САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №4

по курсу «Алгоритмы и структуры данных»

Тема: Тема работы

Вариант 7

Выполнил:

Крылов Михаил Максимович

К3240

Проверил:

Афанасьев А.В.

Санкт-Петербург

2023 г.

**Содержание отчета**

Оглавление

[**Содержание отчета** 2](#_Toc179020149)

[**Задачи по варианту** 3](#_Toc179020150)

[**Задача №1. Максимальная стоимость добычи [0.5 баллов]** 3](#_Toc179020151)

[**Задача №6. Максимальная зарплата [0.5 баллов]** 6](#_Toc179020152)

[**Задача №13. Сувениры [1.5 баллов]** 8](#_Toc179020153)

[**Вывод** 10](#_Toc179020154)

**Задачи по варианту**

**Задача №1. Наивный поиск подстроки в строке [2 s, 256 Mb, 1 балл]**

Даны строки p и t. Требуется найти все вхождения строки p в строку t в качестве подстроки.

• Формат ввода / входного файла (input.txt). Первая строка входного файла содержит p, вторая – t. Строки состоят из букв латинского алфавита.

• Ограничения на входные данные. 1 ≤ |p|, |t| ≤ 104 .

• Формат вывода / выходного файла (output.txt). В первой строке выведите число вхождений строки p в строку t. Во второй строке выведите в возрастающем порядке номера символов строки t, с которых начинаются вхождения p. Символы нумеруются с единицы.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб

Листинг кода

def func(substring, string):

    result = []

    for i in range(len(string)):

        new\_substring = string[i:len(substring)+i]

        if new\_substring == substring:

            result.append(i+1)

    return str(len(result)), ' '.join(map(str, result))

def main():

    with open('input.txt') as file:

        substring = file.readline().strip()

        string = file.readline().strip()

    result = func(substring, string)

    with open('output.txt', 'w') as file:

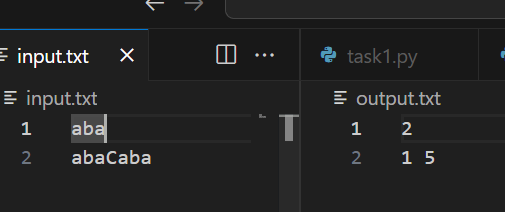
        file.write(result[0]+'\n')

        file.write(result[1])

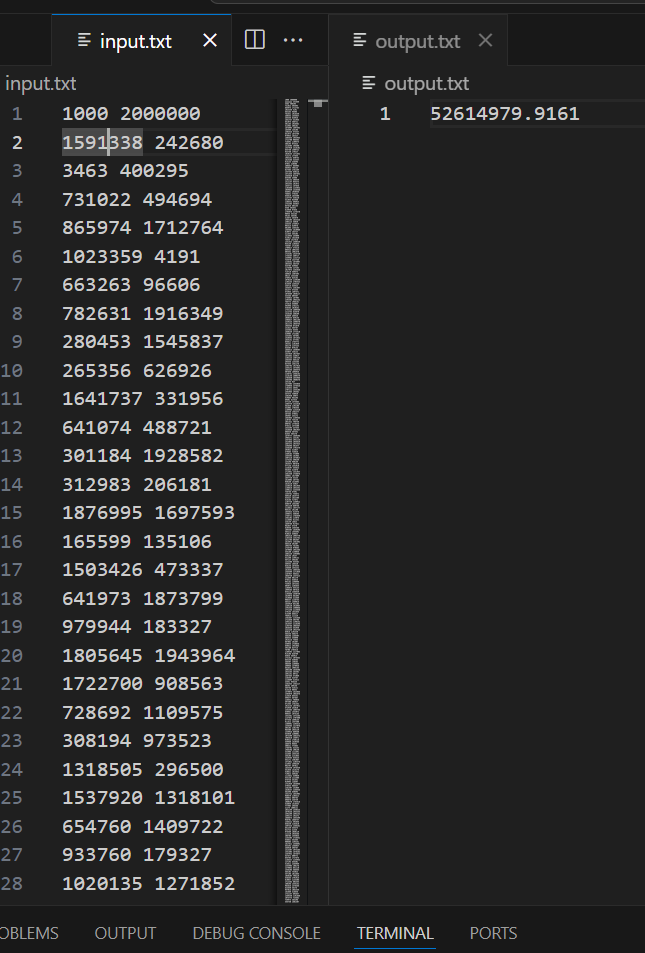
if \_\_name\_\_ == "\_\_main\_\_":

    main()

Считываем данные с файла. Далее запускаем основную функцию. С помощью среза строк. Все вхождения подстроки в строку записываем в список, который потом выводим.



Результат работы кода на максимальных и минимальных значениях:



|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.007 секунды | 0.017 МБ |
| Пример из задачи | 0.008 секунды | 0.018 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.054 секунды | 0.050 МБ |

Вывод по задаче: закрепили работу со строками.

**Задача №3. Паттерн в тексте [2 s, 256 Mb, 1 балл]**

В этой задаче ваша цель – реализовать алгоритм Рабина-Карпа для поиска заданного шаблона (паттерна) в заданном тексте.

• Формат ввода / входного файла (input.txt). На входе две строки: паттерн P и текст T. Требуется найти все вхождения строки P в строку T в качестве подстроки.

• Ограничения на входные данные. 1 ≤ |P|, |T| ≤ 106 . Паттерн и текст содержат только латинские буквы.

• Формат вывода / выходного файла (output.txt). В первой строке выведите число вхождений строки P в строку T. Во второй строке выведите в возрастающем порядке номера символов строки T, с которых начинаются вхождения P. Символы нумеруются с единицы.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб

Листинг кода

def func(substring, string):

    result = []

    for i in range(len(string)-len(substring)+1):

        for j in range(len(substring)):

            if string[i+j] != substring[j]:

                break

        else:

            result.append(i+1)

    return str(len(result)), ' '.join(map(str, result))

def main():

    with open('input.txt') as file:

        substring = file.readline().strip()

        string = file.readline().strip()

    result = func(substring, string)

    with open('output.txt', 'w') as file:

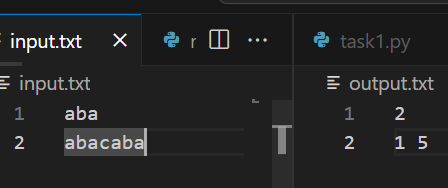
        file.write(result[0]+'\n')

        file.write(result[1])

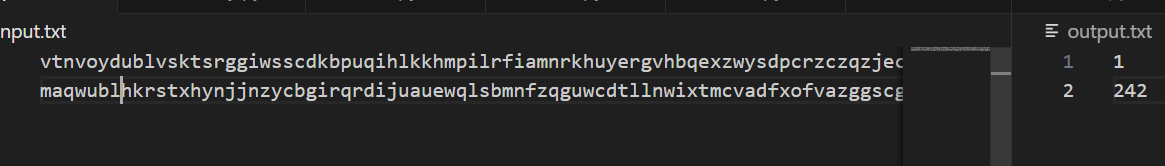
if \_\_name\_\_ == "\_\_main\_\_":

    main()

Считываем данные с файла. Далее запускаем основную функцию, которая с помощью алгоритма Рабина-Карпа находит подстроки в строке. Когда подстрока найдена – алгортм записываем её нндекс+1 в список. После чего выводим всё в текстовый файл.



Результат работы кода на максимальных и минимальных значениях:



|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.003 секунды | 0.018 МБ |
| Пример из задачи | 0.003 секунды | 0.018 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.055 секунды | 0.038 МБ |

Вывод по задаче: реализовали алгоритм поиска подстроки с помощью алгоритма Рабина-Карпа.

**Задача №6. Z-функция [2 s, 256 Mb, 1.5 балла]**

Постройте Z-функцию для заданной строки s.

• Формат ввода / входного файла (input.txt). Одна строка входного файла содержит s. Строка состоит из букв латинского алфавита.

• Ограничения на входные данные. 2 ≤ |s| ≤ 106 .

• Формат вывода / выходного файла (output.txt). Выведите значения Z-функции для всех индексов 1, 2, ..., |s| строки s, в указанном порядке.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

Листинг кода

def func(string):

    n = len(string)

    Z = [0] \* n

    for i in range(n):

        for j in range(i, n):

            if string[j] == string[j - i]:

                Z[i] += 1

            else:

                break

    return Z

def main():

    with open('input.txt') as file:

        string = file.readline().strip()

    result = func(string)

    with open('output.txt', 'w') as file:

        file.write(' '.join(map(str, result)))

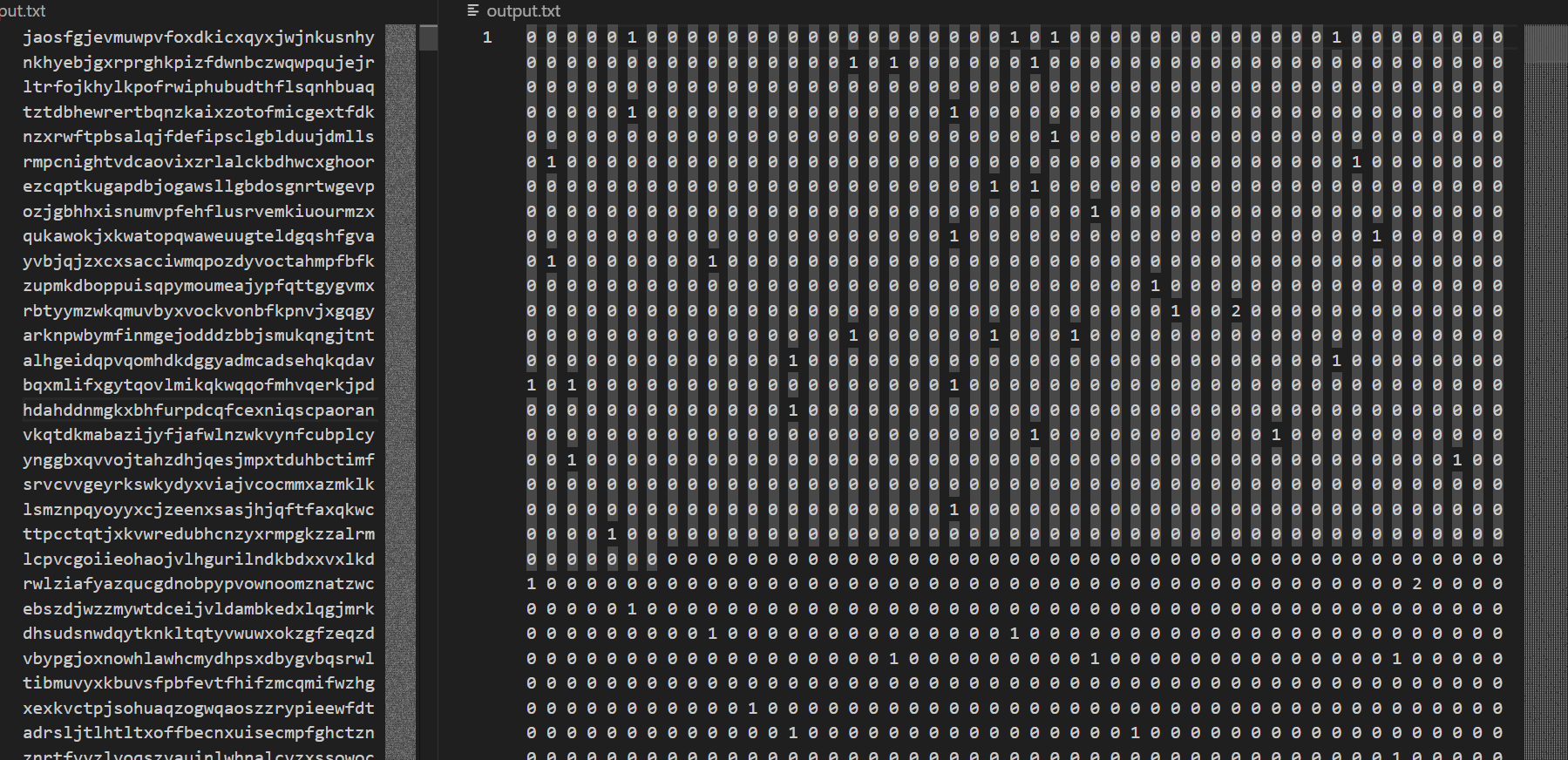
if \_\_name\_\_ == "\_\_main\_\_":

    main()

Считываем данные. Запускаем основную функцию. С помощью вложенной функции проверяем сколько элементов строки содержат подстроку. Эти данные записываем в массив Z по индексу. Выводим результат в файл.



Результат работы кода на максимальных и минимальных значениях:



|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.055 секунды | 0.018 МБ |
| Пример из задачи | 0.055 секунды | 0.018 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 1.356 секунды | 42.986 МБ |

Вывод по задаче: вспомнили как решать задачи на динамическое программирование через рекурсивные функции.

**Вывод**

Данная лабораторная работа напомнила и закрепила работу со строками.