Rose-Hulman Undergraduate Mathematics Journal

Volume 12
Issue 2

Article 11

Exploring String Patterns with Trees

Deborah Sutton
Purdue University, dasutton@purdue.edu

Booi Yee Wong
Purdue University, wongb@purdue.edu

Follow this and additional works at: https://scholar.rose-hulman.edu/rhumj

Recommended Citation
Sutton, Deborah and Wong, Booi Yee (2011) "Exploring String Patterns with Trees," *Rose-Hulman Undergraduate Mathematics Journal*: Vol. 12 : Iss. 2 , Article 11.
Available at: https://scholar.rose-hulman.edu/rhumj/vol12/iss2/11

Rose-
Hulman
Undergraduate
Mathematics
Journal

Exploring String Patterns with Trees

Deborah Sutton[a] Booi Yee Wong[b]

Volume 12, No. 2, Fall 2011

Sponsored by

Rose-Hulman Institute of Technology
Department of Mathematics
Terre Haute, IN 47803
Email: mathjournal@rose-hulman.edu
http://www.rose-hulman.edu/mathjournal

[a]Purdue University
[b]Purdue University

Rose-Hulman Undergraduate Mathematics Journal
Volume 12, No. 2, Fall 2011

Exploring String Patterns with Trees

Deborah Sutton Booi Yee Wong

Abstract. The combination of the fields of probability and combinatorics is currently an object of much research. However, not many undergraduates or lay people have the opportunity to see how these areas can work together. We present what we hope is an accessible introduction to the possibilities easily available to many more people through the use of many examples and understandable explanations. We introduce topics of generating functions and tree structures formed through both independent strings and suffixes, as well as how we can find correlation polynomials, expected values, second moments, and variances of the number of nodes in a tree using indicator functions. Then we show a higher order example that includes matrices as the basis for its generating functions. The study of this unique field has many applications in areas including data compression, computational biology with the human genome, and computer science with binary strings.

Acknowledgements: The first author is thankful for the generosity of Kathryn and Art Lorenz through the Joseph Ruzicka Undergraduate Fellowship, which made this project possible.
We dedicate this paper to the memory of Dr. Philippe Flajolet (1948–2011). His work in combinatorics was very influential in our research.
Both authors would like to thank Dr. Mark Daniel Ward for all his guidance, support, teaching, and just being an all-around fun person who made our research experience special. Dr. Ward's work is supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.
We also thank Dr. David Rader for his helpful editorial feedback and guidance throughout the process of revision.

1 Introduction

This is a survey of techniques for the analysis of strings and tree structures. We survey ways to analyze the repeating occurrence of a specific pattern in a longer string. Applications related to the analysis of repeated patterns include mapping DNA sequences, identifying genes that carry particular biological information, compressing data with redundant information, synchronizing codes in computer programming, and searching for information in telecommunications. In these fields, we can think of DNA sequences, text, binary data, and programming codes as strings, and these strings contain repeated patterns. As we examine such strings we also account for the possibility that a given pattern overlaps with itself, which makes our analysis more complicated than the scenarios typically studied at the undergraduate level. We consider using combinatorial analysis together with trees to study repeated patterns.

In fact, the analysis of repeated patterns using combinatorial structures has been well studied by many authors. This analysis has introduced a few languages to denote occurrences of patterns in a string by taking into account the overlaps of the pattern with itself and the overlaps with another pattern [2, 3]. These languages can be translated into generating functions so that probabilities, expected values, and variances for the number of occurrences can be found for any length of string [1]. We also show a two-dimensional example to analyze the recurrence of two particular patterns in a longer string. In [4], the pattern of occurrences of two DNA base sequences in a DNA strand is analyzed through correlational languages to further account for the overlaps between two patterns besides the overlap with itself. However, we further develop the analysis of DNA repeated patterns by using trees.

We use a suffix tree, which is built from a common string, to allow us to recognize repeated patterns and combinations of patterns within the string. Repeated patterns can be identified by the presence of internal nodes in the tree, which is shown through indicator functions. (In contrast, a trie is built from independently-generated strings.)

Although the study of combinatorial analysis and trees is well known, the connection between these two fields in the analysis of repeated patterns is seldom emphasized. A tree helps yield insights about repeated patterns, and the combinatorial analysis gives methods for finding probabilities, means, and variances of the mapped patterns.

In this paper, we first present definitions of combinatorial languages, their associated generating functions, tree structures, and some associated indicator functions. Then, we present a few examples to illustrate how these techniques work together to aid in analyzing repeating patterns in a string.

2 Background Material

In this section, we define the languages and the associated generating functions that will be used throughout the rest of the paper. Example 2.1, found at the end of this section, gives a concrete example of many of the definitions in this section.

2.1 Languages and Probability Generating Functions

A language is just a set of words that each have finite length. Throughout this paper, we use $\mathcal{A}$ as the alphabet, i.e., as the set of characters from which letters in the words are drawn. We also use concatenation of words and languages when two words or two languages are written side-by-side. For instance, if v and w are words of length 3 and 7, with letters from $\mathcal{A}$, then vw is a word of length 10 that also has letters from $\mathcal{A}$. Similarly, if $\mathcal{V}$ and $\mathcal{W}$ are two languages (i.e., two collections of words), then $\mathcal{VW}$ is the set of words formed by concatenating a word from $\mathcal{V}$ with a word from $\mathcal{W}$. I.e., $\mathcal{VW} = \{vw \mid v \in \mathcal{V}, w \in \mathcal{W}\}$.

We also use the Kleene-star notation, commonly found in automata theory. Namely, $\mathcal{A}^*$ is the set of all words with finitely many letters from $\mathcal{A}$. Similarly, $\mathcal{W}^*$ consists of the set of words obtained by concatenating together finitely many words from $\mathcal{W}$, i.e.,

$$\mathcal{W}^* = \bigcup_{j \geq 0} \{w_1 w_2 \dots w_j \mid w_k \in \mathcal{W}\}.$$

We use ε to denote the empty word, i.e., the word of length 0.

A note about absolute value signs: With collections of letters, such as $|\mathcal{A}|$, the absolute value sign denotes the number of letters, e.g., if $\mathcal{A}$ is the English alphabet, then $|\mathcal{A}| = 26$. On the other hand, the absolutely value of a word is the length of the word, i.e., if w is a word of length 8, then $|w| = 8$. This notation is standard throughout the literature on combinatorics on words.

We use three key languages while analyzing the occurrences of word w in a string. We follow the notation in [2]. We define

$$\begin{aligned}
\mathcal{R}_w &= \{\text{all words that have exactly one } w\text{, occurring at right hand end}\},\\
\mathcal{M}_w &= \{\text{set of words } v \text{ with property } wv \text{ has exactly two } w\text{'s, one at left, one at right.}\},\\
\mathcal{U}_w &= \{\text{set of words } v \text{ with property } wv \text{ has exactly one } w\}.
\end{aligned}$$

Using the languages $\mathcal{R}_w, \mathcal{M}_w$, and $\mathcal{U}_w$, we can describe different types of words. For example,

1. If we want a symbol to express the set of words that contains just one occurrence of w, we can use the language $\mathcal{R}_w\mathcal{U}_w$ because $\mathcal{R}_w$ has only one w at the end of it and $\mathcal{U}_w$ contains no additional occurrences of w.

2. If we want to have a language that has one or more occurrences of w, we can use the language of words of the form $\mathcal{R}_w\mathcal{A}^*$.

3. The language of words that has exactly two occurrences of w is $\mathcal{R}_w\mathcal{M}_w\mathcal{U}_w$.

Another concept we need to address before we can go very far into any combinatorial languages is the notion of probabilistic generating functions. First we need the notation of probability. We assume throughout the paper that the letters of a word are selected independently (although more general models are possible—such as Markov models, with dependencies among the letters—we do not consider such models in this paper). To each

letter in $\mathcal{A}$, there is an associated probability of occurrence. I.e., to the letters $a_1, \ldots, a_{|\mathcal{A}|} \in \mathcal{A}$, we assign probabilities $p_1, \ldots, p_{|\mathcal{A}|} \geq 0$ such that $p_1 + \cdots + p_{|\mathcal{A}|} = 1$. Thus, for example, if w is a word of length 8, with 3 occurrences of a_1 and 5 occurrences of a_2, then $\mathbf{P}(w) = p_1^3 p_2^5$. In general, if w has exactly c_j occurrences of letter a_j, then

$$\mathbf{P}(w) = \prod_{j=1}^{|\mathcal{A}|} p_j^{c_j}.$$

In particular, the probability of the empty word is the trivial product—the multiplicative identity 1—since the empty word has no letters. I.e., $\mathbf{P}(\varepsilon) = 1$.

The probability generating function of any language $\mathcal{L}$ is

$$L(z) = \sum_{w \in \mathcal{L}} \mathbf{P}(w) z^{|w|}.$$

Using this approach we can write generating functions for any of the languages we described above. For example, the probability generating function associated with the language $\mathcal{U}_w$ is

$$U_w(z) = \sum_{u \in \mathcal{U}_w} \mathbf{P}(u) z^{|u|}.$$

The probability generating functions for other languages are defined in analogous ways. The variable z is related to the size of the string, as the power n to which z is raised marks the objects of size n in the set. Also, a very beneficial quality of generating functions is that they can be easily manipulated as power series.

2.2 Combinatorial and Probabilistic Analysis on Words

A central topic of the analysis of words is the method of characterizing the overlap(s) of a word with itself. We consider overlaps by introducing the autocorrelation polynomial $S_w(z)$, which is the degree to which a word w overlaps with itself. We define

$$S_w(z) = \sum_{i \in \mathcal{P}(w)} \mathbf{P}(w_{i+1}, \ldots, w_m) z^{m-i}, \tag{1}$$

where $\mathcal{P}(w)$ is the set of sizes of the overlaps, and $m = |w|$ is the length of word w. We are again following the notation from [2]. For instance, if $w = \mathbf{abcabcabca}$, then w has overlaps with itself of lengths 1, 4, 7, and 10. So $\mathcal{P}(w) = \{1, 4, 7, 10\}$, and the autocorrelation polynomial of w is

$$S_w(z) = \mathbf{P}(\mathbf{bcabcabca}) z^1 + \mathbf{P}(\mathbf{bcabca}) z^4 + \mathbf{P}(\mathbf{bca}) z^7 + \mathbf{P}(\varepsilon) z^{10}.$$

We can create probabilistic generating functions, $M_w(z), U_w(z), R_w(z)$, for the three languages we introduced above, $\mathcal{R}_w, \mathcal{M}_w$, and $\mathcal{U}_w$. In finding the generating functions for the

above languages, we have to find the generating functions for $\mathcal{M}_w$ followed by $\mathcal{U}_w$ and $\mathcal{R}_w$. The prior generating function is needed in calculating the next generating function. These formulae, and the reasoning for their derivations, are given in [2]. We have:

$$\begin{aligned}\frac{1}{1-M_w(z)} &= S_w(z) + \frac{\mathbf{P}(w)z^m}{1-z},\\ U_w(z) &= \frac{M_w(z)-1}{z-1},\\ R_w(z) &= \mathbf{P}(w)z^m U_w(z).\end{aligned}$$

These generating functions will be the fundamental building blocks in constructing larger probability generating functions. In many of these constructions, the denominator has a common form that is so pervasive that we go ahead and define it here. We let $D_w(z)$ denote

$$D_w(z) = (1-z)S_w(z) + z^m\mathbf{P}(w).$$

(Again, the notation $D_w(z)$ follows the notation in [2].)

To find the expected number of occurrences of w in a string of length n, we first add up the product of the generating functions and the probabilities for any number j of occurrences of w, along with u raised to that number. This results in multivariate generating functions. We will further illustrate why we add the variable u into the equations as we give more examples. These generating functions are not limited to the case where we are looking for one occurrence of w in a string of length n, but indeed will work for any number of occurrences of w in a random string of length n. Define X_n as the number of occurrence(s) of w in a random word of length n. Then we define the multivariate generating function $h(u,z)$, where the power of u marks the number of occurrences of w in a word v, and the power of z marks the length of a word v. I.e., we define

$$h(u,z) := \sum_{v\in\mathcal{A}^*} \mathbf{P}(v)u^{\#\text{ of occurrences of } w \text{ in } v}z^{|v|}.$$

Collecting the words v according to their length n, we have

$$h(u,z) = \sum_{n=0}^{\infty}\sum_{j=1}^{\infty} P(X_n = j)u^j z^n,$$

or equivalently,

$$h(u,z) = \sum_{n=0}^{\infty} \mathbb{E}(u^{X_n})z^n.$$

Since $\mathcal{R}$ is the set of words that have exactly one w at the right, and each subsequent (concatenated) occurrence of a word from $\mathcal{M}_w$ yields another w, and finally a concatenated

word from $\mathcal{U}_w$ does not yield any additional occurrences of w, it follows that

$$\begin{aligned} h(u,z) &= \sum_{j=1}^{\infty} R_w(z) u \overbrace{M_w(z)u, \ldots, M_w(z)u}^{j-1} U_w(z) \\ &= R_w(z)u\Big(\sum_{j=1}^{\infty}(M_w(z)u)^{j-1}\Big)U_w(z) \\ &= R_w(z)u\frac{1}{1-uM_w(z)}U_w(z). \end{aligned} \tag{2}$$

Then, we take the first derivative of $h(u,z)$ with respect to u and plug in $u=1$. Here is the explanation for this technique: By looking at the original form of the generating function,

$$\sum_{n=0}^{\infty}\sum_{j=1}^{\infty} P(X_n = j)u^j z^n.$$

When we take the first derivative with respect to u, we get

$$\sum_{n=0}^{\infty}\sum_{j=1}^{\infty} P(X_n = j)ju^{j-1} z^n.$$

Substituting $u=1$ turns this expression into

$$\sum_{n=0}^{\infty}\sum_{j=1}^{\infty} P(X_n = j)jz^n = \sum_{n=0}^{\infty} E(X_n)z^n.$$

Here we see how the variable u helps us to transform the multivariate generating function $h(u,z)$ into the generating function for the expected value by multiplying $P(X_n = j)$ with j.

Since we now have the expected value, it does not take too much more to be able to get the variance. Indeed, the variance can be expressed as this sum:

$$\mathrm{Var}(X_n) = E[(X_n)(X_n-1)] + E[X_n] - (E[X_n])^2.$$

All that remains to calculate the variance is the second falling moment, $E[(X_n)(X_n-1)]$. Once again we go back to our original generating function. The second falling moment comes from the second derivative with respect to u with the substitution $u=1$. So the second derivative looks like

$$\sum_{n=0}^{\infty}\sum_{j=1}^{\infty} P(X_n = j)j(j-1)u^{j-2} z^n,$$

and substituting $u=1$ gives us

$$\sum_{n=0}^{\infty}\sum_{j=1}^{\infty} P(X_n = j)j(j-1)z^n = \sum_{n=0}^{\infty} E(X_n)(X_n-1)z^n.$$

After defining functions for probability, expected value and variance for one repeating pattern in a string, we can now move on to two dimensional cases with two repeating patterns in a string. We not only want to look at how the word w overlaps with itself and the word v overlaps with itself, but we would also like to analyze how words w and v overlap with each other. This leads us into a discussion of the correlation polynomial.

Definition: The correlation set, $A_{v,w}$ is the set of suffixes of w that follow the overlapping portion of a suffix of v with a prefix of word w. For example, if $v =$ `bbbabcab`, and $w =$ `abcabcabca`, then there are two ways in which a suffix of v overlaps with a prefix of w, namely, the overlap could be `abcab` or `ab`. These overlaps correspond to the following suffixes of w: `cabca` and `cabcabca`. Thus $A_{v,w}(z) = \mathbf{P}(cabca)z^5 + \mathbf{P}(cabcabca)z^8$ in this example.

In general, for $w = w$, the correlation set is the same as the autocorrelation set introduced in (1) (which measures the degree to which a word overlaps with itself).

The correlation polynomial of words v and w is

$$A_{v,w}(z) = \sum_{u \in A_{v,w}} \mathbf{P}(u) z^{|u|}.$$

The correlation matrix is

$$\mathbb{A}_{w,v}(z) = \begin{bmatrix} A_{w,w}(z) & A_{w,v}(z) \\ A_{v,w}(z) & A_{v,v}(z) \end{bmatrix}.$$

In particular, $A_{w,w}(z)$ is the autocorrelation polynomial for word w with itself, i.e., $A_{w,w}(z) = S_w(z)$.

In the multivariate case, instead of one autocorrelation polynomial $S_w(z)$, we have the correlation matrix $\mathbb{A}_{w,v}(z)$. (We can suppress the w and v from the notation when the two words are clear from the context.) Because of this, $D_w(z), U_w(z)$, and $R_w(z)$ all get analogous matrix representations too, as seen below.

We use here the notation from Regniér and Denise [4], in the multivariate generating functions (see their section 4).

Define $\widetilde{\mathcal{R}}_i$ as the set of all words with exactly one occurrence of the ith word, with this occurrence happening at the end of the word *and no occurrences of the jth word.* (This is analogous to the one dimensional $\mathcal{R}$, but with an additional restriction—which we have italicized for emphasis—on word j.)

As before, $\mathcal{A}^*$ corresponds to the set of all words of finite length, and so the probability generating function of $\mathcal{A}^*$ is $\frac{1}{1-z}$.

To get the generating functions for the above languages, we need to find $\mathbb{D}_w(z)$. In the one dimensional case, we could express $M_w(z)$ as $1-\frac{1-z}{D_w(z)}$, where $D_w(z) = S_w(z)(1-z)+\mathbf{P}(w)z^m$. Then $U_w(z) = \frac{1}{D_w(z)}$ and $R_w(z) = \mathbf{P}(w)z^m - \frac{1}{D_w(z)}$. In the multivariate case, instead of one autocorrelation polynomial $S_w(z)$, we have the correlation matrix $\mathbb{A}(z)$. Because of this, $D_w(z), U_w(z)$, and $R_w(z)$ all become matrices. Thus we can find $\mathbb{D}(z)$ corresponding to words w_i and w_j by:

$$\mathbb{D}(z) = (1-z)\mathbb{A}(z) + \begin{bmatrix} \mathbf{P}(w_i)z^{m_1} & \mathbf{P}(w_j)z^{m_2} \\ \mathbf{P}(w_i)z^{m_1} & \mathbf{P}(w_j)z^{m_2} \end{bmatrix},$$

where m_1 is the length of w_i and m_2 is the length of w_j.

Since we know $\mathbb{D}(z)$, we can use its matrix inverse $\mathbb{D}(z)^{-1}$ to compute the generating functions for $\widetilde{\mathcal{R}}_i$, $\widetilde{\mathcal{R}}_j$, $\mathcal{M}_{i,j}$, $\widetilde{\mathcal{U}}_i$, and $\widetilde{\mathcal{U}}_j$ as shown in the formulae below.

$$\begin{aligned}(\widetilde{R}_i(z), \widetilde{R}_j(z)) &= (\mathbf{P}(w_i)z^{m_1}, \mathbf{P}(w_j)z^{m_2})\mathbb{D}(z)^{-1},\\ \mathbb{M}(z) &= \begin{bmatrix} M_{i,i}(z) & M_{i,j}(z) \\ M_{j,i}(z) & M_{j,j}(z) \end{bmatrix}\\ &= \mathbb{I} - (1-z)\mathbb{D}(z)^{-1},\\ \begin{bmatrix} \widetilde{U}_i(z) \\ \widetilde{U}_j(z) \end{bmatrix} &= \mathbb{D}(z)^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.\end{aligned}$$

The matrix $\mathbb{M}$ contains $M_{1,1}(z)$, $M_{1,2}(z)$, $M_{2,1}(z)$, and $M_{2,2}(z)$ as its terms. Thus

$$\begin{aligned}M^*_{1,1}(z) &= \frac{1}{1 - M_{1,1}(z)},\\ M^*_{2,2}(z) &= \frac{1}{1 - M_{2,2}(z)}.\end{aligned}$$

2.3 Tree Structures

After introducing definitions related to combinatorial analysis, we now introduce definitions regarding tree structures. We consider two kinds of trees: retrieval trees and suffix trees. A tree built from independent strings is a retrieval tree, commonly referred to as a trie. A tree built from the suffixes of a common string is a suffix tree. By separating the strings according to their characters at each level of the tree, all strings will be allocated to distinct locations. Splitting at the kth level is dependent on the kth character. The level of the tree represents the length of the pattern. If there are two or more strings beginning with the same pattern of length k, there will be an internal node at level k of the tree. We use indicator functions to denote the presence of the internal nodes. I.e.,

$$I_w = \begin{cases} 1 & \text{if the internal node corresponding to } w \text{ is present,} \\ 0 & \text{otherwise.} \end{cases}$$

To find the expected number of internal nodes at level k, we multiply the value of each indicator, either 0 or 1, by the probability of two or more occurrences of each pattern at level k and find their sum. We denote the pattern corresponding to a node at level k of the tree as $y \in \mathcal{A}^k$, where $\mathcal{A}^k$ is the set of all words of length k. If the internal node of a particular pattern is absent, it will not contribute to the expected number of internal nodes because of its 0 indicator value. (We are suppressing dependence on k throughout the following notation.) Here we define $X_{n,k}$ as the number of internal nodes at level k, when n

strings are inserted into a trie or suffix tree. Thus

$$X_{n,k} = \text{number of internal nodes at level } k \text{ when tree is built over } n \text{ nodes}$$
$$= \sum_{y\in\mathcal{A}^k} I_{n,y}, \text{ where } I_{n,y} = \begin{cases} 1 & \text{if the internal node corresponding to } y \text{ is present,} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\begin{aligned} E(X_{n,k}) &= \sum_{y\in\mathcal{A}^k} E(I_{n,y}) \\ &= \sum_{y\in\mathcal{A}^k} \mathbf{P}(I_{n,y}=1) \qquad \text{since } E(I_{n,y}) = P(I_{n,y}=1) \\ &= \sum_{y\in\mathcal{A}^k} (\text{probability of two or more strings beginning with } y). \end{aligned}$$

Next, we are going to find variance of the number of expected nodes at level k of the tree, $\mathrm{Var}(X_{n,k})$.

$$\begin{aligned} \mathrm{Var}(X_{n,k}) &= \mathrm{Var}\Big(\sum_{y\in\mathcal{A}^k} I_{n,y}\Big) \\ &= E\Big(\Big(\sum_{y\in\mathcal{A}^k} I_{n,y}\Big)^2\Big) - \Big(E\Big(\sum_{y\in\mathcal{A}^k} I_{n,y}\Big)\Big)^2 \\ &= E\Big[\sum_{y\in\mathcal{A}^k} I_{n,y} \sum_{z\in\mathcal{A}^k} I_{n,z}\Big] - \Big(\sum_{y\in\mathcal{A}^k} E(I_{n,y})\Big)^2 \\ &= E\Big(\sum_{y\neq z} I_{n,y}I_{n,z} + \sum_{y} I_{n,y}\Big) - (E(X_{n,k}))^2 \\ &= E\Big(\sum_{y\neq z} I_{n,y}I_{n,z}\Big) + E\Big(\sum_{y} I_{n,y}\Big) - (E(X_{n,k}))^2 \\ &= \sum_{y\neq z} E(I_{n,y}I_{n,z}) + E(X_{n,k}) - (E(X_{n,k}))^2. \end{aligned} \tag{3}$$

We can obtain the values of $E(I_{n,y}) = P(I_{n,y}=1)$ and $E(I_{n,y}I_{n,z}) = P(I_{n,y}=1 \text{ and } I_{n,z}=1)$ from the probability generating functions discussed earlier. Both internal nodes $I_{n,y}$ and $I_{n,z}$ will occur when there are two or more occurrences of each pattern of y and z. The probability of two or more occurrences of each pattern y and z can be calculated through the correlational languages introduced earlier.

Example 2.1 *Now we present a concrete example to demonstrate many of the aspects of the definitions and concepts. We would like to analyze the number of occurrences of a word,* $w = \texttt{aaaaa}$ *in a longer text.*

aaaaa	
aaaaa	1
aaaaa	1
aaaaa	1
aaaaa	1
aaaaa	1

Figure 1: Self-overlaps of the word aaaaa

In a stochastic setting, we could find the probability of the word occurring a certain number of times in a randomly generated string of length n. We are using the Bernoulli (i.e. memoryless) model for the string, so the letters are independently generated according to a possibly biased source. According to the Bernoulli model, we have $\mathbf{P}(\mathtt{a}) = p$, $\mathbf{P}(w) = p^5$. *The first step in any such problem is to find the autocorrelation polynomial, $S_w(z)$, which sums up the probability of the remaining letters at each overlap. In our case,* $\mathcal{P}(w) = \{1, 2, 3, 4, 5\}$ *and* $m = 5$.

Although one may easily get caught up in all this notation, finding the autocorrelation polynomial is quite simple. All it takes is looking at the possible ways a word overlaps with itself. (See Figure 1.) Start at the left of the word. All words have the trivial overlap that is represented by 1 *in $S_w(z)$; that is, all words overlap perfectly with themselves when lined up directly on top of each other. Then move to the second letter. Is it possible for the word to overlap starting at the second letter if you just add one letter at the end? In the case of* aaaaa, *it is. The probability of this overlap is the probability of the letter you would need to have at the end, in this case,* a. *So* $\mathbf{P}(\mathtt{a}) = p$ *is the coefficient of z. Then you move another letter to the right and see if it would be possible for the word to overlap there. Follow the same procedure until you have checked all the letters in the word for possible overlaps. Figure 1 is a chart which shows the overlaps of* aaaaa *with itself. The presence of an overlap is indicated by a* 1 *on the right. The case of* aaaaa *is very special, because it can always overlap with itself just by adding some number of* a*'s on the end. The autocorrelation polynomial turns out to be*

$$S_w(z) = 1 + pz + p^2z^2 + p^3z^3 + p^4z^4.$$

To become more familiar with the idea of autocorrelation polynomials, it may be helpful to think of some general cases. As we have already seen, every word will have the trivial overlap.

1. *What characteristics must a word have to overlap at the second letter? Upon inspection, we see that this can only happen if the word is made of only one letter.*

2. *What about words that overlap at the third letter? These words are characterized by two repeating letters.*

3. *How about words that overlap at the last letter? This set of words is much more common. Any words that have the same first and last letter will overlap at the end,*

so this is a much easier constraint to satisfy. For these reasons, most autocorrelation polynomials have 1 *as their first term, then have zero or very few terms until the higher order terms at the end.*

Through Maple, we find the generating functions for $\mathcal{R}_w, \mathcal{M}_w$, *and* $\mathcal{U}_w$ *for* $w =$ `aaaaa` *are:*

$$\begin{aligned} R_w(z) &= \frac{p^5 z^5}{D_w(z)}, \\ M_w(z) &= \frac{pz(D_w(z) + p^4 z^4 - p^5 z^5)}{D_w(z)}, \\ U_w(z) &= \frac{1}{D_w(z)}. \end{aligned}$$

In this case $D_w(z) = (1-z)(1+pz+p^2z^2+p^3z^3+p^4z^4)+p^5z^5$. *Now that we have languages and generating functions, we can start to find some probabilities. Define*

$$X_n = \textit{(number of occurrence(s) of } w \textit{ in a randomly generated word of length } n\textit{)}.$$

The probability generating function for the probability of exactly one occurrence of w *is*

$$\sum_{n=0}^{\infty} \mathbf{P}(X_n = 1) z^n = R_w(z) U_w(z) = \frac{p^5 z^5}{D_w(z)^2}.$$

The generating function for the probability of exactly two occurrences of w*'s is*

$$\sum_{n=0}^{\infty} \mathbf{P}(X_n = 2) z^n = R_w(z) M_w(z) U_w(z) = \frac{p^6 z^6 (D_w(z) + p^4 z^4 - p^5 z^5)}{D_w(z)^3}.$$

3 Examples

In this section, we give many examples of applications of combinatorics and probability using words and trees. We make the examples as specific as possible. The motivations for the examples, we hope, will be broadly understandable to a wide class of readers. Our goal is that these examples will pique the reader's interest in studying distributions and moments of the number of occurrences of patterns in words and trees. This paper should prepare the reader to be able to move on to more advanced study of the mathematical, asymptotic theory of these data structures.

Example 3.1 *The following example comes from the popular game, Dance Dance Revolution (DDR). In this game players follow the dance pattern set up on an electronic screen by moving their feet four different directions—up, down, right, and left—on the dance pad.*

$$
\begin{aligned}
S_1 &= \texttt{R U R R R L L D D R} \\
S_2 &= \texttt{U L R R R U L D D L} \\
S_3 &= \texttt{L L L R L R D U R U} \\
S_4 &= \texttt{D D L D L D L R L L} \\
S_5 &= \texttt{L R R U U U L D U U} \\
S_6 &= \texttt{D R R R D U D R U L} \\
S_7 &= \texttt{R R D R R D R U D R} \\
S_8 &= \texttt{L U U L L L D U U D} \\
S_9 &= \texttt{L L R D U U U D R U} \\
S_{10} &= \texttt{R R R L R D U D D R} \\
S_{11} &= \texttt{L L U R R L U L L R} \\
S_{12} &= \texttt{L L U U D R D L D R} \\
S_{13} &= \texttt{U D R D U R D D R R} \\
S_{14} &= \texttt{R U R R R U D D U D} \\
S_{15} &= \texttt{R R D R U U L R D D}
\end{aligned}
$$

Figure 2: DDR Dance Pattern Strings

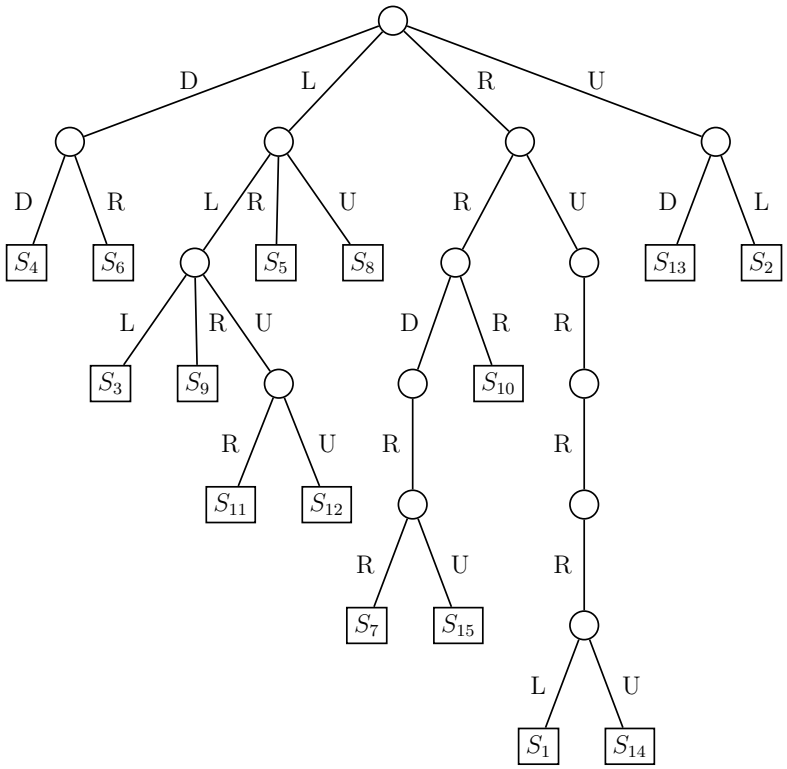

Figure 3: DDR Dance Pattern Tree

We generated 15 *independent, random dancing patters, each of length* 10*. To analyze the occurrences of specific dance patterns, we built a trie by using the dancing patterns as strings. The lengths could be extended if necessary. By separating the strings according to their type of directions,* {`U,D,R,L`} *at each level of the tree, all strings will be allocated to distinct locations. For example,* S_2 *is the only string that starts with* `UL`*, so the first level goes to the* `U` *node, and at the second level it goes to* `L`*, where the reader can see the leaf labeled* S_2*. Each string starts at the top of the tree (the root) and goes in the direction of each successive letter in the string until it reaches the point where there is no other string starting with that same pattern. At that point, we have drawn a box with the appropriate label,* S_i *inside. This way we can see how many strings begin with the same patterns. For example, we can tell that there are four strings that begin with* `LL` *and two that begin with* `RRDR`*.*

We want to analyze the occurrences of a specific dance pattern that begins with `RRDR`*. The level of the tree represents the length of the directions.* `RRDR` *ends at level* 4 *of the tree. If there are at least two strings that begin with* `RRDR`*, there will be an internal node at level* 4 *of the tree. An internal node is the point where the tree branches to its children nodes. In our tree diagram, we have two strings that begin with* `RRDR`*, and its internal node at level* 4

branches to `R`(S_7) *and* `U`(S_{15})*. As before, we use an indicator random variable*

$$I_{w=RRDR} = \begin{cases} 1 & \text{if the internal node is present,} \\ 0 & \text{otherwise.} \end{cases}$$

Instead of focusing on the `RRDR` *dance pattern, we can expand our focus to all strings that end at level* 4 *of the tree to find the number of internal nodes at a specific level. There are four specific dance patterns that reach level* 4 *of the tree, which are* {`LLUR,LLUU,RRDR,RURR`}*.*

1. `RRDR` *and* `RURR` *have internal nodes because each of them is a prefix of two or more dance patterns.*

2. `LLUR` *and* `LLUU` *do not have internal nodes because each of them is a prefix of only one dance pattern.*

3. *Some dance patterns do not reach level* 4 *because their combinations of the first four directions do not appear in our strings, e.g.,* `UUUU` *and* `URUD`*.*

4. *Some dance patterns do appear in our strings, e.g.,* `DDLD` *and* `LUUL`*, however they have already been allocated to a previous level of the tree. There is only one string that begins with* `DD` *or* `LU`*. Therefore, they end up at level* 2 *of the tree.*

Using the 15 *strings generated above, we end up with two internal nodes at level* 4 *of the tree.*

Now we explain in a broader way, which is not specific to the 15 strings that we generated above. It applies to all possible strings that are made up of {`U,D,R,L`}*. Since there are four possible directions to build the strings, there are* $4^4 = 256$ *ways to build the first four directions of the strings. I.e.,* $\mathcal{A}^4$ *has 256 strings, each of length 4. This means that there are 256 possible internal nodes at level 4 of the tree. Here, we are going to show how we calculate the expected value for the number of internal nodes at level 4 of the tree for all possible strings. We have*

$$\begin{aligned} X_{n,4} &= \text{number of internal nodes at level 4,} \\ &= \sum_{y \in \mathcal{A}^4} I_{n,y}, \text{ where } I_{n,y} = \begin{cases} 1 & \text{if the } y\text{th internal node is present,} \\ 0 & \text{otherwise,} \end{cases} \end{aligned}$$

and thus

$$\begin{aligned}
E(X_{n,4}) &= \sum_{y\in\mathcal{A}^4} E(I_{n,y}) \\
&= \sum_{y\in\mathcal{A}^4} \mathbf{P}(I_{n,y}=1) \\
&= \sum_{y\in\mathcal{A}^4} \textit{(probability two or more strings begin with dance pattern } y \textit{ of length 4)} \\
&= \sum_{y\in\mathcal{A}^4} \Big(1 - \mathbf{P}\textit{(no strings begin with dance pattern } y\textit{)} \\
&\qquad - \mathbf{P}\textit{(one string begins with dance pattern } y\textit{)}\Big) \\
&= \sum_{y\in\mathcal{A}^4} \Big(1-(1-\mathbf{P}(y))^{15} - 15\mathbf{P}(y)(1-\mathbf{P}(y))^{14}\Big).
\end{aligned}$$

For example, if the letters in $\mathcal{A}$ are equally likely, then $\mathbf{P}(y) = 1/256$, and $E(X_{n,4}) = 0.3965$, i.e., we expect less than 1 internal node at level 4, when only $n = 15$ strings are inserted into the tree.

Example 3.2 *Here, we want to analyze the occurrence of $w =$* `baaba` *in a word of length n, which is made up of letters a and b. E.g., $w =$* `baababbbaabaa`*. Working with the binary alphabet has many applications, especially in the area of computer science because you can just substitute [0,1] for $[a, b]$ and find word patterns.*

This example has several parts. We will discuss about probabilities, expected value, variance, and the mean number of internal nodes. In the first part, we want to find the probability of at least one occurrence of w and the probability of exactly one occurrence of w in a word of length n. We also want to compare both probabilities when the length of the word grows larger.

In a word of length n, `baaba` *can occur without self-overlap, such as* ***baaba****ba****baaba****a* *has 2 occurrences of w. However,* `baaba` *can also occur in an overlapping way, such as* ***baabaaba*** *has 2 overlapping occurrences of w. The overlapping occurrences make the analysis much more complicated. Again, using the combinatorial analysis, we are going to find the autocorrelation for* `baaba`*. Looking at Figure 4, the word* `baaba` *repeats at the first and fourth letters. Therefore, besides the trivial overlap, it also overlaps at the last two letters. The set for the size of overlaps is*

$$\mathcal{P}(w) = \{2, 5\}.$$

Let $\mathbf{P}(a) = p$ and $\mathbf{P}(b) = 1 - p = q$. Then the autocorrelation polynomial is

$$\begin{aligned}
S_w(z) &= 1 + \mathbf{P}(aba)z^3 \\
&= 1 + p^2qz^3.
\end{aligned}$$

baaba	
baaba	1
baaba	0
baaba	0
baaba	1
baaba	0

Figure 4: Autocorrelation of baaba

In Example 1, we have introduced $\mathcal{R}$, $\mathcal{M}$, $\mathcal{U}$, and $\mathcal{A}^*$ languages. To find the probability of at least one occurrence of $w =$ baaba in word of length n, its language representation is $\mathcal{R}\mathcal{A}^*$ ($\mathcal{R}$ represents one occurrence of w at the right end and $\mathcal{A}^*$ represents the remaining letters, which may or may not have additional occurrences of w). To find its generating function, $\frac{R_w(z)}{1-z}$, we need to find $M_w(z)$ followed by $U_w(z)$ and $R_w(z)$. We have

$$M_w(z) = \frac{p^2qz^3(1-z+pqz^2)}{D_w(z)},$$
$$U_w(z) = \frac{1}{D_w(z)},$$

$$R_w(z) = \frac{p^3q^2z^5}{D_w(z)},$$

$$\frac{R_w(z)}{(1-z)} = \frac{p^3q^2z^5}{D_w(z)(1-z)},$$

$$D_w(z) = 1 - z + p^2qz^3 - p^2qz^4 + p^3q^2z^5.$$

By substituting $p = \frac{2}{3}$ and $q = \frac{1}{3}$, and using Maple, we see that the series of $\frac{R_w(z)}{(1-z)}$ is

$$0.0329z^5 + 0.6584z^6 + 0.9877z^7 + 0.1268z^8 + 0.1549z^9 + 0.1818z^{10} + 0.2084z^{11} + \cdots .$$

If we want to find the probability of at least one occurrence of w in a word of length 10, we extract the coefficient value from z^{10}, which is 0.1818.

By putting the length n of the word on the x-axis and $\mathbf{P}$(at least one occurrence of w) on the y-axis, we generated the graph in Figure 5 through Maple. Looking at the graph,

$$\mathbf{P}(\textit{at least one occurrence of } w) = 0, \qquad \textit{for } 1 \le n \le 4.$$

When the length of the word is shorter than the length of w, it is impossible to observe the occurrence of baaba*. When the length of the word gets longer, it is more likely to observe at*

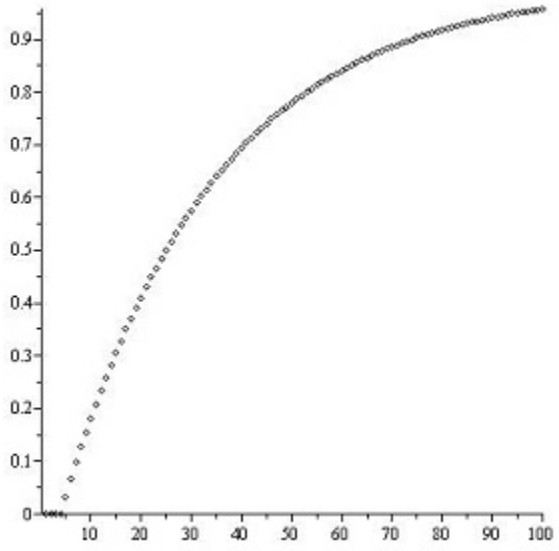

Figure 5: Probability of at least one occurrence of `baaba` in a string of length n

least one occurrence of w. When the length of the word is 100,

$$\mathbf{P}(\textit{at least one occurrence of } w) \approx 1.$$

We can compare $\mathbf{P}(\textit{at least one occurrence of } w)$ *to* $\mathbf{P}(\textit{exactly one occurrence of } w)$ *when n grows larger. First, we need to find the generating function for the probability of exactly one occurrence of w, which is $R_w(z)U_w(z)$.*

$$R_w(z)U_w(z) = \frac{p^3q^2z^5}{(1 - z + p^2qz^3 - p^2qz^4 + p^3q^2z^5)^2}.$$

By substituting $p = \frac{2}{3}$ and $q = \frac{1}{3}$, the series of $R_w(z)U_w(z)$ is computed (by Maple) to be

$$0.3292z^5 + 0.6584z^6 + 0.9877z^7 + 0.1219z^8 + 0.1451z^9 + 0.1661z^{10} + 0.1871z^{11} + \cdots .$$

The probability of exactly one occurrence of w in a word of length 10 *is* 0.1661.

We generated the graph in Figure 6 by putting the length n of word w on the x-axis and putting the probability $\mathbf{P}(\textit{exactly one occurrence of } w)$ *on the y-axis.*

Obviously, Figure 6 looks very different than Figure 5. When n grows larger (up to 35), $\mathbf{P}(\textit{exactly one occurrence of } w)$ *increases to around 0.36. However, when n grows larger than 35,* $\mathbf{P}(\textit{exactly one occurrence of } w)$ *starts decreasing. As the length of a word increases, the probability of observing w increases.*

In the second part of this example, using $w =$ `baaba` *again, we want to find (1) the expected number of occurrences of w in a word of length n and (2) the variance for the number of occurrences of w. Recall that, using $h(u, z) = \sum_{v \in \mathcal{A}^*} \mathbf{P}(v)u^{\#\text{ of occurrences of } w \text{ in } v}z^{|v|}$,*

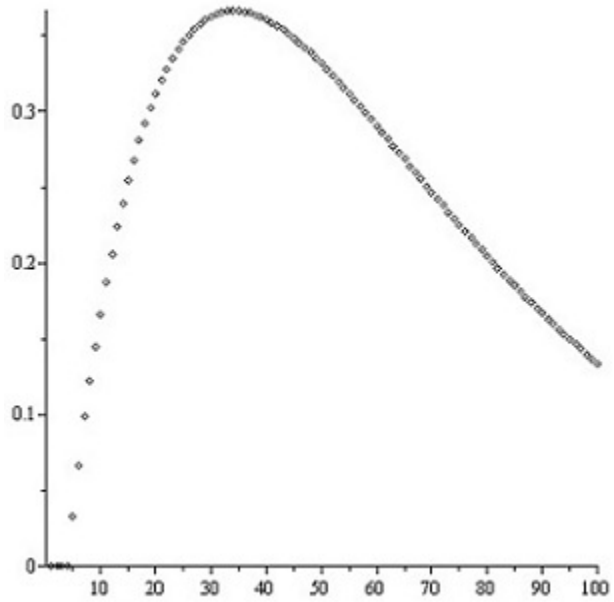

Figure 6: Probability of exactly one occurrence of `baaba` in a string of length n

equation (2) gives $h(u,z) = R_w(z)u\frac{1}{1-uM_w(z)}U_w(z)$*, which, in this case, is*

$$h(u,z) = -\frac{up^3q^2z^5}{D_w(z)(-1+z-p^2qz^3+p^2qz^4-p^3q^2z^5+up^2qz^3-up^2qz^4+up^3q^2z^5)}.$$

Using $X_{n,w}$ *as the number of occurrences of* w *in a word of length* n*, we have*

$$\sum_{n=0}^{\infty} E(X_{n,w})z^n = -\frac{p^3q^2z^5}{D_w(z)(-1+z)} + \frac{p^3q^2z^5(p^2qz^3-p^2qz^4+p^3q^2z^5)}{D_w(z)(-1+z)^2}.$$

Through Maple, the series turns out to be

$$\sum_{n=0}^{\infty} E(X_{n,w})z^n = p^3q^2z^5 + 2p^3q^2z^6 + 3p^3q^2z^7 + 4p^3q^2z^8 + 5p^3q^2z^9 + 6p^3q^2z^{10} + \cdots.$$

(In general, the expected number of occurrences of w *in a word of length* n *is* $(n-4)p^3q^2$*. If* $p = 0.5, q = 0.5$*, the expected number of occurrences of* w *in a word of length* 10 *is* 0.1875*.)*

$$\sum_{n=0}^{\infty} E[(X_{n,w})(X_n-1)]z^n = \frac{2p^3q^2z^5(p^2qz^3-p^2qz^4+p^3q^2z^5)}{D_w(z)(-1+z)^2} - \frac{2p^3q^2z^5(p^2qz^3-p^2qz^4+p^3q^2z^5)^2}{D_w(z)(-1+z)^3}.$$

Suffix 1 = `abaababbbaabaa`
Suffix 2 = `baababbbaabaa`
Suffix 3 = `aababbbaabaa`
Suffix 4 = `ababbbaabaa`
Suffix 5 = `babbbaabaa`
Suffix 6 = `abbbaabaa`
Suffix 7 = `bbbaabaa`
Suffix 8 = `bbaabaa`
Suffix 9 = `baabaa`
Suffix 10 = `aabaa`

Figure 7: Suffixes from the string `baababbbaabaa`

Through Maple, the series turns out to be

$$\sum_{n=0}^{\infty} E[(X_n)(X_n-1)]z^n = p^5q^3z^8 + 4p^5q^3z^9 + (2p^6q^4 + 6p^5q^3)z^{10} + (6p^6q^4 - 12p^5q^3 + (2(10p^3q^2 - p^5q^3))p^2q + 2p^7q^4)z^{11} + \cdots .$$

If $p = 0.5, q = 0.5$*, the value for the second falling moment for a word of length* 10 *is* 0.2539.

To compute the variance for the number of occurrences of w *in a word of a specific length* n*, we need to extract the coefficients of* n*th term from both series of expected value and second falling moment.*

For $n = 10$*, we obtain* $\mathrm{Var}(X_{10}) = 0.2539 + 0.1875 - 0.1875^2 = 0.4062$.

In the third part of this example, we want to find the mean number of internal nodes at level 5 of the tree. We build a suffix tree by using the random word **`baababbbaabaa`** *as the string. By repeatedly chopping off the first letter from the previous string, we build a set of suffixes from the original string. The strings are given in Figure 7.*

Looking at the suffix tree diagram in Figure 8, there are only 3 prefixes that reach level 5 of the suffix tree {**`aabaa`**, **`aabab`**, **`baaba`**}*. Both* **`aabaa`** *and* **`aabab`** *do not have internal nodes because each of them is a prefix of one suffix string. On the other hand,* **`baaba`** *is a prefix of two suffix strings. Its internal node at level 5 branches to* `a` *(String 9) and* `b` *(String 2). Therefore, we have one internal node at level 5 of the suffix tree.*

Now we explain in a broader way how this idea applies to all possible strings made up of letters a *and* b*. It is not specific to the case where the string is* **`baababbbaabaa`***. Since we have two possible letters to build the string, there are* $2^5 = 32$ *different ways to build the first five letters of the strings. This means that there are 32 possible internal nodes.*

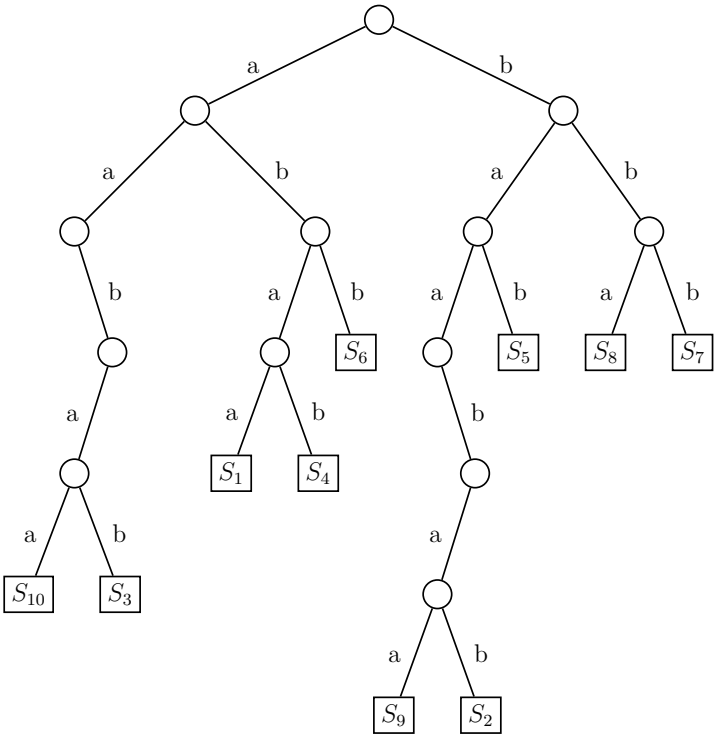

Figure 8: Suffix tree from `baababbbaabaa`

Again, we use the indicator function to denote the presence of the internal nodes. If the internal node is present, it indicates that the specific word pattern has two or more occurrences in the original string. E.g., ***baaba****ba****baaba****a. However, in the trie example, if the internal node is present, it indicates that there are at least two independent strings that begin with the the specific word pattern. E.g.,* ***baaba****aaaaba,* ***baaba****aabbbb. We define*

$$X_{n,5} = \text{number of internal nodes at level 5 in a suffix tree,}$$
$$= \sum_{y\in\mathcal{A}^5} I_{n,y}, \text{ where } I_{n,y} = \begin{cases} 1 & \text{if the internal node corresponding to } y \text{ is present,} \\ 0 & \text{otherwise,} \end{cases}$$

and therefore

$$\begin{aligned} E(X_{n,5}) &= \sum_{y\in\mathcal{A}^5} E(I_{n,y}) \\ &= \sum_{y\in\mathcal{A}^5} \mathbf{P}(I_y = 1) \\ &= \sum_{y\in\mathcal{A}^5} \text{(probability of two or more strings beginning with } y\text{).} \end{aligned}$$

Let w be one of these y's of length 5, e.g., let $w =$ `baaba`*. To calculate the probability of two or more occurrences of $w =$* `baaba` *in a word of length n, we use the values of $R(z)$, $M(z)$, and $A^*(z)$ for $w =$* `baaba` *that we have calculated previously,*

$$R_{w=baaba}(z)M_{w=baaba}(z)\Big(\frac{1}{1-z}\Big) = \frac{p^5q^3z^8(1-z+pqz^2)}{(1-z+p^2qz^3-p^2qz^4+p^3q^2z^5)^2(1-z)},$$

and we just extract the relevant coefficient of z.

To get the expected total number of internal nodes at level 5, we would still have 31 *other sums to calculate.*

Example 3.3 *Now we fix one specific word $w =$* `aaaaa`*. We can find the expected value and variance of the number of occurrences of* `aaaaa` *in a word of length n the same way we did with* `baaba`*.*

The generating function for all numbers of occurrences of w is

$$\begin{aligned} \sum_{n=0}^{\infty}\sum_{j=1}^{\infty} \mathbf{P}(X_n = j)u^j z^n &= \frac{1}{1-uM_w(z)} R_w(z)uU_w(z) \\ &= \frac{p^5z^5u}{D_w(z)^2(1-\frac{pz(D_w(z)+p^4z^4-p^5z^5)u}{D_w(z)})}, \end{aligned}$$

and

$$\sum_{n=0}^{\infty} E(X_n)z^n = \frac{p^5z^5}{(1-z)^2}.$$

The series is $\sum_{n\geq 5}(n-4)p^5z^n$. *The expected number of occurrences of* $w =$ `aaaaa` *in a word of length* n *is* $(n-4)p^5$.

$$\sum_{n=0}^{\infty} E[(X_n)(X_n-1)]z^n = \frac{-(2(1-z+pz-pz^2+p^2z^2-p^2z^3+p^3z^3-p^3z^4+p^4z^4))z^6p^6}{(-1+z)^3}.$$

To compute the variance for the occurrences of w *in a word of a specific length of* n, *we need to extract the coefficients of the* n*th terms from both series for the expected value and for the second falling moment. These series can be generated through Maple. To find the variance for the occurrences of* $w =$`aaaaa` *in a word of length* 10, *the coefficient of the 10th term from the series of expected values is* $a = 6p^5$. *The coefficient of the 10th term from the series of the second falling moment is*

$$b = 2(-p^3+p^4)p^6 + 20(-1+p)p^6 + 30p^6 + 12(-p+p^2)p^6 + 6(-p^2+p^3)p^6.$$

So we find that

$$\begin{aligned}\mathrm{Var}(X_{10}) &= b + a - a^2 \\ &= 2p^6(-p^3+p^4) + 20p^6(-1+p) + 30p^6 + 12p^6(-p+p^2) \\ &\quad + 6p^6(-p^2+p^3) + 6p^5 - 36p^{10}.\end{aligned}$$

Example 3.4 *Now we build a suffix tree by using music notes as an example for the string. We use 'Happy Birthday to You' song as an example for the string.*

$$\textit{String} = \texttt{GGAGCBGGAGDCGGHECBAFFECDC}.$$

(The G+ note is replaced with H.) By repeatedly chopping off the first letter from the previous string, 25 suffixes are generated. (See Figure 9.)

Example 3.5 *Here's an example similar to one found in [4]. This example deals with multivariate generating functions and so takes our work up to one higher dimension. This is also a very useful example because it is taken from the genetic alphabet in DNA: A, T, G, and C.*

We would like to analyze the occurrences of $w = AAG$ *and* $v = GAA$ *in a DNA sequence of length* n, *which contains G,A,T, and C as its bases (e.g., we might be performing pattern matches in a longer string, such as* $AAGGCGATGAAGTGC$*). Suppose that* $\mathbf{P}(A) = p, \mathbf{P}(G) = q$.

In this higher dimensional example we not only want to look at how the word w *overlaps with itself and the word* v *overlaps with itself, but we would also like to analyze how words* w *and* v *overlap with each other. This leads us into a discussion of the correlation polynomial.*

The overlapping set of w *with itself is the entire word* $\{AAG\}$. *Therefore, the correlation set is an empty string. The overlapping set of* w *followed by* v *is* $\{G\}$, *namely, the last letter of* w *and the first letter of* v. *Its correlation set is* $\{AA\}$, *i.e., the last two letters of* v.

Suffix 1 = GGAGCBGGAGDCGGHECBAFFECDC#
Suffix 2 = GAGCBGGAGDCGGHECBAFFECDC#
Suffix 3 = AGCBGGAGDCGGHECBAFFECDC#
Suffix 4 = GCBGGAGDCGGHECBAFFECDC#
Suffix 5 = CBGGAGDCGGHECBAFFECDC#
Suffix 6 = BGGAGDCGGHECBAFFECDC#
Suffix 7 = GGAGDCGGHECBAFFECDC#
Suffix 8 = GAGDCGGHECBAFFECDC#
Suffix 9 = AGDCGGHECBAFFECDC#
Suffix 10 = GDCGGHECBAFFECDC#
Suffix 11 = DCGGHECBAFFECDC#
Suffix 12 = CGGHECBAFFECDC#
Suffix 13 = GGHECBAFFECDC#
Suffix 14 = GHECBAFFECDC#
Suffix 15 = HECBAFFECDC#
Suffix 16 = ECBAFFECDC#
Suffix 17 = CBAFFECDC#
Suffix 18 = BAFFECDC#
Suffix 19 = AFFECDC#
Suffix 20 = FFECDC#
Suffix 21 = FECDC#
Suffix 22 = ECDC#
Suffix 23 = CDC#
Suffix 24 = DC#
Suffix 25 = C#

Figure 9: Suffixes from the notes in "Happy Birthday"

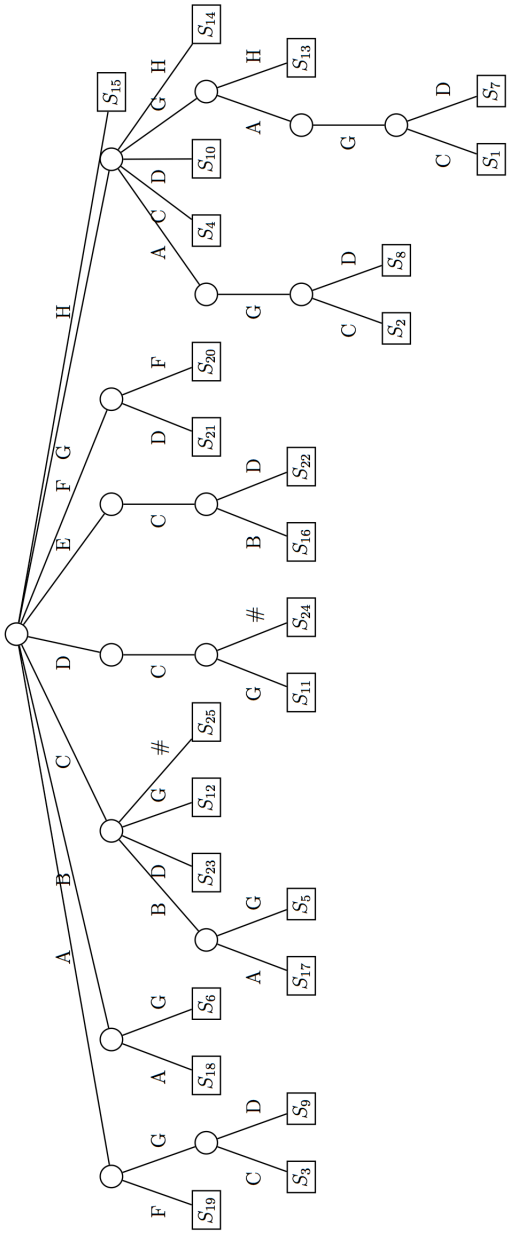

Figure 10: Suffix tree from "Happy Birthday"

Similarly, the overlapping set of v followed by w is $\{AA, A\}$. Its correlation set is $\{G, AG\}$. The overlapping set of v with itself is the entire word $\{GAA\}$. Its correlation set is the empty string $\{\varepsilon\}$.

The correlation sets are

$$\begin{aligned} A_{w,w} &= \{\varepsilon\}, \\ A_{w,v} &= \{AA\}, \\ A_{v,w} &= \{G, AG\}, \\ A_{v,v} &= \{\varepsilon\}. \end{aligned}$$

Therefore for our w and v we have the following correlation polynomials:

$$\begin{aligned} A_{w,w}(z) &= 1, \\ A_{w,v}(z) &= \mathbf{P}(AA)z^2, \\ A_{v,w}(z) &= \mathbf{P}(G)z + \mathbf{P}(AG)z^2, \\ A_{v,v}(z) &= 1. \end{aligned}$$

Here $\mathbf{P}(AA)$, $\mathbf{P}(G)$, and $\mathbf{P}(AG)$ are the probabilities of AA, G, and AG occurring, respectively. We will take $\mathbf{P}(A) = p$ and $\mathbf{P}(G) = q$. From these four values we can form a correlation matrix $\mathbb{A}$:

$$\begin{bmatrix} 1 & p^2z^2 \\ qz + pqz^2 & 1 \end{bmatrix}.$$

Define $\mathcal{M}_{i,j}$ (for $i, j \in 1, 2$) as the language of words that begin with word i and end with word j with no other occurrences in between.

So we find that

$$\mathbb{D} = (1-z)\mathbb{A} + \begin{bmatrix} z^3p^2q & z^3p^2q \\ z^3p^2q & z^3p^2q \end{bmatrix} = \begin{bmatrix} 1 - z + z^3p^2q & (1-z)p^2z^2 + z^3p^2q \\ (1-z)(qz + pqz^2) + z^3p^2q & 1 - z + z^3p^2q \end{bmatrix}.$$

Now we can compute the matrix inverse of $\mathbb{D}$, namely

$$\mathbb{D}^{-1} = \frac{1}{k}\begin{bmatrix} 1 - z + z^3p^2q & -p^2z^2(1 - z + qz) \\ -qz(1 + pz - z - pz^2 + p^2z^2) & 1 - z + z^3p^2q \end{bmatrix},$$

where

$$\begin{aligned} k = 1 &- 2z + z^3p^2q + z^2 - p^3z^4q + 2p^3z^5q - p^4z^5q - p^2z^5q - p^3z^6q + p^4z^6q \\ &- z^4p^2q^2 - z^5p^3q^2 + z^5p^2q^2 + z^6p^3q^2. \end{aligned}$$

Now that we have $\mathbb{D}$ and its inverse, all we have to do to get $\widetilde{R}_1$ and $\widetilde{R}_2$ is some simple matrix multiplication: $\begin{bmatrix} p^2qz^3 & p^2qz^3 \end{bmatrix}\mathbb{D}^{-1}$. We therefore find that

$$(\widetilde{R}_1(z), \widetilde{R}_2(z)) = \begin{bmatrix} \frac{z^3p^2q(1-z+z^3p^2q) - z^4p^2q^2(1+pz-z-pz^2+p^2z^2)}{k^*} & \frac{-z^5p^4q(1-z+qz) + z^3p^2q(1-z+z^3p^2q)}{k^*} \end{bmatrix}.$$

Suffix 1 = `AAGTCTAAGAAGTC`
Suffix 2 = `AGTCTAAGAAGTC`
Suffix 3 = `GTCTAAGAAGTC`
Suffix 4 = `TCTAAGAAGTC`
Suffix 5 = `CTAAGAAGTC`
Suffix 6 = `TAAGAAGTC`
Suffix 7 = `AAGAAGTC`
Suffix 8 = `AGAAGTC`

Figure 11: Suffixes from `AAGTCTAAGAAGTC`

Next we can find the $\mathbb{M}$ *matrix.*

$$\mathbb{M} = \begin{bmatrix} \frac{-(1-z)(1-z+z^3p^2q)}{k^*} + 1 & \frac{(1-z)p^2z^2(1-z+qz)}{k^*} \\ \frac{(1-z)qz(1+pz-z-pz^2+p^2z^2)}{k^*} & \frac{-(1-z)(1-z+z^3p^2q)}{k^*} + 1 \end{bmatrix}.$$

To analyze the occurrences of $w = AAG$ *and* $v = GAA$*, we can build a suffix tree by using the DNA sequence (e.g.,* $AAGGCGATGAAGTGC$*) as the string. By chopping off the first letter from the previous string, we build a set of suffixes from the original string. The suffixes are given in Figure 11. The suffix tree built from these suffixes is given in Figure 12. In the general case, we can look at any string using the language based on* $\{G, A, T, C\}$*. Since we have four possible characters* $\{G, A, T, C\}$ *to build the strings, there are* 64 *different ways to build the first three characters of the suffix strings. For instance,* AAG *and* GAA *are 2 out of the 64 possible choices. If the word* $w = AAG$ *occurs twice or more in the string, there would be at least 2 suffixes that start with* AAG*, which leads to an internal node at level 3. Once again, we use the indicator function to help us in finding the expected number of internal nodes in the third level of the suffix tree. The generating function whose terms are the expected values of the number of occurrences of the* j*th word out of the 64 possible choices is* $\mathcal{R}_j(z)\mathcal{M}_j(z)\frac{1}{1-z}$*, since we need two or more occurrences of the* j*th word, for it to correspond to an internal node.*

From the 64 *sums that need to be calculated, we demonstrate 1 of them, in which the* j*th word is* $w = AAG$*. Through Maple, by substituting* $p = q = 1/4$*, the series of*

$$\widetilde{R}_1(z)M_{1,1}(z)A^*(z) = \frac{1}{8192}z^7 + \frac{3}{8192}z^8 + \frac{47}{65536}z^9 + \frac{613}{524288}z^{10} + \frac{225}{131072}z^{11} + \frac{617}{262144}z^{12} + \cdots.$$

From this series, we can extract a certain coefficient, like $n = 10$*. In this situation, we find that the coefficient for the tenth term is* $\frac{613}{524288}$*. Next, using equation (3), we are going to find* $\mathrm{Var}(X_n)$*. From the earlier calculations, we have computed* $E(X_n)$*. In order to find the*

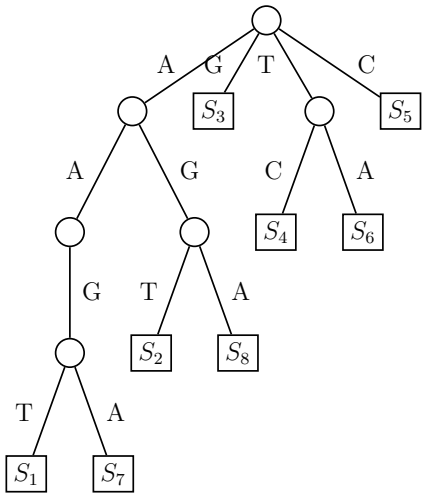

Figure 12: Suffix Tree for `AAGTCTAAGAAGTC`

variance, we still have to find the value for $\sum_{y \neq z} E(I_{n,y} I_{n,z})$. *To calculate all the values for* $E(I_y I_z)$ *is tedious, so, we only demonstrate* 1 *out of the* $1952 = (\binom{64}{2} - 64)$ *combinations of* y *and* z. *Let*

$$I_1 = \textit{the node corresponding to } AAG \textit{ on level 3 of the tree},$$
$$I_2 = \textit{the node corresponding to } GAA \textit{ on level 3 of the tree}.$$

If both I_1 *and* I_2 *are internal nodes,* $I_1 I_2 = 1$. *Otherwise,* $I_1 I_2 = 0$. *Thus*

$$E(I_1 I_2) = \mathbf{P}(\textit{at least two occurrences of } w = AAG \textit{ and at least two occurrences of } v = GAA).$$

There are 6 possible ways that the first two w*'s and the first two* v*'s can appear, namely,* $wwvv, wvvw, wvwv, vvww, vwvw, vwwv$.

Since we already have notation to describe these different combinations of words, we can then express all the possibilities of strings that contain two or more occurrences of both AAG *and* GAA. *The ways in which the first occurrence of* w *appears before the first occurrence of* v *are:*

In Figure 13, the chart illustrates the possible orders in which w*'s and* v*'s can occur in a given string. The lines in the chart correspond to the above languages, with the first two lines following the pattern of the first language, the third and fourth lines following the pattern of the second language, and the last two lines following the pattern of the third language. There are three other possible combinations, which are the three cases that are the reverse of the three above (i.e., switch all the* v*'s and* w*'s, and consequently switch all the* 1*'s and* 2*'s in the subscripts of the languages).*

$\widetilde{\mathcal{R}}_1\mathcal{M}_{1,1}\mathcal{M}_{1,1}^*\mathcal{M}_{1,2}(\mathcal{M}_{2,1}\mathcal{M}_{1,1}^*\mathcal{M}_{1,2}+\mathcal{M}_{2,2})\mathcal{A}^*$

(no w's, v's)	**w**	(no w's, v's)	**w**	(w's allowed)	**v**	(w's allowed)	**v**	(w's, v's allowed)

$\widetilde{\mathcal{R}}_1\mathcal{M}_{1,2}\mathcal{M}_{2,2}\mathcal{M}_{2,2}^*\mathcal{M}_{2,1}\mathcal{A}^*$

(no w's, v's)	**w**	(no w's, v's)	**v**	(no w's, v's)	**v**	(v's allowed)	**w**	(w's, v's allowed)

$\widetilde{\mathcal{R}}_1\mathcal{M}_{1,2}\mathcal{M}_{2,1}\mathcal{M}_{1,1}^*\mathcal{M}_{1,2}\mathcal{A}^*$

(no w's, v's)	**w**	(no w's, v's)	**v**	(no w's, v's)	**w**	(w's allowed)	**v**	(w's, v's allowed)

Figure 13: Possible orders of the occurrences of w's and v's, with w first

Therefore we are now able to find $E(I_iI_j)$ as the sum of all six possible combinations of matrices added together. If we substitute $p = q = .25$ into our final result we find that the series begins like this:

$$E(I_iI_j) = \frac{1}{16384}z^7 + \frac{3}{16384}z^8 + \frac{45}{131072}z^9 + \frac{285}{524288}z^{10} + \frac{825}{1048576}z^{11} + \cdots .$$

4 Conclusion

Our paper focuses on combinatorial analysis and trees to find repeating occurrences of patterns. We give explanations of several examples. Combinatorial languages are used together with generating functions to find the probabilities, expected values, and variances of repeating patterns. As patterns take various forms, trees help us to locate all kinds of patterns at a particular level of the tree. Then, repeated patterns are identified through the presence of internal nodes. Although patterns can be identified by visual inspection of a string, our method of using trees requires less work in identifying occurrences of repeated strings.

References

[1] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge, 2009.

[2] P. Jacquet and W. Szpankowski. Analytic approach to pattern matching. In M. Lothaire, editor, *Applied Combinatorics on Words*, chapter 7. Cambridge, 2005. See [3].

[3] M. Lothaire, editor. *Applied Combinatorics on Words*. Cambridge, 2005.

[4] M. Régnier and A. Denise. Rare events and conditional events on random strings. *Discrete Mathematics and Theoretical Computer Science*, 2004.