COMP284 Scripting Languages
Lecture 4: Perl (Part 3)
Handouts

Department of Computer
School of Electrical Engineering, Electronics, an
University of Liverpo

Contents

Regular expressions: Motivation

Suppose you are testing the performance of a new sorting algorithm by measuring its runtime on randomly generated arrays of numbers of a given length:


```
Generating an unsorted array with 10000 elements took 1.250 seconds
Sortin
Genera
Sortin
Genera
Sorting took 8.951 seconds
```


Your task is to write a program that determines the aver the sorting algorithm:


```
Average runtime for 10000 elements is 8.886 seconds
```

Solution: The regular expression `/^Sorting took (\d+\.\d+) seconds/` allows us to get the required information

$\rightsquigarrow$ Regular expressions are useful for information extraction

Regular expressions: Motivation

Suppose you have recently taken over responsibility for a company's website. You note that their HTML files contain a large number of URLs containing superfluous occurrences of `/../`, e.g.

```
http://www.myorg.co.uk/info/refund/../vat.html
```

Your task i
equivale

```
http:/
```

while making sure that relative URLs like

```
../video/disk.html
```

are preserved

Solution: `s!/[^\/]+/\.\.!!;` removes a superfluous dot-segment

$\rightsquigarrow$ Substitution of regular expressions is useful for text manipulation

Regular expressions: Introductory example

```
\Ahttps?:\/\/[^\/]+\/.\w.\/(cat|dog)\/\1
```


- `\A` is an assertion or anchor
- `h`, `t`, `p`, `s`, `:`, `\/`, `c`, `a`, `t`, `d`, `o`, `g` are characters
- `?` and
- `[^\/`
- `.` is a me
- `(cat|dog)` is alternation within a capture grou
- `\1` is a backreference to a capture group

Pattern match operation

- To match a regular expression *regexpr* against the special variable $_ simply use one of the expressions /*regexpr*/ or m/*regexpr*/
  - This is called a pattern match
  - $_ is the target string of the pattern match
- In a scal ’’)
  depen


```
if (/\Ahttps
    ... }

if (m/\Ahttps?:\/\/[^\/]+\/\w+\/(cat|dog)\/\1/)
    ... }
```

Regular expressions: Characters

The simplest regular expression just consists of a sequence of

- alphanumberic characters and
- non-alphanumeric characters escaped by a backslash

that matches exactly this sequence of characters occurring as a substring in the targ


```
$_ = "ababcbcd
if (/cbc/) { print "Match\n"} else { print "No␣match\n" }
```

Output:


```
Match
```

```
$_ = "ababcbcdcde";
if (/dbd/) { print "Match\n"} else { print "No␣match\n" }
```

Output:

```
No match
```

Regular expressions: Special variables

- Often we do not just want to know whether a regular expession matches a target string, but retrieve additional information
  - The special variable `$-[0]` can be used to retrieve the start position of the match

    Note th
- The sp
  after th
- The special variable `$&` returns the match it


```
$_ = "ababcbcdcde";
if (/cbc/) { print "Match found at position $-[0]: $&\n"}
```

Output:

```
Match found at position 4: cbc
```

Regular expressions: Special escapes

There are various special escapes and metacharacters that match more then one character:

.	Matches any character except \n
\w	Matches a 'word' character (alphanumeric
\W	
\s	Match a whitesp
\S	Match a non-whi
\d	Match a decimal d
\D	Match a non-digi
\p{*UnicodeProperty*}	Match *UnicodeProperty* characters
\P{*UnicodeProperty*}	Match non-*UnicodeProperty* characters

Regular expressions: Unicode properties

- Each unicode character has one or more properties, for example, which script it belongs it
- \p{*UnicodeProperty*} matches all characters that have a particular property
- \P{*U*
- Exam

Ara	
ASCII	ASCII characters
Currency_Symbol	Currency symbols
Digit	Digits in all scripts
Greek	Greek characters
Han	Chinese kanxi or Japanese kanji characters
Space	Whitespace characters

See http://perldoc.perl.org/perluniprops.html for a complete list

Regular expressions: Character class

- A character class, a list of characters, special escapes, metacharacters and unicode properties enclosed in square brackets, matches any single character from within the class, for example, `[ad\t\n\-\\09]`
- One m
  for exa
- A caret
  that is, it matches any single character that is not from within the class, for example, `[^01a-z]`


```
$_ = "ababcbcdcde";
if (/[bc][b-e][^bcd]/) {
    print "Match at positions $-[0] to ",$+[0]-1,": $&\n"};
```

Output:

```
Match at positions 8 to 10: cde
```

Quantifiers

- The constructs for regular expressions that we have so far are not sufficient to match, for example, natural numbers of arbitrary size
- Also, writing a regular expressions for, say, a nine digit number would be tedious

This is ma

regex	
regexpr+	Match *regexpr* 1 or more times
regexpr?	Match *regexpr* 1 or 0 tim
regexpr{*n*}	Match *regexpr* exactl
regexpr{*n*,}	Match *regexpr* at least n
regexpr{*n*,*m*}	Match *regexpr* at least n but not more than m times

Quantifiers are greedy by default and match the longest leftmost sequence of characters possible

Quantifiers

*regexpr**	Match *regexpr* 0 or more times
regexpr+	Match *regexpr* 1 or more times
regexpr?	Match *regexpr* 1 or 0 times
regexpr{*n*}	Match *regexpr* exactly n times
regex	
regex	

Example

```
$_ = "Sorting took 10.486 seconds";
if (/\d+\.\d+/) {
    print "Match at positions $-[0] to $+[0]-1;
$_ = "E00481370";
if (/[A-Z]0{2}(\d+)/) {
    print "Match at positions $-[1] to ",$+[1]-1,": $1\n"};
```

Output:

```
Match at positions 13 to 18: 10.486
Match at positions 3 to 8: 481370
```

Quantifiers

Example:

```
$_ = "E00481370";
if (/\d+/) {
    print "Match at positions $-[0] to ",$+[0]-1,": $&\n";}
```

Output:

```
Match at posit
```


- The reg
- As the example illustrates, the regular expression
  - matches as early as possible
  - matches as many digits as possible

    ⇝ quantifiers are greedy by default

Revision

Read

- Chapter 7: In the World of Regular Expressions
- Chapter 8: Matching with Regular Expressions

of

R. L. Sch

Learning Perl.

O'Reilly, 2011

- `http://perldoc.perl.org/perlre.html`
- `http://perldoc.perl.org/perlretut.html`
- `http://www.perlfect.com/articles/regextutor.shtml`