Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Рубежный контроль №2

по курсу «Анализ алгоритмов»

Конечные автоматы и регулярные выражения

Выполнил: Харламов П.С., гр. ИУ7-55Б

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

2020 г.

Оглавление

Введение **2**

1 Аналитический раздел **3**
1.1 Конечные автоматы 3
1.2 Регулярные выражения 3
1.3 Вывод 4

2 Технологический раздел **5**
2.1 Требования к программному обеспечению 5
2.2 Средства реализации 5
2.3 Листинг кода 5
2.4 Вывод 11

Заключение **12**

Литература **13**

Введение

Задача лабораторной работы состоит в том, что нужно при помощи конечного автомата и регулярного выражения написать программу поиска всех групп вуза: для специалитета, бакалавров, магистров в тексте.

1. Аналитический раздел

В данном разделе будут описаны конечные автоматы и регулярные выражения.

1.1 Конечные автоматы

Конечные автоматы - это до предела упрощенная модель компьютера имеющая конечное число состояний, которая жертвует всеми особенностями компьютеров такие как ОЗУ, постоянная память, устройства ввода-вывода и процессорными ядрами в обмен на простоту понимания, удобство рассуждения и легкость программной или аппаратной реализации[1].

С помощью КА можно реализовать такие вещи как, регулярные выражения, лексический анализатор, ИИ в играх и тд.

Для реализации поставленной задачи был спроектирован конечный автомат, представленный на рисунке 1.1

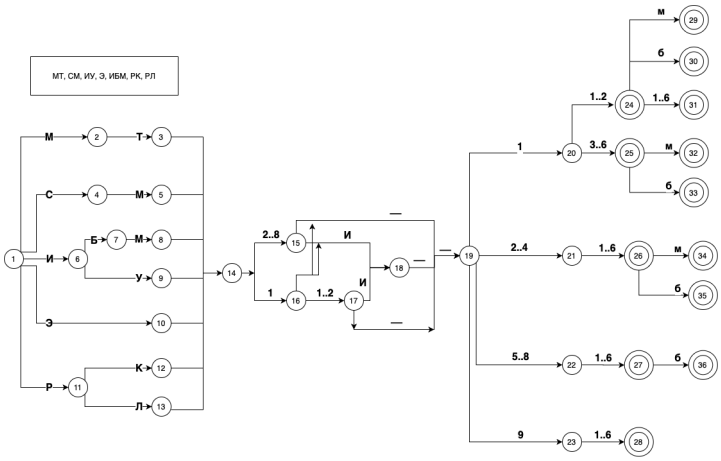

Рис. 1.1: Конечный автомат для поиска группы вуза

1.2 Регулярные выражения

Регулярные выражения — язык поиска подстроки или подстрок в тексте. Для поиска используется паттерн (шаблон, маска), состоящий из символов и метасимволов (символы, которые обозначают не сами себя, а набор символов).

Это довольно мощный инструмент, который может пригодиться во многих случая — поиск, проверка на корректность строки и т.д. Спектр его возможностей трудно уместить в одну статью[2].

Для реализации поставленной задачи был спроектировано регулярное выражение по образу конечный автомата, представленного на рисунке 1.1

$$(МТ+И(У + БМ)+Р(Л + К + КТ)+$$

$$+\text{ФН}+\text{С}(\text{М}\ +\ \text{ГН})+\text{Э}+\text{БМТ}+\text{Л}+\text{АК})(1+...+12)(\text{И}\ +\ '\ ')-$$
$$-(1+...+12)(\text{Б}\ +\ \text{М}\ +\ \text{б}\ +\ \text{м}\ +\ '\ ')$$

1.3 Вывод

В данном разделе были описаны конечные автоматы и регулярные выражения.

2. Технологический раздел

В данном разделе будут предъявлены требования к программному обеспечению, средства реализации и листинги кода.

2.1 Требования к программному обеспечению

Программное обеспечение должно реализовавать поиск подстроки в строке с помощью конечного автомата и регулярного выражения. Причем, для конечного автомата не должно исполизоваться никаких библиотек, а реализация регулярного выражения может быть осуществлена с использованием библиотек.

2.2 Средства реализации

Для выполнения поставленной задачи был использован язык программирования Python. Среда для разработки IDLE. Для измерения времени была взята функция time.time() из библиотеки time.

Данный язык обусловлен тем, что функции необходимые для реализации регулярного выражения находятся в встроенной библиотеке re.

2.3 Листинг кода

В данном подразделе представлены листинги кода реализации конечного автомата и регулярного выражения, которые были представлены в аналитическом разделе.

Листинг 2.1: Реализация конечного автомата

```
1  import time
2
3  def GetFSM(stroke):
4
5  state = 1
6  groupName = ''
7
8  for value in stroke:
9  # print(state)
10 # print("value = " + value)
11 if state == 1:
12 if value == 'M':
13 state = 2
14 elif value == 'C':
15 state = 4
16 elif value == 'N':
17 state = 6
```

```
18 elif value == 'E':
19 state = 14
20 elif value == 'P':
21 state = 11
22 else:
23 break
24 groupName += value
25
26 elif state == 2:
27 if value == 'T':
28 state = 14
29 else:
30 break
31 groupName += value
32
33 elif state == 4:
34 if value == 'M':
35 state = 14
36 else:
37 break
38 groupName += value
39
40 elif state == 6:
41 if value == 'B':
42 state = 7
43 elif value == 'Y':
44 state = 14
45 else:
46 break
47 groupName += value
48
49 elif state == 11:
50 if value == 'K':
51 state = 14
52 elif value == 'L':
53 state = 14
54 else:
55 break
56 groupName += value
57
58 elif state == 7:
59 if value == 'M':
60 state = 14
61 else:
62 break
63 groupName += value
64
65 elif state == 14:
66 if value.isdigit() and int(value) in [2,3,4,5,6,7,8] :
67 state = 15
68 elif value.isdigit() and int(value) == 1:
69 state = 16
70 else:
```

```
71 break
72 groupName += value
73
74 elif state == 15:
75 if value == '-':
76 state = 19
77 elif value == 'N':
78 state = 18
79 else:
80 break
81 groupName += value
82
83 elif state == 16:
84 if value.isdigit() and int(value) in [1,2]:
85 state = 17
86 elif value == 'N':
87 state = 18
88 elif value == '-':
89 state = 19
90 else:
91 break
92 groupName += value
93
94 elif state == 17:
95 if value == 'N':
96 state = 18
97 elif value == '-':
98 state = 19
99 else:
100 break
101 groupName += value
102
103 elif state == 18:
104 if value == '-':
105 state = 19
106 else:
107 break
108 length += 1
109
110 elif state == 19:
111 if value.isdigit() and int(value) in [1]:
112 state = 20
113 elif value.isdigit() and int(value) in [2,3,4]:
114 state = 21
115 elif value.isdigit() and int(value) in [5,6,7,8]:
116 state = 22
117 elif value.isdigit() and int(value) in [9]:
118 state = 23
119 else:
120 break
121 groupName += value
122
123 elif state == 20:
```

```
124 if value.isdigit() and int(value) in [1,2]:
125 state = 24
126 elif value.isdigit() and int(value) in [3,4,5,6]:
127 state = 25
128 else:
129 break
130 groupName += value
131
132 elif state == 21:
133 if value.isdigit() and int(value) in [1,2,3,4,5,6]:
134 state = 26
135 else:
136 break
137 groupName += value
138
139 elif state == 22:
140 if value.isdigit() and int(value) in [1,2,3,4,5,6]:
141 state = 27
142 else:
143 break
144 groupName += value
145
146 elif state == 23:
147 if value.isdigit() and int(value) in [1,2,3,4,5,6]:
148 state = 28
149 else:
150 break
151 groupName += value
152
153 elif state == 24:
154 if value == "M":
155 state = 29
156 elif value == "B":
157 state = 30
158 elif value.isdigit() and int(value) in [1,2,3,4,5,6]:
159 state = 31
160 else:
161 break
162 groupName += value
163
164 elif state == 25:
165 if value == "M":
166 state = 32
167 elif value == "B":
168 state = 33
169 else:
170 break
171 groupName += value
172
173 elif state == 26:
174 if value == "M":
175 state = 34
176 elif value == "B":
```

```
177 state = 35
178 else:
179 break
180 groupName += value
181
182 elif state == 27:
183 if value == "B":
184 state = 36
185 else:
186 break
187 groupName += value
188
189 # print("State = " + str(state))
190 # print("Group = " + str(groupName))
191
192 if state >= 24 and state <= 36:
193 return groupName
194 else: return ''
195
196 def FindGroups(text):
197
198 groups = []
199 text += "*******"
200
201 i = 0
202
203 while(1):
204
205 if len(text) <= i + 10:
206 break
207
208 tmpStroke = text[i: i + 10]
209 tmpStroke = tmpStroke.upper()
210
211 group = GetFSM(tmpStroke)
212 if group != '':
213 groups.append(group)
214
215 i += 1
216
217 return groups
218
219 if __name__ == '__main__':
220 f = open('/kek/text.txt', 'r')
221
222 text = f.read()
223
224 f.close()
225
226 start_time = time.time()
227
228 findedGroups = FindGroups(text)
229
```

```
totalTime = time.time() - start_time
totalTime = round(totalTime * 1000, 4)
print("--- %s seconds ---" % totalTime)

# for group in findedGroups:
# print(group)
```

Листинг 2.2: Реализация регулярного выражения

```
import re
import time

def FindGroups(text, doPrint):

text += "*******"

i = 0

while(1):

if len(text) <= i + 10:
break

tmpStroke = text[i: i + 10]
tmpStroke = tmpStroke.upper()

match = re.match(r'(MT|IU|RL|FN|CM|E|RK|BMT|IBM|L|SGN|RKT|AK)(10|11|12|[1-9])
    -((10|11|12|[1-9]))[1-6][bmBM]?',
tmpStroke)

if doPrint == 1 and match != None:
print(match[0])

i += 1


if __name__ == '__main__':

f = open('/kek/text.txt', 'r')

text = f.read()

f.close()

start_time = time.time()

FindGroups(text, 0)

totalTime = time.time() - start_time
totalTime = round(totalTime * 1000, 4)
print("--- %s seconds ---" % totalTime)
```

2.4 Вывод

В данном разделе были предъявлены требования к программному обеспечению, средства реализации и листинги кода.

Заключение

Были изучены основные принципы работы регулярных выражений и конечных автоматов для поиска подстроки в тексте, а также разработана программа, решающая задау поиска в тексте групп университета МГТУ им. Баумана.

Литература

[1] Конечные автоматы (finite-state machine) [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/358304/, свободный. (Дата обращения: 5.2.2020 г.)

[2] Регулярные выражения [Электронный ресурс]. - Режим доступа: https://habr.com/ru/company/badoo/blog/343310/, свободный. (Дата обращения: 5.2.2020 г.)