Pattern Matching and Grep

Grep
Regular expressions
Nondeterministic finite state automata

Reference: Chapter 7.1, *Introduction to Computer Science*, R. Sedgewick and K. Wayne.

Pattern Matching

String search. Search for given string in a large text file.

Regular expression.

- Natural and compact way to express multiple text patterns.
- Quintessential programmer's tool.

Ex: fragile X syndrome is a common cause of mental retardation.

- Human genome contains triplet repeats of `CGG` or `AGG`, starting with `GCG` and ending with `CTG`.
- Number of repeats is variable, and correlated with syndrome.
- Use regular expression to specify pattern: `GCG(CGG|AGG)*CTG`.

More Pattern Matching Applications

Test if a string matches some pattern.

- Process natural language.
- Scan for virus signatures.
- Search for information using Google.
- Access information in digital libraries.
- Find Java file containing certain string.
- Retrieve information from Lexis/Nexis.
- Search-and-replace in a word processors.
- Filter text (SpamAssassin, NetNanny, Carnivore, AdAware).
- Validate data-entry fields (dates, email, URL, credit card).
- Search for markers in human genome using PROSITE patterns.

Parse text files.

- Compile a Java program.
- Crawl and index the Web.
- Read in data stored in some file format.
- Automatically create Java documentation from Javadoc comments.

Regular Expressions: Basic Operations

Regular expression.

- Compact notation to specify a set of strings.
- Core operations.

Operation	RE	Yes	No
Concatenation	aabaab	aabaab	every other string
Wildcard	.abba..	babbaaa aabbaab	abba abbabb
Union	aa | baab	aa baab	every other string
Closure	ab*a	aa abbbba	ε ababa
Grouping	a(a|b)aab	aaaab abaab	every other string
	(ab)*a	a ababa	ε aa

Regular Expressions: Examples

Regular expression examples.

- Notation is surprisingly expressive.

Regular Expression	Yes	No
`a* | (a*ba*ba*ba*)*` multiple of three b's	ε bbb abbbaababb	b bb baabbbaa
`a | a.*a` begins and ends with a	a aba abbaabba	ε ab ba
`.*abba.*` contains the substring abba	abba bbabbabb abbaabba	ε abb bbaaba

Using Regular Expressions

Additional operations typically added for convenience.

- Ex: `[a-e]+` is shorthand for `(a|b|c|d|e)(a|b|c|d|e)*`.

Operation	Regular Expression	Yes	No
Any single character	`..oo..oo.`	spoonfood	choochoo
One or more	`a(bc)+de`	abcbcde	ade
Character classes	`[a-e]+`	decade	Upper45
Exactly N times	`[a-e]{6}`	decade	ade
Negations	`[^aeiou]{6}`	rhythm	decade

Regular Expressions in Java

Validity checking. Is `text` in the set described by the `pattern`?

```
public class Validator {
   public static void main(String[] args) {
      String pattern = args[0];
      String text    = args[1];
      System.out.println(text.matches(pattern));
   }
}
```

```
% java Validator "0*10*10*10*" 010000011
true
                              legal Java identifier
% java Validator "[$_A-Za-z][$_A-Za-z0-9]*" ident123
true
                              legal email address (simplified)
% java Validator "[a-z]+@([a-z]+\.)+(edu|com)" rs@cs.princeton.edu
true
                              Social Security numbers
% java Validator "[0-9]{3}-[0-9]{2}-[0-9]{4}" 166-11-5655
true
```


Regular Expressions in Java

Broadly applicable programmer's tool.

- Many other languages also support extended regular expressions.
- Built into grep, awk, emacs, Perl, PHP, Python, JavaScript.

```
grep NEWLINE */*.java
```

print all lines containing `NEWLINE` which occurs in any file with a `.java` extension

```
egrep '^[qwertyuiop]*[zxcvbnm]*$' dict.txt | egrep '..........'
```

PERL. Practical Extraction and Report Language.

```
perl -pi -e 's|from|to|g' input.txt
```

replace all occurrences of `from` with `to` in the file `input.txt`

```
perl -ne 'print if /^[A-Z][A-Za-z]*$/' dict.txt
```

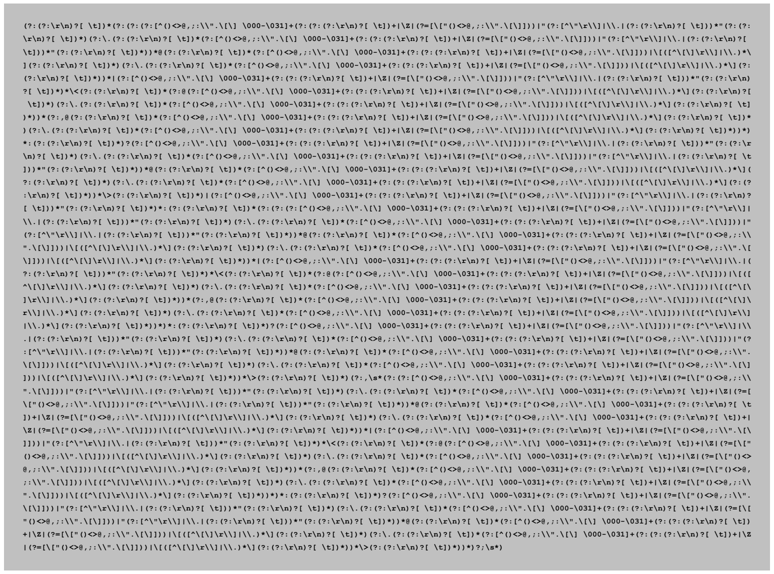
Regular Expression Caveats

Writing a RE is like writing a program.

- Need to learn syntax.
- Can be easier to write than read.

> "Sometimes you have a programming problem and it seems like the best solution is to use regular expressions; now you have two problems."

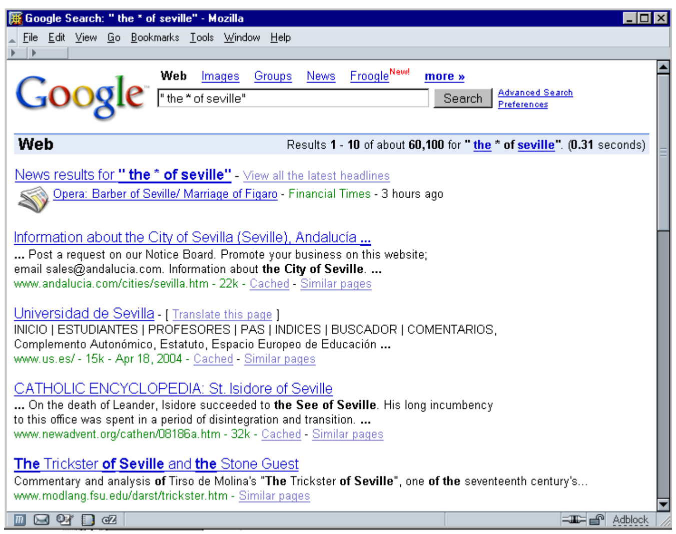
Perl RE for Valid RFC822 Email Addresses



Reference: http://www.ex-parrot.com/~pdw/Mail-RFC822-Address.html

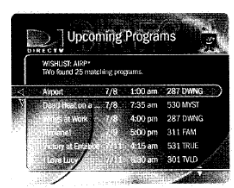
Pattern Matching in Google

Google supports * for full word wildcard and | for union.



Pattern Matching in TiVo

TiVo WishList has very limited pattern matching.



Using * in WishList Searches. To search for similar words in Keyword and Title WishList searches, use the asterisk (*) as a special symbol that replaces the endings of words. For example, the keyword *AIRP** would find shows containing "airport," "airplane," "airplanes," as well as the movie "Airplane!" To enter an asterisk, press the SLOW ( ) button as you are spelling out your keyword or title.

The asterisk can be helpful when you're looking for a range of similar words, as in the example above, or if you're just not sure how something is spelled. Pop quiz: is it "irresistible" or "irresistable?" Use the keyword *IRRESIST** and don't worry about it! Two things to note about using the asterisk:

- It can only be used at a word's end; it cannot be used to omit letters at the beginning or in the middle of a word. (For example, *AIR*NE* or **PLANE* would not work.)

Reference: page 76, Hughes DirectTV TiVo manual

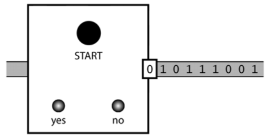
Engineering Grep

Generalized regular expression print.

- First implemented in 1973 by Ken Thompson for text-to-speech.
- Quintessential programmer's tool.

Approach to develop grep algorithm.

- Define class of abstract machines.
- Write simulator for machine.
- Write translator from REs to machines.



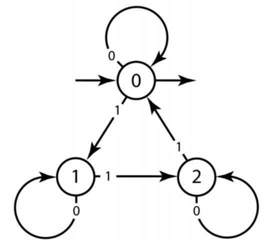
An abstract machine

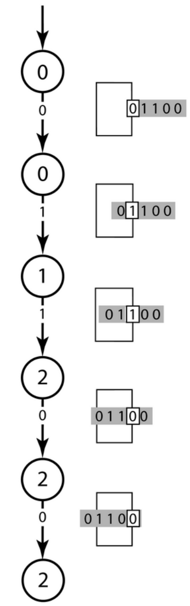
Example of essential paradigm in computer science.

- Build intermediate abstractions.
- Pick the right ones!
- Solve important practical problem.

Deterministic Finite State Automata

DFA review.



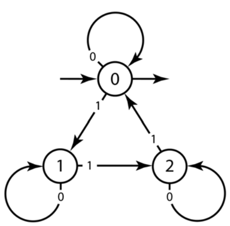


```
int pc = 0;
while (!tape.isEmpty()) {
   boolean bit = tape.read();
   if      (pc == 0) { if (bit) pc = 0; else pc = 1; }
   else if (pc == 1) { if (bit) pc = 1; else pc = 2; }
   else if (pc == 2) { if (bit) pc = 2; else pc = 0; }
}
if (pc == 0) System.out.println("accepted");
else         System.out.println("rejected");
```


Duality

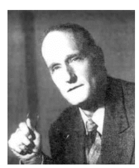
RE. Concise way to describe a set of strings.
DFA. Machine to recognize whether a given string is in a given set.

Kleene's theorem (1956): for any DFA, there exists a RE that describes the same set of strings; for any RE, there exists a DFA that recognizes the same set of strings.



```
0* | (0*10*10*10*)*
```

multiple of three 1's



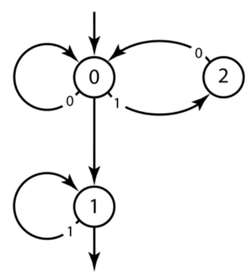
Stephen Kleene

Good news: to match RE build DFA and simulate DFA on input string.
Bad news: the DFA can be exponentially large.
Consequence: need more efficient abstract machine.

Nondeterministic Finite State Automata

NFA.

- Finite state automata.
- May have 0, 1, or more transitions for each input symbol.
- May have ε-transitions.
- Accept if any sequence of transitions leads to accept state.

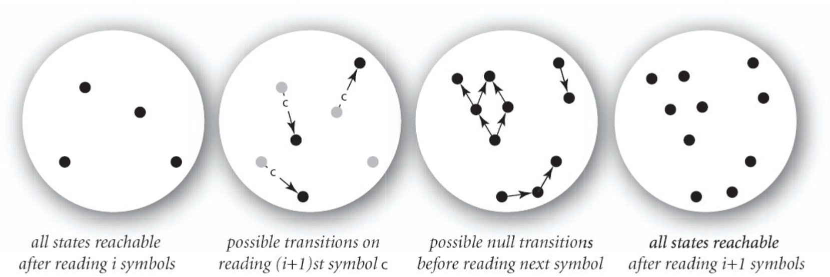


yes: 111, 00011, 101001011

no: 110, 00011011, 00110

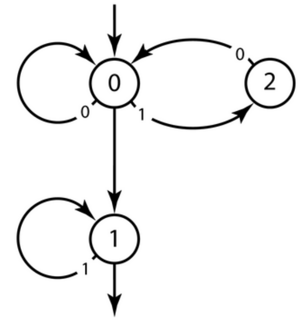
Simulating an NFA

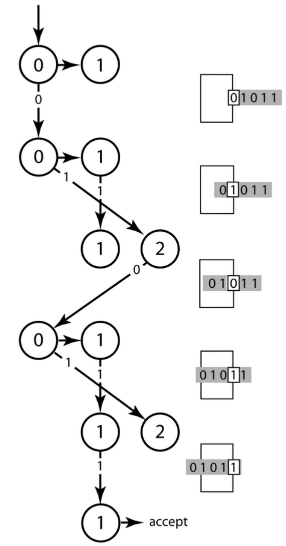
How to simulate an NFA? Maintain list of all possible states that NFA could be in after reading in the first i symbols.



One step in simulating an NFA

NFA Simulation

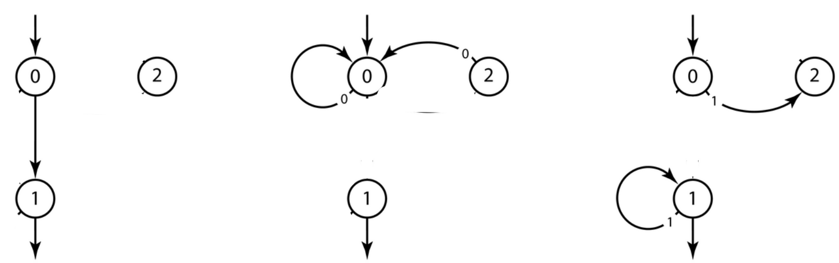




An NFA trace

NFA Representation

NFA representation. Maintain several graphs, one for each symbol in the alphabet, plus one for ε.



ε-graph

0-graph

1-graph

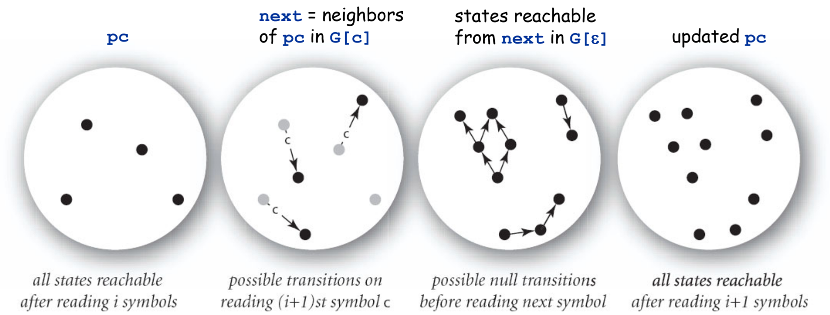
NFA: Java Implementation

```java
public class NFA {
   private int START  = 0;                  // start state
   private int ACCEPT = 1;                  // accept state
   private int N      = 2;                  // number of states
   private String ALPHABET = "01";          // RE alphabet
   private int EPS = ALPHABET.length(); // symbols in alphabet
   private Graph G[];

   public NFA(String re) {
      G = new Graph[EPS + 1];
      for (int i = 0; i <= EPS; i++)
         G[i] = new Graph();
      build(0, 1, re);
   }

   private void build(int from, int to, String re) { }
   public boolean simulate(Tape tape)              { }
```

NFA Simulation

How to simulate an NFA?

- Maintain **List** of all possible states that NFA could be in after reading in the first i symbols.
- Use **Graph** adjacency and reachability ops to update.



NFA Simulation: Java Implementation

```
public boolean simulate(Tape tape) {
   List pc = G[EPS].reachable(START);

   // simulate NFA using input from tape
   while (!tape.isEmpty()) {
      char c = tape.read();
      int  i = ALPHABET.indexOf(c);
      List next = G[i].neighbors(pc);

      pc = G[EPS].reachable(next);
   }

   while (!pc.isEmpty())
      if (pc.remove() == ACCEPT) return true;
   return false;
}
```

states reachable from start by ε-transitions

all possible states after reading in c

follow ε-transitions

check if end in an accept state

NFA Simulation Running Time

Input: Text with N characters, NFA with M transitions.

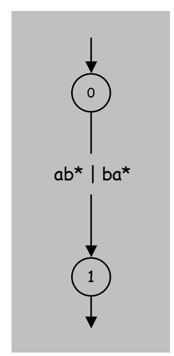
Running time. O(M N)

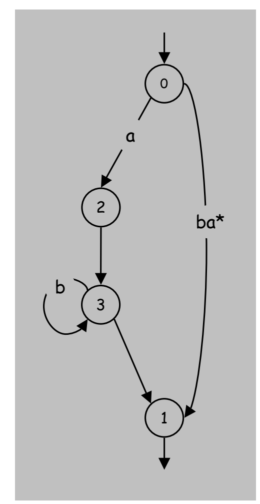
- Bottleneck = 1 graph reachability per input character.
- Can be substantially faster in practice if few ε-transitions.

Note: Easy to extend graph search to handle multiple sources.
Implicit assumption: alphabet size is a small constant.

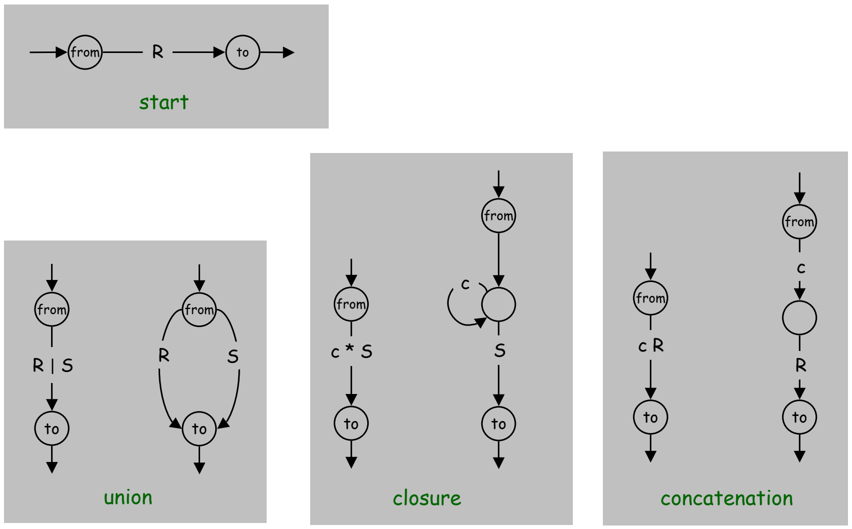
Extended NFA

Some extended NFAs.



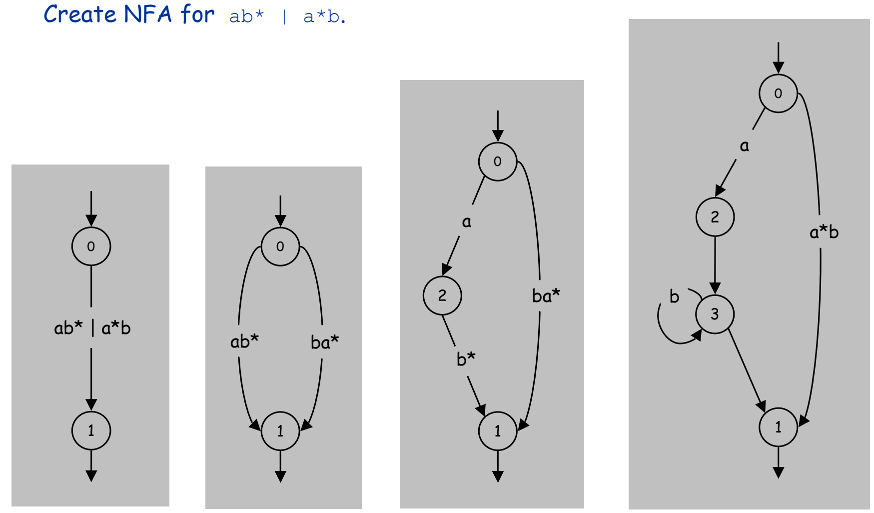


Converting from an RE to an NFA: Basic Transformations



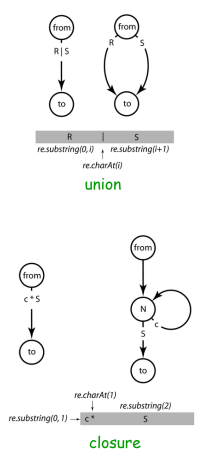
Converting from an RE to an NFA

Create NFA for `ab* | a*b`.



NFA Construction: Java Implementation

```
private void build(int from, int to, String re) {
   if (re.length() == 0) G[EPSILON].insert(from, to);

   else if (re.length() == 1) {                    single char
      char c = re.charAt(0);
      for (int i = 0; i < EPSILON; i++)
         if (c == ALPHABET.charAt(i) || c == '.')
            G[i].insert(from, to);
   }

   int or = re.indexOf('|');                       union
   else if (or > 0) {
      build(from, to, re.substring(0, or));
      build(from, to, re.substring(or + 1));
   }

   else if (re.charAt(1) == '*') {                 closure
      G[EPSILON].insert(from, N);
      build(N, N, re.substring(0, 1));
      build(N++, to, re.substring(2));
   }

   else {                                          concatenation
      build(from, N, re.substring(0, 1));
      build(N++, to, re.substring(1));
   }
}
```



Grep Running Time

Input: Text with N characters, RE with M characters.

Claim. The number of edges in the NFA is at most 2M.

- Single character: consumes 1 symbol, creates 1 edge.
- Wildcard character: consumes 1 symbol, creates 2 edges.
- Concatenation: consumes 1 symbols, creates 0 edges.
- Union: consumes 1 symbol, creates 1 edges.
- Closure: consumes one symbol, creates 2 edges.

NFA simulation: O(MN) since NFA has 2M transitions.
NFA construction: Ours is O(M^2) but not hard to make O(M).

Surprising bottom line. Worst case cost for grep is the same as for elementary string match!

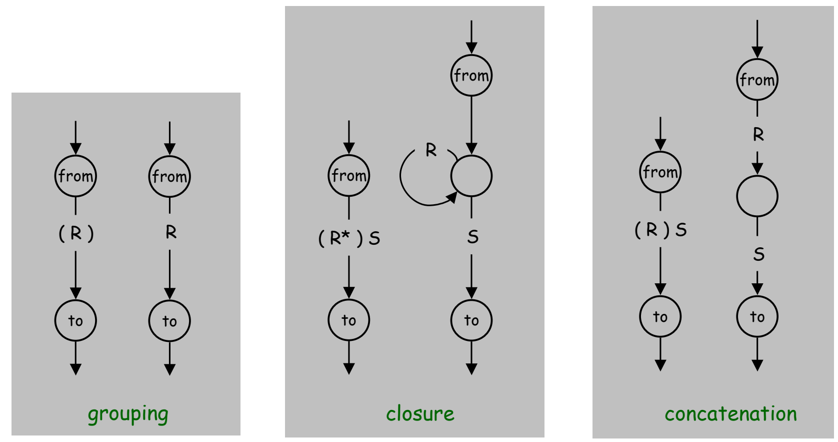
Industrial Strength Grep Implementation

To complete grep implementation.

- Parentheses.
- Documentation.
- Extend the alphabet.
- Add character classes.
- Add capturing capabilities.
- Deal with meta characters.
- Extend the closure operator.
- Error checking and recovery.
- Greedily match longest possible match.

Converting from an RE to an NFA

Transformations for parsing parentheses.



Application: Harvester

Harvesting info: Print all occurrences of `regexp` from `text` file or URL.

Pattern. Compiles RE to an NFA.
Matcher. Simulate the NFA.

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class Harvester {
   public static void main(String[] args) {
      String regexp = args[0];
      In in = new In(args[1]);
      String text = in.readAll();
      Pattern pattern = Pattern.compile(regexp);
      Matcher matcher = pattern.matcher(text);
      while (matcher.find())
         System.out.println(matcher.group());
   }
}
```


Application: Harvester

Harvesting info: Print all occurrences of `regexp` from `text` file or URL.

- Word puzzles.

```
java Harvester ".*hh.*" dictionary.txt
beachhead
highhanded
withheld
```

- Harvest email addresses for spam campaign.

```
java Harvester "[a-z]+@([a-z]+\.)+(edu|com|net|tv)" http://www.princeton.edu
mdudik@cs.princeton.edu
nailon@cs.princeton.edu
wayne@cs.princeton.edu
```

simple email validator

Application: Data File Parser

Parsing input files: Internet movie database, NCBI genome file, ...

```
LOCUS AC146846 128142 bp DNA linear HTG 13-NOV-2003
DEFINITION Ornithorhynchus anatinus clone CLM1-393H9,
ACCESSION AC146846
VERSION AC146846.2 GI:38304214
KEYWORDS HTG; HTGS_PHASE2; HTGS_DRAFT.
SOURCE Ornithorhynchus anatinus (platypus)
ORIGIN
     1 tgtatttcat ttgaccgtgc tgtttttcc cggtttttca gtacggtgtt agggagccac
    61 gtgattctgt ttgttttatg ctgccgaata gctgctcgat gaatctctgc atagacagct  // a comment
   121 gccgcaggga gaaatgacca gtttgtgatg acaaaatgta ggaaagctgt ttcttcataa
   ...
128101 ggaaatgcga cccccacgct aatgtacagc ttctttagat tg
//
```

```
String  regexp  = "[ ]*[0-9]+([actg ]*).*";
Pattern pattern = Pattern.compile(regexp);
In in = new In(filename);
String line;
while ((line = in.readLine()) != null) {
   Matcher matcher = pattern.matcher(line);
   if (matcher.find()) {      just the RE part in parentheses
      String s = matcher.group(1).replaceAll(" ", "");
      // do something with s
   }                                   replace this RE   with this string
}
```


Application: Web Crawler

Crawling the Web: Find all web page reachable from site s.

```
Queue q = new Queue();                  // queue of sites to crawl
HashSet visited = new HashSet();        // ST of visited websites
q.enqueue(s);                           // start crawl from site s
visited.add(s);
while (!q.isEmpty()) {
   String v = (String) q.dequeue();
   System.out.println(v);
   In in = new In(v);                   read in raw html
   String input = in.readAll();         http://xxx.yyy.zzz
   String regexp = "http://(\\w+\\.)*(\\w+)";
   Pattern pattern = Pattern.compile(regexp);
   Matcher matcher = pattern.matcher(input);
   while (matcher.find()) {
      String w = matcher.group();        search using regular expression
      if (!visited.contains(w)) {
            visited.add(w);
            q.enqueue(w);                if unvisited, mark as visited
                                         and put on queue
      }
   }
}
```


Algorithmic Complexity Attacks

Warning: most everyday implementations (Unix grep, Java, Perl) do not guarantee performance!

```
java Validator "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac            1.6 seconds
java Validator "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac          3.7 seconds
java Validator "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac        9.7 seconds
java Validator "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac     23.2 seconds
java Validator "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac   62.2 seconds
java Validator "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac 161.6 seconds
```

SpamAssassin regular expression.

```
java RE "[a-z]+@[a-z]+([a-z\.]+\.)+[a-z]+" spammer@x......................
```

- Takes exponential time.
- Spammer could use a pathological return email addresses to DOS a mail server running SpamAssassin.

Not-So-Regular Expressions

Back-references.

- `\1` notation matches subexpression that was matched earlier.
- Supported by most RE implementations.

```
java Harvester "^(.*)\1$"  dictionary.txt
beriberi
couscous
```

Some non-regular languages.

- All strings of the form ww for some string w: `couscous`, `beriberi`.
- All bitstring with an equal number of 0s and 1s: `10`, `01110100`.
- All Watson-Crick complemented palindromes: `atttcggaaat`.
- . . .

Remark. Pattern matching with back-references is NP-hard.

Context

Abstract machines, languages, and nondeterminism.
- Basis of the theory of computation.
- Intensively studied since the 1930s.

Compiler: a program that translates from one language to another.
- `grep`: RE ⇒ NFA.
- `javac`: Java language ⇒ Java byte code.

Abstract Machine	NFA	Computer
Pattern	Word in CFL	Word in CFL
Parser	Check if legal RE	Check if legal Java program
Compiler	Output NFA	Output machine executable
Simulator	Find match	Run program in hardware

Summary

Programmer.
- REs are a powerful pattern matching tool.
- Implement regular expressions with NFAs.

Theoretician.
- RE is a compact description of a set of strings.
- NFA is an abstract machine equivalent in power to RE.
- DFAs and REs have limitations.

You. Practical application of core CS principles.