Computing the Pattern Waiting Time: A Revisit of the Intuitive Approach[*]

Kai Jin

Department of Computer Science, University of Hong Kong, Hong Kong
cscjjk@gmail.com

Abstract

We revisit the waiting time of patterns in repeated independent experiments. We show that the most intuitive approach for computing the waiting time, which reduces it to computing the stopping time of a Markov chain, is optimum from the perspective of computational complexity. For the single pattern case, this approach requires us to solve a system of m linear equations, where m denotes the length of the pattern. We show that this system can be solved in $O(m+n)$ time, where n denotes the number of possible outcomes of each single experiment. The main procedure only costs $O(m)$ time, while a preprocessing procedure costs $O(m+n)$ time. For the multiple pattern case, our approach is as efficient as the one given by Li [14].

Our method has several advantages over other methods. First, it extends to compute the variance or even higher moment of the waiting time for the single pattern case. Second, it is more intuitive and does not entail tedious mathematics and heavy probability theory. Our main result (Theorem 2) might be of independent interest to the theory of linear equations.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Pattern matching, G.3 [Probability and Statistics] Markov processes, G.1.3 [Numerical Linear Algebra] Linear systems

Keywords and phrases Pattern Occurrence, Waiting Time, Penney's Game, Markov Chain

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.39

1 Introduction

In this paper, we revisit the waiting time of patterns in repeated independent experiments. Consider an experiment with countably many *outcomes*, denoted by $1, \ldots, n$, where the probabilities of the outcomes might be nonidentical. Let the experiment be performed repeatedly. Given a collection of t finite sequences of possible outcomes, called *patterns*, we compute the expected waiting time till one of the pattern is observed in a run of experiments. We also compute the probability for each pattern to be the first to appear.

These quantities have been extensively studied in history and different approaches were invented for computing them. The existing approaches apply complicated mathematical tools, e.g. the generating functions applied in [12] and [4], the martingale theory applied in [14], and the renewal theory applied in [4]. In this paper we revisit a much more intuitive approach for computing them and we show that this approach is computationally optimal.

For the single pattern case, we use the following approach to compute the expected waiting time. We reduce it to compute the expected stopping time of a Markov chain (see its construction in Subsection 1.2), which consists of m nonterminal states, where m is

[*] This work is done at Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University. Supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, and the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174.

© Kai Jin;
licensed under Creative Commons License CC-BY

the length of the given pattern. The expected stopping time can easily be computed in polynomial time by standard methods, since it reduces to solve a system of linear equations with m equations and m variables. We propose a novel method to solve this system and thus compute the expected stopping time starting from any state of the Markov chain, which only costs $O(m+n)$ time and thus is computationally optimal. This time complexity cannot be guaranteed by standard linear system solving; for achieving it we exploit structural properties of the Markov chain and some insights about pattern matching.

Our method easily extends to compute the variance or even higher moments of the waiting time for a single pattern, starting from any state of the Markov chain. The running time is still $O(m+n)$ for the variances, and is $O(m \cdot k^2 + n)$ for the k-th moments.

For the multiple pattern case, computing the expected waiting time and the probability for each pattern to be first to appear reduces to solving a system of linear equations with $t+1$ variables and $t+1$ equations, in which the coefficients can be computed by our algorithm for the single pattern case. Our algorithm has the same time complexity as the one given in [14].

1.1 Related work and applications

The waiting time of patterns is a classic problem in probability theory which arises in 1960s ([17, 23]). Since then it has drawn a lot of attentions and has been studied extensively. It has many applications in different fields, including computer science, telecommunication, molecular biology, statics and applied probability; see [22, 11, 16, 15] and the references within.

Due to the practical importance of the waiting time, several approaches were invented for studying it. Guibas and Odlyzko [12] studied it via a combinatorial approach. They computed the generating function of the number of strings with any fixed length which contain none of the given patterns. Breen, Waterman and Zhang [4] obtained similar results via a probabilistic approach, in which the renewal theory of Feller [7] is applied. Li [14] studied it via martingale theory and general Markov chain theory (See also [19]). Most approaches entail tedious mathematics and heavy probability theory, even for the single pattern case. Useful reviews of different approaches can be found in two recent books: [3, 8].

To compute the waiting time, there is an intuitive approach as aforementioned. It appears in several literatures and is usually called Markov Embedding approach. However, to the best of our knowledge, researches have not recognized that this approach is optimum from the perspective of computational complexity. (Usually, the stopping time of a Markov chain cannot be computed in linear time).

Many variants of the original problem have been studied. For example, Fudos, Pitoura and Szpankowski [9] studied the probability of exactly r occurrences of a pattern in a fixed number of Bernoulli experiments (independent identical experiments). In addition, people also considered the waiting time in Markovian experiments (Markovian dependent experiments) instead of Bernoulli experiments; see [5, 18, 2] and the references within.

A closely related topic to the waiting time is the frequency of patterns. Régnier and coauthors made important contributions to this topic. Let O_k denote the number of occurrence of a given pattern in k Markovian experiments when overlapping is counted separately. [21] computed the mean and variance of O_k. This result was extended to the higher order Markovian experiments, for non-overlap or overlap counting, in [20].

The literature of the problem concerning the probability for each pattern to be the first to appear starts from [17]. This problem is referred to as Penney's Game ([1]). Conway gave a beautiful formula for the 2-players Penney's Game (See [10]). The best strategy is known for choosing the pattern so that the winning probability is optimal; see [10, 12, 6].

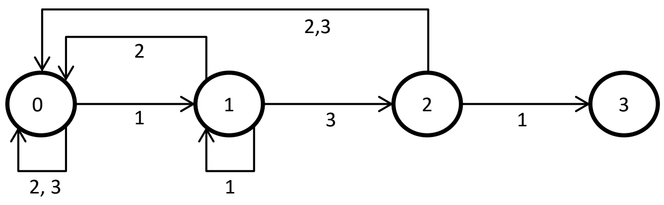

Figure 1 The Markov chain corresponding to the above example.

1.2 Technique overview

Now, we define the aforementioned Markov chain corresponding to a single pattern, state our main result and present the overview of our techniques.

The Markov chain corresponding to pattern W

Denote $m = |W|$ and assume $W = W_1W_2..W_m$. Denote $W^{(i)} = W[1..i]$, which is the prefix of W with length i. Recall that there are n possible outcomes and assume that the probability of outcome i is $Pr(i)$. Note that $Pr(1), \ldots, Pr(n)$ are fixed parameters.

The Markov chain consists of $m+1$ states, state 0 to state m, each of which corresponds to a prefix of W. State m is the *terminal state.* For each state $i (0 \le i \le m-1)$, there are n outgoing edges whose ending points are denoted by $\tau(i,1), \ldots, \tau(i,n)$, respectively. Specifically, if it's now at state i and the next outcome is j, the next state should be

$$\tau(i,j) = \max\{h \mid W^{(h)} \text{ is a suffix of } (W^{(i)} + \text{`}j\text{'})\}. \tag{1}$$

The j-th outgoing edge of each state has an associated probability $Pr(j)$.

Note: The symbol '+' in Equation 1 indicates concatenation of strings.

▶ **Example 1.** Let $W =$ "131". $n = 3$. The corresponding Markov chain is drawn in Figure 1.

Clearly, computing the expected waiting time of pattern W reduces to computing the expected stopping time of the preceding Markov chain.

Our Main Results

Suppose $(c_0, \ldots, c_{m-1})$ are m constants. We consider the following system of linear equations.

$$x_i = \begin{cases} 0, & i = m; \\ c_i + \sum_{j=1}^{n} Pr(j) \cdot x_{\tau(i,j)}, & i = 0..m-1. \end{cases} \tag{2}$$

Note that there are $m+1$ variables in this system, which are $x_0, \ldots, x_m$, and there are also $m+1$ equations. Our main result is the following.

▶ **Theorem 2.** *The solution of system (2) can be computed in $O(m+n)$ time.*

Denote e_i as the expected "stopping time" starting from state i; and denote f_i as the expected "square of stopping time" starting from state i. Formally,

$$e_i := \sum_{t \ge 0} t \cdot Pr(\text{It takes } t \text{ steps to get the terminal state from state } i); \tag{3}$$

$$f_i = \sum_{t \ge 0} t^2 \cdot Pr(\text{It takes } t \text{ steps to get the terminal state from state } i). \tag{4}$$

The following equations are easy to prove.

$$e_i = \begin{cases} 0, & i = m; \\ \left(\sum_{j=1}^{n} Pr(j) \cdot e_{\tau(i,j)}\right) + 1, & i = 0..m-1. \end{cases} \tag{5}$$

$$f_i = \begin{cases} 0, & i = m; \\ \left(\sum_{j=1}^{n} Pr(j) \cdot f_{\tau(i,j)}\right) + (2e_i - 1), & i = 0..m-1. \end{cases} \tag{6}$$

Proof of (5). Assume $i < m$; otherwise it is trivial. By First-Step-Analysis,

$$\begin{aligned} e_i &= \sum_{j=1}^{n} Pr(j) \sum_{t \geq 0} (t+1) \cdot Pr(\text{It takes } t \text{ steps to get state } m \text{ from state } \tau(i,j)) \\ &= \sum_{j=1}^{n} Pr(j) \cdot \left(e_{\tau(i,j)} + 1\right) \\ &= \left(\sum_{j=1}^{n} Pr(j) \cdot e_{\tau(i,j)}\right) + 1. \end{aligned}$$

◀

Proof of (6). Assume $i < m$; otherwise it is trivial. By First-Step-Analysis,

$$\begin{aligned} f_i &= \sum_{j=1}^{n} Pr(j) \sum_{t \geq 0} (t+1)^2 \cdot Pr(\text{It takes } t \text{ steps to get state } m \text{ from state } \tau(i,j)) \\ &= \sum_{j=1}^{n} Pr(j) \cdot \left(f_{\tau(i,j)} + 2e_{\tau(i,j)} + 1\right) \\ &= \left(\sum_{j=1}^{n} Pr(j) \cdot f_{\tau(i,j)}\right) + 2(e_i - 1) + 1. \end{aligned}$$

◀

According to Theorem 2, we can compute array $e = (e_0, \ldots, e_m)$ in $O(m+n)$ time. Moreover, after array e has been computed, we can further compute array $f = (f_0, \ldots, f_m)$ in $O(m+n)$ time. The expected stopping time is e_0, and its variance is $f_0 - e_0^2$.

▶ **Corollary 3.** *Given a pattern W with length m, the expected waiting time of W and the variance of the waiting time of W can be computed in $O(m+n)$ time.*

The above method for computing the expectation and variance can be easily generalized to compute the higher moments. We show this generalization in Section 5.

Technique overview – solve system (2)

System (2) has m unknowns $x_0, \ldots, x_{m-1}$ to compute. Rather than compute them directly, we define $y_i = x_0 - x_i$ and compute the unknowns $y_1, \ldots, y_m$. The difficulty on computing y_i lies in computing the term $\sum_{j=1..n, j \neq W_{i+1}} Pr(j) \cdot y_{\tau(i,j)}$, which is denoted by z_i.

Let $\pi : \{0, .., m\} \to \{-1, .., m-1\}$ be the well-known prefix function of W. Formally,

$$\pi_i = \begin{cases} -1, & i = 0; \\ \max\left\{h \mid h < i \text{ and } W^{(h)} \text{ is a suffix of } W^{(i)}\right\}, & i > 0. \end{cases} \tag{7}$$

Our key observation is that z_i can be computed from z_{π_i} in $O(1)$ time. More specifically,

$$z_i = z_{\pi_i} + [y_{\pi_i+1} \cdot Pr(W_{\pi_i+1})] - [y_{\sigma_i} \cdot Pr(W_{i+1})] \quad (\text{for } 1 \leq i < m),$$

where $\sigma_i = \tau(\pi_i, W_{i+1})$.

Based on this observation, we compute y and z in $O(m)$ time. Besides, we spend $O(n+m)$ time to compute the auxiliary array σ. Our method is computationally optimum, since inputting the task instance would already cost $O(n+m)$ time.

Technique overview – the multiple pattern case

For the multiple pattern case, we only consider the expectation of waiting time and the probability for each pattern to be first to appear; we do not consider variance of even higher moments of the waiting time. Our method is similar to the one of Li [14]. Specifically, we shall solve the same system of linear equations (which consists of $t+1$ variables) proposed by Li. However, we use a different approach to compute the coefficients of this system.

Outline. We prove Theorem 2 in Section 2. We present an alternative linear algorithm for computing $e_0, \dots, e_{m-1}$ in Section 3. We consider the multiple pattern case in Section 4. We compute the higher moments of the waiting time of a single pattern in Section 5.

2 Solving system (2) in linear time

▶ **Definition 4.** For $0 \le i \le m$, we define a *periods set*

$$\mathcal{P}(i) = \begin{cases} \{0\}, & i = 0; \\ \mathcal{P}(\pi_i) \cup \{i\}, & i > 0. \end{cases}$$

The following lemmas can be found in any textbook on finite automata and is the basis of the famous KMP (Knuth-Morris-Pratt) algorithm ([13]). We omit their trivial proofs.

▶ **Lemma 5.** *Suppose $0 \le i \le m$ and $0 \le h \le i$. Then,*

$h \in \mathcal{P}(i)$ if and only if $W^{(h)}$ is a suffix of $W^{(i)}$.

▶ **Lemma 6.** *For $1 \le i < m$ and $1 \le j \le n$, we have*

$$\tau(i,j) = \begin{cases} i+1, & j = W_{i+1}; \\ \tau(\pi_i, j), & j \ne W_{i+1}. \end{cases} \tag{8}$$

In this section, we prove Theorem 2, which states that the system of linear equations stated in (2) can be solved in linear time.

We do not compute x directly; instead, we compute another array $y = (y_0, \dots, y_m)$, which are defined as follows. To compute y efficiently, we introduce two more arrays $z = (z_0, \dots, z_{m-1})$ and $\sigma = (\sigma_1, \dots, \sigma_{m-1})$ as follows.

$$y_i = x_0 - x_i \quad (0 \le i \le m) \tag{9}$$

$$z_i = \sum_{j=1..n, j \ne W_{i+1}} Pr(j) \cdot y_{\tau(i,j)} \quad (0 \le i < m) \tag{10}$$

$$\sigma_i = \tau(\pi_i, W_{i+1}) \quad (\forall 1 \le i < m) \tag{11}$$

We state two formulas. (Their proofs are deferred for a moment.)

$$y_i = (c_{i-1} + y_{i-1} - z_{i-1})/Pr(W_i) \quad (\text{for } 1 \le i \le m)\,. \tag{12}$$

$$z_i = z_{\pi_i} + [y_{\pi_i+1} \cdot Pr(W_{\pi_i+1})] - [y_{\sigma_i} \cdot Pr(W_{i+1})] \quad (\text{for } 1 \le i < m) \tag{13}$$

We give the algorithm in Algorithm 1.

Note that when we compute z_i in Line 5, we have $\pi_i < i, \pi + 1 \le i$ and $\sigma_i \le i$. These inequalities respectively imply that the quantities z_{π_i}, y_{π_i+1}, and y_{σ_i} have been computed before we compute z_i. Therefore, Algorithm 1 is correct according to (12) and (13).

Algorithm 1 Algorithm for computing x

```
1: Compute $\pi$ and $\sigma$;                          ▷ See the detailed algorithm in the next subsection.
2: $y_0, z_0 \leftarrow 0$;
3: for $i = 1..m-1$ do
4:     $y_i \leftarrow (c_{i-1} + y_{i-1} - z_{i-1})/Pr(W_i)$.                 ▷ Applying (12).
5:     $z_i \leftarrow z_{\pi_i} + y_{\pi_i+1} \cdot Pr(W_{\pi_i+1}) - y_{\sigma_i} \cdot Pr(W_{i+1})$       ▷ Applying (13).
6: end for
7: $y_m \leftarrow (c_{m-1} + y_{m-1} - z_{m-1})/Pr(W_m)$;                      ▷ Applying (12).
8: $x_0 \leftarrow y_m$.                          ▷ Because $y_m = x_0 - x_m = x_0 - 0 = x_0$.
9: Compute $x_1, \ldots, x_{m-1}$ from $y$ by formula $x_i = x_0 - y_i$.
```

Proof of (12). Assume $1 \le i \le m$. Then,

$$\begin{aligned}
y_{i-1} + c_{i-1} &= x_0 - x_{i-1} + c_{i-1} && \text{(due to (9))}\\
&= x_0 - \sum_{j=1..n} Pr(j) \cdot x_{\tau(i-1,j)} && \text{(due to (2))}\\
&= (\sum_{j=1..n} Pr(j)) \cdot x_0 - \sum_{j=1..n} Pr(j) \cdot x_{\tau(i-1,j)} && \text{(since } \sum_{j=1}^{n} Pr(j) = 1\text{)}\\
&= \sum_{j=1..n} Pr(j) \cdot (x_0 - x_{\tau(i-1,j)})\\
&= \sum_{j=1..n} Pr(j) \cdot y_{\tau(i-1,j)} && \text{(due to (9))}\\
&= \sum_{j=1..n, j\neq W_i} Pr(j) \cdot y_{\tau(i-1,j)} + Pr(W_i) \cdot y_{\tau(i-1,W_i)}\\
&= z_{i-1} + Pr(W_i) \cdot y_{\tau(i-1,W_i)} && \text{(due to (10))}\\
&= z_{i-1} + Pr(W_i) \cdot y_i. && \text{(since } \tau(i-1, W_i) = i\text{)}
\end{aligned}$$

This further implies (12). ◀

Proof of (13). Assume $1 \le i < m$. We have

$$z_i = \sum_{1\le j\le n, j\neq W_{i+1}} Pr(j) \cdot y_{\tau(i,j)} = \sum_{1\le j\le n, j\neq W_{i+1}} Pr(j) \cdot y_{\tau(\pi_i,j)}$$

The first equation follows from the definition (10), whereas the second follows from (8).

On the other side, by the definition (10) of z_{π_i}, we have

$$z_{\pi_i} = \sum_{1\le j\le n, j\neq W_{\pi_i+1}} Pr(j) \cdot y_{\tau(\pi_i,j)} .$$

Therefore,

$$z_i - z_{\pi_i} = Pr(W_{\pi_i+1}) \cdot y_{\tau(\pi_i, W_{\pi_i+1}))} - Pr(W_{i+1}) \cdot y_{\tau(\pi_i, W_{i+1})}.$$

Substituting $\tau(\pi_i, W_{i+1})$ and $\tau(\pi_i, W_{\pi_i+1}))$ by σ_i and $\pi_i + 1$ respectively, we obtain (13). ◀

▶ Remark. Proving Formula 13 is the key step in designing our linear time algorithm. This formula shows that we can rapidly compute z_i from z_{π_i}. The proof of this formula mainly applies the properties of τ stated in Lemma 6.

In a more straight-forward way, we may apply Formula 10 instead of Formula 13 to compute z_i at Line 5. This would produce an alternative algorithm that runs in $O(m^2)$ time.

2.1 Preprocessing procedure – computing π and σ in linear time

The prefix function π can be computed in $O(m)$ time by using the famous KMP (Knuth-Morris-Pratt) algorithm, see [13]. Next, we assume that π is computed and we present the algorithm for computing $\sigma = (\sigma_1, \ldots, \sigma_{m-1})$. Recall that $\sigma_i = \tau(\pi_i, W_{i+1})$.

We describe the algorithm in Algorithm 2. It uses a Depth-First-Search (DFS).

Algorithm 2 Compute σ by a DFS

```
1:  Tr[1..n] is a temporary array.
2:  procedure ComputeSigma(i)          ▷ When entering this procedure, $\mathsf{Tr}[*] = \tau(i,*)$
3:      for $j : \pi(j) = i$ do
4:          $\sigma_j \leftarrow \mathsf{Tr}[W_{j+1}]$;
5:          $\mathsf{Tr}[W_{j+1}] \leftarrow j+1$;          ▷ Here, $\mathsf{Tr}[*]$ becomes $\tau(j,*)$
6:          ComputeSigma(j);
7:          $\mathsf{Tr}[W_{j+1}] \leftarrow \sigma_j$;          ▷ Let $\mathsf{Tr}[*]$ return to $\tau(i,*)$
8:      end for
9:  end procedure
10: $\mathsf{Tr}[1], \ldots, Tr[n] \leftarrow 0$;
11: $\mathsf{Tr}[W_1] \leftarrow 1$;          ▷ The last two lines make $\mathsf{Tr}[*] = \tau(0,*)$
12: ComputeSigma(0)
```

The correctness of Algorithm 2 is obvious. When entering the procedure ComputeSigma with parameter i, the temporary array $\mathsf{Tr}[1, \ldots, n]$ stores $\tau(i,1), \ldots, \tau(i,n)$.

This preprocessing procedure runs in $O(n+m)$ time.

▶ Remark. There is a chance that this preprocessing procedure may be improved to $O(m)$ time. However, for two reasons the $O(n+m)$ time solution is just fine. First, inputting the task instance would cost $O(n+m)$ time. Second, usually we can assume that $n \le m$. (If $n > m$, we may modify the input parameters and merge the redundant outcomes at first.)

3 An alternative method followed by Li's approach

In this section, we restate some beautiful results proved by Shuo-yen Robert Li in [14]. Then, we show that based on these results, we can design an alternative algorithm which computes $e_0, \ldots, e_{m-1}$ in $O(m)$ time. We then compare this algorithm to Algorithm 1.

▶ **Definition 7.** Let $A = A_1A_2 \ldots A_s$ and $W = W_1W_2 \ldots W_m$ be two strings over $\{1..n\}$. For every pair (i,j) of integers, write

$$\delta_{ij} = \begin{cases} Pr(W_j)^{-1}, & \text{if } 1 \le i \le s, 1 \le j \le m \text{ and } A_i = W_j; \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Then define

$$A * W = \delta_{11}\delta_{22} \ldots \delta_{ss} + \delta_{21}\delta_{32} \ldots \delta_{s,s-1} + \ldots + \delta_{s1} \tag{15}$$

▶ **Example 8.** Let $A =$ "2113", $W =$ "131". Assume that $Pr(1) = \frac{1}{2}, Pr(2) = \frac{1}{3}, Pr(3) = \frac{1}{6}$. Then $W * W = 2 \cdot 6 \cdot 2 + 0 \cdot 0 + 2 = 26$, and $A * W = 0 \cdot 0 \cdot 2 \cdot 0 + 2 \cdot 0 \cdot 0 + 2 \cdot 6 + 0 = 12$.

▶ **Lemma 9** ([14]). *Given a starting string A, the expected waiting time for a pattern W is*

$$W * W - A * W,$$

provided that W is not a substring of A. In particular, the waiting time of pattern W (without a starting sequence) is $W * W$.

By Lemma 9, we have the following equation for $e_0, \ldots, e_{m-1}$.

$$e_i = W * W - W^{(i)} * W. \tag{16}$$

In the following, we prove the following result.

▶ **Theorem 10.** *We can compute the values of $W^{(0)} * W = 0, W^{(1)} * W, \ldots, W^{(m)} * W = W * W$ altogether in $O(m)$ time, and thus compute $e_0, \ldots, e_{m-1}$ in $O(m)$ time.*

To prove this result, we first state the following equation of $W^{(i)} * W$.

Recall the periods set $\mathcal{P}(i)$ defined in Definition 4. Denote $\mathsf{prod}_h = \prod_{j=1}^{h} Pr(W_j)^{-1}$. Then,

$$W^{(i)} * W = \sum_{h \in \mathcal{P}(i)} \mathsf{prod}_h. \tag{17}$$

The proof of Equation 17 is deferred for a moment.

Furthermore, recall that

$$\mathcal{P}(i) = \begin{cases} \{0\}, & i = 0; \\ \mathcal{P}(\pi_i) \cup \{i\}, & i > 0. \end{cases}$$

We obtain the following equation based on (17).

$$W^{(i)} * W = \begin{cases} 0, & i = 0; \\ W^{(\pi_i)} * W + \mathsf{prod}_i, & i > 0. \end{cases} \tag{18}$$

According to this recursive equation, we obtain Theorem 10 immediately. The detailed algorithm is omitted. In the following we prove Equation 17.

Proof of (17). Set $A = W^{(i)}$. According to the definition of $A * W$ in (15),

$$\begin{aligned} W^{(i)} * W &= A * W \\ &= \delta_{11}\delta_{22} \ldots \delta_{ii} + \delta_{21}\delta_{32} \ldots \delta_{i,i-1} + \ldots + \delta_{i1} \\ &= [W^{(i)} \text{ is a suffix of } W^{(i)}] \cdot Pr(W_1)^{-1} \ldots Pr(W_i)^{-1} + \\ &\quad [W^{(i-1)} \text{ is a suffix of } W^{(i)}] \cdot Pr(W_1)^{-1} \ldots Pr(W_{i-1})^{-1} + \ldots \\ &\quad + [W^{(1)} \text{ is a suffix of } W^{(i)}] \cdot Pr(W_1)^{-1} \\ &= \sum_{1 \le h \le i} [W^{(h)} \text{ is a suffix of } W^{(i)}] \cdot \mathsf{prod}_i \\ &= \sum_{h \in \mathcal{P}(i)} \mathsf{prod}_i \end{aligned}$$

The last step is due to Lemma 5. ◀

▶ Remark. In designing the above linear algorithm for computing $W^{(0)} * W, \ldots, W^{(m)} * W$, the key is to apply the recursive formula (18). This formula tells us that we can compute $W^{(i)} * W$ rapidly from $W^{(\pi(i))} * W$. We note that the same technique is applied in the previous algorithm for computing $z_1, \ldots, z_{m-1}$ shown in Section 2.

Lemma 9 provides a concise and beautiful formula, which reveals many insights of the problem. However, its proof requires some advanced mathematic tools; for example, the Doob's fundamental theorem on stopping times of martingales. In addition, it only computes the expected waiting time, while our approach easily extends to compute the variance.

4 The multiple pattern case

In this section, we consider the multiple pattern case. Given a set of t patterns $W_1, \dots, W_t$. Suppose that they are reduced, which means that no pattern is a substring of another. We compute the expected waiting time till one of the pattern is observed in a run of experiments. We also compute the probability for each pattern to be first to appear.

Our method is similar to the one given by Li [14].

▶ **Lemma 11.** *For $1 \le i \le t$, let N_i be the random variable which indicate the waiting time till W_i is observed, and let p_i be the probability that W_i precedes the remaining $t-1$ patterns and be the first pattern to appear. Let $N = min(N_1, \dots, N_t)$. Let $e_{i,k}$ denote the expected stopping time of the Markov chain corresponding to W_i, starting from its state k. For every i, j such that $1 \le i, j \le t$, we denote by $k(i,j)$ the largest k such that $W_i^{(k)}$ is a suffix of W_j. Then*

$$e_{i,0} = EN + \sum_{j=1}^{t} p_j e_{i,k(i,j)} \quad (1 \le i \le t) \tag{19}$$

Proof.

$$\begin{aligned} e_{i,0} &= E(N_i) \\ &= EN + E(N_i - N) \\ &= EN + \sum_{j=1}^{t} E(N_i - N \mid N = N_j)) \cdot Pr(N = N_j) \\ &= EN + \sum_{j=1}^{t} p_j e_{i,k(i,j)} \end{aligned}$$ ◀

Note that the value of e can be computed by our algorithm for the simple pattern case. Also note that we have another simple equation as follows.

$$\sum_{j=1}^{t} p_j = 1. \tag{20}$$

In the matrix form, the t equations in (19) and the equation in (20) become:

$$\begin{pmatrix} 0 & 1 & \dots & 1 \\ 1 & e_{1,k(1,1)} & \dots & e_{1,k(1,t)} \\ \dots & \dots & \dots & \dots \\ 1 & e_{t,k(t,1)} & \dots & e_{t,k(t,t)} \end{pmatrix} \begin{pmatrix} EN \\ p_1 \\ \dots \\ p_t \end{pmatrix} = \begin{pmatrix} 1 \\ e_{1,0} \\ \dots \\ e_{t,0} \end{pmatrix}. \tag{21}$$

Let M denote the coefficient matrix of this system. Li [14] proved that M is nonsingular. So, we can compute $EN, p_1, \dots, p_t$ by solving this system of linear equations.

5 Higher moments of the waiting time of a single pattern

For any integer $k \ge 0$ and for any $0 \le i \le m$, denote

$$g_{k,i} := \sum_{t \ge 0} t^k \cdot Pr(\text{It takes } t \text{ steps to get the terminal state from state } i). \tag{22}$$

In Subsection 1.2, we defined array $e = g_1$ and array $f = g_2$ and we show that both of them can be computed efficiently by applying Theorem 2. In the following we show that arrays $g_1, \ldots, g_k$ can be computed altogether in $O(m \cdot k^2)$ time (after the preprocessing procedure for computing σ). As a corollary, we obtain:

▶ **Corollary 12.** *Given a pattern W with length m, the first k moments of the waiting time of W can be computed in $O(n + m \cdot k^2)$ time.*

We state an equation of g_k at first. For $k \geq 1$,

$$g_{k,i} = \begin{cases} 0, & i = m; \\ \left(\sum_{j=1}^{n} Pr(j) \cdot g_{k,\tau(i,j)}\right) + \sum_{h=0}^{k-1} \binom{k}{h} (-1)^{k-1-h} g_{h,i}, & i = 0..m-1. \end{cases} \tag{23}$$

▶ **Example 13.** For $i < m$, we have

$$\begin{aligned}
g_{1,i} &= \left(\sum_{j=1}^{n} Pr(j) \cdot g_{1,\tau(i,j)}\right) + g_{0,i}; \\
g_{2,i} &= \left(\sum_{j=1}^{n} Pr(j) \cdot g_{2,\tau(i,j)}\right) + 2g_{1,i} - g_{0,i}; \\
g_{3,i} &= \left(\sum_{j=1}^{n} Pr(j) \cdot g_{3,\tau(i,j)}\right) + 3g_{2,i} - 3g_{1,i} + g_{0,i}; \\
g_{4,i} &= \left(\sum_{j=1}^{n} Pr(j) \cdot g_{4,\tau(i,j)}\right) + 4g_{3,i} - 6g_{2,i} + 4g_{1,i} - g_{0,i}; \\
&\ldots
\end{aligned}$$

According to (23), we can apply Theorem 2 to compute g_k after we have computed $g_0, \ldots, g_{k-1}$. Therefore, we obtain the aforementioned results and Corollary 12.

We apply the following identity to prove (23).

$$\sum_{h=l}^{k} \binom{k}{h} \binom{h}{l} (-1)^h = 0 \qquad \text{(when } k > l \geq 0\text{)} \tag{24}$$

Proof of (24).

$$\sum_{h=l}^{k} \binom{k}{h} \binom{h}{l} (-1)^h = \sum_{h=l}^{k} \binom{k}{l} \binom{k-l}{h-l} (-1)^h = \binom{k}{l} \sum_{h=l}^{k} \binom{k-l}{h-l} (-1)^h = 0$$ ◀

Proof of (23). Assume $i < m$; otherwise it is trivial. By First-Step-Analysis,

$$\begin{aligned}
g_{k,i} &= \sum\nolimits_{j=1}^{n} Pr(j) \sum\nolimits_{t\geq 0} (t+1)^k \cdot Pr(\text{It takes } t \text{ steps to get state } m \text{ from state } \tau(i,j)) \\
&= \sum\nolimits_{j=1}^{n} Pr(j) \sum\nolimits_{t\geq 0} \cdot (t^k + \sum\nolimits_{h=0}^{k-1} \binom{k}{h} t^h) \cdot Pr(\text{It takes } t \text{ steps ... from state } \tau(i,j)) \\
&= \left(\sum\nolimits_{j=1}^{n} Pr(j) \cdot g_{k,\tau(i,j)}\right) + \sum\nolimits_{h=0}^{k-1} \binom{k}{h} \sum\nolimits_{j=1}^{n} Pr(j) g_{h,\tau(i,j)} \\
&= \left(\sum\nolimits_{j=1}^{n} Pr(j) \cdot g_{k,\tau(i,j)}\right) + \sum\nolimits_{h=0}^{k-1} \binom{k}{h} \left(g_{h,i} - \sum\nolimits_{l=0}^{h-1} \binom{h}{l} (-1)^{h-1-l} g_{l,i} \right) \\
&= \left(\sum\nolimits_{j=1}^{n} Pr(j) \cdot g_{k,\tau(i,j)}\right) - \sum\nolimits_{h=0}^{k-1} \binom{k}{h} \sum\nolimits_{l=0}^{h} \binom{h}{l} (-1)^{h-1-l} g_{l,i} \\
&= \left(\sum\nolimits_{j=1}^{n} Pr(j) \cdot g_{k,\tau(i,j)}\right) - \sum\nolimits_{l=0}^{k-1} g_{l,i} \cdot \sum\nolimits_{h=l}^{k-1} \left(\binom{k}{h} \binom{h}{l} (-1)^{h-1-l} \right) \\
&= \left(\sum\nolimits_{j=1}^{n} Pr(j) \cdot g_{k,\tau(i,j)}\right) - \sum\nolimits_{l=0}^{k-1} g_{l,i} \cdot \left(\binom{k}{l} (-1)^{k-l} \right) \\
&= \left(\sum\nolimits_{j=1}^{n} Pr(j) \cdot g_{k,\tau(i,j)}\right) + \sum\nolimits_{h=0}^{k-1} g_{h,i} \cdot \left(\binom{k}{h} (-1)^{k-1-h} \right)
\end{aligned}$$

Note that the second last step applies Identity 24. ◀

Acknowledgements. The author would like to thank Zhiyi Huang for taking part in fruitful discussions, thank Jian Li and Andrew Yao and for much help, and thank several anonymous reviewers from MFCS16 and ISAAC16 for much great suggestions.

References

1 Anonymous. Penney's game. Technical report, Wikipedia, 2016. URL: https://en.wikipedia.org/wiki/Penney%27s_game.

2 J. A. D. Aston and D. E. K. Martin. Waiting time distributions of competing patterns in higher-order markovian sequences. *Journal of Applied Probability*, 42(4):977–988, 2005.

3 N. Balakrishnan and M. V. Koutras. *Runs and Scans with Applications*. Wiley, 2002.

4 S. Breen, M. S. Waterman, and N. Zhang. Renewal theory for several patterns. *Journal of Applied Probability*, 22(1):228–234, 1985.

5 O. Chrysaphinou and S. Papastavridis. The occurrence of sequence patterns in repeated dependent experiments. *Theory of Probability & Its Applications*, 35(1):145–152, 1991. doi:10.1137/1135015.

6 J. A. Csirik. Optimal strategy for the first player in the penney ante game. *Combinatorics, Probability and Computing*, 1:311–321, 1992. doi:10.1017/S0963548300000365.

7 W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. Wiley, 3 edition, 1968.

8 J. C. Fu and W. Y. W. Lou. *Distribution Theory of Runs and Patterns and Its Applications*. World Scientific Publishing, 2003.

9 I. Fudos, E. Pitoura, and W. Szpankowski. On pattern occurrences in a random text. *Information Processing Letters*, 57(6):307–312, 1996.

10 M. Gardner. On the paradoxical situations that arise from nontransitive relations. *Scientific American*, 231(4), 1974.

11 R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

12 L. J. Guibas and A. M. Odlyzko. String overlaps, pattern matching, and nontransitive games. *Journal of Combinatorial Theory, Series A*, 30(2):183–208, 1981. doi:10.1016/0097-3165(81)90005-4.

13 D. E. Knuth, Jr. J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977. doi:10.1137/0206024.

14 S. R. Li. A martingale approach to the study of occurrence of sequence patterns in repeated experiments. *The Annals of Probability*, 8(6):1171–1176, 1980.

15 M. E. Lladser, M. D. Betterton, and R. Knight. Multiple pattern matching: a markov chain approach. *Journal of Mathematical Biology*, 56(1):51–92, 2007. doi:10.1007/s00285-007-0109-3.

16 M. Lothaire. *Applied Combinatorics on Words.* Cambridge University Press, 2005.

17 W. Penney. Problem 95: Penney-ante. *Journal of Recreational Mathematics*, page 241, 1969.

18 V. Pozdnyakov. On occurrence of patterns in markov chains: Method of gambling teams. *Statistics & Probability Letters*, 78(16):2762–2767, 2008. doi:10.1016/j.spl.2008.03.023.

19 V. Pozdnyakov and M. Kulldorff. Waiting times for patterns and a method of gambling teams. *The American Mathematical Monthly*, 113(2):134–143, 2006.

20 M. Régnier. A unified approach to word occurrence probabilities. *Discrete Applied Mathematics*, 104:259–280, 2000.

21 M. Régnier and W. Szpankowski. On pattern frequency occurrences in a markovian sequence. *Algorithmica*, 22(4):631–649, 1998. doi:10.1007/PL00009244.

22 G. Reinert, S. Schbath, and M. S. Waterman. Probabilistic and statistical properties of words: An overview. *Journal of Computational Biology*, 7(2):1–46, 2000.

23 A. D. Solov'ev. A combinatorial identity and its application to the problem concerning the first occurrence of a rare event. *Theory of Probability & Its Applications*, 11(2):276–282, 1966. doi:10.1137/1111022.