Algorithms Seminar Session 1

Scribe: Sichen Zhong

8/29/2014

Prefix Matching Problem

Simple Version

Given a set of n strings, order the strings from top to bottom so there is maximum overlap starting from left to right.

ex.

$ABBA$
$ACBA$
$AZBA$
$ABCD$

- A matches with A, but B does not match with C, so number matchings between 1st and 2nd line is 1.
- A matches A but C does not match with Z, so number of matchings between 2nd and 3rd line is 1.
- A matches A but Z does not match with B, so number of matchings between 3rd and 4th line is 1.
-Total number of matchings is 1 + 1 + 1 = 3

However, we can do better if we shift ABCD to line 2.

$ABBA$
$ABCD$
$ACBA$
$AZBA$

- Total number of matchings is now 2 + 1 + 1 = 4.

Question 1
Given n strings, sorting maximizes the total matching.

Proof: We group the n strings into blocks of matching characters and recur within each block. For example, if we have the following strings

$ABCD$
$ABRT$
$BBUI$
$BAIO$
$CDFK$
$CRTS$

Then by matching the first character, we have 3 blocks, namely a block with the first 2 strings, a block with the middle 2 strings, and a third block with the last 2 strings. We then recur in each block and look at the second character. Repeat the process. This gives us an ordering of the n strings in some lexicographic order. Note that if our n strings are in lexicographic order, then we cannot get a higher matching score since the total matching score never increases if any character of a string we move is shifted out of its own block. Since sorting gives us an ordering of the n strings in lexicographic order, it follows that sorting will give us a maximum matching.

Difficult Version

Given a set of n strings, order the strings from top to bottom so there is maximum overlap starting from left to right. In addition, we are now allowed to reverse individual strings to find the maximum overlap (Total strings $2n$).

Open Question 1
What is the Complexity Class of the above problem?

Open Question 2
What are some approximation algorithms to the difficult version of the problem?

The following are some approximation algorithms determined to be bad:

Algorithm 1
-Given n strings, find the maximum matching value using sorting -Now reverse each string and calculate the total matching again.

ex.

AB
BA

-Using the above algorithm, the maximum matching is 0.
-Reversing each string, we still get a maximum matching of 0.
-However, the optimal is to reverse the 2nd string, and get a score of 2.
-Hence, $\frac{OPT}{APX} = \frac{2}{0}$, and $\frac{OPT}{APX}$ can be arbitrarily bad.

Algorithm 2: Greedy Approach

ex.

$ABC - - - ZZ$
$ABC - - - ZZ$
$BCD - - - ZZ$
$BCD - - - ZZ$
$CDE - - - ZZ$
$CDE - - - ZZ$

(The dashed lines represent arbitrary letters which do not match with the string immediately following it.)

- Optimal value is to flip each string. This gives us a total score of 2 + 2 + 2 + 2 + 2 = 10 - However, if we were to use a Greedy approach, We would get a value of 3 + 0 + 3 + 0 + 3 = 9 - Hence, in the above example, $\frac{OPT}{APX} = \frac{10}{9}$.

However, the above example can be generalized to make Greedy arbitrarily bad, where $\frac{OPT}{APX}$ converges to infinity. Consider the following example:

ex.

123456789
123456780
193456780
193456780
193456780
...
...

Then Greedy will first find the 2 strings that has the most number of matchings, namely the first 2 strings. However, the rest of the strings can be constructed such that there is

a very small number of prefix matchings, but when each is reversed, the number of prefix matchings can be very large (depending on the string length). This can make $\frac{OPT}{APX} \to \infty$ as string length increases.