ЛАБОРАТОРНАЯ РАБОТА №8

Тема: Алгоритмы поиска.

Цель: Научиться программировать простейшие алгоритмы поиска элементов в массиве и подстрок в строках.

Ход работы

Вариант 8

1. Реализовать алгоритмы поиска: линейный, бинарный, интерполяционный. Данные взять из файла *sorted.dat*, созданного в лабораторной работе №7. Для каждого алгоритма должны выводиться на консоль следующие данные: позиция найденного элемента (или сообщение «Не найдено»), время работы алгоритма («секунды : миллисекунды»), количество сравнений.

Код:

```
using System;
using System.Diagnostics;
using System.IO;

Ссылок: 0
```

```
static void Main()
{
    string filePath = "sorted.dat";

    if (!File.Exists(filePath))
    {
        Console.WriteLine("Файл sorted.dat не найден.");
        return;
    }

    string[] parts = File.ReadAllText(filePath)
        .Split(new char[] { ' ', ',', '\n', '\t' }, StringSplitOptions.RemoveEmptyEntries);

    int[] numbers = new int[parts.Length];
    int count = 0;

    for (int i = 0; i < parts.Length; i++)
    {
        if (int.TryParse(parts[i], out int num))
        {
            numbers[count] = num;
            count++;
        }
    }

    Array.Resize(ref numbers, count);

    if (numbers.Length == 0)
    {
        Console.WriteLine("Файл не содержит чисел.");
        return;
    }
```

```
    Console.Write("Введите число для поиска: ");
    if (!int.TryParse(Console.ReadLine(), out int target))
    {
        Console.WriteLine("Некорректный ввод.");
        return;
    }

    // Поиск
    LinearSearch(numbers, target);

    BinarySearch(numbers, target);

    InterpolationSearch(numbers, target);
}
```

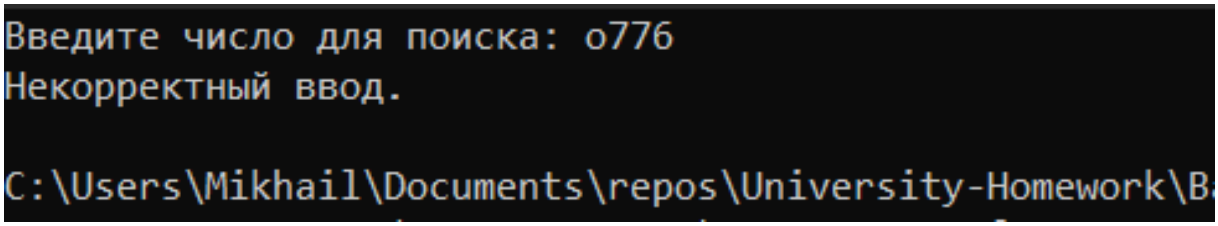
```
static void LinearSearch(int[] arr, int x)
{
    int comparisons = 0;
    Stopwatch sw = Stopwatch.StartNew();

    for (int i = 0; i < arr.Length; i++)
    {
        comparisons++;
        if (arr[i] == x)
        {
            sw.Stop();
            PrintResult("Линейный", i, sw.Elapsed, comparisons);
            return;
        }
    }

    sw.Stop();
    PrintResult("Линейный", -1, sw.Elapsed, comparisons);
}
```

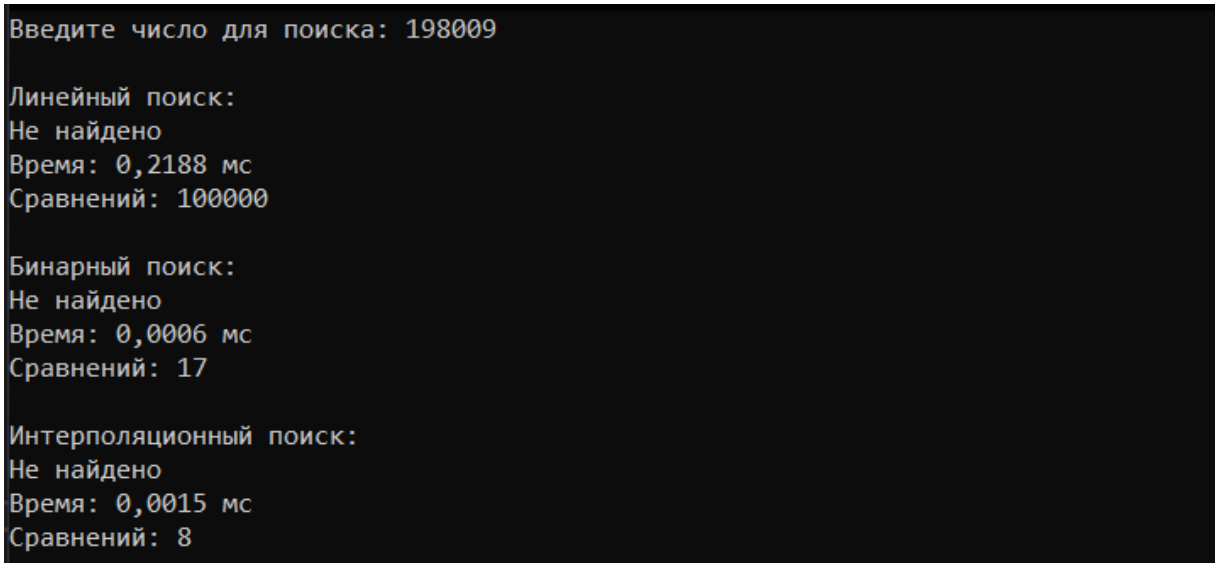
```
static void BinarySearch(int[] arr, int x)
{
    int left = 0, right = arr.Length - 1;
    int comparisons = 0;
    Stopwatch sw = Stopwatch.StartNew();

    while (left <= right)
    {
        comparisons++;
        int mid = (left + right) / 2;
        if (arr[mid] == x)
        {
            sw.Stop();
            PrintResult("Бинарный", mid, sw.Elapsed, comparisons);
            return;
        }
        else if (arr[mid] < x)
        {
            left = mid + 1;
        }
        else
        {
            right = mid - 1;
        }
    }

    sw.Stop();
    PrintResult("Бинарный", -1, sw.Elapsed, comparisons);
}
```

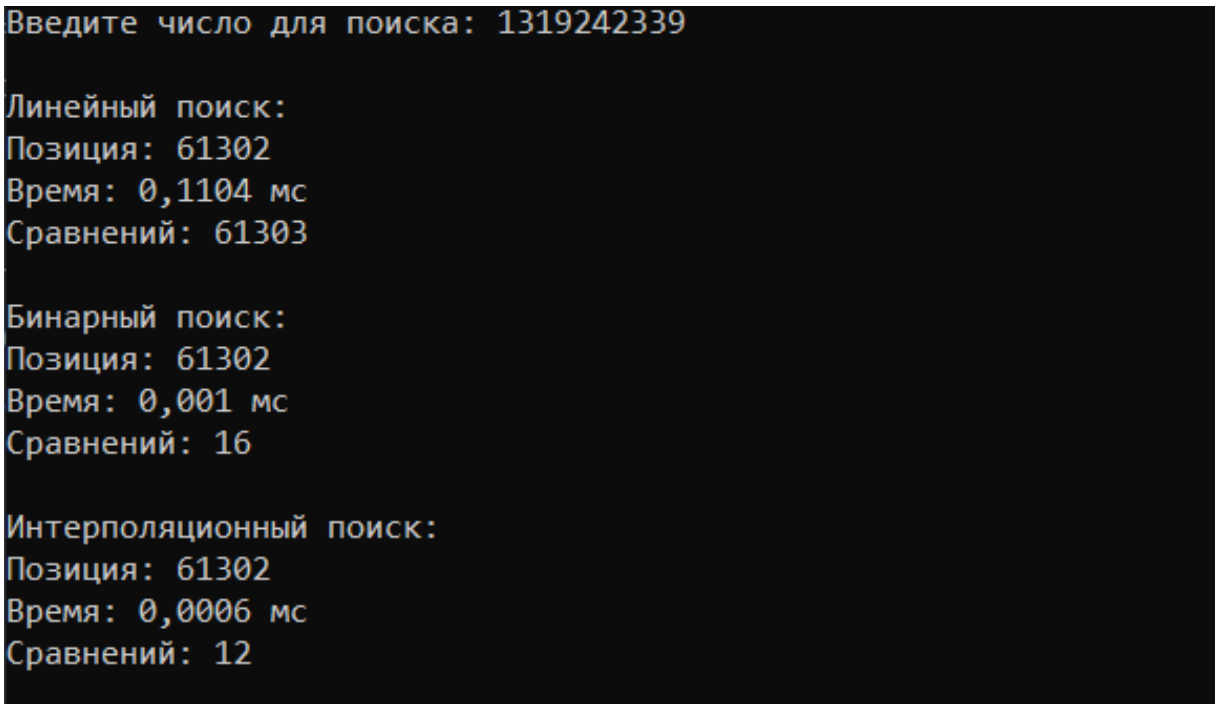
```
static void InterpolationSearch(int[] arr, int x)
{
    int comparisons = 0;
    int low = 0, high = arr.Length - 1;
    Stopwatch sw = Stopwatch.StartNew();

    while (low <= high && x >= arr[low] && x <= arr[high])
    {
        comparisons++;

        if (arr[high] == arr[low])
        {
            if (arr[low] == x)
            {
                sw.Stop();
                PrintResult("Интерполяционный", low, sw.Elapsed, comparisons);
                return;
            }
            else
                break;
        }

        int pos = low + (int)(((long)(x - arr[low]) * (high - low)) / (arr[high] - arr[low]));

        // Защита от выхода за границы
        if (pos < low || pos > high)
            break;

        comparisons++;

        if (arr[pos] == x)
        {
            sw.Stop();
            PrintResult("Интерполяционный", pos, sw.Elapsed, comparisons);
            return;
        }

        if (arr[pos] < x)
            low = pos + 1;
        else
            high = pos - 1;
    }

    sw.Stop();
    PrintResult("Интерполяционный", -1, sw.Elapsed, comparisons);
}
```

```
Ссылок: 6
static void PrintResult(string method, int index, TimeSpan time, int comparisons)
{
    Console.WriteLine($"\n{method} поиск:");

    if (index != -1)
        Console.WriteLine("Позиция: " + index);
    else
        Console.WriteLine("Не найдено");

    Console.WriteLine("Время: " + time.TotalMilliseconds.ToString("0.###") + " мс");

    Console.WriteLine("Сравнений: " + comparisons);
}
```

Тестирование:







2. Реализовать алгоритмы КМП, БМ, простого поиска подстроки и проверить на различных тестах. Тестовые строки (искомые подстроки) должны быть представлены произвольными строками и строками с повторяющимися фрагментами. Сравнить эффективность одних тех же алгоритмов для разных подстрок. Для каждого алгоритма должны выводиться на консоль следующие данные: позиция найденного элемента (или сообщение «Не найдено»), время работы алгоритма («секунды : миллисекунды»), количество сравнений.

Код:

```
using System;
using System.Diagnostics;
using System.IO;

Ссылок: 0
class Program
{
    Ссылок: 0
    static void Main()
    {
        string[] files = { "test1.txt", "test2.txt", "test3.txt", "test4.txt", "test5.txt" };

        Console.Write("Введите подстроку для поиска: ");
        string pattern = Console.ReadLine();

        foreach (var file in files)
        {
            if (!File.Exists(file))
            {
                Console.WriteLine($"Файл {file} не найден.");
                continue;
            }

            string text = File.ReadAllText(file);
            Console.WriteLine($"\n===== {file} =====");

            RunSearch("Простой", SimpleSearch, text, pattern);
            RunSearch("КМП", KMPSearch, text, pattern);
            RunSearch("Бойер-Мур", BoyerMooreSearch, text, pattern);
        }
    }
```

```
static void RunSearch(string name, Func<string, string, (int pos, int comparisons)> searchFunc, string text, string pattern)
{
    Stopwatch sw = Stopwatch.StartNew();
    var (position, comparisons) = searchFunc(text, pattern);
    sw.Stop();

    Console.WriteLine($"\n{name} поиск:");
    Console.WriteLine(position != -1 ? $"Позиция: {position}" : "Не найдено");

    // Точное отображение времени: до микросекунд (6 знаков)
    Console.WriteLine($"Время: {sw.Elapsed.TotalMilliseconds:0.0000} мс");

    Console.WriteLine($"Сравнений: {comparisons}");
}
```

```
static (int, int) SimpleSearch(string text, string pattern)
{
    int comparisons = 0;
    for (int i = 0; i <= text.Length - pattern.Length; i++)
    {
        int j;
        for (j = 0; j < pattern.Length; j++)
        {
            comparisons++;
            if (text[i + j] != pattern[j])
                break;
        }

        if (j == pattern.Length)
            return (i, comparisons);
    }
    return (-1, comparisons);
}
```

```
static (int, int) KMPSearch(string text, string pattern)
{
    int[] lps = BuildLPS(pattern);
    int i = 0, j = 0, comparisons = 0;

    while (i < text.Length)
    {
        comparisons++;
        if (text[i] == pattern[j])
        {
            i++; j++;
            if (j == pattern.Length)
                return (i - j, comparisons);
        }
        else
        {
            if (j != 0)
                j = lps[j - 1];
            else
                i++;
        }
    }
    return (-1, comparisons);
}
```

```
static int[] BuildLPS(string pattern)
{
    int[] lps = new int[pattern.Length];
    int len = 0, i = 1;
    while (i < pattern.Length)
    {
        if (pattern[i] == pattern[len])
        {
            len++;
            lps[i] = len;
            i++;
        }
        else
        {
            if (len != 0)
                len = lps[len - 1];
            else
            {
                lps[i] = 0;
                i++;
            }
        }
    }
    return lps;
}
```

```
static (int, int) BoyerMooreSearch(string text, string pattern)
{
    int comparisons = 0;
    var badChar = BuildBadCharTable(pattern);
    int shift = 0;

    while (shift <= text.Length - pattern.Length)
    {
        int j = pattern.Length - 1;

        while (j >= 0 && text[shift + j] == pattern[j])
        {
            comparisons++;
            j--;
        }

        if (j < 0)
        {
            return (shift, comparisons);
        }

        comparisons++;

        char mismatchedChar = text[shift + j];
        int badCharIndex = badChar.ContainsKey(mismatchedChar) ? badChar[mismatchedChar] : -1;

        shift += Math.Max(1, j - badCharIndex);
    }

    return (-1, comparisons);
}
```

```
static Dictionary<char, int> BuildBadCharTable(string pattern)
{
    var table = new Dictionary<char, int>();
    for (int i = 0; i < pattern.Length; i++)
    {
        table[pattern[i]] = i;
    }
    return table;
}
```

Тестирование:

```
У лукоморья дуб зелёный;
Златая цепь на дубе том:
И днём и ночью кот учёный
Всё ходит по цепи кругом;
Идёт направо — песнь заводит,
Налево — сказку говорит.
Там чудеса: там леший бродит,
Русалка на ветвях сидит;
Там на неведомых дорожках
Следы невиданных зверей;
Избушка там на курьих ножках
Стоит без окон, без дверей;
Там лес и дол видений полны;
Там о заре прихлынут волны
На брег песчаный и пустой,
И тридцать витязей прекрасных
Чредой из вод выходят ясных,
И с ними дядька их морской;
Там королевич мимоходом
Пленяет грозного царя;
Там в облаках перед народом
Через леса, через моря
Колдун несёт богатыря;
В темнице там царевна тужит,
А бурый волк ей верно служит;
Там ступа с Бабою Ягой
Идёт, бредёт сама собой,
Там царь Кащей над златом чахнет;
Там русский дух… там Русью пахнет!
И там я был, и мёд я пил;
У моря видел дуб зелёный;
Под ним сидел, и кот учёный
Свои мне сказки говорил.
```

test2 – Блокнот

Файл Правка Формат Вид Справка

Найти тему

Сайт психологов b17.ru
119,9 тыс подписчиков
Подписаться
Какого цвета ваша грусть
4 февраля 2024
111
1 мин
Вы никогда не задумывались, какого цвета ваша грусть? Ну, например: серый - обыденная, фиолетовый - экзистенциальная, зелёный - природная, красный - чувственная и пр. Сегодня мы попробуем проанализироваить ощущение эмоций с точки зрения цветовых ассоциаций, и начнем с грусти.

Действительно, иногда нам бывает просто грустно, непонятно от чего. Так сидишь дома, у телевизора, пьешь чай, о чем-то думаешь - и как-то муторно на душе. Грустно, что уже 40, а не 20 лет, что очередные отношения не сложились, что острые снежинки скребут по стеклу.... А так всё - ничего.

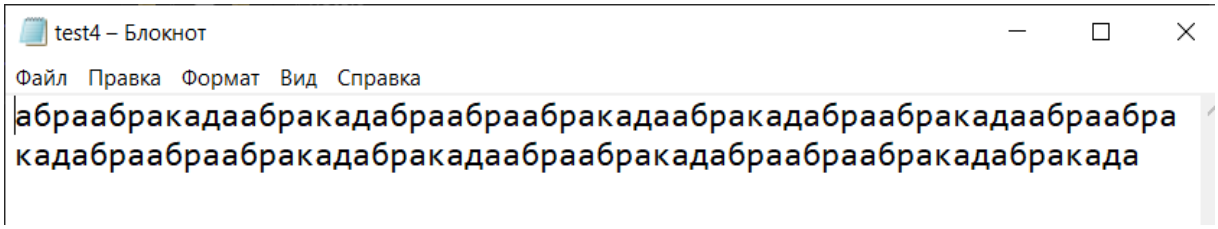
Это серый обеденный оттенок грусти, она как бы есть, и ее нет - ведь все находится на своих местах, идет своим чередом

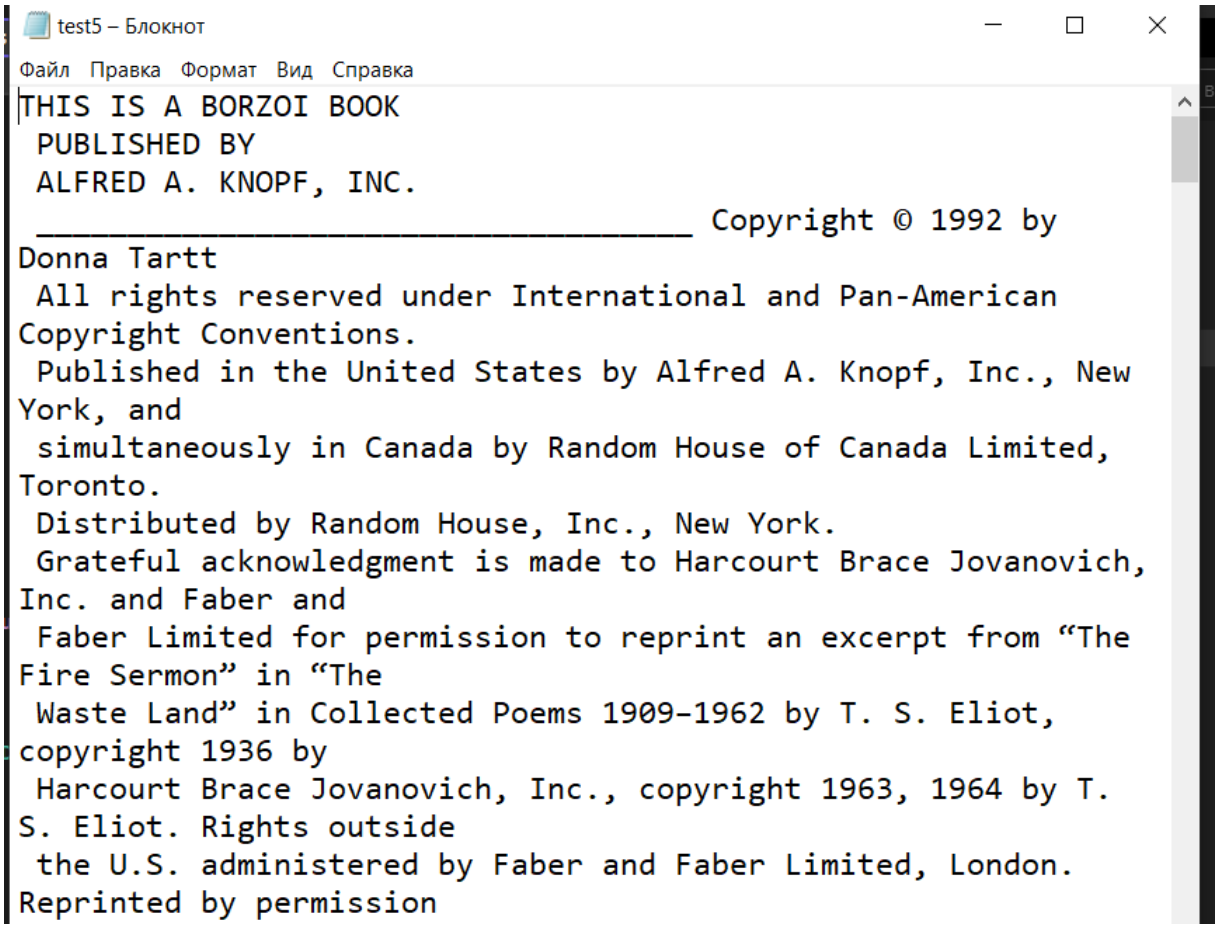
Условия и особенности выращивания граната
Уход в открытом грунте
Полив и мульчирование
Подкормки
Как обрезать гранат
Укрытие на зиму
Выращивание в домашних условиях граната
Выращивание экзотических культур в саду, к примеру, граната, уже не кажется бесперспективным благодаря селекционным работам. Культура плодоносит даже в условиях укороченного лета, а в холодные зимы не вымерзает. Какие особенности выращивания и ухода за кустом граната нужно знать?

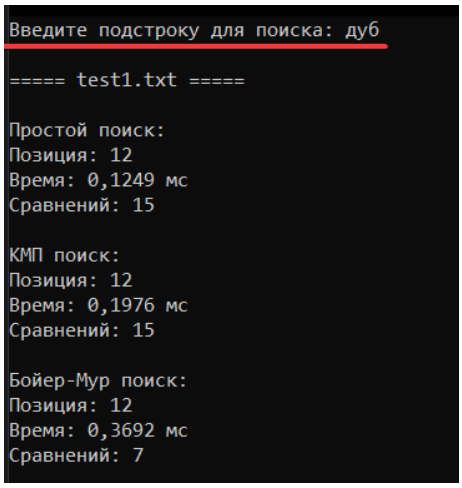
granat5-min.jpg

Условия и особенности выращивания граната

Традиционно выращивание граната осуществляют в южных регионах с мягкими, теплыми зимами. Однако в процессе длительных селекционных работ были выведены новые, более морозостойкие


test4 – Блокнот
Файл Правка Формат Вид Справка
абраабракадаабракадабраабраабракадаабракадабраабракадаабраабра
кадабраабраабракадабракадаабраабракадабраабраабракадабракада


test5 – Блокнот
Файл Правка Формат Вид Справка
THIS IS A BORZOI BOOK
PUBLISHED BY
ALFRED A. KNOPF, INC.
Copyright © 1992 by
Donna Tartt
All rights reserved under International and Pan-American
Copyright Conventions.
Published in the United States by Alfred A. Knopf, Inc., New
York, and
simultaneously in Canada by Random House of Canada Limited,
Toronto.
Distributed by Random House, Inc., New York.
Grateful acknowledgment is made to Harcourt Brace Jovanovich,
Inc. and Faber and
Faber Limited for permission to reprint an excerpt from “The
Fire Sermon” in “The
Waste Land” in Collected Poems 1909–1962 by T. S. Eliot,
copyright 1936 by
Harcourt Brace Jovanovich, Inc., copyright 1963, 1964 by T.
S. Eliot. Rights outside
the U.S. administered by Faber and Faber Limited, London.
Reprinted by permission


Введите подстроку для поиска: дуб
===== test1.txt =====
Простой поиск:
Позиция: 12
Время: 0,1249 мс
Сравнений: 15
КМП поиск:
Позиция: 12
Время: 0,1976 мс
Сравнений: 15
Бойер-Мур поиск:
Позиция: 12
Время: 0,3692 мс
Сравнений: 7

```
===== test2.txt =====

Простой поиск:
Не найдено
Время: 0,0482 мс
Сравнений: 5449

КМП поиск:
Не найдено
Время: 0,0265 мс
Сравнений: 5440

Бойер-Мур поиск:
Не найдено
Время: 0,3176 мс
Сравнений: 1844

===== test3.txt =====

Простой поиск:
Не найдено
Время: 0,0651 мс
Сравнений: 6035

КМП поиск:
Не найдено
Время: 0,0332 мс
Сравнений: 6024

Бойер-Мур поиск:
Не найдено
Время: 0,0482 мс
Сравнений: 2028

===== test4.txt =====

Простой поиск:
Не найдено
Время: 0,0021 мс
Сравнений: 133

КМП поиск:
Не найдено
Время: 0,0016 мс
Сравнений: 135

Бойер-Мур поиск:
Не найдено
Время: 0,0033 мс
Сравнений: 57
```

```
===== test5.txt =====

Простой поиск:
Не найдено
Время: 0,5225 мс
Сравнений: 50057

КМП поиск:
Не найдено
Время: 0,2333 мс
Сравнений: 50059

Бойер-Мур поиск:
Не найдено
Время: 0,3881 мс
Сравнений: 16686
```

```
Введите подстроку для поиска: аг

===== test1.txt =====

Простой поиск:
Не найдено
Время: 0,1796 мс
Сравнений: 979

КМП поиск:
Не найдено
Время: 0,2592 мс
Сравнений: 980

Бойер-Мур поиск:
Не найдено
Время: 0,6061 мс
Сравнений: 482

===== test2.txt =====

Простой поиск:
Не найдено
Время: 0,0499 мс
Сравнений: 5624

КМП поиск:
Не найдено
Время: 0,0295 мс
Сравнений: 5625

Бойер-Мур поиск:
Не найдено
Время: 0,0773 мс
Сравнений: 2790

===== test3.txt =====

Простой поиск:
Позиция: 478
Время: 0,0053 мс
Сравнений: 528

КМП поиск:
Позиция: 478
Время: 0,0046 мс
Сравнений: 528

Бойер-Мур поиск:
Позиция: 478
Время: 0,0087 мс
Сравнений: 252

===== test4.txt =====

Простой поиск:
Не найдено
Время: 0,0024 мс
Сравнений: 181

КМП поиск:
Не найдено
Время: 0,0021 мс
Сравнений: 182

Бойер-Мур поиск:
Не найдено
Время: 0,0040 мс
Сравнений: 79
```

```
===== test5.txt =====

Простой поиск:
Не найдено
Время: 0,4186 мс
Сравнений: 50058

КМП поиск:
Не найдено
Время: 0,3715 мс
Сравнений: 50059

Бойер-Мур поиск:
Не найдено
Время: 0,5206 мс
Сравнений: 25029
```

```
Введите подстроку для поиска: home

===== test1.txt =====

Простой поиск:
Не найдено
Время: 0,1300 мс
Сравнений: 920

КМП поиск:
Не найдено
Время: 0,1978 мс
Сравнений: 923

Бойер-Мур поиск:
Не найдено
Время: 0,3793 мс
Сравнений: 230
```

```
===== test5.txt =====

Простой поиск:
Позиция: 7166
Время: 0,0665 мс
Сравнений: 7475

КМП поиск:
Позиция: 7166
Время: 0,0478 мс
Сравнений: 7453

Бойер-Мур поиск:
Позиция: 7166
Время: 0,0692 мс
Сравнений: 2143
```

Контрольные вопросы

Лабораторная работа №8

Тема: алгоритмы поиска

Цель: научиться прогр. программировать простейшие алг. поиска элементов в массиве и подстрок в строках.

Контрольные вопросы

1. Описание и сравнение простых алг. поиска:

- Линейный поиск: алг. перебирает все элементы массива по очереди, начиная с первого, пока не найдёт искомый элемент.

Время выполнения: $O(n)$

- Бинарный поиск: исп. для поиска эл. в отсорт. массиве. Алгоритм делит массив пополам и выбирает сторону, в к-й имен находится искомый элемент. Дальше он продолжает

эти операции надо остановиться частью и т.д., пока не найдёт эл.

Время выполнения: $O(\log n)$

Преимущ.: Быстрый, эффективен на отсорт. данных.

Недостатки: требует предв. сортировка.

- Интерполяционный поиск: модификация бинарного поиска, где при каждом шаге исп. оценка, где может находиться искомый эл., исходя из его знач. и знач. концов массива.

Время выполн: ~~$O(\log \log$~~ $O(\log \log n)$ в лучшем случае, $O(n)$ - в худшем;

+ быстрее бинарного, если данные равномерно распр.;

- требует равномерности распр. данных.

Красько Михаил ИВТ-4

2. Алг. простого поиска подстроки - это метод, при к-м для каждой позиции в строке ищется совпадение подстроки с осн. строкой. Для этого сдвигали подстроку по всей строке и на каждом шаге сравнивали символы.

Преимущ.:

- простота реализации,
- хорошо работает для коротких строк.

Недостатки:

- сложность вычисл: $O(n \cdot m)$, где n - длина осн. строки, а m - длина подстроки.

3. Алгоритм КМП

Для каждой позиции в подстроке вычисл. длина наиб. суффикса, к-й явл.

префиксам этой подстроки. Это позволяет при несов. не начинать поиск с первого символа, а сразу с позиции, соотв. префикс-ф-ции.

Преимущ.:

- более быстрый: $O(n+m)$, где n - длина строки, а m - длина подстроки.

Недостатки: - требует предв. обр. подстроки, что увел. сложность при маленьких строках.

4. Алгоритм Бойера - Мура

Это алг. поиска подстроки, к-й выполняет поиск с конца подстроки, двигаясь справа налево. Он исп.:

- эвристику хорошего суффикса: если часть подстроки совпала, но потом произошел сбой, алг. сдвигает подстроку;
- эвристику плохого символа: если

символ в строке не совпал с символом подстроки, алгоритм сдвигает подстроку таким образом, чтобы символ, не совпавший, стал в нужной позиции.

Преимущества:

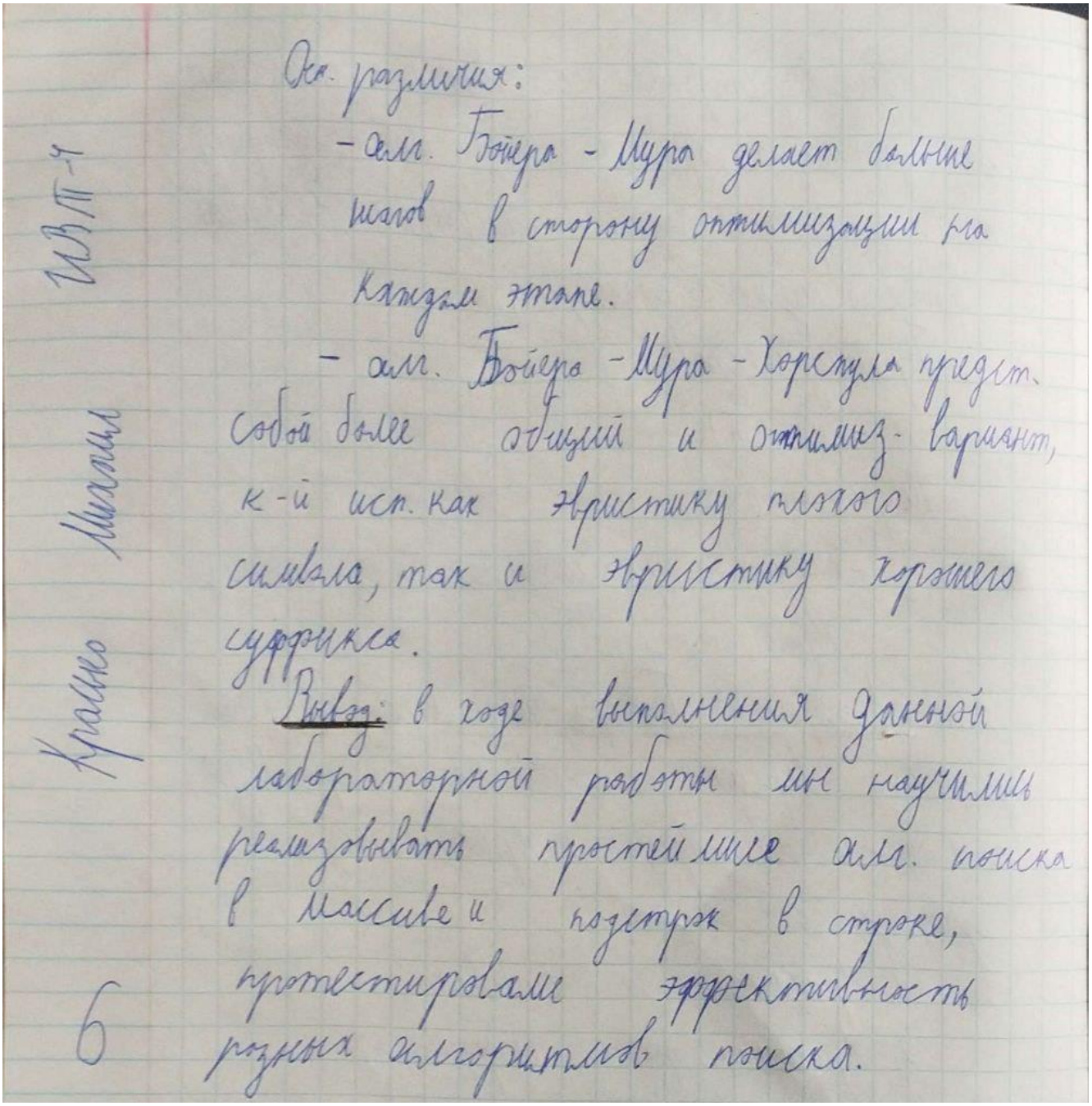
- $O(\frac{n}{m})$.

Недостатки:

- сложная реализация из-за таблиц эвристик.

5. Алгоритм Бойера-Мура-Хорспула - это улучш. версия алгоритма Бойера-Мура, к-я ~~исп~~ комбинирует 2 эвристики из предыдущего алг.:

1. эвристика плохого символа;
2. эвристика хорошего суффикса;
3. доп. эвристика смещения: алг. исп. прямой переход к позиции, где будет возможен след. успешный поиск.



Вывод: в ходе выполнения данной лабораторной работы мы научились реализовывать простейшие алгоритмы поиска в массиве и подстрок в строке, протестировали эффективность разных алгоритмов поиска.