Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

**Домашняя работа №1**

**Вариант № 4**

Выполнили:

Бакланова А.Г,

Крылов Д.С,

Улитина М.С

Проверил Мусаев А.А.

Санкт-Петербург, 2023

Оглавление

[Оглавление 2](#_Toc128350629)

[Введение. 3](#_Toc128350630)

[Алгоритмы выполнения заданий. 4](#_Toc128350631)

[Задание 1. 4](#_Toc128350632)

[Задание 2. 7](#_Toc128350633)

[Заключение. 9](#_Toc128350634)

[Источники информации. 10](#_Toc128350635)

# 

# Введение.

Цель лабораторной работы – выполнение заданий с помощью Python. А именно создать программы с алгоритмом поиска подстрок. Оба задания выполняются с учетом варианта. Вариант работы: 4; тема реферата для второго задания: рентгеновское излучение.Первое задание требует найти количество наиболее часто встречающихся двузначных чисел из строки простых чисел. Сравнить изученные алгоритмы поиска подстроки, сделать вывод о достоинствах и недостатках. Вторая задача – это антиплагиат. В ней необходимо сравнить две статьи и вывести процент плагиата.

# 

# Алгоритмы выполнения заданий.

## Задание 1.

В первом задании нам необходимо создать массив и заполнить его первыми 500 простыми числами, написанными слитно. После, используя такие методы поиска подстрок: наивный, Рабина-Карпа, Бойера-Мура, Кнута-Морриса-Пратта, вычислить количество наиболее часто встречающихся двузначных чисел в этой строке.

На рисунке 1 представлен алгоритм для поиска всех простых чисел до n-ого значения. Происходит ввод числа n, цикл while работает до тех пор, пока оно больше счетчика k, поочередно добавляем все просты числа, которые получаем, используя sympy.isprime из библиотеку sympy, после каждого добавления счетчик k увеличивается на 1, также увеличивается и значение i.

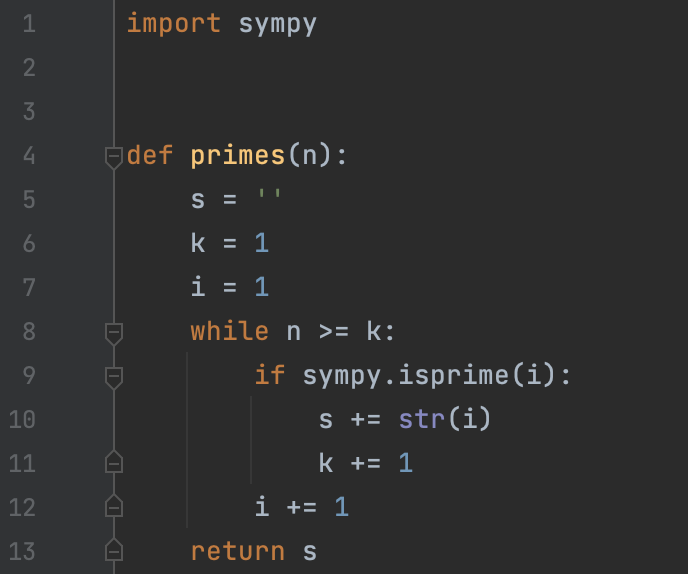


Рисунок 1 – Алгоритм поиска простых чисел

Первый реализованный алгоритм – наивный (рисунок 2). Создаем массив, который заполнен нулями. В нем 90 элементов, так как всего существует 90 различных двузначных чисел. С помощью цикла for проходим по длине простых чисел, переменной number поочередно присваиваем значения двух соседних элементов. Если оно больше или равно 10, то в созданном массиве прибавляем в соответствующий раздел единицу. Таким образом, нам вернется массив, где будет видно, какое число сколько раз встречается.

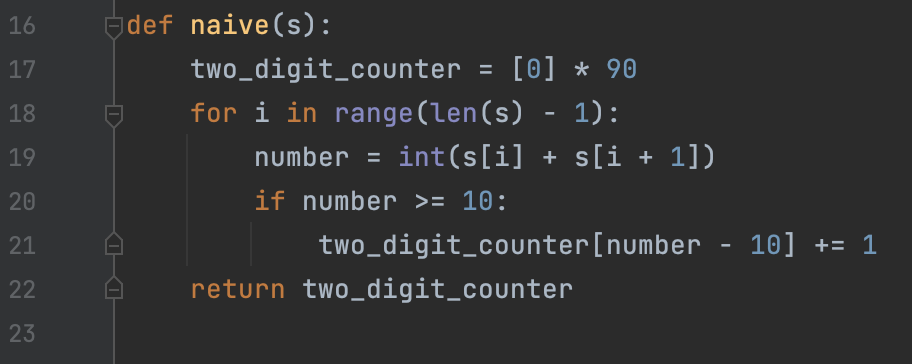


Рисунок 2 – Наивный алгоритм поиска подстрок

На рисунке 3 представлен алгоритм поиска Рабина-Карпа. Также создается массив из 90 ячеек, заполненных нулями. Дальше записываем размер алфавита в переменную dim, который равен 10. Создаем массив с цифрами от 0 до 9. После этого производим расчет хэша для шаблона. Затем расчет хэша для элементов строки, равных длине шаблона. И поочередно их сравниваем. На выходе так же получаем массив, в котором заключена информация о количестве повторений.

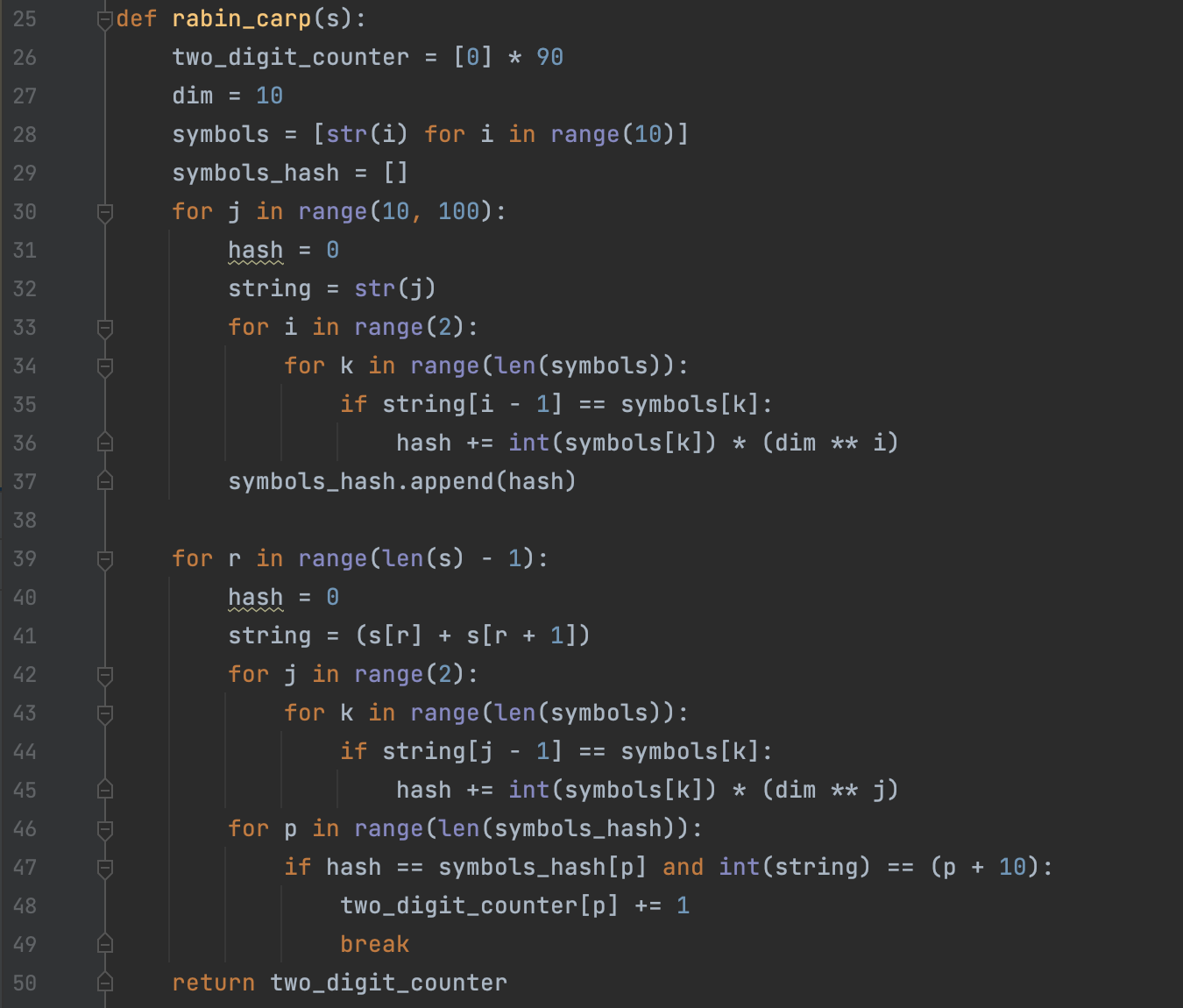


Рисунок 3 – Алгоритм поиска подстрок Рабина-Карпа

На рисунке 4 можем видеть алгоритм поиска подстрок Бойера-Мура. Аналогично с предыдущими создается массив. Также формируем число из двух соседних элементов. В случае если оно двузначное, отображаем это добавлением единицы в нужное место массива и увеличиваем переменную i на один. В случае если число больше или равно единице также увеличиваем i на один. Если число не двузначное и не больше или равно 1, увеличиваем i на два. Возвращается в результате также массив с информацией о повторениях.

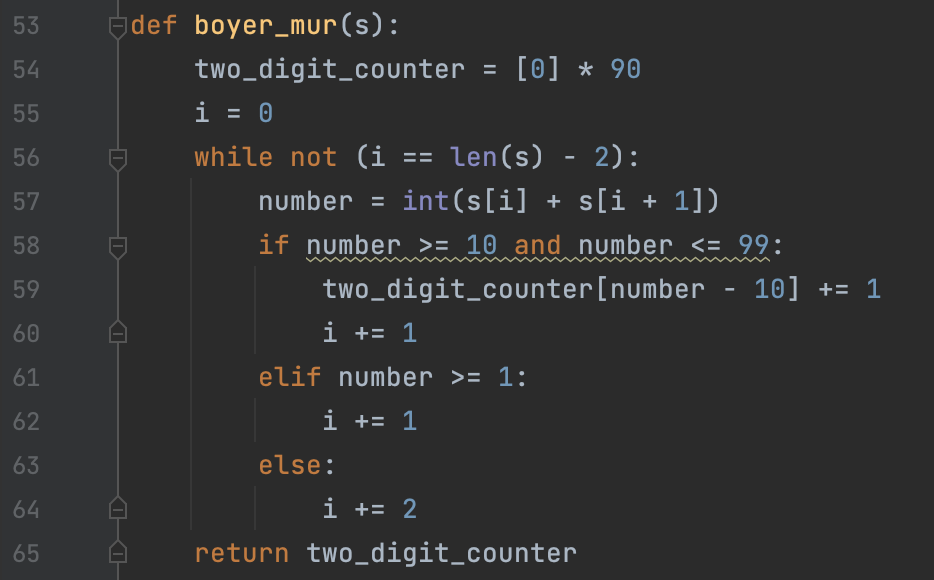


Рисунок 4 – Алгоритм поиска подстрок Бойера-Мура

На рисунке 5 приведен алгоритм поиска Кнута-Морриса-Пратта. Аналогично создаем массив, делаем проверку на ‘00’, в этом случае префикс будет равен 1, так как длина совпавшего участка 0. Числа от 10 до 99 при повторениях также отмечаем добавлением единица в массив на необходимое место. В случае больше или равно единице, префикс приравниваем 1.

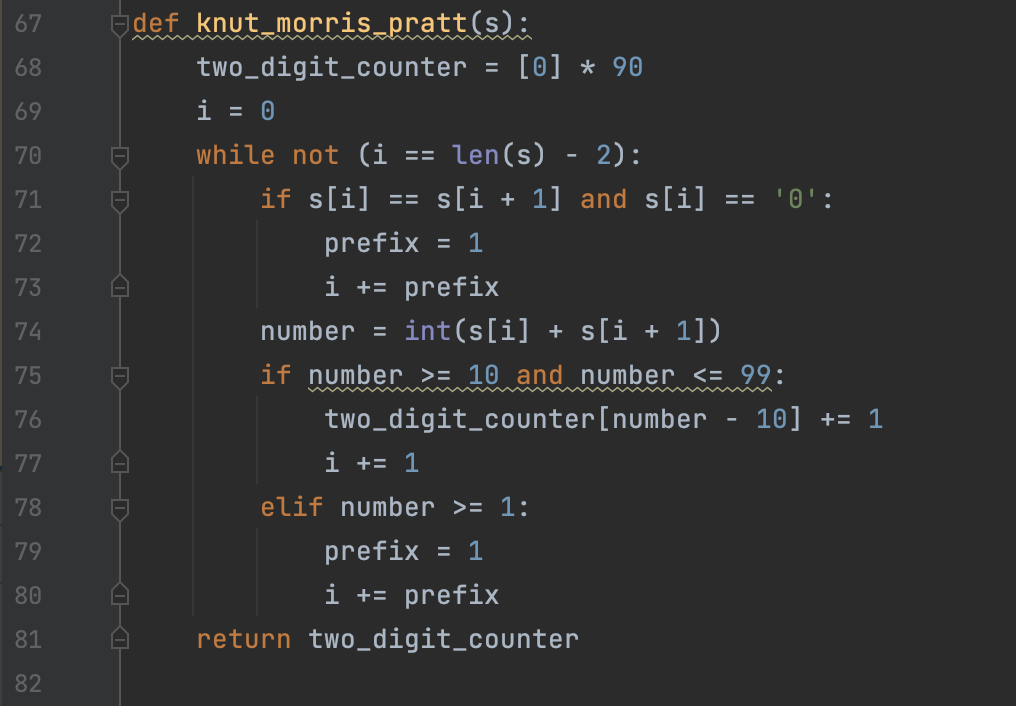


Рисунок 5 – Алгоритм Кнута-Морриса-Пратта

На рисунке 6 видно вызовы функций с выводом максимального числа повторений из массива, при использовании реализованных алгоритмов.

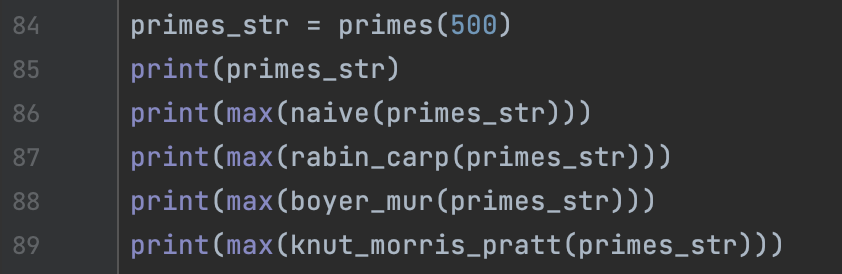


Рисунок 6 – Вывод

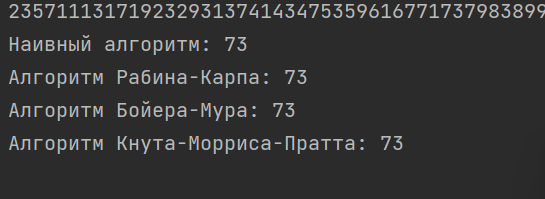
Рисунок 7 показывает вывод программы: первая строка – сам список слитно записанных простых чисел, далее – результат работы каждого алгоритма поиска подстрок (выведено максимальное число повторений).

Рисунок 7 – Результат работы программы.

Итак, первое задание выполнено.

Сравнение алгоритмов:

Наивный алгоритм:

Достоинства

• Легко реализуем

• С небольшими объёмами данных показывает наиболее эффективный результат

Недостатки

• Работает медленно при больших объёмах данных

Алгоритм Рабина-Карпа:

Достоинства

• В некоторых практических случаях работает быстрее чем наивных алгоритм

Недостатки

• Часто оказывается самым медленным алгоритмом

Алгоритм Бойера-Мура:

Достоинства

• В среднем достаточно быстр

• Быстр в работе с большими алфавитами

Недостатки

• В некоторых ситуациях оказывается достаточно медленным

Алгоритм Кнута-Морриса-Пратта:

Достоинства

• Быстро работает на больших объёмах памяти

Недостатки

• На небольших объёмах памяти может уступать другим алгоритмам

## Задание 2.

Задачей второго задания является создание программы антиплагиата. В качестве алгоритма поиска был использован наивный алгоритм.

Сначала считываем оба файла: “википедия” и “реферат”. Далее смотрим общую длину реферата и создаем 2 массива, в которых будут объединены 3 слова (в соответствии с критериями задачи о том, что плагиатом считается 3 идущих подряд одинаковых слова) (рисунок 8).

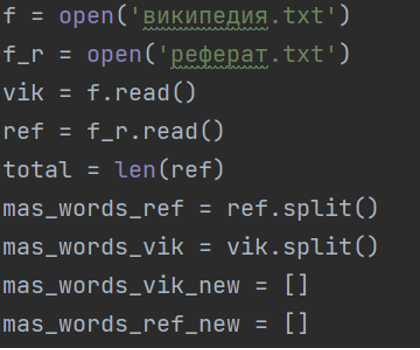


Рисунок 8 – Начало программы.

Далее, проходимся по обоим файлам последовательно и добавляем в массивы склеенные 3 слова подряд (рисунок 9).

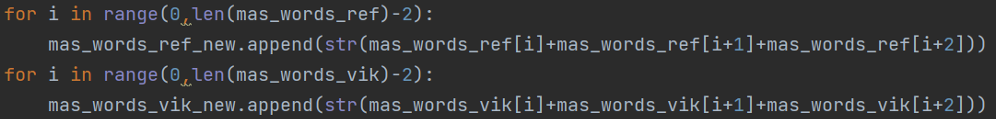


Рисунок 9 – Объединение трех слов.

В следующем цикле сравниваем элементы в каждом из двух массивов, и, если они совпадают добавляем выявление плагиата (рисунок 10). И в самом конце выводим процент плагиата.

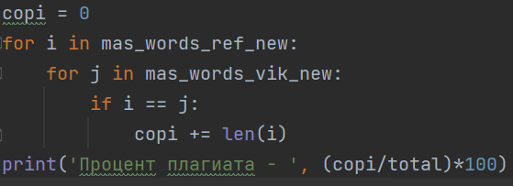


Рисунок 10 – Подсчет совпадений.

Результат работы программы (рисунок 11):

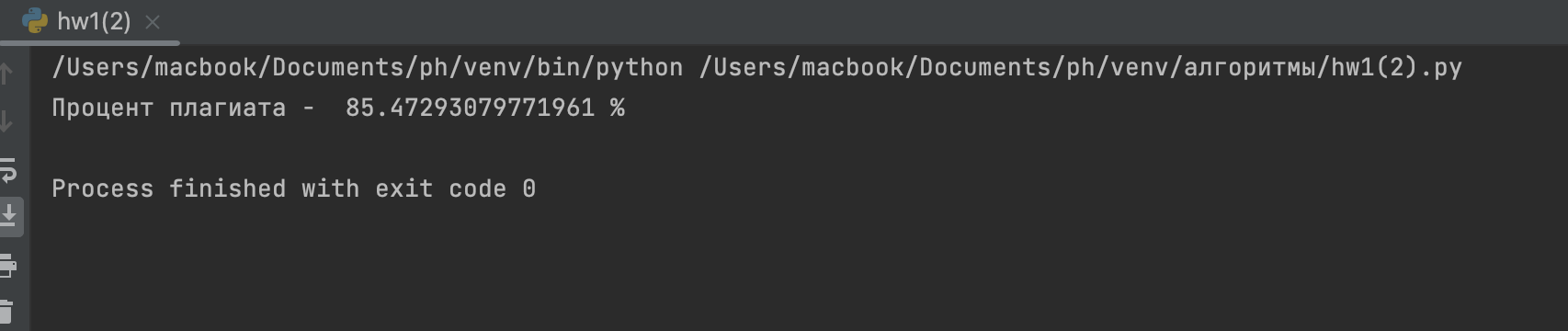


Рисунок 11 – Результат работы программы.

Таким образом цель второй задачи достигнута.

# 

# Заключение.

Были выполнены все задания, таким образом, можно считать, что цель лабораторной работы выполнена. Создана программа для нахождения количества наиболее часто встречающихся двузначных чисел в строке простых чисел. Были сравнены изученные алгоритмы поиска подстроки и сделан вывод о достоинствах и недостатках. А также была создана программа - антиплагиат. В ней были сравнены две статьи для выявления процента плагиата. Программы находятся по ссылке: <https://github.com/Anasstasssia/->

# Источники информации.

1. Python.ru[Электронный ресурс] - https://pythonru.com/biblioteki/sympy-v-python (Дата последнего обращения по ссылке: 25.02.2023)