Multiple Pattern Matching with the Aho-Corasick Algorithm

Tom Spreen, Andre Van Slyke
CSC 428 (Spring 2010)
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.
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`

\[
\begin{align*}
l &:= 1; \quad \text{// } l : \text{starting pos of current search in the text} \\
c &:= 1; \quad \text{// } c : \text{current character position in the text} \\
w &:= \text{root}; \quad \text{// } w : \text{the node we are currently at in the tree} \\
\end{align*}
\]

repeat
    \[
    \begin{align*}
    &\text{while there is an edge } (w, w') \text{ labeled } T(c) \\
    &\quad \text{begin} \quad \text{// } w' : \text{some child of } w \text{ that fits the description} \\
    &\quad \quad \text{IF } (w' \text{ is a numbered node}), \text{ OR} \\
    &\quad \quad \quad \text{(there is a directed path of failure links} \\
    &\quad \quad \quad \quad \text{from } w' \text{ to a numbered node)} \\
    &\quad \quad \quad \quad \text{THEN} \\
    &\quad \quad \quad \quad \quad \text{report occurrence of } P_i, \text{ ending at position } c; \\
    &\quad \quad \quad \quad \quad w = w', \text{ and } c = c + 1; \\
    &\quad \quad \text{end;}
    \end{align*}
\]
\[
\begin{align*}
w &:= n_w \text{ and } l := c - \text{lp}(w); \quad \text{// ask us about } \text{lp}(w)! \ :-) \\
\text{until } c > n;
\end{align*}
\]
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.
**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:

Questions?