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.

Студентка гр. 1304 _____ Чернякова В.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2022

Цель работы.

Освоить работу с хеш-таблицами. Реализовать алгоритм Рабина-Карпина, основываясь на знаниях о хэш-таблицах.

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.

Примечание: в работе запрещено использовать библиотечные реализации алгоритмов и структур.

Ограничения

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

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

Пример.

Вход:

aba

abacaba

Выход:

0 4

Подсказки:

1. Будьте осторожны с операцией взятия подстроки — она может оказаться дорогой по времени и по памяти.

2. Храните степени x ** p в списке - тогда вам не придется вычислять их каждый раз заново.

Выполнение работы.

На вход программе с помощью функции *input()* подается 2 строки: подстрока и текст, котором нужно найти вхождения введённой подстроки.

Полученный результат работы функции, осуществляющей реализацию работы алгоритма Рабина-Карпа, выводится на экран.

Функции.

Функция *def hasing(string).* Принимает на вход строку и считает ее хэш. Полиномиальный хэш рассчитывается по формуле:

$$hash(p[1..m]) = \left(\sum_{i=1}^{m} p[i]x^{m-i}\right) \bmod q,$$

Для корректного подсчета добавляется код символа: *hashing_value = (x * hashing_value + ord(string[k])) % prime*. Значение берется по модулю *prime* – простое число.

Функция *def algorithmRabinKarp(pattern, text)*. Функция принимает на вход подстроку и текст, в котором необходимо найти ее вхождения. В функции реализован алгоритм Рабина-Карпа. Переменные *pattern_hash* – хэш введенной подстроки, *win_hash* – хеш в тексте по срезу от начала строки до длины введенной подстроки.

В основном цикле алгоритма сравниваются два полученных хэша. Если они равны, то сравнивается введенная строка и срез текста *pattern == text[j: j + pl]*. И если это условие тоже выполняется, тогда в список результатов добавляется индекс, с которого начинается вхождение подстроки *result.append(j).* Для дальнейшего поиска вхождений вначале проверяется не произойдёт ли выход за границы текста, если нет, то хэш пересчитывается по формуле:

$$hash(s[i+1..i+m]) = \left((hash(s[i..i+m-1]) - s[i] \cdot x^{m-1}) \cdot x + s[i+m]\right) \bmod q.$$

Если же значение хэша было отрицательным, то к нему просто прибавляется простое число. Алгоритм работает дальше.

Согласно данной формуле x^{m-1}(в коде программы это переменная *h*) было рассчитано заранее для оптимизации *for _ in range(pl - 1): h = (h * x) % prime.*

Функция возвращает список с индексами вхождений подстроки.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	aba abacaba	0 4	Проверка работы алгоритма для стандартного случая.
2.	cde aacdecdecdepppcde	2 5 8 14	Проверка работы алгоритма для стандартного случая.
3.	пти На ферме большой птичий двор. На дворе гуляют гуси и гусята, утки и утята, куры и цыплята. Птиц кормит птичница бабушка Настя. Ей помогают Таня и Катя. Они кормят гусят, утят и цыплят.	17 103	Проверка работы алгоритма для небольшого текста на русском языке.
4.	cdb opoaserdjsnc	Вхождения не найдены!	Проверка работы алгоритма для случая, когда подстроки нет в тексте.
5.	Hello hi	Длина подстроки превышает текст	Проверка работы алгоритма для случая, когда длина строки больше текста.

Выводы.

Была изучена хэш-функция, с помощью которой реализован алгоритм Рабина-Карпа. Корректность работы проверена с помощью тестов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
prime = 101
x = 128


def hashing(string):
    hashing_value = 0
    for k in range(len(string)):
        hashing_value = (x * hashing_value + ord(string[k])) %
prime
    return hashing_value


def algorithm_rabin_karp(pattern, text):
    result = list()
    pl = len(pattern)
    tl = len(text)
    h = 1

    if pl > tl:
        print('Длина подстроки превышает текст')
        return -1

    if pl == 0 or tl == 0:
        print('Введены пустые строки')
        return -1

    pattern_hash = hashing(pattern)
    win_hash = hashing(text[:pl])

    for _ in range(pl - 1):
        h = (h * x) % prime

    for j in range(0, tl - pl + 1):
```

```
            if pattern_hash == win_hash:
                if pattern == text[j: j + pl]:
                    result.append(j)

            if j + pl < tl:
                win_hash = (x*(win_hash-ord(text[j])*h) + ord(text[j +
pl])) % prime

                if win_hash < 0:
                    win_hash += prime

        return result


    if __name__ == '__main__':
        Pattern = input()
        Text = input()
        answer = algorithm_rabin_karp(Pattern, Text)
        if answer != -1:
            if len(answer) != 0:
                for i in range(len(answer)):
                    print(answer[i], end=' ')
            else:
                print('Вхождения не найдены!')
```

Название файла: test.py

```
from main import algorithm_rabin_karp
import pytest


def test1():
    Pattern = 'aba'
    Text = 'abacaba'
    assert algorithm_rabin_karp(Pattern, Text) == [0, 4]


def test2():
    Pattern = 'cde'
    Text = 'aacdecdecdepppcde'
```

```
        assert algorithm_rabin_karp(Pattern, Text) == [2, 5, 8, 14]


    def test3():
        Pattern = 'пти'
        Text = 'На ферме большой птичий двор. На дворе гуляют гуси и гусята, утки и утята, куры и цыплята. Птиц кормит птичница бабушка Настя. Ей помогают Таня и Катя. Они кормят гусят, утят и цыплят.'
        assert algorithm_rabin_karp(Pattern, Text) == [17, 103]


    def test4():
        Pattern = 'cdb'
        Text = 'opoaserdjsnc'
        assert algorithm_rabin_karp(Pattern, Text) == []


    def test5():
        Pattern = 'hello'
        Text = 'hi'
        assert algorithm_rabin_karp(Pattern, Text) == -1
```