Минобрнауки РФ

ФГБОУ ВО Тверской Государственный Технический Университет

Кафедра  “Программное Обеспечение”

Лабораторная работа N4

По дисциплине “Структуры и алгоритмы обработки данных”

Выполнил: студент группы Б.ПИН.РИС-22.06

Иванов Алексей Михайлович

Проверил: Мальков Александр Анатольевич

г. Тверь, 2023

Оглавление

[**Часть 1. Постановка задачи.** 3](#_Toc145406338)

[**Часть 2. Алгоритм решения.** 3](#_Toc145406339)

[**Часть 3. Оценка временной эффективности в терминах О-функций.** 4](#_Toc145406340)

[**Часть 4. Программный код.** 5](#_Toc145406341)

[**Часть 5. Тестирование и результаты работы.** 9](#_Toc145406342)

# **Часть 1. Постановка задачи.**

Алгоритмы поиска

- 4 алгоритма+рекурсивный вариант.

# **Часть 2. Алгоритм решения.**

# В данной лабораторной работе будут рассмотрены следующие алгоритмы поиска элемента в массиве:

1. Линейный поиск
2. Бинарный поиск
3. Рекурсивный вариант бинарного поиска
4. Интерполяционный поиск
5. Эксполяционный поиск

Алгоритм линейного поиска – самый простой в реализации и самый универсальный. Он сводится к последовательному сравнению каждого элемента массива с образцом. Он также является самым универсальным, так как не требует того, чтобы массив был упорядочен.

Алгоритм бинарного поиска требует упорядоченности массива, зато его алгоритмическая сложность значительно ниже. Так как массив упорядочен, нет необходимости сравнивать все элементы с образцом: вместо этого можно сравнить образец с центральным элементом, если он окажется больше, то необходимость просматривать первую половину массива отпадает. Тогда можно принять за начало нового отрезка поиска середину массива, и далее повторять аналогичные действия, делать это можно либо с помощью цикла, либо рекурсивно. В случае, если образец меньше центрального элемента, конец исследуемого отрезка ставится на середину просматриваемого, и далее действия аналогичны.

Интерполяционный поиск - это алгоритм, который использует специальную формулу для приблизительного нахождения элемента. В моей работе он основан на принципе линейной интерполяции, который используется для приближенного нахождения значения функции в промежутке между двумя известными точками. Это нужно для приблизительного нахождения позиции элемента.

Он заключается в следующем:

Сначала определяется диапазон, где находится искомый элемент, затем сравнивается значение искомого элемента с элементом массива с найденным индексом. Если они равны, то образец найден. Если значение искомого элемента меньше, чем значение элемента с найденным индексом, то мы уменьшаем диапазон поиска, устанавливая верхнюю границу равной найденному индексу минус один. Если значение искомого элемента больше, то величивается диапазон поиска, устанавливая нижнюю границу равным найденному индексу плюс один.

Эксполяционный поиск также основан на бинарном. Только для него еще необходимо определить k такое, чтобы элементы массива на отрезке [2^k-1; 2^k] содержали образец. То бишь элемент по индексу 2^k >= искомого, а по индексу 2^(k-1) <= искомого. Далее выполняется бинарный поиск на этом отрезке. Такой поиск быстрее бинарного, но также логарифмически зависим.

# **Часть 3. Оценка временной эффективности в терминах О-функций.**

Составим сводную таблицу эффективности для матрицы в обоих типах представлений.

|  |  |
| --- | --- |
|  | Эффективность (n длина массива) |
| Линейный | O(n) |
| Бинарный | О(log(n)) |
| Эксполяционный | O(log(n)) |
| Интерполяционныхй | O(log(log(n)) |

# **Часть 4. Программный код.**

Итоговая библиотека классов имеет следующий вид:

/// <summary>

/// Класс, добавляющий в функционал класса Sorter

/// возможность различных видов поиска элемента

/// </summary>

/// <typeparam name="T">Тип данных в массива</typeparam>

public class FinderInArray<T> : Sorter<T> where T : IComparable<T>

{

/// <summary>

/// Создает экземпляр класса на основе

/// данного массива

/// </summary>

/// <param name="arr">Данный массив</param>

public FinderInArray(T[] arr) : base(arr) { }

/// <summary>

/// Осуществляет линейный поиск в данном массиве

/// </summary>

/// <param name="elem">Искомый элемент</param>

/// <returns>

/// Индекс элемента если такой присутствует,

/// иначе -1

/// </returns>

public int LinearSearch(T elem)

{

for (int i = 0; i < len; i++)

if (arr[i].Equals(elem))

return i;

return -1;

}

/// <summary>

/// Осуществляет бинарный поиск в отсортированном

/// по возрастанию массиве

/// </summary>

/// <param name="elem">Искомый элемент</param>

/// <returns>

/// Индекс элемента если такой присутствует,

/// иначе -1

/// </returns>

public int BinarySearch(T elem)

{

int left = 0, right = len, center;

while (right > left)

{

center = (left + right) / 2;

if (arr[center].Equals(elem))

return center;

if (arr[center].CompareTo(elem) > 0)

right = center;

else

left = center + 1;

}

return -1;

}

/// <summary>

/// Осуществляет бинарный рекурсивный

/// поиск в отсортированном

/// по возрастанию массиве

/// </summary>

/// <param name="elem">Искомый элемент</param>

/// <returns>

/// Индекс элемента если такой присутствует,

/// иначе -1

/// </returns>

public int BinaryRecursiveSearch(T elem)

{

return BinRecS(elem, this.arr, 0, len);

}

/// <summary>

/// Осуществляет бинарный рекурсивный

/// поиск в отсортированном

/// по возрастанию массиве

/// </summary>

/// <param name="elem">Искомый элемент</param>

/// <param name="arr">Массив, в котором осуществляется поиск</param>

/// <param name="start">Начало проверяемого отрезка</param>

/// <param name="end">Конец проверяемого отрезка</param>

/// <returns>

/// Индекс элемента если такой присутствует,

/// иначе -1

/// </returns>

private static int BinRecS(T elem, T[] arr, int start, int end)

{

int center = (start + end) / 2;

if (start == end)

return -1;

if (arr[center].Equals(elem))

return center;

if (arr[center].CompareTo(elem) < 0)

return BinRecS(elem, arr, center + 1, end);

else

return BinRecS(elem, arr, start, center);

}

/// <summary>

/// Осуществляет интерполяционный поиск в задаваемом массиве,

/// интерполяция линейная

/// </summary>

/// <param name="elem">Искомый элемент</param>

/// <param name="array">Массив, в котором происходит поиск</param>

/// <returns>

/// Индекс элемента если такой присутствует,

/// иначе -1

/// </returns>

public static int InterpolateSearch(int elem, int[] array)

{

int left = 0, right = array.Length - 1, center;

while (right > left)

{

center = left + ((elem - array[left]) \* (right - left)) / ((array[right] - array[left]));

if (array[center].Equals(elem))

return center;

if (array[center].CompareTo(elem) > 0)

right = center;

else

left = center + 1;

}

return -1;

}

/// <summary>

/// Осуществляет экстраполяционный поиск в задаваемом массиве

/// </summary>

/// <param name="elem">Искомый элемент</param>

/// <param name="array">Массив, в котором происходит поиск</param>

/// <returns>

/// Индекс элемента если такой присутствует,

/// иначе -1

/// </returns>

public static int ExponentialSearch(int elem, int[] array)

{

if (array[0] == elem)

return 0;

int k = 1;

while (k < array.Length && array[k] < elem)

k \*= 2;

int end = k < array.Length - 1 ? k : array.Length - 1;

end++;

return FinderInArray<int>.BinRecS(elem, array, k / 2, end);

}

}

**Часть 5. Тестирование и результаты работы.**

Для тестирования было построено консольное приложение, которая позвляет искать элементы в массиве, задаваемом пользователем.

