МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе № 4
по дисциплине «Построение и Анализ Алгоритмов»
Тема: «Поиск подстроки в строке»

Студент гр. 3343 Жучков О.Д.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2025

Цель работы

Изучить принцип работы алгоритма Кнута-Морриса-Пратта (КМП). Написать программу, которая:

1) Находит поиск индексов вхождений подстроки в строку.

Определить, являются ли строки циклическим сдвигом друг друга, найти первый индекс начала вхождения второй строки в первую.

Задание

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P ($|P| \leq 15000$) и текста T ($|T| \leq 5000000$) найдите все вхождения P в T.

Вход:

Первая строка - P

Вторая строка - T

Выход:

индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести -1

Sample Input:

ab

abab

Sample Output:

0,2

Заданы две строки A ($|A| \leq 5000000$) и B ($|B| \leq 5000000$).

Определить, является ли A циклическим сдвигом B (это значит, что A и B имеют одинаковую длину и A состоит из суффикса B, склеенного с префиксом B). Например, defabc является циклическим сдвигом abcdef.

Вход:

Первая строка - A

Вторая строка - B

Выход:

Если A является циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

Sample Input:

defabc

abcdef

Sample Output:

3

Выполнение работы

Префикс функция calculate_lps () начинается с инициализации трех переменных:

- пустой список lps, заполненный нулями длиной строки, для которой нужно найти префикс функцию.

- i – индекс, для прохождения по строке

- j – переменная, хранящая в себе текущую длину совпадений суффикса с префиксом.

Далее, пока не достигнут конец строки, проверяем максимальное число совпадений символов, попутно увеличивая счетчики i,j. Как только мы нашли первые неравные символы, появляется два исхода: 1) совпадений не было, т. е. Можем просто продолжить перебор, оставив текущий префикс нулевым. 2) Совпадения были и нам требуется вернуть значение j на prefixes[j-1]. Откат на prefixes[j-1] символов позволяет нам эффективно продолжать поиск с максимально возможной позиции в подстроке, не повторяя уже выполненных проверок.:

Принцип работы алгоритма Кнута-Морриса-Пратта заключается в эффективном использовании префикс-функции. Вместо того чтобы возвращаться к уже проверенным символам при несоответствии, алгоритм использует префикс-функцию для сдвига подстроки на максимально возможную позицию.

Для поиска всех вхождений подстроки в строку, мы одновременно идем по строке и подстроке, сравнивая символы на каждой позиции. Если символы совпадают, переходим к следующей позиции. В случае несовпадения применяем префикс-функцию, чтобы определить, на какую позицию нужно сдвинуть подстроку вправо. Это позволяет продолжить сравнение с максимально возможной позицией, не теряя информации о возможных совпадениях. Сдвиг осуществляется на значение prefixes[j-1], где j — позиция, на которой произошло несовпадение.

Алгоритм продолжает сравнение символов до тех пор, пока не будут найдены все вхождения подстроки в строку. Если по завершению строки вхождений не найдено, возвращается пустой массив, а затем выводится значение -1.

Задачу определения циклического сдвига можно свести к задаче нахождения шаблона, равного изначальной длине текста, внутри удвоенного текста.

Оценка сложности алгоритма.

По времени:

Линейная сложность O(n+m), построение префикс-функции для шаблона длиной n и проход по тексту длиной m.

Для алгоритма циклического сдвига аналогично O(n + 2*m)

По памяти:

O(n) для хранения значений префикс-функции

Тестирование

Таблица 1 – Тестирование алгоритма

№ п/п	Входные данные	Выходные данные	Комментарии
1	lolol hellololololohellololhello	3,5,14	Тест первого задания
2	abadab ababadadab	-1	Шаблон не встречается в тексте
3	babbab bbabba	2	Верно, оптимизация для простых чисел.
4	aaaaaaaa aaaaaaaa	0	Одинаковые строки
5	abcdef defacb	-1	Не являются циклическим сдвигом друг друга

Выводы

Изучен принцип работы алгоритма Кнута-Морриса-Пратта. Написаны функции, решающие задачу по нахождению строки с помощью префикс-функции и задачу по нахождению индекса циклического сдвига.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from array import array


def calculate_lps(pattern):
    print("Вычисление префикс функции")
    n = len(pattern)
    lps = array('i', [0 for i in range(n)])
    j = 0
    i = 1
    while i < n:
        print(f"{i}: Длина префикса {j}; Сравниваем pattern[{i}] =
{pattern[i]} с pattern[{j}] = {pattern[j]}")
        if pattern[i] == pattern[j]:
            j += 1
            print(f"\tСовпало; записываем {j} в pattern[{i}]; уве-
личиваем i")
            lps[i] = j
            i += 1
        else:
            if j != 0:
                print(f"\tНесовпадение; ставим длину {lps[j - 1]}
(lps[{j} - 1])")
                j = lps[j - 1]
            else:
                print(f"\tНесовпадение; обнуляем lps[{i}]; увеличи-
ваем i")
                lps[i] = 0
                i += 1
    print("Массив lps:")
    print(', '.join(map(str, lps)))
    return lps


def kmp(pattern, text):
    lps = calculate_lps(pattern)
    n = len(pattern)
    appearances = []
    i, j = 0, 0
    print("Поиск pattern в text")
    while i < len(text):
        print(f"Позиция в тексте {i} - {text[i]}, длина шаблона {j}
- {pattern[j]}")
        if text[i] == pattern[j]:
            print("\tСовпадение, увеличиваем счетчики")
            i += 1
            j += 1
            if j == n:
                print(f"\tПолное вхождение шаблона с индекса {i-j},
переходим на длину шаблона lps[{j-1}] - {lps[j-1]}")
                appearances.append(i - j)
                j = lps[j - 1]
        else:
            if j != 0:
```

```
                    print(f"Несовпадение, переходим на длину шаблона
lps[{j-1}] - {lps[j-1]}")
                    j = lps[j - 1]
                else:
                    print("Несовпадение, увеличиваем счетчик текста")
                    i += 1
        return appearances if appearances else [-1]


    pattern = input()
    text = input()
    print(calculate_lps(pattern))
    print(",".join(map(str, kmp(pattern, text))))

    if len(pattern) == len(text):
        if pattern == text:
            print("Строки совпадают")
            print(0)
        else:
            print("Длины совпадают; проверка, является ли первая строка
циклическим сдвигом второй:")
            print("Найдем вхождения второй строки в удвоенную первую
строку")
            print(",".join(map(str, kmp(text, pattern*2))))
```