Федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт: ИВТИ Кафедра: ВМСС

Направление подготовки: 09.03.01 Информатика и вычислительная техника

ОТЧЕТ по практике

Наименование практики: Производственная практика: *научно-исследовательская работа*

СТУДЕНТ

Доржу Н.Ш.
(подпись) (*Фамилия и инициалы*)

Группа А-12-19
(номер учебной группы)

ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ПРАКТИКЕ

(отлично, хорошо, удовлетворительно, неудовлетворительно)

/ /
(подпись) (*Фамилия и инициалы члена комиссии*)

/ /
(подпись) (*Фамилия и инициалы члена комиссии*)

Москва
2023

СОДЕРЖАНИЕ ОТЧЕТА

1.Задание 3
Введение 3
2.Основная часть 3
3.Эффективность методов 5
Заключение 5
4.Листинг программы 6

Задание:

Реализовать метод поиска строки в словаре существительных последовательным доступом и методом дихотомии. Проверить эффективность двух способов поиска. Словарь может храниться либо в файле, либо полностью быть загруженным в массив.

1.Введение:

Метод дихотомии (половинного деления): это технический прием, при котором можно исключать половину отрезка при каждом прохождении функции. Функция будет работать непрерывно до тех пор, пока оставшийся промежуток от деления не будет слишком маленького значения. При каждом прохождении промежуток делится пополам — одна половина «отбрасывается», а вторая снова делится ровно посередине. Нужно реализовать этот метод для поиска слова в словаре.

2.Основная часть:

Для выполнения данной работы я разработал несколько методов, такие как ***Find_serially()***, предназначенный для последовательного поиска строки в файле; ***Seek()*** для нахождения строки на определенной позиции в файле; ***Compare()*** нужный для того, чтобы сравнивать две строки и узнать какой из двух слов «больше» другого; ***Dyhot()***, содержащий метод поиска строки методом дихотомии в файле; ***Find_serially_arr()*** для поиска строки последовательным доступом в массиве строк; ***Dyhot_arr()*** тот же метод дихотомии только для массива строк.

Рассмотрим каждый метод подробнее.

Bool Find_serially(StreamReader file, string word);

Функция принимает на вход поток файла и искомое слово в потоке. Выводит функция значение **True**, если искомое слово было найдено или же **False**, если функция не смогла найти эту строку. Алгоритм работы функции крайне прост. Начиная с первого элемента функция сравнивает его с исходным словом, если они равны, то выход из функции со значением **True**, если слово не подходит, тогда переход на следующий элемент. Если после последнего элемента искомое слово не было найдено, то происходит выход из функции со значением **False**.

string Seek(StreamReader file, int n);

Функция принимает на вход такие аргументы как поток файла и индекс строки, которую мы просим вывести. Функция выводит строку, стоящую на том индексе, который был нам дан. Алгоритм работы функции состоит из цикла считывающий поочередно строки и инкрементируя значение i от нуля. При достижении **i=n**, функция выводит прочитанную строку.

int Compare(string w1, string w2);

Функция принимает на вход две строки. Если первая строка равняется второй, функция выводит значение 0. Если же строка в алфавитном порядке будет выше, чем вторая, тогда функция выдает значение -1, иначе 1.

bool Dyhot(string Find_word, StreamReader file, double Nmax);

Функция принимает на вход искомую строку, поток файла, а также количество элементов в этом файле. Если слово найдено, тогда True, иначе **False**. В данную функцию также включены функции ***Seek()*** и ***Compare()***. Метод выбирает строку лежащую посередине между **Nmax** и **Nmin** (на первой итерации равный 0), посредством функции ***Seek()*** и сравнивает его с искомым словом. Если искомое слово будет выше, тогда при следующей итерации, будет происходить тот же процесс только с **Nmax** равному индексу строки бывшей по середине предыдущей итерации. Если же искомое слово было ниже, производим аналогичную операцию, однако **Nmin** будет равно индексу строки бывшей по середине предыдущей итерации.

bool Find_serially_arr(string[] file, string word);

Функция выполняет аналогичную работу функции **Find_serially()** только в массиве строк.

bool Dyhot_arr(string[] file, string Find_word);

Функция аналогичная функции **Dyhot()**, только с в массиве строк.

Для удобства использования пользователем программы был создан выбор метода поиска строки и поиском в массиве или файле. А также пользователь сам вводит искомое слово и в конце программы получает результат **True** или **False**.

3.Эффективность методов:

При поиске строки в файле поиск методом последовательного доступа время работы функции лежало в промежутке от 0,88 до 7,827 миллисекунд. При методе дихотомии время будет лежать в промежутке от 7,647 до 93 миллисекунд.

При поиске строки в массиве строк поиск методом последовательного доступа время работы функции лежало в промежутке от 0,78 до 0,853 миллисекунд. При методе дихотомии время будет лежать в промежутке от 0,767 до 0,827 миллисекунд.

Результаты показывают, что при поиске из массива времени на поиск уходит гораздо меньше, однако траты памяти получаются большие. При поиске из файла выходит большое время так как нам приходится постоянно начинать считывать файл с начала и до указанного индекса.

В теории метод последовательного доступа должен иметь временную сложность **N**, а временная сложность метода дихотомии должна быть равной **Log(N).**

Заключение:

Исходя из проделанной мной работы можно понять, что поиск в массиве строк гораздо выгоднее по времени выполнения. А также мы узнали, что на поиск строки методом дихотомии уходит меньше времени, нежели чем поиск последовательным доступом.

4.Листинг программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Net.Security;

namespace NIR
{
    internal class Program
    {
        static bool Find_serially(StreamReader file, string word)//from file of
string
        {
            string temp;
            while (file.EndOfStream != true)
            {
                temp = file.ReadLine();
                if (word == temp)
                {
                    return true;
                }
            }
            file.DiscardBufferedData();
            file.BaseStream.Position = 0;
            return false;
        }

        static string Seek(StreamReader file, int n)//find n word from file
        {
            string temp="";
            for (int i = 0; i < n; i++)
            {
                temp = file.ReadLine();
            }
            file.DiscardBufferedData();
            file.BaseStream.Position = 0;
            return temp;
        }

        static int Compare(string w1, string w2)//w1=искомое w2=полученное
        {
            if (w1 == w2) return 0;
            for (int i = 0; i < w1.Length && i < w2.Length; i++)
            {
                if (w1[i] != w2[i])
                {
                    if (w1[i] < w2[i])
                    {
                        return -1;
                    }
                    else return 1;
```

```
                }
            }
            if (w1.Length < w2.Length) return -1;
                else return 1;
            //0=равные слова 1=первая половина -1=вторая половина
        }

        static bool Dyhot(string Find_word, StreamReader file, double Nmax)//for
file
        {
            string Temp_word = "";
            int Sign = 3;
            double N = Math.Ceiling(Nmax / 2);
            int z = Convert.ToInt32(N);
            double Nmin = 1;
            while (Sign!=0)
            {
                Temp_word = Seek(file, z);
                Sign = Compare(Find_word, Temp_word);
                if (Sign == 0)
                {
                    return true;
                }
                if (N == Nmin || N == Nmax)
                {
                    return false;
                }
                if (Sign == -1)
                {
                    Nmax = N;
                    N = Math.Floor(Nmax - (Nmax - Nmin) / 2);
                    z = Convert.ToInt32(N);
                }
                if (Sign == 1)
                {
                    Nmin = N;
                    N = Math.Ceiling(Nmax - (Nmax - Nmin) / 2);
                    z = Convert.ToInt32(N);
                }
            }
            return false;
        }

        static bool Find_serially_arr(string[] file, string word)//for array
        {
            for(int i = 0; i < file.Length; i++)
            {
                if (file[i] == word) return true;
            }
            return false;
        }

        static bool Dyhot_arr(string[] file, string Find_word)//for array
        {
            string Temp_word;
            double Nmax = file.Length;
```

```
            double Nmin = 0;
            double N = Math.Ceiling(Nmax / 2);
            int z = Convert.ToInt32(N);
            int Sign = 3;
            while (Sign != 0)
            {
                Temp_word = file[z];
                Sign = Compare(Find_word, Temp_word);
                if (Sign == 0)
                {
                    return true;
                }
                if (N == Nmin || N == Nmax)
                {
                    return false;
                }
                if (Sign == -1)
                {
                    Nmax = N;
                    N = Math.Floor(Nmax - (Nmax - Nmin) / 2);
                    z = Convert.ToInt32(N);
                }
                if (Sign == 1)
                {
                    Nmin = N;
                    N = Math.Ceiling(Nmax - (Nmax - Nmin) / 2);
                    z = Convert.ToInt32(N);
                }
            }
            return false;
        }
        static void Main(string[] args)
        {
            string Find_word;
            string path = "russian_words_sorted.txt";
            string[] file = File.ReadAllLines(path);
            double N = file.Length;
            Console.WriteLine("Выберите откуда искать слово:");
            Console.WriteLine("1 - Из файла");
            Console.WriteLine("2 - Из массива");
            switch (Console.ReadKey(true).Key)
            {
                case ConsoleKey.D1:
                    {
                        Console.WriteLine("Выберите способ поиска слова:");
                     Console.WriteLine("1 - Методом последовательного поиска");
                        Console.WriteLine("2 - Методом Дихотомии");
                        switch (Console.ReadKey(true).Key)
                        {
                            case ConsoleKey.D1:
                                {
                                    Console.WriteLine("Введите искомое слово");
                                    Find_word = Console.ReadLine();
                                using (StreamReader sr = new StreamReader(path))
                                Console.WriteLine(Find_serially(sr, Find_word));
                                    break;
```

```
                                    }
                                case ConsoleKey.D2:
                                    {
                                        Console.WriteLine("Введите искомое слово");
                                        Find_word = Console.ReadLine();
                                    using (StreamReader sr = new StreamReader(path))
                                         Console.WriteLine(Dyhot(Find_word, sr, N));
                                        break;
                                    }
                            }
                            break;
                        }
                    case ConsoleKey.D2:
                        {
                            Console.WriteLine("Выберите способ поиска слова:");
                          Console.WriteLine("1 - Методом последовательного поиска");
                            Console.WriteLine("2 - Методом Дихотомии");
                            switch (Console.ReadKey(true).Key)
                            {
                                case ConsoleKey.D1:
                                    {
                                        Console.WriteLine("Введите искомое слово");
                                        Find_word = Console.ReadLine();
                                  Console.WriteLine(Find_serially_arr(file, Find_word));
                                        break;
                                    }
                                case ConsoleKey.D2:
                                    {
                                        Console.WriteLine("Введите искомое слово");
                                        Find_word = Console.ReadLine();
                                     Console.WriteLine(Dyhot_arr(file, Find_word));
                                        break;
                                    }
                            }
                            break;
                        }
                }
                Console.ReadKey();
            }
        }
}
```

ГРАФИК ПРОХОЖДЕНИЯ НИР

Номер п/п	Перечень работ в соответствии с заданием	Отметка о выполнении работы (выполнено /не выполнено)
1		
2		
3		
4		
5		

Руководитель практики (от МЭИ) _____ / _____ /

(подпись) (*Фамилия и инициалы*)