The Efficiency of Discrete Convolution Method on Solving Exact String Matching Problem

K. W. Hou* (侯冠維) and R. C. T. Lee**(李家同)
*Department of Electrical Engineering,
**Department of Computer Science
National Tsing Hua University
s9961702@m99.nthu.edu.tw

Abstract

In this paper, we first derived an equation which can approach the probability of appearing of a pattern string in a text string. From this equation, we see that the probability that a pattern string appears in a text string reduces to 0 quickly as the length of the pattern string increases. Because of this observation, we introduce an algorithm based on the discrete convolution method with an early termination. The algorithm stops as soon as it discovers that a prefix of the pattern string does not appear in the text string. In this paper, we show that the discrete convolution method with an early termination is quite efficient to solve exact string matching problem.

1 Introduction

The exact string matching problem considered here is to find the occurrences of a pattern string $P = p_1p_2\ldots p_m$ in a text string $T = t_1t_2\ldots t_n$, where string P and T are both generated from a finite alphabet Σ whose size is σ, and $n \geq m$.

In the following, we first introduce one of those methods, called discrete convolution method [1]. In this paper, we introduce an algorithm based on the discrete convolution method with an early termination, and show that it is an efficient way to solve exact string matching problem.

2 Discrete Convolution Method

It is quite interesting that convolution was originally widely used in the areas of communication. It was first presented in [1] that we can use it to solve exact string matching problem. In this section, we first show the definition of discrete convolution of strings, and then we introduce that how can we apply it to solve exact string matching problem.

Definition 1: Discrete Convolution of Strings

Given two strings $X = x_1x_2...x_n$ and $Y = y_1y_2...y_m$, where x_i and y_j are characters from a finite alphabet Σ whose size is σ. The result of discrete convolution of X and Y is $Z = z_2z_3\ldots z_{n+m}$.

$$Z_k = \sum_{i+j=k} c(x_i, y_j)$$

$$c(x_i, y_j) = \begin{cases} 1, & \text{if } x_i = y_j \\ 0, & \text{otherwise} \end{cases}$$

Given a text string $T = t_1t_2\ldots t_n$ and a pattern string $P = p_1p_2\ldots p_m$, where $n \geq m$. In order to use discrete convolution to solve the exact string matching problem, we need to reverse the pattern string P to a string $P' = p'_1p'_2\ldots p'_m = p_m\ldots p_2p_1$. And then by applying discrete convolution on string T and P', we have the following results.

$z_2 = c(t_1, p'_1)$
$z_3 = c(t_1, p'_2) + c(t_2, p'_1)$
…
$z_k = c(t_{k-m}, p'_m) + c(t_{m-k+1}, p'_{m-1}) + \ldots + c(t_{k-1}, p'_1)$
…
$z_{m+n} = c(t_n, p'_m)$

Figure 1 to 4 illustrates the meaning of the above equations. Notes that because $P' = p'_1p'_2\ldots p'_m = p_m\ldots p_2p_1$, the index number of P are reversed, i.e. $P = p'_mp'_{m-1}\ldots p'_1$.

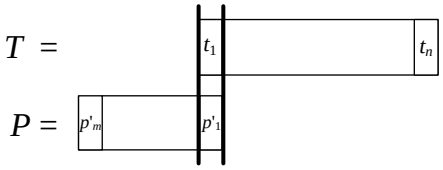

Figure 1: Deriving of $z_2 = c(t_1, p'_1)$

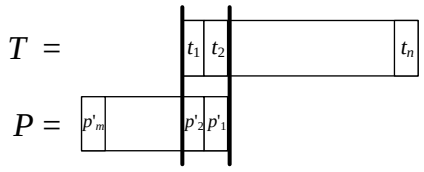

Figure 2: Deriving of $z_3 = c(t_1, p'_2) + c(t_2, p'_1)$

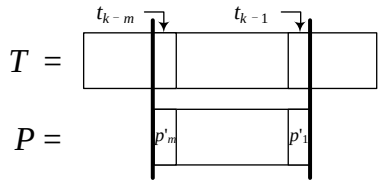

Figure 3: Deriving of $z_k = c(t_{k-m}, p'_m) + c(t_{m-k+1}, p'_{m-1}) + \ldots + c(t_{k-1}, p'_1)$

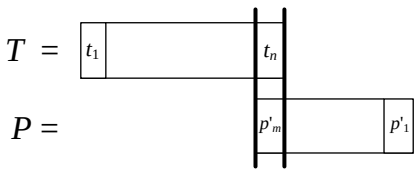

Figure 4: Deriving of $z_{m+n} = c(t_n, p'_m)$

From the results, we can see the following results.

Case 1: For $m + 1 \le k \le n + 1$, if z_k is equal to m, there exists an exact string matching of P at position $k - m$ of T.

Case 2: For $2 \le k < m + 1$ or $n + 1 < k \le m + n$, if z_k is equal to $k - 1$, there exists a suffix of P which is equal to a prefix of T.

Case 3: For $n + 1 < k \le m + n$, if z_k is equal to $m + n - k + 1$, there exists a prefix of P which is equal to a suffix of T.

Therefore, discrete convolution can be used to solve exact string matching problems.

Example 1:

Let $T = acgacgta$ and $P = cgac$. First, we reverse the pattern string $P = cgac$ to $P' = cagc$. By applying discrete convolution of strings on strings T and P', we have the following results.

$z_2 = c(a, c) = 0$
$z_3 = c(a, a) + c(c, c) = 2$
$z_4 = c(a, g) + c(c, a) + c(g, c) = 0$
$z_5 = c(a, c) + c(c, g) + c(g, a) + c(a, c) = 0$
$z_6 = c(c, c) + c(g, g) + c(a, a) + c(c, c) = 4$
$z_7 = c(g, c) + c(a, g) + c(c, a) + c(g, c) = 0$
$z_8 = c(a, c) + c(c, g) + c(g, a) + c(t, c) = 0$
$z_9 = c(c, c) + c(g, g) + c(t, a) + c(a, c) = 2$
$z_{10} = c(g, c) + c(t, g) + c(a, a) = 1$
$z_{11} = c(t, c) + c(a, g) = 0$
$z_{12} = c(a, c) = 0$

The discrete convolution between T and P' can be viewed in a graphical way. Before introducing this graphical way, we define a vector which is called incidence vector.

Definition 2: Incidence Vector
Given a string $S = s_1 s_2 \ldots s_n$ and a character x, the incidence vector of x on S is $IV(x, S) = (iv_1, iv_2, \ldots, iv_n)$, where $iv_i = c(x, s_i)$.

Thus, for $T = acgacgta$ and $P = cgac$, $IV(a, T) = (10010001)$, $IV(c, T) = (01001000)$, $IV(g, T) = (00100100)$, and $IV(t, T) = (00000010)$.

Figure 5 to 8 show the discrete convolution between T and P' in a graphical way.

T	*a*	*c*	*g*	*a*	*c*	*g*	*t*	*a*
P'					*c*	*a*	*g*	*c*
IV(c, T)	0	1	0	0	1	0	0	0
Z	0	0	0	0	1	0	0	1	0	0	0

Figure 5: Step 1

T	*a*	*c*	*g*	*a*	*c*	*g*	*t*	*a*
P'					*c*	*a*	*g*	*c*
IV(c, T)	0	1	0	0	1	0	0	0
IV(g, T)	0	0	1	0	0	1	0	0
Z	0	0	0	0	2	0	0	2	0	0	0

Figure 6: Step 2

T	*a*	*c*	*g*	*a*	*c*	*g*	*t*	*a*
P'					*c*	*a*	*g*	*c*
IV(c, T)	0	1	0	0	1	0	0	0
IV(g, T)	0	0	1	0	0	1	0	0
IV(a, T)	1	0	0	1	0	0	0	1
Z	0	1	0	0	3	0	0	2	1	0	0

Figure 7: Step 3

T			*a*	*c*	*g*	*a*	*c*	*g*	*t*	*a*
P'							*c*	*a*	*g*	*c*
IV(c, T)			0	1	0	0	1	0	0	0
IV(g, T)		0	0	1	0	0	1	0	0	
IV(a, T)	1	0	0	1	0	0	0	1		
IV(c, T)	0	1	0	0	1	0	0	0		
Z	0	2	0	0	4	0	0	2	1	0	0

Figure 8: Step 4

From the above steps, we see that $Z_6 = m = 4$, and there is an exact string matching of *P* in position 2 of *T*.

3 The Probability for a String to Exactly Appear in Another String

Given a string *T* of length *n* and a pattern *P* of length *m*, and all characters used in *T* and *P* are from an alphabet Σ whose size is σ, we wish to find the probability that *P* exactly appears in *T*. In the following, we use Inclusion-Exclusion Principle, which is shown below, to calculate the probability.

3.1 Inclusion-Exclusion Principle

For events A_1, A_2, and A_3, the Inclusion-Exclusion Principle shows that

$$\Pr(A_1 \cup A_2 \cup A_3) = \Pr(A_1) + \Pr(A_2) + \Pr(A_3) - \Pr(A_1 \cap A_2) - \Pr(A_1 \cap A_3) - \Pr(A_2 \cap A_3) + \Pr(A_1 \cap A_2 \cap A_3).$$

The above equation can be extended to any positive integer number of events. Therefore, we have the general form of the equation as below.

$$\Pr(A_1 \cup A_2 \cup \ldots \cup A_n) = \sum_{1 \le i \le n} \Pr(A_i) - \sum_{1 \le i1 < i2 \le n} \Pr(A_{i1} \cap A_{i2}) + \sum_{1 \le i1 < i2 < i3 \le n} \Pr(A_{i1} \cap A_{i2} \cap A_{i3}) - \cdots + (-1)^n \Pr(A_1 \cap A_2 \cap \ldots \cap A_n)$$

In the following, we calculate the probability that a string *P* exactly appears in another string *T* by using this Inclusion-Exclusion principle.

3.2 Appearing of a String in Another String

Let A_i represent the event that *P* exactly appears at position *i* of *T*.

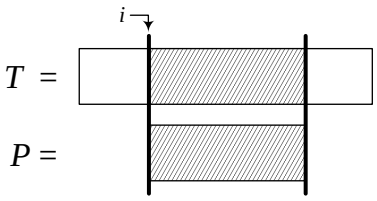

Figure 9: Event A_i represents that *P* exactly appears at position *i* of *T*

If *P* appears in *T*, it means that *P* appears at position 1, or position 2, …, or position $n - m + 1$ of *T*. Therefore, the union of events A_1, A_2, …, and A_{n-m+1} means that *P* appears in *T*. The probability that *P* appears in *T* can be represented by $\Pr(A_1 \cup A_2 \cup \ldots \cup A_{n-m+1})$. By Inclusion-Exclusion Principle, the probability of appearing of *P* in *T* is shown as follows.

$$\Pr(A_1 \cup A_2 \cup \ldots \cup A_{n-m+1}) = \sum_{1 \le i \le n-m+1} \Pr(A_i) - \sum_{1 \le i1 < i2 \le n-m+1} \Pr(A_{i1} \cap A_{i2}) + \sum_{1 \le i1 < i2 < i3 \le n-m+1} \Pr(A_{i1} \cap A_{i2} \cap A_{i3}) - \cdots + (-1)^{n-m+1} \Pr(A_1 \cap A_2 \cap \ldots \cap A_{n-m+1})$$

4 Derivation of an Equation for the Probability

4.1 Dividing the Problem into Two Parts

In the derived equation in Section 3.2, we see that we have to consider the intersection of several events, for example $A_1 \cap A_2$, to calculate the probability. $A_1 \cap A_2$ means that *P* occurs in position 1 and position 2 at the same time, and the occurrences of *P* in *T* overlap.

Therefore, to calculate the probability of *P* exactly appears in *T*, we divide the problem into two different parts.

Part 1: The occurrences of *P* in *T* do not overlap with each other.

Part 2: There exist overlapping occurrences of *P* in *T*.

In the following, we show that for randomly generated text and pattern strings, the probability of appearing of *P* in *T* is quite small under the condition of Part 2. Note that if *P* appears in *T* twice in an overlapping form, a suffix of the pattern string must be equal to a prefix of it. There are two cases.

Case 1: The suffix of *P*, which is equal to a prefix of it, is "long".

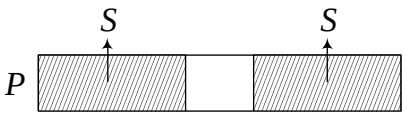

Figure 10: The suffix of *P*, which is equal to a prefix of it, is "long"

In Figure 10, because the "long" string S must appear twice in *P*, we can easily see that the probability for the happening of this case is quite small.

Case 2: The suffix of *P*, which is equal to a prefix of it, is "short".

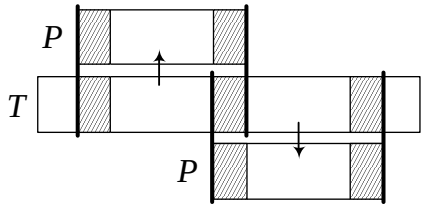

Figure 11: The suffix of *P*, which is equal to a prefix of it, is "short"

In Figure 11, we see that *P* appears twice in *T* in an overlapping form. It means that there appears a string in *T* whose length is about twice of the length of the pattern string. Thus, the probability of this case is quite small.

The above discussions are inexact. In later sections, we shall experimentally prove that for random cases, the probability that a pattern string appears exactly in a text string reduces to 0 quickly as the length of the pattern string increases. Therefore, the probability that Case 2 occurs is even smaller.

From the above discussions, we know that the probability for Part 2 (there exist overlapping occurrences of *P* in *T*) can be ignored.

4.2 Derivation of the Probability for Part 1

Assume that the length of string *T* is *n*, the length of string *P* is *m*, and all characters used in *T* and *P* are from an alphabet Σ whose size is σ.

In order to get the value of the probability, we have to know the values of $\Pr(A_i)$.

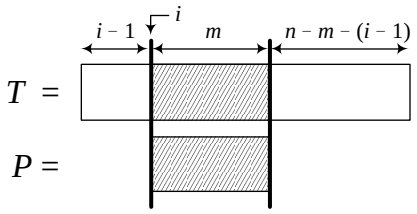

Figure 12: *P* appears at position *i* of *T*

In Figure 12, we see that there is a substring of *T* which is an exact string matching of *P* at position *i* of *T*, whose length is *m*. The probability that *P* appears at position *i* of *T* is as follow.

$$\Pr(A_i) = \frac{\sigma^{i-1} \cdot 1 \cdot \sigma^{n-m-(i-1)}}{\sigma^n} = \frac{\sigma^{n-m}}{\sigma^n} = \frac{1}{\sigma^m}$$

$$\Pr(A_1) + \mathrm{P}(A_2) + \cdots + \mathrm{P}(A_{n-m+1})$$

$$= (n-m+1) \cdot \frac{1}{\sigma^m} = \frac{n-m+1}{\sigma^m}$$

And then, we have to know the values of $\Pr(A_{i1} \cap A_{i2})$.

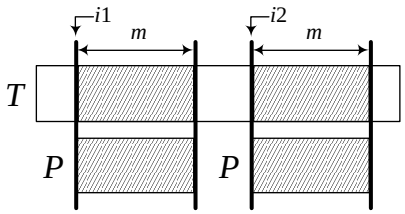

Figure 13: *P* appears at position *i*1 and *i*2 of *T*

Under the assumption that the occurrences of *P* in *T* do not overlap with each other, we can obtain $\Pr(A_{i1} \cap A_{i2})$ as follows:

$$\Pr(A_{i1} \cap A_{i2}) = \begin{cases} \frac{\sigma^{n-2m}}{\sigma^n} = \frac{1}{\sigma^{2m}} & , |i1 - i2| \geq m \\ 0 & , |i1 - i2| < m \end{cases}$$

For calculating the value of $\mathrm{P}(A_1 \cup A_2 \cup \ldots \cup A_{n-m+1})$, we have to know the following value.

$$\sum_{1 \leq i1 < i2 \leq n-m+1} \Pr(A_{i1} \cap A_{i2})$$

In order to calculate this value, we have to know how many different pairs of $i1$ and $i2$ are there such that $\mathrm{P}(A_{i1} \cap A_{i2}) \neq 0$.

In Figure 13, we see that T consists of 2 substrings, and other $n - 2m$ characters, a total of $n - 2m + 2$ objects. The number of different pairs of $i1$ and $i2$ such that $\mathrm{P}(A_{i1} \cap A_{i2}) \neq 0$ is equal to the number of permutations of these $n - 2m + 2$ objects. It is a permutation problem with 2 objects of the same kind and $n - 2m$ objects of another kind.

$$\text{The number of permutations} = \frac{(n-2m+2)!}{(n-2m)!\,2!}$$

$$\sum_{1 \le i1 < i2 \le n-m+1} \Pr(A_{i1} \cap A_{i2})$$

$$= \frac{(n-2m+2)!}{(n-2m)!\,2!} \cdot \frac{1}{\sigma^{2m}}$$

And then, we have to know the values of $\mathrm{P}(A_{i1} \cap A_{i2} \cap \ldots \cap A_{ik})$, where k is a positive integer such that $n - km \geq 0$ and $1 \leq ik \leq n - m - 1$.

$$\mathrm{P}(A_{i1} \cap A_{i2} \cap \ldots \cap A_{ik}) = \begin{cases} \frac{1}{\sigma^{km}} & , \ |i_x - i_y| \geq m, \forall x \neq y \\ 0 & , \ \text{otherwise} \end{cases}$$

By Inclusion-Exclusion Principle and the above results, we can obtain the probability that P appears in T under the assumption that the occurrences of P in T do not overlap with each other.

$$\mathrm{P}(A_1 \cup A_2 \cup \ldots \cup A_{n-m+1}) = (n-m+1) \cdot \frac{1}{\sigma^m}$$

$$- \frac{(n-2m+2)!}{(n-2m)!\,2!} \cdot \frac{1}{\sigma^{2m}} + \frac{(n-3m+3)!}{(n-3m)!\,3!} \cdot \frac{1}{\sigma^{3m}}$$

$$- \cdots + (-1)^{k+1} \cdot \frac{(n-km+k)!}{(n-km)!\,k!} \cdot \frac{1}{\sigma^{km}}$$

where k is the largest positive integer such that $n - km \geq 0$.

5 The Value of the Probability from the Derived Equation and from Experiments

In Section 5.1, we use the derived equation in Section 4.2 to calculate the probability at different n and m. In Section 5.2, we randomly generate 10000 pairs of T and P, and calculate the probability of appearing of P in T.

5.1 Calculating the Probability by the Derived Equation

We calculate the probability at different values of n and m. In this paper, we use n which is equal to 30, 100, 300, 1000, 3000, and 10000, and for each n, we use m which is integers from 1 to 20. The results are shown in Figure 14.

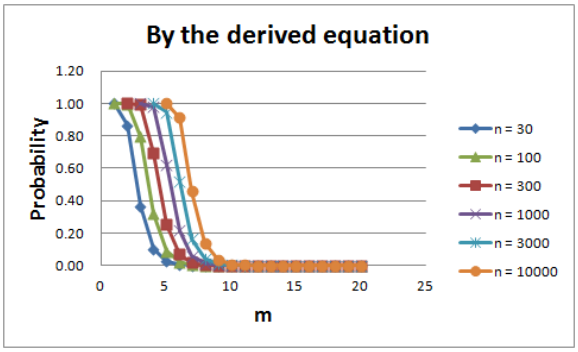

Figure 14: the probability from derived equation at different n and m

From the above figure, we can see that the probability that a pattern string appears in a text string reduces to 0 quickly as the length of the pattern string increases.

5.2 Calculating the probability by randomly generated text and pattern strings

We wrote a program in C language. And for each pair of n and m, we randomly generated 10000 pairs of T and P, and calculated the probability of appearing of P in T. In this paper, we used rand() function defined in the library <stdlib.h> to generate random strings for our experiments. In Figure 15, we show the results for different n and m.

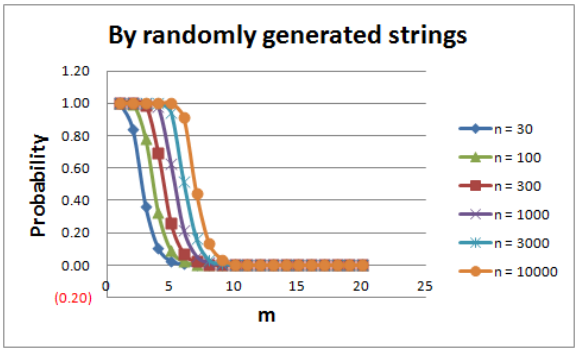

Figure 15: the probability from randomly generated strings at different n and m

In Figure 14 and 15, we see that the derived

probability is very close to the real cases under the assumption that P and T are generated randomly. And we see that it is quite difficult for a "long" pattern string to appear in a text string T.

From the above theoretical and experimental results, we can conclude that we may ignore Part 2 (there exist overlapping occurrences of P in T).

6 Improving the Algorithm by Discrete Convolution Method with Early Termination

From the results in Section 3, we know that it is quite difficult for a "long" pattern string to appear in a text string. Thus, we have an idea. Can we terminate the process of discrete convolution earlier? In this section, we introduce a way to achieve our purpose.

When we are performing a convolution process, we continually check all digits of Z. When we find that all digits of Z are smaller than k in step k, we terminate the process and output that there is no exact string matching for the corresponding text and pattern strings.

Let's show this idea by a simple example.

Example 2:

Let T = *acgacgta* and P = *ctac*. First, we reverse the pattern string P = *ctac* to P' = *catc*. The incidence vectors of every characters in Σ on T are $IV(a, T)$ = (10010001), $IV(c, T)$ = (01001000), $IV(g, T)$ = (00100100), and $IV(t, T)$ = (00000010).

Figure 16 and 17 show the discrete convolution between T and P' in a graphical way.

T				*a*	*c*	*g*	*a*	*c*	*g*	*t*	*a*
P'								*c*	*a*	*t*	*c*
$IV(c, T)$				0	1	0	0	1	0	0	0
Z	0	0	0	0	1	0	0	1	0	0	0

Figure 16: Step 1

T				*a*	*c*	*g*	*a*	*c*	*g*	*t*	*a*
P'								*c*	*a*	*t*	*c*
$IV(c, T)$				0	1	0	0	1	0	0	0
$IV(g, T)$			0	0	0	0	0	0	1	0	
Z	0	0	0	0	1	0	0	1	1	0	0

Figure 17: Step 2

In Figure 17, we see that all digits of Z are smaller than 2 at step 2. Therefore, we know that the prefix "*ct*" of P does not appear in T and can terminate the process earlier.

7 Experiments for Our Algorithm Based on Discrete Convolution Method with Early Termination

In this program, we randomly generated 10,000 pairs of T and P, and calculated the average number of used steps and used comparisons to solve the exact string matching problem. In Figure 18, we show the results of the program for different n and m.

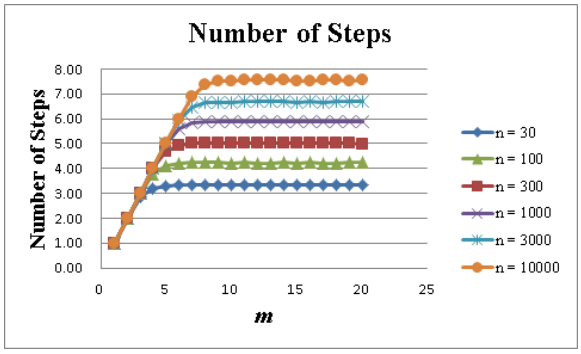

Figure 18: The number of steps used at different n and m

If we use the original algorithm without early termination, the number of used steps is equal to the length of the pattern string m. For example, for a pattern string of length 15, the number of used steps is 15.

In Figure 19 and 20, we show the number of comparisons used of the algorithm with and without early termination respectively.

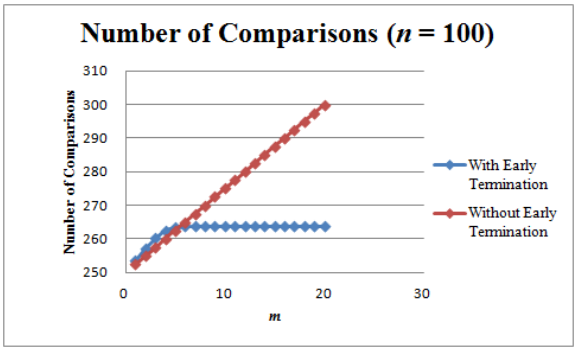

Figure 19: The number of comparisons used at different m when n = 100

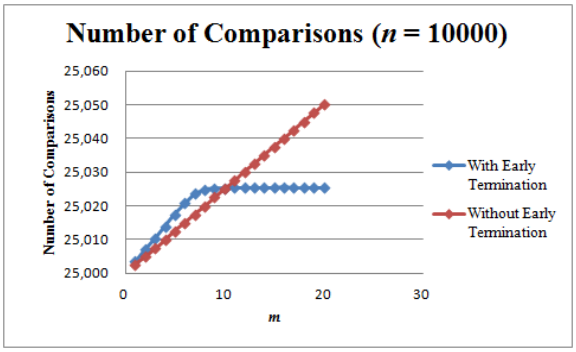

Figure 20: The number of comparisons used at different m when n = 10000

In Figure 19 and 20, we see that when m is small, the number of comparisons used in the algorithm with early termination is worse than the original algorithm. It is because that when m is small, the probability of appearing of a short pattern string in a text string is large. For a pattern string which appears in the text string, the algorithm with early termination spends more comparisons on checking all digits of Z in each step.

But when the length of the pattern string increase, the number of comparisons used is better than the original algorithm.

From the above experiments, we know that the discrete convolution method with early termination is quite efficient to solve exact string matching problem for a long pattern string.

8 Conclusion

In Section 5, we show that the probability that a pattern string appears in a text string reduces to 0 quickly as the length of the pattern string increases. Therefore, it is quite difficult for a "long" pattern string to appear in a text string. Because of this observation, we introduce an algorithm based on the discrete convolution method with early termination. In Section 7, our experiments show that the discrete convolution method with early termination is quite efficient to solve exact string matching problem.

References

[1] M. J. Fischer, and M. S. Paterson, String-Matching and other products. SIAM-AMS Proceedings, Vol. 7, 1974, pp. 113-125 *(*In *Complexity of Computation*, R. M. Karp.)

[2] R. C. T. Lee. *String matching*. 2011. (Unpublished)

[3] B. H. Wu. Convolution and Its Applications to Sequence Analysis. M.D., National Chi-Nan University, 2004.

[4] Z. H. Chen. The Application of Convolution to Suffix to Prefix Rule for the Exact String Matching Problem. M.D., National Chi-Nan University, 2007.