Tree Enumeration Polynomials on Separable Permutations

Yibo Gao† and Siyu Liu‡

†Beijing International Center for Mathematical Research, Peking University, Beijing 100084, China
Email: gaoyibo@bicmr.pku.edu.cn

‡Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139
Email: eliu24@mit.edu

Received: May 12, 2023, Accepted: August 15, 2023, Published: August 25, 2023

ABSTRACT: Pak and Postnikov introduced a tree enumeration polynomial $f_G$ on graphs, as a multivariate generalization of Cayley's formula, and demonstrated an amazing reciprocity property. In this paper, we prove that this tree enumeration polynomial can be factorized into linear factors for the inversion graph of separable permutations. We derive an explicit formula for this factorization and provide three proofs: one using the reciprocity theorem, one algebraic, and another one bijective. We also prove its converse: the tree enumeration polynomials for all other graphs cannot be factored into linear factors.

Keywords: Matrix-Tree Theorem; Reciprocity Theorem; Separable Permutation

2020 Mathematics Subject Classification: 05C05; 05C30; 05C31

1. Introduction

Cayley's formula [3] states that the number of spanning trees of the (labeled) complete graph $K_n$ equals $n^{n-2}$. One simple proof of this result uses the Matrix-Tree Theorem, which expresses the number of spanning trees of a graph as a determinant. However, a deterministic formula does not provide any algorithms to enumerate spanning trees. The first enumerative proof of Cayley's formula is given by Pr"ufer [13] via a simple bijective encoding of all labeled trees. In particular, he encodes trees on $n$ labeled vertices using sequences $(a_1, \ldots, a_n)$ of positive integers between 1 and $n$. The conversion between spanning trees and sequences can be done by well-defined, bijective, iterative algorithms. Numerous generalizations of Cayley's formula have then been found, including R"enyi's coding for spanning trees of the complete bipartite graphs [14].

In particular, Pak and Postnikov [12] presented a novel approach to this problem by introducing a polynomial $f_G$ that enumerates spanning forests of $G$ according to degrees of all vertices. Moreover, they showed that the polynomial $f_G$ satisfies a remarkable reciprocity theorem (Theorem 2.1), which allowed them to obtain a formula for the number of spanning trees of the complete multipartite graphs very easily. Combinatorial proofs of this reciprocity theorem are given by Huang-Postnikov [10] and Hoey-Xiao [9]. Additionally, [11] proposed a linear-time algorithm by applying the Matrix Tree Theorem to determine the number of spanning trees in cographs—an equivalent combinatorial object to the inversion graphs of separable permutations.

In this paper, we study the problem of enumeration of spanning forests of $G_w$, the inversion graph of a permutation $w \in S_n$. Especially, we focus on the case when $w$ is separable. The family of graphs $G_w$ when $w$ is separable includes all complete multipartite graphs with arbitrary part sizes. Our main theorem (Theorem 1.1) provides an explicit formula for $f_w := f_{G_w}$ with $w$ separable, which factors into linear factors, as follows.

Theorem 1.1. For a separable permutation $w \in S_n$,

$$f_w(x_0; x_1, \ldots, x_n) = \prod_{i=1}^{n-1} (x_0 + f^1_i + f^2_i),$$

where $f^1_i = \sum_{j \leq i, w(j) > w(i+1)} x_j$ and $f^2_i = \sum_{j \geq i+1, w(j) < w(i)} x_j$.

We provide three proofs: one uses the reciprocity theorem of Pak and Postnikov [12]; one uses a determinantal argument; and the other one is bijective, which can be thought of as encoding spanning forests of $G_w$ via a
sequence of positive integers, in a much more general context than the Prüfer code. It is surprising that a converse of Theorem 1.1 holds:

**Theorem 1.2.** For a permutation \( w \in S_n \), if \( w \) is not separable, then \( f_w \) cannot be factored into linear factors.

Separable permutations, which are defined as permutations that avoid the patterns 2413 and 3142, have received a lot of attention in algebraic combinatorics. Arising from the study of pop-stack sorting [1], they have nice recursive structures similar to binary trees, and have applications to bootstrap percolation [15] and pattern matching [2]. Wei [16] showed that for a separable permutation \( w \), its principal order ideal \([id, w]\) and principal order filter \([w, w_0]\) in the weak Bruhat order are rank symmetric, and the product of their rank generating function equals \([n]_q!\). This result is generalized to other Weyl groups by Gaetz and the first author [5], who introduced the notion of a separable element. Further studies [6] made connections between separable elements and faces of generalized permutahedron, splittings of Weyl groups, etc. And it is shown that separable permutations have nice actions on canonical bases by bijections up to lower-order terms [8]. Our results (Theorem 1.1 and Theorem 1.2) provide yet another characterization of separable permutations, by considering whether the tree enumeration polynomial \( f_w \) factors linearly.

**Remark 1.1.** As our helpful referee pointed out, much of this paper can be written in the language of cographs as well. For a graph \( G \) on labels \( \{y_1, \ldots, y_m\} \) and a graph \( H \) on labels \( \{z_1, \ldots, z_n\} \), let \( G \oplus H \) and \( G \ominus H \) be the graphs on \( \{y_1, \ldots, y_m, z_1, \ldots, z_n\} \), where \( \oplus \) is the disjoint union, and \( \ominus \) is the join, connecting every vertex in \( G \) to every vertex in \( H \). The permutation graph of a separable permutation is called a cograph [2]. Cographs can be generated from the single-vertex graph \( K_1 \) by joins (\( \ominus \)) and disjoint unions (\( \oplus \)). They are equivalently characterized as graphs that don’t contain \( P_4 = \bullet \bullet \longrightarrow \bullet \bullet \) as a subgraph. Specifically, \( P_4 \) is the permutation graph of 2413 and 1324. The equivalence between labeled cographs and separable permutations allows us to rewrite Theorem 1.2:

**Theorem.** The polynomial \( f_G \) can be written as a product of linear factors if and only if \( G \) is a cograph.

This paper will be organized in the following fashion. In Section 2, we provide the necessary background on the polynomial \( f_G \), which is the main object of study, and on separable permutations. In Section 3, we provide a recursive proof of Theorem 1.1, using the reciprocity theorem in [12]; and in Section 4, we provide another short proof using ideas in Matrix-Tree Theorem. A bijective proof is given in two sections: in Section 5, we prove a recurrence formula bijectively, which is used in Section 6 to ensemble a bijective proof. In Section 7, we discuss non-separable permutations and prove Theorem 1.2. Finally, we talk about further directions in Section 8.

## 2. Background

We mainly follow notations in [12] on the enumeration of spanning trees of graphs.

**Definition 2.1.** For a graph \( G \) on \( n \) vertices \( V = [n] := \{1, \ldots, n\} \) and a spanning tree \( T \) of \( G \), we define a monomial \( m(T) = \prod_{v \in V} x_v^{\rho_T(v)-1} \), where \( \rho_T(v) \) denotes the degree of the vertex \( v \) in the tree \( T \). Define a polynomial

\[
\tau_G := \sum_T m(T)
\]

where the sum is over all spanning trees \( T \) of \( G \).

As suggested in [12], it is often more convenient to work with the extended graph \( \tilde{G} \) as follows.

**Definition 2.2.** The extended graph \( \tilde{G} \) on vertices \( \tilde{V} := \{0\} \cup V \) is obtained from \( G \) by adding a vertex, 0, connected with all vertices of \( G \). We construct another polynomial \( f_{\tilde{G}} \) of variables \( x_v \) for \( v \in \tilde{V} \) such that

\[
f_{\tilde{G}} := \tau_{\tilde{G}}.
\]

Write \( f_G = f_G(x_0; x_1, x_2, \ldots, x_n) \). One should think of \( \tau_G \) as a polynomial enumerating spanning trees of \( G \), while \( f_G \) as a polynomial enumerating spanning forests of \( G \). We can recover \( \tau_G \) from \( f_G \) by

\[
\tau_G(x_1, x_2, \ldots, x_n) \cdot (x_1 + x_2 + \cdots + x_n) = f_G(0; x_1, x_2, \ldots, x_n).
\]

Pak and Postnikov developed the following reciprocity theorem, which is a powerful tool for the computation of spanning trees of graphs. For a graph \( G = (V, E) \), denote its compliment as \( \overline{G} = (V, \overline{E}) \) which contains all edges not in \( G \).
Theorem 2.1 ([12]). Let $G$ be a graph on vertices $V = \{1, 2, \ldots, n\}$, then
\[
f_G(x_0; x_1, \ldots, x_n) = (-1)^{n-1}f_{\overline{x}}(-x_0 - x_1 - \cdots - x_n; x_1, \ldots, x_n).
\]

Bijective proofs of Theorem 2.1 are given in [9,10].

In this paper, we primarily focus on graphs that are the inversion graphs of permutations and classify whether the polynomial $f_{\overline{x}}$ can be factored linearly by properties on $w \in S_n$.

**Definition 2.3.** For a permutation $w \in S_n$, define the inversion graph $G_w$ to be a graph on $[n]$ such that there is an edge between $v_i$ and $v_j$ if $i < j$ and $w(i) > w(j)$, i.e. $(i,j)$ is an inversion of $w$. For simplicity, we write $f_w$ for $f_{G_w}$.

Note that $G_{w_0} = G_{w_0w}$ where $w_0 = n \cdot 1 \in S_n$ is the longest permutation. Because of this, we also write $G_w = G_{\overline{x}}$.

**Example 2.1.** For $w = 3412$, $G_w$ and $G_{\overline{x}}$ are the following.

![Diagram](image)

Figure 1: $G_w$ and its extended graph, $G_{\overline{x}}$ for $w = 3412$.

**Example 2.2.** We provide two examples of spanning trees of $G_w$ in Fig. 2.

![Diagram](image)

Figure 2: Two examples of spanning trees, $T_1, T_2$, of $G_w$.

A particularly nice family of permutations relevant to our study is the set of separable permutations.

**Definition 2.4.** We say that a permutation $w \in S_n$ avoids the pattern $\pi \in S_k$ if there does not exist indices $i_1 < \cdots < i_k$ such that $w(i_1), \ldots, w(i_k)$ have the same relative order as $\pi(1), \ldots, \pi(k)$. A permutation $w \in S_n$ is separable if it avoids 2413 and 3142.

Separable permutations have nice recursive structures as follows.

**Proposition 2.1.** For $n > 1$ and a separable permutation $w \in S_n$, we can write $w = w_A w_B$ (concatenation of words), where both $w_A$ and $w_B$ are separable permutations satisfying one of the following two properties, with some $1 < m < n$:

- $w_A$ is a permutation of $1, \ldots, m$ and $w_B$ is a permutation of $m + 1, \ldots, n$;
- $w_A$ is a permutation of $n - m + 1, \ldots, n$ and $w_B$ is a permutation of $1, \ldots, n - m$.

Proposition 2.1 is well-known and easy to prove, so we omit the proof here. See for example [16]. In light of Proposition 2.1, we say that a separable permutation $w$ has an inversion cut at index $i$ if $w(a) > w(b)$ for all $a \leq i$ and $b \geq i + 1$, and $w$ has a non-inversion cut if $w(a) < w(b)$ for all $a \leq i$ and $b \geq i + 1$. For any separable permutation $w$, a sequence of cuts can be made until $w$ is cut into singletons.

One main goal of this paper is to provide an explicit formula for $f_w$ when $w$ is separable. See the following for an example of the main result, Theorem 1.1.

**Example 2.3.** For $w = 3241$, $f_w = (x_0 + x_1 + x_2 + x_4)(x_0 + x_4)(x_0 + x_1 + x_2 + x_3 + x_4)$.

1. At the first interval, $w(1) > w(2)$, which creates an inversion. $f_1 = x_0, f_2 = x_4$, because $w(1) > w(4)$. Therefore, the factor is $x_0 + x_1 + x_2 + x_4$. 

ECA 4:1 (2024) Article #S2R6 3
2. At the second interval, $w(2) < w(3)$, which does not create an inversion. $f_2^1 = x_0, f_2^2 = x_4$, because $w(2) > w(4)$. Therefore, the factor is $x_0 + x_4$.

3. At the third interval, $w(3) > w(4)$, which creates an inversion. $f_3^1 = x_1 + x_2, f_3^2 = x_0$, because $w(1) > w(4)$ and $w(2) > w(4)$. Therefore, the factor is $x_0 + x_1 + x_2 + x_3 + x_4$.

$f_w$ is obtained by multiplying all three factors.

3. Recursive Formula for Separable Permutations

In this section, we present a recursive formula for $f_w$ for separable permutations and provide an algebraic proof of Theorem 1.1.

**Theorem 3.1.** Let $w \in S_n$ be separable. Suppose $w = w_A w_B$ in the sense of Proposition 2.1, the following result is true.

- If $w_A$ is a permutation of \{1, 2, \ldots, m\} and $w_B$ a permutation of \{m + 1, m + 2, \ldots, n\}, then $f_w = x_0 f_{w_A}(x_0; x_1, x_2, \ldots, x_m) f_{w_B}(x_0; x_{m+1}, x_{m+2}, \ldots, x_n)$.
- If $w_A$ is a permutation of \{1, 2, \ldots, m\} and $w_B$ a permutation of \{m + 1, m + 2, \ldots, n\}, then $f_w = (x_0 + x_1 + \cdots + x_n) f_{w_A}(x_0 + x_{m+1} + \cdots + x_n; x_1, x_2, \ldots, x_{m}) f_{w_B}(x_0 + x_{m+1} + \cdots + x_n; x_{m+1}, x_{m+2}, \ldots, x_n)$.

In the graph induced by the permutation in the second case, all vertices in $w_A$ are connected to all vertices in $w_B$, since every $w_A(i), w_B(j)$ is an inversion.

**Proof.** The non-inversion cut case follows from Theorem 2.1, as \{1, 2, \ldots, m\} and \{m + 1, m + 2, \ldots, n\} are two sets of vertices where there are no edges between the two sets.

In the inversion cut case, $G_w = G_{w_A} \cup G_{w_B}$. By the Reciprocity Theorem in [12],

$$f_{\Pi w} = (-1)^{|A|-1} f_{w_A}(-x_0 - x_{m+1} - \cdots - x_n; x_1, x_2, \ldots, x_m),$$

$$f_{\Pi w} = (-1)^{|B|-1} f_{w_A}(-x_0 - x_1 - x_2 - \cdots - x_{m}; x_{m+1}, x_{m+2}, \ldots, x_n).$$

Apply case 1 and the reciprocity theorem one more time, and substitute in $-x_0 - x_1 - \cdots - x_n$ for $x_0$ to conclude the proof.

**Remark 3.1.** Theorem 3.1 can be restated in the language of cographs as follows.

Let $G, H$ be cographs on $m$ and $n$ vertices, respectively. Set $y := y_1 + \cdots + y_m$ and $z := z_1 + \cdots + z_n$. Then the polynomials can be written as

$$f_{G \oplus H}(x_0; y_1, \ldots, y_m, z_1, \ldots, z_n) = x_0 \cdot f_G(x_0; y_1, \ldots, y_m) \cdot f_H(x_0; z_1, \ldots, z_n),$$

$$f_{G \oplus H}(x_0; y_1, \ldots, y_m, z_1, \ldots, z_n) = (x_0 + y + z) \cdot f_G(x_0 + z; y_1, \ldots, y_m) \cdot f_H(x_0 + y; z_1, \ldots, z_n).$$

Theorem 1.1 can be proved with this recursive formula by strong induction on the number of vertices.

**Proof of Theorem 1.1.** The case when $w$ is of size 1 is trivial. Suppose Theorem 1.1 is true on permutations of size 1, 2, \ldots, $k$. Let $w$ be a permutation of size $k + 1$. Since $w$ is separable, let $w = w_A w_B$, where $w_A$ and $w_B$ have sizes $m$ and $n - m$.

If $w_A w_B$ is a non-inversion cut, Theorem 3.1 asserts that

$$f_w = x_0 \cdot f_{w_A}(x_0; x_1, x_2, \ldots, x_m) \cdot f_{w_B}(x_0; x_{m+1}, x_{m+2}, \ldots, x_n).$$

In this instance, all factors $f_1^1$ and $f_2^2$ in Theorem 1.1 remain unchanged, and $f_m^1 = f_m^2 = 0$, as no $w_A(j)$ with $j \in [1, m]$ is greater than $w_B(1)$ and no $w_B(j)$ with $j \in [1, n - m]$ is less than $w_A(m)$. So $x_0$ is the linear factor at the $m$-th interval of $w$.

If $w_A w_B$ is an inversion cut, Theorem 3.1 asserts that $f_w = (x_0 + x_1 + \cdots + x_n) \cdot f_{w_A}(x_0 + x_{m+1} + \cdots + x_n; x_1, x_2, \ldots, x_m) \cdot f_{w_B}(x_0 + x_1 + \cdots + x_n; x_{m+1}, x_{m+2}, \ldots, x_n)$.

For each interval $i = 1, 2, \ldots, m - 1$ in $w_A$, $w_A(i)$ is greater than all of $w_B$, which adds $x_{m+1} + x_{m+2} + \cdots + x_n$ to every linear factor in $w_A$. Similarly, the inversion cut adds $x_1 + x_2 + \cdots + x_m$ to every linear factor in $w_B$. Additionally, $f_m^1 = x_{m+1} + x_{m+2} + \cdots + x_n$ and $f_m^2 = x_1 + x_2 + \cdots + x_m$, so the linear factor at the $m$-th interval is $x_0 + x_1 + \cdots + x_n$.

The induction step concludes the proof of Theorem 1.1. 

ECA 4:1 (2024) Article #S2R6
4. Recursive Formula Proof 2: the Matrix-Tree Theorem

In this section, we provide a second proof of Theorem 3.1 with the Weighted Matrix-Tree Theorem.

**Definition 4.1.** The Laplacian matrix $L(G)$ of $G$ on $n$ vertices is a $n \times n$ matrix whose $(i, j)$-entry $L_{i,j}$ is given by

$$L_{i,j} = \begin{cases} -x_ix_j & i \neq j \text{ and there is an edge connecting } i, j, \\ 0 & i \neq j \text{ and there is no edge connecting } i, j, \\ -\sum_{h \neq i} L_{i,h} & i = j. \end{cases}$$

The reduced Laplacian matrix $\widetilde{L}(G)$ is obtained from deleting the $i$-th row and $i$-th column from $L(G)$.

$L(G)$ can also be seen as the matrix representation of a tree with edge weights $-L_{i,j}$ for $i \neq j$. We introduce the Weighted Matrix-Tree Theorem before proceeding to the second proof.

**Theorem 4.1.** [Weighted Tree Theorem [4]] For a graph $G$, the number of spanning trees of $\tilde{G}$ is equal to the determinant of the reduced Laplacian of $\tilde{G}$, where $-L_{i,j}$ is the edge weight between vertices $i \neq j$ and $L_{i,i} = -\sum_{h \neq i} L_{i,h}$.

In particular, as every row and column of $L$ sum to 0, linear dependence guarantees that the determinant of the reduced Laplacian is identical regardless of the choice of $i$. Without loss of generality, we choose to always eliminate the row and column that records the connections of $x_0$, and write the corresponding reduced Laplacian as $L(\tilde{G})$.

**Proposition 4.1.** For a graph $G$,

$$f_G = \frac{\det \left( \widetilde{L}(\tilde{G}) \right)}{\prod_{v \in \tilde{V}} x_v} = \frac{\det \left( L(\tilde{G}) \right)}{\prod_{v \in \tilde{V}} x_v}.$$

**Proof.** For a spanning tree $T$ in the graph $G$ with the degrees of vertices $v_i$ being $\deg(v_i)$, the monomial in $\det(L(\tilde{G}))$ associated to $T$ is equal to the product of the weights on all edges of $T$ by Theorem 4.1. As the weight on each edge equals the product of the two vertices that it connects, this monomial is $\prod_{v \in \tilde{V}} x_{\deg(v)}$, which is equal to $m(T) \cdot \prod_{v \in \tilde{V}} x_v$.

We return to the proof of Theorem 3.1.

**Example 4.1.** For $w = 2143$, the Laplacian matrix for $\tilde{G}_w$ is

$$L(\tilde{G}_w) = \begin{pmatrix} x_0(x_1 + x_2 + x_3 + x_4) & -x_0x_1 & -x_0x_2 & -x_0x_3 & -x_0x_4 \\ -x_0x_1 & x_1(x_0 + x_2) & -x_1x_2 & 0 & 0 \\ -x_0x_2 & -x_1x_2 & x_2(x_0 + x_1) & 0 & 0 \\ -x_0x_3 & 0 & 0 & x_0x_3 & 0 \\ -x_0x_4 & 0 & 0 & 0 & x_0x_4 \end{pmatrix}$$

And the reduced Laplacian is

$$\widetilde{L}(\tilde{G}_w) = \begin{pmatrix} x_1(x_0 + x_2) & -x_1x_2 & 0 & 0 \\ -x_1x_2 & x_2(x_0 + x_1) & 0 & 0 \\ 0 & 0 & x_0x_3 & 0 \\ 0 & 0 & 0 & x_0x_4 \end{pmatrix}$$

We return to the proof for Theorem 3.1.

**Proof of Theorem 3.1.** Case 1: $w_A, w_B$ are permutations of permutation of $\{1, \ldots, m\}$ and of $\{m + 1, \ldots, n\}$, respectively. As such, their corresponding reduced Laplacian matrix can be written as $\tilde{A}(x_0, x_1, \ldots, x_m)$ and $\tilde{B}(x_0, x_1, \ldots, x_{n-m})$.

Let $\tilde{L}$ be the reduced Laplacian matrix for permutations $G_w$. Define $\tilde{A} = \tilde{A}$ and $\tilde{B} = \tilde{B}(x_0, x_{m+1}, \ldots, x_n)$.

Then $\tilde{L}$ is a block diagonal matrix with blocks $\tilde{A}$ and $\tilde{B}$.

By Proposition 4.1,

$$\det(\tilde{L}) = x_0x_1 \ldots x_nf_w$$
and 
\[ \det(\tilde{A}) \det(\tilde{B}) = x_0^2 x_1 \ldots x_n f_{w_A} f_{w_B} (x_0; x_{m+1}, x_{m+2}, \ldots, x_n). \]

Therefore, \( f_w = x_0 f_{w_A} f_{w_B} (x_0; x_{m+1}, x_{m+2}, \ldots, x_n) \), as desired.

Case 2: \( w_A, w_B \) are permutations of \( \{m+1, \ldots, n\} \) and permutations of \( \{1, \ldots, m\} \), respectively. Define \( \tilde{A}, \tilde{B}, \tilde{L} \) similarly for permutations \( w_A \) of length \( n-m \), \( w_B \) of length \( m \), and \( w = w_B w_A \). Define matrices \( \tilde{A}, \tilde{B} : \)

\[
\tilde{A}_{i,j}(x_0 + x_{m+1} + \cdots + x_n; x_1, \ldots, x_m)
\]

and

\[
\tilde{B}_{1,j}(x_0 + x_{m+1} + \cdots + x_m; x_{m+1}, \ldots, x_n).
\]

Therefore

\[
\det(\tilde{A}) = (x_0 + x_{m+1} + \cdots + x_n) f_{w_A} (x_0 + x_{m+1} + \cdots + x_n; x_1, \ldots, x_m)
\]

\[
\det(\tilde{B}) = (x_0 + x_{m+1} + \cdots + x_m) f_{w_B} (x_0 + x_{m+1} + \cdots + x_m; x_{m+1}, \ldots, x_n)
\]

We use these to compute the determinant of \( \tilde{L} \). Define

\[
\tilde{L}' = \begin{pmatrix}
\tilde{A} & -x_{m+1} & -x_{m+2} & \cdots & -x_n \\
-x_1 & -x_2 & \cdots & -x_n \\
0 & -x_2 & \cdots & -x_n \\
0 & 0 & \cdots & -x_n \\
\end{pmatrix}
\begin{pmatrix}
M_x \\
M_y \\
M_z \\
M_w
\end{pmatrix}
\]

Notably, \( \det(\tilde{L}) = x_1 x_2 \ldots x_n \det(\tilde{L}') \). The factor \( x_1 x_2 \ldots x_n \) compensates for the division in Eq. (1) and Eq. (2). To compute for \( \det(\tilde{L}') \), we define \( \tilde{L}'' \) with elementary row operations matrices \( M_1, M_2, M_3 \) and \( M_4 \) that divides the first row by \( x_0 \). Specifically,

\[
\tilde{L}'' = \begin{pmatrix}
I_m & 0 \\
M_1 & M_2 \\
M_3 & M_4
\end{pmatrix}
\tilde{L}' M_3 \cdot M_4,
\]

where \( M_1 = \begin{pmatrix}
-1 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 0
\end{pmatrix}, \quad M_2 = \begin{pmatrix}
1 & 0 & \ldots & 0 \\
-1 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
-1 & 0 & \ldots & 1
\end{pmatrix}, \quad M_3 = \begin{pmatrix}
1 & 0 & \ldots & 0 \\
1 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 0 & \ldots & 1
\end{pmatrix}, \quad M_4 = \begin{pmatrix}
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1
\end{pmatrix}
\]

Since \( M_1, M_2, \) and \( M_3 \) are elementary row operations and have determinants equal to 1 and \( \det(M_4) = \frac{1}{x_0} \),

\[
\det(\tilde{L}'') = \frac{\det(\tilde{L}')}{x_0}
\]

Substituting the \( \tilde{A} \) and \( \tilde{B} \) into the expression for \( \tilde{L}' \) in Eq. (3), \( \tilde{L}'' \) simplifies to \( \begin{pmatrix}
N_1 & N_2 \\
N_3 & N_4
\end{pmatrix} \), where \( N_1, N_2, N_3, N_4 \) are matrices of size \( m \times m, m \times n, n \times m, \) and \( n \times n \) respectively, defined by

\[
N_1 = \begin{pmatrix}
1 & \tilde{A}_{1,2} & \tilde{A}_{1,3} & \ldots & \tilde{A}_{1,m} \\
0 & \tilde{A}_{2,2} & \tilde{A}_{2,3} & \ldots & \tilde{A}_{2,m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \tilde{A}_{m,2} & \tilde{A}_{m,3} & \ldots & \tilde{A}_{m,m}
\end{pmatrix},
\]

\[
N_2 = \begin{pmatrix}
-x_{m+1} & -x_{m+2} & -x_{m+3} & \cdots & -x_n \\
0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 0
\end{pmatrix},
\]

\[
N_3 = \begin{pmatrix}
0 & -x_2 & -x_3 & \ldots & -x_m \\
0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 0
\end{pmatrix}, \quad \text{and}
\]
As such, there exists a bijection between the set of tree-sequence tuples and trees in Theorem 5.1. This proof takes inspiration from a special case of [12]. We start in this section, we provide an alternative proof of the second case of Theorem 3.1 with a bijection between the set of tuples and the set \( N_3 \) with the first row and column removed. As such, the block matrix \( \tilde{L} \) has a determinant equal to

\[
\det (N_1 - N_2 N_4^{-1} N_3) = \det (N_1) = \det (N_4).
\]

Hence it suffices to find \( \det (N_1) \) and \( \det (N_4) \).

Upon division by \( x_0 \) in the first column,

\[
\det (N_1) = \frac{\det \tilde{A}}{x_0} = f_{w_A}(x_0 + x_{m+1} + \ldots x_n; x_1, \ldots, x_m).
\]

On the other hand, let \( K = x_0 + x_1 + \cdots + x_m \),

\[
N_4 = \frac{1}{K} \begin{pmatrix}
\tilde{B}_{m+1,m+1} + x_{m+1} & \tilde{B}_{m+1,m+2} + x_{m+2} & \tilde{B}_{m+1,m+3} + x_{m+3} & \ldots & \tilde{B}_{m+1,n} + x_n \\
\tilde{B}_{m+2,m+1} + x_{m+1} & \tilde{B}_{m+2,m+2} + x_{m+2} & \tilde{B}_{m+2,m+3} + x_{m+3} & \ldots & \tilde{B}_{m+2,n} + x_n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\tilde{B}_{n,m+1} + x_{m+1} & \tilde{B}_{n,m+2} + x_{m+2} & \tilde{B}_{n,m+3} + x_{m+3} & \ldots & \tilde{B}_{n,n} + x_n
\end{pmatrix} \cdot \mathbb{B},
\]

where the eigenvalues of the matrix are \((K + x_{m+1} + \cdots + x_n)\) with multiplicity 1, and \(K\) with multiplicity \((n-m-1)\). Therefore,

\[
\det (N_4) = K^{m-n} \cdot (K + x_{m+1} + \cdots + x_n) \cdot K^{n-m-1},
\]

\[
= x_0 + x_1 + \cdots + x_n \cdot \det \mathbb{B} = (x_0 + x_1 + \cdots + x_n) f_{w_B}(x_0 + x_1 + \cdots + x_m; x_{m+1}, \ldots, x_n).
\]

As such,

\[
\det (\tilde{L}) = x_0 x_1 \ldots x_n \det (\tilde{L}')
\]

\[
= x_0 x_1 x_2 \ldots x_n \det (\tilde{L}') = \det N_1 \det N_4
\]

\[
= (x_0 + x_1 + \cdots + x_n) f_{w_A}(x_0 + x_{m+1} + \ldots x_n; x_1, \ldots, x_m) f_{w_B}(x_0 + x_1 + \cdots + x_m; x_{m+1}, \ldots, x_n).
\]

\[
\Box
\]

5. Recursive Formula Proof 3: Bijection

In this section, we provide an alternative proof of the second case of Theorem 3.1 with a bijection between tree-sequence tuples and trees in Theorem 5.1. This proof takes inspiration from a special case of [12]. We start by defining the combinatorial objects for this bijection.

Consider the permutation \( w = w_A w_B \), where \( w_A \) is a permutation of \( \{1, 2, \ldots, m\} \) and \( w_B \) a permutation of \( \{m+1, m+2, \ldots, n\} \). Let \( T \) be a spanning tree of \( G_w, T_A \) be the induced subgraph of \( T \) on \( G_{w_A} \), and \( T_B \) be the induced subgraph of \( T \) on \( G_{w_B} \). Since \( w_A w_B \) is an non-inversion cut, \( T_A, T_B \) are both spanning trees. Let \( \deg_{T_A}(v_0) = k_1 \) in \( T_A \) and \( \deg_{T_B}(v_0) = k_2 \).

**Theorem 5.1.** Let \( u = u_A u_B u_C \) be a sequence of length \( k_1 + k_2 - 1 \). In particular, \( u_A \) is a length-\((k_1 - 1)\) sequence with entries from the set \( \{0, m+1, m+2, \ldots, n\} \). \( u_B \) is a length-\((k_2 - 1)\) sequence with entries from the set \( \{0, 1, 2, \ldots, m\} \). \( u_C \) is a length-1 sequence whose only entry is from \( \{0, 1, 2, \ldots, n\} \).

There exists a bijection between the set of \((T, u)\) tuples and the set of trees \( T' \), where \( T, T' \) are spanning trees of \( G_w, G_{w'}, \) and \( u \) a sequence of length \( \deg_T(v_0) - 1 \). Explicitly, this bijection is defined by the following map:

\[
\phi : (T, u) \rightarrow T',
\]

\[
\psi : T' \rightarrow (T, u).
\]

\( \Box \)
The intuition for \( u \) arises from the formula in Theorem 3.1. As \( x_0 \) is replaced by \( x_0 + x_{-m+1} + \cdots + x_n \) in \( f_{w_A} \) and \( x_0 \) is replaced by \( x_0 + x_1 + \cdots + x_m \) in \( f_{w_B} \), sequences \( u_A \) and \( u_B \) determines the destination of each \( x_0 \) under such replacement. The factor of \( (x_0 + x_1 + \cdots + x_n) \) is represented by \( u_C \), as it can choose from any of \( \{0, 1, 2, \ldots, n\} \).

To further explain the intuition, the bijection re-directs every edge that connects a vertex in \( G_{w_A} \) and 0 according to \( u_A \). These vertices can remain connected to 0, or connect to some vertex in \( G_{w_B} \), as \( u_A \) chooses vertices from 0 or \( V_B \). Similarly, edges between \( G_{w_B} \) and 0 are redirected according to \( u_B \). Since \( u_A \) and \( u_B \) have lengths \( k_1 - 1 \) and \( k_2 - 1 \), \( u_C \) determines how 0, the last 0-adjacent vertex in \( G_{w_A} \) and \( G_{w_B} \), and the remainder of \( G_w \) are connected.

In this proof, we define \( \phi \) and \( \psi \) and show they are well-defined. We will show that they are inverses of each other by arguing that each step in \( \phi \) can be inverted by a step in \( \psi \).

In defining \( \phi \), the first step is to decompose \( T \) into components and assign a value to each. In the second step, we re-direct edges based on \( u_A \). In the third step, we devise an algorithm inspired by the [13] to re-direct edges based on \( u_B \). In the fourth step, we make the final two connections according to \( u_C \).

Map \( \psi \) first identifies the components of \( G_{w_A} \) and \( G_{w_B} \), then reverse \( \phi \) step by step. Together, the bijection gives us explicit constructions of all monomials corresponding to the spanning trees, which can be summed and factored to obtain the desired polynomial.

5.1 Define \( \phi \)

We begin the proof by defining \( \phi \) given the tuple \((T, u)\). \( T \) is a spanning tree in \( \widetilde{G}_w \). \( \phi \) is defined in 4 steps:

1. Step 1: Identify the components of \( T_A \) and \( T_B \) and their representatives (“root”). Assign a value to each component.
2. Step 2: Add \( k_1 - 1 \) edges between \( T_A \) and \( T_B \) according to \( u_A \).
3. Step 3: Add \( k_2 - 1 \) edges between \( T_A \) and \( T_B \) according to \( u_B \).
4. Step 4: Add the final 2 edges according to \( u_C \).

Readers are welcome to refer to the universal examples starting at Example 5.1 throughout the steps. This configuration will be used in all examples when possible.

**Example 5.1.** Let \( w_A = 2431 \) and \( w_B = 576 \). Let \( T \) be the tree in Fig. 3 and \( u = 532 \).

5.1.1 Step 1 of \( \phi \)

Consider the \( k_1 \) vertices in \( G_{w_A} \) that are connected to 0, and the \( k_2 \) vertices in \( G_{w_B} \) that are connected to 0. Call these vertices “roots” in \( A \) and \( B \). The induced graph of \( T \) on \( G_w \) has \( k_1 + k_2 \) disconnected components, where each is connected to 0 through the unique root in their component.

To assign values to the components, we rank the components in \( A \) by the smallest vertex in each component, in increasing order. The “value” of this component is its rank. For each component, label its root \( r^A_z \), where \( z \) is the value of this component. This procedure finds \( r^A_1, r^A_2, \ldots, r^A_{k_1} \), which are roots of components valued \( 1, 2, \ldots, k_1 \). The value of each root is the value of the component that contains this root.

Repeat this procedure for \( B \) and obtain \( r^B_1, r^B_2, \ldots, r^B_{k_2} \).

**Example 5.2.** This example is a continuation of Example 5.1. We provide an example of this root-identification and component value-assigning process for a spanning tree \( T \) of the graph \( G_{2431576} \). In particular, the roots, which are vertices connected to \( v_0 \), are \( v_3, v_4, v_5, v_6 \). Considering the smallest vertex in the component attached to each root, we see that the roots ranked from the smallest-valued to the largest-valued, are \( v_4, v_3, v_5, v_6 \). Root \( v_3 \) is ranked after \( v_4 \) because the smallest vertex in the component of \( v_4 \) is \( v_1 \), smaller than the smallest vertex in the component of \( v_3 \).
We illustrate \( \phi \) using this example, following each step.

**Example 5.3.** We provide an example of this root-identification and component value-assigning process for a spanning tree \( T \) of the graph \( \tilde{G}_{12543} \).

![Spanning tree](image)

**Example 5.4.** We give this example as a continuation of Example 5.2. In Fig. 5, \( A \) has two roots: \( v_3, v_4 \). Then \( u_A \) must be of length 1. In this example, \( u = 532 \), so \( u_A = 5 \), \( v_4 \) connects to \( u_A(1) = v_5 \), \( \ast_A = v_3 \).

![Diagram](image)

**Example 5.5.** Suppose \( A \) has four roots: \( r_1^A, r_2^A, r_3^A, r_4^A \). Then \( u_A \) must be of length 3. In this example, \( r_1^A, r_2^A, r_3^A \) connect to \( u_A(1), u_A(2), u_A(3) \).
5.1.3 Step 3 of $\phi$

This step connects $k_2 - 1$ components in $B$ with either 0 or some component in $A$ by running an algorithm inspired by the Prüfer code bijection.

During the iteration of this algorithm, we call a root “available” if it is not in the set $V$. Initially, the set of all roots in $B$ are in $V : V = \{v_1^B, \ldots, v_{k_2}^B\}$. Note that no two trees in $B$ are connected, as each disconnected tree in $A$ connects to at most one tree in $B$.

Run the algorithm starting with $i = 0$. While $|V| > 1$:

1. Copy $V$ into $V'$, which contains all vertices that are currently available.
2. For each entry $u_B(j)$ for $j > i$, if $u_B(j)$ is connected to some vertex $v \in V'$, remove $v$ from $V'$.
3. For all remaining vertices in $V'$, connect $u_B(i)$ with the first vertex in the ordered set $V'$. Remove this vertex from $V$.
4. Increment $i$ by 1.

At the end of this process, there will be one vertex in $V$. Let this vertex be $*_B$.

This process is well-defined because all vertices in $V$ are disconnected at all times, and no cycles are produced through this procedure.

To see this, whenever a connection is made in Step 3, it either connects a root in $V$ that is disconnected from all of $B$, or it indirectly connects two roots in $V$, with one of them removed from $V$ before the next iteration. This ensures that vertices in $V$ remain disconnected at all times. Additionally, no cycles are created because, at the end of each iteration, $|V|$ is always one greater than $(k_2 - i)$ — the remaining number of connections to be made.

At the end of Step 3, the degree of all vertices in $u_A, u_B, and all roots except for $*_A$ and $*_B$ has increased by 1. We finish $\phi$ in Step 4 by adding two edges to increase the degree of $*_A, *_B, 0$, and $u_C$ by 1 each. This is illustrated in Example 5.6.

**Example 5.6.** Continuing from Example 5.4, in Fig. 7, $B$ has two roots: $v_5, v_6$. Then $u_B$ must be of length 1. In this example, $u_B = 3$. Initially, $V' = V = \{v_5, v_6\}$. Since $v_3$ is not connected to $v_5$, we connect $v_3$ to $v_5$ and remove it from $V$, as shown by the purple edge in Fig. 7. $V$ now contains a singular element $v_6$, which we assign to $*_B$.

![Figure 7: $G_{2431576}$ after step 3 of $\phi$. The purple edge is added in step 3.](image-url)
5.1.4 Step 4 of $\phi$

In this step, we make the final 2 connections according to $u_C$, which induces three cases to be handled differently to avoid cycles. Prior to this step, $*A$ and $*B$ are disconnected from each other and from 0. We use the same incomplete spanning tree Fig. 8 as an alternative example, and show how $u_C$ affects the last two edges drawn to finish $\phi$.

Figure 8: Incomplete spanning tree $T^*$ before Step 3. In particular, $*A = r_A^3$ and $*B = r_B^2$. Edges drawn in Steps 2 and 3 are in orange and blue, respectively.

Case 1: $u_C = 0$. In this case, connect both $*A$ and $*B$ with 0. The two added edges create no cycles and connect all components of $A$ and $B$ and 0.

**Example 5.7.** For $u_C = 0$, the two edges added to $T*$ are $(0, r_A^3)$ and $(0, r_B^2)$.

Figure 9: $T^*$ with two added edges (in red) when $u_C = 0$.

Case 2: $u_C = i$, where vertex $i$ is connected to 0 either directly or through some other vertices. In this case, if $i \in A$, connect $*A$ with 0 and connect $i$ with $*B$. If $i \in B$, connect $*B$ with 0 and connect $i$ with $*A$.

**Example 5.8.** For $u_C = r_B^1$, the two edges added to $T*$ are $(0, r_A^3)$ and $(r_A^3, r_B^2)$.

Figure 10: $T^*$ with two added edges (in red) when $u_C = r_B^1$.

The vertex $i$ being connected to 0 prior to this step entails that $u_A$ made this connection in step 2. All other connections $i$ has with $B$ are through step 3, which does not involve $*B$. As such, connecting $i$ and $*B$ does not create cycles. After making these two connections, both $*A$ and $*B$ are connected to 0.

Case 3: this case deals with all other scenarios. That is when $u_C = i \neq 0$ and $i$ is not connected to 0. In this case, connect $*A$ with $*B$ and vertex 0 with vertex $i$.

**Example 5.9.** For $u_C = r_B^1$, the two edges added to $T*$ are $(0, r_A^3)$ and $(r_A^3, r_B^2)$.
Before this step, there are exactly 3 disconnected components: one that contains $*_{A}$, one that contains $*_{B}$, and one that contains 0. Making two connections between 3 disconnected components without creating cycles makes the graph fully connected.

We finish off this section with another example in Example 5.10, continuing from Example 5.6.

**Example 5.10.** With $u_{C} = 2$, $*_{A} = v_{3}$ and $*_{B} = v_{6}$. This scenario falls into Case 3. As such, we connect both $v_{0}$ and $v_{6}$ with $v_{2}$, as done in Fig. 12.

These 4 steps concludes the definition for $\phi$. We proceed to define $\psi$: $T' \rightarrow (T, u)$.

**5.2 Define $\psi$**

The definition will similarly be separated into 4 steps. These 4 steps in $\psi$ reverse the 4 steps in $\phi$.

In the first step, we separate $T'$ into disconnected components and determine the value of each component. In the second and third step, we recover the roots $r_{A1}, \ldots, r_{A k}, r_{B1}, \ldots, r_{B k}$, and $u_{C}$, then reverse the Step 2 and 3 in $\phi$ to recover $u_{A}$ and $u_{B}$. Finally, we put all these information together to recover $T$ and $u$ in the final step.

**5.2.1 Step 1 of $\psi$**

Given $T'$, copy all vertices and edges into $T$, except the edges that are connected to 0 and those that connect a vertex in $A$ to a vertex in $B$. Rank the components in $A$ and $B$ separately by their smallest vertex. Now $T$ has all components and rankings as it does after Step 1 of $\phi$. Steps 2, 3, and 4 fill in the remaining edges between these components and 0 to recover $T$.

**5.2.2 Step 2 of $\psi$**

In this step, we determine the roots and $u_{C}$. As $*_{A}$ was the root in the highest-ranking component in the map $\phi$, we can identify the component that contains $*_{A}$, call it $T_{*_{A}}$. Similarly, we call the component that contains $*_{B}$ $T_{*_{B}}$.

To find $*_{A}$ and $*_{B}$, consider sets $P, Q, R, S$:

$P := \{\text{vertices in } A \text{ that are connected to both 0 and some vertex in } B\}$

$Q := \{\text{vertices in } B \text{ that is connected to } P\}$

$R := \{\text{vertices in } B \text{ connected to 0}\}$

$S := \{\text{vertices in } B \text{ connected to } T_{*_{A}}\}$

We proceed by separating into cases based on whether $T_{*_{A}}$ is to the vertex 0 (directly or indirectly).

Case 1: $T_{*_{A}}$ is connected to the vertex 0.

There are two scenarios where this case could arise:
• when \( u_C = 0 \),
• when \( u_C = i \) where \( i \in B \) and \( i \) and 0 were already connected prior to step 4 of \( \phi \).

We first determine which scenario our case falls into. In this case, \( T_{*,B} \) is the largest component in \( Q \cup R \). If \( T_{*,B} \in R \), then \( u_C = 0 \), and \( *B \) is the vertex that 0 is connected to in \( T_{*,B} \). This is case 1 in step 3 of \( \phi \). If \( T_{*,B} \in Q \), then \( u_C \) is the vertex in \( A \) that connects to \( *B \).

Case 2: \( T_{*,A} \) is not connected to the vertex 0.

The two scenarios that lead to this case are:

• when \( u_C = i \) and \( i \) is not connected to 0 prior to step 4 of \( \phi \),
• when \( u_C = i \) where \( i \in A \) and \( i \) and 0 were already connected prior to step 4 of \( \phi \).

Similarly, we start by determining which scenario our case falls into. In this case, \( T_{*,B} \) is the largest component in \( S \cup R \). If \( T_{*,B} \in S \), \( *B \) is the vertex in \( T_{*,B} \) that connects to 0. The largest component in \( R \) contains \( u_C \), which is connected to 0. If \( T_{*,B} \in R \), there is exactly one vertex in \( B \) that \( T_{*,A} \) connects to. Specifically, this edge that connects \( u_C \) and \( *A \), \( *B \) is the vertex that 0 is connected to in \( T_{*,B} \).

With the knowledge of \( *A \) and \( *B \), we proceed to identify all other roots. We assign all other inter-tree edges in \( T' \) as follows.

For an oriented edge from \( v_s \) to \( v_t \), we call \( v_s \) the “head” and \( v_t \) the “tail.”

If there is an edge between \( *A \) and \( *B \), remove this edge from \( T' \), and orient all remaining edges towards \( *A \) and \( *B \).

If there is no edge connecting \( *A \) and \( *B \), remove all inter-component edges incident to \( *A \) and \( *B \). This separates \( T' \) to either 2 or 3 disconnected sub-trees: one containing \( *A \), one containing \( *B \), and if there is a third sub-tree, it contains 0. Orient the remaining inter-component edges towards \( *A \) and \( *B \), as well as 0, if applicable.

We can identify the roots of \( T \) by looking at oriented edges. The tails of these oriented edges are the roots of \( T' \). Knowing the value of each component helps us rank the roots and obtain \( r_1^A, r_2^A, \ldots, r_{k_1}^A \) and \( r_1^B, r_2^B, \ldots, r_{k_2}^B \).

5.2.3 \( \text{Step 3 of } \psi \)

This step determines \( u_A \) and \( u_B \).

To determine \( u_A \), it suffices to look at the vertices that \( r_1^A, \ldots, r_{k_1}^A \) are connected to, as to reverse Step 2 in \( \phi \). Since the assignment process in Step 2 of \( \phi \) follows the order, \( u_A(i) \) is the vertex in \( B \) that \( r_i^A \) connects to.

To determine \( u_B \), we reverse the Prüfer code inspired algorithm in Step 3 of \( \phi \).

First, initiate an empty sequence \( u_B \). We call a component a “component-leaf” in \( T' \) if it is only connected to one other component (or 0) in \( T' \).

While the length of \( u_B \) is less than \( k_2 - 1 \):

1. Remove all component leaves that are in \( A \).

2. Find the component leaf with the smallest value in \( T' \). Suppose this component-leaf is connected to another component through edge \((r, v_i)\), where \( r \) is the root in this component-leaf. Concatenate \( v_i \) to the end of \( u_B \) and remove this component-leaf from \( T' \).

The iterations end when there are exactly 2 component leaves left in \( T' \). This generates the sequence \( u_B \) of length \( k_2 - 1 \).

5.2.4 \( \text{Step 4 of } \psi \)

Finally, to recover \( T \), we remove all inter-component edges from \( T' \) and connect all roots with 0. The sequence \( u \) is recovered by concatenating \( u_A, u_B, u_C \), found in steps 2 and 3.

As each step in \( \psi \) reverses a step in \( \phi \), they form a bijection between tree-sequence pairs in \( w_Aw_B \) and trees in \( w_Bw_A \).

We end this section by illustrating \( \psi \) with Example 5.11.

Example 5.11. Given the tree \( T' \) Fig. 13, this example illustrates the process to recover \( T \) and the sequence \( u \).
In this section, we extend the bijection in Section 5 to construct a bijection between sequences $u$ of length $n - 1$ and spanning trees $T$ given a permutation $w$ of length $n$. In particular, $u$ is a permutation of $\{1, 2, \ldots, n - 1\}$. This bijection proves Theorem 1.1.

We first set up some notations for this bijection. Given separable permutation $w$, number the intervals between each pair of adjacent entries of $w$ from 1 to $n - 1$. Obtain sets $S^1, S^2, \ldots, S^h$ as follows.

1. Find all intervals where an inversion cut is possible. There may be multiple, exactly one, or none. From left to right, record the intervals in $S^1$.

2. Consider all segments of the permutation separated by $S^1$. For each segment, find all intervals where a non-inversion cut is possible and record their indices in $S^2$.

3. Consider all segments of the permutation separated by $S^1$ and $S^2$. For each segment, find all intervals where an inversion cut is possible and record their indices in $S^3$.

4. Continue this process until all segments are of length 1.

By the end of this process, we should have sets $S^1, S^2, \ldots, S^h$ where $|S^1| + |S^2| + \cdots + |S^h| = n - 1$. Example 6.1 illustrates how $S^1, S^2, \ldots$ are obtained for permutation $w = 7635421$.

**Example 6.1.** For a permutation $w = 7635421$, the inversion cuts are in the second and fifth intervals (solid lines). This determines $S^1 = \{2, 5\}$ and the three segments, $(76), (354), (21)$. Now find the non-inversion cuts, which only exist at the third interval, between 3 and 54 in the second segment (dashed line). Record $S^2 = \{3\}$.
Finally, inversion cuts at the first, fourth, and sixth intervals separate \( w \) into singletons (dotted lines), \( S^3 = \{1, 4, 6\} \).

\[
\begin{array}{cccccc}
7 & 6 & 3 & 5 & 4 & 2 & 1 \\
1 & 2 & 3 & 4 & 5 & 6 & \end{array}
\]

**Definition 6.1.** A decorated spanning tree \( T^* \) is a tree with labeled intervals. The decorated version of a normal spanning tree \( T \) can be obtained by adding labels between the edges that are incident to 0.

In particular, a normal spanning tree-label pair both uniquely determines and can be recovered from a decorated spanning tree.

Given \( T \), suppose the vertex 0 has a degree of \( k_1 + k_2 \). Order the \( k_1 + k_2 - 1 \) vertices along a line by their corresponding values. This creates \( k_1 + k_2 - 1 \) intervals at \( T \). Given a sequence \( u = u_Au_Bu_C \) of length \( k_1 + k_2 - 1 \), write \( u_A(i) \) at the \( i \)-th interval for \( i = 1, 2, \ldots, k_1 - 1 \), write \( u_C \) at the \( k_1 \)-th interval, and write \( u_B(i) \) at the \( (k_1 + i) \)-th interval for \( i = 1, 2, \ldots, k_2 - 1 \). We say the labels of \( T^* \) is a sequence \( l^* \) of length \( \deg(0) - 1 \). This process is illustrated in Example 6.2.

**Example 6.2.** The decorated tree \( T^* \) is obtained from tree \( T \) and \( u = 43551 \). In this example, \( k_1 = 3, k_2 = 3 \). \( u_A = 43, u_B = 55, u_C = 1 \).

![Figure 15: \( T^* \) given \( T \) and \( u = 43551 \).](image)

Before continuing with the proof, we first define \( \phi_1, \phi_2, \psi_1, \psi_2 \) by modifying \( \phi \) and \( \psi \) so that the maps are compatible with the interval labeling. In particular, \( \phi_1 \) and \( \psi_1 \) handles inversion cuts; \( \phi_2 \) and \( \psi_2 \) handles non-inversion cuts.

**Remark 6.1.** For a tree \( T \) and sequence \( u = u_Au_Bu_C \), the labels \( l^* = u_Au_Cu_B \) if all entries in \( u \) are in \( T \).

### 6.1 Define \( \phi_1(T^*), \psi_1(T^*) \)

Consider a separable permutation \( w = w_Aw_B \), where \( w_A \mid w_B \) is an inversion cut. Let \( w' = w_Bw_A \).

The function \( \phi_1 \) maps decorated spanning trees \( T^* \) in the graph \( G_w \) to decorated trees \( T^{**} \) in \( G_{w'} \). The definition of \( \phi_1 \) leverages from \( \phi : (T, u) \rightarrow T' \).

We can obtain an undecorated tree \( T \) and sequence \( u \) from \( T^* \). \( T \) can be obtained by erasing all interval labels in \( T^* \).

Let \( l^* \) be the labels in \( T^* \). For each entry \( l^*(i) \) that is not a vertex of \( T \), we replace that entry with a 0 and record \( l^*(i) \) in \( l^* \). We also record all 0’s in \( l^* \) in \( l^* \). The permutation \( w \) allows us to identify all the components and determine \( k_1 \) and \( k_2 \) — the number of components in \( A \) and \( B \). As such,

\[
u(i) = \begin{cases} 
l^*(i) & i \leq k_1 - 1 \text{ and } l^*(i) \text{ is in } T^* \\
l^*(i + 1) & k_1 \leq i < k_1 + k_2 - 1 \text{ and } l^*(i) \text{ is in } T^* \\
l^*(k_1) & i = k_1 + k_2 - 1 \text{ and } l^*(i) \text{ is in } T^* \\0 & \text{otherwise} \end{cases}
\]

Find \( T' = \phi(T, u) \). The degree of 0 in \( T' \) is one greater than the length \( t^* \). Decorate \( T' \) with \( l^* \) to obtain \( T^{**} \). Example 6.3 gives an example of \( T^{**} \rightarrow T^{**} \).

**Example 6.3.** Suppose \( w = 43512, w_A = 435, w_0 = 12 \). Given \( T^* \) on 6 vertices and \( l^* = 8502 \). Since 8 is not in \( T \), we replace 8 with 0, and record 8 and 0 in \( l^* \). From Eq. (4), \( u = 0520 \).
Example 6.4. Suppose \(w = 6435\). Given separable permutation \(w = w_A w_B\), where \(w_A|w_B\) is a non-inversion cut, we can “glue” the two components in \(A, B\) to obtain \(T^*\). An example is given in Example 6.4.

Example 6.4. Suppose \(w = 12435\), \(w_A = 12\), \(w_B = 435\). Given \(T_A^*\) and \(T_B^*\), and \(u^* = 6\), we can “glue” the two \(v_0\)'s together, and add 6 as the label to the interval in between.

\[
T^* = \begin{array}{c}
v_0 \\
v_1 & v_2 & v_3 \\
v_4 & v_5
\end{array}
\]

\[
T^* = \phi(T, 0520)
\]

\[
T^* = \begin{array}{c}
v_0 \\
v_1 & v_2 & v_3 \\
v_4 & v_5
\end{array}
\]

Figure 16: \(T^*\) obtained from \(\phi(T, u)\) given \(T\).

The map \(\psi_1\) is the inverse of \(\phi_1\), mapping decorated spanning trees \(T''\) in \(G''_{w}\) to decorated spanning trees \(T^*\) in \(G_{w}\). Similarly, we use \(\psi\) to define \(\psi_1\).

We first remove the labels \(l''\) on \(T^*\) to obtain \(T''\). The degree of 0 in \(T''\) is exactly one more than the length of \(l''\). Obtain \((T, u)\) from \(\psi(T'')\). The number of 0's in \(u\) is identical to the length of \(l''\). Decorate \(T\) with \(u\), replacing the first 0 in \(l''\) (1), the second 0 by \(l''(2)\), and so on. The decorated spanning tree is the image of \(\psi_1(T'')\).

6.2 Define \(\phi_2(T_A^*, T_B^*, u^*), \psi_2(T'')\)

We similarly define how to execute and repair non-inversion cuts. Given separable permutation \(w = w_A w_B\), where \(w_A|w_B\) is a non-inversion cut, we define \(\phi_2, \psi_2\) based on \(w\).

\(\phi_2\) repairs the cut by putting combining two decorated trees. For decorated spanning trees \(T_A^*, T_B^*\) of \(A, B\), and a length -1 sequence \(u^*\), \(\phi_2(T_A^*, T_B^*, u^*) = T^*\). \(T^*\) identifies the 0 vertex in \(T_A^*\) with the 0 vertex in \(T_B^*\), and writes \(u^*\) on the interval in between to obtain \(T^*\). An example is given in Example 6.4.

Example 6.4. Suppose \(w = 12435\), \(w_A = 12\), \(w_B = 435\). Given \(T_A^*\) and \(T_B^*\), and \(u^* = 6\), we can “glue” the two \(v_0\)'s together, and add 6 as the label to the interval in between.

\[
T^* = \begin{array}{c}
v_0 \\
v_1 & v_2 & v_3 \\
v_4 & v_5
\end{array}
\]

\[
T^* = \phi_2(T^*)
\]

\[
T^* = \begin{array}{c}
v_0 \\
v_1 & v_2 & v_3 \\
v_4 & v_5
\end{array}
\]

Figure 17: Adding \(u^* = 6\) to the interval between \(T_A^*\) and \(T_B^*\) to obtain \(T^*\).

Map \(\psi_2\) is the inverse of \(\phi_2\) and executes the cut. For a decorated tree \(T^*\), \(\psi_2(T^*) = (T_A^*, T_B^*, u^*)\), where \(T_A^*\) is a spanning tree in \(A\), \(T_B^*\) a spanning tree in \(B\), and \(u^*\) a length -1 sequence. Given \(w\), components in \(A, B\) can be separated easily to obtain \(T_A^*\) and \(T_B^*\). \(u^*\) is the label between these two trees.

Given \(w\), we can obtain sets \(S^1, S^2, \ldots\). We proceed to define \(\Phi\) and \(\Psi\) using \(\phi_1, \phi_2, \psi_1, \psi_2\).

6.3 Define \(\Phi : u \rightarrow T\)

Given separable permutation \(w\) of length \(n\), \(\Phi\) constructs the spanning tree \(T\) given a length-\(n\) sequence \(u\) by applying \(\phi_1\) and \(\phi_2\) to repair the cuts.

1. First consider the last set of cuts: \(S^h = \{c^h_1, c^h_2, \ldots, c^h_{n_h}\}\). Do \(\phi_1\) or \(\phi_2\) (depending on whether the last set of cuts is inversion or not) at \(c^h_{n_h}, \ldots, c^h_2, c^h_1\), in that order, with inputs \(u_{c^h_1}, \ldots, u_{c^h_{n_h}}, u_{c^h_2}, u_{c^h_1}\). This adds 2\(n_h\) edges to the tree.

2. Do \(S^{h-1}\) with the other kind of cut, and continue. For every \(\phi_1(T^*)\), suppose the cut entry is \(c_i\). Particularly, the first the first \(k_1 - 1\) entries of \(u^*\) are labels in \(T_A^*\), the next \(k_2 - 1\) entries are labels in \(T_B^*\), and the \((k_1 + k_2 - 1)\)-th entry is \(w_{c_i}\).

3. Continue until all cuts have been repaired.

We provide an example of these steps with sequence \(u = 530172\) for permutation \(w = 5762431\) in Example 6.5. At the end of this process, we should obtain a spanning tree whose corresponding monomial is \(x_0 x_1 x_2 x_3 x_5 x_7\).
Example 6.5. To construct the spanning tree for \( w = 5762431 \) and \( u = 530172 \), we first determine the interval sets: \( S_1 = \{3, 6\}, S_2 = \{1, 4\}, S_3 = \{2, 5\} \).

We start with \( T^*_0 \):

\[
T^*_0 = \begin{array}{c}
\begin{array}{cccccccc}
& v_1 & & v_2 & & v_3 & & v_4 & & v_5 & & v_6 & & v_7 \\
& & & & & & & & & & & & \\
\end{array}
\end{array}
\]

Figure 18: \( T^*_0 \) between the preparation process begins for a tree on 8 vertices.

To construct the tree \( T' \) with \( \Phi \), we repair cuts in order: 5, 2, 4, 1, 6, 3.

We start by repairing cuts in \( S_3 \). Since \( S_3 \) contains inversion cuts, we compute \( \phi_1(T^*_0) = T^*_1 \), at the fifth interval with \( u(5) = 7 \). This connects leaves 5 and 6. Since 7 is not in the set \( \{5, 6\} \), we treat it as a 0 to obtain the tree where both \( v_5 \) and \( v_6 \) are connected to \( v_0 \), then write 7 on their interval. Next, we repair the cut at the second interval with \( u(2) = 3 \). Fig. 19 is the tree after these two steps.

\[
T^*_1 = \begin{array}{c}
\begin{array}{cccccccc}
& v_1 & & v_2 & & v_3 & & v_4 & & v_5 & & v_6 & & v_7 \\
& & & & & & & & & & & & \\
\end{array}
\end{array}
\]

Figure 19: \( T^*_1 \) after executing repairs in \( S_3 \).

The second step is to repair cuts in \( S_2 \) with \( \phi_2 \). With \( u(4) = 1 \) and \( u(1) = 5 \), we obtain Fig. 20

\[
T^*_2 = \begin{array}{c}
\begin{array}{cccccccc}
& v_1 & & v_2 & & v_3 & & v_4 & & v_5 & & v_6 & & v_7 \\
& & & & & & & & & & & & \\
\end{array}
\end{array}
\]

Figure 20: \( T^*_2 \) after executing repairs in \( S_2 \).

The third step is to repair cuts in \( S_1 \) using \( \phi_1 \). We first repair the cut at the 6-th interval with \( u(6) = 2 \). Since 1 and 2 are not in the set of vertices \( \{5, 6, 7\} \), we replace them with 0 to get \( l = 070 \). Hence, by Eq. (4), \( u = 007 \). We find \( \phi(T_{4,5,6,7}, 007) \).

\[
T_{4,5,6,7} = \begin{array}{c}
\begin{array}{cccccccc}
& v_0 & & v_4 & & v_5 & & v_6 & & v_7 \\
& & & & & & & & & & & & \\
\end{array}
\end{array} \quad T^*_3 = \phi(T_{4,5,6,7}, 007)
\]

Figure 21: Adding \( u^* = 6 \) to the interval between \( T^*_A \) and \( T^*_B \) to obtain \( T^* \). The two labels that will go on \( T^*_3 \) are 1 and 2.

Finally, we repair the cut at interval 3 with \( u(3) = 0 \). In this case, the labels are 5012, hence \( u = 5021 \).
\[
T^* = \begin{array}{ccc}
& v_0 & \\
v_1 & 5 & 0 \\
v_2 & 3 & 4 \\
v_3 & & v_7 \\
v_4 & & 2 \\
v_5 & & 1 \\
v_6 & & \\
\end{array}
\begin{array}{ccc}
T'^* = \phi_1(T^*) \\
\begin{array}{ccc}
& v_0 & \\
v_1 & 5 & 0 \\
v_2 & 3 & 4 \\
v_3 & & v_7 \\
v_4 & & 2 \\
v_5 & & 1 \\
v_6 & & \\
\end{array}
\end{array}
\]

Figure 22: Final repair at the third interval to complete \(\Phi(530172)\) to obtain the tree representing monomial \(x_0x_1x_2x_3x_5x_7\).

6.4 Define \(\Psi\)

Take tree \(T\) with labeled vertices. Order the labeled vertices in a line. This creates \(deg(x_0) - 1\) intervals. Label all of them 0.

1. Do \(\psi_1\) at \(c_1^1, c_2^1, \ldots, c_{n_1}^1\), in that order. This determines the \(c_1^1\)th, \(c_2^1\)th, \ldots, \(c_{n_1}^1\)th entry in \(u\). At each cut, separate the tree into two.

2. Do \(\psi_2\) at \(c_1^2, c_2^2, \ldots, c_{n_1}^2\). This determines the \(c_1^2\)th, \(c_2^2\)th, \ldots, \(c_{n_1}^2\)th entry in \(u\). At each cut, separate the tree into two.

3. Repeat Steps 1 and 2 for each tree until the entire sequence \(w\) is obtained.

Note that this process changes 0 to other vertices without causing conflicts. This is because at each inversion, according to Theorem 1.1, the linear factor associated with all intervals to the right of the inversion has every vertex to the left of the inversion, and the linear factor associated with all intervals to the left of the inversion has every vertex to the right of the inversion.

7. Non-Separable Permutations

In this section, we prove that \(f_w\) for non-separable permutations \(w\) are not linearly factorizable with two lemmas.

**Lemma 7.1.** Let \(G\) be a graph on \([n]\) where the vertex \(i\) has degree \(k_i\). Then the monomial \(x_0^{n-r}x_i^r\) in the polynomial \(f_G\) has coefficient \(\binom{k_i}{r}\).

**Proof.** Consider a spanning tree \(T\) where \(m(T) = x_0^{n-r}x_i^r\). \(T\) has at most two non-leaf vertices: 0 and \(i\), which are connected to each other. For the remaining \(n-1\) vertices, \(r-1\) are connected to \(i\), and the rest are connected to 0 in \(T\). Therefore, it suffices to count the number of ways of selecting the \(r\) vertices to connect to \(i\). Since \(k_i\) vertices are connected to \(i\), and all vertices are connected to 0 in \(G\), there are \(\binom{k_i}{r}\) ways to choose such \(r\) vertices. \(\square\)

**Lemma 7.2.** For any graph \(G = (V,E)\) where \(f_G\) is linearly factorizable, any induced subgraph of \(\tilde{G}\) is also linearly factorizable. That is, for a subset of vertices \(S \subset V\), let \(\tilde{G}_S\) be the induced subgraph on \(\tilde{S}\). Then \(f_{\tilde{S}}\) can also be linearly factored.

**Proof.** Consider the restriction \(f^R = f_G|_{x_i=0} \notin S\), which must still be linearly factorizable. \(f^R\) is the sum of the spanning trees of \(\tilde{G}\) trees where all vertices except \(v_0\) and those in \(S\) is a leaf. Since \(f_{\tilde{S}}\) gives us all spanning trees of \(\tilde{G}_S\), to complete \(f^R\), it suffices to consider the parents of the leaves. As such, \(f^R = f_{\tilde{S}} \cdot h(V)\), where each monomial in \(h(V)\) is the product of the vertices of the parent of all vertices in \(V\setminus S\). As illustrated by Example 7.1. Given \(f^R\) has linear factorization, \(f_{\tilde{S}}\) must likewise. \(\square\)

**Example 7.1.** Let \(S\) contain \(4\) vertices in \(G\). All vertices in \(V\setminus S\) are leaves and connected to the spanning tree via either \(v_0\) or \(S\), as illustrated in Fig. 23.
Figure 23: An example of a monomial in $h(V)$, where $v_0$ and vertices in $S$ are in black and vertices in $V\setminus S$ and edges incident to them are in blue. In this case, the corresponding monomial in $h(V)$ is $x_0 x_{S_1}^2 x_{S_2}$, as the 4 blue vertices are connected to $v_{S_1}, v_{S_2}, v_{S_3}, v_0$, respectively.

We are now ready to prove Theorem 1.2.

Proof of Theorem 1.2. Suppose $f_w$ has linear factorization. Since each spanning tree connects $n+1$ vertices, $f_w$ is of degree $n$. Let $f_w = (x_0 + a_1 x_1 + b_1 x_2 + c_1 x_3 + \ldots)(x_0 + a_2 x_1 + b_2 x_2 + c_2 x_3 + \ldots)\ldots(x_0 + a_n x_1 + b_n x_2 + c_n x_3 + \ldots)$. Specifically, the coefficient for $x_0$ is 1 for all linear factors, because there is exactly one spanning tree of $G_w$ where the degree of vertex 0 is $n+1$ and the degree of every other vertex is 1.

Let $k_i$ denote the degree of vertex $i$ in the graph $G_w$. Consider the product of these linear factors and apply Lemma 7.1 to $x_0$ to obtain the system of equations as follows, where $r = 1, 2, \ldots, n$.

$$\sum_{S \subseteq V, |S| = r} \prod_{x \in S} a_x = \binom{k_1}{r} = 1$$

As such, for some subset $s_1 \subseteq S$ of size $k_1$,

$$\begin{cases} a_i = 1 & i \in s_1 \\ a_i = 0 & \text{otherwise} \end{cases}$$

By symmetry, this result applies to other vertices as well.

For a 2413 or 3142-pattern containing permutation $w$, let the first occurrence of the pattern be $w_a, w_b, w_c, w_d$ for $a < b < c < d$ and apply Lemma 7.2 to $v_0, v_a, v_b, v_c, v_d$. Thus, it suffices to show that $f_{w_1}$ and $f_{w_2}$ for $w_1 = 2413, w_2 = 3142$ cannot be linearly factored.

Consider inseparable permutation $w_1 = 2413$. Suppose $f_{w_1}$ can be factored into linear factors. The degrees of the vertices in $G_{w_1}$ are $k_1 = 1, k_2 = 2, k_3 = 2, k_4 = 1$. Hence, by Lemma 7.1, we assign 0’s and 1’s to coefficients $a_1, a_2, a_3, \ldots, d_1, d_2, d_3$ in the expression for $f_{w_1}$.

$$f_{w_1} = (x_0 + a_1 x_1 + b_1 x_2 + c_1 x_3 + d_1 x_4) \cdot (x_0 + a_2 x_1 + b_2 x_2 + c_2 x_3 + d_2 x_4) \cdot (x_0 + a_3 x_1 + b_3 x_2 + c_3 x_3 + d_3 x_4)$$

Without loss of generality, let $a_1 = 1$ and $a_2 = a_3 = 0$. Since the tree in Fig. 24 is a spanning tree of $G_{w_1}$, $x_0 x_1 x_4$ is a monomial in $f_{w_1}$. Since only one of $d_1, d_2, d_3$ is equal to 1, $d_1 = 0$. Again, without loss of generality, let $d_2 = 1, d_3 = 0$ is the only valid assignment for the remaining coefficients. Notice that $x_1 x_2 x_2, x_1 x_3 x_3, x_4 x_4 x_2, x_4 x_3 x_3$ are not spanning trees of $G_{w_1}$, hence $b_1 = b_2 = c_1 = c_2 = 1$ and $b_3 = c_3 = 0$. Therefore, $f_{w_1} = (x_0 + x_1 + x_2 + x_3 + x_4) \cdot (x_0 + x_1 + x_2 + x_3 + x_4) \cdot x_0$.

However, $f_{w_1}$ does not contain the monomial $x_1 x_2 x_3$, which is a spanning tree of $G_{w_1}$ as shown in Fig. 24. This invalidates the only coefficient assignment and concludes the proof that $f_{w_1}$ cannot be factored linearly.

Figure 24: Trees corresponding to the monomial $x_0 x_1 x_4$ and $x_1 x_2 x_3$ in $G_{2413}$. 

ECA 4:1 (2024) Article #S2R6 19
A similar strategy proves that \( f_w \) cannot be factored linearly. The graph \( G_w \) has degrees \( k_1 = k_4 = 2, k_2 = k_3 = 1 \). Assume linearly factorizability and set up

\[
f_w = (x_0 + a_1 x_1 + b_1 x_2 + c_1 x_3 + d_1 x_4) \cdot (x_0 + a_2 x_1 + b_2 x_2 + c_2 x_3 + d_2 x_4) \\
\quad \cdot (x_0 + a_3 x_1 + b_3 x_2 + c_3 x_3 + d_3 x_4)
\]

Given the monomial \( x_0 x_2 x_3 \) is a valid tree where the degree on \( v_2 \) and \( v_3 \) are both 2, without loss of generality, let \( b_1 = b_2 = b_3 = c_1 = c_3 = 0 \) and \( a_1 = a_2 = c_1 = d_3 = 0 \). Since \( a_1 + a_2 + a_3 = d_1 + d_2 + d_3 = 2 \), we must assign \( a_1 = a_2 = d_1 = d_2 = 1 \). Therefore, \( f_w = (x_0 + x_1 + x_2 + x_4) \cdot (x_0 + x_1 + x_3 + x_4) \cdot x_0 \).

However, \( f_w \) does not contain the monomial \( x_1 x_2 x_4 \), which is a spanning tree of \( G_{w_2} \) as shown in Fig. 25.

This contradiction concludes the proof that \( f_w \) cannot be factored linearly.

\[
\text{Figure 25: Trees corresponding to the monomial } x_0 x_2 x_3 \text{ and } x_1 x_2 x_4 \text{ in } G_{3142}.
\]

By Lemma 7.2, \( f_w \) for all permutations \( w \) containing pattern 2413 and 3142 cannot be factored into linear factors.

8. Further Discussions

**Definition 8.1.** For a non-separable permutation \( w \), let \( g_w \) be the polynomial derived according to Theorem 1.1.

**Remark 8.1.** Since \( g_w \) is a product of linear factors and that Theorem 1.2 asserts that \( f_w \) is not linearly factorizable for non-separable permutations, \( f_w \neq g_w \) when \( w \) is not separable.

Define \( d_w = f_w - g_w \). We deduce the following conjecture for 25314 and 41352-avoiding inseparable permutations.

**Conjecture 8.1.** The polynomial \( d_w \) has positive coefficients for permutations \( w \), where \( w \) contains patterns 2413 or 3142, but avoids 25314 and 41352.

We prove a weaker version of this conjecture.

**Proposition 8.1.** The polynomial \( d_w \) has positive coefficients for permutations \( w \), if \( w \) contains exactly one occurrence of 2413 or 3142 and avoids 25314 and 41352.

**Lemma 8.1.** For permutation \( w \) with exactly one occurrence of 2413 that avoids 3142, 25314, and 41352. Suppose it occurs at \( a, b, c, d \) — that is, \( a < b < c < d \) and \( w(c) < w(a) < w(d) < w(b) \). Then we must have \( a + 3 = b + 2 = c + 1 = d \) and \( w(c) + 3 = w(a) + 2 = w(d) + 1 = w(b) \).

Lemma 8.1 similarly applies to permutations \( w \) that contain exactly one occurrence of 3142 and avoid 2413, 25314, and 41352. The 3142 pattern must occur at some \( a, b, c, d \), where \( b = a + 1, c = a + 2, d = a + 3 \), with \( w(b) + 3 = w(d) + 2 = w(a) + 2 = w(c) \).

With Lemma 8.1, we return to the proof for Proposition 8.1.

**Proof.** As a result of Lemma 7.2, for an induced subgraph on vertices \( S \subseteq V \) of \( G = (V, E) \), \( f_G = f_S \cdot h(G) \), where \( h(G) \) is some polynomial that specifies the connection for vertices in \( V \setminus S \).

For a permutation \( w \) that contains the pattern 2413, we compare the factored polynomial \( g \) obtained from \( w \) and \( w' \), where \( w'(i) = w(i) \) for \( i \neq a, a + 3 \), and \( w'(a + 3) = w(a), w'(a) = w(a + 3) \). \( h(w) \) is a product of the linear factors at all other intervals. Let \( S = \{ v_a, v_b, v_c, v_d \} \) denote this set of vertices in both \( G_w \) and \( G_w' \).

Lemma 8.1 asserts that for all vertices \( v_i \notin S \) and vertices \( v_j \in S \), if \( (v_i, v_j) \) is an edge in \( G_w \), it must also be an edge in \( G_{w'} \). As such, the same polynomial \( h(G) \) holds for both the expression for \( w \)

\[
f_{G_w} = f_{S_w} \cdot h(G) \tag{5}
\]

and the expression for \( w' \)

\[
f_{G_{w'}} = f_{S_{w'}} \cdot h(G) \tag{6}
\]
By Theorem 1.1, the linear factor at each interval for \( w \) and \( w' \) are identical with the except of between \( a \)-th and \( (a + 2) \)-th interval. In \( g(w') \), these two intervals have an additional term due to the added inversion by interchanging the \( w(a) \) with \( w(c) \). Therefore, let

\[
g(S_w) = (x_0 + x_a)(x_0 + x_a + x_c + x_d)(x_0 + x_b),
\]

and

\[
g(S_{w'}) = (x_0 + x_a + x_b + x_c + x_d)(x_0 + x_a + x_b)(x_0 + x_a + x_b + x_c).
\]

Locally, \( G_{S_w} \) and \( G_{S_{w'}} \) correspond to the following graphs.

\[\text{Figure 26: The induced graph on vertices } v_a, v_b, v_c, v_d \text{ for } G_w \text{ and } G_{w'}.\]

The only edge added by interchanging the entries \( a \) and \( d \) is the edge \( (v_a, v_d) \), hence all spanning trees that are in \( f(w') \) but not \( f(w) \) must contain this edge.

However, expanding Eq. (7) and Eq. (8), \( g(S_{w'}) - g(S_w) =

\[
(x_0 + x_a + x_b + x_c + x_d)(x_0 + x_a + x_c + x_d + x_a x_c + x_b x_d + x_a x_d)
\]

(9)

We notice that, \( g(S_{w'}) - g(S_w) \), which is a monomial that corresponds to the tree Fig. 27 which does not have \( (v_a, v_d) \) as an edge. All other monomials in \( g(S_{w'}) - g(S_w) \) correspond to trees that contain the edge \( (v_a, v_d) \).

\[\text{Figure 27: The induced subtrees on vertices } v_0, v_a, v_b, v_c, v_d \text{ for trees that are in } f(w) \text{ but not } g(w).\]

As such, \( d(w) = f(w') - g(w) \) must have positive coefficients.

For the second part, we consider permutations that contain pattern 3142. Let \( w \) be the permutation whose only occurrence of 3142 at positions \( a, b, c, d \) and avoid patterns 25314 and 41352. Consider permutation \( w' \), where \( w'(i) = w(i) \) for \( i \neq a, d \) and \( w'(a) = w(d), w'(d) = w(a) \). By Lemma 8.1, \( b = a + 1, c = a + 2, d = a + 3 \) and \( w(b), w(d), w(a), w(c) \) are consecutive integers. Since \( G_{S_w} \) contains one additional edge, \( (v_a, v_d) \), than \( G_{S_{w'}} \), the spanning trees that terms in \( f_{S_w} - f_{S_{w'}} \) must contain this edge.

Similarly, by Theorem 1.1, the linear factor at each interval for \( w \) and \( w' \) are identical with the except of between \( a \)-th and \( c \)-th interval. In \( g(S_w) \), these two intervals have an additional term due to the added inversion by interchanging the \( w(a) \) with \( w(c) \).

Specifically,

\[
g(S_w) = (x_0 + x_a + x_b + x_d) \cdot x_0 \cdot (x_0 + x_a + x_c + x_d)
\]

and

\[
g(S_{w'}) = (x_0 + x_a + x_b) \cdot x_0 \cdot (x_0 + x_c + x_d).
\]

\( G_{S_w} \) and \( G_{S_{w'}} \) correspond to the following graphs.
Figure 28: The induced graph on vertices $v_0, v_a, v_b, v_c, v_d$ for $G_w$ and $G_{w'}$.

Expanding Eq. (10) and Eq. (11) finds that there are 7 spanning trees in $\widetilde{G}_{S_w}$, which correspond to the monomials in the difference

$$g(S_w) - g(S_{w'}) = x_0(x_0x_d + x_cx_d + x_d^2 + x_0x_a + x_d^2 + x_ax_b + x_ax_d)$$

However, we find 12 spanning trees of $\widetilde{G}_{S_w}$ that contain the edge $(v_a, v_d)$ via enumeration. In addition to the 7 monomials in $g(S_w) - g(S_{w'})$, the 5 trees that contain $(v_a, v_d)$ correspond to the monomials in

$$x_ax_d(x_0 + x_a + x_b + x_c + x_d).$$

As such, the factored polynomial $g(S_w)$ undercounts its spanning trees. $d(w) = f(w) - g(w)$ have positive coefficients.

References