

Санкт-Петербургский Национальный Исследовательский Университет  
Информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

**Домашняя работа №1**  
**Вариант №8**

Выполнили:  
Смирнов И.И.  
Телунц Э.Р.  
Царёв А.С.  
Проверил:  
Мусаев А.А.

Санкт-Петербург  
2022

# СОДЕРЖАНИЕ

Стр.

<b>ВВЕДЕНИЕ .....</b>	<b>3</b>
<b>1 Задание 1 .....</b>	<b>4</b>
1.1 Нахождение простых чисел .....	4
1.2 Алгоритм наивного поиска .....	4
1.3 Алгоритм Рабина-Карпа .....	4
1.4 Алгоритм Бойера-Мура .....	5
1.5 Алгоритм Кнута-Мориса-Пратта .....	5
<b>2 Задание 2 .....</b>	<b>7</b>
2.1 Выбор алгоритма для работы .....	8
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>9</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....</b>	<b>10</b>

## ВВЕДЕНИЕ

Для становления хорошим специалистом в области программирования на языке Python необходимо знать основные алгоритмы и функционал языка.

Цель данной работы – ознакомление с алгоритмами поиска в глубину и ширину на языке программирования Python.

В ходе выполнения лабораторной работы были решены следующие задачи:

- создание программ с использованием алгоритмов поиска подстроки в строке(наивный алгоритм, алгоритм Рабина-Карпа, алгоритм Бойера-Мура, алгоритм Кнутта-Мориса-Пратта);

- создание программы для обнаружения плагиата из сайта Wikipedia в тексте реферата. За плагиат считалось совпадение 3 идущих подряд слов.

Задания, которые необходимо выполнить:

1. Задание 1: составить строку из ряда первых 500 простых чисел. Выяснить, какое двузначное число(пара цифр) встречается в этой строке наибольшее количество раз, используя алгоритм наивного поиска, алгоритм Рабина-Карпа, алгоритм Бойера-Мура, алгоритм Кнутта-Мориса-Пратта.

2. Задание 2: написать программу, которая сопоставит текст реферата и текст из сайта Wikipedia и оценит в процентах количество плагиата в реферате. За плагиат считалось совпадение 3 идущих подряд слов.

## 1 Задание 1

### 1.1 Нахождение простых чисел

Для поиска первых 500 простых чисел и добавления их в массив была написана функция, которая проверяет число на делимость на все числа до корня из этого числа. При положительном результате число добавляется в массив. Когда массив достигает необходимой длины в 500 чисел, массив преобразовывается в строку.

### 1.2 Алгоритм наивного поиска

Поиск наиболее частой пары чисел в строке был начат с использованием алгоритма наивного поиска. Для этого была написана функция, которая с помощью цикла проходила по всей строке. Также был заранее создан массив из 89 элементов (отрезок [10, 99]). При каждой итерации два элемента строки склеивались и переводились в тип `int`, а значение массива, равное значению получившегося числа, из которого было вычтено число 10, прибавлялось на один. По истечении работы алгоритма, было выяснено, что наиболее часто в строке появляется число 73.

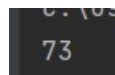


Рисунок 1 - вывод алгоритма наивного поиска

### 1.3 Алгоритм Рабина-Карпа

Для выполнения данного алгоритма, необходимо рассчитывать хэш у всех рассматриваемых подстрок. Именно поэтому была написана функция, которая рассчитывает это значение по формуле:  $\sum_{i=0}^n m * n^{(n-i)}$ . Далее используется цикл для прохода по всем натуральным двузначным числам. За одну итерацию для числа рассчитывается хэш, после чего начинается второй цикл для прохода по строке чисел, в котором рассчитывается хэш каждой пары

чисел и сравнивается с хэшем числа. В случае совпадения хэшей, происходит рассмотрение самих подстрок, чтобы избежать ошибочных совпадений в том случае, когда хэш совпал, а подстроки являются разными. По истечении работы алгоритма, было выяснено, что наиболее часто в строке появляется число 73.

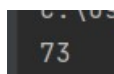


Рисунок 2 - вывод алгоритма Рабина-Карпа

#### 1.4 Алгоритм Бойера-Мура

Первым делом в данном алгоритме для каждой цифры составляется минимальное расстояние до конца числа. Далее с помощью цикла алгоритм проходит по всей строке элементов, начиная сравнение первой подстроки и заданного шаблона. В случае совпадения счетчик прибавится на единицу и произойдет сдвиг строки на единицу вправо. В случае не совпадения, происходит проверка наличия какого-либо элемента из шаблона в рассматриваемой подстроке. Если такой элемент удастся найти, то строка смещается до момента, пока одинаковые элементы подстроки и шаблона не окажутся на одинаковой позиции. Если такой элемент не удалось найти, то вся строка смещается на длину шаблона. По истечении работы алгоритма, было выяснено, что наиболее часто в строке появляется число 73.

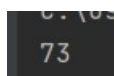


Рисунок 3 - вывод алгоритма Бойера-Мура

#### 1.5 Алгоритм Кнутта-Мориса-Пратта

В данном алгоритме для каждого числа из отрезка  $[10,99]$  рассчитывается префикс, после чего происходит сравнение шаблона и подстроки с использованием рассчитанных префиксов. В случае совпадения счетчик прибавляется на единицу, в ином случае строка смещается вправо на один или

до нового совпавшего префикса. По истечении работы алгоритма, было выяснено, что наиболее часто в строке появляется число 73.

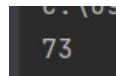
A small, dark rectangular box containing the number 73. The box is positioned centrally below the text paragraph and above the caption. The number 73 is white and stands out against the dark background of the box.

Рисунок 4 - вывод алгоритма Кнута-Мориса-Пратта

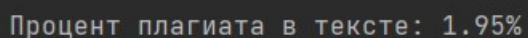
## 2 Задание 2

Задача по нахождению плагиата была разделена на 3 основных этапа: Обработка статьи "Стресс" с сайта Wikipedia, обработка текста реферата на тему "Стресс сравнение двух текстов.

Для обработки текста с Wikipedia была использована библиотека с названием Wikipedia. С ее помощью весь текстовый контент страницы был пере-записан в файл для дальнейшего использования. Далее из созданного файла все строки были перенесены в массив и объединены в строку, из которой впоследствии были удалены все символы кроме букв и цифр, а также лишние пробелы. Полученная строка была переведена в массив. Далее с помощью цикла элемента массива образовывались в тройки элементов, для которых высчитывался хэш. Такие тройки и хэши записывались в новые массивы и имели в них одинаковые индексы.

Для обработки текста реферата была использована библиотека docx (python-docx). С её помощью все абзацы текста были перенесены как элементы массива, который далее был объединен в строку, из которой впоследствии были удалены все символы кроме букв и цифр, а также лишние пробелы. Полученная строка была переведена в массив. Далее с помощью цикла элемента массива образовывались в тройки элементов, для которых высчитывался хэш. Такие тройки и хэши записывались в новые массивы и имели в них одинаковые индексы.

На этапе сравнения был создан счетчик для отслеживания повторов. Далее двойным циклом происходит проверка соответствий хэша троек элементов из Википедии с хэшем троек элементов из реферата. Счетчик прибавляется на один в случае соответствия хэшей и соответствующих им троек элементов. Чтобы вычислить процент плагиата, значение счетчика было разделено на общее количество слов в реферате и умножено на 100%. Результат программы: 1.95%.



Процент плагиата в тексте: 1.95%

Рисунок 5 - вывод программы по поиску плагиата

## 2.1 Выбор алгоритма для работы

Сравнение двух текстов было выполнено при помощи алгоритма Рабина-Карпа, работающего на основе хэшей. Выбор данного алгоритма обусловлен тем, что вычисление и сравнение хэшей занимает меньше времени, чем лексикографическое сравнение слов при больших объёмах данных.



## ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был получен опыт работы на языке Python и написания и применения алгоритмов для поиска подстрок в строке, а именно алгоритма наивного поиска, алгоритма Рабина-Карпа, алгоритма Бойера-Мура, алгоритма Кнутта-Мориса-Пратта.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Wikipedia: официальный сайт: <https://ru.wikipedia.org/wiki> (Дата обращения 19.02.2023)

Ссылка на полный код

[https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%  
2Fteecortex%2FTasksForAaDS&cc\\_key=](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2Fteecortex%2FTasksForAaDS&cc_key=)