МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Поиск образца в тексте. Алгоритм Рабина-Карпа

Студент гр. 0304 _____ Докучаев Р.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Реализовать поиск фрагмента, который ввёл пользователь, в вводимой пользователем строки при помощи алгоритма Рабина-Карпа.

Задание.

Напишите программу, которая ищет все вхождения строки Pattern в строку Text, используя алгоритм Карпа-Рабина.

На вход программе подается подстрока Pattern и текст Text. Необходимо вывести индексы вхождений строки Pattern в строку Text в возрастающем порядке, используя индексацию с нуля.

Ограничения

$1 \leq |Pattern| \leq |Text| \leq 5 \cdot 10^5$.

Суммарная длина всех вхождений образца в текст не превосходит 10^8. Обе строки содержат только буквы латинского алфавита.

Основные теоретические положения.

Были использованы стандартные возможности языка Python: функции, классы, циклы. Также была использована утилита pytest для тестирования работы программы.

Алгоритм Карпа – Рабина это алгоритм поиска строки создан Ричард М. Карп и. Майкл О. Рабин (1987) который использует хеширование чтобы найти точное совпадение строки шаблона в тексте. Он использует скользящий хеш чтобы быстро отфильтровать позиции текста, которые не могут соответствовать шаблону, а затем проверяет соответствие в оставшихся позициях.

Выполнение работы.

1. Был реализована функция hash_str(sentence, power, mod), которая принимает на вход строку, список степеней и остаток, при делении на который и будет результат работы функции. Функция вычисляет полиномиальный хеш строки с помощью полученных данных
2. Была реализована функция sliding_hash(first, last, prev, power, mod, x), которая вычисляет хеш строки на основе предыдущего .
3. В основном модуле программы будет произведено считывание строки и искомой подстроки с консоли. Затем формируется список степеней, после чего подсчитывается первое хеш значение подстроки, которое также для удобства дальнейшего поиска было сохранено в отдельную переменную. Затем происходит вывод результатов поиска в виде индекса первого элемента искомой подстроки в строке.
4. Был создан тестовый модель test.py, который на примерах проверяет корректность выполнения поиска подстроки в строке.

Исходный код программы представлен в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	‘ ‘, ‘asd’		OK
2.	‘no’, ‘yes’		OK
3.	‘a’, ‘abcda’	0, 4	
4.	‘roman’, ‘roman’	0	
5.	‘111’, ‘1111’	0, 1	
6.	‘dr’, ‘0123dr’	4	

Выводы.

Был изучен алгоритм Рабина-Карпа, а также реализована функция, реализующая поиск подстроки в строке вводимой пользователем при помощи этого алгоритма

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: laba4.py

```
def hash_str(sent, power, mod):
    result = 0
    index = 0
    for letter in sent:
        result += ord(letter) * power[index]
        index += 1
    return result % mod


def sliding_hash(first, last, prev, power, mod, x):
    return ((prev - first * power[0]) * x + last) % mod


if __name__ == '__main__':
    a = 11
    b = 2 ** 31 - 1
    pattern = input()
    text = input()

    powers = [a ** i for i in range(len(pattern))][::-1]

    pattern_hash = hash_str(pattern, powers, b)
    len_pattern = len(pattern)
    temp = text[0:len_pattern]
    prev = hash_str(text[0:len_pattern], powers, b)
    if prev == pattern_hash:
        if temp == pattern:
            print(0, end=' ')
    for ind in range(len(text) - len_pattern):
        temp = temp[1:] + text[ind + len_pattern]
        new_hash = sliding_hash(ord(text[ind]), ord(text[ind + 
len_pattern]), prev, powers, b, a)
        prev = new_hash
        if pattern_hash == new_hash:
            if temp == pattern:
                print(ind + 1, end=' ')
```