Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____«Информатика и системы управления»_____

КАФЕДРА _____«Программное обеспечение ЭВМ и информационные технологии»_____

Рубежный контроль №1

По дисциплине «Анализ алгоритмов»

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Выполнила: студентка группы ИУ7-54(Б)
Серёгина Д.В.

Москва, 2020

Задание

Реализовать оптимальный алгоритм решения уравнений из слов, включая = и -.

Есть словарь стран и столиц. Пример уравнения: Париж-Франция = Рим-Италия.

Пользователь может подставить X в любое из полей.

Аналитическая часть

Для решения поставленной задачи мною был использован алгоритм Кнута-Морриса-Пратта, который находит все вхождения заданного образца в строку. В данном алгоритме ключевым элементом является префикс-функция, ещё говорят, что это функция построения конечного автомата.

Перед выполнением основной части, должна выполниться префикс-функция или функция для нахождения перехода по несовпадению, данные несовпадения записываются в массив fail. После нахождения массива fail выполняется основная часть алгоритма, в которой, как в конченом автомате происходит переход из одного состояния в другое: если сравнение успешно, то переход к следующему состоянию автомата, иначе выбранный символ используется повторно.

Технологическая часть

Код программы:

```
def getFail(substring):
    fail = [0]*len(substring)
    for i in range(1,len(substring)):
        k = fail[i-1]
        while k > 0 and substring[k] != substring[i]:
            k = fail[k-1]
        if substring[k] == substring[i]:
            k = k + 1
        fail[i] = k
```

```
    return fail


def kmp(substring, text):
    index = -1
    f = getFail(substring)
    k = 0
    for i in range(len(text)):
        while k > 0 and substring[k] != text[i]:
            k = f[k-1]
        if substring[k] == text[i]:
            k = k + 1
        if k == len(substring):
            index = i - len(substring) + 1
            break
    return index


def get_position(text):
    pos = 0
    if (first < kmp('X', equ)):
        if (ravno < kmp('X', equ)):
            if ((len(equ) - 1) == (kmp('X', equ))):
                pos = 4
            else:
                pos = 3
        else:
            pos = 2
```

```
    else:
        pos = 1
    return pos

def find_in_text(pos, text, first, ravno, second):
    if (pos == 1):
        word = equ[first + 1 : ravno]
    if (pos == 2):
        word = equ[0 : first]
    if (pos == 3):
        word = equ[second + 1 : len(equ)]
    if (pos == 4):
        word = equ[ravno + 1 : second]
    return word

def get_key(d, value):
    for Country, Sity in d.items():
        if Sity == value:
            return Country

def get_answer(pos, word):
    if ((pos % 2) == 0):
        answer = d.get(word)
    else:
        answer = get_key(d, word)
    return answer
```

```
d = {'Russia': 'Moscow', 'Canada': 'Ottawa', 'Italy': 'Roma', 'Spain': 'Barselona'}

print("Make an equation of countries and their capitals!\n")
print("Available countries and capitals: \n", d)
print("\nEnter an equation of the form:\n")
print("Russia-Moscow = Canada-X or X-Roma = Spain-Barselona\n")
equ = input("Your variant: ")

first = kmp('-', equ)
ravno = kmp('=', equ)
piece = equ[ravno + 1 : len(equ)]
second = (kmp('-', piece) + ravno + 1)

pos = get_position(equ)
word = find_in_text(pos, equ, first, ravno, second)

print("\nGot it! X = ", get_answer(pos, word))
```

Экспериментальный раздел

Пример работы программы:

```
Make an equation of countries and their capitals!

Available countries and capitals:
 {'Russia': 'Moscow', 'Canada': 'Ottawa', 'Italy': 'Roma', 'Spain': 'Barselona'}

Enter an equation of the form:

Russia-Moscow = Canada-X or X-Roma = Spain-Barselona

Your variant: Russia-Moscow = Canada-X

Got it! X = Ottawa
>>>
```

```
Make an equation of countries and their capitals!

Available countries and capitals:
 {'Russia': 'Moscow', 'Canada': 'Ottawa', 'Italy': 'Roma', 'Spain': 'Barselona'}

Enter an equation of the form:

Russia-Moscow = Canada-X or X-Roma = Spain-Barselona

Your variant: X-Roma = Spain-Barselona

Got it! X = Italy
>>>
```

```
Make an equation of countries and their capitals!

Available countries and capitals:
 {'Russia': 'Moscow', 'Canada': 'Ottawa', 'Italy': 'Roma', 'Spain': 'Barselona'}

Enter an equation of the form:

Russia-Moscow = Canada-X or X-Roma = Spain-Barselona

Your variant: Russia-X = Canada-Ottawa

Got it! X = Moscow
>>>
```

Заключение

В ходе выполнения данного задания был изучен алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке.

Во время разработки программного обеспечения были получены практические навыки реализации указанных алгоритмов на языке Python.