

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: Алгоритм Кнута-Морриса-Пратта

Студент гр. 9383

Соседков К.С.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить алгоритм 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 .

Задание 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.

Описание работы алгоритма.

Входные данные: текст(text) и подстрока(sub_string).

1) Вычисление префикс функции для строки `sub_string+'#'+text`. Результатом префикс функции является массив(`prefix_function_table`) длины `len(sub_string+'#'+text)`.

2) На основе полученного массива строится новый массив. Значениями этого массива являются индексы(i) элементов массива `prefix_function_table` для которых выполнено: `prefix_function_table[i] ==` длина подстроки.

3) Из каждого значения в полученном массиве вычитается длина подстроки умноженная на два.

Для выполнения второго задания выполняются такие же шаги с небольшими отличиями. Текстом является строка `sub_string+sub_string`, подстрокой является строка `text`.

Анализ алгоритма.

Префикс функция от строки `sub_string+'#'+text` строится за $O(s+t)$ операций (s — длина подстроки, t — длина текста).

Поиск подстроки содержит цикл по тексту, значит сложность равна $O(t)$.

Итоговая сложность алгоритма равна $O(s+t)$.

Описание основных функций и переменных.

Переменные:

`sub_string` — подстрока

`text` — текст

`indices` — массив содержащий индексы начал вхождения подстроки в текст.

Функции:

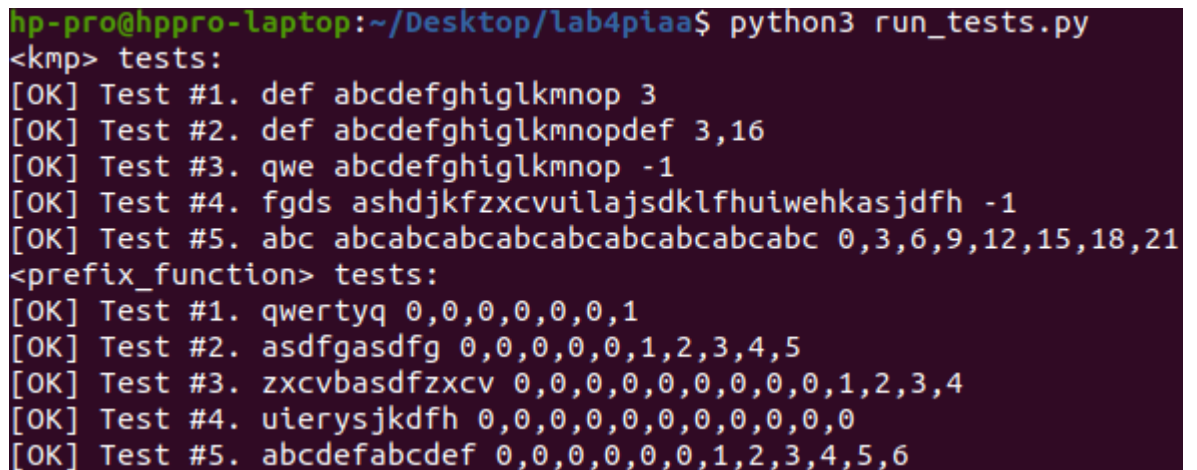
`prefix_function(string)` — префикс функция.

`kmp(string, sub_string)` — алгоритм Кнута-Морриса-Пратта.

Результаты тестирования представлены в таблице 1.

Тестирование.

Для основных функций `kmp` и `prefix_function` были написаны тесты. Для тестирования был написан Python-скрипт - `run_tests.py`. Результаты тестирования представлены на Рисунке 1.



```
hp-pro@hpro-laptop:~/Desktop/lab4p1aa$ python3 run_tests.py
<kmp> tests:
[OK] Test #1. def abcdefghiglkmnop 3
[OK] Test #2. def abcdefghiglkmnopdef 3,16
[OK] Test #3. qwe abcdefghiglkmnop -1
[OK] Test #4. fgds ashdkfzxcvuilajsdklfhuiwehkasjdfh -1
[OK] Test #5. abc abcabcabcabcabcabcabcabcabc 0,3,6,9,12,15,18,21
<prefix_function> tests:
[OK] Test #1. qwertyq 0,0,0,0,0,0,1
[OK] Test #2. asdfgasdfg 0,0,0,0,0,1,2,3,4,5
[OK] Test #3. zxcvbasdfzxcv 0,0,0,0,0,0,0,0,0,1,2,3,4
[OK] Test #4. uierysjkdfh 0,0,0,0,0,0,0,0,0,0,0
[OK] Test #5. abcdefabcdef 0,0,0,0,0,0,1,2,3,4,5,6
```

Рисунок 1 Результаты тестирования

Таблица 1. Результаты тестирования

Ввод	Вывод
def abcdefghiglkmnop	3
def abcdefghiglkmnopdef	3 16
fgds ashdkfzxcvuilajsdklfhuiwehkasjdfh	-1

Выводы.

При выполнении работы был изучен и реализован алгоритм Кнута-Морриса-Пратта, а также была изучена префикс функция и ее приложения.

**ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД.**

Название файла: lab4.py

```
import time
```

```
import sys
```

```
def prefix_function(s, max_len=sys.maxsize):
```

```
    p = [0]*len(s)
```

```
    for i in range(1, len(s)):
```

```
        k = p[i-1]
```

```
        while k > 0 and s[i] != s[k]:
```

```
            k = p[k-1]
```

```
        p[i] = k+1 if s[i] == s[k] else k
```

```
    return p
```

```
def kmp(string, sub_string):
```

```
    prefix_function_table = prefix_function(sub_string+'#'+string)
```

```
    return [index-len(sub_string)*2 for index, value in enumerate(prefix_function_table) if  
value==len(sub_string)]
```

```
if __name__ == '__main__':
```

```
    sub_string = input()
```

```
    string = input()
```

```
    indices = kmp(string, sub_string)
```

```
    if indices:
```

```
        print(indices[0], end=" ", sep=" ")
```

```
        for i in range(1, len(indices)):
```

```
            print(',', indices[i], end=" ", sep=" ")
```

```
        print("\n")
```

```
    else:
```

```
        print(-1)
```