# 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. 2384 | Кузьминых Е.М |
| Преподаватель | Иванов Д.В. |

Санкт-Петербург 2023

# Цель работы.

Реализовать алгоритм Рабина-Карпа, использовать его в программе, ищущей вхождения подстроки в строку.

# Задачи.

# 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 ≤ |Pattern| ≤ |Text| ≤ 5 · 105

# Суммарная длина всех вхождений образца в текста не превосходит 108. 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.

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

# Выполнение работы.

Код представляет собой реализацию алгоритма Рабина-Карпа для поиска подстроки в тексте. Алгоритм Рабина-Карпа - это эффективный алгоритм для поиска подстроки в тексте, который использует хеширование для ускорения процесса.

Функция rabin\_karp\_search принимает два аргумента: pattern и text. pattern - это подстрока, которую мы хотим найти в text. base (d) - это количество символов в алфавите, в данном случае 256 для расширенного ASCII. prime (q) - это простое число, используемое для вычисления хеш-значений. pattern\_length (M) и text\_length (N) - это длины pattern и text соответственно. pattern\_hash (p) и text\_hash (t) - это хеш-значения для pattern и text. hash\_value (h) используется для вычисления хеш-значений.

Сначала вычисляются хеш-значения для pattern и первого окна text. Затем алгоритм "скользит" по text, сравнивая хеш-значения pattern и текущего окна text. Если хеш-значения совпадают, алгоритм проверяет каждый символ по отдельности. Если все символы совпадают, индекс начала окна добавляется в список indices.

В конце вычисляется хеш-значение для следующего окна text. Если полученное значение text\_hash отрицательное, оно преобразуется в положительное. Функция возвращает список indices, содержащий индексы начала каждого вхождения pattern в text.

После мы получаем от пользователя два входных значения: шаблон для поиска и текст, в котором будет производиться поиск. Затем мы вызываем функцию rabin\_karp\_search с этими входными значениями и выводим результат.

Алгоритм Рабина-Карпа является эффективным способом поиска подстроки в тексте, особенно когда текст и шаблон достаточно большие. Он использует хеширование для ускорения процесса поиска, что делает его более эффективным по сравнению с простым алгоритмом поиска подстроки. Однако, как и любой алгоритм, он имеет свои ограничения. Например, он может дать ложные срабатывания, если простое число, используемое для вычисления хеш-значений, недостаточно большое.

# Тестирование.

Тестирование работоспособности написанного кода проводилось с помощью pytest

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Комментарий |
| 1 | aba  abacaba | 0 4 | Ответ верный |
| 2 | row  myrowcrowcrowded | 2 6 10 | Ответ верный |
| 3 | a  bababafkagafmgfakgmaadsaads | 1 3 5 8 10 15 19 20 23 24 | Ответ верный |

Среднее время работы программы (проводилось тестирование с случайными строками, количество тестов – 1000) : 0.00160923470450957 сек.

**Вывод.**

В ходе выполнения лабораторной работы был реализован алгоритм Рабина-Карпа, написана программа, использующая этот алгоритм и вычисляющая вхождения подстроки в строку.

# Приложение А.

**Исходный код программы.**

**Main.py**

def rabin\_karp\_search(pattern, text):

base = 256 # d

prime = 101 # q

pattern\_length = len(pattern) # M

text\_length = len(text) # N

pattern\_hash = 0 # p

text\_hash = 0 # t

hash\_value = 1 # h

for i in range(pattern\_length - 1):

hash\_value = (hash\_value \* base) % prime

for i in range(pattern\_length):

pattern\_hash = (base \* pattern\_hash + ord(pattern[i])) % prime

text\_hash = (base \* text\_hash + ord(text[i])) % prime

indices = []

for i in range(text\_length - pattern\_length + 1):

if pattern\_hash == text\_hash:

for j in range(pattern\_length):

if text[i + j] != pattern[j]:

break

else:

indices.append(i)

if i < text\_length - pattern\_length:

text\_hash = (base \* (text\_hash - ord(text[i]) \* hash\_value) + ord(text[i + pattern\_length])) % prime

if text\_hash < 0:

text\_hash = (text\_hash + prime)

return indices

search\_pattern = input()

search\_text = input()

print(' '.join(map(str, rabin\_karp\_search(search\_pattern, search\_text))))

**Tests.py**

from main import rabin\_karp\_search

def test\_rabin\_karp\_search\_no\_match():

pattern = "abc"

text = "def"

assert rabin\_karp\_search(pattern, text) == []

def test\_rabin\_karp\_search\_one\_match():

pattern = "abc"

text = "abcdef"

assert rabin\_karp\_search(pattern, text) == [0]

def test\_rabin\_karp\_search\_multiple\_matches():

pattern = "abc"

text = "abcabcabc"

assert rabin\_karp\_search(pattern, text) == [0, 3, 6]

def test\_rabin\_karp\_search\_overlapping\_matches():

pattern = "abc"

text = "abcabcabc"

assert rabin\_karp\_search(pattern, text) == [0, 3, 6]