CS 663: Pattern Matching Algorithms

Scribe: Mohit Dia

Lecture#5
10/14/2010

Pattern Renaming

The Naive algorithm, KMP and the Wave method are inherently serial. So the motivation behind Pattern Renaming is Parallel Pattern Matching. We want a method that uses "local" matches.

The Renaming idea which was given by **Karp-Miller-Rosenberg** in 1972 is -

Rename pairs of symbols as a single symbol consistently.

Renaming:

$P = P_1\ P_2\ P_3 .... P_{m-1}\ P_m$

Let $P^0 = P$

Consider : $<P_1, P_2> <P_3,P_4> ....<P_{2i-1}, P_{2i}>...<P_{m-1},P_m>$

There are m/2 such pairs. Number the different pairs by different numbers from $\Sigma_1 = \{1,....,m/2\}$

Create the new pattern as:

$P^1 = P^1{}_1 P^1{}_2 P^1{}_3 ... P^1{}_{m/2}$

where $P^1{}_i\ \varepsilon\ \Sigma_1$
$i = 1,...,m/2$

Now rename all pairs $<P^1{}_{2i-1}, P^1{}_{2i}>$
$i = 1,....,m/2^2$

by elements from $\Sigma_2 = \{1,....,m/2^2\}$

....

Proceed for log m iterations where in iteration i+1:

Rename all pairs $<P^i{}_{2j-1}, P^i{}_{2j}>$
$j = 1,...., m/2^i$

by elements from $\Sigma_{i+1} = \{1,....,m/2^{i+1}\}$

After iteration log m: Pattern is reduced to a single symbol "1"
Time : Renaming can be done in linear time at every step by radix sorting the pairs.

Total Pattern preprocessing time: $\sum_{i=0}^{logm} m/2^i$ = O(m)

Text Processing:

We would like a similar renaming in text as we did in pattern BUT we do not know where the pattern starts.
Unlike pattern, we need to rename at every location.

We have log m steps.

At step j:

For i = 1 to n

if $<t^{j-1}_{i}, t^{j-1}_{i+2^{j}-1}>$ is one of the pairs that was renamed in pattern step j, then t^{j}_{i} is the name of that pair in P^j.

otherwise t^{j}_{i} <- B

end

After step log m:

There is an occurence of P in location i of T iff

$t^{\log m}_{i} = 1$

EXAMPLE:

T = abababab ccababca

P = babc

Pattern Renaming:

Step 1: <b,a> <b,c>
P^1 = 1 2

Step 2: <1,2>

$P^2 = 1$

Text Scanning:

Step 1:
<ab><ba><ab><ba><ab><ba><ab><bc><cc><ca><ab><ba><ab><bc><ca>

T^1 = B 1 B 1 B 1 B 2 B B B 1 B 2 B

Step 2:
<BB><11><BB><11><BB><12><BB><2B><BB><B1><BB><12><BB>

T^2 = B B B B B 1 B B B B B 1 B

So the pattern occurs at positions 6 and 12 in the text.

Time for Text scanning:

Verifying if a pair <x,y> of text is one of the pattern pairs can be done in time O(log m).

Now, this is done for every text location. So the time is O(nlog m).

So, for each of the log m steps the time is $O(n \log^2 m)$; which can be done by n processors in time $O(\log^2 m)$

But this can be done in better time serially:

1. In time O(nlog m):

    Convert pattern alphabet to {1,2,....,m}
    Convert text alphabet to {1,2,.......,m,B}

2. At every one of the log m steps:

    Radix sort text and pattern pairs:
    Time : O(n+m)
    Then merge:
    Time: O(n+m)

Total time per step: O(n)

Total algorithm time (for log m steps):
O(nlog m)