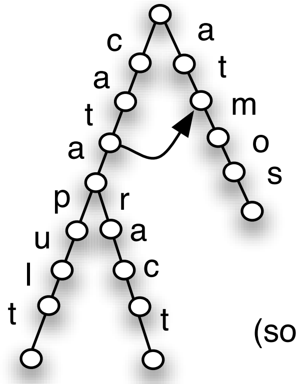
Aho-Corasick (part 2)

Take a keyword tree:



(some failure transitions omitted)

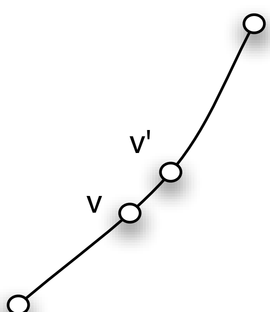
We can match against a pattern in linear time for much the same reason as with Knuth-Morris-Pratt
But can we construct the failure links in linear time? Here is the algorithm (claim: it's linear time):

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

find failure links of node v:

    v' = parent of v
    while ∃ n_v' and n_v' ≠ Root
        if ∃ n_v' → w labeled same as v' → w
            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:

$L_p(v)$ = length of label of v

W_v = # of failure links followed to find failure link for v

$L_p(v) \le L_p(v')+1-W_v$

$W_v \le \cancel{L_p(v')}-L_p(v)+1$

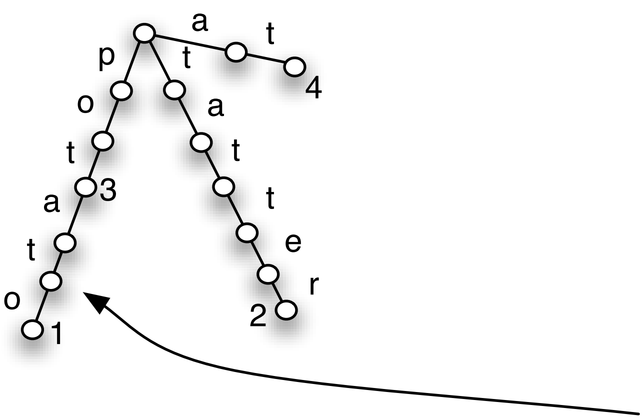
$W_{v'} \le \cancel{L_p(v'')}-\cancel{L_p(v')}+1$

$+ \quad \vdots$

$W_{tot} \le$ length of path $- L_p(v) \le$ length of path

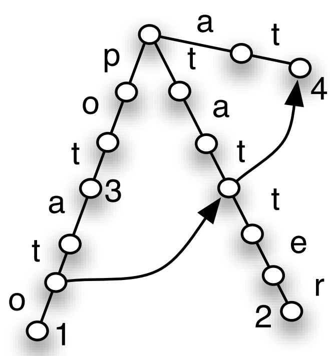
So far the discussion of Aho-Corasick has assumed that no pattern can be a substring of another pattern. Take this pattern set and the corresponding keyword tree:

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 correpsonding to all such numbered nodes.



Aho-Corasick for matching with don't-cares

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

P = A T G A N N N G C N T C A N G G

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

P = [A T G A] N N N [G C] N [T C A] N [G G]

P1 = A T G A, P2 = G C, P3 = T C A, P4 = G G

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

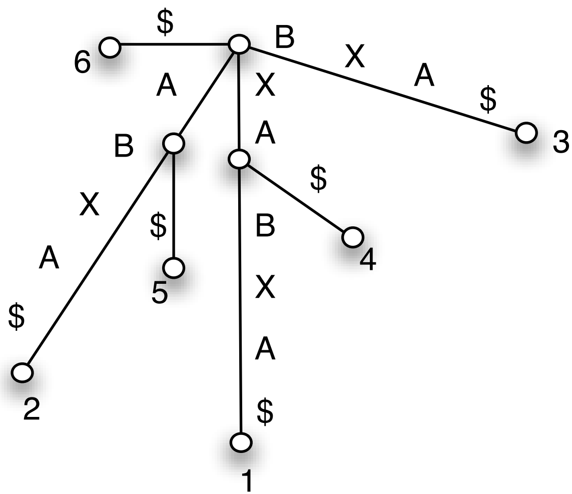
Aho-Corasick can only tell us when a pattern matches in its entirety. It cannot tell us whether a *substring* of the pattern matches. Suffix trees can.

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.

X A B X A $
1 2 3 4 5 6



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

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