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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Построение и анализ алгоритмов»

Тема: Кнут-Моррис-Пратт

Студент гр. 3388 _____ Еникеев А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Цель работы

Изучение и практическое применение алгоритма Кнута–Морриса–Пратта (КМП) для эффективного поиска образца в строке, а также разработка метода определения циклического сдвига одной строки относительно другой.

Задание

Пункт 1

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P ($|P| \leq 25000$) 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, разделённые запятой; если P не входит в T, то вывести -1.

Пункт 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, то индекс начала строки B в A; иначе вывести -1. Если возможно несколько сдвигов, вывести первый индекс.

Выполнение работы

Пусть задан шаблон P длиной m и текст T длиной n.

Префикс-функция

Для каждой позиции i (от 1 до m−1) префикс-функция π[i] равна длине максимального собственного префикса P[0..k−1], который одновременно является суффиксом P[0..i].

Есть два указателя q = 1, k = 0, π[0]=0. Идем по шаблону:

- Если k = 0 и P[q]!=P[k]: π[q]=0, q++
- Если k > 0 и P[q]!=P[k]: k=π[k-1]
- Если P[q]=P[k]: k++, q++, π[q]=k+1

Возвращаем массив π. Отдельно стоит пояснить момент перехода k=π[k-1], который обеспечивает линейную сложность: если символы указателей q, k не совпали и указатель k не в начале шаблона, то мы не сбрасываемся сразу в начало, а откатываем указатель k к позиции π[k−1]. Потому что внутри суффикса P[0…k] у нас уже есть удостоенный доверием префикс длины π[k−1], который также является префиксом суффикса-подстроки P[0…q-1].

Поиск вхождений

После того, как построили массив π по строке шаблона, заводим указатель i=0 (для текста) и q=0 (текущая длина сопоставленного префикса шаблона). Идем по тексту:

- Пока q>0 и P[q]!=T[i]: q=π[q−1]
- Если P[q]=T[i]: q++
- Если q=m: добавить индекс i - m + 1 в список результатов, q=π[q−1]

Функция возвращает список индексов вхождений шаблона P в текст T. Здесь при несовпадении символов выполняется переход к наибольшему

префиксу совпадающей подстроки шаблона, тем самым сокращая количество сравнений.

Является ли строка A циклическим сдвигом строки B

Здесь A рассматривается как шаблон, а в качестве текста будет использоваться B+B. Алгоритмом КМП находим вхождение A в BB, если оно найдено, значит A - циклический сдвиг B и наоборот (учитывая, что длина A равна длине B). Подробнее:

- Если A и B не одной длины, они не могут быть циклическими сдвигами друг друга => сразу -1.
- На основе строки A строится префикс-функция.
- Поиск происходит аналогично алгоритму КМП, но строка B проходится дважды (это эквивалентно конкатенации B+B, но не потребует дополнительных затрат памяти).

  Алгоритм вернет индекс начала строки A в B или -1.

Оценка сложности

1. Построение префикс функции для шаблона длины m
   - Время: O(m)
   - Память: O(m)
2. Алгоритм КМП для шаблона длины m и текста длины n
   - Время: O(m + n)
   - Память: O(m)
3. Проверка циклического сдвига на основе КМП, где длина A и B равна n
   - Время: O(n)
   - Память: O(n)

В алгоритме КМП суммарное число переходов по префикс функции шаблона не превышает длину текста, поэтому сложность остается линейной.

Тестирование

Результаты тестирования программы представлены в табл. 1.

Табл. 1

Входные данные	Выходные данные
alksdmlaksmdlkamdlksamdlkamdlkamdaldml akdmlakdmalkmd amdaldmlakdmlakdmalkmdalksdmlaksmdlka mdlksamdlkamdlk	30
aba ababababa	0,2,4,6
bab bbaababbabaabab	4,7,12

Исходный код программы см. в прил. А.

Выводы

В лабораторной работе был реализован алгоритм Кнута–Морриса–Пратта с построением префикс-функции и его использованием для проверки циклического сдвига строк.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
DEBUG = 1


def prefix_function(pattern):
    m = len(pattern)
    pi = [0] * m
    k = 0
    if DEBUG:
        print("Построение pi-функции:")
    for q in range(1, m):
        if DEBUG:
            action = []
        while k > 0 and pattern[k] != pattern[q]:
            if DEBUG:
                    action.append(f"{pattern[k]}!={pattern[q]}:
откат k->pi[{k-1}]={pi[k-1]}")
            k = pi[k - 1]
        if pattern[k] == pattern[q]:
            k += 1
            if DEBUG:
                  action.append(f"{pattern[q]}={pattern[q]}: k++
-> {k}")
        else:
            if DEBUG and not action:
                    action.append(f"{pattern[k]}!={pattern[q]} и
k=0")
        pi[q] = k
        if DEBUG:
            print(f"q={q}: {'; '.join(action)}; pi[{q}]={k}")
    if DEBUG:
        print("Результат pi:", pi, "\n")
    return pi


def kmp_search(pattern, text):
    n, m = len(text), len(pattern)
    if m == 0:
        return [0] if n == 0 else [-1]

    pi = prefix_function(pattern)
    q = 0
    result = []
    if DEBUG:
        print("Поиск вхождений:")
    for i, ch in enumerate(text):
        while q > 0 and pattern[q] != ch:
            if DEBUG:
                    print(f"i={i}: {pattern[q]}!={ch}, откат
q->pi[{q-1}]={pi[q-1]}")
            q = pi[q - 1]
```

```
            if pattern[q] == ch:
                q += 1
                if DEBUG:
                    print(f"i={i}: {ch} совпало, q-> {q}")
            if q == m:
                pos = i - m + 1
                result.append(pos)
                if DEBUG:
                    print(f"Найдено вхождение с {pos}")
                q = pi[q - 1]
        return result if result else [-1]


    def is_cyclic_shift(A, B):
        if len(A) != len(B):
            return -1
        if not A:
            return 0
        if DEBUG:
            print("Проверка циклического сдвига:")
        pi = prefix_function(A)
        q = 0
        n = len(A)
        for i in range(2*n):
            ch = B[i%n]
            while q > 0 and A[q] != ch:
                if DEBUG:
                                print(f"i={i}:  {A[q]}!={ch},  откат
q->pi[{q-1}]={pi[q-1]}")
                q = pi[q-1]
            if A[q] == ch:
                q += 1
                if DEBUG:
                    print(f"i={i}: {ch} совпало, q-> {q}")
            if q == n:
                pos = i - n + 1
                if DEBUG:
                    print(f"Найден сдвиг {pos}")
                return pos if pos < n else -1
        return -1

    if __name__ == "__main__":
        A = input().strip()
        B = input().strip()
        result = kmp_search(A, B)
        #result = is_cyclic_shift(B, A)
        #print(result)
        print(','.join(map(str, result)))
```