Aho-Corasick (part 2)

Take a keyword tree:

We can match against a pattern in linear time for much the same reason as with Knuth-Morris-Pratt. But can we construct the failure links in linear time? Here is the algorithm (claim: it's linear time):

("nv" denotes that a failure link maps node v to some other node nv)

find failure links of node v:

\[
v' = \text{parent of } v
\]

while \( \exists \text{nv and } v' \neq \text{Root} \)

\[
\text{if } \exists \text{nv} \rightarrow w \text{ labeled same as } v' \rightarrow w
\]

\[
\text{then } n_v = w
\]

\[
v' = n_v
\]

It's not clear that this is linear time. To show that it is, we consider a single path (equivalently: pattern):

Definitions:

\[
L_p(v) = \text{length of label of } v
\]

\[
W_v = \# \text{ of failure links followed to find failure link for } v
\]

\[
L_p(v) \leq L_p(v') + 1 - W_v
\]

\[
W_v \leq L_p(v') - L_p(v) + 1
\]

\[
W_v' \leq L_p(v'') - L_p(v') + 1
\]

\[
\vdots
\]

\[
W_{tot} \leq \text{length of path} - L_p(v) \leq \text{length of path}
\]
So far the discussion of Aho-Corasick has assumed that no pattern can be a substring of another pattern. Take this pattern set and the corresponding keyword tree:

1. potato
2. tatter
3. pot
4. at

But, for example, matching "potato" misses an intermediate match for "at"

Solution: if you reach a node that has an outgoing path of failure links leading through one or more numbered nodes, then report the matches corresponding to all such numbered nodes.

Aho-Corasick for matching with don't-cares

Say that "N" = "don't care which character"

\[ P = ATGANNNGNCTGNGCNGG \]

Make keywords out of all maximal substrings that don't include N

\[ P = \begin{array}{c|c|c|c|c|c|c} A & T & G & A & N & N & G \\ \hline P1 & P2 & P3 & P4 \end{array} \]

Create a keyword tree including all such substrings and match against text using "match array" M:

T: 
M: 

M is initialized to all 0s. When pattern PN matches at offset i in the text, and if the final character of pattern PN appears at offset o within P, then update \( M[i - o + 1]++ \). A match of P occurs at all k where \( M[k] = \# \) patterns.
Introduction to Suffix Trees

Aho-Corasick can only tell us when a pattern matches in its entirety. It cannot tell us whether a substring of the pattern matches. Suffix trees can.

Also, Aho-Corasick spends $O(n)$ time in preprocessing and $O(m)$ time in matching. Also, Suffix trees spend $O(m)$ time in preprocessing and $O(n)$ time in matching, which can be desirable if many patterns are being matched against the same text, or if the patterns are not known at preprocessing time.

A Suffix Tree is similar to a keyword tree for all suffixes.

Problems:
- Tree seems to be quadratic space!
- Construction must therefore be quadratic time!

The cool thing about suffix trees is that they are linear space and can be built in linear time.