String Search & Pattern Matching

Wednesday, October 23, 2008
Reading: Stoughton 3.14, Kozen Chs. 7-8

CS235 Languages and Automata

Department of Computer Science
Wellesley College

Some Applications of Regular Languages

Today:

- Efficient string searching
- Pattern matching with regular expressions
  (example: Unix grep utility)

Tomorrow:

- Lexical analysis (a.k.a. scanning, tokenizing) in a compiler
  (example: ML-Lex).

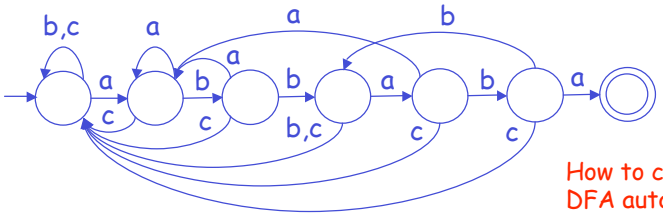
Naïve String Searching

How to search for abbaba in abbabcabbabbaba?

a	b	b	a	b	c	a	b	b	a	b	b	a	b	a
a	b	b	a	b	a									
	a	b	b	a	b	a								
		a	b	b	a	b	a							
			a	b	b	a	b	a						
				a	b	b	a	b	a					
					a	b	b	a	b	a				
						a	b	b	a	b	a			
							a	b	b	a	b	a		
								a	b	b	a	b	a	
									a	b	b	a	b	a

More Efficient String Searching

Knuth-Morris-Pratt algorithm: construct a DFA for searched-for string, and use it to do searching.



How to construct this DFA automatically?

a	b	b	a	b	c	a	b	b	a	b	b	a	b	a
a	b	b	a	b	a									
						a	b	b	a	b	a			
									a	b	b	a	b	a

Pattern Matching with Regular Expressions

Can turn any regular expression (possibly extended with complement, intersection, and difference) into a DFA and use it for string searching.

This idea is used in many systems/languages:

- **grep**: Unix utility that searches for lines in files matching a pattern. ("grep" comes from g/re/p command in the ed editor.)
- **sed**: Unix stream editor
- **awk**: text-manipulation language
- **Perl**: general-purpose programming language with built-in pattern matching
- **Java**: recent versions have support for regular expressions.

Some grep Patterns

Pattern	Matches
c	the character 'c'
.	any character except newline
[a-zA-Z0-9]	any alphanumeric character
[^d-g]	any character except lowercase d,e,f,g
\w	synonym for [a-zA-Z0-9]
\W	synonym for [^a-zA-Z0-9]
^	beginning of line
$	end of line
\<	beginning of word
\>	end of word
r_1r_2	r_1 followed by r_2, where r_1, r_2 are reg. exps.
$r_1\|r_2$	r_1 or r_2
r^*	zero or more rs, where r a reg. exp.
r^+	one or more rs
$r^?$	zero or one rs
$r\{n\}$	exactly n rs
$r\{n,\}$	n or more rs
$r\{n,m\}$	between n and m rs
(r)	r (parens for grouping)
\n	the substring previously matched by the nth parenthesized subexpression of the regular expression

Some grep Examples

As a rule, grep patterns should be double-quoted to prevent Linux from interpreting certain characters specially. (But \ is still a problem, as we'll soon see.)

grep "a.*b.*c.*d" words.txt

grep "^a.*b.*c.*d" words.txt

grep "a.*b.*c.*d$" words.txt

grep "^a.*b.*c.*d$" words.txt

grep "a.*b.*c.*d" * *(in Scowl final database)*

grep "delete[[:space:]]*(Object" *.java

grep "//.*sorted" *.java

A Powerful Combination: find With grep

Unix's `find` command enumerates all files in a directory.

In combination with `grep`, it can search all these files!

find . | xargs grep "delete[[:space:]]*(Object"

find -exec grep "delete[[:space:]]*(Object" {} \;

Escapes in Grep Patterns

grep patterns use special metacharacters that (at least in some contexts) do not stand for themselves:

? + | ( ) { } . * ^ $ \ [ ]

In order to reference the blue characters as themselves, it is necessary to escape them with a backslash. E.g.,

$ is a pattern that matches the end of line
\$ is a pattern that matches the dollar sign character
\\ is a pattern that matches the backslash character
\\\\ is a pattern that matches two backslash characters in a row

But the backslash character is also an escape character in Linux. To safely pass backslashes from Linux to grep, you should* type *two* backslashes for every backslash you wish to send to grep. E.g.

grep "\\$" searches for the dollar sign character
grep "\\\\" searches for a single backslash
grep "\\\\\\\\" searches for two backslash characters in a row

*In some, but not all cases, a single backslash will suffice.

What About the Red Metacharacters?

The red metacharacters are handled in a rather confusing way:

? + | ( ) {

In the **basic regular expressions** used by **grep**, these characters stand for themselves and must be escaped to have the metacharacter meaning. E.g.

grep "(ab)+" searches for the substring "(ab)+"
grep "(ab){2}" searches for the substring "(ab){2}"
grep "\\(ab\\)\\+" searches for any nonempty sequence of abs.
grep "\\(ab\\)\\{2\\}" searches two abs in a row.
grep "\\(.\\)\\1" searches for two consecutive occurrences of the same character

In the **extended regular expressions** used by **grep -E** and **egrep**, these characters are metacharacters and must be escaped to stand for themselves.

egrep "(ab)+" searches for any nonempty sequence of abs.
egrep "(ab){2}" searches two abs in a row.
egrep "\\(ab\\)\\+" searches for the substring "(ab)+"
grep "\\(ab\\)\\{2\\}" searches for the substring "(ab){2}"
egrep "(.)\\1" searches for two consecutive occurrences of the same character

Moral of the story: use egrep instead of grep!

Applications of Search/Pattern Matching

- Document/file search
- Virus signature matching in antivirus software
- DNA/protein analysis