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. 1304 _____ Поршнев Р.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2022

Цель работы.

Изучить хеш-таблицы в алгоритме Рабина-Карпа для нахождения образца в тексте.

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 108. Обе строки содержат только буквы латинского алфавита.

Выполнение работы.

В *main* происходит считывание образца в переменную *pattern* и строки, в которой нужно найти образец *pattern*, — в переменную *text*. Далее создаётся экземпляр класса *RabinKarp*, в конструктор которого передаётся подстрока и строка, считанные ранее. Затем вызывается метод *hashing* класса *RabinKarp*, который высчитывает значение хеш-функции для каждого окна в исходной строке. Последним шагом в *main* выводится ответ на исходную задачу.

Класс *RabinKarp* имеет следующие методы:

1) *get_start_hash_value(self)* – данный метод высчитывает начальное значение хеш-функции от окна размером в подстроку, которую нужно искать в исходной строке. Хеш-функция имеет следующий вид: $\sum_{i=L-l}^{L-1} t_i * x^{i-(L-l)} \ mod \ p$, где L – длина строки, l – длина подстроки, t_i – код i-го символа исходной строки, x – произвольное значение, меньшее p.

2) *get_hash_pattern(self)* – данный метод высчитывает значение хеш-функции от образца. Хеш-функция имеет следующий вид: $\sum_{i=0}^{l-1} t_i * x^i$, где l – длина подстроки, t_i – код i-го символа исходной подстроки, x – произвольное значение, меньшее p.
3) *is_match(self, start)* – данный метод посимвольно сравнивает окно исходной строки, равное длине образца, с подстрокой непосредственно, начиная с индекса *start*. Если все символы совпали, то номер элемента (*start*), начиная с которого прошло совпадение, добавляется в список ответов.
4) *hashing(self)* – данный метод высчитывает значение хеш-функции от каждого окна исходной строки, длиной равной абзацу. Обход окон производится справа налево по исходной строке. Если на текущем окне значение хеш-функции равно значению хеш-функции образца, то вызывается метод *is_match(self, start)*, в который в качестве аргумента передаётся начало текущего окна.
5) *get_ans(self)* – данный метод сортирует массив из индексов, начиная с которых образец входит в строку, а затем список приводит к строке и возвращает её.

Также в данном классе присутствует конструктор, в котором происходит инициализация параметра p, а также параметра x, который должен быть меньше p. Происходит инициализации строки и образца, которые были переданы в качестве аргументов, а также высчитывается значение функции x^i, где $i = 0 \ldots l - 1$, где l – длина строки образца, и заносится в список *pows*.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	drain drain gang	0	Ответ правильный, тестирование одного

			вхождения, при котором подстрока входит в исходную строку с 0-го индекса
2.	ecco2k bladee ecco2k whitearmor	7	Ответ правильный, тестирование одного вхождения, при котором подстрока находится в середине строки
3.	thaiboy digital i'm fresh - thaiboy digi-tal	12	Ответ правильный, тестирование одного вхождения, при котором подстрока находится в конце исходной строки
4.	aba abacaba	0 4	Ответ правильный, тестирование примера из задания
5.	ab abababab	0 2 4 6	Ответ правильный, тестирование случая, при котором исходная строка состоит только из образцов
6.	ab agsdfksahfaks		Ответ правильный, тестирование случая, при котором образец не входит в строку

Вывод.

В ходе выполнения лабораторной работы были изучены хеш-функции и хеш-таблицы, а также изучен принцип работы алгоритма Рабина-Карпа.

Реализована программа, ищущая индексы вхождений образца в исходную строку по алгоритму Рабина-Карпа.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class RabinKarp:
    def __init__(self, str, substr):
        self.x = 263
        self.p = 10000007
        self.str = str
        self.substr = substr
        self.pows = [pow(self.x, i, self.p) for i in
range(len(self.substr))]
        self.ans = []

    def get_start_hash_value(self):
        start_hash = 0
        for i in range(len(self.str) - len(self.substr),
len(self.str), 1):
            start_hash += (ord(self.str[i]) * self.pows[i -
(len(self.str) - len(self.substr))]) % self.p
        return start_hash % self.p

    def get_hash_pattern(self):
        hash_pattern = 0
        for i in range(len(self.substr)):
            hash_pattern += (ord(self.substr[i]) * self.pows[i]) %
self.p
        return hash_pattern % self.p

    def is_match(self, start):
        flag = True
        for j in range(len(self.substr)):
            if self.substr[j] != self.str[start + j]:
                flag = False
                break
        if flag:
            self.ans.append(start)

    def hashing(self):
        hash_prev = self.get_start_hash_value()
```

```
            hash_pattern = self.get_hash_pattern()
            if hash_prev == hash_pattern:
                self.is_match(len(self.str) - len(self.substr))
            for i in range(len(self.str) - len(self.substr) - 1, -1, -
1):
                flag = True
                hash_current = ((hash_prev - ord(self.str[i +
len(self.substr)]) * self.pows[
                    len(self.pows) - 1]) * self.x + ord(self.str[i])) %
self.p
                if hash_current == hash_pattern:
                    self.is_match(i)
                hash_prev = hash_current

        def get_ans(self):
            self.ans.sort()
            return ' '.join(map(str, self.ans))


    if __name__ == "__main__":
        pattern = input()
        text = input()
        rk = RabinKarp(text, pattern)
        rk.hashing()
    print(rk.get_ans())
```

Название файла: tests.py

```
    from main import RabinKarp


    def test_1():
        rk = RabinKarp("drain gang", "drain")
        rk.hashing()
        assert rk.get_ans() == "0"


    def test_2():
        rk = RabinKarp("bladee ecco2k whitearmor", "ecco2k")
        rk.hashing()
```

```
        assert rk.get_ans() == "7"


    def test_3():
        rk = RabinKarp("i'm fresh - thaiboy digital", "thaiboy digital")
        rk.hashing()
        assert rk.get_ans() == "12"


    def test_4():
        rk = RabinKarp("abacaba", "aba")
        rk.hashing()
        assert rk.get_ans() == "0 4"


    def test_5():
        rk = RabinKarp("abababab", "ab")
        rk.hashing()
        assert rk.get_ans() == "0 2 4 6"


    def test_6():
        rk = RabinKarp("agsdfksahfaks", "ab")
        rk.hashing()
    assert rk.get_ans() == ""
```