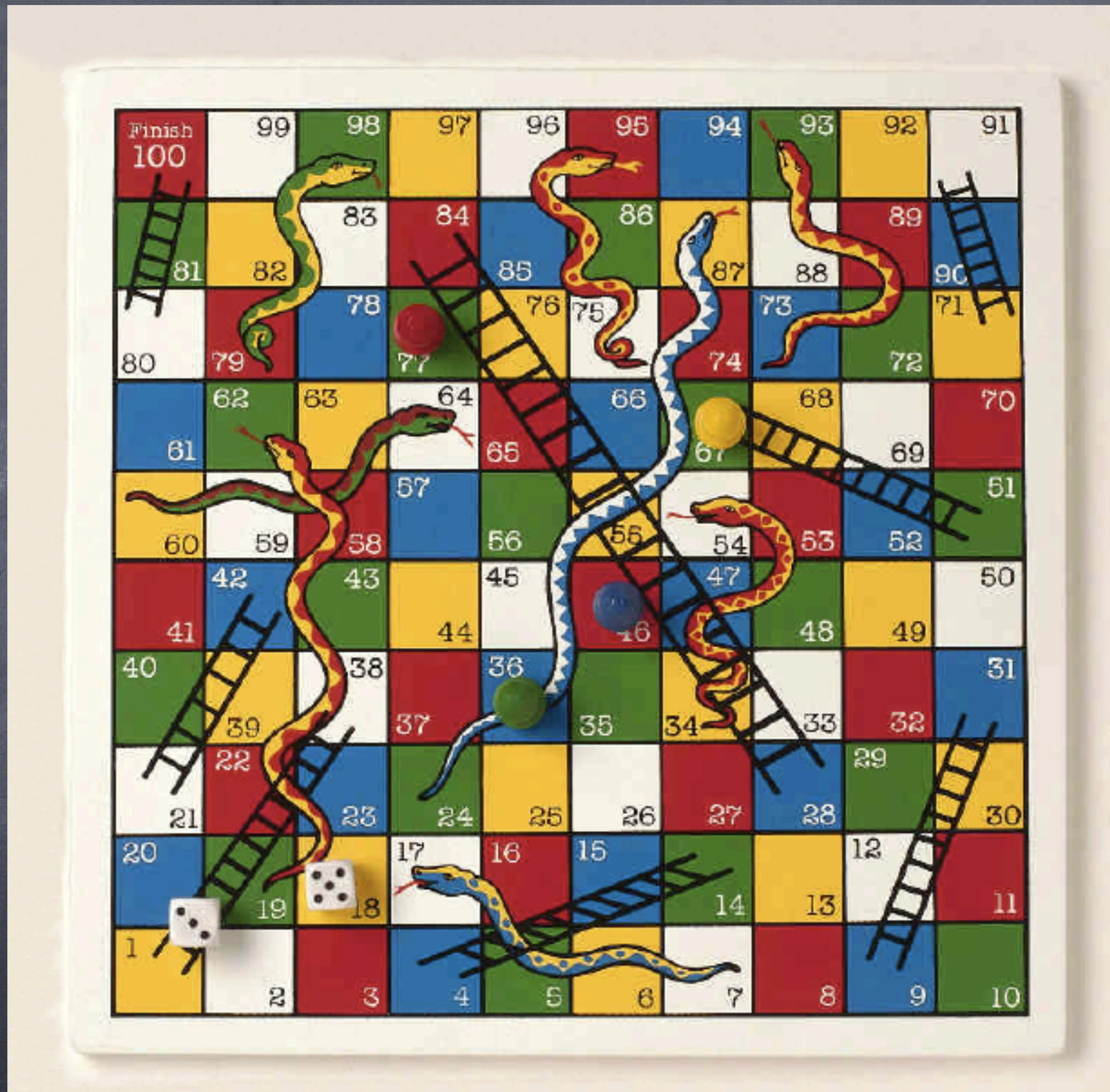
Multiple Pattern Matching with the Aho-Corasick Algorithm

Tom Spreen, Andre Van Slyke

CSC 428 (Spring 2010)


Finish 100
99 98 97 96 95 94 93 92 91
81 82 83 84 85 86 87 88 89 90
80 79 78 77 76 75 74 73 72 71
61 62 63 64 65 66 67 68 69 70
60 59 58 57 56 55 54 53 52 51
41 42 43 44 45 46 47 48 49 50
40 39 38 37 36 35 34 33 32 31
21 22 23 24 25 26 27 28 29 30
20 19 18 17 16 15 14 13 12 11
1 2 3 4 5 6 7 8 9 10

Intro: A Few Definitions

- TEXT: a (usually large) set of characters that we wish to search through (the "haystack")
- PATTERN: a smaller set of characters that we are looking for in the text (the "needle")
- DICTIONARY: a set of (distinct) patterns that we are looking for (a "handful of different needles")

Motivation

- There are many real-world cases whereby we need to search for instances of not one, but many different patterns in a given text (exact-set matching)
- Problem: large sets, long patterns and huge texts result in unacceptable (s-l-o-w-w-w-w) performance using naive methods

Motivation

- Example 1: DNA Contamination
- The Question: “Did we find Dinosaur DNA?”
- TEXT: a candidate DNA sample from a paleontological dig site
- DICTIONARY: several small snippets of human mitochondrial DNA
- http://www.dinosauria.com/jdp/misc/dna.htm

Motivation

- Example 2: Computer Virus Detection
- Question: “Is my program infected?”
- TEXT: the complete code of a suspect program (eg. Microsoft Word)
- DICTIONARY: the set of all known computer viruses which could infect the given system

Implementation

- Clearly, multiple pattern matching is important
- How do we do FAST multiple pattern searches?


CONFUSION
You're not making any sense at all.
DIY.DESPAIR.COM

Aho-Corasick Algorithm

- due to Alfred V. Aho and Margaret J. Corasick (Bell Labs)
- first published in June 1975

Aho-Corasick Algorithm

- MAIN IDEA: go through the text just ONCE, searching for all of the patterns in the dictionary at once

Aho-Corasick Algorithm

- Question: How do we examine a given text for instances of an entire dictionary, ALL AT ONCE?
- Answer: Smart pre-processing!

Aho-Corasick Algorithm

- STEP 1: Build a KEYWORD TREE **K** from the dictionary elements
- Label certain nodes of the keyword tree **K** with the index of that particular pattern in the dictionary **P** (starting at 1). These will be the NUMBERED NODES.

Aho-Corasick Algorithm

- STEP 2: Create FAILURE LINKS within the keyword tree **K**
- FAILURE LINK: a link from the longest suffix of the current pattern that also exists as a prefix in the keyword tree, to that prefix in the tree.
- THEOREM: Failure links are unique

Aho-Corasick Algorithm

- STEP 3: Using the A-C Algorithm, search the text **T** using the pre-constructed keyword tree for the dictionary **P**

Algorithm **full_AC_search**

```
l := 1;         // l : starting pos of current search in the text
c := 1;         // c : current character position in the text
w := root; // w : the node we are currently at in the tree
repeat
  while there is an edge (w, w') labeled T(c)
    begin          // w' : some child of w that fits the description
      IF (w' is a numbered node), OR
        (there is a directed path of failure links
             from w' to a numbered node)
      THEN
        report occurrence of P_i, ending at position c;
      w = w', and c = c + 1;
    end;
  w := n_w and l := c - lp(w);   // ask us about lp(w)! :-)
until c > n;
```

Running Time

- Preprocessing: O(n) time to create prefix tree and failure links, where n is the total length of the dictionary **P**
- Searching: we proceed through the text **T** exactly once, possibly reporting occurrences of **P** in **T**
- Thus, the total running time is $O(n) + O(m+k)$, where $m = |T|$ and $k = \#$ occurrences

Running Time

- **Theorem:** If **P** is a set of patterns with total length n, and **T** is a text of total length m, then one can find all occurrences of **T** in patterns from **P** in O(n) preprocessing time plus O(m+k) search time, where k is the number of occurrences found.

Aho-Corasick Algorithm

- One last real-world application:
- `grep -F` (UNIX and derivatives; search a document for a list of fixed strings) makes use of the Aho-Corasick algorithm
- if you run Mac OS X or any other -nix system, you have Aho-Corasick!

Primary Reference:

- Gusfield, Dan. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge, England: Cambridge University Press, 2005.

Questions?