**Федеральное агентство связи  
Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования   
«Московский технический университет связи и информатики»**

Кафедра «Теории вероятностей и прикладной математики»

**Лабораторная работа №3**

По дисциплине «Структуры и алгоритмы обработки данных»   
по теме «Методы поиска подстроки в строке»  
Вариант №6

Выполнила: Гердт Таисия

Группа: БСТ1955

Руководитель: Кутейников Иван Алексеевич

## Задание

Реализовать метод Кнута-Морриса-Пратта поиска подстроки в строке в соответствии с индивидуальным заданием. Для всех вариантов добавить реализацию добавления строк, ввода подстроки и поиска подстроки. Предусмотреть возможность существования пробела. Ввести опцию чувствительности / нечувствительности к регистру. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

Код программы:

#функция КМП

def kmp(pat, txt):

M = len(pat)

N = len(txt)

lps = [0]\*M

j = 0

lsp(pat, M, lps)

i = 0

flag = 0

while i < N:

if pat[j] == txt[i]:

i += 1

j += 1

if j == M:

print ("KMP: Found pattern at index " + str(i-j))

j = lps[j-1]

flag += 1

elif i < N and pat[j] != txt[i]:

if j != 0:

j = lps[j-1]

else:

i += 1

if flag == 0:

print("KMP: Pattern is not found")

#функция нахождения массива длиннейших префиксов являющихся суффиксами

def lsp(pat, M, lps):

len = 0

lps[0]

i = 1

while i < M:

if pat[i]== pat[len]:

len += 1

lps[i] = len

i += 1

else:

if len != 0:

len = lps[len-1]

else:

lps[i] = 0

i += 1

import time, random, string

flag = input("Enter 1 for case-sensitive search\nEnter 2 for non-case sensetive search\nEnter 0 for quit\n")

while(True):

if flag == "1":

#with open("verylongstr.txt", mode="r", encoding="utf-8") as f:

# for line in f:

# txt = line

txt = input("Enter string:\n")

pat = input("Enter pattern:\n")#"myUniquePattern"

t = time.time()

kmp(pat, txt)

t = time.time() - t

print(f"KMP execurtion time: {t\*1000} milliseconds")

t = time.time()

m = txt.index(pat)

t = time.time() - t

print(".index(): Found pattern at index ", m)

print(f".index() execurtion time: {t\*1000} milliseconds")

elif flag == "2":

txt = input("Enter string:\n").lower()

pat = input("Enter pattern:\n").lower()

t = time.time()

kmp(pat, txt)

t = time.time() - t

print(f"KMP execurtion time: {t\*1000} milliseconds")

t = time.time()

m = txt.index(pat)

t = time.time() - t

print(".index(): Found pattern at index ", m)

print(f".index() execurtion time: {t\*1000} milliseconds")

elif flag == "0":

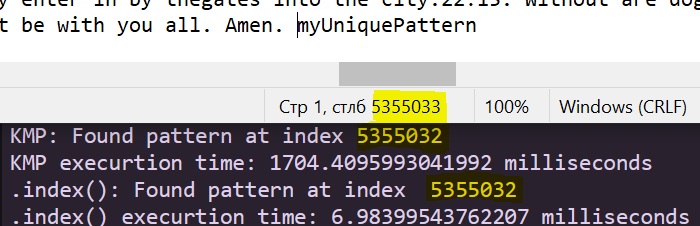
quit()

else:

print("Invalid command")

flag = input("Enter 1 for case-sensitive search\nEnter 2 for non-case sensetive search\nEnter 0 for quit\n")

Для проверки правильности работы программы использовали текстовый файл в 5355032 символа, в конце которого единожды встречается «myUniquePattern»:



Так в программе первый символ имеет нулевой индекс, индексы найденного паттерна отличаются на 1.

Составим таблицу времени выполнения поиска:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Поиск\размер** | **44661** | **531160** | **1275785** | **2579515** | **5355047** |
| **KMP** | 18.9614 | 174.5324 | 375.0100 | 740.0553 | 1704.4096 |
| 20.9434 | 169.5590 | 412.9248 | 749.0301 | 1893.9054 |
| 11.9879 | 190.5267 | 416.8835 | 730.0839 | 1642.6046 |
| **Среднее** | 17.2976 | 178.2060 | 401.6061 | 739.7231 | 1746.9732 |
| **Скорость обработки** | 2581.9210 | 2980.5947 | 3176.7074 | 3487.1358 | 3065.3287 |
| **Index** | 0.0000 | 0.0000 | 0.9630 | 2.9881 | 6.9840 |
| 0.0000 | 0.0000 | 0.9682 | 2.9600 | 4.9877 |
| 0.0000 | 0.9990 | 0.9983 | 1.9584 | 5.0151 |
| **Среднее** | 0.0000 | 0.3330 | 0.9765 | 2.6355 | 5.6623 |
| Скорость обработки | #ДЕЛ/0!(быстро!) | 1595116.83 | 1306510 | 978765.2 | 945740.07 |

**Вывод:** Стандартная функция поиска подстроки в строке показала лучшие результаты, чем КМП алгоритм.