else if  $arr[m] < x$  then
8:        $l \leftarrow m + 1$ 
9:     else
10:       $h \leftarrow m - 1$ 
11:    end if
12:  end while
13: return  $-1$ 
14: end function
```

На рисунке 2 представлен псевдокод алгоритма двоичного поиска с двумя сравнениями с медианным элементом.

Листинг 2 Псевдокод алгоритма бинарного поиска с одним сравнением с медианным элементом

Вход: ссылка на массив arr размером n и искомый элемент x

Выход: индекс найденного элемента ind

```
1: function BINSEARCHONECOMP( $arr, n, x$ )
2:    $l \leftarrow 0$ 
3:    $h \leftarrow n - 1$ 
4:   while  $l \neq h$  do
5:      $m \leftarrow \text{ceil}(l + h)/2$ 
6:     if  $arr[m] > x$  then
7:        $h \leftarrow m - 1$ 
8:     else
9:        $l \leftarrow m$ 
10:    end if
11:  end while
12:  if  $arr[l] = x$  then return  $l$ 
13:  end if
14:  return  $-1$ 
15: end function
```

Вывод

В данном разделе были перечислены требования к программному обеспечению и предоставлены псевдокоды реализуемых алгоритмов.

3 Технологическая часть

В данном разделе описаны средства реализации программного обеспечения, а также представлены листинги.

3.1 Средства реализации

В качестве языка программирования, используемого при написании данной лабораторной работы, был выбран C++ [3], так как в нем имеется контейнер `std::vector`, представляющий собой динамический массив данных произвольного типа.

3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

- `main.cpp` — файл, содержащий точку входа в программу;
- `algorithms.cpp` — файл, содержащий реализации алгоритмов бинарного поиска.

3.3 Реализация алгоритмов

На листинге 3.1 представлена реализация алгоритма двоичного поиска с двумя сравнениями с медианным элементом.

Листинг 3.1 – Реализация алгоритма двоичного поиска с двумя сравнениями с медианным элементом

```
1 | int Algs::bin_search_two_comp(const std::vector<int> &arr, int x)
2 | {
3 |     int low = 0;
4 |     int high = arr.size() - 1;
5 |
6 |     while (low <= high)
7 |     {
8 |         int mid = low + (high - low) / 2;
9 |         if (arr[mid] == x)
10 |         {
11 |             return mid;
12 |         }
13 |         else if (arr[mid] < x)
```



```

14         {
15             low = mid + 1;
16         }
17         else
18         {
19             high = mid - 1;
20         }
21     }
22
23     return -1;
24 }

```

На листинге 3.1 представлена реализация алгоритма двоичного поиска с одним сравнением с медианным элементом.

Листинг 3.2 – Реализация алгоритма двоичного поиска с одним сравнением с медианным элементом

```

1  int Algs::bin_search_one_comp(const std::vector<int> &arr, int
   x)
2  {
3      int low = 0;
4      int high = arr.size() - 1;
5
6      while (low != high)
7      {
8          int mid = ceil(double(low + high) / 2.0);
9          if (arr[mid] > x)
10         {
11             high = mid - 1;
12         }
13         else
14         {
15             low = mid;
16         }
17     }
18     if (arr[low] == x)
19     {
20         return low;
21     }
22
23     return -1;
24 }

```

Вывод

В данном разделе были рассмотрены средства реализации, а также представлен листинг реализаций алгоритмов бинарного поиска с одним и двумя сравнениями с медианным элементом.

4 Исследовательская часть

В данном разделе приведены результаты подсчета количества сравнений при поиске элементов в худшем и лучшем случаях.

4.1 Демонстрация работы программы

На рисунке 4.1 продемонстрирована работа программы для случая, когда пользователь выбрал пункт 1 «Выполнить поиск элемента алгоритмом бинарного поиска», ввел массив $[10, 20, 30, 40, 50]$ и запросил поиск элемента 50. Далее пользователь выбрал тот же пункт, ввел массив $[1, 2, 3]$ и запросил поиск элемента -1 , которого нет в массиве.

Меню

1. Выполнить поиск элемента алгоритмом бинарного поиска
 - а) с двумя сравнениями с медианным элементом;
 - б) с одним сравнением с медианным элементом.
2. Подсчитать количество сравнений при выполнении поиска элементов.
0. Выход.

Выберите опцию (0–2): 1

Введите к-во элементов массива: 5

Введите элементы массива (в порядке возрастания): 10 20 30 40 50

Введите искомый элемент: 50

Индекс найденного элемента (2 сравнения): 4

Индекс найденного элемента (1 сравнение): 4

Меню

1. Выполнить поиск элемента алгоритмом бинарного поиска
 - а) с двумя сравнениями с медианным элементом;
 - б) с одним сравнением с медианным элементом.
2. Подсчитать количество сравнений при выполнении поиска элементов.
0. Выход.

Выберите опцию (0–2): 1

Введите к-во элементов массива: 3

Введите элементы массива (в порядке возрастания): 1 2 3

Введите искомый элемент: -1

Элемент не найден.

Меню

1. Выполнить поиск элемента алгоритмом бинарного поиска
 - а) с двумя сравнениями с медианным элементом;
 - б) с одним сравнением с медианным элементом.
2. Подсчитать количество сравнений при выполнении поиска элементов.
0. Выход.

Выберите опцию (0–2): 0

Рисунок 4.1 – Демонстрация работы программы

4.2 Количество сравнений

Исследование реализуемых алгоритмов по количеству выполняемых сравнений производилось 2 раза:

- 1) при варьировании числа элементов в массиве 512, 1024, 2048, 4096, 8192 в лучшем и худших случаях;

- 2) при варьировании числа элементов в массиве 513, 1025, 2049, 4097, 8193 в лучшем и худших случаях;

На рисунке 4.2 изображены результаты исследования для лучшего случая.

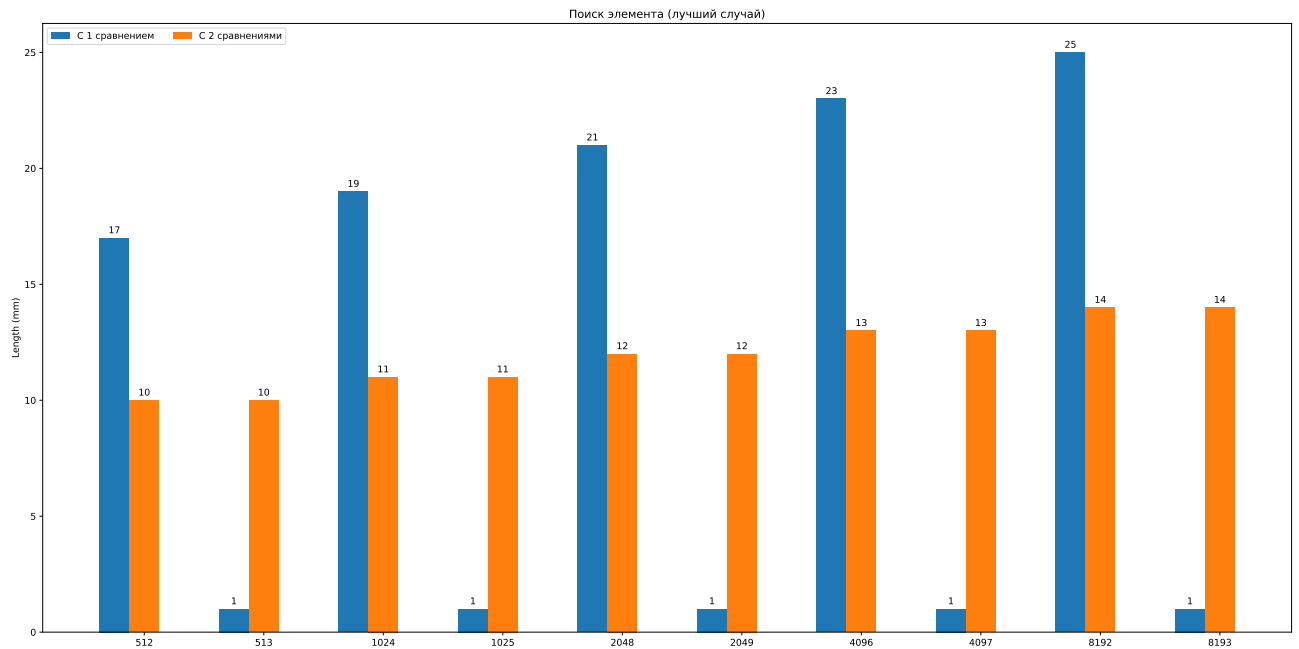


Рисунок 4.2 – Сравнение количества сравнений при работе алгоритмов для лучшего случая

На рисунке 4.3 изображены результаты исследования для худшего случая, когда искомого элемента нет.

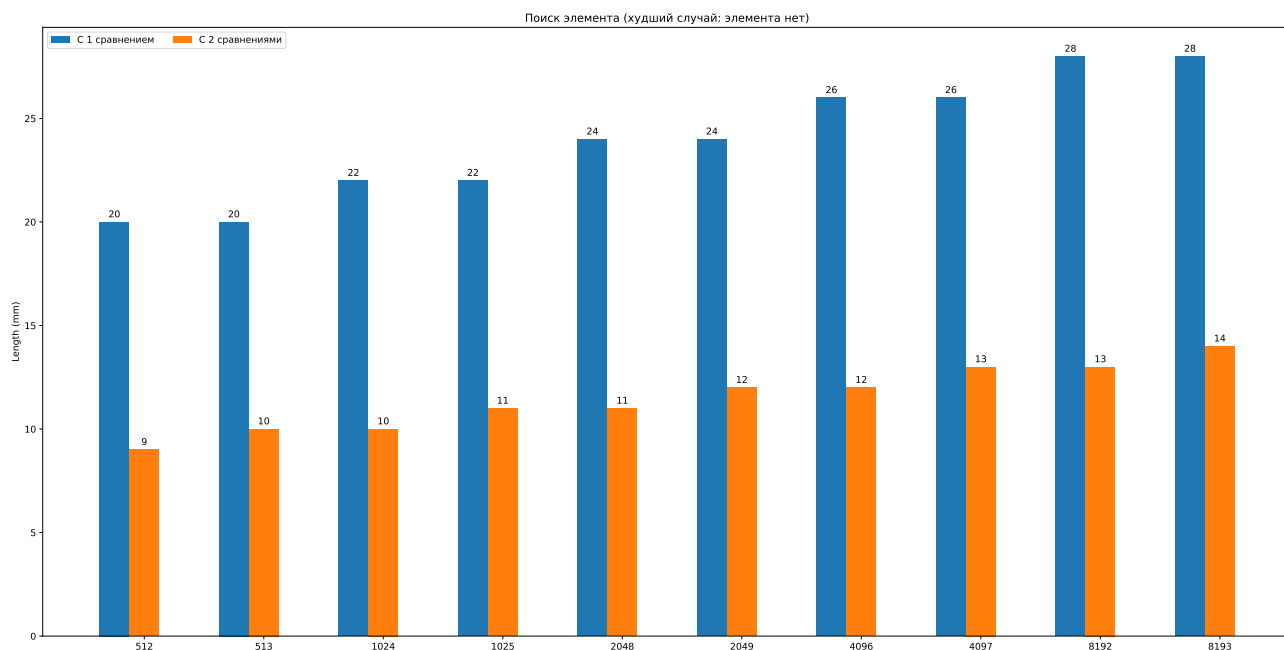


Рисунок 4.3 – Сравнение количества сравнений при работе алгоритмов для худшего случая, когда искомого элемента нет

На рисунке 4.4 изображены результаты исследования для худшего случая, когда искомый элемент последний.

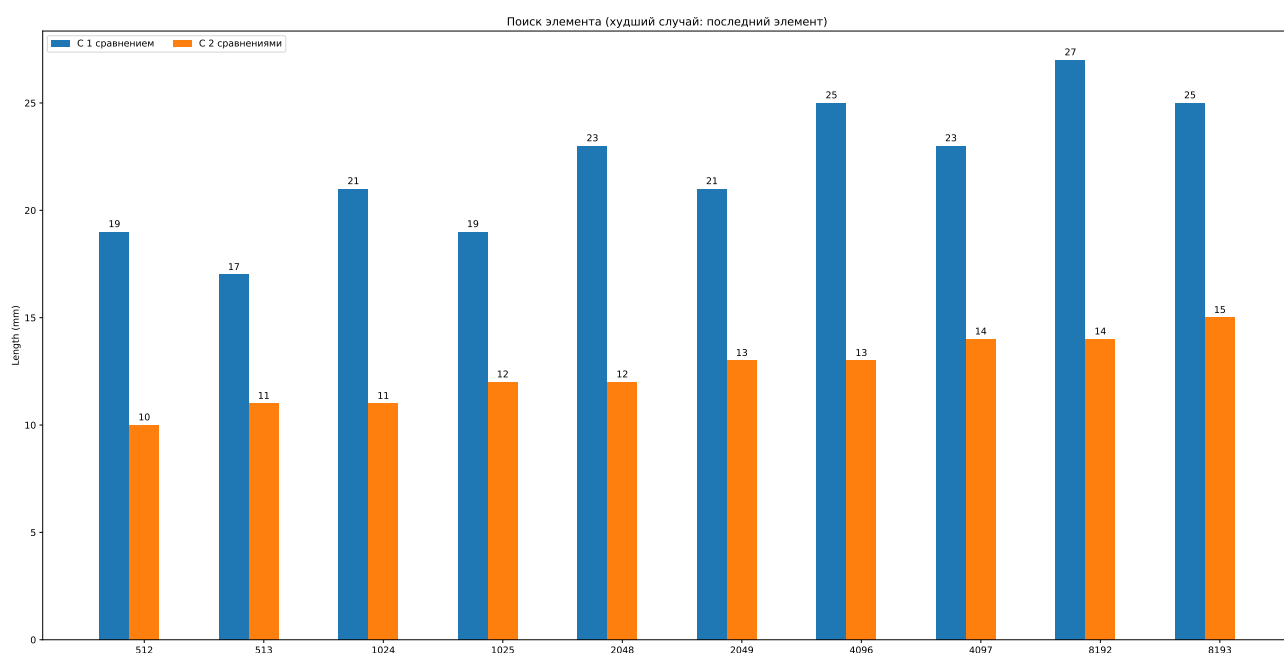


Рисунок 4.4 – Сравнение количества сравнений при работе алгоритмов для худшего случая, когда искомый элемент последний

4.3 Вывод

При нечетных размерностях алгоритм поиска с одним сравнением выполняется за одну операцию сравнения. Однако на четных размерностях

количество выполняемых сравнений больше, чем у алгоритма, выполняющего поиск за два сравнения с медианным элементом.

Для реализации бинарного поиска, использующей два сравнения с медианным элементом, самым худшим случаем работы является поиск последнего элемента массива, а для реализации с одним сравнением — при поиске элемента, которого нет в массиве.

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы по исследованию алгоритмов сортировок решены следующие задачи:

- 1) описаны используемые алгоритмы поиска;
- 2) описаны худший и лучший случаи работы рассмотренных алгоритмов;
- 3) определены средства программной реализации;
- 4) реализованы алгоритмы двоичного поиска с одним и двумя сравнениями с медианным элементом;
- 5) проанализированы алгоритмы по количеству сравнений и сделаны следующие выводы:
 - при нечетных размерностях алгоритм поиска с одним сравнением выполняется за одну операцию сравнения; однако на четных размерностях количество выполняемых сравнений больше, чем у алгоритма, выполняющего поиск за два сравнения с медианным элементом;
 - для реализации бинарного поиска, использующей два сравнения с медианным элементом, самым худшим случаем работы является поиск последнего элемента массива, а для реализации с одним сравнением — при поиске элемента, которого нет в массиве.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Chang S.-K.* Data Structures and Algorithms. Т. 13. — Singapore : World Scientific, 2003. — ISBN 978-981-238-348-8.
2. *Knuth D.* Sorting and Searching. The Art of Computer Programming. Vol. 3. — 2nd. — Reading, MA : Addison-Wesley Professional, 1998. — ISBN 978-0-201-89685-5.
3. C++ language. — [Электронный ресурс]. — Режим доступа: <https://en.cppreference.com/w/cpp/language> (дата обращения: 21.12.2023).