The \( i \)-largest Number Domination Sequence and Its Application to the Average Case Analysis of the Horspool Algorithm

S. C. Chen, G. S. Huang and R. C. T. Lee
Department of Department of Computer Science and Information Engineering,
National ChiNan University, Puli, Nantou Hsien, Taiwan, 54561
{s94321905, shieng, rctlee}@ncnu.edu.tw

Abstract

In this paper, we define the \( i \)-largest number domination sequence and compute the number of \( i \)-largest number domination sequence with length \( L \). We use this result to analyze the average case analysis of the Horspool algorithm when we are given a random pattern and a random text.

1 Introduction

We define the \( i \)-largest number domination sequence as follows:

An \( i \)-largest number domination sequence is a sequence \( S \) consisting of integers 1, 2, ..., \( i \) satisfying the following conditions:

1. The integer \( i \) is the largest in the sequence, appears at the last position and appears only once and the integer at the first position must be 1.
2. For every positive integer \( k \) smaller than \( i \), there exists a \( k \)-largest number domination sequence as a prefix of \( S \).

Because the number of \( i \)-largest number domination sequence with \( L \) can be proved to be equivalent to Stirling numbers of the second kind (the number of ways of partitioning a set of \( n \) elements into \( m \) nonempty sets), the recurrence formula to compute the number of \( i \)-largest number domination sequences with length \( L \) is similar to the recurrence formula of Stirling numbers of the second kind. This formula is related to the average case analysis of the Horspool algorithm [4].

The exact string matching problem is a classical problem in computer science and can be applied to many fields. For example, DNA sequence analysis, searching engines, subcircuit extraction problem, image processing and virus scanning all apply this technique. There are many algorithms to solve this problem. The famous algorithms are KMP algorithm [2] and BM algorithm [3]. Horspool algorithm is one of the algorithms to solve the exact string matching and this algorithm is the simplified version of the Boyer-Moore algorithm.

The time complexity of the worst case for KMP Algorithm and BM Algorithm is also \( O(n) \) where \( n \) is the length of the text. The time complexity of the worst case for Horspool Algorithm is \( O(mn) \) where \( m \) is the length of the pattern. In [1], Yao showed that the average-case complexity of exact string matching problem is \( \Omega \left( \frac{n}{m \log m} \right) \). In [5],...
they showed that the expected number of comparison for Horspool Algorithm is related to the average shift where shift is the distance to move P, hence we want to compute the average shift in our paper.

The organization of this paper is as follows. Section 2 presents a recurrence formula to compute the number of i-largest number domination sequences with length L. Section 3 presents the string matching problem and the Horspool algorithm. Section 4 presents the relation between the i-largest number domination sequence and the average case analysis of the Horspool algorithm. Section 5 presents the detail of the average case analysis of the Horspool algorithm. Section 6 is the experiment to verify our theorem. The last section is our conclusion.

2 A Recurrence Formula to Compute the Number of i-largest Number Domination Sequences with Length L

According to the definition given in Section 1, we give several examples of the i-largest number domination sequences. The following sequences are all i-largest number domination sequences for some i:

\{1, 12, 123, 1234, 112, 11112, 1111112, 1223, 1213, 12223, 12221334\}

The following sequences are not i-largest number domination sequences:

\{11, 121, 1122344, 1233, 213, 2213\}

The i-largest number domination sequence problem is to determine the total number of i-largest number domination sequences with length L for a given i and a given L. For instance, let i=3 and L=3. Then there is only one 3-largest number domination sequences with length 3, namely 123. If i=2 and L=3, there is also one 2-largest number domination sequences with length 3, namely 112. For i=3 and L=4, there are three 3-largest number domination sequences with length 4, namely 1123, 1213 and 1223.

How do we compute the total number of i-largest number domination sequences with length L? Let D(i, L) be the number of all i-largest number domination sequences with length L. That is, i is the largest number and the length of the sequences are all L. Then we have the following recurrence formula:

$$D(i, L) = D(i-1, L-1) + (i-1) D(i, L-1)$$

for $$i \geq 2$$ and $$L \geq i$$

with boundary conditions $$D(1, L)=0$$ for $$L>1$$ and $$D(i, i)=1$$ for $$i \geq 1$$. This recurrence formula can be derived from the following reasoning. Let $$a_1 a_2 \ldots a_L$$ be a i-largest number domination sequence with length L. Clearly, $$a_L$$ must be i. Now consider the position of the first occurrence of i-1 in $$a_1 a_2 \ldots a_L$$. There are two possibilities: it can be at position L-1 or prior to position L-1. As for the first case, $$a_1 a_2 \ldots a_{L-1}$$ is an (i-1)-largest number domination sequence with length L-1. Hence there are D(i-1, L-1) such sequences. As for the second case, $$a_{L-1}$$ must be one of 1, 2, ..., i-1. It follows that $$a_1 a_2 \ldots a_{L-2} i$$ (note that $$a_{L-1}$$ is replaced by the number i) is an i-largest number domination sequence with length L-1. Since there are D(i, L-1) i-largest number domination sequences with length L-1 and each one further contributes (i-1) sequences to $$a_1 a_2 \ldots a_{L-1}$$, the term (i-1) D(i, L-1) follows. The boundary condition $$D(i, i)=1$$
because there is only one $i$-largest number domination sequence with length $i$ for all $i$ and $D(1, L)=0$ for $L>1$ because there is no 1-largest number domination sequence whose length is larger than 1.

For instance,

$D(4, 6) = D(3, 5) + 3D(4, 5) = \ldots = 25.$

We list $D(i, L)$ for $1 \leq i \leq 7$ and $1 \leq L \leq 7$ in Table 1.

Table 1. The number of $i$-largest number domination sequence with length $L$ for $1 \leq i \leq 7$ and $1 \leq L \leq 7$.

<table>
<thead>
<tr>
<th>$L$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>15</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>6</td>
<td>25</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>10</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

3 The Exact String Matching Problem and the Horspool Algorithm

For the exact string matching problem, we are given a text $T=t_1t_2\ldots t_n$ and a pattern $P=p_1p_2\ldots p_m$ for $n \geq m$. Our job is to find all occurrences of $P$ in $T$. The Horspool algorithm is one of the algorithms to solve the exact string matching problem and it can be considered as a simplified version of the Boyer-Moore algorithm. The idea of the Horspool algorithm is follows.

Let $W$ be a substring of $T$ with length $m$, the last character of $W$ be $x$ and $P(i, j)$ be the substring of $P$ with length $j-i+1$ whose first character is $p_i$ and the last character is $p_j$. If we have to move $P$ in order to find an occurrence of $P$ after $W$ in $T$, we must align the rightmost $x$ in $P(1, m-1)$ to the last character $x$ in $W$ as shown in Figure 2. Let $shift$ be the distance to move $P$. If $x$ does not occur in $P(1, m-1)$, we move $P$ to the next position of $x$ of $W$ as shown in Figure 3. If $x$ occurs in $P(1, m-1)$ and the location of $x$ in $P(1, m-1)$ is $m-L$, the $shift=m-(m-L)=L$, otherwise $shift= m$.

The pseudo code of the Horspool algorithm [4] is extremely simple and is presented in the following.

Program The Horspool Algorithm

**Input:** A text string $T$ and a pattern string $P$ with lengths $n$ and $m$ respectively

**Output:** All occurrences of $P$ in $T$.

```
bmsearch( text, n, pattern, m) /* Search
```
pattern[1…m] in text[1…n] /*
char text[], pattern[];
int n, m;
{
    /*Preprocessing*/
    int d[alphabet_size], i, j, k;
    for (j=0; j<alphabet_size; j=j+1) d[j]=m;
    for (j=1; j<m; j=j+1) d[pattern[j]]=m-j;
    /*Search*/
    for (i=m; i<=n; i=i+d[text[i]])
    {
        k=i;
        for (j=m; j>0&&text[k]==pattern[j]; j=j-1) k=k-1;
        if (j = 0)
            Report_match_at_position(k+1);
    }
}

As can be seen, the Horspool Algorithm is actually a window sliding algorithm. The average number of steps of the shifting of the window is therefore very important. If, in average, the number of steps of the window being shifted is large, the algorithm is efficient. For the Horspool Algorithm, the number of steps of shifting is determined by how distinct characters are arranged in the pattern $P$. Let us consider

$P$=TCAACGTTTTTTTTTTT.

We can easily see that if the last character of the window $W$ is not $T$, the number of steps of this pattern shifting is quite large. On the other hand, suppose that

$P$=ACCGTGTACCCACGTT

In this case, no matter what the last character of the window $W$ is, the number of steps of the pattern shifting is relatively small.

4 Relations of the $i$-largest Number Domination Sequences to Average Case Analysis of the Horspool Algorithm

We are facing an interesting problem. Suppose that the alphabet is $\Sigma = \{x_1, x_2, \cdots, x_c\}$. Without losing generality, we may assume that when we scan from right to the left in the $P$, starting from $P_{m-1}$, the distinct characters we encounter are ordered as $x_1, x_2, \cdots, x_c$. That is, $P_{m-1} = x_1$. Then the second distinct character we encounter is $x_2$. For example, let $P_1$ = ACCGTTGTAC.

Then, $x_1$=A, located at $p_5$; $x_2$=T, located at $p_6$; $x_3$=G, located at $p_7$; $x_4$=C, located at $p_3$.

For each $x_i$, we want to know the location of the rightmost $x_i$ in $P_{1}(1, m-1)$ if it does exist, counted from location $m-1$. Let us denote this number as $shift_i$. For the same example $P_1$ = ACCGTTGTAC, we have

$shift_1 = 1$;
$shift_2 = 2$;
$shift_3 = 3$;
$shift_4 = 7$.

In order to find the average case performance of the Horspool Algorithm, we have to find the average values of $shift_i$'s, “average shift” for short. It will be informative for us to code the string $P(1, m-1)$ into a string consisting of positive integers only. Let us code $x_i$ by $i$. For $P_1 = ACGTTGTAC$, the coding is as follows:

- $x_1 = A \rightarrow 1$;
- $x_2 = T \rightarrow 2$;
- $x_3 = G \rightarrow 3$;
- $x_4 = C \rightarrow 4$.

Thus the original pattern $P_1(1, m-1)$ becomes: $144322321$. Let us now reverse it and we have $123223441$. We use the notation $N(P)$ to denote the inverted sequence of $P$.

Scanning from the left on the inverted sequence, let us single out four prefix sequences: the prefix sequence where the first 1 appears and 1 is the largest, the sequence where the first 2 appears and 2 is the largest and so on. We have the following sequences:

- $S_1: 1$ (the first 1 appears at location 1.)
- $S_2: 12$ (the first 2 appears at location 2.)
- $S_3: 123$ (The first 3 appears at location 3.)
- $S_4: 1232234$ (The first 4 appears at location 7.)

We shall point out that these sequences have a common property. Before doing that, let us consider another example. Let $P_2 = ACTGGGATCAGAGAAT$. It can be seen that $P_2(1, m-1)$ becomes $1324221431211$. We reverse the above sequence into $N(P_2) = 112121341224231$ under the coding $\{A \rightarrow 1, G \rightarrow 2, C \rightarrow 3, T \rightarrow 4,\}$. Then we have the following sequences:

- $S_1: 1$ (the first 1 appears at location 1.)
- $S_2: 112$ (the first 2 appears at location 3.)
- $S_3: 112123$ (the first 3 appears at location 6.)
- $S_4: 11212134$ (the first 4 appears at location 8.)

The physical meaning of each sequence listed above is as follows:

- $S_1$: The first distinct character appears in $P_2$ at location $m-1$;
- $S_2$: The second distinct character appears in $P_2$ at location $m-3$;
- $S_3$: The third distinct character appears in $P_2$ at location $m-6$;
- $S_4$: The third distinct character appears in $P_2$ at location $m-8$.

Hence, $shift_1 = 1$, $shift_2 = 3$, $shift_3 = 6$ and $shift_4 = 8$.

In other words, that the sequence $S_1$ for the first distinct character appears at location $m-1$ in $P$ is equal to 1-largest number domination sequence with length 1. That the sequence $S_2$ for the second distinct character appears at location $m-3$ in $P$ is equal to 2-largest number domination sequence with length 3. In general, that the sequence for the $i$th distinct character appears at location $m-L$ in $P$ is equal to the $i$-largest number domination sequence with length $L$.

**5 On the Average Case Analysis of the Horspool Algorithm**

From the above discussion, we can see that the first distinct character in $P(1, m-1)$, counted from the right, must be located at $m-1$ in $P$ with
$\text{shift} = 1$ as shown in Figure 4. But the second distinct character may appear at anywhere. To analyze the average case performance of the Horspool Algorithm, we must know the average shift of the $i$-th distinct character for a random pattern and a random text. It turns out that this problem can be formulated as the $i$-largest number domination problem.

If we are given random numeral sequences with length $L$, the probability that an $i$-largest number domination sequence with length $L$ occurs is

$$\frac{D(i, L)}{c^L} \quad (3)$$

When the $i$-th distinct character is equal to $x$, the last character of $W$, its shift is equal to $L$, which is also the length of the $i$-largest number domination sequence with length $L$ where $L \leq m$.

The average shift for the $i$-th distinct symbol in a random pattern with length $m$ is

$$\sum_{L=i}^{m-1} \left( \frac{L}{c^L} \right) \quad (4)$$

If $x$ does not occur in $P(1, m-1)$, then $\text{shift} = m$.

Experiments

We do an experiment to get the average shift.
using DNA sequences. The input is a random
text with length 10000 and all combinations of
pattern with length 6, 7 and 11. Totally, we do
this experiment for each length 10 times. The
results are in the following Table 2.

<table>
<thead>
<tr>
<th>The length of pattern</th>
<th>Experiment value</th>
<th>Theoretical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m=6</td>
<td>3.287554</td>
<td>3.012695</td>
</tr>
<tr>
<td>m=7</td>
<td>3.466941</td>
<td>3.303711</td>
</tr>
<tr>
<td>m=11</td>
<td>3.836701</td>
<td>3.814986</td>
</tr>
</tbody>
</table>

7 Conclusion

In this paper, we define the \( i \)-largest number
domination sequences and compute the number
of the \( i \)-largest number domination sequences
with length \( L \). We use this result to analyze the
average shift of the Horspool algorithm. In the
future, we will investigate on how to simplify
the formula (9) and to discover more
applications of the \( i \)-largest domination
sequences.

Reference

Matching for a Random String, *SIAM

Fast Pattern Matching in Strings, *SIAM
Journal on Computing*, Vol. 6, pp. 323-350,
1977.

Searching Algorithm, *Communications of

Strings, *Software Practice and Experience*,

[5] Ricardo A. Baeza-Yates and Mireille Régnier, Average Running Time of the
Boyer Moore Horspool Algorithm,