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. 1304 _____ Мамин Р.А.

Преподаватель _____ Шевелева А.М.

Санкт-Петербург
2023

Цель работы

Изучить и на практике освоить азы нахождения подстрок в строках посредством алгоритма Кнута-Морриса-Пратта.

Задание 1

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.

Задание 2

Заданы две строки A ($|A| \leq 5000000$) и BB ($|B| \leq 5000000$). 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 вляется циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

Выполнение работы

Задание 1

Программа содержит следующие функциии:

Функция *compute_LPS_array(pattern, pattern_length)* вычисляет массив LPS (*Longest Proper Prefix Suffix*) для заданного шаблона. Массив LPS представляет собой массив, в котором на каждой позиции *i* указывается длина наибольшего префикса шаблона *pattern[0:i]*, который также является суффиксом этого же шаблона. Алгоритм вычисления массива LPS использует два указателя - *i* и *j*, которые двигаются по шаблону и сравнивают символы между собой. Временная сложность алгоритма - *O(m)*, где *m* - длина шаблона.

Функция *KMP(string, pattern)* реализует алгоритм Кнута-Морриса-Пратта для поиска всех вхождений образца в строку. Алгоритм использует вычисленный массив LPS для быстрого нахождения совпадений между строкой и образцом. Временная

сложность алгоритма - *O(m+n)*, где *m* и *n* - длины образца и строки соответственно. Алгоритм заключается в движении двух указателей - *i* и *j*, которые двигаются по строке и образцу соответственно. Если символы на текущих позициях совпадают, то оба указателя сдвигаются на одну позицию. В противном случае указатель *j* сдвигается на значение *LPS[j-1]*.

Аргумент *pattern* в обе функции представляет собой шаблон, для которого вычисляется массив LPS и который ищется в строке. Аргумент *string* представляет собой строку, в которой ищутся вхождения образца. Обе функции возвращают результат в виде массива LPS или строки, содержащей индексы всех вхождений образца в строке через запятую. Если образец не найден в строке, функция *KMP()* возвращает -1.

Использование алгоритма Кнута-Морриса-Пратта позволяет быстро и эффективно находить вхождения заданного образца в строку, что находит применение во многих задачах, например, в обработке текста или поиске паттернов в изображениях.

Задание 2

Функция *get_longest_prefix_suffix(pattern, pattern_length)* вычисляет массив *LPS (Longest Proper Prefix Suffix),* который представляет собой массив, содержащий длины наибольшего префикса образца *pattern[0:i]*, который является суффиксом этого же образца. Алгоритм вычисления массива LPS использует индексы, которые движутся по образцу и сравнивают символы между собой. Эта функция возвращает список, содержащий значения LPS для заданного образца.

Функция *circular_shift(string, pattern)* проверяет, можно ли получить заданный образец путем циклического сдвига (вращения) входной строки. Алгоритм использует массив LPS, вычисленный с помощью функции *get_longest_prefix_suffix(),* для сравнения символов образца и входной строки. Если образец можно получить путем циклического сдвига, функция возвращает индекс первого символа во входной строке после сдвига, иначе возвращает -1. Алгоритм заключается в движении двух указателей - *string_idx* и *pattern_idx*, которые двигаются по строке и образцу

соответственно.

Аргумент *pattern* в обе функции представляет собой образец, для которого вычисляется массив LPS и который может быть получен путем циклического сдвига входной строки. Аргумент *string* представляет собой строку, в которой производится поиск циклического сдвига образца.

В случае если образец нельзя получить путем сдвига, функция *circular_shift()* возвращает -1. Иначе, функция возвращает индекс первого символа во входной строке после сдвига в виде строки.

Тестирование

Тестирование программы см. в таблицах 1 и 2 Приложения Б.

Выводы

В результате выполнения лабораторной работы по алгоритму Кнута-Морриса-Пратта были представлены две реализации этого алгоритма. Было научено реализовывать функцию *compute_LPS_array()* для вычисления массива LPS для заданного шаблона и функцию *KMP()* для поиска всех вхождений образца в строку. Также был изучен алгоритм, используемый в функции *circular_shift(),* который позволяет проверить, можно ли получить заданный образец путем циклического сдвига входной строки. Для проверки корректности работы алгоритмов было необходимо составлять и запускать тесты. После успешного выполнения лабораторной работы были получены навыки, позволяющие применять алгоритм Кнута-Морриса-Пратта для решения задач, связанных с поиском подстроки в строке. Также было изучено, какие алгоритмы используются в функциях *compute_LPS_array()* и *circular_shift()* и как они могут быть применены в других задачах.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла - lab4_1.py.

```
def get_lps(pattern, M):
    len = 0
    lps = [0] * M
    i = 1
    while i < M:
        if pattern[i] == pattern[len]:
            len += 1
            lps[i] = len
            i += 1
        else:
            if len != 0:
                len = lps[len - 1]
            else:
                lps[i] = 0
                i += 1
    return lps

def KMP(string, pattern):
    lps = get_lps(pattern,len(pattern))
    m = len(string)
    n= len(pattern)
    i = 0
    j = 0
    res = []

    while i < m:
        if string[i] == pattern[j]:
            if j == n-1:
                res.append(str(i-j))
                j = lps[j]
            else:
                j+=1
        else:
            if j != 0:
                j = lps[j-1]
                continue
        if i == m - 1 and j == n - 1:
            break
        i+=1
    if len(res) == 0:
        return '-1'
    return ','.join(res)

if __name__ == '__main__':
```

```
    pattern = input()
    string = input()
    print(KMP(string,pattern))
```

Название файла - lab4_2.py.

```
def get_lps(P, M):
    lps = [0]*M
    for q in range(1,M):
        k = lps[q-1]
        while k > 0 and not P[k] == P[q]:
           k  = lps[k-1]
        if P[k] == P[q]:
            k = k+1
        lps[q] = k
    return lps

def Circle_shift(string, pattern):
    if len(string) != len(pattern):
        return '-1'
    lps = get_lps(pattern,len(pattern))
    m = len(string)
    n = len(pattern)
    i = 0
    j = 0
    while j < n:
        if string[i%m] == pattern[j]:
            if j == n-1:
                return str((i+1)%m)
            else:
                j+=1
        else:
            if j == n-1:
                return '-1'
            if j != 0:
                j = lps[j-1]
                continue
        i+=1
    return '-1'

if __name__ == '__main__':
    A = input()
    B = input()
    print(Circle_shift(A,B))
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ ПРОГРАММЫ

Таблица 1 – Резальтаты тестирования задачи №1.

№ теста	Входные данные	Выходные данные	Комментарий
1	"aba", "abababa"	"0,2,4"	Шаблон "aba" встречается в строке "abababa" на позициях 0, 2 и 4.
2	"aa", "aaaa"	"0,1,2"	Шаблон "aa" встречается в строке "aaaa" на позициях 0, 1 и 2.
3	"abc", "def"	"-1"	Шаблон "abc" не встречается в строке "def".
4	"abc", "abcdef"	"0"	Шаблон "abc" встречается в начале строки "abcdef".
5	"", "abc"	"-1"	Шаблон пустой, следовательно его вхождения в строку нет.

Таблица 2 – Результаты тестирования задачи №2.

№ теста	Входные данные	Выходные данные	Комментарий
1	"abcd", "abcd"	"0"	Образец "abcd" встречается в начале строки "abcd", после циклического сдвига индекс первого символа в строке равен 0.
2	"abcd", "bcda"	"1"	Образец "abcd" можно получить из строки "bcda" путем циклического сдвига на одну позицию вправо, после сдвига индекс первого символа в строке равен 1.
3	"aaa", "aaaa"	"-1"	Образец "aaa" нельзя получить из строки "aaaa" путем циклического сдвига,

			поэтому возвращается -1.
4	"abcd", "cdab"	"2"	Образец "abcd" можно получить из строки "cdab" путем циклического сдвига на две позиции вправо, после сдвига индекс первого символа в строке равен 2.
5	"abab", "baba"	"-1"	Образец "abab" нельзя получить из строки "baba" путем циклического сдвига, поэтому возвращается -1.