**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. 1303 |  | Чубан Д.В. |
| --- | --- | --- |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2022

* 1. **Цель работы.**

Изучение хеширования данных, реализация алгоритма Рабина-Карпа, основанного на хешировании, для эффективного поиска заданной подстроки в строке.

* 1. **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.

* 1. **Выполнение работы.**

Для решения данной задачи были реализованы следующие функции*:*

* *hashing(str, k) –* принимая на вход список множителей и строку, возвращает хеш-значение
* *substringRK(text, pattern, coefs) –* функция, реализующая алгоритм Карпа-Рабина, а именно ищет все вхождения подстроки *pattern* в строку text.

В функции *main()* считываются строка-шаблон и строка, в которой осуществляется поиск. Число X положим равным мощности алфавита введенной строки. В *main()* производится вычисление списка множителей вида {x^i | i = 0,1..m-1}.

В функции *hashing(str, k)* производится вычисление хеш-функции следующего вида:

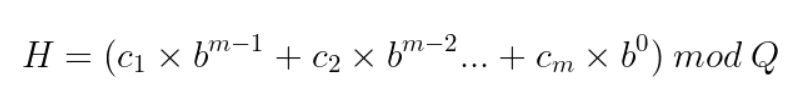


Рисунок 1 – Формула хеш-функции

Где – символы в строке, – элементы списка множителей, полученного в *main().*

В функции *substringRK(text, pattern, coefs)* сначала копируется подстрока из *text* длины подстроки *pattern.* Для *pattern* и *substr* вычисляется хеш. Затем в цикле сравниваются полученные значения, в случае совпадения – в список *res[]* сохраняется индекс вхождения подстроки. Затем, к *substr* добавляется очередной символ из *text* и хеш пересчитывается. Можно считать, что в *substringRK(text, pattern, coefs)* используется скользящая хеш-функция, т.к. значение её вычисляется по формуле:

Где *Hp –* предыдущее значение хеша, *Cp –* символ, который необходимо удалить из подстроки, *Cn –* символ, который добавили в строку.

Данная реализация позволяет избежать затрат ресурсов и времени на взятие подстрок на каждом шаге работы данного алгоритма.

**Тестирование программы.**

Тестирование функции производится с помощью unit-тестов, описанных в файле *test.py.* Данные тесты покрывают следующие случаи:

* На вход подается строка из одинаковых символов и шаблон из тех же символов, но меньшей длины
* На вход подается строка и шаблон, входящий в подстроку несколько раз.
* На вход подается строка и шаблон, не входящий в подстроку.
* Строка и шаблон совпадают

**Вывод.**

Была освоена работа с хеш-функциями. Реализована программа, основанная на алгоритме Рабина-Карпа, находящая вхождения подстроки в строку.

1. **Приложение А  
   Исходный код программы**

Название файла: main.py

def hashing(str, k):

res = 0

for i in range(len(str)):

res += ord(str[i]) \* k[len(str) - i - 1]

return res

def substringRK(text, pattern, coefs):

res = []

pat\_len = len(pattern)

substr = text[:pat\_len]

patt\_hash = hashing(pattern, coefs)

substr\_hash = hashing(substr, coefs)

for i in range(pat\_len, len(text) + 1):

if patt\_hash == substr\_hash:

res.append(i - pat\_len)

if i < len(text):

substr += text[i]

substr\_hash = (substr\_hash - ord(substr[i - pat\_len]) \* coefs[pat\_len - 1]) \* coefs[1] + ord(text[i])

return res

def main():

pattern = input()

str = input()

m = len(set(str))

coefs = [m \*\* i for i in range(len(pattern))]

res = substringRK(str, pattern, coefs)

print(\*res)

main()

1. **Приложение Б  
   Исходный код UNIT-ТЕСТОВ**

Название файла: test.py

import pytest

from substringRK import \*

def test\_screaming():

alph = len(set('аааааа'))

coefs = [alph \*\* i for i in range(len('aaa'))]

res = substringRK('aaaaaa', 'aaa', coefs)

assert res == [0, 1, 2, 3]

def test\_common():

alph = len(set('аbacaba'))

coefs = [alph \*\* i for i in range(len('aba'))]

res = substringRK('abacaba', 'aba', coefs)

assert res == [0, 4]

def test\_empty():

alph = len(set('аbacaba'))

coefs = [alph \*\* i for i in range(len('aaa'))]

res = substringRK('abacaba', 'aaa', coefs)

assert res == []

def test\_equal():

alph = len(set('аbacaba'))

coefs = [alph \*\* i for i in range(len('аbacaba'))]

res = substringRK('аbacaba', 'аbacaba', coefs)

assert res == [0]