Lecture 5

Multiple keyword (pattern) matching

Bibliographic search in libraries: you give several keywords, and the computer finds all entries containing the keywords.

Problem: Given text T and patterns $P_1, P_2, \ldots, P_k$, find all occurrences of the patterns in the text.

Search for each keyword – very slow.
One scan of the database T?

Aho-Corasick Algorithm

(1) AC scans text from left to right as in KMP.

(2) When there is a mismatch, AC tries to obtain the longest shift as in KMP.

- For (1), KMP had a line of states for one pattern. AC needs a tree of states for many patterns.
- For (2), KMP had the failure function. AC needs an extension of the failure function

Pattern matching machine for aabbaab (failure function)

```
P: aabbaab
f: 0100123
```

Extend to many patterns, e.g., (1) he, (2) she, (3) his, (4) hers.

- Let P' be the prefix of a pattern such that i is the state number corresponding to P'.
- $f(i)$ is the state number corresponding to the longest prefix of some pattern which is a proper suffix of P'.

Failure function

```
i: 0 1 2 3 4 5 6 7 8 9
f: 0 0 0 0 1 2 0 3 0 3
```

Output function

```
2 - he (1)
5 - she, he (1,2)
7 - his (3)
9 - hers (4)
```

Another example: cacbaa, acb, aba, acbab, ccbab

Failure function

```
i: 0 1 2 3 4  5 6 7 8 9 10 11 12 13 14 15 16 17
f: 0 0 7 8 9 12 7 0 1 0  0  7  7 10  1  0  7 10
```

Output function

```
6 - cacbaa
9 - acb
11 - aba
13 - acbab
17 - ccbab
```


Searching Text

```
procedure AC
  s = 0 (current state of pattern matching machine)
  i = 1 (current text position)
  while i <= n do
    if current text char has a match then
      go to next state s'
      i = i+1
      if output(s') is not empty
      then print output(s') fi
    else
      if s == 0 then i = i+1
      else s = f(s) fi
    fi
  od
```

First example with $T = ushers$

```
  u s h e r s
0 0 3 4 5 8 9
      2
```

Constructing the tree:

1. Initially, only state 0
2. For each pattern, search down the tree and create states as necessary. The last state gets an output value.

Example with he, she, his, hers. See p335 of AC.

Let $m = |P_1| + ... + |P_k|$. The number of states (nodes) in the tree is $O(m)$.

Implementation

- 2D array $g(s, x)$ for each state s and each char x. Space: $|\Sigma|$ size array for each node, $O(m|\Sigma|)$ in total. Branching: O(1) time
- A linear list for children. Space: $|\Sigma|$ for each node, but better than 2D arrary for large alphabets. Branching: $O(|\Sigma|)$ time
- Binary search trees. Space: bounded by the number of children for each node, $O(m)$ in total. Branching: $O(\log |\Sigma|)$ time.

For implementation, use 1 or 2.

For analysis, use 1 or 3.

Computing Failure Function

For a state s, depth(s) is the length of the path from state 0 to s. That is, depth(0) = 0, depth(1) = 1, ...

Compute f from smallest depth to largest.

1. For all states s such that depth(s) ≤ 1, $f(s) = 0$.
2. Suppose f has been computed for all states of depth $< d$. For each state s of depth d, let s' be the previous state and let x be the character on edge (s', s). Let $r' = f(s')$. If r' has x-branch (leading to state r), then $f(s) = r$. When r' doesn't have x-branch, if $r' = 0$ then $f(s) = 0$; otherwise, assign $s' = r'$ and repeat.

Computing Output Function

During the computation of f, we also update output function: Once we determine $f(s) = s'$, new output(s) is the union of old output(s) and output(s').

Example for failure and output: he, she, his, hers

Time Complexity

Array implementation:

- Constructing tree: The number of states is $O(m)$. $O(|\Sigma|m)$ time.
- Computing f: $O(m)$ time – similar argument to KMP
- Search: $O(n)$ time - similar argument to KMP
- Total time $O(|\Sigma|m + n)$.

Binary search tree implementation:

- Constructing tree: $O(m \log |\Sigma|)$ time
- Computing f: $O(m \log |\Sigma|)$ time
- Search: $O(n \log |\Sigma|)$ time
- Total time $O((m + n) \log |\Sigma|)$ time

Commentz-Walter Algorithm

Boyer-Moore approach for multiple-keyword matching

Commentz-Walter (one person) automaton for cacbaa, acb, aba, acbab, ccbab

Failure function is harder to compute. (If interested, see Aho's survey paper for details)

Worst case $O(mn)$ time.

Average case: depends on the smallest pattern.