CS260 Design and Analysis of Algorithms
Project Proposal
Approximate String Matching

Gang Liao[*] Fatima Zohra Smaili[†] Wentao Hu[‡] Guangming Zang[§]
Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division
King Abdullah University of Science and Technology (KAUST), Saudi Arabia

September 2014

1 Introduction

String matching algorithms are an important class of algorithms in Computer Science used to solve some famous problems mainly DNA strings matching, text processing, spell checking, spam filtering. The idea behind them is to quickly find the first or all occurrences of a string in a text. In other words, given a text string T and a pattern P, we need to find a quick way to find whether there is any occurrence in P and where it appears in case it exists.

A slightly different but more interesting problem is approximate string matching problem, which is the problem we chose to work on for our project. For the approximate string matching problem we look for a substring that is similar to pattern P in text T. The word similar here refers to a string that needs a minimum number of operations (insertion, deletion and substitution) to be converted to P. This minimum number of operations is what we refer to as the edit distance. In the following sections, we will try to give a brief overview on the two approximate string matching algorithms we chose to work with.

2 Methodology

Approximate string matching is one of the main problems in classical algorithms, with applications to text searching, computational biology, pattern recogintion, etc. Many algorithms have been presented that improve approximate string matching, for instance [1–6]. We decide to implement two of them and compare them via the time and space complexity.

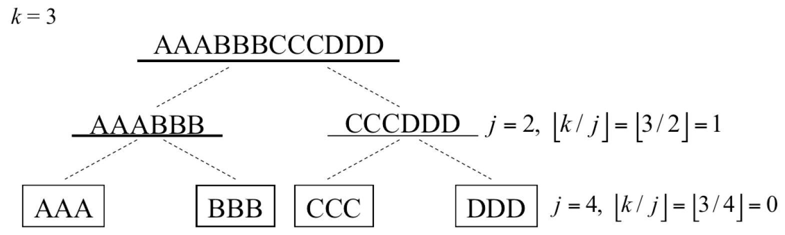
[*]liao.gang@kaust.edu.sa
[†]fatimazohra.smaili@kaust.edu.sa
[‡]wentao.hu@kaust.edu.sa
[§]guangming.zang@kaust.edu.sa

2.1 Very fast and simple approximate string matching

In 1999, G. Navarro and R. Baeza-Yates presented a very fast and simple approximate string matching [2]. The original idea of this algorithm was presented in [7]. This algorithm is based upon the following lemma:

Lemma 1 *Let T and P be two strings. Let P be divided into j pieces $p_1, p_2, \ldots, p_j$. If $ed(T, P) \leq k$, then there exists at least one p_i and a substring S in T such that $ed(S, p_i) \leq \lfloor \frac{k}{j} \rfloor$.*

We let $j = k + 1$, in this case, if $ed(T, P) \leq k$, then at least one p_i occurs in T exactly. If, in a certain window, we find an exact matching of a p_i inside the window, we use the dynamic programming approach to determine whether there exists an approximate matching of P allowing k errors in this window. After determining the occurrences of exact matching of small pieces, we start to determine the occurrences of larger piece of P in T.



$k = 3$
AAABBBCCCDDD
AAABBB
CCCDDD
$j = 2,\ \lfloor k/j \rfloor = \lfloor 3/2 \rfloor = 1$
AAA
BBB
CCC
DDD
$j = 4,\ \lfloor k/j \rfloor = \lfloor 3/4 \rfloor = 0$

2.2 Bit-parallel approximate string matching

This algorithm is based on "Fast text searching with errors" [6]. There are three operation which are insertion, deletion and substitution. It uses a new operation transposition (It is transposing two adjacent characters in the string). We will use those operations to solve the approximate string matching problem under bit-parallel.

To solve our approximate string matching problem, we use a table, called $R^k[n, m]$.

$$R^k[i, j] = \begin{cases} 1 & if\ there\ exists\ a\ suffix\ A\ of\ T_{1,i}\ such\ that\ d(A, P_{1,j}) \leq k. \\ 0 & otherwise \end{cases}$$

where $1 \leq i \leq n, 1 \leq j \leq m.$

3 Project Timeline

3.1 September

(1) Review and research on the topic and its applications

(2) Review specific papers on the two algorithms chosen

(3) Task dispatching among team members

3.2 October

(1) Implementation of the algorithms

(2) Preparation of Midterm Presentation

(3) Preparation of Midterm Report

3.3 November

(1) Evaluation of the implemented algorithms on different case studies

(2) Analysis of time and space complexity of the algorithms

(3) Synthesis and comparison of the algorithms

(4) Work on improving the existing algorithms as future work

(5) Preparation of Final Presentation and Report

References

[1] E. Ukkonen, "Finding approximate patterns in strings," *Journal of algorithms*, vol. 6, no. 1, pp. 132–137, 1985.

[2] G. Navarro and R. Baeza-Yates, "Very fast and simple approximate string matching," *Information Processing Letters*, vol. 72, no. 1, pp. 65–70, 1999.

[3] G. M. Landau and U. Vishkin, "Fast parallel and serial approximate string matching," *Journal of algorithms*, vol. 10, no. 2, pp. 157–169, 1989.

[4] R. A. Baeza-Yates and C. H. Perleberg, "Fast and practical approximate string matching," in *Combinatorial Pattern Matching.* Springer, 1992, pp. 185–192.

[5] R. Baeza-Yates and G. Navarro, "Faster approximate string matching," *Algorithmica*, vol. 23, no. 2, pp. 127–158, 1999.

[6] H. Hyyrö, "Bit-parallel approximate string matching algorithms with transposition," in *String Processing and Information Retrieval.* Springer, 2003, pp. 95–107.

[7] S. Wu and U. Manber, *Fast text searching with errors.* University of Arizona, Department of Computer Science, 1991.