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. 1384 _____ Усачева Д.В.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург2022

Цель работы.

Научиться работе с хэш-функцией, с её помощью решить алгоритмическую задачу.

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 тексте не превосходит 10^8. Обе строки содержат только буквы латинского алфавита.

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.

Первой строкой добавьте #python или #c++, чтобы проверяющая система знала, каким языком вы пользуетесь.

Выполнение работы.

Для решения задачи был применён алгоритм Рабина-Карпа. Суть данного алгоритма заключается в хеширование строк и сравнении полученных

значений. Хэш-значение вычисляется по формуле $\Sigma_{i=1}^{n} \left(ord(str[i]) * x^{i}\right) mod\ p$.

Алгоритм начинается с хеширования искомой строки и подстроки длины искомой строки из исследуемой строки. Программа заходит в цикл, который проходит по всем подстрокам исследуемой строки, смещаясь каждый виток на один символ. Если хэши совпадают, в ответ записывается индекс начала подстроки в тексте. Далее берётся подстрока, сдвинутая на один символ относительно старой подстроки в исследуемой строке. Считается хэш такой подстроки и цикл повторяется.

Тестирование.

Чтобы удостовериться в правильности работы программы, она была протестирована на следующих случаях:

1. В тексте не встречается требуемой подстроки
2. Подстрока находится в начале текста
3. Подстрока находится на конце текста
4. Подстрока находится на конце текста
5. Подстрока является одной буквой
6. Подстрока длиннее текста

Выводы.

В ходе выполнения лабораторной работы был изучен алгоритм Рабина-Карпа, были применены навыки работы с хэш-функцией. Была успешно решена поставленная задача

ПРИЛОЖЕНИЕ 1
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
#python
result = []
x = 241
p = 1000000007


def get_hash(string):
    st = 0
    for j in range(len(string)):
        cur = string[j]
        res = result[j]
        st += ord(cur) * res
    return st


def RabinKarp(pattern, text):
    answer = []
    txt_len = len(text)
    pat_len = len(pattern)
    if txt_len > 0 and pat_len > 0:
        for j in range(0, len(pattern)):
            result.append(x ** j % p)
        hashPattern = get_hash(pattern)
        for j in range(0, txt_len - pat_len + 1):
            hashText = get_hash(text[j:j + pat_len])
            if hashText == hashPattern:
                answer.append(j)
    return answer


if __name__ == '__main__':
    pattern = input()
```

```
    text = input()
    print(' '.join(list(map(str, RabinKarp(pattern, text)))))
```

Название файла: tests.py

```
from main import *

def test_no_entry():
    assert RabinKarp('ex', 'testnoentry') == []

def test_prefix():
    assert RabinKarp('test', 'testprefix') == [0]

def test_suffix():
    assert RabinKarp('suffix', 'testsuffix') == [4]

def test_center():
    assert RabinKarp('AA', 'testAAcenter') == [4]

def test_letters():
    assert RabinKarp('a', 'aaaaAa') == [0,1,2,3,5]

def test_pattern_longer_than_text():
    assert RabinKarp('heHEhehehehhe', 'he') == []
```