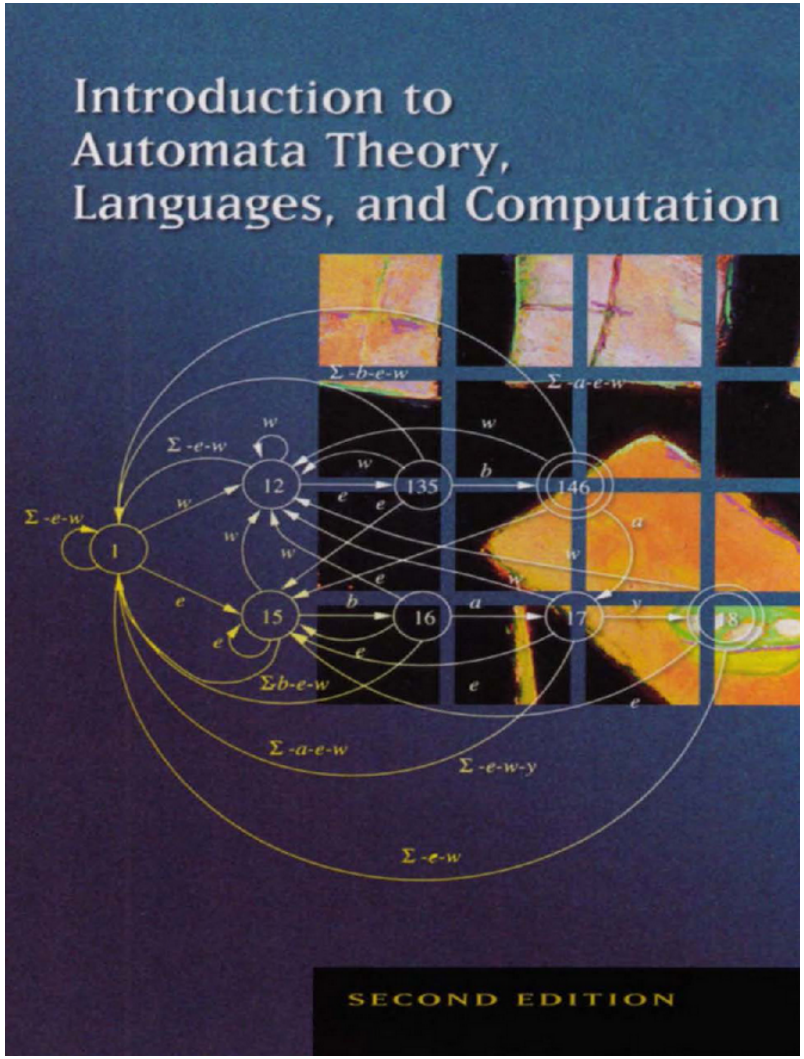
Регулярные выражения

@pvavilin

25 марта 2022 г.

Outline

Что такое регулярные выражения?



Варианты регулярок

Метасимволы	BRE	ERE
Точка, ^, $, [...], [^...]	✓	✓
Произвольное число	*	*
Квантификаторы + и ?		+ ?
Интервальный квантификатор	\{*min*, *max*\}	{*min*, *max*}
Группировка	\(...\)	(...)
Применение квантификаторов к скобкам	✓	✓
Обратные ссылки	от \1 до \9	
Конструкция выбора		✓

PCRE

Библиотеки RegEx в Python

- стандартная
- regex

Глобы

```
man 7 glob
```

```
import glob

print(glob.glob("*.py"))
```

```
['automaton_kmp.py', 'naive_re.py', 'naive_grep.py']
```

grep

g/<Regular Expression>/p

```
echo "g/def/p" | ed naive_re.py
1234
def match(regexp: str, text: str) -> bool:
def matchhere(regexp: str, text: str) -> bool:
def matchstar(c: str, regexp: str, text: str) -> bool:
```

```
./naive_grep.py '^def .*(.*):$' \
  ../praktikum_project_5/*/*.py \
  | head -n 5
```

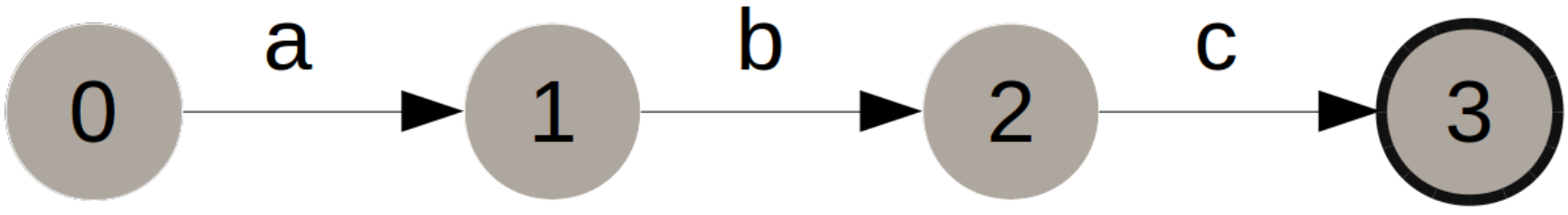
Наивная реализация регулярок

```
print(match("abc", "abc"))
print(match("abc$", "xyzabc"))
print(match("^abc", "xyzabc"))
print(match("^abc", "abcx"))
print(match("a*b", "bcd"))
print(match("a*b", "aaaaabcd"))
```

```
True
True
False
True
True
True
```

Конечные Автоматы

	a	b	c
0	1	0	0
1	1	2	0
2	1	0	3



Поиск подстроки на DFA

```
kmp1 = KMP("abc")
print(kmp1.search("abcd"))
print(kmp1.search("fooabcd"))
print(kmp1.search("foobar"))
```

```
0
3
-1
```

Реализация RegExp на конечных автоматах

для особо пытливых

Примеры использования RE

Задача: написать регулярку, проверяющую, что в строке корректный email адрес

Задача решена

```
print(re.search(
      r"\w+@\w+\.\w+",
      "example@gmail.com"
)[0])
print(re.search(
      r"\w[\w_.]+@\w+\.\w+",
      "example.1.2.3@gmail.com"
)[0])
```

```
example@gmail.com
example.1.2.3@gmail.com
```

Задача решена?

```
print(re.search(
      r"\w+@\w+\.\w+",
      "example@foo.gmail.com"
)[0])
print(re.search(
      r"\w[\w_.]+@\w+\.\w+",
      "example.1.2.3@foo.gmail.com"
)[0])
```

```
example@foo.gmail
example.1.2.3@foo.gmail
```

Задача решена

```
rg = re.compile(
  r"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+"
  r"(?:\.[a-zA-Z0-9-]+)+$"
)

print(re.match(
     rg, "example@foo.gmail.com"
)[0])
print(re.match(
     rg, "example.1.2.3@foo.gmail.com"
)[0])
```

```
example@foo.gmail.com
example.1.2.3@foo.gmail.com
```

Опережающие и ретроспективные проверки

Задача: заменить переводы строк на
,
за исключением случая, если перед этим шел html-тэг

Задача решена

```
import re

print(re.sub(
      r"([^>])\n",
      r"\1<br\\>",
      "<p>Привет\ndрузья</p>"
))
```

```
<p>Привет<br\>друзья</p>
```

Задача решена?

```
import re

print(re.sub(
      r"([^>])\n",
      r"\1<br\>",
      "<p>Привет\n\nдрузья</p>"
))
```

```
<p>Привет<br\>
друзья</p>
```

Задача решена!

```
import re

print(re.sub(
      r"(?<=[^>])\n",
      r"<br\>",
      "<p>Привет\n\nдрузья</p>"
))
```

```
<p>Привет<br\><br\>друзья</p>
```

lookahead & lookbehind

(?<=...)	Должно совпасть слева (Позитивная ретроспективная проверка).
(?<!...)	Не должно совпасть слева (Негативная ретроспективная проверка).
(?=...)	Должно совпасть справа (Позитивная опережающая проверка).
(?!...)	Не должно совпасть справа (Негативная опережающая проверка).

Отладка RegExp

RegeExer
Как это отладить?!

Практика

Задача: преобразовать все ссылки в тексте в html-тэги url

Практика

Задача: является ли текст числом (в том числе, дробным)?

Практика

Задача: поставить пробелы после запятых, если их там нет.

Практика

Задача: заменить идущие подряд знаки ,.!? на один

Литература

- import re
- lookahead & lookbehind
- хорошая статья
- Практика программирования
- Регулярные выражения
- NFA & RE
- Алгоритмы