The chromatic symmetric function in the star basis

By
Michael Gonzalez

SENIOR HONORS THESIS
ROSA C. ORELLANA, ADVISOR
DEPARTMENT OF MATHEMATICS
DARTMOUTH COLLEGE
JUNE, 2023

Contents

Acknowledgements

I would like to thank Professor Orellana for the time she put into working with me on this project and for the passion for mathematics that she has displayed as a teacher and mentor. She helped me undertand what math is during my first year at Dartmouth, when I took Math 28 Introduction to Combinatorics with her. Since then, I've had an amazing journey learning and teaching math to others. Along this journey, I've met some other incredible professors who have helped to propel me further and seek out the beauties that math has to offer. Professor Jayanti and Professor Chakrabarti have been two such professors, who I thank greatly.

I would also like to thank Mario ('25), who has motivated and inspired me to think about this project. My discussions with him have led to many insights, some of which are presented in this thesis.

I would also like to thank my parents, sister, and grandma, whose endless sacrifices and support have allowed me to graduate from Dartmouth College and to write this thesis. Finally, thank you to my friends who helped me build a home away from home. I love you all.

Abstract

This thesis approaches Stanley's tree isomorphism conjecture by determining properties of a tree that are recoverable from its chromatic symmetric function (CSF) in the star basis. First, we explore and modify a recursive relation introduced in a recent paper to compute the CSF and specify two algorithms that relate the relation to CSF calculations. We use these algorithms to prove some simple yet insightful results about coefficients in the CSF. We then show that the leading term of the CSF in the star basis is completely determined by properties of the tree, allowing us to distinguish large classes of trees. We then consider a matrix representation of the chromatic symmetric functions of trees on n vertices and show that the subspace of Λ_k spanned by these vectors has dimension $p(n) - n + 1$. We also give a construction for a basis for this space, which permit us to write CSFs of trees in terms of CSFs of these basis elements, an area for future study.

Chapter 1

Introduction

Graphs are ubiquitous in mathematics and computer science. They model relationships in solutions to various mapping and scheduling problems. Beyond their utility, they are beautiful mathematical objects to study. They have very interesting properties and a satisfying visual representation. In 1995, Stanley introduced the chromatic symmetric function, a generalization of a graph's chromatic polynomial [St1]. Unlike the chromatic polynomial, however, the chromatic symmetric function does not satisfy a deletion-contraction recurrence.

In [St1], Stanley expresses the chromatic symmetric function in terms of different bases for symmetric functions. An interesting open problem (and the focus of this thesis) is whether the chromatic symmetric function determines trees. That is, for any two trees, are their chromatic symmetric functions distinct? In [**?**, **?**, bHJ] Heil shows that this question can be answered in the affirmative for trees having up to 29 vertices. Given that there are 8 billion trees with at most 29 vertices, this evidence is quite convincing. That said, a more generalize argument is necessary.

In 2016, Cho in [CvW] introduces a new basis for the algebra of symmetric functions constructed from star graphs, called the star-basis. Though the chromatic symmetric function does not satisfy a deletion-contraction formula, in 2022 Aliste-Prieto, de Mier, Orellana, and Zamora showed that, when expressed in terms of the star-basis, it does satisfy a modified variation of this relation called the deletion-near-contraction relation [AMOZ].

The main objective of this thesis is to use the deletion-near-contraction relation to study the properties of a tree that can be determined from the coefficients of its chromatic symmetric function. I start my thesis by connecting a derivation of the chromatic symmetric function of a tree with ternary trees and explain how this process may be useful in the context of Stanley's conjecture. Then, in Chapters 4 and 5 I go on to describe two major results that follow using these techniques and discuss how these results allow us to distinguish classes of trees. The thesis ends with a Python implementation of the deletion-near-contraction relation and data tables that have been useful in this work.

Chapter 2

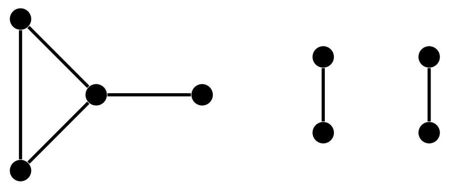
Preliminaries

2.1 Forests

In this section, we review some fundamental definitions and statements about forests, a special kind of graph. Most of these terms can be found in any introductory graph theory textbook like [West]. Some possibly new terms like "leaf edge" and "internal edge" are defined here.

Definition 1. *A* **simple graph** *G is a pair $(V(G), E(G))$, where $V(G)$ is a nonempty, finite set of elements called* **vertices** *and $E(G) \subseteq \{\{u, v\}|u, v \in V, u \neq v\}$. The elements of $E(G)$ are called* **edges***. A graph with no edges and exactly one vertex is called the* **trivial graph***.*

In this thesis we use uv (or, equivalently, vu) to denote the edge $\{u, v\}$.

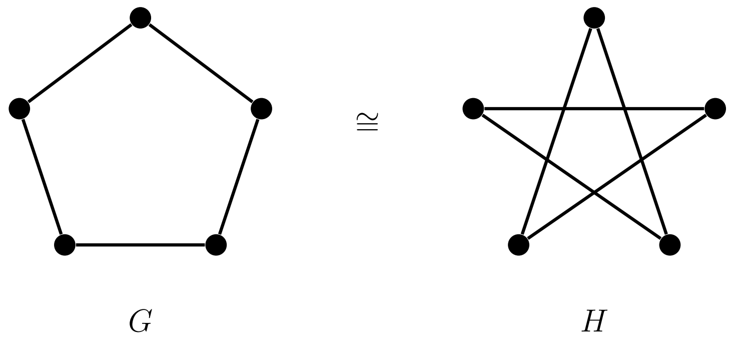


Above is an example of a graph. Observe that the graph is **unlabeled**, i.e., the vertices do not have any labels. In fact, all of the graphs we consider in this thesis are unlabeled. This may be counterintuitive as a graph's vertex set is defined as a set of labels. As a convention, when we refer to a vertex in an unlabeled graph, we assume an arbitrary, temporary labeling of the vertex set that "expires" once the discussion of said vertex is complete.

There are more general types of graphs that involve multiple edges between pairs of vertices and edges between a single vertex. We won't consider such graphs as they lack relevance in the domain of proper coloring, the underlying theme of this thesis.

Definition 2. *Two graphs G and H are* **isomorphic***, denoted $G \cong H$, if there exists a bijection $\phi : V(G) \to V(H)$ such that for all $u, v \in V(G)$, $uv \in E(G)$ if and only if $\phi(u)\phi(v) \in E(H)$.*

Essentially, two graphs are isomorphic if one can be obtained from the other by a relabeling of vertices.



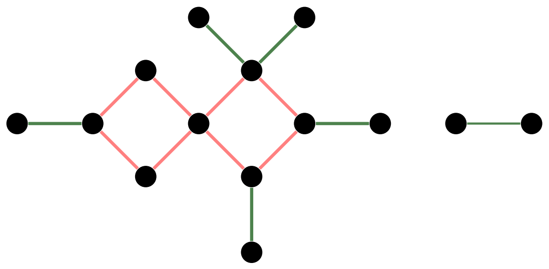
The above two graphs are isomorphic. Though visually we can convince ourselves this is true, formally $G \cong H$ since for an arbitrary vertex labeling of $V(G)$ and $V(H)$, Definition 2 is satisfied.

For convenience, we define here some commonly used terminology in graph theory that will be used throughout the thesis. Two vertices $u, v \in V(G)$ are **adjacent** if $uv \in E(G)$. We say an edge uv is **incident** to its **endpoints**: u and v. An edge e is incident to an edge e' if e and e' share exactly one endpoint. The **degree** of a vertex

u, denoted $\deg(u)$, is the number of edges incident to u. The **neighborhood** of a vertex u, denoted $N(u)$, is the set of vertices adjacent to u. A vertex is **isolated** if $\deg(u) = 0$, and a vertex u is a **leaf** if $\deg(u) = 1$. Otherwise, i.e., if $\deg(u) \geq 2$, we call u an **internal vertex**.

In this thesis, it will be useful to consider edges whose endpoints are internal vertices. To distinguish these edges from all other edges in a graph, we introduce the following vocabulary.

Definition 3. *A* **leaf edge** *is any edge incident to a leaf. An* **internal edge** *is any edge uv, where $\deg(u), \deg(v) > 1$. We denote the set of leaf edges of a graph G by $L(G)$ and the set of internal edges of G by $I(G)$.*

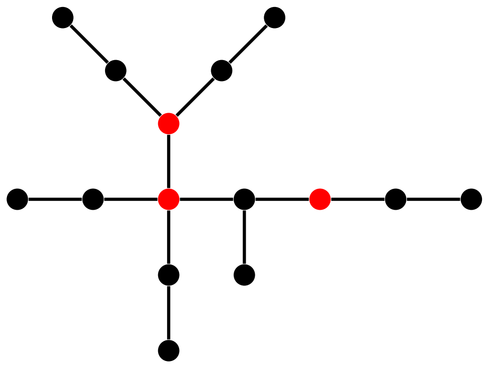


In the graph G above, the leaf edges are colored green and internal edges are colored red. By definition, it follows that every edge in a graph is either a leaf edge or an internal edge.

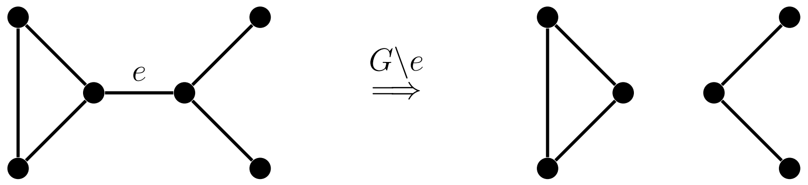
It will also be useful to distinguish internal vertices that have no leaves in their neighborhoods.

Definition 4. *A* **deep vertex** *is an internal vertex whose neighborhood contains only internal vertices.*

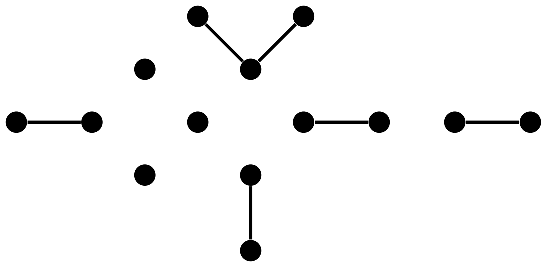
In the following graph the deep vertices are colored red:



Definition 5. *In a graph G,* **deletion of edge** *$e = uv$ is simply the removal of uv from $E(G)$. The resulting graph is denoted $G \backslash e$.*



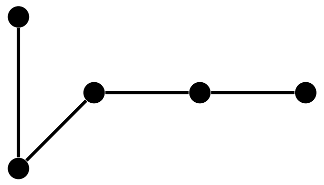
More generally, given a graph G and a subset of edges $E' \subseteq E(G)$, we define $G \backslash E'$ to be the graph obtained by deleting every edge of G in E'. For example, letting G be the graph from Definition 3, $G \backslash I(G)$ is the following graph:



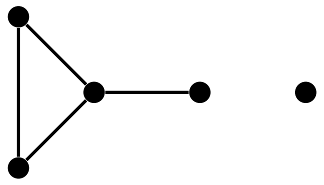
Definition 6. *A* **path** *in G is a finite sequence of distinct edges $v_0v_1, v_1v_2, \ldots v_{m-1}v_m$, where $v_0, \ldots, v_{m-1}$ are distinct vertices in G. A* **cycle** *is a nonempty path where $v_0 = v_m$.*

Definition 7. *A graph G is* **acyclic** *if there are no cycles in G. Otherwise, G is* **cyclic***. We say G is* **connected** *if for every $u, v \in V(G)$, there exists a path in G from u to v. Otherwise, G is* **disconnected***.*

The following graph is acyclic and connected:



The graph below is cyclic and disconnected:



Definition 8. *A* **subgraph** *of G is any graph G' for which $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. A* **connected component** *of G is a maximal, connected subgraph of G.*

For example, the cyclic and disconnected graph in the preceding example consists of two connected components.

In this thesis, we consider a special class of simple graphs called forests.

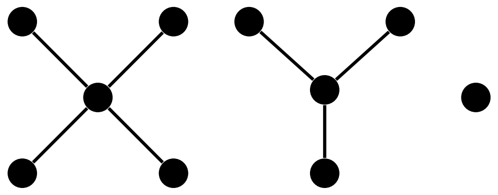
Definition 9. *A* **forest** *is an acyclic graph. A* **tree** *is a connected, acyclic graph.*

The **union of graphs** $G_1, G_2, \ldots, G_k$, denoted $\bigcup_{i=1}^{k} G_i$, is the graph consisting of connected components $G_1, G_2, \ldots, G_k$. Thus, a forest can be equivalently defined as a union of one or more trees.

We will frequently refer to a special type of tree called a "star graph", as well as the forest equivalent, a "star forest".

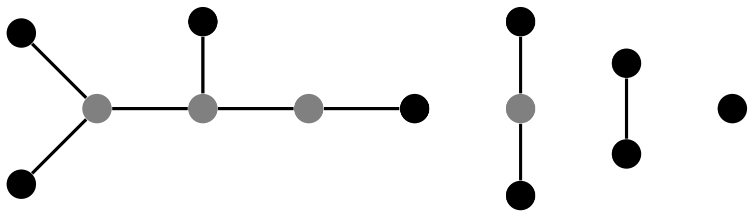
Definition 10. *The* **star graph** *on n vertices, denoted St_n, is the tree on n vertices with no internal edges. A star forest is a union of one or more star graphs.*

The star forest $St_5 \cup St_4 \cup St_1$ is drawn below as an example. Note that St_1 is, by definition, the trivial graph since this is the only tree with 1 vertex and no internal edges.



Drawn below is an example of a proper forest. Observe that there are four connected components, each of which is a proper tree. The internal vertices are shaded grey.

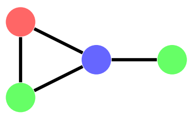
Definition 11. *A* **proper tree** *is a tree T such that for any internal vertex $u \in V(T)$, there is a leaf in $N(u)$. A* **proper forest** *is a union of one or more proper trees.*



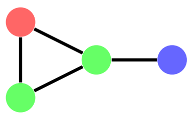
So far, we have colored vertices to showcase certain properties of a graph. In fact, vertex coloring is an important area of study in graph theory and underlies all of the work in this thesis.

Definition 12. *A* **coloring** *of a graph G with k colors is a function $\kappa : [k] \to V(G)$. A coloring κ of G with k colors is* **proper** *if for all $uv \in E(G)$, $\kappa(u) \neq \kappa(v)$.*

Note that in this thesis, we use the shorthand $[k] = \{1, 2, \ldots, k\}$ for any $k \geq 0$. Below is a proper coloring of a graph:



The following, however, would not be a proper coloring of the graph since there exists an edge whose endpoints are assigned the same color:

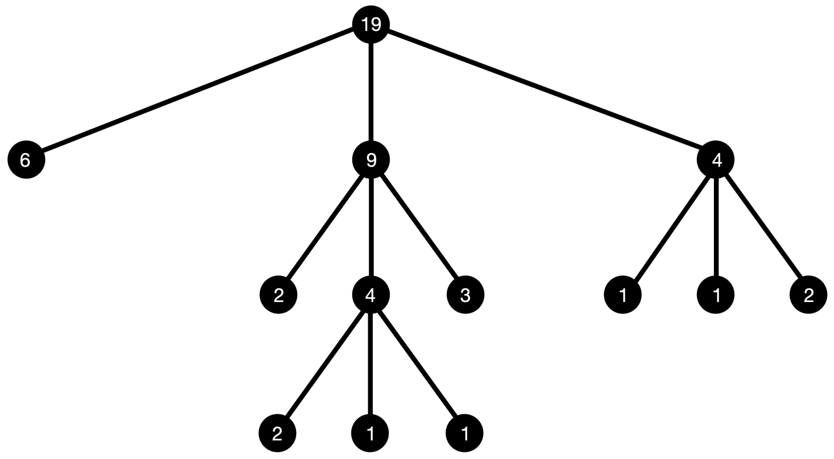


2.2 Rooted Trees

A binary tree, a type of rooted tree wherein each node has at most two children, is a ubiquitous data structure in computer science. In this thesis, we make use of full ternary trees, a relative of the familiar binary tree.

Definition 13. *A* **full ternary tree** *is a rooted tree, wherein every node has either* 0 *or* 3 *children and stores some* **data**.

Let $\mathcal{T}$ denote the following ternary tree, which will be referenced in the remainder of this section:



Definition 14. *A node in a full ternary tree is a string of Ls (left-steps), Ms (middle-steps), and Rs (right-steps), encoding the path from the root to the node. The root is denoted by ϵ.*

In the ternary tree above, the vertex storing data 3 is MR.

Definition 15. *Given two vertices $\mathtt{u}, \mathtt{v}$ in $\mathcal{T}$, we say $(\mathtt{u}, \mathtt{v})$ is an edge in $\mathcal{T}$ if $\mathtt{u}L = \mathtt{v}$, $\mathtt{u}M = \mathtt{v}$, or $\mathtt{u}R = \mathtt{v}$.*

In $\mathcal{T}$, (M, MR) and (ϵ, L) are edges. We can generalize the notion of edges by considering paths between vertices.

Definition 16. *Given two vertices* $\mathtt{u}, \mathtt{v}$ *in* $\mathcal{T}$*, if* $\mathtt{uw} = \mathtt{v}$ *for some string* $\mathtt{w}$ *of* L*s,* M*s, and* R*s, we say there is a path from* $\mathtt{u}$ *to* $\mathtt{v}$ *in* $\mathcal{T}$*, namely* $\mathtt{w}$*.*

In $\mathcal{T}$ above, ML is the path from M to MML.

The relevance of this discussion will be made clear in Section 3, when we describe the Star-Expansion algorithm, which constructs full ternary trees, wherein the data being stored are unlabeled forests.

2.3 Symmetric Functions

In this thesis, we study the chromatic symmetric functions of graphs in the star basis. In this section, we define symmetric functions, including the chromatic symmetric function of a graph. Before this, we define integer partitions, which function as indices of symmetric functions.

Definition 17. *Given a positive integer* n*, a* **partition** λ *of* n *is a nonincreasing list of positive integers, called* **parts***, that sum to* n*. If* $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_k)$ *is a partition of* n*, we write* $\lambda \vdash n$ *and define the* **length** *of* λ*, denoted* $\ell(\lambda)$*, to be the number of parts of* λ*. Furthermore, we define* $p(n)$ *to be the number of partitions of* n*.*

For example, $(4, 3, 3, 1)$ and $(3, 3, 3, 1, 1)$ are partitions of 11. As shorthand, we can condense repeated parts, equivalently expressing these partitions as $(4, 3^2, 1)$ and $(3^3, 1^2)$, respectively.

Definition 18. *For any* n*, the hook partitions of* n *are* $\{(m, 1^{n-m}) : m \in [n-1]\}$*.*

For example, $(7, 1)$ and $(5, 1^3)$ are hook partitions of 8, while $(4, 2, 1, 1, 1, 1)$ is not a hook partition. We will now proceed with our discussion of functions.

Definition 19. *A polynomial* $f(x_1, x_2, \dots, x_n)$ *in which every term has total degree exactly* k *is* **homogeneous** *of degree* k*.*

The polynomial $x_1x_3^3 + x_2^2x_4^2$, for example, is homogeneous of degree 4.

Definition 20. *Let $X = \{x_j\}_{j=1}^{\infty}$ be a set of variables and f be a function of X. We say f is a* **symmetric function** *in X if, for all $n \geq 1$ and all permutations $\sigma : [n] \to [n]$, we have $f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots) = f(x_1, x_2, \ldots)$.*

For example, consider the function

$$f(x_1, x_2, \ldots) = \sum_{i \geq 1} x_i^2 = x_1^2 + x_2^2 + x_3^2 + \cdots$$

We observe that for any $n \geq 1$ and any permutation σ of $[n]$, $f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots) = f(x_1, x_2, \ldots)$ and so f is a symmetric function.

Definition 21. *We write Λ_k to denote the set of all symmetric polynomials in $x_1, x_2, x_3, \ldots$ which are homogeneous of degree k.*

Λ_k is a vector space with many bases. One such basis is the power-sum basis, which we now introduce.

Definition 22. *Let $p_k = \sum_{j=1}^{\infty} x_j^k$. The* **power sum symmetric function** *p_λ indexed by a partition λ is defined as follows:*

$$p_\lambda = \prod_{j=1}^{\ell(\lambda)} p_{\lambda_j}$$

Theorem 23. *For all $k \geq 1$, the set $\{p_\lambda | \lambda \vdash k\}$ is a basis for Λ_k. We call this basis the power-sum basis.*

Now, the chromatic symmetric function is a symmetric function that is generated from proper colorings of graphs.

Definition 24. *The chromatic symmetric function of a graph G, written $\mathbf{X}_G$ is defined over a set of variables $X = \{x_j\}_{j=1}^{\infty}$ as follows:*

$$\mathbf{X}_G = \sum_{\kappa} \prod_{v \in V} x_{\kappa(v)},$$

where the sum is over all proper colorings of G, with any number of colors.

Observe that for every proper coloring κ of G, a term with coefficient 1 is added to $\mathbf{X}_G$. The following remark follows:

Remark. $\mathbf{X}_G(x_1 = 1, x_2 = 1, \ldots, x_k = 1, 0, 0, 0 \ldots) = \chi_G(k)$, *the number of proper colorings of G using k colors.*

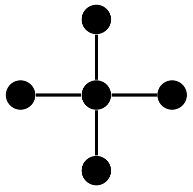
We can better describe and understand symmetric functions by expanding them in different bases. In [St1], Stanley describes how to write $\mathbf{X}_G$ in terms of the power-sum basis:

Theorem 25.

$$\mathbf{X}_G = \sum_{S \subseteq E(G)} (-1)^{|S|} p_{\pi(S)},$$

where $\pi(S)$ is the partition of $|V(G)|$ consisting of the number of vertices in each connected component of $(V(G), S)$ as parts.

We can use Theorem 25 to expand $\mathbf{X}_G$ in the power-sum basis for small graphs. For example consider $G = St_5$, the star graph on 5 vertices drawn below.



Note that if $0 \leq k \leq 4$, for any k-element subset of $E(St_5)$, $p_{\pi(S)} = (5-k, 1^k)$. From this observation, we have the following chain of equalities:

$$\begin{aligned}
\mathbf{X}_{St_5} &= \sum_{S \subseteq E(St_5)} (-1)^{|S|} p_{\pi(S)} \\
&= (-1)^0 p_{(5)} + \binom{4}{1}(-1)^1 p_{(4,1)} + \binom{4}{2}(-1)^2 p_{(3,1^2)} + \binom{4}{3}(-1)^3 p_{(2,1^3)} + (-1)^4 p_{(1^5)} \\
&= p_5 - 4p_{(4,1)} + 6p_{(3,1^2)} - 4p_{(2,1^3)} + p_{(1^5)}
\end{aligned}$$

In fact, in [CvW], Cho writes a general formula for $\mathbf{X}_{St_n}$ in terms of the power-sum basis:

Theorem 26. *If St_n is the star graph on n vertices, then*

$$\mathbf{X}_{St_n} = \sum_{r=0}^{n-1} \binom{n-1}{r} p_{(r+1, 1^{(n-1)-r})}$$

Theorem 27. *Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$. For any $k \geq 1$, define $\mathfrak{st}_k = \mathbf{X}_{St_k}$. Then, define $\mathfrak{st}_\lambda = \mathfrak{st}_{\lambda_1}\mathfrak{st}_{\lambda_2} \cdots \mathfrak{st}_{\lambda_k}$. For all $k \geq 1$, the set $\{\mathfrak{st}_\lambda | \lambda \vdash k\}$ is a basis for Λ_k. We call this basis the star basis.*

In the following chapters, we will use the star basis to expand the chromatic symmetric function of trees and discover properties that can be determined from the coefficients. By this approach, we can conclude that certain trees are determined by their chromatic symmetric functions. As we can predict from the small example for St_5, for very large trees it will become very burdensome to compute the chromatic symmetric by hand. That said, in the next chapter we will study a novel approach introduced in [AMOZ] to compute the chromatic symmetric function of a graph in the star-basis.

Chapter 3

Deletion-Near-Contraction

In this chapter, we will study the deletion-near-contraction relation as introduced by Aliste-Prieto, de Mier, Orellana, and Zamora. This powerful relation serves as a new tool to compute the chromatic symmetric function of trees in the star basis efficiently. First, we describe this relation and then we apply it to compute chromatic symmetric functions. Finally, we determine some coefficients of the chromatic symmetric function of a tree by using inferences about how deletion-near-contraction is applied in the derivation of the CSF.

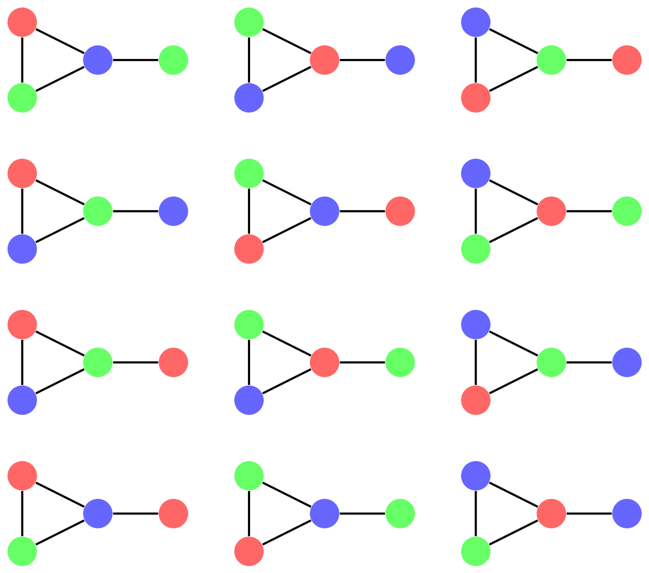
3.1 DNC Relation

The chromatic symmetric function of a graph is tied to its chromatic polynomial. We will start this section by reviewing properties of the chromatic polynomial of a graph and then considering the deletion-near-contraction relation in parallel.

Definition 28. *The* **chromatic polynomial** *of a graph G is a function $\chi_G : \mathbb{N} \cup \{0\} \to \mathbb{N} \cup \{0\}$ such that $\chi_G(k)$ is the number of proper colorings of G using k colors.*

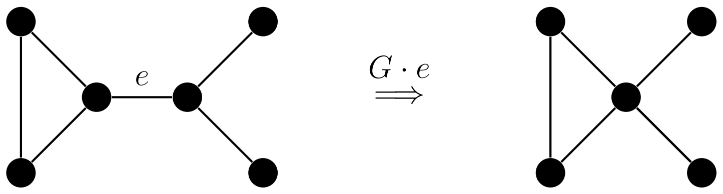
For example, consider the graph G drawn below with chromatic polynomial

$\chi_G(k) = k(k-1)^2(k-2)$.



$\chi_G(2) = 0$, so G cannot be properly colored with just 2 colors. $\chi_G(3) = 12$, so there are 12 ways to properly color the vertices of G with 3 colors, and so on.

Definition 29. *In a graph G,* **contraction of edge** $e = uv$ *is the replacement of u and v with a single vertex whose incident edges are the edges other than e that were incident to u or v. The resulting graph $G \cdot e$ has one less edge than G.*

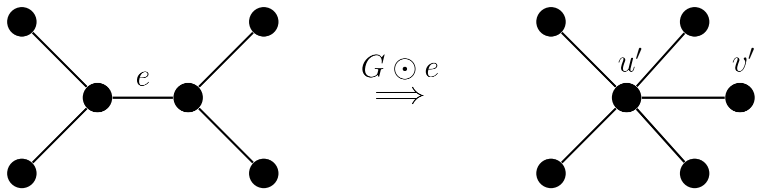


Theorem 30. *For any graph G with edge e, $\chi_G(k) = \chi_{G \setminus e}(k) - \chi_{G/e}(k)$.*

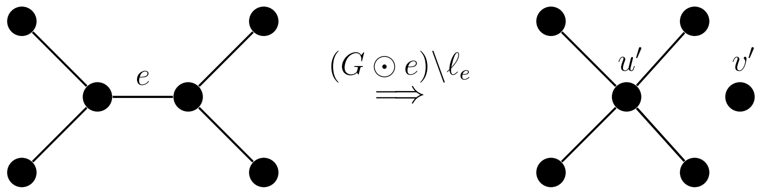
Let's apply this to G above. By a simple counting argument, we observe that $\chi_{G\setminus e}(k) = k(k-1)(k-2) \cdot k(k-1)^2$ and $\chi_{G \cdot e}(k) = k(k-1)^3(k-2)$. Thus, by Theorem 30, $\chi_G(k) = k^2(k-1)^3(k-2) - k(k-1)^3(k-2) = k(k-1)^4(k-2)$.

This relation can't possibly hold for the chromatic symmetric function since the CSF is homogeneous of degree $|V(G)|$ and performing contraction decreases the number of vertices. That said, in [AMOZ] a modified reccurence relation is proven, which we now introduce.

Definition 31. *In a graph G, near-contraction of edge $e = uv$ is the contraction of e, where u and v are replaced with u', followed by the insertion of a vertex v' and an edge $u'v'$. The resulting graph is denoted $G \odot e$.*



Definition 32. *In a graph G, dot-contraction of edge $e = uv$ is the near-contraction of e, followed by deletion of the introduced edge $u'v'$. The resulting graph is denoted $(G \odot e)\backslash \ell_e$.*



Lemma 1. *For any simple graph G having internal edge e, $G \odot e$ and $(G \odot e)\backslash \ell_e$ each has exactly one fewer internal edge than G.*

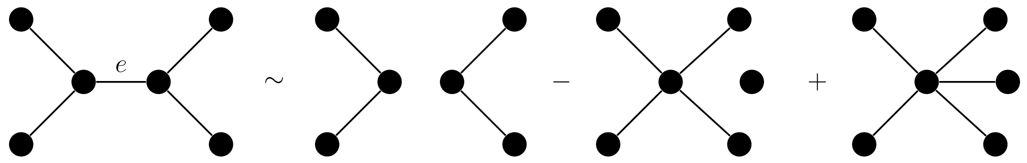
Proof. Assume we label the vertices of G and apply leaf-contraction to an internal edge uv, replacing u and v with u'. Then, every internal edge of G of the form ux appears as $u'x$ in $G \odot e$, and every internal edge of G of the form vx appears as $u'x$ in $G \odot e$. Every other internal edge in G, i.e. every internal edge not incident to e, remains an internal edge in $G \odot e$. Leaf-contraction does not increase the number of internal edges and so the edge uv in G does not map to any internal edge in $G \odot e$. By a symmetric argument for dot-contraction, the lemma holds. □

We are now ready to see the deletion-near-contraction relation. From here forward, let $\mathbf{X}_G$ denote the chromatic symmetric function of a simple graph G in the star-basis.

Theorem 33. *For any simple graph G with edge e,*

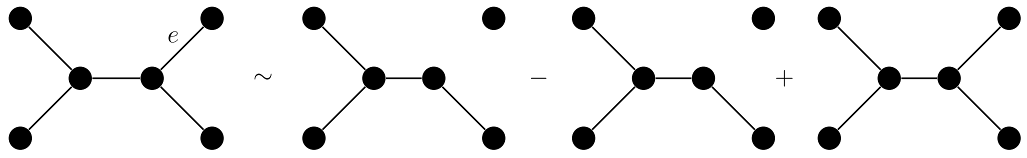
$$\mathbf{X}_G = \mathbf{X}_{G\backslash e} - \mathbf{X}_{(G\odot e)\backslash \ell_e} + \mathbf{X}_{G\odot e}$$

Letting G be the graph in the above example, we can write:



If we replace each graph separated by $+$ or $-$, then $\sim$ can be rewritten as an equality. That is, the chromatic symmetric function of the graph on the left-hand side can be expressed in terms of the chromatic symmetric functions of the three graphs on the right-hand side. Note that all of the graphs on the right-hand side are star graphs and so we have successfully expressed $\mathbf{X}_G$ in terms of star graphs. That is, $\mathbf{X}_G = \mathfrak{st}_{(3,3)} - \mathfrak{st}_{(5,1)} + \mathfrak{st}_{(6)}$.

Suppose, instead, we tried to apply the relation to a leaf edge:



As depicted in this picture, performing deletion-near-contraction on a leaf-edge gives us no information. In the next section, we will see how to apply the relation more generally to determine $\mathbf{X}_G$.

3.2 Star-Expansion and Path-Counting

Toward the end of the previous section, we saw an example of using the deletion-near-contraction (DNC) relation to express the chromatic symmetric function (CSF) of a graph in the star basis. In that example, the cardinality of the internal edge set $|\mathcal{I}(G)| = 1$ and so (as long as we operate on this internal edge), the derivation of $\mathbf{X}_G$ is very simple. More generally, given a graph G, we apply a variant of the star-expansion algorithm discussed in [AMOZ] to produce a ternary tree from which we compute $\mathbf{X}_T$.

Definition 34. *Given a graph G, a* **DNC-tree** *for $\mathbf{X}_G$ is the output of a Star-Expansion algorithm.*

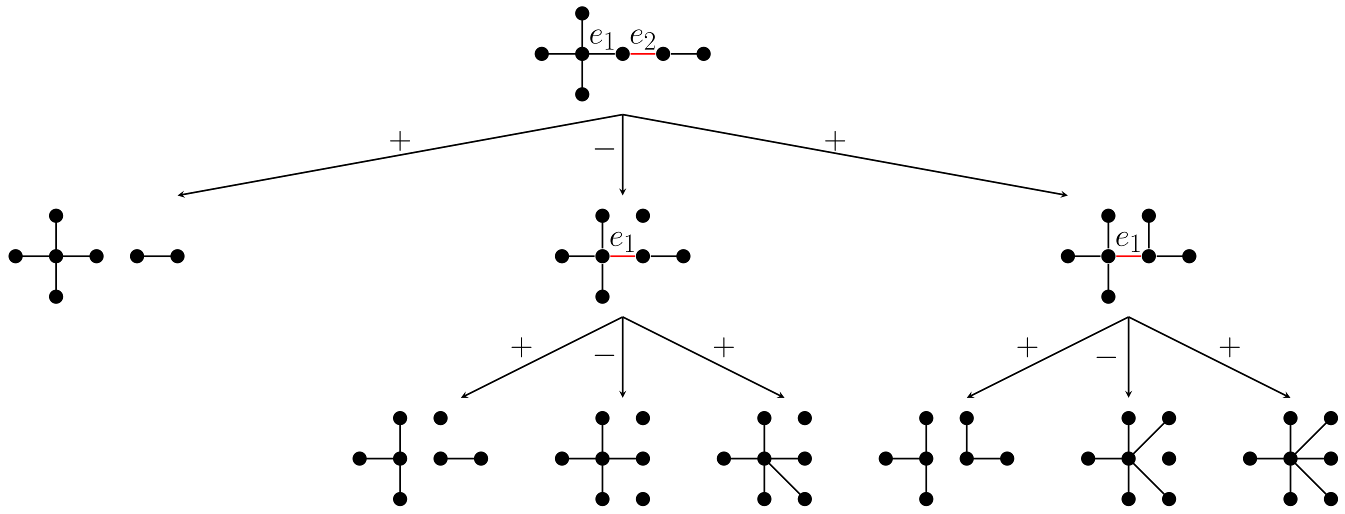
Algorithm 1: Star-Expansion (Iterative)

```
input  : a forest $F$, a permutation $\sigma$ of $\mathcal{I}(F)$
output: visualization of a ternary tree $\mathcal{D}$ for $\mathbf{X}_F$ with leaves labeled by star
        forests
draw a ternary tree with root $F$ and no children.
for $e \in \sigma$ (in order) do
    for each leaf node $F'$ in $\mathcal{D}$ do
        if $e \in E(F')$ then
            draw $F'$'s left-child $F' \backslash e$, inserting $+$ along the edge
            draw $F'$'s middle-child $(F' \odot e) \backslash \ell_e$, inserting $-$ along the edge
            draw $F'$'s right-child $F' \odot e$, inserting $+$ along the edge
    end
end
```

As written, these algorithms require a permutation of the set of internal edges of the input forest. That said, as described in [AMOZ], one could avoid this by searching for an arbitrary internal edge at every step. On the following page is an example of a ternary tree that is returned by the above algorithms with input G and permutation (e_2, e_1).

Algorithm 2: Star-Expansion (Recursive)

```
input  : a forest $F$, a permutation $\sigma$ of $\mathcal{I}(F)$
output: a ternary tree $\mathcal{D}$ for $\mathbf{X}_F$ with leaves labeled by star forests
if $\sigma$ is empty then
    return $(F, \lambda, \lambda, \lambda)$
else
    let $\sigma = e_1, e_2, \ldots, e_{|\mathcal{I}(F)|}$
    let $i$ be smallest int s.t. $e_i \in E(F)$.
    let $j > i$ be smallest int s.t. $e_j \in E(F \backslash e_i)$.
    $\mathcal{D}_1 \leftarrow$ Star-Expansion($F \backslash e_i, (e_j, \ldots e_{|\mathcal{I}(F)|})$)
    $\mathcal{D}_2 \leftarrow$ Star-Expansion($(F \odot e_i) \backslash \ell_{e_i}, (e_{i+1}, \ldots, e_{|\mathcal{I}(F)|})$)
    $\mathcal{D}_3 \leftarrow$ Star-Expansion($F \odot e_i, (e_{i+1}, \ldots, e_{|\mathcal{I}(F)|})$)
    return $(F, \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$
```



$$\mathbf{X}_G = -\mathfrak{st}_{(4,2,1)} + \mathfrak{st}_{(4,3)} + \mathfrak{st}_{(5,1,1)} + \mathfrak{st}_{(5,2)} - 2\mathfrak{st}_{(6,1)} + \mathfrak{st}_{(7)}$$

The vertices of $\mathcal{D}$ in the example abovee are $L, ML, MM, MR, RL, RM, RR$. The corresponding data stored at each of these vertices is $St_5 \cup St_2$, $St_4 \cup St_2 \cup St_1$, $St_5 \cup St_1 \cup St_1$, and so on, respectively.

Definition 35. *Given a star forest F, define $\lambda(F)$ to be the partition of $|V(F)|$ whose parts are the orders of the connected components of F. Given a partition*

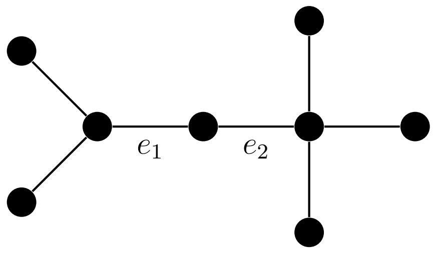
$\lambda = (\lambda_1, \ldots, \lambda_k)$, *define* $F(\lambda)$ *to be the star forest* $St_{\lambda_1} \cup \cdots \cup St_{\lambda_k}$.

Given $\mathcal{D}$ above, we compute $\mathbf{X}_G$ by taking the sum of $\mathfrak{st}_{\lambda(F)}$ over each label of each leaf of $\mathcal{D}$. $\mathbf{X}_G$ is given right below $\mathcal{D}$.

Theorem 36. *There are no cancellations in the star-expansion of the chromatic symmetric function of a simple graph.*

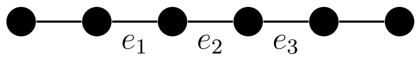
The above fact implies that the coefficients store a lot of information about the input tree. In particular, we have that for any forest F be with DNC-tree $\mathcal{D}$ and $\mathbf{X}_F = \sum_{\lambda \vdash |V(F)|} c_\lambda \mathfrak{st}_\lambda$, for any $\lambda \vdash |V(F)|$, $|c_\lambda|$ is the number of paths from the root F to vertices labeled $F(\lambda)$. Furthermore, $\operatorname{sign}(c_\lambda) = (-1)^{m_1}$, where x is the multiplicity of 1 in λ.

The order that internal edges are considered may greatly impact the specific DNC-tree obtained, as demonstrated in the following two examples. This fact will become relevant as we begin to count paths in DNC-trees. Consider the following tree T_1:

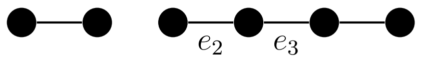


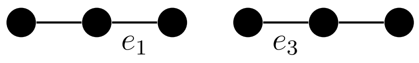
When producing the DNC-tree for T_1, $\mathcal{D}$, if edges are operated in the order (e_1, e_2), then the left-child of the root will be $St_3 \cup St_5$. If edges are instead operated in the order (e_2, e_1), the the left-child of the root will be $St_4 \cup St_4$. Since the algorithm in [AMOZ] has no restriction on the order of internal edges considered, both DNC-trees are correct in the sense that their leaves will yield a correct star-expansion of T_1. That said, we should be aware that vertex labels of $\mathcal{D}$ will differ in each case.

Consider the following tree T_2:



In the DNC-tree for $\mathbf{X}_T$, the graph below on the left is the left-child of the root if the first internal edge considered is e_1. The graph on the right is the left-child of the root if the first internal edge is e_2.





Note that the graph on the left has an internal edge while the graph on the right does not; thus, the DNC-trees in each of these cases will be structurally different (beyond just different edge labels). Hence, when arguing about paths in DNC-trees, we need to be careful about the ordering of internal edges.

Chapter 4

Leading Term

The discussion of lexicographic ordering in Section 2.3 equips us to consider the "leading term" of a chromatic symmetric function. In particular, we will show that the leading term of $\mathbf{X}_T$ is completely determined by characteristics of T. At the end of this chapter, we will see how this enables us to distinguish classes of trees.

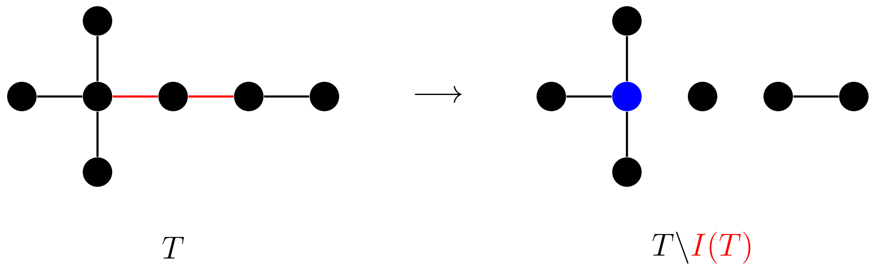
Definition 1. *Let F be a forest on n vertices. The leading partition of $\mathbf{X}_F = \sum_{\lambda \vdash n} c_\lambda \mathfrak{st}_\lambda$ is the lexicographically smallest partition λ such that $c_\lambda \neq 0$. The leading coefficient is c_λ and the leading term is $c_\lambda \mathfrak{st}_\lambda$.*

Recall that a star forest $\tilde{F}$ is a forest with no internal edges, and $\lambda(\tilde{F})$ is the partition obtained by listing the sizes of the connected components in nonincreasing order. We refer to the connected components of $F \backslash I(F)$ as the **leaf components** of F, and we denote by $\lambda_{\text{lead}}(F)$ the partition obtained by listing the orders of the leaf components of $F \backslash I(F)$ in nonincreasing order:

$$\lambda_{\text{lead}}(F) := \lambda(F \backslash I(F)).$$

For example, consider the following forest F with leaf components St_4, St_2, St_1

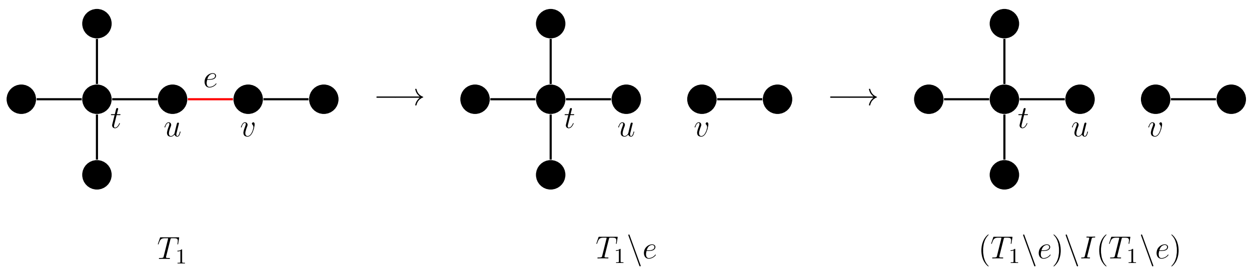
and $\lambda_{\text{lead}}(F) = (4, 2, 1)$:



We will soon relate $\lambda_{\text{lead}}(F)$ to the leading partition of $\mathbf{X}_F$. First, we will study the relationship between the deletion-near-contraction relations and $\lambda_{\text{lead}}(F)$ by considering how deletion, dot-contraction, and leaf-contraction affect leaf components.

4.1 DNC operations and λ_{lead}

We will consider six cases to fully characterize how the DNC operations change λ_{lead}. First, consider T_1 drawn below.



Note that $\lambda_{\text{lead}}(T_1) = (4, 2, 1) < \lambda_{\text{lead}}(T_1 \backslash e) = (5, 2)$. For this example, deleting e decreases the number of singleton leaf components by 1 and increases the order of the leaf component containing u's neighbor t by 1. We formalize this observation with the following lemma:

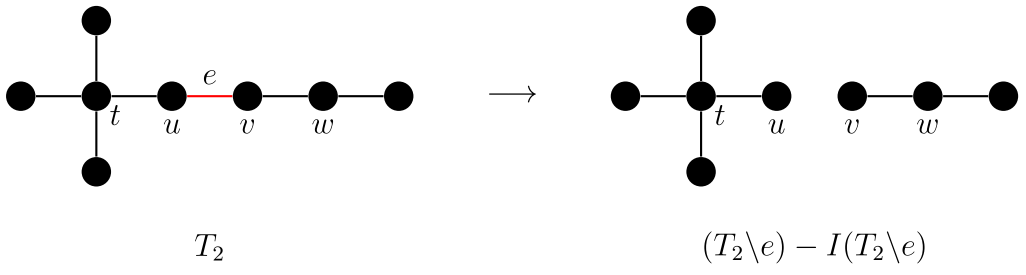
Lemma 2. *Suppose T has an internal edge $e = uv$, where u is a degree-2 deep vertex with $N(u) = \{t, v\}$ and v is not a degree-2 deep vertex. If $\lambda_{\text{lead}}(T) = (k_1, \ldots, k_i, \ldots, k_m)$,* *where k_i is the order of the leaf component that t belongs to, then* $\lambda_{lead}(T\backslash e) = \operatorname{sort}(k_1, \ldots, k_{i-1}, k_i + 1, k_{i+1}, \ldots, k_{m-1})$.

Proof. Recall that, by definition, the leaf components of T are the connected components of $T\backslash I(T)$ having orders $k_1 \geq \cdots \geq k_m$. Note that $k_m = 1$ since u is a connected component in $T\backslash I(T)$. In $T\backslash e$, the leaf component that t belongs to has order $k_i + 1$ since there is an additional leaf u in $N(t)$. Since u is not a leaf component in $T\backslash e$, k_m does not appear in $\lambda_{\text{lead}}(T\backslash e)$.

In T, v is an internal vertex that is not a degree-2 deep vertex, so it has a leaf in its neighborhood or has degree at least 3. Either way, the order of the leaf component that v belongs to doesn't change in $T\backslash e$ since v remains an internal vertex with the same leaves adjacent.

All other leaf components have the same order in T as in $T\backslash e$. The lemma follows. $\square$

Now, consider T_2 drawn below with $\lambda_{\text{lead}}(T_2) = (4, 2, 1, 1)$. Observe that $\lambda_{\text{lead}}(T_2\backslash e) = (5, 3)$.



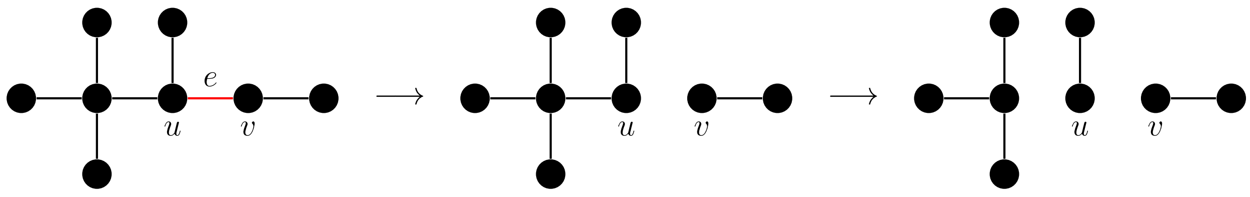
We observe that in this example, deleting e decreases the number of 1s in the leading partition by 2 and increases the parts of $\lambda_{\text{lead}}(F)$ corresponding to the leaf components of neighboring vertices by 1. This is formalized in Lemma 3 below:

Lemma 3. *Suppose T has an internal edge $e = uv$, where u and v are degree-2 deep vertices with $N(u) = \{t, v\}$ and $N(v) = \{u, w\}$. If $\lambda_{lead}(T) = (k_1, \ldots, k_i, \ldots, k_j, \ldots, k_m)$, where k_i and k_j are the orders of the leaf components that t and w belong to, then $\lambda_{lead}(T \backslash e) = \text{sort}(k_1, \ldots, k_{i-1}, k_i + 1, k_{i+1}, \ldots, k_{j-1}, k_j + 1, k_{j+1}, \ldots, k_{m-2})$.*

Proof. The leaf components of T have orders $k_1 \geq \cdots \geq k_m$. Note that $k_m = k_{m-1} = 1$ since u and v are each connected components in $T \backslash I(T)$. If in T the leaf components that t and w belong to have orders k_i and k_j, respectively, then in $T \backslash e$, the leaf components that t and w belong to have orders $k_i + 1$ and $k_j + 1$, respectively, since u and v are now leaves. Since u and v are not leaf components in $T \backslash e$, k_m and k_{m-1} do not appear in $\lambda_{\text{lead}}(T \backslash e)$.

All other leaf components have the same order in T as in $T \backslash e$. □

We have considered the cases when the deleted internal edge has one or two endpoints that are degree-2 deep vertices. Now, consider the final case for T_3 below:



T_3 $\quad$ $T_3 \backslash e$ $\quad$ $(T_3 \backslash e) \backslash I(T_3 \backslash e)$

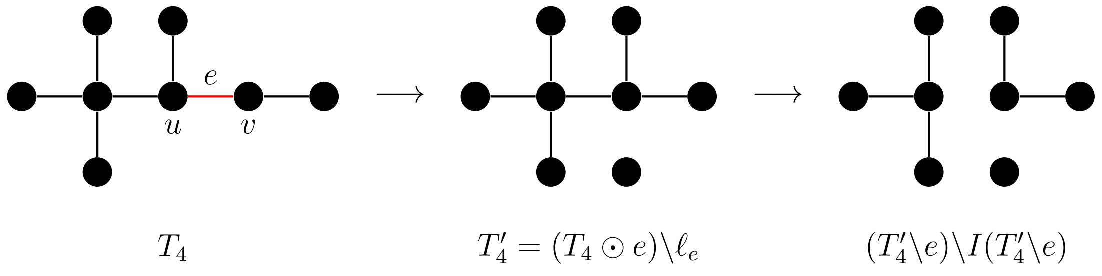
We observe that in this example, deleting e does not change the leading partition at all, guiding us to the following lemma.

Lemma 4. *Let $e = uv$ be an internal edge in T such that neither u nor v is a degree-2 deep vertex. Then, $\lambda_{lead}(T) = \lambda_{lead}(T \backslash e)$.*

Proof. The leaf components of T have orders $k_1 \geq \cdots \geq k_m$. The leaf components that u and v belong to have the same order in T as in $T \backslash e$ since these vertices remain internal with the same leaves adjacent.

All other leaf components have the same order in T as in $T \backslash e$. □

We have fully specified how λ_{lead} is affected by deletion of an internal edge. Now we consider the dot-contraction operation. First, think about the following process:


e
u
v
$\longrightarrow$
$\longrightarrow$
T_4
$T_4' = (T_4 \odot e) \backslash \ell_e$
$(T_4' \backslash e) \backslash I(T_4' \backslash e)$

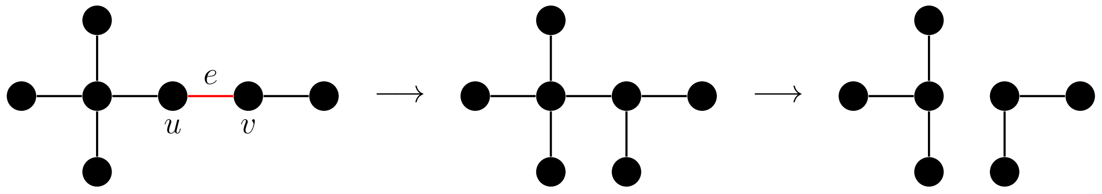
In this example, $\lambda_{\text{lead}}(T_4) = (4, 2, 2)$, and $\lambda_{\text{lead}}((T_4 \odot e) \backslash \ell_e) = (4, 3, 1)$. Thus, performing dot-contraction on e in T_4 appears to merge two leaf components and add a singleton leaf component. We now formalize and generalize this notion.

Lemma 5. *Suppose F has an internal edge $e = uv$. If $\lambda_{lead}(F) = (k_1, \ldots, k_i, \ldots, k_j, \ldots, k_m)$, where k_i and k_j are the orders of the leaf components that u and v belong to, then $\lambda_{lead}((F \odot e) \backslash \ell_e) = \text{sort}(k_1, \ldots, k_{i-1}, k_{i+1}, k_{i+2}, \ldots, k_{j-1}, k_{j+1}, k_{j+2}, \ldots, k_m, k_i + k_j - 1, 1)$.*

Proof. The leaf components of F have orders $k_1, \ldots, k_m$. After dot-contraction of e, the adjacent leaf components merge into a single leaf component of order $k_i + k_j - 1$, following by definition of the dot-contraction operation. A singleton is also produced, adding a 1 to λ_{lead}.

All other leaf components have the same order in T as in $(T \odot e) \backslash \ell_e$ and so the claim follows. □

Finally, consider the following example for leaf-contraction:



T_5 $\qquad$ $T_5' = T_5 \odot e$ $\qquad$ $(T_5' \backslash e) \backslash I(T_5' \backslash e)$

Again, it appears two leaf components merge in the process of dot-contraction, which we formalize now.

Lemma 6. *Suppose F has an internal edge $e = uv$. If $\lambda_{lead}(F) = (k_1, \ldots, k_i, \ldots, k_j, \ldots, k_m)$, where k_i and k_j are the orders of leaf components that u and v belong to, then $\lambda_{lead}(F \odot e) = \mathrm{sort}(k_1, \ldots, k_{i-1}, k_{i+1}, k_{i+2}, \ldots k_{j-1}, k_{j+1}, k_{j+2}, \ldots, k_m, k_i + k_j)$.*

Proof. By definition, leaf-contraction of e merges the two components into a single component of order $k_i + k_j$, and all other leaf components stay the same. Hence, the lemma follows. □

We have reinterpreted the deletion-near-contraction operations in terms of how they change leaf components of a forest. Now, we summarize and modify these results to make them useful for a future theorem.

Theorem 37. *For any internal edge $e = uv$ in a forest F:*

1. *If either u or v is a degree-2 deep vertex, then $\lambda_{\text{lead}}(F) < \lambda_{\text{lead}}(F \backslash e)$.*

2. *If neither u nor v is a degree-2 deep vertex, then $\lambda_{\text{lead}}(F) = \lambda_{\text{lead}}(F \backslash e)$.*

3. *If both u and v have leaves in their neighborhoods, then $\lambda_{\text{lead}}(F) < \lambda_{\text{lead}}((F \odot e) \backslash \ell_e)$.*

4. *If either u or v has no leaf in its neighborhood, then $\lambda_{\text{lead}}(F) = \lambda_{\text{lead}}((F \odot e) \backslash \ell_e)$.*

5. $\lambda_{\text{lead}}(F) < \lambda_{\text{lead}}(F \odot e)$.

Proof. 1 and 2 follow immediately from the arguments in Lemmas 4.1-4 above.

For 3, observe that if u and v both have leaves in their neighborhoods and belong to leaf components of orders k_i and k_j, then $k_i, k_j \geq 2$ and so the order of the leaf component containing the contracted vertex will be $k_i + k_j - 1 \geq 3$. Since $k_i + k_j - 1 > k_i, k_j$ it follows that $\lambda_{\text{lead}}((F \odot e)\backslash \ell_e) > \lambda_{\text{lead}}(F)$.

On the other hand, for 4, suppose (WLOG) that u has no leaf in its neighborhood (so it belongs to a leaf component of order 1), and suppose v belongs to a leaf component of order k_j. Then, $k_i + k_j - 1 = k_j$ and so it follows that $\lambda_{\text{lead}}((F \odot e)\backslash \ell_e) = \lambda_{\text{lead}}(F)$.

For 5, if u and v belong to leaf components of orders k_i and k_j, then the order of the leaf component containing the contracted vertex will be $k_i + k_j > k_i, k_j$ since $k_i, k_j \geq 1$. It follows that $\lambda_{\text{lead}}(F \odot e) > \lambda_{\text{lead}}(F)$. □

4.2 The Leading Term and λ_{lead}

We observe trivially that for any edge $e = uv$, exactly one of the premises in Theorem 37.1 and Theorem 37.2 holds, and exactly one of the premises in Theorem 37.3 and Theorem 37.4 holds. That is, for any internal edge e, $\lambda_{\text{lead}}(F) \leq \lambda_{\text{lead}}(F\backslash e), \lambda_{\text{lead}}((F \odot e)\backslash \ell_e), \lambda_{\text{lead}}(F \odot e)$. The following lemma holds immediately.

Lemma 7. *Let λ denote the leading partition of $\mathbf{X}_F$. Then, $\lambda \geq \lambda_{\text{lead}}(F)$.*

Proof. In any DNC-tree, every leaf node will consist of star graphs $\tilde{F}$ for which $\lambda_{\text{lead}}(F) \leq \lambda(\tilde{F})$ since DNC operations do not decrease λ_{lead}. □

We have just established a lower bound on the leading partition of $\mathbf{X}_{\mathrm{F}}$. To conclude that $\lambda_{\mathrm{lead}}(F)$ is, in fact, the leading partition of $\mathbf{X}_{\mathrm{F}}$, we must show that there exists a path in any DNC-tree of F from the root F to the star-forest $F \backslash I(F)$. This follows almost immediately from the following lemma.

Lemma 8. *Any internal edge uv satisfies at least one of the following: (1) neither u nor v is a degree-2 deep vertex, (2) either u or v has no leaf in its neighborhood.*

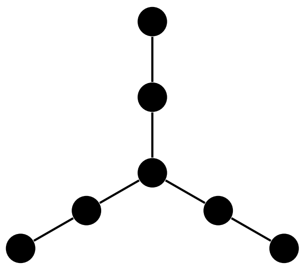
Proof. Let uv be an internal edge in F. That is, $\deg(u), \deg(v) \geq 2$. Assume, to the contrary, and without loss of generality that u is a degree-2 deep vertex and that u and v both have leaves in their neighborhoods. That is, the vertex u is a deep vertex with a leaf in its neighborhood, a contradiction by definition of deep vertex. Hence, we have the lemma. $\square$

As stated previously, the proof of the following theorem follows simply from the previous results.

Theorem 38. *The leading partition of $\mathbf{X}_F$ is $\lambda_{\mathrm{lead}}(F)$.*

Proof. Given a forest F, by Corollary 1 we have for all $\lambda \vdash |V(F)|$ with $\lambda < \lambda_{\mathrm{lead}}(F)$ that $c_\lambda = 0$. On the other hand letting $\mathcal{D}$ denote a fixed DNC-tree of F, we observe that at any level k of $\mathcal{D}$, there exists an edge (corresponding to deletion or dot-contraction) between a forest at level k, F_k, and forest at level $k+1$, F_{k+1}, for which $\lambda_{\mathrm{lead}}(F_k) = \lambda_{\mathrm{lead}}(F_{k+1})$. It follows that there exists a path in $\mathcal{D}$ from F to a star-forest $\tilde{F}$ such that $\lambda_{\mathrm{lead}}(F) = \lambda_{\mathrm{lead}}(\tilde{F}) = \lambda(\tilde{F})$. That is $c_{\lambda_{\mathrm{lead}}(F)} \neq 0$. It follows by Lemma [?] that $\lambda_{\mathrm{lead}}(F)$ is the leading partition of $\mathbf{X}_{\mathrm{F}}$. $\square$

Let T denote the tree below.



By definition, we have that $\lambda_{\text{lead}}(T) = (2,2,2,1)$. By Theorem 35, then, we have that the leading partition of $\mathbf{X}_{\mathrm{T}}$ is $(2,2,2,1)$.

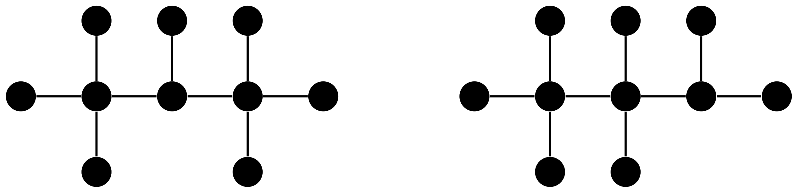
Given any tree T, we can now determine the leading partition of X_T based on properties of the tree T itself. This allows us to distinguish certain trees as explained in the following theorem:

Theorem 39. *If T_1 and T_2 are trees with $T_1 - I(T_1) \neq T_2 - I(T_2)$, then $\mathbf{X}_{\mathrm{T}_1} \neq \mathbf{X}_{\mathrm{T}_2}$. Equivalently, if $\lambda_{lead}(T_1) \neq \lambda_{lead}(T_2)$, then $\mathbf{X}_{\mathrm{T}_1} \neq \mathbf{X}_{\mathrm{T}_2}$.*

Proof. This follows trivially since if two trees have the same chromatic symmetric function then the same terms must appear in each function. In particular, their leading terms must be equal and so their leading partitions must be equal. □

We apply Theorem 39 directly in the following example.

Example 1. *Consider the two trees below. Let T_1 denote the tree on the left and T_2 the tree on the right.*

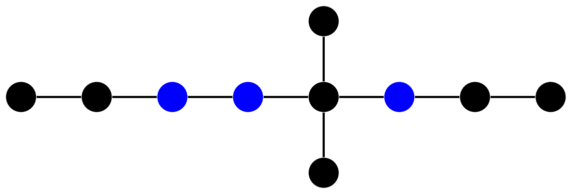


By Theorem 39, we can conclude that $\mathbf{X}_{\mathrm{T}_1} \neq \mathbf{X}_{\mathrm{T}_2}$ since $\lambda_{lead}(T_1) = (4,4,2)$ while $\lambda_{lead}(T_2) = (4,3,3)$.

Corollary 1. *For any nontrivial tree T, the multiplicity of 1 in $\lambda_{lead}(T)$ is the number of deep vertices of T.*

Proof. By definition, $\lambda_{\text{lead}}(T) = \lambda(T \backslash I(T))$. Since T is nontrivial, every singleton component in $T \backslash I(T)$ must arise after deletion of all edges incident to an internal vertex in T and all of these edges are internal. That is, there is a natural correspondence between 1's in $\lambda_{\text{lead}}(T)$ and deep vertices since these vertices are exactly those that are surrounded by internal edges in T. □

Example 2. *In the tree below, the three deep vertices are colored blue.*



Observe that the leading partition of this tree is $(3, 2, 2, 1, 1, 1)$. As expected, the number of 1's equals the number of deep vertices in T.

Corollary 2. *For any nontrivial tree T, the multiplicity of 1 in $\lambda_{lead}(T)$ is 0 if and only if T is a proper tree.*

Proof. Let T be a nontrivial tree. If the multiplicity of 1 in $\lambda_{\text{lead}}(T) = 0$, then there are no deep vertices in T. By a previous result, this is equivalent to the tree being proper. If T is a nontrivial proper tree, then every component in $T \backslash I(T)$ has order at least 2. □

Definition 2. *A bi-star is a tree consisting of two star graphs separated by an internal edge. An extended bi-star is a tree consisting of two star graphs separated by two or more internal edges.*

Corollary 3. *For some tree T, $\lambda_{lead}(T) = (i, j, 1^{|V(T)|-i-j})$ for some $i, j > 1$ if and only if T is a bi-star or extended bi-star with leaf-stars St_i and St_j separated by $|V(T)| - i - j$ internal vertices.*

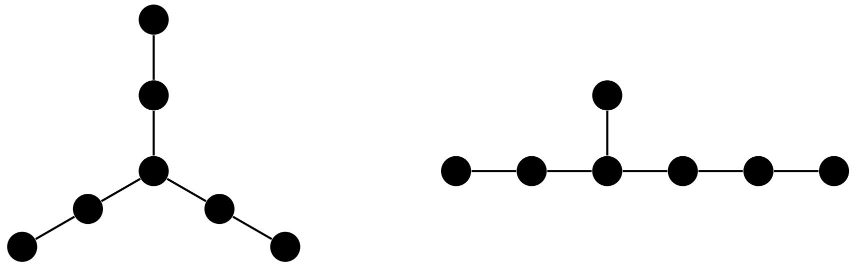
Proof. Let T be a tree with $\lambda_{\text{lead}}(T) = (i, j, 1^{|V(T)-i-j})$ for some $i, j > 1$. It follows that the connected components of $T \backslash I(T)$ are St_i, St_j, and $|V(T)|-i-j$ components of order 1. Since these order-1 components cannot be leaves of T and deep vertices, it follows that St_i and St_j are at the ends of T, and the St_1 graphs are connected by a path in between these larger star graphs. The converse follows simply by an application of the previous theorem. □

Corollary 4. *Bi-stars and extended bi-stars are distinguished by their chromatic symmetric functions.*

Proof. As shown in the previous result, given a tree T, we can determine whether it's a bi-star/extended bi-star based on its leading partition. Using the values of the leading partition, we can fully reconstruct T. □

A natural direction to follow in the study of Stanley's conjecture is to try and distinguish trees T_1, T_2 satisfying $T_1 - I(T_1) = T_2 - I(T_2)$. To this end, we will strengthen our result about the leading partition. In particular, we will show that the leading coefficient is fully determined by properties of the input tree thus allowing us to distinguish more types of trees.

Example 3. *Consider the two trees drawn below.*



Based on our work so far, we cannot conclude that these two trees have distinct chromatic symmetric functions without computing the respective CSFs since the leading partitions of both trees are the same. We will now prove a result that allows us to conclude that these two trees do, in fact, have differents CSFs.

Recall from the discussion in Chapter 3 for any $\lambda \vdash n$ that $|c_\lambda|$ is equal to the number of paths in a DNC-tree for $\mathbf{X}_{\mathrm{T}}$ from the root T to $\overset{*}{F}(\lambda)$. In particular, $|c_{\lambda_{\mathrm{lead}}(T)}|$ is the number of paths in a DNC-tree for $\mathbf{X}_{\mathrm{T}}$ from T to $\overset{*}{F}(\lambda_{\mathrm{lead}}(T))$.

Lemma 9. *Let $T_1, \ldots, T_k$ be trees, and let F be the forest consisting of these trees. That is, $F = \bigcup_{i=1}^{k} T_i$. Then, $c_{\lambda_{lead}(F)} = \prod_{i=1}^{k} c_{\lambda_{lead}(T_i)}$.*

Proof. (direct)

Let $\mathcal{D}$ be the DNC-tree obtained by operating on all edges in $I(T_1)$, followed by all edges in $I(T_2)$, …, followed by all edges in $I(T_k)$. Since $I(F) = \bigcup_{i=1}^{k} I(T_i)$, observe that $\mathcal{D}$ is, in fact, a DNC-tree for $\mathbf{X}_{\mathrm{F}}$.

Let R denote the set of paths in $\mathcal{D}$ from F to $F \backslash I(F)$. For any $j \in [k]$, let S_j denote the set of paths in $\mathcal{D}$ from $F - \bigcup_{i=1}^{j-1} I(T_j)$ to $F - \bigcup_{i=1}^{j} I(T_j)$. Let $S = S_1 \circ S_2 \circ \cdots \circ S_k$. It follows that $S = T$ and so the lemma holds. □

Lemma 10. *If T is a tree with no deep vertices, then $c_{\lambda_{lead}(T)} = 1$.*

Proof. (direct)

Let T be an arbitrary tree with no deep vertices, and suppose we fix a DNC-tree $\mathcal{D}$ for $\mathbf{X}_{\mathrm{T}}$. We will show that there is exactly one path from T to $T \backslash I(T)$ in $\mathcal{D}$.

Since F has no deep vertices, every internal vertex in F has a leaf in its neighborhood. Let e_i be the i-th internal edge operated on in F. We must begin the path with a deletion of e_1 since any other operation results in a forest with greater leading partition than F. Let $k \geq 1$ be arbitrary and assume we have deleted $e_1, \ldots, e_k$. Since deleting internal edges does not remove leaves from the neighborhoods of internal vertices, it follows that e_{k+1} must be deleted as well. It follows by induction that the only path from F to $F\backslash I(F)$ is the path of repeated deletions. Thus, $|c_{\text{lead}(F)}| = 1$. Since the number of 1s in $\lambda_{\text{lead}}(F)$ is equal to the number of deep vertices in F, $c_{\lambda_{\text{lead}}(T)} = (-1)^0 \cdot 1$. Hence, we have the claim. □

Lemma 11. *If F is a forest with a deep vertex, then there exists a deep vertex in F with at most one deep vertex in its neighborhood.*

Proof. Let F be s forest with a deep vertex and assume, to the contrary, that each vertex in F has at least two deep vertices in its neighborhood. Since F is finite, it follows that F has a cycle consisting of deep vertices, a contradiction. Hence, the Lemma follows. □

Define a predicate $P(m)$, over $\mathbb{Z}$, as follows: $P(m) \equiv$ for all forests F with exactly m deep vertices, the number of paths from F to $F\backslash I(F)$ in any DNC-tree for $\mathbf{X}_{\mathrm{F}}$ is

$$\prod_{i=1}^{m} (\deg(u_i) - 1)\,,$$

where $u_1, \ldots, u_m$ are the deep vertices of F.

Claim: $\forall m \geq 0 : P(m)$.

Proof. (by induction on m)

Base Rule $(P(0))$:

Let F denote an arbitrary forest with no deep vertices consisting of trees $T_1, \ldots, T_k$. By Lemma 17, we have that $c_{\lambda_{\text{lead}}(T_i)} = 1$ for every $1 \leq i \leq k$. Then, by Lemma 14 above $c_{\lambda_{\text{lead}}(F)} = 1$. Hence, $P(0)$ holds.

Inductive Step $(\forall k \geq 0 : P(k) \implies P(k+1))$:

Let $k \geq 0$ be arbitrary, and assume $P(k)$. Let F be an arbitrary forest with exactly $k+1$ deep vertices, call them $u_1, \ldots, u_{k+1}$. Since $k+1 \geq 1$, it follows by Lemma 18 that there exist a deep vertex in F, say u_1 (WLOG), such that u_1 has at most one deep vertex in its neighborhood. Suppose we fix a DNC-tree $\mathcal{D}$ for $\mathbf{X}_{\mathrm{F}}$ that starts by performing the deletion-near-contraction relation on $\deg(u_1) - 1$ edges of the form $u_1 v$, where v is not a deep vertex.

Fix an arbitrary path from F to $F \backslash I(F)$ in $\mathcal{D}$, and consider the first $\deg(u_1) - 1$ steps along this path, $x_1, x_2, \ldots, x_{\deg(u_1)-1}$. Since leaf-contractions result in forests with greater leading partitions, each of these steps must be deletions (D) or dot-contractions (DC). Furthermore, if all of these steps were deletions, then the final deletion will occur on an edge $u_1 v$, where u_1 is a degree-2 deep vertex. By Corollary 1, the resulting forest has a greater leading partition than F and so for some smallest $i \in [\deg(u_1) - 1]$, $x_i = DC$.

Let $u_1 v_i$ denote the edge that is dot-contracted at step i. Since v_i is an internal vertex that is not a deep vertex, it must have a leaf in its neighborhood. Then, after step x_i, u_1 has a leaf in its neighborhood and so for all $j \in \{i+1, \ldots, \deg(u_1) - 1\}$, $x_j \neq DC$ by Corollary 1. That is, for all $j \in \{i+1, \ldots, \deg(u_1) - 1\}$, $x_j = D$.

For any $i \in [\deg(u_1) - 1]$ performing dot-contraction will result in a forest with the same leading partition as F by Corollary 1 since, at this step, u has no leaf in its neighborhood. Thus, we have shown that the set of prefixes of length $\deg(u_1) - 1$ of paths from F to $F \backslash I(F)$ is given by:

$$S = \{D^{i-1}, DC, D^{\deg(u_1)-1-i} : i \in [\deg(u_1) - 1]\}.$$

Let F' denote the forest at the end of some arbitrary sequence from S. Note that in F', u_1 is no longer a deep vertex since a leaf was added to its neighborhood after the dot-contraction is performed, and every other deep vertex present in F appears in F' with same degree as in F. Then, by the Induction Hypothesis, there are $\prod_{i=2}^{m-1}(\deg(u_i)-1)$ paths from F' to $F'-I(F')$ in any DNC-tree for $\mathbf{X}_{\mathrm{F}'}$, reusing vertex labels. Since there are $\deg(u_1)-1$ possibilities for F', it follows that the number of paths from F to $F\backslash I(F)$ in any DNC-tree for $\mathbf{X}_{\mathrm{F}}$ is $(\deg(u_1)-1)\cdot\prod_{i=2}^{m-1}(\deg(u_i)-1)$, by the product principle. Hence, we have the claim.

□

Corollary 5. *For some tree T, if $|c_{\lambda_{lead}(T)}|=1$, then each internal vertex of T has a leaf in its neighborhood or is a degree-2 deep vertex.*

Proof. Let T be a tree with $|\lambda_{\text{lead}}(T)|=1$. Assume, to the contrary, that there exists some internal vertex $u\in V(T)$ such that u has no leaf in its neighborhood and is not a degree-2 deep vertex. Then u must be a deep vertex of degree at least 3. It follows by the previous theorem that $|c_{\lambda_{\text{lead}}(T)}|\geq 2$, a contradiction. □

We now apply λ_{lead} to show an interesting result for proper trees, which will be useful in a following theorem.

Lemma 12. *If T is a proper tree, then in X_T, $c_{\lambda_{\text{lead}}}=1$.*

Proof. Let T be an arbitrary proper tree, and fix a DNC-tree $\mathcal{D}$. Recall from Chapter 2 that all nodes in $\mathcal{D}$ are proper forests. That is, for every node F in $\mathcal{D}$, the endpoints of every internal edge of F has a leaf in its neighborhood. It follows that the only path in $\mathcal{D}$ from F to $F\backslash I(F)$ consists of repeated deletions. Thus, $c_{\lambda_{\text{lead}}(F)}=1$ □

Lemma 13. *For any non-star tree T, the leading partition of $\mathbf{X}_T$ is never a hook partition.*

Chapter 5

Chromatic Subspace and Basis

Let n be an integer and $\lambda^1 < \lambda^2 < \cdots < \lambda^k$ be the partitions of n in increasing order. For example, if $n = 4$, $\lambda^1 = (1,1,1,1), \lambda^2 = (2,1,1), \lambda^3 = (2,2), \lambda^4 = (3,1), \lambda^5 = (4)$. For any tree T on n vertices, we have $\mathbf{X}_{\mathrm{T}} = \sum_{j=1}^{k} c_{\lambda^j} \mathfrak{st}_{\lambda^j}$ for some integers $c_{\lambda^1}, \ldots, c_{\lambda^k}$. Using vector notation, we have:

$$\mathbf{X}_{\mathrm{T}} = \begin{bmatrix} c_{\lambda^1} & c_{\lambda^2} & \cdots & c_{\lambda^k} \end{bmatrix} \begin{bmatrix} \mathfrak{st}_{\lambda^1} \\ \mathfrak{st}_{\lambda^2} \\ \vdots \\ \mathfrak{st}_{\lambda^k} \end{bmatrix}.$$

We define the vector form of the chromatic symmetric function of T in the star-basis to be the vector $[c_{\lambda^1}\ c_{\lambda^2}\ \cdots\ c_{\lambda^k}]$ above. We will refer to this vector as *the CSF vector of* T.

Let $T_1, T_2, \ldots, T_q$ be the trees on n vertices. Then, as above, we have for any $1 \leq i \leq q$ that $\mathbf{X}_{T_i} = \sum_{j=1}^{k} c_{\lambda^{i,j}} \mathfrak{st}_{\lambda^j}$ for some integers $c_{\lambda^{i,1}}, \ldots, c_{\lambda^{i,k}}$. Using matrix notation, we have the following:

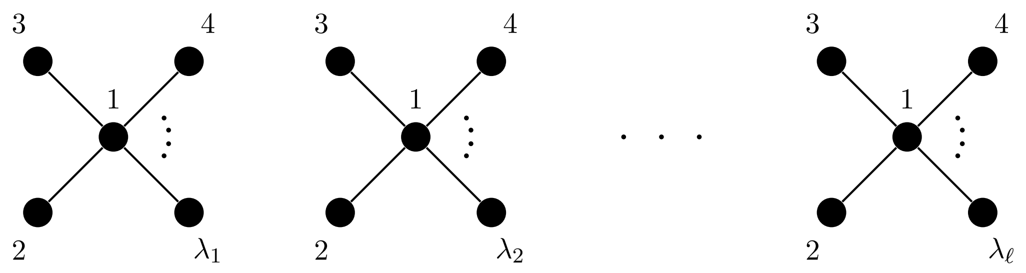
$$\begin{bmatrix} \mathbf{X}_{T_1} \\ \mathbf{X}_{T_2} \\ \vdots \\ \mathbf{X}_{T_q} \end{bmatrix} = \begin{bmatrix} c_{\lambda^1,1} & c_{\lambda^1,2} & \cdots & c_{\lambda^1,k} \\ c_{\lambda^2,1} & c_{\lambda^2,2} & \cdots & c_{\lambda^2,k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{\lambda^q,1} & c_{\lambda^q,2} & \cdots & c_{\lambda^q,k} \end{bmatrix} \begin{bmatrix} \mathfrak{st}_{\lambda^1} \\ \mathfrak{st}_{\lambda^2} \\ \vdots \\ \mathfrak{st}_{\lambda^k} \end{bmatrix}.$$

We define a matrix form of the chromatic symmetric functions of trees on n vertices to be any matrix equivalent to the $q \times k$ matrix above, up to row interchange so that the order of the trees doesn't matter. We will refer to any such matrix as *an n-CSF matrix.*

Recall that $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_\ell) \vdash n$ is a non-hook partition if $\ell = 1$ or $\lambda_2 > 1$. Recall also that $F(\lambda)$ is the star forest consisting of the star graphs $St_{\lambda_1}, \cdots, St_{\lambda_\ell}$. Let $\lambda \vdash n$ be a non-hook partition, and consider $F(\lambda)$. There are the following two possibilities:

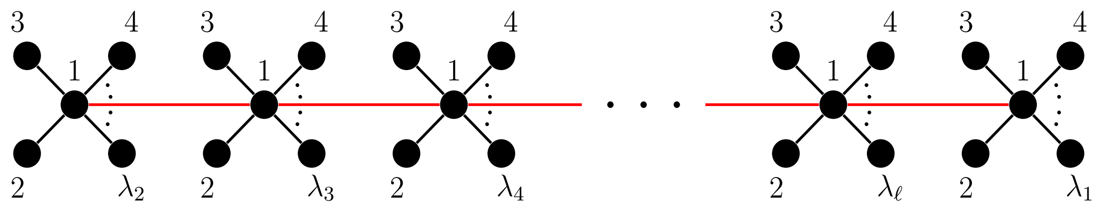
Case I: $\ell = 1$. Then, $F = St_n$.

Case II: $\ell > 1$ and $\lambda_2 > 1$. Then, we can visualize F as the following star forest:



The picture is a helpful visual aid, but is somewhat incomplete. In particular, we must recall that $\lambda_2 > 1$ and so the first two star graphs are both trees with at least two vertices. This implies that each of these trees has at least two leaves. We know nothing about the orders of the remaining star graphs besides that they are nonempty and in non-increasing order; in particular, the final graph St_{λ_ℓ} may be the trivial graph.

F has n vertices and $\sum_{i=1}^{\ell}(\lambda_i - 1) = n - \ell$ edges. It follows that we can convert it into a tree by inserting exactly $\ell - 1$ edges without introducing cycles. Suppose we do these edge insertions by translating the leftmost star graph all the way to the right of the rightmost star graph and then connecting the centers of the star graphs in a path as follows:



Let T denote this newly constructed tree. Observe that the endpoints for each of the inserted edges (in red) are internal vertices in T. This fact holds trivially for the inner vertices along the red path; it also holds for the vertices at the endpoints of the path since these vertices have at least two neighbors: another vertex along the path (since $\ell > 1$) and a leaf (since $\lambda_1, \lambda_2 > 1$). Furthermore, every other edge (in black) remains a leaf edge since red edges aren't inserted between two leaves. It follows immediately that $T \backslash I(T) = F$ and so $\lambda_{\text{lead}}(T) = \lambda(F) = \lambda$.

We have just shown something really cool! For any non-hook partition λ, the argument above gives us a procedure to construct a tree with leading partition λ. We call this procedure *build-leading-tree* and specify it formally on the following page.

Algorithm 3: Build-Leading-Tree (BLT)

```
input  : a non-hook partition $\lambda \vdash n$
output: a tree $T$ such that $\lambda_{\text{lead}}(T) = \lambda$
$F \leftarrow F(\lambda)$
$\ell \leftarrow \ell(\lambda)$
if $\ell = 1$ then
    return $F$
else
    Let $T_1, \ldots, T_\ell$ denote the star graphs of $F$ such that
      $|V(T_1)| \geq \cdots \geq |V(T_\ell)|$
    for $i \in \{2, 3, \ldots, \ell - 1\}$ do
        Let $u_i$ and $u_{i+1}$ be centers of $T_i$ and $T_{i+1}$, respectively.
        $F \leftarrow F \cup u_i u_{i+1}$
    end
    $F \leftarrow F \cup u_\ell u_1$
    return $F$
end
```

The following lemma follows by the previous discussion surrounding Algorithm 3 and the definition of the CSF vector.

Lemma 14. *Let $\mathcal{M}$ be an n-CSF matrix for some $n \geq 1$. Then,* $\text{rank}(\mathcal{M}) \geq p(n) - n + 1$.

Proof. Fix an n-CSF matrix $\mathcal{M}$. Recall that there are $n - 1$ hook partitions of n. For each of the $p(n) - n + 1$ remaining (non-hook) partitions $\lambda \vdash n$, there is a tree T with $\lambda_{\text{lead}}(T) = \lambda$ such that the CSF vector of $\mathbf{X}_{\text{T}}$ is a row of $\mathcal{M}$.

Recall that for any n, there are $n - 1$ hook partitions of n. It follows immediately that there are $p(n) - n + 1$ non-hook partitions of n. For each such partition λ, we can construct a tree with leading partition λ. Since each partition is distinct, each leading partition will be distinct, which implies the existence of $p(n) - n + 1$ pivot columns in $\mathcal{M}$. Thus, we have the lemma. $\square$

We have just shown an upper bound on the rank of any n-CSF matrix. We will also provide a lower bound (which happens to be the same as the upper bound!), but first we need the following lemma.

Let $cc(F)$ denote the number of connected components of a forest F.

Claim 1. *For any forest F with chromatic symmetric function $\mathbf{X}_{\mathrm{F}} = \sum_{\lambda \vdash n} c_\lambda \mathfrak{st}_\lambda$, for all $m > cc(F)$:*

$$\sum_{\lambda \vdash n, \ell(\lambda) = m} c_\lambda = 0.$$

Proof. We will proceed by induction on the number of internal edges of F.

- Base Case: Let F be an arbitrary star forest. Then, $\mathbf{X}_{\mathrm{F}} = \mathfrak{st}_{\lambda(F)}$. Trivially $\ell(\lambda(F)) = cc(F)$ and so the claim holds vacuously.

- Induction Step: Let $k \geq 0$ be arbitrary, and assume the induction hypothesis holds for forests with at most k internal edges. Let F be an arbitrary forest with exactly $k+1 \geq 1$ internal edges. Selecting an arbitrary internal edge e, we have by the deletion near-contraction relation that $\mathbf{X}_{\mathrm{F}} = \mathbf{X}_{F-e} - \mathbf{X}_{(F \odot e)\backslash \ell_e} + \mathbf{X}_{F \odot e}$. Note that $F - e$, $(F \odot e)\backslash \ell_e$, and $F \odot e$ each has at most k internal edges. That is, the induction hypothesis applies for each of these forests.

  If $m > cc(F) + 1$, by the induction hypothesis the claim holds since the coefficients indexed by partitions with more than $cc(F) + 1$ parts in $\mathbf{X}_{F\backslash e}$, $\mathbf{X}_{(F\odot e)\backslash \ell_e}$, and $\mathbf{X}_{F\odot e}$ are all 0.

  Otherwise, $m = cc(F) + 1$. In this case, the induction hypothesis still applies for $F \odot e$. Let $n = |V(F\backslash e)|$, and let $\mathbf{X}_{F\backslash e} = \sum\limits_{\lambda \vdash n} c_{\lambda, F\backslash e} \mathfrak{st}_\lambda$ and $\mathbf{X}_{(F\odot e)\backslash \ell_e} = \sum\limits_{\lambda \vdash n} c_{\lambda,(F\odot e)\backslash \ell_e} \mathfrak{st}_\lambda$.

  Note that $cc(F\backslash e) = cc((F \odot e)\backslash \ell_e) = m$. Since deletions and dot contractions increase the number of connected components, the only path to a star forest of length m is the sequence of repeated leaf contractions in both $F\backslash e$ and $(F\odot e)\backslash \ell_e$.

Then, the number of partitions of n of length m is the same in $\mathbf{X}_{F\odot e}$ as in $\mathbf{X}_{(F\backslash e)\backslash \ell_e}$.

That is, $\sum_{\lambda\vdash n,\ell(\lambda)=m} c_{\lambda,F\backslash e} = \sum_{\lambda\vdash n,\ell(\lambda)=m} c_{\lambda,(F\odot e)\backslash \ell_e} = 1$. Since $\mathbf{X}_{F\backslash e}$ and $\mathbf{X}_{(F\odot e)\backslash \ell_e}$ have opposite parities in the DNC relation, this case follows.

By the Principles of Mathematical Induction, the claim holds.

□

Corollary 6. *If T is a tree with $\mathbf{X}_{\mathrm{T}} = \sum_{\lambda\vdash n} c_\lambda \mathfrak{st}_\lambda$, then for all $m > 1$:*

$$\sum_{\lambda\vdash n,\ell(\lambda)=m} c_\lambda = 0.$$

Lemma 15. *Let $\mathcal{M}$ be an n-CSF matrix for some $n \geq 1$. Then,* $\operatorname{rank}(\mathcal{M}) \leq p(n) - n + 1$

Proof. Recall that $\mathcal{M}$ has $p(n)$ columns. Let $k \in [n-1]$ be arbitrary, and let h_k denote the hook partition with biggest part k. By Corollary 2, we have that the sum of the column vectors in $\mathcal{M}$ corresponding to partitions of length k is the zero vector. That is, h_k can be expressed as a linear combination of the column vectors corresponding to non-hook partitions of length k. Thus, we can express $n-1$ columns of $\mathcal{M}$ as linear combinations of other columns in $\mathcal{M}$. The claim follows immediately. □

Theorem 40. *Let $\mathcal{M}$ be an n-CSF matrix for some $n \geq 1$. Then,* $\operatorname{rank}(\mathcal{M}) = p(n) - n + 1$. *Furthermore, we have a simple procedure to explicitly construct a basis for the row space of $\mathcal{M}$, referred to as the* **caterpillar basis** *for CSFs of trees on n vertices.*

Proof. We have shown that $\operatorname{rank}(\mathcal{M}) \geq p(n) - n + 1$ and $\operatorname{rank}(\mathcal{M}) \leq p(n) - n + 1$. Iterating over the $p(n) - n + 1$ non-hook partitions of n and applying the *Build-*

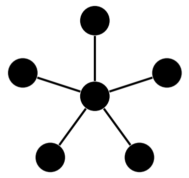
Leading-Tree procedure, we construct $p(n) - n + 1$ linearly independent CSF vectors, which must form a basis for the row space of n-CSF matrices. □

In the following example, we construct the caterpillar basis for CSFs on 6 vertices, consisting of 6 CSF-vectors.

- non-hook partition: (6)

  basis element: [0 0 0 0 0 0 0 0 0 0 1]

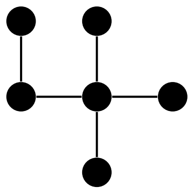
  tree:



- non-hook partition: $(4, 2)$

  basis element: [0 0 0 0 0 0 0 0 1 -1 1]

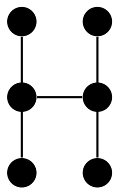
  tree:



- non-hook partition: $(3, 3)$

  basis element: [0 0 0 0 0 0 1 0 0 -1 1]

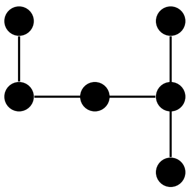
  tree:



- non-hook partition: $(3, 2, 1)$

  basis element: [0 0 0 0 0 -1 1 1 1 -2 1]

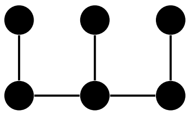
  tree:



- non-hook partition: $(2,2,2)$

  basis element: [0 0 0 1 0 -2 0 1 2 -2 1]

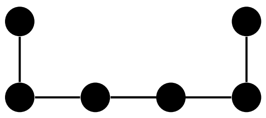
  tree:



- non-hook partition: $(2,2,1,1)$

  basis element: [0 0 1 1 -1 -4 1 3 2 -3 1]

  tree:



Chapter 6

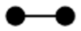
Code and Data

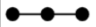
On the following pages, I include code that I wrote with Mario Tomba ('25) to efficiently compute the CSF in the star basis using the deletion-near-contraction relation. It makes use of dynamic programming (memoization) and is written in Python 3.

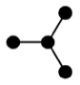
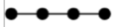
After the code, I include screenshots of the data that we constructed. The spreadsheets contain images of the trees, as well as their chromatic symmetric functions in the star basis, written in vector form. They are categorized by number of vertices.

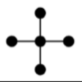
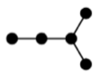
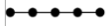
```
from sage.all import *


# Deletion on e

def left_operation(F, e):

    F.delete_edge(e)

    return F


# Contraction on e plus singleton

def middle_operation(F, e):

    F.contract_edge(e)
    F.add_vertex()
    return F

# Contraction on e plus singleton and edge
def right_operation(F, e):
    F.contract_edge(e)
    F.add_vertex()
    F.add_edge(e)
    return F

# If forest F has an internal edge, it returns one; otherwise, returns None
def get_internal_edge(F):
    for e in F.edges(labels=False):
        if F.degree(e)[0] > 1 and F.degree(e)[1] > 1:
            return e
    return None

# Returns number of 1's in an array
def num_singletons(p):
    num = 0
    for i in p:
        if i == 1:
            num += 1
    return num

# Returns the index of a partition p in Partitions(n), assuming
# lexicographic ordering and zero-indexing
def get_index(p, partitions):
    i = 0
    for partition in partitions:
        if partition == p:
            return i
        else:
            i += 1


# Returns number of internal edges in forest F
def num_int_edges(F):
    F = F.copy()
    i=0
    e = get_internal_edge(F)
    while (e != None):
        i += 1
        T = middle_operation(F, e)
        e = get_internal_edge(F)
    return i


def create_tree(n, edge_list):
    G = Graph()

    for v in range(1,n+1):
        G.add_vertex()
    for e in edge_list:
        G.add_edge(e)

    return G

# Input: A forest F on n>=2 vertices, a list of forests seen so far
# Output: Returns the CSF of F as a vector of length p(n),
# in lexicographic order
def CSF_helper(F,n,seen_list):
    partitions = Partitions(n)
    # Base case: F has already been seen
    for seen_forest in seen_list:
        if seen_forest.is_isomorphic(F):
            return seen_list[seen_forest]

    CSF = zero_vector(ZZ, len(partitions))

    # Base case: F has no internal edges
    e = get_internal_edge(F)
    if e == None:
        p = F.connected_components_sizes()
        sign = (-1)**num_singletons(p)
        index = get_index(p,partitions)
        CSF[index] = sign
        seen_list[F.copy(immutable=True)] = CSF
        return CSF
    # Recursive case: F has not been seen and has an internal edge
    else:
        F1 = F.copy()
        F2 = F.copy()
        F3 = F.copy()
        CSF = CSF_helper(left_operation(F1,e),n,seen_list) +
            CSF_helper(middle_operation(F2,e),n,seen_list) +
            CSF_helper(right_operation(F3,e),n,seen_list)
        seen_list[F.copy(immutable=True)] = CSF
        return CSF

# Input: n>=0
# Output: prints the CSF of all trees on n vertices
def CSF(n, int_edge_cap = None, print_bool = False, count = False):
    CSF_matrix = Matrix(0,len(Partitions(n)))
    seen_list = {}
    tree_iterator = graphs.trees(n)
    if print_bool or count:
        i = 0 #helps with plotting
    for tree in tree_iterator:
        if (int_edge_cap == None) or (num_int_edges(tree) <= int_edge_cap):
            if print_bool or count:
                i += 1
                if print_bool:
                    tree.plot().show()
                    print(i)
            tree = tree.copy()
            CSF = CSF_helper(tree, n, seen_list)
            print(CSF)
            CSF_matrix = CSF_matrix.stack(CSF)
    if count:
        print(i)
    return CSF_matrix


def CSF_tree(n, edge_list):
    seen_list = {}


    tree = create_tree(n, edge_list)
    tree.plot().show()
    tree = tree.copy()


    CSF_vector = CSF_helper(tree, n, seen_list)


    return CSF_vector


print(CSF_tree(3, ((0,1),(1,2))))
```

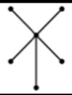
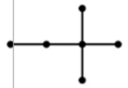
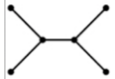
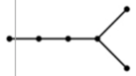
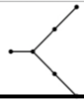
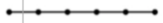
n=1		
#I(T)	T	(1)
0		1

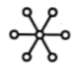
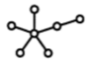
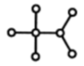
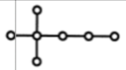
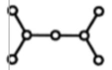
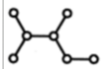
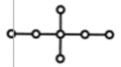
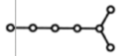
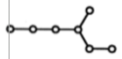
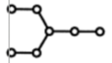
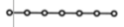
n=2			
#I(T)	T	(2)	(1²)
0		1	

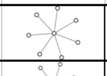
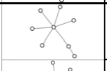
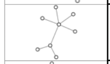
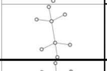
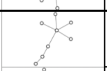
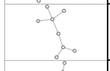
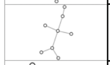
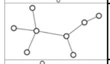
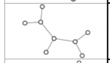
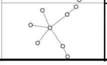
n=3				
#I(T)	T	(3)	(2,1)	(1³)
0		1		

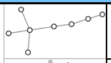
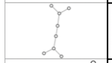
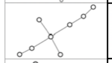
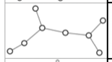
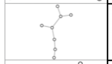
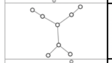
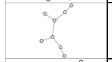
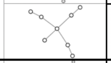
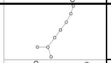
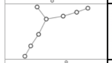
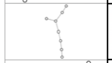
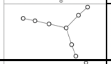
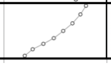
n=4						
#I(T)	T	(4)	(3,1)	(2²)	(2,1²)	(1⁴)
0		1				
1		1	-1	1		

n=5								
#I(T)	T	(5)	(4,1)	(3,2)	(3,1²)	(2²,1)	(2,1³)	(1⁵)
0		1						
1		1	-1	1				
2		1	-2	2	1	4		

n=6												
#I(T)	T	(6)	(5,1)	(4,2)	$(4,1^2)$	(3^2)	(3,2,1)	$(3,1^3)$	(2^3)	$(2^2,1^2)$	$(2,1^4)$	(1^6)
0		1										
1		1	-1	1								
		1	-1			1						
2		1	-2	1	1	1	-1					
		1	-2	2	1		-2		1			
3		1	-3	2	3	1	-4	-1	1	1		

n=7																
#I(T)	T	(7)	(6,1)	(5,2)	$(5,1^2)$	(4,3)	(4,2,1)	$(4,1^3)$	$(3^2,1)$	$(3,2^2)$	$(3,2,1^2)$	$(3,1^4)$	$(2^3,1)$	$(2^2,1^3)$	$(2,1^5)$	(1^7)
0		1														
1		1	-1	1												
		1	-1			1										
2		1	-2	1	1	1	-1									
		1	-2		1	2			-1							
		1	-2	1	1	1	-1		-1	1						
		1	-2	2	1		-2			1						
3		1	-3	1	3	2	-2	-1	-2	1	1					
		1	-3	2	3	1	-4	-1	-1	2	2		-1			
		1	-3	3	3		-6	-1		3	3		-2			
4		1	-4	2	6	2	-6	-4	-3	3	6	1	-2	-1		

n=8																							
#I(T)	T	(8)	(7,1)	(6,2)	$(6,1^2)$	(5,3)	(5,2,1)	$(5,1^3)$	(4^2)	(4,3,1)	$(4,2^2)$	$(4,2,1^2)$	$(4,1^4)$	$(3^2,2)$	$(3^2,1^2)$	**$(3,2^2,1)$**	$(3,2,1^3)$	$(3,1^5)$	(2^4)	$(2^3,1^2)$	$(2^2,1^4)$	$(2,1^6)$	(1^8)
0		1																					
1		1	-1	1																			
		1	-1			1																	
		1	-1						1														
2		1	-2	1	1	1	-1																
		1	-2		1	1			1	-1													
		1	-2	1	1	1	-1			-1				1									
		1	-2	1	1		-1		1	-1	1												
		1	-2		1	2				-2				1									
		1	-2	2	1		-2				1												

		n=8 (cont.)																					
3		1	-3	1	3	1	-2	-1	1	-2	1	1											
		1	-3		3	2		-1	1	-4				1	1								
		1	-3	2	3	1	-4	-1		-1	1	2		1		-1							
		1	-3	1	3	1	-2	-1	1	-3	1	1		1	1	-1							
		1	-3	1	3	2	-2	-1		-3		1		2	1	-1							
		1	-3	2	3	1	-4	-1		-2	1	2		2	1	-2							
		1	-3	2	3		-4	-1	1	-2	3	2			1	-3			1				
		1	-3	3	3		-6	-1			3	3				-3			1				
4		1	-4	1	6	2	-3	-4	1	-6	1	3	1	2	3	-2	-1						
		1	-4	2	6	2	-6	-4		-4	1	6	1	3	2	-4	-2			1			
		1	-4	2	6	1	-6	-4	1	-4	3	6	1	1	2	-5	-2		1	1			
		1	-4	3	6	1	-9	-4		-2	3	9	1	2	1	-7	-3		1	2			
5		1	-5	2	10	2	-8	-10	1	-8	3	12	5	3	6	-9	-8	-1	1	3	1		

Chapter 7

Open Questions

There remain several open questions related to the chromatic symmetric function. In fact, in this thesis, we have introduced some new questions and new means of exploring preexisting questions. For example, we have shown various properties that can be extracted from the CSF using the star-basis. It will likely be useful to consider other properties that can be extracted using other bases. In particular, is there enough information that can be extract to reconstruct the graph?

In Chapter 3, we formalize the notion of a DNC-tree and then proceed to prove results about the chromatic symmetric function using the formalization. Following the example of the leading term in Chapter 4, can we find other coefficients that distinguish classes of trees using DNC-trees? What counting arguments can we make, and how can we use DNC-trees to aid proofs about coefficients of terms in the star basis?

In Chapter 5, we introduce the Caterpillar Basis, which allows us to express the chromatic symmetric function of a tree as a linear combination of chromatic symmetric functions of Caterpillar Basis elements. By finding patterns in the ways CSFs can be expressed using this basis, we may be able to distinguish large classes of trees.

Bibliography

[AMOZ] Aliste-Prieto, J., de Mier, A., Orellana, R., & Zamora, J. (2022). *Marked graphs and the chromatic symmetric function*. ArXiv. /abs/2202.11787.

[CvW] Cho, S., & van Willigenburg, S. (2015). *Chromatic bases for symmetric functions.*

[HJ] Heil, S., & Ji, C. (2018). *On an Algorithm for Comparing the Chromatic Symmetric Functions of Trees.*

[MMA] Martin, J., Morin, M. and Wagner, J. *On distinguishing Trees by their Chromatic Symmetric Functions*, Journal of Combinatorial Theory, Series A 115, 237-253 (2008).

[St1] Stanley, Richard P. *A symmetric function generalization of the chromatic polynomial of a graph*, Adv. Math. 111 (1995), no. 1, 166–194.

[St2] Stanley, Richard P. *Graph colorings and related symmetric functions: ideas and applications. A description of results, interesting applications, and notable open problems*, Discrete Math. 193 (1998), no. 1-3, 267–286.

[West] West, D. B. (2000). *Introduction to Graph Theory.* Prentice Hall. ISBN: 0130144002.