Государственное образовательное учреждение высшего профессионального образования
“Московский государственный технический университет имени Н.Э.Баумана”

Дисциплина: Анализ алгоритмов

Рубежный контроль №2

Нахождение подстроки в строке с помощью регулярных выражений и конечного автомата

Студент группы ИУ7-55Б,
Руднев К. К.,

Преподаватель,
Волкова Л. Л.,
Строганов Ю. В.

2019 г.

1 Аналитическая часть

В рамках раздела будет дано аналитическое описание регулярных выражений и конечного автомата.

1.1 Описание алгоритмов

1.1.1 Регулярные выражения

Регулярные выражения — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

1.1.2 Конечный автомат

Конечный автомат можно охарактеризовать множеством состояний (вершин) и переходов (дуг, соединяющие вершины). Среди состояний есть два специальных - состояние начала и конца. Если строка читается данным автоматом, то после прохода по строке, автомат должен оказаться в одном из заключительных состояний. На этом основывается алгоритм поиска подстроки в строке с помощью конечного автомата.

2 Конструкторская часть

В дальнейшем на рисунке 1 будет представлена схема разработанного автомата.

2.1 Разработка алгоритмов

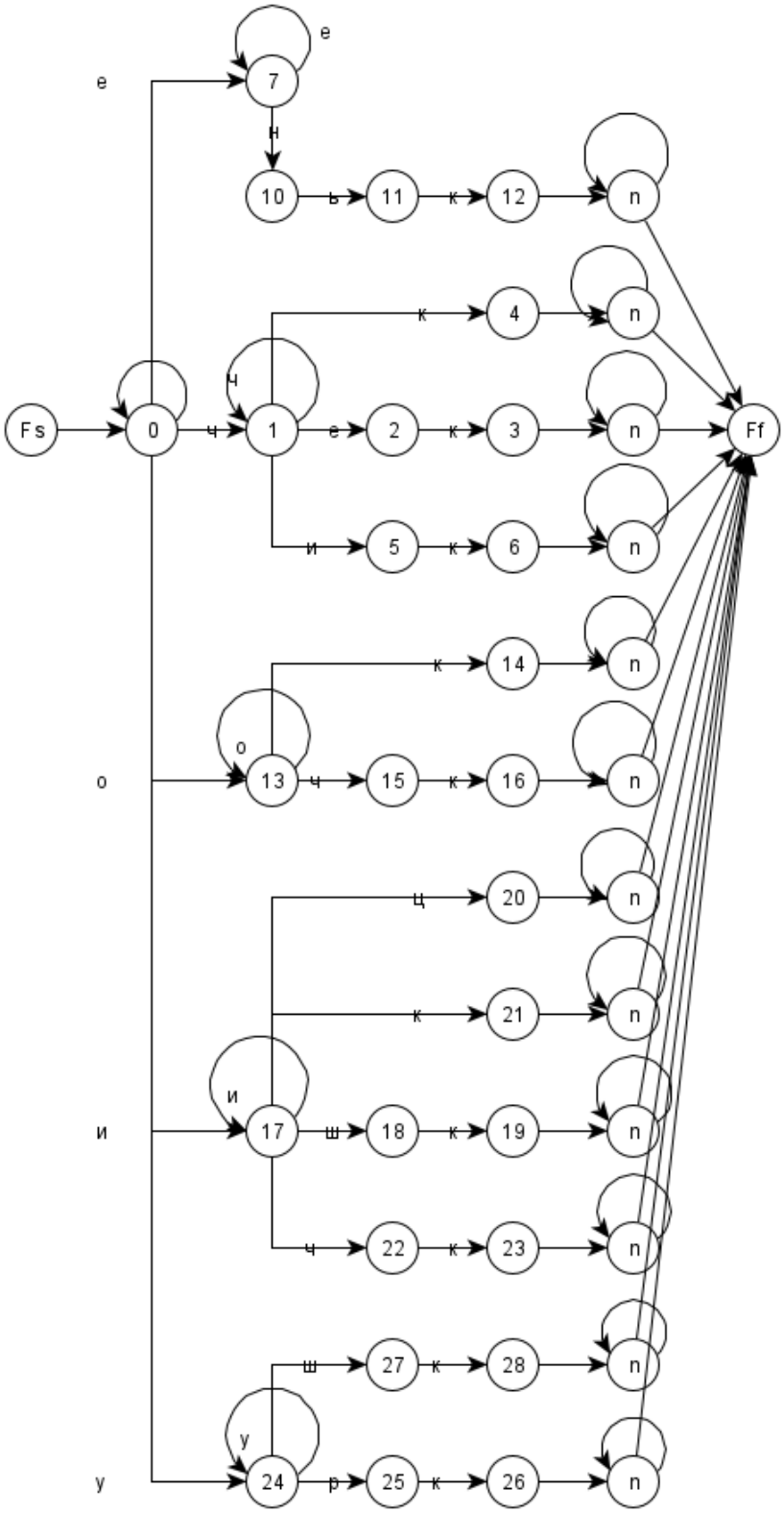

Рис. 1: Автомат для нахождения слов с уменьшительно-ласкательными суффиксами

3 Технологическая часть

В рамках раздела будут описаны инструментарии разработки, выбор среды, требования к ПО. Также будут предоставлены листинги конкретных реализаций алгоритмов.

3.1. Средства реализации

Для реализации алгоритмов использовался язык программирования Python 3.8.0 и среда разработки PyCharm Community Edition 2019.3.1 by JetBrains. У меня есть определенный опыт работы с данным языком, которого будет достаточно для реализации текущей работы, а среда разработки имеет бесплатную комьюнити версию и удобный интерфейс, упрощающий разработку приложения/скрипта.

3.2. Требования к программному обеспечению

На вход программа должна получать текст, на выход отображать все найденные слова с уменьшительно-ласкательными суффиксами.

3.3. Листинг кода

На листингах 1-3 представлена реализация поиска слова с определенным суффиксом с помощью регулярного выражения и конечного автомата. На листинге 1 представлен код регулярных выражений и всего, что с ними связано. На листинге 2 представлен код класса автомата для нумерации его состояний. На листинге 3 представлен код функций для работы с автоматом.

Листинг 1: Листинг разработанных регулярных выражений и функций с ними работающих

```
import re
import enum

def get_suffix_str(suffix):
    string_res = ''
    for i in suffix:
        string_res += r'\b\w+' + i + r'\w*|'

    return string_res[:-1]

suffix = ['чек', 'ецк', 'ишк', 'чк', 'очк',\
                'иц', 'урк', 'чик', 'ик', 'ушк', 'ок', 'ичк', 'еньк']
suffix_str3 = get_suffix_str(suffix)

res2 = re.findall(suffix_str3, text)
```

Листинг 2: Листинг класса для нумерации состояний автомата

```
class Automat(enum.Enum):
        start = 0
        ########
        ch = 1
        ch_e = 2
        ch_e_k = 3
        ch_k = 4
        ch_i = 5
        ch_i_k = 6
```

```
        ########
        e_n = 10
        e_n_soft = 11
        e_n_soft_k = 12
        ########
        o = 13
        o_k = 14
        o_ch = 15
        o_ch_k = 16
        ########
        i = 17
        i_sh = 18
        i_sh_k = 19
        i_c = 20
        i_k = 21
        i_ch = 22
        i_ch_k = 23
        ########
        u = 24
        u_r = 25
        u_r_k = 26
        u_sh = 27
        u_sh_k = 28
        ########
        finish = 29
```

Листинг 3: Листинг функций работающих с автоматом

```
def check_words(word, automat):
    current_state = automat.start
    original_word = word
    return_flag = 0
    a = 0
    word = word.lower()

    while a <= len(word):
        try:
            i = word[a]
        except:
            pass

        if current_state == automat.start:
            if i == 'ч' and a:
                current_state = automat.ch
            elif i == 'о' and a:
                current_state = automat.o
            elif i == 'у' and a:
                current_state = automat.u
            elif i == 'и' and a:
                current_state = automat.i
            elif i == 'е' and a:
                current_state = automat.e
```

```
        elif current_state != automat.finish:
            if current_state == automat.ch:
                if i == 'ч':
                    pass
                elif i == 'е':
                    current_state = automat.ch_e
                elif i == 'и':
                    current_state = automat.ch_i
                elif i == 'к':
                    current_state = automat.ch_k
                else:
                    current_state = automat.start
                    a -= 1

            elif current_state == automat.ch_e:
                if i == 'к':
                    current_state = automat.ch_e_k
                else:
                    current_state = automat.start
                    a -= 2

            elif current_state == automat.ch_e_k:
                current_state = automat.finish

            elif current_state == automat.ch_k:
                current_state = automat.finish

            elif current_state == automat.ch_i:
                if i == 'к':
                    current_state = automat.ch_i_k
                else:
                    current_state = automat.start
                    a -= 2

            elif current_state == automat.ch_i_k:
                current_state = automat.finish

            elif current_state == automat.o:
                if i == 'ч':
                    current_state = automat.o_ch
                elif i == 'к':
                    current_state = automat.o_k
                elif i == 'о':
                    pass
                else:
                    current_state = automat.start
                    a -= 1

            elif current_state == automat.o_k:
                current_state = automat.finish

            elif current_state == automat.o_ch:
```

```
        if i == 'к':
            current_state = automat.o_ch_k
        else:
            current_state = automat.start
            a -= 2

    elif current_state == automat.o_ch_k:
        current_state = automat.finish

    elif current_state == automat.u:
        if i == 'у':
            pass
        elif i == 'р':
            current_state = automat.u_r
        elif i == 'ш':
            current_state = automat.u_sh
        else:
            current_state = automat.start
            a -= 1

    elif current_state == automat.u_r:
        if i == 'к':
            current_state = automat.u_r_k
        else:
            current_state = automat.start
            a -= 2

    elif current_state == automat.u_r_k:
        current_state = automat.finish

    elif current_state == automat.u_sh:
        if i == 'к':
            current_state = automat.u_sh_k
        else:
            current_state = automat.start
            a -= 2

    elif current_state == automat.u_sh_k:
        current_state = automat.finish

    elif current_state == automat.i:
        if i == 'ч':
            current_state = automat.i_ch
        elif i == 'к':
            current_state = automat.i_k
        elif i == 'ш':
            current_state = automat.i_sh
        elif i == 'ц':
            current_state = automat.i_c
        elif i == 'и':
            pass
        else:
```

```
            current_state = automat.start
            a -= 1

    elif current_state == automat.i_sh:
        if i == 'к':
            current_state = automat.i_sh_k
        else:
            current_state = automat.start
            a -= 2

    elif current_state == automat.i_sh_k:
        current_state = automat.finish

    elif current_state == automat.i_c:
        current_state = automat.finish

    elif current_state == automat.i_k:
        current_state = automat.finish

    elif current_state == automat.i_ch:
        if i == 'к':
            current_state = automat.i_ch_k
        else:
            current_state = automat.start
            a -= 2

    elif current_state == automat.i_ch_k:
        current_state = automat.finish

    elif current_state == automat.e:
        if i == 'е':
            pass
        elif i == 'н':
            current_state = automat.e_n
        else:
            current_state = automat.start
            a -= 1

    elif current_state == automat.e_n:
        if i == 'ь':
            current_state = automat.e_n_soft
        else:
            current_state = automat.start
            a -= 2

    elif current_state == automat.e_n_soft:
        if i == 'к':
            current_state = automat.e_n_soft_k
        else:
            current_state = automat.start
            a -= 3
```

```
                elif current_state == automat.e_n_soft_k:
                    current_state = automat.finish

            else:
                break

            a += 1

        if current_state == automat.finish:
            return_flag = 1

        return return_flag

    def find_suffixes(words, automat):
        res = []
        for i in words:
            if check_words(i, automat):
                res.append(i)

    return res
```

Заключение

Выполнен рубежный контроль по нахождению подстроки в строке с помощью регулярных выражений и конечного автомата.